OS3 - FreeRTOS Programming

Programming applications using the FreeRTOS operating system

Objectives

- Understanding the FreeRTOS architecture.
- Discovering the various FreeRTOS services and APIs.
- Learning how to develop FreeRTOS applications.
- Learning how to debug FreeRTOS applications.
- Becoming familiar with FreeRTOS APIs.
- Managing the Task priorities and clarifying the scheduling algorithm.
- Explaining the inter-task communication.
- Developping interrupt service routines.

Practical labs are conducted on STM32 boards under System Workbench for STM32.
(Keil or IAR can also be used for on site courses).
ACSYS is able to adapt the course to the Cortex-M based microcontroller chosen by the customer.
See our website for the various families we are covering: ST STM32, NXP LPC and Kinetis, TI Stellaris.

Course environment

- Example code, labs and solutions are provided to the attendees.
- An ARM-based target board (Cortex/M) with a JTAG probe.
- For open courses labs are done on STM32 boards using Sytem Workbench for STM32.
- For on site courses, the customer may select Keil or IAR IDEs.

Prerequisites

- Familiarity with embedded C concepts and programming
- Basic knowledge of embedded processors

Target Audience

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

Course Outline

First day

The FreeRTOS source code

- Introduction to FreeRTOS
  - The FreeRTOS architecture and features
  - The FreeRTOS license
- Getting FreeRTOS source code
  - Files and directories structure
  - The demo applications
- Data types and coding style
  - Naming conventions
- FreeRTOS on the Cortex/M processors

Task Management

- The Task life-cycle
  - Creating tasks
  - Deleting tasks
  - The Endless-loop pattern
- Task Priorities
  - Assigning task priorities
  - Changing task priorities
- The idle task
  - Idle task hooks

Scheduling

- Deterministic preemptive scheduling
  - Scheduling strategies
  - Cyclic scheduling (RMA)
  - Deadline scheduling
- Cooperative scheduling
  - Hybrid scheduling

Queue Management

- Purpose of queue management
- Basic use
  - Creation
  - Sending on a queue
  - Receiving from a queue
- Data management
  - Sending compound types
  - Transfering large data

Second day

Resource Management

- Mutual exclusion
  - Mutexes and binary semaphores
  - Deadlocks
  - Priority inversion
  - Priority inheritance
- Critical sections
  - Critical sections
  - Suspending (locking) the scheduler
  - Gatekeeper tasks

Memory Management

- FreeRTOS-provided memory allocation schemes
  - Allocate-only scheme
  - Best-fit without coalescing
  - Thread-safe default malloc
- Checking remaining free memory
- Adding an application-specific memory allocator

Interrupt Management

- FreeRTOS interrupt processing
  - Writing ISRs in C
  - Interrupt safe functions
  - Interrupt nesting
- Deferred interrupt processing
  - Using semaphores within an ISR
  - Counting semaphores
  - Using queues within an ISR
  - Tasks with interrupt synchronization

Software Timers

- The timer daemon task
- Timer configuration
- One-Shot and Auto-reload timers
- The Software Timer API

Trouble Shooting

- Checking for stack problems
- Common pitfalls

FreeRTOS-MPU

- The Cortex/M MPU
  - User and privileged modes
  - Access permissions
- Defining MPU regions
  - Overlapping regions
  - Predefined regions
  - Programmer-defined regions
- Needed linker configuration
- Practical usage tips