e2 studio

Partner RTOS Aware Debugging for RA

Introduction

Renesas e^2 studio is a development environment based on the popular Eclipse CDT (C/C++ Development Tool). It includes a build (editor, compiler and linker control) functions as well as debug interface. It also supports integrating the Renesas GitHub FreeRTOS (with IoT libraries) demo applications and runs them on Renesas boards.

The Partner OS debugging plug-in provides a view in e^2 studio named RTOS Resources view. This view displays information on the usage of resources by the real-time OS. Items that can be displayed vary according to the real-time OS.

Objectives

This document introduces the usage of RTOS Resource view in e^2 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

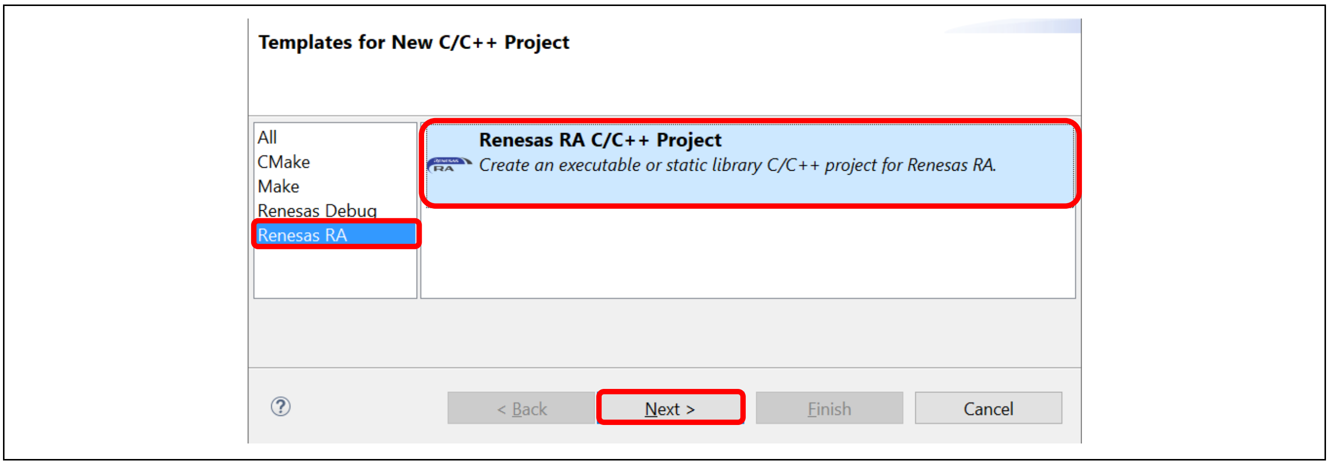
IDE	e^2 studio v2020-10 + FSP v2.2.0 e^2 studio v7.8 + FSP v1.0.0
Toolchains	GNU-ARM Embedded Toolchain version 9-2019-q4-major
Target devices	Renesas RA Family (EK-RA6M3)
Debuggers	SEGGER J-Link
Target OS	FreeRTOS

Contents

1. Creating the FreeRTOS project

The following steps show how to create a FreeRTOS project.

1. Launch e^2 studio.
2. Select **File** → **New** → **C/C++ Project** from the menu.
   Select **Renesas RA** → **Renesas RA C/C++ Project** and click **Next**.

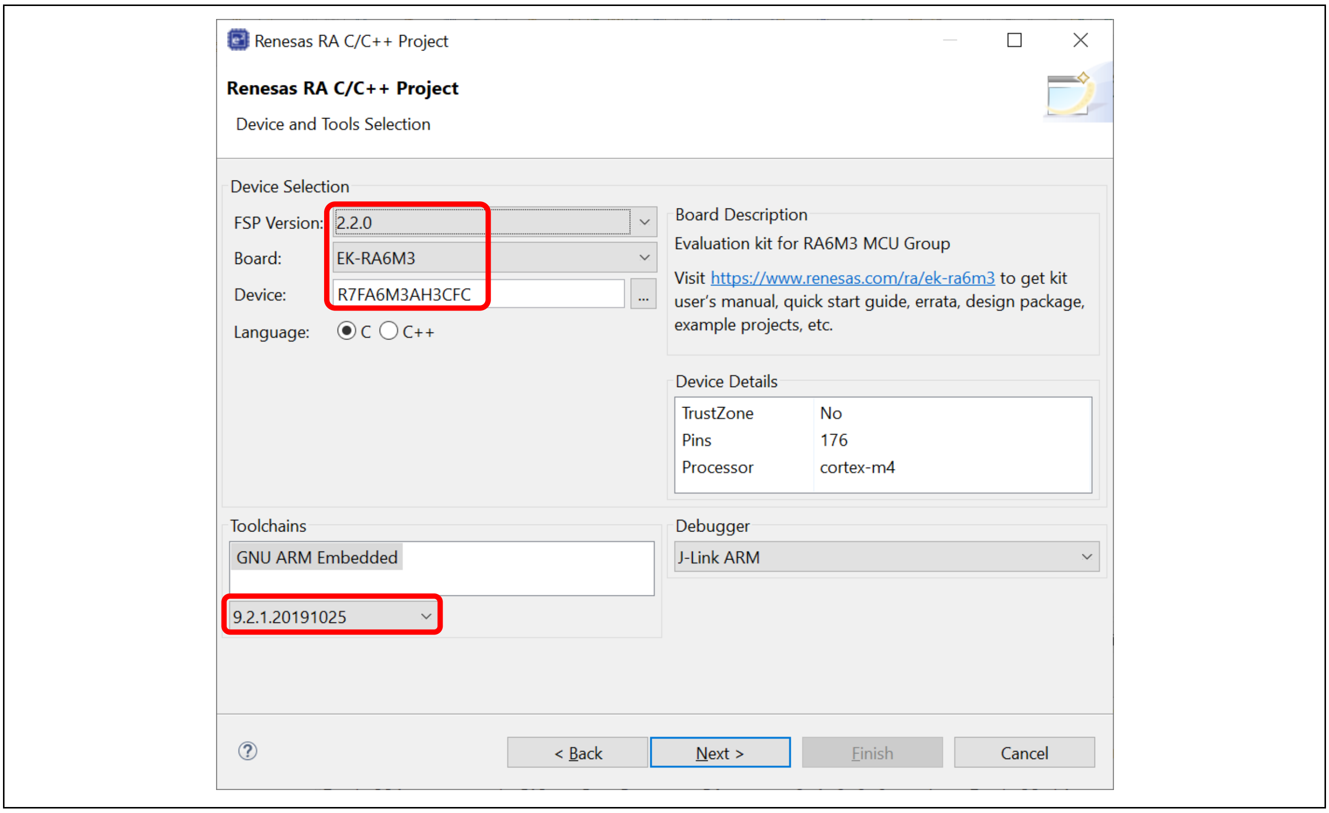


Templates for New C/C++ Project

Renesas RA C/C++ Project
Create an executable or static library C/C++ project for Renesas RA.

Figure 1-1. Select project template

3. Name the project and click **Next**.
4. Specify the following information and click **Next**:
   - FSP Version: 2.2.0
   - Board: EK-RA6M3
   - Toolchain: 9.2.1.20191025
   - Debugger: J-Link ARM



Renesas RA C/C++ Project

Renesas RA C/C++ Project

Device and Tools Selection

Device Selection

FSP Version: 2.2.0
Board: EK-RA6M3
Device: R7FA6M3AH3CFC
Language: ● C ○ C++

Board Description

Evaluation kit for RA6M3 MCU Group

Visit https://www.renesas.com/ra/ek-ra6m3 to get kit user's manual, quick start guide, errata, design package, example projects, etc.

Device Details

TrustZone	No
Pins	176
Processor	cortex-m4

Toolchains

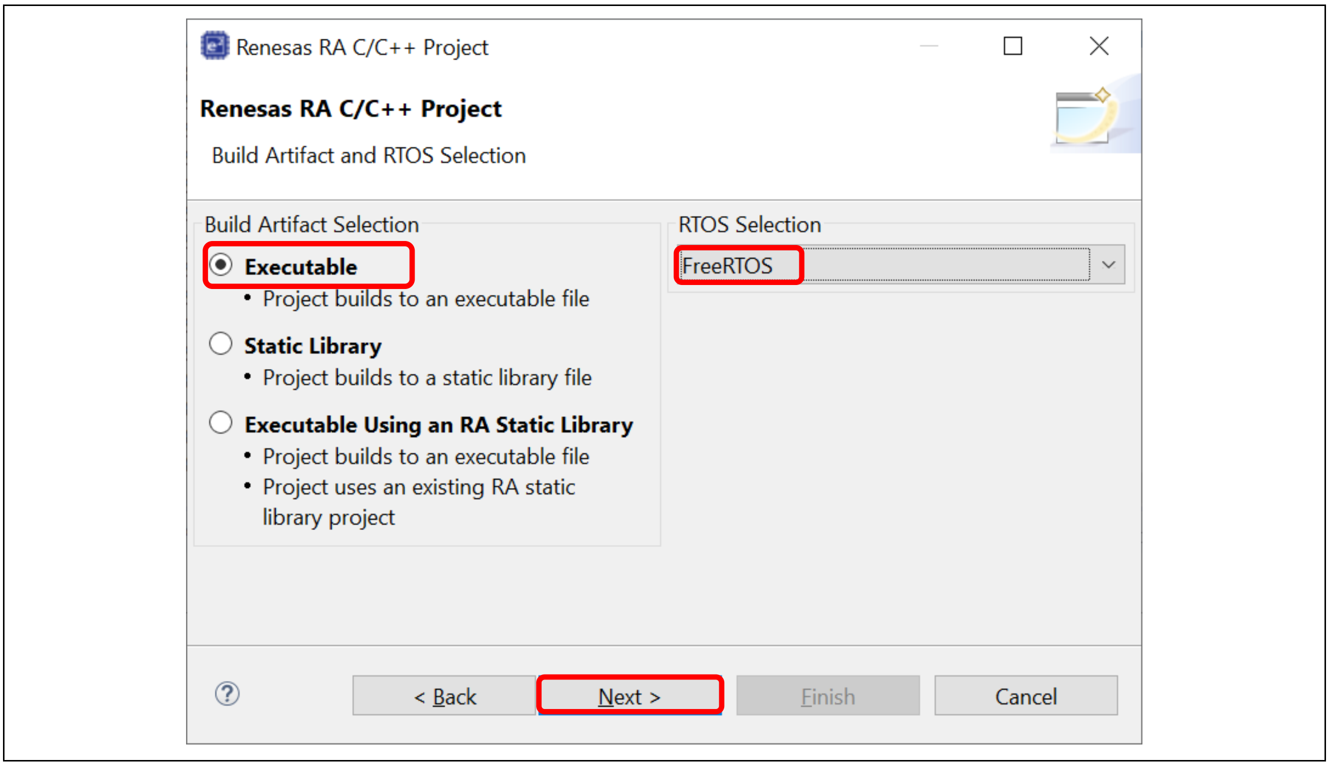
GNU ARM Embedded
9.2.1.20191025

Debugger

J-Link ARM

Figure 1-2. Select device and tool

5. Select the build artifact and RTOS, then click **Next**.



Renesas RA C/C++ Project

Renesas RA C/C++ Project

Build Artifact and RTOS Selection

Build Artifact Selection

● **Executable**
- Project builds to an executable file

○ **Static Library**
- Project builds to a static library file

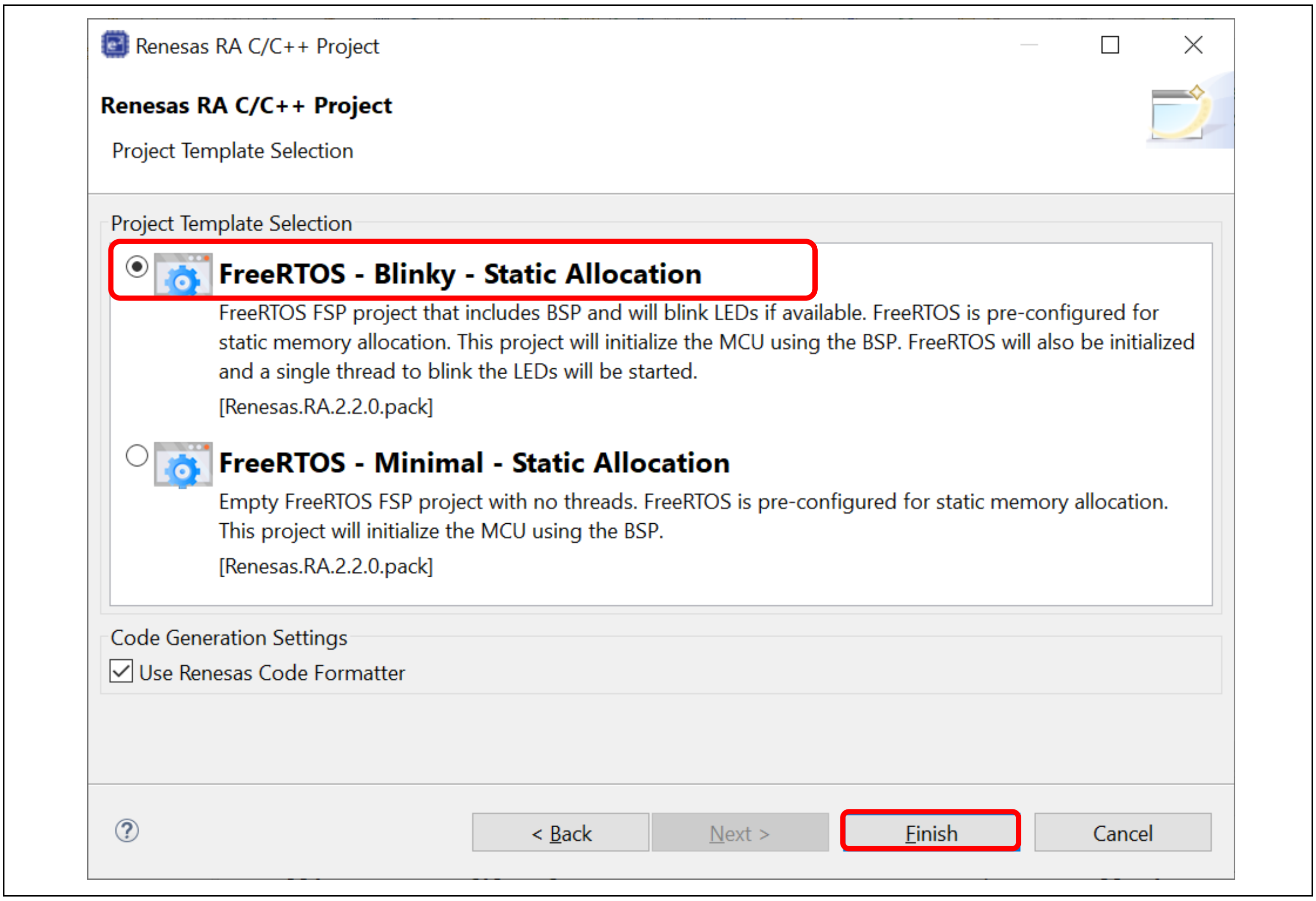
○ **Executable Using an RA Static Library**
- Project builds to an executable file
- Project uses an existing RA static library project

RTOS Selection

FreeRTOS

Figure 1-3. Select build artifact and FreeRTOS

6. In the **Project Template Selection** dialog, select **FreeRTOS – Blinky – Static Allocation**. Click **Finish**.



Renesas RA C/C++ Project

Renesas RA C/C++ Project

Project Template Selection

Project Template Selection

● **FreeRTOS - Blinky - Static Allocation**

FreeRTOS FSP project that includes BSP and will blink LEDs if available. FreeRTOS is pre-configured for static memory allocation. This project will initialize the MCU using the BSP. FreeRTOS will also be initialized and a single thread to blink the LEDs will be started.

[Renesas.RA.2.2.0.pack]

○ **FreeRTOS - Minimal - Static Allocation**

Empty FreeRTOS FSP project with no threads. FreeRTOS is pre-configured for static memory allocation. This project will initialize the MCU using the BSP.

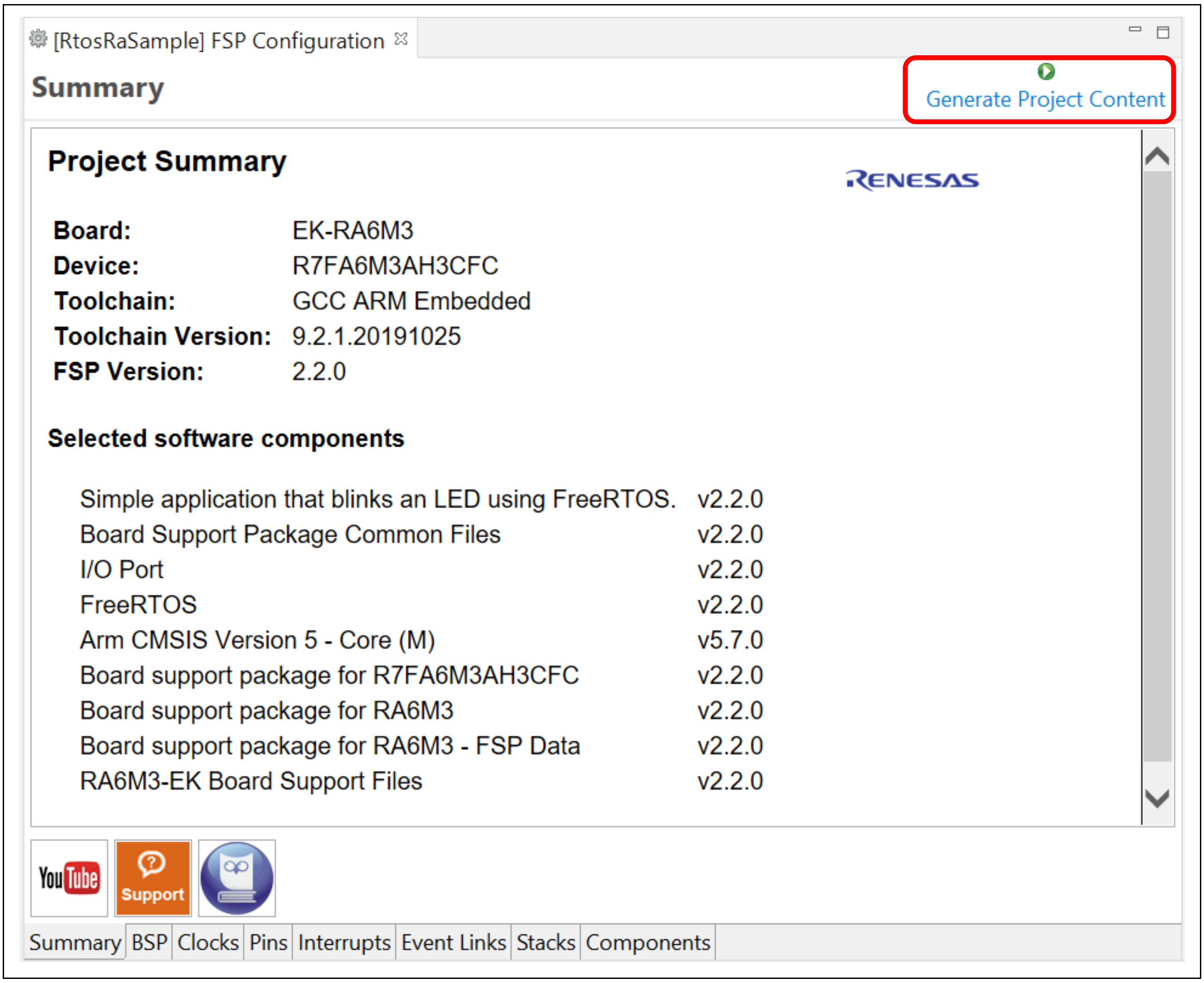
[Renesas.RA.2.2.0.pack]

Code Generation Settings

☑ Use Renesas Code Formatter

Figure 1-4. Select project template

7. After the project is created, click **Generate Project Content** in the **RA Configuration** window.



Project Summary

Board:	EK-RA6M3
Device:	R7FA6M3AH3CFC
Toolchain:	GCC ARM Embedded
Toolchain Version:	9.2.1.20191025
FSP Version:	2.2.0

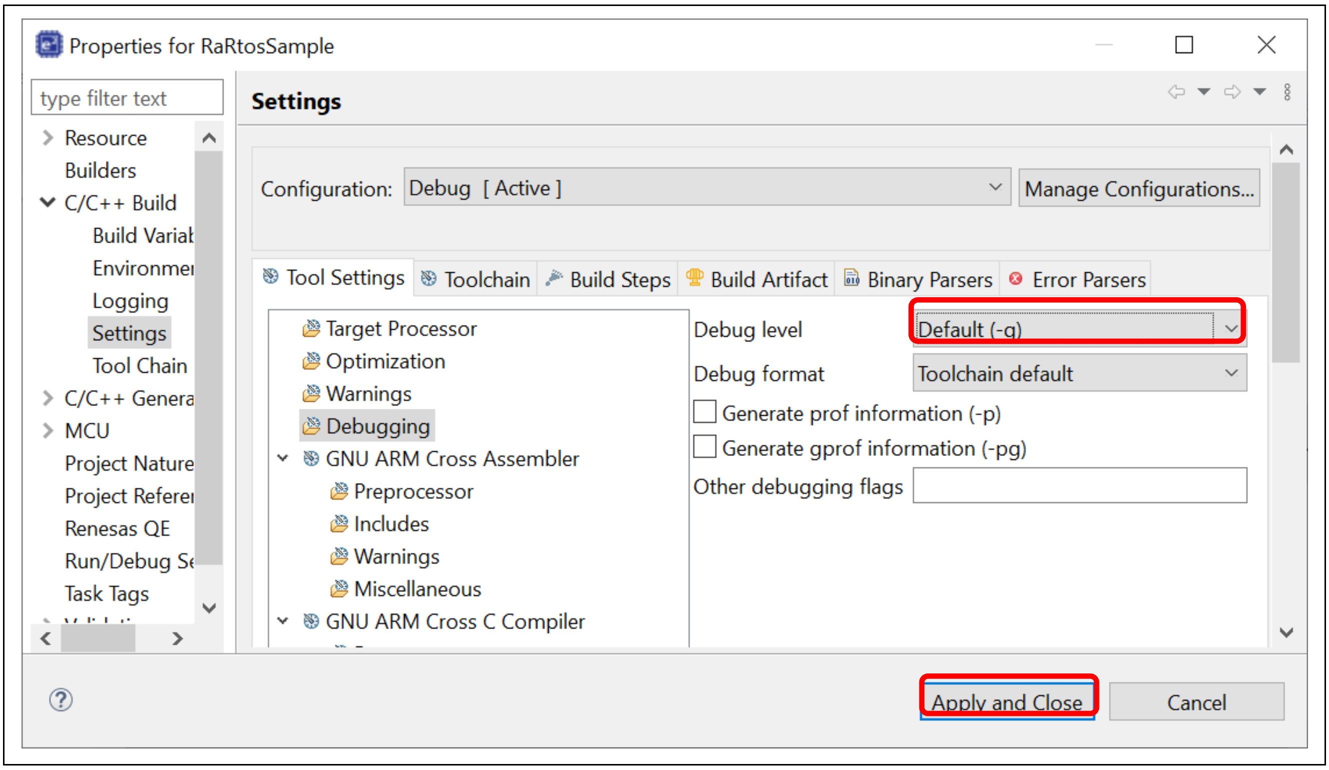
Selected software components

Simple application that blinks an LED using FreeRTOS.	v2.2.0
Board Support Package Common Files	v2.2.0
I/O Port	v2.2.0
FreeRTOS	v2.2.0
Arm CMSIS Version 5 - Core (M)	v5.7.0
Board support package for R7FA6M3AH3CFC	v2.2.0
Board support package for RA6M3	v2.2.0
Board support package for RA6M3 - FSP Data	v2.2.0
RA6M3-EK Board Support Files	v2.2.0

Figure 1-5. Generate project content

To use the **RTOS Resources** view, downloaded programs must have been compiled with the output of debugging information. For RA project, open project **Properties > C/C++ Build > Settings > Tool Settings > Debugging** and select at least **Default (-g)** for **Debug level** (do not select **None**). For further details, refer to the user manual of the GCC compiler.

Finally, build the project.



Configuration: Debug [Active]

Debug level: Default (-g)

Debug format: Toolchain default

□ Generate prof information (-p)

□ Generate gprof information (-pg)

Other debugging flags:

Figure 1-6. Build program with output of debugging information

2. Introduction of RTOS Resources view

The **RTOS Resources** view displays information about the resources (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.

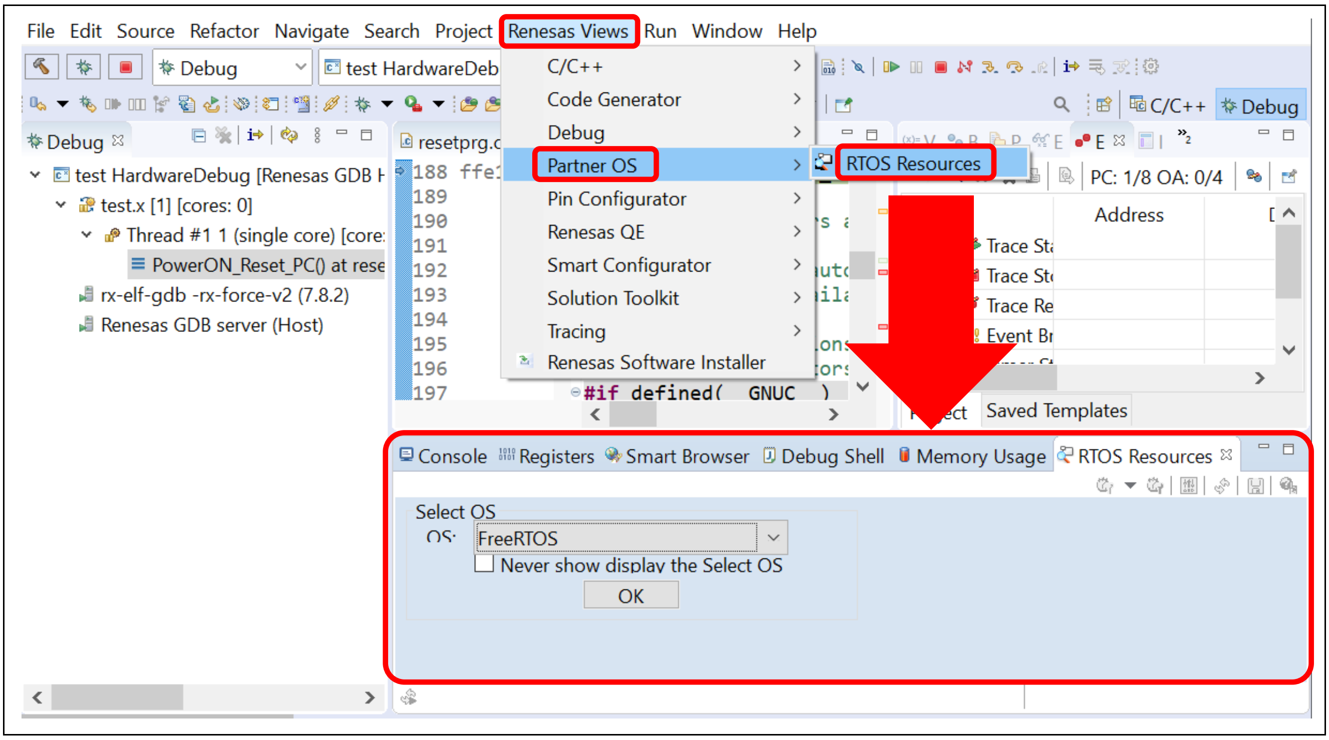

File Edit Source Refactor Navigate Search Project Renesas Views Run Window Help
C/C++
Code Generator
Debug
Partner OS
Pin Configurator
Renesas QE
Smart Configurator
Solution Toolkit
Tracing
Renesas Software Installer
RTOS Resources
Console Registers Smart Browser Debug Shell Memory Usage RTOS Resources
Select OS
OS: FreeRTOS
□ Never show display the Select OS
OK

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.

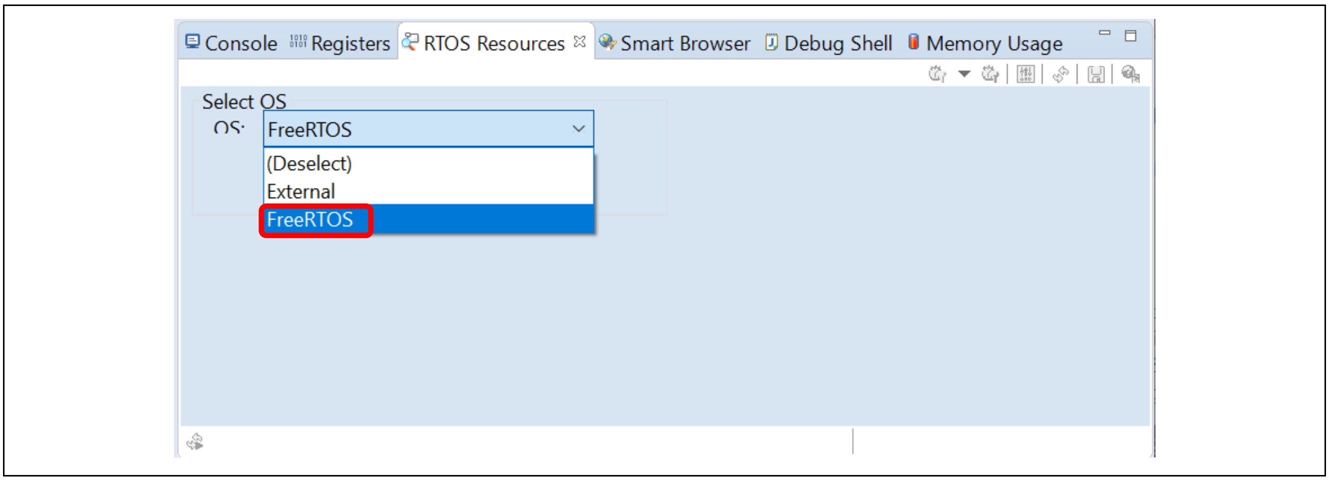

Console Registers RTOS Resources Smart Browser Debug Shell Memory Usage
Select OS
OS: FreeRTOS
(Deselect)
External
FreeRTOS

Figure 2-2. Select OS

2.3 Context menu

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

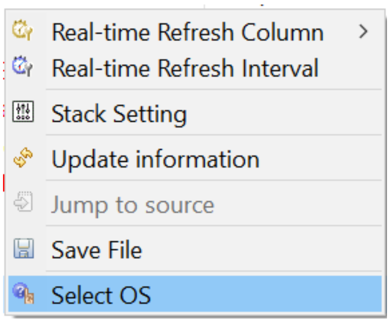


Figure 2-3. Context menu

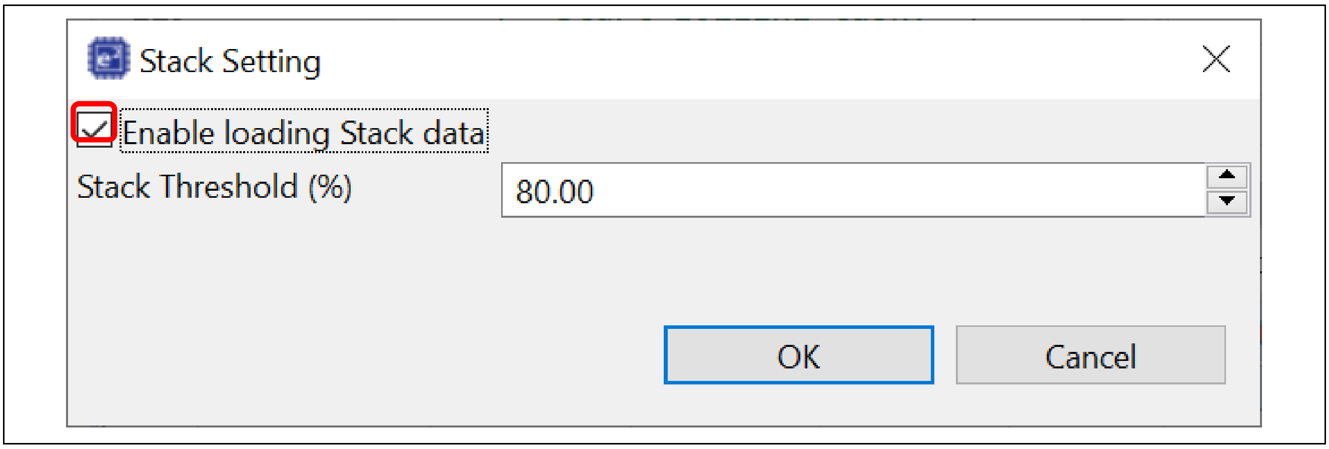
Explanation:

- **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 500 ms to 10000 ms.
  This is not valid while the program is running.
- **Stack Setting:**
  Enables/disables Stack Loading and stack threshold setting for stack alert function.
  This is not valid 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 not valid while the program is running.
- **Save File:**
  Saves the data of the current tab in the text file (*.txt).
  This is not valid while the program is running.
- **Select OS:**
  Opens the **Select OS** dialog box.
  This is not valid 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 the **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.



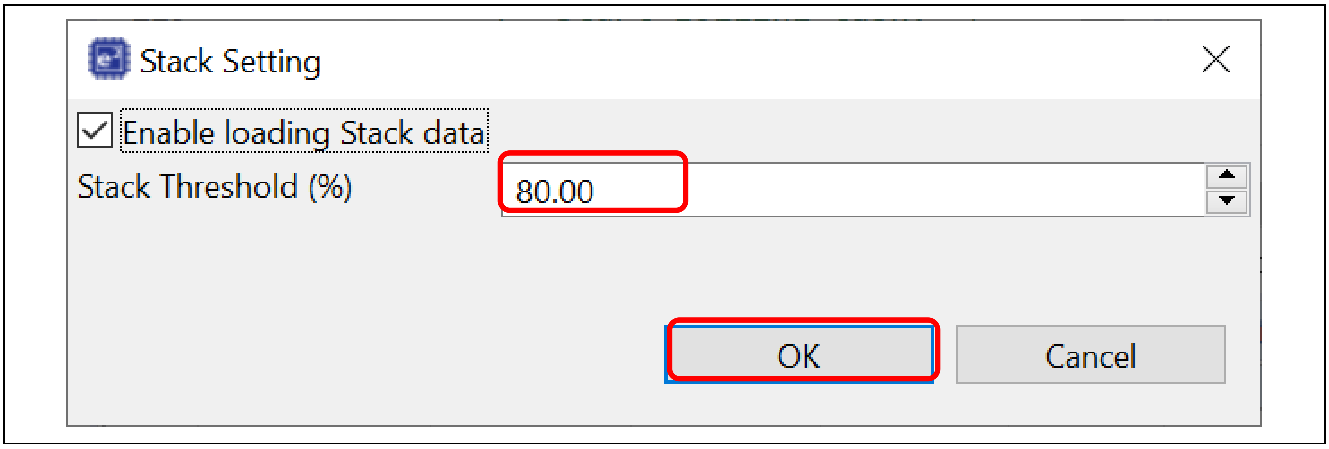
Stack Setting

☑ Enable loading Stack data

Stack Threshold (%) 80.00

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.



Stack Setting

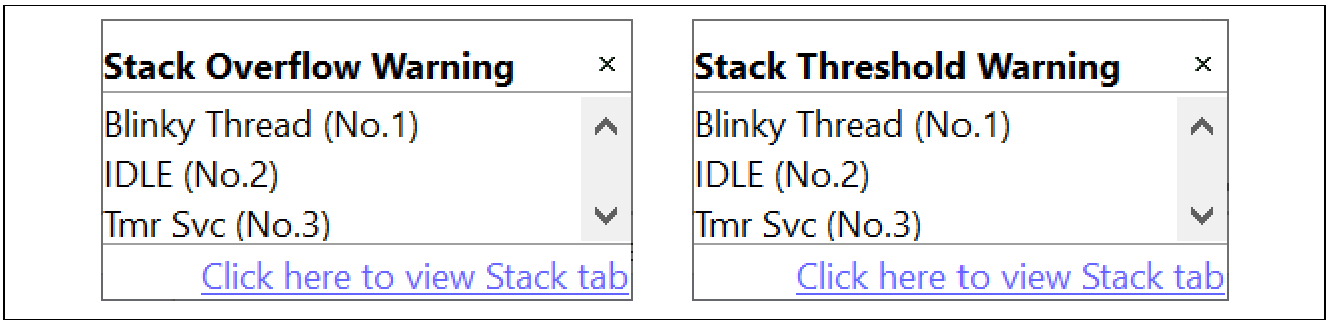
☑ Enable loading Stack data

Stack Threshold (%) 80.00

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



Stack Overflow Warning

Blinky Thread (No.1)
IDLE (No.2)
Tmr Svc (No.3)

Click here to view Stack tab

Stack Threshold Warning

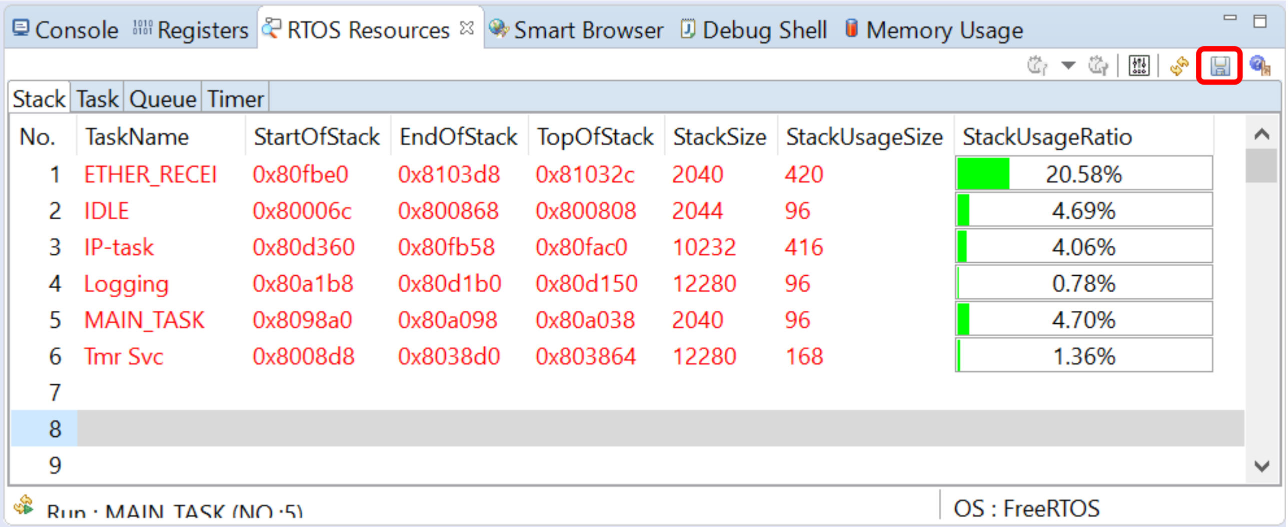
Blinky Thread (No.1)
IDLE (No.2)
Tmr Svc (No.3)

Click here to view Stack tab

Figure 2-6. Stack overflow warning popup (left) and Stack threshold warning popup (right)

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.



No.	TaskName	StartOfStack	EndOfStack	TopOfStack	StackSize	StackUsageSize	StackUsageRatio
1	ETHER_RECEI	0x80fbe0	0x8103d8	0x81032c	2040	420	20.58%
2	IDLE	0x80006c	0x800868	0x800808	2044	96	4.69%
3	IP-task	0x80d360	0x80fb58	0x80fac0	10232	416	4.06%
4	Logging	0x80a1b8	0x80d1b0	0x80d150	12280	96	0.78%
5	MAIN_TASK	0x8098a0	0x80a098	0x80a038	2040	96	4.70%
6	Tmr Svc	0x8008d8	0x8038d0	0x803864	12280	168	1.36%
7							
8							
9							

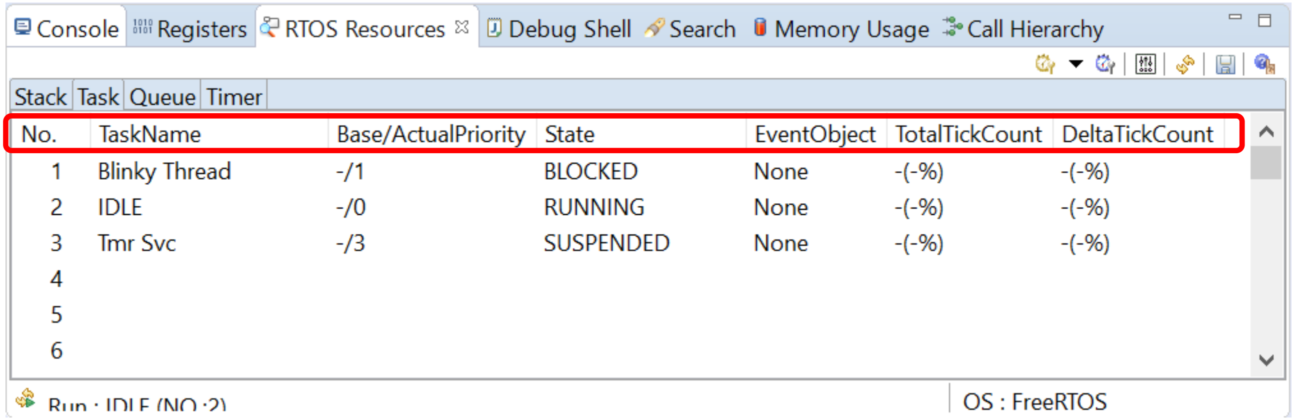
Run : MAIN_TASK (NO :5) OS : FreeRTOS

Figure 2-7. Save File button

3. Using RTOS Resources view with FreeRTOS

3.1 Task tab

The **Task** tab lists all tasks that existed in the program with the following information:



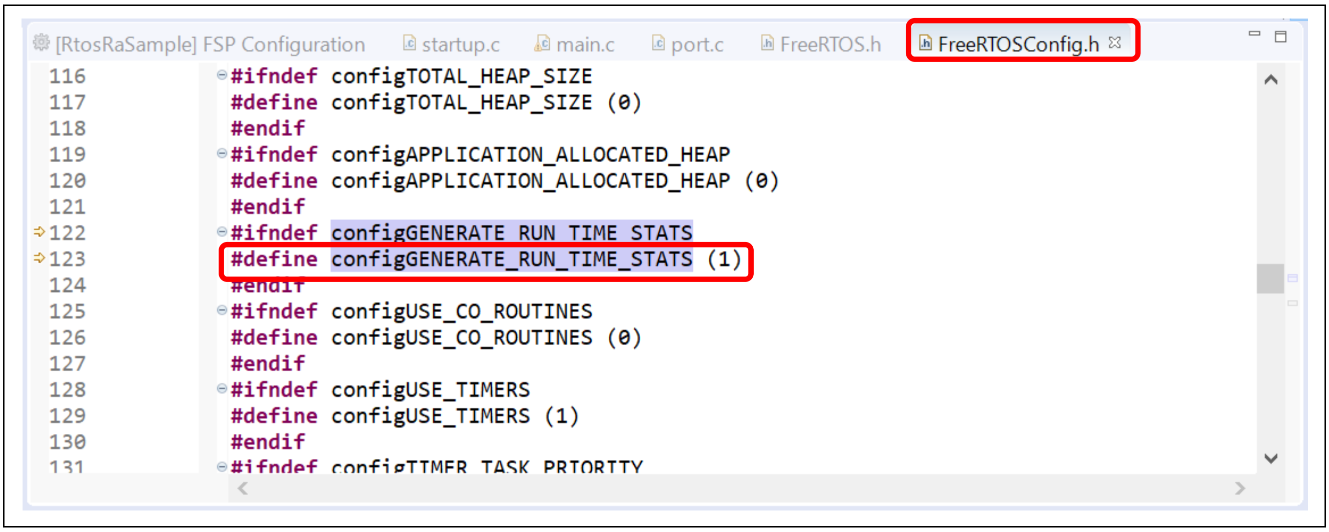
No.	TaskName	Base/ActualPriority	State	EventObject	TotalTickCount	DeltaTickCount
1	Blinky Thread	-/1	BLOCKED	None	-(-%)	-(-%)
2	IDLE	-/0	RUNNING	None	-(-%)	-(-%)
3	Tmr Svc	-/3	SUSPENDED	None	-(-%)	-(-%)
4						
5						
6						

Run : IDLE (NO :2) OS : FreeRTOS

Figure 3-1. Task tab

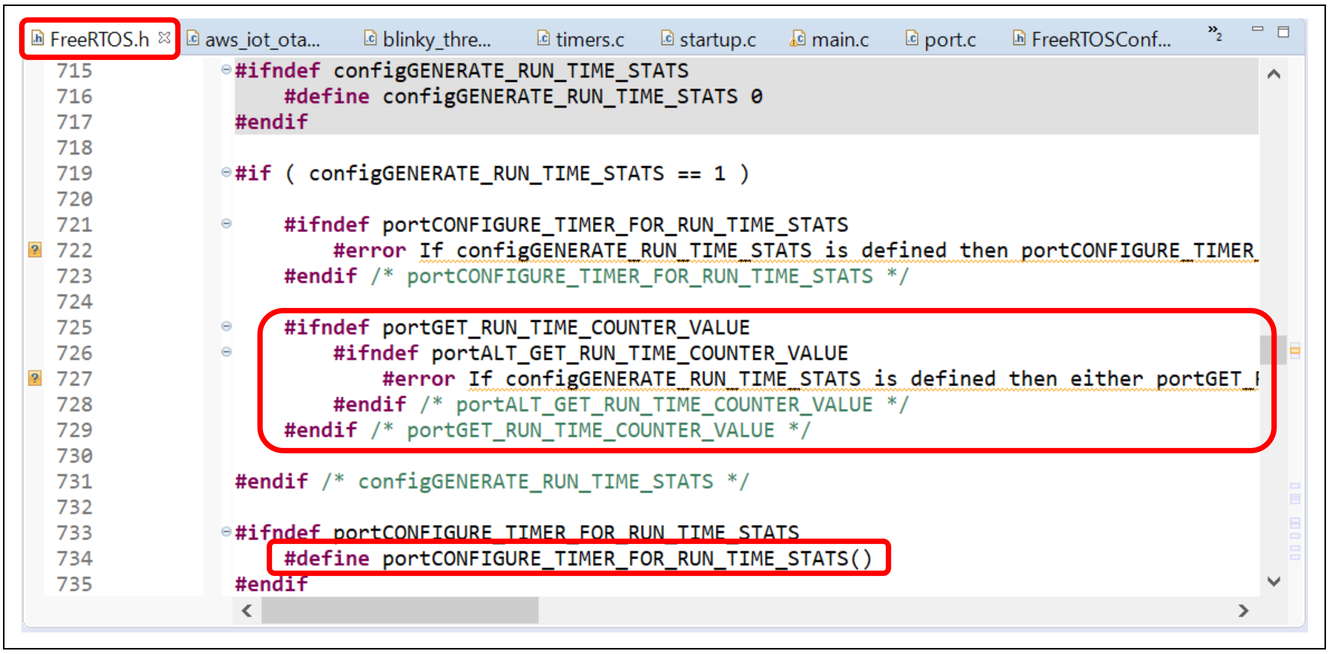
- **No.**: Row index.
- **TaskName**: The name assigned to the task upon creation.
- **Base/ActualPriority**: The base priority used by the priority inheritance mechanism/The actual priority used by the task.
- **State**: State of the task which includes RUNNING, READY, BLOCKED and SUSPENDED.
- **EventObject**: 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` to 1 (in `<project>/ra_cfg/aws/FreeRTOSConfig.h`), and implement the macros `portCONFIGURE_TIMER_FOR_RUN_TIME_STATS()` and `portGET_RUN_TIME_COUNTER_VALUE()` (in `<project>/ra/aws/amazon-freertos/freertos_kernel/include/FreeRTOS.h`). To configure these parameters, please refer to FreeRTOS guidelines at: https://www.freertos.org/rtos-run-time-stats.html.



```
#ifndef configTOTAL_HEAP_SIZE
#define configTOTAL_HEAP_SIZE (0)
#endif
#ifndef configAPPLICATION_ALLOCATED_HEAP
#define configAPPLICATION_ALLOCATED_HEAP (0)
#endif
#ifndef configGENERATE_RUN_TIME_STATS
#define configGENERATE_RUN_TIME_STATS (1)
#endif
#ifndef configUSE_CO_ROUTINES
#define configUSE_CO_ROUTINES (0)
#endif
#ifndef configUSE_TIMERS
#define configUSE_TIMERS (1)
#endif
#ifndef configTIMER_TASK_PRIORITY
```

Figure 3-2. Define configGENERATE_RUN_TIME_STATS in FreeRTOSConfig.h



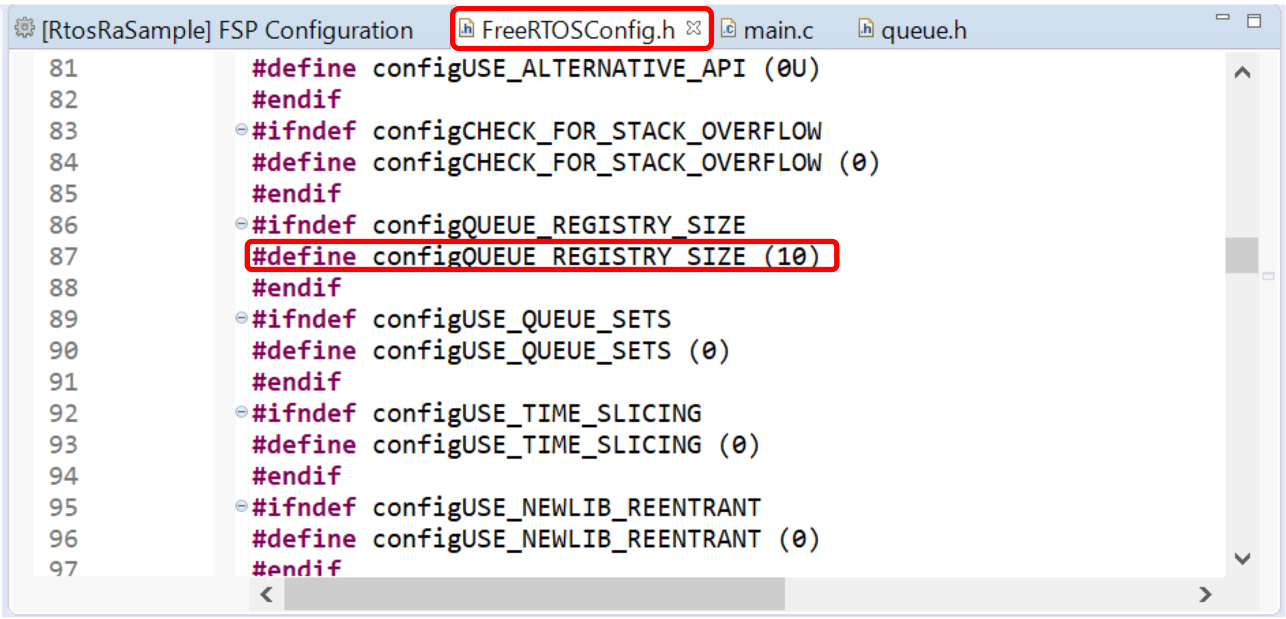
```
#ifndef configGENERATE_RUN_TIME_STATS
    #define configGENERATE_RUN_TIME_STATS 0
#endif

#if ( configGENERATE_RUN_TIME_STATS == 1 )

    #ifndef portCONFIGURE_TIMER_FOR_RUN_TIME_STATS
        #error If configGENERATE_RUN_TIME_STATS is defined then portCONFIGURE_TIMER_
    #endif /* portCONFIGURE_TIMER_FOR_RUN_TIME_STATS */

    #ifndef portGET_RUN_TIME_COUNTER_VALUE
        #ifndef portALT_GET_RUN_TIME_COUNTER_VALUE
            #error If configGENERATE_RUN_TIME_STATS is defined then either portGET_
        #endif /* portALT_GET_RUN_TIME_COUNTER_VALUE */
    #endif /* portGET_RUN_TIME_COUNTER_VALUE */

#endif /* configGENERATE_RUN_TIME_STATS */

#ifndef portCONFIGURE_TIMER_FOR_RUN_TIME_STATS
    #define portCONFIGURE_TIMER_FOR_RUN_TIME_STATS()
#endif
```

Figure 3-3. Configure portCONFIGURE_TIMER_FOR_RUN_TIME_STATS() and portGET_RUN_TIME_COUNTER_VALUE() in FreeRTOS.h

3.2 Queue tab

The **Queue** tab lists all queues/semaphores/mutexes used in the program.

Note: To display queue information, specify `configQUEUE_REGISTRY_SIZE` with value greater than 0 (in `<project>/ra_cfg/aws/FreeRTOSConfig.h`), and call the function `vQueueAddToRegistry()`. Note that this function call is already implemented in the demo code.

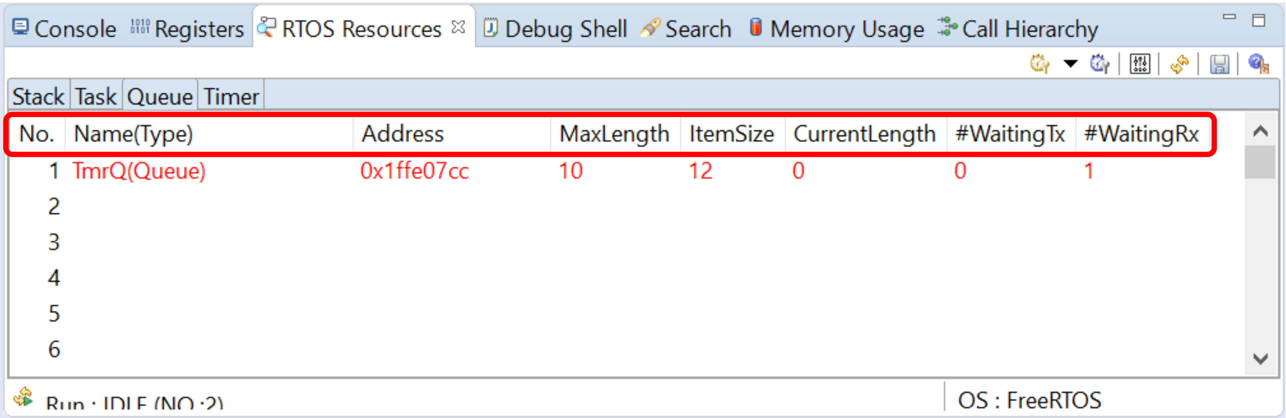


```
#define configUSE_ALTERNATIVE_API (0U)
#endif
#ifndef configCHECK_FOR_STACK_OVERFLOW
#define configCHECK_FOR_STACK_OVERFLOW (0)
#endif
#ifndef configQUEUE_REGISTRY_SIZE
#define configQUEUE_REGISTRY_SIZE (10)
#endif
#ifndef configUSE_QUEUE_SETS
#define configUSE_QUEUE_SETS (0)
#endif
#ifndef configUSE_TIME_SLICING
#define configUSE_TIME_SLICING (0)
#endif
#ifndef configUSE_NEWLIB_REENTRANT
#define configUSE_NEWLIB_REENTRANT (0)
#endif
```

Figure 3-4. Define configQUEUE_REGISTRY_SIZE in FreeRTOSConfig.h

This 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	TmrQ(Queue)	0x1ffe07cc	10	12	0	0	1
2							
3							
4							
5							
6							

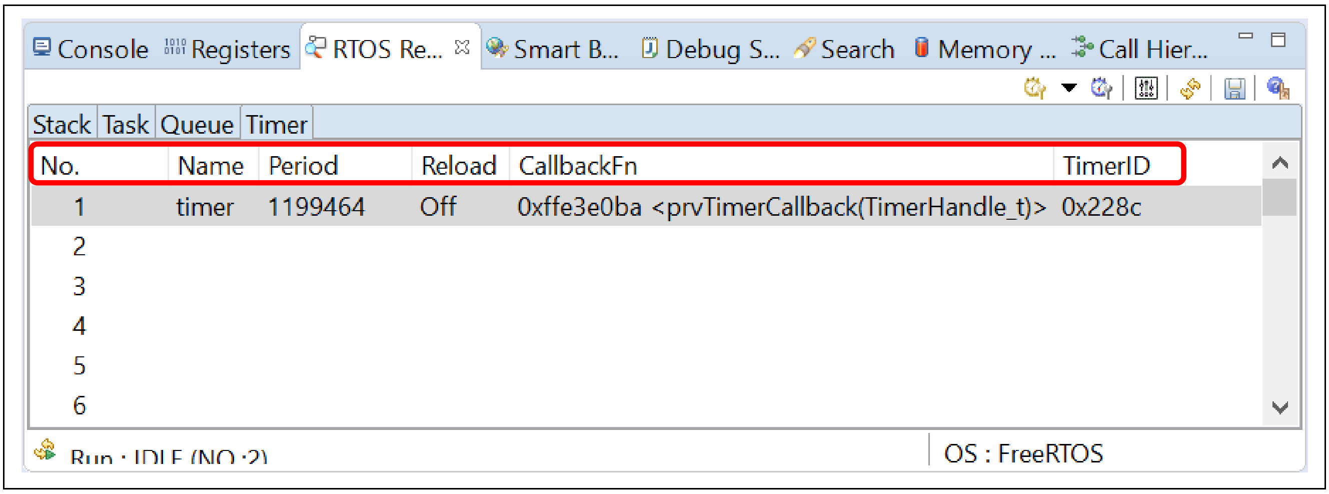
Run : IDLE (NO :2)

OS : FreeRTOS

Figure 3-5. Queue tab

3.3 Timer tab

The **Timer** tab lists all timers that existed 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					

Run : IDLE (NO :2)

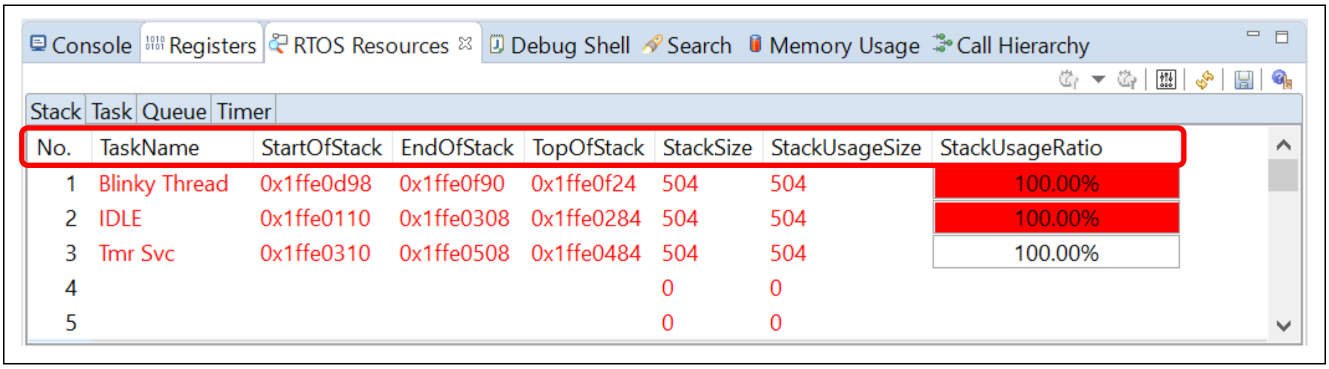
OS : FreeRTOS

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

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



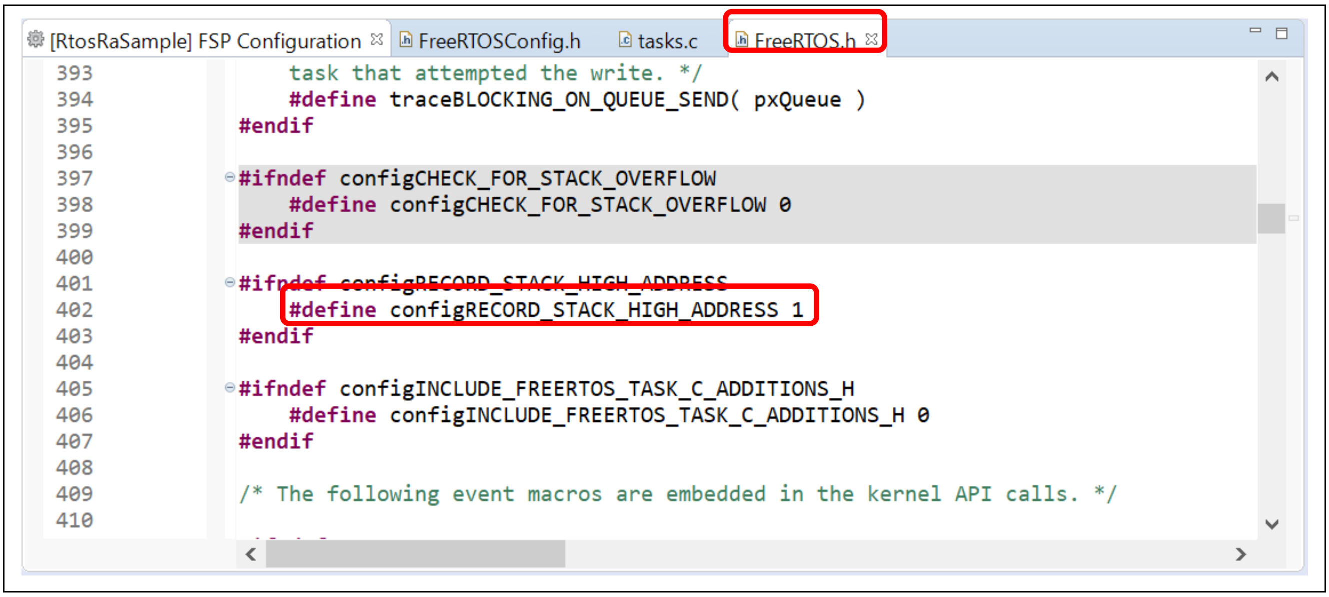
No.	TaskName	StartOfStack	EndOfStack	TopOfStack	StackSize	StackUsageSize	StackUsageRatio
1	Blinky Thread	0x1ffe0d98	0x1ffe0f90	0x1ffe0f24	504	504	100.00%
2	IDLE	0x1ffe0110	0x1ffe0308	0x1ffe0284	504	504	100.00%
3	Tmr Svc	0x1ffe0310	0x1ffe0508	0x1ffe0484	504	504	100.00%
4					0	0	
5					0	0	

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:

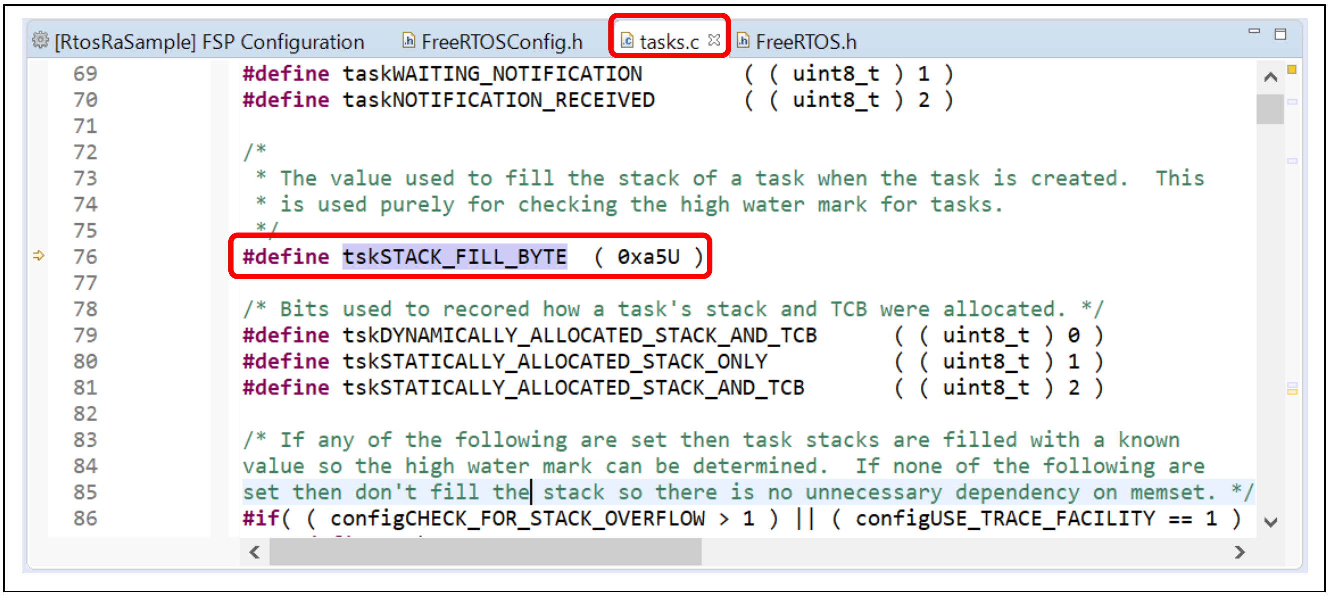
- To display **EndOfStack** and **StackSize**, define `configRECORD_STACK_HIGH_ADDRESS` as 1 in the `<project>/ra/aws/amazon-freertos/freertos_kernel/include/FreeRTOS.h` file.



```
    task that attempted the write. */
    #define traceBLOCKING_ON_QUEUE_SEND( pxQueue )
#endif

#ifndef configCHECK_FOR_STACK_OVERFLOW
    #define configCHECK_FOR_STACK_OVERFLOW 0
#endif

#ifndef configRECORD_STACK_HIGH_ADDRESS
    #define configRECORD_STACK_HIGH_ADDRESS 1
#endif

#ifndef configINCLUDE_FREERTOS_TASK_C_ADDITIONS_H
    #define configINCLUDE_FREERTOS_TASK_C_ADDITIONS_H 0
#endif

/* The following event macros are embedded in the kernel API calls. */
```

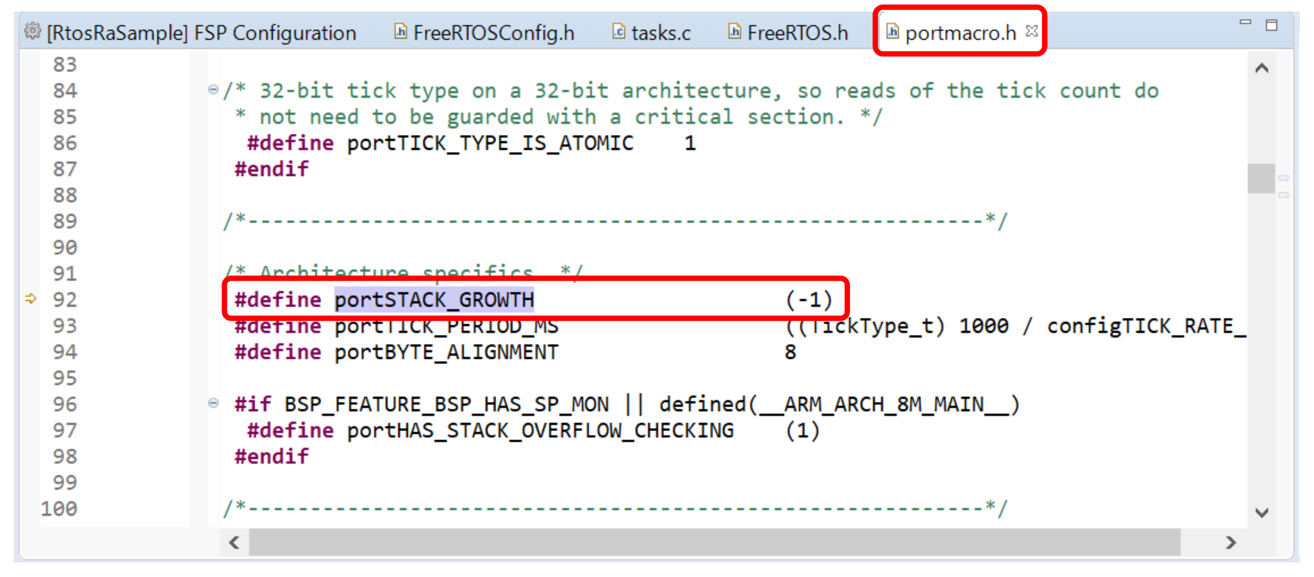
Figure 3-8. Define configRECORD_STACK_HIGH_ADDRESS in FreeRTOS.h

- To display **StackUsageSize** and **StackUsageRatio**, define `configRECORD_STACK_HIGH_ADDRESS` as 1 in the `<project>/ra/aws/amazon-freertos/freertos_kernel/include/FreeRTOS.h` file, and `tskSTACK_FILL_BYTE` to 0xA5U in the `<project>/ra/aws/amazon-freertos/freertos_kernel/task.c` file.
  Only devices with `portSTACK_GROWTH` defined as -1 are supported (in `<project>/ra/fsp/src/rm_freertos_port/portmacro.h`).



```
#define taskWAITING_NOTIFICATION        ( ( uint8_t ) 1 )
#define taskNOTIFICATION_RECEIVED       ( ( uint8_t ) 2 )

/*
 * The value used to fill the stack of a task when the task is created.  This
 * is used purely for checking the high water mark for tasks.
 */
#define tskSTACK_FILL_BYTE   ( 0xa5U )

/* Bits used to recored how a task's stack and TCB were allocated. */
#define tskDYNAMICALLY_ALLOCATED_STACK_AND_TCB      ( ( uint8_t ) 0 )
#define tskSTATICALLY_ALLOCATED_STACK_ONLY          ( ( uint8_t ) 1 )
#define tskSTATICALLY_ALLOCATED_STACK_AND_TCB       ( ( uint8_t ) 2 )

/* If any of the following are set then task stacks are filled with a known
value so the high water mark can be determined.  If none of the following are
set then don't fill the stack so there is no unnecessary dependency on memset. */
#if( ( configCHECK_FOR_STACK_OVERFLOW > 1 ) || ( configUSE_TRACE_FACILITY == 1 )
```

Figure 3-9. Define tskSTACK_FILL_BYTE in task.c



```
/* 32-bit tick type on a 32-bit architecture, so reads of the tick count do
 * not need to be guarded with a critical section. */
  #define portTICK_TYPE_IS_ATOMIC    1
 #endif

/*-----------------------------------------------------------*/

/* Architecture specifics. */
 #define portSTACK_GROWTH                    (-1)
 #define portTICK_PERIOD_MS                  ((TickType_t) 1000 / configTICK_RATE_
 #define portBYTE_ALIGNMENT                  8

 #if BSP_FEATURE_BSP_HAS_SP_MON || defined(__ARM_ARCH_8M_MAIN__)
  #define portHAS_STACK_OVERFLOW_CHECKING    (1)
 #endif

/*-----------------------------------------------------------*/
```

Figure 3-10.   Define portSTACK_GROWTH in portmacro.h

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Dec.24.2020		First creation

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

Corporate Headquarters

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

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/.

Trademarks

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