RZ/T1 Group

Multi-function Timer Pulse Unit (MTU3a)

May. 14, 2021
R01AN2653EJ0150
May. 14, 2021

Introduction

This application note describes a sample program that outputs three-phase (positive and negative, in total of 6 phases) dead-time PWM waveform by using complementary PWM mode of the multi-function timer pulse unit (MTU3a), MTU3 and MTU4.

The major features of the program are listed below:

- Output of complementary PWM waveform with the cycle of the carrier (100 µs) by using MTU3 and MTU4 and dead time (2µs)
- Switching of PWM duty ratio by every press-down of SW2 to 25%, 50%, and 75% (repeatedly)

Target Devices for Operation Checking

RZ/T1 Group

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

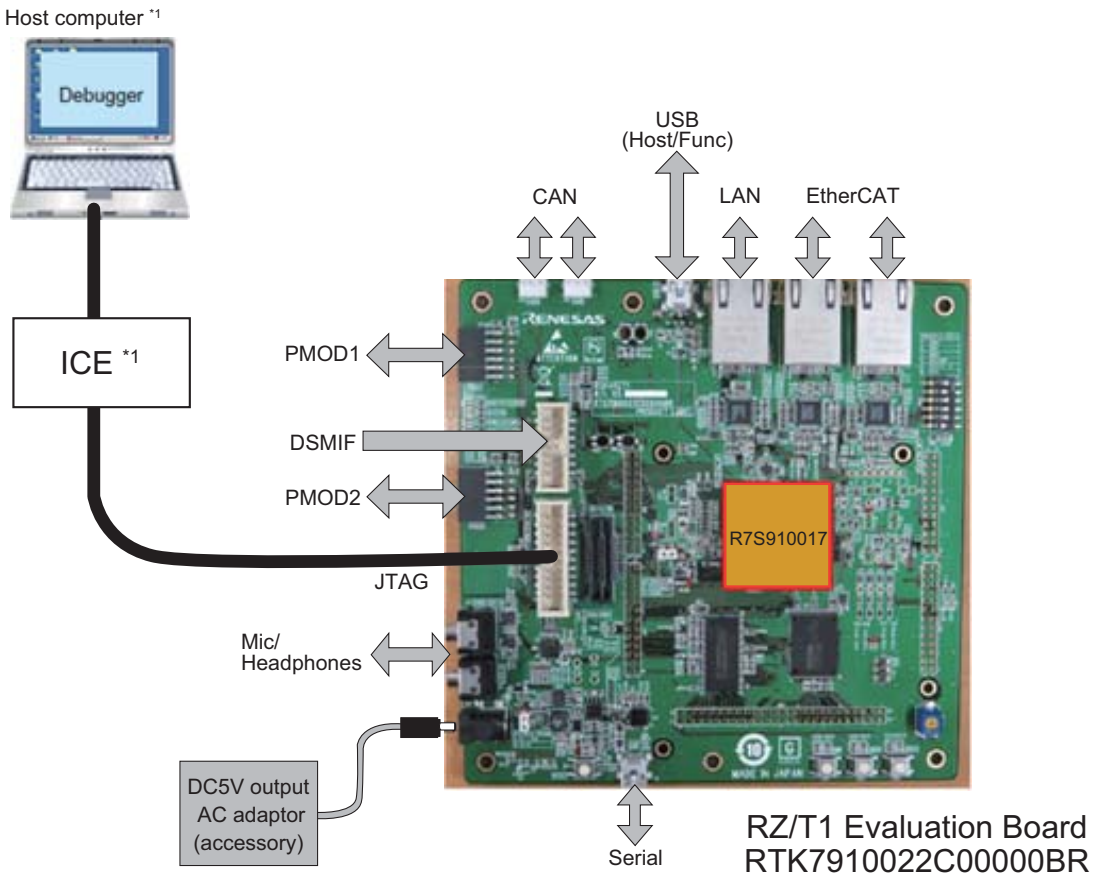
Table of Contents

1. Specifications

Table 1.1 lists the peripheral functions to be used and their applications and Figure 1.1 shows the operating environment.

Table 1.1 Peripheral Functions and Applications

Peripheral Function	Application
Clock Pulse Generator (CPG)	The CPG produces the CPU clock and low-speed on-chip oscillator clock signals
Interrupt Control Unit A (ICUA)	The ICUA is used for the external interrupt input pin (IRQ5)
Multi-function timer pulse unit (MTU3a)	The MTU3a is used for complementary PWM output of the MTU3 and MTU4
Error Control Module (ECM)	The ECM is used to initialize the ERROROUT# pins



Note 1. Indicates the device that the user needs to prepare.

Figure 1.1 Operating Environment

2. Operating Environment

The sample program of this application is for the environment below.

Table 2.1 Operating Environment

Item	Description
Microcomputer	RZ/T1 Group
Operating frequency	CPUCLK = 450 MHz
Operating voltage	3.3 V
Integrated development environment	Manufactured by IAR Systems Embedded Workbench® for Arm Version 8.20.2 Manufactured by Arm DS-5™ 5.26.2 Manufactured by RENESAS e2studio 6.1.0
Operating modes	SPI boot mode 16-bit bus boot mode
Board	RZ/T1 Evaluation board (RTK7910022C00000BR)
Devices (functions to be used on the board)	• NOR flash memory (connected to CS0/CS1 space) Manufacturer: Macronix International Co. Ltd. Model: MX29GL512FLT2I-10Q • SDRAM (connected to CS2/CS3 space) Manufacturer: Integrated Silicon Solution Inc. Model: IS42S16320D-7TL • Serial flash memory Manufacturer: Macronix International Co. Ltd. Model: MX25L51245G

3. Related Application Note

Refer to the relevant application note given below.

- RZ/T1 Group Initial Settings

4. Peripheral Functions

For the basics of the clock pulse generator (CPG), multi-function timer pulse unit (MTU3a), interrupt control unit A (ICUA), and error control module (ECM), refer to the RZ/T1 Group User's Manual: Hardware.

5. Hardware

5.1 Example of Hardware Configuration

Figure 5.1 shows an example of the hardware configuration.

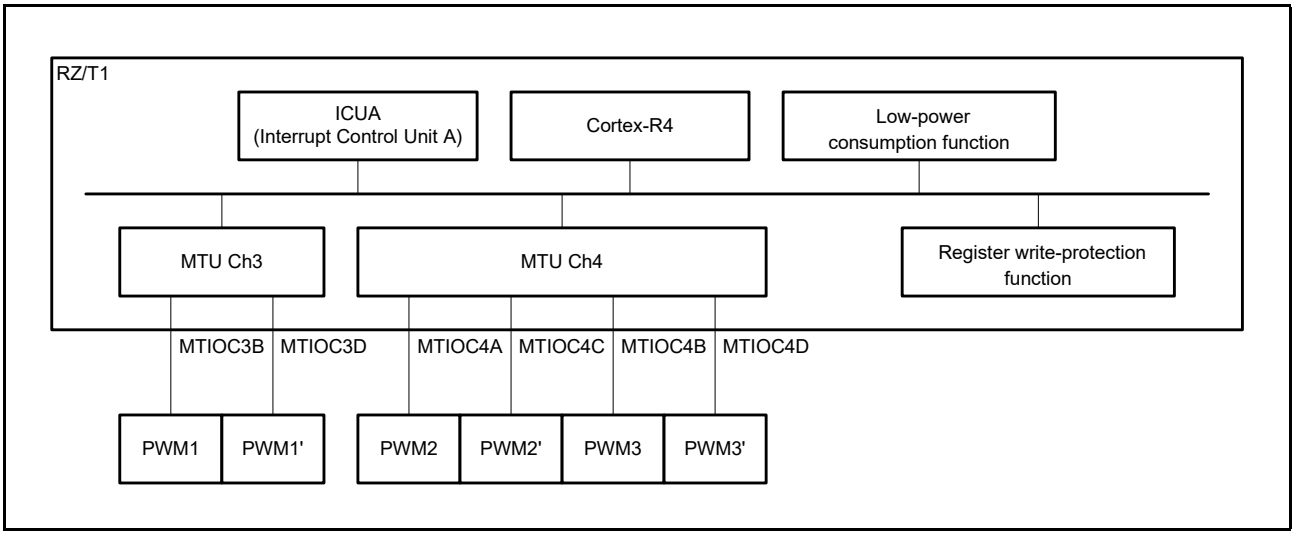


Figure 5.1 Example of Hardware Configuration

5.2 Pins

Table 5.1 shows pins to be used and their functions.

Table 5.1 Pins and Pin Functions

Pin Name	I/O	Function
MD0	Input	Selection of operating modes MD0 = L, MD1 = L, MD2 = L (SPI boot mode) MD0 = L, MD1 = H, MD2 = L (16-bit bus boot mode)
MD1	Input	
MD2	Input	
IRQ5	Input	SW2 (IRQ pin interrupt)
MTIOC3B	Output	PWM output 1
MTIOC3D	Output	PWM output 1' (inverted-phase waveform output of PWM output 1)
MTIOC4A	Output	PWM output 2
MTIOC4C	Output	PWM output 2' (inverted-phase waveform output of PWM output 2)
MTIOC4B	Output	PWM output 3
MTIOC4D	Output	PWM output 3' (inverted-phase waveform output of PWM output 3)

6. Software

6.1 Operation Overview

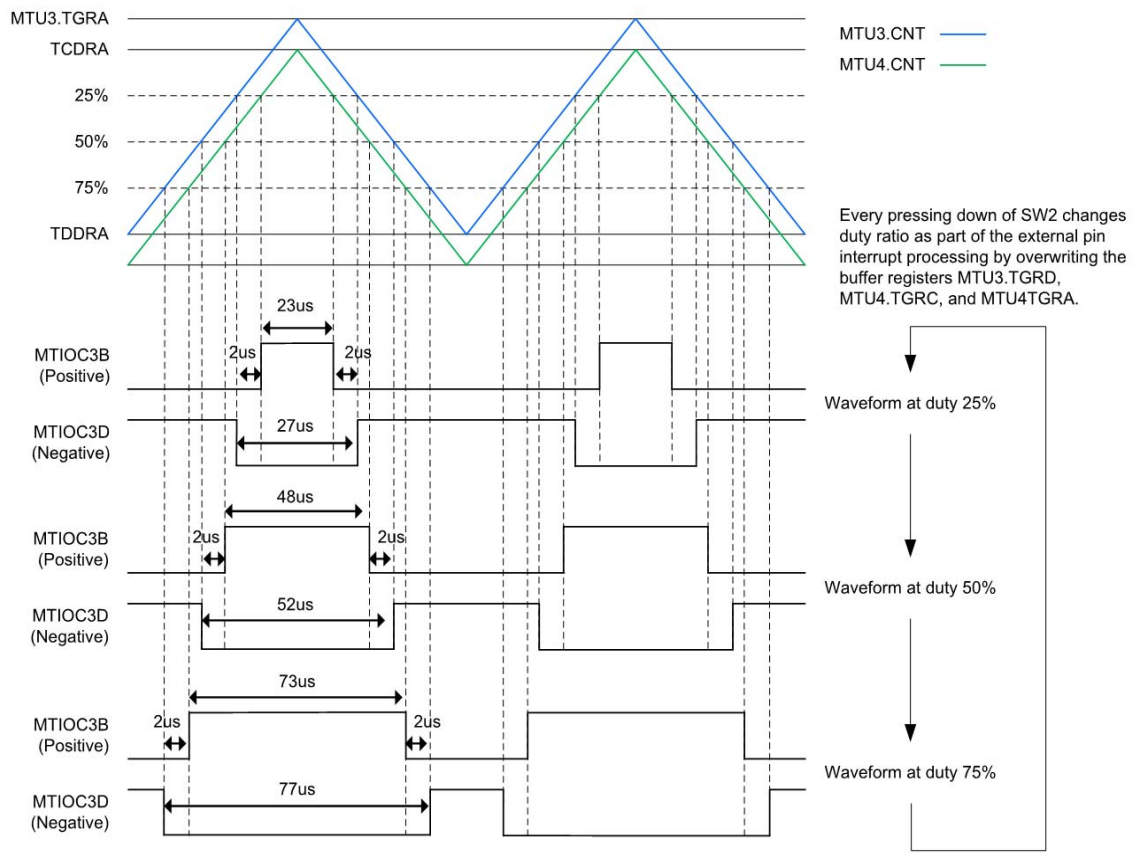
The sample program makes the initial settings of the multi-function timer pulse unit (MTU3a), and then outputs complementary PWM signals.

Pressing SW2 down generates an external pin interrupt 5 and switches duty ratio by writing the buffer register (25% →50% → 75% → 25%→(repeatedly)).

Table 6.1 shows the functional overview of the sample program and Figure 6.1 shows the timing diagram.

Table 6.1 Functional Overview

Function	Description
Channel	Channel 3 (MTU3), channel 4 (MTU4)
PWM output	Three phases for positive and negative (in total of 6), cycle of carrier (100 µs), dead time (2 µs)
Operating mode	Complementary PWM mode 1 (transferred at crest)
Clock	PCLKC/4 (= 37.5 MHz), rising edge
Duty ratio	25%, 50%, and 75% common to all phases



Note: Note. The following MTU output pins have the same output.
MTIOC4A (normal) and MTIOC4C (inverted)
MTIOC4B (normal) and MTIOC4D (inverted)

Figure 6.1 Timing Diagram

6.1.1 Project Settings

For the settings of the project to be used on the EWARM for development environment, refer to the Application Note: RZ/T1 Group Initial Settings.

6.2 Memory Map

For the address space of the RZ/T1 Group and a memory map of the RZ/T1 evaluation board, refer to the Application Note: RZ/T1 Group Initial Settings.

6.2.1 Assignment to Sections of Sample Program

Refer to the Application Note: RZ/T1 Group Initial Settings for the sections to be used in the sample 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).

6.2.2 MPU Settings

Refer to the Application Note: RZ/T1 Group Initial Settings for MPU settings.

6.2.3 Exception Processing Vector Table

Refer to the Application Note: RZ/T1 Group Initial Settings for the vector table for exception processing.

6.3 Interrupts

Table 6.2 shows interrupts to be used in the sample program.

Table 6.2 Interrupts for Sample Program

Interrupts (Source ID)	Priority	Description
IRQ9 interrupt (IRQ pin interrupt 5)	15	Pressing down SW2 switches duty ratio to 25%, 50%, and 75% by writing the buffer register, MTU3.TGRD, MTU4.TGRC, and MTU4.TGRD

6.4 Fixed-Width Integer Types

Table 6.3 shows fixed width integers to be used in the sample program.

Table 6.3 Fixed-Width Integers for 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)

6.5 Constants/Error Codes

Table 6.4 shows constants to be used in the sample program.

Table 6.4 Constants for Sample Program

Constant Name	Setting Value	Description
MTU3_CFG_PARAM_CHECKING_ENABLE	(1)	Displays enabling (1)/disabling (0) of parameter checking via the API function of the MTU
MTU_COUNT_STOP	(0)	Stops the operation of the MTU counter
MTU_CMPLMT_PWM_START	(0xC0)	Starts complementary PWM
MTU_C_SET_CYCLE	(0xEA6)	Specifies the cycle of the carrier
MTU_C_CYCLE	(0x753)	Specifies 1/2 of the cycle of the carrier
MTU_DEAD_TIME	(0x4B)	Specifies dead time
MTU_DUTY_25	((uint16_t) MTU_C_CYCLE * 3/4 + MTU_DEAD_TIME * 1/2) *1	Specifies duty ratio 25%
MTU_DUTY_50	((uint16_t) MTU_C_CYCLE * 2/4 + MTU_DEAD_TIME * 1/2) *1	Specifies duty ratio 50%
MTU_DUTY_75	((uint16_t) MTU_C_CYCLE * 1/4 + MTU_DEAD_TIME * 1/2) *1	Specifies duty ratio 75%
MTU_ENABLE_OUTPUT	(1)	Enables MTU output
MTU_DISABLE_OUTPUT	(0)	Disables MTU output
DUTY_CNT_MAX	(3)	The maximum value for switching duty ratio

Note 1: The setting value of duty ratio n% is MTU_DUTY_n = MTU_C_CYCLE * (1-n/100) + MTU_DEAD_TIME * 1/2

6.6 Structures/Unions/Emulate Types

Figure 6.2 to Figure 6.14 show the structures, unions, and enumerated types to be used in the sample program.

```
/* 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 channnel 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 6.2 Structures/Unions/Enumerated Types for Sample Program (1)

```
/* The possible return codes from the API functions. */
typedef enum mtu_pwm_err_e
{
    MTU_SUCCESS = 0,
    MTU_ERR_BAD_CHAN,                   // Invalid channel number.
    MTU_ERR_CH_NOT_OPENED,              // Channel not yet opened.
    MTU_ERR_CH_NOT_CLOSED,              // Channel still open from previous open.
    MTU_ERR_UNKNOWN_CMD,                // Control command is not recognized.
    MTU_ERR_INVALID_ARG,                // Argument is not valid for parameter.
    MTU_ERR_ARG_RANGE,                  // Argument is out of range for parameter.
    MTU_ERR_NULL_PTR,                   // Received null pointer; missing required argument.
    MTU_ERR_LOCK,                       // The lock procedure failed.
    MTU_ERR_UNDEF                       // Undefined/unknown error
} mtu_err_t;

/* 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 6.3 Structures/Unions/Enumerated Types for Sample Program (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_4,      // PCLK/4
    MTU_SRC_CLK_DIV_16,     // PCLK/16
    MTU_SRC_CLK_DIV_64,     // PCLK/64
    MTU_SRC_CLK_DIV_256,    // PCLK/256
    MTU_SRC_CLK_DIV_1024,   // PCLK/1024
    MTU_SRC_CLK_DIV_2,      // PCLK/2
    MTU_SRC_CLK_DIV_8,      // PCLK/8
    MTU_SRC_CLK_DIV_32      // PCLK/32
} 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 6.4 Structures/Unions/Enumerated Types for Sample Program (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 channnel not support */
    MTU_GRP_CH6 = 0x4000,
    MTU_GRP_CH7 = 0x8000,
    MTU_GRP_CH8 = 0x0008
} mtu_group_t;
```

Figure 6.5 Structures/Unions/Enumerated Types for Sample Program (4)

```
typedef struct mtu_timer_status_s
{
      uint32_t timer_count;              // The current channel counter value.
      bool     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.
      uint8_t  capture_flags;             // 1 if a capture event occurred, 0 if still waiting.
} mtu_capture_status_t;

typedef struct mtu_pwm_status_s
{
      bool    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 6.6 Structures/Unions/Enumerated Types for Sample Program (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_NUM_TIMERS
} 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 6.7 Structures/Unions/Enumerated Types for Sample Program (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             filter_enable;        // Noise filter on or off.
} mtu_capture_settings_t;
```

Figure 6.8 Structures/Unions/Enumerated Types for Sample Program (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 6.9 Structures/Unions/Enumerated Types for Sample Program (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 6.10 Structures/Unions/Enumerated Types for Sample Program (9)

```
/************ Type defines used with the R_MTU_PWM_Complement_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,  // Complementary PWM mode 1
    MTU_CMPL_PWM_MODE_2 = 0x0E,  // Complementary PWM mode 2
    MTU_CMPL_PWM_MODE_3 = 0x0F,  // Complementary 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 6.11 Structures/Unions/Enumerated Types for Sample Program (10)

```
typedef enum
{
    MTU_CMPL_PWM_D_BF_OFF = 0,  // OFF
    MTU_CMPL_PWM_D_BF_ON_SYNC,   // SYNC
    MTU_CMPL_PWM_D_BF_ON_ASYNC,  // ASYNC
}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 6.12 Structures/Unions/Enumerated Types for Sample Program (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 6.13 Structures/Unions/Enumerated Types for Sample Program (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;             // Complementary 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 6.14 Structures/Unions/Enumerated Types for Sample Program (13)

6.7 Global Variables

Table 6.5 shows global variables.

Table 6.5 Global Variables

Model	Variable	Description	Function
uint16_t	g_duty_rate[DUTY_CNT_MAX]	Data of duty ratio	R_IRQ9_isr
uint8_t	g_duty_cnt	Switching counter of duty ratio	R_IRQ9_isr

6.8 Functions

Table 6.6 lists the functions to be used.

Table 6.6 Functions

Function	Page Number
main	26
init_mtu3	26
R_MTU_PWM_Complement_Open	27
R_MTU_PWM_Complement_Close	27
R_MTU_PWM_Complement_Control	28
R_IRQ9_isr	28
R_MTU_Timer_Open	29
R_MTU_Capture_Open	30
R_MTU_PWM_Open	31
R_MTU_Close	32
R_MTU_Control	33

6.9 Specifications of Functions

6.9.1 main

main	
Synopsis	Main processing
Declaration	int main (void)
Description	This function make settings of complementary PWM mode 1 and outputs three-phase PWM by using the MTU3 and MTU4.
Arguments	None
Return value	None
Supplement	None

6.9.2 init_mtu3

init_mtu3	
Synopsis	Initializing MTU
Declaration	void init_mtu3 (void)
Description	This function initializes the MTU3 and MTU4.
Arguments	None
Return value	None
Supplement	None

6.9.3 R_MTU_PWM_Complement_Open

R_MTU_PWM_Complement_Open		
Synopsis	Setting Complementary PWM mode	
Header	r_mtu3_if.h	
Declaration	mtu_err_t R_MTU_PWM_Complement_Open(mtu_cmpl_pwm_channel_t channel, mtu_cmpl_pwm_chnl_settings_t * pconfig)	
Description	This function specifies complementary PWM output for each specified MTU channel.	
Arguments	mtu_cmpl_pwm_channel_t channel	Specifies complement PWM channels MTU_CHANNEL_3_4 MTU_CHANNEL_6_7
	mtu_cmpl_pwm_chnl_settings_t * pconfig	The configuration setting pointer to output complementary PWM. For details, see Section 6.11, R_MTU_PWM_Complement_Open Parameters.
Return value	The result of execution of the complementary PWM open function is returned. MTU_SUCCESS: Normal termination MTU_ERR_BAD_CHAN: Specified channel invalid MTU_ERR_CH_NOT_CLOSED: Channel operation in progress MTU_ERR_NULL_PTR: pconfig pointer null MTU_ERR_INVALID_ARG: Invalid argument value	
Supplement	Setting MTU3_CFG_PARAM_CHECKING_ENABLE that is defied by r_mtu3_config.h to 1 enables checking of the parameters of the argument. Relevant functions: R_MTU_PWM_Complement_Close(), R_MTU_PWM_Complement_Control()	

6.9.4 R_MTU_PWM_Complement_Close

R_MTU_PWM_Complement_Close		
Synopsis	Complementary PWM end processing	
Header	r_mtu3_if.h	
Declaration	mtu_err_t R_MTU_PWM_Complement_Close(mtu_cmpl_pwm_channel_t channel)	
Description	This functions ends complementary PWM settings for each specified MTU channel.	
Arguments	mtu_cmpl_pwm_channel_t channel	Specifies complementary PWM channels MTU_CHANNEL_3_4 MTU_CHANNEL_6_7
Return value	The result of execution of the complementary PWM close function is returned. MTU_SUCCESS: Normal termination MTU_ERR_CH_NOT_OPEN: Channel operation stopped MTU_ERR_BAD_CHAN: Specified channel invalid	
Supplement	Setting MTU3_CFG_PARAM_CHECKING_ENABLE that is defined by r_mtu3_config.h to 1 enables checking of the parameters of the arguments.	

6.9.5 R_MTU_PWM_Complement_Control

R_MTU_PWM_Complement_Control		
Synopsis	Controlling complementary PWM	
Header	r_mtu3_if.h	
Declaration	mtu_err_t R_MTU_PWM_Complement_Control(mtu_cmpl_pwm_channel_t channel, mtu_cmd_t cmd, void * pcmd_data)	
Description	This function acquires information on the starting and stopping of complementary PWM during PWM operation.	
Arguments	mtu_cmpl_pwm_channel_t channel	Specifies complementary PWM channels MTU_CHANNEL_3_4 MTU_CHANNEL_6_7
	mtu_cmd_t cmd	Specifies a command to be controlled MTU_CMD_START MTU_CMD_STOP MTU_CMD_GET_STATUS
	void * pcmd_data	The pointer to the storage position of the acquired information. See Section 6.12, R_MTU_PWM_Complement_Control Parameters.
Return value	The result of execution of the complementary PWM control function is returned. MTU_SUCCESS: Normal termination MTU_ERR_BAD_CHAN: Specified channel invalid MTU_ERR_CH_NOT_OPEN: Channel operation stopped MTU_ERR_UNKNOWN_CMD: Invalid command MTU_ERR_NULL_PTR: pcmd_data pointer null	
Supplement	Setting MTU3_CFG_PARAM_CHECKING_ENABLE that is defined by r_mtu3_config.h to 1 enables checking of the parameters of the arguments.	

6.9.6 R_IRQ9_isr

R_IRQ9_isr	
Synopsis	IRQ9 interrupt (IRQ pin interrupt 5) processing
Declaration	void R_IRQ9_isr(void)
Description	This function overwrites the buffer registers MTU3.TGRD, MTU4.TGRC, and MTU4.TGRD to switch duty ratio to 25%, 50%, or 75%.
Arguments	None
Return value	None
Supplement	None

6.9.7 R_MTU_Timer_Open

R_MTU_Timer_Open		
Synopsis	Setting MTU compare/match	
Header	r_mtu3_if.h	
Declaration	mtu_err_t R_MTU_Timer_Open(mtu_channel_t channel, mtu_timer_chnl_settings_t *pconfig, void (*pcallback)(void *pdata))	
Description	This function makes settings of compare match timing operations for each specified MTU channel.	
Arguments	mtu_channel_t channel	Specifies MTU channel for compare match MTU_CHANNEL_0 MTU_CHANNEL_1 MTU_CHANNEL_2 MTU_CHANNEL_3 MTU_CHANNEL_4 MTU_CHANNEL_6 MTU_CHANNEL_7 MTU_CHANNEL_8
	mtu_timer_chnl_settings_t *pconfig	The pointer for configuration for compare match. For details, see Section 6.13, R_MTU_Timer_Open Parameters.
	void (*pcallback)(void *pdata)	Specifies the pointer to the callback function.
Return value	The result of execution of the MTU compare/match function is returned. MTU_SUCCESS: Normal termination MTU_TIMERS_ERR_BAD_CHAN: Specified channel invalid MTU_TIMERS_ERR_CH_NOT_CLOSED: Channel operation in progress MTU_TIMERS_ERR_NULL_PTR: pconfig pointer null MTU_ERR_ARG_RANGE: Range specified by argument invalid MTU_TIMERS_ERR_INVALID_ARG: Invalid argument value MTU_TIMERS_ERR_LOCK:	
Supplement	Setting MTU3_CFG_PARAM_CHECKING_ENABLE that is defined by r_mtu3_config.h to 1 enables checking of the parameters of the arguments. Relevant functions: R_MTU_Close(), R_MTU_Control()	

6.9.8 R_MTU_Capture_Open

R_MTU_Capture_Open		
Synopsis	Setting MTU capture	
Header	r_mtu3_if.h	
Declaration	mtu_err_t R_MTU_Capture_Open(mtu_channel_t channel, mtu_capture_chnl_settings_t *pconfig, void (*pcallback)(void *pdata))	
Description	This function makes settings of input capture for each specified MTU channel.	
Arguments	mtu_channel_t channel	Specifies MTU channel to be captured. MTU_CHANNEL_0 MTU_CHANNEL_1 MTU_CHANNEL_2 MTU_CHANNEL_3 MTU_CHANNEL_4 MTU_CHANNEL_6 MTU_CHANNEL_7 MTU_CHANNEL_8
	mtu_capture_chnl_settings_t *pconfig	The pointer of configuration for capturing For details, see Section 6.14, R_MTU_Capture_Open Parameters
	void (*pcallback)(void *pdata)	Specifies the pointer to the callback function.
Return value	The result of execution of the MTU capture open function is returned. MTU_SUCCESS: Normal termination MTU_TIMERS_ERR_BAD_CHAN: Specified channel invalid MTU_TIMERS_ERR_CH_NOT_CLOSED: Channel operation in progress MTU_TIMERS_ERR_NULL_PTR: pconfig pointer null MTU_TIMERS_ERR_INVALID_ARG: Invalid argument value MTU_TIMERS_ERR_LOCK:	
Supplement	Setting MTU3_CFG_PARAM_CHECKING_ENABLE that is defined by r_mtu3_config.h to 1 enables checking of the parameters of the arguments. Relevant functions: R_MTU_Close(), R_MTU_Control()	

6.9.9 R_MTU_PWM_Open

R_MTU_PWM_Open		
Synopsis	Setting MTU PWM	
Header	r_mtu3_if.h	
Declaration	mtu_err_t R_MTU_PWM_Open (mtu_channel_t channel, mtu_pwm_chnl_settings_t *pconfig, void (*pcallback)(void *pdata))	
Description	This function makes setting for basic PWM operations on each specified MTU channel.	
Arguments	mtu_channel_t channel	Specifies a port for PWM output. MTU_CHANNEL_0*1 MTU_CHANNEL_1*1 MTU_CHANNEL_2*1 MTU_CHANNEL_3 MTU_CHANNEL_4 MTU_CHANNEL6 MTU_CHANNEL7 Note 1. When PWM mode 2 is selected, only channel 0, 1, or 2 can be set.
	mtu_pwm_chnl_settings_t * pconfig	The pointer of configuration for basic PWM output. For details, see Section 6.15, R_MTU_PWM_Open Parameters.
	void (*pcallback)(void *pdata)	Specifies the pointer to the callback function.
Return value	The result of execution of the MTU open function is returned. MTU_SUCCESS: Normal termination MTU_ERR_BAD_CHAN: Specified channel invalid MTU_ERR_CH_NOT_CLOSED: Channel operation in progress MTU_ERR_NULL_PTR: pconfig pointer null MTU_ERR_ARG_RANGE: Range specified by argument invalid MTU_ERR_INVALID_ARG: Invalid argument value MTU_ERR_LOCK:	
Supplement	Setting MTU3_CFG_PARAM_CHECKING_ENABLE that is defined by r_mtu3_config.h to 1 enables checking of the parameters of the arguments. Relevant functions: R_MTU_Close(), R_MTU_Control()	

6.9.10 R_MTU_Close

R_MTU_Close		
Synopsis	MTU end processing	
Header	r_mtu3_if.h	
Declaration	mtu_err_t R_MTU_Close (mtu_channel_t channel)	
Description	This function disables the settings of each channel set through compare match, input capture, or PWM.	
Arguments	mtu_channel_t channel	Specifies a MTU channel to be disabled. MTU_CHANNEL_0 MTU_CHANNEL_1 MTU_CHANNEL_2 MTU_CHANNEL_3 MTU_CHANNEL_4 MTU_CHANNEL_6 MTU_CHANNEL_7 MTU_CHANNEL_8
Return value	The result of execution of the MTU end function is returned. MTU_SUCCESS: Normal termination MTU_TIMERS_ERR_CH_NOT_OPEN: Channel operation in progress MTU_TIMERS_ERR_BAD_CHAN: Specified channel invalid	
Supplement	Setting MTU3_CFG_PARAM_CHECKING_ENABLE that is defined by r_mtu3_config.h to 1 enables checking of the parameters of the arguments. Relevant functions: R_MTU_Timer_Open(), R_MTU_Capture_Open(), R_MTU_PWM_Open()	

6.9.11 R_MTU_Control

R_MTU_Control		
Synopsis	MTU control processing	
Header	r_mtu3_if.h	
Declaration	mtu_err_t R_MTU_Control(mtu_channel_t channel, mtu_cmd_t cmd, void * pcmd_data)	
Description	This function handles special operations of the hardware or software for each specified MTU channel.	
Arguments	mtu_channel_t channel	Specifies an MTU channel. MTU_CHANNEL_0 MTU_CHANNEL_1 MTU_CHANNEL_2 MTU_CHANNEL_3 MTU_CHANNEL_4 MTU_CHANNEL_6 MTU_CHANNEL_7 MTU_CHANNEL_8
	mtu_cmd_t cmd	Specifies a command to be controlled MTU_CMD_START MTU_CMD_STOP MTU_CMD_SAFE_STOP MTU_CMD_RESTART MTU_CMD_SYNCHRONIZE MTU_CMD_GET_STATUS MTU_CMD_CLEAR_EVENTS MTU_CMD_SET_CAPT_EDGE
	void * pcmd_data	A specific format must be specified according to the command set by cmd. For details, see Section 6.16, R_MTU_Control Parameters.
Return value	The result of execution of the MTU control function is returned. MTU_SUCCESS: Normal termination MTU_TIMERS_ERR_CH_NOT_OPEN: Channel operation stopped MTU_TIMERS_ERR_BAD_CHAN: Specified channel invalid MTU_TIMERS_ERR_UNKNOWN_CMD: Invalid command MTU_TIMERS_ERR_NULL_PTR: pcmd_data pointer null MTU_TIMERS_ERR_INVALID_ARG: Invalid argument value MTU_TIMERS_ERR_LOCK:	
Supplement	Setting MTU3_CFG_PARAM_CHECKING_ENABLE that is defined by r_mtu3_config.h to 1 enables checking of the parameters of the arguments. Relevant functions: R_MTU_Timer_Open(), R_MTU_Capture_Open(), R_MTU_PWM_Open()	

6.10 Flowcharts

6.10.1 Main Processing

Figure 6.15 show a flow chart of main processing.

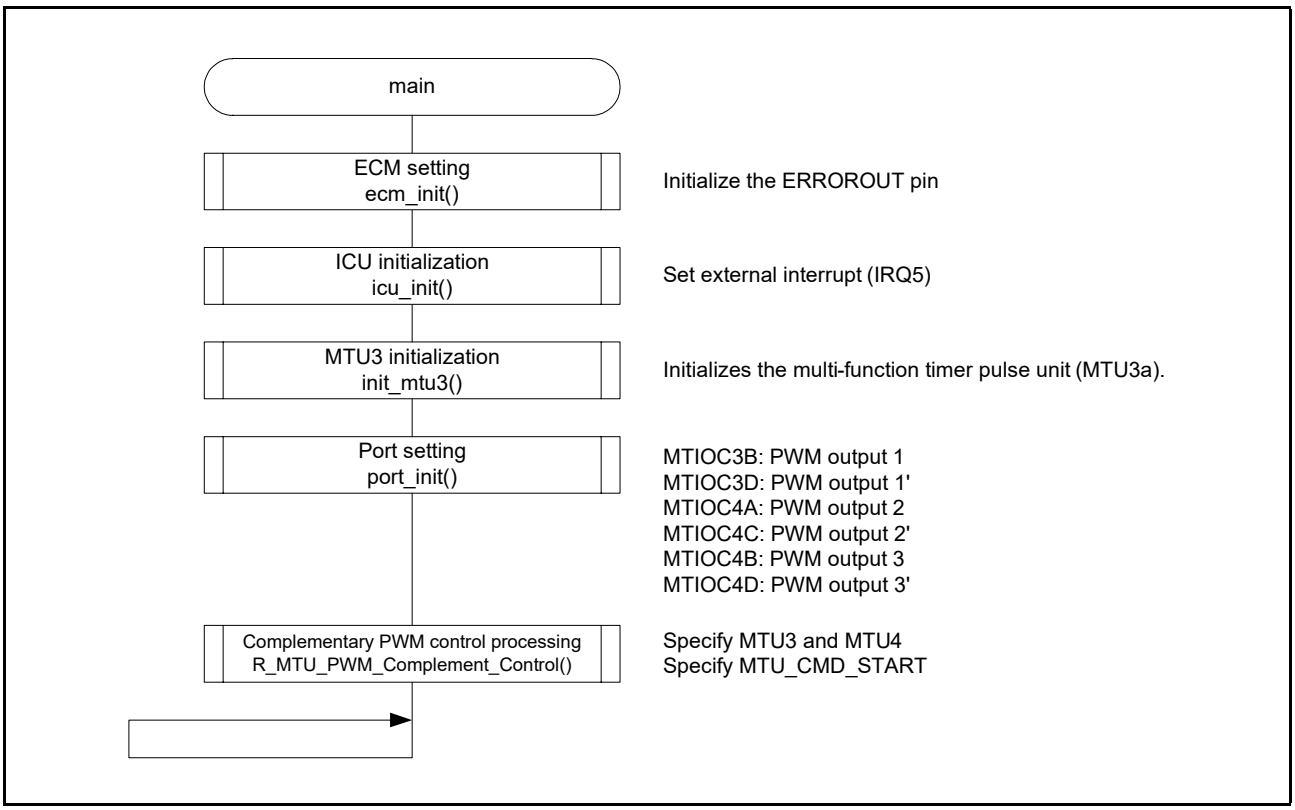


Figure 6.15 Main Processing

6.10.2 Initialization of MTU

Figure 6.16 shows a flowchart of processing of Initialization of the MTU.

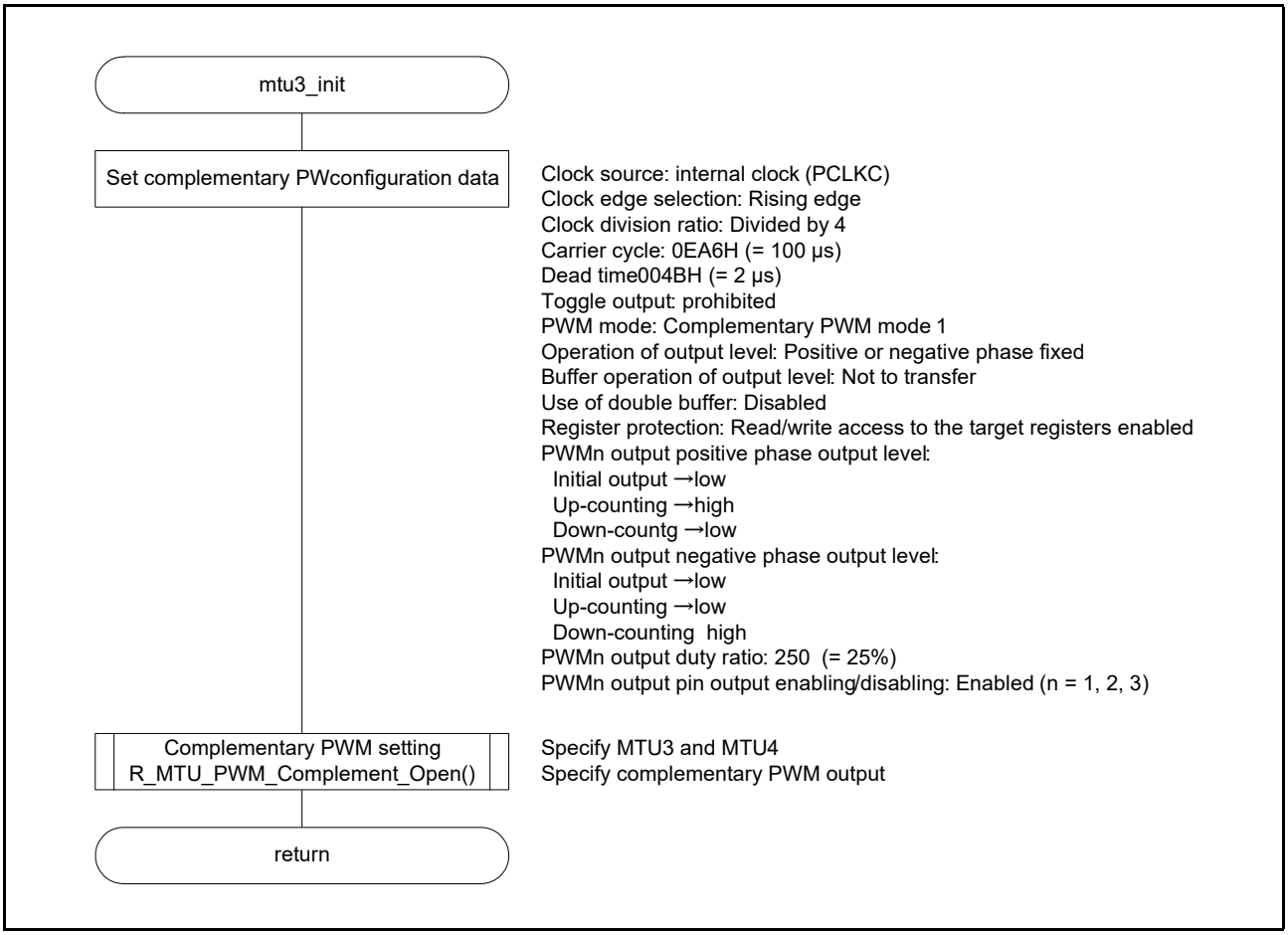
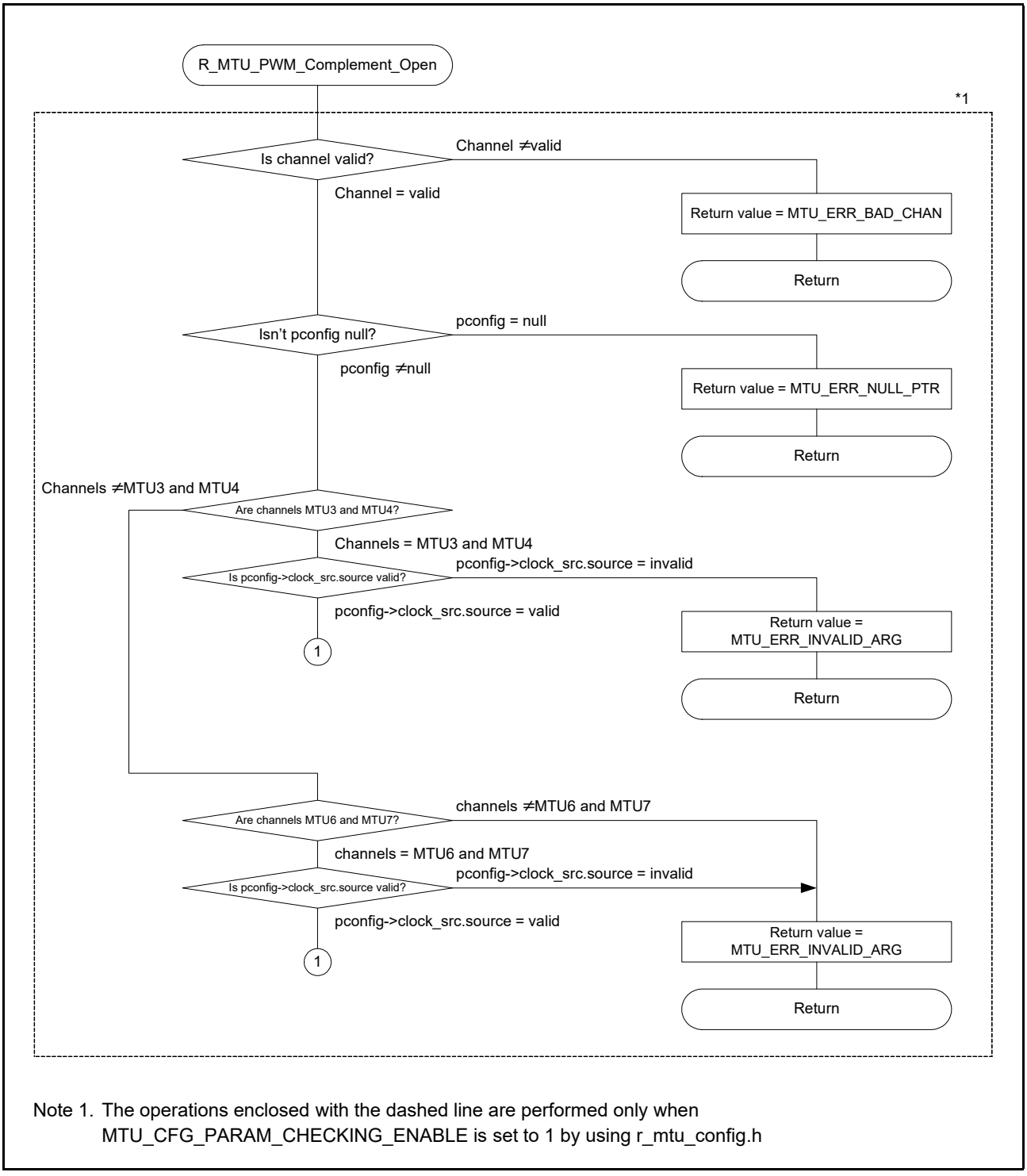


Figure 6.16 Initialization of MTU

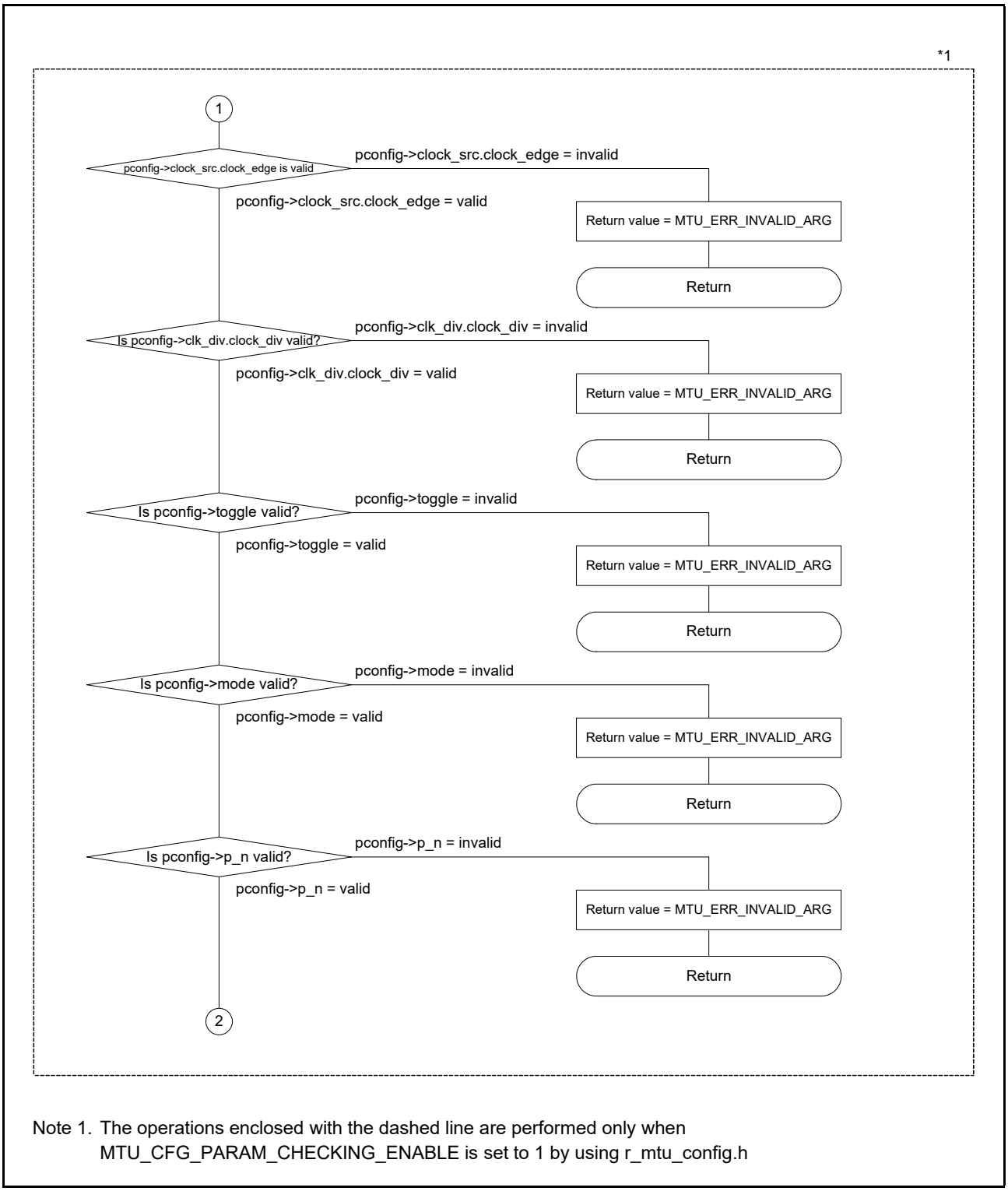
6.10.3 Complementary PWM Mode Setting

Figure 6.17 to Figure 6.21 show flows of processing of complementary PWM mode.



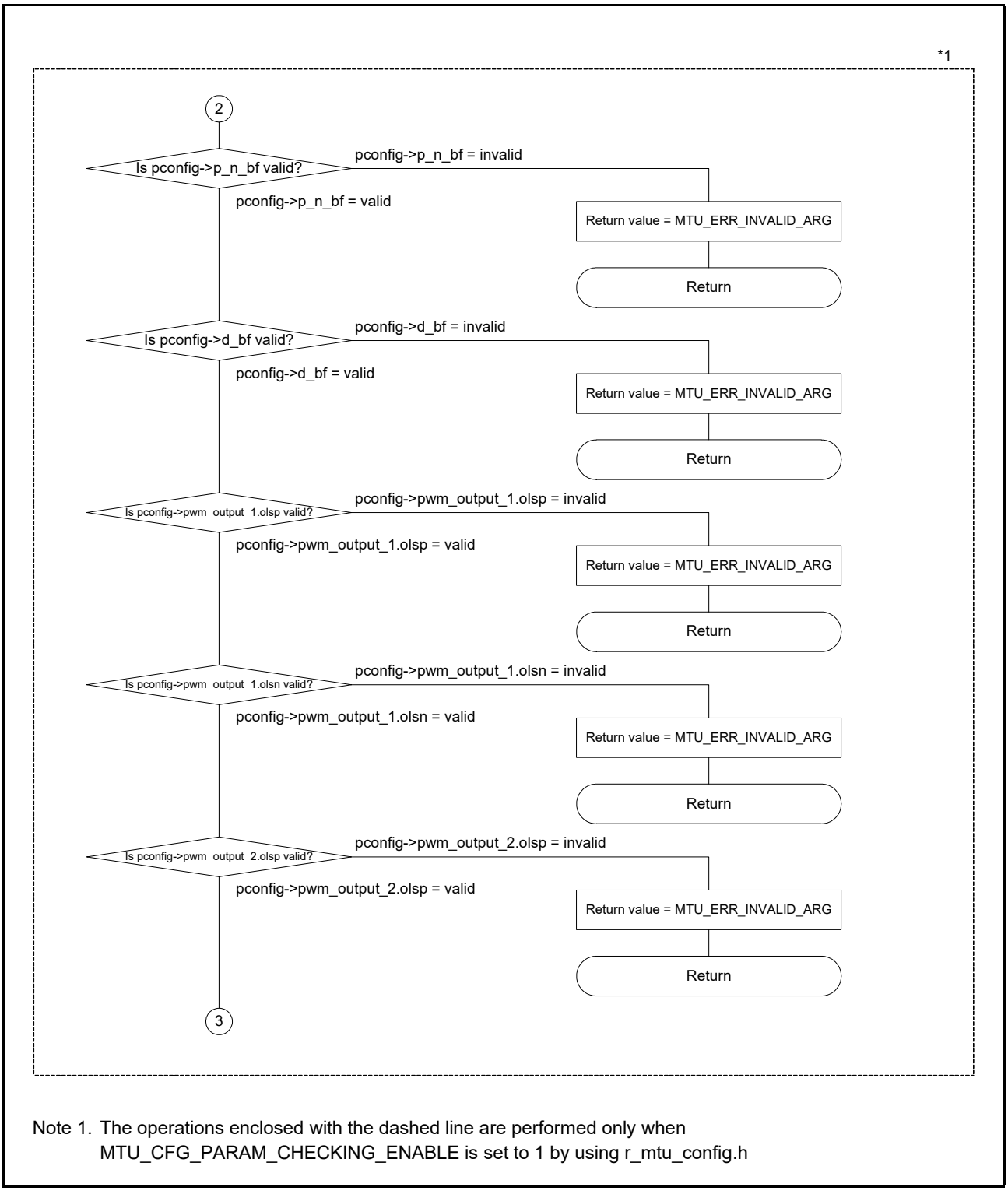
Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.17 Complementary PWM Mode Setting (1)



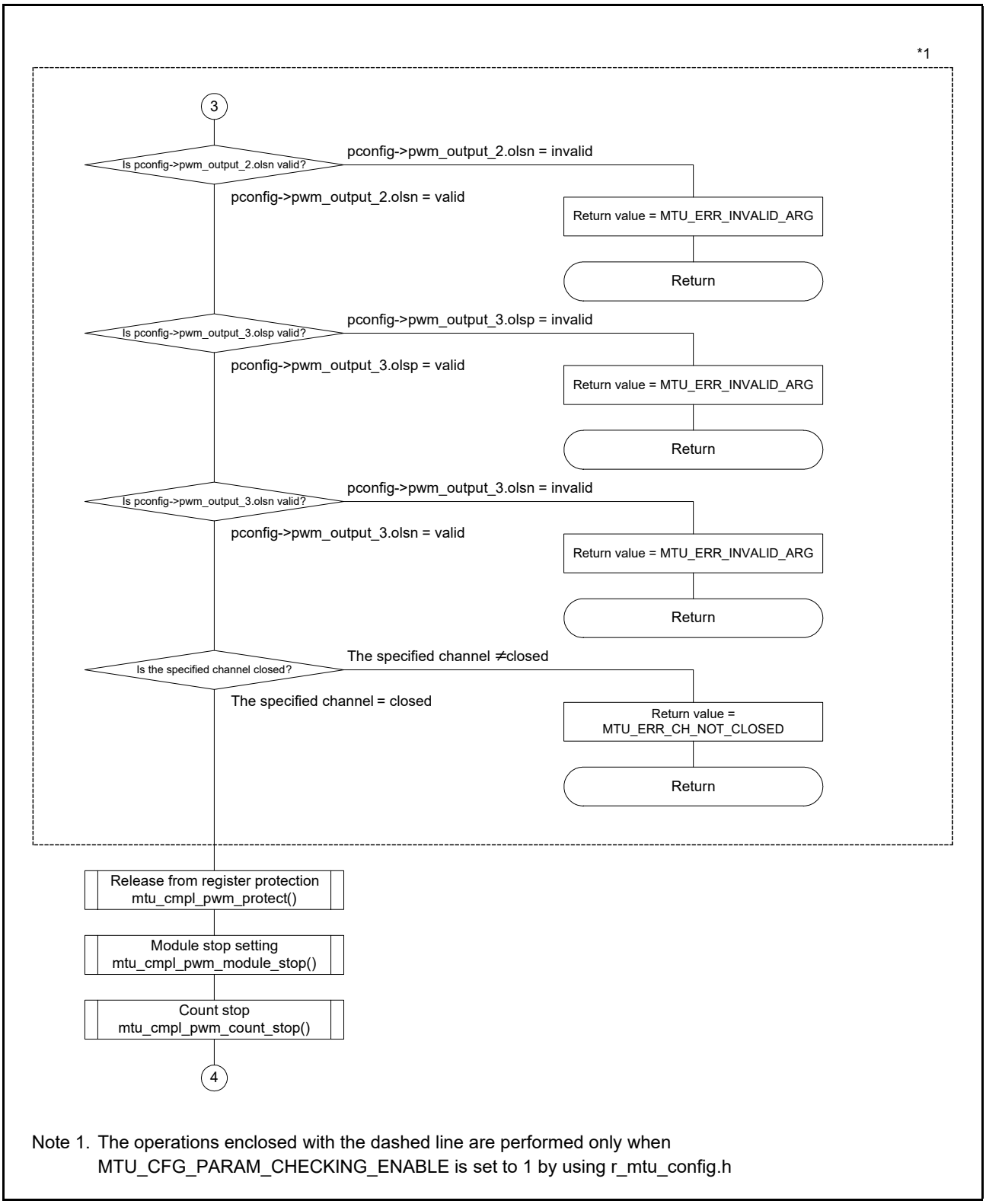
Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.18 Complementary PWM Mode End Processing (2)



Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.19 Complementary PWM Mode End Processing (3)



Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.20 Complementary PWM Mode End Processing (4)

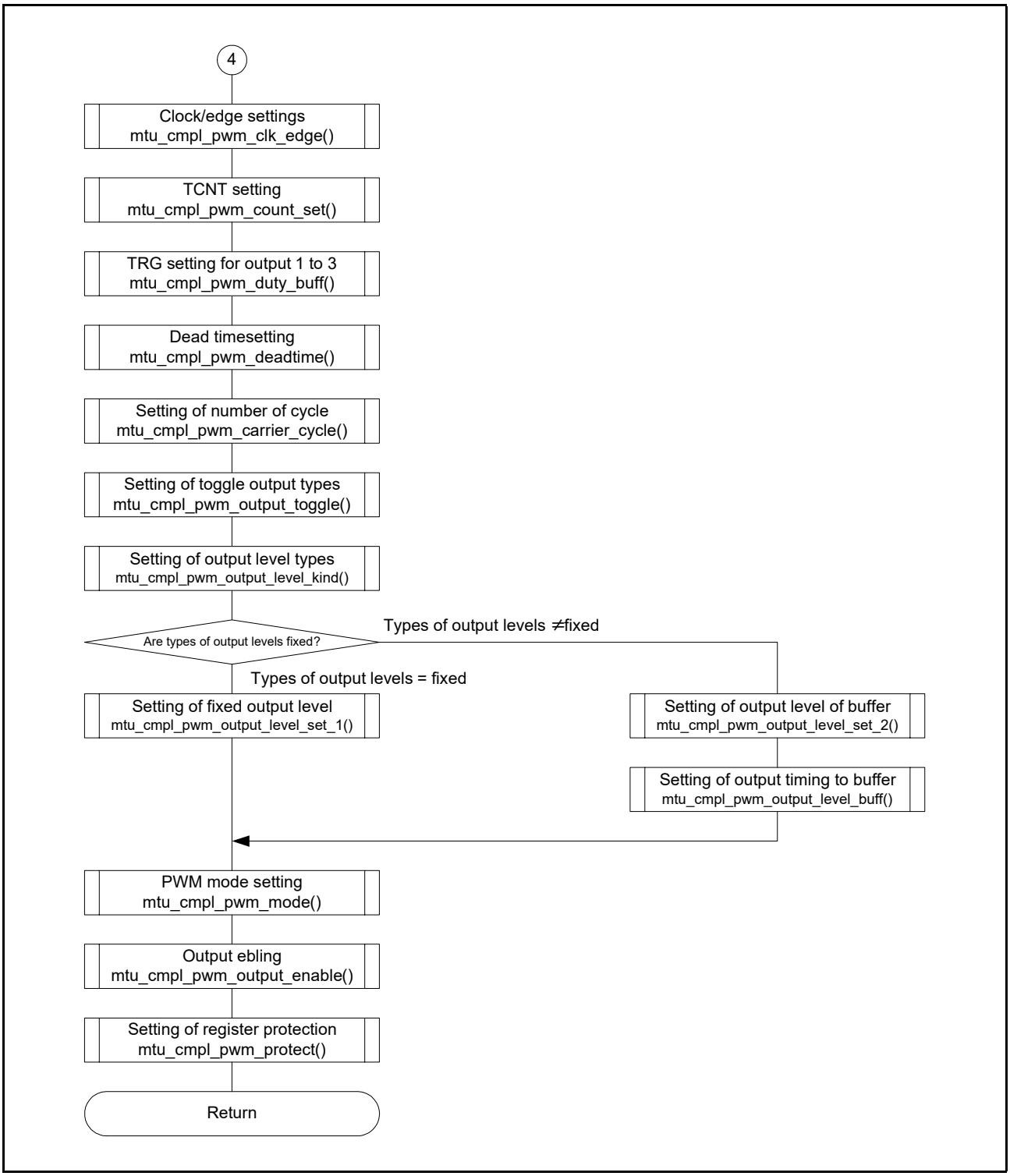
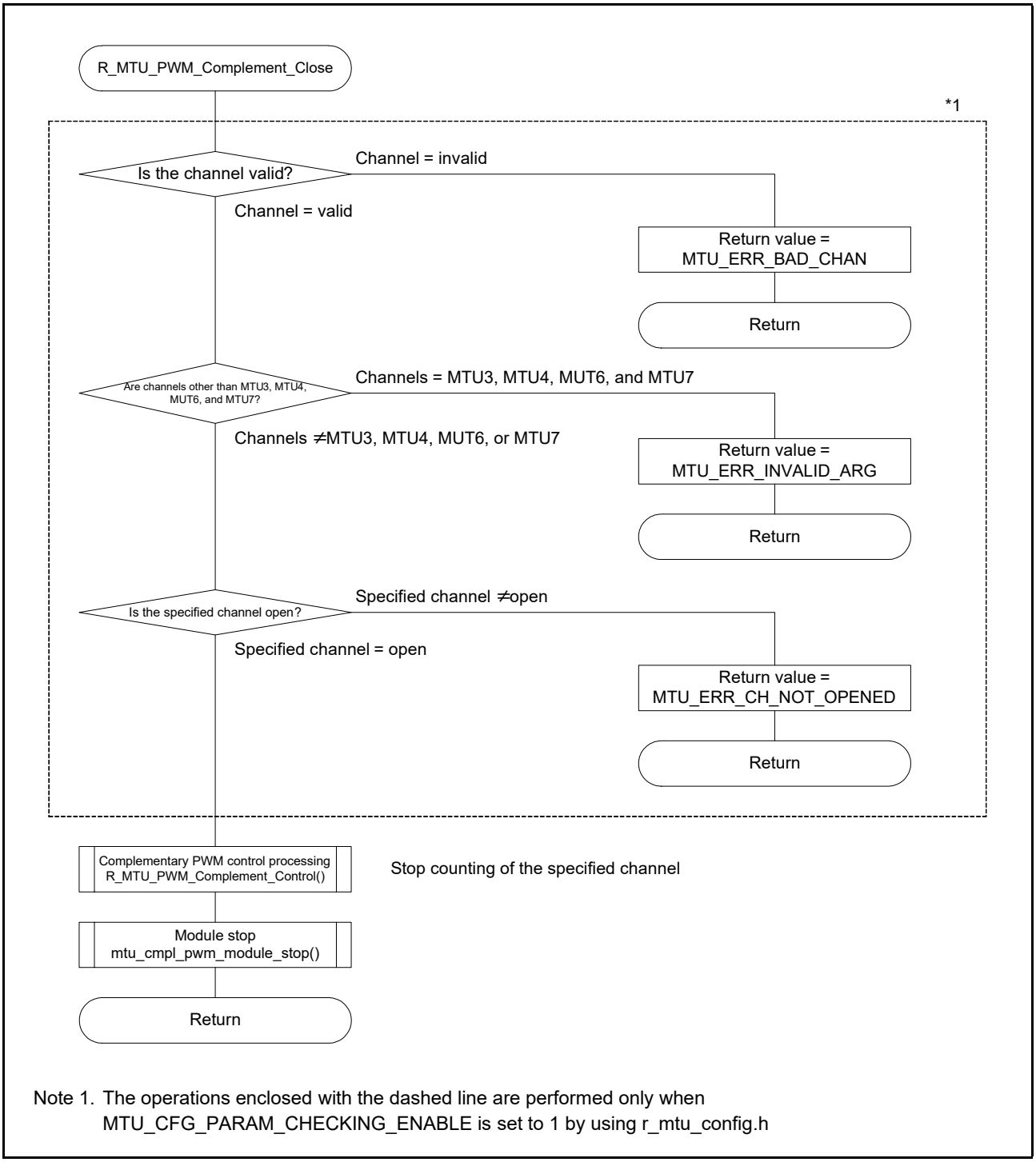


Figure 6.21 Complementary PWM Mode End Processing (5)

6.10.4 Complementary PWM Mode End Processing

Figure 6.22 shows a flowchart of Complementary PWM mode end processing.

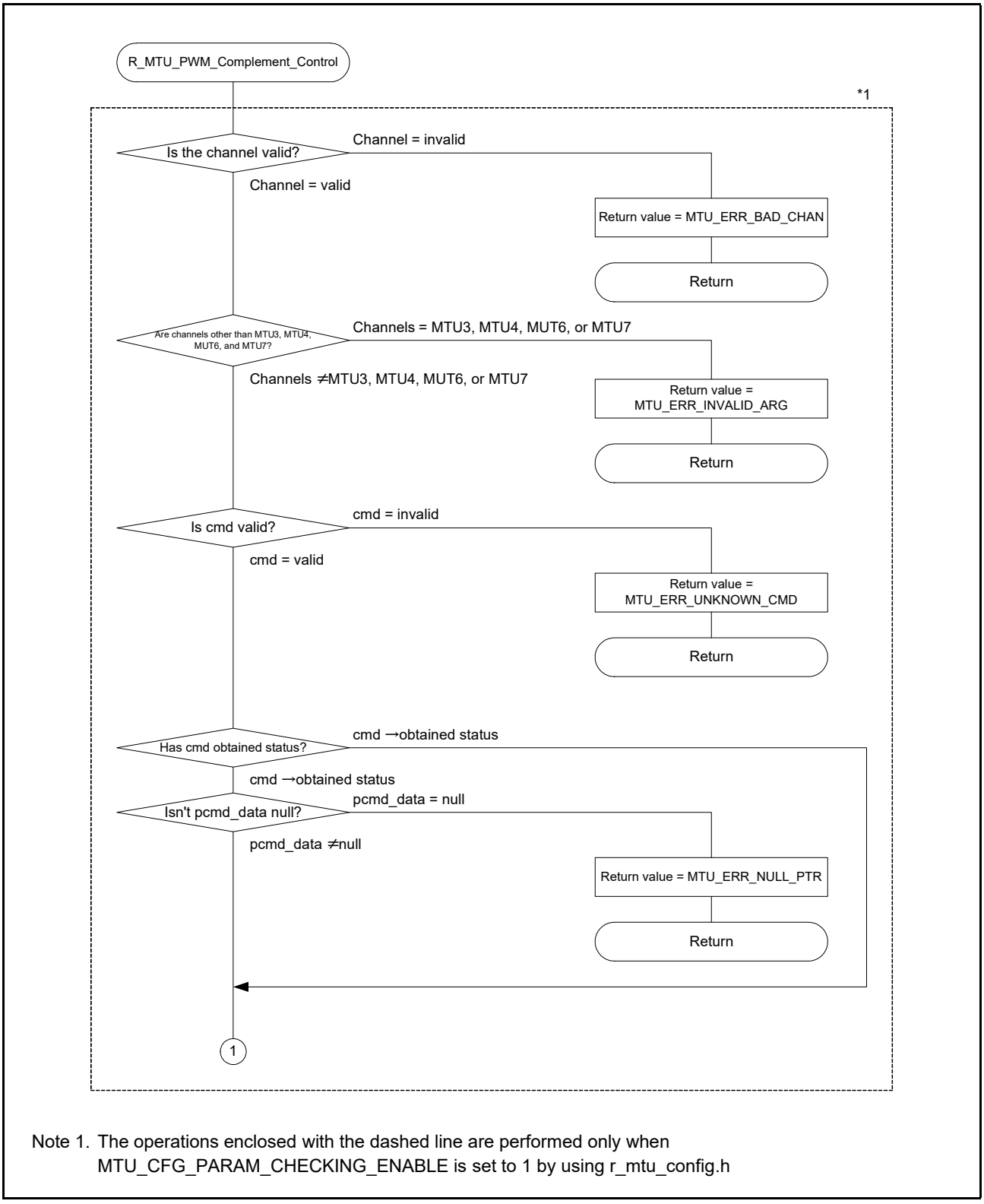


Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.22 Complementary PWM Mode End Processing

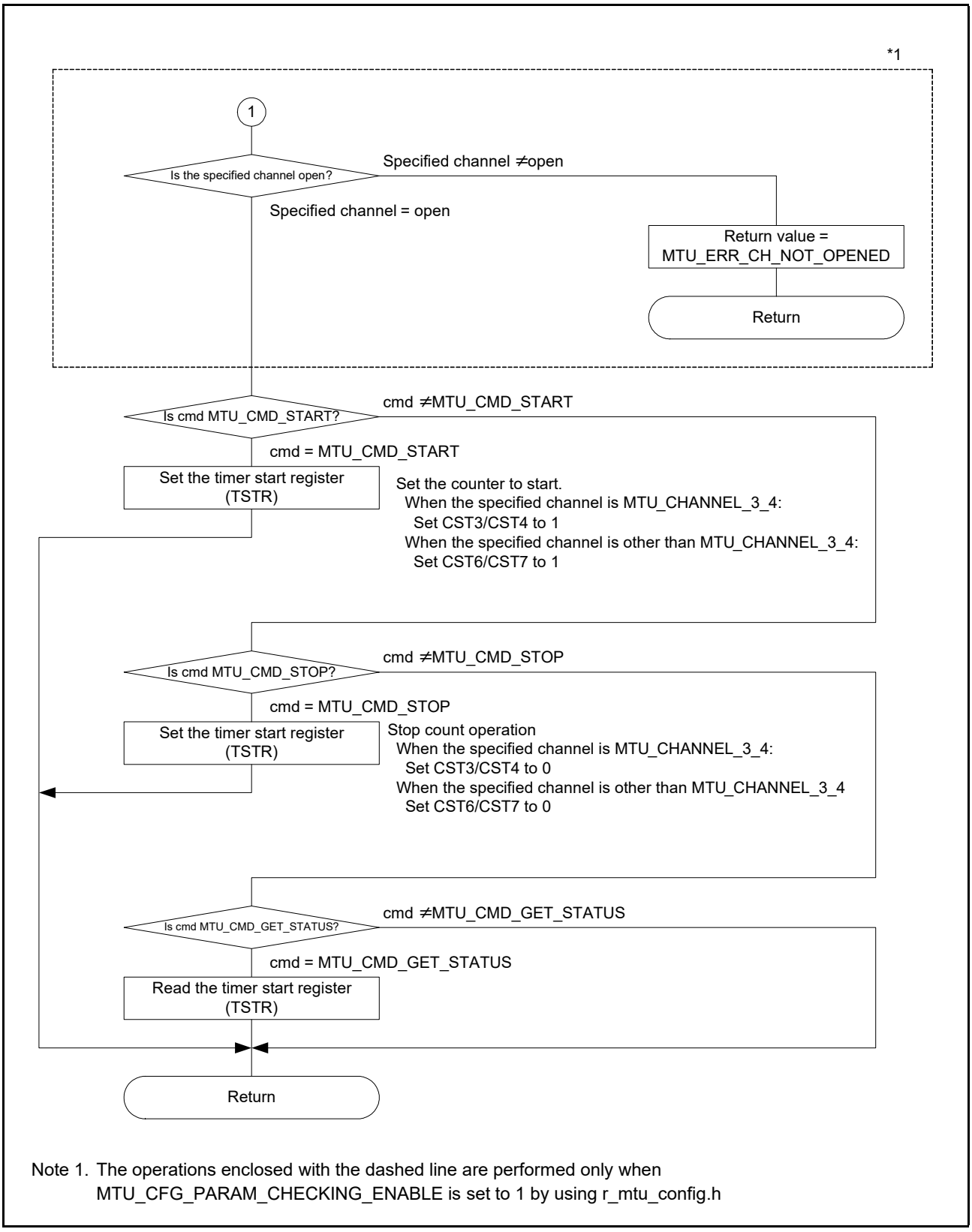
6.10.5 Complementary PWM Controlling

Figure 6.23 shows a flowchart of complementary PWM controlling.



Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.23 Complementary PWM Controlling (1)



Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.24 Complementary PWM Controlling (2)

6.10.6 IRQ9 Interrupt (IRQ Pin Interrupt 5) Processing

Figure 6.25 show a flowchart of IRQ9 interrupt (IRQ pin interrupt 5) processing.

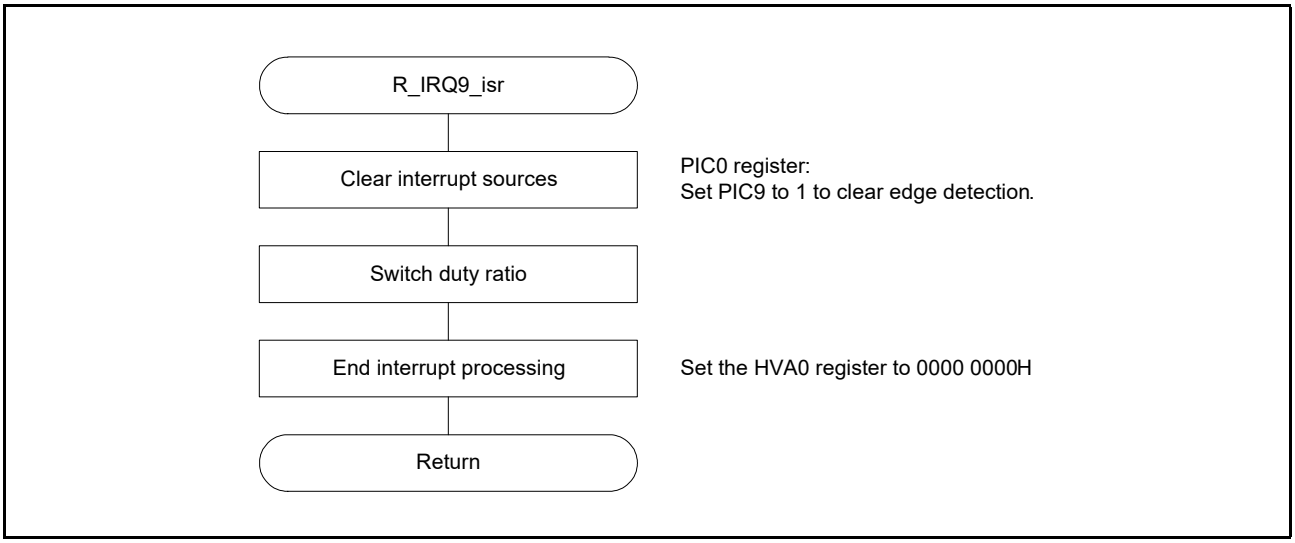
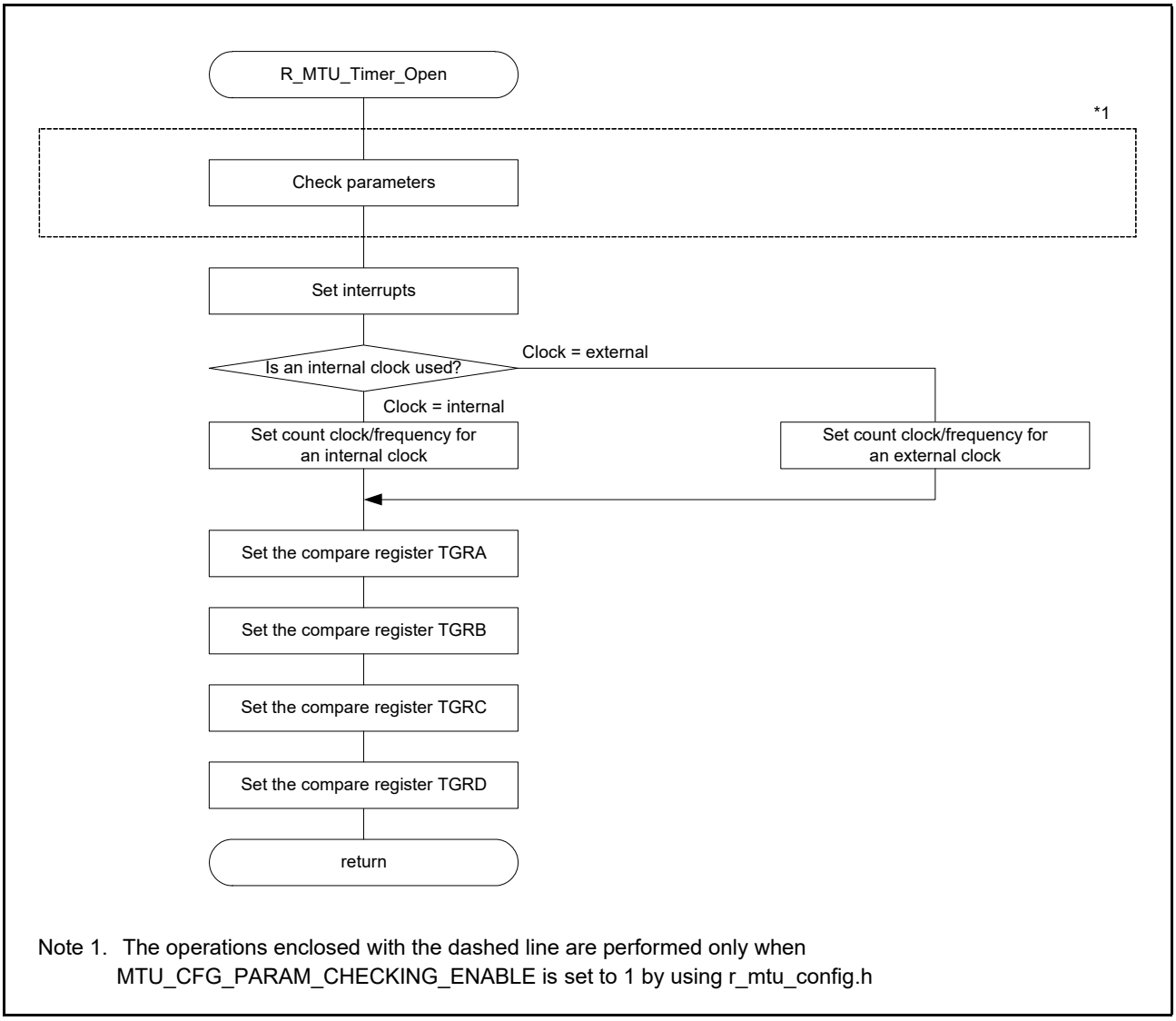


Figure 6.25 IRQ9 Interrupt (IRQ Pin Interrupt 5) Processing

6.10.7 MTU Compare Match Setting

Figure 6.26 shows a flowchart of MTU compare match setting.

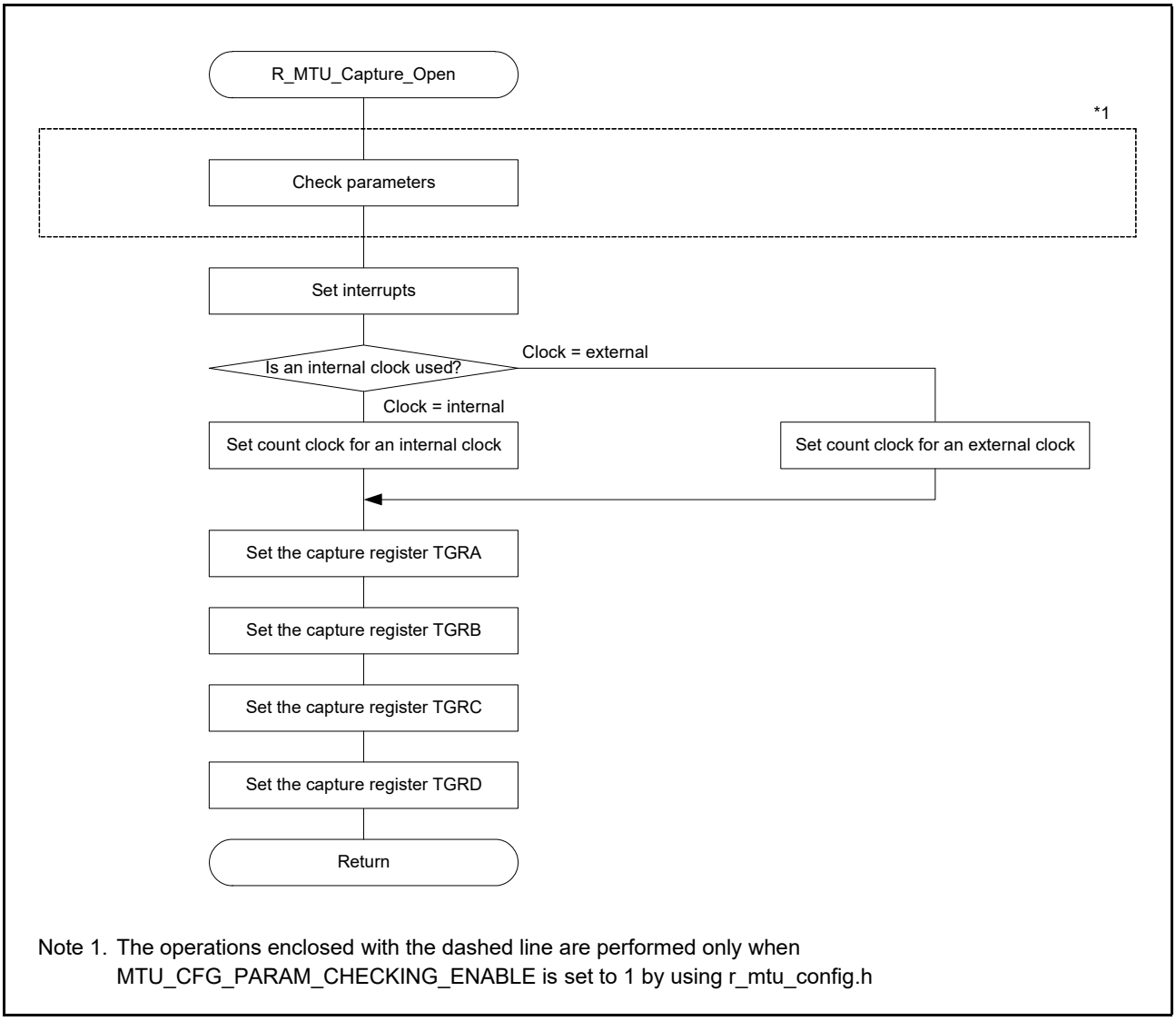


Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.26 MTU Compare Match Setting

6.10.8 MTU Capture Processing

Figure 6.27 shows a flowchart of MTU capture processing.

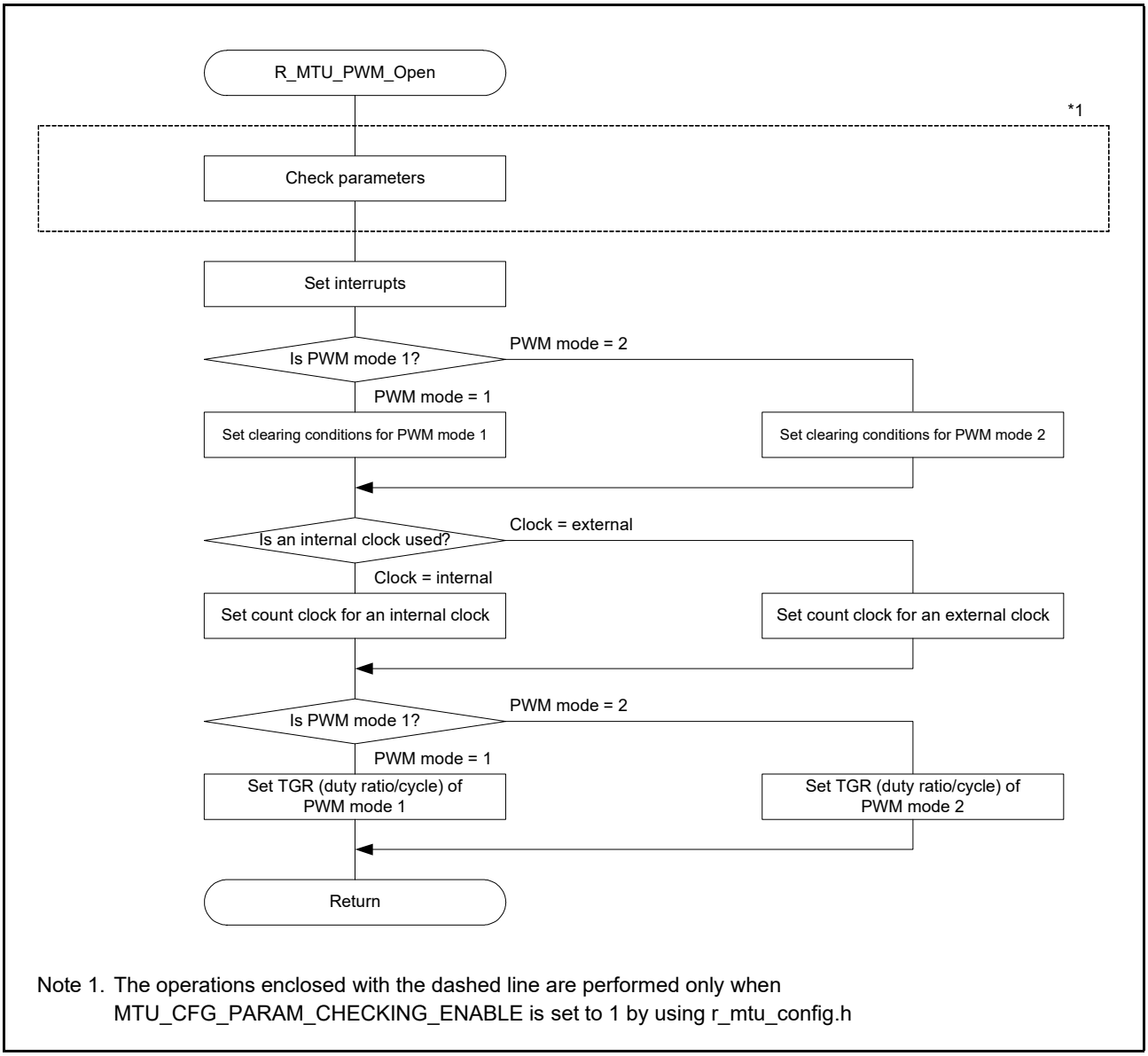


Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.27 MTU Capture Processing

6.10.9 MTU PWM Processing

Figure 6.28 shows a flowchart of MTU PWM processing.

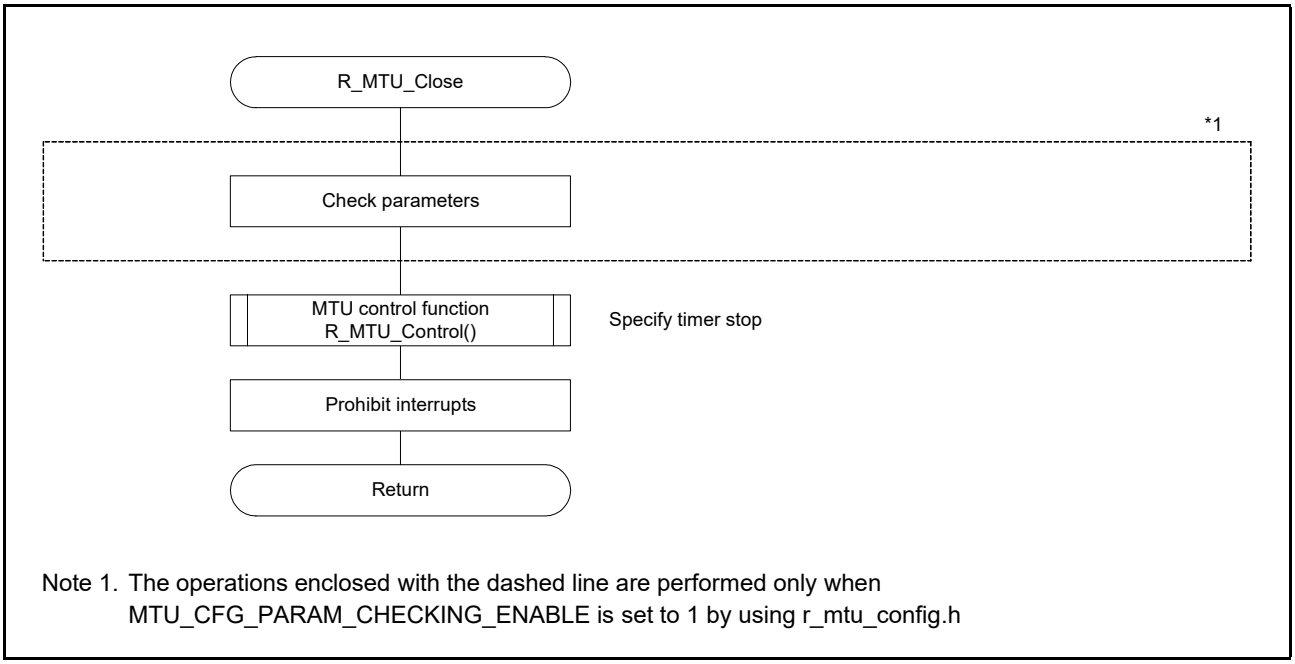


Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.28 MTU PWM Processing

6.10.10 MTU End Processing

Figure 6.29 shows a flowchart of MTU end processing.

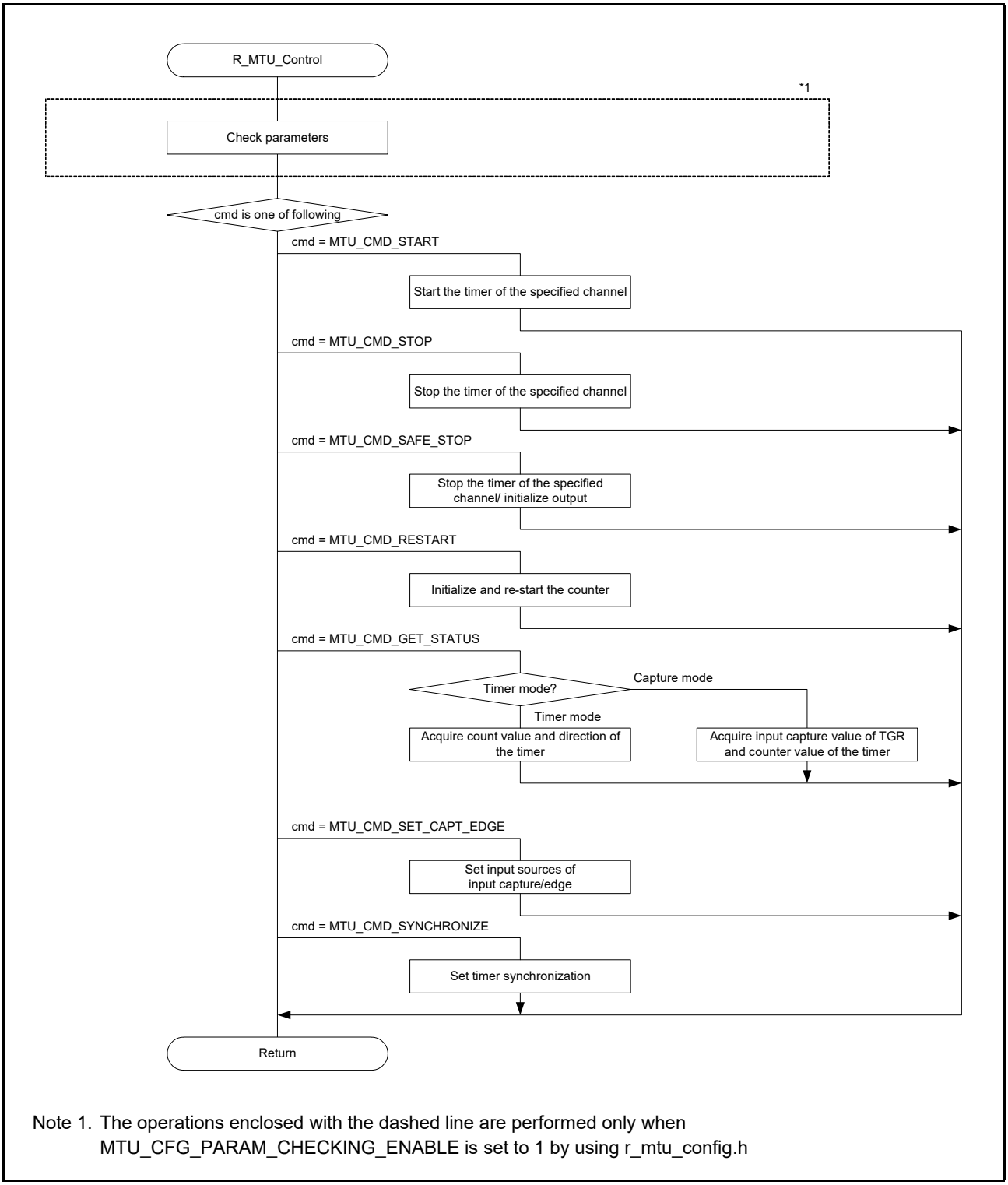


Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.29 MTU End Processing

6.10.11 MTU Controlling

Figure 6.30 shows a flowchart of MTU controlling.



Note 1. The operations enclosed with the dashed line are performed only when MTU_CFG_PARAM_CHECKING_ENABLE is set to 1 by using r_mtu_config.h

Figure 6.30 MTU Controlling

6.11 R_MTU_PWM_Complement_Open Parameters

The parameters (*pconfig) that are used for R_MTU_PWM_Complement_Open are listed in the table below.

Table 6.7 List of R_MTU_PWM_Complement_Open Parameters

Parameter	Description
clock_src.source	Specifies a count clock
clock_src.clock_edge	Specifies a clock edge
clk_div.clock_div	Specifies division ratio of the count clock
clk_div.cycle_freq	Specifies clock count
dead_time	Specifies dead time
toggle	Specifies toggle output synchronized to the cycle of the carrier
mode	Specifies operating modes of complementary PWM
p_n	Specifies operation of the output level
p_n_bf	Specifies operations of the buffer for output level
d_bf	Enables or disables the double buffer functions
protect	Sets write-protection capability against accidental modification in the MTU3 and MTU4
pwm_output_X.olsp (X = 1 to 3)	Specifies output level of the normal phase of PWM output X (X = 1 to 3)
pwm_output_X.olsn (X = 1 to 3)	Specifies output level of the inverted phase of PWM output X (X = 1 to 3)
pwm_output_X.duty (X = 1 to 3)	Specifies duty ratio of PWM output X (X = 1 to 3)
pwm_output_X.output (X = 1 to 3)	Enables or disables pin output of PWM output X (X = 1 to 3)

6.11.1 clock_src.source

clock_src.source		
Synopsis	Setting count clock	
Header	r_mtu3_if.h	
Description	This parameter specifies a count clock on MTU channels (MTU3 and MTU4, or MTU6 and MTU7) When an external clock is selected, the I/O port of the target pin and multi-function pin controller (MPC) must be set in advance.	
Parameters	MTU_CLK_SRC_EXT_MTCLKA*1	Specifies an external clock (MTCLKA pin input)
	MTU_CLK_SRC_EXT_MTCLKB*1	Specifies an external clock (MTCLKB pin input)
	MTU_CLK_SRC_INTERNAL	Specifies an internal clock (PCLKC)
Supplement	Note 1. Available only when channel 3 and 4 are selected.	

6.11.2 clock_src.clock_edge

clock_src.clock_edge		
Synopsis	Setting clock edge	
Header	r_mtu3_if.h	
Description	This parameter specifies an edge that counts an internal clock on the MTU channels (MTU3 and MTU4, or MTU6 and MTU7)	
Parameters	MTU_CLK_RISING_EDGE	Specifies counting at the rising edge
	MTU_CLK_FALLING_EDGE	Specifies counting at the falling edge
	MTU_CLK_ANY_EDGE	Specifies counting at the rising/falling edges
Supplement	When the clock source to drive counting of the specified timer cycle (clk_div) is PCLKC/1, specification of an edge is ignored and handling is as the initial value (rising edge).	

6.11.3 clk_div.clock_div

clk_div.clock_div		
Synopsis	Setting division ratio of count clock	
Header	r_mtu3_if.h	
Description	This parameter selects division ratio of the clock for the MTU channels (MTU3 and MTU4, or MTU6 and MTU7).	
Parameters	MTU_SRC_CLK_DIV_1:	Count at PCLKC/1
	MTU_SRC_CLK_DIV_2:	Count at PCLKC/2
	MTU_SRC_CLK_DIV_4:	Count at PCLKC/4
	MTU_SRC_CLK_DIV_8:	Count at PCLKC/8
	MTU_SRC_CLK_DIV_16:	Count at PCLKC/16
	MTU_SRC_CLK_DIV_32:	Count at PCLKC/32
	MTU_SRC_CLK_DIV_64:	Count at PCLKC/64
	MTU_SRC_CLK_DIV_256:	Count at PCLKC/256
	MTU_SRC_CLK_DIV_1024:	Count at PCLKC/1024
Supplement	None	

6.11.4 clk_div.cycle_freq

clk_div.cycle_freq		
Synopsis	Setting carrier cycle of complementary PWM	
Header	r_mtu3_if.h	
Description	This parameter specifies the cycle of the carrier for the MTU channels (MTU3 and MTU4, or MTU6 and MTU7). One half of the value specified here is set in the timer cycle data register (TCDRA or TCDRB).	
Parameters	Value	Setting value of the cycle of the carrier (2 to 1FFFEh)
Supplement	None	

6.11.5 dead_time

dead_time		
Synopsis	Setting dead time of complementary PWM	
Header	r_mtu3_if.h	
Description	This parameter specifies a setting value as complementary PWM dead time to the dead time data register (TDDRA or TDDRB).	
Parameters	Value	Value set in the dead time data register (0 to FFFFh)
Supplement	None	

6.11.6 toggle

toggle		
Synopsis	Setting toggled output synchronized with cycle of carrier	
Header	r_mtu3_if.h	
Description	This parameter selects toggled output that is synchronized with the cycle of the carrier from MTIOC3A or MTIOC6A. When MTIOC3A or MTIOC6A is in use, the I/O port of the target pin and multi-function pin controller (MPC) must be set in advance.	
Parameters	MTU_TOGGLE_OFF	Toggle output OFF
	MTU_TOGGLE_ON	Toggle output ON
Supplement	PWM modes are basically not used. This is provided for used in evaluation and so on.	

6.11.7 mode

mode		
Synopsis	Setting operating mode of complementary PWM	
Header	r_mtu3_if.h	
Description	This parameter selects an operating mode for complementary PWM from 3 operating 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 /trough)
Supplement	None	

6.11.8 p_n

p_n		
Synopsis	Setting output level operation	
Header	r_mtu3_if.h	
Description	This parameter selects output level of complementary PWM output (normal or inverted phase) either of fixed setting or buffer operation setting. When the output level is set to fixed, the output level can be determined by setting the timer output control register 1 (TOCR1A or TOCR1B). When output level is set to buffer operation, the output level can be determined by setting the timer output control register 2 (TOCR2A, TOCR2B) or buffer register (TOLBRA, TOLBRB).	
Parameters	MTU_PIN_P_N_1*1	Fixes the output level of normal- or inverted-phase.
	MTU_PIN_P_N_2	Selects the output level (normal- or inverted-phase), that is specified in the buffer operation
Supplement	Note 1. When the output level is fixed, the output level of normal- or inverted-phase becomes common to PWM output 1 to 3.	

6.11.9 p_n_bf

p_n_bf		
Synopsis	Setting buffer operation of output level	
Header	r_mtu3_if.h	
Description	This parameter selects a transfer trigger of the output level (normal- or inverted-phase) from the buffer of the 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/trough
Supplement	Note 1. Enabled only when MTU_PIN_P_N_2 is selected by using p_n	

6.11.10 d_bf

d_bf		
Synopsis	Setting usage of double buffer function	
Header	r_mtu3_if.h	
Description	This parameter enables or disables the double buffer function of the complementary PWM output.*1 Enabling this parameter changes the minimum resolution of the PWM output from ±2 to ±1 when PWM output is changed.	
Parameters	MTU_CMPL_PWM_D_BF_OFF	Disables the double buffer function.
	MTU_CMPL_PWM_D_BF_ON	Enables the double buffer function.
Supplement	Note 1. The double buffer function can be set only when complementary PWM mode 3 (to be transferred at crest/trough) is selected.	

6.11.11 protect

protect		
Synopsis	Setting write-protection capability against accidental modification in MTU3 and MTU4	
Header	r_mtu3_if.h	
Description	This parameter sets write-protection capability against accidental modification in the target registers of MTU3 and MTU4 and target counter.	
Parameters	MTU_CMPL_PWM_PROTECT_OFF	Enables read/write
	MTU_CMPL_PWM_PROTECT_ON	Disables read/write
Supplement	For details of the target registers and counters for write-protection capability against accidental modification, refer to section of the timer read/write enable registers (TRWERA, TRWERB) in the user's manual.	

6.11.12 pwm_output_X.olsp (X = 1 to 3)

pwm_output_X.olsp		
Synopsis	Setting output level of normal phase for PWM output X (X = 1 to 3)	
Header	r_mtu3_if.h	
Description	This parameter specifies the output level of the normal phase for complementary PWM output X (X = 1 to 3).	
Parameters	MTU_PIN_OLSP_HI_UPLO_DNHI	Initial output: High, count-up: Low, count-down: High
	MTU_PIN_OLSP_LO_UPHI_DNLO	Initial output: Low, count-up: High, count-down: Low
Supplement	None	

6.11.13 pwm_output_X.olsn (X = 1 to 3)

pwm_output_X.olsn		
Synopsis	Setting output level of the inverted phase of complementary PWM output X (X = 1 to 3)	
Header	r_mtu3_if.h	
Description	This parameter selects output level of the inverted phase for complementary PWM output X (X = 1 to 3).	
Parameters	MTU_PIN_OLSN_HI_UPHI_DNLO	Initial output: High, count-up: High, count-down: Low
	MTU_PIN_OLSN_LO_UPLO_DNHI	Initial output: Low, count-up: Low, count-down: High
Supplement	None	

6.11.14 pwm_output_X.duty (X = 1 to 3)

pwm_output_X.duty		
Synopsis	Setting duty ratio of complementary PWM output X (X = 1 to 3)	
Header	r_mtu3_if.h	
Description	This parameter sets duty ratio of the complementary PWM output X (X = 1 to 3). Duty ratio is specified in 0.1% units and can be set in the range from 0% to 100%.	
Parameters	Value (in 0.1% units)	Specifies duty ratio in 0.1% units. (0 to 1000)
Supplement	None	

6.11.15 pwm_output_X.output (X = 1 to 3)

pwm_output_X.output		
Synopsis	Enabling/disabling pin output for PWM output X (X = 1 to 3)	
Header	r_mtu3_if.h	
Description	This parameter enables or disables pin output of complementary PWM output X (X = 1 to 3) for each pair of normal and inverted phases. When MTU3 and 4 are selected: PWM output 1: MTIOC3B (normal phase) MTIOC3D (inverted phase) PWM output 2: MTIOC4A (normal phase) MTIOC4C (inverted phase) PWM output 3: MTIOC4B (normal phase) MTIOC4D (inverted phase) When MTU6 and 7 are selected: PWM output 1: MTIOC6B (normal phase) MTIOC6D (inverted phase) PWM output 2: MTIOC7A (normal phase) MTIOC7C (inverted phase) PWM output 3: MTIOC7B (normal phase) MTIOC7D (inverted phase) When the PWM output pin is used, the I/O port of the target pin and multi-function pin controller (MPC) must be set in advance.	
Parameters	MTU_CMPL_PWM_OUTPUT_OFF	Not use as complementary PWM output
	MTU_CMPL_PWM_OUTPUT_ON	Uses as complementary PWM output
Supplement	None	

6.12 R_MTU_PWM_Complement_Control Parameters

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

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

6.12.1 When cmd is MTU_CMD_START

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

6.12.2 When cmd is MTU_CMD_STOP

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

6.12.3 When cmd is MTU_CMD_GET_STATUS

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

Table 6.8 Parameters When cmd = MTU_CMD_GET_STATUS

Parameter	Description
c_st	Counting state of the timer (operating, stop)
d_st	Counting direction of the timer

6.13 R_MTU_Timer_Open Parameters

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

Table 6.9 R_MTU_Timer_Open Parameters

Parameter	Description
clock_src.source	Sets a count clock
clock_src.clock_edge	Sets an edge of the clock
clear_src	Specifies a source for clearing the counter
timer_X.actions.freq (X = a, b, c, d)	Specifies the compare match cycle (Hz) of the timer general register TGRX (X = a, b, c, d)
timer_X.actions.do_action (X = a, b, c, d)	Specifies operations of the timer general register TGRX (X = a, b, c, d)
timer_X.actions. output (X = a, b, c, d)	Specifies pin output level of the timer general register TGRX (X = a, b, c, d)

6.13.1 clock_src.source

clock_src.source

Synopsis: Setting count clock

Header: r_mtu3_if.h

Description: This parameter sets a count clock on each MTU channel.
When the internal clock (PCLKC) is selected, the I/O port of the target pin and multi-function pin controller (MPC) must be set in advance.
When the internal clock (PCLKC) is selected, a division ratio that is more suitable than the specified timer cycle (timer_X.actions.freq) is automatically selected

Parameters:

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

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

6.13.2 clock_src.clock_edge

clock_src.clock_edge

Synopsis: Setting clock edge

Header: r_mtu3_if.h

Description: This parameter sets an edge for counting an internal clock on each MTU channel.

Parameters:

MTU_CLK_RISING_EDGE	Specifies counting at the rising edge
MTU_CLK_FALLING_EDGE	Specifies counting at the falling edge
MTU_CLK_ANY_EDGE	Specifies counting at the rising/falling edges

Supplement: 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, specification of an edge is ignored and handling is as the initial value (rising edge).

6.13.3 clear_src

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

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

timer_X.actions.freq		
Synopsis	Setting compare match cycle (Hz) of timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies the TGR cycle (Hz) on each MTU channel. When the internal clock (PCLKC) is selected, a PCLKC division ratio and TGR setting value (count value) is automatically calculated. When an external clock is selected, the value of this parameter is simply set to TGR and handled as the value for counting until a compare-match.	
Parameters	Value (in Hz units)	Specifies a cycle in Hz units Specifies the TGR count value (up to FFFFh) for an external clock
Supplement	Setting range is from 3 to 100000000 Hz (1 to 100000000Hz for channel 8). When both edges are set by using clock_src.clock_edge, the upper limit becomes 60000000Hz.	

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

timer_X.actions. do_action		
Synopsis	Setting operation of timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies TGR operation on each MTU channel. When the MTU pin is specified as an output pin, the I/O port of the target pin and multi-function pin controller (MPC) must be set in advance. Multiple operations can be set 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 executes the callback function that is specified by the user in the interrupt service routine.
	MTU_ACTION_TRIGGER_ADC*2	Specifies a trigger startup of the A/D converter due to a TGRA compare match.
	MTU_ACTION_REPEAT*3	Continues timer count operation after the first compare match.
	MTU_ACTION_NONE	This parameter is used when TGR is not used. Specifying the parameter with other parameters is prohibited.
Supplement	Note 1. Interrupt priority needs to be specified in advance by using r_mtu3_config.h. Note 2. Available for channels 0 to 4, 6, and 7 Note 3. When not specified, this parameter stops timer count operation in the interrupt service routine. For this reason, timer count operation always continues if interrupt enabling (MTU_ACTION_INTERRUPT or MTU_ACTION_CALLBACK) is not specified.	

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

timer_X.actions. output		
Synopsis	Setting pin output level of timer general register TGRX(X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies TGR pin output level on each MTU channel.	
Parameters	MTU_PIN_NO_OUTPUT	Disables output
	MTU_PIN_LO_GOLO	Specifies initial output as low, and output of the low level upon a compare match.
	MTU_PIN_LO_GOHI	Specifies initial output as low, and output of the high level upon a compare match.
	MTU_PIN_LO_TOGGLE	Specifies initial output as low and output of toggle upon a compare match.
	MTU_PIN_HI_GOLO	Specifies initial output as high and output of the low upon a compare match.
	MTU_PIN_HI_GOHI	Specifies initial output as high and output of the high upon a compare match.
	MTU_PIN_HI_TOGGLE	Specifies initial output as high and output of toggle upon a compare match.
Supplement	When _GOLO or _GOHI is specified by the parameter, the output level is retained after the MTU pin output is set at low or high at the first compare match. The output level will not be change even after another compare match. When _TOGGLE is specified, toggle is output by switching the output level of the MTU pin level at every compare match.	

6.14 R_MTU_Capture_Open Parameters

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

Table 6.10 R_MTU_Capture_Open Parameters

Parameter	Description
clock_src.source	Sets a count clock.
clock_src.clock_edge	Sets an edge of the clock
clock_div	Specifies division ratio of an internal clock.
clear_src	Specifies a source for clearing the counter
capture_X.actions (X = a, b, c, d)	Specifies operations of the timer general register TGRX (X = a, b, c, d)
capture_X.capture_edge (X = a, b, c, d)	Specifies an effective edge of the pin output level of the timer general register TGRX (X = a, b, c, d)
capture_X.filter_enable (X = a, b, c, d)	Specifies a noise filter of the corresponding MTU input pin MTIOCnX (X = a, b, c, d) (n = 0 to 4, 6 to 8)

6.14.1 clock_src.source

clock_src.source

Synopsis: Setting count clock

Header: r_mtu3_if.h

Description: This parameter specifies a count clock on each MTU channel.
When an external clock is selected, the I/O port of the target pin and multi-function pin controller (MPC) must be set in advance.

Parameters:

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

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

6.14.2 clock_src.clock_edge

clock_src.clock_edge

Synopsis: Selecting clock edge

Header: r_mtu3_if.h

Description: This parameter specifies an edge for counting an internal clock on each MTU channel.

Parameters:

MTU_CLK_RISING_EDGE	Specifies counting at the rising edge
MTU_CLK_FALLING_EDGE	Specifies counting at the falling edge
MTU_CLK_ANY_EDGE	Specifies counting at the rising/falling edges

Supplement: When the clock source to drive the division ratio of the specified internal clock (clock_div) is PCLKC/1 or is specified as overflows or underflows of MTU2.TCNT, specification of an edge is ignored and handling is as the initial value (rising edge).

6.14.3 clock_div

clock_div		
Synopsis	Setting division ratio of internal clock	
Header	r_mtu3_if.h	
Description	This parameter specifies division ratio of an internal clock on each MTU channel. This parameter setting is disabled when an external clock is selected by using clock_src.clock_edge.	
Parameters	MTU_SRC_CLK_DIV_1	Specifies an internal clock (PCLKC/1)
	MTU_SRC_CLK_DIV_2	Specifies an internal clock (PCLKC/2)
	MTU_SRC_CLK_DIV_4	Specifies an internal clock (PCLKC/4)
	MTU_SRC_CLK_DIV_8	Specifies an internal clock (PCLKC/8)
	MTU_SRC_CLK_DIV_16	Specifies an internal clock (PCLKC/16)
	MTU_SRC_CLK_DIV_32	Specifies an internal clock (PCLKC/32)
	MTU_SRC_CLK_DIV_64	Specifies an internal clock (PCLKC/64)
	MTU_SRC_CLK_DIV_256	Specifies an internal clock (PCLKC/256)
	MTU_SRC_CLK_DIV_1024	Specifies an internal clock (PCLKC/1024)
Supplement	None	

6.14.4 clear_src

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

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

capture_X.actions		
Synopsis	Setting operation of timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies TGR operation on each MTU channel. When MTIOCnm pin (n = 0 to 4, 6, 7, 8; m = a, b, c, d) is used as input capture input, the I/O port of the target pin and multi-function pin controller (MPC) must be set 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 TGR as input capture operation
	MTU_ACTION_INTERRUPT*2	Enables input capture interrupts
	MTU_ACTION_CALLBACK	Enables input capture interrupts and executes the callback function that is specified by the user in the interrupt service routine.
	MTU_ACTION_TRIGGER_ADC*3	Specifies trigger startup of the A/D converter upon TGRA compare match.
	MTU_ACTION_REPEAT*4	Continues timer count operation even after the first input capture
	MTU_ACTION_NONE	This parameter is specified when TGR is not used. Specifying the parameter with other parameters is prohibited.
Supplement	Note 1. When the input capture function is used, MTU_ACTION_CAPTURE must always be specified. Note 2. Interrupt priority needs to be specified in advance by using 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 interrupt enabling (MTU_ACTION_INTERRUPT or MTU_ACTION_CALLBACK) is not specified.	

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

capture_X.capture_edge		
Synopsis	Setting effective edge of input capture input signal of timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies an effective edge for an input capture signal on each MTU channel.	
Parameters	MTU_CAP_RISING_EDGE	Specifies counting at the rising edge
	MTU_CAP_FALLING_EDGE	Specifies counting at the falling edge
	MTU_CAP_ANY_EDGE	Specifies counting at the rising/falling edges
Supplement	None	

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

capture_X.filter_enable		
Synopsis	Setting noise filter of MTU input pin MTIOCnX (X = a, b, c, d) (n = 0 to 4, 6 to 8)	
Header	r_mtu3_if.h	
Description	This parameter specifies a digital noise filter for an input capture signal on each MTU channel.	
Parameters	true	Enables the noise filter.
	false	Disables the noise filter.
Supplement	The sampling clock of the noise filter must be specified in advance by using r_mtu3_config.h.	

6.15 R_MTU_PWM_Open Parameters

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

Table 6.11 R_MTU_PWM_Open Parameters

Parameter	Description
clock_src.source	Sets a count clock.
clock_src.clock_edge	Sets an edge of the clock
cycle_freq	Specifies the PWM cycle (Hz)
clear_src	Specifies a source for clearing the counter
pwm_mode	Set PWM mode
pwm_X.duty (X = a, b, c, d)	Specifies duty ratio for the timer general register TGRX (X = a, b, c, d)
pwm_X.actions (X = a, b, c, d)	Specifies operations of the timer general register TGRX (X = a, b, c, d)
pwm_X.outputs (X = a, b, c, d)	Specifies pin level of the PWM output for the timer general register TGRX (X = a, b, c, d)

6.15.1 clock_src.source

clock_src.source

Synopsis: Setting count clock

Header: r_mtu3_if.h

Description: This parameter specifies a count clock on each MTU channel.
When an external clock is selected, the I/O port of the target pin and multi-function pin controller (MPC) must be set in advance.
When an internal clock (PCLKC) is selected, a division ratio that is more suitable than the specified PWM cycle (cycle_freq) is automatically selected

Parameters:

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

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

6.15.2 clock_src.clock_edge

clock_src.clock_edge

Synopsis: Setting clock edge

Header: r_mtu3_if.h

Description: This parameter specifies an edge for counting an input clock on each MTU channel.

Parameters:

MTU_CLK_RISING_EDGE	Specifies counting at the rising edge
MTU_CLK_FALLING_EDGE	Specifies counting at the falling edge
MTU_CLK_ANY_EDGE	Specifies counting at the rising/falling edges

Supplement: When the clock source to drive counting of the specified PWM cycle (cycle_freq) is PCLKC/1, specification of an edge is ignored and handling is as the initial value (rising edge).

6.15.3 cycle_freq

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

6.15.4 clear_src

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

6.15.5 pwm_mode

pwm_mode		
Synopsis	Setting PWM mode	
Header	r_mtu3_if.h	
Description	This parameter specifies PWM mode on each MTU channel.	
Parameters	MTU_PWM_MODE_1*1	Specifies PWM mode 1
	MTU_PWM_MODE_2*2	Specifies PWM mode 2
Supplement	Note 1. Only available for channels 0 to 4, 6, and 7 Note 2. Only available for channels 0 to 2	

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

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

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

pwm_X.actions	
Synopsis	Setting operation of timer general register TGRX (X = a, b, c, d)
Header	r_mtu3_if.h
Description	This parameter specifies TGR operation on each MTU channel. When the MTU pin is specified as an output pin, the I/O port of the target pin and multi-function pin controller (MPC) must be set in advance. Multiple operations can be set simultaneously by this parameter as follows. 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 executes the callback function specified by the user in the interrupt service processing routine. MTU_ACTION_TRIGGER_ADC*2 — Specifies trigger startup of the A/D converter upon TGRA compare match. MTU_ACTION_REPEAT*3 — Continues PWM output after the first PWM cycle has elapsed. MTU_ACTION_NONE — This parameter is specifies when TGR is not used. Setting this parameter with other parameters is prohibited.
Supplement	Note 1. Interrupt priority must be specified in advance by using r_mtu3_config.h. Note 2. Available for channel 0 to 4, 6, and 7 Note 3. When this parameter is not specified, timer count operation is stopped in the interrupt processing routine. For this reason, timer count operation always continues when interrupt is not enabled (MTU_ACTION_INTERRUPT or MTU_ACTION_CALLBACK).

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

pwm_X.outputs		
Synopsis	Setting pin level of PWM output for timer general register TGRX (X = a, b, c, d)	
Header	r_mtu3_if.h	
Description	This parameter specifies initial output level of the PWM output pin on each MTU channel and output level upon a compare match.	
Parameters	MTU_PIN_NO_OUTPUT	Disables output
	MTU_PIN_LO_GOLO	Specifies initial output as low, and low upon a compare match.
	MTU_PIN_LO_GOHI	Specifies initial output as low, and high upon a compare match.
	MTU_PIN_LO_TOGGLE	Specifies initial output as low, and toggle output upon a compare match.
	MTU_PIN_HI_GOLO	Specifies initial output as high, and low upon a compare match.
	MTU_PIN_HI_GOHI	Specifies initial output as high, and high upon a compare match.
	MTU_PIN_HI_TOGGLE	Specifies initial output as high, and toggle output upon a compare match.
Supplement	PWM mode 1, PWM output is from the MTIOCnA pin when TGRA and TGRB are in use. Furthermore, PWM output is from the MTIOCnA pin when TGRC and TGRD are in use. In PWM mode 2, PWM output is in accord with the single TGR set in the period 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. This is provided for used in evaluation and so on.	

6.16 R_MTU_Control Parameters

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

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

6.16.1 When cmd is MTU_CMD_START

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

pcmd_data

pcmd_data	
Synopsis	Specifying multiple channel numbers
Header	r_mtu3_if.h

Description	This parameter specifies multiple MTU channels for their simultaneous operation.
	
	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
Supplement	Note: When a single channel is used, set only the first channel argument of the R_MTU_Control function 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.

6.16.2 When cmd is MTU_CMD_STOP

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

pcmd_data

pcmd_data		
Synopsis	Setting multiple channel numbers	
Header	r_mtu3_if.h	

Description	This parameter specifies multiple MTU channels for their simultaneous operation.
	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
Supplement	Note:	When only a single channel is used, set only the first argument channel of the R_MTU_Control function and set the third argument pcmd_data null. When multiple channels are specified by using the third argument pcmd_data, the value in the first argument becomes invalid.

6.16.3 When cmd is MTU_CMD_SAFE_STOP

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

6.16.4 When cmd is MTU_CMD_RESTART

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

6.16.5 When cmd is MTU_CMD_SYNCHRONIZE

The first address of the mtu_group_t structure is specified to parameter (*pcmd_data).

pcmd_data

pcmd_data		
Synopsis	Setting multiple channels	
Header	r_mtu3_if.h	

Description	This parameter specifies multiple MTU channels to operate them simultaneously.
	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
Supplement	Note:	When only a single channel is used, set only the first argument channel of the R_MTU_Control function and set the third argument pcmd_data null. When multiple channels are set in the third argument pcmd_data, the value in the first argument becomes null.

6.16.6 When cmd is MTU_CMD_GET_STATUS (in timer mode)

The first address of the mtu_timer_status_t structure is specified as parameter (*pcmd_data) in timer_mode.

When the command is executed, information of the parameters is acquired and the value is returned to the specified structure variable.

Table 6.12 Parameters when cmd is MTU_CMD_GET_STATUS (in timer mode)

Parameter	Description
timer_count	Count value of the timer
timer_running	Count direction of the timer

6.16.7 When cmd is MTU_CMD_GET_STATUS (in input capture mode)

The first address of the mtu_capture_status_t structure is specified to parameter (*pcmd_data) in input capture mode.

When the command is executed, information of the parameters is acquired and the value is returned to the specified structure variable.

Table 6.13 Parameters when cmd is MTU_CMD_GET_STATUS (in input capture mode)

Parameter	Description
capt_X_count (X = a, b, c, d)	Value of input capture of TGRX (X = a, b, c, d)
timer_count	Count value of the timer

6.16.8 When cmd is MTU_CMD_SET_CAPT_EDGE

The first address of the mtu_capture_set_edge_t structure is specified as parameter (*pcmd_data) when this command is set.

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

Table 6.14 Parameters when cmd is MTU_CMD_SET_CAPT_EDGE

Parameter	Description
capture_src	Setting of 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	Setting of edge for input capture MTU_CAP_RISING_EDGE: Rising edge MTU_CAP_FALLING_EDGE: Falling edge MTU_CAP_ANY_EDGE: Rising/falling edges

7. Sample Program

The sample program is available on the Renesas Electronics website.

8. Reference Documents

User's manual: hardware:
RZ/T1 Group User's Manual: Hardware
(Download the latest version of the manual from the Renesas Electronics website.)

RZ/T1 Evaluation Board RTK7910022C00000BR User's Manual
(Download the latest version of the manual from the Renesas Electronics website.)

Technical Update / Technical News
(Download the latest version of the update or news from the Renesas Electronics website.)

User's manual: Development Environment
For IAR integrated development environment (IAR Embedded Workbench® for Arm), visit the IAR Systems website.
(Download the latest version from the IAR Systems website.)

Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry

Revision History	Application Note: Multi-function Timer Pulse Unit (MTU3a)

Rev.	Date	Description	
		Page	Summary
0.10	Mar. 27, 2015	—	First Edition issued
1.00	Apr. 10, 2015	—	Only the revision number was changed to be posted on a website.
1.10	Aug. 18, 2015	2. Operating Environment	
		6	Table 2.1 Operating Environment: Integrated Development Environment, partially amended and added
		6. Software	
		11	6.2.4 Required Memory Size: Description and reference added
		11	Table 6.2: Table title was partially amended
		12	Table 6.3 added
		12	Table 6.4 added
1.20	Dec. 04, 2015	2. Operating Environment	
		6	Table 2.1 Operating Environment: Integrated Development Environment, information partially amended
1.30	Apr. 05, 2017	2. Operating Environment	
		6	Table 2.1 Operating Environment: Integrated Development Environment, modified
		6. Software	
		—	6.2.4 Required Memory Size, deleted
1.40	Jun. 07, 2018	2. Operating Environment	
		6	Table 2.1 Operating Environment: The description on the integrated development environment, modified
		5. Hardware	
		9	Figure 5.1 Hardware configuration example: The name of module, modified
		8. Related Documents	
		74	The name of IAR Embedded Workbench, modified
1.50	May.14.2021	6. Software	
		10	Figure 6.1 Timing Diagram, modified
		12	Table 6.4 Constants for Sample Program: The setting Value of duty ratio, modified

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 claims 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, drawings, charts, programs, algorithms, and 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 reverse engineer 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, copying or reverse engineering.
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; 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.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; 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 any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using 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 dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of 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. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a 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. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using 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 comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
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 directly or indirectly controlled subsidiaries.

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

(Rev.4.0-1 November 2017)

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.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

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: +44-1628-651-700, Fax: +44-1628-651-804

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, 200333 P. R. China
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 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.0