OPEN**RTOS** APPLICATION NOTE: #34-185-AN-001

INSTALLING AND USING THE WHIS STATE**VIEWER** KERNEL AWARE PLUG-IN FOR ECLIPSE

Issue 3.0 – 7th November 2017

CONTENTS

FIGURES

CHAPTER 1

INTRODUCTION

1.1 DESCRIPTION

The STATE**VIEWER** Kernel Aware Plug-in for Eclipse provides a snapshot view of the FreeRTOS/OPEN**RTOS**/SAFE**RTOS** kernel data structures presented in a meaningful tabular format. This allows the current state of tasks, queues and the various forms of semaphore to be inspected directly from within the Eclipse Workbench IDE.

WITTENSTEIN high integrity systems have made the STATE**VIEWER** plug-in available for use by the FreeRTOS.org™ community as well.

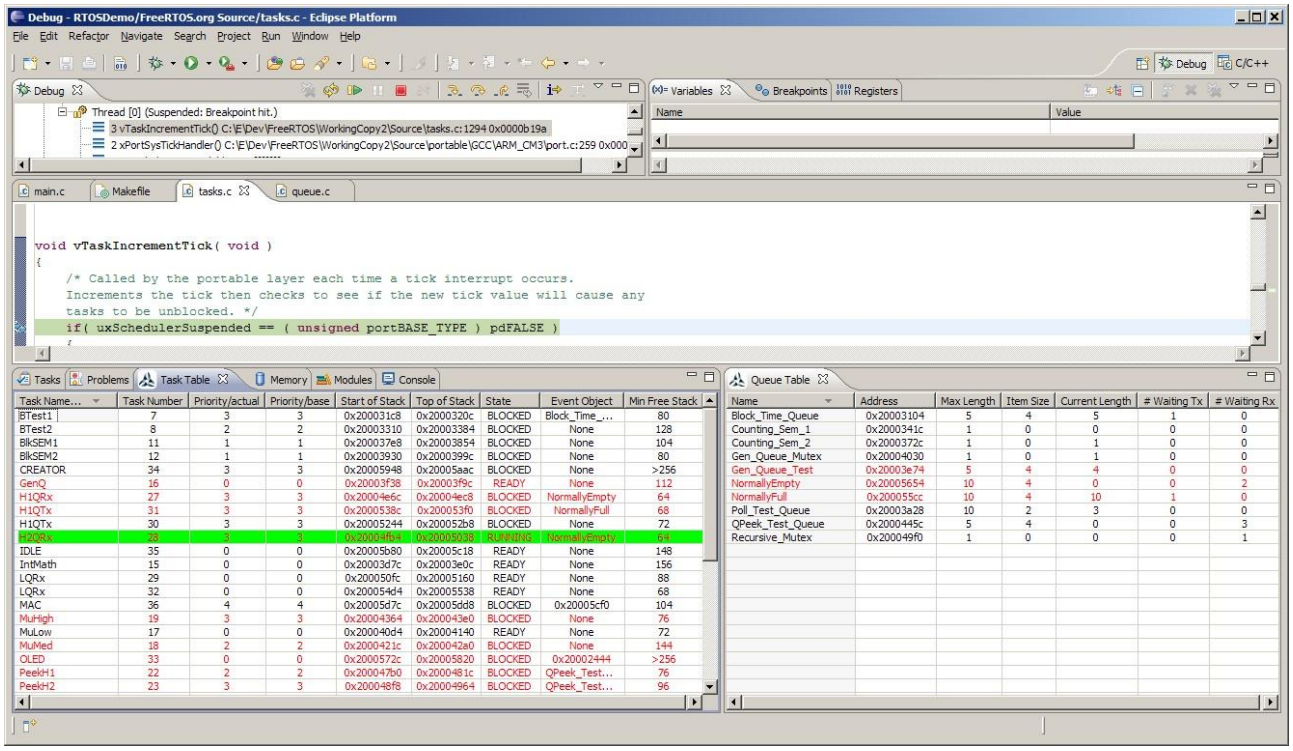


Figure 1-1 The Eclipse Workbench IDE with the Task and Queue viewers visible

CHAPTER 2

INSTALLATION & USAGE

2.1 INSTALLING THE PLUG-IN

Step 1. Within Eclipse, from the 'Help' menu select 'Install New Software…' item.

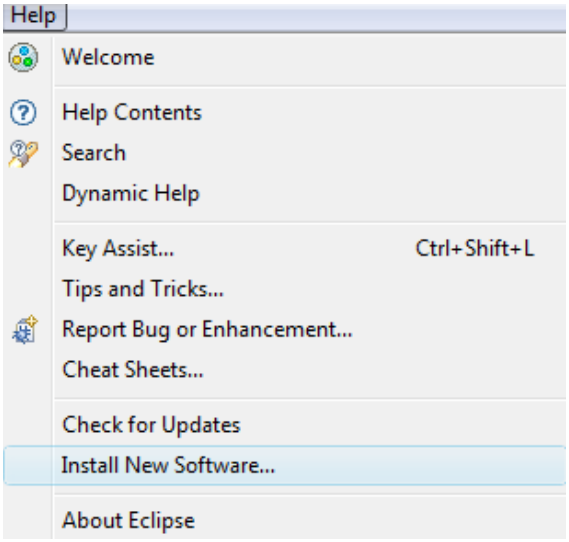


Figure 2-1: 'Install New Software...' menu item.

Step 2. In the 'Work with:' category, click the 'Add…' button. See Figure 2-3.

Step 3. In the 'Add Repository' dialog window, specify plug-in name as 'Stateviewer' and location path as http://stateviewer.highintegritysystems.com and click 'OK'

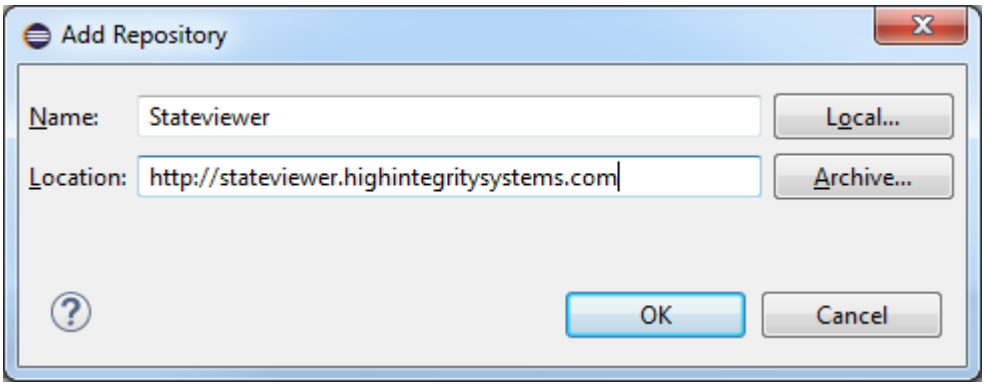


Figure 2-2: 'Add Repository' Dialog.

Step 1. Tick the ‘RTOS Tools’ category and click the ‘Next >’ button.

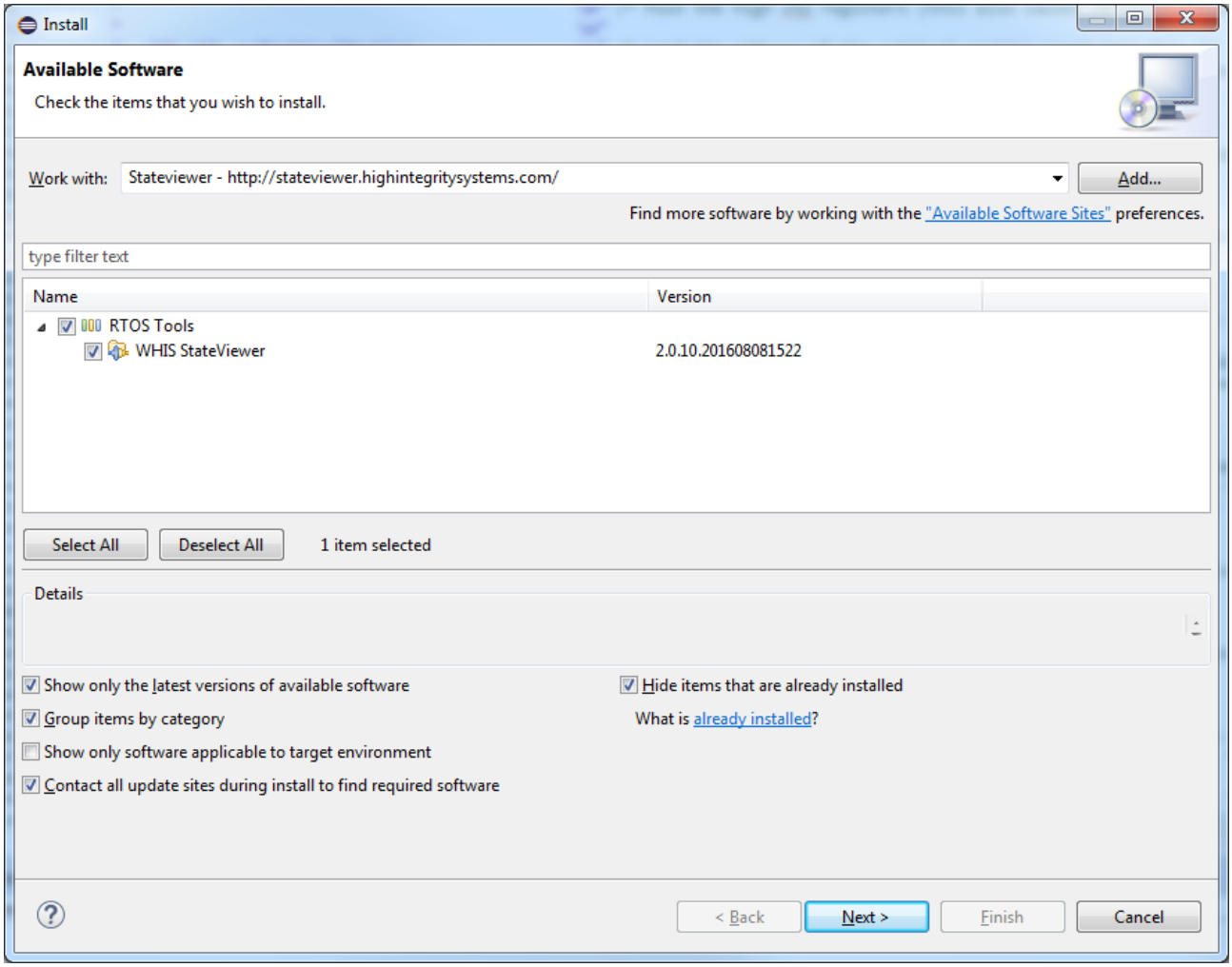


Figure 2-3: ‘Available Software’ window.

Step 2. The ‘Install Details’ window displays the items to be installed. Click ‘Next >’.

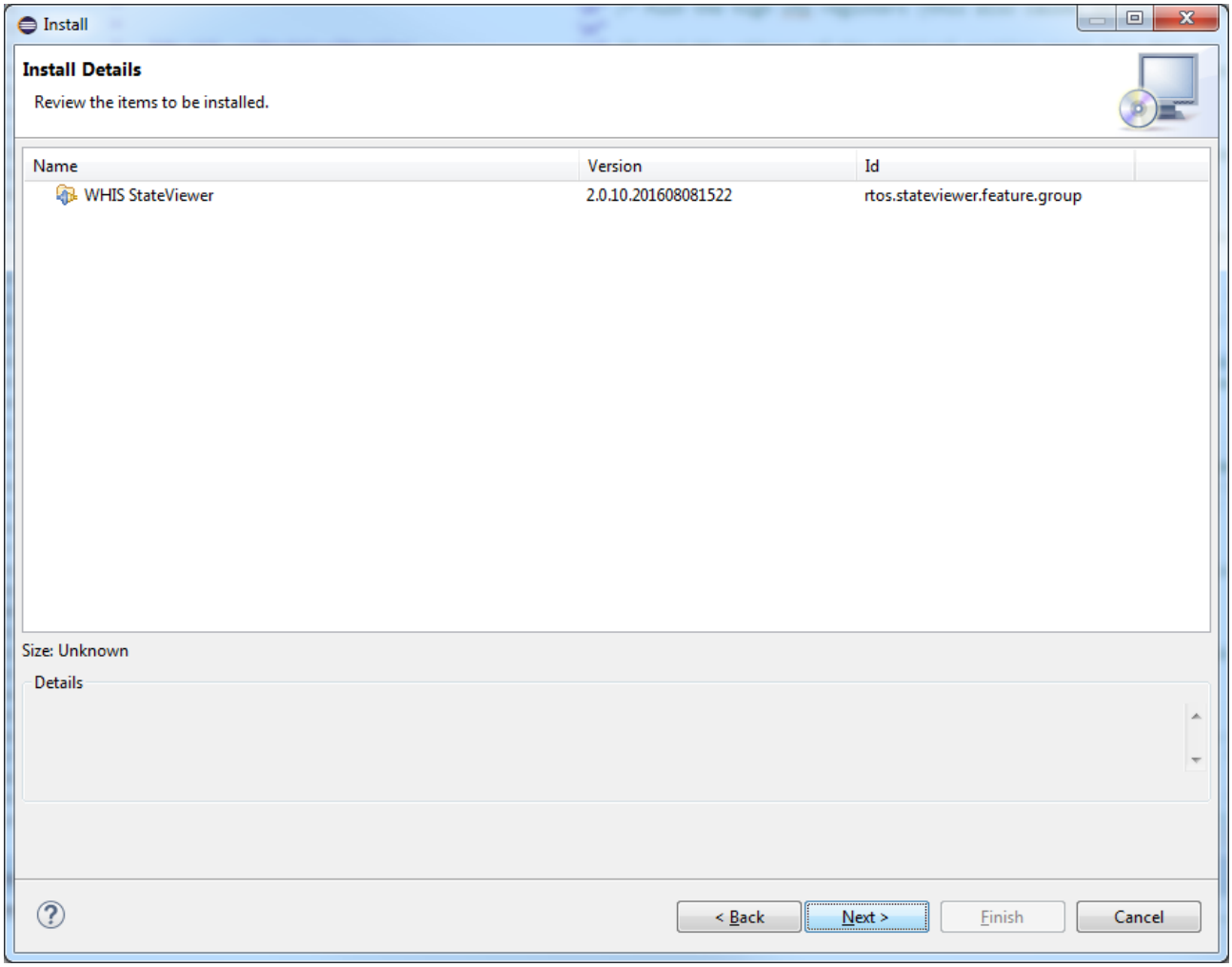


Figure 2-4: 'Install Details' window.

Step 3. The ‘Review Licenses’ window requires accepting the License to use the software to proceed. Read the ‘License Text’ for WHIS StateViewer. If you agree with the terms and conditions, select the ‘I accept the terms of the license agreements’ pushbutton and click ‘Finish’ to install the software.

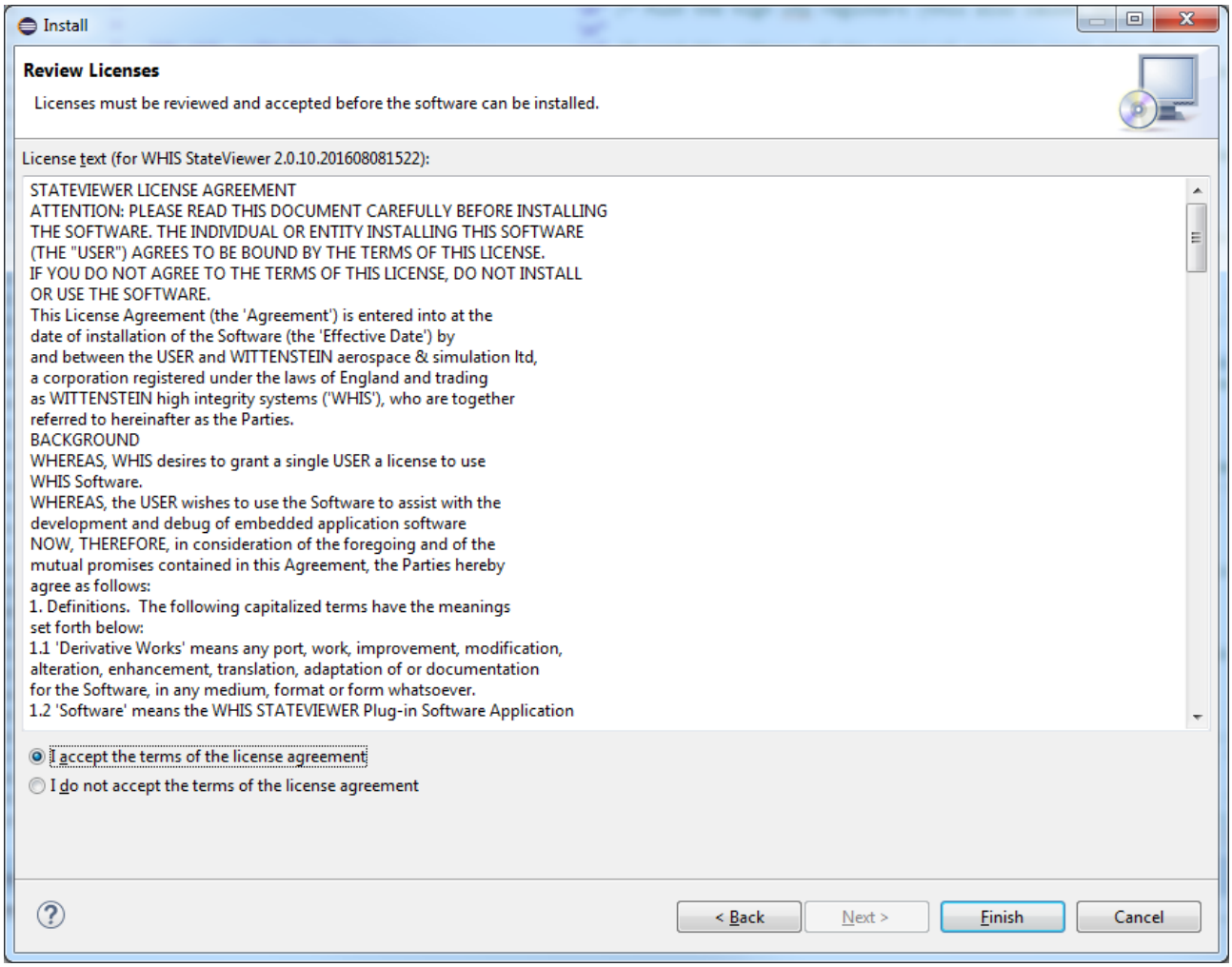


Figure 2-5: 'Review Licenses' window.

Step 4. Check for certificate details as in Figure 2-6 and tick the certificate “WITTENSTEIN High Integrity Systems; High Integrity Systems; WITTENSTEIN aerospace & simulation Ltd.” and then click ‘OK’.

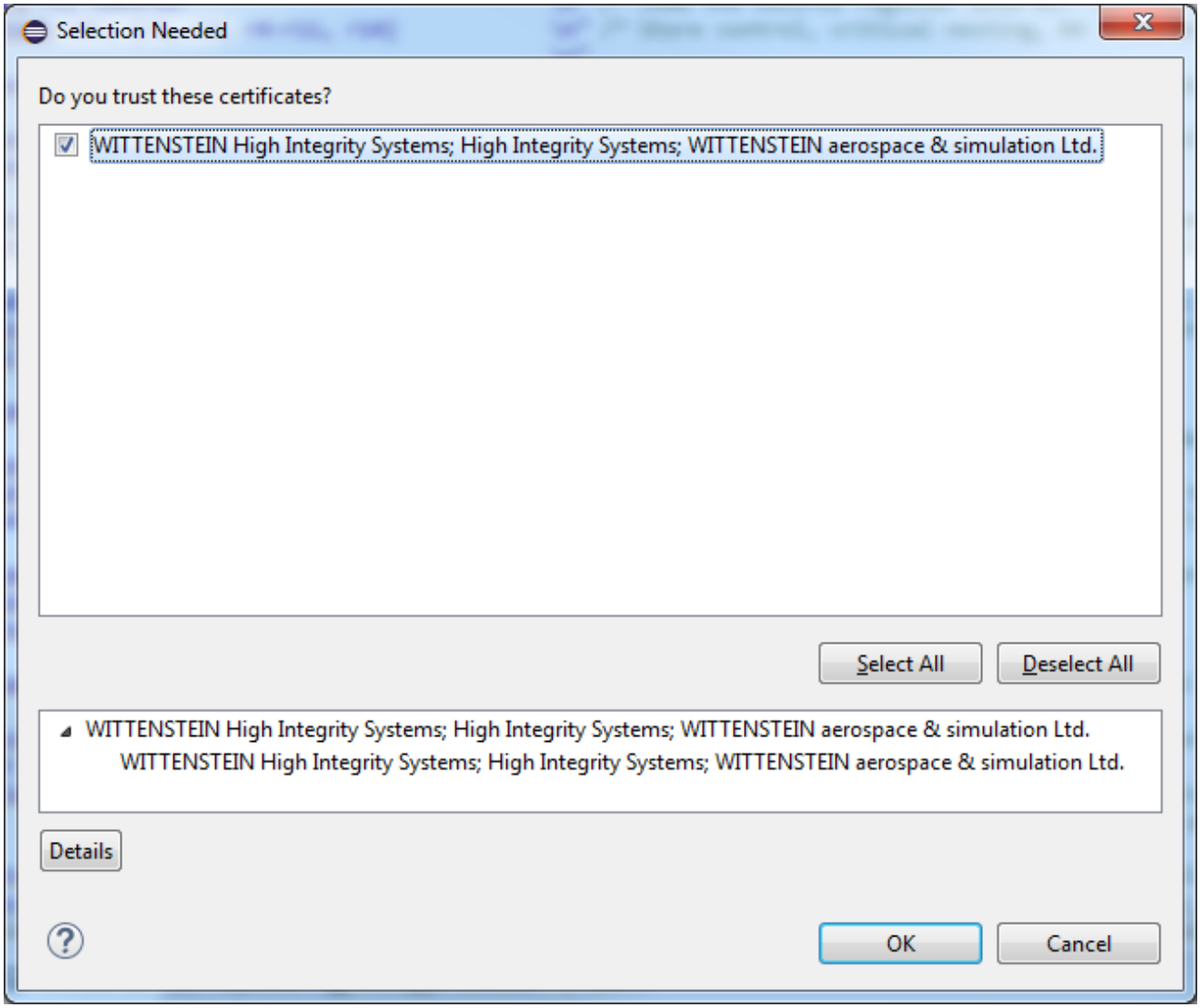


Figure 2-6: Certificate Selection Window

Step 5. When the Installation has finished, Eclipse prompts for restart. Now restart the Eclipse by clicking ‘Restart Now’.

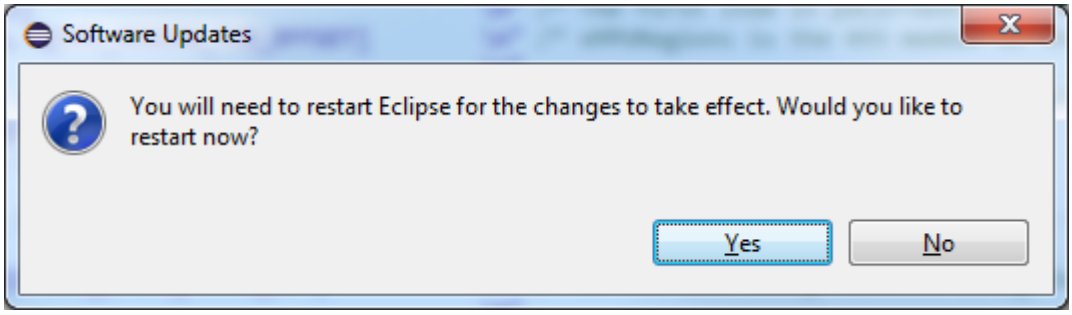


Figure 2-7: Restart Eclipse notice.

Step 6. When WHIS StateViewer plug-in is installed correctly you will see ‘OpenRTOS Viewer’ and ‘SafeRTOS Viewer’ in Eclipse ‘Show View’ window as in Figure 2-8.

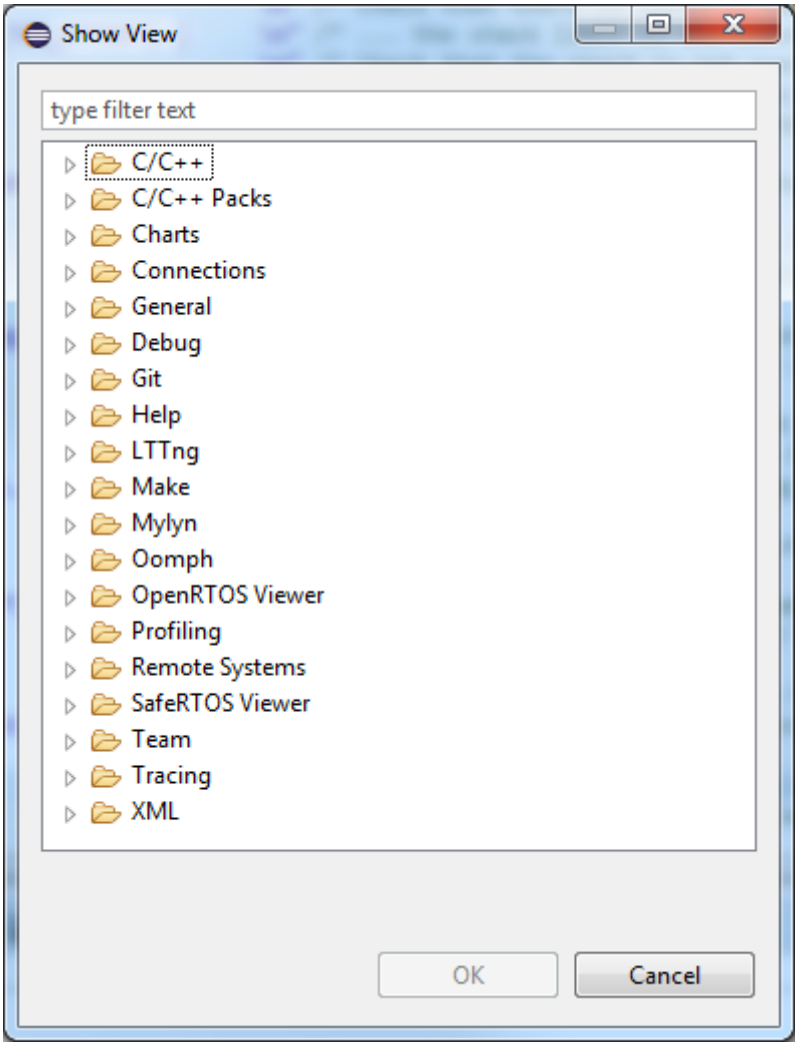


Figure 2-8 Show View window

2.2 OPENING THE RTOS VIEWER TABLES

When you are ready to start a debug session:

Step 1. Start a debug session as normal.

Step 2. Ensure ‘Debug’ is the current Eclipse perspective.

Step 3. From the Eclipse Workbench “Window” menu, select “Show View” → “Other” (Figure 2-9).

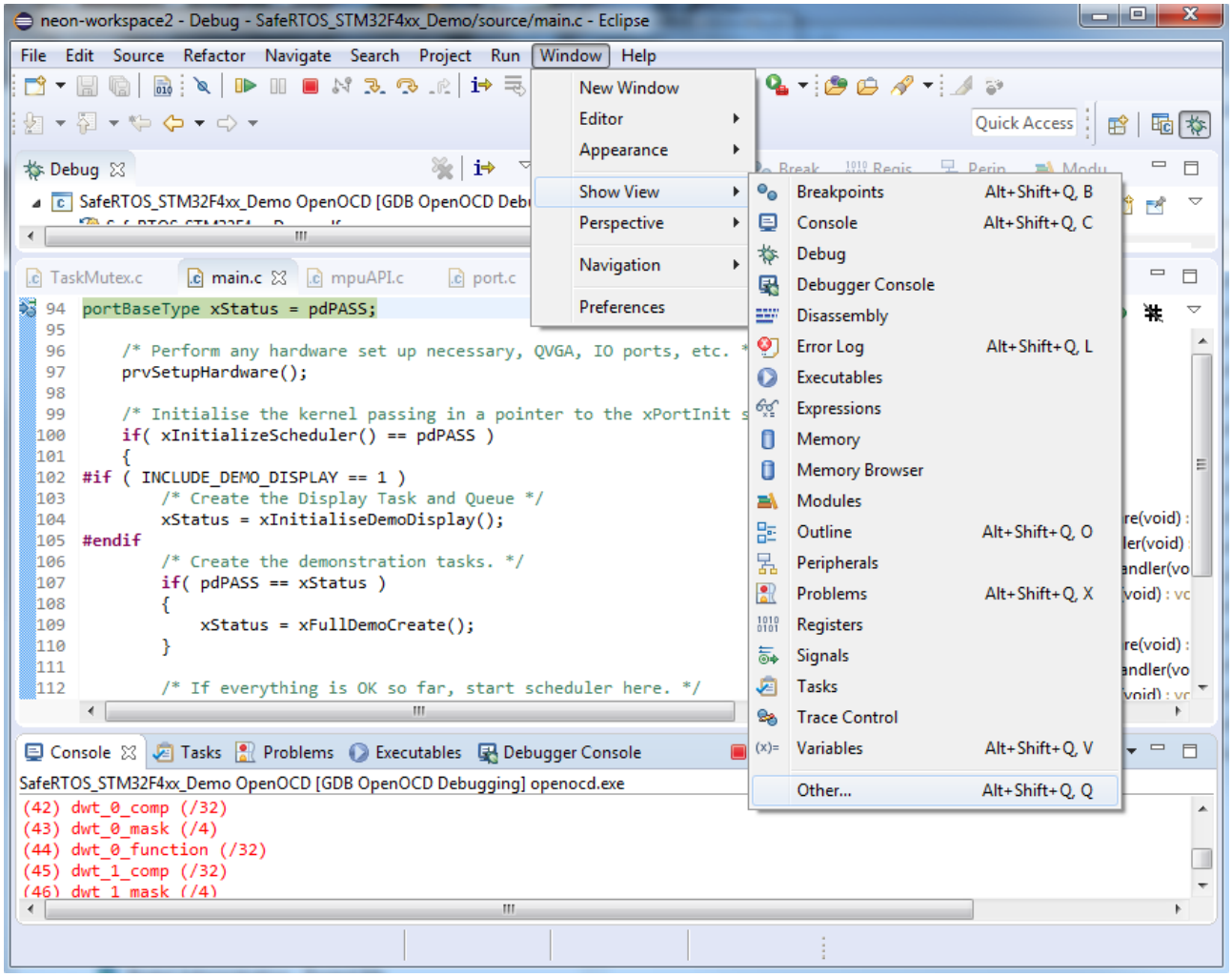


Figure 2-9: Opening up the "Other" view dialogue

Step 4. From the resultant dialogue box, expand the required either OpenRTOS or SafeRTOS Viewer tree (Figure 2-10) and select whichever table is required and click OK to add to the Eclipse perspective.

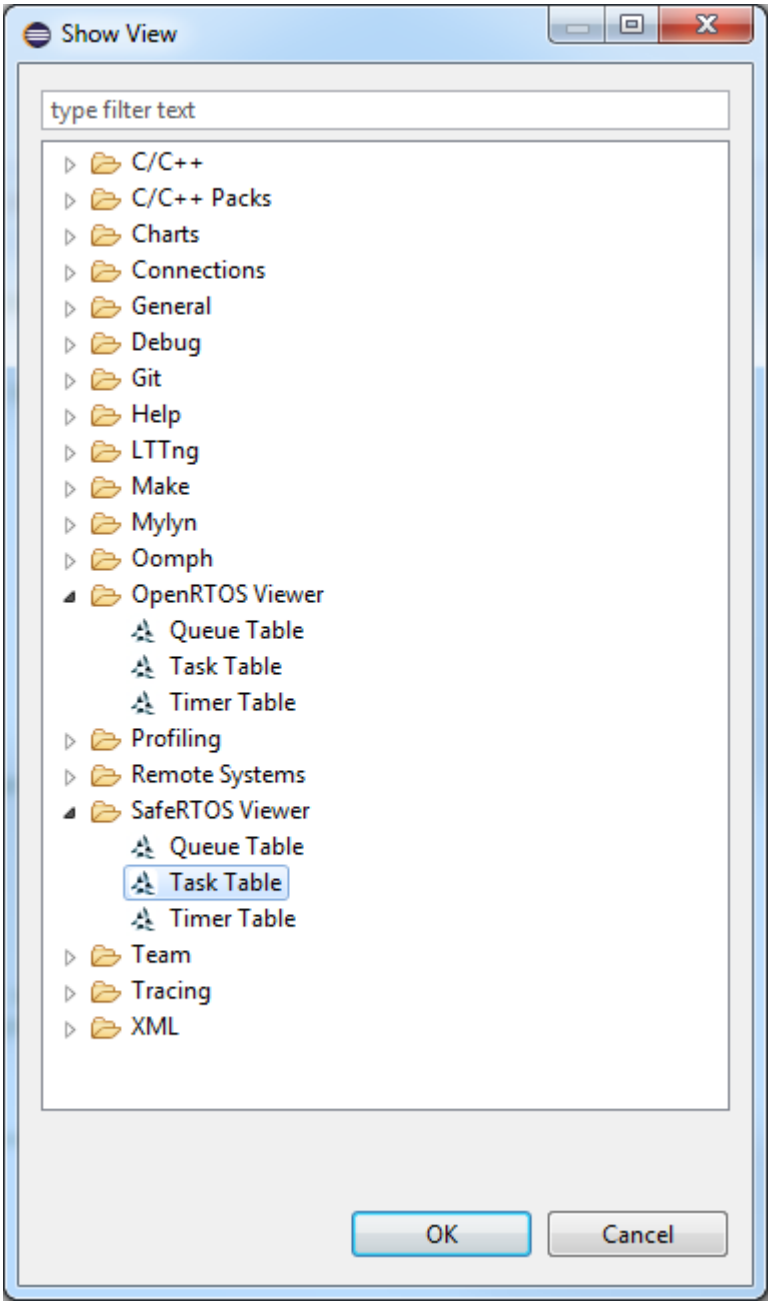
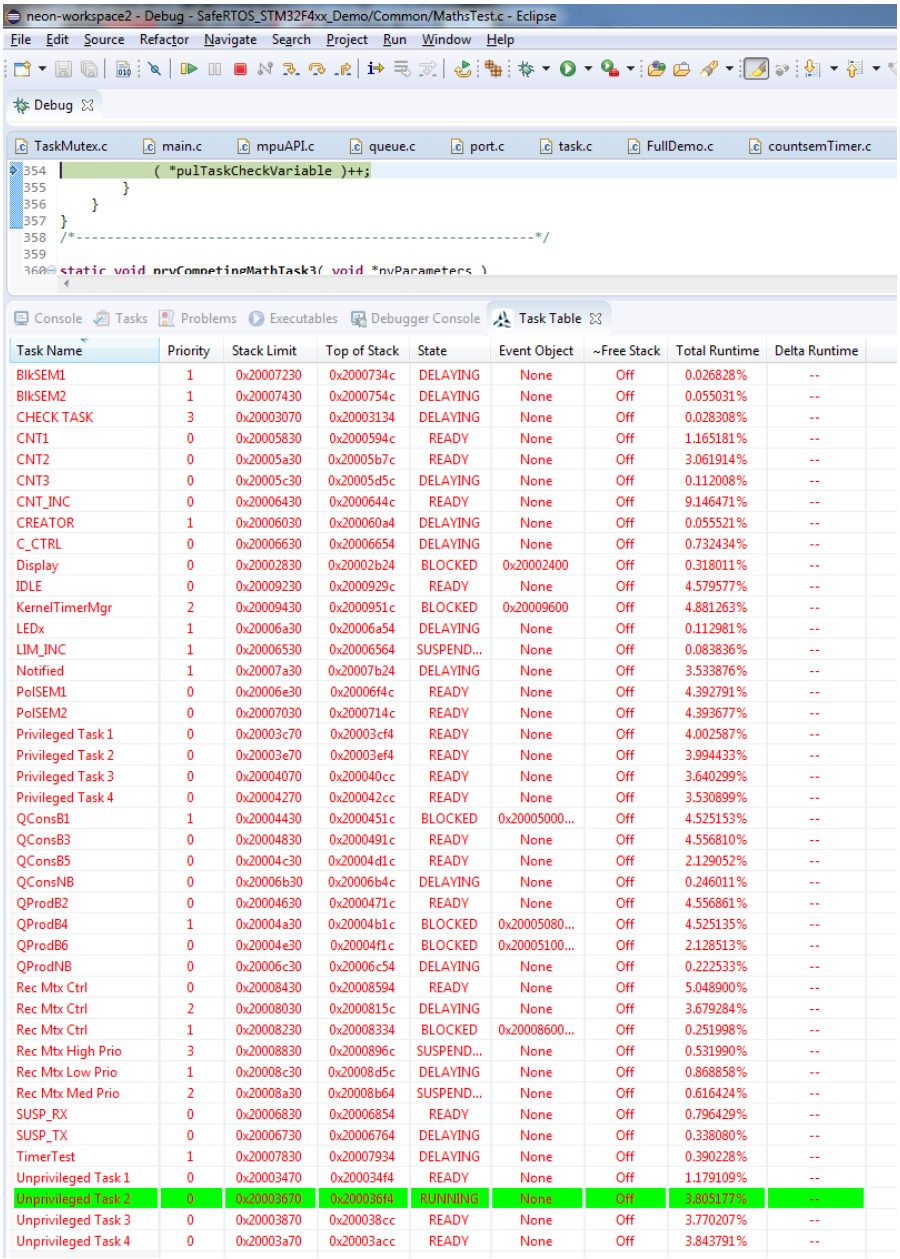


Figure 2-10: StateViewer Tables in the Show View window.

The tables will refresh themselves each time the debugger is manually suspended, hits a break point, or is single stepped.

The tabs that contain the tables can be docked in any position within the Eclipse Workbench IDE simply by dragging and dropping the tabs to your desired location.

2.3 INTERPRETING TASK TABLE DATA



Task Name	Priority	Stack Limit	Top of Stack	State	Event Object	~Free Stack	Total Runtime	Delta Runtime
BlkSEM1	1	0x20007230	0x2000734c	DELAYING	None	Off	0.026828%	--
BlkSEM2	1	0x20007430	0x2000754c	DELAYING	None	Off	0.055031%	--
CHECK TASK	3	0x20003070	0x20003134	DELAYING	None	Off	0.028308%	--
CNT1	0	0x20005830	0x2000594c	READY	None	Off	1.165181%	--
CNT2	0	0x20005a30	0x20005b7c	READY	None	Off	3.061914%	--
CNT3	0	0x20005c30	0x20005d5c	DELAYING	None	Off	0.112008%	--
CNT_INC	0	0x20006430	0x2000644c	READY	None	Off	9.146471%	--
CREATOR	1	0x20006030	0x200060a4	DELAYING	None	Off	0.055521%	--
C_CTRL	0	0x20006630	0x20006654	DELAYING	None	Off	0.732434%	--
Display	0	0x20002830	0x20002b24	BLOCKED	0x20002400	Off	0.318011%	--
IDLE	0	0x20009230	0x2000929c	READY	None	Off	4.579577%	--
KernelTimerMgr	2	0x20009430	0x2000951c	BLOCKED	0x20009600	Off	4.881263%	--
LEDx	1	0x20006a30	0x20006a54	DELAYING	None	Off	0.112981%	--
LIM_INC	1	0x20006530	0x20006564	SUSPEND...	None	Off	0.083836%	--
Notified	1	0x20007a30	0x20007b24	DELAYING	None	Off	3.533876%	--
PolSEM1	0	0x20006e30	0x20006f4c	READY	None	Off	4.392791%	--
PolSEM2	0	0x20007030	0x2000714c	READY	None	Off	4.393677%	--
Privileged Task 1	0	0x20003c70	0x20003cf4	READY	None	Off	4.002587%	--
Privileged Task 2	0	0x20003e70	0x20003ef4	READY	None	Off	3.994433%	--
Privileged Task 3	0	0x20004070	0x200040cc	READY	None	Off	3.640299%	--
Privileged Task 4	0	0x20004270	0x200042cc	READY	None	Off	3.530899%	--
QConsB1	1	0x20004430	0x2000451c	BLOCKED	0x20005000...	Off	4.525153%	--
QConsB3	0	0x20004830	0x2000491c	READY	None	Off	4.556810%	--
QConsB5	0	0x20004c30	0x20004d1c	READY	None	Off	2.129052%	--
QConsNB	0	0x20006b30	0x20006b4c	DELAYING	None	Off	0.246011%	--
QProdB2	0	0x20004630	0x2000471c	READY	None	Off	4.556861%	--
QProdB4	1	0x20004a30	0x20004b1c	BLOCKED	0x20005080...	Off	4.525135%	--
QProdB6	0	0x20004e30	0x20004f1c	BLOCKED	0x20005100...	Off	2.128513%	--
QProdNB	0	0x20006c30	0x20006c54	DELAYING	None	Off	0.222533%	--
Rec Mtx Ctrl	0	0x20008430	0x20008594	READY	None	Off	5.048900%	--
Rec Mtx Ctrl	2	0x20008030	0x2000815c	DELAYING	None	Off	3.679284%	--
Rec Mtx Ctrl	1	0x20008230	0x20008334	BLOCKED	0x20008600...	Off	0.251998%	--
Rec Mtx High Prio	3	0x20008830	0x2000896c	SUSPEND...	None	Off	0.531990%	--
Rec Mtx Low Prio	1	0x20008c30	0x20008d5c	DELAYING	None	Off	0.868858%	--
Rec Mtx Med Prio	2	0x20008a30	0x20008b64	SUSPEND...	None	Off	0.616424%	--
SUSP_RX	0	0x20006830	0x20006854	READY	None	Off	0.796429%	--
SUSP_TX	0	0x20006730	0x20006764	DELAYING	None	Off	0.338080%	--
TimerTest	1	0x20007830	0x20007934	DELAYING	None	Off	0.390228%	--
Unprivileged Task 1	0	0x20003470	0x200034f4	READY	None	Off	1.179109%	--
Unprivileged Task 2	0	0x20003670	0x200036f4	RUNNING	None	Off	3.805177%	--
Unprivileged Task 3	0	0x20003870	0x200038cc	READY	None	Off	3.770207%	--
Unprivileged Task 4	0	0x20003a70	0x20003acc	READY	None	Off	3.843791%	--

Figure 2-11: Example task table screen shot

Figure 2-11 provides an example task table screen shot where each table row provides information on a single task. If the row is black then the information it contains has not changed since the debugger was last suspended. Conversely, if the row is red then at least one of the columns in that row has changed since the debugger was last suspended. The green highlight indicates the currently executing task – the task that is in the Running state.

Table 2-1 OpenRTOS Task Table Columns

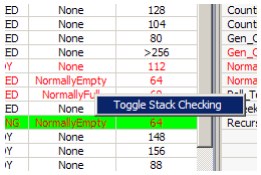
Column Heading	Description
Task Name	The plain text name assigned to the task when the task was created.
Base/Actual Priority	Base Priority: The priority assigned to the task. A priority is assigned when the task is created and each time the task is the subject of a call to vTaskPrioritySet(). Actual Priority: The priority currently being used by the task. The OPEN**RTOS** priority inheritance mechanism means the actual priority can transiently be higher than the tasks base (assigned) priority.
Start of Stack	The address of the beginning of the stack assigned to the task. The beginning of the stack is the extreme of the stack region that gets written into first.
Top of Stack	The address of the current top of stack – that is the position last written to when the context of the stack was last saved. This value has little meaning for the current Running state task.
State	A text description of the task state, as specified in the OPEN**RTOS** user manual.
Event Object	The name or address of the queue or semaphore on which the task is blocked – if any. The name is used if the queue or semaphore has been registered within the queue registry.
Min Free Stack	The stack 'high water mark'. This is the minimum number of bytes there have been between the task stack pointer and the end of the stack assigned to the task. The lower this value the closer the task has come to overflowing its stack. A value of 0 indicates that the task has overflowed its stack at some time. Calculating the stack high water mark takes a noticeable amount of time so it is sometimes desirable to disable the calculation altogether. Right clicking on the task table displays a menu that allows the feature to be toggle on and off. 
Total Runtime	The amount of time that the task has been executing since reset (expressed as a percentage).
Delta Runtime	The amount of time that the task has been executing since the last time the debugger was paused (expressed as a percentage).

Figure 2-12: Right clicking on the Task Table to reveal the "Toggle Stack Checking" pop up menu

Table 2-2 SAFE**RTOS** Task Table Columns

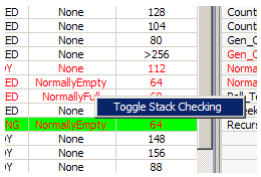
Column Heading	Description
Task Name	The plain text name assigned to the task when the task was created.
Priority	The priority assigned to the task. A priority is assigned when the task is created and each time the task is the subject of a call to xTaskPrioritySet().
Stack Limit	The address of the lowest allowable stack usage assigned to the task. This value is calculated to ensure that the stack contains enough free space to allow the context to be saved when the task is switched out.
Top of Stack	The address of the current top of stack – that is the position last written to when the context of the stack was last saved. This value has little meaning for the current Running state task.
State	A text description of the task state, as specified in the SAFE**RTOS** user manual.
Event Object	The name or address of the queue or semaphore on which the task is blocked – if any. The name is used if the queue or semaphore has been registered within the queue registry.
~Free Stack	The stack 'high water mark'. This is the minimum number of bytes there have been between the task stack pointer and the end of the stack assigned to the task. The lower this value the closer the task has come to overflowing its stack. A value of 0 indicates that the task has overflowed its stack at some time. Calculating the stack high water mark takes a noticeable amount of time so it is sometimes desirable to disable the calculation altogether. Right clicking on the task table displays a menu that allows the feature to be toggle on and off. 
Total Runtime	The amount of time that the task has been executing since reset (expressed as a percentage).
Delta Runtime	The amount of time that the task has been executing since the last time the debugger was paused (expressed as a percentage).

Figure 2-13: Right clicking on the Task Table to reveal the "Toggle Stack Checking" pop up menu

2.4 INTERPRETING QUEUE TABLE DATA

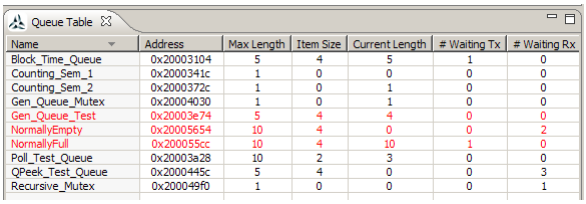


Figure 2-14: Example queue table screen shot

The queue table displays data on all queues and semaphores that have been added to the queue registry by a call to vQueueAddToRegistry(). The following description refers only to queues, but is equally applicable to binary semaphores, counting semaphores and mutexes. Please note that for SAFE**RTOS**, the queue registry functions are distributed separately to the RTOS kernel, whereas they are distributed as part of the OpenRTOS kernel.

Figure 2-14 provides an example queue table screen shot where each table row provides information on a single queue or semaphore. If the row is black then the information it contains has not changed since the debugger was last suspended. Conversely, if the row is red then at least one of the columns in that row has changed since the debugger was last suspended.

Table 2-3 Queue Table Columns

Column Heading	Description
Name	The name that was assigned to the queue when the queue was registered.
Address	The address of the queue structure. This is also the queue handle value.
Max Length	The maximum number of items the queue can hold at any one time.
Item Size	The size in bytes that each queued item occupies.
Current Length	The number of items currently in the queue. This can only be less than or equal to the Max Length.
# Waiting Tx	The number of tasks currently blocked waiting for space to become available on the queue. That is – tasks that are blocked waiting to send to the queue.
# Waiting Rx	The number of tasks currently blocked waiting for data to become available from the queue. That is – tasks that are blocked waiting to receive from the queue.

2.5 INTERPRETING TIMER TABLE DATA

The timer table displays data on all timers that are currently active.

Table 2-4 Timer Table Columns

Column Heading	Description
Name	The name that was assigned to the timer when the timer was created.
Period	The timeout period of the timer in ticks.
Reload	Indicates whether the timer will automatically reload (periodic or single shot).
Callback Fn	The address of the routine that will be executed when the timer expires.
ID	An ID that has been assigned to the timer instance (useful when multiple timers share the same callback function).

CONTACT INFORMATION

User feedback is essential to the continued maintenance and development of our products. Please provide all software and documentation comments and suggestions to the most convenient contact point listed below.

Address:	WITTENSTEIN high integrity systems Brown's Court, Long Ashton Business Park Yanley Lane, Long Ashton Bristol, BS41 9LB England
Phone:	+44 (0)1275 395 600
Fax:	+44 (0)1275 393 630
Email:	support@HighIntegritySystems.com
Website	www.HighIntegritySystems.com

All Trademarks acknowledged.