oRT5 - Programmation avec Zephyr RTOS

Formation Zephyr OS complète avec approche pratique

Objectifs

- Développer, configurer, déboguer et tracer des applications Zephyr
- Utilisation et développement de Devicetree et Kconfig
- Vue d'ensemble du multitâche dans Zephyr
- Utilisation de west et écriture d'un west manifest
- Comprendre les services noyau (kernel services) et l'écosystème de Zephyr
- Apprendre les mécanismes de communication et de synchronisation
- Comprendre la gestion de la mémoire et les structures de données de Zephyr
- Comprendre la séparation entre User mode et Kernel mode
- Écrire un device tree et développer des drivers Zephyr
- Utiliser et intégrer les subsystems de Zephyr

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.
  - Un PC Linux en ligne par stagiaire pour les activités pratiques
  - Exemples de code, exercices et solutions
  - 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 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 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.

Course Outline

Première session

Introduction to Zephyr

- Zephyr Ecosystem
- Why use Zephyr
  - Drivers' API abstraction
  - Hardware-agnostic configuration
  - Modular Architecture
- Host tools dependencies
- Install and Use Zephyr

Zephyr Build System

- Overview
  - West
  - CMake
  - Toolchains and Zephyr SDK
- Development environment
  - VSCode Configuration
  - Workbench for Zephyr
  - Debugging Tools and runners
- Components of a Zephyr application
  - Application structure
  - Application types
  - Samples & Tests
- Code structure

Exercise : Getting started with Zephyr

Configure Zephyr: Kconfig & Devicetree

- Overview
- Kconfig
  - Default configuration
  - Interactive configuration tools
  - Config fragments
- Devicetree
  - Syntax
  - Standard properties
  - DeviceTree node structure
  - Devicetree bindings
  - Initial devicetree source
  - Access devicetree from source code
  - Overlays
  - Best practices
- Snippets

Exercise : Write a devicetree overlay

Zephyr Fundamentals

- Operations without Threads
- Common subsytems

    - GPIOs
    - DeviceTree specification structures dt_spec
    - I2C
- Utilities
- Preprocessor meta-programming macros
- Data Structures
    - Single-linked List
    - Double-linked List
    - Ring Buffers
- Shell

Exercise : Using X-Macros in Zephyr and understanding CONTAINER_OF
Exercise : Writing custom shell commands

Deuxième session

West

- Why West? Problems solved
- Alternatives and limitations
- West structure
    - West manifest
    - West workspace
- Anatomy of west.yml
- West topologies
- Writing custom manifests
- Specific commands and common extensions
    - Init, update, list, config
    - Build, debug, attach, flash
    - Other common commands
- Extending West with custom commands

Thread Management

- Thread Fundamentals
    - Thread Control Block
    - Creating Threads
    - Threads Priorities
- Main and Idle Threads
- System Initialization
    - SYS_INIT
    - Initialization levels
    - Initialization order
- Delays and timeout
- Scheduling
    - Kernel Timing
    - States
    - Scheduler Backends
- 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 overlay for visual trace diagnostics using Tracealyzer

Memory Management in Zephyr

- Memory Overview
- Dynamic memory managers
  - K_heap
  - System heap
  - Memory Slabs
  - Memory Blocks
- Heap Listeners
- Thread Resource Pools
- RAM/ROM reports
- Stack information
  - Stack analysis
  - Puncover
  - High watermark
- Stack overflow detection

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

Troisième session

User Mode

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

Traditional Multithreading Primitives

- 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

Inter-Thread Communication

- 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

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

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 session

Modules

- Why to use modules?
- Module structure
- Out-of-tree module
- Module’s YAML
- Module CMakeLists.txt

Exercise : Create a basic module

Writing Kconfig symbols

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

Exercise : Create and configure a module that uses custom Kconfig options

Device Driver Architecture

- Zephyr Device Driver Model
    - Overview and its role

- Standard Drivers
  - The struct device
  - Subsystems
  - Device definition
- API Extensions
- Devices allocation and initialization
- Using drivers in application
- Initialization Levels
  - Dependencies between device drivers

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

Cinquième session

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

Writing device tree compatible driver

- 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

Developing Custom Boards

- Zephyr Board Architecture Overview

- Understanding SoC vs Board
- Role of a Board in Zephyr
- Structure and components of a board port
- Creating a New Board Definition
- Qualifiers and Revisions
- Integration and Creation Workflow
- Hardware Adaptation and Configuration Tuning

Exercise : Write a custom board

Sujets optionnels

Testing: ZTest & Twister

- Testing Fundamentals
  - Zephyr test infrastructure overview
  - Zephyr Test Framework (Ztest)
  - Test Runner (Twister) overview
- Creating a test suite

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