

## RT5 - Programmation avec Zephyr RTOS

### Usage complet de Zephyr: De la théorie à la pratique

#### 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 imprimé et au format PDF (en anglais).
  - 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 ou Windows) par binôme de stagiaires (si plus de 6 stagiaires) pour les activités pratiques avec, si approprié, une carte cible embarquée.
  - Le formateur accède aux PC des stagiaires pour l'assistance technique et pédagogique.
- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque demi-journée 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 [oL2 - Langage C pour les MCUs embarqués](#))

#### Déroulé pédagogique

- Durée totale : 5 jours
- De 40% à 50% du temps de formation est consacré aux activités pratiques qui servent à valider la bonne compréhension des concepts enseignés.

#### 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

### First Day

#### **Introduction to Zephyr**

- Zephyr Project
- Zephyr Ecosystem
- Why use Zephyr
- Install and use Zephyr
- Build and Configuration Systems
  - West
  - CMake
  - Zephyr SDK
- Application components and structure
- West manifest

#### **Configure Zephyr**

- Overview
- Kconfig
  - Default configuration
  - Interactive configuration tools
  - Config fragments
- Devicetree
  - Syntax
  - Standard properties
  - Initial devicetree source
  - Access devicetree from source code
  - Best practices

*Exercise : Write a device tree overlay*

#### **Zephyr Without Threads**

- Operation without Threads
- GPIO subsystem
- Utilities
  - Container\_of
  - For\_each
- 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*

## Second jour

### **Thread Management**

- Thread Fundamentals
  - Thread Control Block
  - Creating Threads
  - Threads Priorities
  - Changing Thread Priority
  - Thread States
- Main and Idle Threads
- Delays
- Suspending Threads
- Kernel Structures
  - Simple linked-list ready queue
  - Red/black tree ready queue
  - Traditional multi-queue ready queue
- Thread Custom Data

*Exercise : Create and manage threads*

*Exercise : Create periodic threads*

### **Tracing and logging**

- Runtime Statistics
- Scheduling Traces
  - User-Defined Tracing
  - Percepio Tracealyzer

*Exercise : Create config fragment 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
- RAM/ROM reports
- Stack information
  - Stack Overflow detection
  - Stack analysis

*Exercise : Understand dynamic memory allocation in Zephyr*

*Exercise : Display threads information and detect stack overflow*

## Troisième jour

### **User Mode**

- Overview
- Memory Domains
  - Partitions
  - Logical apps
- Syscalls
  - Kernel objects
  - Permissions

## Resource Management and Synchronization

- Mutual Exclusion
- Mutexes
- Gatekeeper threads
- Critical Sections
- Atomic
- SpinLocks
- Semaphores
- Events
- Polling

*Exercise : The producer-consumer problem, synchronize and avoid concurrent access problems*

*Exercise : Understanding event bit group by synchronizing several threads*

## Data Passing

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

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

*Exercise : Synchronous communication using mailboxes*

## Quatrième jour

## Interrupt Management

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

*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*

## Modules

- Why to use modules?

- Module structure
- Out-of-tree module
- YAML files
- Module CMakeLists.txt

*Exercise : Create a simple hello world module*

## **Kconfig**

- Advantages
- Kconfig Options in Zephyr RTOS
- Configuration System
- Writing custom Kconfig Options
- Kconfig extension
- Using Kconfigs

*Exercise : Create a module that uses custom Kconfig options*

## **Cinquième jour**

### **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

*Exercise : Create a driver that respects the Zephyr Device Driver Model and define devices*

### **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
- Adding In-Tree Code to Zephyr Source Code
- Common properties
  - compatible
  - reg
  - interrupts

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

*Exercise : Writing in-tree drivers*

### **Power Management**

- Overview
- System Power Management
- Device Power Management
  - System-Managed
  - Runtime
- Power domains

*Exercise : Write a driver compatible with power management subsystem*

## Renseignements pratiques

**Renseignements : 5 jours**