NR6 - NXP + ThreadX + West

ThreadX with NXP MCUXpresso

Objectives

- Understand MCUXpresso SDK (MCUXSDK) structure
- Manage multi-repository projects using Zephyr West
- Use Kconfig and prj.conf for configuration
- Create and integrate custom boards
- Extend projects with Eclipse ThreadX
- Get an overview of Cortex-M architecture
- Discover the concepts of real time multitasking
- Understand Real Time constraints
- Understand the ThreadX architecture
- Discover the various ThreadX services and APIs
- Learn how to develop, debug and trace ThreadX applications
- Best practices for large MCUXpresso/ ThreadX projects

Prerequisite

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

Course Environment

- Theoretical course
  - PDF course material (in English) supplemented by a printed version for face-to-face courses.
  - Online courses are dispensed using the Teams video-conferencing system.
  - 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
  - For remote trainings:
    - One Online Linux PC per trainee for the practical activities.
    - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
    - QEMU Emulated board or physical board connected to the online PC (depending on the course).
    - Some Labs may be completed between sessions and are checked by the trainer on the next session.
  - For face-to-face trainings:
    - 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

Outline

Second day

Third day

Target Audience

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

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

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
  - 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: *Getting started with West and MCUXpresso SDK*
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: *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: *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: *Extend the workflow with ThreadX and advanced tools*
Exercise: *Using west memory analysis features*

Second day

Integration and Analysis

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

Exercise: *Extend the workflow with ThreadX and advanced tools*
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: *Customize prj.conf*
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: *Write a custom board*

External MCUX Modules

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

Exercise: *Create a custom library module*

Cortex-M resources used by ThreadX

- Cortex-M Architecture Overview
  - Two stacks pointers
  - Different Running-modes and Privileged Levels
  - MPU Overview
  - Systick Timer Description
  - ARMv8-M evolutions
- Exception / Interrupt Mechanism Overview
  - Interrupt entry and return Overview
  - SVC / PendSV / Systick Interrupt Presentation
- Developing with the IDE

Exercise: *Interrupt Management on Cortex-M*

Third Day

Introduction to Real Time

- Base real time concepts
- The Real Time constraints
- Multi-task and real time

Thread safe data structures

- Need for specific data structures
- Thread safe data structures
  - Linked lists
  - Circular lists
  - FIFOs
  - Stacks

Exercise: *Build a general purpose thread safe linked list*

Element of a real time system

- Tasks and Task Descriptors
  - Content of the task descriptor
  - List of task descriptors
- Context Switch

- Task Scheduling and Preemption
  - Tick based or tickless scheduling
- Scheduling systems and schedulability proof
  - Fixed priorities scheduling
  - RMA and EDF scheduling
- Scheduling through ThreadX
  - Deterministic preemptive scheduling
  - Scheduling strategies
  - Cooperative scheduling
  - Hybrid scheduling

Exercise: *Implement a Context Switch routine*

FreeRTOS Task Management

- The Task life-cycle
- Thread Control Block
- Thread States
- Thread Creation
- Preemption-Threshold
- Changing Thread Priority
  - Suspending Threads
  - Resume Thread Execution
  - Thread Sleep
- Terminate Thread Execution
- Time-Slice
- Thread States and Thread Design
- Thread Statistics
- Visual trace diagnostics using Tracealyzer

Exercise: *Managing tasks*

Fourth day

Memory Management in ThreadX

- ThreadX Memory Managers
  - Memory Byte Pool
  - Memory Block Pool
- Out of Memory management
- Stack overflow detection

Exercise: *Enhance the memory manager for memory error detection*
Exercise: *Detect stack overflow*

Resource Management with ThreadX

- Critical sections
  - Critical sections
  - Suspending (locking) the scheduler
- Mutual Exclusion
  - Spinlocks and interrupt masking
  - Mutex or Semaphore
  - Recursive or not recursive mutexes
  - Priority inversion problem
  - Priority inheritance (the automatic answer)
  - Priority ceiling (the design centric answer)
- Gatekeeper tasks

Exercise: *Gatekeeper tasks*

ThreadX Synchronization Primitives

- Queues
- Synchronization
- Semaphores
- Binary and counting semaphores
- Events and Event Groups
- The Readers/writer problem

Exercise: *Synchronizing a task with another one through binary semaphores*
Exercise: *Synchronizing a task with another one through queues*

Fifth Day

Parallelism Problems Solutions

- Parallel programming problems
  - Uncontrolled parallel access
  - Deadlocks
  - Livelocks
  - Starvation

Exercise: *The producer-consumer problem, illustrating (and avoiding) concurrent access problems*
Exercise: *The philosophers' dinner problem, illustrating (and avoiding) deadlock, livelock and starvation*
Exercise: *The readers-writer problem, illustrating complex concurrent access solving*

Interrupt Management

- Need for interrupts in a real time system
  - Software Interrupt
  - Time Interrupts
  - Device Interrupts
- Level or Edge interrupts
- Hardware and Software acknowledge
- Interrupt vectoring
- Interrupts and scheduling
- Deferred interrupt processing through ThreadX
  - Tasks with interrupt synchronization
  - Using semaphores within an ISR
  - Counting semaphores
  - Using queues within an ISR
- ThreadX interrupt processing
  - Writing ISRs in C
  - Interrupt safe functions
  - Interrupt nesting

Exercise: *Synchronize Interrupts with tasks*

Application Timer

- Application Timers
- System Timer Thread
- One-shot timers
- Auto-reload timers
- Application Timer Commands

Exercise: *Implement Soft Timers & Synchronize a task with a timer*

Renseignements pratiques

Inquiry : 5 days