RX Family

WDT Module Using Firmware Integration Technology

Introduction

This application note describes the Watch Dog Timer (WDT) module which uses Firmware Integration Technology (FIT). This module uses WDT to control counting operation of the WDT peripheral. In this document, this module is referred to as the WDT FIT module.

Target Devices

- RX230, RX231 Groups
- RX23W Group
- RX26T Group
- RX64M Group
- RX65N, RX651 Group
- RX66T Group
- RX66N Group
- RX660 Group
- RX671 Group
- RX71M Group
- RX72T Group
- RX72M Group
- RX72N Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

For details of the confirmed operation contents of each compiler, refer to “6.1 Confirmed Operation Environment".

Contents

1. Overview

1.1 WDT FIT Module

The WDT FIT module can be used by being implemented in a project as an API. See section 2.13, Adding the FIT Module to Your Project for details on methods to implement this FIT module into a project.

1.2 Overview of the WDT FIT Module

This WDT FIT driver supports WDT peripherals. WDT details are described in the Hardware User's Manual. This driver supports both Auto-Start and Register-Start modes. By selecting Auto-Start mode via a compile-time equate, the R_WDT_Open() code is removed from the build. With Auto-Start mode, the WDT down-counter is started automatically after a reset. With Register-Start mode, the WDT down-counter is started after a call to R_WDT_Open() and an R_WDT_Control() refresh operation.

The R_WDT_Control() refresh command must be made periodically to refresh the WDT counter. If no call is made, the WDT counter will underflow, and the reset signal or non-maskable interrupt (NMI) signal will be output.

If the NMI signal is selected, an interrupt handler must be created and registered to handle this interrupt. When using Auto-Start mode, the application must also enable the underflow/refresh error interrupt in the Interrupt Controller Unit (ICU).

This is handled by the R_WDT_Open() function in Register-Start mode.

1.3 Using the FIT WDT module

1.3.1 Using FIT WDT module in C++ project

For C++ project, add FIT WDT module interface header file within extern “C”{}:

```
Extern “C”
{
  #include “r_smc_entry.h”
  #include “r_wdt_rx_if.h”
}
```

1.4 API Overview

Table 1.1 lists the API functions included in this module.

Table 1.1 API Functions

Function	Contents
R_WDT_Open()	This function initializes the WDT FIT module. This function must be executed before other API functions. This Open function is not used when WDT Auto-Start mode is enabled in the OFS0 register in r_bsp_config.h.
R_WDT_Control()	This function gets WDT status (underflow error status, refresh error status, and WDT counter value) and refreshes the WDT down-counter.
R_WDT_GetVersion()	This function returns the driver version number.

1.5 Limitations

The WDT FIT module does not support maskable interrupts.

2. API Information

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

2.1 Hardware Requirements

The MCU used must support the following functions:

- WDTA

2.2 Software Requirements

This driver is dependent upon the following FIT module:

- Renesas Board Support Package (r_bsp) v5.20 or higher

2.3 Limitations

2.3.1 RAM Location Limitations

In FIT, if a value equivalent to NULL is set as the pointer argument of an API function, error might be returned due to parameter check. Therefore, do not pass a NULL equivalent value as pointer argument to an API function.

The NULL value is defined as 0 because of the library function specifications. Therefore, the above phenomenon would occur when the variable or function passed to the API function pointer argument is located at the start address of RAM (address 0x0). In this case, change the section settings or prepare a dummy variable at the top of the RAM so that the variable or function passed to the API function pointer argument is not located at address 0x0.

In the case of the CCRX project (e2 studio V7.5.0), the RAM start address is set as 0x4 to prevent the variable from being located at address 0x0. In the case of the GCC project (e2 studio V7.5.0) and IAR project (EWRX V4.12.1), the start address of RAM is 0x0, so the above measures are necessary.

The default settings of the section may be changed due to the IDE version upgrade. Please check the section settings when using the latest IDE.

2.4 Supported Toolchain

This driver has been confirmed to work with the toolchain listed in 6.1, Confirmed Operation Environment.

2.5 Interrupt Vector

The Non-maskable interrupt (NMI) is enabled by executing the R_WDT_Open function (with argument).

Table 2.1 lists the interrupt vector used in the WDT FIT Module.

Table 2.1 List of Usage of Exception Vector

Device	Contents
RX230	Non-maskable interrupt
RX231	
RX23W	
RX26T	
RX64M	
RX65N	
RX66T	
RX66N	
RX660	
RX671	
RX71M	
RX72T	
RX72M	
RX72N	

2.6 Header Files

All API calls and their supporting interface definitions are located in r_wdt_rx_if.h.

2.7 Integer Types

This project uses ANSI C99. These types are defined in stdint.h.

2.8 Configuration Overview

The configuration option settings of this module are located in r_wdt_rx_config.h. The option names and setting values are listed in the table below:

Configurable options in r_wdt_rx_config.h	
WDT_CFG_PARAM_CHECKING_ENABLE 1	1: Compile time parameter check processing is included in the code. 0: Compile time parameter check processing is excluded in the code. BSP_CFG_PARAM_CHECKING_ENABLE (default): Use this as the system default. Note: The size of code can be reduced by excluding the compile time parameter check.

Configuration options in r_bsp_config.h	
BSP_CFG_OFS0_REG_VALUE 0xFFFFFFFF	If this definition is set to 0xFFFFFFFF, the WDT is disabled at powerup and must be initialized using the R_WDT_Open() function. If configured to enable WDT auto-start, the R_WDT_Open() code is removed from the build, and the count automatically starts after a reset. See r_bsp_config.h for configuration options.

2.9 Code Size

Typical code sizes associated with this module are listed below.

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options described in 2.8, Configuration Overview. The table lists reference values when the C compiler's compile options are set to their default values, as described in 2.4, Supported Toolchain. The compile option default values are optimization level: 2, optimization type: for size, and data endianness: little-endian. The code size varies depending on the C compiler version and compile options.

ROM, RAM and Stack Code Sizes				
Device	**Category**	**Memory used**		**Remarks**
		Renesas Compiler		
		With Parameter Checking	**Without Parameter Checking**	
RX230	ROM	316 bytes	177 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function
RX231	ROM	316 bytes	177 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function
RX23W	ROM	316 bytes	178 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function
RX64M	ROM	316 bytes	177 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function
RX65N	ROM	316 bytes	177 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function
RX66T	ROM	316 bytes	177 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function
RX66N	ROM	316 bytes	178 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function
RX71M	ROM	316 bytes	177 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function
RX72T	ROM	316 bytes	177 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function

ROM, RAM and Stack Code Sizes				
Device	**Category**	**Memory used**		**Remarks**
		Renesas Compiler		
		With Parameter Checking	**Without Parameter Checking**	
RX72M	ROM	316 bytes	178 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control Function
RX72N	ROM	316 bytes	178 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	28 bytes		When using R_WDT_Control function
RX671	ROM	307 bytes	175 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	20 bytes		When using R_WDT_Control function
RX660	ROM	307 bytes	176 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	12		When using R_WDT_Control function
RX26T	ROM	307 bytes	176 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	12		When using R_WDT_Control function

ROM, RAM and Stack Code Sizes				
Device	Category	Memory used		Remarks
		GCC		
		With Parameter Checking	Without Parameter Checking	
RX230	ROM	600 bytes	328 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX231	ROM	600 bytes	328 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX64M	ROM	600 bytes	328 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX65N	ROM	600 bytes	328 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX66T	ROM	600 bytes	328 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX66N	ROM	632 bytes	336 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX71M	ROM	600 bytes	328 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX72T	ROM	600 bytes	328 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX72M	ROM	632 bytes	336 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function

ROM, RAM and Stack Code Sizes				
Device	Category	Memory used		Remarks
		GCC		
		With Parameter Checking	Without Parameter Checking	
RX72N	ROM	632 bytes	336 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX671	ROM	648 bytes	352 bytes	Register-Start mode
	RAM	4 bytes	4 bytes	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX660	ROM	640 bytes	352 bytes	Register-Start mode
	RAM	0 byte	0 byte	already_opened only
	Maximum stack usage	-		When using R_WDT_Control function
RX26T	ROM	376 bytes	216 bytes	Register-Start mode
	RAM	0 byte	0 byte	already_opened only
	Maximum stack usage	-	-	When using R_WDT_Control function

ROM, RAM and Stack Code Sizes				
Device	Category	Memory used		Remarks
		IAR Compiler		
		With Parameter Checking	Without Parameter Checking	
RX230	ROM	496 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	156 bytes		When using R_WDT_Control function
RX231	ROM	496 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	156 bytes		When using R_WDT_Control function
RX64M	ROM	496 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	156 bytes		When using R_WDT_Control function
RX65N	ROM	496 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	156		When using R_WDT_Control function
RX66T	ROM	496 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	156 bytes		When using R_WDT_Control function
RX66N	ROM	496 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	160 bytes		When using R_WDT_Control function
RX71M	ROM	496 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	156 bytes		When using R_WDT_Control function
RX72T	ROM	496 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	156 bytes		When using R_WDT_Control function
RX72M	ROM	496 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	160 bytes		When using R_WDT_Control function

ROM, RAM and Stack Code Sizes				
Device	**Category**	**Memory used**		**Remarks**
		IAR Compiler		
		With Parameter Checking	**Without Parameter Checking**	
RX671	ROM	465 bytes	245 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	172 bytes		When using R_WDT_Control function
RX660	ROM	492 bytes	292 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	156 bytes		When using R_WDT_Control function
RX26T	ROM	500 bytes	300 bytes	Register-Start mode
	RAM	1 byte	1 byte	already_opened only
	Maximum stack usage	96 bytes		When using R_WDT_Control function

2.10 Parameters

This section describes the parameter structure used by the API functions in this module. The structure is located in r_wdt_rx_if.h as are the prototype declarations of API functions.

2.11 Return Values

This section describes return values of API functions. This enumeration is located in r_wdt_rx_if.h as are the prototype declarations of API functions.

```
typedef enum e_wdt_err        // WDT API error codes
{
    WDT_SUCCESS=0,
    WDT_ERR_OPEN_IGNORED,     // The module has already been opened.
    WDT_ERR_INVALID_ARG,      // Argument is not valid.
    WDT_ERR_NULL_PTR,         // Received null pointer.
    WDT_ERR_NOT_OPENED,       // Open function has not been called.
    WDT_ERR_BUSY,             // WDT resource is locked.
} wdt_err_t;
```

2.12 Callback Function

None.

2.13 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) or (3) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) or (4) for RX devices that are not supported by the Smart Configurator.

(1) Adding the FIT module to your project using the Smart Configurator in e^2 studio
By using the Smart Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(2) Adding the FIT module to your project using the FIT Configurator in e^2 studio
By using the FIT Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using the Smart Configurator in CS+
By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “Renesas e^2 studio Smart Configurator User Guide (R20AN0451)” for details.

(4) Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

2.14 “for”, “while” and “do while” statements

In this module, “for”, “while” and “do while” statements (loop processing) are used in processing to wait for register to be reflected and so on. For these loop processing, comments with “WAIT_LOOP” as a keyword are described. Therefore, if user incorporates fail-safe processing into loop processing, user can search the corresponding processing with “WAIT_LOOP”.

The following shows example of description.

```
while statement example :
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for statement example :
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while statement example :
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. API Functions

R_WDT_Open()

This function initializes the WDT FIT module. This function must be executed before other API functions. This Open function is not used when WDT Auto-Start mode is enabled in the OFS0 register in r_bsp_config.h.

Format

wdt_err_t R_WDT_Open (

void * const p_cfg

)

Parameters

*void *p_cfg*

Pointer to configuration structure of type wdt_config_t (see below).

The following figure illustrates the complete runtime configurable options for Register-Start mode.

The structure is cast into a void pointer in the Open() call.

```
typedef enum e_wdt_timeout          // WDT Time-Out Period
{
    WDT_TIMEOUT_1024 =0x0000u,      // 1024 (cycles)
    WDT_TIMEOUT_4096 =0x0001u,      // 4096 (cycles)
    WDT_TIMEOUT_8192 =0x0002u,      // 8192 (cycles)
    WDT_TIMEOUT_16384=0x0003u,      // 16,384 (cycles)
WDT_NUM_TIMEOUTS
} wdt_timeout_t;

typedef enum e_wdt_clock_div        // WDT Clock Division Ratio
{
    WDT_CLOCK_DIV_4  =0x0010u,      // PCLK/4
    WDT_CLOCK_DIV_64 =0x0040u,      // PCLK/64
    WDT_CLOCK_DIV_128 =0x00F0u,     // PCLK/128
    WDT_CLOCK_DIV_512 =0x0060u,     // PCLK/512
    WDT_CLOCK_DIV_2048=0x0070u,     // PCLK/2048
    WDT_CLOCK_DIV_8192=0x0080u      // PCLK/8192
} wdt_clock_div_t;

typedef enum e_wdt_window_end       // Window End Position
{
    WDT_WINDOW_END_75=0x0000u,      // 75%
    WDT_WINDOW_END_50=0x0100u,      // 50%
    WDT_WINDOW_END_25=0x0200u,      // 25%
    WDT_WINDOW_END_0 =0x0300u       // 0%  (window end position is not
specified)
} wdt_window_end_t;

typedef enum e_wdt_window_start     // Window Start Position
{
    WDT_WINDOW_START_25 =0x0000u,   // 25%
    WDT_WINDOW_START_50 =0x1000u,   // 50%
    WDT_WINDOW_START_75 =0x2000u,   // 75%
    WDT_WINDOW_START_100=0x3000u    // 100%(window start position is not
specified)
} wdt_window_start_t;
```

```
typedef enum e_wdt_timeout_control  // Signal control when Time-out and
Refresh error
{
    WDT_TIMEOUT_NMI   =0x00u         // NMI request output is enabled
    WDT_TIMEOUT_RESET=0x80u,         // Reset output is enabled
} wdt_timeout_control_t;



typedef struct st_wdt_config
{
    wdt_timeout_t          timeout;           /* Timeout period */
    wdt_clock_div_t        wdtcks_div;        /* WDT clock division ratio */
    wdt_window_start_t     window_start;      /* Window start position */
    wdt_window_end_t       window_end;        /* Window end position */
    wdt_timeout_control_t  timeout_control;   /* Reset or NMI at timeout */
} wdt_config_t;
```

Return Values

[WDT_SUCCESS]	*/* WDT initialized*	**/*
[WDT_ERR_OPEN_IGNORED]	*/* Error: The module has already been opened*	**/*
[WDT_ERR_INVALID_ARG]	*/* Error: Argument is not valid.*	**/*
[WDT_ERR_NULL_PTR]	*/* Error: Received null pointer*	**/*
[WDT_ERR_BUSY]	*/* Error: WDT resource is locked*	**/*

Properties

Prototyped in file "r_wdt_rx_if.h".

Description

This function initializes associated WDT registers. Options can be selected per MCU.

Example

```
    wdt_config_t  config;
    wdt_err_t     err;

   /*
    * Configure the WDT for:
    * - A 2.6 ms timer period:
    *      PCLKB is 50MHz clock = 20 ns/tick; So 20 ns/tick / 128 * 1024 = 2.6 ms
    * - A 100% refresh-permitted window (start 100% end 0%)
    * - Reset on counter underflow
    * - Count stop disabled
    */
    config.timeout = WDT_TIMEOUT_1024;
    config.wdtcks_div = WDT_CLOCK_DIV_128;
    config.window_start = WDT_WINDOW_START_100;
    config.window_end = WDT_WINDOW_END_0;
    config.timeout_control = WDT_TIMEOUT_RESET;

    err = R_WDT_Open(&config);
```

Special Notes:

The Open function is only available in Register-Start mode (BSP_CFG_OFS0_REG_VALUE = 0xFFFFFFFF in r_bsp_config.h).

This function configures the WDT counter without starting the WDT counter. The WDT_CMD_REFRESH_COUNTING argument must be specified in the R_WDT_Control function to start the WDT counter.

The R_WDT_Open() function should be called only once after a reset.

Any additional calls will return WDT_ERR_OPEN_IGNORED.

The setting to enable WDT underflow/refresh error interrupt in Interrupt Controller module (ICU) must be enabled when non-maskable interrupts are selected and Auto-Start mode is enabled. A sample is provided here.

```
    ICU.NMIER.BIT.WDTEN = 1;    // Enable WDT underflow/refresh error interrupt
```

In both Auto-Start mode and Register-Start mode, the user application should have a function to handle this interrupt. A sample implementation of an NMI handler is provided here.

```
void wdt_nmi_func(void *p_args)
{
    /* Do some processing here */

    while(WDT.WDTSR.BIT.REFEF == 1)
    {
       WDT.WDTSR.BIT.REFEF = 0;  // clear Refresh Error Flag
    }
    while(WDT.WDTSR.BIT.UNDFF == 1)
    {
       WDT.WDTSR.BIT.UNDFF = 0;  // clear Underflow Flag
    }
}
```

In Register-Start mode, the function should be registered right after calling R_WDT_Open() as shown in the example above. In Auto-Start mode, the R_BSP_InterruptWrite() should occur right after enabling WDT interrupt.

For example:

```
    err = R_WDT_Open(&config);

    /* Register wdt_nmi_func() to be called whenever WDT underflow occurs. */
    bsp_err = R_BSP_InterruptWrite(BSP_INT_SRC_WDT_ERROR, wdt_nmi_func);
```

R_WDT_Control()

This function gets the WDT status and refreshes the WDT down-counter. This function may be used in both Auto-Start and Register-Start modes.

Format

wdt_err_t R_WDT_Control (

wdt_cmd_t const cmd,

uint16_t * p_status

)

Parameters

wdt_cmd_t cmd

Run command (see below).

*uint16_t *p_status*

Pointer to the storage of the counter and status flags.

The following code is used for the cmd argument.

```
WDT_CMD_GET_STATUS,          // Get WDT status
WDT_CMD_REFRESH_COUNTING,  // Refresh the counter
```

Return Values

[WDT_SUCCESS]	*/* Command completed successfully*	**/*
[WDT_ERR_INVALID_ARG]	*/* Error: Argument is not valid.*	**/*
[WDT_ERR_NULL_PTR]	*/* Error: Received null pointer*	**/*
[WDT_ERR_NOT_OPENED]	*/* Error: Open function has not yet been called*	**/*
[WDT_ERR_BUSY]	*/* Error: WDT resource is locked*	**/*

Properties

Prototyped in file "r_wdt_rx_if.h".

Description

If command WDT_CMD_REFRESH_COUNTING is selected, the watchdog counter is initialized to its starting value.

If command WDT_CMD_GET_STATUS is selected, the WDT status (underflow error status, refresh error status, and WDT counter value) register is loaded into *p_status. The two high-order bits indicate whether a refresh error occurred (b15) or an underflow occurred (b14). The remaining bits (b13 - b0) indicate the current counter value.

Example

```
wdt_config_t  config;
wdt_err_t     err;
uint16_t       status;
  :
err = R_WDT_Open(&config);                             /* Register-Start mode */
  :
err = R_WDT_Control(WDT_CMD_REFRESH_COUNTING, NULL); /* Start counting */
  :
```

```
err = R_WDT_Control(WDT_CMD_GET_STATUS, &status);     /* Get WDT status */
```

Special Notes:
The second argument is ignored for the WDT_CMD_REFRESH_COUNTING command.

R_WDT_GetVersion()

This function returns the driver version number at runtime.

Format

uint32_t R_WDT_GetVersion (void)

Parameters

None.

Return Values

Version number.

Properties

Prototyped in file "r_wdt_rx_if.h".

Description

This function returns the driver version number.

Example

```
uint32_t version;
:
version = R_WDT_GetVersion();
```

Special Notes:

None

4. Pin Setting

WDT FIT module don’t use pin setting.

5. Demo Projects

Demo projects include function main() that utilizes the FIT module and its dependent modules (e.g. r_bsp). This FIT module includes the following demo projects.

5.1 wdt_demo_rskrx230, wdt_demo_rskrx231, wdt_demo_rskrx64m, wdt_demo_rskrx71m, wdt_demo_rskrx72m, wdt_demo_rskrx671, wdt_demo_rskrx230_gcc, wdt_demo_rskrx231_gcc, wdt_demo_rskrx64m_gcc, wdt_demo_rskrx71m_gcc, wdt_demo_rskrx72m_gcc, wdt_demo_rskrx671_gcc

In the demonstration, program will request user to select options to configure WDT (Timeout Cycles, Clock Division Ratio, Window Start, Window End, Timeout Control). After configuring WDT, it begins counting down and LED0 is blinked until underflow occurs. When counter value is in refresh-permitted period or in refresh-prohibited period, Renesas Debug Virtual Console will print out to notify user. If it is in refresh-permitted period and SW1 is pressed, WDT is refreshed. Otherwise, if it is in refresh- prohibited period and SW1 is pressed, refresh error occurs.

In addition, after WDT underflow error or refresh error occurs, RESET button should be pressed to release WDT register protection so that user can input new WDT configuration.

Setup and Execution

1. Compile and download the sample code.
2. Click 'Reset Go' to start the software. If PC stops at Main, press F8 to resume.
3. Select WDT input options according to Renesas Debug Virtual Console messages so that WDT can be configured and start counting.
   Input Example:

```
Please select timeout cycles:
[0] WDT_TIMEOUT_1024  (1024 cycles)
[1] WDT_TIMEOUT_4096  (4096 cycles)
[2] WDT_TIMEOUT_8192  (8192 cycles)
[3] WDT_TIMEOUT_16384 (16,384 cycles)
Your input: 3
Please select Clock Division Ratio:
[0] WDT_CLOCK_DIV_4      (PCLK/4)
[1] WDT_CLOCK_DIV_64     (PCLK/64)
[2] WDT_CLOCK_DIV_128    (PCLK/128)
[3] WDT_CLOCK_DIV_512    (PCLK/512)
[4] WDT_CLOCK_DIV_2048   (PCLK/2048)
[5] WDT_CLOCK_DIV_8192   (PCLK/8192)
Your input: 5
Please select Window Start:
[0] WDT_WINDOW_START_25    (25%)
[1] WDT_WINDOW_START_50    (50%)
[2] WDT_WINDOW_START_75    (75%)
[3] WDT_WINDOW_START_100   (100%)
Your input: 3
Please select Window End:
[0] WDT_WINDOW_END_75    (75%)
[1] WDT_WINDOW_END_50    (50%)
[2] WDT_WINDOW_END_25    (25%)
[3] WDT_WINDOW_END_0     (0%)
Your input: 3
Please select Timeout Control:
[0] WDT_TIMEOUT_RESET    (Reset output)
[1] WDT_TIMEOUT_NMI      (NMI output)
Your input: 1
```

With above inputs, WDT configuration will be set as below:

- WDT Time-Out Period: 16384 cycles
- WDT Clock Division Ratio: PCLK/8192
- Window Start Position: 100%
- Window End Position: 0%
- WDT Time-Out Control: 1 (Non-maskable interrupt request output is enabled)

Boards Supported

RSKRX230

RSKRX231

RSKRX64M

RSKRX71M

RSKRX72M

RSKRX671

5.2 Adding a Demo to a Workspace

Demo projects are found in the FITDemos subdirectory of the distribution file for this application note. To add a demo project to a workspace, select *File* >> *Import* >> *General* >> *Existing Projects into Workspace*, then click “Next”. From the Import Projects dialog, choose the “Select archive file” radio button. “Browse” to the FITDemos subdirectory, select the desired demo zip file, then click “Finish”.

5.3 Downloading Demo Projects

Demo projects are not included in the RX Driver Package. When using the demo project, the FIT module needs to be downloaded. To download the FIT module, right click on this application note and select “Sample Code (download)” from the context menu in the *Smart Browser* >> *Application Notes* tab.

6. Appendices

6.1 Confirmed Operation Environment

This section describes confirmed operation environment for the WDT FIT module.

Table 6.1 Confirmed Operation Environment (Rev.3.00)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 2022-10 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.05.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202204 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
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.3.00
Board used	Renesas Flexible Motor Control Kit for RX26T(product No.:RTK0EMXE70S00020BJ)

Table 6.2 Confirmed Operation Environment (Rev.2.90)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 2022-07 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202104 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
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.2.90
Board used	Renesas Starter Kit for RX64M (product No.: R0K50564MxxxxBE) Renesas Starter Kit for RX71M (product No.: R0K50571MCxxxBE) Renesas Starter Kit+ for RX72M (product No.: RTK5572MNDCxxxxxBJ) Renesas Starter Kit+ for RX671 (product No.: RTK55671EDCxxxxxBJ) Renesas Starter Kit+ for RX230 (product No.: RTK505230xxxxxxxx) Renesas Starter Kit for RX231 (product No.: R0K505231SxxxBE)

Table 6.3 Confirmed Operation Environment (Rev.2.80)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 2022-04 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202104 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
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.2.80
Board used	Renesas Starter Kit for RX660 (product No: RTK556609HCxxxxxBJ)

Table 6.4 Confirmed Operation Environment (Rev.2.70)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 2021-10 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202104 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
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.2.70
Board used	Renesas Starter Kit for RX66T (product No.: RTK50566T0SxxxxxBE)

Table 6.5 Confirmed Operation Environment (Rev.2.60)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202004 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
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.2.60
Board used	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxxx)

Table 6.6 Confirmed Operation Environment (Rev.2.50)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202004 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
	IAR C/C++ Compiler for Renesas RX version 4.20.3 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.2.50
Board used	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxxx)

Table 6.7 Confirmed Operation Environment (Rev.2.40)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.8.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.02.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.201904 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	Little endian
Revision of the module	Rev.2.40
Board used	Renesas Starter Kit+ for RX72M (product No.: RTK5572Mxxxxxxxxxx)

Table 6.8 Confirmed Operation Environment (Rev.2.30)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.7.0 IAR Embedded Workbench for Renesas RX 4.12.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 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
	IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.2.30
Board used	Renesas Starter Kit+ for RX72N (product No.: RTK5572Nxxxxxxxxxx)

Table 6.9 Confirmed Operation Environment (Rev.2.20)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.5.0 IAR Embedded Workbench for Renesas RX 4.12.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 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
	IAR C/C++ Compiler for Renesas RX version 4.12.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.2.20
Board used	Renesas Starter Kit+ for RX72M (product No.: RTK5572Mxxxxxxxxxx)

Table 6.10 Confirmed Operation Environment (Rev.2.10)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.5.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.2.10
Board used	Renesas Solution Starter Kit for RX23W (product No.: RTK5523Wxxxxxxxxxx)

Table 6.11 Confirmed Operation Environment (Rev.2.00)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.4.0 IAR Embedded Workbench for Renesas RX 4.10.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 4.8.4.201803 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
	IAR C/C++ Compiler for Renesas RX version 4.10.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.2.00
Board used	Renesas Starter Kit+ for RX65N-2MB (product No.: RTK50565Nxxxxxxxxx)

Table 6.12 Confirmed Operation Environment (Rev.1.40)

Item	Details
Integrated development environment	Renesas Electronics e2 studio version 7.3.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
	Compiler option: The following option is added to the default settings of the integrated development environment. • -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.40
Board used	Renesas Starter Kit for RX72T (product No.: RTK5572Txxxxxxxxxx)

Table 6.13 Confirmed Operation Environment (Rev.1.31)

Item	Details
Integrated development environment	Renesas Electronics e^{2}studio Version 7.3.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
	Compiler option: The following option is added to the default settings of the integrated development environment. • -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.31
Board used	Renesas Starter Kit for RX66T (product No.: RTK50566T0SxxxxxBE)

Table 6.14 Confirmed Operation Environment (Rev.1.30)

Item	Details
Integrated development environment	Renesas Electronics e^{2}studio Version 7.0.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.00.00
	Compiler option: The following option is added to the default settings of the integrated development environment. • -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.30
Board used	Renesas Starter Kit for RX66T (product No.: RTK50566T0SxxxxxBE)

Table 6.15 Confirmed Operation Environment (Rev.1.20)

Item	Details
Integrated development environment	Renesas Electronics e^{2}studio Version 6.0.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.07.00
	Compiler option: The following option is added to the default settings of the integrated development environment. • -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.20
Board used	Renesas Starter Kit+ for RX231 (product No.: R0K505231SxxxBE) Renesas Starter Kit+ for RX64M (product No.: R0K50564MSxxxBE) Renesas Starter Kit+ for RX71M (product No.: R0K50571MSxxxBE)

Table 6.16 Confirmed Operation Environment (Rev.1.10)

Item	Details
Integrated development environment	Renesas Electronics e^{2}studio Version 6.0.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.07.00
	Compiler option: The following option is added to the default settings of the integrated development environment. • -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.10
Board used	Renesas Starter Kit+ for RX65N-2MB (product NO.: RTK50565N2SxxxxxBE)

Table 6.17 Confirmed Operation Environment (Rev.1.00)

Item	Details
Integrated development environment	Renesas Electronics e^{2}studio Version 5.0.1.005
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.05.00
	Compiler option: The following option is added to the default settings of the integrated development environment. • -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.00
Board used	Renesas Starter Kit+ for RX65N (product NO.: RTK500565NSxxxxxBE)

6.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file “platform.h”.

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:

  Application note “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)”

- Using e^2 studio:

  Application note “Adding Firmware Integration Technology Modules to Projects (R01AN1723)”

When using this FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note “Board Support Package Module Using Firmware Integration Technology (R01AN1685)”.

(2) Q: I have added the FIT module to the project and built it. Then I got the error: This MCU is not supported by the current r_wdt_rx module.

A: The FIT module you added may not support the target device chosen in your project. Check the supported devices of added FIT modules.

(3) Q: I have added the FIT module to the project and built it. Then I got an error for when the configuration setting is wrong.

A: The setting in the file “r_wdt_rx_config.h” may be wrong. Check the file “r_wdt_rx_config.h”. If there is a wrong setting, set the correct value for that. Refer to 2.7, Configuration Overview for details.

7. Reference Documents

User’s Manual: Hardware
The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

User’s Manual: Development Tools
RX Family CC-RX Compiler User’s Manual (R20UT3248)
The latest version can be downloaded from the Renesas Electronics website.

Related Technical Updates

This module reflects the content of the following technical updates.

- None

Revision History

Rev.	Date	Description Page	Summary
1.00	Oct 1, 2016	—	First edition issued
1.10	Oct 1, 2017	—	Added mention of support for RX65N-2MB
		3	1.1, WDT FIT Nodule, added 1.2, Outline of the API, added 1.3, Limitations, added
		4	2.2, Hardware Resource Requirements, deleted 2.3, Software Requirements The chapter number changed. The sentence changed. 2.4, Limitations, deleted 2.5, Supported Toolchains The chapter number changed. The sentence changed. 2.6, Header Files The chapter number changed. The sentence changed. 2.7 Integer Types The chapter number changed. The sentence changed. 2.4, Usage of Interrupt Vector and Exception Vector, added
		5	2.8 Configuration Overview The sentence changed. The table title changed.
		6	2.9 Code Size The chapter number changed. The sentence changed. 2.10 API Data Types, deleted 2.9 Arguments, added 2.10 Return Values, added
		7	2.11 Adding a FIT Module to Your Project The sentence changed.
		8	3.1 Overview, deleted 3.2 Return Values, deleted 3.3 R_WDT_Open() The chapter number changed. The contents modified.
		11	3.4 R_WDT_Control() The chapter number changed.
		12	3.5 R_WDT_GetVersion() The chapter number changed.
		13	4. Appendices, added
		15	5. Reference Documents, added
		Program	Since RX65N - 2MB was added, the version number of R_WDT_GetVersion function changed.
1.20	Oct 31, 2017	1	Added mention of support for RX230, RX231, RX64M, RX71M
		4	2.4 Usage of Interrupt Vector and Exception Vector The contents modified.
		6	2.8 Code Size: ROM sizes modified.
		14	4. Demo Projects Added.
1.20	Oct 31, 2017	16	5.1 Operation Confirmation Environment: Added table for Rev.1.20

		17	5.2 Troubleshooting: Added 2 more questions
		18	6 Reference Documents: Updated contents.
		Program	Changed global variable name
1.30	Sep 28, 2018	1,4	Added support for RX66T.
		6	Added code size corresponding to RX66T
		18	6.1 Confirmed Operation Environment: Added Table for Rev.1.30
1.31	Nov 16, 2018	—	Added document number in XML
		1	Added support for RX651.
			Changed Renesas Starter Kit Product No for RX66T
			Added Table for Rev.1.30
1.40	Feb 01, 2019	Program	Added support for RX72T.
		1, 4, 6	Added support for RX72T.
		9-14	Removed 'Reentrant' description in each API function.
		18	6.1 Confirmed Operation Environment: Added Table for Rev.1.40
2.00	May.20.19	—	Supported the following compilers: - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX
		1	Added the section of Target compilers.
			Deleted related documents.
			2.2 Software Requirements
		4	Requires r_bsp v5.20 or higher
		6-8	Updated the section of 2.8 Code Size
		20	Table 6.1 Confirmed Operation Environment: Added table for Rev.2.00
		24	Deleted the section of Website and Support.
		Program	Changed below for support GCC and IAR compiler: 1. Deleted the inline expansion of the R_WDT_GetVersion function. 2. Replaced nop with the intrinsic functions of BSP.
2.10	Jun.28.19	1, 4	Added support for RX23W
		7	Added code size corresponding to RX23W
		21	6.1 Confirmed Operation Environment: Added Table for Rev.2.10
		Program	Added support for RX23W.
2.20	Aug.15.19	1, 4	Added support for RX72M
		7-9	Added code size corresponding to RX72M
		21	6.1 Confirmed Operation Environment: Added Table for Rev.2.20 Table 6.2: Corrected board name for RX23W
		Program	Added support for RX72M.
2.30	Dec.30.19	1, 5	Added support for RX66N, RX72N
		4	2.3 Limitations Added limitations
		8-13	Added code size corresponding to RX66N, RX72N
		25	6.1 Confirmed Operation Environment: Added Table for Rev.2.30
		Program	Added support for RX66N, RX72N.
2.40	Jun.30.20	23,24	Added RSKRX72M to "5. Demo Projects"
		25	6.1 Confirmed Operation Environment: Added Table for Rev.2.40
		Program	Updated and added new demo project

2.50	Mar.31.21	1	Added support for RX671
		3	Added 1.3 Using the FIT WDT module. Added 1.3.1 Using FIT WDT module in C++ project.
		9,11,13	Added code size corresponding to RX671
		25	6.1 Confirmed Operation Environment: Added Table for Rev.2.50
		Program	Added support for RX671
2.60	Sep.13.21	24,25	Added RSKRX671 to “5. Demo Projects”.
		26	Table 6.1: Confirm Operation Environment: Added Table for Rev. 2.60
		Program	Updated and added new demo projects
2.70	Mar.14.22	23	Table 6.1: Confirm Operation Environment: Added Table for Rev. 2.70.
		Program	Added support for RX66T-48Pin
2.80	Mar.31.22	1, 6	Added support for RX660.
		9, 11, 13	Added code size corresponding to RX660.
		25	Table 6.1: Confirm Operation Environment: Added Table for Rev. 2.80.
		Program	Added support for RX660.
2.90	Jun.28.22	25	Table 6.1: Confirm Operation Environment: Added Table for Rev. 2.90.
		Program	Updated demo projects
3.00	Aug.15.22	1, 6	Added support for RX26T.
		9, 11, 13	Added code size corresponding to RX26T.
		25	Table 6.1: Confirm Operation Environment: Added Table for Rev. 3.00.
		Program	Added support for RX26T.

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.