

e² studio

Partner RTOS Aware Debugging for RX

Introduction

Renesas e² studio is a development environment based on the popular Eclipse CDT (C/C++ Development Tooling). It covers build (editor, compiler and linker control) as well as debug interface. It also supports integrating the Renesas GitHub FreeRTOS (with IoT libraries) demo applications and running them on Renesas boards.

Partner OS Debugging Plug-in within e² studio can be used during debugging session by clicking Renesas Views > Partner OS > RTOS Resources. This view displays information on the usage of resources by the RTOS operation. Items that can be displayed vary according to the real-time OS.

Objectives

This document introduces the usage of RTOS Resource view in e² studio as follows:

- How to create an RTOS project
- Introduction of RTOS Resource view
- Using the RTOS Resource view with FreeRTOS (Task, Queue, Timer, Stack)

Operating Environment

The operation was confirmed in the following environments.

IDE	e ² studio 2020-07 e ² studio v7.8.0
Toolchains	CCRX Compiler v3.0.2
Target devices	Renesas RX Family (RX65N-2MB RSK)
Debuggers	E2 emulator , E2 emulator Lite(E2 Lite) , E1 emulator
Target OS	FreeRTOS

Contents

1. Create the FreeRTOS project.....	3
2. Introduction of RTOS Resources view	8
2.1 Opening the RTOS Resources view	8
2.2 Selecting the OS	8
2.3 Context menu	9
2.4 Stack setting.....	10
3. Using RTOS Resources view with FreeRTOS	12
3.1 Task tab.....	12
3.2 Queue tab.....	13
3.3 Timer tab	14
3.4 Stack tab.....	15
Revision History	17

1. Create the FreeRTOS project

To create a new FreeRTOS project, follow the steps below.

1. Launch e² studio.
2. Select [File] → [New] → [C/C++ Project]
3. Select [Renesas RX] → [Renesas CC-RX C/C++ Executable Project]

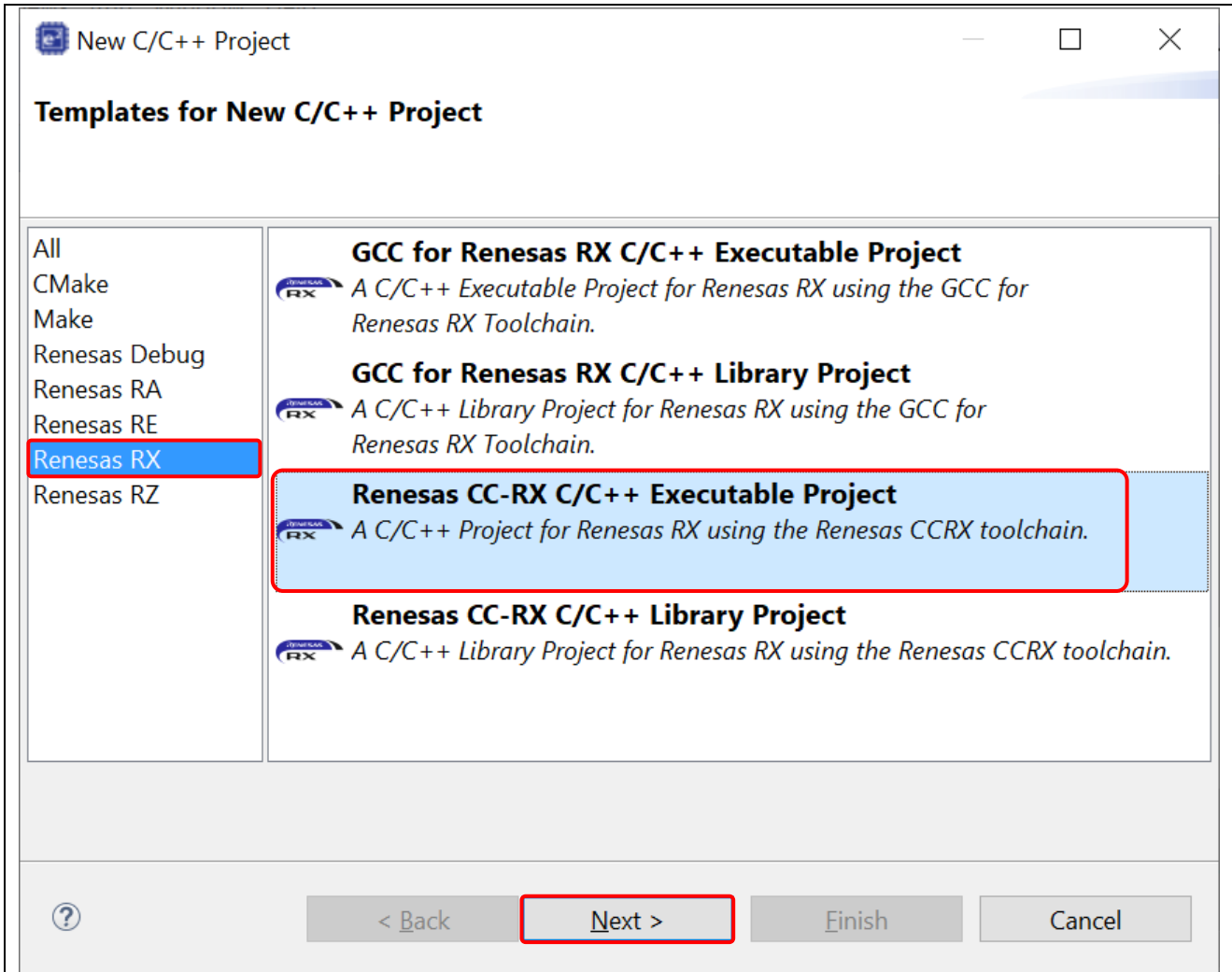


Figure 1-1 Select project template

4. Name the project and click [Next].

5. Specify the following information and click [Finish]:

- Language: C
- Toolchain: Renesas RX
- Toolchain Version: v3.02.00
- Target Board: RSKRX65N-2MB
- Configuration: Tick Create Hardware Debug Configuration and select the emulator (e.g. E2 Lite (RX))
- RTOS: FreeRTOS (with IoT libraries)
- RTOS Version: v202002.00-rx-1.0.1 (or latest) for e² studio 2020-07 (64-bit), and v201908.00-rx-0.1.17 for e² studio v7.8 (32-bit); v201908.00-rx-0.1.17 for e² studio 7.7.

If there is a warning message that the FreeRTOS package is not found or to check and download the latest FreeRTOS package, click [Manage Toolchains...] and follow steps 6 to 9 before clicking [Finish].

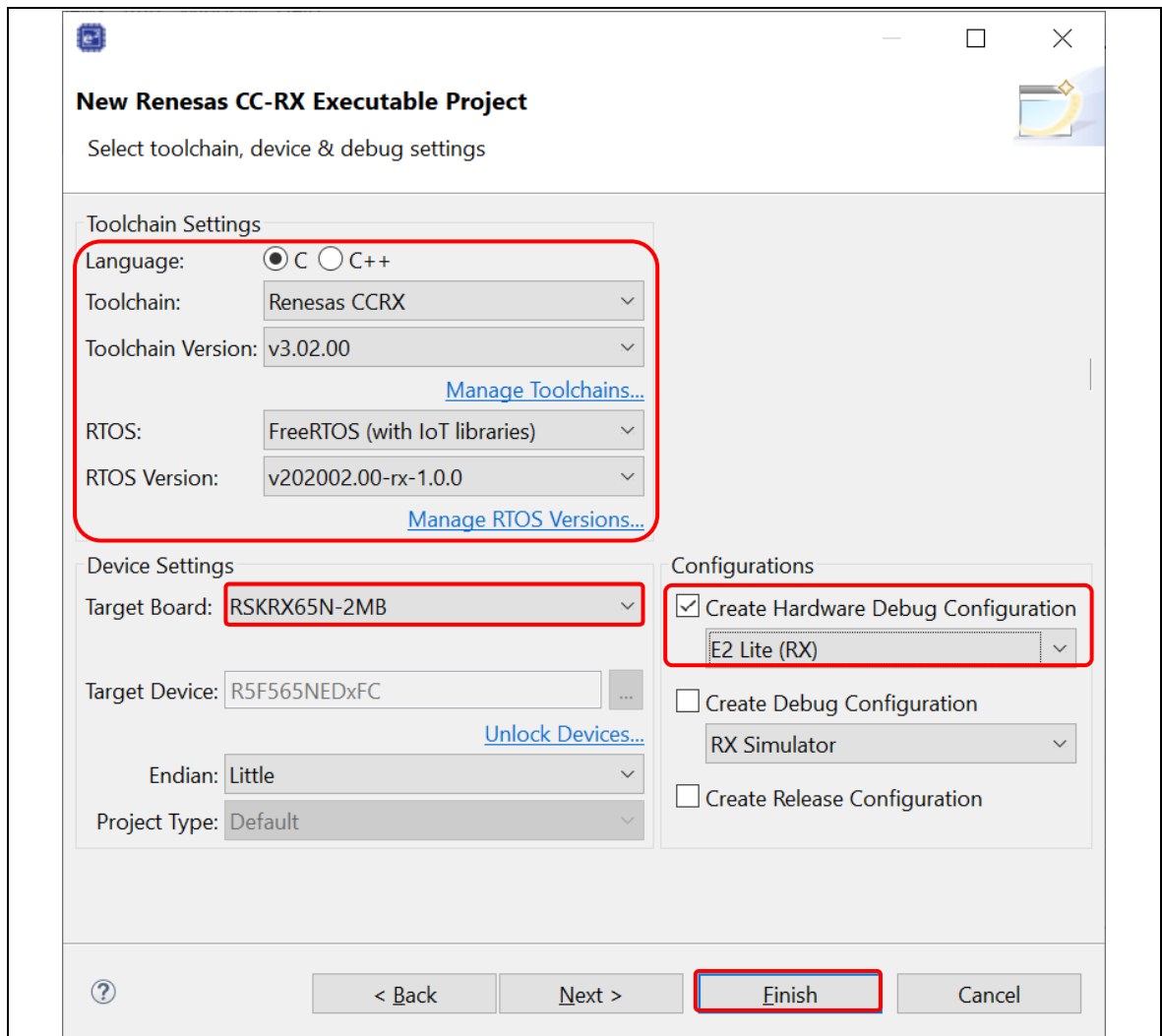


Figure 1-2 Select toolchain, device and RTOS

- 6. Select "Manage RTOS Versions..." to download Renesas FreeRTOS (with IoT libraries) package from the GitHub.

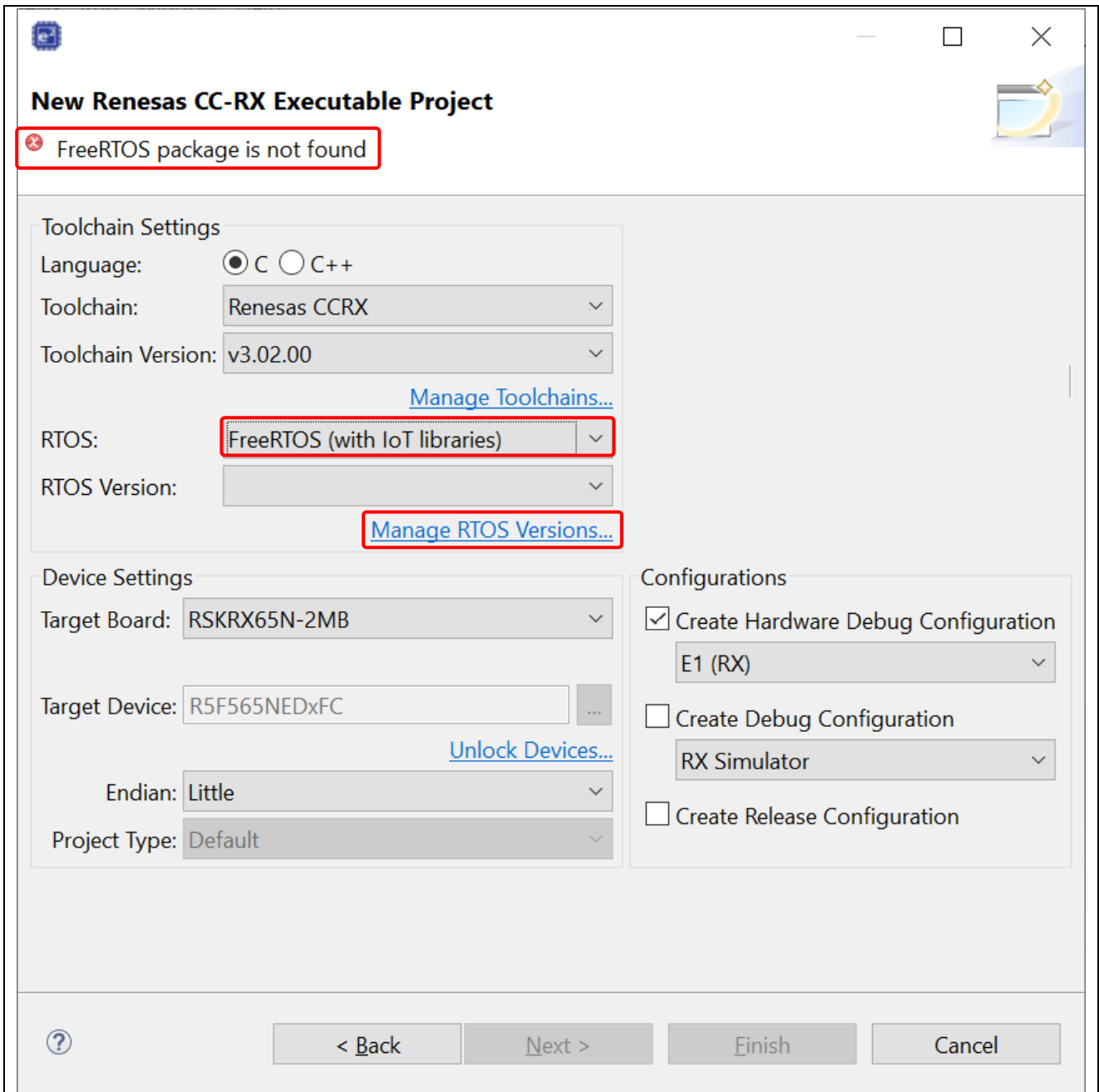


Figure 1-3 Renesas FreeRTOS (with IoT libraries) package needs to be downloaded

7. The download dialog will be shown. Select the latest FreeRTOS module and click [Download].

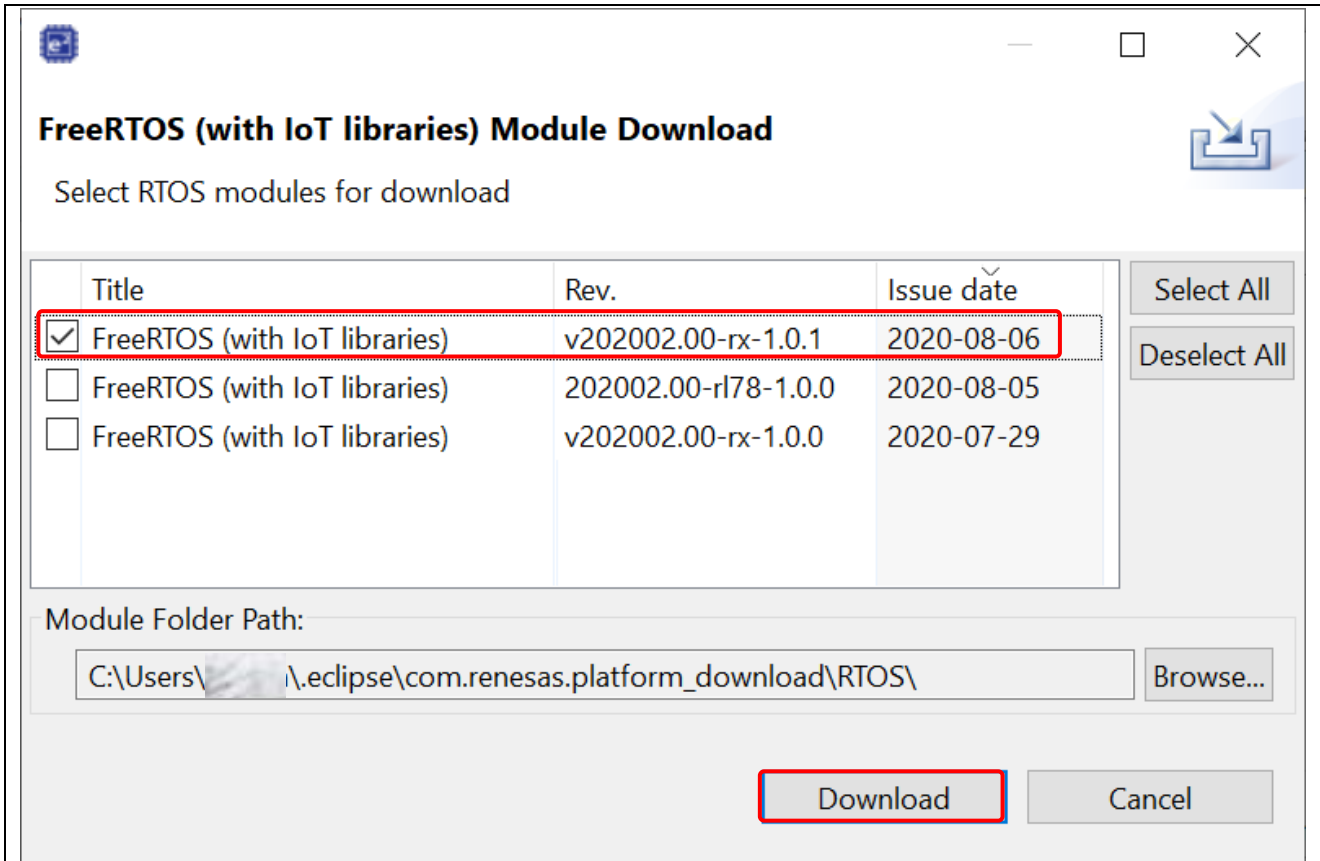


Figure 1-4 Select FreeRTOS module to download

8. Read and click [Agree] to the end-user license agreement. Wait for the download to be completed.

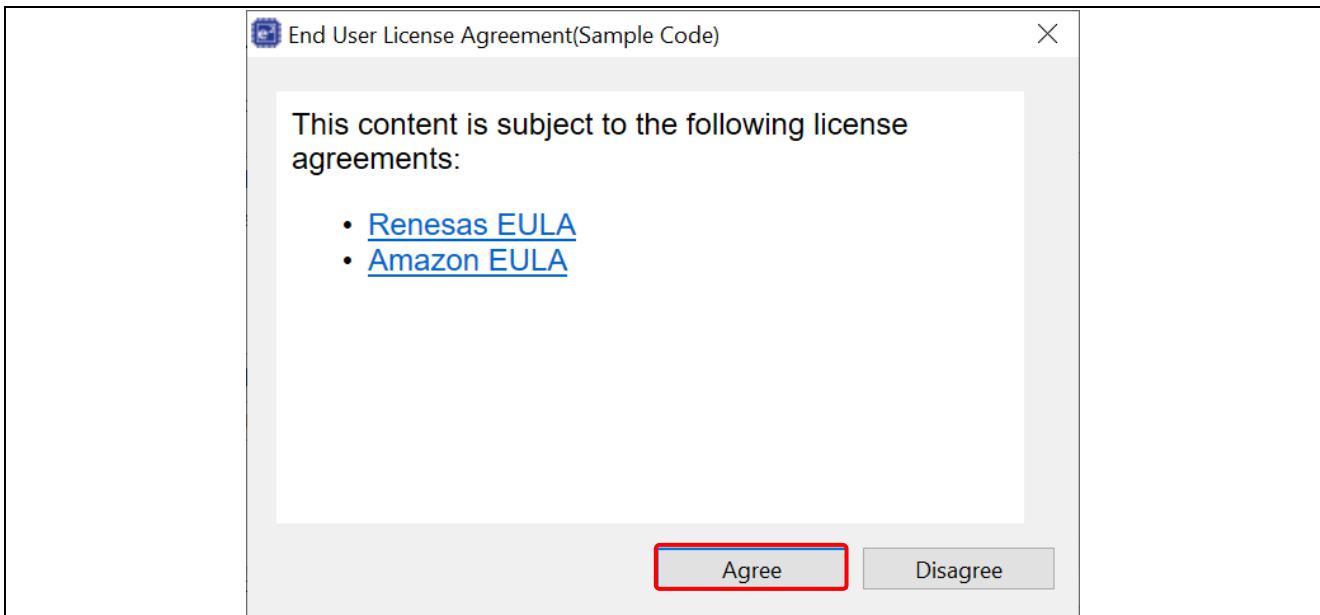


Figure 1-5 User license agreement

9. The FreeRTOS source code from Github will be downloaded to C:\Users\\.eclipse\org.eclipse.platform_download\RTOS\- 10. To use the RTOS Resources view, compile the project with output debugging information. For CC-RX, open project properties > [C/C++ Build] > [Settings] > [Tool Settings] > [Compiler] > [Object] and tick “Outputs debugging information (-debug/-nodebug)”.

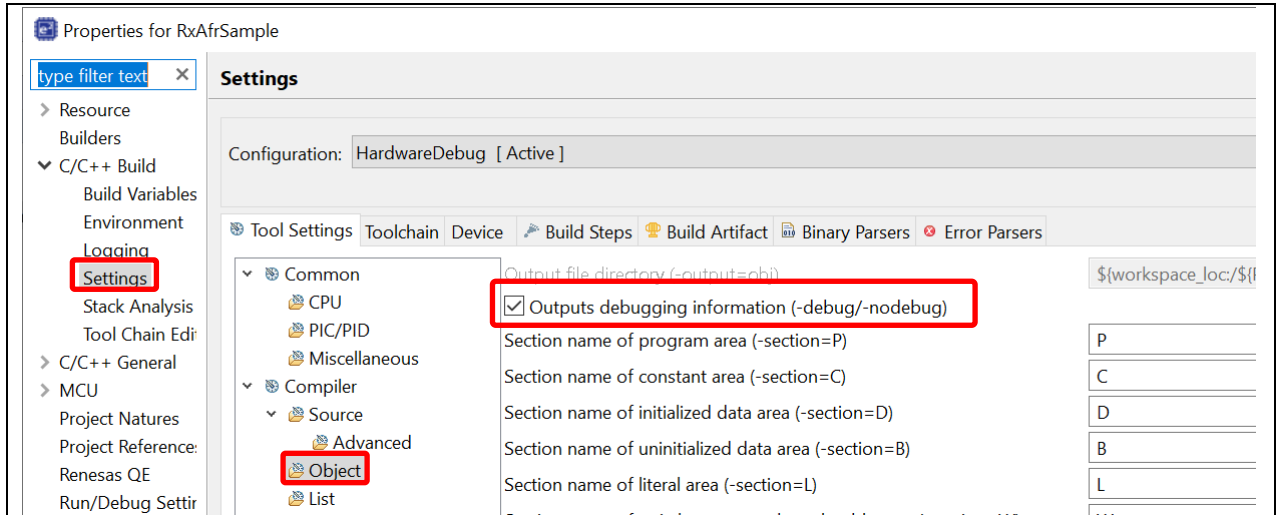


Figure 1-6 Build program with output of debugging information in CC-RX compiler

11. Perform other necessary settings (if any) and build the project.

Note: To connect to AWS, more configuration should be carried out. For further details refer to the application note [r20an0543ejxxxx](#).

<https://www.renesas.com/sg/en/search/keyword-search.html?q=r20an0543ej>

2. Introduction of RTOS Resources view

The RTOS Resources view displays information about the resources (i.e. system information and task/thread information) used by the real-time OS.

2.1 Opening the RTOS Resources view

It can be opened during the debugging session. Select menu [Renesas Views] > [Partner OS] > [RTOS Resources]. The view has a [Select OS] box for selecting the real-time OS used in the project.

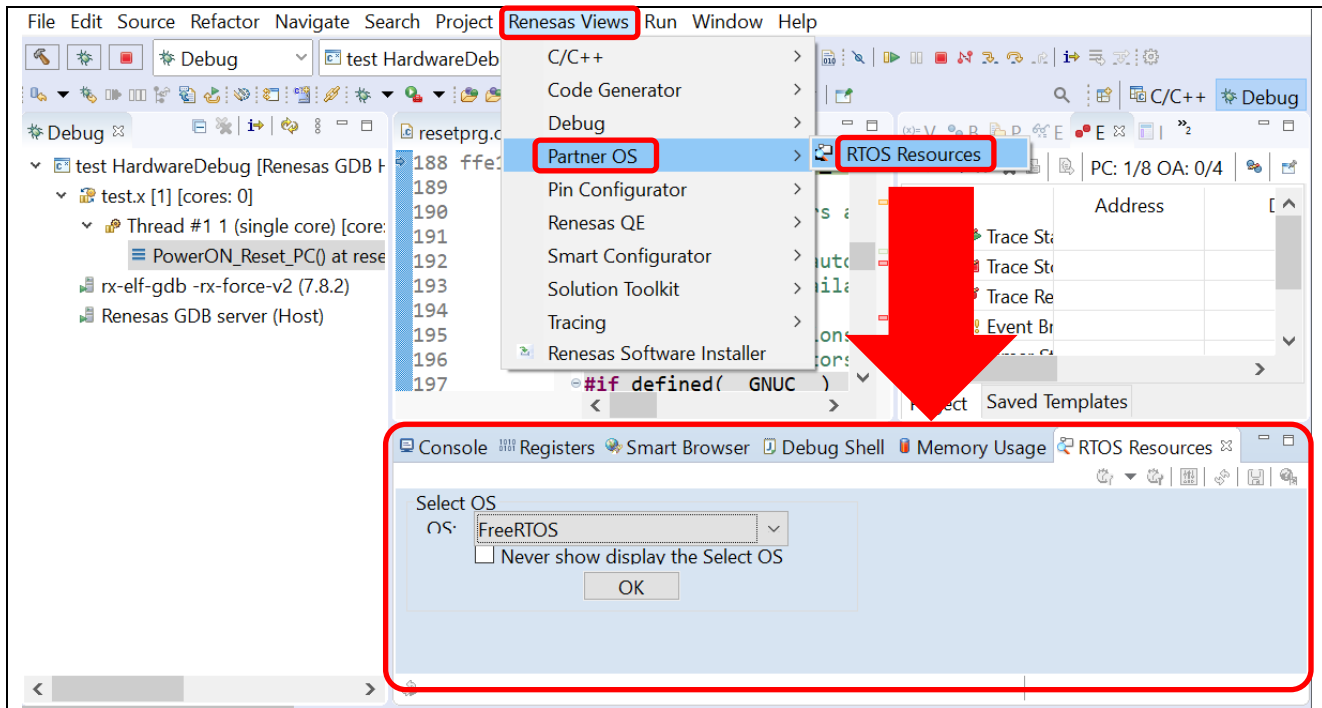


Figure 2-1 Open RTOS Resources view

2.2 Selecting the OS

After opening the view, select the real-time OS to be used. Currently, only "FreeRTOS" is supported.

Select "FreeRTOS" from the list box and click [OK].

Note: Please do not select "External" as it is for real-time OS developers.

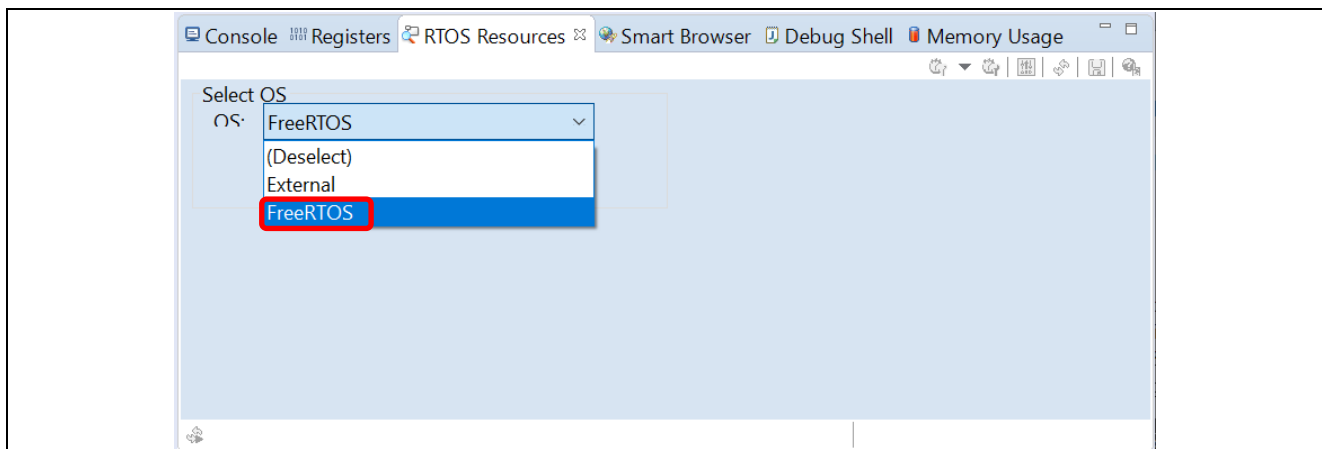


Figure 2-2 Select OS

2.3 Context menu

The context menu is displayed by right-clicking the mouse on the resource information view.

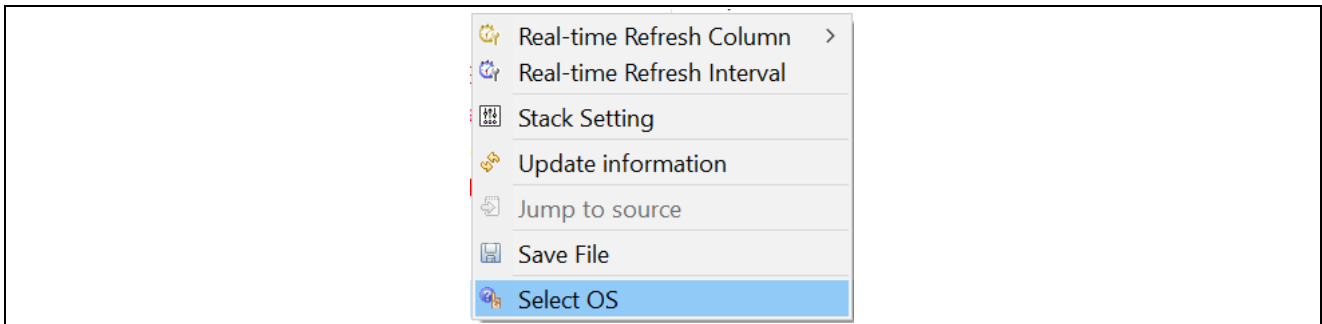


Figure 2-3 Context menu

Explanation:

- **Real-time Refresh Column:**
Allows real-time display for the displayed items.
This is not valid while the program is running.
- **Real-time Refresh Interval:**
Specifies interval time for updating of the real-time display. The specifiable range is 500ms to 10000ms.
This is invalid while the program is running.
- **Stack Setting:**
Enables/disables Stack Loading and stack threshold setting for stack alert function.
This is invalid while a program is running.
- **Update information:**
Updates the information.
- **Jump to source:**
Opens an editor view in which the source code of the task/thread or handler is displayed. An editor view is also opened by double-clicking the task/thread or handler.
This is invalid while the program is running.
- **Save File:**
Saves the data of the current tab in the text file (*.txt).
This item is invalid while the program is running.
- **Select OS:**
Opens the [Select OS] Dialog Box.
This is invalid while the program is running.

2.4 Stack setting

2.4.1 Enable load stack data and set stack threshold

- (1) Open the context menu and select “Stack Setting”.
- (2) To load stack data to the RTOS Resource view, tick “Enable loading Stack data” checkbox in the “Stack Setting” dialog. If this option is not enabled, stack data will not be loaded in the next debugging session.

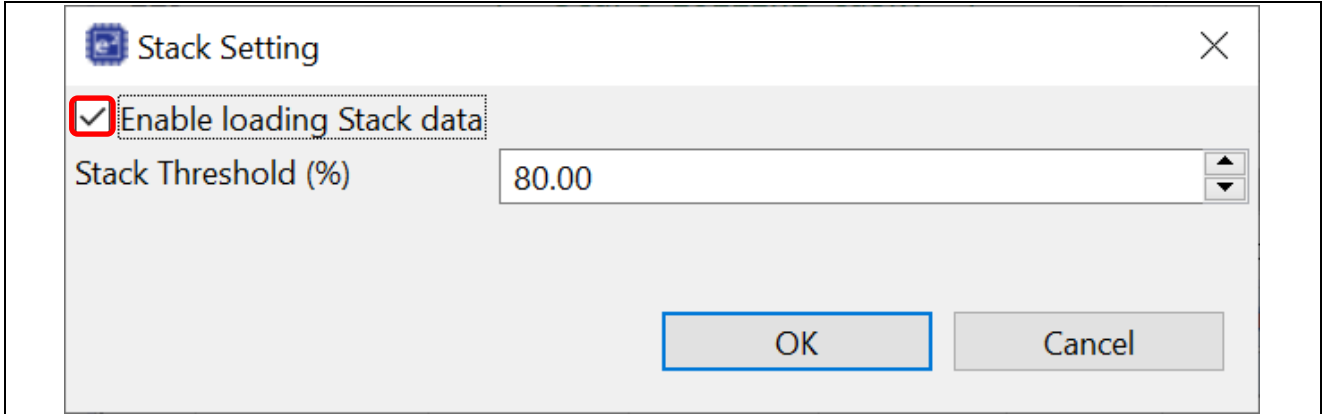


Figure 2-4 Enable loading stack data

- (3) The desired threshold value can be set in the “Stack Threshold (%)” textbox, click [OK] to save the setting.

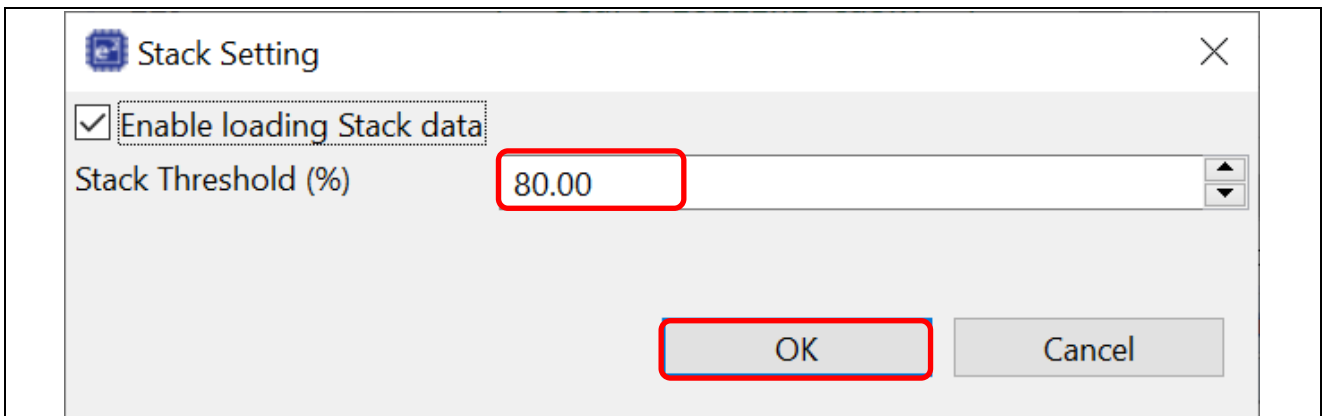


Figure 2-5 Set up threshold value

- (4) Run then suspend the target project to load stack data. The stack threshold warning will pop up if the threshold set is met.

There are 2 types of warning popup: Threshold Warning (list of threads which reached stack threshold value set as above) and Overflow Warning (reached 100%).

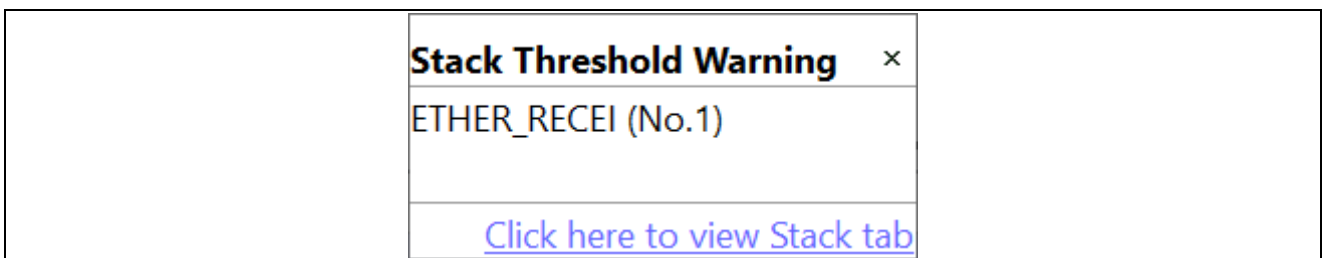


Figure 2-6 Example of Stack threshold warning popup

2.4.2 Save stack data

The stack data can be saved by selecting “Save File” from the context menu (or click the “Save File” button on the toolbar). A “Save As” dialog will be shown for user to enter the file name and location.

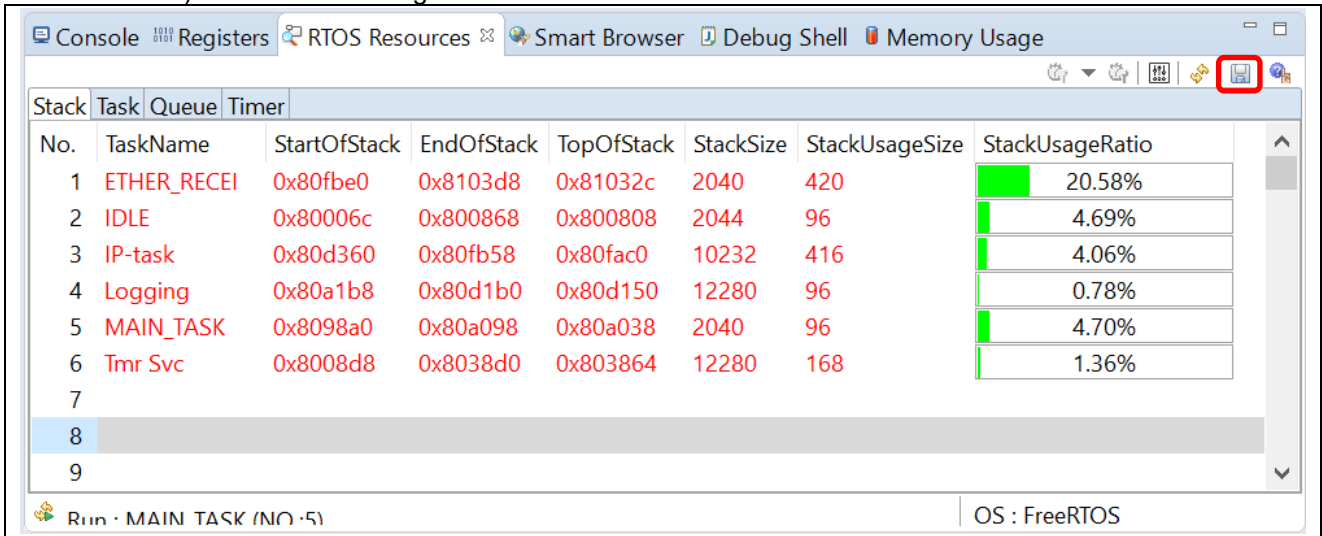


Figure 2-7 Save File button

3. Using RTOS Resources view with FreeRTOS

3.1 Task tab

This tab lists all tasks existed in the program with the following information:

No.	TaskName	Base/ActualPriority	State	EventObject	TotalTickCount	DeltaTickCount
1	ETHER_RECEI	6/6	BLOCKED	None	0x0(0.00%)	0x0(0.00%)
2	IDLE	0/0	RUNNING	None	0x0(0.00%)	0x0(0.00%)
3	IP-task	5/5	BLOCKED	None	0x0(0.00%)	0x0(0.00%)
4	Logging	0/0	SUSPENDED	None	0x0(0.00%)	0x0(0.00%)
5	MAIN_TASK	3/3	BLOCKED	None	0x0(0.00%)	0x0(0.00%)
6	Tmr Svc	6/6	SUSPENDED	None	0x1(100.00%)	0x0(0.00%)
7	TzCtrl	1/1	BLOCKED	None	0x0(0.00%)	0x0(0.00%)

Figure 3-1 Task tab

- **No.:** Row index.
- **TaskName:** The name assigned to the task upon creation.
- **Base/ActualPriority:** The base priority used by the priority inheritance mechanism/The actual priority used by the task.
- **State:** State of the task which includes “RUNNING”, “READY”, “BLOCKED” and “SUSPENDED”.
- **EventObject:** The name of the queue which causes the task to be blocked.
- **TotalTickCount:** The total number of tick count for the task to be active.
- **DeltaTickCount:** The number of tick count for the task to be active since previous suspend event.

Note: To display “TotalTickCount” and “DeltaTickCount”, define `configGENERATE_RUN_TIME_STATS` as 1 and implement the macros `portCONFIGURE_TIMER_FOR_RUN_TIME_STATS()` and `portGET_RUN_TIME_COUNTER_VALUE()` (in `<project>/config_files/FreeRTOSConfig.h`).

To configure these parameters, refer to FreeRTOS guidelines at <https://www.freertos.org/rtos-run-time-stats.html>.

```

108  /* Event group related definitions. */
109  #define configUSE_EVENT_GROUPS          1
110
111  /* Run time stats gathering definitions. */
112  unsigned long ulGetRunTimeCounterValue( void );
113  void vConfigureTimerForRunTimeStats( void );
114  #define configGENERATE_RUN_TIME_STATS  1
115  #define portCONFIGURE_TIMER_FOR_RUN_TIME_STATS()  vConfigureTimerForRunTimeStats()
116  #define portGET_RUN_TIME_COUNTER_VALUE()      ulGetRunTimeCounterValue()
117
118  /* Co-routine definitions. */
119  #define configUSE_CO_ROUTINES          0
120  #define configMAX_CO_ROUTINE_PRIORITIES  ( 2 )
121

```

Figure 3-2 Define `configGENERATE_RUN_TIME_STATS` in `FreeRTOSConfig.h`

After defining the 2 Run time statistics macros as above, user should implement 2 functions, `vConfigureTimerForRunTimeStats()` and `ulGetRunTimeCounterValue()`.

The figure below shows an implementation with empty functions, user should implement the functions according to the project specification.

```

88     *ppxTimerTaskStackBuffer = uxTimerTaskStack;
89
90     /* Pass out the size of the array pointed to by *ppxTimerTaskStackBu
91     * Note that, as the array is necessarily of type StackType_t,
92     * configMINIMAL_STACK_SIZE is specified in words, not bytes. */
93     *pulTimerTaskStackSize = configTIMER_TASK_STACK_DEPTH;
94 }
95 /*-----*/
96 void vConfigureTimerForRunTimeStats( void )
97 {
98     return;
99 }
100 /*-----*/
101 unsigned long ulGetRunTimeCounterValue( void )
102 {
103     return (unsigned long)(1);
104 }
105
106

```

Figure 3-3 Implementation of Run Time Statistics functions

3.2 Queue tab

This tab lists all queues/semaphores/mutexes used in the program.

To display queue information, specify `configQUEUE_REGISTRY_SIZE` with value greater than 0 in `<project>/config_files/FreeRTOSConfig.h`.

In addition, the function `vQueueAddToRegistry()` should be called. Note that this function call is already implemented in the demo code.

```

60 #define configUSE_TRACE_FACILITY 1
61 #define configUSE_16_BIT_TICKS 0
62 #define configIDLE_SHOULD_YIELD 1
63 #define configUSE_CO_ROUTINES 0
64 #define configUSE_MUTEXES 1
65 #define configUSE_RECURSIVE_MUTEXES 1
66 #define configQUEUE_REGISTRY_SIZE 10
67 #define configUSE_APPLICATION_TASK_TAG 0
68 #define configUSE_COUNTING_SEMAPHORES 1
69 #define configUSE_ALTERNATIVE_API 0
70 #define configNUM_THREAD_LOCAL_STORAGE_POINTERS 3 /* FreeRTOS+FP
71 #define configRECORD_STACK_HIGH_ADDRESS 1
72
73 #define configUSE_DAEMON_TASK_STARTUP_HOOK 1
74
75 #define configCPU_CLOCK_HZ (BSP_ICLK_HZ)
76 #define configPERIPHERAL_CLOCK_HZ (BSP_PCLKB_HZ)
77 #define configUSE_QUEUE_SETS 1

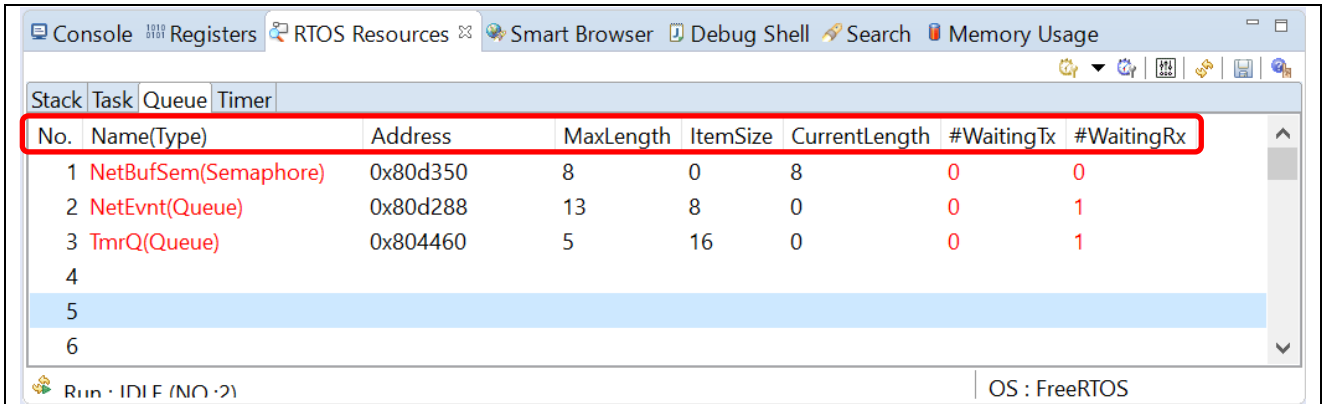
```

Figure 3-4 Define `configQUEUE_REGISTRY_SIZE` in `FreeRTOSConfig.h`

The queue tab displays the following information:

- **No.:** Row index.
- **Name(Type):** The name assigned to the queue upon registration and its type (Queue, Semaphore or Mutex).

- **Address:** The address of the queue handle.
- **MaxLength:** The maximum number of items that can be stored in the queue.
- **ItemSize:** Size per item in the queue (in bytes).
- **CurrentLength:** Number of items currently stored in the queue.
- **#WaitingTx:** Number of tasks blocked while waiting to send to the queue.
- **#WaitingRx:** Number of tasks blocked while waiting to receive from the queue.

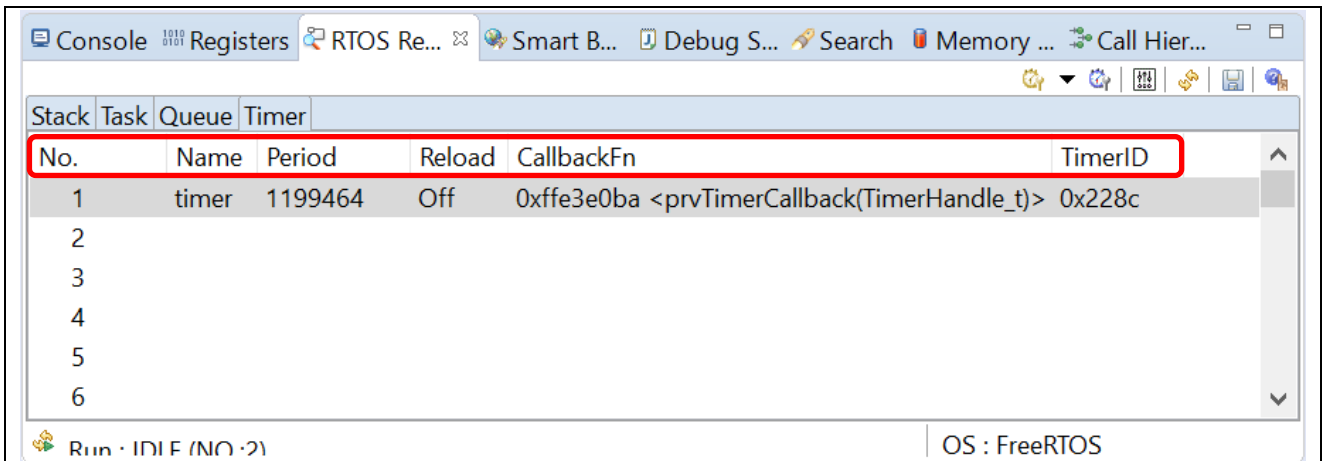


No.	Name(Type)	Address	MaxLength	ItemSize	CurrentLength	#WaitingTx	#WaitingRx
1	NetBufSem(Semaphore)	0x80d350	8	0	8	0	0
2	NetEvt(Queue)	0x80d288	13	8	0	0	1
3	TmrQ(Queue)	0x804460	5	16	0	0	1
4							
5							
6							

Figure 3-5 Queue tab

3.3 Timer tab

This tab lists all timers used in the program. The following information is displayed in the timer tab:



No.	Name	Period	Reload	CallbackFn	TimerID
1	timer	1199464	Off	0xffe3e0ba <prvTimerCallback(TimerHandle_t)>	0x228c
2					
3					
4					
5					
6					

Figure 3-6 Timer tab

- **No.:** Row index.
- **Name:** The name assigned to the software timer upon creation.
- **Period:** The current period of the timer in system ticks.
- **Reload:** Automatic reload Enable / Disable. "On" when auto reload is enabled which resets the timer each time it expires, "Off" when auto reload is disabled which does nothing when the timer expires.
- **CallbackFn:** Address and <Name> of the callback function which executes each time the timer ends.
- **TimerID:** The numeric ID of the timer assigned in hexadecimal format when it was created.

3.4 Stack tab

This tab lists all stacks associated with tasks that existed in the program. The following information is displayed in the stack tab:

No.	TaskName	StartOfStack	EndOfStack	TopOfStack	StackSize	StackUsageSize	StackUsageRatio
1	ETHER_RECEI	0x80fbe0	0x8103d8	0x81032c	2040	464	22.74%
2	IDLE	0x80006c	0x800868	0x800800	2044	148	7.24%
3	IP-task	0x80d360	0x80fb58	0x80faa4	10232	1612	15.75%
4	Logging	0x80a1b8	0x80d1b0	0x80d120	12280	188	1.53%
5	MAIN_TASK	0x8098a0	0x80a098	0x80a02c	2040	136	6.66%
6	Tmr Svc	0x8008d8	0x8038d0	0x803858	12280	1204	9.80%
7	iot_thread	0x810708	0x814700	0x814400	16376	1236	7.54%
8	iot_thread	0x8148e8	0x8158e0	0x815828	4088	212	5.18%
9	iot_thread	0x8158f0	0x8168e8	0x816830	4088	212	5.18%
10							

Figure 3-7 Stack tab

- **No.:** Row index.
- **TaskName:** The name assigned to the task upon creation.
- **StartOfStack:** The address of the start of stack.
- **EndOfStack:** The address of the end of stack.
- **TopOfStack:** The address of the top of the stack where it is last written to when the context of the stack was saved.
- **StackSize:** Total stack size.
- **StackUsageSize:** Stack usage at high water mark.
- **StackUsageRatio:** Percentage of usage at high water mark relative to total stack size.

Note:

- (1) To display "EndOfStack" and "StackSize", define "configRECORD_STACK_HIGH_ADDRESS" as 1 in <project>/config_files/FreeRTOSConfig.h file (this is already set for the existing project).

```

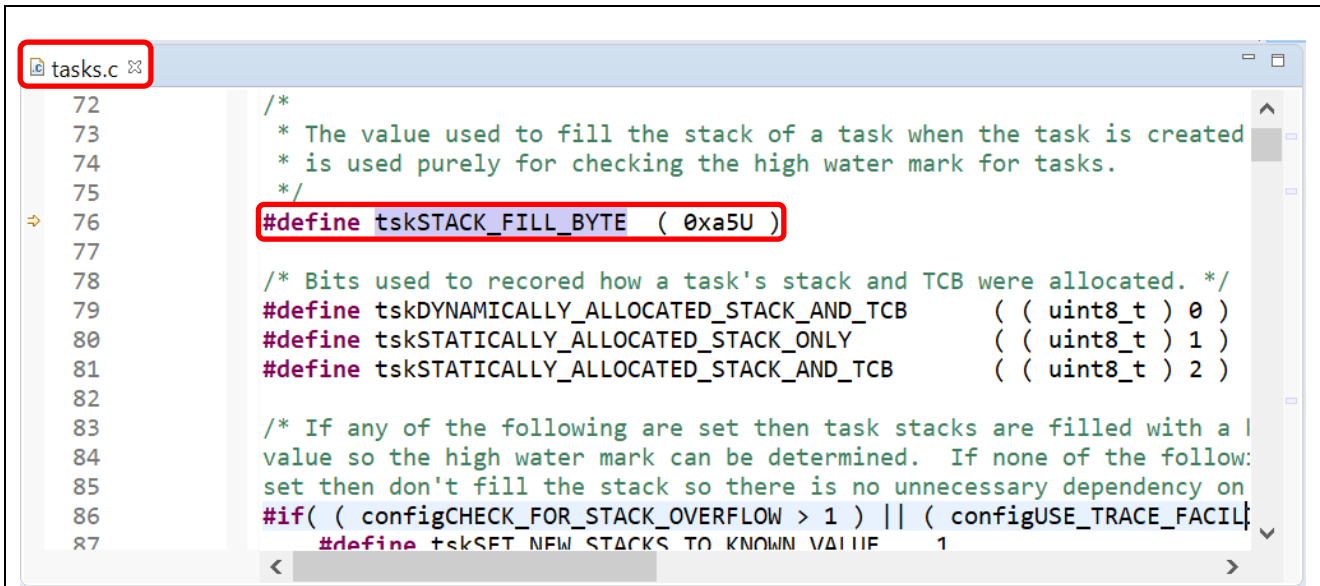
59  #define configMAX_TASK_NAME_LEN          ( 12 )
60  #define configUSE_TRACE_FACILITY        1
61  #define configUSE_16_BIT_TICKS         0
62  #define configIDLE_SHOULD_YIELD        1
63  #define configUSE_CO_ROUTINES          0
64  #define configUSE_MUTEXES              1
65  #define configUSE_RECURSIVE_MUTEXES   1
66  #define configQUEUE_REGISTRY_SIZE      10
67  #define configUSE_APPLICATION_TASK_TAG  0
68  #define configUSE_COUNTING_SEMAPHORES  1
69  #define configUSE_ALTERNATIVE_API      0
70  #define configNUM_THREAD_LOCAL_STORAGE_POINTERS 3 /* FreeRTOS+F
71  #define configRECORD_STACK_HIGH_ADDRESS 1
72
73  #define configUSE_DAEMON_TASK_STARTUP_HOOK 1
74

```

Figure 3-8 Define configRECORD_STACK_HIGH_ADDRESS in FreeRTOSConfig.h

- (2) To display “StackUsageSize” and “StackUsageRatio”, define “configRECORD_STACK_HIGH_ADDRESS” as 1 in FreeRTOSConfig.h file, and “tskSTACK_FILL_BYTE” as 0xA5U in <workspace>/freertos_kernel/task.c file.

Only devices with portSTACK_GROWTH defined as -1 are supported (in <workspace>/freertos_kernel/portable/<compiler name>/<processor name>/portmacro.h).

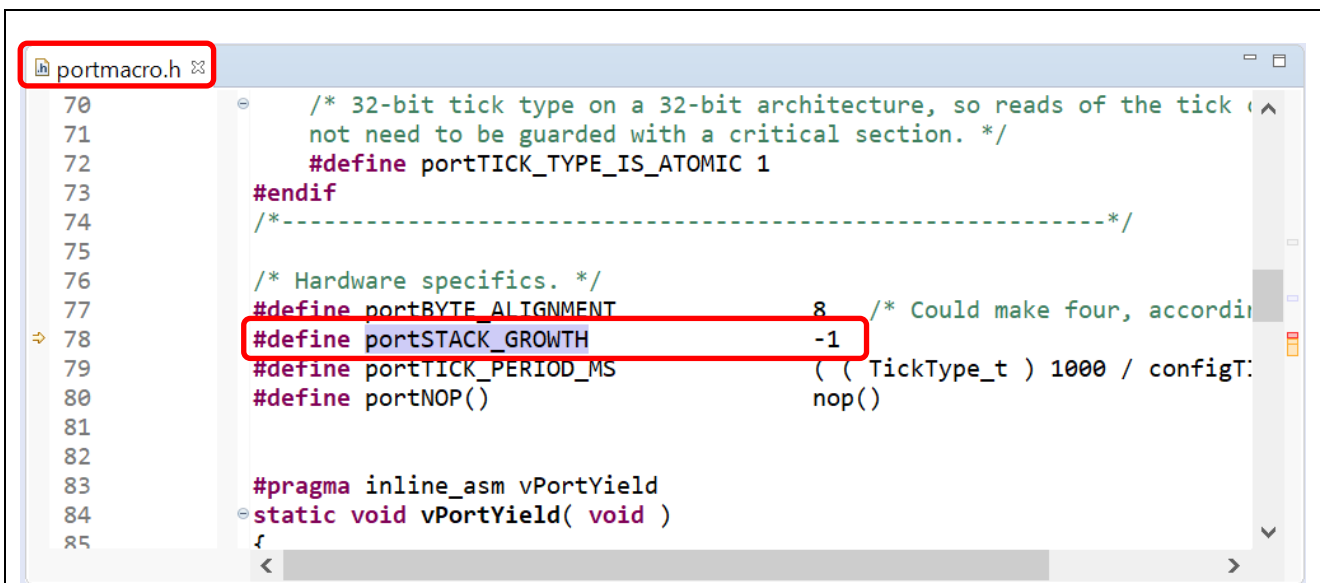


```

72      /*
73       * The value used to fill the stack of a task when the task is created
74       * is used purely for checking the high water mark for tasks.
75       */
76      #define tskSTACK_FILL_BYTE ( 0xa5U )
77
78      /* Bits used to record how a task's stack and TCB were allocated. */
79      #define tskDYNAMICALLY_ALLOCATED_STACK_AND_TCB      ( ( uint8_t ) 0 )
80      #define tskSTATICALLY_ALLOCATED_STACK_ONLY         ( ( uint8_t ) 1 )
81      #define tskSTATICALLY_ALLOCATED_STACK_AND_TCB      ( ( uint8_t ) 2 )
82
83      /* If any of the following are set then task stacks are filled with a
84       value so the high water mark can be determined. If none of the follow:
85       set then don't fill the stack so there is no unnecessary dependency on
86       #if( ( configCHECK_FOR_STACK_OVERFLOW > 1 ) || ( configUSE_TRACE_FACIL
87       #define tskSET_NEW STACKS TO KNOWN VALUE          1

```

Figure 3-9 Define tskSTACK_FILL_BYTE in task.c



```

70      /* 32-bit tick type on a 32-bit architecture, so reads of the tick (
71       not need to be guarded with a critical section. */
72      #define portTICK_TYPE_IS_ATOMIC 1
73      #endif
74      /*-----*/
75
76      /* Hardware specifics. */
77      #define portBYTE_ALIGNMENT      8 /* Could make four, accordin
78      #define portSTACK_GROWTH        -1
79      #define portTICK_PERIOD_MS      ( ( TickType_t ) 1000 / configT
80      #define portNOP()                nop()
81
82
83      #pragma inline_asm vPortYield
84      static void vPortYield( void )
85      {

```

Figure 3-10 Define portSTACK_GROWTH in portmacro.h

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.20.20	-	First release document

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.

Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems.

The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.