Real Time College

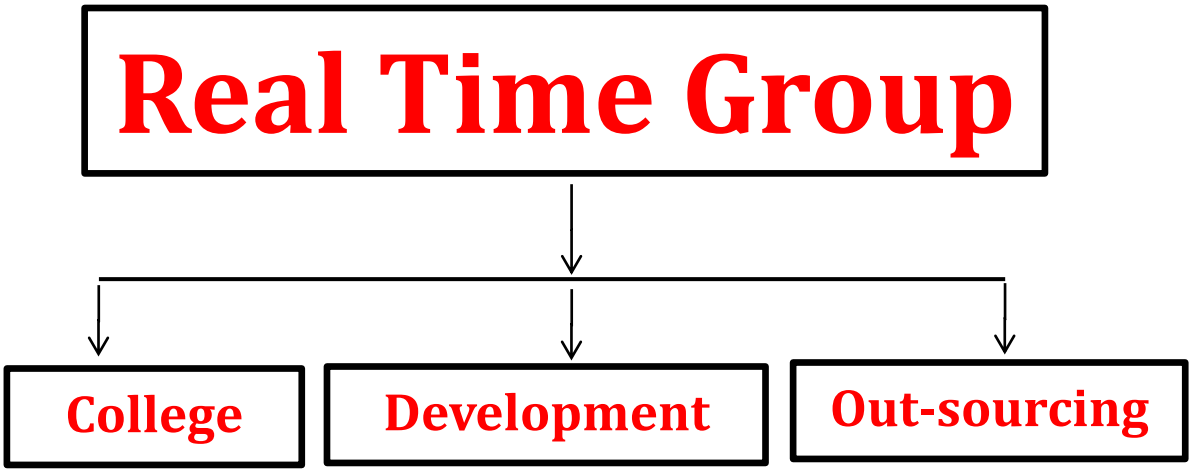
Course: FreeRTOS
Duration: 40 Hours, 5 Days
Hands-On-Training

www.rt-ed.co.il info@rt-ed.co.il 077-7067057
רח' מרדכי רוזנסקי 14, ראשון לציון

Real Time Group is a multi-disciplinary dynamic and innovative Real-Time O.S. and Embedded Software Solutions Center, established in 2007.

Providing Bare-Metal and Embedded Linux solutions, professional services and consulting, end-to-end flexible system infrastructure, outsourcing, integration and training services for Hardware, Software and RT-OS \ Embedded Systems.

The company is divided into the following three Divisions:



rt-ed.co.il rt-dev.com rt-hr.co.il

Training Division:

Professional Training Services for Hardware, Software, RT-OS and Embedded systems industries.

We provide the knowledge and experience needed to enable professional engineers to Develop, Integrate and QA Hardware, Software and Networking Projects.

In order to insure experience, all courses are practical – hands-on-training.
The latest Development, QA and Automation equipment which are adopted by the industry are used.

All students are supplied with Development-Boards for home-work and course projects.

Course Overview:

This Course aims to provide a practical knowledge and understanding of Real-Time kernel usage, Attendees will gain the knowledge and skills needed to develop and maintain applications running industry RTOS application using FreeRTOS.

It starts with the basics, understanding the FreeRTOS Environment, creating new projects, coding styles, task management and drills down to implementation of Interrupts, Queues, communication and synchronization between different tasks, Interfacing Hardware Peripherals in FreeRTOS and implementing secure applications.

This is a Hands-On course, throughout the course we"ll be using Free-RTOS On ARM Cortex M4 evaluation board in order to demonstrate RT Operating system management.

Who should attend:

- HW C\C++ programmers who need to program FreeRTOS systems.
- Engineers who want to Understand RTOS Concepts in FreeRTOS context.
- Anyone who wishes to Use Hardware Peripherals in FreeRTOS Application Development.

Prerequisite:

- Familiarity with developing embedded Systems
- Knowledge in the C programming language.
- Understanding of Operating Systems – advantage but not a must.

FreeRTOS (24 AH)

Day – 1:

1. **Main characteristics**
   (a) Multitasking in Small Embedded Systems.
   (b) Different Real-Time Kernels?
   (c) FreeRTOS overview - why use FreeRTOS.
   (d) Main Features of FreeRTOS.

2. **Scheduling in FreeRTOS**
   (a) Deterministic preemptive scheduling
   (b) Scheduling strategies
   (c) Cooperative scheduling
   (d) Hybrid scheduling

3. **Integrating your project with Free-RTOS**
   (a) Downloading FreeRTOS suitable for your Hardware
   (b) Setting up the hardware for running FreeRTOS based project.
   (c) Locating the project file.
   (d) Building the a new FreeRTOS project.
   (e) Using FreeRTOS APIs.

Lab Exercise-1: Building & Running your first Free-RTOS Application

4. **Coding standards in FreeRTOS**
   (a) Data types.
   (b) Variable Names.
   (c) Function Name.
   (d) Formatting.
   (e) Macro Names.
   (f) Type Casting.

Day – 2:

5. **Task Management**
   (a) Task Functions.
   (b) Creating Tasks.
   (c) Assigning task priorities
   (d) Time Measurement and Tick Interrupt.
   (e) Understanding “no running” state
   (f) The Idle Task.
   (g) Changing priority of a Task.
   (h) Deleting a Task.

   Lab Exercise-2: Running a multi-Tasked program
   Lab Exercise-3: Using Time measurement
   Lab Exercise-4: Deleting Tasks

6. **Queue Management (process to process communication)**
   (a) How to create a Queue.
   (b) How to send\receive Data to\from a Queue.
   (c) How to Block a Queue.
   (d) How to Overwrite\Clear Queue.

   Lab Exercise-5: Communicating between tasks with a Queue
   Lab Exercise-6: Blocking\ Overwriting a Queue

Day – 3:

7. **Resource Management**
   (a) Critical Sections and Suspending Scheduling.
   (b) Mutexes and Task Scheduling.
   (c) Mutexes and Binary Semaphores.
   (d) Counting Semaphores.
   (e) Priority Inversion.
   (f) Deadlocks.
   (g) Livelocks
   (h) Starvation

Lab Exercise-7: Demonstrating a Race Condition
Lab Exercise-8: Using Mutexes for Synchronization
Lab Exercise-9: Demonstrating Priority Inversion
Lab Exercise-9: Demonstrating Deadlocks

8. **Interrupts Management**
   (a) Introduction to Interrupt Management.
   (b) Software Interrupt
   (c) Hardware (Device) Interrupts
   (d) Level or Edge interrupts
   (e) Hardware and Software acknowledge
   (f) Interrupt vectoring
   (g) Interrupts and scheduling
   (h) Deferred Interrupt Proccesing.
   (i) Writing FreeRTOS Interrupts Handler.
   (j) Using Binary\Counting Semaphore in Interrupt Context.
   (k) Using Queues Within Interrupts.
   (l) Nesting Interrupts.

Lab Exercise-10: Implementing an Interrupt handler
Lab Exercise-11: Implementing a deferred Interrupt
Lab Exercise-12: Synchronization within Interrupt context
Lab Exercise-13: Queues within Interrupt context

Day – 4:

9. **Software Timer**
   (a) The Timer Daemon Task
   (b) Timer Configuration
   (c) One-shot / Auto-reload Timer
   (d) Software Timer API
   (e) Deferred interrupt handling

   Lab Exercise-14: Implementing a Soft Timer
   Lab Exercise-15: Implementing an Auto-Reload Timers

10. **Event Groups**
   (a) Characteristics of an Event Group.
   (b) Event Management Using an Event Group.
   (c) Task Synchronization Using an Event Group.

   Lab Exercise-16: Synchronization Using an Event Group

Day – 5:

Lab Exercise-17: Final Project – Putting it all together