

## 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

- Code Composer Studio (CCS) v9.0.1
- LAUNCHXL-CC1352P1(CC1352P microcontroller) Texas Instruments board

## Environnement du cours

- Cours théorique
  - Support de cours au format PDF (en anglais) et une version imprimée lors des sessions en présentiel
  - Cours dispensé via le système de visioconférence Teams (si à distance)
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique
- Activités pratiques
  - Les activités pratiques représentent de 40% à 50% de la durée du cours
  - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
  - Exemples de code, exercices et solutions
  - Pour les formations à distance:
    - ▶ Un PC Linux en ligne par stagiaire pour les activités pratiques, avec tous les logiciels nécessaires préinstallés.
    - ▶ Le formateur a accès aux PC en ligne des stagiaires pour l'assistance technique et pédagogique
    - ▶ Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante.
  - Pour les formations en présentiel:
    - ▶ Un PC (Linux ou Windows) pour les activités pratiques avec, si approprié, une carte cible embarquée.
    - ▶ Un PC par binôme de stagiaires s'il y a plus de 6 stagiaires.
  - Pour les formations sur site:
    - ▶ Un manuel d'installation est fourni pour permettre de préinstaller les logiciels nécessaires.
    - ▶ Le formateur vient avec les cartes cible nécessaires (et les remporte à la fin de la formation).
- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque session (demi-journée en présentiel) une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

## Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

## Modalités d'évaluation

- Les prérequis indiqués ci-dessus sont évalués avant la formation par l'encadrement technique du stagiaire dans son entreprise, ou par le stagiaire lui-même dans le cas exceptionnel d'un stagiaire individuel.
- Les progrès des stagiaires sont évalués de deux façons différentes, suivant le cours:
  - Pour les cours se prêtant à des exercices pratiques, les résultats des exercices sont vérifiés par le formateur, qui aide si nécessaire les stagiaires à les réaliser en apportant des précisions supplémentaires.
  - Des quizz sont proposés en fin des sections ne comportant pas d'exercices pratiques pour vérifier que les stagiaires ont assimilé les points présentés
- En fin de formation, chaque stagiaire reçoit une attestation et un certificat attestant qu'il a suivi le cours avec succès.
  - En cas de problème dû à un manque de prérequis de la part du stagiaire, constaté lors de la formation, une formation différente ou complémentaire lui est proposée, en général pour conforter ses prérequis, en accord avec son responsable en entreprise le cas échéant.

## 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 object 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 (Hwi Global, Swi Global and Tasks)*

### Timing Benchmarks

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

### Renseignements pratiques

**Renseignements : 4 jours**