# oRT5 - Programmation temps réel avec Zephyr

## Il s'agit d'une formation en ligne en direct

## Objectifs

- Apprendre à développer, configurer, déboguer et tracer des applications Zephyr
- Découvrir le concept de multitâche temps réel
- Comprendre les contraintes temps réel, comme le déterminisme, la préemption ou les interruptions
- Comprendre les services du kernel Zephyr
- Apprendre les mécanismes de communication et de synchronisation
- Comprendre la gestion de la mémoire et les structures de données de Zephyr
- Comprendre le mode utilisateur et le mode kernel
- Écrire un device tree
- Écrire un pilote complet

### *Environnement du cours*

- Cours théorique
  - Support de cours au format PDF (en anglais).
  - Cours dispensé via le système de visioconférence Teams.
  - 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.
  - Un PC Linux en ligne par stagiaire pour les activités pratiques.
  - 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.
- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque session 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

### *Pré-requis*

- Bonne maîtrise du langage C (voir notre cours cours oL2 - Langage C pour les MCUs embarqués)

### *Déroulé pédagogique*

- Durée totale : 30 heures en 5 sessions de 6 heures chacune (hors temps de pause).
- De 40% à 50% du temps de formation est consacré aux activités pratiques qui servent à valider la bonne compréhension des concepts enseignés.
- Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante.
- En début de chaque session une interaction avec les stagiaires permet d'adapter si nécessaire le contenu du cours à leurs besoins.

### *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

## Première session

## Real-Time Concepts

- Base real time concepts
- The Real Time constraints
- Multi-task and real time
- Tasks and Task Descriptors
  - Content of the task descriptor
  - List of task descriptors
- Context Switch

## Introduction to Zephyr

- Zephyr Project
- Zephyr Ecosystem
- Why use Zephyr
- Install and use Zephyr
- Build and Configuration Systems
  - West
  - CMake
  - Kconfig and configuration overlay
  - Configuration tools: menuconfig and guiconfig

## Zephyr Without Threads

- Operation without Threads
- GPIO
- Random Number Generation
- Utilities
- Data Structures
  - Single-linked List
  - Double-linked List
  - Ring Buffers

***Exercise :*** *Hello World from Zephyr, configure and blink LEDs using Zephyr*
***Exercise :*** *Manage Zephyr linked list and understand container_of macro*

## Scheduling

- Task Scheduling and Preemption

    - Tick based or tickless scheduling
- Scheduling systems and schedulability proof
    - Fixed priorities scheduling
    - RMA and EDF scheduling
- Scheduling through Zephyr
    - Scheduling Algorithm
    - Cooperative Time Slicing
    - Preemptive Time Slicing

## ***Deuxième session***

### *Thread Management*

- Thread Control Block
- Creating Threads
- Threads Priorities
- Thread States
- Main and Idle Threads
- Delays
- Changing Thread Priority
- Suspending Threads
- Kernel Structures
    - Simple linked-list ready queue
    - Red/black tree ready queue
    - Traditional multi-queue ready queue
- Thread Custom Data
- Scheduling Traces
    - Runtime Statistics
    - User-Defined Tracing
    - Percepio Tracealyzer

***Exercise :*** *Create and manage threads*
***Exercise :*** *Create periodic threads*
***Exercise :*** *Create config overlay for visual trace diagnostics using Tracealyzer*

### *Memory Management in Zephyr*

- Memory Managers
- Dynamic memory managers
    - K_heap
    - System heap
    - Memory Slabs
    - Memory Blocks
- Heap Listeners
- Thread Resource Pools
- Stack Overflow detection
- User Mode
- Memory Domains

***Exercise :*** *Understand dynamic memory allocation in Zephyr*
***Exercise :*** *Display threads information and detect stack overflow*

## ***Troisième session***

### *Resource Management*

- Mutual Exclusion
- Critical Sections
- Mutexes
- Gatekeeper threads

- Atomic
- Lock-Free Data Structures
- SpinLocks

***Exercise :*** *Implement mutual exclusion between threads*

## Synchronization Primitives

- Synchronization
- Semaphores
- The Readers/Writer Problem
- Condition variables
- Events and Event Groups
- Polling

***Exercise :*** *The producer-consumer problem, synchronize and avoid concurrent access problems*
***Exercise :*** *Understanding event bit group by synchronizing several threads*

# Quatrième session

## Data Passing

- Message Queues
- Pipes
- Queues
  - FIFOs
  - LIFOs
- Mailboxes
- Stacks
- Zephyr Bus (Zbus)

***Exercise :*** *Create a print gatekeeper thread using message queue*

## Interrupt Management

- Threads and Interrupts
- Interrupts in zephyr
- Interrupts on ARM Cortex-M
- Handler thread
- Queue within an ISR
- Workqueue Threads
- Power Management

***Exercise :*** *Understand how to wait on multiple events and interrupt safe APIs*
***Exercise :*** *Understand how to pass data using Queues from an interrupt to a thread*
***Exercise :*** *Create and submit work items from interrupts to custom WorkQueue*

## Software Timers

- Timers
  - Defining a Timer
  - Using a Timer Expiry Function
- Timer types
  - One-shot timers
  - Auto-reload timers
- Timer Commands

***Exercise :*** *Understand the use of one-shot and auto-reload timers*

# Cinquième session

## Modules

- Why to use modules?
- Module structure
- Out-of-tree module
- YAML files
- Module CMakeLists.txt
- How to add and use custom Kconfigs

***Exercise :*** *Create a simple hello world module*
***Exercise :*** *Create a module that uses custom Kconfig options*

### *Zephyr device driver model*

- Introduction to Device Drivers
- Overview of the Zephyr device driver model
- Standard Drivers
- The struct device
- Subsystems
- API Extensions
- Initialization Levels
- Dependencies between device drivers
- Define devices programmatically
- Adding In-Tree Code to Zephyr Source Code

***Exercise :*** *Create a driver that respects the Zephyr Device Driver Model and define devices*
***Exercise :*** *Writing in-tree drivers*

### *Device Trees in Zephyr*

- Overview of Device Tree (DT) and its role in Zephyr
- Device Tree VS Kconfig
- Device Tree node structure
- Device Tree bindings
- Overlay and yaml files
- APIs to access device tree properties
- Write device drivers using device tree APIs
- Device Tree in Zephyr VS Linux

***Exercise :*** *Create a driver that uses custom device tree and Kconfig*

## Renseignements pratiques

**Durée :** **30 heures**
**Prix :** **3070 € HT**
**Prochaines sessions :** **du 10 au 14 juin 2024 - Online EurAsia (9h-16h CET)**