



RT5 - Zephyr RTOS Programming

From Theory to Practice

Objectives

- Learn how to develop, configure, debug and trace Zephyr applications
- Devicetree and Kconfig usage and development
- Using west and writing west manifest
- Zephyr real time multitasking overview
- Understand the Zephyr kernel Services and ecosystem
- Learn communication and synchronization mechanisms
- Understand Zephyr memory management and data structures
- Understand User mode and kernel mode
- Writing a device tree, and driver
- Using common subsystems

Course environment

- Theoretical course
 - Printed and PDF course material (in English)
 - The trainer answers trainees questions during the session and provide technical and pedagogical assistance
- Practical activities
 - Practical activities represent from 40% to 50% of course duration
 - Example code, labs and solutions
 - Preconfigured Virtual Machine (VM) for doing labs during of after the training
 - STM32 simulated board using Splat or NXP FRDM-MCXXN947 physical board

Prerequisites

- Good C programming skills (see our [L2 - C language for Embedded MCU](#) course)

Duration

- Total: 5 days
- From 40% to 50% of training time is devoted to practical activities
- Some Labs may be completed between sessions and are checked by the trainer on the next session

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Day

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 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

Second Day

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
 - 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 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

Third Day

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

Fourth Day

Modules

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

Exercise: Create a simple hello world module

Writing Kconfig symbols

- 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

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

Fifth Day

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

Optional Topics

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