



## Cortex-R4 Software Development

### Course Description

Cortex-R4 software development is a three days ARM official course. The course goes into great depth, and provides all necessary know-how to develop software for systems based on Cortex-R4 processor.

The course covers the processor architecture, memory ordering, memory protection unit (MPU), caches and TCMs, Assembler language, synchronization, barriers, debug, power management, C for ARM and exception handling.

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

### Course Duration

3 days (4 days with Hands-on labs)

## Goals

1. Become familiar with ARMv7 architecture
2. Become familiar with Cortex-R4 architecture
3. Become familiar with ARM instruction sets
4. Understand Caches and TCMs structures and maintenance
5. Be able to write assembler code for Cortex-R4
6. Implement synchronization processes using mutex/semaphore
7. Be able to add barriers instructions to control program flow
8. Be able to configure and use the MPU
9. Apply invasive and non-invasive debug techniques
10. Write an efficient C code for Cortex-R4 processor
11. Be familiar with ARM tools for Cortex-R processors
12. Manage Cortex-R4 power modes

## Target Audience

Software engineers that would like developing software and BSP for platforms based on Cortex-R4 processor.

## Prerequisites

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

## Course Material

ARM official course book  
Lab handbook  
ARM DS5 SDK



When innovation meets expertise...

## Agenda

### Main Topics:

- ARM Processor Cores
- ARM System Design
- Introduction to the ARM Architecture
- ISA Assembly
- Exception Handling
- Software Engineer's Guide to the Cortex-R4
- ARM Caches and TCMs
- Using the MPU
- Synchronization
- Barriers
- C/C++ Compiler Hints & Tips
- Linker & Libraries Hints & Tips
- Programming the GIC
- Further Compiler/Linker Hints & Tips
- Embedded Software Development
- Cortex-R4 power management
- Debug and trace

### Day #1

#### ❖ Introduction to the ARM architecture

- Architecture Versions
  - Introduction to the ARM architecture
  - Development of the ARM architecture
  - ARM Cortex processors (A/R/M)
- Registers & Instruction Sets
  - Data sizes and instruction sets
  - The ARM register set
  - Program status register
  - ARM, Thumb, Thumb2, ThumbEE, Jazelle
  - Endianness
  - Assembler syntax examples
  - Floating point and NEON
  - AAPCS

- Exception Model
  - Processor modes
  - Banking of registers
  - Taking an exception
  - Vector table
- Memory Model
  - Memory model overview
  - Memory types (Normal/Device/Strongly Ordered)
  - Memory hierarchy example
  - Data alignment
- Coprocessors
  - Coprocessors overview
  - CP15 example
  - PMU
- Architecture Extensions
  - TrustZone
  - Virtualization
  - Jazelle

## ❖ Cortex-R4 Architecture

- **System Engineer's Guide to the Cortex-R4**
  - Cortex-R4 architecture overview
  - Cortex-R4 pipeline
  - Program flow prediction
  - Divide & floating-point units
  - Level 1 memory system overview
  - Cache organization
  - MPU
  - Default memory map
  - TCM

## ❖ ARMv7-A/R ISA Overview

- ARM Assembler File Syntax
- Load/Store Instructions
  - Single/Double register data transfer
  - Addressing memory
  - Pre and post-indexed addressing
  - Multiple register data transfer
- Data Processing Instructions
  - Arithmetic, logical, move instructions

- Shift/Rotate operations
- The flexible second operand
- Instructions for loading constants
- Multiply/Divide
- Bit manipulation instructions
- Byte reversal
  
- Flow Control Instructions
  - Branch instructions
  - Interworking
  - Compare and branch if Zero
  - Condition codes and flags
  - If-Then instruction
  - Supervisor call instruction (SVC)
  
- Miscellaneous Instructions
  - Coprocessor instructions
  - PSR access
  - Breakpoint instruction (BKPT)
  - Wait for interrupt instruction (WFI)
  - NOP instruction
  - Wait for event & send event instructions (WFE & SEV)
  
- DSP Instructions
  - SIMD
  - Saturated maths and CLZ
  - Data packing/unpacking

## ❖ Exception Handling in Details

- Introduction
  - Exception handling process
  - The ARM register set and modes
  - Exception priorities
  - Vector table
  - Link register adjustments
  - Returning from exceptions
  - Exception state & Endianness
  - Non-makable fast interrupt
  - Low latency interrupt
  
- Interrupts
  - Interrupt & interrupt handler example
  - Interrupt pre-emption
  - Issues with re-enabling interrupts
  - Change processor state (CPS) instruction
  - Stack issues
  - Nested interrupt example

- FIQ vs IRQ
- Interrupt controllers
- Abort Handlers
  - Prefetch and data aborts
  - Data abort types (internal/external, precise/imprecise)
  - Identifying the abort source
  - Example data abort handler
- SVC Handlers
  - What are SVC used for?
  - Example SVC handler
  - Example SVC usage in an OS
- Undef Handlers
  - Undefined instruction
  - Example Undef handler
- Reset Handlers

## Day #2

### ❖ Memory Structure

- ARMv7-R Caches and TCMs
  - Cache basics
  - Caches on ARM processors
  - L1 data cache policies
  - Inner and Outer cache policies
  - Write back and write through
  - L1 memory system buffers
  - Tightly Coupled Memory (TCM)
  - TCM configuration
  - ECC and parity schemes
  - Optimization considerations
  - Cache coherency operations
  - Cache core optimizations
  - Point of Unification (PoU) and Point of Coherency (PoC)

### ❖ Memory Protection Unit (MPU)

- Using the MPU
  - Why do we need memory management?

- Access permissions
- Memory types
- Types & attributes
- Instruction accesses
- Memory Protection Unit (MPU) overview
- Protection regions
- When the MPU is disabled

### ❖ Synchronization Techniques

- Synchronization
  - The need for atomicity
  - Critical sections
  - LDREX and STREX instructions
  - Multi-thread mutex example
  - Coherent and non-coherent multi-core
  - Synchronization in a cluster
  - Memory attributes
  - Context switching

### ❖ Using Barriers

- Understanding Barriers
  - Memory model and access order
  - Barriers overview
  - Data barriers
  - Speculation across barriers
  - Instruction barriers
  - Ordering of maintenance operations

## Day #3

### ❖ C/C++ Compiler Hints & Tips

- Writing C for ARM
  - Parameter passing
    - Parameter passing
    - Passing more than 4 parameters
    - Parameter alignment
  - Floating point linkage
    - HW and SW floating point linkage
    - Floating point linkage example

- Alignment
  - Global data layout
  - Unaligned accesses
  - Packing and alignment of structures
  - Alignment of pointers
- Coding considerations
  - Size of local variables
  - Division
  - Base pointer optimization
  - Using “volatile”
- C/C++ Compiler Hints & Tips
  - Basic compilation
    - Language support
    - Variables types supported
    - Optimization levels
    - Selecting an architecture or processor
  - Compiler optimizations
    - Automatic optimizations
    - Tail-call optimization
    - Instruction scheduling
    - Idiom recognition
    - Inlining of functions
    - Loop transformation
  - Coding considerations
    - Register usage
    - Loop termination
    - Division operations
    - Reminders-Modulo arithmetic
    - C++ support
  - Local and global data issues
    - Variable types
    - Global data layout
    - Optimization of memcpy()
- ❖ **Linker & Libraries Hints & Tips**
  - Linker & Libraries Hints & Tips
    - Linking basics
      - 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 and interworking
  - Dealing with branches
  - Linker generated veneers
  - Veneer types
  - Minimizing the number of veneers
- Linker optimizations and diagnostics
  - Unused section elimination
  - RW data compression
  - Small function inlining
  - Useful linker diagnostics
- ARM supplied libraries
  - ARM compiler standard libraries
  - Microlib
- Mixing C/C++ and Assembler
  - Register usage revisited
  - Mixing C and Assembly
  - Calling Assembly from C/C++
  - Intrinsic
  - Embedded Assembler
  - Inline Assembler
- Stack issues
  - Protecting and measuring stack usage
  - --callgraph example (Dhrystone)
- VFP
  - Floating point capabilities
  - Floating point linkage
- Advanced building facilities
  - Multifile compilation
  - Linker feedback
  - Linking for specific target
  - Debug issues & build time
  - BPABI & SysV

## ❖ Embedded Software Development

- Embedded Software Development
  - Embedded development process
  - An “out-of-the-box” build
    - Default C library
    - Default memory map
    - Application startup
  - Tailoring the C library to your target
    - Retargeting the C library
    - Avoiding C library Semihosting
  - Tailoring image memory map to your target
    - Introduction to Scatterloading
    - Scatter description files
    - Linker placement rules
    - Ordering objects in a Scatter file
    - Root regions
    - Run-time memory management
    - Stack and Heap initialization
    - Run-time memory models
    - Stack and Heap regions
  - Reset and initialization
    - The vector table
    - Initialization steps
    - Initialize stack pointers
    - Local memory setup
    - Extending functions
    - Execute mode considerations
  - Further memory map considerations
    - Long branch veneers
    - Memory mapped registers
  - Building and debugging your image
    - Unused section elimination/entry points
    - Endianness
    - Output options
    - Debugging ROM images
  - Placing stack and heap in scatter file
  - ROM/RAM remapping

## ❖ **Generic Interrupt Controller Programming**

- GIC Overview
  - GIC architecture
  - Sources of interrupt (SGI, PPI, SPI)
- Distributor and CPU Interfaces
  - Register interfaces
  - Distributor interface
  - CPU interface
  - Programming guidelines
- How to Enable and Configure Interrupts
  - Enabling the IC
  - Interrupt configuration
- How to Handle Interrupts
  - Interrupt states
  - Taking an interrupt
  - Which CPU services an SPI?
  - Priority mask register
  - Interrupt priority registers
  - Pre-emption
  - Nesting interrupts
- How to Send Software Interrupts
  - SGI capability
  - Sending a SGI
  - Receiving a SGI
- Security Extensions
  - Group 0 and Group 1
  - Acknowledging interrupts
  - Priority and banking
- Interrupts IDs on Cortex-R4

## ❖ **Debug Overview**

- Debug
  - Types of debug
  - Invasive debug
  - Debug infrastructure and CoreSight overview
  - How do I access the debug logic?
  - Debug events
  - Halt vs. Monitor mode debugging
  - Viewing memory

- Debugger impact on caches
- Vector catch
- Instruction breakpoint types
- Embedded cross trigger- CTI
- Debugger semi-hosting support
- Non-invasive debug (PMU and trace)
- Performance monitoring hardware
- PMU configuration
- Results analysis
- CoreSight trace system
- Other trace resources

## ❖ **Software Power Management for Cortex-R4**

- Cortex-R4 Power Management
  - Processor power consumption
    - Power components
    - Example power contributions
    - Power reduction
  
  - Power modes
    - Standby mode
    - Shutdown mode
    - Dormant mode
  
  - Use cases
  - SoC power modes
  - Entering low power modes
  - Exiting shutdown/dormant mode