



RT2 - MQX Real Time Programming

Real-time programming applied to the MQX operating system

Objectives

- Discover the concepts of real time multitasking
- Understand Real Time constraints
 - Determinism
 - Preemption
 - Interrupts
- Understand the MQX architecture
- Discover the various MQX services and APIs
- Learn how to develop MQX applications
- Learn how to debug MQX applications
- Learn how to use MQX Library (USB, TCP/IP, File System, Embedded GUI)
- Get an overview on Cortex-M4 architecture

Course environment

- Convenient course material with space for taking notes
- Example code, labs and solutions
- A PC under Windows XP for two trainees
- A NXP Kinetis K60 (Cortex/M4) with CodeWarrior IDE

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

MQX at a Glance

- MQX overview
- Organization of MQX
- MQX directory structure
 - RTOS directory
 - PSP, BSP, I/O and others source subdirectories
- Initializing and starting MQX
- Developing with NXP CodeWarrior Development Studio
 - Build projects
 - PSP build-project
 - BSP build-project
 - Post-build processing

Exercise: Creating a simple MQX (only one task) project using NXP CodeWarrior Development Studio

Introduction to Real Time

- Base real time concepts
- The Real Time constraints
- Multi-task and real time

Thread safe data structures

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

Exercise: Build a general purpose thread safe doubly linked list

Second day

Memory Management

- Memory management algorithms
 - Buddy System
 - Best fit
 - First Fit
 - Pools Management
- Understand how MQX RTOS Manages Memory
 - Memory with variable-size blocks
 - Lightweight memory with variable-size blocks
 - Memory with fixed-size blocks
 - Creating partitions
 - Allocating and freeing partition blocks
- Memory management errors
- Stack monitoring
- Controlling caches and MMU Overview (not present on K60 MCU)

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

Exercise: Detect stack overflow

Element of a real time system

- 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

Managing and Scheduling tasks with MQX

- Managing Tasks
 - Creating tasks
 - Managing task errors
 - Terminating tasks
- Scheduling
 - FIFO scheduling
 - Round Robin scheduling

Exercise: Creating your first multi-task project

Third day

Timing with MQX RTOS

- Time components
- Timers
- Lightweight timers
- Watchdogs
- Hardware Timer on Cortex-M4

Exercise: Turn a LED on and off using software timers

Interrupt Management in Real Time Systems

- 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
- Handling Interrupts and Exceptions in MQX RTOS
 - Initializing interrupt handling
 - Restrictions on ISRs
 - Handling exceptions
 - Handling ISR exceptions
 - Handling task exceptions
- Handling Interrupts on Cortex-M4

Exercise: Synchronize Interrupts with tasks

Synchronization Primitives (1/2)

- Waiting and waking up tasks
- Semaphores
- Events
- Semaphores and Events through MQX RTOS
 - Events
 - Lightweight events
 - Lightweight semaphores
 - Semaphores

Exercise: Synchronizing a task with another

Fourth day

Synchronization Primitives (2/2)

- Mutual Exclusion
 - Spinlocks and interrupt masking

- Mutex or Semaphore
- Recursive or not recursive mutexes
- Priority inversion problem
- Priority inheritance (the automatic answer)
- Priority ceiling (the design centric answer)
- Mutexes and condition variables
- Mutual Exclusion through MQX RTOS
 - Priority inversion
 - Priority inheritance
 - Priority protection
 - Mutexes
- Mailboxes
- Using Mailboxes through MQX RTOS
 - Messages
 - Queues
 - Task Queues

Exercise: Implement mutual exclusion between two tasks

Exercise: Synchronizing tasks using queues

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

Debugging the application

- Instrumentation
 - Logs
 - Lightweight logs
 - Kernel logs
 - Stack usage utilities
- Run-Time Testing
- Embedded Debug

Exercise: Debug an application with the log component

Fifth day

Developing a New BSP

- Selecting a Baseline BSP
- Modifying Source Code

USB Library

- USB Host/Device Library Description
- USB Host/Device Stack Directory Structure
- USB Host and USB Device APIs Overview

Exercise: Run a NXP USB HID Example

File System (MFS) Library

- MFS at a Glance

- MFS API Overview

Exercise: Use of MFS accessing the SPI-connect SD Card

Shell Library

- Shell Library Overview

TCP/IP Stack (RTCS) Library

- RTCS Library Overview
- Setting up the RTCS
- Using Sockets
- RTCS API Overview
- RTCS Applications

Exercise: Shell Command line providing microprocessor state information Demo

Exercise: Simple Web Server Demo

Embedded GUI Library

- eGUI Library Overview
- Graphic Object Description
- Structure of Project with eGUI
- Driver API Overview

Exercise: Embedded GUI Demo