APPLICATION NOTE

RX Family

Renesas FreeRTOS

R01AN4307EJ0104
Rev.1.04
Aug 31, 2020

Introduction

This application note describes the Renesas FreeRTOS module which is based on FreeRTOS

Target Devices

- RX72M Group
- RX72N Group
- RX72T Group
- RX71M Group
- RX66T Group
- RX66N Group
- RX65N Group
- RX64M Group
- RX23W Group
- RX231 Group
- RX230 Group
- RX130 Group

Related Documents

- RX Family Board Support Package Module Using Firmware Integration Technology (R01AN1685)
- RX Family CMT Module Using Firmware Integration Technology (R01AN1856)
- Renesas e^2 studio Smart Configurator User Guide (R20AN0451)

References

- FreeRTOS customization: https://www.freertos.org/a00110.html [1]
- FreeRTOS Memory Management: https://www.freertos.org/a00111.html [2]

Contents

1. Overview ... 3
1.1 Renesas FreeRTOS Module ... 3
1.2 Creating a RTOS project with Renesas FreeRTOS ... 3
1.2.1 Creating CCRX Project ... 3
1.2.2 Creating GCC Project ... 4
1.3 BSP setting and Timer Source ... 5
1.3.1 BSP Configuration ... 5
1.3.2 Using CMT module ... 5

2. Requirements ... 6
2.1 Hardware Requirements ... 6
2.2 Software Requirements ... 6
2.3 Supported Toolchain ... 6

3. Configuration Overview ... 7
3.1 Heap Memory Management ... 7
3.1.1 heap_1.c ... 7
3.1.2 heap_2.c ... 7
3.1.3 heap_3.c ... 7
3.1.4 heap_4.c ... 7
3.2 Custom Configuration ... 8
3.3 Heap estimation ... 10
3.4 Add/Remove Objects ... 11

4. User Start Code ... 12
4.1 freertos_start.c, freertos_start.h ... 13
4.1.1 RTOS System Timer Initialization – void vApplicationSetupTimerInterrupt(void) ... 13
4.1.2 Idle Hook Function – void vApplicationIdleHook(void) ... 13
4.1.3 Tick Hook Function – void vApplicationTickHook(void) ... 13
4.1.4 Malloc Failed Hook Function – void vApplicationMallocFailedHook(void) ... 13
4.1.5 void Processing_Before_Start_Kernel(void) ... 13
4.2 void main_task(void *pvParameters) ... 13

5. Appendices ... 14
5.1 Confirmed Operation Environment ... 14
5.2 Notes ... 15
5.2.1 The macro configTOTAL_HEAP_SIZE_N ... 15
5.2.2 e[2]studio version for this application note ... 15

1. Overview

1.1 Renesas FreeRTOS Module

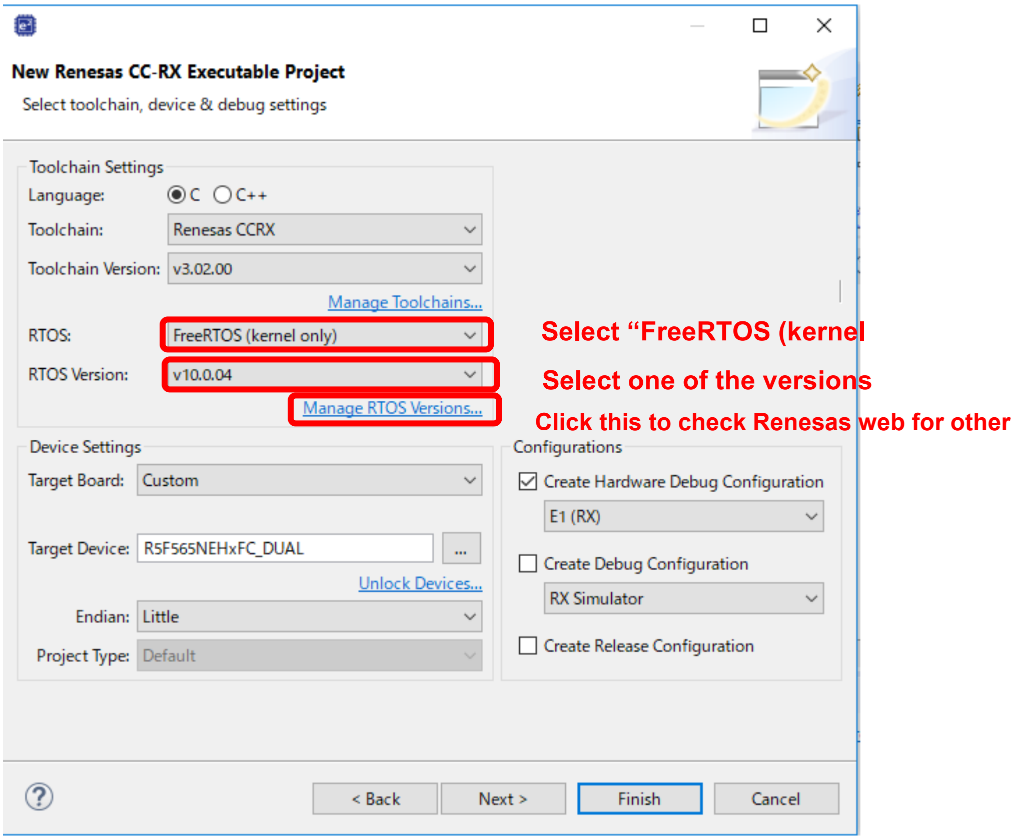
The Renesas FreeRTOS module is FreeRTOS Kernel for RX family.

1.2 Creating a RTOS project with Renesas FreeRTOS

1.2.1 Creating CCRX Project

It is recommended to create a Renesas FreeRTOS project using e2studio, which supports CCRX RTOS project creation. At the start of project creation, user would be able to choose the version of Renesas FreeRTOS package, and the selected version will be imported automatically into the project. This makes it easier for the user, so that they can focus only on FreeRTOS configuration, and writing application code.

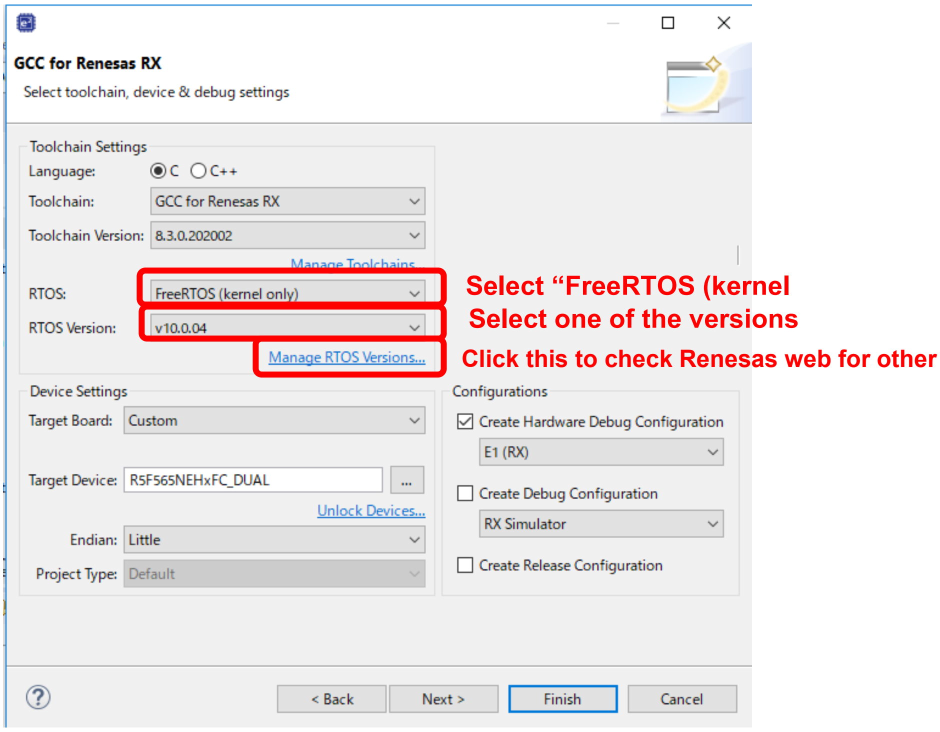
The figure below shows how to select RTOS for CCRX project creation:



1.2.2 Creating GCC Project

GCC RTOS project creation is supported. At the start of project creation, user would be able to choose the version of Renesas FreeRTOS package, and the selected version will be imported automatically into the project. This makes it easier for the user, so that they can focus only on FreeRTOS configuration, and writing application code.

The figure below shows how to select RTOS for GCC project creation:



1.3 BSP setting and Timer Source

1.3.1 BSP Configuration

When using the method mention in Section 1.3, the following BSP configuration will be set automatically:

- BSP_CFG_RTOS_USED is set to ‘1’
- BSP_CFG_RTOS_SYSTEM_TIMER is set to ‘0’ (this is CMT channel 0)

By default, the RTOS kernel is configured to use CMT channel 0 as system timer. In future version of e2studio, user would be able to choose anyone of the available CMT channel

1.3.2 Using CMT module

Renesas FreeRTOS kernel requires the exclusive use of the selected CMT channel. If user would like to use CMT module in the project for purpose other than the kernel, care must be taken not to interfere with the operation of the CMT channel used by the kernel. It is therefore recommended that in such cases, a CMT FIT module which supports RTOS, be used. At this moment the version of CMT FIT which supports RTOS is v3.30

2. Requirements

This FIT module has been confirmed to operate under the following conditions.

2.1 Hardware Requirements

The MCU used must support the following functions:

- CMT

2.2 Software Requirements

This Renesas FreeRTOS module is dependent upon the following FIT module:

- Renesas Board Support Package (r_bsp)

2.3 Supported Toolchain

This driver has been confirmed to work with the toolchain listed in 5.1, Confirmed Operation Environment.

3. Configuration Overview

3.1 Heap Memory Management

3.1.1 heap_1.c

This is the simplest implementation of all. It does not permit memory to be freed once it has been allocated. Despite this, heap_1.c is appropriate for a large number of embedded applications. This is because many small and deeply embedded applications create all the tasks, queues, semaphores, etc. required when the system boots, and then use all of these objects for the lifetime of program (until the application is switched off again or is rebooted). Nothing ever gets deleted

For details, refer to [2]

3.1.2 heap_2.c

This scheme uses a best fit algorithm and, unlike scheme 1, allows previously allocated blocks to be freed. It does not combine adjacent free blocks into a single large block. See heap_4.c for an implementation that does coalescence free blocks. The total amount of available heap space is set by configTOTAL_HEAP_SIZE - which is defined in FreeRTOSConfig.h. The configAPPLICATION_ALLOCATED_HEAP FreeRTOSConfig.h configuration constant is provided to allow the heap to be placed at a specific address in memory.

For details, refer to [2]

3.1.3 heap_3.c

This implements a simple wrapper for the standard C library malloc() and free() functions that will, in most cases, be supplied with your chosen compiler. The wrapper simply makes the malloc() and free() functions thread safe

For details, refer to [2]

3.1.4 heap_4.c

This scheme uses a first fit algorithm and, unlike scheme 2, it does combine adjacent free memory blocks into a single large block (it does include a coalescence algorithm).
The total amount of available heap space is set by configTOTAL_HEAP_SIZE - which is defined in FreeRTOSConfig.h. The configAPPLICATION_ALLOCATED_HEAP FreeRTOSConfig.h configuration constant is provided to allow the heap to be placed at a specific address in memory

For details, refer to [2]

3.2 Custom Configuration

Renesas FreeRTOS is customized using a configuration file called FreeRTOSConfig.h. Every Renesas FreeRTOS application must have a FreeRTOSConfig.h header file in its pre-processor include path. FreeRTOSConfig.h tailors the RTOS kernel to the application being built. It is therefore specific to the application, not the RTOS, and should be located in an application directory, not in one of the RTOS kernel source code directories

Below are some typical FreeRTOSConfig.h definition. Note that some macro configuration has to refer to parameters in BSP; the RTOS kernel will not work as expected if wrong values are set:

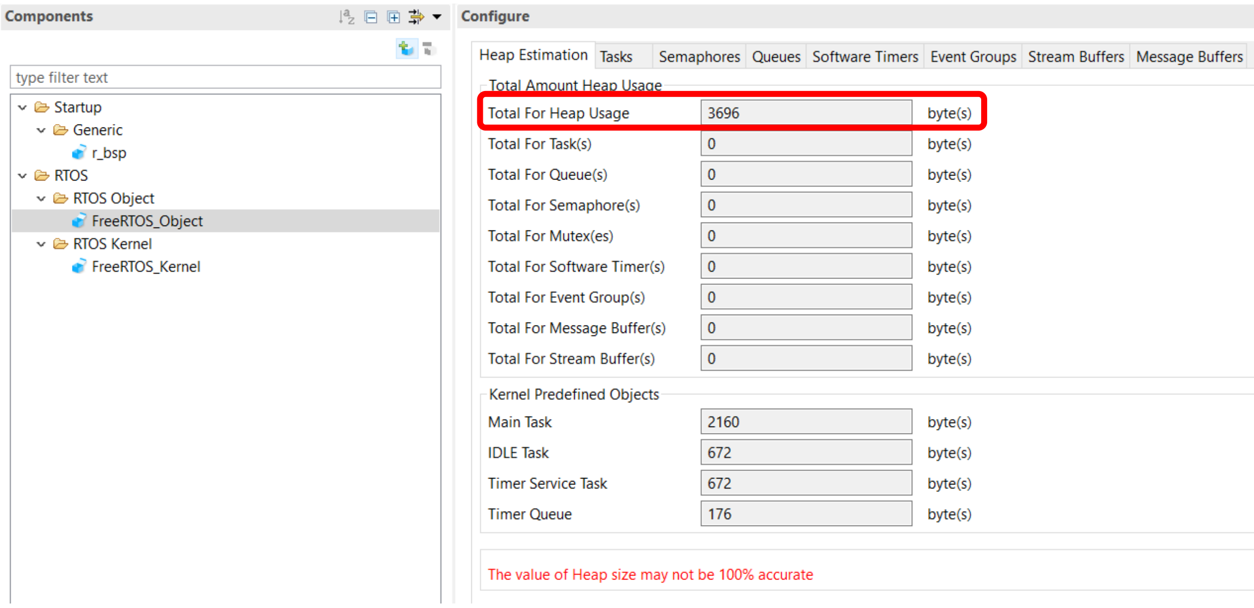
Configuration options in FreeRTOSConfig.h	
configUSE_PREEMPTION	Refer to [1]
configUSE_IDLE_HOOK	Refer to [1]
configUSE_TICK_HOOK	Refer to [1]
configCPU_CLOCK_HZ	This macro defines the MCU system clock speed in Hz. It is required in order to correctly configure timer peripherals. It should be set to **BSP_ICLK_HZ**, which is defined in BSP mcu_info.h
configPERIPHERAL_CLOCK_HZ	The frequency of the peripheral module clock, used by CMT. In the case of RX, it is PCLKB, therefore this macro should be set to **BSP_PCLKB_HZ**, which is defined in mcu_info.h
configTICK_RATE_HZ	The frequency of the RTOS kernel tick interrupt. The tick interrupt is used to measure time. Therefore a higher tick frequency means time can be measured to a higher resolution. However, a high tick frequency also means that the RTOS kernel will use more CPU time so it is less efficient. Typically this value is 1000
configMINIMAL_STACK_SIZE	Refer to [1]
configTOTAL_HEAP_SIZE_N	This macro replaces existing macro "configTOTAL_HEAP_SIZE" Refer to 3.3
configMAX_TASK_NAME_LEN	Refer to [1]
configUSE_TRACE_FACILITY	Refer to [1]
configUSE_16_BIT_TICKS	Refer to [1]
configIDLE_SHOULD_YIELD	Refer to [1]
configUSE_CO_ROUTINES	Refer to [1]
configUSE_MUTEXES	Refer to [1]
configGENERATE_RUN_TIME_STATS	Refer to [1]
configCHECK_FOR_STACK_OVERFLOW	Refer to [1]
configUSE_RECURSIVE_MUTEXES	Refer to [1]
configQUEUE_REGISTRY_SIZE	Refer to [1]
configUSE_MALLOC_FAILED_HOOK	Refer to [1]
configUSE_APPLICATION_TASK_TAG	Refer to [1]
configUSE_QUEUE_SETS	Refer to [1]
configUSE_COUNTING_SEMAPHORES	Refer to [1]
configMAX_PRIORITIES	Refer to [1]
configMAX_CO_ROUTINE_PRIORITIES	Refer to [1]
configUSE_TIMERS	Refer to [1]
configTIMER_TASK_PRIORITY	Refer to [1]
configTIMER_QUEUE_LENGTH	Refer to [1]
configTIMER_TASK_STACK_DEPTH	Refer to [1]
configKERNEL_INTERRUPT_PRIORITY	Refer to [1]
configMAX_SYSCALL_INTERRUPT_PRIORITY	Refer to [1]
configTICK_VECTOR	This macro should be set to the vector number of the CMT

	channel used. The possible values are: _CMT0_CMI0: If CMT0 is used _CMT1_CMI1: If CMT1 is used _CMT2_CMI2: If CMT2 is used _CMT3_CMI3: If CMT3 is used
`configUSE_TASK_NOTIFICATIONS`	Refer to [1]
`configRECORD_STACK_HIGH_ADDRESS`	Refer to [1]
`configNUM_THREAD_LOCAL_STORAGE_POINTERS`	Refer to [1]
`configSUPPORT_DYNAMIC_ALLOCATION`	Refer to [1]
`configSUPPORT_STATIC_ALLOCATION`	Refer to [1]

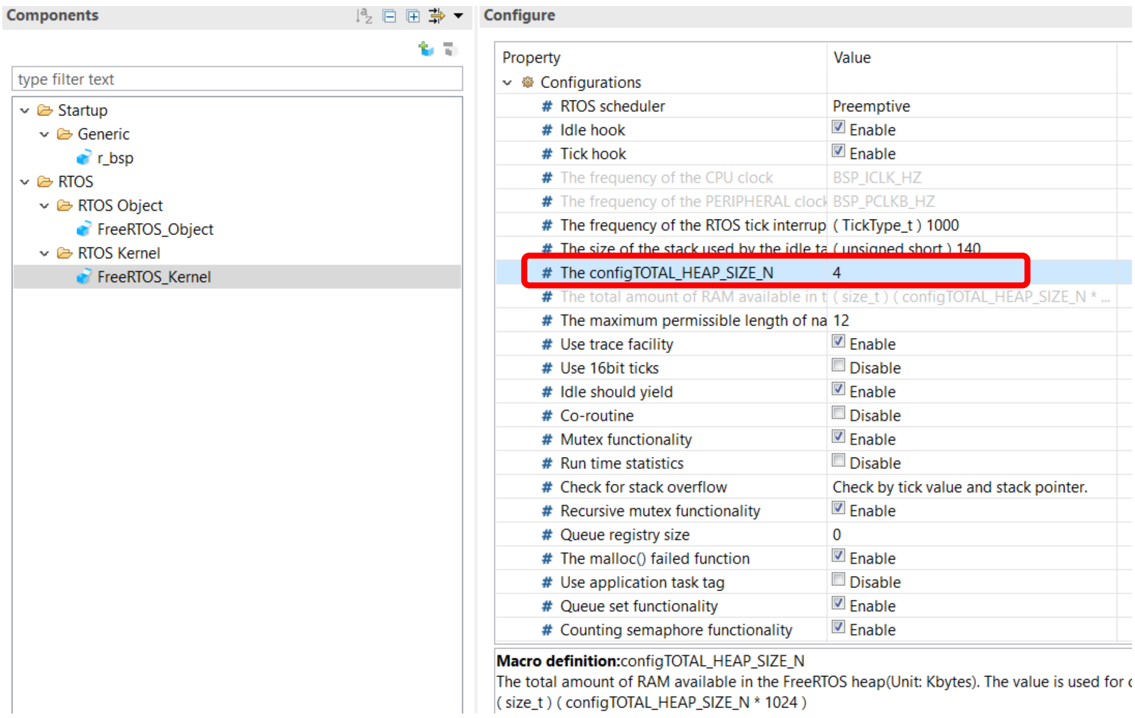
3.3 Heap estimation

From e2studio v7.7.0, the values of heap usage will be summarized and displayed in “Total Amount Heap Usage” group in Smart Configurator. **Total For Heap Usage** displays the estimated total RAM usage for all FreeRTOS objects. Total RAM usage changes when user adds or removes FreeRTOS objects. The purpose of this feature is to estimate the total remaining heap space to save RAM usage in implement application.

Below is what heap estimation looks like in Smart Configurator:



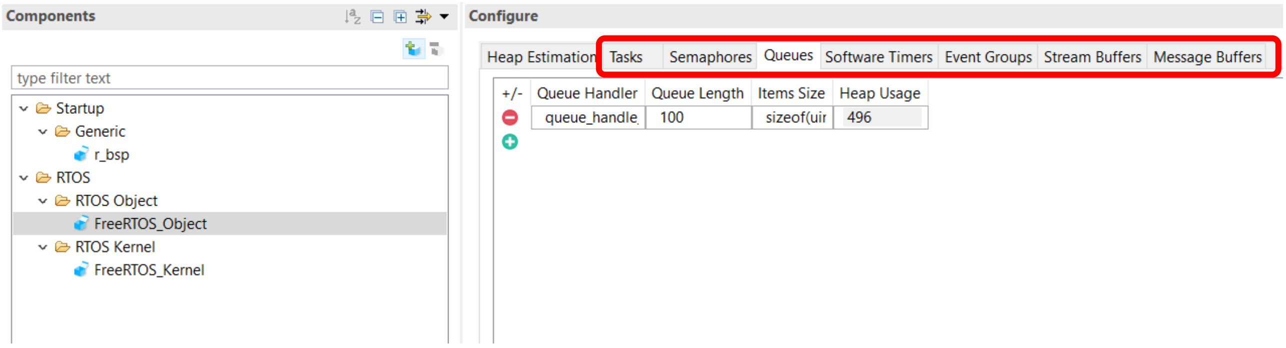
The macro “configTOTAL_HEAP_SIZE_N” in FreeRTOSConfig.h is used to set the total available heap space. This macro can be modified by changing “The configTOTAL_HEAP_SIZE_N” configuration in FreeRTOS_Kernel as shown in the following picture:



Total allocated RAM size is calculated with the following expression: (configTOTAL_HEAP_SIZE_N*1024).

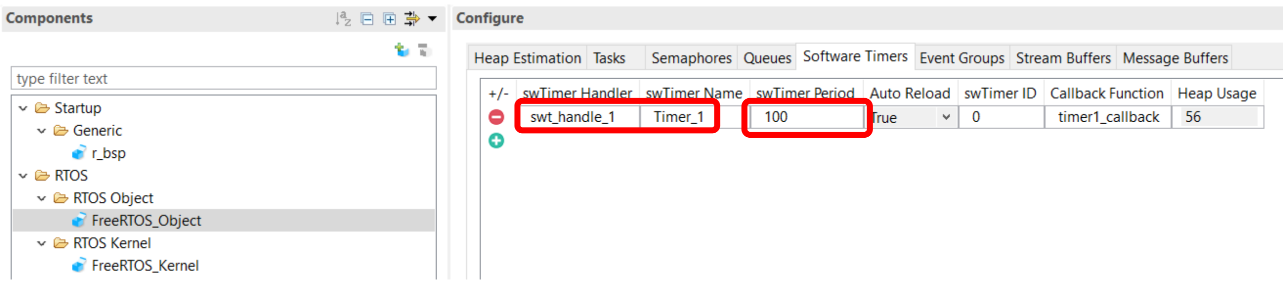
3.4 Add/Remove Objects

User can add or remove FreeRTOS objects within the “FreeRTOS_Object” component in Smart Configurator as shown in the following picture:



User may choose from any of the available objects to modify them. Objects can be added or removed with the (add) or (remove) button.

User can also modify certain parameters of the object by replacing the default value as shown in the following picture:



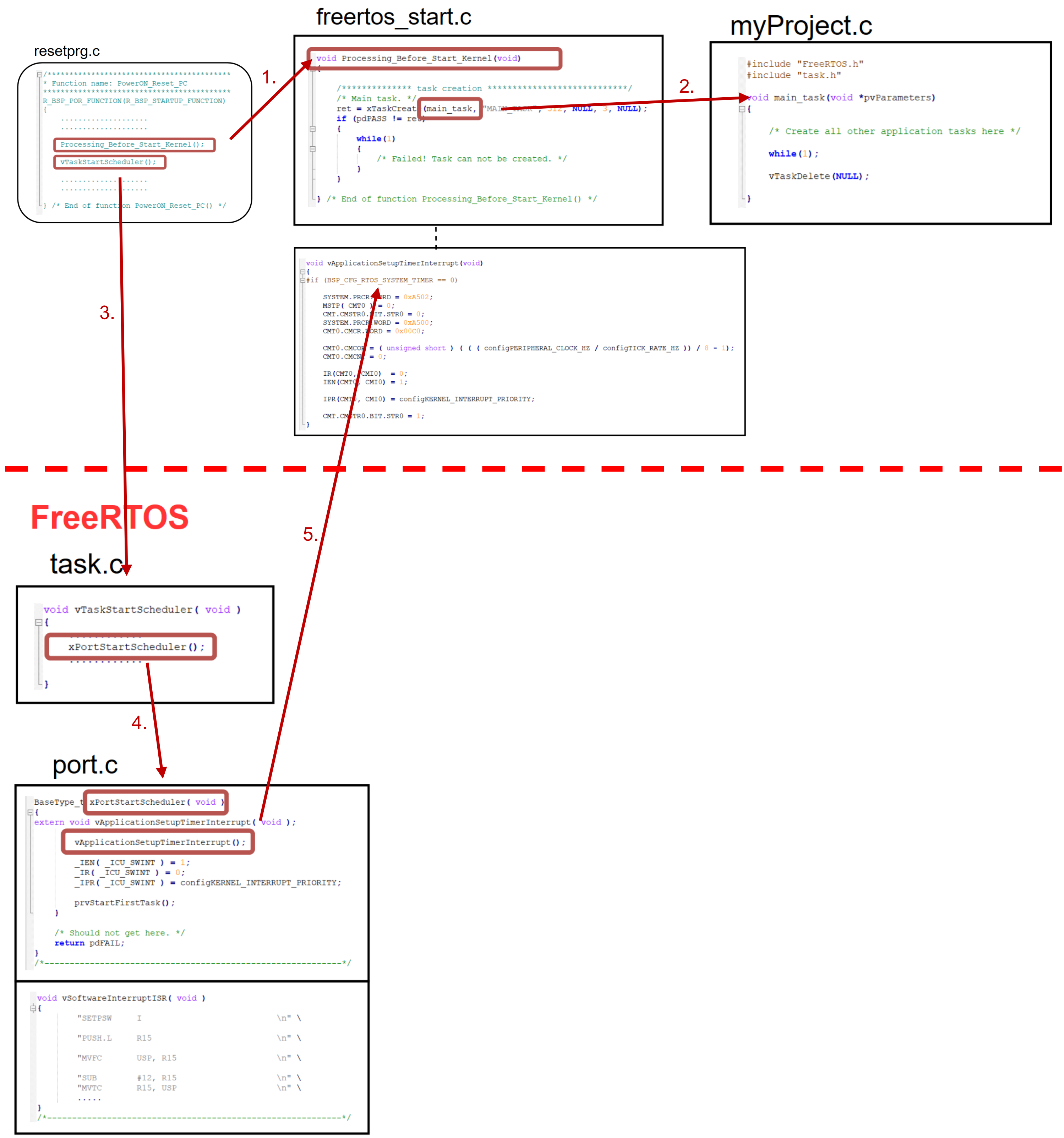
Modification of objects will be reflected after code generation in the following .c file.

Heap usage of modified objects will be displayed in “Total Amount Heap Usage” group after code generation. Object heap usage exceeding the total available heap space will result in build error.

More details are mentioned in [e2studio]→ [Help].

4. User Start Code

The following figure shows the program flow of the Renesas FreeRTOS project, starting from BSP routines, startup of FreeRTOS kernel, and running of user's application code:



4.1 freertos_start.c, freertos_start.h

Before the RTOS kernel starts, user startup functions and CMT configuration routine can be run. The functions are provided below:

4.1.1 RTOS System Timer Initialization – void vApplicationSetupTimerInterrupt(void)

This function is provided and it readily configures the selected CMT channel (#define BSP_CFG_RTOS_SYSTEM_TIMER) for use as RTOS kernel system tick. User can use this function readily without any modification

4.1.2 Idle Hook Function – void vApplicationIdleHook(void)

The idle task is created automatically when the RTOS scheduler is started to ensure there is always at least one task that is able to run. It is created at the lowest possible priority to ensure it does not use any CPU time if there are higher priority application tasks in the ready state. The idle task can optionally call an application defined hook function, which is the idle hook function. The idle task runs at the very lowest priority, so such idle hook function will only get executed when there are no tasks of higher priority that are able to run

The idle hook function will only get called if configUSE_IDLE_HOOK is set to 1 within FreeRTOSConfig.h

4.1.3 Tick Hook Function – void vApplicationTickHook(void)

The tick interrupt can optionally call an application defined hook function, which is the tick hook function. The tick hook provides a convenient place to implement timer functionality

The tick hook will only get called if configUSE_TICK_HOOK is set to 1 within FreeRTOSConfig.h

4.1.4 Malloc Failed Hook Function – void vApplicationMallocFailedHook(void)

The memory allocation schemes implemented by heap_1.c, heap_2.c, heap_3.c, heap_4.c and heap_5.c can optionally include a malloc() failure hook function that can be configured to get called if pvPortMalloc() ever returns NULL

Defining the malloc() failure hook will help identify problems caused by lack of heap memory especially when a call to pvPortMalloc() fails within an API function

The malloc failed hook will only get called if configUSE_MALLOC_FAILED_HOOK is set to 1 within FreeRTOSConfig.h

4.1.5 void Processing_Before_Start_Kernel(void)

In this function, a main task "main_task()" is created. User can also create FreeRTOS objects (mailbox, semaphore, mutex) if required

4.2 void main_task(void *pvParameters)

This task is created in Processing_Before_Start_Kernel(). User should create all other application tasks within this function

5. Appendices

5.1 Confirmed Operation Environment

This section describes confirmed operation environment for the Renesas FreeRTOS module.

Table 5.1 Confirmed Operation Environment

Item	Contents
Integrated development environment	Renesas Electronics e² studio Version 7.8
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.02 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 4.8.4.201902 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99 Linker option: The following user defined option should be added to the default settings of the integrated development environment, if “Optimize size (-Os)” is used: -Wl,--no-gc-sections This is to work around a GCC linker issue whereby the linker erroneously discard interrupt functions declared in FIT peripheral module
Endian	Big endian/little endian
Revision of the module	Renesas FreeRTOS Version 10.0.03
Board used	Renesas Starter Kit+ for RX72N (product No.: RTK5572NNxSxxxxxBE) Target Board for RX23W (product No.: RTK5RX23W0CxxxxxBJ) Renesas Starter Kit+ for RX130 512KB (product No.: RTK5051308CxxxxxBR) Renesas Starter Kit+ for RX72M (product No.: RTK5572MNDCxxxxxBJ) Renesas Starter Kit+ for RX72T (product No.: RTK5572TKCCxxxxxSE) Renesas Starter Kit+ for RX71M (product No.: R0K50571MSxxxBE) Renesas Starter Kit+ for RX66T (product No.: RTK50566T0CxxxxxBE) Renesas Starter Kit+ for RX65N-2M (product No.: RTK50565N2SxxxxxBE) Renesas Starter Kit+ for RX64M (product No.: R0K50564MSxxxBE) Renesas Starter Kit+ for RX231 (product No.: R0K505231CxxxBE)

5.2 Notes

5.2.1 The macro configTOTAL_HEAP_SIZE_N

- The macro configTOTAL_HEAP_SIZE_N is the value of KB, default value = 4.
  (Default heap size 4*1024 = 4096)

5.2.2 e²studio version for this application note

This application note can be used with e2stduio V.7.8 or e2studio 2020-04 or later versions.

Website and Support
Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jan. 21, 2019	—	First edition issued
1.01	Oct. 08. 2019	—	Added supporting device. RX130 Group, RX230 Group, RX231 Group, RX66T Group, RX72T Group, RX72M Group
1.02	Jan. 30, 2020	—	Added supporting device. RX23W Group
1.03	Apr. 30, 2020	—	Replaced macro “configTOTAL_HEAP_SIZE” with “configTOTAL_HEAP_SIZE_N”. Added support for GCC project creation with Smart Configurator.
1.04	Aug. 31, 2020	P.14	Fixed Table 5.1 Confirmed Operation Environment
		P.15	Added 5.2.2

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.

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/