RX Family

IWDT Module Using Firmware Integration Technology

Introduction

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

Target Devices

- RX110, RX111, RX113 Groups
- RX130 Groups
- RX230, RX231 Groups
- RX23W Group
- RX23T Groups
- RX24T Groups
- RX24U Groups
- RX64M Group
- RX651, RX65N Groups
- RX66T Group
- RX71M Group
- RX72T Group
- RX72M 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 IWDT FIT Module

The IWDT FIT module can be used by being implemented in a project as an API. See section 2.12, Adding the FIT Module to Your Project for details on methods to implement this FIT module into a project.

1.2 Overview of the IWDT FIT Module

This IWDT driver supports the IWDT peripheral on the RX110, RX111, RX113, RX130, RX230, RX231, RX23T, RX23W, RX24T, RX24U, RX64M, RX651, RX65N, RX66T, RX71M, RX72T and RX72M.

This driver supports both Auto-Start and Register-Start modes. By selecting Auto-Start mode via a compile-time equate, the R_IWDT_Open() code is removed from the build. For Auto-Start mode, the down-counter of IWDT is started automatically after a reset. For Register-Start mode, the down-counter of IWDT is started after a call to R_IWDT_Open() and an R_IWDT_Control() refresh operation.

The R_IWDT_Control() refresh command must be made periodically to refresh IWDT counter. If no call is made, the IWDT counter will underflow and the reset signal or non-maskable interrupt (NMI) signal will 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_IWDT_Open() function in Register-Start mode.

The IWDT module operates using both the Peripheral Clock B (PCLKB) and the IWDT Dedicated Clock (IWDTCLK). PCLKB should be operating at least four times faster than the IWDTCLK. Both clocks should be operating and settled prior to calling this module.

1.3 API Overview

Table 1.1 lists the API functions included in this module.

Table 1.1 API Functions

Function	Description
R_IWDT_Open()	Configures the IWDT counter options by initializing the associated registers. This Open function is unavailable if the IWDT is initialized by the OFS0 register in r_bsp_config.h (Auto-Start mode).
R_IWDT_Control()	Gets IWDT status (underflow error status, refresh error status and IWDT counter value) and refreshes the down-counter of IWDT.
R_IWDT_GetVersion()	Returns the driver version number at runtime.

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:

- IWDT

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 Supported Toolchain

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

2.4 Interrupt Vector

The IWDT interrupt is enabled by executing the **R_IWDT_Open()** function**.**

Table 2.1 Interrupt Vector Used in the IWDT FIT Module lists the interrupt vector used in the IWDT FIT Module.

Table 2.1 Interrupt Vector Used in the IWDT FIT Module

Device	Interrupt Vector
RX110 RX111 RX113 RX130 RX210 RX231 RX23W RX23T RX24T RX24U RX63N RX631	Non-maskable interrupt (WUNI)
RX64M RX651 RX65N RX66T RX71M RX72T RX72M	Non-maskable interrupt IWUNI*1 interrupt (vector no.: 95)

Note 1. This is the case where the corresponding non-maskable interrupt enable bit is set to 0 (disabled).

2.5 Header Files

All API calls and their supporting interface definitions are located in the r_iwdt_rx_if.h file.

2.6 Integer Types

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

2.7 Configuration Overview

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

Configuration options in r_iwdt_rx_config.h	
IWDT_CFG_PARAM_CHECKING_ENABLE 1	If this equate is set to 0, local parameter checking is excluded in the build. If the equate is set to 1, the parameter checking is included in the build. Use BSP_CFG_PARAM_CHECKING_ENABLE to set to system default.
BSP_CFG_OFS0_REG_VALUE 0xFFFFFFFF	If this equate is set to 0xFFFFFFFF, the watchdog is disabled at power-up and must be initialized using the R_IWDT_Open() function. For any other value, the R_IWDT_Open() code is removed from the build and the watchdog auto-starts at power-up. See r_bsp_config.h for configuration options.

2.8 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 2.7, Configuration Overview. The table lists reference values when the C compiler's compile options are set to their default values, as described in 2.3, 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	Renesas Compiler		Remarks
		With Parameter Checking	Without Parameter Checking	
RX110	ROM	325 bytes	179 bytes	Register Start Mode
	RAM	1 byte	1 byte	Register Start Mode
	Maximum stack usage	28 bytes		R_IWDT_Control func used
RX231	ROM	343 bytes	180 bytes	Register Start Mode
	RAM	1 byte	1 byte	Register Start Mode
	Maximum stack usage	28 bytes		R_IWDT_Control func used
RX23W	ROM	343 bytes	180 bytes	Register Start Mode
	RAM	1 byte	1 byte	Register Start Mode
	Maximum stack usage	28 bytes		R_IWDT_Control func used
RX65N	ROM	302 bytes	139 bytes	Register Start Mode
	RAM	1 byte	1 byte	Register Start Mode
	Maximum stack usage	28 bytes		R_IWDT_Control func used
RX66T	ROM	343 bytes	179 bytes	Register Start Mode
	RAM	1 byte	0 byte	Register Start Mode
	Maximum stack usage	28 bytes		R_IWDT_Control func used
RX72T	ROM	343 bytes	180 bytes	Register Start Mode
	RAM	1 byte	1 byte	Register Start Mode
	Maximum stack usage	28 bytes		R_IWDT_Control func used
RX72M	ROM	343 bytes	180 bytes	Register Start Mode
	RAM	1 byte	1 byte	Register Start Mode
	Maximum stack usage	28 bytes		R_IWDT_Control func used

ROM, RAM and Stack Code Sizes				
Device	Category	GCC		Remarks
		With Parameter Checking	Without Parameter Checking	
RX110	ROM	640 bytes	344 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX231	ROM	640 bytes	344 bytes	
	RAM	0 byte	0 byte	
	Maximum stack usage	-		
RX65N	ROM	640 bytes	344 bytes	
	RAM	4 bytes	4 bytes	
	Maximum stack usage	-		
RX66T	ROM	640 bytes	344 bytes	
	RAM	4 bytes	4 bytes	
	Maximum stack usage	-		
RX72T	ROM	640 bytes	344 bytes	
	RAM	4 bytes	4 bytes	
	Maximum stack usage	-		
RX72M	ROM	688 bytes	352 byte	
	RAM	4 bytes	4 bytes	
	Maximum stack usage	-		

ROM, RAM and Stack Code Sizes				
Device	Category	IAR Compiler		Remarks
		With Parameter Checking	Without Parameter Checking	
RX110	ROM	568 bytes	336 bytes	
	RAM	6 bytes	6 bytes	
	Maximum stack usage	140 bytes		
RX231	ROM	568 bytes	336 bytes	
	RAM	6 bytes	6 bytes	
	Maximum stack usage	144 bytes		
RX65N	ROM	591 bytes	336 bytes	
	RAM	5 bytes	5 bytes	
	Maximum stack usage	148 bytes		
RX66T	ROM	568 bytes	336 bytes	
	RAM	5 bytes	5 bytes	
	Maximum stack usage	148 bytes		
RX72T	ROM	568 bytes	336 bytes	
	RAM	5 bytes	5 bytes	
	Maximum stack usage	148 bytes		
RX72M	ROM	532 bytes	336 bytes	
	RAM	1 bytes	1 bytes	
	Maximum stack usage	160 bytes		

2.9 Parameters

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

2.10 Return Values

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

Below are the different error codes API functions can return. The enum is found in r_iwdt_rx_if.h along with the API function declarations.

```
typedef enum e_iwdt_err       // IWDT API error codes
{
    IWDT_SUCCESS=0,
    IWDT_ERR_OPEN_IGNORED,    // The module has already been opened
    IWDT_ERR_INVALID_ARG,     // Argument is not valid
    IWDT_ERR_NULL_PTR,        // Received null pointer
    IWDT_ERR_NOT_OPENED,      // Open function has not yet been called
    IWDT_ERR_BUSY,            // IWDT resource is locked
} iwdt_err_t;
```

2.11 Callback Function

None.

2.12 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.

2.13 “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_IWDT_Open()

This function configures the IWDT counter options by initializing the associated registers. It is unavailable if the IWDT is initialized by the OFS0 register in r_bsp_config.h (Auto-Start mode).

This function must be called before calling any other API functions.

Format

```
iwdt_err_t        R_IWDT_Open (
        void * const      p_cfg
)
```

Parameters

*void * const p_cfg*

Pointer to configuration structure of type *iwdt_ config_t* (see below).

The complete runtime configurable options for Register-Start mode are declared below. The structure is cast into a void pointer in the Open() call to allow for potential alternate mode configurations in the future.

```
typedef enum e_iwdt_timeout          // IWDT Time-Out Period
{
#if defined (BSP_MCU_RX11_ALL) || defined(BSP_MCU_RX130) ||
defined(BSP_MCU_RX23_ALL) || defined(BSP_MCU_RX24U)
    IWDT_TIMEOUT_128 =0x0000u,       // 128  (cycles)
    IWDT_TIMEOUT_512 =0x0001u,       // 512  (cycles)
    IWDT_TIMEOUT_1024=0x0002u,       // 1024 (cycles)
    IWDT_TIMEOUT_2048=0x0003u,       // 2048 (cycles)
#else /* RX210, RX63N, RX64M, RX71M, RX65N, RX66T, RX72T */
    IWDT_TIMEOUT_1024 =0x0000u,      // 1024 (cycles)
    IWDT_TIMEOUT_4096 =0x0001u,      // 4096 (cycles)
    IWDT_TIMEOUT_8192 =0x0002u,      // 8192 (cycles)
    IWDT_TIMEOUT_16384=0x0003u,      // 16,384 (cycles)
#endif
    IWDT_NUM_TIMEOUTS
} iwdt_timeout_t;

typedef enum e_iwdt_clock_div        // IWDT Clock Division Ratio
{
      IWDT_CLOCK_DIV_1  =0x0000u,    // IWDTCLK
      IWDT_CLOCK_DIV_16 =0x0020u,    // IWDTCLK/16
      IWDT_CLOCK_DIV_32 =0x0030u,    // IWDTCLK/32
      IWDT_CLOCK_DIV_64 =0x0040u,    // IWDTCLK/64
      IWDT_CLOCK_DIV_128=0x00F0u,    // IWDTCLK/128
      IWDT_CLOCK_DIV_256=0x0050u     // IWDTCLK/256
} iwdt_clock_div_t;

typedef enum e_iwdt_window_end       // Window End Position
{
      IWDT_WINDOW_END_75=0x0000u,    // 75%
      IWDT_WINDOW_END_50=0x0100u,    // 50%
      IWDT_WINDOW_END_25=0x0200u,    // 25%
      IWDT_WINDOW_END_0 =0x0300u     // 0%(window end position is not specified)
} iwdt_window_end_t;

typedef enum e_iwdt_window_start     // Window Start Position
{
      IWDT_WINDOW_START_25 =0x0000u, // 25%
```

```
      IWDT_WINDOW_START_50 =0x1000u, // 50%
      IWDT_WINDOW_START_75 =0x2000u, // 75%
      IWDT_WINDOW_START_100=0x3000u  // 100%(window start position is not
                                     // specified)
} iwdt_window_start_t;
typedef enum e_iwdt_timeout_control  // Signal control when Time-out and
                                    // Refresh error
{
      IWDT_TIMEOUT_NMI  =0x00u       // NMI request output is enabled
      IWDT_TIMEOUT_RESET=0x80u,      // Reset output is enabled
} iwdt_timeout_control_t;

typedef enum e_iwdt_count_stop       // Sleep mode count stop
{
      IWDT_COUNT_STOP_DISABLE=0x00u, // Count stop is disabled
      IWDT_COUNT_STOP_ENABLE =0x80u  // Count stop is enabled at a transition
                                     //    to low power consumption mode
} iwdt_count_stop_t;


typedef struct st_iwdt_config
{
    iwdt_timeout_t          timeout;          /* Timeout period */
    iwdt_clock_div_t        iwdtclk_div;      /* IWDT clock division ratio */
    iwdt_window_start_t     window_start;     /* Window start position */
    iwdt_window_end_t       window_end;       /* Window end position */
    iwdt_timeout_control_t  timeout_control;  /* Reset or NMI at timeout */
    iwdt_count_stop_t       count_stop_enable;/* Sleep mode count stop flag */
 } iwdt_config_t;
```

Return Values

[IWDT_SUCCESS]	*/* IWDT initialized*	*/*
[IWDT_ERR_OPEN_IGNORED]	*/* The module has already been opened*	*/*
[IWDT_ERR_INVALID_ARG]	*/* An element of the p_cfg structure contains an invalid value*	*/*
[IWDT_ERR_NULL_PTR]	*/* p_cfg pointer is NULL*	*/*
[IWDT_ERR_BUSY]	/* IWDT resource is locked	*/*

Properties

Prototyped in file “r_iwdt_rx_if.h”.

Description

Sets all configurable options for the Independent Watchdog Timer.

Example

```
    iwdt_config_t  config;
    iwdt_err_t     err;

    /*
     * Configure the IWDT for:
     * - A 8.738 S timer period:
     *      15 kHz clock = 66.67 uS/tick; So 66.67 uS/tick * 128 * 1024 = 8.738 S
     * - A 100% refresh-permitted window (start 100% end 0%)
     * - Reset on counter underflow
     * - Count stop  enabled
     */
    config.timeout = IWDT_TIMEOUT_1024;
    config.iwdtclk_div = IWDT_CLOCK_DIV_128;
    config.window_start = IWDT_WINDOW_START_100;
    config.window_end = IWDT_WINDOW_END_0;
    config.timeout_control = IWDT_TIMEOUT_RESET;
```

```
    config.count_stop_enable = IWDT_COUNT_STOP_ENABLE;

    err = R_IWDT_Open(&config);
```

Special Notes:
The Open function is only available in Register-Start mode (BSP_CFG_OFS0_REG_VALUE = 0xFFFFFFFF). This function configures the IWDT module without starting the IWDT counter. The R_IWDT_Control() function with a refresh command must be issued to start the IWDT counter.

The R_IWDT_Open() function should be called only once after a reset. Any additional calls will return IWDT_ERR_OPEN_IGNORED.

Before calling R_IWDT_Open() function, Peripheral Clock B (PCLKB) and internal clock IWDT-dedicated Clock (IWDTCLK) must be initialized by user application. PCLKB clock can be configured via the compile-time configuration setting in the BSP. The IWDTCLK can be initialized via direct register setting. A sample setting to start operating IWDT-dedicated clock is provided here:

```
R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_CGC); // Register protect off
SYSTEM.ILOCOCR.BIT.ILCSTP = 0;                     // Enable IWDT oscillator
R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_CGC);  // Register protect on
```

The user shall ensure that the PCLKB clock frequency is four times larger than or equal to the IWDTCLK clock frequency after division.

If the NMI signal is selected, additional settings are necessary to configure the NMI handler. The setting to enable IWDT underflow/refresh error interrupt in Interrupt Controller module (ICU) is done in R_IWDT_Open() for Register-Start mode. In Auto-Start mode, it must be done by the user application. A sample setting is provided here:

```
    ICU.NMIER.BIT.IWDTEN = 1;   // Enable IWDT 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 iwdt_nmi_func(void *p_args)
{
    /* Do some processing here */
    while(IWDT.IWDTSR.BIT.REFEF == 1)
    {
       IWDT.IWDTSR.BIT.REFEF = 0;  // clear Refresh Error Flag
    }
    while(IWDT.IWDTSR.BIT.UNDFF == 1)
    {
       IWDT.IWDTSR.BIT.UNDFF = 0;  // clear Underflow Flag
    }
}
```

To register the function, call R_BSP_InterruptWrite(). For example:

```
    err = R_IWDT_Open(&config);

    /* Register iwdt_nmi_func() to be called whenever IWDT underflow occurs. */
    err = R_BSP_InterruptWrite(BSP_INT_SRC_IWDT_ERROR, iwdt_nmi_func);
```

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

R_IWDT_Control()

This function performs getting the IWDT status and refreshing the down-counter of IWDT. This function may be used in both Auto-Start and Register-Start modes.

Format

```
iwdt_err_t      R_IWDT_Control (
        iwdt_cmd_t const        cmd,
        uint16_t                * p_status
)
```

Parameters

iwdt_cmd_t const cmd
Command to run (see enumeration below).

uint16_t p_status
Pointer to the storage of the counter and status flags.

The following enumeration is used for the *cmd* argument:

```
IWDT_CMD_GET_STATUS,        // Get IWDT status
IWDT_CMD_REFRESH_COUNTING,  // Refresh the counter
```

The following masks delineate the fields in the *p_status parameter.

```
#define IWDT_STAT_REFRESH_ERR_MASK      (0x8000)
#define IWDT_STAT_UNDERFLOW_ERR_MASK    (0x4000)
#define IWDT_STAT_ERROR_MASK            (0xC000)
#define IWDT_STAT_COUNTER_MASK          (0x3FFF)
```

Return Values

```
[IWDT_SUCCESS]          /* Command completed successfully      */
[IWDT_ERR_INVALID_ARG]  /* Invalid argument                    */
[IWDT_ERR_NULL_PTR]     /* p_status is NULL                    */
[IWDT_ERR_NOT_OPENED]   /* Open function has not yet been called */
[IWDT_ERR_BUSY]         /* IWDT resource is locked             */
```

Properties

Prototyped in file “r_iwdt_rx_if.h”

Description

If command IWDT_CMD_REFRESH_COUNTING is selected, the watchdog counter is initialized to its start value and counting continues.

If command IWDT_CMD_GET_STATUS is selected, the IWDT status register is loaded into *p_status. The high order 2 bits indicate whether a refresh error occurred (refresh called outside of legal window) or an underflow occurred (counter expired). The remaining bits indicate the current counter value.

Example

```
iwdt_config_t  config;
iwdt_err_t     err;
uint16_t       status;
  :
err = R_IWDT_Open(&config);                              /* Register-Start mode */
  :
err = R_IWDT_Control(IWDT_CMD_REFRESH_COUNTING, NULL); /* Start counting */
  :
err = R_IWDT_Control(IWDT_CMD_GET_STATUS, &status);    /* Get IWDT status */
```

Special Notes:

This second argument is ignored for command IWDT_CMD_REFRESH_COUNTING.
When initializing the watchdog timer with the IWDT_CMD_REFRESH_COUNTING command, up to four count cycles is required (the number of cycles of the IWDT-dedicated clock (IWDTCLK) for one count cycle varies depending on the clock division ratio specified in the Open). Therefore, an execution of the IWDT_CMD_REFRESH_COUNTING command should be completed four count cycles before the end position of the refresh-permitted period or a counter underflow.

R_IWDT_GetVersion()

This function returns the driver version number at runtime.

Format
uint32_t R_IWDT_GetVersion (void)

Parameters
None.

Return Values
Version number.

Properties
Prototyped in file “r_iwdt_rx_if.h”.

Description
Returns the version of this module. The version number is encoded such that the top 2 bytes are the major version number and the bottom 2 bytes are the minor version number.

Example
```
uint32_t   version;
  :
version = R_IWDT_GetVersion();
```

Special Notes:
None.

4. Pin Setting

IWDT 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 iwdt_demo_rskrx113

This is a simple demo of the RX113 Independent Watchdog Timer (IWDT) for the RSKRX113 starter kit (FIT module "r_iwdt_rx"). The demo configures the IWDT for a 68 ms period and to generate an interrupt on underflow/refresh error. The IWDT is then started and refreshed every 50 ms for a total of 10 seconds while flashing LED0. The IWDT refresh is then stopped so that the watchdog timer expires and generates an interrupt. The interrupt handler sets a global flag which causes main() to stop flashing LED0 and begin flashing LED2 as a visual indicator that the IWDT underflow has occurred.

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. Set breakpoints and watch global variables

Boards Supported

RSKRX113

5.2 iwdt_demo_rskrx231

This is a simple demo of the RX231 Independent Watchdog Timer (IWDT) for the RSKRX231 starter kit (FIT module "r_iwdt_rx"). The demo configures the IWDT for a 68 ms period and to generate an interrupt on underflow/refresh error. The IWDT is then started and refreshed every 50 ms for a total of 10 seconds while flashing LED0. The IWDT refresh is then stopped so that the watchdog timer expires and generates an interrupt. The interrupt handler sets a global flag which causes main() to stop flashing LED0 and begin flashing LED2 as a visual indicator that the IWDT underflow has occurred.

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. Set breakpoints and watch global variables

Boards Supported

RSKRX231

5.3 iwdt_demo_rskrx64m

This is a simple demo of the RX64M Independent Watchdog Timer (IWDT) for the RSK RX64M starter kit (FIT module "r_iwdt_rx"). The demo configures the IWDT for a 68 ms period and to generate an interrupt on underflow/refresh error. The IWDT is then started and refreshed every 50 ms for a total of 10 seconds while flashing LED0. The IWDT refresh is then stopped so that the watchdog timer expires and generates an interrupt. The interrupt handler sets a global flag which causes main() to stop flashing LED0 and begin flashing LED2 as a visual indicator that the IWDT underflow has occurred.

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. Set breakpoints and watch global variables

Boards Supported

RSKRX64M

5.4 iwdt_demo_rskrx71m

This is a simple demo of the RX71M Independent Watchdog Timer (IWDT) for the RSK RX71M starter kit (FIT module "r_iwdt_rx"). The demo configures the IWDT for a 68 ms period and to generate an interrupt on underflow/refresh error. The IWDT is then started and refreshed every 50 ms for a total of 10 seconds while flashing LED0. The IWDT refresh is then stopped so that the watchdog timer expires and generates an interrupt. The interrupt handler sets a global flag which causes main() to stop flashing LED0 and begin flashing LED2 as a visual indicator that the IWDT underflow has occurred.

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. Set breakpoints and watch global variables

Boards Supported

RSKRX71M

5.5 iwdt_demo_rskrx65n

This is a simple demo of the RX65N Independent Watchdog Timer (IWDT) for the RSK RX65N starter kit (FIT module "r_iwdt_rx"). The demo configures the IWDT for a 68 ms period and to generate an interrupt on underflow/refresh error. The IWDT is then started and refreshed every 50 ms for a total of 10 seconds while flashing LED0. The IWDT refresh is then stopped so that the watchdog timer expires and generates an interrupt. The interrupt handler sets a global flag which causes main() to stop flashing LED0 and begin flashing LED2 as a visual indicator that the IWDT underflow has occurred.

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. Set breakpoints and watch global variables

Boards Supported

RSKRX65N

5.6 iwdt_demo_rskrx65n_2m

This is a simple demo of the RX65N-2MB Independent Watchdog Timer (IWDT) for the RSK RX65N-2MB starter kit (FIT module "r_iwdt_rx"). The demo configures the IWDT for a 68 ms period and to generate an interrupt on underflow/refresh error. The IWDT is then started and refreshed every 50 ms for a total of 10 seconds while flashing LED0. The IWDT refresh is then stopped so that the watchdog timer expires and generates an interrupt. The interrupt handler sets a global flag which causes main() to stop flashing LED0 and begin flashing LED2 as a visual indicator that the IWDT underflow has occurred.

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. Set breakpoints and watch global variables

Boards Supported

RSKRX65N-2MB

5.7 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".

5.8 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 IWDT FIT module.

Table 6.1 Confirmed Operation Environment (Rev.3.20)

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.3.20
Board used	Renesas Starter Kit+ for RX72M (product No.: RTK5572Mxxxxxxxxxx)

Table 6.2 Confirmed Operation Environment (Rev.3.10)

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.3.10
Board used	Renesas Solution Starter Kit for RX23W (product No.: RTK5523Wxxxxxxxxxx)

Table 6.3 Confirmed Operation Environment (Rev.3.00)

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.3.00
Board used	Renesas Starter Kit+ for RX65N-2MB (product No.: RTK50565Nxxxxxxxxx)

Table 6.4 Confirmed Operation Environment (Rev.2.00)

Item	Contents
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.2.00
Board used	Renesas Starter Kit for RX72T (product No.: RTK5572Txxxxxxxxxx)

Table 6.5 Confirmed Operation Environment (Rev.1.91)

Item	Contents
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.1.91
Board used	Renesas Starter Kit for RX66T (product No.: RTK50566T0SxxxxxBE) Renesas Starter Kit+ for RX 65N-2MB (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308CxxxxxBR)

Table 6.6 Confirmed Operation Environment (Rev.1.90)

Item	Contents
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.1.90
Board used	Renesas Starter Kit for RX66T (product No.: RTK50566T0SxxxxxBE) Renesas Starter Kit+ for RX 65N-2MB (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308CxxxxxBR)

Table 6.7 Confirmed Operation Environment (Rev.1.81)

Item	Contents
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.1.81
Board used	Renesas Starter Kit+ for RX 65N-2MB (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308CxxxxxBR)

Table 6.8 Confirmed Operation Environment (Rev.1.80)

Item	Contents
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.1.80
Board used	Renesas Starter Kit+ for RX 65N-2MB (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX130-512KB (product No.: RTK5051308CxxxxxBR)

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 a 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_iwdt_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_iwdt_rx_config.h” may be wrong. Check the file “r_iwdt_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 version 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 Compiler CC-RX User's Manual (R20UT3248)
The latest versions can be downloaded from the Renesas Electronics website.

Related Technical Updates

This module reflects the content of the following technical updates.

None

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Nov.15, 2013	—	First edition issued
1.20	Mar.25, 2014	1,2	Added mention of support for RX110, RX210, RX63N
		4	Removed OFS0 equates from r_iwdt_config.h table.
		4	Added r_bsp_config.h configuration table.
		6	Added #ifdef to e_iwdt_timeout
1.30	Dec.30, 2014	1,3,5	Added mention of support for RX113
		15	Added Demo Projects section
1.40	Mar.6, 2015	1,3,5,14	Added mention of support for RX64M/RX71M Added Demo Projects section
1.50	Jun.2, 2015	1,3,5,14	Added mention of support for RX231 Added Demo Projects section
1.51	Mar.1, 2016	1,3,4,6	Added mention of support for RX130, RX230, RX23T
1.60	Oct.1, 2016	1,4,6,9	Added mention of support for RX65N Changed Code Size section
1.70	Feb.28, 2017	—	Added support for the RX24T (including ROM 512 KB version) and RX24U Groups.
		—	Corrected some descriptions.
		4	Added RXC v2.06.00 to “2.5 Supported Toolchains”.
		12	Added the description regarding the IWDT_CMD_REFRESH_COUNTING command in the Special Notes in 3.4 R_IWDT_Control()
		Program	The code has been modified to check arguments for both NULL and FIT_NO_PTR.
1.80	July.21, 2017	—	Added support for the RX130-512KB and RX65N-2MB.
		4	Added RXC v2.07.00 to “2.5 Supported Toolchains”.
		5	Added “2.6 Interrupt Vector”.
		8	Added “2.12 Adding the FIT Module to Your Project”.
1.81	Oct.31, 2017	17	Added “4.5 iwdt_demo_rskr65n”
		17	Added “4.6 iwdt_demo_rskr65n_2m”
		18	Added “4.8 Downloading Demo Projects”
		19	Added "5. Appendices”
1.90	Sep 28, 2018	1, 3, 4	Added support for the RX66T.
		6	Added code size corresponding to RX66T
		19	6.1 Confirmed Operation Environment: Added table for Rev.1.90
1.91	Nov 16, 2018	—	Added document number in XML
		19	Changed Renesas Starter Kit Product No for RX66T. Added table for Rev.1.91
2.00	Feb 01, 2019	Program	Added support for the RX72T.
		1, 3, 4, 6	Added support for the RX72T.
		9-14	Removed ‘Reentrant’ description in each API function.
		19	6.1 Confirmed Operation Environment: Added table for Rev.2.00

3.00	May.20.19	—	Supported the following compilers: - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX
		1	Deleted the RX210, RX631, and RX63N in Target Devices for end of update these devices. Added the section of Target compilers. Deleted related documents.
		3	1.2 Overview of the IWDT FIT Module Deleted the description of RX210, RX631, and RX63N.
		4	2.2 Software Requirements Requires r_bsp v5.20 or higher
		6-8	Updated the section of 2.8 Code Size
		21	Table 6.1 Confirmed Operation Environment: Added table for Rev.3.00
		24	Deleted the section of Website and Support.
		Program	Changed bellow for support GCC and IAR compiler: Deleted the inline expansion of the R_IWDT_GetVersion function.
3.10	Jun.28.19	1, 4	Added support for RX23W
		6	Added code size corresponding to RX23W
		11	Added defined(BSP_MCU_RX24U) in R_IWDT_Open function Added comment RX71M, RX65N, RX66T, RX72T in R_IWDT_Open function
		21	6.1 Confirmed Operation Environment: Added Table for Rev.3.10
		Program	Added support for RX23W.
3.20	Aug.15.19	1, 3, 4	Added support for RX72M
		6-8	Added code size corresponding to RX72M
		21	6.1 Confirmed Operation Environment: Added Table for Rev.3.20 Table 6.2: Corrected board name for RX23W
		Program	Added support for RX72M.

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

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

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

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

3. Input of signal during power-off state

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

4. Handling of unused pins

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

5. Clock signals

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

6. Voltage application waveform at input pin

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

7. Prohibition of access to reserved addresses

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

8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.4.0-1 November 2017)

Corporate Headquarters

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

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.