Objectifs

- Although the Cortex-M4 seems to be a simple 32-bit core, it supports sophisticated mechanisms, such as exception pre-emption, internal bus matrix and debug units
- Through a tutorial, the Cortex-M4 low level programming is explained, particularly the ARM linker parameterizing and some tricky assembly instructions
- Discover the concepts of real time multitasking
  - Determinism
  - Preemption
  - Interrupts
- Understand the structure of a TI-RTOS
- Discover the various TI-RTOS services and APIs
- Learn how to develop TI-RTOS applications
- Learn how to debug TI-RTOS applications

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

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

Prerequisites

- Basic Knowledge on C language and microcontrollers

First Day

Introduction to ARM CORTEX-M4/M4F

- ARM Cortex-M4 processor macrocell
- Programmer’s model
- Instruction pipeline
- Fixed memory map
- Privilege, modes and stacks
- Memory Protection Unit
- Interrupt handling
- Nested Vectored Interrupt Controller [NVIC]
- Power management
- Debug

ARM CORTEX-M4 CORE

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

Exception behavior, exception return

- Non-maskable exceptions
- Privilege, modes and stacks
- Fault escalation
- Priority boosting
- Vector table

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

Exercise: *Interruption Management on Cortex-M4*

Second Day

Elements of a real time system

- Tasks and task descriptors
  - Content of the task descriptor
  - Lists of task descriptors
- Context switch
- Task scheduling and preemption
  - Tick based or tickless scheduling
- Scheduling systems and schedulability proofs
  - Fixed priority scheduling
  - RMA and EDF scheduling
  - Adaptive scheduling
- Synchronization primitives
  - Waiting and waking up tasks
  - Semaphores
  - Mutual exclusion
  - The priority inversion problem
  - Priority inheritance (the automagic answer)
  - Priority ceiling (the design centric answer)
  - Mutexes and condition variables
  - Mailboxes

Introduction to Ti-RTOS

- What is TI-RTOS ?
- Overview of TI-RTOS Components
- SYS/BIOS: The TI-RTOS Kernel
- TI-RTOS Networking and Networking Services
- TI-RTOS Drivers

Third Day

SYS/BIOS

- SYS/BIOS Packages
- SYS/BIOS and XDC Tools
- TI-RTOS Startup Sequence
- Configuring SYS/BIOS Using XDCTools
- Overview of Threading Modules
- Thread Characteristics
- Choosing the right Thread
- Thread Priorities preemption and yielding
- Introduction to Hooks

Memory management

- Memory Map
- Placing sections into Memory Segments
- Stacks
- Cache configuration
- Dynamic Memory Allocation
- Heap implementations

Hardware Interrupt Threads (Hwis)

- Introduction to Hwis
- Creating Hwi Objects
- Hwi APIs
- Hwi Interrupt Nesting and System Stack Size
- Hwi Hooks

Exercise: *Configuring Hwis*
Exercise: *Configuring Hwis dynamically*

Software Interrupt Threads (Swis)

- Introduction to Swis
- Creating Swi objects
- SWI interrupts nesting
- Using Swi bbject trigger variables

Exercise: *Configuring Swis*
Exercise: *Configuring Swis dynamically*

Fourth Day

Task Threads

- Introduction to Task Threads
- Creating Tasks
- Task Execution states and scheduling
- Task Stacks
- Testing for stack overflow
- Task Hooks
- Task Yielding for time-slice scheduling
- Idle Loop

Exercise: *Hwi, Swi, Task and Idle Threads*

Synchronization modules

- Semaphores
- Event Module
- Gates
- Mailboxes
- Queues

Exercise: *Synchronization Primitives (Semaphore, Gates and Events)*
Exercise: *Reader / Writers (Mailboxes)*

Timing Services

- Clock
- Timer Module
- Seconds Module
- Timestamp

Exercise: *Clock and Timer Threads*

Instrumentation

- Introduction
- Load Module
- Error Handling
- Instrumentation Tools in Code Composer studio
- Performance optimization

Exercise: *Threads statistics (Hwis Global, Swi Global and Tasks)*

Timing Benchmarks

- Timing Benchmarks
- Interrupt Latency
- Hwi-Hardware Interrupt
- Swi-Software Interrupt Benchmarks
- Task Benchmarks
- Semaphore Benchmarks

Renseignements pratiques

Duration : 4 days
Cost : 2740 € HT