

Using the FreeRTOS™ Real Time Kernel

PIC32 Edition

Richard Barry

Contents

List of Figures	v
List of Code Listings	vii
List of Tables	x
List of Notation.....	xi
Preface	
FreeRTOS and the PIC32.....	1
Multitasking on a PIC32 Microcontroller	2
An Introduction to Multitasking in Small Embedded Systems	2
A Note About Terminology	2
Why Use a Real-time Kernel?	3
The PIC32 Port of FreeRTOS	5
The FreeRTOS, OpenRTOS, and SafeRTOS Family.....	5
Using the Examples that Accompany this Book.....	8
Required Tools and Hardware	8
Opening the Example Workspaces	10
Connecting the PIC32 Starter Kit	10
Starting a Debug Session	10
Chapter 1 Task Management.....	13
1.1 Chapter Introduction and Scope.....	14
Scope	14
1.2 Task Functions.....	15
1.3 Top Level Task States	16
1.4 Creating Tasks.....	17
The xTaskCreate() API Function.....	17
Example 1. Creating tasks	20
Example 2. Using the task parameter	23
1.5 Task Priorities	26
Example 3. Experimenting with priorities.....	27
1.6 Expanding the 'Not Running' State.....	30
The Blocked State.....	30
The Suspended State	31
The Ready State.....	31
Completing the State Transition Diagram.....	31
Example 4. Using the Blocked state to create a delay.....	32
The vTaskDelayUntil() API Function	35
Example 5. Converting the example tasks to use vTaskDelayUntil().....	37
Example 6. Combining blocking and non-blocking tasks.....	38
1.7 The Idle Task and the Idle Task Hook.....	41
Idle Task Hook Functions.....	41

Limitations on the Implementation of Idle Task Hook Functions	42
Example 7. Defining an Idle task hook function	42
1.8 Changing the Priority of a Task	44
The vTaskPrioritySet() API Function	44
The uxTaskPriorityGet() API Function	44
Example 8. Changing task priorities	45
1.9 Deleting a Task	50
The vTaskDelete() API Function	50
Example 9. Deleting tasks	51
1.10 The Scheduling Algorithm—A Summary	54
Prioritized Pre-emptive Scheduling	54
Selecting Task Priorities	56
Co-operative Scheduling	56
Chapter 2 Queue Management	59
2.1 Chapter Introduction and Scope	60
Scope	60
2.2 Characteristics of a Queue	61
Data Storage	61
Access by Multiple Tasks	61
Blocking on Queue Reads	61
Blocking on Queue Writes	62
2.3 Using a Queue	64
The xQueueCreate() API Function	64
The xQueueSendToBack() and xQueueSendToFront() API Functions	65
The xQueueReceive() and xQueuePeek() API Functions	67
The uxQueueMessagesWaiting() API Function	70
Example 10. Blocking when receiving from a queue	71
Using Queues to Transfer Compound Types	75
Example 11. Blocking when sending to a queue or sending structures on a queue	77
2.4 Working with Large Data	83
Chapter 3 Interrupt Management	85
3.1 Chapter Introduction and Scope	86
Events	86
Scope	86
3.2 Deferred Interrupt Processing	88
Binary Semaphores Used for Synchronization	88
Writing FreeRTOS Interrupt Handlers	89
The vSemaphoreCreateBinary() API Function	92
The xSemaphoreTake() API Function	94
The xSemaphoreGiveFromISR() API Function	95
Example 12. Using a binary semaphore to synchronize a task with an interrupt	97
3.3 Counting Semaphores	102

The xSemaphoreCreateCounting() API Function	105
Example 13. Using a counting semaphore to synchronize a task with an interrupt.....	107
3.4 Using Queues within an Interrupt Service Routine	109
The xQueueSendToFrontFromISR() and xQueueSendToBackFromISR() API Functions	109
Efficient Queue Usage	111
Example 14. Sending and receiving on a queue from within an interrupt	111
3.5 Interrupt Nesting	116
Chapter 4 Resource Management	119
4.1 Chapter Introduction and Scope.....	120
Mutual Exclusion.....	122
Scope	123
4.2 Critical Sections and Suspending the Scheduler	124
Basic Critical Sections	124
Suspending (or Locking) the Scheduler	125
The vTaskSuspendAll() API Function.....	126
The xTaskResumeAll() API Function	126
4.3 Mutexes (and Binary Semaphores).....	128
The xSemaphoreCreateMutex() API Function.....	130
Example 15. Rewriting vPrintString() to use a semaphore	130
Priority Inversion	133
Priority Inheritance	134
Deadlock (or Deadly Embrace)	135
4.4 Gatekeeper Tasks.....	137
Example 16. Re-writing vPrintString() to use a gatekeeper task.....	137
Chapter 5 Memory Management.....	143
5.1 Chapter Introduction and Scope.....	144
Scope	145
5.2 Example Memory Allocation Schemes	146
Heap_1.c	146
Heap_2.c	147
Heap_3.c	149
The xPortGetFreeHeapSize() API Function	149
Chapter 6 Trouble Shooting	153
6.1 Chapter Introduction and Scope.....	154
printf-stdarg.c.....	154
6.2 Stack Overflow.....	155
The uxTaskGetStackHighWaterMark() API Function	155
Run Time Stack Checking—Overview	156
Run Time Stack Checking—Method 1	156
Run Time Stack Checking—Method 2	157
6.3 Other Common Sources of Error.....	158

Symptom: Adding a simple task to a demo causes the demo to crash	158
Symptom: Using an API function within an interrupt causes the application to crash....	158
Symptom: Sometimes the application crashes within an interrupt service routine	159
Symptom: Critical sections do not nest correctly	159
Symptom: The application crashes even before the scheduler is started.....	159
Symptom: Calling API functions while the scheduler is suspended causes the application to crash	159
Symptom: The prototype for pxPortInitialiseStack() causes compilation to fail.....	160
6.4 Execution Visualization—Trace Hook Macros	161
Available Trace Hook Macros.....	161
Defining Trace Hook Macros	170
Example 17. Using Trace Hook Macros	171
6.5 Execution Visualization—Run Time Statistics.....	176
Run Time Statistics Time Base	176
Configuring an Application to Collect Run Time Statistics	177
The vTaskGetRunTimeStats() API Function.....	178
Example 18. Generating Run Time Statistics	179
Chapter 7 The FreeRTOS Download	185
7.1 Chapter Introduction and Scope	186
Scope.....	186
7.2 Files and Directories.....	187
Removing Unused Source Files	188
7.3 Demo Applications	189
Removing Unused Demo Files	190
7.4 Creating a FreeRTOS Project.....	191
Adapting One of the Supplied Demo Projects	191
Creating a New Project from Scratch	192
Header Files.....	193
7.5 Data Types and Coding Style Guide.....	194
Data Types.....	194
Variable Names.....	195
Function Names	195
Formatting.....	195
Macro Names.....	195
Rationale for Excessive Type Casting	196
Appendix 1: Licensing Information.....	199
Open Source License Details	200
GPL Exception Text	201
INDEX.....	203

List of Figures

Figure 1. PIC32 USB Starter Kit II (Microchip part number DM320003-2).....	8
Figure 2. Block diagram of the PIC32MX795F512L-80I/PT.....	9
Figure 3. Selecting 'PIC32 Starter Kit' as the debug interface.....	11
Figure 4. The 'Run' speed button (highlighted by the box with round corners)	11
Figure 5. Top level task states and transitions	16
Figure 6. The output produced when Example 1 is executed.....	21
Figure 7. The execution pattern of the two Example 1 tasks	22
Figure 8. The execution sequence expanded to show the tick interrupt executing	27
Figure 9. Running both test tasks at different priorities	28
Figure 10. The execution pattern when one task has a higher priority than the other.....	29
Figure 11. Full task state machine	32
Figure 12. The output produced when Example 4 is executed.....	34
Figure 13. The execution sequence when the tasks use vTaskDelay() in place of the NULL loop.....	34
Figure 14. Bold lines indicate the state transitions performed by the tasks in Example 4	35
Figure 15. The output produced when Example 6 is executed.....	39
Figure 16. The execution pattern of Example 6.....	40
Figure 17. The output produced when Example 7 is executed.....	43
Figure 18. The sequence of task execution when running Example 8.....	48
Figure 19. The output produced when Example 8 is executed.....	49
Figure 20. The output produced when Example 9 is executed.....	52
Figure 21. The execution sequence for Example 9.....	53
Figure 22. Execution pattern with pre-emption points highlighted	55
Figure 23. An example sequence of writes and reads to and from a queue	63
Figure 24. The output produced when Example 10 is executed.....	75
Figure 25. The sequence of execution produced by Example 10	75
Figure 26. An example scenario where structures are sent on a queue	76
Figure 27. The output produced by Example 11	80
Figure 28. The sequence of execution produced by Example 11	81
Figure 29. The interrupt interrupts one task but returns to another.....	88
Figure 30. Using a binary semaphore to synchronize a task with an interrupt.....	93
Figure 31. The output produced when Example 12 is executed.....	100
Figure 32. The sequence of execution when Example 12 is executed	101
Figure 33. A binary semaphore can latch at most one event.....	103
Figure 34. Using a counting semaphore to 'count' events.....	104
Figure 35. The output produced when Example 13 is executed.....	108
Figure 36. The output produced when Example 14 is executed.....	115
Figure 37. The sequence of execution produced by Example 14.....	115
Figure 38. Constants affecting interrupt nesting behavior	117
Figure 39. Mutual exclusion implemented using a mutex.....	129

Figure 40. The output produced when Example 15 is executed	133
Figure 41. A possible sequence of execution for Example 15	133
Figure 42. A worst case priority inversion scenario	134
Figure 43. Priority inheritance minimizing the effect of priority inversion.....	135
Figure 44. The output produced when Example 16 is executed	141
Figure 45. RAM being allocated within the array each time a task is created	146
Figure 46. RAM being allocated from the array as tasks are created and deleted	148
Figure 47. The output produced when Example 17 is executed	175
Figure 48. The output produced when Example 18 is executed	183
Figure 49. The top-level directories—Source and Demo	187
Figure 50. The three core files that implement the FreeRTOS kernel.....	188
Figure 51. The source directories required to build a PIC32 demo application	188
Figure 52. The demo directories required to build the PIC32 demo application.....	190

List of Code Listings

Listing 1. The task function prototype.....	15
Listing 2. The structure of a typical task function.....	15
Listing 3. The xTaskCreate() API function prototype.....	17
Listing 4. Implementation of the first task used in Example 1.....	20
Listing 5. Implementation of the second task used in Example 1.....	20
Listing 6. Starting the Example 1 tasks.....	21
Listing 7. Creating a task from within another task after the scheduler has started.....	23
Listing 8. The single task function used to create two tasks in Example 2.....	24
Listing 9. The main() function for Example 2.....	25
Listing 10. Creating two tasks at different priorities.....	28
Listing 11. The vTaskDelay() API function prototype.....	33
Listing 12. The source code for the example task after the null loop delay has been replaced by a call to vTaskDelay().....	33
Listing 13. vTaskDelayUntil() API function prototype.....	36
Listing 14. The implementation of the example task using vTaskDelayUntil().....	37
Listing 15. The continuous processing task used in Example 6.....	38
Listing 16. The periodic task used in Example 6.....	39
Listing 17. The Idle task hook function name and prototype.....	42
Listing 18. A very simple Idle hook function.....	42
Listing 19. The source code for the example task prints out the ulIdleCycleCount value.....	43
Listing 20. The vTaskPrioritySet() API function prototype.....	44
Listing 21. The uxTaskPriorityGet() API function prototype.....	44
Listing 22. The implementation of Task 1 in Example 8.....	46
Listing 23. The implementation of Task 2 in Example 8.....	47
Listing 24. The implementation of main() for Example 8.....	48
Listing 25. The vTaskDelete() API function prototype.....	50
Listing 26. The implementation of main() for Example 9.....	51
Listing 27. The implementation of Task 1 for Example 9.....	52
Listing 28. The implementation of Task 2 for Example 9.....	52
Listing 29. The xQueueCreate() API function prototype.....	64
Listing 30. The xQueueSendToFront() API function prototype.....	65
Listing 31. The xQueueSendToBack() API function prototype.....	65
Listing 32. The xQueueReceive() API function prototype.....	68
Listing 33. The xQueuePeek() API function prototype.....	68
Listing 34. The uxQueueMessagesWaiting() API function prototype.....	70
Listing 35. Implementation of the sending task used in Example 10.....	72
Listing 36. Implementation of the receiver task for Example 10.....	73
Listing 37. The implementation of main() for Example 10.....	74
Listing 38. The definition of the structure that is to be passed on a queue, plus the declaration of two variables for use by the example.....	77
Listing 39. The implementation of the sending task for Example 11.....	78

Listing 40. The definition of the receiving task for Example 11	79
Listing 41. The implementation of main() for Example 11	80
Listing 42. The template to use for all interrupt wrapper functions	90
Listing 43. Installing an ISR assembly wrapper in the interrupt vector table	90
Listing 44. The entire assembly file used in Example 12	91
Listing 45. The vSemaphoreCreateBinary() API function prototype	92
Listing 46. The xSemaphoreTake() API function prototype	94
Listing 47. The xSemaphoreGiveFromISR() API function prototype	95
Listing 48. Implementation of the task that periodically generates a software interrupt in Example 12	97
Listing 49. The implementation of the handler task (the task that synchronizes with the interrupt) in Example 12	98
Listing 50. The software interrupt handler used in Example 12	99
Listing 51. The implementation of main() for Example 12	100
Listing 52. The xSemaphoreCreateCounting() API function prototype	105
Listing 53. Using xSemaphoreCreateCounting() to create a counting semaphore	107
Listing 54. The implementation of the interrupt service routine used by Example 13	107
Listing 55. The xQueueSendToFrontFromISR() API function prototype	109
Listing 56. The xQueueSendToBackFromISR() API function prototype	109
Listing 57. The implementation of the task that writes to the queue in Example 14	112
Listing 58. The implementation of the interrupt service routine used by Example 14	113
Listing 59. The task that prints out the strings received from the interrupt service routine in Example 14	114
Listing 60. The main() function for Example 14	114
Listing 61. An example read, modify, write sequence	120
Listing 62. An example of a reentrant function	122
Listing 63. An example of a function that is not reentrant	122
Listing 64. Using a critical section to guard access to a register	124
Listing 65. A possible implementation of vPrintString()	124
Listing 66. The vTaskSuspendAll() API function prototype	126
Listing 67. The xTaskResumeAll() API function prototype	126
Listing 68. The implementation of vPrintString()	127
Listing 69. The xSemaphoreCreateMutex() API function prototype	130
Listing 70. The implementation of prvNewPrintString()	131
Listing 71. The implementation of prvPrintTask() for Example 15	131
Listing 72. The implementation of main() for Example 15	132
Listing 73. The name and prototype for a tick hook function	138
Listing 74. The gatekeeper task	138
Listing 75. The print task implementation for Example 16	139
Listing 76. The tick hook implementation	139
Listing 77. The implementation of main() for Example 16	140
Listing 78. The heap_3.c implementation	149
Listing 79. The xPortGetFreeHeapSize() API function prototype	149
Listing 80. The uxTaskGetStackHighWaterMark() API function prototype	155

Listing 81. The stack overflow hook function prototype	156
Listing 82. The malloc() failed hook function name and prototype.....	158
Listing 83. The trace macros demonstrated by Example 17	173
Listing 84. The definition of the task created by Example 17.....	174
Listing 85. The vTaskGetRunTimeStats() API function prototype.....	178
Listing 86. Configuration of the timer 2 (T2) peripheral used in Example 18.....	179
Listing 87. T2 overflow interrupt handler used in Example 18	180
Listing 88. Macros added to FreeRTOSConfig.h to enable the collection of run time statistics in Example 18	180
Listing 89. The definition of two tasks created in Example 18 to use up some processing time.....	181
Listing 90. The task that prints out the collected run time statistics in Example 18	182
Listing 91. The template for a new main() function.....	192

List of Tables

Table 1. Comparing the FreeRTOS license with the OpenRTOS license	7
Table 2. xTaskCreate() parameters and return value	17
Table 3. vTaskDelay() parameters	33
Table 4. vTaskDelayUntil() parameters	36
Table 5. vTaskPrioritySet() parameters	44
Table 6. uxTaskPriorityGet() parameters and return value	45
Table 7. vTaskDelete() parameters	50
Table 8. xQueueCreate() parameters and return value	64
Table 9. xQueueSendToFront() and xQueueSendToBack() function parameters and return value	65
Table 10. xQueueReceive() and xQueuePeek() function parameters and return values	68
Table 11. uxQueueMessagesWaiting() function parameters and return value	71
Table 12. Key to Figure 28	81
Table 13. vSemaphoreCreateBinary() parameters	92
Table 14. xSemaphoreTake() parameters and return value	94
Table 15. xSemaphoreGiveFromISR() parameters and return value	96
Table 16. xSemaphoreCreateCounting() parameters and return value	106
Table 17. xQueueSendToFrontFromISR() and xQueueSendToBackFromISR() parameters and return values	109
Table 18. Constants that affect interrupt nesting	116
Table 19. xTaskResumeAll() return value	126
Table 20. xSemaphoreCreateMutex() return value	130
Table 21. xPortGetFreeHeapSize() return value	150
Table 22. uxTaskGetStackHighWaterMark() parameters and return value	155
Table 23. Trace hook macros	161
Table 24. Macros used in the collection of run time statistics	177
Table 25. vTaskGetRunTimeStats() parameters	178
Table 26. FreeRTOS source files to include in the project	193
Table 27. Special data types used by FreeRTOS	194
Table 28. Macro prefixes	196
Table 29. Common macro definitions	196
Table 30. Comparing the open source license with the commercial license	200