RZ/T1 Group

CMSIS-RTOS RTX for Cortex-R4
CMT(W) & ELC & ADC Sample Programs
(for use with the following combinations of environments and compilers:
EWARM and ICCARM, e2studio andRenesas GCC, DS-5 and ARMCC)

R01AN3539EJ0100
Rev.1.00
Mar.07, 2017

Introduction

This application note describes a sample program for RZ/T1 devices which handles A/D conversion triggered by a timer. A 32-bit timer (compare-match timer W, CMTW) is interlinked with the ADC through the event link controller (ELC).

The features of the CMT(W), ELC, and ADC sample programs are given below.

- Matches in comparison by the timer are detected at 3-second intervals.
- The voltage input from a potentiometer is A/D converted each time the timer detects a match in comparison.

Target Devices

RZ/T1

When applying the program covered in this application note to another microcontroller, modify the program according to the specifications for the target microcontroller and extensively evaluate the modified program.

Table of Contents

1. Specifications

The table below lists the peripheral modules used and their applications

Table 1.1 Peripheral Modules and Their Applications

Peripheral Module	Application
RZ/T1 internal compare-match timer (CMT)	A 16-bit timer counter that outputs an interrupt and issues event signals to the ELC when the specified counter value has been reached.
RZ/T1 internal compare-match timer W (CMTW)	A 32-bit timer counter that outputs an interrupt and issues event signals to the ELC when the specified counter value has been reached.
RZ/T1 internal event link controller (ELC)	Receives event signals from linked modules and conveys them to other modules as triggers for operations.
Low power consumption	Supplies and stops the clock signal to the CMTW and ELC modules.
Interrupt controller (ICUA)	• Control of interrupts for the CMTW (units 0 and 1) Compare match (vectors 25 and 30) Input capture 0 (vectors 26 and 31) Input capture 1 (vectors 27 and 32) Output compare 0 (vectors 28 and 33) Output compare 1 (vectors 29 and 34) • Control of ELC interrupts Interrupt request signal 1 (vector 242) Interrupt request signal 2 (vector 243)
ADC (AN007)	A/D conversion of the voltage input from a potentiometer
Potentiometer	Inputs variable voltages to the ADC.

The basics of the following are described in *RZ/T1 Group User's Manual: Hardware*.

- Operating mode used for event linking
- Compare-match timer W (CMTW)
- Event link controller (ELC)
- Low power consumption
- Interrupt controller (ICUA)
- I/O ports
- Multi-function pin controller (MPC)
- 12-bit A/D converter (S12ADCa)

2. Operating Environment

Operation of the sample program covered in this application note has been confirmed under the conditions below.

Table 2.1 Operating Environment

Item	Description
MCU used	RZ/T1 Group
Operating frequency	CPUCLK = 450 MHz
Operating voltage	3.3 V
Integrated development environment	• Embedded Workbench for ARM ((EWARM), version 7.80.2, from IAR Systems • e2studio, version 5.2.0.020, from Renesas • DS-5, version: 5.25.0, from ARM
Operating mode	SPI boot mode (booting from serial flash memory) SW4: ON/ON/ON
	RAM boot mode (booting by using JTAG-ICE for direct downloading to the internal RAM) or 16-bit bus boot mode (booting from NOR flash memory) SW4: ON/OFF/ON
Board used	RZ/T1 CPU board (RTK7910018C00000BE)
Devices used (functions to be used on the board)	• NOR flash memory (connected to the CS0 and CS1 spaces) Manufacturer: Macronix International Co., Ltd. Model: MX29GL512FLT2I-10Q • SDRAM (connected to the CS2 and CS3 spaces) Manufacturer: Integrated Silicon Solution Inc. Model: IS42S16320D-7TL • Serial flash memory Manufacturer: Macronix International Co., Ltd. Model: MX25L51245GMI-10G • Potentiometer (AN007)
ICE	• I-jet JTAG emulator from IAR Systems • J-Link JTAG emulator from SEGGER • KEIL ULINK2 emulator from ARM

3. Documents

3.1 Related Documents

The documents related to descriptions in this application note are listed below for reference.

- CMSIS-RTOS compliant Kernel Version 4.74
  This is the specification for RTOS for use in this system. The sample application uses the functions of the RTX CMSIS-RTOS. Each sample application initializes the RTX CMSIS-RTOS.

- RZ/T1 Group User's Manual: Hardware (R01UH0483)
  This document describes the hardware specifications of RZ/T1 devices.
  Download the latest version from the Renesas Electronics website.

- Application Note: RZ/T1 Group Initial Settings (R01AN2554)
  This document describes the initial settings for RZ/T1 devices. For those not covered in this application note, the values set in *Application Note: RZ/T1 Group Initial Settings* are used.
  Download the latest version from the Renesas Electronics website.

- Application Note: RZ/T1 Group CMSIS-RTOS RTX for Cortex-R4 RTX Sample Programs (R01AN3538EJ)
  This document describes the RTX CMSIS-RTOS sample applications for RZ/T1 devices.

- Technical Update and Technical News
  Download the latest version from the Renesas Electronics website.

- User's manuals related to the development environment
  The latest version of the IAR integrated development environment (IAR Embedded Workbench for ARM) is available from the IAR Systems website.

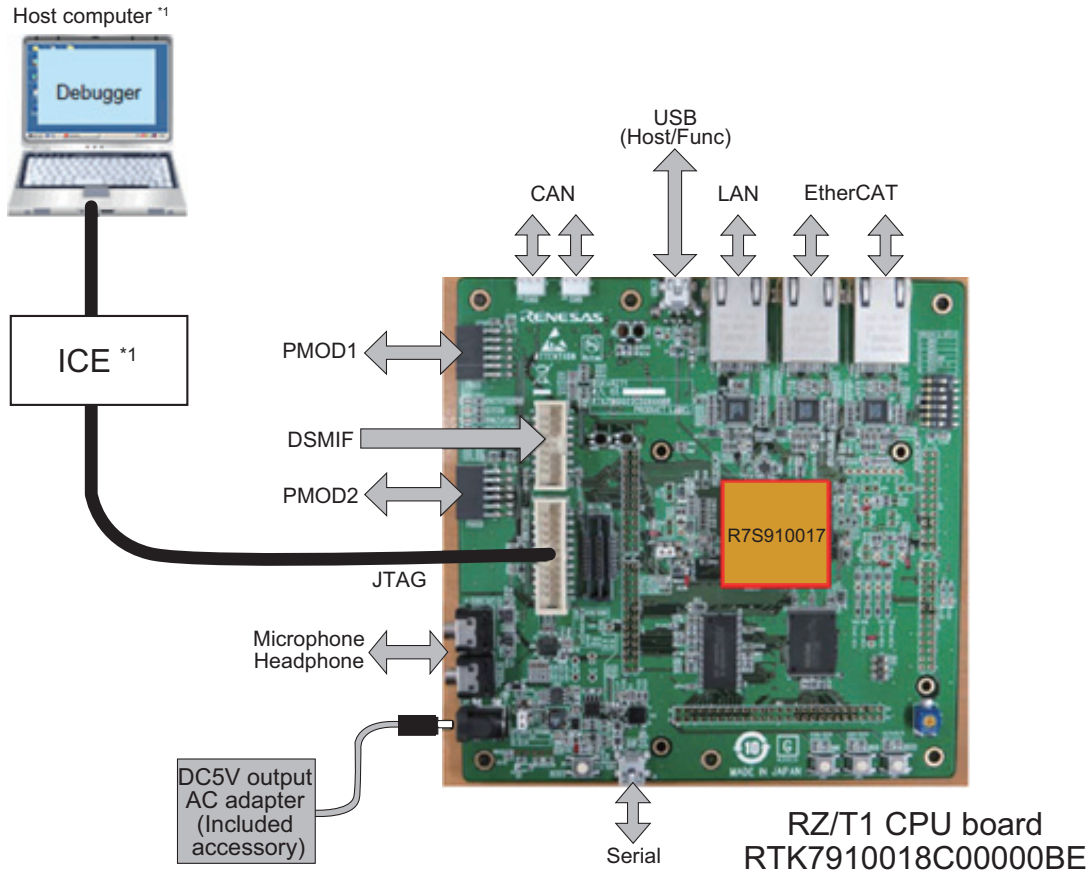
  The latest version of the DS-5 integrated development environment (ARM Development Studio 5) is available from the ARM website.

  The latest version of the Renesas Electronics software development tools (e2studio, etc.) is available from the Renesas Electronics website.

4. Hardware

4.1 Example of the Hardware Configuration

The figure below shows a basic example of connection when the RZ/T1 CPU board is used.



Note 1. You will need to provide this separately.

Figure 4.1 Operating Environment

The figure below shows an example of the hardware configuration for this sample program.

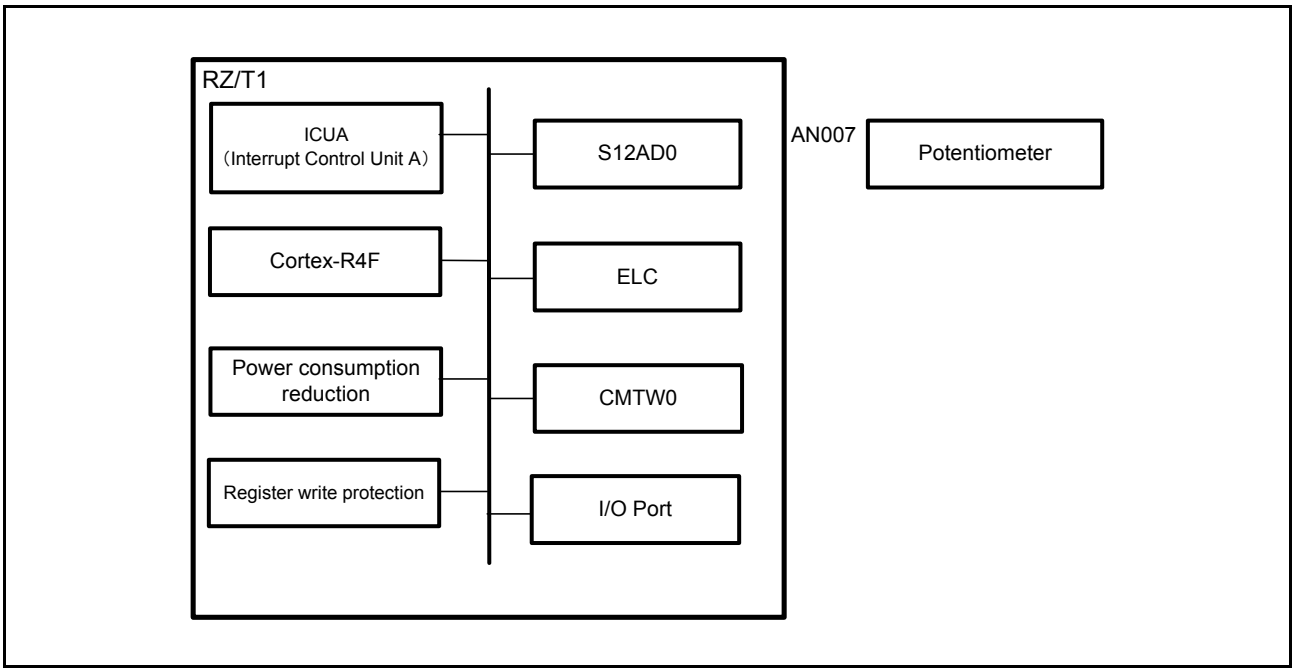


Figure 4.2 Example of the Hardware Configuration

4.2 Pins

Table 4.1 lists the pins used and their functions.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Description
MD0	Input	Selection of the operating mode MD0 = "L", MD1 = "L", MD2 = "L" (SPI boot mode) MD0 = "L", MD1 = "H", MD2 = "L" (RAM boot mode or 16-bit bus boot mode)
MD1	Input	
MD2	Input	
AN007	Input	Potentiometer

5. Software

5.1 Operation in Outline

The table below describes the operation of the sample program in outline. The figure overleaf is a block diagram of the flow in the system set up by this sample program.

Table 5.1 Operation Outline

Function	Outline
Operation outline	• The following items 1 to 4 are executed repeatedly. 1. Event signals are generated in response to matches in comparison by the CMTW. 2. The ELC conveys the event signals to the ADC. 3. A/D conversion by the ADC 4. The results of A/D conversion are displayed by printf.
Channel number (CMTW)	• Channel 0 is selected.
Operating mode (CMTW)	• Compare-match only is set.
Compare-match cycle (CMTW)	• Approx. 3 seconds (Input clock to the timer counter: PCLKD/8; compare match target setting: 28125000)
Source for clearing the timer counter (CMTW)	• Compare-match Operating with the period of the timer counter
Source event for linking (ELC)	• CMTW, channel 0, compare-match
Destination event for linking (ELC)	• S12AD0
Condition for starting A/D conversion	• Activation by a synchronous trigger
ADC driver setting	• The following are set. Input channel: AN007 Operating mode: Single scan mode Trigger to start conversion: Activation by a synchronous trigger (ELC)

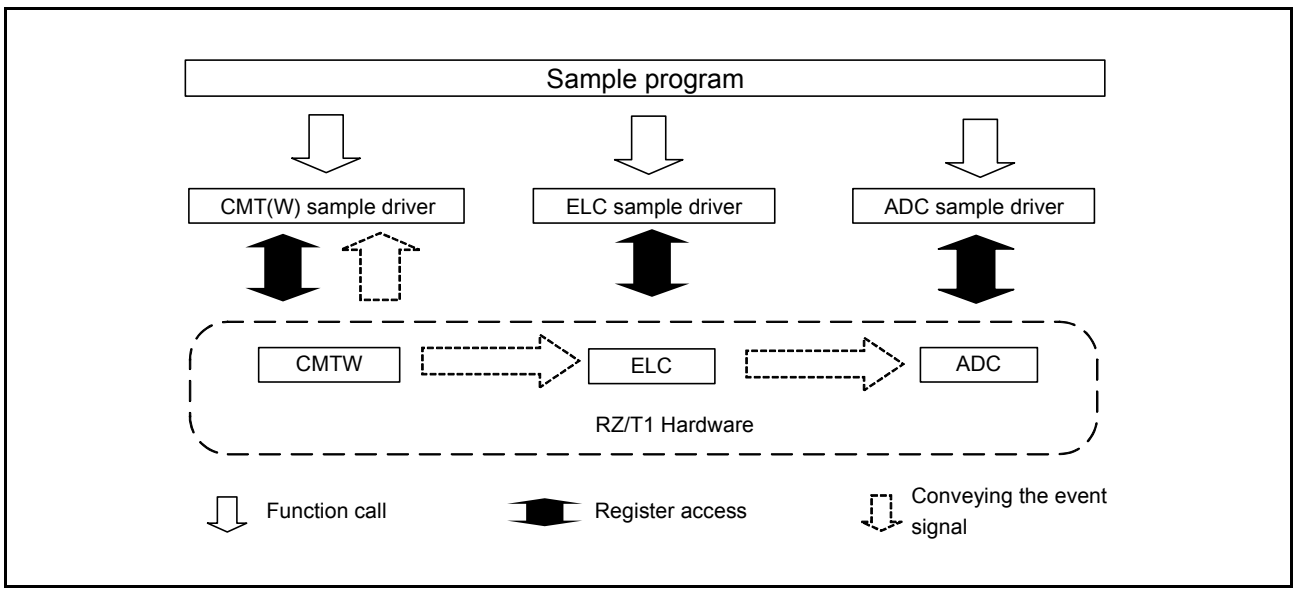


Figure 5.1 System Block Diagram

5.2 Project Settings

Project settings for use in EWARM, e2studio, or DS-5 as the development environment are described in *Application Note: RZ/T1 Group Initial Settings*.

5.3 Memory Map

The address space of the RZ/T1 group and a memory map of the RZ/T1 CPU board are described in *Application Note: RZ/T1 Group Initial Settings*.

5.3.1 Assignment of the Sample Program to Sections

Refer to *Application Note: RZ/T1 Group Initial Settings* for the sections to be used in the program, assignment to sections (loading view) of the sample program in its initial state, and assignment to sections of the sample program following the application of scatter loading (execution view).

5.3.2 MPU Settings

The settings for the MPU are described in *Application Note: RZ/T1 Group Initial Settings*.

5.3.3 Exception Handling Vector Table

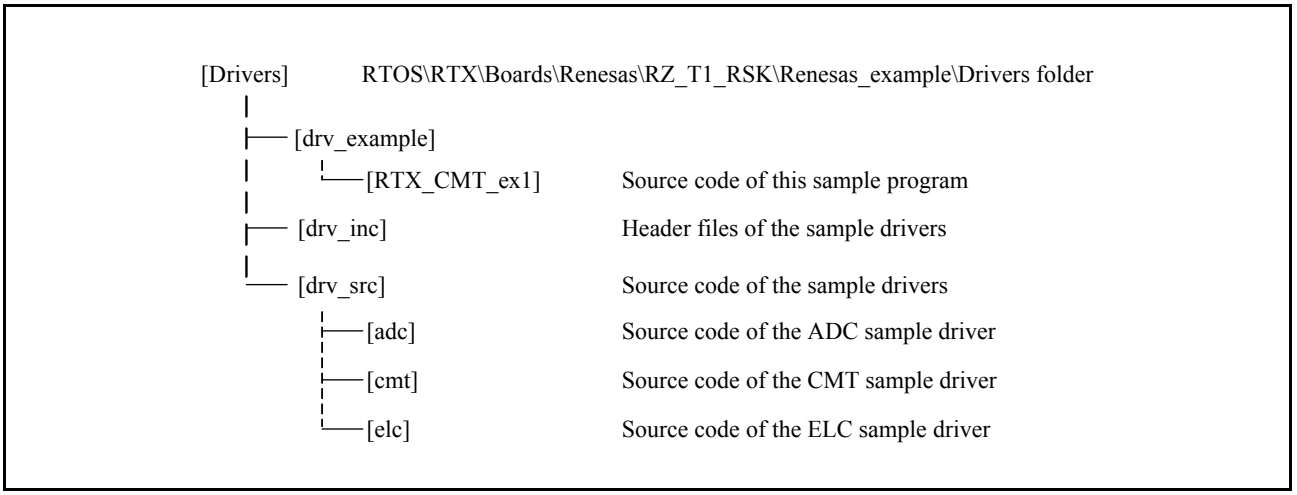
The exception processing vector table is described in *Application Note: RZ/T1 Group Initial Settings*.

5.3.4 Required Memory Size

Regarding the required amounts of memory, example memory sizes for the RTX_ex1 sample program are listed in *Application Note: RZ/T1 Group CMSIS-RTOS RTX for Cortex-R4 RTX Sample Programs*. For details of the required amount of memory, refer to the actual project.

5.4 Folder Structure

The following shows the folder structure of the sample program and sample drivers.



5.5 Interrupts

Table 5.2 Interrupts for the Sample Program

Interrupt (Source ID)	Priority	Outline
Compare-match interrupt (25)	7	The callback function is called.
A/D conversion completed interrupt (35)	7	The callback function is called.

5.6 Fixed-Width Integers

Table 5.3 lists the fixed-width integers for the sample program.

Table 5.3 Fixed-Width Integers for the Sample Program

Symbol	Description
int8_t	8-bit signed integer (defined in the standard library)
int16_t	16-bit signed integer (defined in the standard library)
int32_t	32-bit signed integer (defined in the standard library)
int64_t	64-bit signed integer (defined in the standard library)
uint8_t	8-bit unsigned integer (defined in the standard library)
uint16_t	16-bit unsigned integer (defined in the standard library)
uint32_t	32-bit unsigned integer (defined in the standard library)
uint64_t	64-bit unsigned integer (defined in the standard library)

5.7 Constants and Error Codes

The tables below list the constants and error codes for the sample program and sample drivers.

5.7.1 Constants for the Sample Program

Table 5.4 Constants for the Sample Program

Constant	Setting	Description
CMTW_MATCH_COUNT_3SEC	28125000	Sets the counter value for a compare match. In combination with the frequency division ratio of PCLKD in the sample program, this value corresponds to a compare-match cycle of three seconds.

5.7.2 Error Code Constants for the Sample Program

Table 5.5 Error Code Constants for the Sample Program

Constant	Setting	Description
ESUCCESS	0	Success
EINVAL	-28	Invalid argument
EACCES	-64	Access denied
EBUSY	-67	Busy
EFAULT	-97	Failure

5.7.3 Constants for the ADC Sample Driver

Table 5.6 Constants of Version Definitions (ADC)

Constant	Value	Description
ADC_VERSION_MAJOR	1	Major part of the version number of the ADC sample driver
CMT_VERSION_MINOR	0	Minor part of the version number of the ADC sample driver

Table 5.7 Constants of Mask Definitions (ADC)

Constant	Value	Description
ADC_MASK_GROUPB_OFF	0	Remove masking of group B.
ADC_MASK_ADD_OFF	0	Remove masking of additional channels.
ADC_MASK_CH0	(1<<0)	Mask channel 0.
ADC_MASK_CH1	(1<<1)	Mask channel 1.
ADC_MASK_CH2	(1<<2)	Mask channel 2.
ADC_MASK_CH3	(1<<3)	Mask channel 3.
ADC_MASK_CH4	(1<<4)	Mask channel 4.
ADC_MASK_CH5	(1<<5)	Mask channel 5.
ADC_MASK_CH6	(1<<6)	Mask channel 6.
ADC_MASK_CH7	(1<<7)	Mask channel 7.

Table 5.8 Sampling Time (SST) Constants (ADC)

Constant	Value	Description
ADC_SST_CNT_MIN	5	Minimum sampling time
ADC_SST_CNT_MAX	255	Maximum sampling time
ADC_SST_CNT_DEFAULT	11	Default value of the sampling time

5.7.4 Constants for the CMT(W) Sample Driver

Table 5.9 Constants of Version Definitions CMT(W)

Constant	Value	Description
CMT_VERSION_MAJOR	1	Major part of the version number of the CMT(W) sample driver
CMT_VERSION_MINOR	0	Minor part of the version number of the CMT(W) sample driver

Table 5.10 Values of CMTW Channel Number Definitions CMT(W)

Constant	Value	Description
CMTW_CHANNEL_0	6	CMT_CHANNEL_6
CMTW_CHANNEL_1	7	CMT_CHANNEL_7

Note: For CMT_CHANNEL_6 and CMT_CHANNEL_7, see Section 5.8.2

Table 5.11 Definitions of Commands CMT(W)

Constant	Value	Description
CMT_CMD_SET_TIME_CNT	0	Timer count setting command
CMT_CMD_SET_MODE	1	Timer mode setting command
CMT_CMD_SET_PAUSE	2	Pause command
CMT_CMD_SET_RESUME	3	Resume command
CMT_CMD_SET_RESTART	4	Restart command
CMT_CMD_SET_ECM	5	ECM dynamic mode error setting command
CMT_CMD_GET_STATUS	6	State acquisition command

Table 5.12 Values of Timer Clock Division Ratio Setting Definitions CMT(W)

Constant	Value	Description
CMT_PCLK_DIV_8	0	Selects PCLKD/8 as the timer clock division ratio.
CMT_PCLK_DIV_32	1	Selects PCLKD/32 as the timer clock division ratio.
CMT_PCLK_DIV_128	2	Selects PCLKD/128 as the timer clock division ratio.
CMT_PCLK_DIV_512	3	Selects PCLKD/512 as the timer clock division ratio.
CMT_PCLK_DIV_MAX_OVER	4	For checking that settings are in the normal range

Table 5.13 Values of Timer Counter Size Setting Definitions CMT(W)

Constant	Value	Description
CMTW_CNT_SIZE_32BIT	0	Selects 32 bits as the counter size.
CMTW_CNT_SIZE_16BIT	1	Selects 16 bits as the counter size.
CMTW_CNT_SIZE_MAX_OVER	2	For checking that settings are in the normal range

Table 5.14 Values of Timer Counter Clearing Source Setting Definitions CMT(W)

Constant	Value	Description
CMT_CNT_CLEAR_COMPARE_MATCH	0	Selects compare match as the source to drive counter clearing.
CMTW_CNT_CLEAR_COMPARE_MATCH	0	Selects compare match as the source to drive counter clearing.
CMTW_CNT_CLEAR_INPUT_CAPTURE0	1	Selects input capture 0 as the source to drive counter clearing.
CMTW_CNT_CLEAR_INPUT_CAPTURE1	2	Selects input capture 1 as the source to drive counter clearing.
CMTW_CNT_CLEAR_OUTPUT_COMPARE0	3	Selects output compare 0 as the source to drive counter clearing.
CMTW_CNT_CLEAR_OUTPUT_COMPARE1	4	Selects output compare 1 as the source to drive counter clearing.
CMTW_CNT_CLEAR_NONE	5	Selects no source for counter clearing.
CMTW_CNT_CLEAR_MAX_OVER	6	For checking that settings are in the normal range

Table 5.15 Values of Output Compare Signal Output Setting Definitions CMT(W)

Constant	Value	Description
CMTW_OUT_COMPARE_VALUE_HOLD	0	The output value does not change at the time of output comparison.
CMTW_OUT_COMPARE_VALUE_0_TO_TOGGLE	1	Selects starting with the initial value 0 and toggling at the time of output comparison.
CMTW_OUT_COMPARE_VALUE_1_TO_TOGGLE	2	Selects starting with the initial value 1 and toggling at the time of output comparison.
CMTW_OUT_COMPARE_VALUE_MAX_OVER	3	For checking that settings are in the normal range

Table 5.16 Values of Input Capture Detection Trigger Setting Definitions CMT(W)

Constant	Value	Description
CMTW_IN_CAPTURE_TRIGGER_UP	0	Selects rising edges as input-capture triggers.
CMTW_IN_CAPTURE_TRIGGER_DOWN	1	Selects falling edges as input-capture triggers.
CMTW_IN_CAPTURE_TRIGGER_UP_DOWN	2	Selects both rising and falling edges as input-capture triggers.
CMTW_IN_CAPTURE_TRIGGER_MAX_OVER	3	For checking that settings are in the normal range

Table 5.17 Values of Noise Filter Clock Division Ratio Setting Definitions CMT(W)

Constant	Value	Description
CMTW_NOISE_FILTER_PCLK_DIV_1	0	Selects PCLKD/1 as the frequency division ratio of the noise filter.
CMTW_NOISE_FILTER_PCLK_DIV_8	1	Selects PCLKD/8 as the frequency division ratio of the noise filter.
CMTW_NOISE_FILTER_PCLK_DIV_32	2	Selects PCLKD/32 as the frequency division ratio of the noise filter.
CMTW_NOISE_FILTER_PCLK_DIV_64	3	Selects PCLKD/64 as the frequency division ratio of the noise filter.
CMTW_NOISE_FILTER_PCLK_DIV_MAX_OVER	4	For checking that settings are in the normal range

Table 5.18 Values of Input Capture Number Definitions CMT(W)

Constant	Value	Description
CMTW_INPUT_CAPTURE_NUM_0	0	Select input capture 0.
CMTW_INPUT_CAPTURE_NUM_1	1	Select input capture 1.
CMTW_INPUT_CAPTURE_NUM	2	Used to check that settings are in the normal range and indicates the number of input capture pins.

Table 5.19 Values of Output Compare Number Definitions CMT(W)

Constant	Value	Description
CMTW_OUTPUT_COMPARE_NUM_0	0	Selects output compare 0.
CMTW_OUTPUT_COMPARE_NUM_1	1	Selects output compare 1.
CMTW_OUTPUT_COMPARE_NUM	2	Used to check that settings are in the normal range and indicates the number of output compare pins.

Table 5.20 Values of Timer Operating State Definitions CMT(W)

Constant	Value	Description
CMTW_STATUS_STOP	0	Indicates that the timer is stopped.
CMTW_STATUS_RUNNING	1	Indicates that the timer is operating.

5.7.5 Constants for the ELC Sample Driver

Table 5.21 Values of Driver Version Definitions (ELC)

Constant	Value	Description
ELC_VERSION_MAJOR	1	Major part of the version number of the ELC sample driver
ELC_VERSION_MINOR	0	Minor part of the version number of the ELC sample driver

Table 5.22 Definitions of Commands (ELC)

Constant	Value	Description
ELC_CMD_SET_EVENT_MTU	0	MTU event link setting command
ELC_CMD_SET_EVENT_CMT	1	CMT event link setting command
ELC_CMD_SET_EVENT_DSMIF	2	ΔΣ unit event link setting command
ELC_CMD_SET_EVENT_S12AD	3	12-bit A/D converter event link setting command
ELC_CMD_SET_EVENT_INTR	4	ELC interrupt request signal event link setting command
ELC_CMD_SET_EVENT_OUT_PORT_GROUP	5	Output port group event link setting command
ELC_CMD_SET_EVENT_IN_PORT_GROUP	6	Input port group event link setting command
ELC_CMD_SET_EVENT_SINGLE_PORT	7	Single port registration and event link setting command
ELC_CMD_SET_EVENT_CMTW	8	CMTW event link setting command
ELC_CMD_SET_EVENT_TPU	9	TPU event link setting command
ELC_CMD_SET_EVENT_GPT	10	GPT event link setting command
ELC_CMD_SET_PORT_GROUP	11	Port group setting command
ELC_CMD_SET_SOFTWARE_EVENT	12	Software event issuing command
ELC_CMD_GET_PORT_GROUP_VALUE	13	Input port group value acquisition command

Table 5.23 Values of Event Link Resource Setting Definitions (ELC) (1 / 3)

Constant	Value	Description
ELC_RESOURCE_MTU0_COMPARE_MATCH_0A	0x01	Specifies compare match 0A for MTU0 as the event link source.
ELC_RESOURCE_MTU0_COMPARE_MATCH_0B	0x02	Specifies compare match 0B for MTU0 as the event link source.
ELC_RESOURCE_MTU0_COMPARE_MATCH_0C	0x03	Specifies compare match 0C for MTU0 as the event link source.
ELC_RESOURCE_MTU0_COMPARE_MATCH_0D	0x04	Specifies compare match 0D for MTU0 as the event link source.
ELC_RESOURCE_MTU0_COMPARE_MATCH_0E	0x05	Specifies compare match 0E for MTU0 as the event link source.
ELC_RESOURCE_MTU0_COMPARE_MATCH_0F	0x06	Specifies compare match 0F for MTU0 as the event link source.
ELC_RESOURCE_MTU0_OVERFLOW	0x07	Specifies the overflow of MTU0 as the event link source.
ELC_RESOURCE_MTU3_COMPARE_MATCH_3A	0x10	Specifies compare match 3A for MTU3 as the event link source.
ELC_RESOURCE_MTU3_COMPARE_MATCH_3B	0x11	Specifies compare match 3B for MTU3 as the event link source.
ELC_RESOURCE_MTU3_COMPARE_MATCH_3C	0x12	Specifies compare match 3C for MTU3 as the event link source.
ELC_RESOURCE_MTU3_COMPARE_MATCH_3D	0x13	Specifies compare match 3D for MTU3 as the event link source.
ELC_RESOURCE_MTU3_OVERFLOW	0x14	Specifies overflow of MTU3 as the event link source.
ELC_RESOURCE_MTU4_COMPARE_MATCH_4A	0x15	Specifies compare match 4A for MTU4 as the event link source.
ELC_RESOURCE_MTU4_COMPARE_MATCH_4B	0x16	Specifies compare match 4B for MTU4 as the event link source.
ELC_RESOURCE_MTU4_COMPARE_MATCH_4C	0x17	Specifies compare match 4C for MTU4 as the event link source.
ELC_RESOURCE_MTU4_COMPARE_MATCH_4D	0x18	Specifies compare match 4D for MTU4 as the event link source.
ELC_RESOURCE_MTU4_OVERFLOW	0x19	Specifies the overflow of MTU4 as the event link source.
ELC_RESOURCE_MTU4_UNDERFLOW	0x1A	Specifies the underflow of MTU4 as the event link source.
ELC_RESOURCE_CMT1_COMPARE_MATCH_1	0x1F	Specifies compare match 1 for CMT1 as the event link source.
ELC_RESOURCE_ETHER_TIMER_SYNC	0x22	Specifies the IEEE1588 timer synchronization signal from the Ethernet MAC as the event link source.

Table 5.23 Values of Event Link Resource Setting Definitions (ELC) (2 / 3)

Constant	Value	Description
ELC_RESOURCE_RIIC0_EVENT	0x4E	Specifies the communications error and event signal from RIIC0 as the event link source.
ELC_RESOURCE_RIIC0_RX_DATA_FULL	0x4F	Specifies the received data full signal from RIIC0 as the event link source.
ELC_RESOURCE_RIIC0_TX_DATA_EMPTY	0x50	Specifies the transmission data empty signal from RIIC0 as the event link source.
ELC_RESOURCE_RIIC0_TX_END	0x51	Specifies the transmission end signal from RIIC0 as the event link source.
ELC_RESOURCE_RSPI0_ERROR	0x52	Specifies the error signal from RIIC0 as the event link source.
ELC_RESOURCE_RSPI0_IDLE	0x53	Specifies the idle state of RSPI0 as the event link source.
ELC_RESOURCE_RSPI0_RX_DATA_FULL	0x54	Specifies the received data full signal from RSPI0 as the event link source.
ELC_RESOURCE_RSPI0_TX_DATA_EMPTY	0x55	Specifies the transmission data empty signal from RSPI0 as the event link source.
ELC_RESOURCE_RSPI0_TX_END	0x56	Specifies the transmission end signal from RSPI0 as the event link source.
ELC_RESOURCE_SA12AD0_CONVERT_END	0x58	Specifies the end of A/D conversion by S12AD0 as the event link source.
ELC_RESOURCE_INPUT_PORT_GROUP1	0x63	Specifies the detection of input edges for input port group 1 as the event link source.
ELC_RESOURCE_INPUT_PORT_GROUP2	0x64	Specifies the detection of input edges for input port group 2 as the event link source.
ELC_RESOURCE_SINGLE_PORT0	0x65	Specifies the detection of input edges for input single port 0 as the event link source.
ELC_RESOURCE_SINGLE_PORT1	0x66	Specifies the detection of input edges for input single port 1 as the event link source.
ELC_RESOURCE_SINGLE_PORT2	0x67	Specifies the detection of input edges for input single port 2 as the event link source.
ELC_RESOURCE_SINGLE_PORT3	0x68	Specifies the detection of input edges for input single port 3 as the event link source.
ELC_RESOURCE_ELC_SOFT_EVENT	0x69	Specifies a software event as the event link source.
ELC_RESOURCE_DOC_DATA_CALCULATE	0x6A	Specifies satisfaction of the data operation condition for the DOC as the event link source.
ELC_RESOURCE_SA12AD1_CONVERT_END	0x6C	Specifies the end of A/D conversion by S12AD1 as the event link source.
ELC_RESOURCE_CMTW0_COMPARE_MATCH	0x7E	Specifies compare match by CMTW0 as the event link source.
ELC_RESOURCE_GPT0_COMPARE_MATCH_A	0x80	Specifies compare match A for GPT0 as the event link source.
ELC_RESOURCE_GPT0_COMPARE_MATCH_B	0x81	Specifies compare match B for GPT0 as the event link source.
ELC_RESOURCE_GPT0_COMPARE_MATCH_C	0x82	Specifies compare match C for GPT0 as the event link source.
ELC_RESOURCE_GPT0_COMPARE_MATCH_D	0x83	Specifies compare match D for GPT0 as the event link source.
ELC_RESOURCE_GPT0_OVERFLOW	0x86	Specifies the overflow of GPT0 as the event link source.
ELC_RESOURCE_GPT0_UNDERFLOW	0x87	Specifies the underflow of GPT0 as the event link source.
ELC_RESOURCE_GPT1_COMPARE_MATCH_A	0x88	Specifies compare match A for GPT1 as the event link source.
ELC_RESOURCE_GPT1_COMPARE_MATCH_B	0x89	Specifies compare match B for GPT1 as the event link source.
ELC_RESOURCE_GPT1_COMPARE_MATCH_C	0x8A	Specifies compare match C for GPT1 as the event link source.
ELC_RESOURCE_GPT1_COMPARE_MATCH_D	0x8B	Specifies compare match D for GPT1 as the event link source.
ELC_RESOURCE_GPT1_OVERFLOW	0x8E	Specifies the overflow of GPT1 as the event link source.
ELC_RESOURCE_GPT1_UNDERFLOW	0x8F	Specifies the underflow of GPT1 as the event link source.
ELC_RESOURCE_GPT2_COMPARE_MATCH_A	0x90	Specifies compare match A for GPT2 as the event link source.
ELC_RESOURCE_GPT2_COMPARE_MATCH_B	0x91	Specifies compare match B for GPT2 as the event link source.

Table 5.23 Values of Event Link Resource Setting Definitions (ELC) (3 / 3)

Constant	Value	Description
ELC_RESOURCE_GPT2_COMPARE_MATCH_C	0x92	Specifies compare match C for GPT2 as the event link source.
ELC_RESOURCE_GPT2_COMPARE_MATCH_D	0x93	Specifies compare match D for GPT2 as the event link source.
ELC_RESOURCE_GPT2_OVERFLOW	0x96	Specifies the overflow of GPT2 as the event link source.
ELC_RESOURCE_GPT2_UNDERFLOW	0x97	Specifies the underflow of GPT2 as the event link source.
ELC_RESOURCE_GPT3_COMPARE_MATCH_A	0x98	Specifies compare match A for GPT3 as the event link source.
ELC_RESOURCE_GPT3_COMPARE_MATCH_B	0x99	Specifies compare match B for GPT3 as the event link source.
ELC_RESOURCE_GPT3_COMPARE_MATCH_C	0x9A	Specifies compare match C for GPT3 as the event link source.
ELC_RESOURCE_GPT3_COMPARE_MATCH_D	0x9B	Specifies compare match D for GPT3 as the event link source.
ELC_RESOURCE_GPT3_OVERFLOW	0x9E	Specifies the overflow of GPT3 as the event link source.
ELC_RESOURCE_GPT3_UNDERFLOW	0x9F	Specifies the underflow of GPT3 as the event link source.
ELC_RESOURCE_MTU6_COMPARE_MATCH_6A	0xA0	Specifies compare match 6A for MTU6 as the event link source.
ELC_RESOURCE_MTU6_COMPARE_MATCH_6B	0xA1	Specifies compare match 6B for MTU6 as the event link source.
ELC_RESOURCE_MTU6_COMPARE_MATCH_6C	0xA2	Specifies compare match 6C for MTU6 as the event link source.
ELC_RESOURCE_MTU6_COMPARE_MATCH_6D	0xA3	Specifies compare match 6D for MTU6 as the event link source.
ELC_RESOURCE_MTU6_OVERFLOW	0xA4	Specifies the overflow of MTU6 as the event link source.
ELC_RESOURCE_MTU7_COMPARE_MATCH_7A	0xA5	Specifies compare match 7A for MTU7 as the event link source.
ELC_RESOURCE_MTU7_COMPARE_MATCH_7B	0xA6	Specifies compare match 7B for MTU7 as the event link source.
ELC_RESOURCE_MTU7_COMPARE_MATCH_7C	0xA7	Specifies compare match 7C for MTU7 as the event link source.
ELC_RESOURCE_MTU7_COMPARE_MATCH_7D	0xA8	Specifies compare match 7D for MTU7 as the event link source.
ELC_RESOURCE_MTU7_OVERFLOW	0xA9	Specifies the overflow of MTU7 as the event link source.
ELC_RESOURCE_MTU7_UNDERFLOW	0xAA	Specifies the underflow of MTU7 as the event link source.
ELC_RESOURCE_TPU0_COMPARE_MATCH_A	0xAC	Specifies compare match A for TPU0 as the event link source.
ELC_RESOURCE_TPU0_COMPARE_MATCH_B	0xAD	Specifies compare match B for TPU0 as the event link source.
ELC_RESOURCE_TPU0_COMPARE_MATCH_C	0xAE	Specifies compare match C for TPU0 as the event link source.
ELC_RESOURCE_TPU0_COMPARE_MATCH_D	0xAF	Specifies compare match D for TPU0 as the event link source.
ELC_RESOURCE_TPU0_OVERFLOW	0xB0	Specifies the overflow of TPU0 as the event link source.
ELC_RESOURCE_TPU1_COMPARE_MATCH_A	0xB1	Specifies compare match A for TPU1 as the event link source.
ELC_RESOURCE_TPU1_COMPARE_MATCH_B	0xB2	Specifies compare match B for TPU1 as the event link source.
ELC_RESOURCE_TPU1_OVERFLOW	0xB3	Specifies the overflow of TPU1 as the event link source.
ELC_RESOURCE_TPU1_UNDERFLOW	0xB4	Specifies the underflow of TPU1 as the event link source.
ELC_RESOURCE_TPU2_COMPARE_MATCH_A	0xB5	Specifies compare match A for TPU2 as the event link source.
ELC_RESOURCE_TPU2_COMPARE_MATCH_B	0xB6	Specifies compare match B for TPU2 as the event link source.
ELC_RESOURCE_TPU2_OVERFLOW	0xB7	Specifies the overflow of TPU2 as the event link source.
ELC_RESOURCE_TPU2_UNDERFLOW	0xB8	Specifies the underflow of TPU2 as the event link source.
ELC_RESOURCE_TPU3_COMPARE_MATCH_A	0xB9	Specifies compare match A for TPU3 as the event link source.
ELC_RESOURCE_TPU3_COMPARE_MATCH_B	0xBA	Specifies compare match B for TPU3 as the event link source.
ELC_RESOURCE_TPU3_COMPARE_MATCH_C	0xBB	Specifies compare match C for TPU3 as the event link source.
ELC_RESOURCE_TPU3_COMPARE_MATCH_D	0xBC	Specifies compare match D for TPU3 as the event link source.
ELC_RESOURCE_TPU3_OVERFLOW	0xBD	Specifies the overflow of TPU3 as the event link source.

Table 5.24 Values of MTU Channel Number Definitions (ELC)

Constant	Value	Description
ELC_MTU_CH_0	0	Indicates channel 0 of the MTU.
ELC_MTU_CH_3	1	Indicates channel 3 of the MTU.
ELC_MTU_CH_4	2	Indicates channel 4 of the MTU.
ELC_MTU_CH_NUM	3	For checking that settings are in the normal range

Table 5.25 Values of MTU Event Link Operation Setting Definitions (ELC)

Constant	Value	Description
ELC_MTU_COUNT_START	0	The MTU starts counting in response to the event link signal.
ELC_MTU_COUNT_RESTART	1	The MTU restarts counting in response to the event link signal.
ELC_MTU_INPUT_CAPTURE	2	The MTU executes input capture in response to the event link signal.
ELC_MTU_MAX_OVER	3	For checking that settings are in the normal range

Table 5.26 Values of CMT Event Link Operation Setting Definitions (ELC)

Constant	Value	Description
ELC_CMT_COUNT_START	0	The CMT starts counting in response to the event link signal.
ELC_CMT_COUNT_RESTART	1	The CMT restarts counting in response to the event link signal.
ELC_CMT_COUNT_INCREMENT	2	The CMT counter is incremented in response to the event link signal.
ELC_CMT_MAX_OVER	3	For checking that settings are in the normal range

Table 5.27 Values of ΔΣ Unit Channel Trigger Number Definitions (ELC)

Constant	Value	Description
ELC_DSMIF0_TRIGGER_0	0	Indicates trigger 0 of ΔΣ unit 0.
ELC_DSMIF0_TRIGGER_1	1	Indicates trigger 1 of ΔΣ unit 0.
ELC_DSMIF1_TRIGGER_0	2	Indicates trigger 0 of ΔΣ unit 1.
ELC_DSMIF1_TRIGGER_1	3	Indicates trigger 1 of ΔΣ unit 1.
ELC_DSMIF_TRIGGER_NUM	4	For checking that settings are in the normal range

Table 5.28 Values of 12-Bit A/D Converter Channel Number Definitions (ELC)

Constant	Value	Description
ELC_S12AD_CH_0	0	Indicates channel 0 of the 12-bit A/D converter.
ELC_S12AD_CH_1	1	Indicates channel 1 of the 12-bit A/D converter.
ELC_S12AD_CH_NUM	2	For checking that settings are in the normal range

Table 5.29 Values of Interrupt Number Definitions (ELC)

Constant	Value	Description
ELC_INTR_NUM_1	0	Indicates interrupt request signal 1.
ELC_INTR_NUM_2	1	Indicates interrupt request signal 2.
ELC_INTR_NUM	2	For checking that settings are in the normal range

Table 5.30 Values of Port Group Number Definitions (ELC)

Constant	Value	Description
ELC_PORT_GROUP_NUM_0	0	Indicates output port group number 0.
ELC_PORT_GROUP_NUM_1	1	Indicates output port group number 1.
ELC_PORT_GROUP_NUM	2	For checking that settings are in the normal range

Table 5.31 Values of Output Port Group Event Link Operation Setting Definitions (ELC)

Constant	Value	Description
ELC_OUT_GROUP_OUTPUT_0	0	The output group port outputs 0 in response to the event link signal.
ELC_OUT_GROUP_OUTPUT_1	1	The output group port outputs 1 in response to the event link signal.
ELC_OUT_GROUP_TOGGLE	2	The output port group outputs a toggle value in response to the event link signal.
ELC_OUT_GROUP_BUFFER	3	The buffer value is output in response to the event link signal.
ELC_OUT_GROUP_ROTATE	4	The rotated buffer value is output in response to the event link signal.
ELC_OUT_GROUP_MAX_OVER	5	For checking that settings are in the normal range

Table 5.32 Values of Single Port Number Definitions (ELC)

Constant	Value	Description
ELC_SINGLE_PORT_NUM_0	0	Indicates single port number 0.
ELC_SINGLE_PORT_NUM_1	1	Indicates single port number 1.
ELC_SINGLE_PORT_NUM_2	2	Indicates single port number 2.
ELC_SINGLE_PORT_NUM_3	3	Indicates single port number 3.
ELC_SINGLE_PORT_NUM	4	For checking that settings are in the normal range

Table 5.33 Values of Single Port Event Link Operation Setting Definitions (ELC)

Constant	Value	Description
ELC_SINGLE_PORT_OUTPUT_0	0	0 is output in response to the event link signal.
ELC_SINGLE_PORT_OUTPUT_1	1	1 is output in response to the event link signal.
ELC_SINGLE_PORT_TOGGLE	2	A toggle value is output in response to the event link signal.
ELC_SINGLE_PORT_ACTION_MAX_OVER	3	For checking that settings are in the normal range

Table 5.34 Values of CMTW Event Link Operation Setting Definitions (ELC)

Constant	Value	Description
ELC_CMTW_COUNT_START	0	The CMTW starts counting in response to the event link signal.
ELC_CMTW_COUNT_RESTART	1	The CMTW restarts counting in response to the event link signal.
ELC_CMTW_COUNT_INCREMENT	2	The CMTW counter is incremented in response to the event link signal.
ELC_CMTW_COUNT_MAX_OVER	3	For checking that settings are in the normal range

Table 5.35 Values of TPU Channel Number Definitions (ELC)

Constant	Value	Description
ELC_TPU_CH_0	0	Indicates channel 0 of the TPU.
ELC_TPU_CH_1	1	Indicates channel 1 of the TPU.
ELC_TPU_CH_2	2	Indicates channel 2 of the TPU.
ELC_TPU_CH_3	3	Indicates channel 3 of the TPU.
ELC_TPU_CH_NUM	4	For checking that settings are in the normal range

Table 5.36 Values of TPU Event Link Operation Setting Definitions (ELC)

Constant	Value	Description
ELC_TPU_COUNT_START	0	The TPU starts counting in response to the event link signal.
ELC_TPU_COUNT_RESTART	1	The TPU restarts counting in response to the event link signal.
ELC_TPU_INPUT_CAPTURE	2	The TPU executes input capture in response to the event link signal.
ELC_TPU_MAX_OVER	3	For checking that settings are in the normal range

Table 5.37 Values of GPT Channel Number Definitions (ELC)

Constant	Value	Description
ELC_GPT_CH_0	0	Indicates channel 0 of the GPT.
ELC_GPT_CH_1	1	Indicates channel 1 of the GPT.
ELC_GPT_CH_2	2	Indicates channel 2 of the GPT.
ELC_GPT_CH_3	3	Indicates channel 3 of the GPT.
ELC_GPT_CH_NUM	4	For checking that settings are in the normal range

Table 5.38 Values of GPT Event Link Operation Setting Definitions (ELC)

Constant	Value	Description
ELC_GPT_COUNT_START	0	The GPT starts counting in response to the event link signal.
ELC_GPT_COUNT_RESTART	1	The GPT restarts counting in response to the event link signal.
ELC_GPT_COUNT_STOP	2	The GPT stops counting in response to the event link signal.
ELC_GPT_INPUT_CAPTURE	3	The GPT executes input capture in response to the event link signal.
ELC_GPT_MAX_OVER	4	For checking that settings are in the normal range

Table 5.39 Values of I/O Port Group Symbol Setting Definitions (ELC)

Constant	Value	Description
ELC_PORT_B	1	Specifies PORTB.
ELC_PORT_E	2	Specifies PORTE.
ELC_PORT_NUM	3	For checking that settings are in the normal range

Table 5.40 Values of Input Port Group Event Detection Trigger Setting Definitions (ELC)

Constant	Value	Description
ELC_PORT_GROUP_TRIGGER_UP	0	Selects rising edges as input port group triggers.
ELC_PORT_GROUP_TRIGGER_DOWN	1	Selects falling edges as input port group triggers.
ELC_PORT_GROUP_TRIGGER_UP_DOWN	2	Selects both rising and falling edges as input port group triggers.
ELC_PORT_GROUP_MAX_OVER	3	For checking that settings are in the normal range

Table 5.41 Values of Single Port Event Detection Trigger Setting Definitions (ELC)

Constant	Value	Description
ELC_SINGLE_EVENT_INPUT	0	Selects the input of an event signal.
ELC_SINGLE_EVENT_OUTPUT	1	Selects the output of an event signal.
ELC_SINGLE_EVENT_MAX_OVER	2	For checking that settings are in the normal range

Table 5.42 Values of Single Port Event Detection Trigger Setting Definitions (ELC)

Constant	Value	Description
ELC_SINGLE_PORT_TRIGGER_UP	0	Selects rising edges as input single port triggers.
ELC_SINGLE_PORT_TRIGGER_DOWN	1	Selects falling edges as input single port triggers.
ELC_SINGLE_PORT_TRIGGER_UP_DOWN	2	Selects both rising and falling edges as input single port triggers.
ELC_SINGLE_PORT_TRIGGER_MAX_OVER	3	For checking that settings are in the normal range

5.8 Structures, Unions, and Enumerations

The structures, unions, and enumerations are listed below.

5.8.1 Structures, Unions, and Enumerations for the ADC Sample Driver

The structures, unions, and enumerations for the ADC sample driver are listed below.

```
/* ADC_OPEN() ARGUMENT DEFINITIONS */
typedef enum e_adc_mode
{
    ADC_MODE_SS_TEMPERATURE,                /* single scan temperature sensor */
    ADC_MODE_SS_ONE_CH,                     /* single scan one channel */
    ADC_MODE_SS_ONE_CH_DBLTRIG,             /* on even triggers save to ADDBLDR &
interrupt */
    ADC_MODE_SS_MULTI_CH,                /* 1 trigger source, scan multiple channels */
    ADC_MODE_SS_MULTI_CH_GROUPED,         /* 2 trigger sources, scan multiple channels */
    ADC_MODE_SS_MULTI_CH_GROUPED_DBLTRIG_A,
    ADC_MODE_CONT_ONE_CH,                   /* continuous scan one channel */
    ADC_MODE_CONT_MULTI_CH,                 /* continuous scan multiple channels */
    ADC_MODE_MAX
} adc_mode_t;
/* trigger sources */
typedef enum e_adc_trig
{
    ADC_TRIG_ADTRG0                 = 0,
    ADC_TRIG_TRGA0N                 = 1,
    ADC_TRIG_TRGA1N                 = 2,
    ADC_TRIG_TRGA2N                 = 3,
    ADC_TRIG_TRGA3N                 = 4,
    ADC_TRIG_TRGA4N                 = 5,
    ADC_TRIG_TRGA6N                 = 6,
    ADC_TRIG_TRGA7N                 = 7,
    ADC_TRIG_TRG0N                  = 8,
    ADC_TRIG_TRG4AN                 = 9,
    ADC_TRIG_TRG4BN                 = 10,
    ADC_TRIG_TRG4AN_OR_TRG4BN       = 11,
    ADC_TRIG_TRG4ABN                = 12,
    ADC_TRIG_TRG7AN                 = 13,
    ADC_TRIG_TRG7BN                 = 14,
    ADC_TRIG_TRG7AN_OR_TRG7BN       = 15,
    ADC_TRIG_TRG7ABN                = 16,
    ADC_TRIG_GTADTRA0N              = 17,
    ADC_TRIG_GTADTRB0N              = 18,
```

```
    ADC_TRIG_GTADTRA1N              = 19,
    ADC_TRIG_GTADTRB1N              = 20,
    ADC_TRIG_GTADTRA2N              = 21,
    ADC_TRIG_GTADTRB2N              = 22,
    ADC_TRIG_GTADTRA3N              = 23,
    ADC_TRIG_GTADTRB3N              = 24,
    ADC_TRIG_GTADTRA0N_OR_GTADTRB0N = 25,
    ADC_TRIG_GTADTRA1N_OR_GTADTRB1N = 26,
    ADC_TRIG_GTADTRA2N_OR_GTADTRB2N = 27,
    ADC_TRIG_GTADTRA3N_OR_GTADTRB3N = 28,
    ADC_TRIG_TPTRGAN_0              = 31,
    ADC_TRIG_TPTRG0AN_0             = 32,
    ADC_TRIG_TPTRGAN_1              = 33,
    ADC_TRIG_TPTRG6AN_1             = 34,
    ADC_TRIG_ELCTRG0                = 48,
    ADC_TRIG_SOFTWARE               = 63
} adc_trig_t;
typedef enum e_adc_add
{
    ADC_ADD_OFF           = 0,      /* addition is turned off for chans/sensors */
    ADC_ADD_TWO_SAMPLES,
    ADC_ADD_THREE_SAMPLES,
    ADC_ADD_FOUR_SAMPLES,
    ADC_ADD_MAX
} adc_add_t;

typedef enum e_adc_align
{
    ADC_ALIGN_RIGHT = 0x0000,
    ADC_ALIGN_LEFT  = 0x8000
} adc_align_t;

typedef enum e_adc_clear
{
    ADC_CLEAR_AFTER_READ_OFF = 0x0000,
    ADC_CLEAR_AFTER_READ_ON  = 0x0020
} adc_clear_t;

typedef struct st_adc_cfg
{
    adc_add_t       add_cnt;
    adc_align_t     alignment;          /* ignored if addition used */
```

```
    adc_clear_t     clearing;
    adc_trig_t      trigger;            /* default and Group A trigger source */
    adc_trig_t      trigger_groupb;     /* valid only for group modes */
    uint8_t         priority;           /* S12ADIO interrupt priority; 0-15 */
    uint8_t         priority_groupb;    /* GBADI interrupt priority; 0-15 */
} adc_cfg_t;
/* CALLBACK FUNCTION ARGUMENT DEFINITIONS */
/* callback function events */
typedef enum e_adc_cb_evt
{
    ADC_EVT_SCAN_COMPLETE,              /* normal/Group A scan complete */
    ADC_EVT_SCAN_COMPLETE_GROUPB        /* Group B scan complete */
} adc_cb_evt_t;

typedef struct st_adc_cb_args           /* callback arguments */
{
    adc_cb_evt_t    event;
} adc_cb_args_t;

/* ADC_CONTROL() ARGUMENT DEFINITIONS */
typedef enum e_adc_cmd
{
    ADC_CMD_ENABLE_CHANS           = 1, /* enables chans and INT(s) if priority != 0 */
    ADC_CMD_ENABLE_TEMP_SENSOR,         /* enables sensor and INT if priority != 0 */
    ADC_CMD_SET_SAMPLE_STATE_CNT,
    ADC_CMD_ENABLE_TRIG,                /* allows an async/sync trigger to start scan */
    ADC_CMD_DISABLE_TRIG,              /* prevents an async/sync trigger to start scan */
    ADC_CMD_SCAN_NOW,                   /* issue software trigger */
    ADC_CMD_DISABLE_INT,                /* interrupt disable; ADCSR.ADIE=0 */
    ADC_CMD_ENABLE_INT,                 /* interrupt enable;  ADCSR.ADIE=1 */
    ADC_CMD_DISABLE_INT_GROUPB,         /* interrupt disable; ADCSR.GBADIE=0 */
    ADC_CMD_ENABLE_INT_GROUPB,          /* interrupt enable;  ADCSR.GBADIE=1 */
    ADC_CMD_CHECK_SCAN_DONE,            /* for Normal, GroupA or GroupB scan */
    ADC_CMD_CHECK_SCAN_DONE_GROUPA,
    ADC_CMD_CHECK_SCAN_DONE_GROUPB,
    ADC_CMD_MAX
} adc_cmd_t;

/* for ADC_CMD_ENABLE_CHANS */
typedef struct st_adc_ch_cfg            /* bit 0 is ch0; bit 7 is ch7 */
{
    uint32_t        chan_mask;          /* channels/bits 0-7 */
```

```
    uint32_t        chan_mask_groupb;   /* valid for group modes */
    uint32_t        add_mask;           /* valid if add enabled in Open() */
} adc_ch_cfg_t;
/* for ADC_CMD_SET_SAMPLE_STATE_CNT */
/* sample state registers */
typedef enum e_adc_sst_reg
{
    ADC_SST_CH0 = 0,
    ADC_SST_CH1,
    ADC_SST_CH2,
    ADC_SST_CH3,
    ADC_SST_CH4,
    ADC_SST_CH5,
    ADC_SST_CH6,
    ADC_SST_CH7,
    ADC_SST_TEMPERATURE,
    ADC_SST_NUM_REGS
} adc_sst_reg_t;

typedef struct st_adc_time
{
    adc_sst_reg_t   reg_id;
    uint8_t         num_states;         /* default=11 */
} adc_time_t;
/* ADC_READ() ARGUMENT DEFINITIONS */
typedef enum e_adc_reg
{
    ADC_REG_CH0 = 0,
    ADC_REG_CH1 = 1,
    ADC_REG_CH2 = 2,
    ADC_REG_CH3 = 3,
    ADC_REG_CH4 = 4,
    ADC_REG_CH5 = 5,
    ADC_REG_CH6 = 6,
    ADC_REG_CH7 = 7,
    ADC_REG_TEMP = 8,
    ADC_REG_DBLTRIG = 9,
    ADC_REG_MAX
} adc_reg_t;
/* ADC_READALL() ARGUMENT DEFINITIONS */
typedef struct st_adc_data
{
```

```
    uint16_t    chan[8];
    uint16_t    dbltrig;
} adc_data_t;
```

Figure 5.2 Structures, Unions, and Enumerations for the ADC Sample Driver

5.8.2 Structures, Unions, and Enumerations for the CMT(W) Sample Driver

The structures, unions, and enumerations for the CMT(W) sample driver are listed below.

```
typedef enum e_cmt_channel
{
    CMT_CHANNEL_0,
    CMT_CHANNEL_1,
    CMT_CHANNEL_2,
    CMT_CHANNEL_3,
    CMT_CHANNEL_4,
    CMT_CHANNEL_5,
    CMT_CHANNEL_6,          /* CMTW */
    CMT_CHANNEL_7,          /* CMTW */
    CMT_CHANNEL_MAX
} cmt_channel_t;

/*-----------------------------------------------------------------------------*/
/*   Define cmt(w) timer count setting structure.                              */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t                pclk_div;
    uint32_t                cnt_size;
    uint32_t                clear_factor;
} cmt_time_cnt_t;
#define cmtw_time_cnt_t     cmt_time_cnt_t

/*-----------------------------------------------------------------------------*/
/*   Define cmt(w) compare match setting structure.                            */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    int32_t                 mode_enable;
    uint32_t                compare_match_cnt;
    int32_t                 intr_priority;
    void                    (* p_callback)(void);
} cmt_compare_match_t;
#define cmtw_compare_match_t    cmt_compare_match_t
```

```
/*-------------------------------------------------------------------------*/
/*   Define cmt(w) output compare setting structure.                       */
/*-------------------------------------------------------------------------*/
typedef struct
{
    int32_t                 mode_enable;
    uint32_t                output_compare_cnt;
    uint32_t                output_signal;
    int32_t                 intr_priority;
    void                    (* p_callback)(void);
} cmtw_output_compare_t;

/*-------------------------------------------------------------------------*/
/*   Define cmt(w) input capture setting structure.                        */
/*-------------------------------------------------------------------------*/
typedef struct
{
    int32_t                 mode_enable;
    uint32_t                trigger;
    int32_t                 filter_enable;
    int32_t                 intr_priority;
    void                    (* p_callback)(uint32_t cnt_value);
} cmtw_input_capture_t;

/*-------------------------------------------------------------------------*/
/*   Define ECM setting structure.                                         */
/*-------------------------------------------------------------------------*/
typedef struct
{
    int32_t                 ecm_enable;
    uint32_t                output_compare_num;
} cmtw_ecm_t;
```

```
/*-----------------------------------------------------------------------------*/
/*   Define cmt(w) operation mode setting structure.                           */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    cmt_compare_match_t     compare_match;
    cmtw_output_compare_t   output_compare[CMTW_OUTPUT_COMPARE_NUM];
    cmtw_input_capture_t    input_capture[CMTW_INPUT_CAPTURE_NUM];
    uint32_t                noise_filter_clk;
} cmt_mode_t;



/*-----------------------------------------------------------------------------*/
/*   Define cmt(w) configuration structure.                                    */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    cmt_time_cnt_t          time_cnt_param;
    cmt_mode_t              mode_param;
} cmt_cfg_t;
```

Figure 5.3 Structures, Unions, and Enumerations for the CMT(W) Sample Driver

5.8.3 Structures, Unions, and Enumerations for the ELC Sample Driver

The structures, unions, and enumerations for the ELC sample driver are listed below.

```
/*-------------------------------------------------------------------------*/
/*   Define MTU parameter for event link.                                  */
/*-------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_mtu_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_mtu_t;

/*-------------------------------------------------------------------------*/
/*   Define CMT parameter for event link.                                  */
/*-------------------------------------------------------------------------*/
typedef struct
{
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_cmt_t;

/*-------------------------------------------------------------------------*/
/*   Define delta sigma unit parameter for event link.                     */
/*-------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_dsmif_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
}elc_cmd_dsmif_t;
```

```
/*-----------------------------------------------------------------------------*/
/*   Define 12bit A/D converter parameter for event link.                      */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_s12ad_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
}elc_cmd_s12ad_t;



/*-----------------------------------------------------------------------------*/
/*   Define ELC interrupt for event link.                                      */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_intr_num;
    int32_t         event_link_enable;
    uint32_t        resource;
    int32_t         intr_priority;
    void            (* p_callback)(void);
}elc_cmd_intr_t;

/*-----------------------------------------------------------------------------*/
/*   Define output port group parameter for event link.                        */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_out_port_group_num;
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
    uint8_t         init_value;
}elc_cmd_out_port_group_t;
```

```
/*-------------------------------------------------------------------------*/
/*   Define input port group parameter for event link.                     */
/*-------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_in_port_group_num;
    int32_t         event_link_enable;
    uint8_t         resource;
    int32_t         overwrite_enable;
}elc_cmd_in_port_group_t;









/*-------------------------------------------------------------------------*/
/*   Define input port group parameter for event link.                     */
/*-------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_single_port_num;
    uint32_t        port_symbol;
    uint32_t        port_num;
    int32_t         event_link_enable;
    uint32_t        event_direction;
    uint32_t        resource;
    uint32_t        output_action;
    uint32_t        input_trigger;
}elc_cmd_single_port_t;
```

```
/*-----------------------------------------------------------------------------*/
/*   Define CMTW parameter for event link.                                     */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_cmtw_t;

/*-----------------------------------------------------------------------------*/
/*   Define TPU parameter for event link.                                      */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_tpu_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_tpu_t;

/*-----------------------------------------------------------------------------*/
/*   Define GPT parameter for event link.                                      */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_gpt_ch;
    int32_t         event_link_enable;
    uint32_t        resource;
    uint32_t        action;
}elc_cmd_gpt_t;
/*-----------------------------------------------------------------------------*/
/*   Define port group parameter for port group setting.                       */
/*-----------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        port_group_num;
    uint8_t         port_group_bit;
    uint32_t        trigger;
}elc_cmd_port_group_t;
```

```
/*-------------------------------------------------------------------------*/
/*   Define port group parameter for port group value get.                 */
/*-------------------------------------------------------------------------*/
typedef struct
{
    uint32_t        elc_port_group_num;
    uint8_t         port_value;
}elc_get_port_value_t;
```

Figure 5.4 Structures, Unions, and Enumerations for the ELC Sample Driver

5.9 Global Variables

The table below lists the global variables.

Table 5.43 Global Variables

Type	Variable	Description	Function Used
static volatile int32_t	gb_cmtw_end_flag	Callback information of the CMT(W) sample driver	cmtw_elc_sample_cmtwi_callback
static volatile int32_t	gb_adc_end_flag	Callback information of the ADC sample driver	cmtw_elc_sample_adc_callback

5.10 Functions

The table below lists the functions.

Table 5.44 List of Functions

5.11 Specifications of the Functions of the Sample Application

The specifications of the functions of the sample application code are listed below.

5.11.1 main

main	
Synopsis	Periodical A/D conversion of the voltage input from a potentiometer
Header	—
Declaration	int32_t main(void);
Description	This function handles the following processing. It initializes the following drivers. • Serial • CMT(W) • ELC • ADC CMTW0 is linked with the ADC by the ELC so that the results of A/D conversion of the voltage input from the potentiometer that is connected on the evaluation board can be output through a USB serial port in the cycle for compare-matches in CMTW0.
Arguments	—
Return value	Return from this function does not proceed because it is an endless loop.
Remarks	The ADC driver produces activation by a synchronous trigger in single scan mode.

5.11.2 sample_setup

sample_setup	
Synopsis	Setting the CMT(W), ELC, and ADC drivers
Header	—
Declaration	static int32_t sample_setup(void);
Description	This function handles the following processing. The ADC driver is activated in response to an ELC trigger in single scan mode. The ELC driver is activated to link the CMTW0 compare match and ADC events. The CMT(W) driver is activated in response to matches in comparison by CMTW0 which occur at 3-second intervals.
Arguments	—
Return values	Error codes ESUCCESS: Success EINVAL: Invalid input parameters EACESS: Invalid access EFAULT: Failure EBUSY: Busy For other error codes, see the list of error codes.
Remarks	—

5.11.3 sample_process

sample_process	
Synopsis	Outputs the results from the ADC through the USB serial port after the completion of A/D conversion.
Header	—
Declaration	static void sample_process(void);
Description	This function waits for the event of A/D conversion being completed following the detection of the event signal from CMTW0. After the detecting the event of A/D conversion being completed, it outputs the A/D-converted value through USB serial communications.
Arguments	—
Return values	—
Remarks	—

5.11.4 sample_release

sample_release	
Synopsis	Ending and releasing the CMT(W), ELC, and ADC drivers
Header	—
Declaration	static void sample_release(void);
Description	This function ends operation of the CMT(W), ELC, and ADC drivers and stops the corresponding modules.
Arguments	—
Return values	—
Remarks	—

5.11.5 cmtw_ch0_inhr_callback

cmtw_ch0_inhr_callback	
Synopsis	Interrupt handler for CMTW0
Header	—
Declaration	static void cmtw_ch0_inhr_callback(void);
Description	This callback function indicates detection of a compare-match interrupt from CMTW0 after the timer has been set up by R_CMT_Open. Calling this function conveys detection of the CMTW0 compare-match event signal to the application.
Arguments	—
Return values	—
Remarks	—

5.11.6 adc_inhr_callback

adc_inhr_callback	
Synopsis	Interrupt handler for completion of A/D conversion
Header	—
Declaration	static void adc_inhr_callback(int32_t event);
Description	This callback function indicates detection of an A/D conversion completed interrupt set by R_ADC_Open. Calling this function conveys completion of A/D conversion to the application.
Arguments	—
Return values	—
Remarks	—

5.12 Specifications of the Functions of the ADC Sample Driver

The specifications of the functions of the ADC sample driver are listed below.

5.12.1 R_ADC_Init

R_ADC_Init	
Synopsis	Initializing the ADC driver
Header	r_adc_if.h
Declaration	void R_ADC_Init(void);
Description	This function initializes the ADC driver. A semaphore for exclusive control of the ADC driver interface is raised.
Arguments	—
Return values	—
Remarks	—

5.12.2 R_ADC_Uninit

R_ADC_Uninit	
Synopsis	Ending operation of the ADC driver
Header	r_adc_if.h
Declaration	int32_t R_ADC_Uninit(void);
Description	This function ends the ADC driver. The ADC module is stopped and the semaphore for exclusive control of the interface is lowered.
Arguments	—
Return values	Error codes: ESUCCESS, EACCES, EFAULT
Remarks	—

5.12.3 R_ADC_Open

R_ADC_Open		
Synopsis	Setting to start operation of the ADC	
Header	r_adc_if.h	
Declaration	int32_t R_ADC_Open(adc_mode_t const mode, adc_cfg_t * const p_cfg, void (* const p_callback)(int32_t event));	
Description	This function makes settings to start operation of the ADC.	
Arguments	mode	ADC operation mode
	p_cfg	Pointer to the ADC operation setting information
	p_callback	A/D conversion completion event callback
Return values	Error codes: ESUCCESS, EBUSY, EACESS, EFAULT	
Remarks	Call function R_ADC_Init beforehand to initialize the ADC driver. After the completion of the operations required of the ADC module, call function R_ADC_Close to end operation of the ADC.	

5.12.4 R_ADC_Close

R_ADC_Close	
Synopsis	Ending operation of the ADC module
Header	r_adc_if.h
Declaration	int32_t R_ADC_Close(void);
Description	This function stops operation of the ADC module.
Arguments	—
Return values	Error codes: ESUCCESS, EACESS, EFAULT
Remarks	Call function R_ADC_Open to start operation of the ADC module.

5.12.5 R_ADC_Control

R_ADC_Control		
Synopsis	Setting the operations required of the ADC module	
Header	r_adc_if.h	
Declaration	int32_t R_ADC_Control(adc_cmd_t const cmd, void * const p_args);	
Description	This function sets the operations required of the ADC module. For details of the ADC command, see Section 5.16, Specifications of the R_ADC_Control Commands.	
Arguments	cmd	ADC control command
	p_args	Pointer to the ADC command parameter information
Return values	Error codes: ESUCCESS, EACESS, EFAULT	
Remarks	Call function R_ADC_Open to start operation of the ADC module.	

5.12.6 R_ADC_Read

R_ADC_Read		
Synopsis	Reading the A/D-converted value from a channel	
Header	r_adc_if.h	
Declaration	int32_t R_ADC_Read(adc_reg_t const reg_id, uint16_t * const p_data);	
Description	This function reads the A/D-converted value from a specified channel.	
Arguments	reg_id	Channel for reading the A/D-converted value
	p_data	Pointer to the variable at the destination for storing the result of A/D conversion
Return values	Error codes: ESUCCESS, EINVAL, EACESS, EFAULT	
Remarks	Call function R_ADC_Open to start operation of the ADC module.	

5.12.7 R_ADC_ReadAll

R_ADC_ReadAll		
Synopsis	Reading the A/D-converted values from all channels	
Header	r_adc_if.h	
Declaration	int32_t R_ADC_ReadAll(adc_data_t * const p_all_data);	
Description	This function reads the A/D-converted values from all channels.	
Arguments	p_all_data	Pointer to the variable at the destination for storing the result of A/D conversion
Return values	Error codes: ESUCCESS, EINVAL, EACESS, EFAULT	
Remarks	Call function R_ADC_Open to start operation of the ADC module.	

5.12.8 R_ADC_GetVersion

R_ADC_GetVersion	
Synopsis	Acquiring the version number of the ADC driver
Header	r_adc_if.h
Declaration	int32_t R_ADC_GetVersion(void);
Description	This function returns the version number of the ADC driver.
Arguments	—
Return values	Version number of the ADC driver Bits 0 to 15: Minor part of the version number Bits 16 to 31: Major part of the version number
Remarks	—

5.13 Specifications of the Functions of the CMT(W) Sample Driver

The specifications of the functions of the CMT(W) sample driver are listed below.

5.13.1 R_CMT_Init

R_CMT_Init	
Synopsis	Initializing the CMT(W) driver
Header	r_cmt_if.h
Declaration	void R_CMT_Init(void);
Description	This function initializes the CMT(W) driver. A semaphore for exclusive control of the CMT(W) driver interface is raised.
Arguments	—
Return values	—
Remarks	—

5.13.2 R_CMT_Uninit

R_CMT_Uninit	
Synopsis	Ending operation of the CMT(W) driver
Header	r_cmt_if.h
Declaration	int32_t R_CMT_Uninit(void);
Description	This function ends the CMT(W) driver. The CMT(W) module is stopped and the semaphore for exclusive control of the interface is lowered.
Arguments	—
Return values	Error codes: ESUCCESS, EINVAL (an error occurred during the call of CMT_Close or CMTW_Close), EFAULT
Remarks	—

5.13.3 R_CMT_Open

R_CMT_Open		
Synopsis	Setting to start operation of a CMT(W) channel	
Header	r_cmt_if.h	
Declaration	int32_t R_CMT_Open(cmt_channel_t const channel, cmt_cfg_t * const p_cfg);	
Description	This function makes settings to start operation of a CMT(W) channel. Specifically, it sets starting of the specified channel number in the corresponding CMT or CMTW. The following arguments specify the channel numbers. CMT_CHANNEL_0 to CMT_CHANNEL_5 for the compare-match timers (CMTs) CMT_CHANNEL_6 to CMT_CHANNEL_7 for the compare-match timer Ws (CMTWs)	
Arguments	channel	Channel numbers CMT_CHANNEL_0 to CMT_CANNEL_7
	p_cfg	Pointer to the initial CMT(W) operation setting information
Return values	Error codes: ESUCCESS, EINVAL, EACESS, EBUSY, EFAULT	
Remarks	Call function R_CMT_Init to initialize the CMT(W) driver. After the completion of the operations required of the CMT(W) module, call function R_CMT_Close to end operation of the CMT(W).	

5.13.4 R_CMT_Close

R_CMT_Close		
Synopsis	Ending operation of the CMT(W) module	
Header	r_cmt_if.h	
Declaration	int32_t R_CMT_Close(cmt_channel_t const channel);	
Description	This function stops operation of the specified channel of the CMT or CMTW. Specifically, it stops operation of the CMT or CMTW of the specified channel number. The following arguments specify the channel numbers. CMT_CHANNEL_0 to CMT_CHANNEL_5 for the compare-match timers (CMTs) CMT_CHANNEL_6 to CMT_CHANNEL_7 for the compare-match timer Ws (CMTWs)	
Arguments	channel	Channel numbers CMT_CHANNEL_0 to CMT_CANNEL_7
Return values	Error codes: ESUCCESS, EINVAL, EACESS, EBUSY, EFAULT	
Remarks	Call function R_CMT_Open to start operation of the CMT(W) module.	

5.13.5 R_CMT_Control

R_CMT_Control		
Synopsis	Setting a CMT(W) module	
Header	r_cmt_if.h	
Declaration	int32_t R_CMT_Control(cmt_channel_t const channel, const uint32_t cmd, void * const p_data);	
Description	This function makes settings for a CMT(W) module. It makes settings corresponding to the specified command. For details, see Section 5.17, Specifications of the R_CMT_Control Commands.	
Arguments	channel	Channel numbers CMT_CHANNEL_0 to CMT_CANNEL_7
	cmd	Changes the command for making module settings.
	p_data	Sets parameters for making module settings
Return values	Error codes: ESUCCESS, EACESS, EBUSY, EFAULT	
Remarks	Call function R_CMT_Open to start operation of the CMT(W) module.	

5.13.6 R_CMT_StartPeriodic

R_CMT_StartPeriodic		
Synopsis	Function for starting cyclic operation of the timer counter	
Header	r_cmt_if.h	
Declaration	int32_t R_CMT_StartPeriodic(cmt_channel_t const channel);	
Description	This function starts cyclic operation of a timer counter.	
Arguments	channel	Channel numbers CMT_CHANNEL_0 to CMT_CANNEL_7
Return values	Error codes: ESUCCESS, EINVAL, EACESS, EBUSY, EFAULT	
Remarks	Call function R_CMT_Open to start operation of the CMT(W) module.	

5.13.7 R_CMT_StartOneShot

R_CMT_StartOneShot		
Synopsis	Function for starting non-cyclic (one-shot) operation of the timer counter	
Header	r_cmt_if.h	
Declaration	int32_t R_CMT_StartOneShot(cmt_channel_t const channel);	
Description	This function starts non-cyclic (one-shot) operation of a timer counter. After the condition for clearing the timer counter is met, the timer stops automatically and returns to the initialized state.	
Arguments	channel	Channel numbers CMT_CHANNEL_0 to CMT_CANNEL_7
Return values	Error codes: ESUCCESS, EINVAL, EACESS, EBUSY, EFAULT	
Remarks	Call function R_CMT_Open to start operation of the CMT(W) module.	

5.13.8 R_CMT_GetVersion

R_CMT_GetVersion	
Synopsis	Acquiring the version number of the CMT(W) driver
Header	r_cmt_if.h
Declaration	int32_t R_CMT_GetVersion(void);
Description	This function returns the version number of the CMT(W) driver.
Arguments	—
Return values	Version number of the CMT(W) driver Bits 0 to 15: Minor part of the version number Bits 16 to 31: Major part of the version number
Remarks	—

5.14 Specifications of the Functions of the ELC Sample Driver

The specifications of the functions of the ELC sample driver are listed below.

5.14.1 R_ELC_Init

R_ELC_Init	
Synopsis	Initializing the ELC driver
Header	r_elc_if.h
Declaration	void R_ELC_Init(void);
Description	This function initializes the ELC driver. It generates a semaphore for exclusive control of the ELC driver interface.
Arguments	—
Return values	—
Remarks	—

5.14.2 R_ELC_Uninit

R_ELC_Uninit	
Synopsis	Ending the ELC driver
Header	r_elc_if.h
Declaration	int32_t R_ELC_Uninit(void);
Description	This function ends the ELC driver. The ELC module is stopped and the semaphore for exclusive control of the interface is lowered.
Arguments	—
Return values	Error codes: ESUCCESS, EFAULT
Remarks	—

5.14.3 R_ELC_Open

R_ELC_Open	
Synopsis	Initializing the ELC driver
Header	r_elc_if.h
Declaration	int32_t R_ELC_Open(void);
Description	This function makes settings to start operation of the ELC.
Arguments	—
Return values	Error codes: ESUCCESS, EINVAL (ELC_ERR_MISSING_PTR or ELC_ERR_INVALID_ARG occurred during the call of ELC_Open), EACESS, EBUSY, EFAULT
Remarks	Call function R_ELC_Init beforehand to initialize the ELC driver. After the completion of the operations required of the ELC module, call function R_ELC_Close to end operation of the ELC.

5.14.4 R_ELC_Close

R_ELC_Close	
Synopsis	Ending operation of the ELC module
Header	r_elc_if.h
Declaration	int32_t R_ELC_Close(void);
Description	This function stops operation of the ELC module.
Arguments	—
Return values	Error codes: ESUCCESS, EBUSY, EACESS, EFAULT
Remarks	Call function R_ELC_Open to start operation of the ELC module.

5.14.5 R_ELC_Control

R_ELC_Control		
Synopsis	ELC driver operation settings	
Header	r_elc_if.h	
Declaration	int32_t R_ELC_Control(uint32_t const cmd, void * const p_data);	
Description	This function handles processing for the ELC operation settings. It makes settings corresponding to the specified command. For details, see Section 5.18, Specifications of the R_ELC_Control Commands.	
Arguments	cmd	ELC control command
	p_data	Pointer to the ELC command parameter information
Return values	Error codes: ESUCCESS, EINVAL, EACESS, EBUSY, EFAULT	
Remarks	Call this function after executing R_ELC_Open().	

5.14.6 R_ELC_LinkStart

R_ELC_LinkStart	
Synopsis	Starting event linking by the ELC
Header	r_elc_if.h
Declaration	int32_t R_ELC_LinkStart(void);
Description	This function starts event linking by the ELC module.
Arguments	—
Return values	Error codes: ESUCCESS, EINVAL (ELC_ERR_MISSING_PTR or ELC_ERR_INVALID_ARG occurred during the call of ELC_LinkStart), EACESS, EBUSY, EFAULT
Remarks	Call function R_ELC_Open to start operation of the ELC module.

5.14.7 R_ELC_LinkStop

R_ELC_LinkStop	
Synopsis	Stopping event linking by the ELC
Header	r_elc_if.h
Declaration	int32_t R_ELC_LinkStop(void);
Description	This function stops event linking by the ELC module.
Arguments	—
Return values	Error codes: ESUCCESS, EINVAL (ELC_ERR_MISSING_PTR or ELC_ERR_INVALID_ARG occurred during the call of ELC_LinkStop), EACESS, EFAULT
Remarks	Call function R_ELC_LinkStart to start event linking.

5.14.8 R_ELC_GetVersion

R_ELC_GetVersion	
Synopsis	Acquiring the version number of the ELC driver
Header	r_elc_if.h
Declaration	int32_t R_ELC_GetVersion(void);
Description	This function returns the version number of the ELC driver.
Arguments	—
Return values	Version number of the ELC driver Bits 0 to 15: Minor part of the version number Bits 16 to 31: Major part of the version number
Remarks	—

5.15 Flowcharts

5.15.1 Main Processing of the Sample Program

Figure 5.5 to Figure 5.7 are the flowcharts of main processing of the sample program.

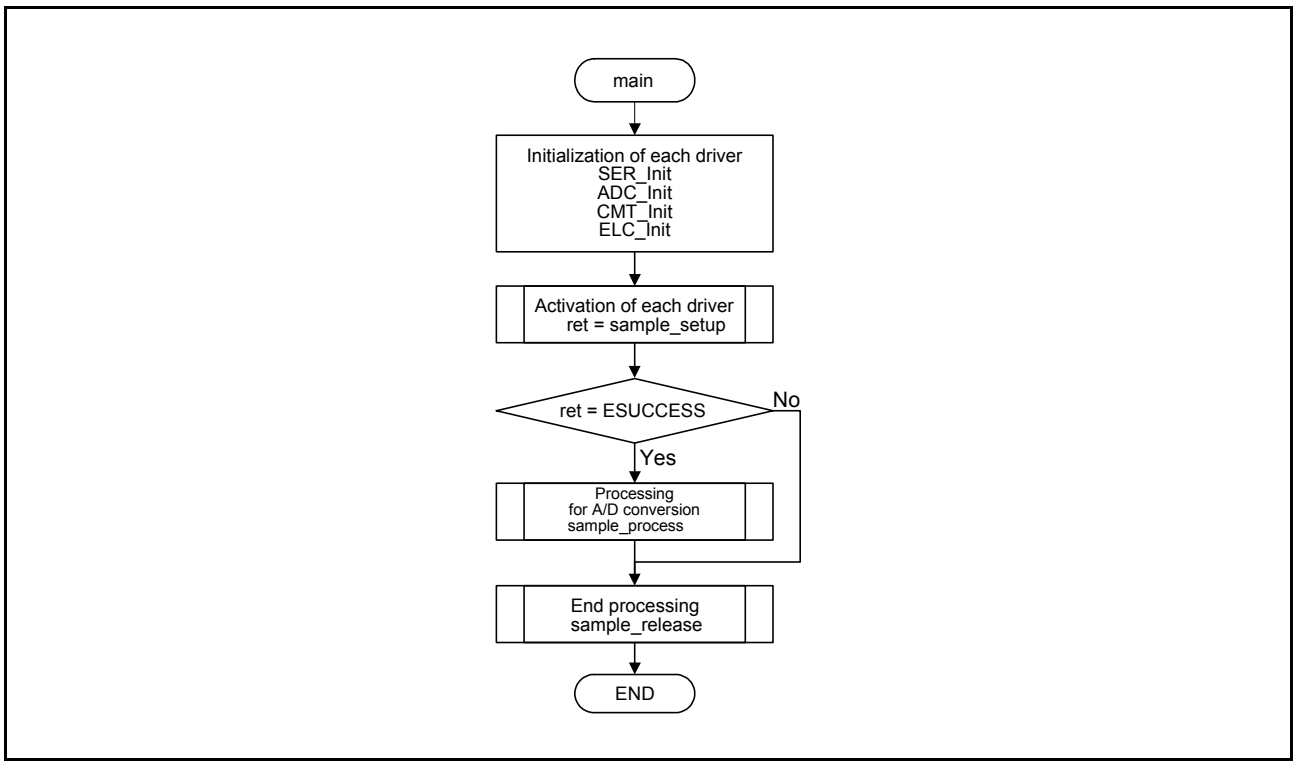


Figure 5.5 Main Processing

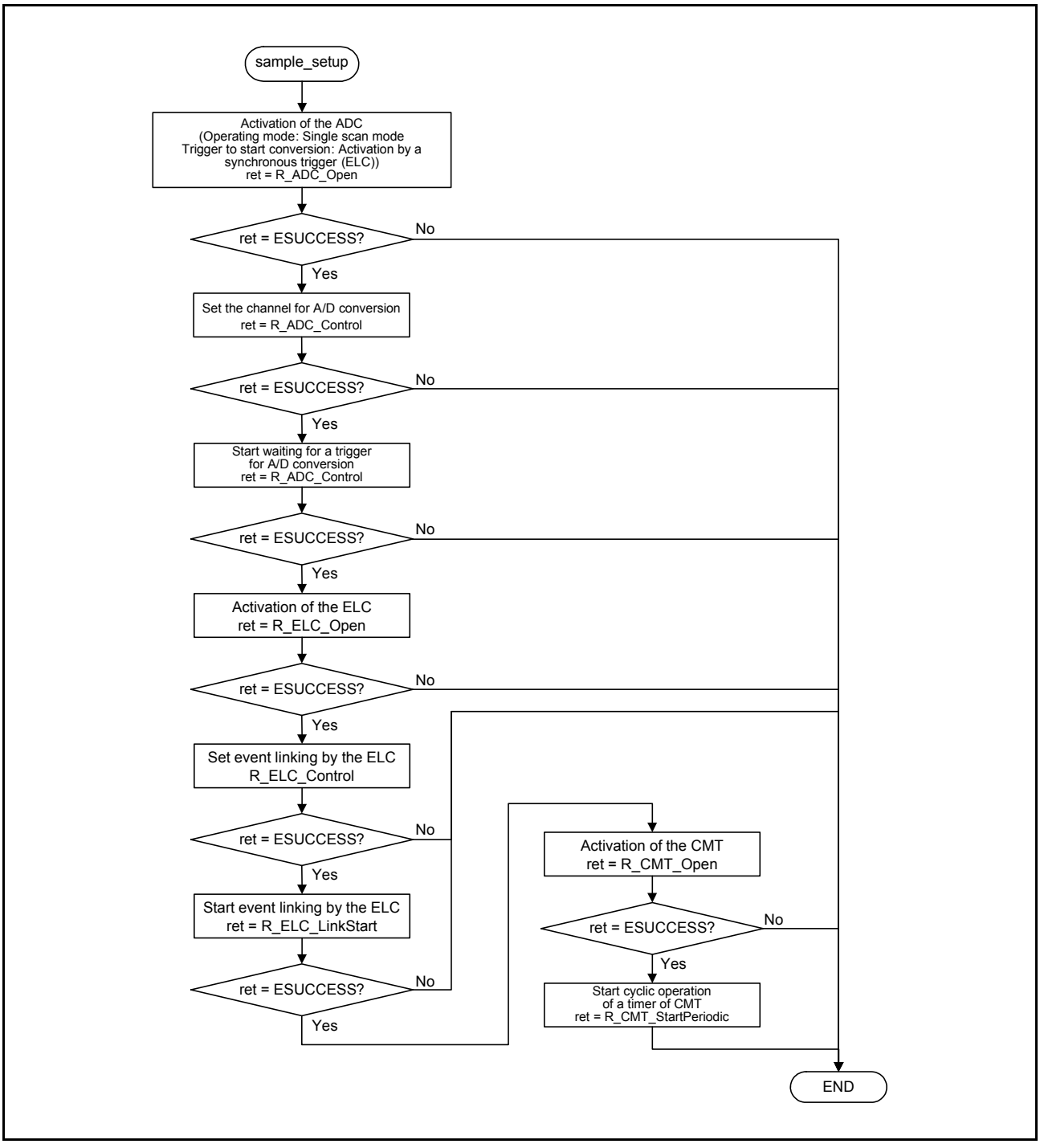


Figure 5.6 Activating the CMTW, ELC, and ADC

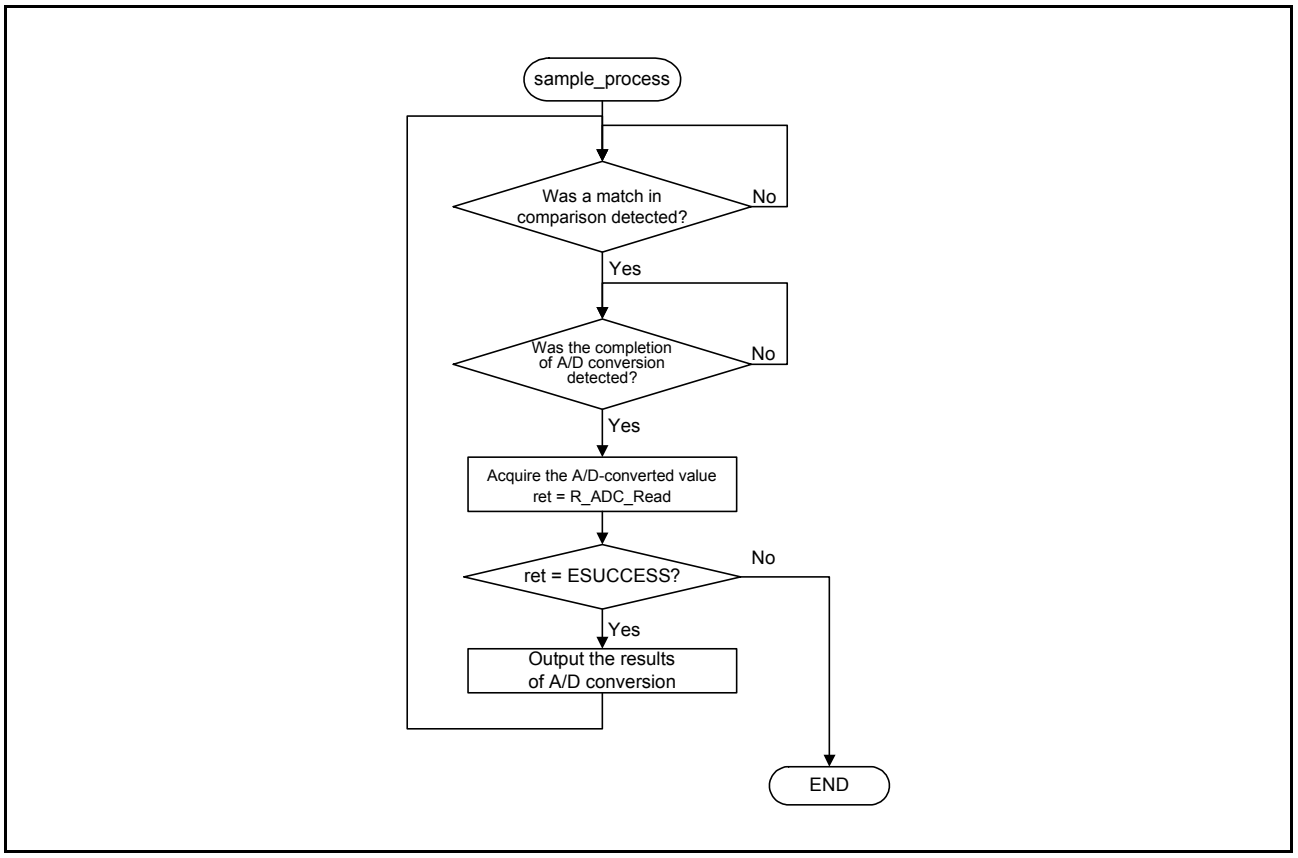


Figure 5.7 Processing for Conversion by the ADC

5.15.2 cmtw_ch0_inhr_callback

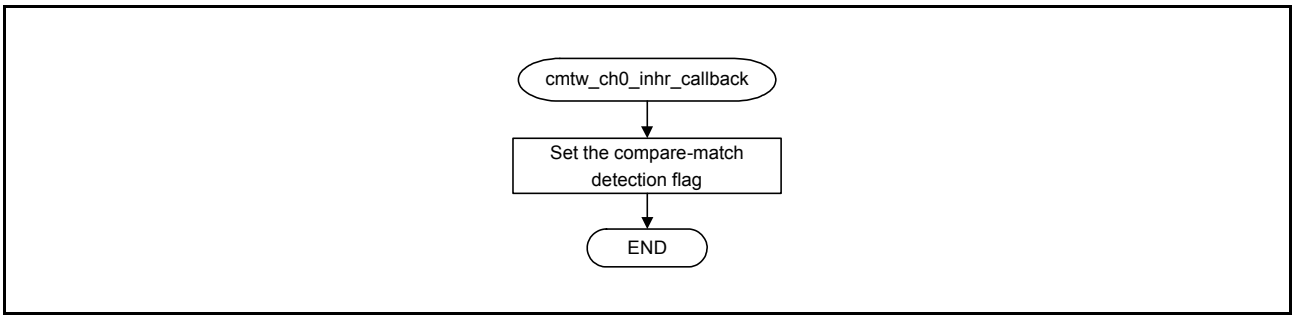


Figure 5.8 cmtw_ch0_inhr_callback

5.15.3 adc_inhr_callback

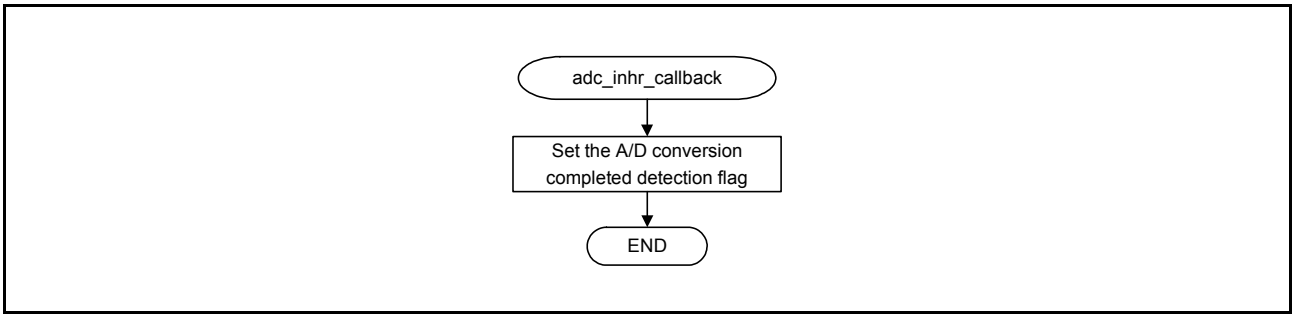


Figure 5.9 adc_inhr_callback

5.16 Specifications of the R_ADC_Control Commands

The specifications of the R_ADC_Control commands are listed below.

Table 5.45 R_ADC_Control Commands

Command	Outline
ADC_CMD_ENABLE_CHANS	Specifies the channel for A/D conversion.
ADC_CMD_ENABLE_TEMP_SENSOR	Makes initial settings for the temperature sensor.
ADC_CMD_SET_SAMPLE_STATE_CNT	Sets the sampling time for analog inputs.
ADC_CMD_ENABLE_TRIG	Enables starting of A/D conversion by a synchronous or asynchronous trigger.
ADC_CMD_DISABLE_TRIG	Disables starting of A/D conversion by a synchronous or asynchronous trigger.
ADC_CMD_SCAN_NOW	Starts A/D conversion by a software trigger.
ADC_CMD_ENABLE_INT	Enables generation of an S12ADI interrupt on completion of scanning.
ADC_CMD_DISABLE_INT	Disables generation of an S12ADI interrupt on completion of scanning.
ADC_CMD_ENABLE_INT_GROUPB	Enables generation of an S12GBADI interrupt on completion of scanning of group B.
ADC_CMD_DISABLE_INT_GROUPB	Disables generation of an S12GBADI interrupt on completion of scanning of group B.
ADC_CMD_CHECK_SCAN_DONE	Checks A/D conversion.
ADC_CMD_CHECK_SCAN_DONE_GROUPA	Checks scanning of group A.
ADC_CMD_CHECK_SCAN_DONE_GROUPB	Checks scanning of group B.

5.16.1 ADC_CMD_ENABLE_CHANS

ADC_CMD_ENABLE_CHANS			
Synopsis	Specifying the channel for A/D conversion		
Header	r_adc_if.h		
Description	This function specifies the channel for A/D conversion. The parameters are passed as an adc_ch_cfg_t type variable.		
Parameters	adc_ch_cfg_t	p_args	Specifies the channel for A/D conversion.
Return values	ADC_SUCCESS:		The channel has been specified.
	ADC_ERR_MISSING_PTR:		Incorrect pointer argument
	ADC_ERR_ILLEGAL_ARG:		The mode of function R_ADC_Open is ADC_MODE_SS_TEMPERATURE.
	ADC_ERR_INVALID_ARG:		Invalid argument value
Remarks	—		

5.16.2 ADC_CMD_ENABLE_TEMP_SENSOR

ADC_CMD_ENABLE_TEMP_SENSOR		
Synopsis	Initial settings for the temperature sensor	
Header	r_adc_if.h	
Description	This function makes initial settings for the temperature sensor. Since there are no parameters, the value of parameters should be null.	
Parameters	NULL	
Return values	ADC_SUCCESS:	The temperature sensor has been initialized.
	ADC_ERR_ILLEGAL_ARG:	The mode of function R_ADC_Open is not ADC_MODE_SS_TEMPERATURE.
Remarks	—	

5.16.3 ADC_CMD_SET_SAMPLE_STATE_CNT

ADC_CMD_SET_SAMPLE_STATE_CNT		
Synopsis	Setting the sampling time for analog inputs	
Header	r_adc_if.h	
Description	This function sets the sampling time for analog inputs. The parameters are passed as an adc_time_t type variable.	
Parameters	adc_time_t p_args	Specifies the channel for which the sampling time is to be set and the sampling time. See the adc_time_t structure.
Return values	ADC_SUCCESS:	The sampling time for analog inputs has been set.
	ADC_ERR_MISSING_PTR:	Incorrect pointer argument
	ADC_ERR_ILLEGAL_ARG:	Invalid argument value
Remarks	—	

5.16.4 ADC_CMD_ENABLE_TRIG

ADC_CMD_ENABLE_TRIG	
Synopsis	Enabling starting of A/D conversion by a synchronous or asynchronous trigger
Header	r_adc_if.h
Description	This function enables starting of A/D conversion by a synchronous or asynchronous trigger. Since there are no parameters, the value of parameters should be null.
Parameters	NULL
Return values	ADC_SUCCESS: Starting of A/D conversion by a synchronous or asynchronous trigger has been enabled.
Remarks	—

5.16.5 ADC_CMD_DISABLE_TRIG

ADC_CMD_DISABLE_TRIG	
Synopsis	Disabling starting of A/D conversion by a synchronous or asynchronous trigger
Header	r_adc_if.h
Description	This function disables starting of A/D conversion by a synchronous or asynchronous trigger. Since there are no parameters, the value of parameters should be null.
Parameters	NULL
Return values	ADC_SUCCESS: Starting of A/D conversion by a synchronous or asynchronous trigger has been disabled.
Remarks	—

5.16.6 ADC_CMD_SCAN_NOW

ADC_CMD_SCAN_NOW	
Synopsis	Starting A/D conversion by a software trigger
Header	r_adc_if.h
Description	This function starts A/D conversion by a software trigger. Since there are no parameters, the value of parameters should be null.
Parameters	NULL
Return values	ADC_SUCCESS: A/D conversion has been started. ADC_ERR_SCAN_NOT_DONE: A/D conversion is in progress.
Remarks	—

5.16.7 ADC_CMD_ENABLE_INT

ADC_CMD_ENABLE_INT	
Synopsis	Enabling generation of an S12ADI interrupt on completion of scanning
Header	r_adc_if.h
Description	This function enables generation of an S12ADI interrupt on completion of scanning. Since there are no parameters, the value of parameters should be null.
Parameters	NULL
Return values	ADC_SUCCESS: Generation of an S12ADI interrupt has been enabled on completion of scanning. ADC_ERR_ILLEGAL_ARG: A callback function is not registered.
Remarks	—

5.16.8 ADC_CMD_DISABLE_INT

ADC_CMD_DISABLE_INT	
Synopsis	Disabling generation of an S12ADI interrupt on completion of scanning
Header	r_adc_if.h
Description	This function disables generation of an S12ADI interrupt on completion of scanning. Since there are no parameters, the value of parameters should be null.
Parameters	NULL
Return values	ADC_SUCCESS: Generation of an S12ADI interrupt has been disabled on completion of scanning.
Remarks	—

5.16.9 ADC_CMD_ENABLE_INT_GROUPB

ADC_CMD_ENABLE_INT_GROUPB		
Synopsis	Enabling generation of an S12GBADI interrupt on completion of scanning of group B	
Header	r_adc_if.h	
Description	This function enables generation of an S12GBADI interrupt on completion of scanning of group B. Since there are no parameters, the value of parameters should be null.	
Parameters	NULL	
Return values	ADC_SUCCESS:	Generation of an S12GBADI interrupt has been enabled on completion of scanning of group B.
	ADC_ERR_ILLEGAL_ARG:	A callback function is not registered.
Remarks	—	

5.16.10 ADC_CMD_DISABLE_INT_GROUPB

ADC_CMD_DISABLE_INT_GROUPB	
Synopsis	Disabling generation of an S12GBADI interrupt on completion of scanning of group B
Header	r_adc_if.h
Description	This function disables generation of an S12GBADI interrupt on completion of scanning of group B. Since there are no parameters, the value of parameters should be null.
Parameters	NULL
Return values	ADC_SUCCESS: Generation of an S12GBADI interrupt has been disabled on completion of scanning of group B.
Remarks	—

5.16.11 ADC_CMD_CHECK_SCAN_DONE

ADC_CMD_CHECK_SCAN_DONE		
Synopsis	Checking A/D conversion	
Header	r_adc_if.h	
Description	This function checks whether A/D is in progress. Since there are no parameters, the value of parameters should be null.	
Parameters	NULL	
Return values	ADC_SUCCESS: ADC_ERR_SCAN_NOT_DONE:	A/D conversion has been completed. A/D conversion is in progress.
Remarks	—	

5.16.12 ADC_CMD_CHECK_SCAN_DONE_GROUPA

ADC_CMD_CHECK_SCAN_DONE_GROUPA		
Synopsis	Checking scanning of group A	
Header	r_adc_if.h	
Description	This function checks whether scanning of group A has been completed. Since there are no parameters, the value of parameters should be null.	
Parameters	NULL	
Return values	ADC_SUCCESS: ADC_ERR_SCAN_NOT_DONE:	Scanning of group A has been completed. Group A is being scanned.
Remarks	—	

5.16.13 ADC_CMD_CHECK_SCAN_DONE_GROUPB

ADC_CMD_CHECK_SCAN_DONE_GROUPB		
Synopsis	Checking scanning of group B	
Header	r_adc_if.h	
Description	This function checks whether scanning of group B has been completed. Since there are no parameters, the value of parameters should be null.	
Parameters	NULL	
Return values	ADC_SUCCESS: ADC_ERR_SCAN_NOT_DONE:	Scanning of group B has been completed. Group B is being scanned.
Remarks	—	

5.17 Specifications of the R_CMT_Control Commands

The specifications of the commands of R_CMT_Control are listed below.

Table 5.46 Commands of R_CMT_Control

Constant	Description
CMT_CMD_SET_TIME_CNT	Sets a timer counter of a CMT(W).
CMT_CMD_SET_MODE	Sets the operating mode and parameters for a CMT(W).
CMT_CMD_SET_PAUSE	Pauses the timer.
CMT_CMD_SET_RESUME	Resumes counter operation from the current counter value.
CMT_CMD_SET_RESTART	Clears the counter to 0 to resume counter operation.
CMT_CMD_SET_ECM	Sets the output of an ECM dynamic mode error.
CMTW_CMD_GET_STATUS	Acquires the operating state of the counter.

5.17.1 CMT_CMD_SET_TIME_CNT

CMT_CMD_SET_TIME_CNT		
Synopsis	Setting a timer counter of a CMT(W)	
Header	r_cmt_if.h	
Description	This function sets a timer counter of a CMTW. The parameters are passed as a cmtw_time_cnt_t type variable.	
Parameters	uint32_t pclk_div	Sets the frequency division ratio of the PCLKD clock.
	uint32_t cnt_size	Sets the counter size.
	uint32_t clear_factor	Sets the source for clearing the timer counter.
Return values	CMT_SUCCESS:	Setting succeeded
	CMT_ERR_INVALID_ARG:	A member of the timer counter information has an invalid value.
	CMT_ERR_TIMER_RUNNING:	The function is executed while the timer counter is running.
	CMT_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	—	

5.17.2 CMT_CMD_SET_MODE

CMT_CMD_SET_MODE		
Synopsis	Setting the operating mode of a CMT(W)	
Header	r_cmt_if.h	
Description	This command sets the operating mode of a CMT(W) and the parameters for each of the operating modes. The parameters are passed as a cmtw_mode_t type variable.	
Parameters	cmtw_compare_match_t compare_match	Stores compare match parameters.
	int32_t mode_enable	Switches the compare match function on or off. true: on; false: off
	uint32_t compare_match_cnt	Sets the value for the counter. When the counter size is 16 bits, values set in the 16 higher-order bits will be ignored.
	int32_t intr_priority	Specifies the priority of the compare match interrupt.
	void (*p_ callback)(void)	Sets the pointer to the compare match callback function. Setting the pointer to NULL will inhibit the notification of the occurrence of a compare match without causing an error.
	cmtw_output_compare_t output_compare[CMTW_OUTPUT_COMPARE_NUM]	Stores output compare parameters. The array number represents whether the parameters are for output compare 0 or output compare 1.
	int32_t mode_enable	Sets the output compare function on or off. true: on; false: off
	uint32_t output_compare_cnt	Sets the value of the output compare register for the counter. When the counter size is 16 bits, values set in the 16 higher-order bits will be ignored.
	uint32_t output_signal	Sets the signal value of the output compare output.
	int32_t intr_priority	Specifies the priority of the output compare interrupt.
	void (*p_ callback)(void)	Sets the pointer to the output compare 0 or 1 callback function. Setting the pointer to NULL will inhibit the notification of the occurrence of an output compare without causing an error.
	cmtw_input_capture_t input_capture[CMTW_INPUT_CAPTURE_NUM]	Stores input capture parameters. The array number represents whether the parameters are for input capture 0 or input capture 1.
	int32_t mode_enable	Sets the input capture function on or off. true: on; false: off
	uint32_t trigger	Sets the trigger for executing input capture.
	int32_t filter_enable	Sets the noise filter function on or off. true: on; false: off
	int32_t intr_priority	Specifies the priority of the input capture interrupt.
	void (*p_ callback) (uint32_t cnt_value)	Sets the pointer to the input capture 0 or 1 callback function. Setting the pointer to NULL will inhibit the notification of the occurrence of an input capture without causing an error.
	uint32_t noise_filter_clk	Sets the frequency division ratio of the PCLKD clock used for the noise filter. This parameter is invalid when multiple noise filters are available in input capture 0 or 1.

Return values	CMT_SUCCESS:	Setting succeeded
	CMT_ERR_INVALID_ARG:	A member of the timer counter information has an invalid value.
	CMT_ERR_TIMER_RUNNING:	This function is executed while the timer counter is running.
	CMT_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	—	

5.17.3 CMT_CMD_SET_PAUSE

CMT_CMD_SET_PAUSE		
Synopsis	Pausing the timer	
Header	r_cmt_if.h	
Description	This command pauses a timer counter of a CMT(W). When called with the timer counter stopping, the command ends without any processing being executed.	
Parameters	None	
Return values	CMT_SUCCESS:	Setting succeeded
	CMT_ERR_TIMER_STOP:	Executed without starting the timer counter once after initialization
Remarks	—	

5.17.4 CMT_CMD_SET_RESUME

CMT_CMD_SET_RESUME		
Synopsis	Resuming counting from the current value of the timer counter	
Header	r_cmt_if.h	
Description	This command resumes counting from the current timer counter value, when a counter of a CMT(W) is paused. When called with the timer counter running, the command ends without any processing being executed.	
Parameters	None	
Return values	CMT_SUCCESS:	Setting succeeded
	CMT_ERR_TIMER_STOP:	Executed without starting the timer counter once after initialization.
	CMT_ERR_TIMER_RUNNING:	This function is executed while the timer counter is running.
Remarks	—	

5.17.5 CMT_CMD_SET_RESTART

CMT_CMD_SET_RESTART		
Synopsis	Restarting counting after clearing of a timer counter	
Header	r_cmt_if.h	
Description	This command restarts counting after clearing the timer counter value when a counter of a CMT(W) has been paused. When called with the timer counter running, the command ends without any processing being executed.	
Parameters	None	
Return values	CMT_SUCCESS:	Setting succeeded
	CMT_ERR_TIMER_STOP:	Execution was without having started the timer counter once after initialization.
	CMT_ERR_TIMER_RUNNING:	This function is executed while the timer counter is running.
Remarks	—	

5.17.6 CMT_CMD_SET_ECM

CMT_CMD_SET_ECM		
Synopsis	Setting the error output in ECM dynamic mode	
Header	r_cmt_if.h	
Description	This command sets the error output in ECM dynamic mode. The parameters are passed as a cmtw_ecm_t type variable.	
Parameters	int32_t ecm_enable	Enables or disables the error output in ECM dynamic mode. true: Enables the error output in ECM dynamic mode. false: Disables the error output in ECM dynamic mode.
	uint32_t output_compare_num	Selects the output compare number to which the ECM dynamic mode error is output.
Return values	CMT_SUCCESS:	Setting succeeded
	CMT_ERR_INVALID_ARG:	The output compare number setting is invalid.
	CMT_ERR_TIMER_RUNNING:	This function is executed while the timer counter is running.
	CMT_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	• Enable the output compare output setting for the output compare number selected in this command separately before starting the timer. • The setting will be initialized when the timer satisfies the stop condition (R_CMT_Close is executed, non-cyclic operation of the timer ends). • Only one output comparison signal at a time in the hardware as a whole can be set as the error output in ECM dynamic mode. A current setting will be overwritten by the latest one. • The error output in ECM dynamic error mode is set to "disabled" immediately after initialization.	

5.17.7 CMTW_CMD_GET_STATUS

CMTW_CMD_GET_STATUS

Synopsis	Acquiring the operating state of the timer	
Header	r_cmt_if.h	
Description	This command acquires the operating state of the timer. The operating state is passed as a uint32_t pointer variable. The parameter name below is an example.	
Parameters	uint32_t * pcmt_status	Stores the operating state of the timer. Either of the following values is returned. CMTW_STATUS_STOP: The timer has been stopped. CMTW_STATUS_RUNNING: The timer is running.
Return values	CMT_SUCCESS: CMT_ERR_INVALID_ARG:	Acquisition succeeded The pointer to the timer operation state acquisition parameter has an invalid value.
Remarks	—	

5.18 Specifications of the R_ELC_Control Commands

The specifications of the R_ELC_Control commands are listed below.

Table 5.47 R_ELC_Control Commands

Constant	Description
ELC_CMD_SET_EVENT_MTU	Sets the event link for the MTU module.
ELC_CMD_SET_EVENT_CMT	Sets the event link for the CMT module.
ELC_CMD_SET_EVENT_DSMIF	Sets the event link for the ΔΣ unit module.
ELC_CMD_SET_EVENT_S12AD	Sets the event link for the 12-bit A/D converter.
ELC_CMD_SET_EVENT_INTR	Sets an interrupt request signal as an event link signal for the ELC.
ELC_CMD_SET_EVENT_OUT_PORT_GROUP	Sets the event link for the output port group.
ELC_CMD_SET_EVENT_IN_PORT_GROUP	Sets the event link for the input port group.
ELC_CMD_SET_EVENT_SINGLE_PORT	Sets a single port and the event link for the port.
ELC_CMD_SET_EVENT_CMTW	Sets event link parameters for CMTW0.
ELC_CMD_SET_EVENT_TPU	Sets the event link for the TPU module.
ELC_CMD_SET_EVENT_GPT	Sets the event link for the GPT module.
ELC_CMD_SET_PORT_GROUP	Sets a port group.
ELC_CMD_SET_SOFTWARE_EVENT	Issues a software event of the ELC.
ELC_CMD_GET_PORT_GROUP_VALUE	Acquires the signal values of a port group.

5.18.1 ELC_CMD_SET_EVENT_MTU

ELC_CMD_SET_EVENT_MTU		
Synopsis	Setting event link parameters for MTU0, MTU3, or MTU4	
Header	r_elc_if.h	
Description	This command sets event link parameters for the ELC to MTU0, MTU3, or MTU4. The parameters are passed as an elc_cmd_mtu_t type variable.	
Parameters	uint32_t elc_mtu_ch	Specifies the MTU unit number to be set.
	int32_t event_link_enable	Switches the event link for the MTU on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
	uint32_t action	Sets the action when an event linked with the MTU occurs.
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	—	

5.18.2 ELC_CMD_SET_EVENT_CMT

ELC_CMD_SET_EVENT_CMT		
Synopsis	Setting event link parameters for CMT1	
Header	r_elc_if.h	
Description	This command sets event link parameters of the ELC for CMT1. The parameters are passed as an elc_cmd_cmt_t type variable.	
Parameters	int32_t event_link_enable	Switches the event link for CMT1 on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
	uint32_t action	Sets the action when an event linked with CMT1 occurs.
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	—	

5.18.3 ELC_CMD_SET_EVENT_DSMIF

ELC_CMD_SET_ EVENT_DSMIF		
Synopsis	Setting event link parameters for trigger 0 or 1 of ΔΣ unit 0 or 1	
Header	r_elc_if.h	
Description	This command sets event link parameters of the ELC for trigger 0 or 1 of ΔΣ unit 0 or 1. The parameters are passed as an elc_cmd_dsmif_t type variable.	
Parameters	uint32_t elc_dsmif_ch	Specifies the number of trigger 0 or 1 of ΔΣ unit 0 or 1 to be set.
	int32_t event_link_enable	Switches the event link for the ΔΣ unit on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	—	

5.18.4 ELC_CMD_SET_EVENT_S12AD

ELC_CMD_SET_EVENT_S12AD		
Synopsis	Sets event link parameters for 12-bit A/D converter 0 or 1	
Header	r_elc_if.h	
Description	This command sets event link parameters of the ELC for 12-bit A/D converter 0 or 1. The parameters are passed as an elc_cmd_s12ad_t type variable.	
Parameters	uint32_t elc_s12ad_ch	Specifies the number of the 12-bit A/D converter to be set.
	int32_t event_link_enable	Switches the event link for the 12-bit A/D converter on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	—	

5.18.5 ELC_CMD_SET_EVENT_INTR

ELC_CMD_SET_EVENT_INTR		
Synopsis	Setting event link parameters for interrupt request signal 1 or 2 of the ELC.	
Header	r_elc_if.h	
Description	This command sets evet link parameters of the ELC for interrupt request signal 1 or 2 of the ELC. The parameters are passed as an elc_cmd_intr_t type variable.	
Parameters	uint32_t elc_intr_num	Specifies the number of the interrupt request signal.
	int32_t event_link_enable	Switches the event link for interrupt request signal 1 or 2 on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
	int32_t intr_priority	Specifies the priority of the interrupt.
	void (*p_ callback)(void)	Sets the pointer to the event link callback function. Setting the pointer to NULL will inhibit the notification of the occurrence of an interrupt without causing an error.
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	—	

5.18.6 ELC_CMD_SET_EVENT_OUT_PORT_GROUP

ELC_CMD_SET_EVENT_OUT_PORT_GROUP		
Synopsis	Setting event link parameters for output port group 1 or 2	
Header	r_elc_if.h	
Description	This command sets event link parameters of the ELC for output port group 1 or 2. The parameters are passed as an elc_cmd_out_port_group_t type variable.	
Parameters	uint32_t elc_out_port_group_num	Specifies the number of the output port group.
	int32_t event_link_enable	Switches the event link for output port group 1 or 2 on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
	uint32_t action	Sets the action when an event linked with output port group 1 or 2 occurs.
	uint8_t init_value	Sets the initial output value of the output port group.
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	After resetting the rotate state to the initial state while rotate output has been set, use this command to set the parameters of the initial state again.	

5.18.7 ELC_CMD_SET_EVENT_IN_PORT_GROUP

ELC_CMD_SET_EVENT_IN_PORT_GROUP		
Synopsis	Setting event link parameters for input port group 1 or 2	
Header	r_elc_if.h	
Description	This command sets event link parameters of the ELC for input port group 1 or 2. The parameters are passed as an elc_cmd_in_port_group_t type variable.	
Parameters	uint32_t elc_in_port_group_num	Specifies the number of the input port group.
	int32_t event_link_enable	Switches the event link for input port group 1 on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
	int32_t overwrite_enable	Sets whether or not to enable overwriting signal values in the buffer in response to events. true: Overwriting enabled false: Overwriting disabled
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	When overwriting signal values in the buffer is disabled in response to events, subsequent events will be ignored until the buffer value is read. Use ELC_OUT_GROUP_BUFFER to read the buffer value.	

5.18.8 ELC_CMD_SET_EVENT_SINGLE_PORT

ELC_CMD_SET_EVENT_SINGLE_PORT		
Synopsis	Registering a single port and setting event link parameters for the single port	
Header	r_elc_if.h	
Description	This command registers I/O ports for single port 0, 1, 2, or 3 and sets event link parameters of the ELC. The parameters are passed as an elc_cmd_single_port_t type variable.	
Parameters	uint32_t elc_single_port_num	Specifies the number of the single port.
	uint32_t port_symbol	Selects the port symbol to be set as a single port.
	uint32_t port_num	Specifies the I/O port number to be set as a single port from 0 to 7.
	int32_t event_link_enable	Switches the event link for a single port on or off. When the event link is switched on, the program waits for an event and outputs data from the single pin in response to an event. When the event link is switched off, the program waits for data input to the single input port and issues an event link request after detecting data input. true: on false: off
	uint32_t event_direction	Selects between event input and event output at the time of an event link.
	uint32_t resource	Sets the event signal of the event link source. This parameter is only valid when event_link_enable is true and signal_direction is ELC_SINGLE_EVENT_OUTPUT.
	uint32 _t output_action	Sets the action of the output single port at the time of event linking. This parameter is only valid when event_link_enable is true and signal_direction is ELC_SINGLE_EVENT_OUTPUT.
	uint32_t input_trigger	Specifies the trigger for detecting data input to a single input port. This parameter is only valid when event_link_enable is true and signal_direction is ELC_SINGLE_EVENT_INPUT.
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	• This sample driver does not set the data input/output direction of the I/O port registered as a single port. Use the I/O driver, etc. to set the direction. • When both a single port and a port group are set for a given I/O port, both functions will be valid when the port is set for input. Only the setting of the port group will be valid when the port is set for output.	

5.18.9 ELC_CMD_SET_EVENT_CMTW

ELC_CMD_SET_EVENT_CMTW		
Synopsis	Setting event link parameters for CMTW0	
Header	r_elc_if.h	
Description	This command sets event link parameters of the ELC for CMTW0. The parameters are passed as an elc_cmd_cmtw_t type variable.	
Parameters	int32_t event_link_enable	Switches the event link for CMTW0 on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
	uint32_t action	Sets the action when an event linked with CMTW0 occurs.
Return values	ELC_SUCCESS: ELC_ERR_INVALID_ARG: ELC_ERR_MISSING_PTR:	Setting succeeded A member of the event link information has an invalid value. Incorrect pointer argument
Remarks	—	

5.18.10 ELC_CMD_SET_EVENT_TPU

ELC_CMD_SET_EVENT_TPU		
Synopsis	Setting event link parameters for TPU0, TPU1, TPU2, or TPU3	
Header	r_elc_if.h	
Description	This command sets event link parameters of the ELC for TPU0, TPU1, TPU2, or TPU3. The parameters are passed as an elc_cmd_tpu_t type variable.	
Parameters	uint32_t elc_tpu_ch	Specifies the number of the TPU.
	int32_t event_link_enable	Switches the event link for TPU0, TPU1, TPU2, or TPU3 on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
	uint32_t action	Sets the action when an event linked withTPU0, TPU1, TPU2, or TPU3 occurs.
Return values	ELC_SUCCESS: ELC_ERR_INVALID_ARG: ELC_ERR_MISSING_PTR:	Setting succeeded A member of the event link information has an invalid value. Incorrect pointer argument
Remarks	—	

5.18.11 ELC_CMD_SET_EVENT_GPT

ELC_CMD_SET_EVENT_GPT		
Synopsis	Setting event link parameters for GPT0, GPT1, GPT2, or GPT3	
Header	r_elc_if.h	
Description	This command sets event link parameters of the ELC for GPT0, GPT1, GPT2, or GPT3. The parameters are passed as an elc_cmd_gpt_t type variable.	
Parameters	uint32_t elc_gpt_ch	Specifies the number of the GPT.
	int32_t event_link_enable	Switches the event link for GPT0, GPT1, GPT2, or GPT3 on or off. true: on false: off
	uint32_t resource	Sets the event signal of the event link source.
	uint32_t action	Sets the action when an event linked with GPT0, GPT1, GPT2, or GPT3 occurs.
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	—	

5.18.12 ELC_CMD_SET_PORT_GROUP

ELC_CMD_SET_PORT_GROUP		
Synopsis	Setting a port group	
Header	r_elc_if.h	
Description	This command sets a port group. The parameters are passed as an elc_cmd_port_group_t type variable.	
Parameters	uint32_t port_group_num	Selects the port group number to be set.
	uint8_t port_group_bit	Specifies the port numbers to be specified as members of a port group by a bit value. Bits 0 to 7 correspond to port numbers 0 to 7 of the I/O port. When a given bit is 1, the corresponding port is set as part of the port group.
	uint32_t trigger	Specifies the trigger for the output of an event signal when the port group can work as an event link source.
Return values	ELC_SUCCESS:	Setting succeeded
	ELC_ERR_INVALID_ARG:	A member of the event link information has an invalid value.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	• This sample driver does not set port groups for input or output. Use the I/O driver, etc. to set the direction. • When both a single port and a port group are set for a given I/O port, the both functions will be valid when the port is set for input. Only the setting of the port group will be valid when the port is set for output. • Port group 1 and port group 2 correspond to "port number: port B" and "port number: port E," respectively.	

5.18.13 ELC_CMD_SET_SOFTWARE_EVENT

ELC_CMD_SET_SOFTWARE_EVENT	
Synopsis	Issuing a software event of the ELC
Header	r_elc_if.h
Description	This command issues a software event of the ELC.
Parameters	—
Return values	—
Remarks	—

5.18.14 ELC_CMD_GET_PORT_GROUP_VALUE

ELC_CMD_GET_PORT_GROUP_VALUE		
Synopsis	Acquiring the signal value of a port group	
Header	r_elc_if.h	
Description	This command acquires the signal value of a port group. The parameters are passed as an elc_get_port_value_t type variable.	
Parameters	uint32_t elc_port_group_num	Specifies the port group number from which the value is to be acquired.
	uint8_t port_value	Stores the signal values of the input port group.
Return values	ELC_SUCCESS:	Acquisition succeeded
	ELC_ERR_INVALID_ARG:	Incorrect port group number is specified.
	ELC_ERR_MISSING_PTR:	Incorrect pointer argument
Remarks	—	

6. Sample Program

The sample program can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry

Revision History	Application Note: CMSIS-RTOS RTX for Cortex-R4 CMT(W) & ELC & ADC Sample Programs (for use with the following combinations of environments and compilers: EWARM and ICCARM, e2studio and Renesas GCC, DS-5 and ARMCC)

Rev.	Date	Description	
		Page	Summary
1.00	Mar. 07, 2017	—	First Edition issued

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

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

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

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

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.3.0-1 November 2016)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

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

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

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

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

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

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

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

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

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

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

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

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

Colophon 6.0