

RX Family

R01AN1856EU0230

Rev.2.30

CMT Module Using Firmware Integration Technology

April 21, 2014

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**
- **RX210 Group**
- **RX63N, RX631 Groups**
- **RX64M 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 (R01AN1833EU0100)
- Board Support Package Firmware Integration Technology Module (R01AN1685EU0250)
- Adding Firmware Integration Technology Modules to Projects (R01AN1723EU0110)

Contents

1. Overview	3
1.1 Using the FIT CMT module	3
1.2 Callback Functions	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 Header Files	4
2.7 Integer Types	4
2.8 Configuration Overview.....	4
2.9 API Data Structures	5
2.10 Return Values	5
2.11 Adding the Module to Your Project	5
3. API Functions	6
3.1 Summary	6
3.2 R_CMT_CreatePeriodic	7
3.3 R_CMT_CreateOneShot.....	8
3.4 R_CMT_Stop().....	9
3.5 R_CMT_Control	10
3.6 R_CMT_GetVersion()	11
Website and Support.....	12
Revision History	1
General Precautions in the Handling of MPU/MCU Products	2

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. There are too many variables

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 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 will be passed as an integer constant in the range of 0-3, so it can be cast to any integer type when you dereference the pointer.

Example:

```
void my_cmt_callback(void * pdata)
{
    uint8_t  cmt_event_channel_number;

    cmt_event_channel_number = *((uint8_t *)pdata); //cast pointer to uint8_t
    ...
}
```

2. API Information

This Driver API follows the Renesas API naming standards.

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 v1.00 and later

2.6 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"

2.7 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*.

2.8 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 <i>r_cmt_rx_config.h</i>		
CMT_RX_CFG_IPR	(5)	Interrupt priority level used for CMT interrupts

Table 1 : List of CMT module configuration options

2.9 API Data Structures

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

2.9.1 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_if.h*.

2.10 Return Values

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

2.11 Adding the Module to Your Project

The driver must be added to an existing e2Studio project. It is best to use the e2Studio FIT plugin to add the driver to your project as that will automatically update the include file paths for you. Alternatively, the driver can be imported from the archive that accompanies this application note and manually added by following these steps:

1. This application note is distributed with a zip file package that includes the FIT CMT support module in its own folder *r_cmt_rx*.
2. Unzip the package into the location of your choice.
3. In a file browser window, browse to the directory where you unzipped the distribution package and locate the *r_cmt_rx* folder
4. Open your e2Studio workspace.
5. In the e2Studio project explorer window, select the project that you want to add the CMT module to.
6. Drag and drop the *r_cmt_rx* folder from the browser window (or copy/paste) into your e2Studio project at the top level of the project.
7. Update the source search/include paths for your project by adding the paths to the module files:
 - a. Navigate to the "Add directory path" control:
 - i. 'project name'->properties->C/C++ Build->Settings->Compiler->Source -Add (green + icon)
 - b. Add the following paths:

```
"${workspace_loc}/${ProjName}/r_cmt_rx"
```



```
"${workspace_loc}/${ProjName}/r_cmt_rx/src"
```

Whether you used the plug-in or manually added the package to your project, it is necessary to configure the driver for your application.

8. Locate the *r_cmt_rx_config_reference.h* file in the *r_cmt_rx/ref/* source folder in your project and copy it to your project's *r_config* folder.
9. Change the name of the copy in the *r_config* folder to *r_cmt_rx_config.h*
10. Make the required configuration settings by editing the copied *r_cmt_rx_config.h* file. See Configuration Overview.

The CMT support module uses the *r_bsp* package for certain MCU information and support functions. The *r_bsp* package is easily configured through the *platform.h* header file which is located in the *r_bsp* folder. To configure the *r_bsp* package, open up *platform.h* and uncomment the #include for the board you are using. For example, to run the demo on a RSKRX111 board, the user would uncomment the #include for `'./board/rskrx111/r_bsp.h'` macro and make sure all other board #includes are commented out.

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.

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 power 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 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 preassign 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

This function is not designed for reentrant operation on the same CMT channel; however it can be reentered for a different channel.

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 power up and starts the timer. The timer is shut down after the first interrupt and callback.

Format

```
bool R_CMT_OneShot( 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 preassign 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

This function is not designed for reentrant operation on the same CMT channel; however it can be reentered for a different channel.

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:

The CMT channel could not be closed or it was not open

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. If this function is called for a CMT channel that is not in the open state then an error code is returned.

Reentrant

Yes.

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.

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 two bytes are the major version number and the bottom two 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;  
  
sprintf(version_str, "CMT v%1.1hu.%2.2hu", version_high, version_low);
```

Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

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

All trademarks and registered trademarks are the property of their respective owners.

Revision History

Rev.	Date	Description	
		Page	Summary
2.00	Nov 6, 2013	—	First GSCE Release
2.10	Nov 15, 2013	---	Formula for CMCOR value corrected.
2.30	April 12, 2014	1	Updated to indicate support for additional MCUs
		2	Added section 1.2 on callback functions.
		7	Added notes on maximum periodic frequency.

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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 an MPU or MCU 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
 2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
 3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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 systems; anti-disaster systems; anti-crime systems; and safety 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 (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable 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. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages 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 its 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 be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, 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 use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
 9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
 11. This document may not be 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, or if you have any other inquiries.
- (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.



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

1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

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 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

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 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jin Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.

12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141