FEABHAS

Developing for Real-Time Operating Systems with FreeRTOS

Course category	RTOS
Training area	Operating Systems
Course code	RTOS-301
Duration	3 days
Price exc VAT	£1600.00
Additional information	Our public course schedule is suspended until 2023. We can still offer this course on-site either remotely delivered or face-to-face.

Course Description

The operating environment for a vast range of embedded systems is event-driven and asynchronous.

Dealing with such an environment with sequential code is challenging and liable to suffer from latency and responsiveness issues. Real-Time Operating Systems (RTOS) provide concurrency services to aid with solving these problems.

Developing concurrent applications with an RTOS is a non-trivial exercise and requires the engineer to understand a new set of problems and master a new set of design and coding mechanisms.

This three-day course provides a practical introduction to the theory, application and development of RTOS-based systems. Particular emphasis is placed on issues relating to resource-constrained embedded applications. Nevertheless, this course is still of significant value to engineers developing non-embedded real-time projects.

The course covers basic concepts, practical issues and coding idioms for concurrent applications. Approximately 50% of the time is given over to practical programming, where students will be developing a case study using the commercial FreeRTOS operating system on an ARM Cortex M target system.

Overview

A three-day course providing a practical introduction to the theory, structure and practice of real-time operating systems.

Course Objectives

After completing the course attendees will:

- Understand the core components and mechanisms of commercial RTOS
- Understand the implications of concurrent design.
- Be able to build efficient, thread-safe code.

Pre-requisites

- A strong working knowledge of C
- An understanding of the fundamentals of modern hardware architecture is useful, but not essential.

Who should attend?

Application programmers and software engineers who are new to real-time system development, or wish to improve their concurrency design skills.

Duration

Three days

Course Material

Delegate handbook

Course workshops

Delegates will spend a large proportion of the course working on a target embedded system (based on an ARM Cortex M microcontroller) using the FreeRTOS operating system. The exercises are designed to give representative experience of concurrent systems development. Attendees will be able to apply their learning experience to real-world system development.

The C memory model

- The C object model
- Sequential code
- Concurrent code

Scheduling principles

- Context switching
- Preemption
- Round-robin scheduling
- Time-triggered scheduling
- Priority pre-emptive scheduling
- Creating tasks

Task management

- Task creation
- Creating task functions
- Static vs dynamic task creation
- Putting a task to sleep
- Terminating tasks
- Priority management

Mutual exclusion

- Race conditions
- Mutexes
- Building thread-safe resources

Mutual exclusion issues

- Priority inversion
- Priority Inheritance Protocol
- Priority Ceiling Protocol
- Recursive deadlock
- Cyclic dependency deadlock
- Deadlock-through-death

Task synchronisation

- Events
- Conjunctive and disjunctive events
- Unilateral and bilateral synchronisation
- Signals
- Condition variables

Event groups

- Events as signals
- Creating events
- Conjunctive and disjunctive waiting
- Synchronising multiple tasks with events

Semaphore as signal

- Unidirectional, persistent consuming signals
- The Semaphore-as-signal pattern

- The Blocking Monitor pattern
- Counting semaphores
- Bi-lateral synchronisation

Condition variables

- The Guarded Suspension pattern
- Condition variables

Resource pools

- Multiple-Reader, Single-Writer pattern
- Readers-Writer locks

Message queues

- Asynchronous communication with data
- Message queues
- Marshalling and non-marshalling queues
- Queuing policies
- The Asynchronous Message pattern
- Dealing with variable-sized data
- Queue Sets
- Mailboxes

Timers

- Software timers
- The FreeRTOS Daemon task
- One-shot and auto-reload timers
- Timer management

RTOS interaction from interrupts

- Interrupts and the OS
- Communicating from an ISR to a task
- Problems with blocking calls
- The ISR-safe API
- The Deferred Interrupt model
- The Deferred Centralised Interrupt model
- Configuring interrupt priorities

Memory management

- Problems with dynamic memory
- The FreeRTOS memory models
- Dynamic-object lifetime management issues