Formation FreeRTOS Real Time Programming: Real-time programming applied to the FreeRTOS operating system - Programmation: Temps réel

RT3 - FreeRTOS Real Time Programming

Real-time programming applied to the FreeRTOS operating system

Objectives

- Get an overview on Cortex-M4 architecture
- Discover the concepts of real time multitasking
- Understand Real Time constraints
  - Determinism
  - Preemption
  - Interrupts
- Understand the FreeRTOS architecture
- Discover the various FreeRTOS services and APIs
- Learn how to develop FreeRTOS applications
- Learn how to debug FreeRTOS applications

Course environment

- Convenient course material with space for taking notes
- Example code, labs and solutions
- A PC under Windows 7 for two trainees
- A ST STM32F4 (Cortex/M4) with System Workbench IDE

Prerequisites

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

Plan

First Day

Cortex-M resources used by RTOS

- Cortex-M Architecture Overview
  - Two stacks pointers
  - Different Running-modes and Privileged Levels
  - MPU Overview
  - Systick Timer Description
- Exception / Interrupt Mechanism Overview

- Interrut entry and return Overview
- SVC / PendSV / Systick Interrupt Presentation
- Developing with the IDE

Exercice : *Interrupt Management on Cortex-M4*

Element of a real time system

- Base real time concepts
- The Real Time constraints
- Multi-task and real time
- 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 FReeRTOS
  - Deterministic preemptive scheduling
  - Scheduling strategies
  - Cooperative scheduling
  - Hybrid scheduling

Exercice : *Analyse a Context Switch*

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
- Timing

Exercice : *Managing tasks*

Second Day

FreeRTOS Memory Management

- FreeRTOS Memory Managers
- Out of Memory management
- Stack Overflow Management

Exercice : *Check stack usage in existing programs*

Resource Management

- Mutual exclusion through FreeRTOS
  - Critical sections (interrupt masking)
  - Suspending (locking) the scheduler
  - Mutexes
- Mutexes concepts

- Mutex or Semaphore
- Recursive or not recursive mutexes
- Priority inversion problem
- Priority inheritance (the automatic answer)
- Priority ceiling (the design centric answer)

- Gatekeeper tasks

Exercice : *Implement mutual exclusion between tasks*

Synchronization Primitives

- Introduction
  - Waiting and waking up tasks
  - Semaphores
  - Events
  - Mailboxes
- Binary Semaphores through FreeRTOS
  - Give a Binary Semaphore
  - Take a binary Semaphore
- Queue Management through FreeRTOS
  - Creation
  - Sending on a queue
  - Receiving from a queue
  - Data management
  - Sending compound types
  - Transfering large data
- Event groups

Exercice : *Synchronizing a task with another one through binary semaphores*

Exercice : *Synchronizing a task with another one through queues*

Parallelism Problems Solution

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

Exercice : *The producer-consumer problem, illustrating (and avoiding) concurrent access problems*

Exercice : *The philosophers dinner problem, illustrating (and avoiding) deadlock, livelock and starvation*

Third Day

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 FreeRTOS
  - Tasks with interrupt synchronization
  - Using semaphores within an ISR
  - Counting semaphores
  - Using queues within an ISR

- FreeRTOS interrupt processing
  - Writing ISRs in C
  - Interrupt safe functions
  - Interrupt nesting

Exercice : *Synchronize Interrupts with tasks*

Software Timer

- The Timer Daemon Task
- Timer Configuration
- One-shot / Auto-reload Timer
- Software Timer API
- Deferred interrupt handling

Exercice : *Implement Soft Timers*

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

Exercice : *Implement protected memory regions*

Annexes

Data structures

- Need for specific data structures
- Data structures
  - Linked lists
  - Circular lists
  - FIFOs
  - Stacks
- Data structures integrity proofs
  - Assertions
  - Pre and post-conditions
- Thread safety

Exercice : *Build a general purpose linked list*

Memory Management

- Memory management algorithms
  - Buddy System
  - Best fit / First Fit
  - Pools 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

- Memory management errors
- Stack monitoring

Exercice : *Write a simple, thread safe, buddy system memory manager*
Exercice : *Write a generic, multi-level, memory manager*
Exercice : *Enhance the memory manager for memory error detection*

Renseignements pratiques

Durée : 3 jours
Prix : 1900 € HT