RM0 - Cortex-M0 / Cortex-M0+ implementation

This course covers both Cortex-M0 and Cortex-M0+ ARM CPUs

Objectives

- This course is split into 3 important parts:
  - Processor architecture
  - Software implementation
  - Hardware implementation.
- A tutorial has been developed by ACSYS to facilitate the understanding of Cortex-M0 low level programming, therefore labs can be replayed after the course.
- The course explains how to design a SoC based on Cortex-M0 / Cortex-M0+, clarifying the operation of the interconnect and the debug facilities integrated in the CPU.
- This training has been delivered several times to companies developing SoCs for wireless / consumer market.

Labs are run under Keil

A more detailed course description is available on request at info@ac6-training.com

Prerequisites

- Basic knowledge of processor or DSP.

Plan

First day

CORTEX-M0/M0+ ARCHITECTURE

- Instruction pipeline
- Internal bus matrix, fixed memory map
- Highlighting the differences between Cortex-M0 and Cortex-M3
- Implementation options
- Cortex-M0+ additional features, dual privilege levels, dual stack

ARM V6-M PROGRAMMING

- Program registers, xPSR format
- Thumb 16-bit instruction set

- Keil library functions, divide
- Barrier instruction, use cases

DEBUG

- Coresight overview
- CPU-dependent coresight units, breakpoints, watchpoints
- Vector catch
- Serial Wire Debug
- Optional Micro Trace Buffer (Cortex-M0+)

MEMORY PROTECTION UNIT - CORTEX-M0+

- Memory protection overview, ARM v7 PMSA
- Cortex-M0 MPU and bus faults
- Region overview, memory type and access control, sub-regions
- Setting up the MPU

Second day

EXCEPTION MECHANISM AND LOW POWER MODES

- Exception vs interrupt
- Automatic state saving on exception entry and exit, CISC approach
- Interrupt priority levels, nesting
- Tail-chaining and late arriving interrupts
- Fault management
- OS system call and task switching

LOW POWER MODES

- Standby and deep sleep with state retention
- Event vs interrupt
- Optional wake-up interrupt controller
- SysTick hardware timer
- Requirements for the Power Management Unit

EMBEDDED SOFTWARE DESIGN

- 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
- Long branch veneers
- CMSIS library

HARDWARE IMPLEMENTATION

- Bus architecture, von Neuman operation
- Single-cycle I/O port (Cortex-M0+)
- Address pipelining
- Sequential transfers
- AHB-lite specification

Renseignements pratiques

Durée : 2 jours
Prix : 1400 € HT