

## RT7 - Real Time Programming with RT-Thread

### From Theory to Practice

#### Objectives

- Understand the RT-Thread architecture and execution model
- Migrate from bare-metal designs to a thread-based RTOS architecture
- Get an overview on Cortex-M architecture used by RT-Thread
- Understand the concepts of real time multithreading
- Learn how to configure, build, develop and debug RT-Thread applications
- Use Env, menuconfig and SCons in a BSP-based workflow
- Understand RT-Thread thread management, synchronization and communication services
- Understand RT-Thread memory management mechanisms
- Understand the RT-Thread device framework and device model
- Learn how to access devices through RT-Thread APIs
- Get a concise overview of the RT-Thread ecosystem

#### 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

#### Prerequisites

- Strong embedded C programming skills
- Basic knowledge of embedded processors

#### Duration

- 3 days

#### 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 trainee in 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 verify that 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

## Cortex-M resources used by RT-Thread

- Cortex-M architecture overview
- Registers and execution states
- Privilege levels and execution modes
  - Thread mode and Handler mode
  - MSP and PSP
- Exception and interrupt mechanism overview
- NVIC and SysTick
- PendSV role in context switching

*Exercise: Exercise: Interrupt management and Cortex-M execution model*

## RT-Thread Architecture and Project Organization

- RT-Thread standard architecture
- RT-Thread kernel architecture overview
- Kernel object model
- Standard BSP / project tree
- Where application code is added in an RT-Thread project
- Main thread, idle thread and timer-related system services
- Differences between a bare-metal superloop design and an RT-Thread application structure

*Exercise: Explore the structure of a Hello World RT-Thread project*

## RT-Thread Startup and Kernel Fundamentals

- System startup sequence
  - Board initialization
  - Scheduler initialization
  - Main thread, timer thread and idle thread
- Automatic initialization mechanism
- Kernel service overview
- Clock and timer management

*Exercise: Follow the boot sequence from reset to main()*

## Development Environment and System Configuration

- RT-Thread Env workflow
- Opening Env from the BSP environment
- menuconfig usage

- Configuration principles in RT-Thread
- Generation and role of .config and rtconfig.h
- Enabling and disabling kernel features and components
- Managing BSP-specific configuration
- Practical differences between source-level customization and package selection

*Exercise: Enable selected RT-Thread features with menuconfig, rebuild and validate the generated configuration*

### **Second Day**

#### **Real-Time Concepts and Scheduling through RT-Thread**

- Base real time concepts
- Tasks and task descriptors
- Context switch
- Task scheduling and preemption
- Tick-based scheduling
- Deterministic preemptive scheduling
- Priority-based scheduling in RT-Thread
- Time-slice rotation for same-priority threads

*Exercise: Analyse scheduling and context switch behavior*

#### **RT-Thread Thread Management**

- Thread control block
- Thread stack layout
- Thread states and life-cycle
- Thread creation
- Thread startup
- Thread delay, yield and suspend / resume
- Main thread and idle thread roles

*Exercise: Create and manage RT-Thread threads*

#### **Synchronization and Resource Management**

- Critical sections
- Interrupt lock and scheduler lock
- Semaphores
- Mutexes
  - Recursive locking
  - Priority inversion
  - Priority inheritance
- Events
  - OR trigger
  - AND trigger
- Choosing the right synchronization primitive

*Exercise: Synchronize threads with events*

#### **Inter-Thread Communication**

- Mailbox
- Message queue
- Signals overview
- Thread-to-thread communication patterns
- ISR-to-thread communication patterns
- Data passing versus synchronization

*Exercise: Implement producer-consumer and command / message exchange patterns*

## Third Day

### **Memory Management in RT-Thread**

- Memory model in embedded real-time systems
- Static objects versus dynamic objects
- Dynamic memory heap
- Small memory management algorithm
- Slab allocator overview
- Memheap for multiple non-contiguous heaps
- Memory pools
- Determinism and fragmentation considerations
- Memory allocation constraints in ISR context

*Exercise: Compare heap and memory pool usage in RT-Thread*

### **Clock, Tick and Timer Management**

- Clock tick and system heartbeat
- Software timers
  - One-shot timers
  - Periodic timers
- HARD\_TIMER versus SOFT\_TIMER
- Timer thread role

*Exercise: Use RT-Thread timers for periodic processing*

### **RT-Thread Device Framework and Device Model**

- I/O device framework
  - I/O device management layer
  - Device driver framework layer
  - Device driver layer
- RT-Thread device object model
- Device registration and lookup
- Standard device operations
  - init()
  - open()
  - close()
  - read()
  - write()
  - control()
- Character devices versus block devices
- Application to driver interaction flow
- Overview of common device classes
- Typical threaded device access patterns

*Exercise: Access and use a device through the RT-Thread device framework*

## Appendix

### **RT-Thread Ecosystem Overview**

- Components and packages overview
  - FinSH / MSH
  - Virtual File System
  - Network-related components
  - Utility components
- When to use components and when to keep a minimal RTOS design

## Advanced Topics Overview

- Kernel porting principles
- BSP porting overview
- Additional device classes available in RT-Thread
- Debug and tracing directions

## Renseignements pratiques

**Inquiry : 3 days**