

# **Using the FreeRTOS™ Real Time Kernel**



# **Using the FreeRTOS™ Real Time Kernel**

**Renesas RX600 Edition**

Richard Barry

First edition published 2011.

All text, source code and diagrams are the exclusive property of Real Time Engineers Ltd. Distribution, use in presentations, or publication in any form is strictly prohibited without prior written authority from Real Time Engineers Ltd.

© Real Time Engineers Ltd. 2011. All rights reserved.

FreeRTOS™, FreeRTOS.org™ and the FreeRTOS logo are trademarks of Real Time Engineers Ltd.

RENESAS® is a registered trademark of Renesas Electronics Corporation.

OPENRTOS™ and SAFERTOS™ are trademarks of WITTENSTEIN Aerospace and Simulation Ltd.

IAR™ and IAR Embedded Workbench® are trademarks or registered trademarks of IAR Systems AB. All other brands or product names are the property of their respective holders.

<http://www.freertos.org>

ISBN 978-1-4467-6274-5

I would like to thank my wife for putting up with a husband who has paid her much less attention than she deserves.



# Contents

---

List of Figures .....	v
List of Code Listings .....	vii
List of Tables .....	x
List of Notation.....	xi
Preface	
FreeRTOS and the RX600.....	1
Multitasking on an RX600 Family Microcontroller.....	2
An Introduction to Multitasking in Small Embedded Systems .....	2
A Note About Terminology .....	2
Why Use a Real-time Kernel? .....	3
The RX600 Port of FreeRTOS .....	5
Resources Used By FreeRTOS .....	6
The FreeRTOS, OpenRTOS, and SafeRTOS Family.....	6
Using the Examples that Accompany this Book.....	9
Required Tools and Hardware .....	9
Opening the Example Workspace.....	9
Building the Examples .....	10
Programming the RX62N Flash Memory.....	10
Starting a Debug Session .....	11
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 .....	36
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 .....	76
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 .....	89
The xSemaphoreTake() API Function .....	92



The xSemaphoreGiveFromISR() API Function .....	93
Example 12. Using a binary semaphore to synchronize a task with an interrupt .....	95
3.3 Counting Semaphores .....	100
The xSemaphoreCreateCounting() API Function .....	103
Example 13. Using a counting semaphore to synchronize a task with an interrupt.....	105
3.4 Using Queues within an Interrupt Service Routine .....	107
The xQueueSendToFrontFromISR() and xQueueSendToBackFromISR() API Functions .....	107
Efficient Queue Usage .....	109
Example 14. Sending and receiving on a queue from within an interrupt .....	109
3.5 Interrupt Nesting .....	114
Chapter 4 Resource Management .....	117
4.1 Chapter Introduction and Scope.....	118
Mutual Exclusion.....	120
Scope .....	121
4.2 Critical Sections and Suspending the Scheduler .....	122
Basic Critical Sections .....	122
Suspending (or Locking) the Scheduler .....	123
The vTaskSuspendAll() API Function.....	124
The xTaskResumeAll() API Function .....	124
4.3 Mutexes (and Binary Semaphores).....	126
The xSemaphoreCreateMutex() API Function.....	128
Example 15. Rewriting vPrintString() to use a semaphore .....	128
Priority Inversion .....	131
Priority Inheritance .....	132
Deadlock (or Deadly Embrace) .....	133
4.4 Gatekeeper Tasks.....	135
Example 16. Re-writing vPrintString() to use a gatekeeper task.....	135
Chapter 5 Memory Management.....	141
5.1 Chapter Introduction and Scope.....	142
Scope .....	143
5.2 Example Memory Allocation Schemes .....	144
Heap_1.c .....	144
Heap_2.c .....	145
Heap_3.c .....	147
The xPortGetFreeHeapSize() API Function .....	147
Chapter 6 Trouble Shooting .....	151
6.1 Chapter Introduction and Scope.....	152
printf-stdarg.c.....	152
6.2 Stack Overflow.....	153
The uxTaskGetStackHighWaterMark() API Function .....	153
Run Time Stack Checking—Overview .....	154

Run Time Stack Checking—Method 1 .....	154
Run Time Stack Checking—Method 2 .....	155
6.3 Other Common Sources of Error .....	156
Symptom: Adding a simple task to a demo causes the demo to crash .....	156
Symptom: Using an API function within an interrupt causes the application to crash....	156
Symptom: Sometimes the application crashes within an interrupt service routine .....	157
Symptom: Critical sections do not nest correctly .....	157
Symptom: The application crashes even before the scheduler is started.....	157
Symptom: Calling API functions while the scheduler is suspended causes the application to crash .....	157
Symptom: The prototype for pxPortInitialiseStack() causes compilation to fail.....	158
6.4 Execution Visualization—Trace Hook Macros .....	159
Available Trace Hook Macros.....	159
Defining Trace Hook Macros .....	168
Example 17. Using Trace Hook Macros .....	169
6.5 Execution Visualization—Run Time Statistics.....	174
Run Time Statistics Time Base .....	174
Configuring an Application to Collect Run Time Statistics .....	175
The vTaskGetRunTimeStats() API Function.....	176
Example 18. Generating Run Time Statistics .....	177
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. Block diagram showing the features, peripherals and interfaces provided by the RX62N microcontroller .....	3
Figure 2. Changing the active project in the High-performance Embedded Workshop .....	11
Figure 3. Top level task states and transitions .....	16
Figure 4. The output produced when Example 1 is executed .....	21
Figure 5. The execution pattern of the two Example 1 tasks .....	22
Figure 6. The execution sequence expanded to show the tick interrupt executing .....	27
Figure 7. Running both test tasks at different priorities .....	28
Figure 8. The execution pattern when one task has a higher priority than the other .....	29
Figure 9. Full task state machine .....	32
Figure 10. The output produced when Example 4 is executed .....	34
Figure 11. The execution sequence when the tasks use vTaskDelay() in place of the NULL loop .....	34
Figure 12. Bold lines indicate the state transitions performed by the tasks in Example 4 .....	35
Figure 13. The output produced when Example 6 is executed .....	40
Figure 14. The execution pattern of Example 6 .....	40
Figure 15. The output produced when Example 7 is executed .....	43
Figure 16. The sequence of task execution when running Example 8 .....	48
Figure 17. The output produced when Example 8 is executed .....	49
Figure 18. The output produced when Example 9 is executed .....	52
Figure 19. The execution sequence for Example 9 .....	53
Figure 20. Execution pattern with pre-emption points highlighted .....	55
Figure 21. An example sequence of writes and reads to and from a queue .....	63
Figure 22. The output produced when Example 10 is executed .....	75
Figure 23. The sequence of execution produced by Example 10 .....	75
Figure 24. An example scenario where structures are sent on a queue .....	76
Figure 25. The output produced by Example 11 .....	80
Figure 26. The sequence of execution produced by Example 11 .....	81
Figure 27. The interrupt interrupts one task but returns to another .....	88
Figure 28. Using a binary semaphore to synchronize a task with an interrupt .....	91
Figure 29. The output produced when Example 12 is executed .....	98
Figure 30. The sequence of execution when Example 12 is executed .....	99
Figure 31. A binary semaphore can latch at most one event .....	101
Figure 32. Using a counting semaphore to 'count' events .....	102
Figure 33. The output produced when Example 13 is executed .....	106
Figure 34. The output produced when Example 14 is executed .....	113
Figure 35. The sequence of execution produced by Example 14 .....	113
Figure 36. Constants affecting interrupt nesting behavior .....	115
Figure 37. Mutual exclusion implemented using a mutex .....	127
Figure 38. The output produced when Example 15 is executed .....	131

Figure 39. A possible sequence of execution for Example 15 .....	131
Figure 40. A worst case priority inversion scenario .....	132
Figure 41. Priority inheritance minimizing the effect of priority inversion.....	133
Figure 42. The output produced when Example 16 is executed .....	139
Figure 43. RAM being allocated within the array each time a task is created .....	144
Figure 44. RAM being allocated from the array as tasks are created and deleted .....	146
Figure 45. The output produced when Example 17 is executed .....	173
Figure 46. The output produced when Example 18 is executed .....	182
Figure 47. The top-level directories—Source and Demo .....	187
Figure 48. The three core files that implement the FreeRTOS kernel.....	188
Figure 49. The source directories required to build an RX600 demo application .....	188
Figure 50. The demo directories required to build an RX600 demo application.....	190

## List of Code Listings

---

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

Listing 39. 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 40. The implementation of the sending task for Example 11. ....	78
Listing 41. The definition of the receiving task for Example 11 .....	79
Listing 42. The implementation of main() for Example 11 .....	80
Listing 43. The vSemaphoreCreateBinary() API function prototype .....	90
Listing 44. The xSemaphoreTake() API function prototype .....	92
Listing 45. The xSemaphoreGiveFromISR() API function prototype .....	93
Listing 46. Implementation of the task that periodically generates an interrupt in Example 12 .....	95
Listing 47. The implementation of the handler task (the task that synchronizes with the interrupt) in Example 12 .....	96
Listing 48. The interrupt handler used in Example 12 .....	97
Listing 49. The implementation of main() for Example 12 .....	98
Listing 50. The xSemaphoreCreateCounting() API function prototype .....	103
Listing 51. Using xSemaphoreCreateCounting() to create a counting semaphore .....	105
Listing 52. The implementation of the interrupt service routine used by Example 13 .....	105
Listing 53. The xQueueSendToFrontFromISR() API function prototype .....	107
Listing 54. The xQueueSendToBackFromISR() API function prototype .....	107
Listing 55. The implementation of the task that writes to the queue in Example 14 .....	110
Listing 56. The implementation of the interrupt service routine used by Example 14 .....	111
Listing 57. The task that prints out the strings received from the interrupt service routine in Example 14 .....	112
Listing 58. The main() function for Example 14 .....	112
Listing 59. An example read, modify, write sequence .....	118
Listing 60. An example of a reentrant function .....	120
Listing 61. An example of a function that is not reentrant .....	120
Listing 62. Using a critical section to guard access to a variable .....	122
Listing 63. A possible implementation of vPrintString() .....	122
Listing 64. The vTaskSuspendAll() API function prototype .....	124
Listing 65. The xTaskResumeAll() API function prototype .....	124
Listing 66. The implementation of vPrintString() .....	125
Listing 67. The xSemaphoreCreateMutex() API function prototype .....	128
Listing 68. The implementation of prvNewPrintString() .....	129
Listing 69. The implementation of prvPrintTask() for Example 15 .....	129
Listing 70. The implementation of main() for Example 15 .....	130
Listing 71. The name and prototype for a tick hook function .....	136
Listing 72. The gatekeeper task .....	136
Listing 73. The print task implementation for Example 16 .....	137
Listing 74. The tick hook implementation .....	137
Listing 75. The implementation of main() for Example 16 .....	138
Listing 76. The heap_3.c implementation .....	147
Listing 77. The xPortGetFreeHeapSize() API function prototype .....	147
Listing 78. The uxTaskGetStackHighWaterMark() API function prototype .....	153

Listing 79. The stack overflow hook function prototype .....	154
Listing 80. The malloc() failed hook function name and prototype.....	156
Listing 81. The trace macros demonstrated by Example 17.....	171
Listing 82. The definition of the task created by Example 17.....	172
Listing 83. The vTaskGetRunTimeStats() API function prototype.....	176
Listing 84. Configuration of the TMR0 and TMR1 peripherals used in Example 18.....	177
Listing 85. TMR0/TMR1 overflow interrupt handler used in Example 18 .....	178
Listing 86. Macros added to FreeRTOSConfig.h to enable the collection of run time statistics in Example 18 .....	179
Listing 87. The definition of two tasks created in Example 18 to use up some processing time.....	180
Listing 88. The task that prints out the collected run time statistics in Example 18.....	181
Listing 89. The template for a new main() function.....	192

## List of Tables

---

Table 1. Comparing the FreeRTOS license with the OpenRTOS license .....	8
Table 2. xTaskCreate() parameters and return value .....	17
Table 3. vTaskDelay() parameters .....	33
Table 4. vTaskDelayUntil() parameters .....	37
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 26 .....	81
Table 13. vSemaphoreCreateBinary() parameters .....	90
Table 14. xSemaphoreTake() parameters and return value .....	92
Table 15. xSemaphoreGiveFromISR() parameters and return value .....	94
Table 16. xSemaphoreCreateCounting() parameters and return value .....	104
Table 17. xQueueSendToFrontFromISR() and xQueueSendToBackFromISR() parameters and return values .....	107
Table 18. Constants that affect interrupt nesting .....	114
Table 19. xTaskResumeAll() return value .....	124
Table 20. xSemaphoreCreateMutex() return value .....	128
Table 21. xPortGetFreeHeapSize() return value .....	148
Table 22. uxTaskGetStackHighWaterMark() parameters and return value .....	153
Table 23. Trace hook macros .....	159
Table 24. Macros used in the collection of run time statistics .....	175
Table 25. vTaskGetRunTimeStats() parameters .....	176
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



## List of Notation

---

API	Application Programming Interface
CMT	RX600 Compare Match Timer Peripheral
FAQ	Frequently Asked Question
FIFO	First In First Out
HEW	High-performance Embedded Workshop
HMI	Human Machine Interface
IDE	Integrated Development Environment
ISR	Interrupt Service Routine
LCD	Liquid Crystal Display
RMS	Rate Monotonic Scheduling
RSK	Renesas Starter Kit
RTOS	Real-time Operating System
SIL	Safety Integrity Level
TCB	Task Control Block
TMR0/1	RX600 8-bit Timer Peripherals Zero and One
UART	Universal Asynchronous Receiver/Transmitter

