

RT5 - Programmation avec Zephyr RTOS

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.
 - Une machine virtuelle préconfigurée
 - Cible Qemu
 - Exemples de code, exercices et solutions
 - Le formateur assiste les stagiaires pendant les exercices
- 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.

Plan du cours

Premier jour

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

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

Testing: ZTest & Twister

- Testing Fundamentals
 - Zephyr test infrastructure overview
 - Zephyr Test Framework (Ztest)
 - Test Runner (Twister) overview
- Creating a test suite
- CI Overview & Pipeline Structure

Second jour

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
- Application components and structure
 - Application structure
 - Application types
 - Samples & Tests
- Code structure

Exercise : Getting started with Zephyr and using west

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 subsystems
 - 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

Troisième jour

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

Exercise : Writing a custom west manifest

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

- Logging subsystem
- Fatal Errors
- Task Watchdog
- Runtime Statistics
- Scheduling Traces
 - User-Defined Tracing
 - ZView

- Percepio Tracealyzer

Exercise : Create config fragment for visual trace diagnostics using Tracealyzer

Memory Management in Zephyr

- Memory Overview
- Iterable Sections
- 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

Quatrième jour

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
- 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

Cinquième jour

Modules

- Why to use modules?
- Module structure
- Out-of-tree module
- module.yml file
- Module CMakeLists.txt
- Blobs

Exercise : Create a basic module

Writing Kconfig symbols

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

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

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

Optional Topics

Secure Boot

- Secure boot fundamentals
 - Chain of trust
 - Root of trust
 - Signature keys
- MCUboot overview
 - Role as Zephyr's bootloader
 - Slots & partitions
- Image signing
- Image verification
- Trusted Firmware-M (TF-M)

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

Common Multithreading Problems

- Dining philosophers
- Lock-Free Data Structures
- The Readers/Writer Problem
- Condition variables

Linkable Loadable Extensions (LLEXT)

- Overview
 - Extending an application at runtime
 - Use cases and limitations
- Building an LLEXT extension
- Extension Development Kit (EDK)
- Loading and linking process