

ARM Architecture and Embedded Software Design

Objective

This comprehensive course is to cover the software aspects of ARM system, and to equip students with practical techniques for application development on real ARM hardware. Upon completion, students will understand the ARM architecture development, and will be familiar with embedded system development process.

Overview

ARM designs the world's most popular processor architecture for embedded systems. The technology can be found at the heart of advanced digital products, from wireless, networking and consumer entertainment solutions to imaging, automotive, security and storage devices. MIRACLE offers this two days training course to provide the necessary knowledge to develop software for an ARM based system targeted at practicing engineers who are interested in embedded software development. This training course will be conducted in two consecutive days. MIRACLE has chosen the current most popular ARM processor family, ARM9 core KB9202, as the hardware platform for this course.

Target Audience

This course is designed for the practicing engineers interested in acquiring in deep knowledge of ARM technology and hands-on experience in embedded software design.

Prerequisites

- A basic understanding of microprocessors or microcontrollers
- Familiarity with assembler or C programming
- Embedded programming experience would be helpful, but is not essential
- A basic awareness of the ARM is useful but not essential

Course Objectives

- The course explains ARM7 and ARM9 pipeline operation
- The workbook provides the student with an introduction to the tools provided with the ARM Developer Suite version 1.2 (ADS)
- The ARM assembly instructions are viewed in detail
- The course focuses on cache operation
- Thumb / ARM interworking is described
- ARM debug solutions are explained

Confidential

Course Contents:

THE ARM ARCHITECTURE

- Overview of ARM
- ARM operation modes
- The ARM registers set, register organization summary according to the current mode
- Program Status Registers
- Exception handling, vector table, automatic switch into ARM mode
- Instruction sets : ARM branches and subroutines

ARM PROCESSOR CORE

- ARM7TDMI core signals
- ARM7TDMI block diagram
- The ARM7TDMI instruction pipeline
- ARM7TDMI memory interface
- ARM9TDMI datapaths
- ARM9TDMI pipeline
- Example ARM9TDMI system
- Overview of ARM9E-S, ARM10, StrongARM and Xscale

ARM DEVELOPPER SUITE (ADS) OVERVIEW

- Using the core tools
- C/C++ compilers key features
- Supplied libraries
- Codewarrior introduction
- Debugging with multi-ICE

ADS INTRODUCTORY WORKBOOK

- Compiling and running an example
- Creating a header file
- Creating a new project
- Viewing registers and memory

ARM AND THUMB INSTRUCTION SETS

- Conditional execution and flags
- Branch instructions
- The barrel shifter
- Immediate constants
- Single register data transfer
- Block data transfer
- Stack management
- Coprocessor instructions
- Register access in Thumb
- ARM architecture V5TE new instructions
- Assembler workbooks

Miracle Corporate Solutions Pvt. Ltd., C-41,GF, Behind Nirula's Hotel, Sec-2, Noida
Call : 0120-3058446/7
Mobile : 9311305846/7

Confidential

ARM / THUMB INTERWORKING

- Switching between states
- Branch exchange example
- Mixing ARM and Thumb subroutines
- ARM to thumb veneer
- Thumb-to-ARM veneer
- Interworking calls
- Interworking using codewarrior

EXCEPTION HANDLING

- Exception return instructions
- Exception priority
- Vector table instructions
- Chaining exception handlers
- Register usage in exception handlers
- FIQ vs IRQ
- Example C interrupt handler
- Software managed interrupt controller
- Issues when reenabling interrupts
- C nested interrupt example
- Invoking SWIs
- Data abort with memory management
- The return address

EMBEDDED SOFTWARE DEVELOPMENT

- ROM or RAM at 0x0 ?
- ROM/RAM remapping
- Exception vector table
- Reset handler
- Initialization : stack pointers, code and data areas
- C library initialization
- Scatterloading
- Linker placement rules
- Long branch veneers
- C library functionality
- Placing the stack and heap
- Debugging ROM images

