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

- 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

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

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

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

Exercise: *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

Exercise: *Analyse a Context Switch*

Task Management

- Creating Tasks
- Task Priorities
- Task States
- The idle task
- Delays
- Changing Task Priority
- Deleting Tasks
- Suspending Tasks
- Kernel Structures
- Thread Local Storage
- Kernel Interrupts on Cortex-M4

- Scheduling Traces
- Visual trace diagnostics using Tracealyzer

Exercise: *Task Management*
Exercise: *Periodic Tasks*
Exercise: *Task Statistics*

Second Day

FreeRTOS Memory Management

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

Exercise: *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

Exercise: *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
- Task Notifications
- Stream Buffers and Message Buffers

Exercise: *Synchronizing a task with another one through binary semaphores*
Exercise: *Synchronizing a task with another one through queues*
Exercise: *Task Notifications*
Exercise: *Properly use stream Buffers*
Exercise: *Message Buffers*

Parallelism Problems Solution

- 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*

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

Exercise: *Synchronize Interrupts with tasks*

Software Timer

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

Exercise: *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

Exercise: *Implement protected memory regions*

appendixes

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

Exercise: *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

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

Renseignements pratiques

Inquiry : 3 days