RZ/T1 Group

CMSIS-RTOS RTX for Cortex-R4
MTU3a Sample Program
(for use with the following combinations of environments and compilers:
EWARM and ICCARM, e2studio and Renesas GCC, DS-5 and ARMCC)

R01AN3540EJ0100
Rev.1.00
Mar. 13, 2017

Introduction

This application note describes a sample program that sets up the output of positive and negative PWM waveforms in three phases (six signals in total) that include dead time, by using the complementary PWM mode of MTU3 and MTU4 of the multi-function timer pulse unit (MTU3a) of RZ/T1 devices.

The feature of the MTU3a sample program:

- Output of complementary PWM waveforms with a carrier cycle of 100 µs and dead time of 2 µs by using MTU3 and MTU4.

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
Clock generation circuit (CPG)	The CPG produces the CPU clock and low-speed on-chip oscillator clock signals.
Interrupt controller (ICUA)	The ICUA is used for the compare-match interrupt from MTU3a.
Multi-function timer pulse unit (MTU3a)	The MTU3a is used for complementary PWM output for MTU3 and MTU4.
Error control module (ECM)	The ECM is used to initialize the ERROROUT# pin.

The details of the above modules are given in *RZ/T1 Group User's Manual: Hardware*.

2. Operating Environment

The sample program covered in this application note is for the environment 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
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.

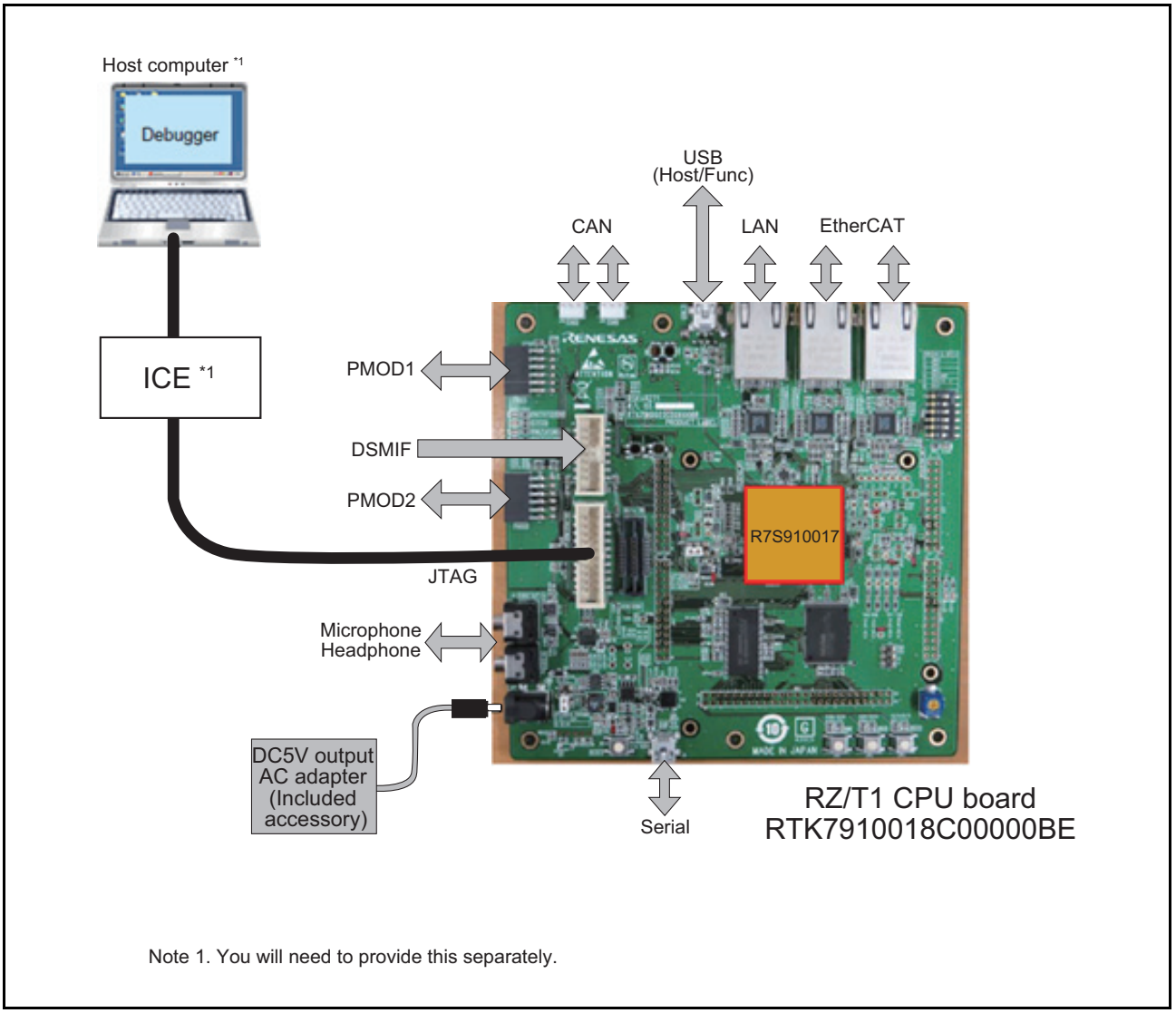

Figure 4.1 Operating Environment

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

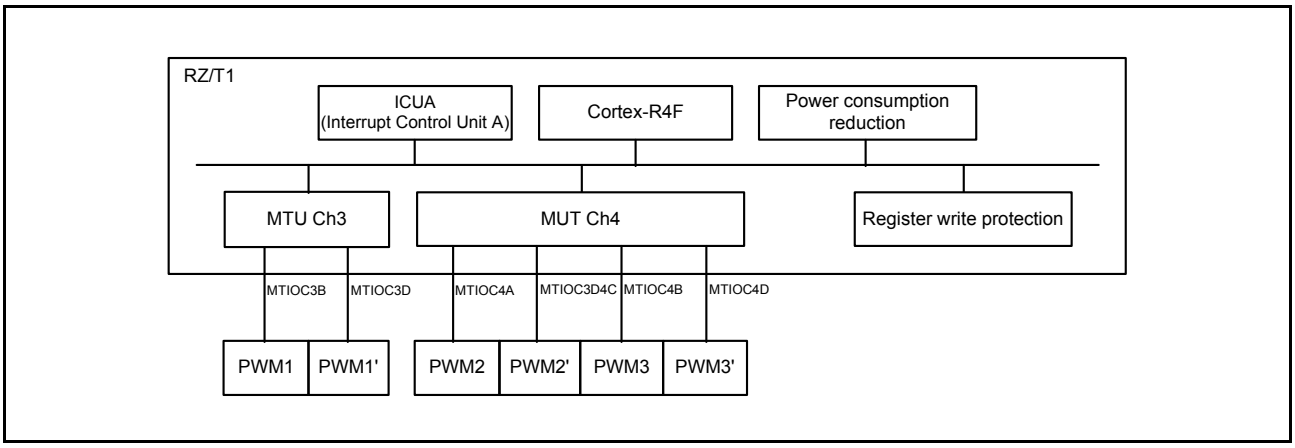

Figure 4.2 Example of the Hardware Configuration

4.2 Pins

The table below 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	
MTIOC3B	Output	PWM output 1
MTIOC3D	Output	PWM output 1' (inverse-phase waveform output for PWM output 1)
MTIOC4A	Output	PWM output 2
MTIOC4C	Output	PWM output 2' (inverse-phase waveform output for PWM output 2)
MTIOC4B	Output	PWM output 3
MTIOC4D	Output	PWM output 3' (inverse-phase waveform output for PWM output 3)

5. Software

5.1 Operation in Outline

This sample program makes initial settings for the multi-function timer pulse unit (MTU3a) to output complementary PWM signals.

The table below lists the operation of the sample program in outline. Figure 5.1 shows the waveforms.

Table 5.1 Operation in Outline

Function	Outline
Channel	Channel 3 (MTU3), channel 4 (MTU4)
PWM output	Positive and negative waveforms in three phases (six signals in total), carrier cycle 100 us, dead time 2 µs
Operating mode	Complementary PWM mode 1 (to be transferred at crest)
Clock	PCLKC/4 (= 37.5 MHz), rising edges
Duty cycle	25%

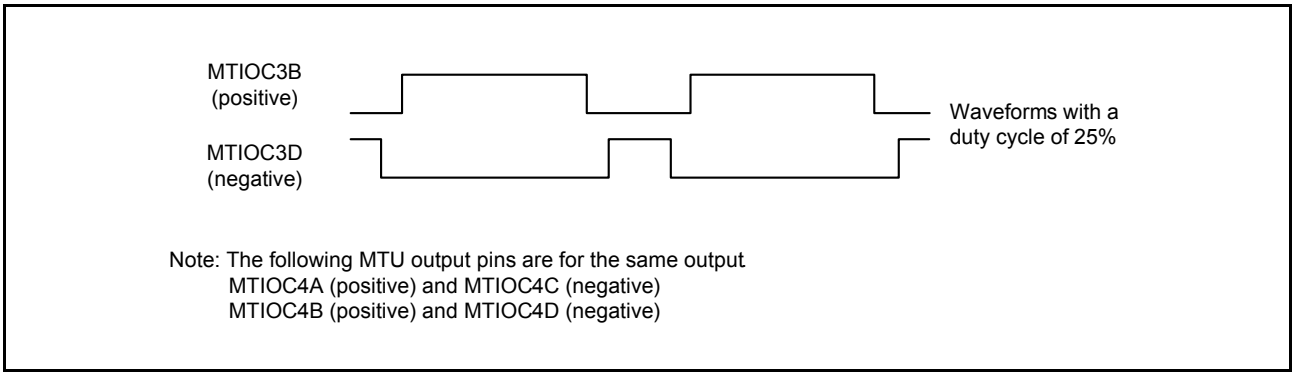

Figure 5.1 Waveforms

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.

```
[Drivers]          RTOS\RTX\Boards\Renesas\RZ_T1_RSK\Renesas_example\Drivers folder
    |
    ├── [drv_example]
    |       └──[RTX_MTU3_ex1]        Source code of this sample program
    |
    ├── [drv_inc]                    Header files of the sample drivers
    |
    └── [drv_src]                    Source code of the sample drivers
            └──[mtu3]                Source code of the SMTU3 sample driver
```

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.

5.6 Fixed-Width Integers

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)
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)

5.7 Constants

The tables below lists 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
MTU3_CFG_PARAM_CHECKING_ ENABLE	1	Indicates enabling (1) or disabling (0) of parameter checking via the API function of the MTU.
MTU_COUNT_STOP	0	Stops counter operation of the MTU.
MTU_CMPLMT_PWM_START	0xC0	Starts complementary PWM.
MTU_C_SET_CYCLE	0xEA6	Sets the carrier cycle.
MTU_C_CYCLE	0x753	This constant sets 1/2 of the carrier cycle.
MTU_DEAD_TIME	0x4B	This constant sets the dead time.
MTU_TGRA_CYCLE	MTU_C_CYCLE + MTU_DEAD_TIME	This constant sets 1/2 of the carrier cycle + the dead time.
MTU_ENABLE_OUTPUT	1	Enables MTU output.
MTU_DISABLE_OUTPUT	0	Disables MTU output.

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
EFAULT	-97	Failure

5.8 Structures, Unions, and Enumerations

Figure 5.2 to Figure 5.14 list the structures, unions, and enumerations for the sample drivers.

```
/* Enumeration of MTU channel numbers. */
typedef enum
{
    MTU_CHANNEL_0 = 0,
    MTU_CHANNEL_1,
    MTU_CHANNEL_2,
    MTU_CHANNEL_3,
    MTU_CHANNEL_4,
    MTU_CHANNEL_5,   /* This channel not support */
    MTU_CHANNEL_6,
    MTU_CHANNEL_7,
    MTU_CHANNEL_8,
    MTU_CHANNEL_MAX
} mtu_channel_t;

/* Clocking source selections. Index into register settings table. */
typedef enum mtu_clk_sources_e
{
    MTU_CLK_SRC_EXT_MTCLKA = 0x00,      // External clock input on MTCLKA pin
    MTU_CLK_SRC_EXT_MTCLKB = 0x01,      // External clock input on MTCLKB pin
    MTU_CLK_SRC_EXT_MTCLKC = 0x02,      // External clock input on MTCLKC pin
    MTU_CLK_SRC_EXT_MTCLKD = 0x03,      // External clock input on MTCLKD pin
    MTU_CLK_SRC_CASCADE    = 0x04,     // Clock by overflow from other channel counter. (only on certain channels)
    MTU_CLK_SRC_INTERNAL               // Use internal clock (PCLK)
} mtu_clk_sources_t;
```

Figure 5.2 Structures, Unions, and Enumerations for the Sample Drivers (1)

```
/* The possible settings for MTU output pins. Register setting values. */
typedef enum mtu_output_states_e
{
    MTU_PIN_NO_OUTPUT = 0x0,    // Output high impedance.
    MTU_PIN_LO_GOLO   = 0x1,    // Initial output is low. Low output at compare match.
    MTU_PIN_LO_GOHI   = 0x2,    // Initial output is low. High output at compare match.
    MTU_PIN_LO_TOGGLE = 0x3,    // Initial output is low. Toggle (alternate) output at compare match.
    MTU_PIN_HI_GOLO   = 0x5,    // Initial output is high. Low output at compare match.
    MTU_PIN_HI_GOHI   = 0x6,    // Initial output is high. High output at compare match.
    MTU_PIN_HI_TOGGLE = 0x7     // Initial output is high. Toggle (alternate) output at compare match.
} mtu_output_states_t;

/* The possible settings for counting clock active edge. Register setting values. */
typedef enum mtu_clk_edges_e
{
    MTU_CLK_RISING_EDGE  = 0x00,
    MTU_CLK_FALLING_EDGE = 0x08,
    MTU_CLK_ANY_EDGE     = 0x10,
} mtu_clk_edges_t;
```

Figure 5.3 Structures, Unions, and Enumerations for the Sample Drivers (2)

```
/* The possible counter clearing source selections. Index into register settings table. */
typedef enum mtu_clear_src_e
{
    MTU_CLR_TIMER_A = 0,    // Clear the channel counter on the "A" compare or capture event.
    MTU_CLR_TIMER_B,        // Clear the channel counter on the "B" compare or capture event.
    MTU_CLR_TIMER_C,        // Clear the channel counter on the "C" compare or capture event.
    MTU_CLR_TIMER_D,        // Clear the channel counter on the "D" compare or capture event.
    MTU_CLR_SYNC,           // Clear the channel counter when another sync'ed channel clears.
    MTU_CLR_DISABLED        // Never clear the channel counter.
} mtu_clear_src_t;
/* PCLK divisor  for internal clocking source. Index into register settings table. */
typedef enum mtu_pclk_divisor_e
{
    MTU_SRC_CLK_DIV_1 = 0, // PCLK/1
    MTU_SRC_CLK_DIV_2,     // PCLK/2
    MTU_SRC_CLK_DIV_4,     // PCLK/4
    MTU_SRC_CLK_DIV_8,     // PCLK/8
    MTU_SRC_CLK_DIV_16,    // PCLK/16
    MTU_SRC_CLK_DIV_32,    // PCLK/32
    MTU_SRC_CLK_DIV_64,    // PCLK/64
    MTU_SRC_CLK_DIV_256,   // PCLK/256
    MTU_SRC_CLK_DIV_1024  // PCLK/1024
} mtu_src_clk_divisor_t;
/* Actions to be done upon timer or capture event. Multiple selections to be ORed together. */
typedef enum mtu_actions_e
{
    MTU_ACTION_NONE      = 0x00,      // Do nothing with this timer.
    MTU_ACTION_OUTPUT    = 0x01,      // Change state of output pin.
    MTU_ACTION_INTERRUPT = 0x02,      // Generate interrupt request.
    MTU_ACTION_CALLBACK  = 0x04,     // Generate interrupt request and execute user-defined callback on interrupt.
    MTU_ACTION_REPEAT    = 0x10,      // Continuously repeat the timer cycle and actions
    MTU_ACTION_TRIGGER_ADC = 0x20,    // Trigger ADC on this event. Timer A events only.
    MTU_ACTION_CAPTURE   = 0x40,     // Default input capture action. Placeholder value, does not to be specified.
} mtu_actions_t;
```

Figure 5.4 Structures, Unions, and Enumerations for the Sample Drivers (3)

```
/************ Type defines used with the R_MTU_Control function. ***************/
/* Control function command codes. */
typedef enum mtu_cmd_e
{
    MTU_CMD_START,              // Activate clocking
    MTU_CMD_STOP,               // Pause clocking
    MTU_CMD_SAFE_STOP,          // Stop clocking and set outputs to safe state
    MTU_CMD_RESTART,            // Zero the counter then resume clocking
    MTU_CMD_SYNCHRONIZE,        // Specify channels to group for synchronized clearing.
    MTU_CMD_GET_STATUS,         // Retrieve the current status of the channel
    MTU_CMD_SET_CAPT_EDGE,      // Sets the detection edge polarity for input capture.
    MTU_CMD_UNKNOWN             // Not a valid command.
} mtu_cmd_t;

/* Used as bit-field identifiers to identify channels assigned to a group for group operations.
 * Add multiple channels to group by ORing these values together. */
typedef enum
{
    MTU_GRP_CH0 = 0x0001,
    MTU_GRP_CH1 = 0x0002,
    MTU_GRP_CH2 = 0x0004,
    MTU_GRP_CH3 = 0x0040,
    MTU_GRP_CH4 = 0x0080,
    PROHIBTID   = 0x0000,   /* This channel not support */
    MTU_GRP_CH6 = 0x4000,
    MTU_GRP_CH7 = 0x8000,
    MTU_GRP_CH8 = 0x0008
} mtu_group_t;
```

Figure 5.5 Structures, Unions, and Enumerations for the Sample Drivers (4)

```
typedef struct mtu_timer_status_s
{
    uint32_t timer_count;           // The current channel counter value.
    Bool_t   timer_running;         // True = timer currently counting, false = counting stopped.
} mtu_timer_status_t;

typedef struct mtu_capture_status_s
{
    uint32_t capt_a_count;          // The count at input capture A event.
    uint32_t capt_b_count;          // The count at input capture B event.
    uint32_t capt_c_count;          // The count at input capture C event.
    uint32_t capt_d_count;          // The count at input capture D event.
    uint32_t timer_count;           // The current channel counter value.
} mtu_capture_status_t;

typedef struct mtu_pwm_status_s
{
    bool_t  running;
    uint16_t pwm_timer_count;      // The current channel counter value.
    uint16_t pwm_a_value;          // The count at input capture A event.
    uint16_t pwm_b_value;          // The count at input capture B event.
    uint16_t pwm_c_value;          // The count at input capture C event.
    uint16_t pwm_d_value;          // The count at input capture D event.
} mtu_pwm_status_t;
```

Figure 5.6 Structures, Unions, and Enumerations for the Sample Drivers (5)

```
/************ Type defines used for callback functions. ***************/
/* Specifies the timer to which an operation is associated. Returned in callback data structure. */
typedef enum
{
    MTU_TIMER_A = 0,    //Corresponds to MTU TGRA register operations
    MTU_TIMER_B,        //Corresponds to MTU TGRB register operations
    MTU_TIMER_C,        //Corresponds to MTU TGRC register operations
    MTU_TIMER_D,        //Corresponds to MTU TGRD register operations
    MTU_TIMERS_MAX
} mtu_timer_num_t;

/************ Type defines used for callback functions. ***************/
/* Data structure passed to User callback upon pwm interrupt. */
typedef struct mtu_callback_data_s
{
    mtu_channel_t   channel;
    mtu_timer_num_t timer_num;
    uint32_t        count;
} mtu_callback_data_t;

/************ Type defines used with the R_MTU_Timer_Open and R_MTU_Capture_Open functions. ***************/
typedef struct mtu_timer_clk_src_s
{
    mtu_clk_sources_t   source;      // Internal clock or external clock input
    mtu_clk_edges_t     clock_edge;  // Specify the clock active edge.
} mtu_clk_src_t;
```

Figure 5.7 Structures, Unions, and Enumerations for the Sample Drivers (6)

```
/************ Type defines used with the R_MTU_Capture_Open function. ***************/
typedef enum
{
    MTU_CAP_SRC_A = 0,
    MTU_CAP_SRC_B,
    MTU_CAP_SRC_C,
    MTU_CAP_SRC_D
} mtu_cap_src_t;

/* The possible settings for input capture signal active edge. Register setting values. */
typedef enum mtu_cap_edges_e
{
    MTU_CAP_RISING_EDGE  = 0x08,
    MTU_CAP_FALLING_EDGE = 0x09,
    MTU_CAP_ANY_EDGE     = 0x0A,
} mtu_cap_edges_t;

typedef struct mtu_capture_set_edge_s // Used with the MTU_TIMER_CMD_SET_CAPT_EDGE command.
{
    mtu_cap_src_t    capture_src;        // The capture source.
    mtu_cap_edges_t  capture_edge;       // Specify transition polarities.
} mtu_capture_set_edge_t;

typedef struct mtu_capture_settings_s
{
    mtu_actions_t    actions;
    mtu_cap_edges_t  capture_edge;       // Specify transition polarities.
    bool_t           filter_enable;      // Noise filter on or off.
} mtu_capture_settings_t;
```

Figure 5.8 Structures, Unions, and Enumerations for the Sample Drivers (7)

```
typedef struct mtu_capture_chnl_settings_s
{
    mtu_clk_src_t     clock_src;   // Specify clocking source.
    mtu_src_clk_divisor_t   clock_div;   // Internal clock divisor selection.
    mtu_clear_src_t         clear_src;   // Specify the counter clearing source.
    mtu_capture_settings_t  capture_a;
    mtu_capture_settings_t  capture_b;
    mtu_capture_settings_t  capture_c;
    mtu_capture_settings_t  capture_d;
} mtu_capture_chnl_settings_t;

/************ Type defines used with the R_MTU_Timer_Open function. ***************/
typedef struct mtu_timer_actions_config_s
{
    mtu_actions_t           do_action;  // Various actions that can be done at timer event.
    mtu_output_states_t     output;     // Output pin transition type when output action is selected.
} mtu_timer_actions_cfg_t;

typedef struct mtu_timer_settings_s
{
    uint32_t   freq; // If internal clock source, the desired event frequency, or if external the Compare-match
count.
    mtu_timer_actions_cfg_t actions;
} mtu_timer_settings_t;

typedef struct mtu_timer_chnl_settings_s
{
    mtu_clk_src_t  clock_src;      // Specify clocking source.
    mtu_clear_src_t      clear_src;      // Specify the counter clearing source.
    mtu_timer_settings_t timer_a;
    mtu_timer_settings_t timer_b;
    mtu_timer_settings_t timer_c;
    mtu_timer_settings_t timer_d;
} mtu_timer_chnl_settings_t;
```

Figure 5.9 Structures, Unions, and Enumerations for the Sample Drivers (8)

```
/************ Type defines used with the R_MTU_PWM_Open function. ***************/
/* Available PWM operating modes. */
typedef enum mtu_pwm_mode_e
{
    MTU_PWM_MODE_1 = 0x02,
    MTU_PWM_MODE_2 = 0x03
} mtu_pwm_mode_t;

typedef struct mtu_pwm_settings_s
{
    uint16_t             duty;
    mtu_actions_t        actions;
    mtu_output_states_t  outputs;       // Specify transition polarities.
} mtu_pwm_settings_t;

typedef struct mtu_pwm_chnl_settings_s
{
    mtu_clk_src_t       clock_src;   // Specify clocking source.
    uint32_t                cycle_freq;  // Cycle frequency for the channel
    mtu_clear_src_t         clear_src;   // Specify the counter clearing source.
    mtu_pwm_mode_t          pwm_mode;    // Specify mode 1 or mode 2
    mtu_pwm_settings_t  pwm_a;
    mtu_pwm_settings_t  pwm_b;
    mtu_pwm_settings_t  pwm_c;
    mtu_pwm_settings_t  pwm_d;
} mtu_pwm_chnl_settings_t;
```

Figure 5.10 Structures, Unions, and Enumerations for the Sample Drivers (9)

```
/************ Type defines used with the R_MTU_PWM_Compliment_Open function. ***************/
typedef enum
{
    MTU_CHANNEL_3_4 = 0,
    MTU_CHANNEL_6_7,
    MTU_CMPL_PWM_CHANNEL_MAX
} mtu_cmpl_pwm_channel_t;

typedef enum
{
    MTU_TOGGLE_OFF = 0x00,  // Output toggle OFF
    MTU_TOGGLE_ON  = 0x01,  // Output toggle ON
} mtu_cmpl_pwm_toggle_t;

typedef enum
{
   MTU_CMPL_PWM_MODE_1 = 0x0D,  // Complimentary PWM mode 1
    MTU_CMPL_PWM_MODE_2 = 0x0E,  // Complimentary PWM mode 2
    MTU_CMPL_PWM_MODE_3 = 0x0F,  // Complimentary PWM mode 3
} mtu_cmpl_pwm_mode_t;

typedef enum
{
   MTU_PIN_P_N_1 = 0x00, // TOCR1
    MTU_PIN_P_N_2 = 0x01, // TOCR2
} mtu_cmpl_pwm_p_n_t;

typedef enum
{
    MTU_PIN_P_N_BF_OFF          = 0x00,  // Does not transfer
    MTU_PIN_P_N_BF_CREST        = 0x01,  // Transfer in TCNT Crest
    MTU_PIN_P_N_BF_TROUGH       = 0x02,  // Transfer in TCNT trough
    MTU_PIN_P_N_BF_CREST_TROUGH = 0x03,  // Transfer in TCNT crest and trough
} mtu_cmpl_pwm_p_n_bf_t;
```

Figure 5.11 Structures, Unions, and Enumerations for the Sample Drivers (10)

```
typedef enum
{
   MTU_CMPL_PWM_D_BF_OFF = 0,  // OFF
    MTU_CMPL_PWM_D_BF_ON        // ON
}mtu_cmpl_pwm_d_bf_t;

typedef enum
{
   MTU_PIN_OLSN_HI_UPHI_DNLO = 0x00, // Init High Active Low Up High Down Low
    MTU_PIN_OLSN_LO_UPLO_DNHI = 0x01, // Init Low Active High Up Low Down High
} mtu_cmpl_pwm_olsn_t;

typedef enum
{
   MTU_PIN_OLSP_HI_UPLO_DNHI = 0x00, // Init High Active Low Up Low Down High
    MTU_PIN_OLSP_LO_UPHI_DNLO = 0x01, // Init Low Active High Up High Down Low
} mtu_cmpl_pwm_olsp_t;

typedef enum
{
    MTU_CMPL_PWM_ST_COUNT_OFF    = 0,
    MTU_CMPL_PWM_ST_COUNT_ON,
} mtu_cmpl_pwm_count_st_t;

typedef enum
{
    MTU_CMPL_PWM_DIRECT_DOWN    = 0,
    MTU_CMPL_PWM_DIRECT_UP,
} mtu_cmpl_pwm_direction_t;
```

Figure 5.12 Structures, Unions, and Enumerations for the Sample Drivers (11)

```
typedef enum
{
    MTU_CMPL_PWM_OUTPUT_OFF    = 0,
    MTU_CMPL_PWM_OUTPUT_ON,
} mtu_cmpl_pwm_output_st_t;

typedef enum
{
    MTU_CMPL_PWM_PROTECT_OFF    = 0,
    MTU_CMPL_PWM_PROTECT_ON,
} mtu_cmpl_pwm_reg_protect_t;

typedef struct mtu_cmpl_pwm_clk_div_s
{
    mtu_src_clk_divisor_t clock_div;       // Internal clock divisor selection.
    uint16_t              cycle_freq;      // Cycle
} mtu_cmpl_pwm_clk_div_t;

typedef struct mtu_cmpl_pwm_settings_s
{
    mtu_cmpl_pwm_olsp_t       olsp; // Output Level Select P
    mtu_cmpl_pwm_olsn_t       olsn; // Output Level Select N
    uint16_t                  duty; // Duty cycle (Unit 0.1%)
    mtu_cmpl_pwm_output_st_t  output; // PWM output
} mtu_cmpl_pwm_settings_t;
```

Figure 5.13 Structures, Unions, and Enumerations for the Sample Drivers (12)

```
typedef struct mtu_cmpl_pwm_chnl_settings_s
{
    mtu_clk_src_t              clock_src;    // Specify clocking source.
    mtu_cmpl_pwm_clk_div_t     clk_div;      // Internal clock divisor selection.
    uint16_t                   dead_time;    // Dead time
    mtu_cmpl_pwm_toggle_t      toggle;       // Output toggle
    mtu_cmpl_pwm_mode_t        mode;         // Complimentary PWM mode
    mtu_cmpl_pwm_p_n_t         p_n;          // TOC Select
    mtu_cmpl_pwm_p_n_bf_t      p_n_bf;       // Buffer output level
    mtu_cmpl_pwm_d_bf_t        d_bf;         // Double buffer select
    mtu_cmpl_pwm_reg_protect_t protect;      // register protect
    mtu_cmpl_pwm_settings_t    pwm_output_1; // PWM output 1
    mtu_cmpl_pwm_settings_t    pwm_output_2; // PWM output 2
    mtu_cmpl_pwm_settings_t    pwm_output_3; // PWM output 3
} mtu_cmpl_pwm_chnl_settings_t;

typedef struct mtu_cmpl_pwm_chnl_status_s
{
    mtu_cmpl_pwm_count_st_t  c_st;
    mtu_cmpl_pwm_direction_t d_st;
} mtu_cmpl_pwm_chnl_status_t;
```

Figure 5.14 Structures, Unions, and Enumerations for the Sample Drivers (13)

5.9 Functions

The table below lists the functions.

Table 5.6 List of Functions

Function	Page Number
main	25
SER_Init	25
mtu3_setup	25
R_MTU_Init	26
R_MTU_Uninit	26
R_MTU_PWM_Compliment_Open	26
R_MTU_PWM_Compliment_Close	27
R_MTU_PWM_Compliment_Control	27
R_MTU_Timer_Open	28
R_MTU_Capture_Open	28
R_MTU_PWM_Open	29
R_MTU_Close	29
R_MTU_Control	30
R_MTU_GetVersion	30

5.10 Specifications of the Functions of the Sample Program

5.10.1 main

main	
Header	
Declaration	int32_t main(void)
Description	This function makes settings for complementary PWM mode 1 and outputs three-phase PWM signals by using MTU3 and MTU4.
Arguments	None
Return values	Return from this function does not proceed because it is an endless loop.
Remarks	—

5.10.2 SER_Init

SER_init	
Header	Serial.h
Declaration	void SERInit(void)
Description	This function initializes the serial port.
Arguments	None
Return values	None
Remarks	—

5.10.3 mtu3_setup

mtu3setup	
Header	
Declaration	int32_t mtu3_setup(void)
Description	This function sets complementary PWM.
Arguments	None
Return values	ESUCCESS, EACESS, EFAULT, EBUSY
Remarks	—

5.11 Specifications of the Functions of the MTU Driver

5.11.1 R_MTU_Init

R_MTU_Init	
Header	r_mtu3_if.h
Declaration	void R_MTU_Init(void)
Description	This function initializes the MTU driver.
Arguments	None
Return values	None
Remarks	—

5.11.2 R_MTU_Uninit

R_MTU_Uninit	
Header	r_mtu3_if.h
Declaration	Int32_t R_MTU_Uninit(void)
Description	This function ends the MTU driver to stop the MTU3a device.
Arguments	None
Return values	ESUCCESS, EACCES, EFAULT
Remarks	—

5.11.3 R_MTU_PWM_Compliment_Open

R_MTU_PWM_Complment_Open		
Synopsis	Processing to set up complementary PWM mode	
Header	r_mtu3_if.h	
Declaration	int32_t R_MTU_PWM_Compliment_Open (mtu_cmpl_pwm_channel_t const channel, mtu_cmpl_pwm_chnl_settings_t * const pconfig)	
Description	This function makes settings to start operations for output in complementary PWM mode.	
Arguments	channel	Complementary PWM mode channels MTU_CHANNEL_3_4 MTU_CHANNEL_6_7
	pconfig	Pointer to the complementary PWM mode setting information
Return values	ESUCCESS, EACESS, EFAULT, EINVAL, EBUSY	
Remarks	Call R_MTU_Init beforehand to initialize the MTU driver. After operations in complementary PWM mode end, call function R_MTU_PWM_Compliment_Close to close the results of this function.	

5.11.4 R_MTU_PWM_Compliment_Close

R_MTU_PWM_Compliment_Close		
Synopsis	Ending complementary PWM operations	
Header	r_mtu3_if.h	
Declaration	int32_t R_MTU_PWM_Compliment_Close(mtu_cmpl_pwm_channel_t const channel)	
Description	This function ends operations for output in complementary PWM mode.	
Arguments	channel	Complementary PWM mode channels MTU_CHANNEL_3_4 MTU_CHANNEL_6_7
Return values	ESUCCESS, EACESS, EFAULT, EINVAL, EBUSY	
Remarks	After executing function R_MTU_PWM_Compliment_Open, the channels will be open.	

5.11.5 R_MTU_PWM_Compliment_Control

R_MTU_PWM_Compliment_Control		
Synopsis	Processing for complementary PWM control	
Header	r_mtu3_if.h	
Declaration	int32_t R_MTU_PWM_Compliment_Control(mtu_cmpl_pwm_channel_t const channel, mtu_cmd_t const cmd, void * const pcmd_data)	
Description	This function makes settings for complementary PWM mode operations by the specified channels. For details of the commands, see Section 5.14.	
Arguments	Channel	Complementary PWM mode channels MTU_CHANNEL_3_4 MTU_CHANNEL_6_7
	Cmd	Complementary PWM control commands MTU_CMD_START MTU_CMD_STOP MTU_CMD_GET_STATUS
	pcmd_data	Pointer to the complementary PWM control command parameter information
Return values	ESUCCESS, EINVAL, EACESS, EFAULT, EBUSY	
Remarks	After executing function R_MTU_PWM_Compliment_Open, the channels will be open.	

5.11.6 R_MTU_Timer_Open

R_MTU_Timer_Open		
Synopsis	Compare match settings for the MTU	
Header	r_mtu3_if.h	
Declaration	int32_t R_MTU_Timer_Open(mtu_channel_t const channel, mtu_timer_chnl_settings_t * const pconfig, void (* const pcallback)(void * pdata))	
Description	This function makes settings for the timing of compare match for each MTU channel. For details of the pointer to the setting information (pconfig), see Section 5.15.	
Arguments	channel	MTU channels: MTU_CHANNEL_0 to MTU_CHANNEL_8
	pconfig	Pointer to the compare-match timing setting information
	pcallback	Pointer to the callback function
Return values	ESUCCESS, EINVAL, EACESS, EFAULT, EBUSY	
Remarks	Call R_MTU_Init beforehand to initialize the MTU driver. After the completion of the timing of matches in comparison by the MTU, call function R_MTU_Close to end operation of the MTU.	

5.11.7 R_MTU_Capture_Open

R_MTU_Capture_Open		
Synopsis	Capture operation settings for the MTU	
Header	r_mtu3_if.h	
Declaration	int32_t R_MTU_Capture_Open(mtu_channel_t const channel, mtu_capture_chnl_settings_t * const pconfig, void (* const pcallback)(void * pdata))	
Description	This function makes settings for capture operation for each specified MTU channel. For details of the pointer to the capture setting information (pconfig), see Section 5.16.	
Arguments	channel	MTU channels: MTU_CHANNEL_0 to MTU_CHANNEL_8
	pconfig	Pointer to the capture setting information
	pcallback	Pointer to the callback function
Return values	ESUCCESS, EINVAL, EACESS, EFAULT, EBUSY	
Remarks	Call R_MTU_Init beforehand to initialize the MTU driver. After capture operation of the MTU ends, call function R_MTU_Close to end operation of the MTU.	

5.11.8 R_MTU_PWM_Open

R_MTU_PWM_Open		
Synopsis	PWM operation settings for the MTU	
Header	r_mtu3_if.h	
Declaration	int32_t R_MTU_PWM_Open(mtu_channel_t const channel, mtu_pwm_chnl_settings_t * const pconfig, void (* const pcallback)(void * pdata))	
Description	This function makes settings for basic operation in the PWM mode for each specified MTU channel. For the PWM mode setting information (pconfig), see Section 5.13.	
Arguments	channel	MTU channels: MTU_CHANNEL_0 to MTU_CHANNEL_8 Note: When PWM mode 2 is specified, only channels 0, 1, and 2 can be set.
	pconfig	Pointer to the PWM mode setting information
	pcallback	Pointer to the callback function
Return values	ESUCCESS, EINVAL, EACESS, EFAULT, EBUSY	
Remarks	Call R_MTU_Init beforehand to initialize the MTU driver. After PWM operation of the MTU ends, call function R_MTU_Close to end operation of the MTU.	

5.11.9 R_MTU_Close

R_MTU_Close		
Synopsis	Ending operation of the MTU	
Header	r_mtu3_if.h	
Declaration	int32_t R_MTU_Close (mtu_channel_t const channel)	
Description	This function ends operation of the specified MTU channels.	
Arguments	channel	MTU channels: MTU_CHANNEL_0 to MTU_CHANNEL_8
Return values	ESUCCESS, EINVAL, EACESS, EFAULT, EBUSY	
Remarks	After executing function R_MTU_Timer_Open, R_MTU_Capture_Open, or R_MTU_PWM_Open, the MTU channels will be open.	

5.11.10 R_MTU_Control

R_MTU_Control		
Synopsis	Processing for MTU control	
Header	r_mtu3_if.h	
Declaration	int32_t R_MTU_Control(mtu_channel_t const channel, mtu_cmd_t const cmd, void * const pcmd_data)	
Description	This function makes settings for operation of each specified MTU channel. For details of the commands, see Section 5.18.	
Arguments	channel	MTU channels: MTU_CHANNEL_0 to MTU_CHANNEL_8
	cmd	MTU control commands MTU_CMD_START MTU_CMD_STOP MTU_CMD_SAFE_STOP MTU_CMD_RESTART MTU_CMD_SYNCHRONIZE MTU_CMD_GET_STATUS MTU_CMD_SET_CAPT_EDGE
	pcmd_data	Pointer to the MTU control command parameter information
Return values	ESUCCESS, EINVAL, EACESS, EFAULT, EBUSY	
Remarks	After executing function R_MTU_Timer_Open, R_MTU_Capture_Open, or R_MTU_PWM_Open, the MTU channels will be open.	

5.11.11 R_MTU_GetVersion

R_MTU_GetVersion		
Synopsis	Acquiring the version number	
Header	r_mtu3_if.h	
Declaration	uint32_t R_MTU_GetVersion(void)	
Description	This function acquires the version number of the MTU driver.	
Arguments	None	
Return values	Version number	0 to 15: Minor part of the version number 16 to 31: Major part of the version number
Remarks	—	

5.12 Flowcharts

5.12.1 Main Processing of the Sample Application

Figure 5.15 is a flowchart of main processing of the sample application.

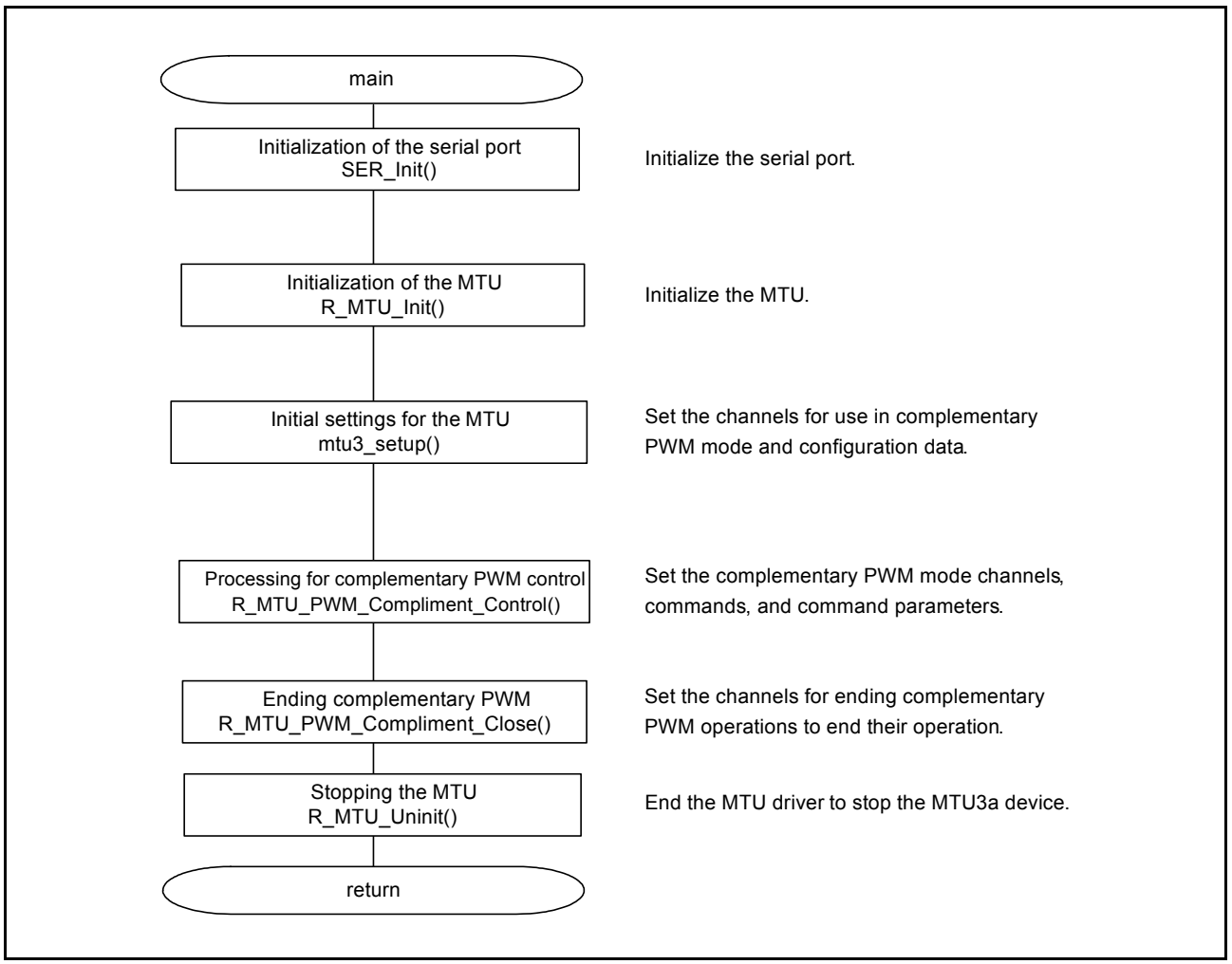

Figure 5.15 Main Processing of the Sample Application

5.12.2 Processing for Complementary PWM Mode Control

Figure 5.16 is a flowchart of processing to set up complementary PWM mode.

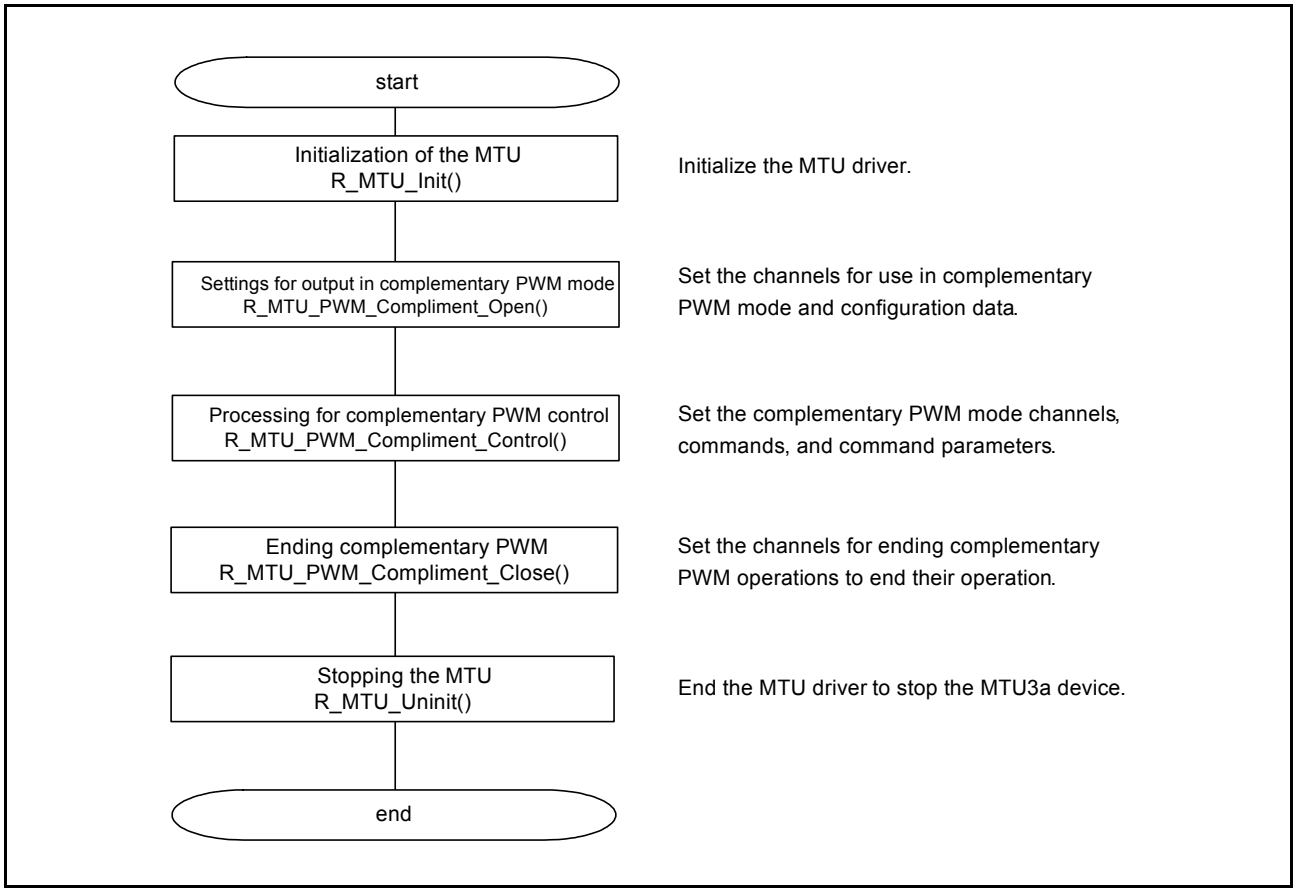

Figure 5.16 Processing to Set Up Complementary PWM Mode

5.12.3 Compare Match Timing

Figure 5.17 is a flowchart of processing for the timing of matches in comparison by the MTU.

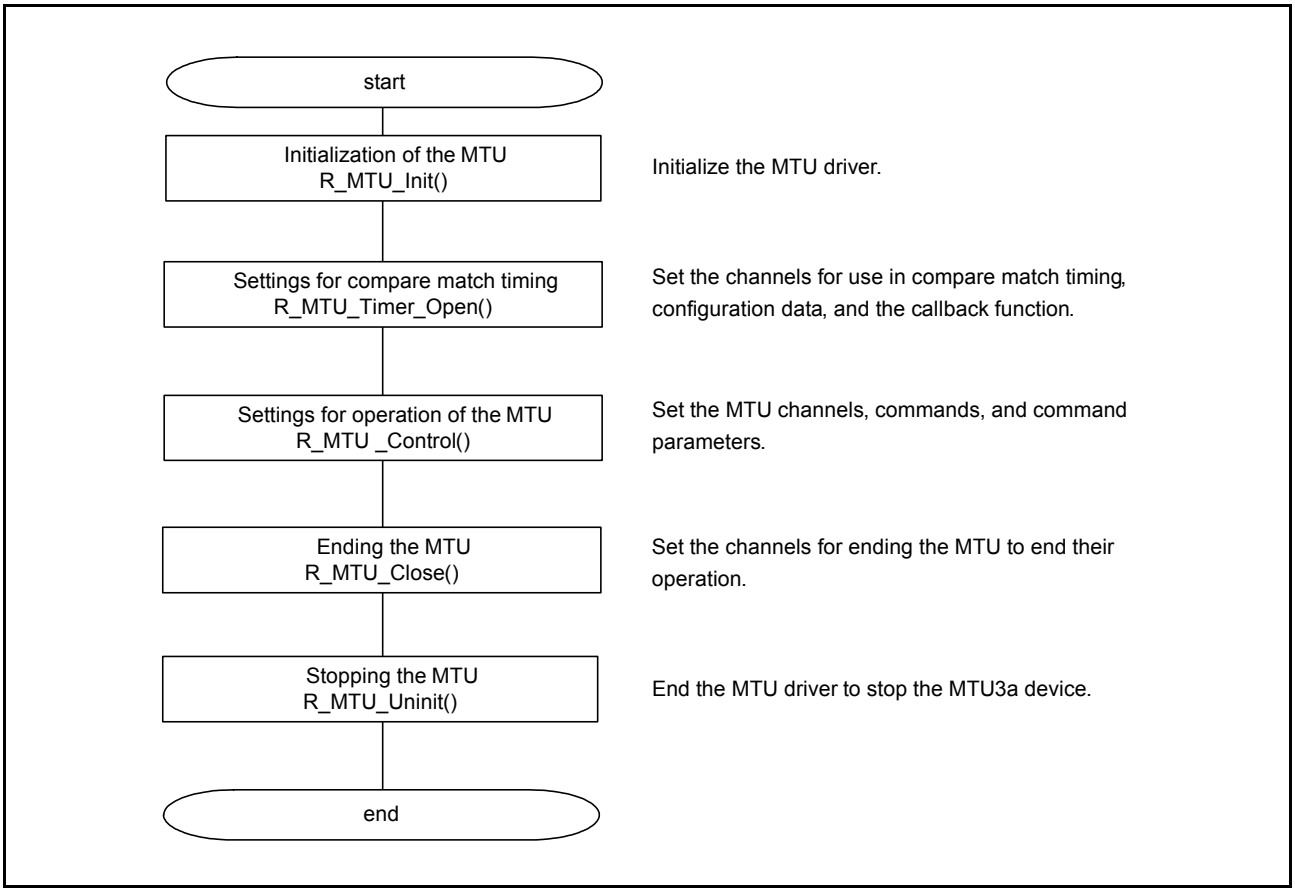

Figure 5.17 Compare Match Timing

5.12.4 Capture Processing

Figure 5.18 is a flowchart of capture processing by the MTU.

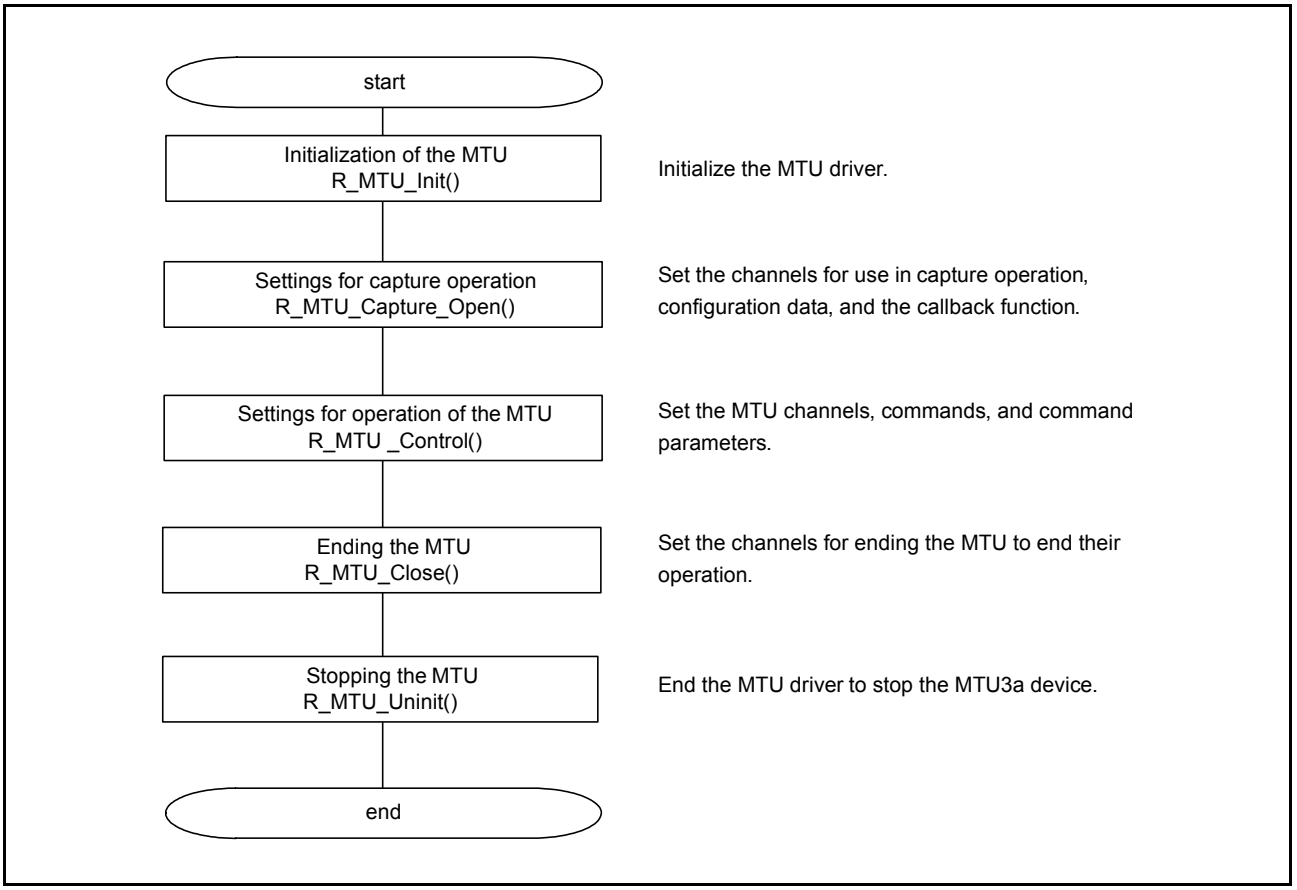

Figure 5.18 Capture Processing

5.12.5 Processing for PWM Mode Control

Figure 5.19 is a flowchart of processing for PWM mode control.

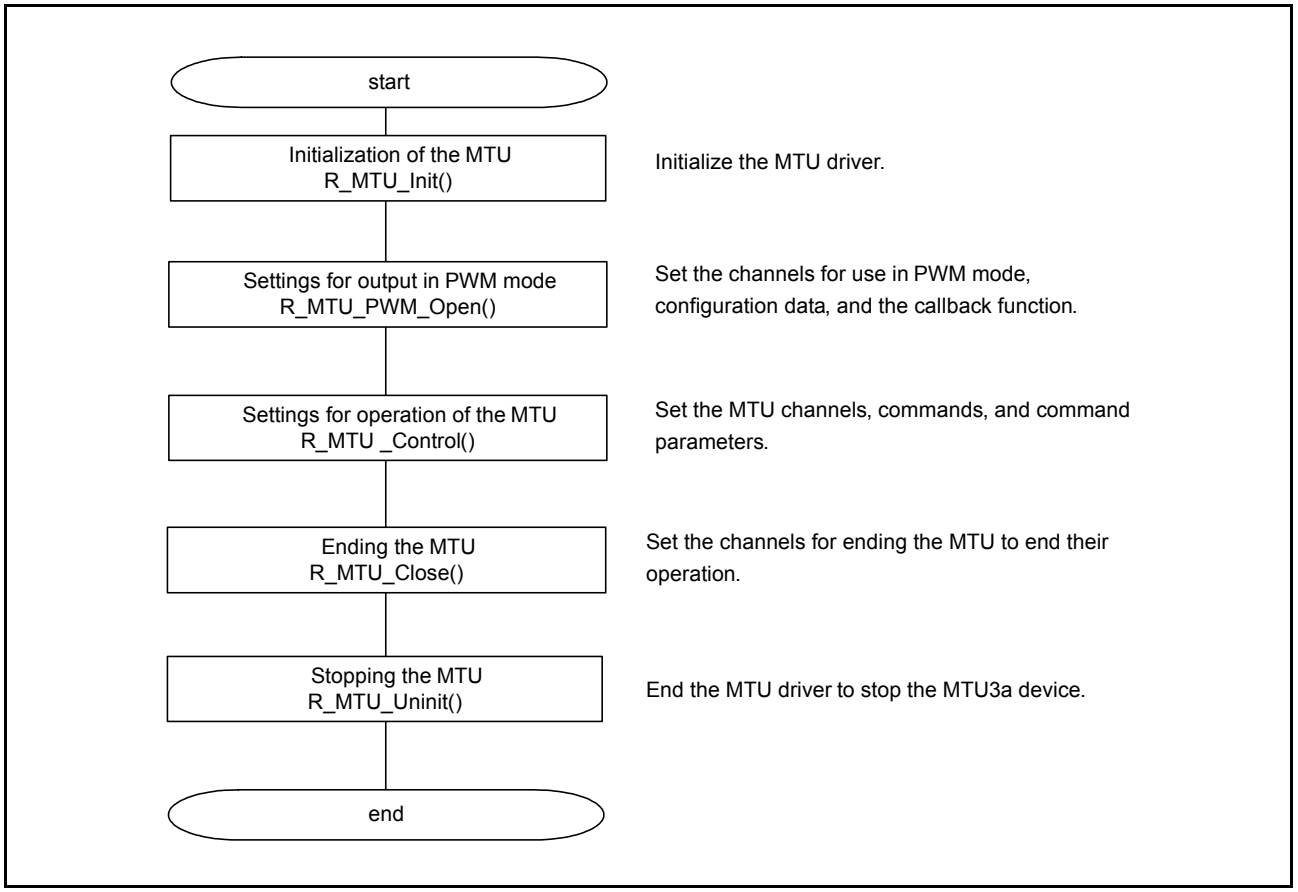

Figure 5.19 Processing for PWM Mode Control

5.13 R_MTU_PWM_Compliment_Open Parameters

The table below list the parameters (*pconfig) that are used for function R_MTU_PWM_Compliment_Open.

Table 5.7 R_MTU_PWM_Compliment_Open Parameters

Parameter	Outline
clock_src.source	Sets the clock source for counting.
clock_src.clock_edge	Selects edges for the clock.
clk_div.clock_div	Sets the frequency division ratio of the clock signal for counting.
clk_div.cycle_freq	Sets the clock counter value.
dead_time	Sets dead time.
toggle	Sets toggle output in synchronization with the carrier cycle.
mode	Sets the operating mode for complementary PWM.
p_n	Sets operation for the output level.
p_n_bf	Sets buffered operation for the output level.
d_bf	Sets the use of buffers.
protect	Sets protection against accidental writing in MTU3 and MTU4.
pwm_output_X.olsp (X = 1 to 3)	Sets the timing of the output levels in the normal phase for PWM output X (X = 1 to 3).
pwm_output_X.olsn (X = 1 to 3)	Sets the timing of the output levels in the inverse phase for PWM output X (X = 1 to 3).
pwm_output_X.duty (X = 1 to 3)	Sets the duty cycle for PWM output X (X = 1 to 3).
pwm_output_X.output (X = 1 to 3)	Enables or disables output from the pins for PWM output X (X = 1 to 3).

The details of these parameters are listed below.

5.13.1 clock_src.source

clock_src.source

Synopsis	Setting the clock source for counting	
Header	r_mtu3_if.h	
Description	This parameter sets the clock source for counting by the MTU channels (MTU3 and MTU4 or MTU6 and MTU7). When the external clock is selected, the I/O port settings for the target pin and in the multi-function pin controller (MPC) must be made in advance.	
Parameters	MTU_CLK_SRC_EXT_MTCLKA*1	Specifies an external clock signal (input on the MTCLKA pin).
	MTU_CLK_SRC_EXT_MTCLKB*1	Specifies an external clock signal (input on the MTCLKB pin).
	MTU_CLK_SRC_INTERNAL	Specifies the internal clock (PCLKC).
Remarks	Note 1. Only available when channels 3 and 4 are selected.	

5.13.2 clock_src.clock_edge

clock_src.clock_edge		
Synopsis	Selecting edges for the clock	
Header	r_mtu3_if.h	
Description	This parameter selects the edge for counting of the input clock cycles by the MTU channels (MTU3 and MTU4 or MTU6 and MTU7).	
Parameters	MTU_CLK_RISING_EDGE	Specifies counting of rising edges.
	MTU_CLK_FALLING_EDGE	Specifies counting of falling edges.
	MTU_CLK_ANY_EDGE	Specifies counting of rising and falling edges.
Remarks	When the timer cycle specified for the counter clock is PCLKC/1 (clk_div), the setting for the selection of edges has no effect and operation is always as if the initial value (rising edges) has remained.	

5.13.3 clk_div.clock_div

clk_div.clock_div		
Synopsis	Setting the frequency division ratio of the clock signal for counting	
Header	r_mtu3_if.h	
Description	This parameter selects the frequency division ratio of the clock signal for the MTU channels (MTU3 and MTU4 or MTU6 and MTU7).	
Parameters	MTU_SRC_CLK_DIV_1	Counted on PCLKC/1.
	MTU_SRC_CLK_DIV_2	Counted on PCLKC/2.
	MTU_SRC_CLK_DIV_4	Counted on PCLKC/4.
	MTU_SRC_CLK_DIV_8	Counted on PCLKC/8.
	MTU_SRC_CLK_DIV_16	Counted on PCLKC/16.
	MTU_SRC_CLK_DIV_32	Counted on PCLKC/32.
	MTU_SRC_CLK_DIV_64	Counted on PCLKC/64.
	MTU_SRC_CLK_DIV_256	Counted on PCLKC/256.
	MTU_SRC_CLK_DIV_1024	Counted on PCLKC/1024.
Remarks	—	

5.13.4 clk_div.cycle_freq

clk_div.cycle_freq		
Synopsis	Setting the carrier cycle for complementary PWM	
Header	r_mtu3_if.h	
Description	This parameter sets the carrier cycle for the MTU channels (MTU3 and MTU4 or MTU6 and MTU7). The timer cycle data registers (TCDRA, TCDRB) are set to 1/2 of this value.	
Parameters	Value	Carrier cycle setting (2 to 1FFFEh)
Remarks	—	

5.13.5 dead_time

dead_time		
Synopsis	Setting dead time for complementary PWM	
Header	r_mtu3_if.h	
Description	This parameter specifies the value to be set in the dead time data registers (TDDRA, TDDRB) as dead time for complementary PWM.	
Parameters	Value	Value to be set in the dead time data registers (0 to FFFFh)
Remarks	—	

5.13.6 toggle

toggle		
Synopsis	Setting toggle output in synchronization with the carrier cycle	
Header	r_mtu3_if.h	
Description	This parameter selects whether to toggle output in synchronization with the carrier cycle from MTIOC3A or MTIOC6A. When MTIOC3A and MTIOC6A are used, the I/O port settings for the target pin and in the multi-function pin controller (MPC) must be made in advance.	
Parameters	MTU_TOGGLE_OFF	Toggle output off
	MTU_TOGGLE_ON	Toggle output on
Remarks	This type of output is basically not used in complementary PWM output. It is provided for use in evaluation and so on.	

5.13.7 mode

mode		
Synopsis	Setting the operating mode for complementary PWM	
Header	r_mtu3_if.h	
Description	This parameter selects the operating mode for complementary PWM from among modes 1 to 3.	
Parameters	MTU_CMPL_PWM_MODE_1	Complementary PWM mode 1 (to be transferred at crest)
	MTU_CMPL_PWM_MODE_2	Complementary PWM mode 2 (to be transferred at trough)
	MTU_CMPL_PWM_MODE_3	Complementary PWM mode 3 (to be transferred at crest or trough)
Remarks	—	

5.13.8 p_n

p_n		
Synopsis	Setting operation for the output level	
Header	r_mtu3_if.h	
Description	This parameter selects the setting to fix the phase of complementary PWM output (normal or inverted) or the setting for buffered operation. When the output level is set to fixed, the output level can be determined by setting timer output control registers 1 (TOCR1A, TOCR1B). When the output level is set for buffered operation, the output level can be determined by setting timer output control registers 2 (TOCR2A, TOCR2B) or the buffer registers (TOLBRA, TOLBRB).	
Parameters	MTU_PIN_P_N_1*1	Fixes the output levels in normal and inverse phases.
	MTU_PIN_P_N_2	Selects buffered operation for the output levels in normal and inverse phases.
Remarks	Note 1. When the output level is set to fixed, the output levels in normal and inverse phases are the common settings for PWM output 1 to 3.	

5.13.9 p_n_bf

p_n_bf		
Synopsis	Setting buffered operation for the output level	
Header	r_mtu3_if.h	
Description	This parameter selects the transfer trigger for setting the output level (normal or inverse phase) from the buffer for complementary PWM output.*1	
Parameters	MTU_PIN_P_N_BF_OFF	No transfer
	MTU_PIN_P_N_BF_CREST	To be transferred at crest.
	MTU_PIN_P_N_BF_TROUGH	To be transferred at trough.
	MTU_PIN_P_N_BF_CREST_TROUGH	To be transferred at crest or trough.
Remarks	Note 1. This parameter is only valid when MTU_PIN_P_N_2 is selected by using p_n.	

5.13.10 d_bf

d_bf		
Synopsis	Setting the use of buffers	
Header	r_mtu3_if.h	
Description	This parameter enables or disables the use of buffers for complementary PWM output.*1 Enabling this setting changes the finest resolution of changes in the PWM output from ±2 to ±1.	
Parameters	MTU_CMPL_PWM_D_BF_OFF	Disables the use of buffers.
	MTU_CMPL_PWM_D_BF_ON	Enables the use of buffers.
Remarks	Note 1. The use of buffers can only be specified when complementary PWM mode 3 (transfer at crest or trough) is selected.	

5.13.11 protect

protect		
Synopsis	Setting protection against accidental writing in MTU3 and MTU4	
Header	r_mtu3_if.h	
Description	This parameter sets protection against accidental writing to the target registers or counters of MTU3 and MTU4.	
Parameters	MTU_CMPL_PWM_PROTECT_OFF	Enables reading and writing.
	MTU_CMPL_PWM_PROTECT_ON	Disables reading and writing.
Remarks	For details of the target registers and counters for protection against accidental writing, see the description of the timer read/write enable registers (TRWERA, TRWERB).	

5.13.12 pwm_output_X.olsp (X = 1 to 3)

pwm_output_X.olsp		
Synopsis	Setting the timing of the output levels in the normal phase for PWM output X (X = 1 to 3)	
Header	r_mtu3_if.h	
Description	This parameter sets the timing of the output levels in the normal phase for complementary PWM output X (X = 1 to 3).	
Parameters	MTU_PIN_OLSP_HI_UPLO_DNHI	High level for initial output, low level while counting up, high level while counting down
	MTU_PIN_OLSP_LO_UPHI_DNLO	Low level for initial output, high level while counting up, low level while counting down
Remarks	—	

5.13.13 pwm_output_X.olsn (X = 1 to 3)

pwm_output_X.olsn		
Synopsis	Setting the timing of the output levels in the inverse phase for PWM output X (X = 1 to 3)	
Header	r_mtu3_if.h	
Description	This parameter sets the timing of the output levels in the inverse phase for complementary PWM output X (X = 1 to 3).	
Parameters	MTU_PIN_OLSN_HI_UPHI_DNLO	High level for initial output, high level while counting up, low level while counting down
	MTU_PIN_OLSN_LO_UPLO_DNHI	Low level for initial output, low level while counting up, high level while counting down
Remarks	—	

5.13.14 pwm_output_X.duty (X = 1 to 3)

pwm_output_X.duty		
Synopsis	Setting the duty cycle for PWM output X (X = 1 to 3)	
Header	r_mtu3_if.h	
Description	This parameter sets the duty cycle for PWM output X (X = 1 to 3). The duty cycle is specified in 0.1% units. The duty cycle can be set in the range from 0% to 100%.	
Parameters	Value (in 0.1% units)	The duty cycle is specified in 0.1% units (0 to 1000).
Remarks	—	

5.13.15 pwm_output_X.output (X = 1 to 3)

pwm_output_X.duty		
Synopsis	Enabling or disabling output from the pins for PWM output X (X = 1 to 3)	
Header	r_mtu3_if.h	
Description	This parameter enables or disables output from the pins for PWM output X (X = 1 to 3) independently for each combination of normal and inverse phases. When MTU3 and MTU4 are selected PWM output 1: MTIOC3B (normal phase), MTIOC3D (inverse phase) PWM output 2: MTIOC4A (normal phase), MTIOC4C (inverse phase) PWM output 3: MTIOC4B (normal phase), MTIOC4D (inverse phase) When MTU6 and MTU7 are selected PWM output 1: MTIOC6B (normal phase), MTIOC6D (inverse phase) PWM output 2: MTIOC7A (normal phase), MTIOC7C (inverse phase) PWM output 3: MTIOC7B (normal phase), MTIOC7D (inverse phase) When PWM output pins are used, the I/O port settings for the target pin and in the multi-function pin controller (MPC) must be made in advance.	
Parameters	MTU_CMPL_PWM_OUTPUT_OFF	Not used as complementary PWM output.
	MTU_CMPL_PWM_OUTPUT_ON	Used as complementary PWM output.
Remarks	—	

5.14 R_MTU_PWM_Compliment_Control Parameters

The parameters that are used for function R_MTU_PWM_Compliment_Control (*pcmd_data) are listed below.

pcmd_data must be written in the format conforming to the command specified by cmd.

5.14.1 When cmd = MTU_CMD_START

This command is only effective for the first argument (specifying the channel) of function R_MTU_PWM_Compliment_Control. In this case, the value of the parameter (*pcmd_data) should be null rather than an actual address.

5.14.2 When cmd = MTU_CMD_STOP

This command is only effective for the first argument (specifying the channel) of function R_MTU_PWM_Compliment_Control. In this case, the value of the parameter (*pcmd_data) should be null rather than an actual address.

5.14.3 When cmd = MTU_CMD_GET_STATUS

The first address of a variable of the mtu_cmpl_pwm_chnl_status_t structure type is specified as the parameter (*pcmd_data). When a command is executed, the following parameters are acquired and the values are returned to the specified structure variable.

Table 5.8 Parameters when cmd = MTU_CMD_GET_STATUS

Parameter	Outline
c_st	State of counting by the timer (operating or stopped)
d_st	Direction of counting by the timer

5.15 R_MTU_Timer_Open Parameters

The parameters that are used for function R_MTU_Timer_Open (*pconfig) are listed below.

Table 5.9 R_MTU_Timer_Open Parameters

Parameter	Outline
clock_src.source	Sets the clock source for counting.
clock_src.clock_edge	Selects edges for the clock.
clear_src	Specifies the source for clearing the counter.
timer_X.actions.freq (X = a, b, c, d)	Sets the cycle for compare-matches (Hz) with a timer general register TGRX (X = a, b, c, d).
timer_X.actions.do_action (X = a, b, c, d)	Sets operation of a timer general register TGRX (X = a, b, c, d).
timer_X.actions.output (X = a, b, c, d)	Sets the pin output level for a timer general register TGRX (X = a, b, c, d).

5.15.1 clock_src.source

clock_src.source

Synopsis: Setting the clock source for counting

Header: r_mtu3_if.h

Description: This parameter sets the clock source for counting by an individual MTU channel.
When the external clock is selected, the I/O port settings for the target pin and in the multi-function pin controller (MPC) must be made in advance.
When the internal clock (PCLKC) is selected, a division ratio that is more suitable for the specified timer cycle (timer_X.actions.freq) is automatically selected.

Parameters:

MTU_CLK_SRC_EXT_MTCLKA	Specifies an external clock signal (input on the MTCLKA pin).
MTU_CLK_SRC_EXT_MTCLKB	Specifies an external clock signal (input on the MTCLKB pin).
MTU_CLK_SRC_EXT_MTCLKC*1	Specifies an external clock signal (input on the MTCLKC pin).
MTU_CLK_SRC_EXT_MTCLKD*2	Specifies an external clock signal (input on the MTCLKD pin).
MTU_CLK_SRC_CASCADE*3	Specifies overflows or underflows of MTU2.TCNT.
MTU_CLK_SRC_INTERNAL	Specifies an internal clock signal (PCLKC).

Remarks:
Note 1. Only available for channels 0 and 2.
Note 2. Only available for channel 0.
Note 3. Only available for channel 1.

5.15.2 clock_src.clock_edge

clock_src.clock_edge		
Synopsis	Selecting edges for the clock	
Header	r_mtu3_if.h	
Description	This parameter selects the edge for counting of the input clock cycles by each MTU channel.	
Parameters	MTU_CLK_RISING_EDGE	Specifies counting of rising edges.
	MTU_CLK_FALLING_EDGE	Specifies counting of falling edges.
	MTU_CLK_ANY_EDGE	Specifies counting of rising and falling edges.
Remarks	When the clock source to drive counting of the specified timer cycle (timer_X.actions.freq) is PCLKC/1 or is specified as overflows or underflows of MTU2.TCNT, the setting for the selection of edges has no effect and operation is always as if the initial value (rising edges) has remained.	

5.15.3 clear_src

clear_src		
Synopsis	Specifying the source for clearing the counter	
Header	r_mtu3_if.h	
Description	This parameter specifies the source for clearing the counter of the TCNT on an individual MTU channel.	
Parameters	MTU_CLR_TIMER_A	Specifies a compare match with TGRA.
	MTU_CLR_TIMER_B	Specifies a compare match with TGRB.
	MTU_CLR_TIMER_C*1	Specifies a compare match with TGRC.
	MTU_CLR_TIMER_D*1	Specifies a compare match with TGRD.
	MTU_CLR_SYNC	Specifies clearing of the counters of other channels that are in synchronous clearing or operation.
	MTU_CLR_DISABLED	Disables clearing of the TCNT.
Remarks	Note 1. Only available for channels 0, 3, 4, 6, 7, and 8.	

5.15.4 timer_X.actions.freq (X = a, b, c, d)

timer_X.actions.freq		
Synopsis	Setting the cycle for compare-matches (Hz) with a timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies the TGR cycle (Hz) for an individual MTU channel. When the internal clock (PCLKC) is selected as the counter clock, an appropriate division ratio of PCLKC and TGR value (for matching with the counter value) are automatically calculated. When an external clock is selected as the counter clock, the value of this parameter is simply set to the current value of the given TGR and is handled as the value for counting until a match in comparison.	
Parameters	Value (in Hz units)	Sets the cycle in Hz units. Specifies the TGR counter value (up to FFFFh) for an external clock.
Remarks	Allowable settings are from 3 to 100000000 Hz (1 to 100000000 Hz for channel 8). When both edges are selected by clock_src.clock_edge, the upper limit is 60000000 Hz.	

5.15.5 timer_X.actions.do_action (X = a, b, c, d)

timer_X.actions. do_action

Synopsis: Setting operation of a timer general register TGRX (X = a, b, c, d)

Header: r_mtu3_if.h

Description: This parameter specifies operation of the TGR on an individual MTU channel.
When the MTU pin is specified as an output pin, the I/O port settings for the target pin and in the multi-function pin controller (MPC) must be made in advance.
Multiple operations can be set simultaneously by this parameter as below.

Example) When my_timer_cfg is used as *pconfig:
my_timer_cfg.timer_a.actions.do_action
= (mtu_actions_t)(MTU_ACTION_OUTPUT | MTU_ACTION_INTERRUPT);

Parameters:

MTU_ACTION_OUTPUT	Specifies the MTU pin as an output pin.
MTU_ACTION_INTERRUPT*1	Enables compare-match interrupts.
MTU_ACTION_CALLBACK	Enables compare-match interrupts and specifies the user-specified callback function to be executed in the interrupt service routine.
MTU_ACTION_TRIGGER_ADC*2	Specifies activation by a trigger for the A/D converter in response to matches in comparison with TGRA.
MTU_ACTION_REPEAT*3	Specifies timer counter operation is to continue after the first match in comparison.
MTU_ACTION_NONE	This parameter is specified when the TGR is not used. Specifying the parameter with other parameters is prohibited.

Remarks:
Note 1. The interrupt priority level must be specified in advance by r_mtu3_config.h.
Note 2. Available for channels 0 to 4, 6, and 7.
Note 3. When this parameter is not specified, timer counter operation is stopped in the interrupt service routine. For this reason, timer counter operation always continues if enabling of interrupts (MTU_ACTION_INTERRUPT or MTU_ACTION_CALLBACK) is not specified.

5.15.6 timer_X.actions.output (X = a, b, c, d)

timer_X.actions.output		
Synopsis	Setting the pin output level for a timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies the pin output level for the TGR for an individual MTU channel.	
Parameters	MTU_PIN_NO_OUTPUT	Disables output.
	MTU_PIN_LO_GOLO	Specifies initial output as low and output of the low level in response to matches in comparison.
	MTU_PIN_LO_GOHI	Specifies initial output as low and output of the high level in response to matches in comparison.
	MTU_PIN_LO_TOGGLE	Specifies initial output as low and toggle output in response to matches in comparison.
	MTU_PIN_HI_GOLO	Specifies initial output as high and output of the low level in response to matches in comparison.
	MTU_PIN_HI_GOHI	Specifies initial output as high and output of the high level in response to matches in comparison.
	MTU_PIN_HI_TOGGLE	Specifies initial output as high and toggle output in response to matches in comparison.
Remarks	If _GOLO or _GOHI is specified in the parameter, the output level is retained after the MTU pin output is set to the low or high level in response to the first match in comparison. The output level will not be changed even after another match in comparison is detected. When _TOGGLE is specified, the output is toggled by switching the output level of the MTU pin every time a match in comparison is detected.	

5.16 R_MTU_Capture_Open Parameters

The parameters that are used for function R_MTU_Capture_Open (*pconfig) are listed below.

Table 5.10 R_MTU_Capture_Open Parameters

Parameter	Outline
clock_src.source	Sets the clock source for counting.
clock_src.clock_edge	Selects edges for the clock.
clock_div	Sets the frequency division ratio of an internal clock.
clear_src	Specifies the source for clearing the counter.
capture_X.actions (X = a, b, c, d)	Sets operation of a timer general register TGRX (X = a, b, c, d).
capture_X.capture_edge (X = a, b, c, d)	Sets the effective edge or edges for the input capture input signal for a timer general register TGRX (X = a, b, c, d).
capture_X.filter_enable (X = a, b, c, d)	Sets noise filtering of a signal from the corresponding MTU input pin MTIOCnX (X = A, B, C, D) (n = 0 to 4, 6, 7, 8).

5.16.1 clock_src.source

clock_src.source

Synopsis: Setting the clock source for counting

Header: r_mtu3_if.h

Description: This parameter sets the clock source for counting by an individual MTU channel.
When the external clock is selected, the I/O port settings for the target pin and in the multi-function pin controller (MPC) must be made in advance.

Parameters:

MTU_CLK_SRC_EXT_MTCLKA	Specifies an external clock signal (input on the MTCLKA pin).
MTU_CLK_SRC_EXT_MTCLKB	Specifies an external clock signal (input on the MTCLKB pin).
MTU_CLK_SRC_EXT_MTCLKC*1	Specifies an external clock signal (input on the MTCLKC pin).
MTU_CLK_SRC_EXT_MTCLKD*2	Specifies an external clock signal (input on the MTCLKD pin).
MTU_CLK_SRC_CASCADE*3	Specifies overflows or underflows of MTU2.TCNT.
MTU_CLK_SRC_INTERNAL	Specifies an internal clock signal (PCLKC).

Remarks:
Note 1. Only available for channels 0 and 2.
Note 2. Only available for channel 0.
Note 3. Only available for channel 1.

5.16.2 clock_src.clock_edge

clock_src.clock_edge		
Synopsis	Selecting edges for the clock	
Header	r_mtu3_if.h	
Description	This parameter selects the edge for counting of the input clock cycles by an individual MTU channel.	
Parameters	MTU_CLK_RISING_EDGE	Specifies counting of rising edges.
	MTU_CLK_FALLING_EDGE	Specifies counting of falling edges.
	MTU_CLK_ANY_EDGE	Specifies counting of rising and falling edges.
Remarks	When the frequency division ratio of the specified internal clock (clock_div) is PCLKC/1 or the clock source for counting is specified as overflows or underflows of MTU2.TCNT, the setting for the selection of edges has no effect and operation is always as if the initial value (rising edges) has remained.	

5.16.3 clock_div

clock_div		
Synopsis	Setting the frequency division ratio of an internal clock	
Header	r_mtu3_if.h	
Description	This parameter specifies the frequency division ratio of an internal clock on an individual MTU channel. The setting of this parameter has no effect when an external clock is selected by clock_src.clock_edge.	
Parameters	MTU_SRC_CLK_DIV_1	Specifies an internal clock signal (PCLKC/1).
	MTU_SRC_CLK_DIV_2	Specifies an internal clock signal (PCLKC/2).
	MTU_SRC_CLK_DIV_4	Specifies an internal clock signal (PCLKC/4).
	MTU_SRC_CLK_DIV_8	Specifies an internal clock signal (PCLKC/8).
	MTU_SRC_CLK_DIV_16	Specifies an internal clock signal (PCLKC/16).
	MTU_SRC_CLK_DIV_32	Specifies an internal clock signal (PCLKC/32).
	MTU_SRC_CLK_DIV_64	Specifies an internal clock signal (PCLKC/64).
	MTU_SRC_CLK_DIV_256	Specifies an internal clock signal (PCLKC/256).
	MTU_SRC_CLK_DIV_1024	Specifies an internal clock signal (PCLKC/1024).
Remarks	—	

5.16.4 clear_src

clear_src		
Synopsis	Selecting the source for clearing the counter	
Header	r_mtu3_if.h	
Description	This parameter specifies the source for clearing the counter of the TCNT on an individual MTU channel.	
Parameters	MTU_CLR_TIMER_A	Specifies input capture by TGRA.
	MTU_CLR_TIMER_B	Specifies input capture by TGRB.
	MTU_CLR_TIMER_C*1	Specifies input capture by TGRC.
	MTU_CLR_TIMER_D*1	Specifies input capture by TGRD.
	MTU_CLR_SYNC	Specifies clearing of the counters of other channels that are in synchronous clearing or operation.
	MTU_CLR_DISABLED	Disables clearing of the TCNT.
Remarks	Note 1. Only available for channels 0, 3, 4, 6, 7, and 8	

5.16.5 capture_X.actions (X = a, b, c, d)

capture_X.actions		
Synopsis	Setting operation of a timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies operation of the TGR on an individual MTU channel. When the MTIOCnm pin (n = 0 to 4, 6, 7, 8; m = A, B, C, D) is used for the input capture input, the I/O port settings for the target pin and in the multi-function pin controller (MPC) must be made in advance. Multiple operations can be set simultaneously by this parameter as below. Example) When my_capture_cfg is used as *pconfig: my_capture_cfg.capture_a.actions = (mtu_actions_t)(MTU_ACTION_CAPTURE | MTU_ACTION_REPEAT);	
Parameters	MTU_ACTION_CAPTURE*1	Specifies input capture operation for the TGR.
	MTU_ACTION_INTERRUPT*2	Enables input capture interrupts.
	MTU_ACTION_CALLBACK	Enables input capture interrupts and specifies the user-specified callback function to be executed in the interrupt service routine.
	MTU_ACTION_TRIGGER_ADC*3	Specifies activation by a trigger for the A/D converter in response to input capture by TGRA.
	MTU_ACTION_REPEAT*4	Specifies timer counter operation is to continue after the first input capture.
	MTU_ACTION_NONE	This parameter is specified when the TGR is not used. Specifying the parameter with other parameters is prohibited.
Remarks	Note 1. When input capture is used, MTU_ACTION_CAPTURE must always be specified. Note 2. The interrupt priority level must be specified in advance by r_mtu3_config.h. Note 3. Available for channels 0 to 4, 6, and 7. Note 4. When this parameter is not specified, timer count operation is stopped in the interrupt service routine. For this reason, timer count operation always continues if enabling of interrupts (MTU_ACTION_INTERRUPT or MTU_ACTION_CALLBACK) is not specified.	

5.16.6 capture_X.capture_edge (X = a, b, c, d)

capture_X.capture_edge		
Synopsis	Setting the effective edge or edges for the input capture input signal for a timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter sets the effective edge or edges for the input capture signal on an individual MTU channel.	
Parameters	MTU_CAP_RISING_EDGE	Specifies counting of rising edges.
	MTU_CAP_FALLING_EDGE	Specifies counting of falling edges.
	MTU_CAP_ANY_EDGE	Specifies counting of rising and falling edges.
Remarks	—	

5.16.7 capture_X.filter_enable (X = a, b, c, d)

capture_X.capture_edge		
Synopsis	Setting noise filtering of a signal from the corresponding MTU input pin MTIOCnX (X = A, B, C, D) (n = 0 to 4, 6, 7, 8)	
Header	r_mtu3_if.h	
Description	This parameter sets digital noise filtering of the input capture signal on an individual MTU channel.	
Parameters	true	Enables noise filtering.
	false	Disables noise filtering.
Remarks	The sampling clock cycle for the noise filter must be specified in advance by r_mtu3_config.h.	

5.17 R_MTU_PWM_Open Parameters

The parameters that are used for function R_MTU_PWM_Open (*pconfig) are listed below.

Table 5.11 R_MTU_PWM_Open Parameters

Parameter	Outline
clock_src.source	Sets the clock source for counting.
clock_src.clock_edge	Sets edges for the clock.
cycle_freq	Sets the PWM cycle (Hz).
clear_src	Specifies the source for clearing the counter.
pwm_mode	Setting the PWM mode
pwm_X.duty (X = a, b, c, d)	Sets the duty cycle for a timer general register TGRX (X = a, b, c, d).
pwm_X.actions (X = a, b, c, d)	Sets operation of a timer general register TGRX (X = a, b, c, d).
pwm_X.outputs (X = a, b, c, d)	Sets the pin level of PWM output for a timer general register TGRX (X = a, b, c, d).

5.17.1 clock_src.source

clock_src.source

Synopsis: Setting the counter clock

Header: r_mtu3_if.h

Description: This parameter sets the clock source for counting by an individual MTU channel.
When the external clock is selected, the I/O port settings for the target pin and in the multi-function pin controller (MPC) must be made in advance.
When the internal clock (PCLKC) is selected, a division ratio that is more suitable for the specified timer cycle (timer_X.actions.freq) is automatically selected.

Parameters:

MTU_CLK_SRC_EXT_MTCLKA	Specifies an external clock signal (input on the MTCLKA pin).
MTU_CLK_SRC_EXT_MTCLKB	Specifies an external clock signal (input on the MTCLKB pin).
MTU_CLK_SRC_EXT_MTCLKC*1	Specifies an external clock signal (input on the MTCLKC pin).
MTU_CLK_SRC_EXT_MTCLKD*2	Specifies an external clock signal (input on the MTCLKD pin).
MTU_CLK_SRC_INTERNAL	Specifies an internal clock (PCLKC).

Remarks: Note 1. Only available for channels 0 and 2.
Note 2. Only available for channel 0.

5.17.2 clock_src.clock_edge

clock_src.clock_edge		
Synopsis	Selecting edges for the clock	
Header	r_mtu3_if.h	
Description	This parameter selects the edge for counting of the input clock cycles by an individual MTU channel.	
Parameters	MTU_CLK_RISING_EDGE	Specifies counting of rising edges.
	MTU_CLK_FALLING_EDGE	Specifies counting of falling edges.
	MTU_CLK_ANY_EDGE	Specifies counting of rising and falling edges.
Remarks	When the PWM cycle specified for the counter clock is PCLKC/1 (cycle_freq), the setting for the selection of edges has no effect and operation is always as if the initial value (rising edges) has remained.	

5.17.3 cycle_freq

cycle_freq		
Synopsis	Setting the PWM cycle (Hz)	
Header	r_mtu3_if.h	
Description	This parameter specifies the PWM cycle (Hz) for an individual MTU channel. When the internal clock (PCLKC) is selected as the counter clock, an appropriate division ratio of PCLKC is automatically calculated. When an external clock is selected as the counter clock, the value of this parameter is simply set to the current value of the given TGR and is handled as the value for counting of the PWM cycle.	
Parameters	Value (in Hz units)	Specifies a cycle in Hz units. Specifies the TGR counter value (up to FFFFh) for an external clock.
Remarks	In PWM mode 1, TGRA and TGRC are the cycle setting registers. In PWM mode 2, TGR specified by clear_src is the cycle setting register.	

5.17.4 clear_src

clear_src		
Synopsis	Specifying the source for clearing the counter	
Header	r_mtu3_if.h	
Description	This parameter specifies the source for clearing the counter of the TCNT on an individual MTU channel.	
Parameters	MTU_CLR_TIMER_A	Specifies a compare match by TGRA.
	MTU_CLR_TIMER_B	Specifies a compare match by TGRB.
	MTU_CLR_TIMER_C*1	Specifies a compare match by TGRC.
	MTU_CLR_TIMER_D*1	Specifies a compare match by TGRD.
	MTU_CLR_SYNC	Specifies clearing of the counters of other channels that are in synchronous clearing or operation.
	MTU_CLR_DISABLED	Disables clearing of the TCNT.
Remarks	Note 1. Only available for channels 0, 3, 4, 6, 7, and 8.	

5.17.5 pwm_mode

pwm_mode		
Synopsis	Setting the PWM mode	
Header	r_mtu3_if.h	
Description	This parameter sets the PWM mode on an individual MTU channel.	
Parameters	MTU_PWM_MODE_1*1	Sets PWM mode 1.
	MTU_PWM_MODE_2*2	Sets PWM mode 2.
Remarks	Note 1. Only available for channels 0 to 4, 6, and 7. Note 2. Only available for channels 0 to 2.	

5.17.6 pwm_X.duty (X = a, b, c, d)

pwm_X.duty		
Synopsis	Setting the duty cycle for a timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter sets the duty cycle (in 0.1% units) of the PWM waveforms from an individual MTU channel. The duty cycle can be set in the range from 0% to 100%.	
Parameters	Value (in 0.1% units)	Specifies the duty cycle in 0.1% units (0 to 1000).
Remarks	In PWM mode 1, TGRB and TGRD are the duty setting registers. In PWM mode 2, registers other than TGR specified by clear_src are the duty setting registers.	

5.17.7 pwm_X.actions (X = a, b, c, d)

pwm_X.actions		
Synopsis	Setting operation of a timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies operation of the TGR on an individual MTU channel. When the MTU pin is specified as an output pin, the I/O port settings for the target pin and in the multi-function pin controller (MPC) must be made in advance. Multiple operations can be set simultaneously by this parameter as below. Example) When simple_pwm _cfg is used as *pconfig: simple_pwm_cfg.pwm_a.actions = (mtu_actions_t)(MTU_ACTION_OUTPUT | MTU_ACTION_INTERRUPT);	
Parameters	MTU_ACTION_OUTPUT	Specifies the MTU pin as an output pin.
	MTU_ACTION_INTERRUPT*1	Enables compare-match interrupts.
	MTU_ACTION_CALLBACK	Enables compare-match interrupts and specifies the user-specified callback function to be executed in the interrupt service routine.
	MTU_ACTION_TRIGGER_ADC*2	Specifies activation by a trigger for the A/D converter in response to matches in comparison with TGRA.
	MTU_ACTION_REPEAT*3	Specifies PWM output is to continue after the first PWM cycle has elapsed.
	MTU_ACTION_NONE	This parameter is specified when the TGR is not used. Specifying the parameter with other parameters is prohibited.
Remarks	Note 1. The interrupt priority level must be specified in advance by r_mtu3_config.h. Note 2. Only available for channels 0 to 4, 6, and 7. Note 3. When this parameter is not specified, timer counter operation is stopped in the interrupt processing routine. For this reason, timer counter operation always continues when enabling of interrupts (MTU_ACTION_INTERRUPT or MTU_ACTION_CALLBACK) is not specified.	

5.17.8 pwm_X.outputs (X = a, b, c, d)

pwm_X.outputs		
Synopsis	Setting the pin level of PWM output for a timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter sets the initial output level of the PWM output pin on an individual MTU channel and the output level transitions at the times of matches in comparison.	
Parameters	MTU_PIN_NO_OUTPUT	Disables output.
	MTU_PIN_LO_GOLO	Specifies initial output as low and output of the low level in response to matches in comparison.
	MTU_PIN_LO_GOHI	Specifies initial output as low and output of the high level in response to matches in comparison.
	MTU_PIN_LO_TOGGLE	Specifies initial output as low and toggle output in response to matches in comparison.
	MTU_PIN_HI_GOLO	Specifies initial output as high and output of the low level in response to matches in comparison.
	MTU_PIN_HI_GOHI	Specifies initial output as high and output of the high level in response to matches in comparison.
	MTU_PIN_HI_TOGGLE	Specifies initial output as high and toggle output in response to matches in comparison.
Remarks	In PWM mode 1, PWM output is from the MTIOCnA pin when TGRA and TGRB are in use. Furthermore, PWM output is from the MTIOCnC pin when TGRC and TGRD are in use. In PWM mode 2, PWM output is in accord with the single TGR set in the cycle register and the TGRs set in other duty-cycle registers. Also, if _TOGGLE is specified, the level output on the MTU pin is switched (toggled) whenever a compare-match is generated, and PWM modes are basically not used. It is provided for use in evaluation and so on.	

5.18 R_MTU_Control Parameters

The parameters that are used for function R_MTU_Control (*pconfig) are listed below.

pcmd_data must be written in the format conforming to the command specified by cmd.

5.18.1 When cmd = MTU_CMD_START

The first address of a variable of the mtu_group_t structure type is specified as the parameter (*pcmd_data).

pcmd_data		
Synopsis	Specifying multiple channel numbers	
Header	r_mtu3_if.h	

Description	This parameter specifies the simultaneous operation of multiple MTU channels.
	Example) When my_group is used as pcmd_data:
	mtu_group_t my_group;
	my_group = (mtu_group_t)(MTU_GRP_CH0 | MTU_GRP_CH3 | MTU_GRP_CH4);
	result = R_MTU_Control(MTU_CHANNEL_0, MTU_CMD_START, &my_group);	

Parameters	MTU_GRP_CH0	Channel number 0
	MTU_GRP_CH1	Channel number 1
	MTU_GRP_CH2	Channel number 2
	MTU_GRP_CH3	Channel number 3
	MTU_GRP_CH4	Channel number 4
	MTU_GRP_CH6	Channel number 6
	MTU_GRP_CH7	Channel number 7
	MTU_GRP_CH8	Channel number 8
Remarks	Note 1. When a single channel is used, set only the first channel argument of function R_MTU_Control and set the third pcmd_data argument null. When multiple channels are specified by using the third pcmd_data argument, the value of the first argument becomes invalid.	

5.18.2 When cmd = MTU_CMD_STOP

The first address of a variable of the mtu_group_t structure type is specified as the parameter (*pcmd_data).

pcmd_data		
Synopsis	Specifying multiple channel numbers	
Header	r_mtu3_if.h	

Description	This parameter specifies the simultaneous operation of multiple MTU channels.
	Example) When my_group is used as pcmd_data:
	mtu_group_t my_group;
	my_group = (mtu_group_t)(MTU_GRP_CH0 | MTU_GRP_CH3 | MTU_GRP_CH4);
	result = R_MTU_Control(MTU_CHANNEL_0, MTU_CMD_STOP, &my_group);	

Parameters	MTU_GRP_CH0	Channel number 0
	MTU_GRP_CH1	Channel number 1
	MTU_GRP_CH2	Channel number 2
	MTU_GRP_CH3	Channel number 3
	MTU_GRP_CH4	Channel number 4
	MTU_GRP_CH6	Channel number 6
	MTU_GRP_CH7	Channel number 7
	MTU_GRP_CH8	Channel number 8
Remarks	When a single channel is used, set only the first channel argument of function R_MTU_Control and set the third pcmd_data argument null. When multiple channels are specified by using the third pcmd_data argument, the value of the first argument becomes invalid.	

5.18.3 When cmd = MTU_CMD_SAFE_STOP

This command is only effective for the first argument (specifying the channel) of function R_MTU_Control.

In this case, the value of the parameter (*pcmd_data) should be null rather than an actual address.

5.18.4 When cmd = MTU_CMD_RESTART

This command is only effective for the first argument (specifying the channel) of function R_MTU_Control.

In this case, the value of the parameter (*pcmd_data) should be null rather than an actual address.

5.18.5 When cmd = MTU_CMD_SYNCHRONIZE

The first address of a variable of the mtu_group_t structure type is specified as the parameter (*pcmd_data).

pcmd_data		
Synopsis	Specifying multiple channel numbers	
Header	r_mtu3_if.h	

Description	This parameter specifies the simultaneous operation of multiple MTU channels.
	Example) When my_group is used as pcmd_data:
	mtu_group_t my_group;
	my_group = (mtu_group_t)(MTU_GRP_CH0 | MTU_GRP_CH3 | MTU_GRP_CH4);
	result = R_MTU_Control(MTU_CHANNEL_0, MTU_CMD_STOP, &my_group);	

Parameters	MTU_GRP_CH0	Channel number 0
	MTU_GRP_CH1	Channel number 1
	MTU_GRP_CH2	Channel number 2
	MTU_GRP_CH3	Channel number 3
	MTU_GRP_CH4	Channel number 4
	MTU_GRP_CH6	Channel number 6
	MTU_GRP_CH7	Channel number 7
Remarks	When multiple channels are specified by using the third pcmd_data argument, the value of the first argument becomes invalid.	

5.18.6 When cmd = MTU_CMD_GET_STATUS (in timer mode)

In timer mode, the first address of a variable of the mtu_timer_status_t structure type is specified as the parameter (*pcmd_data). When a command is executed, the following parameters are acquired and the values are returned to the specified structure variable.

Table 5.12 Parameters when cmd = MTU_CMD_GET_STATUS (in timer mode)

Parameter	Outline
timer_count	Counter value of the timer
timer_running	Counting direction of the timer

5.18.7 When cmd = MTU_CMD_GET_STATUS (in input capture mode)

In input capture mode, the first address of a variable of the mtu_capture_status_t structure type is specified as the parameter (*pcmd_data). When a command is executed, the following parameters are acquired and the values are returned to the specified structure variable.

Table 5.13 Parameters when cmd = MTU_CMD_GET_STATUS (in input capture mode)

Parameter	Outline
capt_X_count (X= a, b, c, d)	Input capture value in TGRX (X = a, b, c, d)
timer_count	Counter value of the timer

5.18.8 When cmd = MTU_CMD_SET_CAPT_EDGE

When this command is specified, the first address of a variable of the mtu_capture_set_edge_t structure type is specified as the parameter (*pcmd_data).

The settings of the input source and the edge for input capture are made in accordance with the following parameters specified in the structure.

Table 5.14 Parameters when cmd = MTU_CMD_SET_CAPT_EDGE

Parameter	Outline
capture_src	Settings for the input source for input capture MTU_CAP_SRC_A: MTIOCnA pin MTU_CAP_SRC_B: MTIOCnB pin MTU_CAP_SRC_C: MTIOCnC pin MTU_CAP_SRC_D: MTIOCnD pin
capture_edge	Settings for the edge for input capture MTU_CAP_RISING_EDGE: Rising edges MTU_CAP_FALLING_EDGE: Falling edges MTU_CAP_ANY_EDGE: Rising and falling edges

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 MTU3a Sample Program (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. 13, 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