RTW - West, MCUXpresso SDK and Kconfig

West fundamentals and NXP ecosystem

Objectives

- Understand MCUXpresso SDK (MCUXSDK) structure
- Manage multi-repository projects using Zephyr West
- Use Kconfig and prj.conf for configuration
- Build, flash and debug on NXP targets
- Create and integrate custom boards
- Extend projects with FreeRTOS
- Integrate with VSCode for development and debugging
- Perform advanced analysis: multicore sysbuild, SPDX and memory footprint

Course environment

- Theoretical course
  - PDF course material (in English)
  - The trainer to answer trainees' questions during the training and provide technical and pedagogical assistance

Exercise: Practical activities

- Practical activities represent from 40% to 50% of course duration
- Example code, labs and solutions
- NXP FRDM Board or simulated board (IMXRT) using Zazu Simulator

Prerequisite

- C Language knowledge (see for example our L2 training course)
- Familiarity with Git and command-line tools

Target Audience

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

Course Outline

First day

Introduction to MCUXpresso SDK

- SDK structure and components
- Toolchains, CMake and Ninja integration
- Application structure and examples

West Tool

- Overview
- Application components and structure
- Application
  - Application
  - Modules

    - West workspace
- Why West? Problems solved
- West as a meta-tool: repository + commands
- Alternatives (git submodules, repo) and limitations
- West
    - West structure
    - Using west
    - West manifest
    - West commands
- West topologies
- Anatomy of west.yml
- Specific commands and common extensions
    - Init, update, list, config
    - Build, debug, attach, flash
    - Other common commands
- Extending West with custom commands

Exercise: Exercise: Getting started with West and MCUXpresso SDK
Exercise: Exercise: Create a custom workspace manifest while importing only required projects

Development Environment

- Setting up host tools (Git, Python, CMake, Ninja)
- Integrating LinkServer, Jlink and other debuggers
- Debugging workflow with GDB
- VSCode integration (tasks, debug sessions)
- MCUXpresso for VSCode

Exercise: Exercise: Build, flash and debug using command line and customize IDE

MCUXpresso Config Tools

- Overview of the configuration tool suite (Pins, Clocks, Peripherals, Device settings)
- How Config Tools integrate with MCUXpresso SDK and West builds
- Generating initialization code (pin_mux.c/h, clock_config.c/h, peripheral setup)
- Using the graphical interface to configure GPIO, UART, and system clocks
- Exporting configuration files and re-integrating them into applications
- Limitations and best practices when combining with Kconfig/prj.conf

Exercise: Exercise: Customize existing boards

Customization and Extensions

- Custom manifests for minimal projects
- Writing custom West commands
- Modifying in-tree applications (LED blinky)
- Freestanding applications outside the SDK

Exercise: Exercise: Extend west commands
Exercise: Exercise: Create a custom freestanding application

Second day

Integration and Analysis

- Adding FreeRTOS using West
- Multicore projects with Sysbuild
- SPDX analysis and compliance check
- Memory footprint and Puncover analysis

Exercise: Exercise: Extend the workflow with FreeRTOS and advanced tools
Exercise: Exercise: Using west memory analysis features

Kconfig and Project Configuration

- Configuration phase in West/CMake
- Kconfig framework:
  - Enabling/disabling global features
  - Tuning and conditional compilation
  - Default values and symbol dependencies
- Role of prj.conf and fragments
- Interactive configuration (menuconfig, guiconfig)
- Generated config files: .config, mcux_config.h
- Writing new Kconfig entries (symbols, menus, defaults)
- Limitations and best practices
- MCUXpresso SDK specifics (custom prefixes, no CONFIG_ macros)

Exercise: Exercise: Customize prj.conf
Exercise: Exercise: Create and use custom kconfig options

Developing Custom Boards

- Board Architecture Overview
- Structure and components of a board port
- Creating a New Board Definition
- Configuring custom boards
- Board debuggers
- Linker Script
- Integrating the custom Board into the SDK

Exercise: Exercise: Write a custom board

External MCUX Modules

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

Exercise: Exercise: Create a custom library module