Cortex-M7 Software Development

Course Description

Cortex-M7 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-M7 processor.

The course covers the Cortex-M7 architecture, development tools, instruction set, CMSIS, Assembly programming, interrupt handling, synchronization, memory model, caches and TCMs, efficient C programming, compiler and linker optimizations, using the MPU, debug, floating point and DSP instructions.

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

Course Duration

4 days

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il

Goals

1. Become familiar with ARMv7-M architecture
2. Become familiar with Cortex-M7 microarchitecture
3. Become familiar with ARMv7-M instruction set
4. Become familiar with the development tools for Cortex-M
5. Be able to handle interrupts and exceptions
6. Be able to configure and use the MPU
7. Understand the memory structure in v7-M architecture
8. Use efficiently caches and TCMs
9. Write an efficient C code for Cortex-M microcontroller
10. Apply optimization techniques for compiler and linker
11. Be able to debug ARMv7-M processors
12. Become familiar with DSP and FP instructions
13. Be able to write software for Cortex-M microcontrollers

Target Audience

Software engineers that would like developing software and firmware for platforms based on Cortex-M7 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 M7 Architecture Overview
- Tools Overview for ARM Microcontrollers
- CMSIS Overview
- ARMv7-M Programmer's Model
- ARMv7-M Assembly Programming
- ARMv7-M Memory Model
- ARMv7-M Exception Handling
- Software Engineer's Guide to Cortex-M7
- ARMv7-M Compiler Hints & Tips
- ARMv7-M Linker & Libraries Hints & Tips
- Embedded Software Development for Cortex-M Processors
- Cortex-M7 Debug
- Cortex-M7 MPU
- Cortex-M7 DSP and SIMD Instructions
- Cortex-M7 Floating Point Instruction Set

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il

Day #1

- **The ARM Architecture**
  - ARM connected community
  - ARM Cortex processors
  - Development of the ARM architecture
  - ARM architecture profiles

- **Cortex-M7 Overview**
  - Cortex-M7 block diagram
  - Architectural features
  - Micro-architectural features
  - Powerful and scalable instruction set
  - Core registers
  - Modes, privilege and stacks
  - Memory map
  - Memory interfaces
  - Tightly Coupled Memory (TCM)
  - DMA interface (AHBS)
  - AHB peripheral port (AHBP)
  - AXI master (AXIM)
  - Caches overview
  - Caches structure
  - Pipeline features
  - Interrupts and exceptions
  - Memory Protection Unit (MPU)
  - Power management
  - System timer (SysTick)
  - Core debug
  - Floating point unit
  - Lock-step
  - Implementation options

- **ARMv7-M Programmer's Model**
  - ARMv7-M profile overview
  - Data types
  - Core registers
    - ARMv7-M register set
    - Floating point extension registers
    - Program Counter (PC)
    - Link Register (LR)
    - Stack Pointer (SP)
    - Program status register (xPSR)
    - Special purpose registers (PRIMASK, BASEPRI, FAULTMASK, CONTROL)

- Modes, privilege and stacks
  - Modes overview
  - Privileged execution
  - Stacks
- Exceptions
- Instruction set support

➢ **Tools Overview for ARM Microcontrollers**
- Introduction to Keil MDK
- ULINK debug adapters
- Development boards
- DS5 and DSTREAM
- Fast Models from ARM

➢ **CMSIS Overview**
- Introduction to CMSIS
  - CMSIS structure
  - CMSIS bundle and documentation
  - CMSIS partners
- CMSIS-CORE
  - CMSIS-CORE overview
  - Using CMSIS-CORE
  - Instruction access
  - DSP/SIMD instructions
  - Special register access
  - NVIC access
  - System and clock configuration
  - SysTick access
  - Debug access
- CMSIS-DSP
  - CMSIS-DSP overview
  - Public header file
  - Pre-processor macros
- CMSIS-RTOS
  - CMSIS-RTOS overview
  - API structure
  - API implementation: RTX
  - CMSIS-RTOS quick reference
- CMSIS-SVD
  - CMSIS-SVD overview
  - Web infrastructure

- File description
- XML snippet
- Validation

- CMSIS-Pack
  - CMSIS-Pack overview
  - Packet description

- CMSIS-Driver
  - CMSIS-Driver overview
  - Functions
  - Example function (controlling USART interface)

- CMSIS-DAP
  - CMSIS-DAP overview
  - CMSIS-DAP benefits

- **Cortex-M7 Processor Core**
  - Cortex-M7 block diagram
  - Processor pipeline
  - Pipeline features
  - Instruction issue
  - Dual issue capability
  - ALU pipelines
  - MAC pipeline
  - Divide unit
  - Load/Store pipeline
  - Branch pipeline
  - FPU pipeline
  - Instruction retrial
  - Prefetch unit (PFU)
  - Branch Target Address Cache (BTAC)
  - Branch resolution
  - Memory mapped registers overview
  - CPUID Base register

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il

Day #2

- **ARMv7-M Assembly Programming**
  - Why do you need to know assembler?
  - Instruction set basics
  - Unified Assembler Language (UAL)
  - Thumb instruction encoding choice
  - Example Assembly file
  - Data processing instructions
  - Load/Store instructions
  - Flow control
  - Miscellaneous instructions

- **ARMv7-M Memory Model**
  - Introduction to Cortex-M memory model
  - Memory address space
  - Memory types and attributes
  - Alignment and endianness
  - Barriers
  - Examples
  - System caches and TCMs
  - Memory access order rules

- **Cortex-M7 Level 1 Subsystems**
  - Cortex-M7 processor block diagram
  - PPB L1 subsystem related regions
  - Caches fundamentals
    - What is a cache?
    - How is a cache accessed by the core?
    - How is a cache populated?
    - Direct mapped cache
    - Set associative cache
    - Example memory access
    - Cache terminology

  - Cortex-M7 cache subsystem
    - Cortex-M7 L1 cache overview
    - I--cache 2-way set associative
    - D--cache 4-way set associative
    - L1 data cache policies
    - Level 2 caches
    - Caching and memory attributes
    - Inner and outer cache policies
    - Cache coherency

- L1 memory system buffers
- Store Buffer (STB)
- Cache ECC overview
- Cache control and identification registers
- Point of Coherency (PoC)
- Point of Unification (PoU)
- Cache maintenance operations
- Initializing and enabling L1 caches

- Tightly Coupled Memory (TCM)
  - What is TCM?
  - Cortex-M7 TCMs
  - Instruction/data TCM control registers
  - TCM timing
  - TCM ECC overview
  - TCM timing with wait/retry

- System considerations
  - Why should I cache?
  - Non-deterministic cache behavior
  - Cache optimizations
  - Cache discovery code
  - Cache ECC considerations
  - TCM ECC considerations

- **ARMv7-M Exception Handling**
  - Exception architecture overview
    - Micro-coded interrupt mechanism
    - Interrupt overheads

  - Exception model
    - Exception types
    - Processor mode usage
    - External interrupts
    - Pre-emption
    - Exception handling example
    - Exception model
    - Exception properties
    - Vector table for ARMv7-M
    - Vector Table Offset Register (VTOR)
    - Reset behavior
    - Exception behavior
    - Exception priorities overview
    - Exception states

  - Exception entry and exit behavior
    - Exception entry behavior

- Stacking on exception entry
- Interrupt entry timing
- ReturnAddress values
- Returning from an exception
- EXC_RETURN
- NMI exception entry example
- NMI exception return example
- Nesting example
- Tail-chaining example
- Late-arriving example
- Exceptions during state restore

o Prioritization and control
- Priority boosting
- Priority boosting example
- Special purpose mask registers
- Priority boosting instructions
- ARMv7-M priority grouping
- Group priority/sub-priority selection
- Interrupt control and status bits
- Interrupt enable registers
- Interrupt pending registers
- Interrupt active registers
- Interrupt priority registers

o Interrupt sensitivity
- Pulse and level sensitive interrupts
- Pulse sensitive interrupts: one pulse
- Pulse sensitive interrupts: multiple pulses
- Level sensitive interrupts
- Pending the same interrupt again

o Writing the vector table and interrupt handlers
- CMSIS-CORE: vector table
- Writing interrupt handlers
- Interrupt management

o Internal exceptions and RTOS support
- Internal exceptions
- SysTick timer
- System Service Call (SVC)
- SVC handlers
- Pended System Call (PendSV)
- Priority escalation
- Internal interrupt registers

o Fault exceptions

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il

- Fault exceptions in ARMv7-M
- Fault escalation
- Fault handling
- The lockup state
- Lockup state behavior
- Exit the lockup state
- Precise (synchronous) exceptions
- Imprecise (asynchronous) exceptions

Day #3

- **ARMv7-M C/C++ Compiler Hints & Tips**
  - Basic compilation
    - Language support
    - Variable types supported
    - Optimization levels
    - Selecting an architecture or processor
  - Compiler optimizations
    - Automatic optimizations
    - Using “volatile”
    - Tail-call optimization
    - Instruction scheduling
    - Idiom recognition
    - Inlining of functions
    - Loop transformation
    - Branch target optimization
    - Multifile compilation
  - Coding considerations
    - Register usage
    - Parameter passing
    - Loop termination
    - Division operations
    - Division by compile-time constants
    - Modulo arithmetic
    - Floating point
    - Floating point linkage
    - C++ support
  - Mixing C/C++ and assembler
    - Mixing C and assembly
    - Calling assembly from C/C++
    - CMSIS

- Intrinsics
- Named register variables
- Embedded assembler
- Inline assembler

- 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

- **ARMv7-M Linker & Libraries Hints & Tips**
  - Linking basics
    - What does a linker do?
    - How does the linker know what to do?
    - Object file structure
    - Library structure
    - Scatter-loading

  - System and user libraries
    - Libraries versus object files
    - Linker library searching
    - Creating and maintaining libraries

  - Veneers
    - Dealing with branches
    - Linker generated veneers
    - Minimizing the number of veneers

  - Stack issues
    - Stack issues considerations
    - Protecting and measuring stack usage
    - --callgraph example

  - Linker optimizations and diagnostics
    - Unused section elimination
    - RW data compression
    - Small function inlining
    - Linker feedback
    - Linking for specific target
    - Debug issues and build time

- Useful linker diagnostics

- ARM supplied libraries
  - ARM compiler standard libraries
  - Microlib

- **ARMv7-M Synchronization**
  - Introduction to synchronization and semaphores
    - The need for atomicity
    - The race for atomicity
    - Critical sections

  - Exclusive accesses
    - Effective atomicity
    - LDREX and STREX instructions
    - Example: lock()
    - Example: unlock()
    - Programs still have to be smart
    - Example: multi-thread Mutex
    - Non-coherent multiprocessor
    - Memory attributes
    - Context switching
    - Exclusive Reservation Granule (ERG)
    - Example: multiprocessor Mutex

- **Embedded Software Development for Cortex-M Processors**
  - Embedded development process
  - Default compilation tool behavior
    - Default memory map
    - Default C library
    - Default C library initialization sequence

  - System startup
    - Reset and initialization
    - CMSIS startup and initialization
    - CMSIS-CORE startup file
    - CMSIS-CORE vector table
    - CMSIS-CORE exception handlers
    - C library startup and initialization

  - Tailoring the C library to your target
    - Custom memory map
    - Introduction to scatter-loading
    - Scatter-loading example
    - Scatter-loading description files
    - Scatter file example
    - Linker placement rules

- Ordering objects in a scatter file
- Root regions
- Special scatter-loading regions
- Run-time memory management
- Run-time memory models
- Stack and heap setup
- CMSIS-CORE stack and heap setup
- Retargeting __user_setup_stackheap()
- Process Stack Pointer (PSP) setup
- MPU initialization
- Memory mapped registers
- Unused sections and entry points
- Long branch veneers

o Post startup initialization
- Extending functions
- 8-byte stack alignment in handlers
- Change thread mode to unprivileged

o Tailoring the C library to your target
- Thumb C libraries provided
- Retargeting the C library
- Avoiding C library semihosting
- Exception table in C/C++

o Building and debugging your image
- Debugging ROM images

Day #4

➢ **Cortex-M7 Debug**

o Introduction to debug
- Basic debug requirements
- ARMv7-M debug features
- Invasive debug
- Non-invasive debug
- PPB debug related regions
- Debug register support in SCS

o CoreSight and Debug Access Port (DAP) overview
- What is CoreSight?
- Example CoreSight system
- Debug Access Port (DAP)
- Cortex-M3/M4 debug components and access paths
- Cortex-M7 debug components and access paths

- Debug events and reset
  - Debug state
  - Debug event sources
  - Halting debug mode
  - Breakpoints versus watchpoints
  - Vector catch
  - Semihosting
  - Reset
  - Downloading boot code

- Flash Patch and Breakpoint Unit (FPB)
  - FPB overview
  - FPB breakpoint example
  - Flash patch remapping support
  - Flash patch register usage
  - FPB instruction remap example

- Data Watchpoint and Trace Unit (DWT)
  - DWT overview
  - DWT: halting debug
  - DWT: halting debug examples
  - DWT: trace and profiling
  - PC sampling register (DWT_PCSR)
  - DWT event counters
  - DWT cycle counter

- Instrumentation Trace Macrocell (ITM)
  - ITM overview
  - Cortex-M7 and ITM
  - ITM stimulus port registers
  - ITM control and setup
  - ITM example

- Embedded Trace Macrocell (ETM)
  - ETM overview
  - Cortex-M7 and ETM
  - ETM-M3 and ETM-M4
  - ETM-M7
  - Basic ETM instruction trace operation
  - ETM-M7 block diagram
  - ETM-M7 interface
  - ETM sharing

- Trace Port Interface Unit (TPIU), Trace Packets, Timestamping & Trace Bandwidth
  - TPIU interface / serial wire output

- Serial Wire Viewer (SWV)
- Trace bandwidth and trace port

- Implementation details
  - Cortex-M7 debug interface
  - Cortex-Mx or CoreSight SoC-400 DAP?
  - ROM tables
  - ROM table entry format
  - Cortex-M7 processor ROM table
  - Cortex-M7 PPB ROM table
  - ROM table hierarchy
  - Linking of ROM tables
  - Example ROM table entry: SCS
  - CoreSight memory map
  - CoreSight identification example
  - Top level ROM table
  - Cortex-M7 Debug Access Port
  - Trace options
  - ETM data trace requirements
  - Off-chip TRACECLK
  -

- **ARMv7-M Memory Protection**
  - Memory protection overview
    - Motivation: memory protection
    - Default system address map
    - PMSAv7
    - Memory Protection Unit (MPU)

  - Memory regions
    - Memory regions overview
    - PRIVDEFENA and HFNMIENA
    - Region attribute control
    - Type extension, C, B and S encodings
    - Normal memory cacheable properties
    - Access permissions
    - Region overlapping overview
    - Overlapping regions examples
    - Sub-regions
    - Sub-regions examples

  - Setting up the MPU
    - MPU registers in SCS
    - MPU type register
    - MPU control register
    - MPU region number register
    - MPU region base address register

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il

- MPU region attribute and size register
- MPU alias register support
- Configuring an MPU region
- Enabling the MPU
- MPU alias register usage
- CMSIS MPU initialization example
- MPU initialization optimization
- MemManage faults

- **ARMv7-M Extensions**
  - Extensions overview
    - Cortex-M extensions
  - DSP extension
    - DSP extension overview
    - SIMD instructions
    - SIMD comparisons
    - Saturating arithmetic
    - ASX instruction
    - SIMD multiplies
  - Floating point extension
    - Floating point extension overview
    - FPU enabling
    - FPSCR
    - Floating point exceptions
    - Floating point instructions
    - Exception handling
    - Extended stack frame
    - Lazy context save

P.O.B. 803 Kfar Saba 441080 ISRAEL | P +972-52-5816791 | F +972-77-4702742 | ContactUs@HandsOnTraining.co.il | HandsOnTraining.co.il