RX Family

CMT Module Using Firmware Integration Technology

R01AN1856EJ0320
Rev. 3.20
July 21, 2017

Introduction

This FIT module provides basic functions for use of the Compare Match Timer (CMT) on RX MCUs. This document describes the CMT Module API.

Target Device

The following is a list of devices that are currently supported by this API:

- **RX110 Group**
- **RX111 Group**
- **RX113 Group**
- **RX130 Group**
- **RX210 Group**
- **RX230 Group**
- **RX231 Group**
- **RX23T Group**
- **RX24T Group**
- **RX24U Group**
- **RX63N, RX631 Groups**
- **RX64M Group**
- **RX65N, RX651 Groups**
- **RX71M Group**

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

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833)
- Board Support Package Firmware Integration Technology Module (R01AN1685)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723)
- Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)

Contents

1. Overview 3
1.1 Using the FIT CMT module 3
1.2 Callback Functions 3
1.2.1 Example callback function prototype declaration. 3
1.2.2 Dereferencing of pdata argument. 3

2. API Information 4
2.1 Hardware Requirements 4
2.2 Hardware Resource Requirements 4
2.3 Software Requirements 4
2.4 Limitations 4
2.5 Supported Toolchains 4
2.6 Interrupt Vector 5
2.7 Header Files 5
2.8 Integer Types 5
2.9 Configuration Overview 5
2.10 Code Size 5
2.11 API Data Structures 6
2.11.1 Special Data Types 6
2.12 Return Values 6
2.13 Adding the FIT Module to Your Project 7

3. API Functions 8
3.1 Summary 8
3.2 R_CMT_CreatePeriodic() 9
3.3 R_CMT_CreateOneShot() 10
3.4 R_CMT_Stop() 11
3.5 R_CMT_Control() 12
3.6 R_CMT_GetVersion() 14

4. Demo Projects 15
4.1 cmt_demo_rskrx113 15
4.2 cmt_demo_rskrx231 15
4.3 cmt_demo_rskrx64M 15
4.4 cmt_demo_rskrx71m 15
4.5 Adding a Demo to a Workspace 15

1. Overview

This software module provides a simple interface to the RX Compare Match Timer (CMT) peripheral. The CMT is a two-channel, 16-bit timer. Each channel contains a free-running counter with a prescaler and a 16-bit compare register. Interrupts can be generated when the free-running counter matches the compare register. The counter is automatically reset and restarted on a compare event making this an ideal timer for pacing repetitive software events like RTOS schedulers. A CMT unit contains two channels; RX MCUs contain either one or two CMT units resulting in either two or four independent CMT channels.

This driver provides functions for creating and starting a CMT channel, pausing and restarting a channel, and shutting down a channel. User application code can be called via a callback function.

1.1 Using the FIT CMT module

The primary use of the CMT module is to make it easy to generate repetitive events and fixed time intervals.

After adding the CMT module to your project you will need to modify the *r_cmt_rx_config.h* file to configure the software for your installation.

Use the functions R_CMT_CreatePeriodic and R_CMT_CreateOneShot to start a timer. Provide a pointer to your callback function as an argument and your callback will be called when the timer expires. Be aware that during execution of your callback, interrupts will be disabled by default, since it is executing from within the context of the ISR. Therefore it is recommended to keep callback functions small so that they complete quickly.

In theory, the CMT timer maximum clocking speed is limited to PCLK/8. When using the periodic timer function to generate a clock, be aware that interrupt and callback processing takes some time. So this will limit the maximum frequency that can be generated.

1.2 Callback Functions

The definition of callbacks follows the FIT 1.0 specification rules:

a. Callback functions take one argument. This argument is ‘void * pdata’.

b. Before calling a callback function the function pointer is checked to be valid. At a minimum the pointer is be checked to be non-**null**, and not equal to **FIT_NO_FUNC** macro.

1.2.1 Example callback function prototype declaration.

You must provide your own callback functions. A callback function is just a normal C function that does not return a value (void) and has one parameter that is a pointer to void, as in the following declaration:

```
void my_cmt_callback(void * pdata);
```

1.2.2 Dereferencing of pdata argument.

When the ISR calls your callback function it will pass a pointer to a value containing the CMT channel number that triggered the interrupt. Since FIT callbacks take a void pointer, you will need to type-cast the pointer so that it can be dereferenced. The CMT channel number is passed as a uint32_t in the range of 0-3.

Example:

```
void my_cmt_callback(void * pdata)
{
    uint32_t  cmt_event_channel_number;

    cmt_event_channel_number = *((uint32_t *)pdata); //cast pointer to uint32_t
    ...
}
```

2. API Information

The sample code in this application note has been run and confirmed under the following conditions.

2.1 Hardware Requirements

This driver requires a RX MCU with the CMT peripheral.

2.2 Hardware Resource Requirements

This driver does not require any resources outside of the CMT. Range and resolution of the CMT timers is determined by the peripheral clock setting of the MCU.

2.3 Software Requirements

This driver is dependent upon the support from the following software:

- This software depends on a FIT compliant BSP module being present that supports the MCU model in use.
- The peripheral clock must be initialized before starting the CMT.

2.4 Limitations

None.

2.5 Supported Toolchains

This driver is tested and working with the following toolchains:

- Renesas RX Toolchain v.2.02.00 (RX110, RX111, RX113, RX210, RX231, RX63N, RX64M, RX71M)
- Renesas RX Toolchain v.2.03.00 (RX130, RX230, RX23T, RX24T)
- Renesas RX Toolchain v.2.05.00 (RX24U, RX651, RX65N)
- Renesas RX Toolchain v.2.06.00 (RX24U)
- Renesas RX Toolchain v.2.07.00 (RX65N-2MB, RX130-512KB)

2.6 Interrupt Vector

CMT interrupt is enabled by execution **R_CMT_CreatePeriodic** funtion and **R_CMT_CreateOneShot** function.

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

Table 2.1 Interrupt Vector Used in the CMT FIT Module

Device	Interrupt Vector
RX110[*1] RX111[*1] RX113 RX130[*1] RX210 RX230 RX231 RX23T RX24T RX24U RX63N RX631	● CMI0 interrupt[channel 0] (vector no.: 28) ● CMI1 interrupt[channel 1] (vector no.: 29) ● CMI2 interrupt[channel 2] (vector no.: 30) ● CMI3 interrupt[channel 3] (vector no.: 31)
RX64M RX651 RX65N RX71M	● CMI0 interrupt[channel 0] (vector no.: 28) ● CMI1 interrupt[channel 1] (vector no.: 29) ● CMI2 interrupt[channel 2] (vector no.: 128) [*2] ● CMI3 interrupt[channel 3] (vector no.: 129) [*2]

Note 1. Only have 2 channels (CMT0, CMT1).
Note 2. The interrupt vector numbers for software configurable interrupt show the default values specified in the board support package FIT module (BSP module).

2.7 Header Files

All API calls are accessed by including a single file "r_cmt_rx_if.h" which is supplied with this software's project code.

Build-time configuration options are selected or defined in the file "r_cmt_rx_config.h"

2.8 Integer Types

This project uses ANSI C99 "Exact width integer types" in order to make the code clearer and more portable. These types are defined in *stdint.h*.

2.9 Configuration Overview

Some features or behavior of the software are determined at build-time by configuration options that the user must select.

Configuration options in *r_cmt_rx_config.h*	
`CMT_RX_CFG_IPR (5)`	Interrupt priority level used for CMT interrupts

Table 2 : List of CMT module configuration options

2.10 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 2.9, Configuration Overview. The table lists reference values when the C compiler's compile options are set to their default values, as described in 2.5, Supported Toolchains. 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**
2 channel parts RX110, RX111, and RX130	ROM	837 bytes	
	RAM	16 bytes	
	Maximum stack usage	64 bytes	
4 channel parts RX113, RX210, RX230, RX231, RX23T, RX24T, RX24U, RX631, RX63N, RX64M, RX651, RX65N, RX71M	ROM	1200 bytes	
	RAM	32 bytes	
	Maximum stack usage	64 bytes	

2.11 API Data Structures

This section details the data structures that are used with the driver's API functions.

2.11.1 Special Data Types

To provide strong type checking and reduce errors, many parameters used in API functions require arguments to be passed using the provided type definitions. Allowable values are defined in the public interface file *r_cmt_rx_if.h*.

2.12 Return Values

All CMT functions return a Boolean value that indicates success or failure of the call.

2.13 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends using “Smart Configurator” described in (1) or (3). However, “Smart Configurator” only supports some RX devices. Please use the methods of (2) or (4) for unsupported RX devices.

(1) Adding the FIT module to your project using “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 “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 “Smart Configurator” on 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.

3. API Functions

3.1 Summary

The following functions are included in this design:

Function	Description
R_CMT_CreatePeriodic()	Finds an unused CMT channel, configures the timer for the desired periodic frequency by setting the appropriate prescaler, associates a user callback function with the timer's interrupt, and starts the timer. The timer continues to run, generating an interrupt and calling the callback at the desired frequency, until the user shuts it down.
R_CMT_CreateOneShot()	Similar to R_CMT_CreatePeriodic; however, the timer is shut down after the first interrupt and callback.
R_CMT_Control()	Commands the timer to pause, restart, or report status.
R_CMT_Stop()	Turns off a CMT channel, disables interrupts, and powers down the CMT unit if it is not in use.
R_CMT_GetVersion()	Returns the driver version number at runtime.

3.2 R_CMT_CreatePeriodic()

This function finds an unused CMT channel, configures it for the requested frequency, associates a user callback function with the timer's interrupt, and powers up and starts the timer.

Format

```
bool R_CMT_CreatePeriodic( uint32_t frequency_hz
                           void (* callback) (void *pdata),
                           uint32_t *channel)
```

Parameters

frequency_hz

Desired frequency in Hz note 1. The range and resolution of the timer is determined by settings of the peripheral clock. The best prescaler for the CMT channel is chosen by the driver

callback

Pointer to the user's callback function. It should receive a single void * argument.

channel

The CMT FIT module finds the first CMT channel that is not in use and assigns it to the caller. This allows multiple drivers to use the CMT driver without having to pre-assign all timer channels. This argument provides a way for the driver to indicate back to the caller which channel has been assigned.

Return Values

true:	*Successful; CMT initialized*
false:	*No free CMT channels available, or invalid settings*

Properties

Prototyped in file "r_cmt_rx_if.h"

Description

The R_CMT_CreatePeriodic function finds an unused CMT channel, assigns it to the caller, and registers a user callback function to be called upon compare match events. The CMT is configured to generate compare matches at the frequency specified in the call.

Reentrant

No

Example

This example sets up 10 Hz (100 ms) compare match operation with callback.
The example shows a user provided callback function cb that will be called to notify the user each time compare match event occurs.

```
uint32_t   ch;
bool       ret;

ret = R_CMT_CreatePeriodic(10, &cb, &ch);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:

1. Maximum periodic frequency
   In hardware, the CMT timer maximum clocking speed is limited to PCLK/8. However, when using the periodic timer function to generate a clock, be aware that interrupt and callback processing takes some time. As requested frequency rises, interrupt and callback processing will take an increasing percentage of the processor's time. At some point, too much time is consumed to leave any time for other useful work. So this will limit the maximum frequency that can be generated. The maximum practical frequency will depend on your system design, but in general, frequencies up to a few kilohertz are reasonable.

3.3 R_CMT_CreateOneShot()

This function finds an unused CMT channel, configures it for the requested period, associates a user callback function with the timer's interrupt, and powers up and starts the timer.

Format

```
bool R_CMT_CreateOneShot( uint32_t period_us,
                     void (* callback)(void *pdata),
                     uint32_t *channel)
```

Parameters

period_us

Desired period in microseconds. The range and resolution of the timer is determined by settings of the peripheral clock. The best prescaler for the CMT channel is chosen by the driver

callback

Pointer to the user's callback function. It should data a single void * argument.

channel

The CMT FIT module finds the first CMT channel that is not in use and assigns it to the caller. This allows multiple drivers to use the CMT driver without having to pre-assign all timer channels. This argument provides a way for the driver to indicate back to the caller which channel has been assigned.

Return Values

true:	*Successful; CMT initialized*
false:	*No free CMT channels available, or invalid settings*

Properties

Prototyped in file "r_cmt_rx_if.h"

Description

The R_CMT_CreateOneShot function finds an unused CMT channel, assigns it to the caller, and registers a user callback function to be called upon the compare match event. The CMT is configured to generate a compare match after the period specified in the call. The timer is shut down after a single compare match event.

Reentrant

No

Example

This example sets up 100 ms compare match operation with callback.

```
uint32_t   ch;
bool       ret;

ret = R_CMT_CreateOneShot(100000, &cb, &ch);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:

None

3.4 R_CMT_Stop()

Stops a CMT channel and powers down the CMT unit if possible.

Format

```
bool      R_CMT_Stop(uint32_t channel);
```

Parameters

channel

The CMT timer channel to stop

Return Values

true:	*Successful; CMT closed*
false:	*Invalid settings*

Properties

Prototyped in file “r_cmt_rx_if.h”

Description

This function frees the CMT channel by clearing its assignment and disabling the associated interrupt. The CMT channel cannot be used again until it has been reopened with either the R_CMT_CreatePeriodic or the R_CMT_CreateOneShot function.

Reentrant

Yes.

Example

This example stops CMT timer channel.

```
uint32_t   ch;
bool       ret;

/* Open and start the timer */
ret = R_CMT_CreatePeriodic(10, &cb, &ch);

/* Stop the timer */
ret = R_CMT_Stop(ch);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:

None

3.5 R_CMT_Control()

This function provides various ways to control and monitor a CMT channel

Format

```
bool R_CMT_Control ( uint32_t channel,
                     cmt_commands_t command,
                     void *pdata);
```

Parameters

channel

CMT channel number to control

command

Command to execute:

CMT_RX_CMD_IS_CHANNEL_COUNTING
CMT_RX_CMD_PAUSE
CMT_RX_CMD_RESUME
CMT_RX_CMD_RESTART
CMT_RX_CMD_GET_NUM_CHANNELS

Return Values

true:	*The command completed properly. Check pdata*
false:	*The command did not complete properly*

Properties

Prototyped in file “r_cmt_rx_if.h”

Description

This function provides a number of commands:

CMT_RX_CMD_IS_CHANNEL_COUNTING tells if a CMT channel is currently running. Check *pdata.
CMT_RX_CMD_PAUSE pauses a timer without closing it (without powering it off).
CMT_RX_CMD_RESUME restarts a paused timer without resetting the counter to zero
CMT_RX_CMD_RESTART restarts a paused timer after resetting the counter to zero
CMT_RX_CMD_GET_NUM_CHANNELS returns the total number of channels available

Reentrant

Yes.

Example 1

This example pauses a timer and restarts a paused timer.

```
uint32_t   ch;
bool       ret;

/* Open and Start the timer */
ret = R_CMT_CreatePeriodic(10, &cb, &ch);

if (true != ret)
{
    /* Handle the error */
}

/* Pause the timer */
ret = R_CMT_Control(ch, CMT_RX_CMD_PAUSE, NULL);

if (true != ret)
{
```

```
    /* Handle the error */
}

/* Restart the timer after resetting the counter to zero */
ret = R_CMT_Control(ch_info, CMT_RX_CMD_RESTART, NULL);

if (true != ret)
{
    /* Handle the error */
}
```

Example 2

This example check state of channel and get total number of channels available.

```
uint32_t   ch;
uint32_t   ch_num;
bool       ret;
bool       data;

/* Open and Start the timer */
ret = R_CMT_CreatePeriodic(10, &cb, &ch);

if (true != ret)
{
    /* Handle the error */
}

/* Check state of channel */
ret = R_CMT_Control(ch, CMT_RX_CMD_IS_CHANNEL_COUNTING, (void*)&data);

if (true != ret)
{
    /* Handle the error */
}

/* Get available of channel */
ret = R_CMT_Control(ch, CMT_RX_CMD_GET_NUM_CHANNELS, (void*)&ch_num);

if (true != ret)
{
    /* Handle the error */
}
```

Special Notes:

None

3.6 R_CMT_GetVersion()

This function returns the driver version number at runtime.

Format

```
uint32_t  R_CMT_GetVersion(void);
```

Parameters

None

Return Values

Version number with major and minor version digits packed into a single 32-bit value.

Properties

Prototyped in file “r_cmt_rx_if.h”

Description

The function 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.

Reentrant

Example

Example showing this function being used.

```
/* Retrieve the version number and convert it to a string. */

uint32_t   version, version_high, version_low;
char       version_str[9];

version = R_CMT_GetVersion();

version_high = (version >> 16)&0xf;
version_low  =  version & 0xff;

sprint(version_str, “CMT v%1.1hu.%2.2hu”, version_high, version_low);
```

Special Notes:

None.

4. Demo Projects

Demo projects are complete stand-alone programs. They include function main() that utilizes the module and its dependent modules (e.g. r_bsp). This FIT module has the following demo projects:

4.1 cmt_demo_rskrx113

The cmt_demo_rskrx113 program demonstrates how to create a timer tick using a CMT channel, how to set up a callback function to handle CMT interrupts and how to de-reference the channel information in the callback argument. As the program runs, the CMT callback function toggles LED0 at a 2 Hz rate.

4.2 cmt_demo_rskrx231

The cmt_demo_rskrx231 program is identical to cmt_demo_rskrx113.

4.3 cmt_demo_rskrx64M

The cmt_demo_rskrx64M program is identical to cmt_demo_rskrx113.

4.4 cmt_demo_rskrx71m

The cmt_demo_rskrx71m program is identical to cmt_demo_rskrx113.

4.5 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”.

Related Technical Updates

This module reflects the content of the following technical updates.

None

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

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

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
2.00	Nov 6, 2013	—	First GSCE Release
2.10	Nov 15, 2013	—	Formula for CMCOR value corrected.
2.30	Apr 12, 2014	1, 2, 7	Updated to indicate support for additional MCUs Added section 1.2 on callback functions Added notes on maximum periodic frequency
2.40	Nov 14, 2014	—	Added support for the RX113 Group.
2.41	Dec 4, 2014	5	Added Code Size section.
2.50	Mar 10, 2015	—	Added support for the RX71M Group.
2.51	Mar 10, 2015	3	Fixed a bug in cmt_isr_common which passed the CMT chnl number instead of a pointer to the CMT chnl number to the callback function. Updated section 1.2.2 regarding pointer cast as uint32_t.
2.60	June 30, 2015	—	Added support for the RX231 Group.
2.70	Sep 30, 2015	—	Added support for the RX23T Group.
		5	Updated the ROM size for 4 channels in 2.9 Code Size.
2.80	Oct 1, 2015	—	Added support for the RX130 Group.
		5	Updated the ROM size for 2 channels in 2.9 Code Size.
2.90	Dec 1, 2015	—	Added support for the RX230 and the RX24T Groups.
		1, 5	Changed the document number for the “Board Support Package Firmware Integration Technology Module” application note.
		4	Changed the description in section 2.
		5	Updated the Code Size table for the RX230 and the RX24T Groups.
		11	Changed description for “false” in Return Values. In Description, deleted the description regarding a call for a CMT channel not in operation.
		12	Added “4. Demo Projects”.
2.91	June 15, 2016	12	Added RSKRX64M to “4. Demo Projects”.
		13	Added “Related Technical Updates”.
3.00	Oct 1, 2016	—	Added support for the RX65N Group
		5	Changed the tabular format of Code Size. Updated the Code Size table for the RX65N Group.
		7, 9, 10, 11,12	Added a description of API function sample code.
3.10	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”.
		Program	Modified the timing to stop the timer to fix the following issue: The compare match counter is operating alone even if the CMT is stopped if an interrupt occurs at an unexpected timing during one-shot timer operation
			Modified the register setting sequence when stopping the timer with the R_CMT_Stop function.
			Modified the register setting sequence when disabling an interrupt with the R_CMT_Stop function.
			Fixed the following issue: When all channels are used, an error occurs if attempting to respecify the one-shot timer in the callback function for the

			one-shot timer.
3.20	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”.
		7	Updated “2.13 Adding the FIT Module to Your Project”.

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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

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

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the 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 disputes 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, drawing, chart, program, algorithm, 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 otherwise misappropriate 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, copy or otherwise misappropriation of Renesas Electronics products.
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; and 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.

   Renesas Electronics products are neither intended nor 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 damages (space and 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 third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the 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 radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond 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. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate 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. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use 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 not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
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.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.3.0-1 November 2016)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0