

# CONTENTS

---

CONTENTS	I
LIST OF FIGURES .....	V
LIST OF CODE LISTINGS .....	VI
LIST OF TABLES .....	IX
LIST OF NOTATION .....	X
CHAPTER 1 TASK MANAGEMENT .....	1-1
1.1 CHAPTER INTRODUCTION AND SCOPE .....	1-2
An Introduction to Multi Tasking in Small Embedded Systems .....	1-2
A Note About Terminology .....	1-2
Scope .....	1-3
1.2 TASK FUNCTIONS .....	1-4
1.3 TOP LEVEL TASK STATES .....	1-5
1.4 CREATING TASKS .....	1-6
xTaskCreate() API Function .....	1-6
Example 1. Creating Tasks .....	1-8
Example 2. Using the Task Parameter .....	1-12
1.5 TASK PRIORITIES .....	1-15
Example 3. Experimenting with priorities .....	1-16
1.6 EXPANDING THE 'NOT RUNNING' STATE .....	1-19
The Blocked State .....	1-19
The Suspended State .....	1-19
The Ready State .....	1-20
Completing the State Transition Diagram .....	1-20
Example 4. Using the Blocked state to create a delay .....	1-20
vTaskDelayUntil() API function .....	1-24
1.7 THE IDLE TASK AND THE IDLE TASK HOOK .....	1-29
Idle Task Hook Functions .....	1-29
Limitations on the Implementation of Idle Task Hook Functions .....	1-29
Example 7. Defining an Idle Task Hook Function .....	1-30
1.8 CHANGING THE PRIORITY OF A TASK .....	1-32
vTaskPrioritySet() API function .....	1-32
uxTaskPriorityGet() API function .....	1-32
Example 8. Changing task priorities .....	1-33
1.9 DELETING A TASK .....	1-38
vTaskDelete() API function .....	1-38

Example 9.  Deleting tasks .....	1-39
1.10  THE SCHEDULING ALGORITHM – A SUMMARY .....	1-42
Prioritized Preemptive Scheduling.....	1-42
Selecting Task Priorities.....	1-43
Co-operative Scheduling .....	1-44
<b>CHAPTER 2  QUEUE MANAGEMENT .....</b>	<b>2-45</b>
2.1  CHAPTER INTRODUCTION AND SCOPE .....	2-46
Scope.....	2-46
2.2  CHARACTERISTICS OF A QUEUE .....	2-47
Data Storage.....	2-47
Access by Multiple Tasks .....	2-47
Blocking on Queue Reads.....	2-47
Blocking on Queue Writes.....	2-47
2.3  USING A QUEUE .....	2-49
xQueueCreate() API Function .....	2-49
xQueueSendToBack() and xQueueSendToFront() API Functions.....	2-50
xQueueReceive() and xQueuePeek() API Functions.....	2-51
uxQueueMessagesWaiting() API Function.....	2-53
Example 10.  Blocking When Receiving From a Queue.....	2-54
Using Queues to Transfer Compound Types .....	2-58
Example 11.  Blocking When Sending to a Queue / Sending Structures on a Queue.....	2-59
2.4  WORKING WITH LARGE DATA .....	2-66
<b>CHAPTER 3  INTERRUPT MANAGEMENT .....</b>	<b>3-67</b>
3.1  CHAPTER INTRODUCTION AND SCOPE .....	3-68
Events.....	3-68
Scope.....	3-68
3.2  DEFERRED INTERRUPT PROCESSING .....	3-69
Binary Semaphores used for Synchronization.....	3-69
vSemaphoreCreateBinary() API Function.....	3-70
xSemaphoreTake() API Function .....	3-72
xSemaphoreGiveFromISR() API Function.....	3-74
Example 12.  Using a Binary Semaphore to Synchronize a Task with an Interrupt.....	3-75
3.3  COUNTING SEMAPHORES.....	3-80
xSemaphoreCreateCounting() API Function .....	3-83
Example 13.  Using a Counting Semaphore to Synchronize a Task with an Interrupt.....	3-84
3.4  USING QUEUES WITHIN AN INTERRUPT SERVICE ROUTINE .....	3-87
xQueueSendToFrontFromISR() and xQueueSendToBackFromISR() API Functions.....	3-87
Efficient Queue Usage .....	3-88
Example 14.  Sending and Receiving on a Queue from Within an Interrupt.....	3-89
3.5  INTERRUPT NESTING .....	3-94

A Note to ARM Cortex M3 Users.....	3-95
CHAPTER 4    RESOURCE MANAGEMENT .....	4-96
4.1    CHAPTER INTRODUCTION AND SCOPE .....	4-97
Mutual Exclusion .....	4-100
Scope.....	4-100
4.2    CRITICAL SECTIONS AND SUSPENDING THE SCHEDULER.....	4-101
Basic Critical Sections.....	4-101
Suspending (or Locking) the Scheduler.....	4-102
vTaskSuspendAll() API Function.....	4-103
xTaskResumeAll() API Function.....	4-103
4.3    MUTEXES (AND BINARY SEMAPHORES).....	4-105
xSemaphoreCreateMutex() API Function .....	4-107
Example 15.  Rewriting vPrintString() to Use a Semaphore.....	4-107
Priority Inversion .....	4-111
Priority Inheritance .....	4-112
Deadlock (or Deadly Embrace) .....	4-113
4.4    GATEKEEPER TASKS .....	4-115
Example 16.  Re-writing vPrintString() to Use a Gatekeeper Task.....	4-115
CHAPTER 5    MEMORY MANAGEMENT .....	5-121
5.1    CHAPTER INTRODUCTION AND SCOPE .....	5-122
Scope.....	5-123
5.2    EXAMPLE MEMORY ALLOCATION SCHEMES.....	5-124
Heap_1.c.....	5-124
Heap_2.c.....	5-124
Heap_3.c.....	5-126
CHAPTER 6    TROUBLE SHOOTING.....	6-128
6.1    CHAPTER INTRODUCTION AND SCOPE .....	6-129
printf-stdarg.c .....	6-129
6.2    STACK OVERFLOW .....	6-130
uxTaskGetStackHighWaterMark() API Function.....	6-130
Run Time Stack Checking - Overview .....	6-131
Run Time Stack Checking - Method 1 .....	6-131
Run Time Stack Checking - Method 2 .....	6-131
6.3    OTHER COMMON SOURCES OF ERROR.....	6-133
Symptom: Adding a Simple Task to a Demo Causes the Demo to Crash.....	6-133
Symptom: Using an API Function Within an Interrupt Causes the Application to Crash.....	6-133
Symptom: Sometimes the Application Crashes within an Interrupt Service Routine .....	6-133
Symptom: The Scheduler Crashes When Attempting to Start the First Task .....	6-133
Symptom: Critical Sections Do Not Nest Correctly .....	6-134

Symptom: The Application Crashes Even Before the Scheduler is Started .....	6-134
Symptom: Calling API Functions While the Scheduler is Suspended Causes the Application to Crash .....	6-134
Symptom: The Prototype For pxPortInitialiseStack() Causes Compilation to Fail .....	6-134
APPENDIX 1: BUILDING THE EXAMPLES .....	6-135
APPENDIX 2: THE DEMO APPLICATIONS.....	6-136
APPENDIX 3: FREERTOS FILES AND DIRECTORIES.....	6-138
Removing Unused Files .....	6-139
APPENDIX 4: CREATING A FREERTOS PROJECT .....	6-140
Adapting One of the Supplied Demo Projects .....	6-140
Creating a New Project from Scratch .....	6-141
Header Files.....	6-142
APPENDIX 5: DATA TYPES AND CODING STYLE GUIDE .....	6-143
Data Types.....	6-143
Variable Names.....	6-144
Function Names.....	6-144
Formatting.....	6-144
Macro Names.....	6-144
Rationale for Excessive Type Casting .....	6-145
APPENDIX 6: LICENSING INFORMATION .....	6-146
Open Source License Details .....	6-147
GPL Exception Text .....	6-147
INDEX	6-148

## LIST OF FIGURES

Figure 1 Top level task states and transitions. ....	1-5
Figure 2 The output produced when Example 1 is executed .....	1-10
Figure 3 The actual execution pattern of the two Example 1 tasks .....	1-11
Figure 4 The execution sequence expanded to show the tick interrupt executing. ....	1-16
Figure 5 Running both test tasks at different priorities.....	1-17
Figure 6 The execution pattern when one task has a higher priority than the other .....	1-18
Figure 7 Full task state machine .....	1-20
Figure 8 The output produced when Example 4 is executed .....	1-22
Figure 9 The execution sequence when the tasks use vTaskDelay() in place of the NULL loop.....	1-23
Figure 10 Bold lines indicate the state transitions performed by the tasks in Example 4 .....	1-24
Figure 11 The output produced when Example 6 is executed. ....	1-28
Figure 12 The execution pattern of Example 6.....	1-28
Figure 13 The output produced when Example 7 is executed .....	1-31
Figure 14 The sequence of task execution when running Example 8.....	1-36
Figure 15 The output produced when Example 8 is executed .....	1-37
Figure 16 The output produced when Example 9 is executed .....	1-40
Figure 17 The execution sequence for example 9.....	1-41
Figure 18 Execution pattern with pre-emption points highlighted.....	1-42
Figure 19 An example sequence of writes and reads to/from a queue .....	2-48
Figure 20 The xQueueReceive() API function prototype .....	2-52
Figure 21 The output produced when Example 10 is executed .....	2-58
Figure 22 The sequence of execution produced by Example 10 .....	2-58
Figure 23 An example scenario where structures are sent on a queue .....	2-59
Figure 24 The output produced by Example 11.....	2-64
Figure 25 The sequence of execution produced by Example 11 .....	2-64
Figure 26 The interrupt interrupts one task, but returns to another.....	3-69
Figure 27 Using a binary semaphore to synchronize a task with an interrupt .....	3-71
Figure 28 The output produced when Example 12 is executed .....	3-79
Figure 29 The sequence of execution when Example 12 is executed .....	3-79
Figure 30 A binary semaphore can latch at most one event.....	3-81
Figure 31 Using a counting semaphore to 'count' events .....	3-82
Figure 32 The output produced when Example 13 is executed .....	3-86
Figure 33 The output produced when Example 14 is executed .....	3-93
Figure 34 The sequence of execution produced by Example 14 .....	3-93
Figure 35 Constants affecting interrupt nesting behavior.....	3-95
Figure 36 Mutual exclusion implemented using a mutex .....	4-106
Figure 37 The output produced when Example 15 is executed .....	4-110
Figure 38 A possible sequence of execution for Example 15 .....	4-111

Figure 39 A worst case priority inversion scenario .....	4-112
Figure 40 Priority inheritance minimizing the effect of priority inversion.....	4-113
Figure 41 The output produced when Example 16 is executed .....	4-120
Figure 42 RAM being allocated within the array each time a task is created .....	5-124
Figure 43 RAM being allocated from the array as tasks are created and deleted .....	5-125
Figure 44 Locating the demo application documentation in the menu frame of the FreeRTOS.org WEB site.....	6-137
Figure 45 The top level directories – Source and Demo.....	6-138
Figure 46 The three core files that implement the FreeRTOS kernel.....	6-139

## LIST OF CODE LISTINGS

---

Listing 1 The task function prototype.....	1-4
Listing 2 The structure of a typical task function.....	1-4
Listing 3 The xTaskCreate() API function prototype .....	1-6
Listing 4 Implementation of the first task used in Example 1 .....	1-9
Listing 5 Implementation of the second task used in Example 1.....	1-9
Listing 6 Starting the Example 1 tasks .....	1-10
Listing 7 Creating a task from within another task – after the scheduler has started.....	1-12
Listing 8 The single task function used to create two tasks in Example 2.....	1-13
Listing 9 The main() function for Example 2. ....	1-14
Listing 10 Creating two tasks at different priorities .....	1-17
Listing 11 The vTaskDelay() API function prototype.....	1-21
Listing 12 The source code for the example task after the null loop delay has been replaced by a call to vTaskDelay().....	1-22
Listing 13 vTaskDelayUntil() API function prototype.....	1-24
Listing 14 The implementation of the example task using vTaskDelayUntil().....	1-26
Listing 15 The continuous processing task used in Example 6.....	1-27
Listing 16 The periodic task used in Example 6. ....	1-27
Listing 17 The idle task hook function name and prototype.....	1-30
Listing 18 A very simple Idle hook function.....	1-30
Listing 19 The source code for the example task now prints out the ulIdleCycleCount value .....	1-31
Listing 20 The vTaskPrioritySet() API function prototype.....	1-32
Listing 21 The uxTaskPriorityGet() API function prototype .....	1-32
Listing 22 The implementation of Task1 in Example 8 .....	1-34
Listing 23 The implementation of Task2 in Example 8 .....	1-35
Listing 24 The implementation of main() for Example 8.....	1-36
Listing 25 The vTaskDelete() API function prototype.....	1-38
Listing 26 The implementation of main() for Example 9.....	1-39

Listing 27 The implementation of Task 1 for Example 9 .....	1-40
Listing 28 The implementation of Task 2 for Example 9 .....	1-40
Listing 29 The xQueueCreate() API function prototype .....	2-49
Listing 30 The xQueueSendToFront() API function prototype .....	2-50
Listing 31 The xQueueSendToBack() API function prototype.....	2-50
Listing 32 The xQueuePeek() API function prototype.....	2-52
Listing 33 The uxQueueMessagesWaiting() API function prototype .....	2-54
Listing 34 Implementation of the sending task used in Example 10.....	2-55
Listing 35 Implementation of the receiver task for Example 10.....	2-56
Listing 36 The implementation of main()Example 10.....	2-57
Listing 37 The definition of the structure that is to be passed on a queue, plus the declaration of two variables for use by the example.....	2-60
Listing 38 The implementation of the sending task for Example 11.....	2-61
Listing 39 The definition of the receiving task for Example 11 .....	2-62
Listing 40 The implementation of main() for Example 11.....	2-63
Listing 41 The vSemaphoreCreateBinary() API function prototype.....	3-70
Listing 42 The xSemaphoreTake() API function prototype .....	3-72
Listing 43 The xSemaphoreGiveFromISR() API function prototype.....	3-74
Listing 44 Implementation of the task that periodically generates a software interrupt in Example 12 . 3-76	
Listing 45 The implementation of the handler task (the task that synchronizes with the interrupt) in Example 12.....	3-76
Listing 46 The software interrupt handler used in Example 12 .....	3-77
Listing 47 The implementation of main() for Example 12.....	3-78
Listing 48 The xSemaphoreCreateCounting() API function prototype.....	3-83
Listing 49 Using xSemaphoreCreateCounting() to create a counting semaphore.....	3-85
Listing 50 The implementation of the interrupt service routine used by Example 13.....	3-85
Listing 51 The xQueueSendToFrontFromISR() API function prototype .....	3-87
Listing 52 The xQueueSendToBackFromISR() API function prototype.....	3-87
Listing 53 The implementation of the task that writes to the queue in Example 14 .....	3-90
Listing 54 The implementation of the interrupt service routine used by Example 14.....	3-91
Listing 55 The task that prints out the strings received from the interrupt service routine in Example 14 .....	3-92
Listing 56 The main() function for Example 14 .....	3-92
Listing 57 An example read, modify, write sequence.....	4-97
Listing 58 An example of a reentrant function .....	4-99
Listing 59 An example of a function that is not reentrant .....	4-99
Listing 60 Using a critical section to guard access to a register.....	4-101
Listing 61 The implementation of vPrintString().....	4-102
Listing 62 The vTaskSuspendAll() API function prototype .....	4-103
Listing 63 The xTaskResumeAll() API function prototype.....	4-103
Listing 64 Suspending the scheduler to guard access to an array.....	4-104

Listing 65 The xSemaphoreCreateMutex() API function prototype .....	4-107
Listing 66 The implementation of prvNewPrintString().....	4-108
Listing 67 The implementation of prvPrintTask() for Example 15.....	4-109
Listing 68 The implementation of main() for Example 15.....	4-110
Listing 69 The name and prototype for a tick hook function.....	4-115
Listing 70 The gatekeeper task .....	4-116
Listing 71 The print task implementation for Example 16 .....	4-117
Listing 72 The tick hook implementation .....	4-118
Listing 73 The implementation of main() for Example 16.....	4-119
Listing 74 The heap_3.c implementation.....	5-127
Listing 75 The uxTaskGetStackHighWaterMark() API function prototype.....	6-130
Listing 76 The stack overflow hook function prototype .....	6-131
Listing 77 The template for a new main() function.....	6-141



## LIST OF TABLES

---

Table 1 xTaskCreate() parameters and return value .....	1-6
Table 2 vTaskDelay() parameters .....	1-21
Table 3 vTaskDelayUntil() parameters .....	1-25
Table 4 vTaskPrioritySet() parameters .....	1-32
Table 5 uxTaskPriorityGet() parameters and return value .....	1-33
Table 6 vTaskDelete() parameters .....	1-38
Table 7 xQueueCreate() parameters and return value .....	2-49
Table 8 xQueueSendToFront() and xQueueSendToBack() function parameters and return value ..	2-50
Table 9 xQueueReceive() and xQueuePeek() function parameters and return values .....	2-52
Table 10 uxQueueMessagesWaiting() function parameters and return value.....	2-54
Table 11 Key to Figure 25.....	2-65
Table 12 vSemaphoreCreateBinary() parameters .....	3-70
Table 13 xSemaphoreTake() parameters and return value .....	3-73
Table 14 xSemaphoreTake() parameters and return value .....	3-75
Table 15 xSemaphoreCreateCounting() parameters and return value .....	3-84
Table 16 xQueueSendToFrontFromISR() and xQueueSendToBackFromISR() parameters and return values .....	3-88
Table 17 Constants that control interrupt nesting .....	3-94
Table 18 xTaskResumeAll() Return Value .....	4-103
Table 19 xSemaphoreCreateMutex() Return Value.....	4-107
Table 20 uxTaskGetStackHighWaterMark() parameters and return value.....	6-130
Table 21 FreeRTOS source files to include in the project.....	6-142
Table 22 Data types used by FreeRTOS .....	6-143
Table 23 Macro prefixes.....	6-145
Table 24 Common macro definitions.....	6-145
Table 25 Open Source Vs Commercial License Comparison.....	6-146