



NRF5 - nRF Connect SDK 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
 - PDF course material (in English) supplemented by a printed version.
 - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- Practical activities
 - Practical activities represent from 40% to 50% of course duration.
 - Code examples, exercises and solutions
 - One PC (Linux ou Windows) for the practical activities with, if appropriate, a target board.
 - One PC for two trainees when there are more than 6 trainees.
- For onsite trainings:
 - An installation and test manual is provided to allow preinstallation of the needed software.
 - The trainer come with target boards if needed during the practical activities (and bring them back at the end of the course).
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Prerequisite

- Good C programming skills (see our L2 - C language for Embedded MCUs 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

Day 1

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

Day 2

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

Day 3

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

Day 4

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

Day 5

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