To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. 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.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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.
5. When exporting the 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. You should not use Renesas Electronics products or the 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. 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.
6. 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.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. 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 categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "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.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. 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.
9. 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 system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. 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.
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.

User's Manual

Renesas Embedded Application Programming Interface

Reference Manual

Renesas Electronics
www.renesas.com

Rev.1.12 2009.05

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.

REVISION HISTORY

Rev.	Summary	Date
1.00		06.04.07
1.01	・ Additions and corrections made for support of libraries for the H8/36094, H8/36077, H8/36109, and R8C/24 and 25 ・ Unnecessary periods in Chapter 1, “Instruction,” deleted ・ Port P73 of H8 deleted ・ Channels A–D written in CreateInputCapture corrected to input captures A–D ・ Error in writing “seroal” corrected to “serial” ・ Erroneous description of RAPI_70_STATES corrected ・ Error in writing of timer channel names corrected ・ Error in writing of input capture mode corrected ・ Error in writing of output compare mode corrected ・ Erroneous description of RAPI_OVERFLOW_BIT15 corrected ・ Support for R8C UART1 clock synchronous mode ・ RAPI_COUNT_CLEAR added to first arguments for the R8C and H8/300H in __CreateInputCapture and __CreateOutputCompare ・ “External signal” deleted from the timer RD count source for the R8C in __CreateInputCapture and __CreateOutputCompare ・ Following statements added to the item for timer RD in __SetTimerRegister and __GetTimerRegister ・ [15]: Timer RD output master enable register 1 ・ [16]: Timer RD output master enable register 2 ・ [17]: Timer RD output control register ・ Explanation for clearing H8/300H module added to the description of __Create..., __Open..., and __BasicOpen... ・ Explanation for setting H8/300H module for standby added to the description of __Destroy..., __Close..., and BasicClose... ・ Return value in __PollingSerialReceiving changed from Boolean to unsigned int, and explanation of return value and example program changed ・ Return value in __PollingSerialSending changed from Boolean to unsigned int, and explanation of return value and example program changed ・ Remarks regarding use of port B on H8/300H added to __ReadIOPort and __ReadIOPortRegister	07.02.16

	・ Description of third argument in __CreateInputCapture corrected ・ Wording “specified” unified to “specified” ・ RAPI_BOTH deleted from the description of timer B1 count edges of H8/300H in __CreateEventCounter ・ RAPI_FTIOA and RAPI_FTIOB added to __CreatePulsePeriodMeasurementMode and __CreatePulseWidthMeasurementMode ・ All occurrences of the word “Tiny” deleted ・ Description of second argument added to __GetPulsePeriodMeasurementMode, __GetPulseWidthMeasurementMode, and __GetEventCounter ・ Description regarding timer V trigger of H8 in __CreateTimer, __CreateEventCounter, and __CreatePulseWidthMeasurementMode corrected ・ Explanation of H8 interrupt settings and interrupt control register settings in __SetSerialInterrupt corrected ・ Remarks on timer V of H8/300H in __EnableTimerRegister added ・ Specification of timer RA input pins of R8C in __CreatePulsePeriodMeasurementMode and __CreatePulseWidthMeasurementMode as well as RAPI_TIOSEL_P1_7 and RAPI_TIOSEL_P1_5 added ・ File name in program example corrected ・ Item “Reference” in __BasicSetSerialFormat deleted ・ __BasicSetSerialFormat added to “Reference” in __SetSerialFormat and __SetSerialInterrupt ・ Error in writing of __CreateInput Capture and __CreateOutputCapture corrected ・ Explanation added to Section 2.1, “Overview” ・ Causes of clearing of timer W and timer RC counters in __CreateInputCapture deleted ・ RAPI_COMPARE_MATCH_A_STOP and RAPI_STOP added to the item for the R8C in __CreateOutputCompare ・ Changed to RAPI_INT_LV_0 and RAPI_INT_LV_1 for H8/300H in __SetSerialInterrupt ・ Description relating to clock in __CreateTimer corrected ・ RAPI_TRC_FILTER and RAPI_TRD_FILTER added to filter specification for H8 in __CreatePulsePeriodMeasurementMode and __CreatePulseWidthMeasurementMode ・ RAPI_TIMER_RD2 and RAPI_TIMER_RD3 added to timer RD of H8/36109 ・ Description relating to timer V trigger input for H8/300H in __CreateTimer deleted ・ Description of pulse output function added to the item for R8C in __CreateEventCounter ・ Description of timer RE usage added to items data3 and data5 for R8C in __CreateOutputCompare ・ Filter function of timer RC and timer RD in __CreatePulsePeriodMeasurementMode, __CreatePulseWidthMeasurementMode, and __CreateInputCapture corrected ・ Description that multiple defined values can be set for data1 in __EnableInterrupt added ・ Description that RAPI_WITHOUT_SAMPLE_HOLD is specifiable in delay trigger modes 0 and 1 of M16C in __CreateADC corrected ・ Description that RAPI_FOCOF is specifiable in repeat mode of R8C in __CreateADC corrected	

	・ RAPI_AN30, RAPI_AN31, RAPI_AN32, and RAPI_P9_GROUP added to the item for data1 of M16C in __CreateADC ・ RAPI_AN30, RAPI_AN31, RAPI_AN32, and RAPI_P9_GROUP added to the item for data1 of M16C in __EnableADC ・ Error in writing of H8/300H interrupt set values in __CreateOutputCompare corrected ・ Error in writing of timer RD and timer RD symbol name corrected ・ Description of gate function of M16C in __CreateEventCounter deleted ・ Description of H8/300H noise rejection function in __BasicSetSerialFormat added ・ Description of timer W input pins of H8/300H in __CreatePulsePeriodMeasurementMode and __CreatePulseWidthMeasurementMode deleted	
1.02	・ Additions and corrections made for support of libraries for the R8C/26, 27, 28, 29, 2A ,2B, 2C and 2D	07.02.26
1.03	・ “Renesas API” corrected to “Renesas Embedded API” ・ “Renesas Embedded” written in “4.1 API List by Peripheral Facility” corrected to “Renesas Embedded API” ・ Program examples in __BasicOpenSerialDriver, __BasicCloseSerialDriver, __OpenSerialDriver and __CloseSerialDriver corrected. ・ Error in writing of Synopsis, Remark and Program example corrected in __BasicStartSerialSending. ・ Error in writing of Synopsis corrected in __BasicStopSerialReceiving. ・ RAPI_TG_TAIS changed from RAPI_TG_TAIIN the default value of Count start condition of M16C in __CreatePulseWidthModulationMode. ・ The default value of Usesing key input interrupt deleted in M16C of __SetInterrupt. ・ “Resolution” of R8C in __CreateADC added.	07.11.30
1.04	・ Additions and corrections made for support of libraries for the H8S/20103 and 20203 ・ Error in writing of R8C/2C and 2D in __GetADCAll. ・ Description of data2 for timer RC of R8C in __GetInputCapture corrected. ・ Description of The default value of count source and output function for timer RE of R8C in __CreateOutputCompare.	08.03.03
1.05	・ RAPI_TMR_TRIGGER of H8S deleted in __CreateADC. ・ Description of interrupt request in repeat mode of H8S deleted in __CreateADC. ・ Description of Input pin of H8S corrected in __CreateADC. ・ Description of RAPI_TIMER_ON and RAPI_TIMER_OFF of H8S corrected in __CreateADC. ・ Description of data3 for rimer RG of H8S corrected in __CreateOutputCompare. ・ Description of gate facility of H8S deleted in __CreateTimer. ・ Description of gate facility of H8S added in __CreateEventCounter, __CreatePulsePeriodMeasurementMode and __CreatePulseWidthMeasurementMode. ・ Error in writing “RAPI_BCSS_F32” corrected to “RAPI_BCSS_F64” for H8 and H8S in __BasicSetSerialFormat. ・ Error in writing “φ/40” corrected to “φ40” for H8S in all Timer function. ・ Error in writing of H8S output function and filter corrected. ・ Description of RAPI_H_L_L and RAPI_L_H_H of H8S added in __CreatePulseWidthModulationMode.	08.05.08
1.06	・ __DestroyADCUnit, __GetADCUnit __GetADCUnitStatus and __ClearADCUnitStatu	08.06.18

	added. ・ Description of R8C/2C and 2D corrected in __GetADC and __GetADCAll. ・ __ConfigurePMC added. ・ __SetEventSequence, __CreateEventGenerateTimer, __EnableEventGenerateTimer, __DestroyEventGenerateTimer, __WritePortBufferRegister and __ReadPortBufferRegister added. ・ 330 states and 166 states of Conversion time of H8S deleted in __CreateADC.	
1.07	・ Header file name of Program examples corrected in all PMC and ELC function. ・ Error in writing "__SetEventSequence" corrected to "__SetEventLink". ・ Description of header files for PMC and ELC driver APIs added in "2.1 Overview". ・ Error in writing "Interrupt_*xxx_yyy*.h" corrected to "Interrupt_*xxx_yyy*.c" in "2.1 Overview". ・ Description and value of unit of H8S added in __CreateADC. ・ RAPI_ADTRG2_TRIGGER of H8S added in __CreateADC. ・ Description and value of unit of H8S added in __EnableADC. ・ Description and value of unit of H8S added in __GetADC. ・ Unit Name of timer RD of H8S corrected. ・ Description of Interrupt control mode of H8S corrected __CreateInputCaptur. ・ Description and value corrected in all PMC functions. ・ Description and value corrected in all ELC functions.	08.07.14
1.08	・ Description of __SetEventLink added.	08.07.18
1.09	・ Number of Timer ELC event corrected.	08.07.29
1.10	・ Number of ELC port-group corrected. (port-group 2 corrected to port-group 1, port-group 3 corrected to port-group 2)	08.11.10
1.11	・ __CreateDTC, __EnableDTC, __DisableDTC, __EnableSWDTC.	08.12.15
1.12	・ Description of Specifing the I/O port added in __ConfigurePMC.	08.03.06

Table of Contents

1. Introduction

The Renesas Embedded Application Programming Interface (API) is a unified API for the microcomputers made by Renesas Technology Corporation.

2. Driver

2.1 Overview

The library described herein provides a peripheral facility control program (peripheral driver) for microcomputers. Use of the Renesas Embedded API permits the peripheral driver to be built into a user program.

Configuration of Renesas Embedded APIs is shown below.

File name	Description
rapi_*xxx_yyy*.lib (*xxx* = family name, *yyy* = series name)	This is the Renesas Embedded API library file. To use Renesas Embedded APIs, specify this file as input file for the linker.
rapi_ad_*xxx_yyy*.h (*xxx* = family name, *yyy* = series name)	This is the header file for A/D converter driver APIs. To use A/D converter driver APIs, be sure to include this file.
rapi_io_port_*xxx_yyy*.h (*xxx* = family name, *yyy*= series name)	This is the header file for I/O port driver APIs. To use I/O port driver APIs, be sure to include this file.
rapi_sif_*xxx_yyy*.h (*xxx* = family name, *yyy*= series name)	This is the header file for serial I/F driver APIs. To use serial I/F driver APIs, be sure to include this file.
rapi_timer_*xxx_yyy*.h (*xxx* = family name, *yyy*= series name)	This is the header file for timer driver APIs. To use timer driver APIs, be sure to include this file.
rapi_pmc_*xxx_yyy*.h (*xxx* = family name, *yyy*= series name)	This is the header file for PMC driver APIs. To use PMC driver APIs, be sure to include this file.
rapi_elc_*xxx_yyy*.h (*xxx* = family name, *yyy*= series name)	This is the header file for ELC driver APIs. To use ELC driver APIs, be sure to include this file
rapi_dtc_*xxx_yyy*.h (*xxx* = family name, *yyy*= series name)	This is the header file for DTC driver APIs. To use DTC driver APIs, be sure to include this file
rapi_io_*xxx_yyy*.h (*xxx* = family name, *yyy*= series name)	This is the CPU control register definition header file for Renesas Embedded APIs.
Interrupt_*xxx_yyy*.h (*xxx* = family name, *yyy*= series name)	This is the interrupt function source file for Renesas Embedded APIs. To use Renesas Embedded APIs that use interrupts, add this file to the user program.

2.2 Driver Features

The library described herein has the following features available as the peripheral driver.

(1) Serial I/O control feature

It comprises a serial interface driver, which sets or clears the conditions of serial communication, as well as controls and manages the transmission/reception of communication data.

(2) Timer control feature

It comprises a timer driver, which sets or clears the operating conditions of timers, as well as controls the timer operation.

(3) I/O port control feature

It comprises an I/O port driver, which sets or clears the usage conditions of I/O ports, as well as control data read/write operation.

(4) External interrupt control feature

It comprises an external interrupt driver, which sets or clears the usage conditions of external interrupts, as well as controls interrupt operation.

(5) A/D converter control feature

It comprises an A/D converter driver, which sets or clears the usage conditions of A/D converters, as well as controls A/D converter operation.

(6) Peripheral I/O mapping controller feature

The peripheral-facility mapping controller driver assigns pin functions.

(7) Event link controller feature

The event link controller driver assigns event signals to be linked to peripheral modules, sets or clears operating conditions of an event timer, and controls the timer and reading from and writing to the port-buffer register.

(8) Data transfer controller feature

The data transfer controller driver allows a transfer of data.

2.3 Serial Interface Driver

The serial interface driver sets serial communication, clears settings, transmit/receives data, and controls the status of serial communication.

There are two kinds of serial interface driver: a single-data transmission/reception API and a multi-data transmission/reception API.

2.4 Timer Driver

The timer driver sets the timer, clears timer settings, controls timer operation, and acquires a counter value with respect to the following modes:

- Timer mode
- Event counter mode
- Pulse width modulation mode (PWM mode)
- Pulse period measurement mode
- Pulse width measurement mode
- Input capture mode
- Output compare mode

2.4.1 Timer Mode

In this mode, the timer counts the internally generated count source. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.2 Event Counter Mode

In this mode, the timer counts the external signal fed in from an input pin or an overflow or underflow from other timer. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.3 Pulse Width Modulation Mode (PWM Mode)

In this mode, the timer outputs pulses in a given width successively. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.4 Pulse Period Measurement Mode

In this mode, the timer measures the pulse period of an external signal fed in from an input pin. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.5 Pulse Width Measurement Mode

In this mode, the timer measures the pulse width of an external signal fed in from an input pin. When an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.6 Input Capture Mode

In this mode, the timer latches the timer value upon an active signal edge or clock pulse at an input pin, thereby generating an interrupt request. When an input capture interrupt or an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.4.7 Output Compare Mode

In this mode, the timer generates an interrupt request when the timer counter and a comparison value match. When a compare match interrupt or an underflow or an overflow interrupt occurs, it calls a preset callback function.

2.5 I/O Port Driver

The I/O port driver sets the I/O port for input or output, writes data to the I/O port, and reads data from the I/O port.

2.6 External Interrupt Driver

The external interrupt driver sets external interrupts, controls external interrupts, acquires the status of external interrupt flags, and clears external interrupt flags.

2.7 A/D Converter Driver

The A/D converter driver sets the A/D converter, controls the A/D converter, clears settings of the A/D converter, acquires the A/D converter value, acquires the status of the A/D converter, and clears the status of the A/D converter.

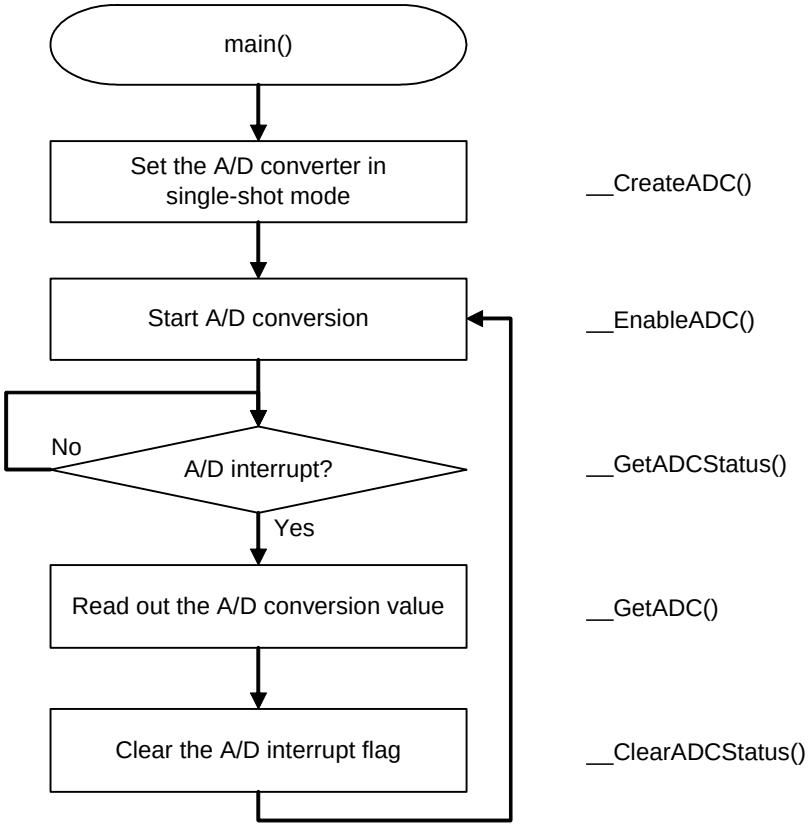
[A/D converter driver usage example (single-shot mode)]

Here, a program example is shown for the A/D converter operating in single-shot mode under the conditions given below.

CPU	:	R8C
Operating clock	:	fAD divided by 2
Resolution	:	10 bits
Analog input pin	:	AN0 pin
A/D conversion start condition	:	Software trigger
Sample-and-hold	:	Enabled

A program flow and a program example are shown below.

(Program flow)



Program example

```
#include "rapi_ad_r8c_13.h"

void main( void )
{
   unsigned int  status, data;

   /* Set up A/D converter as one short mode */
   __CreateADC(                    RAPI_ONE_SHOT|RAPI_AN0|RAPI_FAD2|
RAPI_WITH_SAMPLE_HOLD|
               RAPI_AD_OFF|RAPI_10BIT, 1, 0, 0 );

   while( 1 ){
      /* Disable A/D converter */
      __EnableADC( RAPI_AN0| RAPI_AD_ON, 1 );

      /* Check a flag bit of A/D converter interrupt */
      do{
         __GetADCStatus( &status );
      } while( (*status & 0x0001) == 0 )

      /* Get A/D convertered datas of A/D register */
      __GetADC( &data );

      /* Clear status of A/D convertered */
      __ClearADCStatus( 0 );
   }
}
```

2.8 Peripheral I/O Mapping Controller Driver

The Peripheral I/O Mapping Controller driver assigns pin functions.

2.9 Event Link Controller Driver

The Event Link Controller Controller driver assigns event signals to be linked to peripheral modules, sets or clears operating conditions of an event timer, and controls the timer and reading from and writing to the port-buffer register.

2.10 Data Transfer Controller

The Data Transfer Controller driver allows a transfer of data.

3. Standard Types

This section describes the standard types defined in the library. For details about the set values, refer to the description of each API.

Standard type	Description
Boolean	The Boolean type represents the enum-type data that indicates whether successful (RAPI_TRUE (= 1)) or failed (RAPI_FALSE (= 0)).
VoidFuncNotify	The VoidFuncNotify type represents the type of the notification function to be registered.

4. Library Reference

4.1 API List by Peripheral Facility

The table below lists the Renesas Embedded APIs classified by peripheral facility.

NO	Facility classification	API	API operation
1	Single-data serial I/O	R_BasicOpenSerialDriver	Opens serial port
2		R_BasicCloseSerialDriver	Closes serial port
3		R_BasicSetSerialFormat	Sets serial communication
4		R_BasicStartSerialReceiving	Receives 1 data
5		R_BasicStartSerialSending	Transmits 1 data
6		R_BasicReceivingStatusRead	Reads receive status
7		R_BasicSendingStatusRead	Reads transmit status
8		R_BasicStopSerialReceiving	Stops reception
9		R_BasicStopSerialSending	Stops transmission
10	Multi-data serial I/O	R_OpenSerialDriver	Opens serial port
11		R_CloseSerialDriver	Closes serial port
12		R_ConfigSerialDriverNotify	Registers notification function
13		R_SetSerialFormat	Sets serial communication
14		R_SetSerialInterrupt	Sets transmit/receive interrupt
15		R_StartSerialReceiving	Starts reception
16		R_StartSerialSending	Starts transmission
17		R_StopSerialReceiving	Stops reception
18		R_StopSerialSending	Stops transmission
19		R_PollingSerialReceiving	Receives by polling
20		R_PollingSerialSending	Transmits by polling
21	Timer	R_CreateTimer	Sets timer mode
22		R_EnableTimer	Controls timer mode operation
23		R_DestroyTimer	Clears timer mode setting
24		R_CreateEventCounter	Sets event counter mode
25		R_EnableEventCounter	Controls operation of event counter mode
26		R_DestroyEventCounter	Clears setting of event counter mode
27		R_GetEventCounter	Gets event counter mode counter value
28		R_CreatePulseWidthModulationMode	Sets pulse width modulation mode
29		R_EnablePulseWidthModulationMode	Controls operation of pulse width modulation mode
30		R_DestroyPulseWidthModulationMode	Clears setting of pulse width modulation mode
31		R_CreatePulsePeriodMeasurementMode	Sets pulse period measurement mode
32		R_EnablePulsePeriodMeasurementMode	Controls operation of pulse period measurement mode
33		R_DestroyPulsePeriodMeasurementMode	Clears setting of pulse width measurement mode
34		R_GetPulsePeriodMeasurementMode	Acquires measured value of pulse period measurement mode

35		__CreatePulseWidthMeasurementMode	Sets pulse width measurement mode
36		__EnablePulseWidthMeasurementMode	Controls operation of pulse width measurement mode
37	Timer	__DestroyPulseWidthMeasurementMode	Clears setting of pulse width measurement mode
38		__GetPulseWidthMeasurementMode	Acquires measured value of pulse width measurement mode
39		__CreateInputCapture	Sets input capture mode
40		__EnableInputCapture	Controls operation of input capture mode
41		__DestroyInputCapture	Clears setting of input capture mode
42		__GetInputCapture	Acquires counter value of input capture mode
43		__CreateOutputCompare	Sets output compare mode
44		__EnableOutputCompare	Controls operation of output compare mode
45		__DestroyOutputCompare	Clears setting of output compare mode
46		__SetTimerRegister	Sets timer register
47		__EnableTimerRegister	Controls operation of timer register
48		__ClearTimerRegister	Clears timer register
49		__GetTimerRegister	Gets timer register value
50	I/O port	__SetIOPort	Sets I/O port
51		__ReadIOPort	Reads from I/O port
52		__WriteIOPort	Writes to I/O port
53		__SetIOPortRegister	Sets I/O port register
54		__ReadIOPortRegister	Reads from I/O port register
55		__WriteIOPortRegister	Writes to I/O port register
56	External interrupt	__SetInterrupt	Sets external interrupt
57		__EnableInterrupt	Controls external interrupt
58		__GetInterruptFlag	Gets flag status of external interrupt
59		__ClearInterruptFlag	Clears flag of external interrupt
60	A/D converter	__CreateADC	Sets A/D converter
61		__EnableADC	Controls operation of A/D converter
62		__DestroyADC	Discards settings of A/D converter
63		__DestroyADCUnit	Discards settings of A/D converter (specified unit)
64		__GetADC	Gets A/D conversion value (register specified)
65		__GetADCAll	Gets A/D conversion value (all registers)
66		__GetADCUnit	Gets A/D conversion value (specified unit)
67		__GetADCStatus	Gets status of A/D converter
68		__GetADCUnitStatus	Gets status of A/D converter (specified unit)
69		__ClearADCStatus	Clears status of A/D converter
70		__ClearADCUnitStatus	Clears status of A/D converter (specified unit)

71	Peripheral I/O Mapping Controller	__ConfigurePMC	Assigns pin functions
72	Event Link Controller	__SetEventLink	Assigns event signals to be linked to peripheral modules
73		__CreateEventGenerateTimer	Sets event-generation timer
74		__EnableEventGenerateTimer	Controls event-generation timer
75		__DestroyEventGenerateTimer	Discards settings of event-generation timer
76		__ReadPortBufferRegister	Reads from port buffer registers
77		__WritePortBufferRegister	Writes to port buffer registers
78	Data Transfer Controller	__CreateDTC	Sets DTC register information
79		__EnableDTC	Sets DTC activation Sources
80		__DisableDTC	Sets DTC activation Sources disabled
81		__EnableSWDTC	Sets DTC Software activation

4.2 Description of Each API

This section describes each API and explains how to use them, showing a program example for each.

The description of each API is divided into the following items.

- **Synopsis** : Outlines the content of processing performed by the function. It also shows the syntax of the function, followed by a brief explanation of arguments.
- **Description** : Describes the function and how to use it in detail.
- **Return value** : Explains the returned value of the function.
- **Functionality** : Indicates the functional classification of the function.
- **Reference** : Indicates the related functions.
- **Remark** : Describes the precautions to be taken when using the API.
- **Program example** : Presents a program showing how to use the function.

4.2.1 Serial I/O
__BasicOpenSerialDriver

Synopsis

<Open a serial port>

Boolean __BasicOpenSerialDriver(unsigned long data)

data	Setup data

Description

Opens and initializes a specified serial port.

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S)(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Serial I/O

Reference

__BasicCloseSerialDriver

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- When used for the H8S, H8/300H, this API opens and initializes a specified serial port when freeing it from module standby state.

Program example

```
#include ”rapi_sif_r8c_13.h”

void func( void )
{
    ..........
    /* Open serial driver */
    __BasicOpenSerialDriver( RAPI_COM1 );
    ..........
}
```

__BasicCloseSerialDriver

Synopsis

<Close a serial port>

Boolean __BasicCloseSerialDriver(unsigned long data)

data	Setup data

Description

Closes a specified serial port. For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S)(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Serial I/O

Reference

__BasicOpenSerialDriver

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- When used for the H8S, H8/300H, this API places a specified serial port into module standby state after closing it.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void func( void )
{
    ..........
    /* Close serial driver */
    Rapi_BasicCloseSerialDriver( RAPI_COM1 );
    ..........
}
```

__BasicSetSerialFormat

Synopsis

<Set serial communication>

Boolean __BasicSetSerialFormat(unsigned long data1, unsigned char data2)

data1	Setup data 1
data2	Setup data 2

Description

Sets serial communication according to specified parameters.

[data1]

For data1, the following values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

For serial communication mode, the following values can be set.

(M16C) (UART0, UART1, UART2)

RAPI_SM_SYNC	Clock synchronous serial communication mode
RAPI_SM_ASYNC	Clock asynchronous serial communication mode

(M16C)(SI/O3, SI/O4)

RAPI_SIO_SM_SYNC	Clock synchronous serial communication mode

For the data length format of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_BIT_7	Transfer data length 7 bits	RAPI_BIT_8	Transfer data length 8 bits
RAPI_BIT_9	Transfer data length 9 bits		

For the clock source of serial communication, the following values can be set.

(M16C) (UART0, UART1, UART2)

RAPI_CKDIR_INT	Internal clock is used as the clock source of serial communication.
RAPI_CKDIR_EXT	External clock is used as the clock source of serial communication.

(M16C) (SI/O3, SI/O4)

RAPI_SIO_CKDIR_INT	Internal clock is used as the clock source of serial communication.
RAPI_SIO_CKDIR_EXT	External clock is used as the clock source of serial communication.

For the stop bit length of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_STPB_1	1 stop bit	RAPI_STPB_2	2 stop bits

For the parity bit of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_PARITY_NON	No parity bit	RAPI_PARITY_EVEN	Even parity bit
RAPI_PARITY_ODD	Odd parity bit		

For the clock polarity of serial communication, the following values can be set.

If the API is used in clock asynchronous serial communication mode, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_DPOL_NON	Polarity not inverted	RAPI_DPOL_INV	Polarity inverted

(M16C) (SI/O3, SI/O4)

RAPI_SIO_DPOL_NON	Polarity not inverted	RAPI_SIO_DPOL_INV	Polarity inverted

For the count source of the built-in baud rate generator, the following values can be set.

(M16C) (UART0, UART1, UART2)

RAPI_BCSS_F1	f1SIO	RAPI_BCSS_F2	f2SIO
RAPI_BCSS_F8	f8SIO	RAPI_BCSS_F32	f32SIO

(M16C) (SI/O3, SI/O4)

RAPI_SIO_BCSS_F1	f1SIO	RAPI_SIO_BCSS_F2	f2SIO
RAPI_SIO_BCSS_F8	f8SIO	RAPI_SIO_BCSS_F32	f32SIO

For the _CTS/_RTS function, the following values can be set.

If the internal clock is selected for use in clock synchronous serial communication mode, the _RTS function has no effect.

(M16C) (UART0, UART1, UART2)

RAPI_CTSRTS_DIS	_CTS/_RTS functions are not used.
RAPI_CTS_SEL	_CTS function is selected.
RAPI_RTS_SEL	_RTS function is selected.

For the transfer format, the following values can be set.

If the data length selected for use in clock asynchronous serial communication mode is 7 or 9 bits long, do not set these values.

(M16C) (UART0, UART1, UART2)

RAPI_LSB_SEL	LSB first	RAPI_MSB_SEL	MSB first

(M16C) (SI/O3, SI/O4)

RAPI_SIO_LSB_SEL	LSB first	RAPI_SIO_MSB_SEL	MSB first

For serial data logic switchover, the following values can be set.

(M16C) (UART2)

RAPI_LOGIC_NO_REV	The value written in the transmit buffer register does not have its logic inverted.
RAPI_LOGIC_REV	The value written in the transmit buffer register is inverted before being transmitted.

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1

RAPI_COM3	UART2

For serial communication mode, the following values can be set.

(R8C) (UART0, UART1, UART2)

RAPI_SM_SYNC	Clock synchronous serial communication mode
RAPI_SM_ASYNC	Clock asynchronous serial communication mode

For the data length format of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(R8C) (UART0, UART1, UART2)

RAPI_BIT_7	Transfer data length 7 bits	RAPI_BIT_8	Transfer data length 8 bits
RAPI_BIT_9	Transfer data length 9 bits		

For the clock source of serial communication, the following values can be set.

(R8C) (UART0, UART1, UART2)

RAPI_CKDIR_INT	Internal clock is used as the clock source of serial communication.
RAPI_CKDIR_EXT	External clock is used as the clock source of serial communication.

For the stop bit length of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(R8C) (UART0, UART1, UART2)

RAPI_STPB_1	1 stop bit	RAPI_STPB_2	2 stop bits

For the parity bit of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(R8C) (UART0, UART1, UART2)

RAPI_PARITY_NON	No parity bit	RAPI_PARITY_EVEN	Even parity bit
RAPI_PARITY_ODD	Odd parity bit		

For the clock polarity of serial communication, the following values can be set.

If the API is used in clock asynchronous serial communication mode, do not set these values.

(R8C) (UART0, UART1, UART2)

RAPI_DPOL_NON	Polarity not inverted	RAPI_DPOL_INV	Polarity inverted

For the count source of the built-in baud rate generator, the following values can be set.

(R8C) (UART0, UART1, UART2)

RAPI_BCSS_F1	f1SIO	RAPI_BCSS_F8	f8SIO
RAPI_BCSS_F32	f32SIO		

For the transfer format, the following values can be set.

If the data length selected for use in clock asynchronous serial communication mode is 7 or 9 bits long, do not set these values.

(R8C) (UART0, UART1, UART2)

RAPI_LSB_SEL	LSB first	RAPI_MSB_SEL	MSB first

For the pins used for serial communications, the following values can be set.

(R8C) (UART1)

RAPI_TX_P00_RX_P37	TXD1 is P0_0 and RXD1 is P3_7
RAPI_TX_P37_RX_P45	TXD1 is P3_7 and RXD1 is P4_5
RAPI_TX_P00_RX_P36	TXD1 is P0_0 and RXD1 is P3_6
RAPI_CLK1_P0_5	CLK1 is P0_5
RAPI_CLK1_P6_5	CLK1 is P6_5

(H8S)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

For serial communication mode, the following values can be set.

(H8S) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_SM_SYNC	Clock synchronous serial communication mode
RAPI_SM_ASYNC	Clock asynchronous serial communication mode

For the data length format of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(H8S) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_BIT_7	Transfer data length 7 bits	RAPI_BIT_8	Transfer data length 8 bits

For the clock source of serial communication, the following values can be set.

(H8S) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_CKDIR_INT	Internal clock is used as the clock source of serial communication.
RAPI_CKDIR_EXT	External clock is used as the clock source of serial communication.

For the stop bit length of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(H8S) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_STPB_1	1 stop bit	RAPI_STPB_2	2 stop bits

For the parity bit of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(H8S) (SCI3 channel 1, SCI3 channel 2 SCI3 channel 3)

RAPI_PARITY_NON	No parity bit	RAPI_PARITY_EVEN	Even parity bit
RAPI_PARITY_ODD	Odd parity bit		

For the count source of the built-in baud rate generator, the following values can be set.

(H8S) (SCI3 channel 1, SCI3 channel 2 SCI3 channel 3)

RAPI_BCSS_F1	Φ clock	RAPI_BCSS_F4	φ/4 clock
RAPI_BCSS_F16	φ/16 clock	RAPI_BCSS_F32	φ/64 clock

The noise rejection function can be set to be turned on or turned off.

When used in clock synchronous serial communication mode, the noise rejection function has no effect.

(H8S) (SCI3 channel 1, SCI3 channel 2 SCI3 channel 3)

RAPI_NOISE_CANCEL_ON	Noise rejection function turned on
RAPI_NOISE_CANCEL_OFF	Noise rejection function turned off

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

For serial communication mode, the following values can be set.

(H8/300H) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_SM_SYNC	Clock synchronous serial communication mode
RAPI_SM_ASYNC	Clock asynchronous serial communication mode

For the data length format of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(H8/300H) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_BIT_7	Transfer data length 7 bits	RAPI_BIT_8	Transfer data length 8 bits

For the clock source of serial communication, the following values can be set.

(H8/300H) (SCI3 channel 1, SCI3 channel 2)

RAPI_CKDIR_INT	Internal clock is used as the clock source of serial communication.
RAPI_CKDIR_EXT	External clock is used as the clock source of serial communication.

For the stop bit length of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(H8/300H) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_STPB_1	1 stop bit	RAPI_STPB_2	2 stop bits

For the parity bit of clock asynchronous serial communication, the following values can be set.

If the API is used in clock synchronous serial communication mode, do not set these values.

(H8/300H) (SCI3 channel 1, SCI3 channel 2, SCI3 channel 3)

RAPI_PARITY_NON	No parity bit	RAPI_PARITY_EVEN	Even parity bit
RAPI_PARITY_ODD	Odd parity bit		

For the count source of the built-in baud rate generator, the following values can be set.

(H8/300H) (SCI3 channel 1, SCI3 channel 2 SCI3 channel 3)

RAPI_BCSS_F1	Φ clock	RAPI_BCSS_F4	φ/4 clock
RAPI_BCSS_F16	φ/16 clock	RAPI_BCSS_F32	φ/64 clock

The noise rejection function can be set to be turned on or turned off.

When used in clock synchronous serial communication mode, the noise rejection function has no effect.

(H8/300H) (SCI3 channel 3)

RAPI_NOISE_CANCEL_ON	Noise rejection function turned on
RAPI_NOISE_CANCEL_OFF	Noise rejection function turned off

[data2]

Sets the divide-by-N value of a communication speed.

Return value

If serial communication was successfully set, RAPI_TRUE is returned; if settings failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

Boolean func( void )
{
   /* Set the data of RAPI_COM1 to serial driver */
   return _BasicSetSerialFormat(RAPI_COM1 | RAPI_SM_SYNC | RAPI_CKDIR_INT
|
                              RAPI_BCSS_F1 | RAPI_DPOL_NON | RAPI_LSB_SEL,
20);
}
```

__BasicStartSerialReceiving

Synopsis

<Receive 1 data>

Boolean __BasicStartSerialReceiving(unsigned long data)

data	Setup data

Description

Starts receiving 1 data of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S)(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If data reception in serial communication was successfully started, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__BasicReceivingStatusRead, __BasicStopSerialReceiving

Remark

- For the H8S, H8/300H, wait for at least a 1-bit period before calling this API after __BasicSetSerialFormat was called.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void func( void )
{
    ..........
    __BasicStartSerialReceiving( RAPI_COM1 );
    ..........
}
```

__BasicStartSerialSending

Synopsis

<Transmit 1 data>

Boolean _BasicStartSerialSending(unsigned long data1, unsigned int data2)

data	Setup data
data	Transmit data

Description

Starts sending 1 data of serial communication.

For data1, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SIC3 channel 3		

Return value

If data transmission in serial communication was successfully started, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__BasicSendingStatusRead, __BasicStopSerialSending

Remark

- For the H8S, H8/300H, wait for at least a 1-bit period before calling this API after __BasicSetSerialFormat was called.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void func( void )
{
    ..........
    __BasicStartSerialSending( RAPI_COM1, 0x00AA );
    ..........
}
```

__BasicReceivingStatusRead

Synopsis

<Read receive status>

unsigned int __BasicReceivingStatusRead(unsigned long data)

data	Setup data

Description

Returns the receive status of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

The receive status of serial communication is returned. The returned value is one of the following.

(M16C) (UART0, UART1, UART2)

RAPI_RX_INCOMPLETE	Reception not complete yet.
Other than above	Reception complete. The value read from the UARTi receive buffer register (i = 0 to 2).

(M16C) (SI/O3, SI/O4)

RAPI_RX_INCOMPLETE	Reception not complete yet.
Other than above	Reception complete. Low-order 8 bits: The value read from the SI/Oi transmit/receive register (i = 3, 4).

(R8C)

RAPI_RX_INCOMPLETE	Reception not complete yet.
Other than above	Reception complete. The value read from the UARTi receive buffer register (i = 0, 1).

(H8S) (H8/300H)

RAPI_RX_INCOMPLETE	Reception not complete yet.
Other than above	Reception complete. High-order 8 bits: The value read from the serial status register. Low-order 8 bits: The value read from the receive data register. (Not read if an error occurred.)

Functionality

Serial I/O

Reference

__BasicStartSerialReceiving, __BasicStopSerialReceiving

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void func( void )
{
   unsigned int rcv_data;

   ..........
   rcv_data = __BasicReceivingStatusRead( RAPI_COM1 );
   ..........
}
```

__BasicSendingStatusRead

Synopsis

<Read transmit status>

Boolean __BasicSendingStatusRead(unsigned long data)

data	Setup data

Description

Returns the transmit status of serial communication. For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If no data exists in the transmit buffer, RAPI_TRUE is returned; if data exists, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__BasicStartSerialSending, __BasicStopSerialSending

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_sif_r8c_13.h"

void func( void )
{
    ..........
    if (__BasicSendingStatusRead( RAPI_COM1 ) == RAPI_TRUE) {
        /* Transmission completion */
    }
    ..........
}
```

__BasicStopSerialReceiving

Synopsis

<Stop reception>

Boolean Rapi_BasicStopSerialReceiving(unsigned long data)

data	Setup data

Description

Stops receiving data in serial communication

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If data reception in serial communication was successfully stopped, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__BasicStartSerialReceiving

Remark

- For the M16C SI/03 and SI/04, this API cannot be used.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void func( void )
{
    ……
    /* Stop receiving data in serial communication */
    __BasicStopSerialReceiving ( RAPI_COM1 );
    ……
}
```

__BasicStopSerialSending

Synopsis

<Stop transmission>

Boolean __BasicStopSerialSending(unsigned long data)

data	Setup data

Description

Stops transmitting data in serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
SIO3_COM3	SIO3 channel 3		

Return value

If data transmission in serial communication was successfully stopped, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__BasicStartSerialSending

Remark

- For the M16C SI/03 and SI/04, this API cannot be used.
- The specifiable serial ports differ with each CPU used.
- When operating in clock synchronous serial communication mode, data reception is stopped at the same time by this API.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void func( void )
{
    ……
    /* Stop sending data in serial communication */
    __BasicStopSerialSending ( RAPI_COM1 );
    ……
}
```

__OpenSerialDriver

Synopsis

<Open a serial port>

Boolean __OpenSerialDriver(unsigned long data)

data	Setup data

Description

Opens and initializes a specified serial port.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
SIO3_COM3	SIO3 channel 3		

Return value

If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Serial I/O

Reference

__CloseSerialDriver

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- When used for the H8S, H8/300H, this API opens and initializes a specified serial port when freeing it from module standby state.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void func( void )
{
    ……
    /* Open serial driver */
    __OpenSerialDriver( RAPI_COM1 );
    ……
}
```

__CloseSerialDriver

Synopsis

<Close a serial port>

Boolean __CloseSerialDriver(unsigned long data)

data	Setup data

Description

Closes a specified serial port.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
SIO3_COM3	SIO3 channel 3		

Return value

If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Serial I/O

Reference

__OpenSerialDriver

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- When used for the H8S, H8/300H, this API places a specified serial port into module standby state after closing it.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void func( void )
{
    ……
    /* Close serial driver */
    __CloseSerialDriver( RAPI_COM1 );
    ……
}
```

__ConfigSerialDriverNotify

Synopsis

<Register a notification function>

Boolean __ConfigSerialDriverNotify(unsigned long data, VoidFuncNotify *func)

data	Setup data
func	Function pointer to be registered

Description

Registers the notification function necessary to get various transmit/receive information of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

[func]

The function to be registered in func must be supplied to the serial I/O driver by the user.

The serial I/O driver calls the function registered in func.

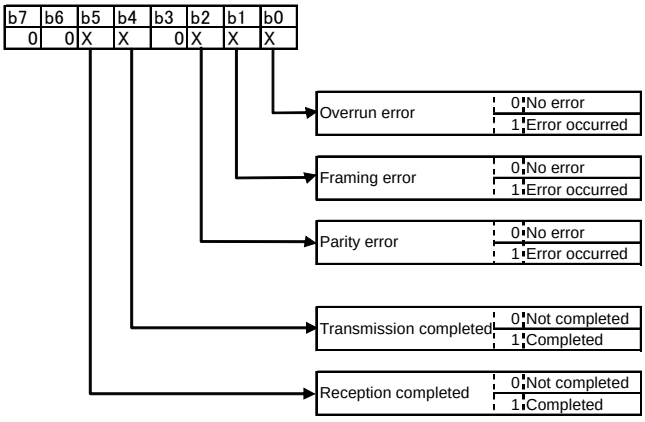
The serial I/O driver notifies the user of the transmit/receive status by an argument.

The type of the function to be registered is shown below.

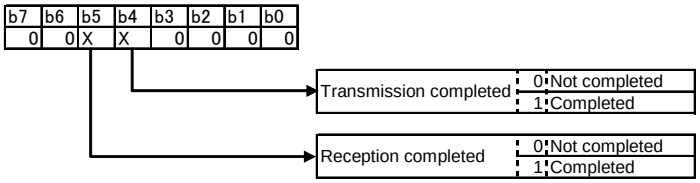
void “any function name” (unsigned char notify);

The argument is detailed below.

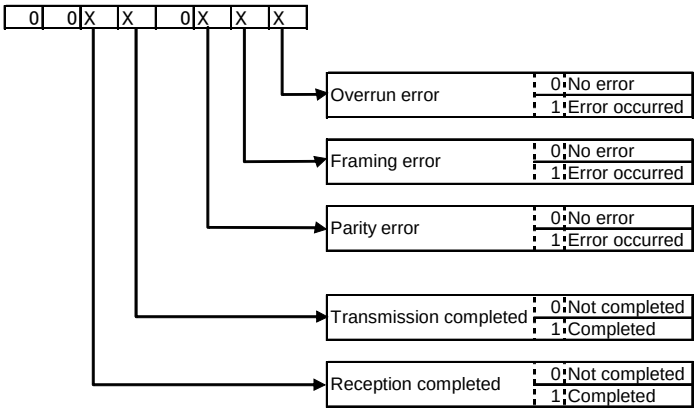
(M16C) (UART0, UART1, UART2)



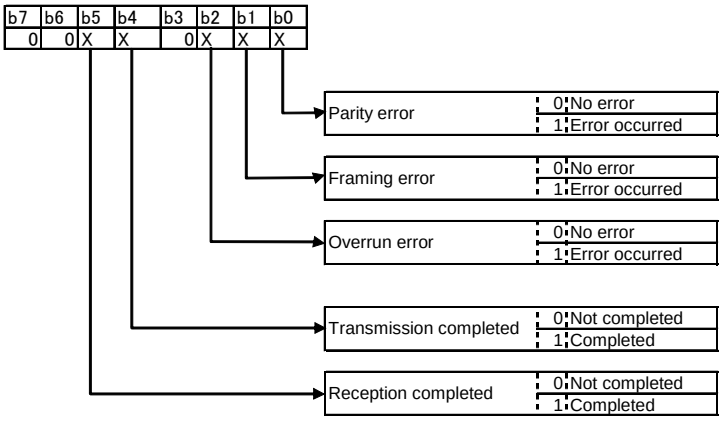
(M16C) (SI/O3,SI/O4)



(R8C)



(H8S) (H8/300H)



Return value	If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	Serial I/O
Reference	StartSerialReceiving, StartSerialSending
Remark	• The specifiable serial ports differ with each CPU used.

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void Notify(unsigned char result) {
    if ((result&RAPI_OVER_ERR) == RAPI_OVER_ERR) {
        /* Overrun error */
    }
    if ((result&RAPI_FRAMING_ERR) == RAPI_FRAMING_ERR) {
        /* Framing error */
    }
    if ((result&RAPI_PARITY_ERR) == RAPI_PARITY_ERR) {
        /* Parity error */
    }
    if ((result&RAPI_TX_END) == RAPI_TX_END) {
        /* Transmission completion */
    }
    if ((result&RAPI_RX_END) == RAPI_RX_END) {
        /* Reception completion */
    }
}

Boolean func( void )
{
    ……
    /* Set callback functions of RAPI_COM1 to serial driver */
    return __ConfigSerialDriverNotify( RAPI_COM1, Notify );
    ……
}
```

__SetSerialFormat

Synopsis

<Set serial communication>

Boolean __SetSerialFormat(unsigned long data1, unsigned char data2)

data1	Setup data 1
data2	Setup data 2

Description

Sets serial communication according to specified parameters.

For details about parameters, refer to the description of __BasicSetSerialFormat.

Return value

If serial communication was successfully set, RAPI_TRUE is returned; if settings failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__BasicSetSerialFormat

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

Boolean func( void )
{
    ……
    /* Set the data of RAPI_COM1 to serial driver */
    return __SetSerialFormat(RAPI_COM1 | RAPI_SM_SYNC | RAPI_CKDIR_INT |
                          RAPI_BCSS_F1 | RAPI_DPOL_NON | RAPI_LSB_SEL, 20);
    ……
}
```

__SetSerialInterrupt

Synopsis

<Set serial interrupts>

Boolean __SetSerialInterrupt(unsigned long data)

data	Setup data

Description

Sets serial interrupts according to specified parameters.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

For interrupt settings, the following values can set.

RAPI_INT_TX_DIS	Transmit interrupt disabled
RAPI_INT_TX_LV_1	Transmit interrupt priority level 1
RAPI_INT_TX_LV_2	Transmit interrupt priority level 2
RAPI_INT_TX_LV_3	Transmit interrupt priority level 3
RAPI_INT_TX_LV_4	Transmit interrupt priority level 4
RAPI_INT_TX_LV_5	Transmit interrupt priority level 5
RAPI_INT_TX_LV_6	Transmit interrupt priority level 6
RAPI_INT_TX_LV_7	Transmit interrupt priority level 7
RAPI_INT_RX_DIS	Receive interrupt disabled
RAPI_INT_RX_LV_1	Receive interrupt priority level 1
RAPI_INT_RX_LV_2	Receive interrupt priority level 2
RAPI_INT_RX_LV_3	Receive interrupt priority level 3
RAPI_INT_RX_LV_4	Receive interrupt priority level 4
RAPI_INT_RX_LV_5	Receive interrupt priority level 5
RAPI_INT_RX_LV_6	Receive interrupt priority level 6
RAPI_INT_RX_LV_7	Receive interrupt priority level 7

(M16C) (UART0, UART1, UART2)

RAPI_INT_TX_DIS	Transmit interrupt disabled
RAPI_INT_TX_LV_1	Transmit interrupt priority level 1
RAPI_INT_TX_LV_2	Transmit interrupt priority level 2
RAPI_INT_TX_LV_3	Transmit interrupt priority level 3
RAPI_INT_TX_LV_4	Transmit interrupt priority level 4
RAPI_INT_TX_LV_5	Transmit interrupt priority level 5
RAPI_INT_TX_LV_6	Transmit interrupt priority level 6
RAPI_INT_TX_LV_7	Transmit interrupt priority level 7
RAPI_INT_RX_DIS	Receive interrupt disabled
RAPI_INT_RX_LV_1	Receive interrupt priority level 1
RAPI_INT_RX_LV_2	Receive interrupt priority level 2

RAPI_INT_RX_LV_3	Receive interrupt priority level 3
RAPI_INT_RX_LV_4	Receive interrupt priority level 4
RAPI_INT_RX_LV_5	Receive interrupt priority level 5
RAPI_INT_RX_LV_6	Receive interrupt priority level 6
RAPI_INT_RX_LV_7	Receive interrupt priority level 7

(M16C) (SI/O3, SI/O4)

RAPI_INT_SIO_DIS	SI/O interrupt disabled
RAPI_INT_SIO_LV_1	SI/O interrupt priority level 1
RAPI_INT_SIO_LV_2	SI/O interrupt priority level 2
RAPI_INT_SIO_LV_3	SI/O interrupt priority level 3
RAPI_INT_SIO_LV_4	SI/O interrupt priority level 4
RAPI_INT_SIO_LV_5	SI/O interrupt priority level 5
RAPI_INT_SIO_LV_6	SI/O interrupt priority level 6
RAPI_INT_SIO_LV_7	SI/O interrupt priority level 7

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

For interrupt settings, the following values can set.

RAPI_INT_TX_DIS	Transmit interrupt disabled
RAPI_INT_TX_LV_1	Transmit interrupt priority level 1
RAPI_INT_TX_LV_2	Transmit interrupt priority level 2
RAPI_INT_TX_LV_3	Transmit interrupt priority level 3
RAPI_INT_TX_LV_4	Transmit interrupt priority level 4
RAPI_INT_TX_LV_5	Transmit interrupt priority level 5
RAPI_INT_TX_LV_6	Transmit interrupt priority level 6
RAPI_INT_TX_LV_7	Transmit interrupt priority level 7
RAPI_INT_RX_DIS	Receive interrupt disabled
RAPI_INT_RX_LV_1	Receive interrupt priority level 1
RAPI_INT_RX_LV_2	Receive interrupt priority level 2
RAPI_INT_RX_LV_3	Receive interrupt priority level 3
RAPI_INT_RX_LV_4	Receive interrupt priority level 4
RAPI_INT_RX_LV_5	Receive interrupt priority level 5
RAPI_INT_RX_LV_6	Receive interrupt priority level 6
RAPI_INT_RX_LV_7	Receive interrupt priority level 7

(H8S)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

For interrupt settings, the following values can set.

RAPI_INT_TX_DIS	Transmit interrupt disabled	RAPI_INT_TX_ENA	Transmit interrupt enabled
RAPI_INT_RX_DIS	Receive interrupt disabled	RAPI_INT_RX_ENA	Receive interrupt enabled

For the interrupt mode and level, the following values can be set.

RAPI_INT_MODE_0	Interrupt control mode 0

RAPI_INT_LV_0	Interrupt control mode 2 and transmit/receive interrupt priority level 0
RAPI_INT_LV_1	Interrupt control mode 2 and transmit/receive interrupt priority level 1
RAPI_INT_LV_2	Interrupt control mode 2 and transmit/receive interrupt priority level 2
RAPI_INT_LV_3	Interrupt control mode 2 and transmit/receive interrupt priority level 3

(H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

For interrupt settings, the following values can set.

RAPI_INT_TX_DIS	Transmit interrupt disabled	RAPI_INT_TX_ENA	Transmit interrupt enabled
RAPI_INT_RX_DIS	Receive interrupt disabled	RAPI_INT_RX_ENA	Receive interrupt enabled

For the CPUs that have an interrupt control register, following values can be set to specify interrupt priority, in addition to ordinary interrupt settings.

RAPI_INT_LV_0	Transmit/Receive interrupt priority level0
RAPI_INT_LV_1	Transmit/Receive interrupt priority level1

Return value

If the serial port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Serial I/O

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include ”rapi_sif_r8c_13.h”

Boolean func( void )
{
   ……
   /* Set interrupt of RAPI_COM1 to serial driver */
   return    __SetSerialInterrupt(    RAPI_COM1    |    RAPI_INT_TX_LV_1    |
RAPI_INT_RX_LV_2 );
   ……
}
```

__StartSerialReceiving

Synopsis

<Start reception>

Boolean __StartSerialReceiving(unsigned long data, unsigned char wordNum, unsigned int *RcvDtBuf)

data	Setup data
wordNum	Number of words received
RcvDtBuf	Pointer to the buffer in which received data is stored

Description

Starts reception of serial communication and gets received data by a specified number of words. When acquisition of received data is complete, this API calls a notification function (if a notification function is registered).

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If reception of serial communication was successfully started, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__ConfigSerialDriverNotify, __StopSerialReceiving

Remark

- For the H8S, H8/300H, wait for at least a 1-bit period before calling this API after __SetSerialFormat was called.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- For the H8S, H8/300H, the following values are stored in the receive buffer.
  High-order 8 bits: The value read from the serial status register.
  Low-order 8 bits: The value read from the receive data register.
  (Not read if an error occurred.)

Program example

```
#include  ”rapi_sif_r8c_13.h”

unsigned int buffer[10];
void func( void )
{
    ……
    /* Get 5 word data received in serial communication */
    __StartSerialReceiving( RAPI_COM1, 5, buffer );
    ……
}
```

__StartSerialSending

Synopsis

<Start transmission>

Boolean __StartSerialSending(unsigned long data, unsigned char wordNum, unsigned int *SndDtBuf)

data	Setup data
wordNum	Number of words transmitted
SndDtBuf	Pointer to the transmit data

Description

Starts transmission of serial communication and writes transmit data to the transmit buffer by a specified number of words. When transmission of all transmit data is complete, this API calls a notification function (if a notification function is registered).

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If transmission of serial communication was successfully started, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__ConfigSerialDriverNotify, __StopSerialSending

Remark

- For the H8S, H8/300H, wait for at least a 1-bit period before calling this API after __SetSerialFormat was called.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

unsigned int buffer[10];
void func( void )
{
    ……
    /* Set 5 word data to transmit buffer of serial communication */
    __StartSerialSending( RAPI_COM1, 5, buffer );
    ……
}
```

__StopSerialReceiving

Synopsis

<Stop reception>

Boolean __StopSerialReceiving(unsigned long data)

data	Setup data

Description

Stops reception of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If reception of serial communication was successfully stopped, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__StartSerialSending

Remark

- For the M16C SI/03 and SI/04, this API cannot be used.
- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include ”rapi_sif_r8c_13.h”

void func( void )
{
    ……
    /* Stop receiving data in serial communication */
    __StopSerialReceiving ( RAPI_COM1 );
    ……
}
```

__StopSerialSending

Synopsis

<Stop transmission>

Boolean __StopSerialSending(unsigned long data)

data	Setup data

Description

Stops transmission of serial communication.

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

If transmission of serial communication was successfully stopped, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Serial I/O

Reference

__StartSerialReceiving

Remark

- For the M16C SI/03 and SI/04, this API cannot be used.
- The specifiable serial ports differ with each CPU used.
- When operating in clock synchronous serial communication mode, data reception is stopped at the same time by this API.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

void func( void )
{
    ……
    /* Stop sending data in serial communication */
    __StopSerialSending ( RAPI_COM1 );
    ……
}
```

__PollingSerialReceiving

Synopsis

<Polling reception>

unsigned int __PollingSerialReceiving(unsigned long data)

data	Setup data

Description

Performs reception of serial communication by polling. This API gets received data by an amount specified by __StartSerialReceiving. When acquisition of received data is complete, it calls a notification function (if a notification function is registered).

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

Out of the receive data counts requested, the number of unreceived data is returned.

Functionality

Serial I/O

Reference

__ConfigSerialDriverNotify, __SetSerialInterrupt, __StartSerialReceiving

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

unsigned int buffer[10], count;

void func( void )
{
    ……
    /* Reception interrupt disable */
    __SetSerialInterrupt( RAPI_COM1 | RAPI_INT_TX_DIS | RAPI_INT_RX_DIS );
    /* Start reception */
    __StartSerialReceiving( RAPI_COM1, 5, buffer );
    do{
        count = __PollingSerialReceiving( RAPI_COM1 );
    }while(count)
    ……
}
```

__PollingSerialSending

Synopsis

<Polling transmission>

Unsigned __PollingSerialSending(unsigned long data)

data	Setup data

Description

Performs transmission of serial communication by polling. This API sends transmit data by an amount specified by __StartSerialSending from the transmit data buffer specified by __StartSerialSending. When transmission of all transmit data is complete, it calls a notification function (if a notification function is registered).

[data]

For data, the following values can be set.

(M16C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2	RAPI_COM4	SI/O3
RAPI_COM5	SI/O4		

(R8C)

RAPI_COM1	UART0	RAPI_COM2	UART1
RAPI_COM3	UART2		

(H8S) (H8/300H)

RAPI_COM1	SCI3 channel 1	RAPI_COM2	SCI3 channel 2
RAPI_COM3	SCI3 channel 3		

Return value

Out of the transmit data counts requested, the number of untransmitted data is returned.

Functionality

Serial I/O

Reference

__ConfigSerialDriverNotify, __SetSerialInterrupt, __StartSerialReceiving

Remark

- The specifiable serial ports differ with each CPU used.
- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

Program example

```
#include  ”rapi_sif_r8c_13.h”

unsigned int buffer[10],count;
void func( void )
{
    ……
    /* Transmission interrupt disable */
    __SetSerialInterrupt( RAPI_COM1 | RAPI_INT_TX_DIS | RAPI_INT_RX_DIS );
    /* Start transmission */
    __StartSerialSending( RAPI_COM1, 5, buffer );
    do{
        count = __PollingSerialSending( RAPI_COM1 );
    }while(count)
    ……
}
```

4.2.2 Timer
__CreateTimer

Synopsis

<Set timer mode>

Boolean __CreateTimer(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to timer mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_A0	Uses timer A channel 0.
RAPI_TIMER_A1	Uses timer A channel 1.
RAPI_TIMER_A2	Uses timer A channel 2.
RAPI_TIMER_A3	Uses timer A channel 3.
RAPI_TIMER_A4	Uses timer A channel 4.
RAPI_TIMER_B0	Uses timer B channel 0.
RAPI_TIMER_B1	Uses timer B channel 1.
RAPI_TIMER_B2	Uses timer B channel 2.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{c32} for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateTimer.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateTimer.
RAPI_GATE_L	Selects a gate facility that counts a period during which input at TA_{iIN} pin remains low.
RAPI_GATE_H	Selects a gate facility that counts a period during which input at TA_{iIN} pin remains high.
RAPI_PULSE_ON	Selects that pulses are output from TA_{iIN} pin.
RAPI_PULSE_OFF	Selects that no pulses are output from TA_{iIN} pin.

• Specifiable definition values when timer A is used (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FC32 }. The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Pulse output state) Specify one from { RAPI_PULSE_ON, RAPI_PULSE_OFF }. The default value is RAPI_PULSE_OFF.

(Gate facility) Specify one from { RAPI_GATE_L, RAPI_GATE_H }. If omitted, “No gate facility” is set.

・ Specifiable definition values when timer B is used (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FC32 }. The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(R8C)

RAPI_TIMER_X	Uses timer X.
RAPI_TIMER_Y	Uses timer Y.
RAPI_TIMER_Z	Uses timer Z.
RAPI_TIMER_RA	Uses timer RA.
RAPI_TIMER_RB	Uses timer RB.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FRING	Selects f_{RING} for the count source.
RAPI_FOCO	Selects foco for the count source.
RAPI_FC32	Selects fC32 for the count source.
RAPI_TIMER_Y_ UNDERFLOW	Selects the underflow of timer Y for the count source.
RAPI_TIMER_RA_UN DERFLOW	Selects the underflow of timer RA for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateTimer.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateTimer.
RAPI_WRITE_ RELOAD_ONLY	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to only the reload register.
RAPI_WRITE_ RELOAD_BOTH	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to both the reload register and the counter, respectively.

• Specifiable definition values when timer X is used (RAPI_TIMER_X specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32 }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

• Specifiable definition values when timer Y is used (RAPI_TIMER_Y specified)

(Count source) Specify one from { RAPI_F1, RAPI_F8, RAPI_FRING }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Write control) Specify one from { RAPI_WRITE_RELOAD_ONLY, RAPI_WRITE_RELOAD_BOTH }. The default value is RAPI_WRITE_RELOAD_BOTH.

- **Specifiable definition values when timer Z is used (RAPI_TIMER_Z specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_TIMER_Y_UNDERFLOW}. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Write control) Specify one from {RAPI_WRITE_RELOAD_ONLY, RAPI_WRITE_RELOAD_BOTH}. The default value is RAPI_WRITE_RELOAD_BOTH.

- **Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_FOCO, RAPI_FC32}. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

- **Specifiable definition values when timer RB is used (RAPI_TIMER_RB specified)**

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_TIMER_RA_UNDERFLOW}. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Write control) Specify one from {RAPI_WRITE_RELOAD_ONLY, RAPI_WRITE_RELOAD_BOTH}. The default value is RAPI_WRITE_RELOAD_BOTH.

(H8S)

RAPI_TIMER_RA	Uses timer RA.
RAPI_TIMER_RB	Uses timer RB.
RAPI_TRA_F1	Timer RA counts with φ.
RAPI_TRA_F2	Timer RA counts with φ/2.
RAPI_TRA_F8	Timer RA counts with φ/8.
RAPI_TRA_F32	Timer RA counts with φ/32.
RAPI_TRA_F40	Timer RA counts with φ40.
RAPI_TRA_F64	Timer RA counts with φ/64.
RAPI_TRA_F128	Timer RA counts with φ/128.
RAPI_TRA_FSUB	Timer RA counts with φsub.
RAPI_TRB_F1	Timer RB counts with φ.
RAPI_TRB_F2	Timer RB counts with φ/2.
RAPI_TRB_F4	Timer RB counts with φ/4.
RAPI_TRB_F8	Timer RB counts with φ/8.

RAPI_TRB_F32	Timer RB counts with φ/32.
RAPI_TRB_F64	Timer RB counts with φ/64.
RAPI_TRB_F128	Timer RB counts with φ/128.
RAPI_TIMER_RA_UNDERFLOW	Timer RB counts with underflow of timer RA.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateTimer.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateTimer.
RAPI_UNDERFLOW	Enables underflow interrupt.
RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.
RAPI_WRITE_RELOAD_ONLY	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to only the reload register.
RAPI_WRITE_RELOAD_BOTH	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to both the reload register and the counter, respectively.

• Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)

(Count source) Specify one from { RAPI_TRA_F1, RAPI_TRA_F2, RAPI_TRA_F8, RAPI_TRA_F32, RAPI_TRA_F40, RAPI_TRA_F64, RAPI_TRA_F128, RAPI_TRA_FSUB }. The default value is RAPI_TRA_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If underflow interrupt requests are enabled, specify RAPI_UNDERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

• Specifiable definition values when timer RB is used (RAPI_TIMER_RB specified)

(Count source) Specify one from { RAPI_TRB_F1, RAPI_TRB_F2, RAPI_TRB_F4, RAPI_TRB_F8, RAPI_TRB_F32, RAPI_TRB_F64, RAPI_TRB_F128, RAPI_TIMER_RA_UNDERFLOW }. The default value is RAPI_TRB_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Write control) Specify one from {RAPI_WRITE_RELOAD_ONLY, RAPI_WRITE_RELOAD_BOTH}. The default value is RAPI_WRITE_RELOAD_BOTH.

(Interrupt) If underflow interrupt requests are enabled, specify RAPI_UNDERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

(H8/300H)

RAPI_TIMER_A	Uses timer A.

RAPI_TIMER_B1	Uses timer B1.
RAPI_TIMER_V	Uses timer V
RAPI_TA_F8	Supplies φ/8 clock to timer A.
RAPI_TA_F32	Supplies φ/32 clock to timer A.
RAPI_TA_F128	Supplies φ/128 clock to timer A.
RAPI_TA_F256	Supplies φ/256 clock to timer A.
RAPI_TA_F512	Supplies φ/512 clock to timer A.
RAPI_TA_F2048	Supplies φ/2048 clock to timer A.
RAPI_TA_F4096	Supplies φ/4096 clock to timer A.
RAPI_TA_F8192	Supplies φ/8192 clock to timer A.
RAPI_TB1_F4	Timer B1 counts with internal clock φ/4.
RAPI_TB1_F16	Timer B1 counts with internal clock φ/16.
RAPI_TB1_F64	Timer B1 counts with internal clock φ/64.
RAPI_TB1_F256	Timer B1 counts with internal clock φ/256.
RAPI_TB1_F512	Timer B1 counts with internal clock φ/512.
RAPI_TB1_F2048	Timer B1 counts with internal clock φ/2048.
RAPI_TB1_F8192	Timer B1 counts with internal clock φ/8192.
RAPI_TV_F4	Timer V counts on falling edges of internal clock φ/4.
RAPI_TV_F8	Timer V counts on falling edges of internal clock φ/8.
RAPI_TV_F16	Timer V counts on falling edges of internal clock φ/16.
RAPI_TV_F32	Timer V counts on falling edges of internal clock φ/32.
RAPI_TV_F64	Timer V counts on falling edges of internal clock φ/64.
RAPI_TV_F128	Timer V counts on falling edges of internal clock φ/128.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateTimer.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateTimer.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_OUT_F4	Outputs φ/4 clock.
RAPI_OUT_F8	Outputs φ/8 clock.
RAPI_OUT_F16	Outputs φ/16 clock.
RAPI_OUT_F32	Outputs φ/32 clock.
RAPI_OUT_8192	Outputs φW/4 clock.
RAPI_OUT_4096	Outputs φW/8 clock.
RAPI_OUT_2048	Outputs φW/16 clock.
RAPI_OUT_1024	Outputs φW/32 clock.
RAPI_AUTO_RELOAD	Uses auto reload facility.

• Specifiable definition values when timer A is used (RAPI_TIMER_A specified)

(Clock) Specify one from { RAPI_TA_F8, RAPI_TA_F32, RAPI_TA_F128, RAPI_TA_F256, RAPI_TA_F512, RAPI_TA_F2048, RAPI_TA_F4096, RAPI_TA_F8192 }. The default value is RAPI_TA_F8192.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Output) Specify one from { RAPI_OUT_F4, RAPI_OUT_F8, RAPI_OUT_F16, RAPI_OUT_F32, RAPI_OUT_8192, RAPI_OUT_4096, RAPI_OUT_2048, RAPI_OUT_1024 }. If this specification is omitted, “Clock output” is set.

• Specifiable definition values when timer B1 is used (RAPI_TIMER_B1 specified)

(Clock) Specify one from { RAPI_TB1_F4, RAPI_TB1_F16, RAPI_TB1_F64, RAPI_TB1_F256, RAPI_TB1_F512, RAPI_TB1_F2048, RAPI_TB1_F8192 }. The default value is RAPI_TB1_F8192.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Reload) To select the auto reload facility, specify RAPI_AUTO_RELOAD. If RAPI_AUTO_RELOAD is not specified, “Select interval facility” is set.

• Specifiable definition values when timer V is used (RAPI_TIMER_V specified)

(Clock) Specify one from { RAPI_TV_F4, RAPI_TV_F8, RAPI_TV_F16, RAPI_TV_F32, RAPI_TV_F64, RAPI_TV_F128}. If this specification is omitted, “Disable clock input” is set.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0‑7) to be set in the interrupt control register.

(H8S)

Specify the interrupt priority level (0-3) to be set in the interrupt priority register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)

Specify the value to be set in the timer register in 16 bits.

(R8C) (H8S)

Specify the set value for the timer or primary register in 8 bits.

(H8/300H)

Specify the set value for the timer reload register in 8 bits. This setting is effective only when timer B1 is used. If any timer other than B1 is used, specify 0.

[data4]

(M16C) (H8/300H)

Specify 0.

(R8C) (H8S)

Specify the set value for the prescaler register in 8 bits.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (timer mode)

Reference

__EnableTimer, __DestroyTimer

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API specify when freeing it from module stanby.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}

void func( void )
{
   /* Set up timer X as timer mode */
   __CreateTimer(  RAPI_TIMER_X|RAPI_TIMER_ON|RAPI_F8,  5,  0x80,  0x80,
TimerIntFunc );
}
```

__EnableTimer

Synopsis

<Control operation of timer mode>

Boolean __EnableTimer(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Controls operation of the timer that is set to specified timer mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_ON	Sets the timer that is set to timer mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to timer mode to stop operating.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_ON	Sets the timer that is set to timer mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to timer mode to stop operating.

(H8S)

RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_ON	Sets the timer that is set to timer mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to timer mode to stop operating.

(H8/300H)

RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_ON	Sets the timer that is set to timer mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to timer mode to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality	Timer (timer mode)
Reference	__CreateTimer, __DestroyTimer
Remark	• If an undefined value is specified in the argument, operation of the API cannot be guaranteed. • The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Disable timer Y as timer mode */
    __EnableTimer( RAPI_TIMER_Y| RAPI_TIMER_OFF );
}
```

__DestroyTimer

Synopsis

<Discard settings of timer mode>

Boolean __DestroyTimer(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Discards settings of the timer that is set to specified timer mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.

(H8S)

RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.

(H8/300H)

RAPI_TIMER_A	Selects timer A.
RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (timer mode)

Reference

__CreateTimer, __EnableTimer

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API places a specified timer into module stanby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
   /* Destroy the setting of timer Z as timer mode */
   __DestroyTimer( RAPI_TIMER_Z );
}
```

__CreateEventCounter

Synopsis

<Set event counter mode>

Boolean __CreateEventCounter(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to event counter mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_A0	Uses timer A channel 0.
RAPI_TIMER_A1	Uses timer A channel 1.
RAPI_TIMER_A2	Uses timer A channel 2
RAPI_TIMER_A3	Uses timer A channel 3.
RAPI_TIMER_A4	Uses timer A channel 4.
RAPI_TIMER_B0	Uses timer B channel 0.
RAPI_TIMER_B1	Uses timer B channel 1.
RAPI_TIMER_B2	Uses timer B channel 2.
RAPI_EV_EXTERNAL	Selects the external signal input to TA_{iIN} pin (when using timer Ai) or TB_{iIN} pin (when using timer Bi) for the count source.
RAPI_EV_TIMER_AJ	Selects overflow or underflow of timer Aj (j = i – 1, however j = 4 if i = 0) for the count source.
RAPI_EV_TIMER_AK	Selects overflow or underflow of timer Ak (k = i – 1, however k = 0 if i = 4) for the count source.
RAPI_EV_TIMER_B2	Selects overflow or underflow of timer B2 for the count source.
RAPI_EV_TIMER_BJ	Selects overflow or underflow of timer Bj (j = i – 1, however j = 2 if i = 0) for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateEventCounter.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateEventCounter.
RAPI_PULSE_ON	Selects that pulses are output from TA_{iIN} pin.
RAPI_PULSE_OFF	Selects that no pulses are output from TA_{iIN} pin.
RAPI_AUTO_RELOAD	Selects reload type for the count type.
RAPI_FREE_RUN	Selects free-run type for the count type.
RAPI_UP_COUNT	Selects up-count for the count operation.
RAPI_DOWN_COUNT	Selects down-count for the count operation.

RAPI_UDF_REGISTER	Selects the UDF register for the cause of up/down switching.
RAPI_TAIOUT	Selects the input signal at TA_{iOUT} pin for the cause of up/down switching.
RAPI_RISING	Selects the rising edge of count source as active edge.
RAPI_FALLING	Selects the falling edge of count source as active edge.
RAPI_BOTH	Selects both rising and falling edges of count source as active edges.

• Specifiable definition values when timer A is used (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)

(Count source) Specify one from { RAPI_EV_EXTERNAL, RAPI_EV_TIMER_AJ, RAPI_EV_TIMER_AK, RAPI_EV_TIMER_B2 }. The default value is RAPI_EV_EXTERNAL.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Pulse output facility) Specify one from { RAPI_PULSE_ON, RAPI_PULSE_OFF }. The default value is RAPI_PULSE_OFF.

(Gate facility) Specify one from { RAPI_GATE_L, RAPI_GATE_H }. If omitted, “No gate facility” is set.

(Count type) Specify one from { RAPI_AUTO_RELOAD, RAPI_FREE_RUN }. The default value is RAPI_AUTO_RELOAD.

(Count direction) Specify one from { RAPI_UP_COUNT, RAPI_DOWN_COUNT }. The default value is RAPI_DOWN_COUNT. The count direction can only be set when the UDF register is used.

(Count direction switching) Specify one from { RAPI_UDF_REGISTER, RAPI_TAIOUT }. The default value is RAPI_UDF_REGISTER.

(Count edge) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

• Specifiable definition values when timer B is used (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

(Count source) Specify one from { RAPI_EV_EXTERNAL, RAPI_EV_TIMER_BJ }. The default value is RAPI_EV_EXTERNAL.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Count edge) Specify one from { RAPI_RISING, RAPI_FALLING, RAPI_BOTH }. The default value is RAPI_FALLING.

(R8C)

RAPI_TIMER_X	Uses timer X.
RAPI_TIMER_Y	Uses timer Y.
RAPI_TIMER_RA	Uses timer RA.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateEventCounter.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateEventCounter.
RAPI_WRITE_ RELOAD_ ONLY	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to only the reload register.

RAPI_WRITE_ RELOAD_ BOTH	Selects operation mode in which writing to the primary register and prescaler causes the set data to be written to both the reload register and the counter, respectively.
RAPI_RISING	Selects the rising edge of count source as active edge.
RAPI_FALLING	Selects the falling edge of count source as active edge.
RAPI_FILTER_F1	Use sampling frequency f1 for digital filter function
RAPI_FILTER_F8	Use sampling frequency f8 for digital filter function
RAPI_FILTER_F32	Use sampling frequency f32 for digital filter function
RAPI_TIOSEL_P1_7	Sets count source input pin to P1_7.
RAPI_TIOSEL_P1_5	Sets count source input pin to P1_5.
RAPI_PULSE_ON	Selects that pulses are output from TRA0 pin.
RAPI_PULSE_OFF	Selects that no pulses are output from TRA0 pin.

• Specifiable definition values when timer X is used (RAPI_TIMER_X specified)

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Count edge) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

• Specifiable definition values when timer Y is used (RAPI_TIMER_Y specified)

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Write control) Specify one from { RAPI_WRITE_RELOAD_ONLY, RAPI_WRITE_RELOAD_BOTH }.
The default value is RAPI_WRITE_RELOAD_BOTH.

(INT2/CNTR1 input) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

• Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Count edge) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Input pin) Specify one from { RAPI_TIOSEL_P1_7, RAPI_TIOSEL_P1_5 }. The default value is RAPI_TIOSEL_P1_7.

(Pulse output function) Specify one from { RAPI_PULSE_ON, RAPI_PULSE_OFF }. The default value is RAPI_PULSE_OFF.

(H8S)

RAPI_TIMER_RA	Uses timer RA.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateEventCounter.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateEventCounter.
RAPI_UNDERFLOW	Enables underflow interrupt.
RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.

RAPI_TRA_GATE_H	Selects a gate facility that counts a period during which input at IRQ2 pin remains high.
RAPI_TRA_RISING	Selects the rising edge of count source as active edge.
RAPI_TRA_FALLING	Selects the falling edge of count source as active edge.
RAPI_TRA_FILTER_F1	Use sampling frequency φ for timer RA filter function.
RAPI_TRA_FILTER_F8	Use sampling frequency φ/8 for timer RA filter function.
RAPI_TRA_FILTER_F32	Use sampling frequency φ/32 for timer RA filter function.
RAPI_TRA_PULSE_ON	Selects that pulses are output from TRA0 pin.
RAPI_TRA_PULSE_OFF	Selects that no pulses are output from TRA0 pin.

• Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)

(Interrupt) If underflow interrupt requests are enabled, specify RAPI_UNDERFLOW. If RAPI_UNDERFLOW is not specified, “No overflow interrupt request” is set.

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Count edge) Specify one from { RAPI_TRA_RISING, RAPI_TRA_FALLING }. The default value is RAPI_TRA_RISING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Pulse output function) Specify one from { RAPI_PULSE_ON, RAPI_PULSE_OFF }. The default value is RAPI_PULSE_OFF.

(Gate function) To use the gate facility, specify RAPI_GATE_H. If RAPI_GATE_H is not specified, "No facility unused" is set.

(H8/300H)

RAPI_TIMER_B1	Uses timer B1.
RAPI_TIMER_V	Uses timer V.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateEventCounter.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateEventCounter.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_AUTO_RELOAD	Uses auto reload facility.
RAPI_RISING	Selects the rising edge of count source as active edge.
RAPI_FALLING	Selects the falling edge of count source as active edge.
RAPI_BOTH	Selects both rising and falling edges of count source as active edges.
RAPI_TRGV_RISING	Selects the rising edge of TRGV pin input signal as active edge.
RAPI_ TRGV_FALLING	Selects the falling edge of TRGV pin input signal as active edge.
RAPI_ TRGV_BOTH	Selects both rising and falling edges of TRGV pin input signal as active edges.
RAPI_TRGV_PROHIBITED	Selects that trigger input from TRGV pin is disabled.

• Specifiable definition values when timer B1 is used (RAPI_TIMER_B1 specified)

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If RAPI_OVERFLOW is not specified, “No overflow interrupt request” is set.

(Reload) To select the auto reload facility, specify RAPI_AUTO_RELOAD. If no interrupts are specified, “No interrupt request” is set.

(Count edge) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING.

• Specifiable definition values when timer V is used (RAPI_TIMER_V specified)

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Count edge) Specify one from { RAPI_RISING, RAPI_FALLING, RAPI_BOTH }. The default value is RAPI_FALLING.

(TRGV pin input) Specify one from { RAPI_TRGV_RISING, RAPI_ TRGV_FALLING, RAPI_ TRGV_BOTH, RAPI_TRGV_PROHIBIT }. The default value is RAPI_TRGV_PROHIBIT.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0‑7) to be set in the interrupt control register.

(H8S)

Specify the interrupt priority level (0-3) to be set in the interrupt priority register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)

Specify the value to be set in the timer register in 16 bits.

(R8C)(H8S)

Specify the set value for the timer or primary register in 8 bits.

When using timer X, specify the set value for the timer register; when using timer Y, specify the set value for the primary register in 8 bits.

(H8/300H)

Specify the set value for the timer reload register in 8 bits. This setting is effective only when timer B1 is used. If any timer other than B1 is used, specify 0.

[data4]

(M16C) (H8/300H)

Specify 0.

(R8C) (H8S)

Specify the set value for the prescaler register in 8 bits.

Return value If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE

is returned.

Functionality Timer (event counter mode)

Reference __EnableEventCounter, __DestroyEventCounter, __GetEventCounter

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API specify when freeing it from module stanby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}

void func( void )
{
    /* Set up timer Y as event counter mode */
    __CreateEventCounter( RAPI_TIMER_Y|RAPI_TIMER_ON|RAPI_FALLING, 5,
                          0x80, 0x80, TimerIntFunc );
}
```

__EnableEventCounter

Synopsis

<Control operation of event counter mode>

Boolean __EnableEventCounter(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Controls operation of the timer that is set to specified timer mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_ON	Sets the timer that is set to event counter mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to event counter mode to stop operating.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_ON	Sets the timer that is set to event counter mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to event counter mode to stop operating.

(H8S)

RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_ON	Sets the timer that is set to event counter mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to event counter mode to stop operating.

(H8/300H)

RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_ON	Sets the timer that is set to event counter mode to start operating.

RAPI_TIMER_OFF	Sets the timer that is set to event counter mode to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (event counter mode)

Reference

__CreateEventCounter, __DestroyEventCounter, __GetEventCounter

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c.h"

void func( void )
{
    /* Disable timer Y as event counter mode */
    __EnableEventCounter( RAPI_TIMER_Y|RAPI_TIMER_OFF );
}
```

__DestroyEventCounter

Synopsis

<Discard settings of event counter mode>

Boolean __DestroyEventCounter(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Discards settings of the timer that is set to specified timer mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_RA	Selects timer RA.

(H8S)

RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (event counter mode)

Reference

__CreateEventCounter, __EnableEventCounter, __GetEventCounter

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API places a specified timer into module stanby state after discarding it.

Program example

```
#include "rapi_timer_r8c.h"

void func( void )
{
    /* Destroy the setting of timer Y as event counter mode */
    __DestroyEventCounter( RAPI_TIMER_Y );
}
```

__GetEventCounter

Synopsis

<Get event counter mode counter value>

Boolean __GetEventCounter(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which counter value is stored

Description

Gets the counter value of the timer that is set to specified event counter mode.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_RA	Selects timer RA.

(H8S)

RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.

[data2]

Specify a pointer to the array in which the acquired counter value is stored.

(M16C)

• When using timer A (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)

[0]: The value of timer Ai register (i = 0–4) is stored.

• When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

[0]: The value of timer Bi register (i = 0–2) is stored.

(R8C)

• When using timer X (RAPI_TIMER_X specified)

[0]: The value of prescaler X register is stored.

[1]: The value of timer X register is stored.

• When using timer Y (RAPI_TIMER_Y specified)

[0]: The value of prescaler Y is stored.

[1]: The value of timer Y primary register is stored.

• When using timer RA (RAPI_TIMER_RA specified)
[0]: The value of timer RA prescaler register is stored.
[1]: The value of timer RA register is stored.
(H8S)
• When using timer RA (RAPI_TIMER_RA specified)
[0]: The value of timer RA prescaler register is stored.
[1]: The value of timer counter RA is stored.
(H8/300H)
• When using timer B1 (RAPI_TIMER_B1 specified)
[0]: The value of timer counter B1 is stored.
• When using timer V (RAPI_TIMER_V specified)
[0]: The value of timer counter V is stored.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (event counter mode)

Reference

__CreateEventCounter, __EnableEventCounter, __DestroyEventCounter

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    unsigned int data[2];

    /* Get the counter of timer Y as event counter mode */
    __GetEventCounter(RAPI_TIMER_Y, data );
}
```

__CreatePulseWidthModulationMode

Synopsis

<Set pulse width modulation mode>

Boolean __CreatePulseWidthModulationMode(unsigned long data1, unsigned int data2, unsigned int* data3, void* data4)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to pulse width modulation mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_A0	Uses timer A channel 0.
RAPI_TIMER_A1	Uses timer A channel 1.
RAPI_TIMER_A2	Uses timer A channel 2.
RAPI_TIMER_A3	Uses timer A channel 3.
RAPI_TIMER_A4	Uses timer A channel 4.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulseWidthModulationMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulseWidthModulationMode.
RAPI_TG_TAIIN	Selects external trigger input from TA_{iIN} pin for the count start condition.
RAPI_EV_ TIMER_AJ	Selects overflow or underflow of timer Aj (j = i – 1, however j = 4 if i = 0) as the trigger for the timer to start counting.
RAPI_EV_TIMER_ AK	Selects overflow or underflow of timer Ak (k = i + 1, however k = 0 if i = 4) as the trigger for the timer to start counting.
RAPI_EV_TIMER_ B2	Selects overflow or underflow of timer B2 as the trigger for the timer to start counting.
RAPI_TG_TAIS	Only writing 1 to the TA_iS bit of the TABSR register causes the timer to start counting.
RAPI_PULSE_ON	Selects that pulses are output from TA_{iIN} pin. Selectable only when timer Ai is used.
RAPI_PULSE_OFF	Selects that no pulses are output from TA_{iIN} pin. Selectable only when timer Ai is used.
RAPI_PWM_16	Selects operation as a 16-bit pulse width modulator.
RAPI_PWM_8	Selects operation as an 8-bit pulse width modulator.

RAPI_RISING	Selects the rising edge of TA_{iIN} pin input signal as active edge.
RAPI_FALLING	Selects the falling edge of TA_{iIN} pin input signal as active edge.

• Specifiable definition values when timer A is used (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FC32 }. The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Count start condition) Specify one from { RAPI_TG_TAIS, RAPI_TG_TAIIN, RAPI_EV_TIMER_AJ, RAPI_EV_TIMER_AK, RAPI_EV_TIMER_B2 }. The default value is RAPI_TG_TAIS.

(Pulse output facility) Specify one from { RAPI_PULSE_ON, RAPI_PULSE_OFF }. The default value is RAPI_PULSE_OFF.

(Modulator) Specify one from { RAPI_PWM_16, RAPI_PWM_8 }. The default value is RAPI_PWM_16.

(TA_{iIN} pin input) Specify one from { RAPI_RISING, RAPI_FALLING }. The default value is RAPI_FALLING. The active edge of TA_{iIN} pin input can only be set when RAPI_TG_TAIIN is selected.

(R8C)

RAPI_TIMER_Y	Uses timer Y.
RAPI_TIMER_Z	Uses timer Z.
RAPI_TIMER_RB	Uses timer RB.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_FRING	Selects f_{RING} for the count source.
RAPI_TIMER_Y_ UNDERFLOW	Selects underflow of timer Y for the count source.
RAPI_TIMER_RA _ UNDERFLOW	Selects underflow of timer RA for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulseWidthModulationMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulseWidthModulationMode.
RAPI_H_L_L	Sets to output a high during the primary period, a low during the secondary period, and a low when the timer is idle.
RAPI_L_H_H	Sets to output a low during the primary period, a high during the secondary period, and a high when the timer is idle.
RAPI_TRB0_P3_1	Select P3_1 for output pin.
RAPI_TRB0_P1_3	Select P1_3 for output pin.

• Specifiable definition values when timer Y is used (RAPI_TIMER_Y specified)

(Count source) Specify one from { RAPI_F1, RAPI_F8, RAPI_FRING }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Output) Specify one from { RAPI_H_L_L, RAPI_L_H_H }. The default value is RAPI_H_L_L.

• Specifiable definition values when timer Z is used (RAPI_TIMER_Z specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_TIMER_Y_UNDERFLOW}. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Output) Specify one from { RAPI_H_L_L, RAPI_L_H_H }. The default value is RAPI_H_L_L.

• Specifiable definition values when timer RB is used (RAPI_TIMER_RB specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_TIMER_RA_UNDERFLOW}. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Output) Specify one from { RAPI_H_L_L, RAPI_L_H_H }. The default value is RAPI_H_L_L.

(Output pin) Specify one from {RAPI_TRBO_P3_1, RAPI_TRBO_P1_3 }. The default value is RAPI_TRBO_P3_1. Valid only in R8C/26 and 27.

(H8S)

RAPI_TIMER_RB	Uses timer RB.
RAPI_TRB_F1	Timer RB counts with φ.
RAPI_TRB_F2	Timer RB counts with φ/2.
RAPI_TRB_F4	Timer RB counts with φ/4.
RAPI_TRB_F8	Timer RB counts with φ/8.
RAPI_TRB_F32	Timer RB counts with φ/32.
RAPI_TRB_F64	Timer RB counts with φ/64.
RAPI_TRB_F128	Timer RB counts with φ/128.
RAPI_TIMER_RA_UNDERFLOW	Timer RB counts with underflow of timer RA.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulseWidthModulationMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulseWidthModulationMode.
RAPI_UNDERFLOW	Enables underflow interrupt.
RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.
RAPI_H_L_L	Sets to output a high during the primary period, a low during the secondary period, and a low when the timer is idle.
RAPI_L_H_H	Sets to output a low during the primary period, a high during the secondary period, and a high when the timer is idle.

• Specifiable definition values when timer RB is used (RAPI_TIMER_RB specified)

(Count source)	Specify one from { RAPI_TRB_F1, RAPI_TRB_F2, RAPI_TRB_F4, RAPI_TRB_F8, RAPI_TRB_F32, RAPI_TRB_F64, RAPI_TRB_F128, RAPI_TIMER_RA_UNDERFLOW }. The default value is RAPI_TRB_F1.
(Operating states set)	Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
(Interrupt)	If underflow interrupt requests are enabled, specify RAPI_UNDERFLOW. If no interrupts are specified, “No interrupt request” is set.
(Interrupt control mode)	Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.
(Output)	Specify one from { RAPI_H_L_L, RAPI_L_H_H }. The default value is RAPI_H_L_L.

(H8/300H)

RAPI_TIMER_V	Uses timer V.
RAPI_TV_F4	Timer V counts on falling edges of internal clock φ/4.
RAPI_TV_F8	Timer V counts on falling edges of internal clock φ/8.
RAPI_TV_F16	Timer V counts on falling edges of internal clock φ/16.
RAPI_TV_F32	Timer V counts on falling edges of internal clock φ/32.
RAPI_TV_F64	Timer V counts on falling edges of internal clock φ/64.
RAPI_TV_F128	Timer V counts on falling edges of internal clock φ/128.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulseWidthModulationMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulseWidthModulationMode.
RAPI_COMPARE_MATCH_A	Enables compare match A interrupt.
RAPI_COMPARE_MATCH_B	Enables compare match B interrupt.
RAPI_OUT_H_L	Sets to output a high the during primary period and a low during the secondary period.
RAPI_OUT_L_H	Sets to output a low the during primary period and a high during the secondary period.
RAPI_RISING	Selects the rising edge of TRGV pin input signal as active edge.
RAPI_FALLING	Selects the falling edge of TRGV pin input signal as active edge.
RAPI_BOTH	Selects both rising and falling edges of TRGV pin input signal as active edge.
RAPI_TRGV_RISING	Selects the rising edge of TRGV pin input signal as active edge.
RAPI_ TRGV_FALLING	Selects the falling edge of TRGV pin input signal as active edge.
RAPI_ TRGV_BOTH	Selects both rising and falling edges of TRGV pin input signal as active edges.
RAPI_TRGV_ PROHIBITED	Selects that trigger input from TRGV pin is disabled.

• Specifiable definition values when timer V is used (RAPI_TIMER_V specified)

(Clock)	Specify one from { RAPI_TV_F4, RAPI_TV_F8, RAPI_TV_F16, RAPI_TV_F32, RAPI_TV_F64, RAPI_TV_F128 }. If clock specification is omitted, specify “Disable clock input.”
(Interrupt)	If compare match A interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A. If compare match B interrupt requests are enabled, specify RAPI_COMPARE_MATCH_B. If no interrupts are specified, “No interrupt request” is set.
(Operating states set)	Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
(Output)	Specify one from { RAPI_OUT_H_L, RAPI_OUT_L_H }. Always be sure to specify this output.
(TRGV pin input)	Specify one from { RAPI_TRGV_RISING, RAPI_ TRGV_FALLING, RAPI_ TRGV_BOTH, RAPI_TRGV_PROHIBITED }. The default value is RAPI_TRGV_PROHIBITED.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0‑7) to be set in the interrupt control register.

(H8S)

Specify the interrupt priority level (0-3) to be set in the interrupt priority register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)

Specify a pointer to the 16-bit variable in which the set value for the timer register is stored.

For 16-bit PWM, specify the value of ‘n’ in “high-level width n/f_j, period 65535/f_j” in 16 bits.

For 8-bit PWM, specify the values of ‘n’ and ‘m’ in “high-level width n (m + 1)/f, period 255 (m + 1)/f_j” in the 8 high-order bits and the 8 low-order bits, respectively.

(R8C)

Specify a pointer to the array in which the set value is stored.

[0]: Specify the set value for the primary register in 8 bits.

[1]: Specify the set value for the secondary register in 8 bits.

[2]: Specify the set value for the prescaler in 8 bits.

(H8S)

Specify a pointer to the array in which the set value is stored.

[0]: Specify the set value for the primary register in 8 bits.

[1]: Specify the set value for the secondary register in 8 bits.

[2]: Specify the set value for the prescaler in 8 bits.

(H8/300H)

Specify a pointer to the array in which the set value is stored.

[0]: Specify the comparison value A in 8 bits.
[1]: Specify the comparison value B in 8 bits.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width modulation mode (PWM mode))

Reference

__EnablePulseWidthModulationMode, __DestroyPulseWidthModulationMode

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- For the H8S, H8/300H, make sure that comparison value A < comparison value B.
- When used for the H8/300H, this API specify when freeing it from module stanby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}

void func( void )
{
    unsigned int p_tim[] =  {0xAA, 0xBB, 0xCC};

    /* Set up timer Z as pulse width modulation mode */

__CreatePulseWidthModulationMode( RAPI_TIMER_Z|RAPI_TIMER_ON|RAPI_F8,
                                  5, p_tim, TimerIntFunc);
}
```

__EnablePulseWidthModulationMode

Synopsis

<Control operation of pulse width modulation mode>

Boolean __EnablePulseWidthModulationMode(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Controls operation of the timer that is set to specified pulse width modulation mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_ON	Sets the timer that is set to pulse width modulation mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width modulation mode to stop operating.

(R8C)

RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_ON	Sets the timer that is set to pulse width modulation mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width modulation mode to stop operating.

(H8S)

RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_ON	Sets the timer that is set to pulse width modulation mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width modulation mode to stop operating.

(H8/300H)

RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_ON	Sets the timer that is set to pulse width modulation mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width modulation mode to stop operating.

Return value	If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.
Functionality	Timer (pulse width modulation mode (PWM mode))
Reference	__CreatePulseWidthModulationMode, __DestroyPulseWidthModulationMode
Remark	• If an undefined value is specified in the argument, operation of the API cannot be guaranteed. • The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Enable timer Y as pulse width modulation mode */
    __EnablePulseWidthModulationMode( RAPI_TIMER_Y|RAPI_TIMER_ON );
}
```

__DestroyPulseWidthModulationMode

Synopsis

<Discard settings of pulse width modulation mode>

Boolean __DestroyPulseWidthModulationMode(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Discards settings of the timer that is set to specified pulse width modulation mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.

(R8C)

RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RB	Selects timer RB.

(H8S)

RAPI_TIMER_RB	Selects timer RB.

(H8/300H)

RAPI_TIMER_V	Selects timer V.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width modulation mode (PWM mode))

Reference

__CreatePulseWidthModulationMode, __EnablePulseWidthModulationMode

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API places a specified timer into module stanby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Destroy the setting of timer Z as pulse width modulation mode */
    __DestroyPulseWidthModulationMode( RAPI_TIMER_Z );
}
```

__CreatePulsePeriodMeasurementMode

Synopsis

<Set pulse period measurement mode>

Boolean __CreatePulsePeriodMeasurementMode(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to pulse period measurement mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_B0	Uses timer B channel 0.
RAPI_TIMER_B1	Uses timer B channel 1.
RAPI_TIMER_B2	Uses timer B channel 2.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_RISING_RISING	Selects measurement of an interval from the rise to the next rise of a measurement pulse.
RAPI_FALLING_FALLING	Selects measurement of an interval from the fall to the next fall of a measurement pulse.

• Specifiable definition values when timer B is used (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

(Count source)	Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FC32 }. The default value is RAPI_F2.
(Operating states set)	Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
(Measurement pulse)	Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_FALLING_FALLING.

(R8C)

RAPI_TIMER_X	Uses timer X.

RAPI_TIMER_RA	Uses timer RA.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.
RAPI_FOCO	Selects fOCO for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_RISING_ RISING	Selects measurement of an interval from the rise to the next rise of a measurement pulse.
RAPI_FALLING_ FALLING	Selects measurement of an interval from the fall to the next fall of a measurement pulse.
RAPI_FILTER_F1	Use sampling frequency f1 for digital filter function
RAPI_FILTER_F8	Use sampling frequency f8 for digital filter function
RAPI_FILTER_F32	Use sampling frequency f32 for digital filter function
RAPI_TIOSEL_P1_7	Sets count source input pin to P1_7.
RAPI_TIOSEL_P1_5	Sets count source input pin to P1_5.

• Specifiable definition values when timer X is used (RAPI_TIMER_X specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32 }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_FALLING_FALLING.

• Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FOCO }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measure pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_FALLING_FALLING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Input pin) Specify one from { RAPI_TIOSEL_P1_7, RAPI_TIOSEL_P1_5 }. The default value is RAPI_TIOSEL_P1_7.

(H8S)

RAPI_TIMER_RA	Uses timer RA.
RAPI_TRA_F1	Timer RA counts with φ.
RAPI_TRA_F2	Timer RA counts with φ/2.
RAPI_TRA_F8	Timer RA counts with φ/8.
RAPI_TRA_F32	Timer RA counts with φ/32.
RAPI_TRA_F40	Timer RA counts with φ40.
RAPI_TRA_F64	Timer RA counts with φ/64.

RAPI_TRA_F128	Timer RA counts with φ/128.
RAPI_TRA_FSUB	Timer RA counts with φsub.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_UNDERFLOW	Enables underflow interrupt.
RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.
RAPI_TRA_GATE_H	Selects a gate facility that counts a period during which input at IRQ2 pin remains high.
RAPI_RISING_ RISING	Selects measurement of an interval from the rise to the next rise of a measurement pulse.
RAPI_FALLING_ FALLING	Selects measurement of an interval from the fall to the next fall of a measurement pulse.
RAPI_TRA_FILTER_F 1	Use sampling frequency φ for timer RA filter function.
RAPI_TRA_FILTER_F 8	Use sampling frequency φ/8 for timer RA filter function.
RAPI_TRA_FILTER_F 32	Use sampling frequency φ/32 for timer RA filter function.

• Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)

(Count source) Specify one from { RAPI_TRA_F1, RAPI_TRA_F2, RAPI_TRA_F8, RAPI_TRA_F32, RAPI_TRA_F40, RAPI_TRA_F64, RAPI_TRA_F128, RAPI_TRA_FSUB }. The default value is RAPI_TRA_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If underflow interrupt requests are enabled, specify RAPI_UNDERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

(Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_RISING_RISING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Gate function) To use the gate facility, specify RAPI_GATE_H. If RAPI_GATE_H is not specified, "No facility unused" is set.

(H8/300H)

RAPI_TIMER_W	Uses timer W.
RAPI_TIMER_Z0	Uses timer Z channel 0.
RAPI_TIMER_Z1	Uses timer Z channel 1.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD0 channel 0.
RAPI_TIMER_RD1	Uses timer RD0 channel 1.
RAPI_TIMER_RD2	Uses timer RD0 channel 2.

RAPI_TIMER_RD3	Uses timer RD0 channel 3.
RAPI_TW_F1	Timer W counts with internal clock φ.
RAPI_TW_F2	Timer W counts with internal clock φ/2.
RAPI_TW_F4	Timer W counts with internal clockφ/4.
RAPI_TW_F8	Timer W counts with internal clock φ/8.
RAPI_TZ_F1	Timer Z counts with internal clock φ.
RAPI_TZ_F2	Timer Z counts with internal clock φ/2.
RAPI_TZ_F4	Timer Z counts with internal clock φ/4.
RAPI_TZ_F8	Timer Z counts with internal clock φ/8.
RAPI_TRC_F1	Timer RC counts with internal clock φ.
RAPI_TRC_F2	Timer RC counts with internal clock φ/2.
RAPI_TRC_F4	Timer RC counts with internal clockφ/4.
RAPI_TRC_F8	Timer RC counts with internal clock φ/8.
RAPI_TRC_F32	Timer RC counts with internal clock φ/8.
RAPI_TRC_F40M	Timer RC counts with internal clock φ40M.
RAPI_TRD_F1	Timer RD counts with internal clock φ.
RAPI_TRD_F2	Timer RD counts with internal clock φ/2.
RAPI_TRD_F4	Timer RD counts with internal clockφ/4.
RAPI_TRD_F8	Timer RD counts with internal clock φ/8.
RAPI_TRD_F32	Timer RD counts with internal clock φ/32.
RAPI_TRD_F40M	Timer RD counts with internal clock φ40M.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_RISING_ RISING	Selects measurement of an interval from the rise to the next rise of a measurement pulse.
RAPI_FALLING_ FALLING	Selects measurement of an interval from the fall to the next fall of a measurement pulse.
RAPI_TRC_FILTER_F 1	Use sampling frequency f1 for timer RC digital filter function
RAPI_TRC_FILTER_F 8	Use sampling frequency f8 for timer RC digital filter function
RAPI_TRC_FILTER_F 32	Use sampling frequency f32 for timer RC digital filter function
RAPI_TRD_FILTER_F	Use sampling frequency f for timer RC digital filter function
RAPI_TRD_FILTER_F 1	Use sampling frequency f1 for timer RD digital filter function
RAPI_TRD_FILTER_F 8	Use sampling frequency f8 for timer RD digital filter function
RAPI_TRD_FILTER_F 32	Use sampling frequency f32 for timer RD digital filter function
RAPI_TRD_FILTER_F	Use sampling frequency f for timer RD digital filter function

RAPI_FTIOA	Use FTIOA pin as input pin.
RAPI_FTIOB	Use FTIOB pin as input pin.

• Specifiable definition values when timer W is used (RAPI_TIMER_W specified)

(Count source) Specify one from { RAPI_TW_F1, RAPI_TW_F2, RAPI_TW_F4, RAPI_TW_F8 }. The default value is RAPI_TW_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_RISING_RISING.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

• Specifiable definition values when timer Z is used (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)

(Count source) Specify one from { RAPI_TZ_F1, RAPI_TZ_F2, RAPI_TZ_F4, RAPI_TZ_F8 }. The default value is RAPI_TZ_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_RISING_RISING.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.

• Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)

(Count source) Specify one from { RAPI_TRC_F1, RAPI_TRC_F2, RAPI_TRC_F4, RAPI_TRC_F8, RAPI_TRC_F32 }. The default value is RAPI_RC_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measurement pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_RISING_RISING.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.

(Clock for digital filter) Specify one from { RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F }. The default value is RAPI_TRC_FILTER_F32.

• Specifiable definition values when timer RD is used (RAPI_TIMER_RD specified)

(Count source) Specify one from { RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M }. The default value is RAPI_RC_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measurement pulse)	Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_RISING_RISING.
(Interrupt)	If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.
(Input pin)	Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.
(Clock for digital filter)	Specify one from { RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F }. The default value is RAPI_TRC_FILTER_F32.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0‑7) to be set in the interrupt control register.

(H8S)

Specify the interrupt priority level (0-3) to be set in the interrupt priority register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C) (H8/300H)

Specify 0.

(R8C) (H8S)

Specify the set value for the timer or primary register in 8 bits.

[data4]

(M16C) (H8/300H)

Specify 0.

(R8C) (H8S)

Specify the set value for the prescaler register.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse period measurement mode)

Reference

__EnablePulsePeriodMeasurementMode, __DestroyPulsePeriodMeasurementMode, __GetPulsePeriodMeasurementMode

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API specify when freeing it from module stanby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}

void func( void )
{
   /* Set up timer X as pulse period measurement mode */
   __CreatePulsePeriodMeasurementMode(
       RAPI_TIMER_X|RAPI_TIMER_ON|RAPI_FALLING_FALLING|RAPI_F8,
       5, 0x80, 0x80, TimerIntFunc);
}
```

__EnablePulsePeriodMeasurementMode

Synopsis

<Control operation of pulse period measurement mode>

Boolean __EnablePulsePeriodMeasurementMode(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Controls operation of the timer that is set to specified pulse period measurement mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_ON	Sets the timer that is set to pulse period measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse period measurement mode to stop operating.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_ON	Sets the timer that is set to pulse period measurement mode to start operating.
RAPI_TIMER_ OFF	Sets the timer that is set to pulse period measurement mode to stop operating.

(H8S)

RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_ON	Sets the timer that is set to pulse period measurement mode to start operating.
RAPI_TIMER_ OFF	Sets the timer that is set to pulse period measurement mode to stop operating.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.

RAPI_TIMER_RD 2	Selects timer RD channel 2.
RAPI_TIMER_RD 3	Selects timer RD channel 3.
RAPI_TIMER_ON	Sets the timer that is set to pulse period measurement mode to start operating.
RAPI_TIMER_ OFF	Sets the timer that is set to pulse period measurement mode to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse period measurement mode)

Reference

__CreatePulsePeriodMeasurementMode, __DestroyPulsePeriodMeasurementMode, __GetPulsePeriodMeasurementMode

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Enable timer X as pulse period measurement mode */
    __EnablePulsePeriodMeasurementMode( RAPI_TIMER_X|RAPI_TIMER_ON );
}
```

__DestroyPulsePeriodMeasurementMode

Synopsis

<Discard settings of pulse period measurement mode>

Boolean __DestroyPulsePeriodMeasurementMode(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Discards settings of the timer that is set to specified pulse period measurement mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.

(H8S)

RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.
RAPI_TIMER_RD 2	Selects timer RD channel 2.
RAPI_TIMER_RD 3	Selects timer RD channel 3.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse period measurement mode)

Reference

__CreatePulsePeriodMeasurementMode, __EnablePulsePeriodMeasurementMode, __GetPulsePeriodMeasurementMode

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.

- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API places a specified timer into module stanby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Destroy the setting of timer X as pulse period measurement mode */
    __DestroyPulsePeriodMeasurementMode( RAPI_TIMER_X );
}
```

__GetPulsePeriodMeasurementMode

Synopsis

<Get measured value in pulse period measurement mode>

Boolean __GetPulsePeriodMeasurementMode(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which counter value is stored

Description

Gets the counter value of the timer that is set to specified pulse period measurement mode.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.

(H8S)

RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.
RAPI_TIMER_RD 2	Selects timer RD channel 2.
RAPI_TIMER_RD 3	Selects timer RD channel 3.

[data2]

Specify a pointer to the array in which the acquired counter value is stored.

(M16C)

• When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

[0]: The value of timer Bi register (i = 0–2) is stored.

(R8C)

• When using timer X (RAPI_TIMER_X specified)

[0]: The value of prescaler X register is stored.

[1]: The value of timer X register is stored.
• When using timer RA (RAPI_TIMER_RA specified)
[0]: The value of timer RA prescaler register is stored.
[1]: The value of timer RA register is stored.
(H8S)
• When using timer RA (RAPI_TIMER_RA specified)
[0]: The value of timer RA prescaler register is stored.
[1]: The value of timer RA register is stored.
(H8/300H)
• When using timer W (RAPI_TIMER_W specified)
[0]: (The value of general register A) – (The value of general register C) is stored.
[1]: (The value of general register B) – (The value of general register D) is stored.
• When using timer Z (RAPI_TIMER_Z0 - RAPI_TIMER_Z1 specified)
[0]: (The value of general register Ai(i=0,1)) – (The value of general register C i(i=0,1)) is stored.
[1]: (The value of general register B i(i=0,1)) – (The value of general register D i(i=0,1)) is stored.
• When using timer RC (RAPI_TIMER_RC specified)
[0]: (The value of general register A) – (The value of general register C) is stored.
[1]: (The value of general register B) – (The value of general register D) is stored.
• When using timer RD (RAPI_TIMER_RD specified)
[0]: (The value of general register A) – (The value of general register C) is stored.
[1]: (The value of general register B) – (The value of general register D) is stored.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse period measurement mode)

Reference

__CreatePulsePeriodMeasurementMode, __EnablePulsePeriodMeasurementMode, __DestroyPulsePeriodMeasurementMode

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    unsigned int data[2];

    /* Get the measured value of timer X as pulse period measurement mode
*/
    __GetPulsePeriodMeasurementMode( RAPI_TIMER_X, data );
}
```

__CreatePulseWidthMeasurementMode

Synopsis

<Set pulse width measurement mode>

Boolean __CreatePulseWidthMeasurementMode(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified timer to pulse with measurement mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_B0	Uses timer B channel 0.
RAPI_TIMER_B1	Uses timer B channel 1.
RAPI_TIMER_B2	Uses timer B channel 2.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulseWidthMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulseWidthMeasurementMode.

• Specifiable definition values when timer B is used (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FC32 }. The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(R8C)

RAPI_TIMER_X	Uses timer X.
RAPI_TIMER_RA	Uses timer RA.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FC32	Selects f_{C32} for the count source.

RAPI_FOCO	Selects fOCO for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_FALLING_ RISING	Measures the low-level width of a pulse.
RAPI_RISING_ FALLING	Measures the high-level width of a pulse.
RAPI_FILTER_F1	Use sampling frequency f1 for digital filter function
RAPI_FILTER_F8	Use sampling frequency f8 for digital filter function
RAPI_FILTER_F32	Use sampling frequency f32 for digital filter function
RAPI_TIOSEL_P1_7	Sets count source input pin to P1_7.
RAPI_TIOSEL_P1_5	Sets count source input pin to P1_5.

• Specifiable definition values when timer X is used (RAPI_TIMER_X specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32 }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measurement pulse) Specify one from { RAPI_FALLING_ RISING, RAPI_RISING_ FALLING }. The default value is RAPI_FALLING_RISING.

• Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2, RAPI_F8, RAPI_F32, RAPI_FOCO }. The default value is RAPI_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Measure pulse) Specify one from { RAPI_RISING_RISING, RAPI_FALLING_FALLING }. The default value is RAPI_FALLING_FALLING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Input pin) Specify one from { RAPI_TIOSEL_P1_7, RAPI_TIOSEL_P1_5 }. The default value is RAPI_TIOSEL_P1_7.

(H8S)

RAPI_TIMER_RA	Uses timer RA.
RAPI_TRA_F1	Timer RA counts with φ.
RAPI_TRA_F2	Timer RA counts with φ/2.
RAPI_TRA_F8	Timer RA counts with φ/8.
RAPI_TRA_F32	Timer RA counts with φ/32.
RAPI_TRA_F40	Timer RA counts with φ40.
RAPI_TRA_F64	Timer RA counts with φ/64.
RAPI_TRA_F128	Timer RA counts with φ/128.
RAPI_TRA_FSUB	Timer RA counts with φsub.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.

RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_UNDERFLOW	Enables underflow interrupt.
RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.
RAPI_TRA_GATE_H	Selects a gate facility that counts a period during which input at IRQ2 pin remains high.
RAPI_FALLING_ RISING	Measures the low-level width of a pulse.
RAPI_RISING_ FALLING	Measures the high-level width of a pulse.
RAPI_TRA_FILTER_F1	Use sampling frequency φ for timer RA filter function.
RAPI_TRA_FILTER_F8	Use sampling frequency φ/8 for timer RA filter function.
RAPI_TRA_FILTER_F32	Use sampling frequency φ/32 for timer RA filter function.

• Specifiable definition values when timer RA is used (RAPI_TIMER_RA specified)

(Count source) Specify one from { RAPI_TRA_F1, RAPI_TRA_F2, RAPI_TRA_F8, RAPI_TRA_F32, RAPI_TRA_F40, RAPI_TRA_F64, RAPI_TRA_F128, RAPI_TRA_FSUB }. The default value is RAPI_TRA_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If underflow interrupt requests are enabled, specify RAPI_UNDERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

(Measurement pulse) Specify one from { RAPI_FALLING_ RISING, RAPI_RISING_ FALLING }. The default value is RAPI_FALLING_RISING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Gate function) To use the gate facility, specify RAPI_GATE_H. If RAPI_GATE_H is not specified, "No facility unused" is set.

(H8/300H)

RAPI_TIMER_W	Uses timer W.
RAPI_TIMER_Z0	Uses timer Z channel 0.
RAPI_TIMER_Z1	Uses timer Z channel 1.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD0 channel 0.
RAPI_TIMER_RD1	Uses timer RD0 channel 1.
RAPI_TIMER_RD2	Uses timer RD0 channel 2.
RAPI_TIMER_RD3	Uses timer RD0 channel 3.
RAPI_TW_F1	Timer W counts with internal clock φ.
RAPI_TW_F2	Timer W counts with internal clock φ/2.

RAPI_TW_F4	Timer W counts with internal clock φ/4.
RAPI_TW_F8	Timer W counts with internal clock φ/8.
RAPI_TZ_F1	Timer Z counts with internal clock φ.
RAPI_TZ_F2	Timer Z counts with internal clock φ/2.
RAPI_TZ_F4	Timer Z counts with internal clock φ/4.
RAPI_TZ_F8	Timer Z counts with internal clock φ/8.
RAPI_TRC_F1	Timer RC counts with internal clock φ.
RAPI_TRC_F2	Timer RC counts with internal clock φ/2.
RAPI_TRC_F4	Timer RC counts with internal clockφ/4.
RAPI_TRC_F8	Timer RC counts with internal clock φ/8.
RAPI_TRC_F32	Timer RC counts with internal clock φ/8.
RAPI_TRC_F40M	Timer RC counts with internal clock φ40M.
RAPI_TRD_F1	Timer RD counts with internal clock φ.
RAPI_TRD_F2	Timer RD counts with internal clock φ/2.
RAPI_TRD_F4	Timer RD counts with internal clockφ/4.
RAPI_TRD_F8	Timer RD counts with internal clock φ/8.
RAPI_TRD_F32	Timer RD counts with internal clock φ/32.
RAPI_TRD_F40M	Timer RD counts with internal clock φ40M.
RAPI_TIMER_ON	Sets the timer to start operating in __CreatePulsePeriodMeasurementMode.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreatePulsePeriodMeasurementMode.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_TRC_FILTER_F1	Use sampling frequency f1 for timer RC digital filter function
RAPI_TRC_FILTER_F8	Use sampling frequency f8 for timer RC digital filter function
RAPI_TRC_FILTER_F32	Use sampling frequency f32 for timer RC digital filter function
RAPI_TRD_FILTER_F	Use sampling frequency f for timer RC digital filter function
RAPI_TRD_FILTER_F1	Use sampling frequency f1 for timer RD digital filter function
RAPI_TRD_FILTER_F8	Use sampling frequency f8 for timer RD digital filter function
RAPI_TRD_FILTER_F32	Use sampling frequency f32 for timer RD digital filter function
RAPI_TRD_FILTER_F	Use sampling frequency f for timer RD digital filter function
RAPI_FTIOA	Use FTIOA pin as input pin.
RAPI_FTIOB	Use FTIOB pin as input pin.

• Specifiable definition values when timer W is used (RAPI_TIMER_W specified)

(Count source) Specify one from { RAPI_TW_F1, RAPI_TW_F2, RAPI_TW_F4, RAPI_TW_F8 }. The default value is RAPI_TW_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

• Specifiable definition values when timer Z is used (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)

(Count source) Specify one from { RAPI_TZ_F1, RAPI_TZ_F2, RAPI_TZ_F4, RAPI_TZ_F8 }. The default value is RAPI_TZ_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.

• Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)

(Count source) Specify one from { RAPI_TRC_F1, RAPI_TRC_F2, RAPI_TRC_F4, RAPI_TRC_F8, RAPI_TRC_F32, RAPI_TRC_F40M }. The default value is RAPI_RC_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.

(Clock for digital filter) Specify one from { RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F }. The default value is RAPI_TRC_FILTER_F32.

• Specifiable definition values when timer RD is used (RAPI_TIMER_RD specified)

(Count source) Specify one from { RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M }. The default value is RAPI_RC_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If no interrupts are specified, “No interrupt request” is set.

(Input pin) Specify one from { RAPI_FTIOA, RAPI_FTIOB }. The default value is RAPI_FTIOA.

(Clock for digital filter) Specify one from { RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F }. The default value is RAPI_TRC_FILTER_F32.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0–7) to be set in the interrupt control register.

(H8S)

Specify the interrupt priority level (0-3) to be set in the interrupt priority register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)(H8/300H)

Specify 0.

(R8C)(H8S)

Specify the set value for the timer or primary register in 8 bits.

[data4]

(M16C) (H8/300H)

Specify 0.

(R8C) (H8S)

Specify the set value for the prescaler register.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width measurement mode)

Reference

__EnablePulseWidthMeasurementMode, __DestroyPulseWidthMeasurementMode, __GetPulseWidthMeasurementMode

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API specify when freeing it from module stanby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}

void func( void )
{
   /* Set up timer X as pulse width measurement mode */
   __CreatePulseWidthMeasurementMode(
      RAPI_TIMER_X|RAPI_TIMER_ON|RAPI_RISING_FALLING|RAPI_F8,
      5, 0x80, 0x80, TimerIntFunc);
}
```

__EnablePulseWidthMeasurementMode

Synopsis

<Control operation of pulse width measurement mode>

Boolean __EnablePulseWidthMeasurementMode(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Controls operation of the timer that is set to specified pulse width measurement mode.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_ON	Sets the timer that is set to pulse width measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width measurement mode to stop operating.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_ON	Sets the timer that is set to pulse width measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width measurement mode to stop operating.

(H8S)

RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_ON	Sets the timer that is set to pulse width measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width measurement mode to stop operating.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.
RAPI_TIMER_RD 2	Selects timer RD channel 2.

RAPI_TIMER_RD 3	Selects timer RD channel 3.
RAPI_TIMER_ON	Sets the timer that is set to pulse width measurement mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to pulse width measurement mode to stop operating.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width measurement mode)

Reference

__CreatePulseWidthMeasurementMode, __DestroyPulseWidthMeasurementMode, __GetPulseWidthMeasurementMode

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Disable timer X as pulse width measurement mode */
    __EnablePulseWidthMeasurementMode( RAPI_TIMER_X|RAPI_TIMER_OFF );
}
```

__DestroyPulseWidthMeasurementMode

Synopsis

<Discard settings of pulse width measurement mode>

Boolean __DestroyPulseWidthMeasurementMode(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Discards settings of the timer that is set to specified pulse width measurement mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.

(H8S)

RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width measurement mode)

Reference

__CreatePulseWidthMeasurementMode, __EnablePulseWidthMeasurementMode, __GetPulseWidthMeasurementMode

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

- When used for the H8S, H8/300H, this API places a specified timer into module stanby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Destroy the setting of timer X as pulse width measurement mode */
    __DestroyPulseWidthMeasurementMode( RAPI_TIMER_X );
}
```

__GetPulseWidthMeasurementMode

Synopsis

<Get measured value in pulse width measurement mode>

Boolean __GetPulseWidthMeasurementMode(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which counter value is stored

Description

Gets the counter value of the timer that is set to specified pulse width measurement mode.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.

(R8C)

RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_RA	Selects timer RA.

(H8S)

RAPI_TIMER_RA	Selects timer RA.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD 0	Selects timer RD channel 0.
RAPI_TIMER_RD 1	Selects timer RD channel 1.
RAPI_TIMER_RD 2	Selects timer RD channel 2.
RAPI_TIMER_RD 3	Selects timer RD channel 3.

[data2]

Specify a pointer to the array in which the acquired counter value is stored.

(M16C)

• When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

[0]: The value of timer Bi register (i = 0–2) is stored.

(R8C)

• When using timer X (RAPI_TIMER_X specified)

[0]: The value of prescaler X register is stored.

[1]: The value of timer X register is stored.
• When using timer RA (RAPI_TIMER_RA specified)
[0]: The value of timer RA prescaler register is stored.
[1]: The value of timer RA register is stored.
(H8S)
[0]: The value of timer RA prescaler register is stored.
[1]: The value of timer RA register is stored.
(H8/300H)
• When using timer W (RAPI_TIMER_W specified)
[0]: (The value of general register A) – (The value of general register C) is stored.
[1]: (The value of general register B) – (The value of general register D) is stored.
• When using timer Z (RAPI_TIMER_Z0 - RAPI_TIMER_Z1 specified)
[0]: (The value of general register Ai(i=0,1)) – (The value of general register C i(i=0,1)) is stored.
[1]: (The value of general register B i(i=0,1)) – (The value of general register D i(i=0,1)) is stored.
• When using timer RC (RAPI_TIMER_RC specified)
[0]: (The value of general register A) – (The value of general register C) is stored.
[1]: (The value of general register B) – (The value of general register D) is stored.
• When using timer RD (RAPI_TIMER_RD specified)
[0]: (The value of general register A) – (The value of general register C) is stored.
[1]: (The value of general register B) – (The value of general register D) is stored.

Return value

If the timer specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (pulse width measurement mode)

Reference

__CreatePulseWidthMeasurementMode, __EnablePulseWidthMeasurementMode, __DestroyPulseWidthMeasurementMode

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    unsigned int data[2];

    /* Get the measured value of timer X as pulse width measurement mode
*/
    __GetPulseWidthMeasurementMode( RAPI_TIMER_X, data );
}
```

__CreateInputCapture

Synopsis

<Set input capture mode>

Boolean __CreateInputCapture(unsigned long data1, unsigned int* data2, unsigned int* data3, unsigned int* data4, void data5)**

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
data5	Setup data 5 (content differs with MCU type)

Description

Sets a specified timer to input capture mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_S	Uses timer S.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateInputCapture.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateInputCapture.
RAPI_OVERFLOW_BIT14	Selects overflow of bit 14 for the base timer interrupt.
RAPI_OVERFLOW_BIT15	Selects overflow of bit 15 for the base timer interrupt.

• Specifiable definition values when timer S is used (RAPI_TIMER_S specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2 }. The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Base timer) Specify one from { RAPI_OVERFLOW_BIT14, RAPI_OVERFLOW_BIT15 }. The default value is RAPI_OVERFLOW_BIT15.

(R8C)

RAPI_TIMER_C	Uses timer C.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD channel 0.
RAPI_TIMER_RD1	Uses timer RD channel 1.
RAPI_TIMER_RF	Uses timer RF.
RAPI_F1	Selects f_1 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FRING_FAST	Selects $f_{RING\text{-}fast}$ for the count source.
RAPI_TRC_F1	Selects f_1 for the timer RC count source.
RAPI_TRC_F2	Selects f_2 for the timer RC count source.
RAPI_TRC_F4	Selects f_4 for the timer RC count source.

RAPI_TRC_F8	Selects f_8 for the timer RC count source.
RAPI_TRC_F32	Selects f_{32} for the timer RC count source.
RAPI_TRC_F40M	Selects fOCO40M for the timer RC count source.
RAPI_TRD_F1	Selects f_1 for the timer RD count source.
RAPI_TRD_F2	Selects f_2 for the timer RD count source.
RAPI_TRD_F8	Selects f_8 for the timer RD count source.
RAPI_TRD_F32	Selects f_{32} for the timer RD count source.
RAPI_TRD_F40M	Selects fOCO40M for the timer RD count source.
RAPI_TRF_F1	Selects f_1 for the timer RF count source.
RAPI_TRF_F8	Selects f_8 for the timer RF count source.
RAPI_TRF_F32	Selects f_{32} for the timer RF count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateInputCapture.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateInputCapture.
RAPI_RISING	Selects the rising edge of measurement pulse as active edge.
RAPI_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_TRC_RISING	Selects the rising edge of measurement pulse as active edge in timer RC.
RAPI_TRC_FALLING	Selects the falling edge of measurement pulse as active edge in timer RC.
RAPI_TRC_BOTH	Selects both rising and falling edges of measurement pulse as active edge in timer RC.
RAPI_TRF_RISING	Selects the rising edge of measurement pulse as active edge in timer RF.
RAPI_TRF_FALLING	Selects the falling edge of measurement pulse as active edge in timer RF.
RAPI_TRF_BOTH	Selects both rising and falling edges of measurement pulse as active edge in timer RF.
RAPI_FILTER_F1	Uses the digital filter facility that has a sampling frequency f_1.
RAPI_FILTER_F8	Uses the digital filter facility that has a sampling frequency f_8.
RAPI_FILTER_F32	Uses the digital filter facility that has a sampling frequency f_{32}.
RAPI_TRC_FILTER_F1	Use sampling frequency f_1 for timer RC digital filter function.
RAPI_TRC_FILTER_F8	Use sampling frequency f_8 for timer RC digital filter function.
RAPI_TRC_FILTER_F32	Use sampling frequency f_{32} for timer RC digital filter function.
RAPI_TRC_FILTER_F	Use timer RC and the digital filtering function with the sampling frequency as that of the count source.
RAPI_INT3_TRIGGER	Selects the trigger facility of TC_{IN} input.
RAPI_FRING128	Selects the trigger facility of $f_{RING128}$ input.
RAPI_OVERFLOW	Enable overflow interrupt.

RAPI_INPUT_CAPTURE_A	Enable GRA input capture interrupt.
RAPI_INPUT_CAPTURE_B	Enable GRB input capture interrupt.
RAPI_INPUT_CAPTURE_C	Enable GRC input capture interrupt.
RAPI_INPUT_CAPTURE_D	Enable GRD input capture interrupt.
RAPI_COUNT_CLEAR_A	Selects GRA input capture to counter clear factor
RAPI_COUNT_CLEAR_B	Selects GRB input capture to counter clear factor
RAPI_COUNT_CLEAR_C	Selects GRC input capture to counter clear factor
RAPI_COUNT_CLEAR_D	Selects GRD input capture to counter clear factor
RAPI_COUNT_CLEAR_SYNC	Clear counter in sync with the synchronized other timer counter
RAPI_TIMER_SYNC	Synchronize timer on channels A and B.
RAPI_TRD_FILTER_F1	Use sampling frequency f1 for digital filter function
RAPI_TRD_FILTER_F8	Use sampling frequency f8 for digital filter function
RAPI_TRD_FILTER_F32	Use sampling frequency f32 for digital filter function
RAPI_TRD_FILTER_F	Use timer RD and the digital filtering function with the sampling frequency as that of the count source.

• Specifiable definition values when timer C is used (RAPI_TIMER_C specified)

(Count source) Specify one from [RAPI_F1, RAPI_F8, RAPI_F32, RAPI_FRING_FAST]. The default is RAPI_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF

(Measure pulse) Specify one from { RAPI_RISING, RAPI_FALLING, RAPI_BOTH }. The default value is RAPI_RISING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(Input trigger) Specify one from { RAPI_INT3_TRIGGER, RAPI_FRING128 }. The default value is RAPI_INT3_TRIGGE.

• Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)

(Count source) Specify one from [RAPI_TRC_F1, RAPI_TRC_F2, RAPI_TRC_F4, RAPI_TRC_F8, RAPI_TRC_F32, RAPI_TRC_F40M]. The default is RAPI_TRC_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPURE_A, RAPI_INPUT_CAPURE_B, RAPI_INPUT_CAPURE_C, or RAPI_INPUT_CAPURE_D, respectively. When not specifying interrupts, select “No interrupt requests.”

(Clock for digital filter) Specify one from [RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F]. The default is RAPI_TRC_FILTER_F32.

• Specifiable definition values when timer RD is used (RAPI_TIMER_RD0 ～ RAPI_TIMER_RD1 specified)

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPURE_A, RAPI_INPUT_CAPURE_B, RAPI_INPUT_CAPURE_C, or RAPI_INPUT_CAPURE_D, respectively. When not specifying interrupts, select “No interrupt requests.”

(Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively. If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_COUNT_CLEAR_SYNC.

(Synchronization) If timers on channels 0 and 1 are to be synchronized, select RAPI_TIMER_SYNC. When not specifying synchronization, select “Channels 0 and 1 operate independently.”

(Clock for digital filter) Specify one from [RAPI_TRD_FILTER_F1, RAPI_TRD_FILTER_F8, RAPI_TRD_FILTER_F32, RAPI_TRD_FILTER_F]. The default is RAPI_TRD_FILTER_F32.

• Specifiable definition values when timer RF is used (RAPI_TIMER_RF specified)

(Count source) Specify one from [RAPI_TRF_F1, RAPI_TRF_F8, RAPI_TRF_F32]. The default is RAPI_TRF_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF

(Measure pulse) Specify one from { RAPI_TRF_RISING, RAPI_TRF_FALLING, RAPI_TRF_BOTH }. The default value is RAPI_TRF_RISING

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32}. The default value is no filter.

(H8S)

RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD channel 0.
RAPI_TIMER_RD1	Uses timer RD channel 1.
RAPI_TIMER_RD2	Uses timer RD channel 2.
RAPI_TIMER_RD3	Uses timer RD channel 3.
RAPI_TIMER_RG	Uses timer RG.
RAPI_TRC_F1	Timer RC counts with internal clock φ.
RAPI_TRC_F2	Timer RC counts with internal clock φ/2.
RAPI_TRC_F4	Timer RC counts with internal clockφ/4.
RAPI_TRC_F8	Timer RC counts with internal clock φ/8.
RAPI_TRC_F32	Timer RC counts with internal clock φ/8.
RAPI_TRC_F40	Timer RC counts with internal clock φ40.
RAPI_TRD_F1	Timer RD counts with internal clock φ.
RAPI_TRD_F2	Timer RD counts with internal clock φ/2.
RAPI_TRD_F4	Timer RD counts with internal clockφ/4.
RAPI_TRD_F8	Timer RD counts with internal clock φ/8.
RAPI_TRD_F32	Timer RD counts with internal clock φ/32.
RAPI_TRD_F40	Timer RD counts with internal clock φ40.
RAPI_TRG_F1	Timer RG counts with internal clock φ.
RAPI_TRG_F2	Timer RG counts with internal clock φ/2.
RAPI_TRG_F4	Timer RG counts with internal clockφ/4.
RAPI_TRG_F8	Timer RG counts with internal clock φ/8.
RAPI_TRG_F32	Timer RG counts with internal clock φ/32.
RAPI_TRG_F40	Timer RG counts with internal clock φ40.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateInputCapture.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateInputCapture.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_INPUT_ CAPTURE_A	Enables GRA input capture interrupt.
RAPI_INPUT_ CAPTURE_B	Enables GRB input capture interrupt.
RAPI_INPUT_ CAPTURE_C	Enables GRC input capture interrupt.
RAPI_INPUT_ CAPTURE_D	Enables GRD input capture interrupt.
RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.
RAPI_TRC_COUNT_CLEAR_A	Selects GRA input capture to timer RC counter clearing source.
RAPI_TRD_COUNT_CLEAR_A	Selects GRA input capture to timer RD counter clearing source.
RAPI_TRD_COUNT_CLEAR_B	Selects GRB input capture to timer RD counter clearing source.

RAPI_TRD_COUNT_CLEAR_C	Selects GRC input capture to timer RD counter clearing source.
RAPI_TRD_COUNT_CLEAR_D	Selects GRD input capture to timer RD counter clearing source.
RAPI_TRD_COUNT_CLEAR_SYNC	Clear timer RD counter in sync with the synchronized other timer counter
RAPI_TRG_COUNT_CLEAR_A	Selects GRA input capture to timer RG counter clearing source.
RAPI_TRG_COUNT_CLEAR_B	Selects GRB input capture to timer RG counter clearing source.
RAPI_TIMER_SYNC	Synchronizes timer on channels 0 and 1.
RAPI_TRC_FILTER_F1	Use sampling frequency φ for timer RC digital filter function.
RAPI_TRC_FILTER_F8	Use sampling frequency φ/8 for timer RC digital filter function.
RAPI_TRC_FILTER_F32	Use sampling frequency φ/32 for timer RC digital filter function.
RAPI_TRC_FILTER_F	Use timer RC and the digital filtering function with the sampling frequency as that of the count source.
RAPI_TRD_FILTER_F1	Use sampling frequency φ for timer RD digital filter function.
RAPI_TRD_FILTER_F8	Use sampling frequency φ/8 for timer RD digital filter function.
RAPI_TRD_FILTER_F32	Use sampling frequency φ/32 for timer RD digital filter function.
RAPI_TRD_FILTER_F	Use timer RD and the digital filtering function with the sampling frequency as that of the count source.
RAPI_TRG_FILTER_F1	Use sampling frequency φ for timer RG digital filter function.
RAPI_TRG_FILTER_F8	Use sampling frequency φ/8 for timer RG digital filter function.
RAPI_TRG_FILTER_F32	Use sampling frequency φ/32 for timer RG digital filter function.
RAPI_TRG_FILTER_F	Use timer RG and the digital filtering function with the sampling frequency as that of the count source.

• Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)

(Count source) Specify one from [RAPI_TRC_F1, RAPI_TRC_F2, RAPI_TRC_F4, RAPI_TRC_F8, RAPI_TRC_F32, RAPI_TRC_F40]. The default is RAPI_TRC_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPURE_A, RAPI_INPUT_CAPURE_B, RAPI_INPUT_CAPURE_C, or RAPI_INPUT_CAPURE_D, respectively. When not specifying interrupts, select “No interrupt requests.”

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

(Counter clear) To specify GRA input capture for the cause of counter clear, specify RAPI_TRC_COUNT_CLEAR_A.

(Clock for digital filter) Specify one from [RAPI_TRC_FILTER_F1, RAPI_TRC_FILTER_F8, RAPI_TRC_FILTER_F32, RAPI_TRC_FILTER_F]. The default is RAPI_TRC_FILTER_F32.

• Specifiable definition values when timer RD is used (RAPI_TIMER_RD0-RAPI_TIMER_RD3 specified)

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40]. The default is RAPI_TRD_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPURE_A, RAPI_INPUT_CAPURE_B, RAPI_INPUT_CAPURE_C, or RAPI_INPUT_CAPURE_D, respectively. When not specifying interrupts, select “No interrupt requests.”

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

(Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively.
If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_COUNT_CLEAR_SYNC.

(Synchronization) If timers on channels 0 and 1 are to be synchronized, select RAPI_TIMER_SYNC. When not specifying synchronization, select “Channels 0 and 1 operate independently.”

(Clock for digital filter) Specify one from [RAPI_TRD_FILTER_F1, RAPI_TRD_FILTER_F8, RAPI_TRD_FILTER_F32, RAPI_TRD_FILTER_F]. The default is RAPI_TRD_FILTER_F32.

• Specifiable definition values when timer RG is used (RAPI_TIMER_RG specified)

(Count source) Specify one from [RAPI_TRG_F1, RAPI_TRG_F2, RAPI_TRG_F4, RAPI_TRG_F8, RAPI_TRG_F32, RAPI_TRG_F40]. The default is RAPI_TRG_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if input capture A or input capture B interrupt requests are enabled, specify RAPI_INPUT_CAPTURE_A or RAPI_INPUT_CAPTURE_B respectively. If no interrupts are specified, “No interrupt request” is set.

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

(Counter clear) To specify GRA or GRB input capture for the cause of counter clear, specify RAPI_COUNT_CLEAR_A or RAPI_COUNT_CLEAR_B respectively.

(Clock for digital filter) Specify one from [RAPI_TRG_FILTER_F1, RAPI_TRG_FILTER_F8, RAPI_TRG_FILTER_F32, RAPI_TRG_FILTER_F]. The default is RAPI_TRG_FILTER_F32.

(H8/300H)

RAPI_TIMER_W	Uses timer W.
RAPI_TIMER_Z0	Uses timer Z channel 0.
RAPI_TIMER_Z1	Uses timer Z channel 1.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD channel 0.
RAPI_TIMER_RD1	Uses timer RD channel 1.
RAPI_TIMER_RD2	Uses timer RD channel 2.
RAPI_TIMER_RD3	Uses timer RD channel 3.
RAPI_TW_F1	Timer W counts with internal clock φ.
RAPI_TW_F2	Timer W counts with internal clock φ/2.
RAPI_TW_F4	Timer W counts with internal clock φ/4.
RAPI_TW_F8	Timer W counts with internal clock φ/8.
RAPI_TZ_F1	Timer Z counts with internal clock φ.
RAPI_TZ_F2	Timer Z counts with internal clock φ/2.
RAPI_TZ_F4	Timer Z counts with internal clock φ/4.
RAPI_TZ_F8	Timer Z counts with internal clock φ/8.
RAPI_TRC_F1	Timer RC counts with internal clock φ.
RAPI_TRC_F2	Timer RC counts with internal clock φ/2.
RAPI_TRC_F4	Timer RC counts with internal clock φ/4.
RAPI_TRC_F8	Timer RC counts with internal clock φ/8.
RAPI_TRC_F32	Timer RC counts with internal clock φ/32.
RAPI_TRC_F40M	Timer RC counts with internal clock φ40M.
RAPI_TRD_F1	Timer RD counts with internal clock φ.
RAPI_TRD_F2	Timer RD counts with internal clock φ/2.
RAPI_TRD_F4	Timer RD counts with internal clockφ/4.
RAPI_TRD_F8	Timer RD counts with internal clock φ/8.
RAPI_TRD_F32	Timer RD counts with internal clock φ/32.
RAPI_TRD_F40M	Timer RD counts with internal clock φ40M.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateInputCapture.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateInputCapture.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_INPUT_ CAPTURE_A	Enables GRA input capture interrupt.
RAPI_INPUT_ CAPTURE_B	Enables GRB input capture interrupt.
RAPI_INPUT_ CAPTURE_C	Enables GRC input capture interrupt.
RAPI_INPUT_ CAPTURE_D	Enables GRD input capture interrupt.

RAPI_COUNT_CLEAR_A	Specifies GRA input capture for the cause of counter clear.
RAPI_COUNT_CLEAR_B	Specifies GRB input capture for the cause of counter clear.
RAPI_COUNT_CLEAR_C	Specifies GRC input capture for the cause of counter clear.
RAPI_COUNT_CLEAR_D	Specifies GRD input capture for the cause of counter clear.
RAPI_COUNT_CLEAR_SYNC	Clear counter in sync with the synchronized other timer counter
RAPI_TIMER_SYNC	Synchronizes timer on channels 0 and 1.
RAPI_TRC_FILTER_F1	Use sampling frequency φ for timer RC digital filter function
RAPI_TRC_FILTER_F8	Use sampling frequency φ/8 for timer RC digital filter function
RAPI_TRC_FILTER_F32	Use sampling frequency φ/32 for timer RC digital filter function
RAPI_TRC_FILTER_F	Use timer RC and the digital filtering function with the sampling frequency as that of the count source.
RAPI_TRD_FILTER_F1	Use sampling frequency φ for timer RD digital filter function
RAPI_TRD_FILTER_F8	Use sampling frequency φ/8 for timer RD digital filter function
RAPI_TRD_FILTER_F32	Use sampling frequency φ/32 for timer RD digital filter function
RAPI_TRD_FILTER_F	Use timer RD and the digital filtering function with the sampling frequency as that of the count source.

• Specifiable definition values when timer W is used (RAPI_TIMER_W specified)

(Count source) Specify one from { RAPI_TW_F1, RAPI_TW_F2, RAPI_TW_F4, RAPI_TW_F8 }. The default value is RAPI_TW_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPTURE_A, RAPI_INPUT_CAPTURE_B, RAPI_INPUT_CAPTURE_C, or RAPI_INPUT_CAPTURE_D, respectively. If no interrupts are specified, “No interrupt request” is set.

(Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively. If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_COUNT_CLEAR_SYNC.

• Specifiable definition values when timer Z is used (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)

(Count source) Specify one from { RAPI_TZ_F1, RAPI_TZ_F2, RAPI_TZ_F4, RAPI_TZ_F8 }. The default value is RAPI_TZ_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPTURE_A, RAPI_INPUT_CAPTURE_B, RAPI_INPUT_CAPTURE_C, or RAPI_INPUT_CAPTURE_D, respectively. If no interrupts are specified, “No interrupt request” is set.

(Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause of counter clear, specify RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively.

(Synchronization) If the timer is synchronized on channels 0 and 1, specify RAPI_TIMER_SYNC. If synchronization is not specified, “Channels 0 and 1 operate independently” is set.

• Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)

(Count source) Specify one from [RAPI_TZ_F1, RAPI_TZ_F2, RAPI_TZ_F4, RAPI_TZ_F8, RAPI_TZ_F32, RAPI_TZ_F40M]. The default is RAPI_TZ_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPURE_A, RAPI_INPUT_CAPURE_B, RAPI_INPUT_CAPURE_C, or RAPI_INPUT_CAPURE_D, respectively. When not specifying interrupts, select “No interrupt requests.”

(Clock for digital filter) Specify one from [RAPI_TRD_FILTER_F1, RAPI_TRD_FILTER_F8, RAPI_TRD_FILTER_F32, RAPI_TRD_FILTER_F]. The default is RAPI_TRD_FILTER_F32.

• Specifiable definition values when timer RD is used (RAPI_TIMER_RD specified)

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPURE_A, RAPI_INPUT_CAPURE_B, RAPI_INPUT_CAPURE_C, or RAPI_INPUT_CAPURE_D, respectively. When not specifying interrupts, select “No interrupt requests.”

(Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively. If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_COUNT_CLEAR_SYNC.

(Synchronization)	If timers on channels 0 and 1 are to be synchronized, select RAPI_TIMER_SYNC. When not specifying synchronization, select “Channels 0 and 1 operate independently.”
(Clock for digital filter)	Specify one from [RAPI_TRD_FILTER_F1, RAPI_TRD_FILTER_F8, RAPI_TRD_FILTER_F32, RAPI_TRD_FILTER_F]. The default is RAPI_TRD_FILTER_F32.

[data2]

(M16C)

Specify a pointer to the array in which the interrupt priority level is stored.

[0]: Specify the IC/OC base timer interrupt priority level (0–7).

[1]: Specify the IC/OC interrupt 0 priority level (0–7).

[2]: Specify the IC/OC interrupt 1 priority level (0–7).

(R8C)

• When using timer C, timer RC or timer RD (RAPI_TIMER_C, RAPI_TIMER_RC, or RAPI_TIMER_RD)

Specify a pointer to the variable that contains the interrupt priority level (0–7) to be set in the interrupt control register.

・When using timer timer RF (RAPI_TIMER_RF)

Specify a pointer to the array in which the interrupt priority level is stored.

[0]：Specify the timer RF interrupt priority level (0–7).

[1]：Specify the input capture interrupt priority level (0–7).

(H8S)

Specify the interrupt priority level (0-3) to be set in the interrupt priority register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)

Specify a pointer to the array in which the set value for the time measurement control register is stored.

[0]: Specify the set value for time measurement control register 0.

[1]: Specify the set value for time measurement control register 1.

[2]: Specify the set value for time measurement control register 2.

[3]: Specify the set value for time measurement control register 3.

[4]: Specify the set value for time measurement control register 4.

[5]: Specify the set value for time measurement control register 5.

[6]: Specify the set value for time measurement control register 6.

[7]: Specify the set value for time measurement control register 7.

For each element of the array, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

RAPI_IC_RISING	Selects the rising edge of measurement pulse as active edge.

RAPI_IC_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_IC_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_FILTER_F1_F2	Uses the digital filter facility that has a sampling frequency f_1 or f_2.
RAPI_FILTER_FBT1	Uses the digital filter facility that has a sampling frequency f_{BT1}.
RAPI_GATE	Uses a gate facility.
RAPI_GATE_CLEAR	Clears the gate facility upon a match of base timer and G1POk register.
RAPI_PRESCALER	Uses a prescaler facility.

• Specifiable definition values for time measurement registers 0-7

(Measurement pulse) Specify one from { RAPI_IC_RISING, RAPI_IC_FALLING, RAPI_IC_BOTH }. If no measurement pulses are specified, the time measurement register is set to “No time measurement performed.”

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_FBT1 }. If no filters are specified, “No digital filter” is set.

• Specifiable definition values for time measurement registers 6-7

(Gate) To use the gate facility, specify RAPI_GATE. If RAPI_GATE is not specified, “No facility unused” is set. Make sure RAPI_GATE_CLEAR is specified at the same time RAPI_GATE is specified.

(Prescaler) To use the prescaler facility, specify RAPI_PRESCALER. If RAPI_PRESCALER is not specified, “No prescaler facility” is set.

(R8C)

• When using timer C or timer RF (RAPI_TIMER_C or RAPI_TIMER_RF)

Specify 0.

• When using timer RC (RAPI_TIMER_RC)

Specify a pointer to the array in which each active edge setting is stored.

[0]: Specify the active edge of TRCIOA pin.

[1]: Specify the active edge of TRCIOB pin.

[2]: Specify the active edge of TRCIOC pin.

[3]: Specify the active edge of TRCIOD pin.

For each element of the array, one of [RAPI_TRC_RISING, RAPI_TRC_FALLING, RAPI_TRC_BOTH] can be set as the active edge of measured pulses.

Furthermore, if the digital filter function is enabled, select RAPI_FILTER_ON.

For the elements corresponding to unused channels, set 0.

RAPI_TRC_RISING	Selects the rising edge of measurement pulse as active edge.
RAPI_TRC_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_TRC_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_FILTER_ON	Selects digital filter on. If not select “RAPI_FILTER_ON”, “No digital filter” is set.

• When using timer RD (RAPI_TIMER_RD0 - RAPI_TIMER_RD1)

Specify a pointer to the array in which each active edge setting is stored.

[0]: Specify the active edge of TRDIOAi (i=0,1) pin.

[1]: Specify the active edge of TRDIOBi (i=0,1) pin.

[2]: Specify the active edge of TRDIOCi (i=0,1) pin.

[3]: Specify the active edge of TRDIODi (i=0,1) pin.

For each element of the array, one of [RAPI_TRD_RISING, RAPI_TRD_FALLING, RAPI_TRD_BOTH] can be set as the active edge of measured pulses.

Furthermore, if the digital filter function is enabled, select RAPI_FILTER_ON.

To set the f0C0128 signal for the TRDI0A0 pin on channel 0, specify RAPI_F0C0128. For the elements corresponding to unused channels, set 0.

RAPI_TRD_RISING	Selects the rising edge of measurement pulse as active edge.
RAPI_TRD_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_TRD_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_FILTER_ON	Selects digital filter on. If not select “RAPI_FILTER_ON”, “No digital filter” is set.
RAPI_FOCO128	Selects fOCO128 as TRDIOA0 pin input. Specifiable only active edge of channel0 TRDIOA0 pin.

(H8S)

Specify a pointer to the array in which each active edge setting is stored.

• When using timer RC or timer RD

(RAPI_TIMER_RC, RAPI_TIMER_RD0 or RAPI_TIMER_RD1)

[0]: Specify the active edge of FTIOA pin.

[1]: Specify the active edge of FTIOB pin.

[2]: Specify the active edge of FTIOC pin.

[3]: Specify the active edge of FTIOD pin.

• When using timer RG (RAPI_TIMER_RG)

[0]: Specify the active edge of FTIOA pin.

[1]: Specify the active edge of FTIOB pin.

For each element of the array, one of the following definition values { RAPI_RISING, RAPI_FALLING, RAPI_BOTH } can be set.

For the array elements corresponding to unused channels, set 0.

RAPI_RISING	Selects the rising edge of measurement pulse as active edge.
RAPI_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_FILTER_ON	Selects digital filter on. If not select “RAPI_FILTER_ON”, “No digital filter” is set.

(H8/300H)

Specify a pointer to the array in which each active edge setting is stored.

[0]: Specify the active edge of TRDIOA pin.

[1]: Specify the active edge of TRDIOB pin.

[2]: Specify the active edge of TRDIOC pin.

[3]: Specify the active edge of TRDIOD pin.

For each element of the array, one of the following definition values { RAPI_RISING, RAPI_FALLING, RAPI_BOTH } can be set.

For the array elements corresponding to unused channels, set 0.

RAPI_RISING	Selects the rising edge of measurement pulse as active edge.

RAPI_FALLING	Selects the falling edge of measurement pulse as active edge.
RAPI_BOTH	Selects both rising and falling edges of measurement pulse as active edge.
RAPI_FILTER_ON	Selects digital filter on. If not select “RAPI_FILTER_ON”, “No digital filter” is set.

[data4]

(M16C)

Specify a pointer to the array in which the set value for each register or timer S is stored.

[0]: Specify the set value for the facility select and facility enable registers.
Specify the channel for which the time measurement facility is enabled.

[1]: Specify the set value for interrupt enable register 0.
Specify the channel for which IO/CO interrupt 0 request is enabled.

[2]: Specify the set value for interrupt enable register 1.
Specify the channel for which IO/CO interrupt 1 request is enabled.

[3]: Specify the set value for the count source divide-by-N register.
Specify the value of ‘n’ in the formula “count source divided by (n ÷ 1)” in 8 bits.

[4]: Specify the set value for time measurement prescaler register 6.
Specify the value of ‘n’ in the prescaler period “n ÷ 1” in 8 bits.

[5]: Specify the set value for time measurement prescaler register 7.
Specify the value of ‘n’ in the prescaler period “n ÷ 1” in 8 bits.

For the channels to be specified in each array element, use the following definition values. To specify multiple definition values at the same time, use the symbol “|” to separate each specified value. If 0 is specified, the value 0 is set in the corresponding register.

RAPI_CHANNEL0	Selects channel 0.
RAPI_CHANNEL1	Selects channel 1.
RAPI_CHANNEL2	Selects channel 2.
RAPI_CHANNEL3	Selects channel 3.
RAPI_CHANNEL4	Selects channel 4.
RAPI_CHANNEL5	Selects channel 5.
RAPI_CHANNEL6	Selects channel 6.
RAPI_CHANNEL7	Selects channel 7.

(R8C) (H8S) (H8/300H)

Specify 0.

[data5]

(M16C)

Specify a pointer to the array in which the callback function is stored.

[0]: Specify a pointer to the callback function for IC/OC base timer interrupt. If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for IC/OC interrupt 0. If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for IC/OC interrupt 1. If this pointer is not specified, 0 is set.

(R8C)

• When using timer C, timer RC or timer RD (RAPI_TIMER_C, RAPI_TIMER_RC, or RAPI_TIMER_RD0 - RAPI_TIMER_RD1)

Specify a pointer to the variable in which the callback function is stored.

If this pointer is not specified, RAPI_NULL is set.

• When using timer RF (RAPI_TIMER_RF)

Specify a pointer to the array in which the callback function is stored.

[0]: Specify a pointer to the callback function for timer RF interrupt.
If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for input capture interrupt.
If this pointer is not specified, 0 is set.

(H8S)

Specify a pointer to the array in which the callback function is stored.

• When using timer RC or timer RD (RAPI_TIMER_RC or RAPI_TIMER_RD0 - RAPI_TIMER_RD3)

[0]: Specify a pointer to the callback function for counter overflow interrupt.
If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for input capture interrupt A.
If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for input capture interrupt B.
If this pointer is not specified, 0 is set.

[3]: Specify a pointer to the callback function for input capture interrupt C.
If this pointer is not specified, 0 is set.

[4]: Specify a pointer to the callback function for input capture interrupt D.
If this pointer is not specified, 0 is set.

• When using timer RG (RAPI_TIMER_RG)

[0]: Specify a pointer to the callback function for counter overflow interrupt.
If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for input capture interrupt A.
If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for input capture interrupt B.
If this pointer is not specified, 0 is set.

(H8/300H)

Specify a pointer to the array in which the callback function is stored. If this pointer is not specified, RAPI_NULL is set.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (input capture mode)

Reference

EnableInputCapture, DestroyInputCapture, GetInputCapture

Remark

- If an undefined value is specified in the first, third and fourth arguments, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When use CPU which has no digital filter function, cannot specify digital filter setting.
- When used for the H8S, H8/300H, this API specify when freeing it from module stanby state.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc( void ){}

void func( void )
{
    /* Set up timer C as imput capture mode */
    __CreateInputCapture(
        RAPI_TIMER_C|RAPI_TIMER_ON|RAPI_BOTH|RAPI_F32| RAPI_FRING128,
        5, TimerIntFunc);
}
```

__EnableInputCapture

Synopsis

<Control operation of input capture mode>

Boolean __EnableInputCapture(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Controls operation of the timer that is set to specified input capture mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_S	Selects timer S.
RAPI_TIMER_ON	Sets the timer that is set to input capture mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to input capture mode to stop operating.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RF	Selects timer RF.
RAPI_TIMER_ON	Sets the timer that is set to input capture mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to input capture mode to stop operating.

(H8S)

RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.
RAPI_TIMER_RG	Selects timer RG.
RAPI_TIMER_ON	Sets the timer that is set to input capture mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to input capture mode to stop operating.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RD0.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.
RAPI_TIMER_ON	Sets the timer that is set to input capture mode to start operating.

RAPI_TIMER_OFF	Sets the timer that is set to input capture mode to stop operating.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (input capture mode)

Reference

CreateInputCapture, DestroyInputCapture, GetInputCapture

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Enable timer C as input capture mode */
    __EnableInputCapture( RAPI_TIMER_C|RAPI_TIMER_ON );
}
```

__DestroyInputCapture

Synopsis

<Discard settings of input capture mode>

Boolean __DestroyInputCapture(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Discards settings of the timer or Intelligent I/O that is set to specified input capture mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_S	Selects timer S.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RF	Selects timer RF.

(H8S)

RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channe 0.
RAPI_TIMER_RD1	Selects timer RD channe 1.
RAPI_TIMER_RD2	Selects timer RD channe 2.
RAPI_TIMER_RD3	Selects timer RD channe 3.
RAPI_TIMER_RG	Selects timer RG.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (input capture mode)

Reference

__CreateInputCapture, __EnableInputCapture, __GetInputCapture

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API places a specified timer into module stanby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Destroy the setting of timer C as input capture mode */
    __DestroyInputCapture( RAPI_TIMER_C );
}
```

__GetInputCapture

Synopsis

<Get input capture mode counter value>

Boolean __GetInputCapture(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)

Description

Gets the counter value of the timer or Intelligent I/O that is set to specified input capture mode.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_S	Selects timer S.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RF	Selects timer RF.

(H8S)

RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channe 0.
RAPI_TIMER_RD1	Selects timer RD channe 1.
RAPI_TIMER_RD2	Selects timer RD channe 2.
RAPI_TIMER_RD3	Selects timer RD channe 3.
RAPI_TIMER_RG	Selects timer RG.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.

[data2]

(M16C)

Specify a pointer to the array in which the acquired counter value is stored.

[0]: Stores the value of base timer register 0.

[1]: Stores the value of time measurement register 0.

[2]: Stores the value of time measurement register 1.

[3]: Stores the value of time measurement register 2.
[4]: Stores the value of time measurement register 3.
[5]: Stores the value of time measurement register 4.
[6]: Stores the value of time measurement register 5.
[7]: Stores the value of time measurement register 6.
[8]: Stores the value of time measurement register 7.

(R8C)

Specify a pointer to the array in which the acquired counter value is stored.

• When using timer C or timer RF (RAPI_TIMER_C or RAPI_TIMER_RF)

[0]: Stores the value of timer C counter.
[1]: Stores the value of capture & compare 0 register.

• When using timer RC or timer RD
(RAPI_TIMER_RC or RAPI_TIMER_RD0- RAPI_TIMER_RD1)

[0]: Stores the value of timer counter.
[1]: Stores the value of general register A.
[2]: Stores the value of general register B.
[3]: Stores the value of general register C.
[4]: Stores the value of general register D.

(H8S)

• When using timer RC or timer RD
(RAPI_TIMER_RC or RAPI_TIMER_RD0- RAPI_TIMER_RD3)

[0]: Stores the value of the timer counter.
[1]: Stores the value of general register A.
[2]: Stores the value of general register B.
[3]: Stores the value of general register C.
[4]: Stores the value of general register D.

• When using timer RG (RAPI_TIMER_RG)

[0]: Stores the value of the timer counter.
[1]: Stores the value of general register A.
[2]: Stores the value of general register B.

(H8/300H)

[0]: Stores the value of the timer counter.
[1]: Stores the value of general register A.
[2]: Stores the value of general register B.
[3]: Stores the value of general register C.
[4]: Stores the value of general register D.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (input capture mode)

Reference

__CreateInputCapture, __EnableInputCapture, __DestroyInputCapture

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    unsigned int data[2];

    /* Get the counter of timer C as input capture mode */
    __GetInputCapture( RAPI_TIMER_C, data );
}
```

__CreateOutputCompare

Synopsis

<Set output compare mode>

Boolean __CreateOutputCompare(unsigned long data1, unsigned int* data2, unsigned int* data3, unsigned int* data4, void data5)**

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)
data5	Setup data 5 (content differs with MCU type)

Description

Sets a specified timer or Intelligent I/O to output compare mode.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_S	Uses timer S.
RAPI_F1	Selects f_1 for the count source.
RAPI_F2	Selects f_2 for the count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateOutputCompare.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateOutputCompare.
RAPI_OVERFLOW_BIT14	Selects overflow of bit 14 for the base timer interrupt.
RAPI_OVERFLOW_BIT15	Selects overflow of bit 15 for the base timer interrupt.

• Specifiable definition values when timer S is used (RAPI_TIMER_S specified)

(Count source) Specify one from { RAPI_F1, RAPI_F2 }. The default value is RAPI_F2.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Base timer) Specify one from { RAPI_OVERFLOW_BIT14, RAPI_OVERFLOW_BIT15 }. The default value is RAPI_OVERFLOW_BIT15.

(R8C)

RAPI_TIMER_C	Uses timer C.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD0 channe 0.
RAPI_TIMER_RD1	Uses timer RD0 channe 1.
RAPI_TIMER_RE	Uses timer RE.
RAPI_TIMER_RF	Uses timer RF.
RAPI_F1	Selects f_1 for the count source.
RAPI_F8	Selects f_8 for the count source.
RAPI_F32	Selects f_{32} for the count source.
RAPI_FRING_FAST	Selects $f_{RING-fast}$ for the count source.
RAPI_TRC_F1	Selects f_1 for the timer RC count source.
RAPI_TRC_F2	Selects f_2 for the timer RC count source.

RAPI_TRC_F4	Selects f_4 for the timer RC count source.
RAPI_TRC_F8	Selects f_8 for the timer RC count source.
RAPI_TRC_F32	Selects f_{32} for the timer RC count source.
RAPI_TRC_F40M	Selects fOCO40M for the timer RC count source.
RAPI_TRD_F1	Selects f_1 for the timer RD count source.
RAPI_TRD_F2	Selects f_2 for the timer RD count source.
RAPI_TRD_F4	Selects f_4 for the timer RD count source.
RAPI_TRD_F8	Selects f_8 for the timer RD count source.
RAPI_TRD_F32	Selects f_{32} for the timer RD count source.
RAPI_TRD_F40M	Selects fOCO40M for the timer RD count source.
RAPI_TRE_F4	Selects f_4 for the timer RE count source.
RAPI_TRE_F8	Selects f_8 for the timer RE count source.
RAPI_TRE_F32	Selects f_{32} for the timer RE count source.
RAPI_TRE_FC4	Selects fC4 for the timer RE count source.
RAPI_TRF_F1	Selects f_1 for the timer RF count source.
RAPI_TRF_F8	Selects f_8 for the timer RF count source.
RAPI_TRF_F32	Selects f_{32} for the timer RF count source.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateOutputCompare.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateOutputCompare.
RAPI_TIMER_OVERFLOW	Enables the overflow interrupt.
RAPI_COMPARE_MATCH	Enables the comparematch interrupt.
RAPI_COMPARE_MATCH_A	Enables the GRA compare match interrupt.
RAPI_COMPARE_MATCH_B	Enables the GRB compare match interrupt.
RAPI_COMPARE_MATCH_C	Enables the GRC compare match interrupt.
RAPI_COMPARE_MATCH_D	Enables the GRD compare match interrupt.
RAPI_COUNT_CLEAR_A	Specifies GRA compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_B	Specifies GRB compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_C	Specifies GRC compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_D	Specifies GRD compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_SYNC	Clear counter in sync with the synchronized other timer counter
RAPI_COUNT_CLEAR_CMP1	Clears TRCCR1 to H'0000 through a compare match of GRA.
RAPI_TIMER_SYNC	Synchronize timer on channels 0 and 1.
RAPI_CMP00_DISABLE	Disables CMP output from $CMP0_0$.
RAPI_CMP01_DISABLE	Disables CMP output from $CMP0_1$.
RAPI_CMP02_DISABLE	Disables CMP output from $CMP0_2$.
RAPI_CMP10_DISABLE	Disables CMP output from $CMP1_0$.

RAPI_CMP11_DISABLE	Disables CMP output from $CMP1_1$.
RAPI_CMP12_DISABLE	Disables CMP output from $CMP1_2$.
RAPI_CMP00_ENABLE	Enables CMP output from $CMP0_0$.
RAPI_CMP01_ENABLE	Enables CMP output from $CMP0_1$.
RAPI_CMP02_ENABLE	Enables CMP output from $CMP0_2$.
RAPI_CMP10_ENABLE	Enables CMP output from $CMP1_0$.
RAPI_CMP11_ENABLE	Enables CMP output from $CMP1_1$.
RAPI_CMP12_ENABLE	Enables CMP output from $CMP1_2$.
RAPI_TRFO00_DISABLE	Disables $TRFO0_0$ output
RAPI_TRFO01_DISABLE	Disables $TRFO0_1$ output
RAPI_TRFO02_DISABLE	Disables $TRFO0_2$ output
RAPI_TRFO10_DISABLE	Disables $TRFO1_0$ output
RAPI_TRFO11_DISABLE	Disables $TRFO1_1$ output
RAPI_TRFO12_DISABLE	Disables $TRFO1_2$ output
RAPI_TRFO00_ENABLE	Enables $TRFO0_0$ output
RAPI_TRFO01_ENABLE	Enables $TRFO0_1$ output
RAPI_TRFO02_ENABLE	Enables $TRFO0_2$ output
RAPI_TRFO10_ENABLE	Enables $TRFO1_0$ output
RAPI_TRFO11_ENABLE	Enables $TRFO1_1$ output
RAPI_TRFO12_ENABLE	Enables $TRFO1_2$ output
RAPI_OUTPUT_ REVERSE_0	When using timer C, Inverts CMP output from $CMP0_0$ through $CMP0_2$. When using timer RF, Inverts TRFO output from $TRFO0_0$ through $TRFO0_2$.
RAPI_OUTPUT_ REVERSE_1	When using timer C, Inverts CMP output from $CMP1_0$ through $CMP1_2$. When using timer RF, Inverts TRFO output from $TRFO1_0$ through $TRFO1_2$.
RAPI_RELOAD	Sets TC register to “0x0000” when compare 1 matches.
RAPI_UNCHANGE_0	When using timer C or timer RF, Holds CMP output level at compare 0 match.
RAPI_REVERSE_0	When using timer C or timer RF, Inverts CMP output level at compare 0 match.
RAPI_L_0	When using timer C or timer RF, Sets CMP output level to “L” at compare 0 match.
RAPI_H_0	When using timer C or timer RF, Sets CMP output level to “H” at compare 0 match.
RAPI_UNCHANGE_1	When using timer C or timer RF, Holds CMP output level at compare 1 match.
RAPI_REVERSE_1	When using timer C or timer RF, Inverts CMP output level at compare 1 match.
RAPI_L_1	When using timer C or timer RF, Sets CMP output level to “L” at compare 1 match.

RAPI_H_1	When using timer C or timer RF, Sets CMP output level to “H” at compare 1 match.
RAPI_OUTPUT_DISABLE	Disable output.
RAPI_OUTPUT_F2	Specifies f2 output for output function.
RAPI_OUTPUT_F4	Specifies f4 output for output function.
RAPI_OUTPUT_F8	Specifies f8 output for output function.
RAPI_OUTPUT_COMPARE	Specifies compare output for output function.
RAPI_4BIT_COUNTER	Uses 4 bit counter.
RAPI_COMPARE_MATCH_A_STOP	Stop count when GRA compare match occur
RAPI_STOP	Stop count when clear TSTART bit.
RAPI_STOP_OUTPUT	Holds output level before count stops.
RAPI_STOP_H	“H” output when count stops.
RAPI_STOP_L	“L” output when count stops.

• Specifiable definition values when timer C is used (RAPI_TIMER_C specified)

(Count source)	Specify one from { RAPI_F1, RAPI_F8, RAPI_F32, RAPI_FRING_FAST }. The default value is RAPI_F1.
(Operating states set)	Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.
($CMP0_0$ output)	Specify one from { RAPI_CMP00_DISABLE, RAPI_CMP00_ENABLE }. The default value is RAPI_CMP00_DISABLE.
($CMP0_1$ output)	Specify one from { RAPI_CMP01_DISABLE, RAPI_CMP01_ENABLE }. The default value is RAPI_CMP01_DISABLE.
($CMP0_2$ output)	Specify one from { RAPI_CMP02_DISABLE, RAPI_CMP02_ENABLE }. The default value is RAPI_CMP02_DISABLE.
($CMP1_0$ output)	Specify one from { RAPI_CMP10_DISABLE, RAPI_CMP10_ENABLE }. The default value is RAPI_CMP10_DISABLE.
($CMP1_1$ output)	Specify one from { RAPI_CMP11_DISABLE, RAPI_CMP11_ENABLE }. The default value is RAPI_CMP11_DISABLE.
($CMP1_2$ output)	Specify one from { RAPI_CMP12_DISABLE, RAPI_CMP12_ENABLE }. The default value is RAPI_CMP12_DISABLE.
(CMP0 output inversion)	To invert CMP0 output, specify RAPI_OUTPUT_REVERSE_0. If RAPI_OUTPUT_REVERSE_0 is not specified, “CMP0 output not inverted” is set.
(CMP1 output inversion)	To invert CMP1 output, specify RAPI_OUTPUT_REVERSE_1. If RAPI_OUTPUT_REVERSE_1 is not specified, “CMP1 output not inverted” is set.
(TC reload)	To reload TC register, specify RAPID_RELOAD. If RAPID_RELOAD is not specified, “No reload” is set.
(CMP0 output mode)	Specify one from { RAPI_UNCHANGE_0, RAPI_REVERSE_0, RAPI_L_0, RAPI_H_0 }. The default value is RAPI_UNCHANGE_0.
(CMP1 output mode)	Specify one from { RAPI_UNCHANGE_1, RAPI_REVERSE_1, RAPI_L_1, RAPI_H_1 }. The default value is RAPI_UNCHANGE_1.

• Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)

(Count source) Specify one from [RAPI_TRC_F1, RAPI_TRC_F2, RAPI_TRC_F4, RAPI_TRC_F8, RAPI_TRC_F32, RAPI_TRC_F40M]. The default is RAPI_TRC_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW.
If compare match A interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A.
If compare match B interrupt requests are enabled, specify RAPI_COMPARE_MATCH_B.
If compare match C interrupt requests are enabled, specify RAPI_COMPARE_MATCH_C.
If compare match D interrupt requests are enabled, specify RAPI_COMPARE_MATCH_D.
If no interrupts are specified, “No interrupt request” is set.

(Counter clear) To specify GRA compare match for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A.

• Specifiable definition values when timer RD is used (RAPI_TIMER_RD specified)

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW.
If input capture A, input capture B, input capture C, or input capture D interrupt requests are enabled, specify RAPI_INPUT_CAPURE_A, RAPI_INPUT_CAPURE_B, RAPI_INPUT_CAPURE_C, or RAPI_INPUT_CAPURE_D, respectively. When not specifying interrupts, select “No interrupt requests.”

(Counter clear) To specify GRA, GRB, GRC, or GRD input capture for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively. If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_COUNT_CLEAR_SYNC.

(Synchronization) If timers on channels 0 and 1 are to be synchronized, select RAPI_TIMER_SYNC. When not specifying synchronization, select “Channels 0 and 1 operate independently.”

(count stop) Specify one from [RAPI_COMPARE_MATCH_A_STOP, RAPI_STOP]. The default is RAPI_STOP.

• Specifiable definition values when timer RE is used (RAPI_TIMER_RE specified)

(Count source) Specify one from [RAPI_TRE_F4, RAPI_TRE_F8, RAPI_TRE_F32, RAPI_TRD_FC4]. The default is RAPI_TRD_F4.

(Operating states set)	Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF
(Interrupt)	If compare match interrupt requests are enabled, specify RAPI_COMPARE_MATCH. When not specifying interrupts, select “No interrupt requests.”
(Output function)	Specify one from [RAPI_OUTPUT_DISABLE, RAPI_OUTPUT_F2, RAPI_OUTPUT_F4, RAPI_OUTPUT_F8, RAPI_OUTPUT_COMPARE]. The default is RAPI_OUTPUT_ DISABLE. For the R8C/28 and 29, this API cannot be used.
(4-bit Counter)	If use 4 bit counter, spcify RAPI_4BIT_COUNTER. The default is 8 bit counter.

• Specifiable definition values when timer RF is used (RAPI_TIMER_RF specified)

(Count source)	Specify one from [RAPI_TRF_F1, RAPI_TRF_F8, RAPI_TRF_F32]. The default is RAPI_TRF_F1.
(Operating states set)	Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF
($TRFO0_0$ output)	Specify one from { RAPI_TRFO00_DISABLE, RAPI_TRFO00_ENABLE }. The default is RAPI_TRFO00_DISABLE.
($TRFO0_1$ output)	Specify one from { RAPI_TRFO01_DISABLE, RAPI_TRFO01_ENABLE }. The default is RAPI_TRFO01_DISABLE.
($TRFO0_2$ output)	Specify one from { RAPI_TRFO02_DISABLE, RAPI_TRFO02_ENABLE }. The default is RAPI_TRFO02_DISABLE.
($TRFO1_0$ output)	Specify one from { RAPI_TRFO10_DISABLE, RAPI_TRFO10_ENABLE }. The default is RAPI_TRFO10_DISABLE.
($TRFO1_1$ output)	Specify one from { RAPI_TRFO11_DISABLE, RAPI_TRFO11_ENABLE }. The default is RAPI_TRFO11_DISABLE.
($TRFO1_2$ output)	Specify one from { RAPI_TRFO12_DISABLE, RAPI_TRFO12_ENABLE }. The default is RAPI_TRFO12_DISABLE.
(TRFO0 inverse output)	To inverts TRFO0 output level, select RAPI_OUTPUT_REVERSE_0. RAPI_OUTPUT_REVERSE_0 is not specified,”Holds output level” is set.
(TRFO1 inverse output)	To inverts TRFO1 output level, select RAPI_OUTPUT_REVERSE_1. RAPI_OUTPUT_REVERSE_1 is not specified,”Holds output level” is set.
(Counter clear)	To specify compare match 1 for the cause for which the counter is cleared, select RAPI_COUNT_CLEAR_CMP1.

	RAPI_COUNT_CLEAR_CMP1 is not specified," Disable clearing" is set.
(TRFO0 output mode)	Specify one from { RAPI_UNCHANGE_0, RAPI_REVERSE_0, RAPI_L_0, RAPI_H_0 }. The default is RAPI_UNCHANGE_0.
(TRFO1 output mode)	Specify one from { RAPI_UNCHANGE_1, RAPI_REVERSE_1, RAPI_L_1, RAPI_H_1 }. The default is RAPI_UNCHANGE_1.
(Output level when count stops)	Specify one from { RAPI_STOP_OUTPUT, RAPI_STOP_H, RAPI_STOP_L }. The default is RAPI_STOP_OUTPUT.

(H8S)

RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD channel 0.
RAPI_TIMER_RD1	Uses timer RD channel 1.
RAPI_TIMER_RD2	Uses timer RD channel 2.
RAPI_TIMER_RD3	Uses timer RD channel 3.
RAPI_TIMER_RE	Uses timer RE.
RAPI_TIMER_RG	Uses timer RG.
RAPI_TRC_F1	Timer RC counts with internal clock φ.
RAPI_TRC_F2	Timer RC counts with internal clock φ/2.
RAPI_TRC_F4	Timer RC counts with internal clock φ/4.
RAPI_TRC_F8	Timer RC counts with internal clock φ/8.
RAPI_TRC_F32	Timer RC counts with internal clock φ/8.
RAPI_TRC_F40	Timer RC counts with internal clock φ40.
RAPI_TRD_F1	Timer RD counts with internal clock φ.
RAPI_TRD_F2	Timer RD counts with internal clock φ/2.
RAPI_TRD_F4	Timer RD counts with internal clock φ/4.
RAPI_TRD_F8	Timer RD counts with internal clock φ/8.
RAPI_TRD_F32	Timer RD counts with internal clock φ/32.
RAPI_TRD_F40	Timer RD counts with internal clock φ40.
RAPI_TRE_F4	Timer RE counts with internal clock φ/4.
RAPI_TRE_F8	Timer RE counts with internal clock φ/8.
RAPI_TRE_F32	Timer RE counts with internal clock φ/32.
RAPI_TRE_FSUB	Timer RE counts with Sub-oscillator output clock.
RAPI_TRG_F1	Timer RG counts with internal clock φ.
RAPI_TRG_F2	Timer RG counts with internal clock φ/2.
RAPI_TRG_F4	Timer RG counts with internal clock φ/4.
RAPI_TRG_F8	Timer RG counts with internal clock φ/8.
RAPI_TRG_F32	Timer RG counts with internal clock φ/32.
RAPI_TRG_F40	Timer RG counts with internal clock φ40.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateOutputCompare.

RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateOutputCompare.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_COMPARE_MATCH	Enables compare match interrupt.
RAPI_COMPARE_MATCH_A	Enables GRA compare match interrupt.
RAPI_COMPARE_MATCH_B	Enables GRB compare match interrupt.
RAPI_COMPARE_MATCH_C	Enables GRC compare match interrupt.
RAPI_COMPARE_MATCH_D	Enables GRD compare match interrupt.
RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.
RAPI_TRC_COUNT_CLEAR_A	Specifies GRA compare match for the cause of Timer RC counter clear.
RAPI_TRD_COUNT_CLEAR_A	Specifies GRA compare match for the cause of Timer RD counter clear.
RAPI_TRD_COUNT_CLEAR_B	Specifies GRB compare match for the cause of Timer RD counter clear.
RAPI_TRD_COUNT_CLEAR_C	Specifies GRC compare match for the cause of Timer RD counter clear.
RAPI_TRD_COUNT_CLEAR_D	Specifies GRD compare match for the cause of Timer RD counter clear.
RAPI_TRD_COUNT_CLEAR_SYNC	Clear Timer RD counter in sync with the synchronized other timer counter
RAPI_TRG_COUNT_CLEAR_A	Specifies GRA compare match for the cause of Timer RG counter clear.
RAPI_TRG_COUNT_CLEAR_B	Specifies GRB compare match for the cause of Timer RG counter clear.
RAPI_TIMER_SYNC	Synchronizes timer on channels 0 and 1.
RAPI_OUTPUT_DISABLE	Disable output.
RAPI_OUTPUT_F2	Specifies φ/2 output for output function.
RAPI_OUTPUT_F4	Specifies φ/4 output for output function.
RAPI_OUTPUT_F8	Specifies φ/8 output for output function.
RAPI_OUTPUT_COMPARE	Specifies compare output for output function.
RAPI_4BIT_COUNTER	Uses 4 bit counter.

• Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)

(Count source) Specify one from [RAPI_TRC_F1, RAPI_TRC_F2, RAPI_TRC_F4, RAPI_TRC_F8, RAPI_TRC_F32, RAPI_TRC_F40]. The default is RAPI_TRC_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A, RAPI_COMPARE_MATCH_B, RAPI_COMPARE_MATCH_C, or RAPI_COMPARE_MATCH_D, respectively. When not specifying interrupts, select “No interrupt requests.”

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

(Counter clear) To specify GRA compare match for the cause of counter clear, specify RAPI_TRC_COUNT_CLEAR_A.

• Specifiable definition values when timer RD is used (RAPI_TIMER_RD0 to RAPI_TIMER_RD3 specified)

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40]. The default is RAPI_TRD_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A, RAPI_COMPARE_MATCH_B, RAPI_COMPARE_MATCH_C, or RAPI_COMPARE_MATCH_D, respectively. If no interrupts are specified, “No interrupt request” is set.

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.

(Counter clear) To specify GRA compare match for the cause for which the counter is cleared, select RAPI_TRD_COUNT_CLEAR_A.
To specify GRB compare match for the cause for which the counter is cleared, select RAPI_TRD_COUNT_CLEAR_B.
To specify GRC compare match for the cause for which the counter is cleared, select RAPI_TRD_COUNT_CLEAR_C.
To specify GRD compare match for the cause for which the counter is cleared, select RAPI_TRD_COUNT_CLEAR_D.
If cleared at the same time a synchronously operating counter on another channel is cleared, select RAPI_TRD_COUNT_CLEAR_SYNC.

(Synchronization) If the timer is synchronized on channels 0 and 1, specify RAPI_TIMER_SYNC. If synchronization is not specified, “Channels 0 and 1 operate independently” is set.

• Specifiable definition values when timer RE is used (RAPI_TIMER_RE specified)

(Count source) Specify one from [RAPI_TRE_F4, RAPI_TRE_F8, RAPI_TRE_F32, RAPI_TRE_FSUB]. The default is RAPI_TRE_F4.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If compare match interrupt requests are enabled, specify RAPI_COMPARE_MATCH.
When not specifying interrupts, select “No interrupt requests.”

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }.
The default value is RAPI_INT_MODE_0.

(Output function) Specify one from { RAPI_OUTPUT_DISABLE, RAPI_OUTPUT_F2, RAPI_OUTPUT_F4, RAPI_OUTPUT_F8, RAPI_OUTPUT_COMPARE }
The default is RAPI_OUTPUT_DISABLE.

(4-bit Counter) If use 4 bit counter, spcify RAPI_4BIT_COUNTER. The default is 8 bit counter.

• Specifiable definition values when timer RG is used (RAPI_TIMER_RG specified)

(Count source) Specify one from [RAPI_TRG_F1, RAPI_TRG_F2, RAPI_TRG_F4, RAPI_TRG_F8, RAPI_TRG_F32, RAPI_TRG_F40]. The default is RAPI_TRG_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW.
If compare match A interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A.
If compare match B interrupt requests are enabled, specify RAPI_COMPARE_MATCH_B.
When not specifying interrupts, select “No interrupt requests.”

(Interrupt control mode) Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }.
The default value is RAPI_INT_MODE_0.

(Counter clear) To specify GRA compare match for the cause for which the counter is cleared, select RAPI_TRG_COUNT_CLEAR_A.
To specify GRB compare match for the cause for which the counter is cleared, select RAPI_TRG_COUNT_CLEAR_B.

(H8/300H)

RAPI_TIMER_W	Uses timer W.
RAPI_TIMER_Z0	Uses timer Z channel 0.
RAPI_TIMER_Z1	Uses timer Z channel 1.
RAPI_TIMER_RC	Uses timer RC.
RAPI_TIMER_RD0	Uses timer RD0 channel 0.
RAPI_TIMER_RD1	Uses timer RD0 channel 1.
RAPI_TIMER_RD2	Uses timer RD0 channel 2.
RAPI_TIMER_RD3	Uses timer RD0 channel 3.
RAPI_TW_F1	Timer W counts with internal clock φ.
RAPI_TW_F2	Timer W counts with internal clock φ/2.
RAPI_TW_F4	Timer W counts with internal clock φ/4.
RAPI_TW_F8	Timer W counts with internal clock φ/8.
RAPI_TZ_F1	Timer Z counts with internal clock φ.
RAPI_TZ_F2	Timer Z counts with internal clock φ/2.
RAPI_TZ_F4	Timer Z counts with internal clock φ/4.

RAPI_TZ_F8	Timer Z counts with internal clock φ/8.
RAPI_TRC_F1	Timer RC counts with internal clock φ.
RAPI_TRC_F2	Timer RC counts with internal clock φ/2.
RAPI_TRC_F4	Timer RC counts with internal clockφ/4.
RAPI_TRC_F8	Timer RC counts with internal clock φ/8.
RAPI_TRC_F32	Timer RC counts with internal clock φ/8.
RAPI_TRC_F40M	Timer RC counts with internal clock φ40M.
RAPI_TRD_F1	Timer RD counts with internal clock φ.
RAPI_TRD_F2	Timer RD counts with internal clock φ/2.
RAPI_TRD_F4	Timer RD counts with internal clockφ/4.
RAPI_TRD_F8	Timer RD counts with internal clock φ/8.
RAPI_TRD_F32	Timer RD counts with internal clock φ/32.
RAPI_TRD_F40M	Timer RD counts with internal clock φ40M.
RAPI_TIMER_ON	Sets the timer to start operating in __CreateOutputCompare.
RAPI_TIMER_OFF	Sets the timer to stop operating in __CreateOutputCompare.
RAPI_OVERFLOW	Enables overflow interrupt.
RAPI_COMPARE_ MATCH_A	Enables GRA compare match interrupt.
RAPI_COMPARE_ MATCH_B	Enables GRB compare match interrupt.
RAPI_COMPARE_ MATCH_C	Enables GRC compare match interrupt.
RAPI_COMPARE_ MATCH_D	Enables GRD compare match interrupt.
RAPI_COUNT_CLEAR_A	Specifies GRA compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_B	Specifies GRB compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_C	Specifies GRC compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_D	Specifies GRD compare match for the cause of counter clear.
RAPI_COUNT_CLEAR_S YNC	Clear counter in sync with the synchronized other timer counter
RAPI_TIMER_SYNC	Synchronizes timer on channels 0 and 1.

• Specifiable definition values when timer W is used (RAPI_TIMER_W specified)

(Count source) Specify one from { RAPI_TW_F1, RAPI_TW_F2, RAPI_TW_F4, RAPI_TW_F8 }. The default value is RAPI_TW_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_ATCH_A, RAPI_COMPARE_ATCH_B, RAPI_COMPARE_ATCH_C, or RAPI_COMPARE_ATCH_D, respectively. If no interrupts are specified, “No interrupt request” is set.

(Counter clear) To specify GRA compare match for the cause of counter clear, specify RAPI_COUNT_CLEAR_A.

• Specifiable definition values when timer Z is used (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)

(Count source) Specify one from { RAPI_TZ_F1, RAPI_TZ_F2, RAPI_TZ_F4, RAPI_TZ_F8 }. The default value is RAPI_TZ_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_ATCH_A, RAPI_COMPARE_ATCH_B, RAPI_COMPARE_ATCH_C, or RAPI_COMPARE_ATCH_D, respectively. If no interrupts are specified, “No interrupt request” is set.

(Counter clear) To specify GRA, GRB, GRC, or GRD compare match for the cause of counter clear, specify RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively.

(Synchronization) If the timer is synchronized on channels 0 and 1, specify RAPI_TIMER_SYNC. If synchronization is not specified, “Channels 0 and 1 operate independently” is set.

• Specifiable definition values when timer RC is used (RAPI_TIMER_RC specified)

(Count source) Specify one from [RAPI_TRC_F1, RAPI_TRC_F2, RAPI_TRC_F4, RAPI_TRC_F8, RAPI_TRC_F32, RAPI_TRC_F40M]. The default is RAPI_TRC_F1.

(Operating states set) Specify one from [RAPI_TIMER_ON, RAPI_TIMER_OFF]. The default is RAPI_TIMER_OFF.

(Interrupt) If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. If compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A, RAPI_COMPARE_MATCH_B, RAPI_COMPARE_MATCH_C, or RAPI_COMPARE_MATCH_D, respectively. When not specifying interrupts, select “No interrupt requests.”

(Counter clear) To specify GRA compare match for the cause of counter clear, specify RAPI_COUNT_CLEAR_A.

• Specifiable definition values when timer RD is used (RAPI_TIMER_RD0 to RAPI_TIMER_RD4 specified)

(Count source) Specify one from [RAPI_TRD_F1, RAPI_TRD_F2, RAPI_TRD_F4, RAPI_TRD_F8, RAPI_TRD_F32, RAPI_TRD_F40M]. The default is RAPI_TRD_F1.

(Operating states set) Specify one from { RAPI_TIMER_ON, RAPI_TIMER_OFF }. The default value is RAPI_TIMER_OFF.

(Interrupt)	If overflow interrupt requests are enabled, specify RAPI_OVERFLOW. Similarly, if compare match A, compare match B, compare match C, or compare match D interrupt requests are enabled, specify RAPI_COMPARE_MATCH_A, RAPI_COMPARE_MATCH_B, RAPI_COMPARE_MATCH_C, or RAPI_COMPARE_MATCH_D, respectively. If no interrupts are specified, “No interrupt request” is set.
(Counter clear)	To specify GRA, GRB, GRC, or GRD compare match for the cause of counter clear, specify RAPI_COUNT_CLEAR_A, RAPI_COUNT_CLEAR_B, RAPI_COUNT_CLEAR_C, or RAPI_COUNT_CLEAR_D, respectively.
(Synchronization)	If the timer is synchronized on channels 0 and 1, specify RAPI_TIMER_SYNC. If synchronization is not specified, “Channels 0 and 1 operate independently” is set.

[data2]

(M16C)

Specify a pointer to the array in which the interrupt priority level is stored.

[0]: Specify the IC/OC base timer interrupt priority level (0-7).

[1]: Specify the IC/OC interrupt 0 priority level (0-7).

[2]: Specify the IC/OC interrupt 1 priority level (0-7).

(R8C)

• When using timer C or timer RF (RAPI_TIMER_C or RAPI_TIMER_RF)

Specify a pointer to the array in which the interrupt priority level is stored.

[0]: Specify the timer C interrupt priority level (0-7).

[1]: Specify the compare match 0 interrupt priority level (0-7).

[2]: Specify the compare match 1 interrupt priority level (0-7).

• When using timer RC, timer RD or timer RE (RAPI_TIMER_C, RAPI_TIMER_RD or RAPI_TIMER_RE)

Specify a pointer to the variable that contains the interrupt priority level (0–7) to be set in the interrupt control register.

(H8/300H)

Specify a pointer to the variable that contains the interrupt priority level (0-1) to be set in the interrupt control register.

For the CPUs that do not have an interrupt control register, specify 0.

[data3]

(M16C)

Specify a pointer to the array in which the set value for the waveform generation control register or waveform generation register is stored.

[0]: Specify the set value for waveform generation control register 0.

[1]: Specify the set value for waveform generation control register 1.

[2]: Specify the set value for waveform generation control register 2.

[3]: Specify the set value for waveform generation control register 3.

[4]: Specify the set value for waveform generation control register 4.

[5]: Specify the set value for waveform generation control register 5.

[6]: Specify the set value for waveform generation control register 6.
[7]: Specify the set value for waveform generation control register 7.
[8]: Specify the set value for waveform generation register 0 in 16 bits.
Specify the comparison value of channel 0 in 16 bits.
[9]: Specify the set value for waveform generation register 1 in 16 bits.
Specify the comparison value of channel 1 in 16 bits.
[10]: Specify the set value for waveform generation register 2 in 16 bits.
Specify the comparison value of channel 2 in 16 bits.
[11]: Specify the set value for waveform generation register 3 in 16 bits.
Specify the comparison value of channel 3 in 16 bits.
[12]: Specify the set value for waveform generation register 4 in 16 bits.
Specify the comparison value of channel 4 in 16 bits.
[13]: Specify the set value for waveform generation register 5 in 16 bits.
Specify the comparison value of channel 5 in 16 bits.
[14]: Specify the set value for waveform generation register 6 in 16 bits.
Specify the comparison value of channel 6 in 16 bits.
[15]: Specify the set value for waveform generation register 7 in 16 bits.
Specify the comparison value of channel 7 in 16 bits.

To specify the set value for each waveform generation control register that is an array element, the following definition values can be set. To specify multiple definition values at the same time, use the symbol "|" to separate each specified value.

RAPI_SINGLE	Selects single-phase waveform output mode.
RAPI_SR	Selects SR waveform output mode.
RAPI_PHASE_DELAYED	Selects inverted waveform output mode.
RAPI_OUT_INIT_0	Outputs a 0 as the initial output value.
RAPI_OUT_INIT_1	Outputs a 1 as the initial output value.
RAPI_OUTPUT_ REVERSED	Uses an output inversion facility.
RAPI_RELOAD_WITH_ RESET	Selects a reset of the base timer as the timing with which the G1P0J register value is reloaded.
RAPI_RELOAD_WITH_ WRITE	Selects a write to the base timer as the timing with which the G1P0J register value is reloaded.

• Specifiable definition values for waveform generation control registers 0-7

(Output mode) Specify one from { RAPI_SINGLE, RAPI_SR, RAPI_PHASE_DELAYED }. The default value is RAPI_SINGLE.

(Initial output value) Specify one from { RAPI_OUT_INIT_0, RAPI_OUT_INIT_1 }. The default value is RAPI_OUT_INIT_0.

(Output inversion) To invert the output, specify RAPI_OUTPUT_REVERSED. If output inversion is not specified, "Output not inverted" is set.

(Reload) Specify one from { RAPI_RELOAD_WITH_RESET, RAPI_RELOAD_WITH_WRITE }. The default value is RAPI_RELOAD_WITH_WRITE.

(R8C)

Specify a pointer to the array in which the comparison value is stored.

• When timer C is used (RAPI_TIMER_C specified) 、When timer RF is used (RAPI_TIMER_RF specified)

[0]: Specify comparison value 0 in 16 bits.

[1]: Specify comparison value 1 in 16 bits.

• When timer RC is used (RAPI_TIMER_RC specified)

[0]: Specify the output operation of TRCIOA.

[1]: Specify the output operation of TRCIOB.

[2]: Specify the output operation of TRCIOC.

[3]: Specify the output operation of TRCIOD.

[4]: Specify the comparison value of general register A in 16 bits.

[5]: Specify the comparison value of general register B in 16 bits.

[6]: Specify the comparison value of general register C in 16 bits.

[7]: Specify the comparison value of general register D in 16 bits.

• When timer RD is used (RAPI_TIMER_RD0 to RAPI_TIMER_RD4 specified)

Specify a pointer to the array in which the comparison value is stored.

[0]: Specify the output operation of TRDIOAi (i=0,1).

[1]: Specify the output operation of TRDIOBi (i=0,1).

[2]: Specify the output operation of TRDIOCi (i=0,1).

[3]: Specify the output operation of TRDIODi (i=0,1).

[4]: Specify the comparison value of general register A in 16 bits.

[5]: Specify the comparison value of general register B in 16 bits.

[6]: Specify the comparison value of general register C in 16 bits.

[7]: Specify the comparison value of general register D in 16 bits.

• When timer RE is used (RAPI_TIMER_RE specified)

[0]: Specify the comparison value of the register for storing the Timer RE compare data in 8 bits.

To specify output operation for each output compare of array elements, the following definition values can be used. To specify multiple definition values at the same time, use the symbol "|" to separate each specified value. For elements of output operation corresponding to unused channels, set 0.

RAPI_OUT_0	Selects 0 output for the output waveform.
RAPI_OUT_1	Selects 1 output for the output waveform.
RAPI_OUT_TOGGLE	Selects toggle output for the output waveform.
RAPI_OUT_INIT_0	Selects 0 output for the initial output.
RAPI_OUT_INIT_1	Selects 1 output for the initial output.

(H8S)

Specify a pointer to the array in which the comparison value is stored.

• When timer RC is used (RAPI_TIMER_C specified) 、When timer RD is used (RAPI_TIMER_RD specified)

[0]: Specify the output operation of FTIOA pin.

[1]: Specify the output operation of FTIOB pin.

[2]: Specify the output operation of FTIOC pin.

[3]: Specify the output operation of FTIOD pin.

[4]: Specify the comparison value of general register A in 16 bits.

[5]: Specify the comparison value of general register B in 16 bits.

[6]: Specify the comparison value of general register C in 16 bits.

[7]: Specify the comparison value of general register D in 16 bits.

To specify output operation for each output compare of array elements, the following definition values can be used. To specify multiple definition values at the same time, use the symbol "|" to separate each specified value. For elements of output operation corresponding to unused channels, set 0.

RAPI_OUT_0	Selects 0 output for the output waveform.
RAPI_OUT_1	Selects 1 output for the output waveform.
RAPI_OUT_TOGGLE	Selects toggle output for the output waveform.
RAPI_OUT_INIT_0	Selects 0 output for the initial output.
RAPI_OUT_INIT_1	Selects 1 output for the initial output.
RAPI_AD_TRIGGER_ON	A/D conversion start trigger is generated by compare match.

• When timer RE is used (RAPI_TIMER_RE specified)

[0]: Specify the comparison value of the register for storing the Timer RE compare data in 8 bits.

• When timer RG is used (RAPI_TIMER_RG specified)

[0]: Specify the output operation of FTIOA pin.

[1]: Specify the output operation of FTIOB pin.

[2]: Specify the comparison value of general register A in 16 bits.

[3]: Specify the comparison value of general register B in 16 bits.

To specify output operation for each output compare of array elements, the following definition values can be used. To specify multiple definition values at the same time, use the symbol "|" to separate each specified value. For elements of output operation corresponding to unused channels, set 0.

RAPI_OUT_0	Selects 0 output for the output waveform.
RAPI_OUT_1	Selects 1 output for the output waveform.
RAPI_OUT_TOGGLE	Selects toggle output for the output waveform.

(H8/300H)

Specify a pointer to the array in which the comparison value is stored.

[0]: Specify the output operation of FTIOA pin.

[1]: Specify the output operation of FTIOB pin.

[2]: Specify the output operation of FTIOC pin.

[3]: Specify the output operation of FTIOD pin.

[4]: Specify the comparison value of general register A in 16 bits.

[5]: Specify the comparison value of general register B in 16 bits.

[6]: Specify the comparison value of general register C in 16 bits.

[7]: Specify the comparison value of general register D in 16 bits.

To specify output operation for each output compare of array elements, the following definition values can be used. To specify multiple definition values at the same time, use the symbol "|" to separate each specified value. For elements of output operation corresponding to unused channels, set 0.

RAPI_OUT_0	Selects 0 output for the output waveform.

RAPI_OUT_1	Selects 1 output for the output waveform.
RAPI_OUT_TOGGLE	Selects toggle output for the output waveform.
RAPI_OUT_INIT_0	Selects 0 output for the initial output.
RAPI_OUT_INIT_1	Selects 1 output for the initial output.

[data4]

(M16C)

Specify a pointer to the array in which the set value for each register of timer S is stored.

[0]: Specify the set value for the facility select and facility enable register.
Specify the channel for which the waveform generation facility is enabled.

[1]: Specify the set value for interrupt enable register 0.
Specify the channel for which IO/CO interrupt 0 request is enabled.

[2]: Specify the set value for interrupt enable register 1.
Specify the channel for which IO/CO interrupt 1 request is enabled.

[3]: Specify the set value for the count source divide-by-n register. Specify the value of 'n' in the formula "count source divided by (n + 1)" in 8 bits.

For the channels to be specified in each array element, use the following definition values. To specify multiple definition values at the same time, use the symbol "|" to separate each specified value. If 0 is specified, the value 0 is set in the corresponding register.

RAPI_CHANNEL0	Selects channel 0.
RAPI_CHANNEL1	Selects channel 1.
RAPI_CHANNEL2	Selects channel 2.
RAPI_CHANNEL3	Selects channel 3.
RAPI_CHANNEL4	Selects channel 4.
RAPI_CHANNEL5	Selects channel 5.
RAPI_CHANNEL6	Selects channel 6.
RAPI_CHANNEL7	Selects channel 7.

(R8C) (H8S)(H8/300H)

Specify 0.

[data5]

(M16C)

Specify a pointer to the array in which the callback function is stored.

[0]: Specify a pointer to the callback function for IC/OC base timer interrupt.
If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for IC/OC interrupt 0.
If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for IC/OC interrupt 1.
If this pointer is not specified, 0 is set.

(R8C)

• When timer C is used (RAPI_TIMER_C specified) 、When timer RF is used (RAPI_TIMER_RF specified)

Specify a pointer to the array in which the callback function is stored.

[0]: Specify a pointer to the callback function for timer C interrupt.
If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for compare match interrupt 0.
If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for compare match interrupt 1.
If this pointer is not specified, 0 is set.

• When timer RC, timer RD, or timer RE is used (RAPI_TIMER_RC, RAPI_TIMER_RD0 to RAPI_TIMER_RD1, or RAPI_TIMER_RE specified)

Specify a pointer to the variable in which the callback function is stored.
If this pointer is not specified, RAPI_NULL is set.

• When timer RD is used (RAPI_TIMER_RD0 to RAPI_TIMER_RD4 specified)

Specify a pointer to the array in which the callback function is stored.
If this pointer is not specified, RAPI_NULL is set.

• When timer RE is used (RAPI_TIMER_RE specified)

Specify a pointer to the array in which the callback function is stored.
If this pointer is not specified, RAPI_NULL is set.

(H8S)

• When timer RC is used (RAPI_TIMER_RC specified) 、When timer RD is used (RAPI_TIMER_RD specified)

[0]: Specify a pointer to the callback function for overflow interrupt.
If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for input compare match A interrupt.
If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for input compare match B interrupt.
If this pointer is not specified, 0 is set.

[3]: Specify a pointer to the callback function for input compare match C interrupt.
If this pointer is not specified, 0 is set.

[4]: Specify a pointer to the callback function for input compare match D interrupt.
If this pointer is not specified, 0 is set.

• When timer RE is used (RAPI_TIMER_RE specified)

Specify a pointer to the array in which the callback function is stored.
If this pointer is not specified, RAPI_NULL is set.

• When timer RG is used (RAPI_TIMER_RG specified)

[0]: Specify a pointer to the callback function for overflow interrupt.
If this pointer is not specified, 0 is set.

[1]: Specify a pointer to the callback function for input compare match A interrupt.
If this pointer is not specified, 0 is set.

[2]: Specify a pointer to the callback function for input compare match B interrupt.
If this pointer is not specified, 0 is set.

(H8/300H)

Specify a pointer to the variable in which the callback function is stored. If this pointer is not specified, RAPI_NULL is set.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (output compare mode)

Reference

__EnableOutputCompare, __DestroyOutputCompare

Remark

- If an undefined values are specified in the arguments, operation of the API cannot be guaranteed.
- When using H8S or H8/300H, this API cancels the module standby state of timer before setting up.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void TimerIntFunc0( void ){}
void TimerIntFunc1( void ){}
void TimerIntFunc2( void ){}
void func( void )
{
    unsigned int *p_func[] =  {(void*) TimerIntFunc0,(void*) TimerIntFunc1,
                               (void*) TimerIntFunc2};
    unsigned char p_ic[] =  {1,2,3};
    unsigned int p_cmp[] =  {0x1234, 0x9876};

    /* Set up timer C as output compare mode */
    __CreateOutputCompare( RAPI_TIMER_C|RAPI_TIMER_ON|RAPI_RELOAD|
        RAPI_L_1|RAPI_L_0|RAPI_F32|
        RAPI_CMP02_ENABLE|RAPI_CMP12_ENABLE, p_cmp, p_ic, p_func );
}
```

__EnableOutputCompare

Synopsis

<Control operation of output compare mode>

Boolean __EnableOutputCompare(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Controls operation of the timer that is set to specified output compare mode by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_S	Selects timer S.
RAPI_TIMER_ON	Sets the timer that is set to output compare mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to output compare mode to stop operating.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RF	Selects timer RF.
RAPI_TIMER_ON	Sets the timer that is set to output compare mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to output compare mode to stop operating.

(H8S)

RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RG	Selects timer RG.
RAPI_TIMER_ON	Sets the timer that is set to output compare mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to output compare mode to stop operating.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RD0.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.

RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.
RAPI_TIMER_ON	Sets the timer that is set to output compare mode to start operating.
RAPI_TIMER_OFF	Sets the timer that is set to output compare mode to stop operating.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (output compare mode)

Reference

__CreateOutputCompare, __DestroyOutputCompare

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Enable timer C as output compare mode */
    __EnableOutputCompare( RAPI_TIMER_C|RAPI_TIMER_ON );
}
```

__DestroyOutputCompare

Synopsis

<Discard settings of output compare mode>

Boolean __DestroyOutputCompare(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Discards settings of the timer or Intelligent I/O that is set to specified output compare mode.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_S	Selects timer S.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RF	Selects timer RF.

(H8S)

RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RG	Selects timer RG.

(H8/300H)

RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RD0.
RAPI_TIMER_RD0	Selects timer RD0 channe 0.
RAPI_TIMER_RD1	Selects timer RD0 channe 1.
RAPI_TIMER_RD2	Selects timer RD0 channe 2.
RAPI_TIMER_RD3	Selects timer RD0 channe 3.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (output compare mode)

Reference

__CreateOutputCompare, __EnableOutputCompare

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- When used for the H8S, H8/300H, this API places a specified timer into module stanby state after discarding it.

Program example

```
#include "rapi_timer_r8c_13.h"

void func( void )
{
    /* Destroy the setting of timer C as output compare mode */
    __DestroyOutputCompare( RAPI_TIMER_C );
}
```

__SetTimerRegister

Synopsis

<Set timer register>

Boolean __SetTimerRegister(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which register value is stored

Description

Sets the registers of a specified timer or Intelligent I/O.

[data1]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_S	Selects timer S.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RF	Selects timer RF.

(H8S)

RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RG	Selects timer RG.

(H8/300H)

RAPI_TIMER_A	Selects timer A.
RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.

[data2]

The content of a pointer to the buffer in which the register value is stored must be specified as described below. The value is set in each register in order of buffer pointer elements.

(M16C)

- **When using timer A (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)**

[0]: Specify the set value for the timer Ai mode register (i = 0–4).
[1]: Specify the set value for the timer Ai register (i = 0–4).
[2]: Specify the set value for the up/down flag register.
[3]: Specify the set value for the one-shot start flag register.
[4]: Specify the set value for the trigger select register.
[5]: Specify the set value for the time-clock prescaler reset register.
[6]: Specify the set value for the count start flag register.

- **When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2)**

[0]: Specify the set value for the timer Bi mode register (i = 0–2).
[1]: Specify the set value for the timer Bi register (i = 0–2).
[3]: Specify the set value for the time-clock prescaler reset register.
[4]: Specify the set value for the count start flag register.

- **When using timer S (RAPI_TIMER_S specified)**

[0]: Specify the set value for the base timer register.
[1]: Specify the set value for the base timer reset register.
[2]: Specify the set value for base timer control register 0.
[3]: Specify the set value for base timer control register 1.
[4]: Specify the set value for the count source divide-by-n register.
[5]: Specify the set value for time measurement control register 0.
[6]: Specify the set value for time measurement control register 1.
[7]: Specify the set value for time measurement control register 2.
[8]: Specify the set value for time measurement control register 3.
[9]: Specify the set value for time measurement control register 4.
[10]: Specify the set value for time measurement control register 5.
[11]: Specify the set value for time measurement control register 6.
[12]: Specify the set value for time measurement control register 7.

[13]: Specify the set value for time measurement prescaler register 6.
[14]: Specify the set value for time measurement prescaler register 7.
[15]: Specify the set value for waveform generation control register 0.
[16]: Specify the set value for waveform generation control register 1.
[17]: Specify the set value for waveform generation control register 2.
[18]: Specify the set value for waveform generation control register 3.
[19]: Specify the set value for waveform generation control register 4.
[20]: Specify the set value for waveform generation control register 5.
[21]: Specify the set value for waveform generation control register 6.
[22]: Specify the set value for waveform generation control register 7.
[23]: Specify the set value for waveform generation register 0.
[24]: Specify the set value for waveform generation register 1.
[25]: Specify the set value for waveform generation register 2.
[26]: Specify the set value for waveform generation register 3.
[27]: Specify the set value for waveform generation register 4.
[28]: Specify the set value for waveform generation register 5.
[29]: Specify the set value for waveform generation register 6.
[30]: Specify the set value for waveform generation register 7.
[31]: Specify the set value for the facility select register.
[32]: Specify the set value for the facility enable register.
[33]: Specify the set value for the interrupt request register.
[34]: Specify the set value for interrupt enable register 0.
[35]: Specify the set value for interrupt enable register 1.

(R8C)

• When using timer C (RAPI_TIMER_C specified)

[0]: Specify the set value for the timer C output control register.
[1]: Specify the set value for timer C control register 0.
[2]: Specify the set value for timer C control register 1.
[3]: Specify the set value for the capture & compare 0 register.
[4]: Specify the set value for the compare 1 register.

• When using timer X (RAPI_TIMER_X specified)

[0]: Specify the set value for the timer count source setup register.
[1]: Specify the set value for the prescaler X register.
[2]: Specify the set value for the timer X register.
[3]: Specify the set value for the timer X mode register.

• When using timer Y (RAPI_TIMER_Y specified)

[0]: Specify the set value for the timer count source setup register.
[1]: Specify the set value for the prescaler Y register.
[2]: Specify the set value for the timer Y primary register.
[3]: Specify the set value for the timer Y secondary register.
[4]: Specify the set value for the timer Y & Z waveform output control register.
[5]: Specify the set value for the timer Y & Z output control register.
[6]: Specify the set value for the timer Y & Z mode register.

• When using timer Z (RAPI_TIMER_Z specified)

[0]: Specify the set value for the timer count source setup register.

[1]: Specify the set value for the prescaler Z register.
[2]: Specify the set value for the timer Z primary register.
[3]: Specify the set value for the timer Z secondary register.
[4]: Specify the set value for the timer Y & Z waveform output control register.
[5]: Specify the set value for the timer Y & Z output control register.
[6]: Specify the set value for the timer Y & Z mode register.

• When using timer RA (RAPI_TIMER_RA specified)

[0]: Specify the set value for the timer RA I/O control register.
[1]: Specify the set value for the timer RA prescaler register.
[2]: Specify the set value for the timer RA register.
[3]: Specify the set value for the timer RA mode register.
[4]: Specify the set value for the timer RA control register.

• When using timer RB (RAPI_TIMER_RB specified)

[0]: Specify the set value for the timer RB one shot control register.
[1]: Specify the set value for the timer RB I/O control register.
[2]: Specify the set value for the timer RB prescaler register.
[3]: Specify the set value for the timer RB primary register.
[4]: Specify the set value for the timer RB secondary register.
[5]: Specify the set value for the timer RB mode register.
[6]: Specify the set value for the timer RB control register.

• When using timer RC (RAPI_TIMER_RC specified)

[0]: Specify the set value for the timer RC general register A.
[1]: Specify the set value for the timer RC general register B.
[2]: Specify the set value for the timer RC general register C.
[3]: Specify the set value for the timer RC general register D.
[4]: Specify the set value for the timer RC digital filter function select register.
[5]: Specify the set value for the timer RC control register 1.
[6]: Specify the set value for the timer RC control register 2.
[7]: Specify the set value for the timer RC I/O control register 0.
[8]: Specify the set value for the timer RC I/O control register 1.
[9]: Specify the set value for the timer RC status register.
[10]: Specify the set value for the timer RC interrupt enable register.
[11]: Specify the set value for the timer RC counter.
[12]: Specify the set value for the timer RC output master enable register.
[13]: Specify the set value for the timer RC mode register.

• When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD1)

[0]: Specify the set value for the timer RD mode register.
[1]: Specify the set value for the timer RD PWM mode register.
[2]: Specify the set value for the timer RD function control register.
[3]: Specify the set value for the timer RD general register Ai (i=0,1).
[4]: Specify the set value for the timer RD general register Bi (i=0,1).
[5]: Specify the set value for the timer RD general register Ci (i=0,1).
[6]: Specify the set value for the timer RD general register Di (i=0,1).
[7]: Specify the set value for the timer RD digital filter function select register i(i=0,1).

[8]: Specify the set value for the timer RD control register i(i=0,1).

[9]: Specify the set value for the timer RD I/O control register Ai(i=0,1).

[10]: Specify the set value for the timer RD I/O control register Ci(i=0,1).

[11]: Specify the set value for the timer RD status register i(i=0,1).

[12]: Specify the set value for the timer RD interrupt enable register i(i=0,1).

[13]: Specify the set value for the timer RD counter i(i=0,1).

[14]: Specify the set value for the timer RD start register.

[15]: Specify the set value for the timer RD output master enable register 1.

[16]: Specify the set value for the timer RD output master enable register 2.

[17]: Specify the set value for the timer RD output control register.

• When using timer RE (RAPI_TIMER_RE)

[0]: Specify the set value for the timer RE second data register.

[1]: Specify the set value for the timer RE minute data register.

[2]: Specify the set value for the timer RE hour data register.

[3]: Specify the set value for the timer RE day of week data register.

[4]: Specify the set value for the timer RE control register 2.

[5]: Specify the set value for the timer RE count source select register.

[6]: Specify the set value for the timer RE control register 1.

• When using timer RF (RAPI_TIMER_RF)

[0]: Specify the set value for the timer RF control register 0.

[1]: Specify the set value for capture & compare 0 register.

[2]: Specify the set value for compare 1 register.

[3]: Specify the set value for the timer RF output control register 0.

[4]: Specify the set value for the timer RF control register 1.

(H8S)

• When using timer RA (RAPI_TIMER_RA)

[0]: Specify the set value for the timer RA I/O control register.

[1]: Specify the set value for the timer RA prescaler register.

[2]: Specify the set value for the timer RA timer register.

[3]: Specify the set value for the timer RA mode register.

[4]: Specify the set value for the timer RA interrupt request status register.

[5]: Specify the set value for the timer RA control register.

• When using timer RB (RAPI_TIMER_RB)

[0]: Specify the set value for the timer RB one-shot control register.

[1]: Specify the set value for the timer RB I/O control register.

[2]: Specify the set value for the timer RB prescaler register.

[3]: Specify the set value for the timer RB primary register.

[4]: Specify the set value for the timer RB secondary register.

[5]: Specify the set value for the timer RB mode register.

[6]: Specify the set value for the timer RB interrupt request status register.

[7]: Specify the set value for the timer RB control register.

• When using timer RC (RAPI_TIMER_RC)

[0]: Specify the set value for timer RC control register 1.

[1]: Specify the set value for timer RC control register 2.

[2]: Specify the set value for timer RC interrupt enable register.

[3]: Specify the set value for timer RC status register.
[4]: Specify the set value for timer RC I/O control register 0.
[5]: Specify the set value for timer RC I/O control register 1.
[6]: Specify the set value for timer RC output enable register.
[7]: Specify the set value for timer RC digital filter function select register.
[8]: Specify the set value for timer RC A/D conversion start trigger control register.
[9]: Specify the set value for timer RC counter.
[10]: Specify the set value for general register A.
[11]: Specify the set value for general register B.
[12]: Specify the set value for general register C.
[13]: Specify the set value for general register D.
[14]: Specify the set value for timer RC mode register.

• When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD3 specified)

[0]: Specify the set value for the timer RD mode register.
[1]: Specify the set value for the timer RD PWM mode register.
[2]: Specify the set value for the timer RD function control register.
[3]: Specify the set value for the timer RD output master enable register 1.
[4]: Specify the set value for the timer RD output master enable register 2.
[5]: Specify the set value for the timer RD output control register.
[6]: Specify the set value for timer RD A/D conversion start trigger control register.
[7]: Specify the set value for the timer RD counter i(i=0,1).
[8]: Specify the set value for general register A_i(i=0,1).
[9]: Specify the set value for general register B_i(i=0,1).
[10]: Specify the set value for general register C_i(i=0,1).
[11]: Specify the set value for general register D_i(i=0,1).
[12]: Specify the set value for timer RD control register_i (i = 0, 1).
[13]: Specify the set value for timer RD I/O control register A_i (i = 0, 1).
[14]: Specify the set value for timer RD I/O control register C_i (i = 0, 1).
[15]: Specify the set value for timer RD status register_i (i = 0, 1).
[16]: Specify the set value for timer RD interrupt enable register_i (i = 0, 1).
[17]: Specify the set value for PWM mode output level control register_i (i = 0, 1).
[18]: Specify the set value for timer RD digital filter function select register i(i=0, 1).
[19]: Specify the set value for the timer RD start register.

• When using timer RE (RAPI_TIMER_RE)

[0]: Specify the set value for timer RE second data register/counter data register.
[1]: Specify the set value for timer RE minute data register/compare data register.
[2]: Specify the set value for timer RE hour data register.
[3]: Specify the set value for timer RE day-of-week data register.
[4]: Specify the set value for timer REcontrol register 2.
[5]: Specify the set value for timer RE clock source select register.
[6]: Specify the set value for timer RE interrupt flag register.
[7]: Specify the set value for timer REcontrol register 1.

• When using timer RG (RAPI_TIMER_RG)

[0]: Specify the set value for timer RG counter control register.
[1]: Specify the set value for timer RG control register.

[2]: Specify the set value for timer RG I/O control register.

[3]: Specify the set value for timer RG status register.

[4]: Specify the set value for timer RG interrupt enable register.

[5]: Specify the set value for timer RG counter.

[6]: Specify the set value for General register A.

[7]: Specify the set value for General register B.

[8]: Specify the set value for GRA buffer register.

[9]: Specify the set value for GRA buffer register.

[10]: Specify the set value for timer RG mode register.

(H8/300H)

• When using timer A (RAPI_TIMER_A specified)

[0]: Specify the set value for timer mode register A.

[1]: Specify the set value for timer counter A.

• When using timer B1 (RAPI_TIMER_B1 specified)

[0]: Specify the set value for timer mode register B1.

[1]: Specify the set value for timer load register B1.

• When using timer V (RAPI_TIMER_V specified)

[0]: Specify the set value for timer counter V.

[1]: Specify the set value for time constant register A.

[2]: Specify the set value for time constant register B.

[3]: Specify the set value for timer control register V0.

[4]: Specify the set value for timer control register V1.

[5]: Specify the set value for timer control/status register V.

• When using timer W (RAPI_TIMER_W specified)

[0]: Specify the set value for timer control register W.

[1]: Specify the set value for timer interrupt enable register W.

[2]: Specify the set value for timer status register W.

[3]: Specify the set value for timer I/O control register 0.

[4]: Specify the set value for timer I/O control register 1.

[5]: Specify the set value for the timer counter.

[6]: Specify the set value for general register A.

[7]: Specify the set value for general register B.

[8]: Specify the set value for general register C.

[9]: Specify the set value for general register D.

[10]: Specify the set value for the timer mode register W.

• When using timer Z (RAPI_TIMER_Z0 to RAPI_TIMER_Z1 specified)

[0]: Specify the set value for the timer mode register.

[1]: Specify the set value for the timer PWM mode register.

[2]: Specify the set value for the timer function control register.

[3]: Specify the set value for the timer output master enable register.

[4]: Specify the set value for the timer output control register.

[5]: Specify the set value for the timer counter.

[6]: Specify the set value for general register A i(i=0,1).

[7]: Specify the set value for general register B i(i=0,1).

[8]: Specify the set value for general register C i(i=0,1).

[9]: Specify the set value for general register D i(i=0,1).
[10]: Specify the set value for timer control register_i (i = 0, 1).
[11]: Specify the set value for timer I/O control register A_i (i = 0, 1).
[12]: Specify the set value for timer I/O control register C_i (i = 0, 1).
[13]: Specify the set value for timer status register_i (i = 0, 1).
[14]: Specify the set value for timer interrupt enable register_i (i = 0, 1).
[15]: Specify the set value for PWM mode output level control register_i (i = 0, 1).
[16]: Specify the set value for the timer start register.

• When using timer RC (RAPI_TIMER_RC)

[0]: Specify the set value for timer RC control register 1.
[1]: Specify the set value for timer RC control register 2.
[2]: Specify the set value for timer RC interrupt enable register.
[3]: Specify the set value for timer RC status register.
[4]: Specify the set value for timer RC I/O control register 0.
[5]: Specify the set value for timer RC I/O control register 1.
[6]: Specify the set value for timer RC output enable register.
[7]: Specify the set value for timer RC digital filter function select register.
[8]: Specify the set value for timer RC counter.
[9]: Specify the set value for general register A.
[10]: Specify the set value for general register B.
[11]: Specify the set value for general register C.
[12]: Specify the set value for general register D.
[13]: Specify the set value for timer RC mode register.

• When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD3 specified)

[0]: Specify the set value for the timer RD mode register.
[1]: Specify the set value for the timer RD PWM mode register.
[2]: Specify the set value for the timer RD function control register.
[3]: Specify the set value for the timer RD output master enable register 1.
[4]: Specify the set value for the timer RD output master enable register 2.
[5]: Specify the set value for the timer RD output control register.
[6]: Specify the set value for the timer RD counter i(i=0,1).
[7]: Specify the set value for general register A_i(i=0,1).
[8]: Specify the set value for general register B_i(i=0,1).
[9]: Specify the set value for general register C_i(i=0,1).
[10]: Specify the set value for general register D_i(i=0,1).
[11]: Specify the set value for timer RD control register_i (i = 0, 1).
[12]: Specify the set value for timer RD I/O control register A_i (i = 0, 1).
[13]: Specify the set value for timer RD I/O control register C_i (i = 0, 1).
[14]: Specify the set value for timer RD status register_i (i = 0, 1).
[15]: Specify the set value for timer RD interrupt enable register_i (i = 0, 1).
[16]: Specify the set value for PWM mode output level control register_i (i = 0, 1).
[17]: Specify the set value for the timer RD digital filter function select register i(i = 0, 1).
[18]: Specify the set value for the timer RD start register.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (register manipulation)

Reference

__EnableTimerRegister, __ClearTimerRegister, __GetTimerRegister

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include " rapi_timer_r8c_13.h"

void func( void )
{
    unsigned char data[] =  {0,0,0,0,0,0,0};

    /* Set up timer Z register */
    __SetTimerRegister( RAPI_TIMER_Z, data );
}
```

__EnableTimerRegister

Synopsis

<Control operation of timer register>

Boolean __EnableTimerRegister(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Controls operation of a specified timer or Intelligent I/O by starting or stopping it.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_S	Selects timer S.
RAPI_TIMER_ON	Sets the selected timer to start operating.
RAPI_TIMER_OFF	Sets the selected timer to stop operating.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RF	Selects timer RF.
RAPI_TIMER_ON	Sets the selected timer to start operating.
RAPI_TIMER_OFF	Sets the selected timer to stop operating.

(H8S)

RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.

RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RG	Selects timer RG.
RAPI_TIMER_ON	Sets the selected timer to start operating.
RAPI_TIMER_OFF	Sets the selected timer to stop operating.

(H8/300H)

RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.
RAPI_TIMER_ON	Sets the selected timer to start operating.
RAPI_TIMER_OFF	Sets the selected timer to stop operating.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (register manipulation)

Reference

__SetTimerRegister, __ClearTimerRegister, __GetTimerRegister

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.
- To specify commencement of timer operation when using timer V of the H8/300H, set the clock and count condition to be supplied to TCNTV that are specified in __SetTimerRegister immediately preceding this API.

Program example

```
#include " rapi_timer_r8c_13.h"

void func( void )
{
    /* Activate timer C */
    __EnableTimerRegister( RAPI_TIMER_C|RAPI_TIMER_ON );
}
```

__ClearTimerRegister

Synopsis

<Clear timer register>

Boolean __ClearTimerRegister(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Sets the timer register of a specified timer or Intelligent I/O to its initial value after reset.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_S	Selects timer S.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RF	Selects timer RF.

(H8S)

RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RG	Selects timer RG.

(H8/300H)

RAPI_TIMER_A	Selects timer A.
RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (register manipulation)

Reference

__SetTimerRegister, __EnableTimerRegister, __GetTimerRegister

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include " rapi_timer_r8c_13.h"

void func( void )
{
    /* Clear the setting of timer C */
    __ClearTimerRegister( RAPI_TIMER_C );
}
```

__GetTimerRegister

Synopsis

<Get timer register value>

Boolean __GetTimerRegister(unsigned long data1, unsigned int *data2)

data1	Setup data (content differs with MCU type)
data2	Pointer to the buffer in which register value is stored

Description

Gets the counter value of a specified timer or Intelligent I/O.

[data]

For data, the following definition values can be set.

(M16C)

RAPI_TIMER_A0	Selects timer A channel 0.
RAPI_TIMER_A1	Selects timer A channel 1.
RAPI_TIMER_A2	Selects timer A channel 2.
RAPI_TIMER_A3	Selects timer A channel 3.
RAPI_TIMER_A4	Selects timer A channel 4.
RAPI_TIMER_B0	Selects timer B channel 0.
RAPI_TIMER_B1	Selects timer B channel 1.
RAPI_TIMER_B2	Selects timer B channel 2.
RAPI_TIMER_S	Selects timer S.

(R8C)

RAPI_TIMER_C	Selects timer C.
RAPI_TIMER_X	Selects timer X.
RAPI_TIMER_Y	Selects timer Y.
RAPI_TIMER_Z	Selects timer Z.
RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RF	Selects timer RF.

(H8S)

RAPI_TIMER_RA	Selects timer RA.
RAPI_TIMER_RB	Selects timer RB.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.
RAPI_TIMER_RE	Selects timer RE.
RAPI_TIMER_RG	Selects timer RG.

(H8/300H)

RAPI_TIMER_A	Selects timer A.
RAPI_TIMER_B1	Selects timer B1.
RAPI_TIMER_V	Selects timer V.
RAPI_TIMER_W	Selects timer W.
RAPI_TIMER_Z0	Selects timer Z channel 0.
RAPI_TIMER_Z1	Selects timer Z channel 1.
RAPI_TIMER_RC	Selects timer RC.
RAPI_TIMER_RD0	Selects timer RD channel 0.
RAPI_TIMER_RD1	Selects timer RD channel 1.
RAPI_TIMER_RD2	Selects timer RD channel 2.
RAPI_TIMER_RD3	Selects timer RD channel 3.

[data2]

Specify a pointer to the array in which the acquired register value is stored.

The content of the array is described below.

(M16C)

• When using timer A (RAPI_TIMER_A0 to RAPI_TIMER_A4 specified)

[0]: Store the value of timer Ai mode register (i = 0–4).

[1]: Store the value of timer Ai register (i = 0–4).

[2]: Store the value of the up/down flag register.

[3]: Store the value of the one-shot start flag register.

[4]: Store the value of the trigger select register.

[5]: Store the value of the time-clock prescaler reset flag register.

[6]: Store the value of the count start flag register.

• When using timer B (RAPI_TIMER_B0 to RAPI_TIMER_B2 specified)

[0]: Store the value of timer Bi mode register (i = 0–2).

[1]: Store the value of timer Bi register (i = 0–2).

[2]: Store the value of the time-clock prescaler reset flag register.

[3]: Store the value of the count start flag register.

• When using timer S (RAPI_TIMER_S specified)

[0]: Store the value of the base timer register.

[1]: Store the value of the base timer reset register.

[2]: Store the value of base timer control register 0.

[3]: Store the value of base timer control register 1.

[4]: Store the value of the count source divide-by-n register.

[5]: Store the value of time measurement control register 0.

[6]: Store the value of time measurement control register 1.

[7]: Store the value of time measurement control register 2.

[8]: Store the value of time measurement control register 3.

[9]: Store the value of time measurement control register 4.

[10]: Store the value of time measurement control register 5.

[11]: Store the value of time measurement control register 6.

[12]: Store the value of time measurement control register 7.

[13]: Store the value of time measurement prescaler register 6.

[14]: Store the value of time measurement prescaler register 7.

[15]: Store the value of waveform generation control register 0.

[16]: Store the value of waveform generation control register 1.

[17]: Store the value of waveform generation control register 2.

[18]: Store the value of waveform generation control register 3.

[19]: Store the value of waveform generation control register 4.

[20]: Store the value of waveform generation control register 5.

[21]: Store the value of waveform generation control register 6.

[22]: Store the value of waveform generation control register 7.

[23]: Store the value of time measurement register 0/waveform generation register 0.

[24]: Store the value of time measurement register 1/waveform generation register 1.

[25]: Store the value of time measurement register 2/waveform generation register 2.

[26]: Store the value of time measurement register 3/waveform generation register 3.

[27]: Store the value of time measurement register 4/waveform generation register 4.

[28]: Store the value of time measurement register 5/waveform generation register 5.

[29]: Store the value of time measurement register 6/waveform generation register 6.

[30]: Store the value of time measurement register 7/waveform generation register 7.

[31]: Store the value of the facility select register.

[32]: Store the value of the facility enable register.

[33]: Store the value of the interrupt request register.

[34]: Store the value of interrupt enable register 0.

[35]: Store the value of interrupt enable register 1.

(R8C)

• When using timer C (RAPI_TIMER_C specified)

[0]: Store the value of the timer C register.

[1]: Store the value of the capture & compare 0 register.

[2]: Store the value of the compare 1 register.

[3]: Store the value of the timer C output control register.

[4]: Store the value of timer C control register 1.

[5]: Store the value of timer C control register 0.

• When using timer X (RAPI_TIMER_X specified)

[0]: Store the value of the timer count source setup register.

[1]: Store the value of the prescaler X register.

[2]: Store the value of the timer X register.

[3]: Store the value of the timer X mode register.

• When using timer Y (RAPI_TIMER_Y specified)

[0]: Store the value of the timer count source setup register.

[1]: Store the value of the prescaler Y register.

[2]: Store the value of the timer Y primary register.

[3]: Store the value of the timer Y secondary register.

[4]: Store the value of the timer Y & Z waveform output control register.

[5]: Store the value of the timer Y & Z output control register.

[6]: Store the value of the timer Y & Z mode register.

• When using timer Z (RAPI_TIMER_Z specified)

[0]: Store the value of the timer count source setup register.

[1]: Store the value of the prescaler Z register.

[2]: Store the value of the timer Z primary register.

[3]: Store the value of the timer Z secondary register.

[4]: Store the value of the timer Y & Z waveform output control register.

[5]: Store the value of the timer Y & Z output control register.

[6]: Store the value of the timer Y & Z mode register.

• When using timer RA (RAPI_TIMER_RA specified)

[0]: Store the value of the timer RA I/O control register.

[1]: Store the value of the timer RA prescaler register.

[2]: Store the value of the timer RA register.

[3]: Store the value of the timer RA mode register.

[4]: Store the value of the timer RA control register.

• When using timer RB (RAPI_TIMER_RB specified)

[0]: Store the value of the timer RB one shot control register.

[1]: Store the value of the timer RB I/O control register.

[2]: Store the value of the timer RB prescaler register.

[3]: Store the value of the timer RB primary register.

[4]: Store the value of the timer RB secondary register.

[5]: Store the value of the timer RB mode register.

[6]: Store the value of the timer RB control register.

• When using timer RC (RAPI_TIMER_RC specified)

[0]: Store the value of the timer RC general register A.

[1]: Store the value of the timer RC general register B.

[2]: Store the value of the timer RC general register C.

[3]: Store the value of the timer RC general register D.

[4]: Store the value of the timer RC digital filter function select register.

[5]: Store the value of the timer RC control register 1.

[6]: Store the value of the timer RC control register 2.

[7]: Store the value of the timer RC I/O control register 0.

[8]: Store the value of the timer RC I/O control register 1.

[9]: Sotre the value of the timer RC status register.

[10]: Sotre the value of the timer RC interrupt enable register.

[11]: Store the value of the timer RC counter.

[12]: Store the value of the timer RC output master enable register.

[13]: Store the value of the timer RC mode register.

• When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD1)

[0]: Store the value of the timer RD mode register.

[1]: Store the value of the timer RD PWM mode register.

[2]: Store the value of the timer RD function control register.

[3]: Store the value of the timer RD general register Ai (i=0,1).

[4]: Store the value of the timer RD general register Bi (i=0,1).

[5]: Store the value of the timer RD general register Ci (i=0,1).

[6]: Store the value of the timer RD general register Di (i=0,1).

[7]: Store the value of the timer RD digital filter function select register i(i=0,1).

[8]: Store the value of the timer RD control register i(i=0,1).

[9]: Store the value of the timer RD I/O control register Ai(i=0,1).

[10]: Store the value of the timer RD I/O control register Ci(i=0,1).

[11]: Store the value of the timer RD status register i(i=0,1).

[12]: Store the value of the timer RD interrupt enable register i(i=0,1).

[13]: Store the value of the timer RD counter i(i=0,1).

[14]: Store the value of the timer RD start register.

[15]: Store the value of the timer RD output master enable register 1.

[16]: Store the value of the timer RD output master enable register 2.

[17]: Store the value of the timer RD output control register.

• When using timer RE (RAPI_TIMER_RE)

[0]: Store the value of the timer RE second data register.

[1]: Store the value of the timer RE minute data register.

[2]: Store the value of the timer RE hour data register.

[3]: Store the value of the timer RE day of week data register.

[4]: Store the value of the timer RE control register 2.

[5]: Store the value of the timer RE count source select register.

[6]: Store the value of the timer RE control register 1.

• When using timer RF (RAPI_TIMER_RF)

[0]: Store the value of the timer RF control register 0.

[1]: Store the value of capture & compare 0 register.

[2]: Store the value of compare 1 register.

[3]: Store the value of the timer RF output control register 0.

[4]: Store the value of the timer RF control register 1.

(H8S)

• When using timer RA (RAPI_TIMER_RA specified)

[0]: Store the value of the timer RA I/O control register.

[1]: Store the value of the timer RA prescaler register.

[2]: Store the value of the timer RA timer register.

[3]: Store the value of the timer RA mode register.

[4]: Store the value of the timer RA interrupt request status register.

[5]: Store the value of the timer RA control register.

• When using timer RB (RAPI_TIMER_RB specified)

[0]: Store the value of the timer RB one-shot control register.

[1]: Store the value of the timer RB I/O control register.

[2]: Store the value of the timer RB prescaler register.

[3]: Store the value of the timer RB primary register.

[4]: Store the value of the timer RB secondary register.

[5]: Store the value of the timer RB mode register.

[6]: Store the value of the timer RB interrupt request status register.

[7]: Store the value of the timer RB control register.

• When using timer RC (RAPI_TIMER_RC)

[0]: Store the value of timer RC control register 1.

[1]: Store the value of timer RC control register 2.

[2]: Store the value of timer RC interrupt enable register.

[3]: Store the value of timer RC status register.

[4]: Store the value of timer RC I/O control register 0.

[5]: Store the value of timer RC I/O control register 1.

[6]: Store the value of timer RC output enable register.

[7]: Store the value of timer RC digital filter function select register.

[8]: Store the value of timer RC A/D conversion start trigger control register.

[9]: Store the value of timer RC counter.

[10]: Store the value of general register A.

[11]: Store the value of general register B.

[12]: Store the value of general register C.

[13]: Store the value of general register D.

[14]: Store the value of timer RC mode register.

• When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD3 specified)

[0]: Store the value of the timer RD mode register.

[1]: Store the value of the timer RD PWM mode register.

[2]: Store the value of the timer RD function control register.

[3]: Store the value of the timer RD output master enable register 1.

[4]: Store the value of the timer RD output master enable register 2.

[5]: Store the value of the timer RD output control register.

[6]: Store the value of timer RD A/D conversion start trigger control register.

[7]: Store the value of the timer RD counter i(i=0,1).

[8]: Store the value of general register A_i(i=0,1).

[9]: Store the value of general register B_i(i=0,1).

[10]: Store the value of general register C_i(i=0,1).

[11]: Store the value of general register D_i(i=0,1).

[12]: Store the value of timer RD control register_i (i = 0, 1).

[13]: Store the value of timer RD I/O control register A_i (i = 0, 1).

[14]: Store the value of timer RD I/O control register C_i (i = 0, 1).

[15]: Store the value of timer RD status register_i (i = 0, 1).

[16]: Store the value of timer RD interrupt enable register_i (i = 0, 1).

[17]: Store the value of PWM mode output level control register_i (i = 0, 1).

[18]: Store the value of the timer RD digital filter function select register i(i = 0, 1).

[19]: Store the value of the timer RD start register.

• When using timer RE (RAPI_TIMER_RE specified)

[0]: Store the value of timer RE second data register/counter data register.

[1]: Store the value of timer RE minute data register/compare data register.

[2]: Store the value of timer RE hour data register.

[3]: Store the value of timer RE day-of-week data register.

[4]: Store the value of timer REcontrol register 2.

[5]: Store the value of timer RE clock source select register.

[6]: Store the value of timer RE interrupt flag register.

[7]: Store the value of timer REcontrol register 1.

• When using timer RG (RAPI_TIMER_RG specified)

[0]: Store the value of timer RG counter control register.

[1]: Store the value of timer RG control register.

[2]: Store the value of timer RG I/O control register.

[3]: Store the value of timer RG status register.

[4]: Store the value of timer RG interrupt enable register.

[5]: Store the value of timer RG counter.

[6]: Store the value of General register A.

[7]: Store the value of General register B.

[8]: Store the value of GRA buffer register.

[9]: Store the value of GRA buffer register.

[10]: Store the value of timer RG mode register.

(H8/300H)

• When using timer A (RAPI_TIMER_A specified)

[0]: Store the value of timer mode register A.

[1]: Store the value of timer counter A.

• When using timer B1 (RAPI_TIMER_B1 specified)

[0]: Store the value of timer mode register B1.

[1]: Store the value of timer counter B1.

• When using timer V (RAPI_TIMER_V specified)

[0]: Store the value of timer counter V.

[1]: Store the value of time constant register A.

[2]: Store the value of time constant register B.

[3]: Store the value of timer control register V0.

[4]: Store the value of timer control register V1.

[5]: Store the value of timer control/status register V.

• When using timer W (RAPI_TIMER_W specified)

[0]: Store the value of timer mode register W.

[1]: Store the value of timer control register W.

[2]: Store the value of timer interrupt master enable register W.

[3]: Store the value of timer status register W.

[4]: Store the value of timer I/O control register 0.

[5]: Store the value of timer I/O control register 1.

[6]: Store the value of the timer counter.

[7]: Store the value of general register A.

[8]: Store the value of general register B.

[9]: Store the value of general register C.

[10]: Store the value of general register D.

• When using timer Z (RAPI_TIMER_Z specified)

[0]: Store the value of the timer start register.

[1]: Store the value of the timer mode register.

[2]: Store the value of the timer PWM mode register.

[3]: Store the value of the timer function control register.

[4]: Store the value of the timer output master enable register.

[5]: Store the value of the timer output control register.

[6]: Store the value of timer counter_i (i = 0, 1).

[7]: Store the value of general register A_i (i = 0, 1).

[8]: Store the value of general register B_i (i=0, 1).

[9]: Store the value of general register C_i (i=0, 1).

[10]: Store the value of general register D_i (i=0, 1).

[11]: Store the value of timer control register_i (i = 0, 1).

[12]: Store the value of timer I/O control register A_i (i = 0, 1).

[13]: Store the value of timer I/O control register B_i (i = 0, 1).

[14]: Store the value of timer status register_i (i = 0, 1).

[15]: Store the value of timer interrupt enable register_i (i = 0, 1).

[16]: Store the value of PWM mode output level control register_i (i = 0, 1).

• When using timer RC (RAPI_TIMER_RC)

[0]: Store the value of timer RC control register 1.

[1]: Store the value of timer RC control register 2.

[2]: Store the value of timer RC interrupt enable register.

[3]: Store the value of timer RC status register.

[4]: Store the value of timer RC I/O control register 0.

[5]: Store the value of timer RC I/O control register 1.

[6]: Store the value of timer RC output enable register.

[7]: Store the value of timer RC digital filter function select register.

[8]: Store the value of timer RC counter.

[9]: Store the value of general register A.

[10]: Store the value of general register B.

[11]: Store the value of general register C.

[12]: Store the value of general register D.

[13]: Store the value of timer RC mode register.

• When using timer RD (RAPI_TIMER_RD0 to RAPI_TIMER_RD3 specified)

[0]: Store the value of the timer RD mode register.

[1]: Store the value of the timer RD PWM mode register.

[2]: Store the value of the timer RD function control register.

[3]: Store the value of the timer RD output master enable register 1.

[4]: Store the value of the timer RD output master enable register 2.

[5]: Store the value of the timer RD output control register.

[6]: Store the value of the timer RD counter i(i=0,1).

[7]: Store the value of general register A_i(i=0,1).

[8]: Store the value of general register B_i(i=0,1).

[9]: Store the value of general register C_i(i=0,1).

[10]: Store the value of general register D_i(i=0,1).

[11]: Store the value of timer RD control register_i (i = 0, 1).

[12]: Store the value of timer RD I/O control register A_i (i = 0, 1).

[13]: Store the value of timer RD I/O control register C_i (i = 0, 1).

[14]: Store the value of timer RD status register_i (i = 0, 1).

[15]: Store the value of timer RD interrupt enable register_i (i = 0, 1).

[16]: Store the value of PWM mode output level control register_i (i = 0, 1).

[17]: Store the value of the timer RD digital filter function select register i(i = 0, 1).

[18]: Store the value of the timer RD start register.

Return value

If the timer or Intelligent I/O specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Timer (register manipulation)

Reference

__SetTimerRegister, __EnableTimerRegister, __ClearTimerRegister

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable timers differ with each CPU used.

Program example

```
#include " rapi_timer_r8c_13.h"

void func( void )
{
   unsigned int data[7];

   /* Get the value of timer Z registers */
    __GetTimerRegister( RAPI_TIMER_Z, data );
}
```

4.2.3 I/O Port
__SetIOPort

Synopsis

<Set I/O port>

Boolean __SetIOPort(unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)

Description

Sets the operating conditions of a specified I/O port.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value. Note, however, that multiple ports cannot be specified at the same time.

(M16C)

The definition values corresponding to each I/O port are listed below.

RAPI_PORT_0_0	Port $P0_0$	RAPI_PORT_0_1	Port $P0_1$
RAPI_PORT_0_2	Port $P0_2$	RAPI_PORT_0_3	Port $P0_3$
RAPI_PORT_0_4	Port $P0_4$	RAPI_PORT_0_5	Port $P0_5$
RAPI_PORT_0_6	Port $P0_6$	RAPI_PORT_0_7	Port $P0_7$
RAPI_PORT_1_0	Port $P1_0$	RAPI_PORT_1_1	Port $P1_1$
RAPI_PORT_1_2	Port $P1_2$	RAPI_PORT_1_3	Port $P1_3$
RAPI_PORT_1_4	Port $P1_4$	RAPI_PORT_1_5	Port $P1_5$
RAPI_PORT_1_6	Port $P1_6$	RAPI_PORT_1_7	Port $P1_7$
RAPI_PORT_2_0	Port $P2_0$	RAPI_PORT_2_1	Port $P2_1$
RAPI_PORT_2_2	Port $P2_2$	RAPI_PORT_2_3	Port $P2_3$
RAPI_PORT_2_4	Port $P2_4$	RAPI_PORT_2_5	Port $P2_5$
RAPI_PORT_2_6	Port $P2_6$	RAPI_PORT_2_7	Port $P2_7$
RAPI_PORT_3_0	Port $P3_0$	RAPI_PORT_3_1	Port $P3_1$
RAPI_PORT_3_2	Port $P3_2$	RAPI_PORT_3_3	Port $P3_3$
RAPI_PORT_3_4	Port $P3_4$	RAPI_PORT_3_5	Port $P3_5$
RAPI_PORT_3_6	Port $P3_6$	RAPI_PORT_3_7	Port $P3_7$
RAPI_PORT_6_0	Port $P6_0$	RAPI_PORT_6_1	Port $P6_1$
RAPI_PORT_6_2	Port $P6_2$	RAPI_PORT_6_3	Port $P6_3$
RAPI_PORT_6_4	Port $P6_4$	RAPI_PORT_6_5	Port $P6_5$
RAPI_PORT_6_6	Port $P6_6$	RAPI_PORT_6_7	Port $P6_7$
RAPI_PORT_7_0	Port $P7_0$	RAPI_PORT_7_1	Port $P7_1$
RAPI_PORT_7_2	Port $P7_2$	RAPI_PORT_7_3	Port $P7_3$
RAPI_PORT_7_4	Port $P7_4$	RAPI_PORT_7_5	Port $P7_5$
RAPI_PORT_7_6	Port $P7_6$	RAPI_PORT_7_7	Port $P7_7$
RAPI_PORT_8_0	Port $P8_0$	RAPI_PORT_8_1	Port $P8_1$
RAPI_PORT_8_2	Port $P8_2$	RAPI_PORT_8_3	Port $P8_3$
RAPI_PORT_8_4	Port $P8_4$	RAPI_PORT_8_5	Port $P8_5$
RAPI_PORT_8_6	Port $P8_6$	RAPI_PORT_8_7	Port $P8_7$

RAPI_PORT_9_0	Port $P9_0$	RAPI_PORT_9_1	Port $P9_1$
RAPI_PORT_9_2	Port $P9_2$	RAPI_PORT_9_3	Port $P9_3$
RAPI_PORT_9_5	Port $P9_5$	RAPI_PORT_9_6	Port $P9_6$
RAPI_PORT_9_7	Port $P9_7$	RAPI_PORT_10_0	Port $P10_0$
RAPI_PORT_10_1	Port $P10_1$	RAPI_PORT_10_2	Port $P10_2$
RAPI_PORT_10_3	Port $P10_3$	RAPI_PORT_10_4	Port $P10_4$
RAPI_PORT_10_5	Port $P10_5$	RAPI_PORT_10_6	Port $P10_6$
RAPI_PORT_10_7	Port $P10_7$		

The definition values related to port settings are described below.

RAPI_PORT_INPUT	Sets a selected port for input.
RAPI_PORT_OUTPUT	Sets a selected port for output.
RAPI_PULLED_HIGH	Sets a selected port to be pulled high.
RAPI_NOT_PULLED _HIGH	Sets a selected port not to be pulled high.
RAPI_LATCH	Sets a selected port to read the port latch regardless of whether it is set for input or output. Specifiable only when port P1 is used.

(R8C)

The definition values corresponding to each I/O port are listed below.

RAPI_PORT_0_0	Port $P0_0$	RAPI_PORT_0_1	Port $P0_1$
RAPI_PORT_0_2	Port $P0_2$	RAPI_PORT_0_3	Port $P0_3$
RAPI_PORT_0_4	Port $P0_4$	RAPI_PORT_0_5	Port $P0_5$
RAPI_PORT_0_6	Port $P0_6$	RAPI_PORT_0_7	Port $P0_7$
RAPI_PORT_1_0	Port $P1_0$	RAPI_PORT_1_1	Port $P1_1$
RAPI_PORT_1_2	Port $P1_2$	RAPI_PORT_1_3	Port $P1_3$
RAPI_PORT_1_4	Port $P1_4$	RAPI_PORT_1_5	Port $P1_5$
RAPI_PORT_1_6	Port $P1_6$	RAPI_PORT_1_7	Port $P1_7$
RAPI_PORT_2_0	Port $P2_0$	RAPI_PORT_2_1	Port $P2_1$
RAPI_PORT_2_2	Port $P2_2$	RAPI_PORT_2_3	Port $P2_3$
RAPI_PORT_2_4	Port $P2_4$	RAPI_PORT_2_5	Port $P2_5$
RAPI_PORT_2_6	Port $P2_6$	RAPI_PORT_2_7	Port $P2_7$
RAPI_PORT_3_0	Port $P3_0$	RAPI_PORT_3_1	Port $P3_1$
RAPI_PORT_3_2	Port $P3_2$	RAPI_PORT_3_3	Port $P3_3$
RAPI_PORT_3_4	Port $P3_4$	RAPI_PORT_3_5	Port $P3_5$
RAPI_PORT_3_6	Port $P3_6$	RAPI_PORT_3_7	Port $P3_7$
RAPI_PORT_4_3	Port $P4_3$	RAPI_PORT_4_4	Port $P4_4$
RAPI_PORT_4_5	Port $P4_5$	RAPI_PORT_5_0	Port $P5_0$
RAPI_PORT_5_1	Port $P5_1$	RAPI_PORT_5_2	Port $P5_2$
RAPI_PORT_5_3	Port $P5_3$	RAPI_PORT_5_4	Port $P5_4$
RAPI_PORT_5_5	Port $P5_5$	RAPI_PORT_5_6	Port $P5_6$
RAPI_PORT_5_7	Port $P5_7$	RAPI_PORT_6_0	Port $P6_0$
RAPI_PORT_6_1	Port $P6_1$	RAPI_PORT_6_2	Port $P6_2$
RAPI_PORT_6_3	Port $P6_3$	RAPI_PORT_6_4	Port $P6_4$
RAPI_PORT_6_5	Port $P6_5$	RAPI_PORT_6_6	Port $P6_6$

RAPI_PORT_6_7	Port $P6_7$	RAPI_PORT_7_0	Port $P7_0$
RAPI_PORT_7_1	Port $P7_1$	RAPI_PORT_7_2	Port $P7_2$
RAPI_PORT_7_3	Port $P7_3$	RAPI_PORT_7_4	Port $P7_4$
RAPI_PORT_7_5	Port $P7_5$	RAPI_PORT_7_6	Port $P7_6$
RAPI_PORT_7_7	Port $P7_7$	RAPI_PORT_8_0	Port $P8_0$
RAPI_PORT_8_1	Port $P8_1$	RAPI_PORT_8_2	Port $P8_2$
RAPI_PORT_8_3	Port $P8_3$	RAPI_PORT_8_4	Port $P8_4$
RAPI_PORT_8_5	Port $P8_5$	RAPI_PORT_8_6	Port $P8_6$
RAPI_PORT_8_7	Port $P8_7$	RAPI_PORT_9_0	Port $P9_0$
RAPI_PORT_9_1	Port $P9_1$	RAPI_PORT_9_2	Port $P9_2$
RAPI_PORT_9_3	Port $P9_3$		

The definition values corresponding to each I/O port are listed below.

RAPI_PORT_INPUT	Sets a selected port for input.
RAPI_PORT_OUTPUT	Sets a selected port for output.
RAPI_PULLED_HIGH	Sets a selected port to be pulled high.
RAPI_NOT_PULLED_HIGH	Sets a selected port not to be pulled high.
RAPI_DRIVE_CAPACITY_H	Sets the N-channel output transistor drive capacity of a selected port to High. Specifiable only when port P1 is used.
RAPI_DRIVE_CAPACITY_L	Sets the N-channel output transistor drive capacity of a selected port to Low. Specifiable only when port P1 is used.

(H8S)

The definition values corresponding to each I/O port are listed below.

RAPI_PORT_1_0	Port P10	RAPI_PORT_1_1	Port P11
RAPI_PORT_1_2	Port P12	RAPI_PORT_1_3	Port P13
RAPI_PORT_1_4	Port P14	RAPI_PORT_1_5	Port P15
RAPI_PORT_1_6	Port P16	RAPI_PORT_1_7	Port P17
RAPI_PORT_2_0	Port P20	RAPI_PORT_2_1	Port P21
RAPI_PORT_2_2	Port P22	RAPI_PORT_2_3	Port P23
RAPI_PORT_2_4	Port P24	RAPI_PORT_2_5	Port P25
RAPI_PORT_2_6	Port P26	RAPI_PORT_2_7	Port P27
RAPI_PORT_3_0	Port P30	RAPI_PORT_3_1	Port P31
RAPI_PORT_3_2	Port P32	RAPI_PORT_3_3	Port P33
RAPI_PORT_3_4	Port P34	RAPI_PORT_3_5	Port P35
RAPI_PORT_3_6	Port P36	RAPI_PORT_3_7	Port P37
RAPI_PORT_5_0	Port P50	RAPI_PORT_5_1	Port P51
RAPI_PORT_5_2	Port P52	RAPI_PORT_5_3	Port P53
RAPI_PORT_5_4	Port P54	RAPI_PORT_5_5	Port P55
RAPI_PORT_5_6	Port P56	RAPI_PORT_5_7	Port P57
RAPI_PORT_6_0	Port P60	RAPI_PORT_6_1	Port P61
RAPI_PORT_6_2	Port P62	RAPI_PORT_6_3	Port P63
RAPI_PORT_6_4	Port P64	RAPI_PORT_6_5	Port P65
RAPI_PORT_6_6	Port P66	RAPI_PORT_6_7	Port P67
RAPI_PORT_8_5	Port P85	RAPI_PORT_8_6	Port P86

RAPI_PORT_8_7	Port P87	RAPI_PORT_9_0	Port P90
RAPI_PORT_9_1	Port P91	RAPI_PORT_9_2	Port P92
RAPI_PORT_9_3	Port P93	RAPI_PORT_9_4	Port P94
RAPI_PORT_9_5	Port P95	RAPI_PORT_9_6	Port P96
RAPI_PORT_9_7	Port P97	RAPI_PORT_A_0	Port PA0
RAPI_PORT_A_1	Port PA1	RAPI_PORT_A_2	Port PA2
RAPI_PORT_A_3	Port PA3	RAPI_PORT_A_4	Port PA4
RAPI_PORT_A_5	Port PA5	RAPI_PORT_A_6	Port PA6
RAPI_PORT_A_7	Port PA7	RAPI_PORT_B_0	Port PB0
RAPI_PORT_B_1	Port PB1	RAPI_PORT_B_2	Port PB2
RAPI_PORT_B_3	Port PB3	RAPI_PORT_B_4	Port PB4
RAPI_PORT_B_5	Port PB5	RAPI_PORT_B_6	Port PB6
RAPI_PORT_B_7	Port PB7	RAPI_PORT_J_0	Port PJ0
RAPI_PORT_J_1	Port PJ1		

The definition values related to port settings are described below.

RAPI_PORT_INPUT	Sets a selected port for input.
RAPI_PORT_OUTPUT	Sets a selected port for output.
RAPI_PULLED_HIGH	Sets a selected port to be pulled high.
RAPI_NOT_PULLED_HIGH	Sets a selected port not to be pulled high.
RAPI_DRIVE_CAPACITY_H	Sets the drive capacity of a selected port to High.

(H8/300H)

The definition values corresponding to each I/O port are listed below.

RAPI_PORT_1_0	Port P10	RAPI_PORT_1_1	Port P11
RAPI_PORT_1_2	Port P12	RAPI_PORT_1_4	Port P14
RAPI_PORT_1_5	Port P15	RAPI_PORT_1_6	Port P16
RAPI_PORT_1_7	Port P17	RAPI_PORT_2_0	Port P20
RAPI_PORT_2_1	Port P21	RAPI_PORT_2_2	Port P22
RAPI_PORT_2_3	Port P23	RAPI_PORT_2_4	Port P24
RAPI_PORT_2_5	Port P25	RAPI_PORT_2_6	Port P26
RAPI_PORT_2_7	Port P27	RAPI_PORT_3_0	Port P30
RAPI_PORT_3_1	Port P31	RAPI_PORT_3_2	Port P32
RAPI_PORT_3_3	Port P33	RAPI_PORT_3_4	Port P34
RAPI_PORT_3_5	Port P35	RAPI_PORT_3_6	Port P36
RAPI_PORT_3_7	Port P37	RAPI_PORT_5_0	Port P50
RAPI_PORT_5_1	Port P51	RAPI_PORT_5_2	Port P52
RAPI_PORT_5_3	Port P53	RAPI_PORT_5_4	Port P54
RAPI_PORT_5_5	Port P55	RAPI_PORT_5_6	Port P56
RAPI_PORT_5_7	Port P57	RAPI_PORT_6_0	Port P60
RAPI_PORT_6_1	Port P61	RAPI_PORT_6_2	Port P62
RAPI_PORT_6_3	Port P63	RAPI_PORT_6_4	Port P64
RAPI_PORT_6_5	Port P65	RAPI_PORT_6_6	Port P66
RAPI_PORT_6_7	Port P67	RAPI_PORT_7_0	Port P70
RAPI_PORT_7_1	Port P71	RAPI_PORT_7_2	Port P72

RAPI_PORT_7_4	Port P74	RAPI_PORT_7_5	Port P75
RAPI_PORT_7_6	Port P76	RAPI_PORT_7_7	Port P77
RAPI_PORT_8_0	Port P80	RAPI_PORT_8_1	Port P81
RAPI_PORT_8_2	Port P82	RAPI_PORT_8_3	Port P83
RAPI_PORT_8_4	Port P84	RAPI_PORT_8_5	Port P85
RAPI_PORT_8_6	Port P86	RAPI_PORT_8_7	Port P87
RAPI_PORT_9_0	Port P90	RAPI_PORT_9_1	Port P91
RAPI_PORT_9_2	Port P92	RAPI_PORT_9_3	Port P93
RAPI_PORT_9_4	Port P94	RAPI_PORT_9_5	Port P95
RAPI_PORT_9_6	Port P96	RAPI_PORT_9_7	Port P97
RAPI_PORT_C_0	Port PC0	RAPI_PORT_C_1	Port PC1
RAPI_PORT_C_2	PortPC2	RAPI_PORT_C_3	PortPC3
RAPI_PORT_D_0	PortPD0	RAPI_PORT_D_1	PortPD1
RAPI_PORT_D_2	PortPD2	RAPI_PORT_D_3	PortPD3
RAPI_PORT_D_4	PortPD4	RAPI_PORT_D_5	PortPD5
RAPI_PORT_D_6	PortPD6	RAPI_PORT_D_7	PortPD7
RAPI_PORT_E_0	PortPE0	RAPI_PORT_E_1	PortPE1
RAPI_PORT_E_2	PortPE2	RAPI_PORT_E_3	PortPE3
RAPI_PORT_E_4	PortPE4	RAPI_PORT_E_5	PortPE5
RAPI_PORT_E_6	PortPE6	RAPI_PORT_E_7	PortPE7
RAPI_PORT_F_0	PortPF0	RAPI_PORT_G_0	PortPG0
RAPI_PORT_G_1	PortPG1	RAPI_PORT_G_2	PortPG2
RAPI_PORT_G_3	PortPG3	RAPI_PORT_G_4	PortPG4
RAPI_PORT_G_5	PortPG5	RAPI_PORT_G_6	PortPG6
RAPI_PORT_G_7	PortPG7	RAPI_PORT_H_0	PortPH0
RAPI_PORT_H_1	PortPH1	RAPI_PORT_H_2	PortPH2
RAPI_PORT_H_3	PortPH3	RAPI_PORT_H_4	PortPH4
RAPI_PORT_H_5	PortPH5	RAPI_PORT_H_6	PortPH6
RAPI_PORT_H_7	PortPH7	RAPI_PORT_J_0	PortPJ0
RAPI_PORT_J_1	PortPJ1		

The definition values related to port settings are described below.

RAPI_PORT_INPUT	Sets a selected port for input.
RAPI_PORT_OUTPUT	Sets a selected port for output.
RAPI_PULLED_HIGH	Sets a selected port to be pulled high.
RAPI_NOT_PULLED_HIGH	Sets a selected port not to be pulled high.

[data2]

(M16C)

Specify the digital filter width of the digital debounce facility assigned to _NMI/_SD. Specifiable only when port P85 is used.

Specify the digital filter width of the digital debounce facility assigned to INPC17/_INT5. Specifiable only when port P17 is used.

When using any other port, set 0 for this data.

(R8C) (H8S)(H8/300H)

Specify 0.

Return value

If the I/O port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

I/O port

Reference

__ReadIOPort, __WriteIOPort, __SetIOPortRegister, __ReadIOPortRegister, __WriteIOPortRegister

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O ports differ with each CPU used.
- The API cannot set function that the specified I/O port dont have.

Program example

```
#include " rapi_io_port_r8c_13.h

void func( void )
{
   /* Set up port P03 as input port */
   __SetIOPort(RAPI_PORT_0_3| RAPI_PORT_INPUT| RAPI_PULLED_HIGH, 0, 0 );
}
```

__ReadIOPort

Synopsis

<Read from I/O port>

Boolean __ReadIOPort(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the variable in which the value read from I/O port is stored.

Description

Gets the value of a specified I/O port.

[data1]

Specify an I/O port from which data is read. The definition values corresponding to each I/O port are listed below.

(M16C)

RAPI_PORT_0_0	Port $P0_0$	RAPI_PORT_0_1	Port $P0_1$
RAPI_PORT_0_2	Port $P0_2$	RAPI_PORT_0_3	Port $P0_3$
RAPI_PORT_0_4	Port $P0_4$	RAPI_PORT_0_5	Port $P0_5$
RAPI_PORT_0_6	Port $P0_6$	RAPI_PORT_0_7	Port $P0_7$
RAPI_PORT_1_0	Port $P1_0$	RAPI_PORT_1_1	Port $P1_1$
RAPI_PORT_1_2	Port $P1_2$	RAPI_PORT_1_3	Port $P1_3$
RAPI_PORT_1_4	Port $P1_4$	RAPI_PORT_1_5	Port $P1_5$
RAPI_PORT_1_6	Port $P1_6$	RAPI_PORT_1_7	Port $P1_7$
RAPI_PORT_2_0	Port $P2_0$	RAPI_PORT_2_1	Port $P2_1$
RAPI_PORT_2_2	Port $P2_2$	RAPI_PORT_2_3	Port $P2_3$
RAPI_PORT_2_4	Port $P2_4$	RAPI_PORT_2_5	Port $P2_5$
RAPI_PORT_2_6	Port $P2_6$	RAPI_PORT_2_7	Port $P2_7$
RAPI_PORT_3_0	Port $P3_0$	RAPI_PORT_3_1	Port $P3_1$
RAPI_PORT_3_2	Port $P3_2$	RAPI_PORT_3_3	Port $P3_3$
RAPI_PORT_3_4	Port $P3_4$	RAPI_PORT_3_5	Port $P3_5$
RAPI_PORT_3_6	Port $P3_6$	RAPI_PORT_3_7	Port $P3_7$
RAPI_PORT_6_0	Port $P6_0$	RAPI_PORT_6_1	Port $P6_1$
RAPI_PORT_6_2	Port $P6_2$	RAPI_PORT_6_3	Port $P6_3$
RAPI_PORT_6_4	Port $P6_4$	RAPI_PORT_6_5	Port $P6_5$
RAPI_PORT_6_6	Port $P6_6$	RAPI_PORT_6_7	Port $P6_7$
RAPI_PORT_7_0	Port $P7_0$	RAPI_PORT_7_1	Port $P7_1$
RAPI_PORT_7_2	Port $P7_2$	RAPI_PORT_7_3	Port $P7_3$
RAPI_PORT_7_4	Port $P7_4$	RAPI_PORT_7_5	Port $P7_5$
RAPI_PORT_7_6	Port $P7_6$	RAPI_PORT_7_7	Port $P7_7$
RAPI_PORT_8_0	Port $P8_0$	RAPI_PORT_8_1	Port $P8_1$
RAPI_PORT_8_2	Port $P8_2$	RAPI_PORT_8_3	Port $P8_3$
RAPI_PORT_8_4	Port $P8_4$	RAPI_PORT_8_5	Port $P8_5$
RAPI_PORT_8_6	Port $P8_6$	RAPI_PORT_8_7	Port $P8_7$
RAPI_PORT_9_0	Port $P9_0$	RAPI_PORT_9_1	Port $P9_1$
RAPI_PORT_9_2	Port $P9_2$	RAPI_PORT_9_3	Port $P9_3$

RAPI_PORT_9_5	Port $P9_5$	RAPI_PORT_9_6	Port $P9_6$
RAPI_PORT_9_7	Port $P9_7$	RAPI_PORT_10_0	Port $P10_0$
RAPI_PORT_10_1	Port $P10_1$	RAPI_PORT_10_2	Port $P10_2$
RAPI_PORT_10_3	Port $P10_3$	RAPI_PORT_10_4	Port $P10_4$
RAPI_PORT_10_5	Port $P10_5$	RAPI_PORT_10_6	Port $P10_6$
RAPI_PORT_10_7	Port $P10_7$		

(R8C)

RAPI_PORT_0_0	Port $P0_0$	RAPI_PORT_0_1	Port $P0_1$
RAPI_PORT_0_2	Port $P0_2$	RAPI_PORT_0_3	Port $P0_3$
RAPI_PORT_0_4	Port $P0_4$	RAPI_PORT_0_5	Port $P0_5$
RAPI_PORT_0_6	Port $P0_6$	RAPI_PORT_0_7	Port $P0_7$
RAPI_PORT_1_0	Port $P1_0$	RAPI_PORT_1_1	Port $P1_1$
RAPI_PORT_1_2	Port $P1_2$	RAPI_PORT_1_3	Port $P1_3$
RAPI_PORT_1_4	Port $P1_4$	RAPI_PORT_1_5	Port $P1_5$
RAPI_PORT_1_6	Port $P1_6$	RAPI_PORT_1_7	Port $P1_7$
RAPI_PORT_2_0	Port $P2_0$	RAPI_PORT_2_1	Port $P2_1$
RAPI_PORT_2_2	Port $P2_2$	RAPI_PORT_2_3	Port $P2_3$
RAPI_PORT_2_4	Port $P2_4$	RAPI_PORT_2_5	Port $P2_5$
RAPI_PORT_2_6	Port $P2_6$	RAPI_PORT_2_7	Port $P2_7$
RAPI_PORT_3_0	Port $P3_0$	RAPI_PORT_3_1	Port $P3_1$
RAPI_PORT_3_2	Port $P3_2$	RAPI_PORT_3_3	Port $P3_3$
RAPI_PORT_3_4	Port $P3_4$	RAPI_PORT_3_5	Port $P3_5$
RAPI_PORT_3_6	Port $P3_6$	RAPI_PORT_3_7	Port $P3_7$
RAPI_PORT_4_2	Port $P4_2$	RAPI_PORT_4_3	Port $P4_3$
RAPI_PORT_4_4	Port $P4_4$	RAPI_PORT_4_5	Port $P4_5$
RAPI_PORT_4_6	Port $P4_6$	RAPI_PORT_4_7	Port $P4_7$
RAPI_PORT_5_0	Port $P5_0$	RAPI_PORT_5_1	Port $P5_1$
RAPI_PORT_5_2	Port $P5_2$	RAPI_PORT_5_3	Port $P5_3$
RAPI_PORT_5_4	Port $P5_4$	RAPI_PORT_5_5	Port $P5_5$
RAPI_PORT_5_6	Port $P5_6$	RAPI_PORT_5_7	Port $P5_7$
RAPI_PORT_6_0	Port $P6_0$	RAPI_PORT_6_1	Port $P6_1$
RAPI_PORT_6_2	Port $P6_2$	RAPI_PORT_6_3	Port $P6_3$
RAPI_PORT_6_4	Port $P6_4$	RAPI_PORT_6_5	Port $P6_5$
RAPI_PORT_6_6	Port $P6_6$	RAPI_PORT_6_7	Port $P6_7$
RAPI_PORT_7_0	Port $P7_0$	RAPI_PORT_7_1	Port $P7_1$
RAPI_PORT_7_2	Port $P7_2$	RAPI_PORT_7_3	Port $P7_3$
RAPI_PORT_7_4	Port $P7_4$	RAPI_PORT_7_5	Port $P7_5$
RAPI_PORT_7_6	Port $P7_6$	RAPI_PORT_7_7	Port $P7_7$
RAPI_PORT_8_0	Port $P8_0$	RAPI_PORT_8_1	Port $P8_1$
RAPI_PORT_8_2	Port $P8_2$	RAPI_PORT_8_3	Port $P8_3$
RAPI_PORT_8_4	Port $P8_4$	RAPI_PORT_8_5	Port $P8_5$
RAPI_PORT_8_6	Port $P8_6$	RAPI_PORT_8_7	Port $P8_7$
RAPI_PORT_9_0	Port $P9_0$	RAPI_PORT_9_1	Port $P9_1$

RAPI_PORT_9_2	Port $P9_2$	RAPI_PORT_9_3	Port $P9_3$

(H8S)

RAPI_PORT_1_0	Port P10	RAPI_PORT_1_1	Port P11
RAPI_PORT_1_2	Port P12	RAPI_PORT_1_3	Port P13
RAPI_PORT_1_4	Port P14	RAPI_PORT_1_5	Port P15
RAPI_PORT_1_6	Port P16	RAPI_PORT_1_7	Port P17
RAPI_PORT_2_0	Port P20	RAPI_PORT_2_1	Port P21
RAPI_PORT_2_2	Port P22	RAPI_PORT_2_3	Port P23
RAPI_PORT_2_4	Port P24	RAPI_PORT_2_5	Port P25
RAPI_PORT_2_6	Port P26	RAPI_PORT_2_7	Port P27
RAPI_PORT_3_0	Port P30	RAPI_PORT_3_1	Port P31
RAPI_PORT_3_2	Port P32	RAPI_PORT_3_3	Port P33
RAPI_PORT_3_4	Port P34	RAPI_PORT_3_5	Port P35
RAPI_PORT_3_6	Port P36	RAPI_PORT_3_7	Port P37
RAPI_PORT_5_0	Port P50	RAPI_PORT_5_1	Port P51
RAPI_PORT_5_2	Port P52	RAPI_PORT_5_3	Port P53
RAPI_PORT_5_4	Port P54	RAPI_PORT_5_5	Port P55
RAPI_PORT_5_6	Port P56	RAPI_PORT_5_7	Port P57
RAPI_PORT_6_0	Port P60	RAPI_PORT_6_1	Port P61
RAPI_PORT_6_2	Port P62	RAPI_PORT_6_3	Port P63
RAPI_PORT_6_4	Port P64	RAPI_PORT_6_5	Port P65
RAPI_PORT_6_6	Port P66	RAPI_PORT_6_7	Port P67
RAPI_PORT_8_5	Port P85	RAPI_PORT_8_6	Port P86
RAPI_PORT_8_7	Port P87	RAPI_PORT_9_0	Port P90
RAPI_PORT_9_1	Port P91	RAPI_PORT_9_2	Port P92
RAPI_PORT_9_3	Port P93	RAPI_PORT_9_4	Port P94
RAPI_PORT_9_5	Port P95	RAPI_PORT_9_6	Port P96
RAPI_PORT_9_7	Port P97	RAPI_PORT_A_0	Port PA0
RAPI_PORT_A_1	Port PA1	RAPI_PORT_A_2	Port PA2
RAPI_PORT_A_3	Port PA3	RAPI_PORT_A_4	Port PA4
RAPI_PORT_A_5	Port PA5	RAPI_PORT_A_6	Port PA6
RAPI_PORT_A_7	Port PA7	RAPI_PORT_B_0	Port PB0
RAPI_PORT_B_1	Port PB1	RAPI_PORT_B_2	Port PB2
RAPI_PORT_B_3	Port PB3	RAPI_PORT_B_4	Port PB4
RAPI_PORT_B_5	Port PB5	RAPI_PORT_B_6	Port PB6
RAPI_PORT_B_7	Port PB7	RAPI_PORT_J_0	Port PJ0
RAPI_PORT_J_1	Port PJ1		

(H8/300H)

RAPI_PORT_1_0	Port P10	RAPI_PORT_1_1	Port P11
RAPI_PORT_1_2	Port P12	RAPI_PORT_1_4	Port P14
RAPI_PORT_1_5	Port P15	RAPI_PORT_1_6	Port P16
RAPI_PORT_1_7	Port P17	RAPI_PORT_2_0	Port P20
RAPI_PORT_2_1	Port P21	RAPI_PORT_2_2	Port P22

RAPI_PORT_2_3	Port P23	RAPI_PORT_2_4	Port P24
RAPI_PORT_2_5	PortP25	RAPI_PORT_2_6	PortP26
RAPI_PORT_2_7	PortP27	RAPI_PORT_3_0	PortP30
RAPI_PORT_3_1	PortP31	RAPI_PORT_3_2	PortP32
RAPI_PORT_3_3	PortP33	RAPI_PORT_3_4	PortP34
RAPI_PORT_3_5	PortP35	RAPI_PORT_3_6	PortP36
RAPI_PORT_3_7	PortP37	RAPI_PORT_5_0	PortP50
RAPI_PORT_5_1	PortP51	RAPI_PORT_5_2	PortP52
RAPI_PORT_5_3	PortP53	RAPI_PORT_5_4	PortP54
RAPI_PORT_5_5	PortP55	RAPI_PORT_5_6	PortP56
RAPI_PORT_5_7	PortP57	RAPI_PORT_6_0	PortP60
RAPI_PORT_6_1	PortP61	RAPI_PORT_6_2	PortP62
RAPI_PORT_6_3	PortP63	RAPI_PORT_6_4	PortP64
RAPI_PORT_6_5	PortP65	RAPI_PORT_6_6	PortP66
RAPI_PORT_6_7	PortP67	RAPI_PORT_7_0	PortP70
RAPI_PORT_7_1	PortP71	RAPI_PORT_7_2	PortP72
RAPI_PORT_7_4	PortP74	RAPI_PORT_7_5	PortP75
RAPI_PORT_7_6	PortP76	RAPI_PORT_7_7	PortP77
RAPI_PORT_8_0	PortP80	RAPI_PORT_8_1	PortP81
RAPI_PORT_8_2	PortP82	RAPI_PORT_8_3	PortP83
RAPI_PORT_8_4	PortP84	RAPI_PORT_8_5	PortP85
RAPI_PORT_8_6	PortP86	RAPI_PORT_8_7	PortP87
RAPI_PORT_9_0	PortP90	RAPI_PORT_9_1	PortP91
RAPI_PORT_9_2	PortP92	RAPI_PORT_9_3	PortP93
RAPI_PORT_9_4	PortP94	RAPI_PORT_9_5	PortP95
RAPI_PORT_9_6	PortP96	RAPI_PORT_9_7	PortP97
RAPI_PORT_B_0	PortPB0	RAPI_PORT_B_1	PortPB1
RAPI_PORT_B_2	PortPB2	RAPI_PORT_B_3	PortPB3
RAPI_PORT_B_4	PortPB4	RAPI_PORT_B_5	PortPB5
RAPI_PORT_B_6	PortPB6	RAPI_PORT_B_7	PortPB7
RAPI_PORT_C_0	PortPC0	RAPI_PORT_C_1	PortPC1
RAPI_PORT_C_2	PortPC2	RAPI_PORT_C_3	PortPC3
RAPI_PORT_D_0	PortPD0	RAPI_PORT_D_1	PortPD1
RAPI_PORT_D_2	PortPD2	RAPI_PORT_D_3	PortPD3
RAPI_PORT_D_4	PortPD4	RAPI_PORT_D_5	PortPD5
RAPI_PORT_D_6	PortPD6	RAPI_PORT_D_7	PortPD7
RAPI_PORT_E_0	PortPE0	RAPI_PORT_E_1	PortPE1
RAPI_PORT_E_2	PortPE2	RAPI_PORT_E_3	PortPE3
RAPI_PORT_E_4	PortPE4	RAPI_PORT_E_5	PortPE5
RAPI_PORT_E_6	PortPE6	RAPI_PORT_E_7	PortPE7
RAPI_PORT_F_0	PortPF0	RAPI_PORT_F_1	PortPF1
RAPI_PORT_F_2	PortPF2	RAPI_PORT_F_3	PortPF3
RAPI_PORT_F_4	PortPF4	RAPI_PORT_F_5	PortPF5

RAPI_PORT_F_6	PortPF6	RAPI_PORT_F_7	PortPF7
RAPI_PORT_G_0	PortPG0	RAPI_PORT_G_1	PortPG1
RAPI_PORT_G_2	PortPG2	RAPI_PORT_G_3	PortPG3
RAPI_PORT_G_4	PortPG4	RAPI_PORT_G_5	PortPG5
RAPI_PORT_G_6	PortPG6	RAPI_PORT_G_7	PortPG7
RAPI_PORT_H_0	PortPH0	RAPI_PORT_H_1	PortPH1
RAPI_PORT_H_2	PortPH2	RAPI_PORT_H_3	PortPH3
RAPI_PORT_H_4	PortPH4	RAPI_PORT_H_5	PortPH5
RAPI_PORT_H_6	PortPH6	RAPI_PORT_H_7	PortPH7
RAPI_PORT_J_0	PortPJ0	RAPI_PORT_J_1	PortPJ1

Return value

If the I/O port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

I/O port

Reference

__SetIOPort, __WriteIOPort, __SetIOPortRegister, __ReadIOPortRegister, __WriteIOPortRegister

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O ports differ with each CPU used.
- Ports in port B of the H8/300H that are used as analog input pins cannot be used as input ports.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
   /* Get the value of port P12 */
   __ReadIOPort(RAPI_PORT_1_2, &data );
}
```

__WriteIOPort

Synopsis

<Write to I/O port>

Boolean __WriteIOPort(unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Data to be written to I/O port

Description

Writes data to a specified I/O port.

[data1]

Specify an I/O port to which data is written. The definition values corresponding to each I/O port are listed below.

(M16C)

RAPI_PORT_0_0	Port $P0_0$	RAPI_PORT_0_1	Port $P0_1$
RAPI_PORT_0_2	Port $P0_2$	RAPI_PORT_0_3	Port $P0_3$
RAPI_PORT_0_4	Port $P0_4$	RAPI_PORT_0_5	Port $P0_5$
RAPI_PORT_0_6	Port $P0_6$	RAPI_PORT_0_7	Port $P0_7$
RAPI_PORT_1_0	Port $P1_0$	RAPI_PORT_1_1	Port $P1_1$
RAPI_PORT_1_2	Port $P1_2$	RAPI_PORT_1_3	Port $P1_3$
RAPI_PORT_1_4	Port $P1_4$	RAPI_PORT_1_5	Port $P1_5$
RAPI_PORT_1_6	Port $P1_6$	RAPI_PORT_1_7	Port $P1_7$
RAPI_PORT_2_0	Port $P2_0$	RAPI_PORT_2_1	Port $P2_1$
RAPI_PORT_2_2	Port $P2_2$	RAPI_PORT_2_3	Port $P2_3$
RAPI_PORT_2_4	Port $P2_4$	RAPI_PORT_2_5	Port $P2_5$
RAPI_PORT_2_6	Port $P2_6$	RAPI_PORT_2_7	Port $P2_7$
RAPI_PORT_3_0	Port $P3_0$	RAPI_PORT_3_1	Port $P3_1$
RAPI_PORT_3_2	Port $P3_2$	RAPI_PORT_3_3	Port $P3_3$
RAPI_PORT_3_4	Port $P3_4$	RAPI_PORT_3_5	Port $P3_5$
RAPI_PORT_3_6	Port $P3_6$	RAPI_PORT_3_7	Port $P3_7$
RAPI_PORT_6_0	Port $P6_0$	RAPI_PORT_6_1	Port $P6_1$
RAPI_PORT_6_2	Port $P6_2$	RAPI_PORT_6_3	Port $P6_3$
RAPI_PORT_6_4	Port $P6_4$	RAPI_PORT_6_5	Port $P6_5$
RAPI_PORT_6_6	Port $P6_6$	RAPI_PORT_6_7	Port $P6_7$
RAPI_PORT_7_0	Port $P7_0$	RAPI_PORT_7_1	Port $P7_1$
RAPI_PORT_7_2	Port $P7_2$	RAPI_PORT_7_3	Port $P7_3$
RAPI_PORT_7_4	Port $P7_4$	RAPI_PORT_7_5	Port $P7_5$
RAPI_PORT_7_6	Port $P7_6$	RAPI_PORT_7_7	Port $P7_7$
RAPI_PORT_8_0	Port $P8_0$	RAPI_PORT_8_1	Port $P8_1$
RAPI_PORT_8_2	Port $P8_2$	RAPI_PORT_8_3	Port $P8_3$
RAPI_PORT_8_4	Port $P8_4$	RAPI_PORT_8_5	Port $P8_5$
RAPI_PORT_8_6	Port $P8_6$	RAPI_PORT_8_7	Port $P8_7$
RAPI_PORT_9_0	Port $P9_0$	RAPI_PORT_9_1	Port $P9_1$
RAPI_PORT_9_2	Port $P9_2$	RAPI_PORT_9_3	Port $P9_3$

RAPI_PORT_9_5	Port $P9_5$	RAPI_PORT_9_6	Port $P9_6$
RAPI_PORT_9_7	Port $P9_7$	RAPI_PORT_10_0	Port $P10_0$
RAPI_PORT_10_1	Port $P10_1$	RAPI_PORT_10_2	Port $P10_2$
RAPI_PORT_10_3	Port $P10_3$	RAPI_PORT_10_4	Port $P10_4$
RAPI_PORT_10_5	Port $P10_5$	RAPI_PORT_10_6	Port $P10_6$
RAPI_PORT_10_7	Port $P10_7$		

(R8C)

RAPI_PORT_0_0	Port $P0_0$	RAPI_PORT_0_1	Port $P0_1$
RAPI_PORT_0_2	Port $P0_2$	RAPI_PORT_0_3	Port $P0_3$
RAPI_PORT_0_4	Port $P0_4$	RAPI_PORT_0_5	Port $P0_5$
RAPI_PORT_0_6	Port $P0_6$	RAPI_PORT_0_7	Port $P0_7$
RAPI_PORT_1_0	Port $P1_0$	RAPI_PORT_1_1	Port $P1_1$
RAPI_PORT_1_2	Port $P1_2$	RAPI_PORT_1_3	Port $P1_3$
RAPI_PORT_1_4	Port $P1_4$	RAPI_PORT_1_5	Port $P1_5$
RAPI_PORT_1_6	Port $P1_6$	RAPI_PORT_1_7	Port $P1_7$
RAPI_PORT_2_0	Port $P2_0$	RAPI_PORT_2_1	Port $P2_1$
RAPI_PORT_2_2	Port $P2_2$	RAPI_PORT_2_3	Port $P2_3$
RAPI_PORT_2_4	Port $P2_4$	RAPI_PORT_2_5	Port $P2_5$
RAPI_PORT_2_6	Port $P2_6$	RAPI_PORT_2_7	Port $P2_7$
RAPI_PORT_3_0	Port $P3_0$	RAPI_PORT_3_1	Port $P3_1$
RAPI_PORT_3_2	Port $P3_2$	RAPI_PORT_3_3	Port $P3_3$
RAPI_PORT_3_4	Port $P3_4$	RAPI_PORT_3_5	Port $P3_5$
RAPI_PORT_3_6	Port $P3_6$	RAPI_PORT_3_7	Port $P3_7$
RAPI_PORT_4_3	Port $P4_3$	RAPI_PORT_4_4	Port $P4_4$
RAPI_PORT_4_5	Port $P4_5$	RAPI_PORT_5_0	Port $P5_0$
RAPI_PORT_5_1	Port $P5_1$	RAPI_PORT_5_2	Port $P5_2$
RAPI_PORT_5_3	Port $P5_3$	RAPI_PORT_5_4	Port $P5_4$
RAPI_PORT_5_5	Port $P5_5$	RAPI_PORT_5_6	Port $P5_6$
RAPI_PORT_5_7	Port $P5_7$	RAPI_PORT_6_0	Port $P6_0$
RAPI_PORT_6_1	Port $P6_1$	RAPI_PORT_6_2	Port $P6_2$
RAPI_PORT_6_3	Port $P6_3$	RAPI_PORT_6_4	Port $P6_4$
RAPI_PORT_6_5	Port $P6_5$	RAPI_PORT_6_6	Port $P6_6$
RAPI_PORT_6_7	Port $P6_7$	RAPI_PORT_7_0	Port $P7_0$
RAPI_PORT_7_1	Port $P7_1$	RAPI_PORT_7_2	Port $P7_2$
RAPI_PORT_7_3	Port $P7_3$	RAPI_PORT_7_4	Port $P7_4$
RAPI_PORT_7_5	Port $P7_5$	RAPI_PORT_7_6	Port $P7_6$
RAPI_PORT_7_7	Port $P7_7$	RAPI_PORT_8_0	Port $P8_0$
RAPI_PORT_8_1	Port $P8_1$	RAPI_PORT_8_2	Port $P8_2$
RAPI_PORT_8_3	Port $P8_3$	RAPI_PORT_8_4	Port $P8_4$
RAPI_PORT_8_5	Port $P8_5$	RAPI_PORT_8_6	Port $P8_6$
RAPI_PORT_8_7	Port $P8_7$	RAPI_PORT_9_0	Port $P9_0$
RAPI_PORT_9_1	Port $P9_1$	RAPI_PORT_9_2	Port $P9_2$
RAPI_PORT_9_3	Port $P9_3$		

(H8S)

RAPI_PORT_1_0	Port P10	RAPI_PORT_1_1	Port P11
RAPI_PORT_1_2	Port P12	RAPI_PORT_1_3	Port P13
RAPI_PORT_1_4	Port P14	RAPI_PORT_1_5	Port P15
RAPI_PORT_1_6	Port P16	RAPI_PORT_1_7	Port P17
RAPI_PORT_2_0	Port P20	RAPI_PORT_2_1	Port P21
RAPI_PORT_2_2	Port P22	RAPI_PORT_2_3	Port P23
RAPI_PORT_2_4	Port P24	RAPI_PORT_2_5	Port P25
RAPI_PORT_2_6	Port P26	RAPI_PORT_2_7	Port P27
RAPI_PORT_3_0	Port P30	RAPI_PORT_3_1	Port P31
RAPI_PORT_3_2	Port P32	RAPI_PORT_3_3	Port P33
RAPI_PORT_3_4	Port P34	RAPI_PORT_3_5	Port P35
RAPI_PORT_3_6	Port P36	RAPI_PORT_3_7	Port P37
RAPI_PORT_5_0	Port P50	RAPI_PORT_5_1	Port P51
RAPI_PORT_5_2	Port P52	RAPI_PORT_5_3	Port P53
RAPI_PORT_5_4	Port P54	RAPI_PORT_5_5	Port P55
RAPI_PORT_5_6	Port P56	RAPI_PORT_5_7	Port P57
RAPI_PORT_6_0	Port P60	RAPI_PORT_6_1	Port P61
RAPI_PORT_6_2	Port P62	RAPI_PORT_6_3	Port P63
RAPI_PORT_6_4	Port P64	RAPI_PORT_6_5	Port P65
RAPI_PORT_6_6	Port P66	RAPI_PORT_6_7	Port P67
RAPI_PORT_8_5	Port P85	RAPI_PORT_8_6	Port P86
RAPI_PORT_8_7	Port P87	RAPI_PORT_9_0	Port P90
RAPI_PORT_9_1	Port P91	RAPI_PORT_9_2	Port P92
RAPI_PORT_9_3	Port P93	RAPI_PORT_9_4	Port P94
RAPI_PORT_9_5	Port P95	RAPI_PORT_9_6	Port P96
RAPI_PORT_9_7	Port P97	RAPI_PORT_A_0	Port PA0
RAPI_PORT_A_1	Port PA1	RAPI_PORT_A_2	Port PA2
RAPI_PORT_A_3	Port PA3	RAPI_PORT_A_4	Port PA4
RAPI_PORT_A_5	Port PA5	RAPI_PORT_A_6	Port PA6
RAPI_PORT_A_7	Port PA7	RAPI_PORT_B_0	Port PB0
RAPI_PORT_B_1	Port PB1	RAPI_PORT_B_2	Port PB2
RAPI_PORT_B_3	Port PB3	RAPI_PORT_B_4	Port PB4
RAPI_PORT_B_5	Port PB5	RAPI_PORT_B_6	Port PB6
RAPI_PORT_B_7	Port PB7	RAPI_PORT_J_0	Port PJ0
RAPI_PORT_J_1	Port PJ1		

(H8/300H)

RAPI_PORT_1_0	Port P10	RAPI_PORT_1_1	Port P11
RAPI_PORT_1_2	Port P12	RAPI_PORT_1_4	Port P14
RAPI_PORT_1_5	Port P15	RAPI_PORT_1_6	Port P16
RAPI_PORT_1_7	Port P17	RAPI_PORT_2_0	Port P20
RAPI_PORT_2_1	Port P21	RAPI_PORT_2_2	Port P22
RAPI_PORT_2_3	Port P23	RAPI_PORT_2_4	Port P24

RAPI_PORT_2_5	PortP25	RAPI_PORT_2_6	PortP26
RAPI_PORT_2_7	PortP27	RAPI_PORT_3_0	PortP30
RAPI_PORT_3_1	PortP31	RAPI_PORT_3_2	PortP32
RAPI_PORT_3_3	PortP33	RAPI_PORT_3_4	PortP34
RAPI_PORT_3_5	PortP35	RAPI_PORT_3_6	PortP36
RAPI_PORT_3_7	PortP37	RAPI_PORT_5_0	PortP50
RAPI_PORT_5_1	PortP51	RAPI_PORT_5_2	PortP52
RAPI_PORT_5_3	PortP53	RAPI_PORT_5_4	PortP54
RAPI_PORT_5_5	PortP55	RAPI_PORT_5_6	PortP56
RAPI_PORT_5_7	PortP57	RAPI_PORT_6_0	PortP60
RAPI_PORT_6_1	PortP61	RAPI_PORT_6_2	PortP62
RAPI_PORT_6_3	PortP63	RAPI_PORT_6_4	PortP64
RAPI_PORT_6_5	PortP65	RAPI_PORT_6_6	PortP66
RAPI_PORT_6_7	PortP67	RAPI_PORT_7_0	PortP70
RAPI_PORT_7_1	PortP71	RAPI_PORT_7_2	PortP72
RAPI_PORT_7_4	PortP74	RAPI_PORT_7_5	PortP75
RAPI_PORT_7_6	PortP76	RAPI_PORT_7_7	PortP77
RAPI_PORT_8_0	PortP80	RAPI_PORT_8_1	PortP81
RAPI_PORT_8_2	PortP82	RAPI_PORT_8_3	PortP83
RAPI_PORT_8_4	PortP84	RAPI_PORT_8_5	PortP85
RAPI_PORT_8_6	PortP86	RAPI_PORT_8_7	PortP87
RAPI_PORT_9_0	PortP90	RAPI_PORT_9_1	PortP91
RAPI_PORT_9_2	PortP92	RAPI_PORT_9_3	PortP93
RAPI_PORT_9_4	PortP94	RAPI_PORT_9_5	PortP95
RAPI_PORT_9_6	PortP96	RAPI_PORT_9_7	PortP97
RAPI_PORT_C_0	PortPC0	RAPI_PORT_C_1	PortPC1
RAPI_PORT_C_2	PortPC2	RAPI_PORT_C_3	PortPC3
RAPI_PORT_D_0	PortPD0	RAPI_PORT_D_1	PortPD1
RAPI_PORT_D_2	PortPD2	RAPI_PORT_D_3	PortPD3
RAPI_PORT_D_4	PortPD4	RAPI_PORT_D_5	PortPD5
RAPI_PORT_D_6	PortPD6	RAPI_PORT_D_7	PortPD7
RAPI_PORT_E_0	PortPE0	RAPI_PORT_E_1	PortPE1
RAPI_PORT_E_2	PortPE2	RAPI_PORT_E_3	PortPE3
RAPI_PORT_E_4	PortPE4	RAPI_PORT_E_5	PortPE5
RAPI_PORT_E_6	PortPE6	RAPI_PORT_E_7	PortPE7
RAPI_PORT_G_0	PortPG0	RAPI_PORT_G_1	PortPG1
RAPI_PORT_G_2	PortPG2	RAPI_PORT_G_3	PortPG3
RAPI_PORT_G_4	PortPG4	RAPI_PORT_G_5	PortPG5
RAPI_PORT_G_6	PortPG6	RAPI_PORT_G_7	PortPG7
RAPI_PORT_H_0	PortPH0	RAPI_PORT_H_1	PortPH1
RAPI_PORT_H_2	PortPH2	RAPI_PORT_H_3	PortPH3
RAPI_PORT_H_4	PortPH4	RAPI_PORT_H_5	PortPH5
RAPI_PORT_H_6	PortPH6	RAPI_PORT_H_7	PortPH7

RAPI_PORT_J_0	PortPJ0	RAPI_PORT_J_1	PortPJ1

Return value

If the I/O port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

I/O port

Reference

__SetIOPort, __ReadIOPort, __SetIOPortRegister, __ReadIOPortRegister, __WriteIOPortRegister

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O ports differ with each CPU used.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
   unsigned int data;

   /* Set the data to port P05 */
   __WriteIOPort( RAPI_PORT_0_5, 0 );
}
```

__SetIOPortRegister

Synopsis

<Set I/O port register>

Boolean __SetIOPortRegister(unsigned long data1, unsigned int data2, unsigned int data3, unsigned int data4)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
data4	Setup data 4 (content differs with MCU type)

Description

[data1]

Set the operating condition of a specified I/O port in each relevant register.

(M16C)

The definition values corresponding to each I/O port register are listed below.

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_10	Port P10 register		

The definition values related to port settings are described below.

RAPI_LATCH	Set to read the port latch regardless of whether the port is set for input or output. Specifiable only when port P1 is used.

(R8C)

The definition values corresponding to each I/O port register are listed below.

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_4	Port P4 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register

(H8S)

The definition values corresponding to each I/O port register are listed below.

RAPI_PORT_1	Port P1 register	RAPI_PORT_2	Port P2 register
RAPI_PORT_3	Port P3 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_8	Port P8 register
RAPI_PORT_9	Port P9 register	RAPI_PORT_A	Port PA register
RAPI_PORT_B	Port PB register	RAPI_PORT_J	Port PJ register

(H8/300H)

The definition values corresponding to each I/O port register are listed below.

RAPI_PORT_1	Port P1 register	RAPI_PORT_2	Port P2 register
RAPI_PORT_3	Port P3 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register

RAPI_PORT_C	Port PC register	RAPI_PORT_D	Port PD register
RAPI_PORT_E	Port PE register	RAPI_PORT_F	Port PF register
RAPI_PORT_G	Port PG register	RAPI_PORT_H	Port PH register
RAPI_PORT_J	Port PJ register		

[data2]

(M16C) (R8C)

Specify the set value for the port direction register corresponding to a selected port.

(H8S) (H8/300H)

Specify the set value for the port control register corresponding to a selected port.

[data3]

(M16C) (R8C)

Specify the set value for the pullup control register corresponding to a selected port.

(H8S)

Specify the set value for the port pullup control register corresponding to a selected port.

(H8/300H)

Specify the set value for the pullup control register corresponding to a selected port.

[data4]

(M16C)

Specify the digital filter width of the digital debounce facility assigned to _NMI/_SD. Specifiable only when port P8 is used.

Specify the digital filter width of the digital debounce facility assigned to INPC17/_INT5. Specifiable only when port P1 is used.

When using any other port, set 0 for this data.

(R8C)

Specify the set value for the port P1 drive capacity register. Specifiable only when port P1 is used.

When using any other port, set 0 for this data.

(H8S)

Specify the set value for the port drive control register corresponding to a selected port.

(H8/300H)

Specify 0.

Return value

If the I/O port register specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

I/O port

Reference

__SetIOPort, __ReadIOPort, __WriteIOPort, __ReadIOPortRegister, __WriteIOPortRegister

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O port registers differ with each CPU used.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
    /* Set inputs/outputs of port P1 register */
    __SetIOPortRegister(RAPI_PORT_1, 0xAA, 0, 0 );
}
```

__ReadIOPortRegister

Synopsis

<Read from I/O port register>

Boolean __ReadIOPortRegister(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the variable in which the value read from I/O port register is stored.

Description

Gets the value of a specified I/O port from each relevant register.

[data1]

Specify an I/O port register from which data is read. The definition values corresponding to each I/O port register are listed below.

(M16C)

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_10	Port P10 register		

(R8C)

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_4	Port P4 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register

(H8S)

RAPI_PORT_1	Port P1 register	RAPI_PORT_2	Port P2 register
RAPI_PORT_3	Port P3 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_8	Port P8 register
RAPI_PORT_9	Port P9 register	RAPI_PORT_A	Port PA register
RAPI_PORT_B	Port PB register	RAPI_PORT_J	Port PJ register

(H8/300H)

RAPI_PORT_1	Port P1 register	RAPI_PORT_2	Port P2 register
RAPI_PORT_3	Port P3 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_B	Port PB register	RAPI_PORT_C	Port PC register
RAPI_PORT_D	Port PD register	RAPI_PORT_E	Port PE register
RAPI_PORT_F	Port PF register	RAPI_PORT_G	Port PG register
RAPI_PORT_H	Port PH register	RAPI_PORT_J	Port PJ register

Return value

If the I/O port register specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality I/O port

Reference __SetIOPort, __ReadIOPort, __WriteIOPort, __SetIOPortRegister, __WriteIOPortRegister

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable I/O port registers differ with each CPU used.
- Ports in port B of the H8/300H that are used as analog input pins cannot be used as input ports.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
    unsigned int data;

    /* Get the value of port P1 register */
    __ReadIOPortRegister( RAPI_PORT_1, &data );
}
```

__WriteIOPortRegister

Synopsis

<Write to I/O port register>

Boolean __WriteIOPortRegister(unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Data to be written to I/O port register

Description

Writes the value for a specified I/O port to each relevant register.

[data1]

Specify an I/O port register to which data is written. The definition values corresponding to each I/O port register are listed below.

(M16C)

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_10	Port P10 register		

(R8C)

RAPI_PORT_0	Port P0 register	RAPI_PORT_1	Port P1 register
RAPI_PORT_2	Port P2 register	RAPI_PORT_3	Port P3 register
RAPI_PORT_4	Port P4 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register

(H8S)

RAPI_PORT_1	Port P1 register	RAPI_PORT_2	Port P2 register
RAPI_PORT_3	Port P3 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_8	Port P8 register
RAPI_PORT_9	Port P9 register	RAPI_PORT_A	Port PA register
RAPI_PORT_B	Port PB register	RAPI_PORT_J	Port PJ register

(H8/300H)

RAPI_PORT_1	Port P1 register	RAPI_PORT_2	Port P2 register
RAPI_PORT_3	Port P3 register	RAPI_PORT_5	Port P5 register
RAPI_PORT_6	Port P6 register	RAPI_PORT_7	Port P7 register
RAPI_PORT_8	Port P8 register	RAPI_PORT_9	Port P9 register
RAPI_PORT_B	Port PB register	RAPI_PORT_C	Port PC register
RAPI_PORT_D	Port PD register	RAPI_PORT_E	Port PE register
RAPI_PORT_F	Port PF register	RAPI_PORT_G	Port PG register
RAPI_PORT_H	Port PH register	RAPI_PORT_J	Port PJ register

Return value

If the I/O port register specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality	I/O port
Reference	__SetIOPort, __ReadIOPort, __WriteIOPort, __SetIOPortRegister, __ReadIOPortRegister
Remark	• If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. • The specifiable I/O port registers differ with each CPU used.

Program example

```
#include " rapi_io_port_r8c_13.h"

void func( void )
{
    /* Set the data to port P1 register */
    __WriteIOPortRegister( RAPI_PORT_1, 0xFF );
}
```

4.2.4 External interrupt
__SetInterrupt

Synopsis

<Set external interrupt>

Boolean __SetInterrupt(unsigned long data1, unsigned int data2, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets a specified external interrupt.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value. Note, however, that multiple external interrupts cannot be specified at the same time.

(M16C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_INT4	Uses _INT4 interrupt.
RAPI_INT5	Uses _INT5 interrupt.
RAPI_KEY	Uses key input interrupt.
RAPI_INT_RISING	Specifies a rising edge for the active edge of a selected external interrupt.
RAPI_INT_FALLING	Specifies a falling edge for the active edge of a selected external interrupt.
RAPI_INT_BOTH	Specifies both edges for the active edge of a selected external interrupt.
RAPI_KI0_ENABLE	Uses _KI0 pin input.
RAPI_KI1_ENABLE	Uses _KI1 pin input.
RAPI_KI2_ENABLE	Uses _KI2 pin input.
RAPI_KI3_ENABLE	Uses _KI3 pin input.

• Specifiable definition values when _INT0-5 interrupts are used (RAPI_INT0 to RAPI_INT5 specified)

(Polarity) Specify one from { RAPI_INT_RISING, RAPI_INT_FALLING, RAPI_INT_BOTH }. The default value is RAPI_INT_FALLING.

• Specifiable definition values when key input interrupt is used (RAPI_KEY specified)

(Input pin) To use _KI0, _KI1, _KI2, or _KI3 pin input, specify RAPI_KI0_ENABLE, RAPI_KI1_ENABLE, RAPI_KI2_ENABLE, or RAPI_KI3_ENABLE, respectively. The default value is RAPI_INT_FALLING.

(R8C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.

RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_KEY	Uses key input interrupt.
RAPI_INT_RISING	Specifies a rising edge for the active edge of a selected external interrupt.
RAPI_INT_FALLING	Specifies a falling edge for the active edge of a selected external interrupt.
RAPI_INT_BOTH	Specifies both edges for the active edge of a selected external interrupt.
RAPI_FILTER_F1	Uses the digital filter facility that has a sampling frequency f_1.
RAPI_FILTER_F8	Uses the digital filter facility that has a sampling frequency f_8.
RAPI_FILTER_F32	Uses the digital filter facility that has a sampling frequency f_{32}.
RAPI_KI0_FALLING	Enables _KI0 pin input whose falling edge is the active edge.
RAPI_KI0_RISING	Enables _KI0 pin input whose rising edge is the active edge.
RAPI_KI1_FALLING	Enables _KI1 pin input whose falling edge is the active edge.
RAPI_KI1_RISING	Enables _KI1 pin input whose rising edge is the active edge.
RAPI_KI2_FALLING	Enables _KI2 pin input whose falling edge is the active edge.
RAPI_KI2_RISING	Enables _KI2 pin input whose rising edge is the active edge.
RAPI_KI3_FALLING	Enables _KI3 pin input whose falling edge is the active edge.
RAPI_KI3_RISING	Enables _KI3 pin input whose rising edge is the active edge.
RAPI_INT1_P1_5	Select P1_5 for INT1 pin.
RAPI_INT1_P1_7	Select P1_7 for INT1 pin.
RAPI_INT1_P3_6	Select P3_6 for INT1 pin.
RAPI_INT2_P6_6	Select P6_6 for INT2 pin.
RAPI_INT2_P3_2	Select P3_2 for INT2 pin.

• Specifiable definition values when _INT0 to _INT3 interrupt is used (RAPI_INT0 to RAPI_INT3 specified)

(Polarity) Specify one from { RAPI_INT_RISING, RAPI_INT_FALLING, RAPI_INT_BOTH }. The default value is RAPI_INT_FALLING.

(Filter) Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F8, RAPI_FILTER_F32 }. If no filters are specified, “No filter” is set.

• Specifiable definition values when key input interrupt is used (RAPI_KEY specified)

(_KI0 pin input) Specify one from { RAPI_KI0_FALLING, RAPI_KI0_RISING }. If _KI0 pin input is not specified, “_KI0 pin input disabled” is set.

(_KI1 pin input) Specify one from { RAPI_KI1_FALLING, RAPI_KI1_RISING }. If _KI1 pin input is not specified, “_KI1 pin input disabled” is set.

(_KI2 pin input) Specify one from { RAPI_KI2_FALLING, RAPI_KI2_RISING }. If _KI2 pin input is not specified, “_KI2 pin input disabled” is set.

(_KI3 pin input) Specify one from { RAPI_KI3_FALLING, RAPI_KI3_RISING }. If _KI3 pin input is not specified, “_KI3 pin input disabled” is set.

(H8S)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.

RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_IRQ4	Uses IRQ4 interrupt.
RAPI_IRQ5	Uses IRQ5 interrupt.
RAPI_IRQ6	Uses IRQ6 interrupt.
RAPI_IRQ7	Uses IRQ7 interrupt.
RAPI_NOT_INT_ REQUEST	Does not request assertion of interrupt for a selected external interrupt.
RAPI_INT_REQUEST	Requests assertion of interrupt for a selected external interrupt.
RAPI_INT_RISING	Specifies a rising edge for the active edge of a selected external interrupt.
RAPI_INT_FALLING	Specifies a falling edge for the active edge of a selected external interrupt.
RAPI_INT_BOTH	Specifies both edges for the active edge of a selected external interrupt.
RAPI_FILTER_F1	Uses noise canceler.
RAPI_FILTER_F2	Uses noise canceler. (Twice)
RAPI_FILTER_F4	Uses noise canceler. (Four times)
RAPI_FILTER_F8	Uses noise canceler. (Eight times)
RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.

• Specifiable definition values when _IRQ0–7 interrupts are used (RAPI_IRQ0 to RAPI_IRQ7 specified)

(Polarity)	Specify one from { RAPI_INT_RISING, RAPI_INT_FALLING, RAPI_INT_BOTH }.
(Interrupt request)	Specify one from { RAPI_NOT_INT_REQUEST, RAPI_INT_REQUEST }. The default value is RAPI_NOT_INT_REQUEST.
(Interrupt control mode)	Specify one from { RAPI_INT_MODE_0, RAPI_INT_MODE_2 }. The default value is RAPI_INT_MODE_0.
(Noise Canceler Control)	Specify one from { RAPI_FILTER_F1, RAPI_FILTER_F2, RAPI_FILTER_F4, RAPI_FILTER_F8 }. The default value is RAPI_FILTER_F1.

(H8/300H)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_WKP	Uses WKP interrupt.
RAPI_NOT_INT_ REQUEST	Does not request assertion of interrupt for a selected external interrupt.
RAPI_INT_REQUEST	Requests assertion of interrupt for a selected external interrupt.
RAPI_INT_RISING	Specifies a rising edge for the active edge of a selected external interrupt.
RAPI_INT_FALLING	Specifies a falling edge for the active edge of a selected external interrupt.

RAPI_WKP0_FALLING	Enables _WKP0 pin input whose falling edge is the active edge.
RAPI_WKP0_RISING	Enables _WKP0 pin input whose rising edge is the active edge.
RAPI_WKP1_FALLING	Enables _WKP1 pin input whose falling edge is the active edge.
RAPI_WKP1_RISING	Enables _WKP1 pin input whose rising edge is the active edge.
RAPI_WKP2_FALLING	Enables _WKP2 pin input whose falling edge is the active edge.
RAPI_WKP2_RISING	Enables _WKP2 pin input whose rising edge is the active edge.
RAPI_WKP3_FALLING	Enables _WKP3 pin input whose falling edge is the active edge.
RAPI_WKP3_RISING	Enables _WKP3 pin input whose rising edge is the active edge.
RAPI_WKP4_FALLING	Enables _WKP4 pin input whose falling edge is the active edge.
RAPI_WKP4_RISING	Enables _WKP4 pin input whose rising edge is the active edge.
RAPI_WKP5_FALLING	Enables _WKP5 pin input whose falling edge is the active edge.
RAPI_WKP5_RISING	Enables _WKP5 pin input whose rising edge is the active edge.

• Specifiable definition values when _IRQ0–3 interrupts are used (RAPI_IRQ0 to RAPI_IRQ3 specified)

(Polarity) Specify one from { RAPI_INT_RISING, RAPI_INT_FALLING }. The default value is RAPI_INT_FALLING.

(Interrupt request) Specify one from { RAPI_NOT_INT_REQUEST, RAPI_INT_REQUEST }. The default value is RAPI_NOT_INT_REQUEST.

• Specifiable definition values when WKP interrupt is used (RAPI_WKP specified)

(_WKP0 pin input) Specify one from { RAPI_WKP0_FALLING, RAPI_WKP0_RISING }. If _WKP0 pin input is not specified, "_WKP0 pin input disabled" is set.

(_WKP1 pin input) Specify one from { RAPI_WKP1_FALLING, RAPI_WKP1_RISING }. If _WKP1 pin input is not specified, "_WKP1 pin input disabled" is set.

(_WKP pin input) Specify one from { RAPI_WKP2_FALLING, RAPI_WKP2_RISING }. If _WKP2 pin input is not specified, "_WKP2 pin input disabled" is set.

(_WKP3 pin input) Specify one from { RAPI_WKP3_FALLING, RAPI_WKP3_RISING }. If _WKP3 pin input is not specified, "_WKP3 pin input disabled" is set.

(_WKP4 pin input) Specify one from { RAPI_WKP4_FALLING, RAPI_WKP4_RISING }. If _WKP4 pin input is not specified, "_WKP4 pin input disabled" is set.

(_WKP5 pin input) Specify one from { RAPI_WKP5_FALLING, RAPI_WKP5_RISING }. If _WKP5 pin input is not specified, "_WKP5 pin input disabled" is set.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8S)

Specify the interrupt priority level (0-3) to be set in the interrupt priority register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

Return value

If the external interrupt specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality	External interrupt
Reference	__EnableInterrupt, __GetInterruptFlag, __ClearInterruptFlag
Remark	• If an undefined value is specified in the first argument, operation of the API cannot be guaranteed. • The specifiable external interrupts differ with each CPU used. • For the R8C, if its CPU type is the one that does not accept specification of “both edges” for interrupts to INT2 and INT3, RAPI_INT_BOTH cannot be specified for INT2 and INT3.

Program example

```
#include " rapi_interrupt_r8c_13.h"

void IntFunc( void ){}

void func( void )
{
    /* Set up _INT0 interrupt */
    __SetInterrupt( RAPI_INT0|RAPI_INT_FALLING, 0, IntFunc );
}
```

__EnableInterrupt

Synopsis

<Control external interrupt>

Boolean __EnableInterrupt(unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Setup data 2 (content differs with MCU type)

Description

Changes the operating condition of a specified external interrupt.

[data1]

The following definition values can be set for data1. To specify multiple definition values at the same time, use the symbol "|" to separate each specified value. Note, however, that multiple external interrupts cannot be specified at the same time.

(M16C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_INT4	Uses _INT4 interrupt.
RAPI_INT5	Uses _INT5 interrupt.
RAPI_KEY	Uses key input interrupt.

(R8C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_KEY	Uses key input interrupt.

(H8S)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_IRQ4	Uses IRQ4 interrupt.
RAPI_IRQ5	Uses IRQ5 interrupt.
RAPI_IRQ6	Uses IRQ6 interrupt.
RAPI_IRQ7	Uses IRQ7 interrupt.
RAPI_NOT_INT_ REQUEST	Does not request assertion of interrupt for a selected external interrupt. (Default)
RAPI_INT_REQUEST	Requests assertion of interrupt for a selected external interrupt.
RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.

(H8/300H)

RAPI_IRQ0	Uses IRQ0 interrupt.

RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_WKP	Uses WKP interrupt.
RAPI_NOT_INT_ REQUEST	Does not request assertion of interrupt for a selected external interrupt. (Default)
RAPI_INT_REQUEST	Requests assertion of interrupt for a selected external interrupt.

[data2]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8S)

Specify the interrupt priority level (0-3) to be set in the interrupt priority register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

Return value

If the external interrupt specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

External interrupt

Reference

__SetInterrupt, __GetInterruptFlag, __ClearInterruptFlag

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable external interrupts differ with each CPU used.

Program example

```
#include " rapi_interrupt_r8c_13.h"

void func( void )
{
    /* Activate _INT1 interrupt ( interrupt priority level 5 ) */
    __EnableInterrupt(RAPI_INT1, 5 );
}
```

__GetInterruptFlag

Synopsis

<Get the status of external interrupt flag>

Boolean __GetInterruptFlag(unsigned long data1, unsigned int *data2))

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which the acquired flag data is stored

Description

Gets the value of interrupt request flag of a specified external interrupt.

[data1]

(M16C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_INT4	Uses _INT4 interrupt.
RAPI_INT5	Uses _INT5 interrupt.
RAPI_KEY	Uses key input interrupt.

(R8C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_KEY	Uses key input interrupt.

(H8S)

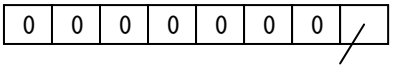
RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_IRQ4	Uses IRQ4 interrupt.
RAPI_IRQ5	Uses IRQ5 interrupt.
RAPI_IRQ6	Uses IRQ6 interrupt.
RAPI_IRQ7	Uses IRQ7 interrupt.

(H8/300H)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_WKP0	Uses WKP0 interrupt.
RAPI_WKP1	Uses WKP1 interrupt.
RAPI_WKP2	Uses WKP2 interrupt.
RAPI_WKP3	Uses WKP3 interrupt.
RAPI_WKP4	Uses WKP4 interrupt.

RAPI_WKP5	Uses WKP5 interrupt.

[data2]



Value of interrupt request flag (0: Interrupt not requested; 1: Interrupt requested)

Return value

If the external interrupt specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

External interrupt

Reference

__SetInterrupt, __EnableInterrupt, __ClearInterruptFlag

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable external interrupts differ with each CPU used.

Program example

```
#include " rapi_interrupt_r8c_13.h"

void func( void )
{
    unsigned char data;

    /* Get flag of _INT2 interrupt */
    __GetInterruptFlag( IS_INT2, &data );
}
```

__ClearInterruptFlag

Synopsis

<Clear external interrupt flag>

Boolean __ClearInterruptFlag(unsigned long data)

data	Setup data (content differs with MCU type)

Description

Clears the interrupt request flag of a specified external interrupt.

[data]

(M16C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_INT4	Uses _INT4 interrupt.
RAPI_INT5	Uses _INT5 interrupt.
RAPI_KEY	Uses key input interrupt.

(R8C)

RAPI_INT0	Uses _INT0 interrupt.
RAPI_INT1	Uses _INT1 interrupt.
RAPI_INT2	Uses _INT2 interrupt.
RAPI_INT3	Uses _INT3 interrupt.
RAPI_KEY	Uses key input interrupt.

(H8S)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_IRQ4	Uses IRQ4 interrupt.
RAPI_IRQ5	Uses IRQ5 interrupt.
RAPI_IRQ6	Uses IRQ6 interrupt.
RAPI_IRQ7	Uses IRQ7 interrupt.

(H8/300H)

RAPI_IRQ0	Uses IRQ0 interrupt.
RAPI_IRQ1	Uses IRQ1 interrupt.
RAPI_IRQ2	Uses IRQ2 interrupt.
RAPI_IRQ3	Uses IRQ3 interrupt.
RAPI_WKP0	Uses WKP0 interrupt.
RAPI_WKP1	Uses WKP1 interrupt.
RAPI_WKP2	Uses WKP2 interrupt.
RAPI_WKP3	Uses WKP3 interrupt.
RAPI_WKP4	Uses WKP4 interrupt.
RAPI_WKP5	Uses WKP5 interrupt.

Return value

If the external interrupt specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

External interrupt

Reference

__SetInterrupt, __EnableInterrupt, __GetInterruptFlag

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable external interrupts differ with each CPU used.

Program example

```
#include " rapi_interrupt_r8c_13.h"

void func( void )
{
    /* Clear status of _INT0 interrupt */
    __ClearInterruptFlag( RAPI_INT0 );
}
```

4.2.5 A/D converter
__CreateADC

Synopsis

<Set A/D converter>

Boolean __CreateADC(unsigned long data1, unsigned int data2, unsigned int data3, void* func)

data1	Setup data 1 (content differs with MCU type)
data2	Number of analog input pins used by A/D converter (differs with MCU type)
data3	Setup data 3 (content differs with MCU type)
func	Callback function pointer (Specify 0 if no callback functions are set.)

Description

Sets the A/D converter to specified mode and operating condition.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value. Note, however, that multiple analog input pin symbols cannot be specified at the same time.

(M16C)

RAPI_ONE_SHOT	Selects one-shot mode.
RAPI_REPEAT	Selects repeat mode.
RAPI_SINGLE_SWEEP	Selects single sweep mode.
RAPI_REPEAT_SWEEP0	Selects repeat sweep mode 0.
RAPI_REPEAT_SWEEP1	Selects repeat sweep mode 1.
RAPI_AN0	Uses AN_0 pin for the analog input pin.
RAPI_AN1	Uses AN_1 pin for the analog input pin.
RAPI_AN2	Uses AN_2 pin for the analog input pin.
RAPI_AN3	Uses AN_3 pin for the analog input pin.
RAPI_AN4	Uses AN_4 pin for the analog input pin.
RAPI_AN5	Uses AN_5 pin for the analog input pin.
RAPI_AN6	Uses AN_6 pin for the analog input pin.
RAPI_AN7	Uses AN_7 pin for the analog input pin.
RAPI_AN00	Uses AN_{00} pin for the analog input pin.
RAPI_AN01	Uses AN_{01} pin for the analog input pin.
RAPI_AN02	Uses AN_{02} pin for the analog input pin.
RAPI_AN03	Uses AN_{03} pin for the analog input pin.
RAPI_AN04	Uses AN_{04} pin for the analog input pin.
RAPI_AN05	Uses AN_{05} pin for the analog input pin.
RAPI_AN06	Uses AN_{06} pin for the analog input pin.
RAPI_AN07	Uses AN_{07} pin for the analog input pin.
RAPI_AN20	Uses AN_{20} pin for the analog input pin.
RAPI_AN21	Uses AN_{21} pin for the analog input pin.
RAPI_AN22	Uses AN_{22} pin for the analog input pin.
RAPI_AN23	Uses AN_{23} pin for the analog input pin.
RAPI_AN24	Uses AN_{24} pin for the analog input pin.
RAPI_AN25	Uses AN_{25} pin for the analog input pin.

RAPI_AN26	Uses AN_{26} pin for the analog input pin.
RAPI_AN27	Uses AN_{27} pin for the analog input pin.
RAPI_AN150	Uses AN_{150} pin for the analog input pin.
RAPI_AN151	Uses AN_{151} pin for the analog input pin.
RAPI_AN152	Uses AN_{152} pin for the analog input pin.
RAPI_AN153	Uses AN_{153} pin for the analog input pin.
RAPI_AN154	Uses AN_{154} pin for the analog input pin.
RAPI_AN155	Uses AN_{155} pin for the analog input pin.
RAPI_AN156	Uses AN_{156} pin for the analog input pin.
RAPI_AN157	Uses AN_{157} pin for the analog input pin.
RAPI_P0_GROUP	Uses port P_0 group for the analog input pin.
RAPI_P2_GROUP	Uses port P_2 group for the analog input pin.
RAPI_P10_GROUP	Uses port P_{10} group for the analog input pin.
RAPI_P15_GROUP	Uses port P_{15} group for the analog input pin.
RAPI_FAD	Sets the AD converter's operating frequency to f_{AD}.
RAPI_FAD2	Sets the AD converter's operating frequency to f_{AD} divided by 2.
RAPI_FAD3	Sets the AD converter's operating frequency to f_{AD} divided by 3.
RAPI_FAD4	Sets the AD converter's operating frequency to f_{AD} divided by 4.
RAPI_FAD6	Sets the AD converter's operating frequency to f_{AD} divided by 6.
RAPI_FAD8	Sets the AD converter's operating frequency to f_{AD} divided by 8.
RAPI_10BIT	Sets the AD converter's resolution to 10 bits.
RAPI_8BIT	Sets the AD converter's resolution to 8 bits.
RAPI_AD_ON	Sets the AD converter to start operating.
RAPI_AD_OFF	Sets the AD converter to stop operating.
RAPI_WITH_SAMPLE_HOLD	Specifies that sample-and-hold action is applied.
RAPI_WITHOUT_SAMPLE_ HOLD	Specifies that sample-and-hold action is not applied.
RAPI_SOFTWARE_TRIGGER	Selects software trigger.
RAPI_EXTERNAL_TRIGGER	Selects external trigger.

• Specifiable definition values when one-shot mode is used (RAPI_ONE_SHOT specified)

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN00, RAPI_AN01, RAPI_AN02, RAPI_AN03, RAPI_AN04, RAPI_AN05, RAPI_AN06, RAPI_AN07, RAPI_AN20, RAPI_AN21, RAPI_AN22, RAPI_AN23, RAPI_AN24, RAPI_AN25, RAPI_AN26, RAPI_AN27, RAPI_AN150, RAPI_AN151, RAPI_AN152, RAPI_AN153, RAPI_AN154, RAPI_AN155, RAPI_AN156, RAPI_AN157 }. The default is RAPI_AN0.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD8 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when repeat mode is used (RAPI_REPEAT specified)

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN00, RAPI_AN01, RAPI_AN02, RAPI_AN03, RAPI_AN04, RAPI_AN05, RAPI_AN06, RAPI_AN07, RAPI_AN20, RAPI_AN21, RAPI_AN22, RAPI_AN23, RAPI_AN24, RAPI_AN25, RAPI_AN26, RAPI_AN27, RAPI_AN150, RAPI_AN151, RAPI_AN152, RAPI_AN153, RAPI_AN154, RAPI_AN155, RAPI_AN156, RAPI_AN157 }. The default is RAPI_AN0.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD8 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when single sweep mode is used (RAPI_SINGLE_SWEEP specified)

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P2_GROUP, RAPI_P10_GROUP, RAPI_P15_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD8 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when repeat sweep mode 0 is used (RAPI_REPEAT_SWEEP0 specified)

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P2_GROUP, RAPI_P10_GROUP, RAPI_P15_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD8 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when repeat sweep mode 1 is used (RAPI_REPEAT_SWEEP1 specified)

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P2_GROUP, RAPI_P10_GROUP, RAPI_P15_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD8 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

(M16C)

RAPI_ONE_SHOT	Selects one-shot mode.
RAPI_REPEAT	Selects repeat mode.
RAPI_SINGLE_SWEEP	Selects single sweep mode.
RAPI_REPEAT_SWEEP0	Selects repeat sweep mode 0.
RAPI_REPEAT_SWEEP1	Selects repeat sweep mode 1.
RAPI_SIMULTANEOUS_ SAMPLE_SWEEP	Selects simultaneous sampling sweep mode.
RAPI_DELAYED_TRIGGER0	Selects delayed trigger mode 0.
RAPI_DELAYED_TRIGGER1	Selects delayed trigger mode 1.
RAPI_AN0	Uses AN_0 pin for the analog input pin.
RAPI_AN1	Uses AN_1 pin for the analog input pin.
RAPI_AN2	Uses AN_2 pin for the analog input pin.
RAPI_AN3	Uses AN_3 pin for the analog input pin.
RAPI_AN4	Uses AN_4 pin for the analog input pin.
RAPI_AN5	Uses AN_5 pin for the analog input pin.
RAPI_AN6	Uses AN_6 pin for the analog input pin.

RAPI_AN7	Uses AN_7 pin for the analog input pin.
RAPI_AN00	Uses AN_{00} pin for the analog input pin.
RAPI_AN01	Uses AN_{01} pin for the analog input pin.
RAPI_AN02	Uses AN_{02} pin for the analog input pin.
RAPI_AN03	Uses AN_{03} pin for the analog input pin.
RAPI_AN04	Uses AN_{04} pin for the analog input pin.
RAPI_AN05	Uses AN_{05} pin for the analog input pin.
RAPI_AN06	Uses AN_{06} pin for the analog input pin.
RAPI_AN07	Uses AN_{07} pin for the analog input pin.
RAPI_AN20	Uses AN_{20} pin for the analog input pin.
RAPI_AN21	Uses AN_{21} pin for the analog input pin.
RAPI_AN22	Uses AN_{22} pin for the analog input pin.
RAPI_AN23	Uses AN_{23} pin for the analog input pin.
RAPI_AN24	Uses AN_{24} pin for the analog input pin.
RAPI_AN25	Uses AN_{25} pin for the analog input pin.
RAPI_AN26	Uses AN_{26} pin for the analog input pin.
RAPI_AN27	Uses AN_{27} pin for the analog input pin.
RAPI_AN30	Uses AN_{30} pin for the analog input pin.
RAPI_AN31	Uses AN_{31} pin for the analog input pin.
RAPI_AN32	Uses AN_{32} pin for the analog input pin.
RAPI_P0_GROUP	Uses port P_0 group for the analog input pin.
RAPI_P10_GROUP	Uses port P_{10} group for the analog input pin.
RAPI_P1P9_GROUP	Uses port P_1/P_9 group for the analog input pin.
RAPI_P9_GROUP	Uses port P_9 group for the analog input pin.
RAPI_FAD	Sets the AD converter's operating frequency to f_{AD}.
RAPI_FAD2	Sets the AD converter's operating frequency to f_{AD} divided by 2.
RAPI_FAD3	Sets the AD converter's operating frequency to f_{AD} divided by 3.
RAPI_FAD4	Sets the AD converter's operating frequency to f_{AD} divided by 4.
RAPI_FAD6	Sets the AD converter's operating frequency to f_{AD} divided by 6.
RAPI_FAD12	Sets the AD converter's operating frequency to f_{AD} divided by 12.
RAPI_10BIT	Sets the AD converter's resolution to 10 bits.
RAPI_8BIT	Sets the AD converter's resolution to 8 bits.
RAPI_AD_ON	Sets the AD converter to start operating.
RAPI_AD_OFF	Sets the AD converter to stop operating.
RAPI_WITH_SAMPLE_HOLD	Specifies that sample-and-hold action is applied.
RAPI_WITHOUT_SAMPLE_ HOLD	Specifies that sample-and-hold action is not applied.
RAPI_SOFTWARE_TRIGGER	Selects software trigger.
RAPI_EXTERNAL_TRIGGER	Selects external trigger.
RAPI_TIMER_B0_TRIGGER	Selects underflow of timer B0 as trigger.
RAPI_TIMER_B2_TRIGGER	Selects underflow of timer B2 as trigger.
RAPI_ICTB2_TRIGGER	Selects underflow of timer B2 interrupt generation frequency set counter as trigger.

• Specifiable definition values when one-shot mode is used (RAPI_ONE_SHOT specified)

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN00, RAPI_AN01, RAPI_AN02, RAPI_AN03, RAPI_AN04, RAPI_AN05, RAPI_AN06, RAPI_AN07, RAPI_AN20, RAPI_AN21, RAPI_AN22, RAPI_AN23, RAPI_AN24, RAPI_AN25, RAPI_AN26, RAPI_AN27, RAPI_AN30, RAPI_AN31, RAPI_AN32 }. The default is RAPI_AN0.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }.
The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when repeat mode is used (RAPI_REPEAT specified)

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN00, RAPI_AN01, RAPI_AN02, RAPI_AN03, RAPI_AN04, RAPI_AN05, RAPI_AN06, RAPI_AN07, RAPI_AN20, RAPI_AN21, RAPI_AN22, RAPI_AN23, RAPI_AN24, RAPI_AN25, RAPI_AN26, RAPI_AN27, RAPI_AN30, RAPI_AN31, RAPI_AN32 }. The default is RAPI_AN0.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }.
The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }.
The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when single sweep mode is used (RAPI_SINGLE_SWEEP specified)

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }.
The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }.
The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when repeat sweep mode 0 is used (RAPI_REPEAT_SWEEP0 specified)

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when repeat sweep mode 1 is used (RAPI_REPEAT_SWEEP1 specified)

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }.
The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER }.
The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when simultaneous sampling sweep mode is used (RAPI_SIMULTANEOUS_SAMPLE_SWEEP specified)

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_EXTERNAL_TRIGGER, RAPI_TIMER_B0_TRIGGER, RAPI_TIMER_B2_TRIGGER, RAPI_ICTB2_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

• Specifiable definition values when delayed trigger mode 0 is used (RAPI_DELAYED_TRIGGER0 specified)

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Conversion method) Specify RAPI_WITH_SAMPLE_HOLD.

• Specifiable definition values when delayed trigger mode 1 is used (RAPI_DELAYED_TRIGGER1 specified)

(Input pin) Specify one from { RAPI_P0_GROUP, RAPI_P10_GROUP, RAPI_P1P9_GROUP, RAPI_P9_GROUP }. The default value is RAPI_P10_GROUP.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD3, RAPI_FAD4, RAPI_FAD6, RAPI_FAD12 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Conversion method) Specify RAPI_WITH_SAMPLE_HOLD.

(R8C)

RAPI_ONE_SHOT	Selects one-shot mode.
RAPI_REPEAT	Selects repeat mode.
RAPI_SINGLE_SWEEP	Selects single sweep mode.
RAPI_REPEAT_SWEEP	Selects repeat sweep mode 0.
RAPI_AN0	Uses AN0 pin for the analog input pin.
RAPI_AN1	Uses AN1 pin for the analog input pin.
RAPI_AN2	Uses AN2 pin for the analog input pin.
RAPI_AN3	Uses AN3 pin for the analog input pin.
RAPI_AN4	Uses AN4 pin for the analog input pin.
RAPI_AN5	Uses AN5 pin for the analog input pin.
RAPI_AN6	Uses AN6 pin for the analog input pin.
RAPI_AN7	Uses AN7 pin for the analog input pin.
RAPI_AN8	Uses AN8 pin for the analog input pin.
RAPI_AN9	Uses AN9 pin for the analog input pin.
RAPI_AN10	Uses AN10 pin for the analog input pin.
RAPI_AN11	Uses AN11 pin for the analog input pin.

RAPI_AN12	Uses AN12 pin for the analog input pin.
RAPI_AN13	Uses AN13 pin for the analog input pin.
RAPI_AN14	Uses AN14 pin for the analog input pin.
RAPI_AN15	Uses AN15 pin for the analog input pin.
RAPI_AN16	Uses AN16 pin for the analog input pin.
RAPI_AN17	Uses AN17 pin for the analog input pin.
RAPI_AN18	Uses AN18 pin for the analog input pin.
RAPI_AN19	Uses AN19 pin for the analog input pin.
RAPI_AN12_GROUP	Uses AN 12 group for the analog input pin.
RAPI_AN16_GROUP	Uses AN 16 group for the analog input pin.
RAPI_FAD	Sets the AD converter's operating frequency to f_{AD}.
RAPI_FAD2	Sets the AD converter's operating frequency to f_{AD} divided by 2.
RAPI_FAD4	Sets the AD converter's operating frequency to f_{AD} divided by 4.
RAPI_FOCOF	Sets the AD converter's operating frequency to fOCO-F.
RAPI_10BIT	Sets the AD converter's resolution to 10 bits.
RAPI_8BIT	Sets the AD converter's resolution to 8 bits.
RAPI_AD_ON	Sets the AD converter to start operating in __CreateAD.
RAPI_AD_OFF	Sets the AD converter to stop operating in __CreateAD
RAPI_WITH_SAMPLE_HOLD	Specifies that sample-and-hold action is applied.
RAPI_WITHOUT_SAMPLE_HOLD	Specifies that sample-and-hold action is not applied.
RAPI_SOFTWARE_TRIGGER	Selects software trigger.
RAPI_TIMER_RD_TRIGGER	Selects timer RD as trigger.

• Specifiable definition values when one-shot mode is used (RAPI_ONE_SHOT specified)

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11, RAPI_AN12, RAPI_AN13, RAPI_AN14, RAPI_AN15, RAPI_AN16, RAPI_AN17, RAPI_AN18, RAPI_AN19 }. Always be sure to specify an input pin.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD4, RAPI_FOCOF }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }. The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_TIMER_RD_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.

For the R8C/13, 26, 27, 28 and 29, this API cannot be used.

• Specifiable definition values when repeat mode is used (RAPI_REPEAT specified)

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11, RAPI_AN12, RAPI_AN13, RAPI_AN14, RAPI_AN15, RAPI_AN16, RAPI_AN17, RAPI_AN18, RAPI_AN19}. Always be sure to specify an input pin.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD4 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }.
The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_TIMER_RD_TRIGGER }. The default value is RAPI_SOFTWARE_TRIGGER.
For the R8C/13, 26, 27, 28 and 29, this API cannot be used.

• Specifiable definition values when single sweep mode is used (RAPI_SINGLE_SWEEP specified)

(Input pin) Specify one from { RAPI_AN12_GROUP, RAPI_AN16_GROUP }.
Always be sure to specify this Input pin.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD4 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_TIMER_RD_TRIGGER }.
The default value is RAPI_SOFTWARE_TRIGGER.
For the R8C/13, 26, 27, 28 and 29, this API cannot be used.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }.
The default value is RAPI_WITHOUT_SAMPLE_HOLD.

• Specifiable definition values when repeat sweep mode 0 is used (RAPI_REPEAT_SWEEP specified)

(Input pin) Specify one from { RAPI_AN12_GROUP, RAPI_AN16_GROUP }.
Always be sure to specify this Input pin.

(Operating frequency) Specify one from { RAPI_FAD, RAPI_FAD2, RAPI_FAD4 }. The default value is RAPI_FAD4.

(Resolution) Specify one from { RAPI_10BIT, RAPI_8BIT }. The default value is RAPI_8BIT.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Conversion method) Specify one from { RAPI_WITH_SAMPLE_HOLD, RAPI_WITHOUT_SAMPLE_HOLD }.
The default value is RAPI_WITHOUT_SAMPLE_HOLD.

(Trigger) Specify one from { RAPI_SOFTWARE_TRIGGER, RAPI_TIMER_RD_TRIGGER }.
The default value is RAPI_SOFTWARE_TRIGGER.
For the R8C/13, 26, 27, 28 and 29, this API cannot be used.

(H8S)

RAPI_ONE_SHOT	Selects one-shot mode.
RAPI_REPEAT	Selects repeat mode.
RAPI_SINGLE_SWEEP	Selects single sweep mode.
RAPI_REPEAT_SWEEP0	Selects repeat sweep mode 0.
RAPI_AD_UNIT_1	Uses A/D converter unit 1.
RAPI_AD_UNIT_2	Uses A/D converter unit 2.
RAPI_AN0	Uses AN0 pin for the analog input pin.
RAPI_AN1	Uses AN1 pin for the analog input pin.
RAPI_AN2	Uses AN2 pin for the analog input pin.
RAPI_AN3	Uses AN3 pin for the analog input pin.
RAPI_AN4	Uses AN4 pin for the analog input pin.
RAPI_AN5	Uses AN5 pin for the analog input pin.
RAPI_AN6	Uses AN6 pin for the analog input pin.
RAPI_AN7	Uses AN7 pin for the analog input pin.
RAPI_AN8	Uses AN8 pin for the analog input pin.
RAPI_AN9	Uses AN9 pin for the analog input pin.
RAPI_AN10	Uses AN10 pin for the analog input pin.
RAPI_AN11	Uses AN11 pin for the analog input pin.
RAPI_AN0_GROUP	Uses AN0~AN3 group for the analog input pin.
RAPI_AN4_GROUP	Uses AN4~AN7group for the analog input pin.
RAPI_AN8_GROUP	Uses AN8~AN11 group for the analog input pin.
RAPI_84_STATES	Conversion time = 84 states
RAPI_43_STATES	Conversion time = 43 states
RAPI_AD_ON	Sets the AD converter to start operating in __CreateAD.
RAPI_AD_OFF	Sets the AD converter to stop operating in __CreateAD.
RAPI_ADTRG1_TRIGGER	Includes an external trigger input on ADTRG1 in the A/D conversion start condition.
RAPI_ADTRG2_TRIGGER	Includes an external trigger input on ADTRG2 in the A/D conversion start condition.
RAPI_TIMER_RC_TRIGGER	Includes an external trigger from timer RC in the A/D conversion start condition.
RAPI_TIMER_RD_0_TRIGGER	Includes an external trigger from timer RD unit 0 in the A/D conversion start condition.
RAPI_TIMER_RD_1_TRIGGER	Includes an external trigger from timer RD unit 1 in the A/D conversion start condition.
RAPI_AD_INT_REQUEST	Requests assertion of interrupt for A/D conversion interrupts.

RAPI_NOT_AD_INT_ REQUEST	Does not request assertion of interrupt for A/D conversion interrupts.

• Specifiable definition values when one-shot mode is used (RAPI_ONE_SHOT specified)

(unit) Specify one from { RAPI_AD_UNIT_1, RAPI_AD_UNIT_2 }. The default value is RAPI_AD_UNIT_1.

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11 }. The default value is RAPI_AN0.

(Conversion time) Specify one from { RAPI_84_STATES, RAPI_43_STATES }.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_ADTRG1_TRIGGER, RAPI_ADTRG2_TRIGGER, RAPI_TIMER_RC_TRIGGER, RAPI_TIMER_RD_0_TRIGGER, RAPI_TIMER_RD_1_TRIGGER }. If no triggers are specified, “Trigger input disabled” is set.

(Interrupt request) Specify one from { RAPI_AD_INT_REQUEST, RAPI_NOT_AD_INT_REQUEST }.
The default value is RAPI_NOT_AD_INT_REQUEST.

• Specifiable definition values when repeat mode is used (RAPI_REPEAT specified)

(unit) Specify one from { RAPI_AD_UNIT_1, RAPI_AD_UNIT_2 }. The default value is RAPI_AD_UNIT_1.

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11 }. The default value is RAPI_AN0.

(Conversion time) Specify one from { RAPI_84_STATES, RAPI_43_STATES }.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_ADTRG1_TRIGGER, RAPI_ADTRG2_TRIGGER, RAPI_TIMER_RC_TRIGGER, RAPI_TIMER_RD_0_TRIGGER, RAPI_TIMER_RD_1_TRIGGER }. If no triggers are specified, “Trigger input disabled” is set.

• Specifiable definition values when single sweep mode is used (RAPI_SINGLE_SWEEP specified)

(unit) Specify one from { RAPI_AD_UNIT_1, RAPI_AD_UNIT_2 }. The default value is RAPI_AD_UNIT_1.

(Input pin) Specify one from { RAPI_AN0_GROUP, RAPI_AN4_GROUP, RAPI_AN8_GROUP }. The default value is RAPI_AN0_GROUP.

(Conversion time) Specify one from { RAPI_84_STATES, RAPI_43_STATES }.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_ADTRG1_TRIGGER, RAPI_ADTRG2_TRIGGER, RAPI_TIMER_RC_TRIGGER, RAPI_TIMER_RD_0_TRIGGER, RAPI_TIMER_RD_1_TRIGGER }. If no triggers are specified, “Trigger input disabled” is set.

(Interrupt request) Specify one from { RAPI_AD_INT_REQUEST, RAPI_NOT_AD_INT_REQUEST }.
The default value is RAPI_NOT_AD_INT_REQUEST.

• Specifiable definition values when repeat sweep mode 0 is used (RAPI_REPEAT_SWEEP0 specified)

(unit) Specify one from { RAPI_AD_UNIT_1, RAPI_AD_UNIT_2 }. The default value is RAPI_AD_UNIT_1.

(Input pin) Specify one from { RAPI_AN0_GROUP, RAPI_AN4_GROUP, RAPI_AN8_GROUP }. The default value is RAPI_AN0_GROUP.

(Conversion time) Specify one from { RAPI_84_STATES, RAPI_43_STATES }.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_ADTRG1_TRIGGER, RAPI_ADTRG2_TRIGGER, RAPI_TIMER_RC_TRIGGER, RAPI_TIMER_RD_0_TRIGGER, RAPI_TIMER_RD_1_TRIGGER }. If no triggers are specified, “Trigger input disabled” is set.

(Interrupt request) Specify one from { RAPI_AD_INT_REQUEST, RAPI_NOT_AD_INT_REQUEST }.
The default value is RAPI_NOT_AD_INT_REQUEST.

(H8/300H)

RAPI_ONE_SHOT	Selects one-shot mode.
RAPI_REPEAT	Selects repeat mode.
RAPI_SINGLE_SWEEP	Selects single sweep mode.
RAPI_REPEAT_SWEEP0	Selects repeat sweep mode 0.
RAPI_AN0	Uses AN0 pin for the analog input pin.
RAPI_AN1	Uses AN1 pin for the analog input pin.
RAPI_AN2	Uses AN2 pin for the analog input pin.
RAPI_AN3	Uses AN3 pin for the analog input pin.
RAPI_AN4	Uses AN4 pin for the analog input pin.
RAPI_AN5	Uses AN5 pin for the analog input pin.
RAPI_AN6	Uses AN6 pin for the analog input pin.
RAPI_AN7	Uses AN7 pin for the analog input pin.
RAPI_AN8	Uses AN8 pin for the analog input pin.
RAPI_AN9	Uses AN9 pin for the analog input pin.
RAPI_AN10	Uses AN10 pin for the analog input pin.
RAPI_AN11	Uses AN11 pin for the analog input pin.
RAPI_AN12	Uses AN12 pin for the analog input pin.
RAPI_AN13	Uses AN13 pin for the analog input pin.
RAPI_AN14	Uses AN14 pin for the analog input pin.

RAPI_AN15	Uses AN15 pin for the analog input pin.
RAPI_AN0_GROUP	Uses AN0-AN3 group for the analog input pin.
RAPI_AN4_GROUP	Uses AN4-AN7 group for the analog input pin.
RAPI_AN8_GROUP	Uses AN8-AN11 group for the analog input pin.
RAPI_AN12_GROUP	Uses AN12-AN15 group for the analog input pin.
RAPI_134_STATES	Conversion time = 134 states
RAPI_70_STATES	Conversion time = 70 states
RAPI_AD_ON	Sets the AD converter to start operating in __CreateAD.
RAPI_AD_OFF	Sets the AD converter to stop operating in __CreateAD.
RAPI_RISING_TRIGGER	Includes a rising-edge external trigger in the A/D conversion start condition.
RAPI_FALLING_TRIGGER	Includes a falling-edge external trigger in the A/D conversion start condition.
RAPI_AD_INT_REQUEST	Requests assertion of interrupt for A/D conversion interrupt.
RAPI_NOT_AD_INT_ REQUEST	Does not request assertion of interrupt for A/D conversion interrupt.

• Specifiable definition values when one-shot mode is used (RAPI_ONE_SHOT specified)

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11, RAPI_AN12, RAPI_AN13, RAPI_AN14, RAPI_AN15 }. The default value is RAPI_AN0.

(Conversion time) Specify one from { RAPI_134_STATES, RAPI_70_STATES }. The default value is RAPI_134_STATES.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_RISING_TRIGGER, RAPI_FALLING_TRIGGER }. If no triggers are specified, “Trigger input disabled” is set.

(Interrupt request) Specify one from { RAPI_AD_INT_REQUEST, RAPI_NOT_AD_INT_REQUEST }. The default value is RAPI_NOT_AD_INT_REQUEST.

• Specifiable definition values when repeat mode is used (RAPI_REPEAT specified)

(Input pin) Specify one from { RAPI_AN0, RAPI_AN1, RAPI_AN2, RAPI_AN3, RAPI_AN4, RAPI_AN5, RAPI_AN6, RAPI_AN7, RAPI_AN8, RAPI_AN9, RAPI_AN10, RAPI_AN11, RAPI_AN12, RAPI_AN13, RAPI_AN14, RAPI_AN15 } The default value is RAPI_AN0.

(Conversion time) Specify one from { RAPI_134_STATES, RAPI_70_STATES }. The default value is RAPI_134_STATES.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_RISING_TRIGGER, RAPI_FALLING_TRIGGER }. If no triggers are specified, “Trigger input disabled” is set.

• Specifiable definition values when single sweep mode is used (RAPI_SINGLE_SWEEP specified)

(Input pin) Specify one from { RAPI_AN0_GROUP, RAPI_AN4_GROUP, RAPI_AN8_GROUP, RAPI_AN12_GROUP }.
The default value is RAPI_AN0_GROUP.

(Conversion time) Specify one from { RAPI_134_STATES, RAPI_70_STATES }. The default value is RAPI_134_STATES.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_RISING_TRIGGER, RAPI_FALLING_TRIGGER }. If no triggers are specified, “Trigger input disabled” is set.

(Interrupt request) Specify one from { RAPI_AD_INT_REQUEST, RAPI_NOT_AD_INT_REQUEST }.
The default value is RAPI_NOT_AD_INT_REQUEST.

• Specifiable definition values when repeat sweep mode 0 is used (RAPI_REPEAT_SWEEP0 specified)

(Input pin) Specify one from { RAPI_AN0_GROUP, RAPI_AN4_GROUP, RAPI_AN8_GROUP, RAPI_AN12_GROUP }. The default value is RAPI_AN0_GROUP.

(Conversion time) Specify one from { RAPI_134_STATES, RAPI_70_STATES }. The default value is RAPI_134_STATES.

(Operating states set) Specify one from { RAPI_AD_ON, RAPI_AD_OFF }. The default value is RAPI_AD_OFF.

(Trigger) Specify one from { RAPI_RISING_TRIGGER, RAPI_FALLING_TRIGGER }. If no triggers are specified, “Trigger input disabled” is set.

[data2]

(M16C)

The set value differs with the A/D conversion mode used.

One-shot mode Repeat mode	Specify 1.
Single sweep mode Repeat sweep mode 0 Simultaneous sampling mode Delayed trigger mode 0 Delayed trigger mode 1	Specify 2, 4, 6, or 8.
Repeat sweep mode	Specify 1, 2, 3, or 4.

(R8C)

One-shot mode Repeat mode	Specify 1.
Single sweep mode Repeat sweep mode 0	Specify 2 or 4.

(H8S)

One-shot mode Repeat mode	Specify 1.
Single sweep mode Repeat sweep mode 0	Specify 1, 2, 3, 4, 5, 6, 7, or 8.

(H8/300H)

One-shot mode Repeat mode	Specify 1.
Single sweep mode Repeat sweep mode 0	Specify 1, 2, 3, or 4.

[data3]

(M16C) (R8C)

Specify the interrupt priority level (0-7) to be set in the interrupt control register.

(H8S)

Specify the interrupt priority level (0-3) to be set in the interrupt priority register.

(H8/300H)

Specify the interrupt priority level (0–1) to be set in the interrupt control register. For the CPUs that do not have an interrupt control register, specify 0.

Return value

If A/D converter was successfully set, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__EnableADC, __DestroyADC, __DestroyADCUnit, __GetADC, __GetADCAll, __GetADCUnit

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable analog input pins differ with each CPU used.
- If you use an R8C MCU which does not support fOCO-F for its operating frequency, you cannot specify RAPI_FOCOF.
- If you use an R8C MCU which does not support timer RD as a trigger, you cannot specify RAPI_TIMER_RD_TRIGGER as a trigger.
- When used for the H8S, H8/300H, this API specify when freeing it from module stanby.

Program example

```
#include " rapi_ad_r8c_13.h"

void AdIntFunc( void ){}

void func( void )
{
   /* Set up A/D converter as one short mode */
   __CreateADC( RAPI_ONE_SHOT|RAPI_AN2|RAPI_FAD2| RAPI_WITH_SAMPLE_HOLD
|
               RAPI_EXTERNAL_TRIGGER   |RAPI_AD_ON|RAPI_10BIT,   1,   5,
AdIntFunc );
}
```

__EnableADC

Synopsis

<Control operation of A/D converter>

Boolean __EnableADC (unsigned long data1, unsigned int data2)

data1	Setup data 1 (content differs with MCU type)
data2	Number of analog input pins used by A/D converter (differs with MCU type)

Description

Controls operation of the A/D converter by starting or stopping it.

[data1]

For data1, the following definition values can be set. To set multiple definition values at the same time, use the symbol "|" to separate each specified value. Note, however, that multiple analog input pin cannot be specified at the same time.

(M16C)

RAPI_AN0	Uses AN_0 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN1	Uses AN_1 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN2	Uses AN_2 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN3	Uses AN_3 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN4	Uses AN_4pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN5	Uses AN_5pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN6	Uses AN_6 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN7	Uses AN_7 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN00	Uses AN_{00} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN01	Uses AN_{01} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN02	Uses AN_{02} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN03	Uses AN_{03} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN04	Uses AN_{04} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN05	Uses AN_{05}pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN06	Uses AN_{06} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN07	Uses AN_{07} pin for the analog input pin.

	Selectable only when one-shot mode or repeat mode is used.
RAPI_AN20	Uses AN_{20} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN21	Uses AN_{21} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN22	Uses AN_{22} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN23	Uses AN_{23} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN24	Uses AN_{24} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN25	Uses AN_{25} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN26	Uses AN_{26} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN27	Uses AN_{27} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN30	Uses AN_{30} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN31	Uses AN_{31} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN32	Uses AN_{32} pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_P0_GROUP	Uses port P_0 group for the analog input pin. Selectable only when sweep mode, repeat sweep mode 0, simultaneous sampling sweep mode, delayed trigger mode 0, delayed trigger mode 1, or repeat sweep mode 1 is used.
RAPI_P10_GROUP	Uses port P_{10} group for the analog input pin. Selectable only when sweep mode, repeat sweep mode 0, simultaneous sampling sweep mode, delayed trigger mode 0, delayed trigger mode 1, or repeat sweep mode 1 is used.
RAPI_P1P9_GROUP	Uses port P_1 /P_9 group for the analog input pin. Selectable only when sweep mode, repeat sweep mode 0, simultaneous sampling sweep mode, delayed trigger mode 0, delayed trigger mode 1, or repeat sweep mode 1 is used.
RAPI_P9_GROUP	Uses port P_9 group for the analog input pin. Selectable only when sweep mode, repeat sweep mode 0, simultaneous sampling sweep mode, delayed trigger mode 0, delayed trigger mode 1, or repeat sweep mode 1 is used.
RAPI_AD_ON	Sets the A/D converter to start operating.
RAPI_AD_OFF	Sets the A/D converter to stop operating.

(R8C)

RAPI_AN0	Uses AN_0 pin for the analog input pin.
RAPI_AN1	Uses AN_1 pin for the analog input pin.

RAPI_AN2	Uses AN_2 pin for the analog input pin.
RAPI_AN3	Uses AN_3 pin for the analog input pin.
RAPI_AN4	Uses AN_4 pin for the analog input pin.
RAPI_AN5	Uses AN_5 pin for the analog input pin.
RAPI_AN6	Uses AN_6 pin for the analog input pin.
RAPI_AN7	Uses AN_7 pin for the analog input pin.
RAPI_AN8	Uses AN_8 pin for the analog input pin.
RAPI_AN9	Uses AN_9 pin for the analog input pin.
RAPI_AN10	Uses AN_{10} pin for the analog input pin.
RAPI_AN11	Uses AN_{11} pin for the analog input pin.
RAPI_AN12	Uses AN_{12} pin for the analog input pin.
RAPI_AN13	Uses AN_{13} pin for the analog input pin.
RAPI_AN14	Uses AN_{14} pin for the analog input pin.
RAPI_AN15	Uses AN_{15} pin for the analog input pin.
RAPI_AN16	Uses AN_{16} pin for the analog input pin.
RAPI_AN17	Uses AN_{17} pin for the analog input pin.
RAPI_AN18	Uses AN_{18} pin for the analog input pin.
RAPI_AN19	Uses AN_{19} pin for the analog input pin.
RAPI_AN12_GROUP	Uses AN_{12} group for the analog input pin.
RAPI_AN16_GROUP	Uses AN_{16} group for the analog input pin.
RAPI_AD_ON	Sets the A/D converter to start operating.
RAPI_AD_OFF	Sets the A/D converter to stop operating.

(H8S)

RAPI_AD_UNIT_1	Uses A/D converter unit 1.
RAPI_AD_UNIT_2	Uses A/D converter unit 2.
RAPI_AN0	Uses AN0 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN1	Uses AN1 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN2	Uses AN2 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN3	Uses AN3 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN4	Uses AN4 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN5	Uses AN5 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN6	Uses AN6 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN7	Uses AN7 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN8	Uses AN8 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.

RAPI_AN9	Uses AN9 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN10	Uses AN10 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN11	Uses AN11 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN0_GROUP	Uses AN0~AN3 group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0, is used.
RAPI_AN4_GROUP	Uses AN4~AN7 group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0, is used.
RAPI_AN8_GROUP	Uses AN8~AN11 group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0, is used.
RAPI_AD_ON	Sets the A/D converter to start operating.
RAPI_AD_OFF	Sets the A/D converter to stop operating.

(H8/300H)

RAPI_AN0	Uses AN0 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN1	Uses AN1 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN2	Uses AN2 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN3	Uses AN3 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN4	Uses AN4 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN5	Uses AN5 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN6	Uses AN6 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN7	Uses AN7 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN8	Uses AN8 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN9	Uses AN9 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN10	Uses AN10 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN11	Uses AN11 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN12	Uses AN12 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN13	Uses AN13 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN14	Uses AN14 pin for the analog input pin.

	Selectable only when one-shot mode or repeat mode is used.
RAPI_AN15	Uses AN15 pin for the analog input pin. Selectable only when one-shot mode or repeat mode is used.
RAPI_AN0_GROUP	Uses AN0-AN3 pins group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0 is used.
RAPI_AN4_GROUP	Uses AN4-AN7 pins group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0 is used.
RAPI_AN8_GROUP	Uses AN8-AN11 pins group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0 is used.
RAPI_AN12_GROUP	Uses AN12-AN15 pins group for the analog input pin. Selectable only when sweep mode or repeat sweep mode 0 is used.
RAPI_AD_ON	Sets the A/D converter to start operating.
RAPI_AD_OFF	Sets the A/D converter to stop operating.

[data2]

(M16C)

The set value differs with the A/D conversion mode used.

One-shot mode Repeat mode	Specify 1.
Single sweep mode Repeat sweep mode 0 Simultaneous sampling mode Delayed trigger mode 0 Delayed trigger mode 1	Specify 2, 4, 6, or 8.
Repeat sweep mode 1	Specify 1, 2, 3, or 4.

(R8C)

The set value differs with the A/D conversion mode used.

One-shot mode Repeat mode	Specify 1
Single sweep mode Repeat sweep mode 0	Specify 2 or 4

(H8S)

The set value differs with the A/D conversion mode used.

One-shot mode Repeat mode	Specify 1.
Single sweep mode Repeat sweep mode 0	Specify 1, 2, 3, 4, 5, 6, 7 or 8.

(H8/300H)

The set value differs with the A/D conversion mode used.

One-shot mode Repeat mode	Specify 1.
Single sweep mode Repeat sweep mode 0	Specify 1, 2, 3, or 4.

Return value

If A/D converter was successfully controlled, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__CreateADC, __DestroyADC, __DestroyADCUnit, __GetADC, __GetADCAll, __GetADCUnit

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- The specifiable analog input pins differ with each CPU used.

Program example

```
#include "rapi_ad_r8c_13.h"

void func( void )
{
    /* Disable A/D converter */
    __EnableADC( RAPI_AN0|RAPI_AD_OFF, 1 );
}
```

__DestroyADC

Synopsis

<Discard settings of A/D converter>

Boolean __DestroyADC(void)

Description

Discards settings of a specified A/D converter.

Return value

If converter setting was successfully discarded, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__CreateADC, __EnableADC, __DestroyADCUnit, __GetADC, __GetADCAll, __GetADCUnit

Remark

- When used for the H8S, H8/300H, this API places a specified timer into module stanby state after discarding it.
- If the MCU has two or more A/D converter units, setting of the unit with the first number will be destroyed.

Program example

```
#include "rapi_ad_r8c_13.h"

void func( void )
{
    /* Destroy A/D converter */
    __DestroyADC();
}
```

__DestroyADCUnit

Synopsis

<Discard settings of A/D converter (specified unit) >

Boolean __DestroyADCUnit(unsigned long unit)

unit	A/D converter unit that you wish to discard settings

Description

Discards settings of a specified A/D converter unit.

[unit]

For unit, the following definition values can be set.

When 0 is specified for a MCU that has two or more A/D converter units, setting of the unit with the first number will be acquired.

(M16C)(R8C)(H8S)(H8/300H)

Specify 0.

(H8S)

RAPI_AD_UNIT_1	Selects A/D converter unit 1.
RAPI_AD_UNIT_2	Selects A/D converter unit 2.

Return value

If converter setting was successfully discarded, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__CreateADC, __EnableADC, __DestroyADC, __GetADC, __GetADCAll, __GetADCUnit

Remark

- When an H8S or H8/300H CPU is in use, the CPU enters the module-standby mode after settings on the A/D converter have been discarded.

Program example

```
#include "rapi_ad_h8s_20223.h"

void func( void )
{
    /* Destroy A/D converter */
    __DestroyADCUnit( RAPI_AD_UNIT_1 );
}
```

__GetADC

Synopsis

<Get A/D converted value (register specified)>

Boolean __GetADC(unsigned long data1, unsigned int *data2)

data1	Setup data 1 (content differs with MCU type)
data2	Pointer to the buffer in which A/D converted value is stored.

Description

Gets the A/D converted value from a specified A/D register.

For data1, the following values can be set.

(M16C)

RAPI_AD0	Selects A/D register 0.
RAPI_AD1	Selects A/D register 1.
RAPI_AD2	Selects A/D register 2.
RAPI_AD3	Selects A/D register 3.
RAPI_AD4	Selects A/D register 4.
RAPI_AD5	Selects A/D register 5.
RAPI_AD6	Selects A/D register 6.
RAPI_AD7	Selects A/D register 7.

(R8C)

Specify the following values for R8C MCU that has two or more A/D data registers.

RAPI_ADDR0	Select A/D data register 0
RAPI_ADDR1	Select A/D data register 1
RAPI_ADDR2	Select A/D data register 2
RAPI_ADDR3	Select A/D data register 3

Other than above: Specify 0

(H8S)

When you specify two or more values at the same time, separate them with “|”.

Note, however, that it is not possible to specify values for multiple analog input pins at the same time.

RAPI_AD_UNIT_1	Selects A/D converter unit 1.
RAPI_AD_UNIT_2	Selects A/D converter unit 2.
RAPI_ADDR0	Selects A/D data register 0.
RAPI_ADDR1	Selects A/D data register 1.
RAPI_ADDR2	Selects A/D data register 2.
RAPI_ADDR3	Selects A/D data register 3.
RAPI_ADDR4	Selects A/D data register 4.
RAPI_ADDR5	Selects A/D data register 5.
RAPI_ADDR6	Selects A/D data register 6.
RAPI_ADDR7	Selects A/D data register 7.

(H8/300H)

RAPI_ADDRA	Selects A/D data register A.
RAPI_ADDRB	Selects A/D data register B.

RAPI_ADDRC	Selects A/D data register C.
RAPI_ADDRD	Selects A/D data register D.

Return value

If A/D converted value was successfully acquired, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__CreateADC, __EnableADC, __DestroyADC, __DestroyADCUnit, __GetADCAll, __GetADCUnit

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.

Program example

```
#include "rapi_ad_r8c_13.h"

void func( void )
{
   unsigned int data;

   /* Get an A/D convertered data of A/D register 0 */
   __GetADC( RAPI_AD0, &data );
}
```

__GetADCAll

Synopsis

<Get A/D converted value (all registers)>

Boolean __GetADCAll(unsigned int *data)

data	Pointer to the buffer in which A/D converted value is stored.

Description

Gets the A/D converted value from all A/D registers.

The A/D registers from which A/D converted values are acquired are listed below.

[M16C] (16 bytes)	:	[0] A/D register 0 [1] A/D register 1 [2] A/D register 2 [3] A/D register 3 [4] A/D register 4 [5] A/D register 5 [6] A/D register 6 [7] A/D register 7
[R8C] <When using the MCU that has two or more data registers> (8 bytes) < other than listed above > (2 bytes)	: :	[0] A/D register 0 [1] A/D register 1 [2] A/D register 2 [3] A/D register 3 A/D register
[H8S] <unit 1> (16 bytes)	:	[0] A/D register 0 [1] A/D register 1 [2] A/D register 2 [3] A/D register 3 [4] A/D register 4 [5] A/D register 5 [6] A/D register 6 [7] A/D register 7
[H8/300H] (8 bytes)	:	[0] A/D register A [1] A/D register B [2] A/D register C [3] A/D register D

Return value

If A/D converted values were successfully acquired, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__CreateADC, __EnableADC, __DestroyADC, __DestroyADCUnit, __GetADC,

__GetADCUnit

Remark

- If the MCU has two or more A/D converter units, A/D conversion values of the unit with the first number will be acquired.

Program example

```
#include "rapi_ad_r8c_13.h"

void func( void )
{
    unsigned int data[8];

    /* Get A/D convertered datas of A/D register */
    __GetADCAll( data );
}
```

__GetADCUnit

Synopsis

<Get A/D converted value (specified unit)>

Boolean __GetADCUnit(unsigned long unit, unsigned int *data)

unit	A/D converter unit to which a conversion values will be acquired.
data	Pointer to the buffer in which A/D converted value is stored.

Description

Gets the A/D conversion value from all A/D registers in specified A/D converter unit.

[unit]

For unit, the following definition values can be set.

When 0 is specified for a MCU that has two or more A/D converter units, A/D conversion value from all A/D registers of the unit with the first number will be acquired.

(M16C)(R8C)(H8S)(H8/300H)

Specify 0.

(H8S)

RAPI_AD_UNIT_1	Selects A/D converter unit 1.
RAPI_AD_UNIT_2	Selects A/D converter unit 2.

[data]

The A/D registers from which A/D converted values are acquired are listed below.

[M16C] (16 bytes)	:	[0] A/D register 0 [1] A/D register 1 [2] A/D register 2 [3] A/D register 3 [4] A/D register 4 [5] A/D register 5 [6] A/D register 6 [7] A/D register 7
[R8C] <When using the MCU that has two or more data registers> (8 bytes)	:	[0] A/D register 0 [1] A/D register 1 [2] A/D register 2 [3] A/D register 3
< other than listed above > (2 bytes)	:	A/D register
[H8S] <unit 1> (16 bytes)	:	[0] A/D register 0 [1] A/D register 1 [2] A/D register 2 [3] A/D register 3 [4] A/D register 4 [5] A/D register 5 [6] A/D register 6

		[7] A/D register 7
<unit 2>	:	[0] A/D register 0
(8 bytes)		[1] A/D register 1
		[2] A/D register 2
		[3] A/D register 3
[H8/300H]	:	[0] A/D register A
(8 bytes)		[1] A/D register B
		[2] A/D register C
		[3] A/D register D

Return value

If A/D converted values were successfully acquired, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__CreateADC, __EnableADC, __DestroyADC, __DestroyADCUnit, __GetADC, __GetADCAll

Program example

```
#include "rapi_ad_h8s_20223.h"

void func( void )
{
    unsigned int data;

    /* Get A/D convertered datas of A/D register */
    __GetADCUnit( RAPI_AD_UNIT_1, &data );
}
```

__GetADCStatus

Synopsis

<Get the status of A/D converter>

Boolean __GetADCStatus(unsigned int *status)

status	Pointer to the buffer in which the register content indicating A/D converter status is stored.

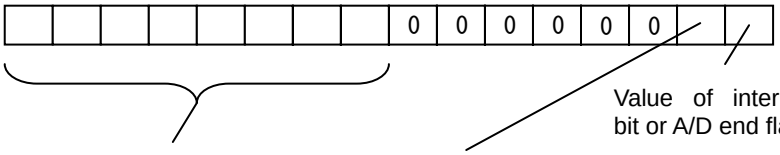
Description

Gets the status of a specified A/D converter.

The status of interrupt bit (when using the M16C or R8C) or the value of A/D end flag (when using the H8S, H8/300H) is stored in the first low-order bit of *status. Furthermore, the status of A/D conversion start flag is stored in the second low-order bit of *status.

When used in the M16C, the value of A/D conversion status register 0 is stored in the 8 high-order bits of *status.

[Configuration of *status]



Value of interrupt bit or A/D end flag

Value of A/D conversion status register 0 (for only the M16C; 0 for the R8C, H8S and H8/300H)

Value of A/D conversion start flag (for only the R8C; 0 for the M16C, H8S and H8/300H)

Return value

If A/D converter status was successfully acquired, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__GetADCUnitStatus, __ClearADCStatus, __ClearADCUnitStatus

Remark

- When there are two or more A/D converter units, information on the status of the unit with the latest number will be acquired.

Program example

```
#include " rapi_ad_r8c_13.h"

void func( void )
{
    unsigned int status;

    /* Get status of A/D convertered */
    __GetADCStatus( &status );
}
```

__GetADCUnitStatus

Synopsis

<Get the status of A/D converter (specified unit) >

Boolean __GetADCUnitStatus(unsigned long unit, unsigned int *status)

unit	A/D converter unit to which information on the status will be acquired.
data	Pointer to the buffer in which A/D converted value is stored.

Description

Gets information on the status of the specified A/D converter unit.

[unit]

For [unit], the values listed below are specifiable.

When 0 is specified for a CPU that has two or more A/D converter units, information on the status of the unit with the latest number will be acquired.

(M16C)(R8C)(H8S)(H8/300H)

Specify 0.

(H8S)

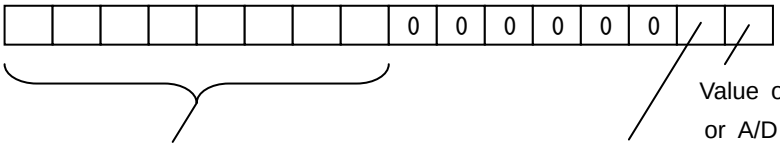
RAPI_AD_UNIT_1	Selects A/D converter unit 1.
RAPI_AD_UNIT_2	Selects A/D converter unit 2.

[status]

The status of interrupt bit (when using the M16C or R8C) or the value of A/D end flag (when using the H8S, H8/300H) is stored in the first low-order bit of *status. Furthermore, the status of A/D conversion start flag is stored in the second low-order bit of *status.

When used in the M16C, the value of A/D conversion status register 0 is stored in the 8 high-order bits of *status.

[Configuration of *status]



Value of A/D conversion status register 0 (for only the M16C; 0 for the R8C, H8S and H8/300H)

Value of A/D conversion start flag (for only the R8C; 0 for the M16C, H8S and H8/300H)

Return value

If A/D converter status was successfully acquired, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__GetADCStatus, __ClearADCStatus, __ClearADCUnitStatus

Remark

- When there are two or more A/D converter units, information on the status of the unit with the latest number will be acquired.

Program example

```
#include " rapi_ad_h8s_20223.h"

void func( void )
{
   unsigned int status;

   /* Get status of A/D convertered */
   __GetADCUnitStatus( RAPI_AD_UNIT_1, &status );
}
```

__ClearADCStatus

Synopsis

<Clear the status of A/D converter>

Boolean __ClearADCStatus(unsigned int status)

status	Status of A/D converter

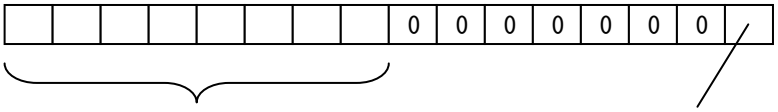
Description

Clears the status flag of a specified A/D converter.

Specify the status of interrupt bit (when using the M16C or R8C) or the value of A/D end flag (when using the H8S or H8/300H) in the first low-order bit of status.

When used in the M16C, specify the value of A/D conversion status register 0 in the 8 high-order bits of status. Write 0 to the bits to be cleared and 1 to the bits that do not need to be cleared.

[Configuration of status]



Value of A/D conversion status register 0 (for only the M16C; 0 for the R8C, H8S and H8/300H)

Value of interrupt bit or A/D end flag

Return value

If A/D converter status flag was successfully cleared, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__GetADCStatus, __GetADCUnitStatus, __ClearADCUnitStatus

Remark

- When there are two or more A/D converter units, information on the status of the unit with the latest number will be acquired.

Program example

```
#include "rapi_ad_r8c_13.h"

void func( void )
{
    /* Clear status of A/D convertered */
    __ClearADCStatus( 0 );
}
```

__ClearADCUnitStatus

Synopsis

<Clear the status of A/D converter (specified unit) >

Boolean __ClearADCUnitStatus(unsigned long unit, unsigned int status)

unit	A/D converter unit to which a status flag will be cleared.
status	Status of A/D converter

Description

Clears the status flag of specified A/D converter unit.

[unit]

For [unit], the values listed below are specifiable.

When 0 is specified for a CPU that has two or more A/D converter units, information on the status of the unit with the latest number will be cleared.

For [unit], the values listed below are specifiable.

(M16C)(R8C)(H8S)(H8/300H)

Specify 0.

(H8S)

RAPI_AD_UNIT_1	Selects A/D converter unit 1.
RAPI_AD_UNIT_2	Selects A/D converter unit 2.

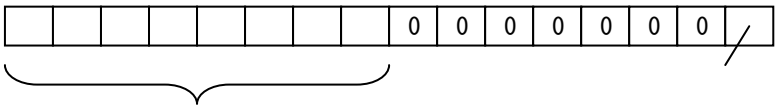
[status]

Specify the status of interrupt bit (when using the M16C or R8C) or the value of A/D end flag (when using the H8S or H8/300H) in the first low-order bit of status.

When used in the M16C, specify the value of A/D conversion status register 0 in the 8 high-order bits of status.

Write 0 to the bits to be cleared and 1 to the bits that do not need to be cleared.

[Configuration of status]



Value of A/D conversion status register 0 (for only the M16C; 0 for the R8C, H8S and H8/300H)

Value of interrupt bit or A/D end flag

Return value

If A/D converter status flag was successfully cleared, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

A/D converter

Reference

__GetADCStatus, __GetADCUnitStatus, __ClearADCStatus

Remark

- If the MCU has two or more A/D converter units, status flags of the unit with the first number will be acquired.

Program example

```
#include "rapi_ad_h8s_20223.h"

void func( void )
{
    /* Clear status of A/D convertered */
    __ClearADCUnitStatus( RAPI_AD_UNIT_1, 0 );
}
```

4.2.6 Peripheral I/O Mapping Controller

__ConfigurePMC

Synopsis

< Assign pin functions >

Boolean __ConfigurePMC(unsigned long port, unsigned int *function)

port	I/O port of the pin to which a function will be assigned
function	Pointer to an array in which the value of [port] will be stored

Description

Assigns a peripheral function to the specified pin.

[port]

Specify the I/O port of the pin to which you wish to assign a function. The following values are specifiable.

(H8S)

RAPI_GR1_PORT_1	Selects Port Group 1 / Port 1.
RAPI_GR1_PORT_2	Selects Port Group 1 / Port 2.
RAPI_GR1_PORT_3	Selects Port Group 1 / Port 3.
RAPI_GR1_PORT_5	Selects Port Group 1 / Port 5.
RAPI_GR1_PORT_6	Selects Port Group 1 / Port 6.
RAPI_GR2_PORT_8	Selects Port Group 2 / Port 8.
RAPI_GR2_PORT_9	Selects Port Group 2 / Port 9.
RAPI_GR2_PORT_A	Selects Port Group 2 / Port A.

[function]

For [function], specify the pointer to an array in which the value of [port] will be stored. The number of elements in [function] is the same as the number of bits in the port register specified for the first parameter [port]. The element number of [function] is the same as the port number of the port register specified for the first parameter [port]. The values listed below are specifiable as [function]. To use the pin as an I/O port pin, specify 0.

(H8S)

RAPI_GR1_IRQ0	Selects Port Group 1 / _IRQ0 input.
RAPI_GR1_IRQ1	Selects Port Group 1 / _IRQ1 input.
RAPI_GR1_IRQ2	Selects Port Group 1 / _IRQ2 input.
RAPI_GR1_IRQ3	Selects Port Group 1 / _IRQ3 input.
RAPI_GR1_IRQ4	Selects Port Group 1 / _IRQ4 input.
RAPI_GR1_IRQ5	Selects Port Group 1 / _IRQ5 input.
RAPI_GR1_IRQ6	Selects Port Group 1 / _IRQ6 input.
RAPI_GR1_IRQ7	Selects Port Group 1 / _IRQ7 input.
RAPI_GR1_ADTRG1	Selects Port Group 1 / _ADTRG1 input.
RAPI_GR1_ADTRG2	Selects Port Group 1 / _ADTRG2 input.
RAPI_GR1_SCK3_1	Selects Port Group 1 / SCK3 input/output.

RAPI_GR1_RXD_1	Selects Port Group 1 / RXD input.
RAPI_GR1_TXD_1	Selects Port Group 1 / TXD output.
RAPI_GR1_SCK3_2	Selects Port Group 1 / SCK3_2 input/output.
RAPI_GR1_RXD_2	Selects Port Group 1 / RXD_2 input.
RAPI_GR1_TXD_2	Selects Port Group 1 / TXD_2 output.
RAPI_GR1_SCK3_3	Selects Port Group 1 / SCK3_3 input/output.
RAPI_GR1_RXD_3	Selects Port Group 1 / RXD_3 input.
RAPI_GR1_TXD_3	Selects Port Group 1 / TXD_3 output.
RAPI_GR1_SSO	Selects Port Group 1 / SSO input/output.
RAPI_GR1_SSCK	Selects Port Group 1 / SSCK input/output.
RAPI_GR1_SCS_SDA	Selects Port Group 1 / SCS/SDA input/output.
RAPI_GR1_SSI_SCL	Selects Port Group 1 / SSI/SCL input/output.
RAPI_GR1_TRAIO	Selects Port Group 1 / TRAIO input/output.
RAPI_GR1_TRAO	Selects Port Group 1 / TRAO output.
RAPI_GR1_TRGB	Selects Port Group 1 / TRGB input.
RAPI_GR1_TRBO	Selects Port Group 1 / TRBO output.
RAPI_GR1_TRCOI	Selects Port Group 1 / _TRCOI input.
RAPI_GR1_FTIOA	Selects Port Group 1 / FTIOA input/output.
RAPI_GR1_FTIOB	Selects Port Group 1 / FTIOB input/output.
RAPI_GR1_FTIOC	Selects Port Group 1 / FTIOC input/output.
RAPI_GR1_FTIOD	Selects Port Group 1 / FTIOD input/output.
RAPI_GR1_FTCI	Selects Port Group 1 / FTCI input.
RAPI_GR1_TRDOI_0	Selects Port Group 1 / _TRDOI_0 input.
RAPI_GR1_FTIOA0	Selects Port Group 1 / FTIOA0 input/output.
RAPI_GR1_FTIOB0	Selects Port Group 1 / FTIOB0 input/output.
RAPI_GR1_FTIOC0	Selects Port Group 1 / FTIOC0 input/output.
RAPI_GR1_FTIOD0	Selects Port Group 1 / FTIOD0 input/output.
RAPI_GR1_FTIOA1	Selects Port Group 1 / FTIOA1 input/output.
RAPI_GR1_FTIOB1	Selects Port Group 1 / FTIOB1 input/output.
RAPI_GR1_FTIOC1	Selects Port Group 1 / FTIOC1 input/output.
RAPI_GR1_FTIOD1	Selects Port Group 1 / FTIOD1 input/output.
RAPI_GR1_TRDOI_1	Selects Port Group 1 / _TRDOI_1 input.
RAPI_GR1_TREO	Selects Port Group 1 / TREO output.
RAPI_GR1_TCLKA	Selects Port Group 1 / TCLKA input.
RAPI_GR1_TCLKB	Selects Port Group 1 / TCLKB input.
RAPI_GR1_TGIOA	Selects Port Group 1 / TGIOA input/output.
RAPI_GR1_TGIOB	Selects Port Group 1 / TGIOB input/output.
RAPI_GR2_IRQ0	Selects Port Group 2 / _IRQ0 input.
RAPI_GR2_IRQ1	Selects Port Group 2 / _IRQ1 input.
RAPI_GR2_IRQ2	Selects Port Group 2 / _IRQ2 input.
RAPI_GR2_IRQ3	Selects Port Group 2 / _IRQ3 input.
RAPI_GR2_IRQ4	Selects Port Group 2 / _IRQ4 input.
RAPI_GR2_IRQ5	Selects Port Group 2 / _IRQ5 input.

RAPI_GR2_IRQ6	Selects Port Group 2 / _IRQ6 input.
RAPI_GR2_IRQ7	Selects Port Group 2 / _IRQ7 input.
RAPI_GR2_FTIOA2	Selects Port Group 2 / FTIOA2 input/output.
RAPI_GR2_FTIOB2	Selects Port Group 2 / FTIOB2 input/output.
RAPI_GR2_FTIOC2	Selects Port Group 2 / FTIOC2 input/output.
RAPI_GR2_FTIOD2	Selects Port Group 2 / FTIOD2 input/output.
RAPI_GR2_FTIOA3	Selects Port Group 2 / FTIOA3 input/output.
RAPI_GR2_FTIOB3	Selects Port Group 2 / FTIOB3 input/output.
RAPI_GR2_FTIOC3	Selects Port Group 2 / FTIOC3 input/output.
RAPI_GR2_FTIOD3	Selects Port Group 2 / FTIOD3 input/output.
RAPI_GR2_SCK3_1	Selects Port Group 2 / SCK3 input/output.
RAPI_GR2_RXD_1	Selects Port Group 2 / RXD input.
RAPI_GR2_TXD_1	Selects Port Group 2 / TXD output.
RAPI_GR2_SCK3_3	Selects Port Group 2 / SCK3_3 input/output.
RAPI_GR2_RXD_3	Selects Port Group 2 / RXD_3 input.
RAPI_GR2_TXD_3	Selects Port Group 2 / TXD_3 output.
RAPI_GR2_TRAIO	Selects Port Group 2 / TRAIO input/output.
RAPI_GR2_TRAO	Selects Port Group 2 / TRAO output.
RAPI_GR2_TRGB	Selects Port Group 2 / TRGB input.
RAPI_GR2_TRBO	Selects Port Group 2 / TRBO output.
RAPI_GR2_TREO	Selects Port Group 2 / TREO output.

Return value

If the I/O port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Peripheral I/O Mapping Controller

Remark

- The correct operation is not guaranteed if any value other than those listed above is specified as the first or second parameter.
- Specifiable I/O ports and functions vary with the CPU.
- This API function is not available with CPUs that do not have a peripheral I/O mapping controller.
- The values specified as the first and second parameters must belong to the same group.

Program example

```
#include " rapi_pmc_h8s_20223.h"
```

```
void func( void )
{
   unsigned int function[8];

   function[0] = RAPI_GR1_IRQ0;
   function[1] = RAPI_GR1_IRQ1;
   function[2] = RAPI_GR1_IRQ2;
   function[3] = RAPI_GR1_IRQ3;
   function[4] = RAPI_GR1_IRQ4;
   function[5] = RAPI_GR1_IRQ5;
   function[6] = RAPI_GR1_IRQ6;
   function[7] = RAPI_GR1_IRQ7;

   /* Set function of port 3 */
   __ConfigurePMC( RAPI_GR1_PORT_3, function );
}
```

4.2.7 Event Link Controller

__SetEventLink

Synopsis

< Assign event signals to be linked to peripheral modules >

Boolean __SetEventLink(unsigned long data1, unsigned long data2, void* func)

data1	Peripheral module, event signal to be linked, and interrupt control mode
data2	Operating conditions for the peripheral module specified for [data1]
func	Pointer to a callback function (specify 0 if there are no interrupts for the peripheral module specified for [data1] or no callback functions are needed)

Description

Assigns event signals to be linked to peripheral modules and sets operating conditions at the time of an input of event signals. Two or more values must be separated by "|".

[data1]

(H8S)

Specify an event signal to be linked to each periheral module.

The following values indicate the corresponding peripheral modules.

RAPI_ELC_TIMER_RA	Selects Timer RA.
RAPI_ELC_TIMER_RB	Selects Timer RB.
RAPI_ELC_TIMER_RC	Selects Timer RC.
RAPI_ELC_TIMER_RD0	Selects Timer RD_0 channel 0.
RAPI_ELC_TIMER_RD1	Selects Timer RD_0 channel 1.
RAPI_ELC_TIMER_RG	Selects Timer RG.
RAPI_ELC_ADC1	Selects AD converter unit 1.
RAPI_ELC_ADC2	Selects AD converter unit 2.
RAPI_ELC_INT1	Selects Interrupts 1.
RAPI_ELC_OUT_PORT_G2	Selects Output port-group 1.
RAPI_ELC_OUT_PORT_G3	Selects Output port-group 2.
RAPI_ELC_IN_PORT_G2	Selects Input port-group 1.
RAPI_ELC_IN_PORT_G3	Selects Input port-group 2.
RAPI_ELC_SINGLE_PORT_1	Selects Single-port 1.
RAPI_ELC_SINGLE_PORT_2	Selects Single-port 2.
RAPI_ELC_SINGLE_PORT_3	Selects Single-port 3.
RAPI_ELC_SINGLE_PORT_4	Selects Single-port 4.
RAPI_ELC_CLOCK	Selects Clock oscillator.
RAPI_ELC_INT2	Selects Interrupts 2.
RAPI_ELC_DAC0	Selects DA converter channel 0.
RAPI_ELC_DAC1	Selects DA converter channel 1.

The following values indicate the corresponding event signals.

RAPI_EV_TIMER_RA_UDF	Selects Timer RA underflow.

RAPI_EV_TIMER_RB_UDF	Selects Timer RB underflow.
RAPI_EV_TIMER_RC_OVF	Selects Timer RC overflow.
RAPI_EV_TIMER_RC_CMA	Selects Timer RC compare-match A.
RAPI_EV_TIMER_RC_CMB	Selects Timer RC compare-match B.
RAPI_EV_TIMER_RC_CMC	Selects Timer RC compare-match C.
RAPI_EV_TIMER_RC_CMD	Selects Timer RC compare-match D.
RAPI_EV_TIMER_RD0_OVF	Selects Timer RD_0 channel 0 overflow.
RAPI_EV_TIMER_RD0_CMA	Selects Timer RD_0 channel 0 compare-match A.
RAPI_EV_TIMER_RD0_CMB	Selects Timer RD_0 channel 0 compare-match B.
RAPI_EV_TIMER_RD0_CMC	Selects Timer RD_0 channel 0 compare-match C.
RAPI_EV_TIMER_RD0_CMD	Selects Timer RD_0 channel 0 compare-match D.
RAPI_EV_TIMER_RD1_OVF	Selects Timer RD_0 channel 1 overflow.
RAPI_EV_TIMER_RD1_UDF	Selects Timer RD_0 channel 1 underflow.
RAPI_EV_TIMER_RD1_CMA	Selects Timer RD_0 channel 1 compare-match A.
RAPI_EV_TIMER_RD1_CMB	Selects Timer RD_0 channel 1 compare-match B.
RAPI_EV_TIMER_RD1_CMC	Selects Timer RD_0 channel 1 compare-match C.
RAPI_EV_TIMER_RD1_CMD	Selects Timer RD_0 channel 1 compare-match D.
RAPI_EV_TIMER_RG_OVF	Selects Timer RG overflow.
RAPI_EV_TIMER_RG_UDF	Selects Timer RG underflow.
RAPI_EV_TIMER_RG_CMA	Selects Timer RG compare-match A.
RAPI_EV_TIMER_RG_CMB	Selects Timer RG compare-match B.
RAPI_EV_AD1	Selects AD conversion end in AD converter unit 1.
RAPI_EV_AD2	Selects AD conversion end in AD converter unit 2.
RAPI_EV_IN_PORT_G1	Selects Input edge detection on input port-group 1.
RAPI_EV_IN_PORT_G2	Selects Input edge detection on input port-group 2.
RAPI_EV_SINGLE_PORT_1	Selects Input edge detection on single input port 1.
RAPI_EV_SINGLE_PORT_2	Selects Input edge detection on single input port 2.
RAPI_EV_SINGLE_PORT_3	Selects Input edge detection on single input port 3.
RAPI_EV_SINGLE_PORT_4	Selects Input edge detection on single input port 4.
RAPI_EV_LVD	Selects Voltage-drop detection in LVD.
RAPI_EV_LVD_RESET	Selects Voltage-drop reset detection in LVD.
RAPI_EV_CPG	Selects CPG backup start.
RAPI_EV_WDT	Selects WDT increment.
RAPI_EV_WDT_RESET	Selects WDT reset.
RAPI_EV_TIMER_RE	Selects Timer RE interval (week, day, hour, minute, or second).
RAPI_EV_DTC	Selects DTC transfer end.
RAPI_EV_IIC2_SSU_T_EMPTY	Selects Transmit-buffer empty in IIC2/SSU.
RAPI_EV_IIC2_SSU_T_END	Selects Transmit end in IIC2/SSU.
RAPI_EV_IIC2_SSU_R_FULL	Selects Receive-buffer full in IIC2/SSU.
RAPI_EV_IIC2_SSU_STOP	Selects Stop-condition detection in IIC2/SSU.
RAPI_EV_IIC2_SSU_ARB_OVE	Selects Arbitration loss/overrun error in IIC2/SSU.
RAPI_EV_IIC2_SSU_NACK_CNF	Selects NACK detection/conflict error in IIC2/SSU.

RAPI_EV_SCI3_1_T_EMPTY	Selects SCI3_1 transmit-buffer empty.
RAPI_EV_SCI3_1_T_END	Selects SCI3_1 transmit end.
RAPI_EV_SCI3_1_R_FULL	Selects SCI3_1 receive-buffer full.
RAPI_EV_SCI3_1_ERR	Selects SCI3_1 transfer error.
RAPI_EV_SCI3_2_T_EMPTY	Selects SCI3_2 transmit-buffer empty.
RAPI_EV_SCI3_2_T_END	Selects SCI3_2 transmit end.
RAPI_EV_SCI3_2_R_FULL	Selects SCI3_2 receive-buffer full.
RAPI_EV_SCI3_2_ERR	Selects SCI3_2 transfer error.
RAPI_EV_SCI3_3_T_EMPTY	Selects SCI3_3 transmit-buffer empty.
RAPI_EV_SCI3_3_T_END	Selects SCI3_3 transmit end.
RAPI_EV_SCI3_3_R_FULL	Selects SCI3_3 receive-buffer full.
RAPI_EV_SCI3_3_ERR	Selects SCI3_3 transfer error.
RAPI_EV_TIMER_ELC_0	Selects Timer ELC event 0.
RAPI_EV_TIMER_ELC_1	Selects Timer ELC event 1.
RAPI_EV_TIMER_ELC_2	Selects Timer ELC event 2.
RAPI_EV_TIMER_ELC_3	Selects Timer ELC event 3.

The following values indicate the corresponding interrupt control modes.

RAPI_INT_MODE_0	Selects interrupt control mode 0.
RAPI_INT_MODE_2	Selects interrupt control mode 2.

The following values indicate whether linkage of all events should be enabled or disabled.

RAPI_ELC_DISABLE	Linkage of all the events are disabled.
RAPI_ELC_ENABLE	Linkage of all the events are enabled.

The following values indicate whether there are event link interrupt requests.

RAPI_NOT_INT_REQUEST	Does not request assertion of interrupt for a selected ELC interrupt.
RAPI_INT_REQUEST	Request assertion of interrupt for a selected ELC interrupt.

[data2]

Specify operating conditions for the peripheral modules specified for the first parameter [data1] at the time of an input of event signals.

For [data 2], specify the interrupt priority level (0 to 3) to be set to the interrupt priority register, or 0.

Two or more values must be separated by “|”.

(H8S)

RAPI_OP_TIMER_RA_ON	Timer RA operation / Timer starts counting.
RAPI_OP_TIMER_RA_EV	Timer RA operation / Timer counts events.
RAPI_OP_TIMER_RA_INVALID	Timer RA operation / Events disabled.
RAPI_OP_TIMER_RB_ON	Timer RB operation / Timer starts counting.
RAPI_OP_TIMER_RB_EV	Timer RB operation / Timer counts events.
RAPI_OP_TIMER_RB_INVALID	Timer RB operation / Events disabled.

RAPI_OP_TIMER_RC_ON	Timer RC operation / Timer starts counting.
RAPI_OP_TIMER_RC_EV	Timer RC operation / Timer counts events.
RAPI_OP_TIMER_RC_IC	Timer RC operation / Timer performs input-capture operation.
RAPI_OP_TIMER_RC_INVALID	Timer RC operation / Events disabled.
RAPI_OP_TIMER_RD0_ON	Timer RD_0 channel 0 operation / Timer starts counting.
RAPI_OP_TIMER_RD0_EV	Timer RD_0 channel 0 operation / Timer counts events.
RAPI_OP_TIMER_RD0_IC	Timer RD_0 channel 0 operation / Timer performs input-capture operation.
RAPI_OP_TIMER_RD0_INVALID	Timer RD_0 channel 0 operation / Events disabled.
RAPI_OP_TIMER_RD1_ON	Timer RD_0 channel 1 operation / Timer starts counting.
RAPI_OP_TIMER_RD1_EV	Timer RD_0 channel 1 operation / Timer counts events.
RAPI_OP_TIMER_RD1_IC	Timer RD_0 channel 1 operation / Timer performs input-capture operation.
RAPI_OP_TIMER_RD1_INVALID	Timer RD_0 channel 1 operation / Events disabled.
RAPI_OP_TIMER_RG_ON	Timer RG operation / Timer starts counting.
RAPI_OP_TIMER_RG_EV	Timer RG operation / Timer counts events.
RAPI_OP_TIMER_RG_IC	Timer RG operation / Timer performs input-capture operation.
RAPI_OP_TIMER_RG_INVALID	Timer RG operation / Events disabled.
RAPI_GR_PORT3_0	Specifies P30 as the member of the same group.
RAPI_GR_PORT3_1	Specifies P31 as the member of the same group.
RAPI_GR_PORT3_2	Specifies P32 as the member of the same group.
RAPI_GR_PORT3_3	Specifies P33 as the member of the same group.
RAPI_GR_PORT3_4	Specifies P34 as the member of the same group.
RAPI_GR_PORT3_5	Specifies P35 as the member of the same group.
RAPI_GR_PORT3_6	Specifies P36 as the member of the same group.
RAPI_GR_PORT3_7	Specifies P37 as the member of the same group.
RAPI_GR_PORT6_0	Specifies P60 as the member of the same group.
RAPI_GR_PORT6_1	Specifies P61 as the member of the same group.
RAPI_GR_PORT6_2	Specifies P62 as the member of the same group.
RAPI_GR_PORT6_3	Specifies P63 as the member of the same group.
RAPI_GR_PORT6_4	Specifies P64 as the member of the same group.
RAPI_GR_PORT6_5	Specifies P65 as the member of the same group.
RAPI_GR_PORT6_6	Specifies P66 as the member of the same group.
RAPI_GR_PORT6_7	Specifies P67 as the member of the same group.
RAPI_PGR3_OUT_0	Output port P3 group.0 is output when an event signal is input.
RAPI_PGR3_OUT_1	Output port 3 group operation / 1 is output when the event is input.
RAPI_PGR3_OUT_TOGGLE	Output port 3 group operation / The toggled (inverted) value is output when the event is input.

RAPI_PGR3_OUT_BUFFER	Output port 3 group operation / The buffer value is output when the event is input.
RAPI_PGR3_OUT_BIT_SHIFT	Output port 3 group operation / The bit value is sifted out in the group (from MSB to LSB) when the event is input.
RAPI_PGR6_OUT_0	Output port 6 group operation / 0 is output when the event is input.
RAPI_PGR6_OUT_1	Output port 6 group operation / 1 is output when the event is input.
RAPI_PGR6_OUT_TOGGLE	Output port 6 group operation / The toggled (inverted) value is output when the event is input.
RAPI_PGR6_OUT_BUFFER	Output port 6 group operation / The buffer value is output when the event is input.
RAPI_PGR6_OUT_BIT_SHIFT	Output port 6 group operation / The bit value is sifted out in the group (from MSB to LSB) when the event is input.
RAPI_PGR3_PDBF_OVER	Input port 3 group operation / Overwriting PDBF is enabled.
RAPI_PGR3_IN_RISING	Input port 3 group operation / Event is generated upon detection of the rising edge of the external input signal.
RAPI_PGR3_IN_FALLING	Input port 3 group operation / Event is generated upon detection of the falling edge of the external input signal.
RAPI_PGR3_IN_BOTH	Input port 3 group operation / Event is generated upon detection of both the rising and falling edge of the external input signal.
RAPI_PGR6_PDBF_VER	Input port 6 group operation / Overwriting PDBF is enabled.
RAPI_PGR6_IN_RISING	Input port 6 group operation / Event is generated upon detection of the rising edge of the external input signal.
RAPI_PGR6_IN_FALLING	Input port 6 group operation / Event is generated upon detection of the falling edge of the external input signal.
RAPI_PGR6_IN_BOTH	Input port 6 group operation / Event is generated upon detection of both the rising and falling edge of the external input signal.
RAPI_SP_OUT_0	Output single port operation / 0 is output when the event is input.
RAPI_SP_OUT_1	Output single port operation / 1 is output when the event is input.
RAPI_SP_OUT_TOGGLE	Output single port operation / The toggled (inverted) value is output when the event is input.
RAPI_SP_IN_ISING	Input single port operation / Event is output upon detection of the rising edge.
RAPI_SP_IN_FALLING	Input single port operation / Event is output upon detection of the falling edge.
RAPI_SP_IN_BOTH	Input single port operation / Event is output upon detection of both the rising and falling edge.

RAPI_SP_P3_0	Specifies P30 as a single-port.
RAPI_SP_P3_1	Specifies P31 as a single-port.
RAPI_SP_P3_2	Specifies P32 as a single-port.
RAPI_SP_P3_3	Specifies P33 as a single-port.
RAPI_SP_P3_4	Specifies P34 as a single-port.
RAPI_SP_P3_5	Specifies P35 as a single-port.
RAPI_SP_P3_6	Specifies P36 as a single-port.
RAPI_SP_P3_7	Specifies P37 as a single-port.
RAPI_SP_P6_0	Specifies P60 as a single-port.
RAPI_SP_P6_1	Specifies P61 as a single-port.
RAPI_SP_P6_2	Specifies P62 as a single-port.
RAPI_SP_P6_3	Specifies P63 as a single-port.
RAPI_SP_P6_4	Specifies P64 as a single-port.
RAPI_SP_P6_5	Specifies P65 as a single-port.
RAPI_SP_P6_6	Specifies P66 as a single-port.
RAPI_SP_P6_7	Specifies P67 as a single-port.

• Operation at the time an event for timer RA (RAPI_ELC_TIMER_RA) is linked

(Operations when Event is Input) Specify one from { RAPI_OP_TIMER_RA_ON, RAPI_OP_TIMER_RA_EV, RAPI_OP_TIMER_RA_INVALID }.

(Callback function pointer) Specify 0.

• Operation at the time an event for Timer RB (RAPI_ELC_TIMER_RB) is linked.

(Operations when Event is Input) Specify one from { RAPI_OP_TIMER_RB_ON, RAPI_OP_TIMER_RB_EV, RAPI_OP_TIMER_RB_INVALID }.

(Callback function pointer) Specify 0.

• Operation at the time an event for Timer RC (RAPI_ELC_TIMER_RC) is linked.

(Operations when Event is Input) Specify one from { RAPI_OP_TIMER_RC_ON, RAPI_OP_TIMER_RC_EV, RAPI_OP_TIMER_RC_IC, RAPI_OP_TIMER_RC_INVALID }.

(Callback function pointer) Specify 0.

• Operation at the time an event for Timer RD_0 channel 0 (RAPI_ELC_TIMER_RD0) is linked.

(Operations when Event is Input) Specify one from { RAPI_OP_TIMER_RD0_ON, RAPI_OP_TIMER_RD0_EV, RAPI_OP_TIMER_RD0_IC, RAPI_OP_TIMER_RD0_INVALID }.

(Callback function pointer) Specify 0.

• Operation at the time an event for Timer RD_0 channel 1 (RAPI_ELC_TIMER_RD1) is linked.

(Operations when Event is Input) Specify one from { RAPI_OP_TIMER_RD1_ON, RAPI_OP_TIMER_RD1_EV, RAPI_OP_TIMER_RD1_IC, RAPI_OP_TIMER_RD1_INVALID }.

(Callback function pointer)	Specify 0.
• Operation at the time an event for Timer RG (RAPI_ELC_TIMER_RG) is linked.	
(Operations when Event is Input)	Specify one from { RAPI_OP_TIMER_RG_ON, RAPI_OP_TIMER_RG_EV, RAPI_OP_TIMER_RG_IC, RAPI_OP_TIMER_RG_INVALID }.
(Callback function pointer)	Specify 0.
• Operation at the time an event for AD converter (RAPI_ELC_ADC1～RAPI_ELC_ADC2) is linked.	
(Operations when Event is Input)	Specify 0.
(Callback function pointer)	Specify 0.
• Operation at the time an event for DA converter (RAPI_ELC_DAC0～RAPI_ELC_DAC1) is linked.	
(Operations when Event is Input)	Specify 0.
(Callback function pointer)	Specify 0.
• Operation at the time an event for Clock oscillator (RAPI_ELC_CLOCK) is linked.	
(Operations when Event is Input)	Specify 0.
(Callback function pointer)	Specify 0.
• Operation at the time an event for ELC Interrupts (RAPI_ELC_INT1～RAPI_ELC_INT2) is linked.	
(Operations when Event is Input)	Specify 0.
(Callback function pointer)	Specify a pointer to the callback function
• Operation at the time an event for Output port-group 1 (RAPI_ELC_OUT_PORT_G2) is linked.	
(Operations when Event is Input)	Select all ports you wish to use from { RAPI_GR_PORT3_0, RAPI_GR_PORT3_1, RAPI_GR_PORT3_2, RAPI_GR_PORT3_3, RAPI_GR_PORT3_4, RAPI_GR_PORT3_5, RAPI_GR_PORT3_6, RAPI_GR_PORT3_7 }. Then select one from { RAPI_PGR3_OUT_0, RAPI_PGR3_OUT_1, RAPI_PGR3_OUT_TOGGLE, RAPI_PGR3_OUT_BUFFER, RAPI_PGR3_OUT_BIT_SHIFT } as the output operation.
(Callback function pointer)	Specify 0.
• Operation at the time an event for Output port-group 2 (RAPI_ELC_OUT_PORT_G3) is linked.	

(Operations when Event is Input)	Select all ports you wish to use from { RAPI_GR_PORT6_0, RAPI_GR_PORT6_1, RAPI_GR_PORT6_2, RAPI_GR_PORT6_3, RAPI_GR_PORT6_4, RAPI_GR_PORT6_5, RAPI_GR_PORT6_6, RAPI_GR_PORT6_7 }. Then select one from { RAPI_PGR6_OUT_0, RAPI_PGR6_OUT_1, RAPI_PGR6_OUT_TOGGLE, RAPI_PGR6_OUT_BUFFER, RAPI_PGR6_OUT_BIT_SHIFT } as the output operation.
(Callback function pointer)	Specify 0.

• Operation at the time an event for Input port-group 1 (RAPI_ELC_IN_PORT_G2) is linked.

(Operations when Event is Input)	Select all ports you wish to use from { RAPI_GR_PORT3_0, RAPI_GR_PORT3_1, RAPI_GR_PORT3_2, RAPI_GR_PORT3_3, RAPI_GR_PORT3_4, RAPI_GR_PORT3_5, RAPI_GR_PORT3_6, RAPI_GR_PORT3_7 }. Then select one from { RAPI_PGR3_PDBF_OVER, RAPI_PGR3_IN_RISING, RAPI_PGR3_IN_FALLING, RAPI_PGR3_IN_BOTH } as the operation.
(Callback function pointer)	Specify 0.

• Operation at the time an event for Input port-group 2 (RAPI_ELC_OUT_PORT_G3) is linked.

(Operations when Event is Input)	Select all ports you wish to use from { RAPI_GR_PORT6_0, RAPI_GR_PORT6_1, RAPI_GR_PORT6_2, RAPI_GR_PORT6_3, RAPI_GR_PORT6_4, RAPI_GR_PORT6_5, RAPI_GR_PORT6_6, RAPI_GR_PORT6_7 }. Then select one from { RAPI_PGR6_PDBF_OVER, RAPI_PGR6_IN_RISING, RAPI_PGR6_IN_FALLING, RAPI_PGR6_IN_BOTH } as the operation.
(Callback function pointer)	Specify 0.

• Operation at the time an event for Single-port (RAPI_ELC_SINGLE_PORT_1 ～ RAPI_ELC_SINGLE_PORT_4) is linked.

(Operations when Event is Input)	Select one port you wish to use from { RAPI_SP_PORT3_0, RAPI_SP_PORT3_1, RAPI_SP_PORT3_2, RAPI_SP_PORT3_3, RAPI_SP_PORT3_4, RAPI_SP_PORT3_5, RAPI_SP_PORT3_6, RAPI_SP_PORT3_7, RAPI_SP_PORT6_0, RAPI_SP_PORT6_1, RAPI_SP_PORT6_2, RAPI_SP_PORT6_3, RAPI_SP_PORT6_4, RAPI_SP_PORT6_5, RAPI_SP_PORT6_6, RAPI_SP_PORT6_7 } Then select one from { RAPI_SP_OUT_0, RAPI_SP_OUT_1, RAPI_SP_OUT_TOGGLE, RAPI_SP_IN_RISING, RAPI_SP_IN_FALLING, RAPI_SP_IN_BOTH } as the operation.
(Callback function pointer)	Specify 0.

Return value

If the event link controller specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned

Functionality

Event Link Controller

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- Specifiable peripheral module ports and event signals vary with the CPU.
- This API function is not available with CPUs that do not have an event link controller.

Program example

```
#include " rapi_elc_h8s_20223.h"

void func( void )
{
   /* Set event of timer RA */
   __SetEventLink(RAPI_ELC_TIMER_RA|RAPI_EV_TIMER_RB_UDF,
   RAPI_OP_TIMER_RA_ON, 0 );
}
```

__CreateEventGenerateTimer

Synopsis

< Set operating conditions for an event timer >

Boolean __CreateEventGenerateTimer(unsigned long data1, unsigned int data2)

data1	Operating condition for an event timer
data2	Value to be set to the timer ELC counter

Description

Sets conditions for a specified timer.

[data1]

The values listed below are specifiable for [data1].

Two or more values must be separated with "|".

(H8S)

RAPI_TIMER_ELC	Uses Event-Generation Timer.
RAPI_ELC_TIMER_ON	Sets the timer to start operating in __CreateEventGenerateTimer.
RAPI_ELC_TIMER_OFF	Sets the timer to stop operating in __CreateEventGenerateTimer.
RAPI_ELC_TIMER_F1	The count source is ϕ
RAPI_ELC_TIMER_F2	Selects φ/2 for the clock source (φELC).
RAPI_ELC_TIMER_F4	Selects φ/4 for the clock source (φELC).
RAPI_ELC_TIMER_F8	Selects φ/8 for the clock source (φELC).
RAPI_ELC_TIMER_F16	Selects φ/16 for the clock source (φELC).
RAPI_ELC_TIMER_F32	Selects φ/32 for the clock source (φELC).
RAPI_ELC_TIMER_F64	Selects φ/64 for the clock source (φELC).
RAPI_ELC_TIMER_F128	Selects φ/128 for the clock source (φELC).
RAPI_ELC_TIMER_F256	Selects φ/256 for the clock source (φELC).
RAPI_ELC_TIMER_F512	Selects φ/512 for the clock source (φELC).
RAPI_ELC_TIMER_F1024	Selects φ/1024 for the clock source (φELC).
RAPI_ELC_TIMER_F2048	Selects φ/2048 for the clock source (φELC).
RAPI_ELC_TIMER_F4096	Selects φ/4096 for the clock source (φELC).
RAPI_ELC_TIMER_F8192	Selects φ/8192 for the clock source (φELC).
RAPI_ELC_CH0_F1	Selects the clock source φELC/1 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F2	Selects the clock source φELC/2 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F4	Selects the clock source φELC/4 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F8	Selects the clock source φELC/8 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F16	Selects the clock source φELC/16 for the channel 0 event-generation interval.

RAPI_ELC_CH0_F32	Selects the clock source φELC/32 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F64	Selects the clock source φELC/64 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F128	Selects the clock source φELC/128 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F256	Selects the clock source φELC/256 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F512	Selects the clock source φELC/512 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F1024	Selects the clock source φELC/1024 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F2048	Selects the clock source φELC/2048 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F4096	Selects the clock source φELC/4096 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F8192	Selects the clock source φELC/8192 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F16384	Selects the clock source φELC/16384 for the channel 0 event-generation interval.
RAPI_ELC_CH0_F32768	Selects the clock source φELC/32768 for the channel 0 event-generation interval.
RAPI_ELC_CH1_F1	Selects the clock source φELC/1 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F2	Selects the clock source φELC/2 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F4	Selects the clock source φELC/4 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F8	Selects the clock source φELC/8 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F16	Selects the clock source φELC/16 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F32	Selects the clock source φELC/32 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F64	Selects the clock source φELC/64 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F128	Selects the clock source φELC/128 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F256	Selects the clock source φELC/256 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F512	Selects the clock source φELC/512 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F1024	Selects the clock source φELC/1024 for the channel 1 event-generation interval.

RAPI_ELC_CH1_F2048	Selects the clock source φELC/2048 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F4096	Selects the clock source φELC/4096 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F8192	Selects the clock source φELC/8192 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F16384	Selects the clock source φELC/16384 for the channel 1 event-generation interval.
RAPI_ELC_CH1_F32768	Selects the clock source φELC/32768 for the channel 1 event-generation interval.
RAPI_ELC_CH2_F1	Selects the clock source φELC/1 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F2	Selects the clock source φELC/2 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F4	Selects the clock source φELC/4 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F8	Selects the clock source φELC/8 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F16	Selects the clock source φELC/16 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F32	Selects the clock source φELC/32 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F64	Selects the clock source φELC/64 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F128	Selects the clock source φELC/128 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F256	Selects the clock source φELC/256 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F512	Selects the clock source φELC/512 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F1024	Selects the clock source φELC/1024 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F2048	Selects the clock source φELC/2048 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F4096	Selects the clock source φELC/4096 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F8192	Selects the clock source φELC/8192 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F16384	Selects the clock source φELC/16384 for the channel 2 event-generation interval.
RAPI_ELC_CH2_F32768	Selects the clock source φELC/32768 for the channel 2 event-generation interval.
RAPI_ELC_CH3_F1	Selects the clock source φELC/1 for the channel 3 event-generation interval.

RAPI_ELC_CH3_F2	Selects the clock source φELC/2 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F4	Selects the clock source φELC/4 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F8	Selects the clock source φELC/8 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F16	Selects the clock source φELC/16 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F32	Selects the clock source φELC/32 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F64	Selects the clock source φELC/64 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F128	Selects the clock source φELC/128 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F256	Selects the clock source φELC/256 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F512	Selects the clock source φELC/512 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F1024	Selects the clock source φELC/1024 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F2048	Selects the clock source φELC/2048 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F4096	Selects the clock source φELC/4096 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F8192	Selects the clock source φELC/8192 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F16384	Selects the clock source φELC/16384 for the channel 3 event-generation interval.
RAPI_ELC_CH3_F32768	Selects the clock source φELC/32768 for the channel 3 event-generation interval.
RAPI_ELC_CH0_NO_DELAY	Channel 0: No delay
RAPI_ELC_CH0_DELAY_1	Channel 0: 1 clock cycle delayed
RAPI_ELC_CH0_DELAY_2	Selects 2 clock cycles for the channel 0 delay.
RAPI_ELC_CH0_DELAY_3	Selects 3 clock cycles for the channel 0 delay.
RAPI_ELC_CH1_NO_DELAY	Selects no delay for the channel 1 delay.
RAPI_ELC_CH1_DELAY_1	Selects 1 clock cycle for the channel 1 delay.
RAPI_ELC_CH1_DELAY_2	Selects 2 clock cycles for the channel 1 delay.
RAPI_ELC_CH1_DELAY_3	Selects 3 clock cycles for the channel 1 delay.
RAPI_ELC_CH2_NO_DELAY	Selects no delay for the channel 2 delay.
RAPI_ELC_CH2_DELAY_1	Selects 1 clock cycle for the channel 2 delay.
RAPI_ELC_CH2_DELAY_2	Selects 2 clock cycles for the channel 2 delay.
RAPI_ELC_CH2_DELAY_3	Selects 3 clock cycles for the channel 2 delay.
RAPI_ELC_CH3_NO_DELAY	Selects no delay for the channel 3 delay.
RAPI_ELC_CH3_DELAY_1	Selects 1 clock cycle for the channel 3 delay.

RAPI_ELC_CH3_DELAY_2	Selects 2 clock cycles for the channel 3 delay.
RAPI_ELC_CH3_DELAY_3	Selects 3 clock cycles for the channel 3 delay.

Return value

If specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned

Functionality

Event Link Controller

Reference

__EnableEventGenerateTimer, __DestroyEventGenerateTimer

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- This API function is not available with CPUs that do not have an event link controller.

Program example

```
#include " rapi_elc_h8s_20223.h"

void func( void )
{
    /* Set up ELC timer */

__CreateEventGenerateTimer( RAPI_TIMER_ELC|RAPI_ELC_TIMER_ON|RAPI_ELC_TIMER_F8,
                        RAPI_ELC_CH0_F16|RAPI_ELC_CH3_F2|RAPI_ELC_CH0_DELAY_2 );
}
```

__EnableEventGenerateTimer

Synopsis

< Control the operation of an event timer >

Boolean __EnableEventGenerateTimer(unsigned long data)

data	Operating condition of an event timer

Description

Controls starting and stopping of an event timer.

[data]

For data, the following definition values can be set. To set multiple definition values at the same time, use the symbol “|” to separate each specified value.

(H8S)

RAPI_TIMER_ELC	Uses Event-Generation Timer.
RAPI_ELC_TIMER_ON	Sets Event-Generation Timer to start operating.
RAPI_ELC_TIMER_OFF	Sets Event-Generation Timer to stop operating.

Return value

If the control the operation of an event timer was successfully set, RAPI_TRUE is returned; if settings failed, RAPI_FALSE is returned.

Functionality

Event Link Controller

Reference

__CreateEventGenerateTimer, __DestroyEventGenerateTimer

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- This API function is not available with CPUs that do not have an event link controller.

Program example

```
#include " rapi_elc_h8s_20223.h"

void func( void )
{
    /* Disable ELC timer */
    __EnableEventGenerateTimer( RAPI_TIMER_ELC|RAPI_ELC_TIMER_OFF );
}
```

__DestroyEventGenerateTimer

Synopsis

< Discard the operating conditions for an event timer >

Boolean __DestroyEventGenerateTimer(unsigned long data)

data	Event timer that you wish to discard settings

Description

Discards settings of the Event-Generation Time.

[data]

For data, the following definition values can be set.

(H8S)

RAPI_TIMER_ELC	Event timer

Return value

If the setting was successfully discarded, RAPI_TRUE is returned; if failed, RAPI_FALSE is returned.

Functionality

Event Link Controller

Reference

__CreateEventCounter, __EnableEventCounter, __GetEventCounter

Remark

- If an undefined value is specified in the argument, operation of the API cannot be guaranteed.
- This API function is not available with CPUs that do not have an event link controller.

Program example

```
#include " rapi_elc_h8s_20223.h"

void func( void )
{
   /* Destroy the setting of ELC timer */
   __DestroyEventGenerateTimer( RAPI_TIMER_ELC );
}
```

__ReadPortBufferRegister

Synopsis

< Read data from a port-buffer register >

Boolean __ReadPortBufferRegister(unsigned long data1, unsigned int *data2)

data1	Port for the port buffer register from which data will be read out
data2	Pointer to a variable in which the read value will be stored

Description

Gets the value of the port-buffer register of the specified port.

[data1]

Specify the port for a port buffer register from which you wish to read data.

The following values are specifiable.

(H8S)

RAPI_PORT_3	Port 3 buffer register
RAPI_PORT_6	Port 6 buffer register

Return value

If the port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Event Link Controller

Reference

__WritePortBufferRegister

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- This API function is not available with CPUs that do not have an event link controller.

Program example

```
#include " rapi_elc_h8s_20223.h"

void func( void )
{
   unsigned int data;

   /* Get the value of port buffer register 2 */
   __ReadPortBufferRegister( RAPI_PORT_6, &data );
}
```

__WritePortBufferRegister

Synopsis

< Writes data to a port-buffer register >

Boolean __WritePortBufferRegister(unsigned long data1, unsigned int data2)

data1	Port for the port buffer register to which data will be written
data2	Data to be written to the port buffer register

Description

Sets the value of the port-buffer register of the specified port.

[data1]

Specify the port for a port buffer register to which you wish to write data.

The following values are specifiable.

(H8S)

RAPI_PORT_3	Port 3 buffer register
RAPI_PORT_6	Port 6 buffer register

Return value

If the port specification is incorrect, RAPI_FALSE is returned; otherwise, RAPI_TRUE is returned.

Functionality

Event Link Controller

Reference

__ReadPortBufferRegister

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- This API function is not available with CPUs that do not have an event link controller.

Program example

```
#include " rapi_elc_h8s_20223.h"

void func( void )
{
   /* Set the data to port buffer register 1 */
   __WritePortBufferRegister( RAPI_PORT_3, 0xFF );
}
```

4.2.8 Data Transfer Controller

__CreateDTC

Synopsis

< Set DTC's register information>

Boolean __CreateDTC(unsigned long RegInfoAddr, unsigned short mra_mrb, unsigned long sar, unsigned long dar, unsigned short cra, unsigned short crb)

RegInfoAddr	Start address of register information(stored in the on-chip RAM)
mra_mrb	Setup data of DTC mode A and B
sar	DTC source address
dar	DTC destination address
cra	DTC transfer count A
crb	DTC transfer count B

Description

Transfers information on DTC registers to a specified address in the on-chip RAM.

[RegInfoAddr]

Specify an address (in the range of H'FFDF80-H'FFFF7F) in the on-chip RAM.

[mra_mrb]

Select transfer mode A or B. The following values are specifiable.

(H8S)

RAPI_DTC_MRA_SAR_FIX	SAR is fixed.
RAPI_DTC_MRA_SAR_INC	SAR is incremented after a transfer. (by +1 when Sz = 0; by +2 when Sz = 1)
RAPI_DTC_MRA_SAR_DEC	SAR is decremented after a transfer. (by –1 when Sz = 0; by –2 when Sz = 1)
RAPI_DTC_MRA_DAR_FIX	DAR is fixed.
RAPI_DTC_MRA_DAR_INC	DAR is incremented after a transfer. (by +1 when Sz = 0; by +2 when Sz = 1)
RAPI_DTC_MRA_DAR_DEC	DAR is decremented after a transfer. (by –1 when Sz = 0; by –2 when Sz = 1)
RAPI_DTC_MRA_NORMAL	Normal mode
RAPI_DTC_MRA_REPEAT	Repeat mode
RAPI_DTC_MRA_BLOCK	Block transfer mode
RAPI_DTC_MRA_SEL_SAR	The origin of the transfer is a repeat or block area.
RAPI_DTC_MRA_SEL_DAR	The destination of the transfer is a repeat or block area.
RAPI_DTC_MRA_BYTE_SIZE	Byte-size transfer
RAPI_DTC_MRA_WORD_SIZE	Word-size transfer
RAPI_DTC_MRB_NO_CHAIN	Disables chain transfer.
RAPI_DTC_MRB_ENABLE_CHAIN	Enables chain transfer.
RAPI_DTC_MRB_COMPLETE_IRQ	An interrupt request to the CPU is generated only when the specified type of data transfer is completed.
RAPI_DTC_MRB_EVERY_IRQ	An interrupt request to the CPU is generated whenever a DTC transfer is attempted.

RAPI_DTC_MRB_SEQ_CHAIN	Chain transfers are performed continuously.
RAPI_DTC_MRB_ZERO_CHAIN	Chain transfers are performed only when the value of the transfer counter is 0.

[sar]

Specify the source address of data to be transferred by the DTC.

[dar]

Specify the destination address of data to be transferred by the DTC.

[cra]

Specify the number of times that data is to be transferred by the DTC.

[crb]

Specify the number of times block data is to be transferred by theDTC in block transfer mode.

Return value

RAPI_TRUE is always returned.

Functionality

Data Transfer Controller

Remark

- The correct operation is not guaranteed if any value other than those listed above is specified as the first to sixth parameters.
- [cra] can be either a 16-bit unit or two 8-bit units depending on the transfer mode.
  Normal mode: 16 bits Repeat or block transfer mode: 8 bits x 2
- Not all transfer modes support [crb].
  Normal or repeat mode: Supported Block transfer mode: Not supported

Program example

```
#include " rapi_dtc_h8s_20203.h"

void main(void)
{
        unsigned long MRA_MRB;
        Boolean RESULT;

        MRA_MRB   =   (RAPI_DTC_MRA_SAR_INC   |   RAPI_DTC_MRA_DAR_INC   |
RAPI_DTC_MRA_BLOCK | RAPI_DTC_MRA_SEL_SAR | RAPI_DTC_MRA_BYTE_SIZE |
RAPI_DTC_MRB_NO_CHAIN | RAPI_DTC_MRB_COMPLETE_IRQ | RAPI_DTC_MRB_SEQ_CHAIN);

         / Start address of register information :*H'FFF000
         SAR is incremented after a transfer.
         DAR is incremented after a transfer.
         Block transfer mode
         The origin of the transfer is a repeat or block area.
         Byte-size transfer
         Disables chain transfer.
         An interrupt request to the CPU is generated only when the specified
type of data transfer is completed.
         Chain transfers are performed continuously.
         the source address:H'00003000
         the destination address:H'00FFE000
         the size of blocksH'FF
         the number of transfers:H'0001*/

         RESULT = __CreateDTC( 0xfff000, MRA_MRB, 0x00003000, 0x00FFE000,
0x0000, 0x0001);
}
```

__EnableDTC

Synopsis

< Enable interrupts to activate the DTC >

Boolean __EnableDTC(unsigned long signal)

signal	Value indicating the source to activate the DTC

Description

Specify the source to activate the DTC so that the corresponding DTCE bit in the DTC enable register will be enabled.

[signal]

The following values are specifiable.

(H8S)

RAPI_DTCERA_IRQ0	IRQ0 is the source to activate the DTC
RAPI_DTCERA_IRQ1	Selects IRQ1 as a DTC activation source.
RAPI_DTCERA_IRQ2	Selects IRQ2 as a DTC activation source.
RAPI_DTCERA_IRQ3	Selects IRQ3 as a DTC activation source.
RAPI_DTCERA_IRQ4	Selects IRQ4 as a DTC activation source.
RAPI_DTCERA_IRQ5	Selects IRQ5 as a DTC activation source.
RAPI_DTCERA_IRQ6	Selects IRQ6 as a DTC activation source.
RAPI_DTCERA_IRQ7	Selects IRQ7 as a DTC activation source.
RAPI_DTCERA_IADEND_1	Selects IADEND_1 as a DTC activation source.
RAPI_DTCERA_IADCMP_1	Selects IADCMP_1 as a DTC activation source.
RAPI_DTCERA_IADEND_2	Selects IADEND_2 as a DTC activation source. (*1)
RAPI_DTCERA_IADCMP_2	Selects IADCMP_2 as a DTC activation source. (*1)
RAPI_DTCERA_ELC1FP	Selects ELC1FP as a DTC activation source.
RAPI_DTCERA_ELC2FP	Selects ELC2FP as a DTC activation source.
RAPI_DTCERA_SCI3_1_RXI	Selects SCI3_1_RXI as a DTC activation source.
RAPI_DTCERA_SCI3_1_TXI	Selects SCI3_1_TXI as a DTC activation source.
RAPI_DTCERA_SCI3_2_RXI	Selects SCI3_2_RXI as a DTC activation source.
RAPI_DTCERA_SCI3_2_TXI	Selects SCI3_2_TXI as a DTC activation source.
RAPI_DTCERA_SCI3_3_RXI	Selects SCI3_3_RXI as a DTC activation source.
RAPI_DTCERA_SCI3_3_TXI	Selects SCI3_3_TXI as a DTC activation source.
RAPI_DTCERA_IIC2_SSU_RXI	Selects IIC2_SSU_RXI as a DTC activation source.
RAPI_DTCERA_IIC2_SSU_TXI	Selects IIC2_SSU_TXI as a DTC activation source.
RAPI_DTCERA_ITCMA	Selects ITCMA as a DTC activation source. (*2)
RAPI_DTCERA_ITCMB	Selects ITCMB as a DTC activation source. (*2)
RAPI_DTCERA_ITCMC	Selects ITCMC as a DTC activation source. (*2)
RAPI_DTCERA_ITCMD	Selects ITCMD as a DTC activation source. (*2)
RAPI_DTCERA_ITDMA0_0	Selects ITDMA0_0 as a DTC activation source.
RAPI_DTCERA_ITDMB0_0	Selects ITDMB0_0 as a DTC activation source.
RAPI_DTCERA_ITDMC0_0	Selects ITDMC0_0 as a DTC activation source.
RAPI_DTCERA_ITDMD0_0	Selects ITDMD0_0 as a DTC activation source.
RAPI_DTCERA_ITDMA0_1	Selects ITDMA0_1 as a DTC activation source.

RAPI_DTCERA_ITDMB0_1	Selects ITDMB0_1 as a DTC activation source.
RAPI_DTCERA_ITDMC0_1	Selects ITDMC0_1 as a DTC activation source.
RAPI_DTCERA_ITDMD0_1	Selects ITDMD0_1 as a DTC activation source.
RAPI_DTCERA_ITDMA1_2	Selects ITDMA1_2 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMB1_2	Selects ITDMB1_2 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMC1_2	Selects ITDMC1_2 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMD1_2	Selects ITDMD1_2 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMA1_3	Selects ITDMA1_3 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMB1_3	Selects ITDMB1_3 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMC1_3	Selects ITDMC1_3 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMD1_3	Selects ITDMD1_3 as a DTC activation source. (*3)
RAPI_DTCERA_ITESC	Selects ITESC as a DTC activation source.
RAPI_DTCERA_ITEMI	Selects ITEMI as a DTC activation source.
RAPI_DTCERA_ITEHR	Selects ITEHR as a DTC activation source.
RAPI_DTCERA_ITEDY	Selects ITEDY as a DTC activation source.
RAPI_DTCERA_ITEWK	Selects ITEWK as a DTC activation source.
RAPI_DTCERA_ITGMA	Selects ITGMA as a DTC activation source.
RAPI_DTCERA_ITGMB	Selects ITGMB as a DTC activation source.
RAPI_SWDTC	

Return value

RAPI_TRUE is returned.

Functionality

Data Transfer Controller

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- Modules with an asterisk (*) are not supported by all device groups.

  *1. Only available in the H8S/20223 group.

  *2. Only available in the H8S/20103 group.

  *3. Not available in the H8S/20103 group.

Program example

```
#include " rapi_dtc_h8s_20203.h"
void main(void)
{
    unsigned long MRA_MRB;
    Boolean RESULT;

    MRA_MRB=(RAPI_DTC_MRA_SAR_INC|RAPI_DTC_MRA_DAR_INC|RAPI_DTC_MRA_BLOCK|
    RAPI_DTC_MRA_SEL_SAR|RAPI_DTC_MRA_BYTE_SIZE|
    RAPI_DTC_MRB_NO_CHAIN|RAPI_DTC_MRB_COMPLETE_IRQ|RAPI_DTC_MRB_SEQ_CHAIN);
    RESULT=__CreateDTC(0xfff000, MRA_MRB, 0x00003000, 0x00004000, 0x0000,
    0x0001);
    /* Selects IRQ0 as a DTC activation source. */
    RESULT = __EnableDTC(RAPI_DTCERA_IRQ0);
}
```

__DisableDTC

Synopsis

< Disable interrupts to activate the DTC >

Boolean __DisableDTC(unsigned long signal)

signal	Value indicating the source to activate the DTC

Description

Specify the source to activate the DTC so that the corresponding DTCE bit in the DTC enable register will be disabled.

[signal]

The following values are specifiable.

(H8S)

RAPI_DTCERA_IRQ0	IRQ0 is the source to activate the DTC
RAPI_DTCERA_IRQ1	Disables IRQ1 as a DTC activation source.
RAPI_DTCERA_IRQ2	Disables IRQ2 as a DTC activation source.
RAPI_DTCERA_IRQ3	Disables IRQ3 as a DTC activation source.
RAPI_DTCERA_IRQ4	Disables IRQ4 as a DTC activation source.
RAPI_DTCERA_IRQ5	Disables IRQ5 as a DTC activation source.
RAPI_DTCERA_IRQ6	Disables IRQ6 as a DTC activation source.
RAPI_DTCERA_IRQ7	Disables IRQ7 as a DTC activation source.
RAPI_DTCERA_IADEND_1	Disables IADEND_1 as a DTC activation source.
RAPI_DTCERA_IADCMP_1	Disables IADCMP_1 as a DTC activation source.
RAPI_DTCERA_IADEND_2	Disables IADEND_2 as a DTC activation source. (*1)
RAPI_DTCERA_IADCMP_2	Disables IADCMP_2 as a DTC activation source. (*1)
RAPI_DTCERA_ELC1FP	Disables ELC1FP as a DTC activation source.
RAPI_DTCERA_ELC2FP	Disables ELC2FP as a DTC activation source.
RAPI_DTCERA_SCI3_1_RXI	Disables SCI3_1_RXI as a DTC activation source.
RAPI_DTCERA_SCI3_1_TXI	Disables SCI3_1_TXI as a DTC activation source.
RAPI_DTCERA_SCI3_2_RXI	Disables SCI3_2_RXI as a DTC activation source.
RAPI_DTCERA_SCI3_2_TXI	Disables SCI3_2_TXI as a DTC activation source.
RAPI_DTCERA_SCI3_3_RXI	Disables SCI3_3_RXI as a DTC activation source.
RAPI_DTCERA_SCI3_3_TXI	Disables SCI3_3_TXI as a DTC activation source.
RAPI_DTCERA_IIC2_SSU_RXI	Disables IIC2_SSU_RXI as a DTC activation source.
RAPI_DTCERA_IIC2_SSU_TXI	Disables IIC2_SSU_TXI as a DTC activation source.
RAPI_DTCERA_ITCMA	Disables ITCMA as a DTC activation source. (*2)
RAPI_DTCERA_ITCMB	Disables ITCMB as a DTC activation source. (*2)
RAPI_DTCERA_ITCMC	Disables ITCMC as a DTC activation source. (*2)
RAPI_DTCERA_ITCMD	Disables ITCMD as a DTC activation source. (*2)
RAPI_DTCERA_ITDMA0_0	Disables ITDMA0_0 as a DTC activation source.
RAPI_DTCERA_ITDMB0_0	Disables ITDMB0_0 as a DTC activation source.
RAPI_DTCERA_ITDMC0_0	Disables ITDMC0_0 as a DTC activation source.
RAPI_DTCERA_ITDMD0_0	Disables ITDMD0_0 as a DTC activation source.
RAPI_DTCERA_ITDMA0_1	Disables ITDMA0_1 as a DTC activation source.

RAPI_DTCERA_ITDMB0_1	Disables ITDMB0_1 as a DTC activation source.
RAPI_DTCERA_ITDMC0_1	Disables ITDMC0_1 as a DTC activation source.
RAPI_DTCERA_ITDMD0_1	Disables ITDMD0_1 as a DTC activation source.
RAPI_DTCERA_ITDMA1_2	Disables ITDMA1_2 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMB1_2	Disables ITDMB1_2 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMC1_2	Disables ITDMC1_2 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMD1_2	Disables ITDMD1_2 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMA1_3	Disables ITDMA1_3 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMB1_3	Disables ITDMB1_3 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMC1_3	Disables ITDMC1_3 as a DTC activation source. (*3)
RAPI_DTCERA_ITDMD1_3	Disables ITDMD1_3 as a DTC activation source. (*3)
RAPI_DTCERA_ITESC	Disables ITESC as a DTC activation source.
RAPI_DTCERA_ITEMI	Disables ITEMI as a DTC activation source.
RAPI_DTCERA_ITEHR	Disables ITEHR as a DTC activation source.
RAPI_DTCERA_ITEDY	Disables ITEDY as a DTC activation source.
RAPI_DTCERA_ITEWK	Disables ITEWK as a DTC activation source.
RAPI_DTCERA_ITGMA	Disables ITGMA as a DTC activation source.
RAPI_DTCERA_ITGMB	Disables ITGMB as a DTC activation source.
RAPI_SWDTC	If the MCU has two or more A/D converter units, status flags of the unit with the first number will be acquired.

Return value RAPI_TRUE is always returned.

Functionality Data Transfer Controller

Remark

- If an undefined value is specified in the first argument, operation of the API cannot be guaranteed.
- Modules with an asterisk (*) are not supported by all device groups.
  *1. Only available in the H8S/20223 group.
  *2. Only available in the H8S/20103 group.
  *3. Not available in the H8S/20103 group.

Program example

```
#include " rapi_dtc_h8s_20203.h"

void main(void)
{
        unsigned long MRA_MRB;
        Boolean RESULT;
        MRA_MRB   =   (RAPI_DTC_MRA_SAR_INC   |   RAPI_DTC_MRA_DAR_INC   |
RAPI_DTC_MRA_BLOCK |
        RAPI_DTC_MRA_SEL_SAR | RAPI_DTC_MRA_BYTE_SIZE | RAPI_DTC_MRB_NO_CHAIN
|
        RAPI_DTC_MRB_COMPLETE_IRQ | RAPI_DTC_MRB_SEQ_CHAIN);
        RESULT  =  __CreateDTC(  0xfff000,  MRA_MRB,  0x00003000,  0x00004000,
0x0000, 0x0001);
        RESULT = __EnableDTC(RAPI_DTCERA_IRQ0);
        /* Disables IRQ0 as a DTC activation source. */
        RESULT = __DisableDTC(RAPI_DTCERA_IRQ0);
}
```

__EnableSWDTC

Synopsis

< Make settings for DTC activation by software >

Boolean __EnableSWDTC(unsigned long vect, VoidFuncNotify *func)

vect	Vector number for DTC activation by software
*func	Pointer to a callback function (or 0 if no callback functions are needed)

Description

Makes settings for DTC activation by software.

[vect]

Vector number for DTC activation by software (when VOFR=H'0000: H'400 + (vector number *2))

[*func]

The user must provide the function of [func].

The DTC driver will call the function of [func] on generation of a software-activated data-transfer end interrupt and pass a parameter to inform the user of the state of transmission or reception.

The declaration of [func] is as follows.

void "function name"(unsigned char notify);

The parameter passed by the DTC driver is as follows.

Vector number for DTC activation (value in DTVECR)

Return value

If the SWDTE is 1, RAPI_FALSE is returned; If the SWDTE is 0, RAPI_TRUE is returned.

Functionality

Data Transfer Controller

Remark

- The correct operation is not guaranteed if any value other than those listed above is specified as the first and second parameters.
- Setting of the vector number and activation of the DTC are performed (written) at the same time.

Program example

```
#include " rapi_dtc_h8s_20203.h"

void main(void)
{
    unsigned long MRA_MRB;
    Boolean RESULT;
    MRA_MRB = (RAPI_DTC_MRA_SAR_INC | RAPI_DTC_MRA_DAR_INC | RAPI_DTC_MRA_BLOCK
    RAPI_DTC_MRA_SEL_SAR|RAPI_DTC_MRA_BYTE_SIZE|RAPI_DTC_MRB_NO_CHAIN|
    RAPI_DTC_MRB_COMPLETE_IRQ | RAPI_DTC_MRB_SEQ_CHAIN);
    RESULT = __CreateDTC( 0xfff000, MRA_MRB, 0x00003000, 0x00004000,
    0x0000, 0x0001);
    /*DTC Vector H'400 + (H'10 * 2) = H'420: no callback functions are set.*/
    RESULT = __EnableSWDTC( 0x00000010, 0 );
}
```

Renesas Embedded Application Programming Interface
Reference Manual

Publication Date: May. 29, 2009 Rev.1.12

Published by: Sales Strategic Planning Div.
Renesas Technology Corp.

Edited by: Multi-purpose MCU Application Engineering Department
Application Engineering Division 1
Renesas Solutions Corp.

Renesas Embedded Application Programming Interface Reference Manual

Renesas Electronics Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8668 Japan

REJ10J2020-0100