RZ/T1 Group

TPUa/PPG Sample Program

R01AN2603EJ0140
Rev.1.40
Jun. 07, 2018

## Introduction

This application note explains the sample program of TPUa (Timer Pulse Unit) and PPG(Programmable Pulse Generator) for the RSK RZ/T1 evaluation board with the RZ/T1 group MCU mounted.

For six output pins, the sample program outputs pulse waveforms by switching output signals every 10 ms.

- Uses TPUa channel 0.
- Uses PPG unit 1 (PO23-28 pin).
- Uses the compare match interrupt function (TGI0A) for TPU0.TGRA
- Uses the overflow interrupt function (TGI0V) for TPU0.TCNT.
- Uses PCLKD/64 (PCLKD = 75 MHz).
- Uses timer operations (normal mode, TGRA compare match operation, and PPG trigger).
- FIT specification compliant API

## Target Devices

RZ/T1

When applying this application note to another microcomputer, modify the contents according to the specifications of the target microcomputer and conduct an extensive evaluation.

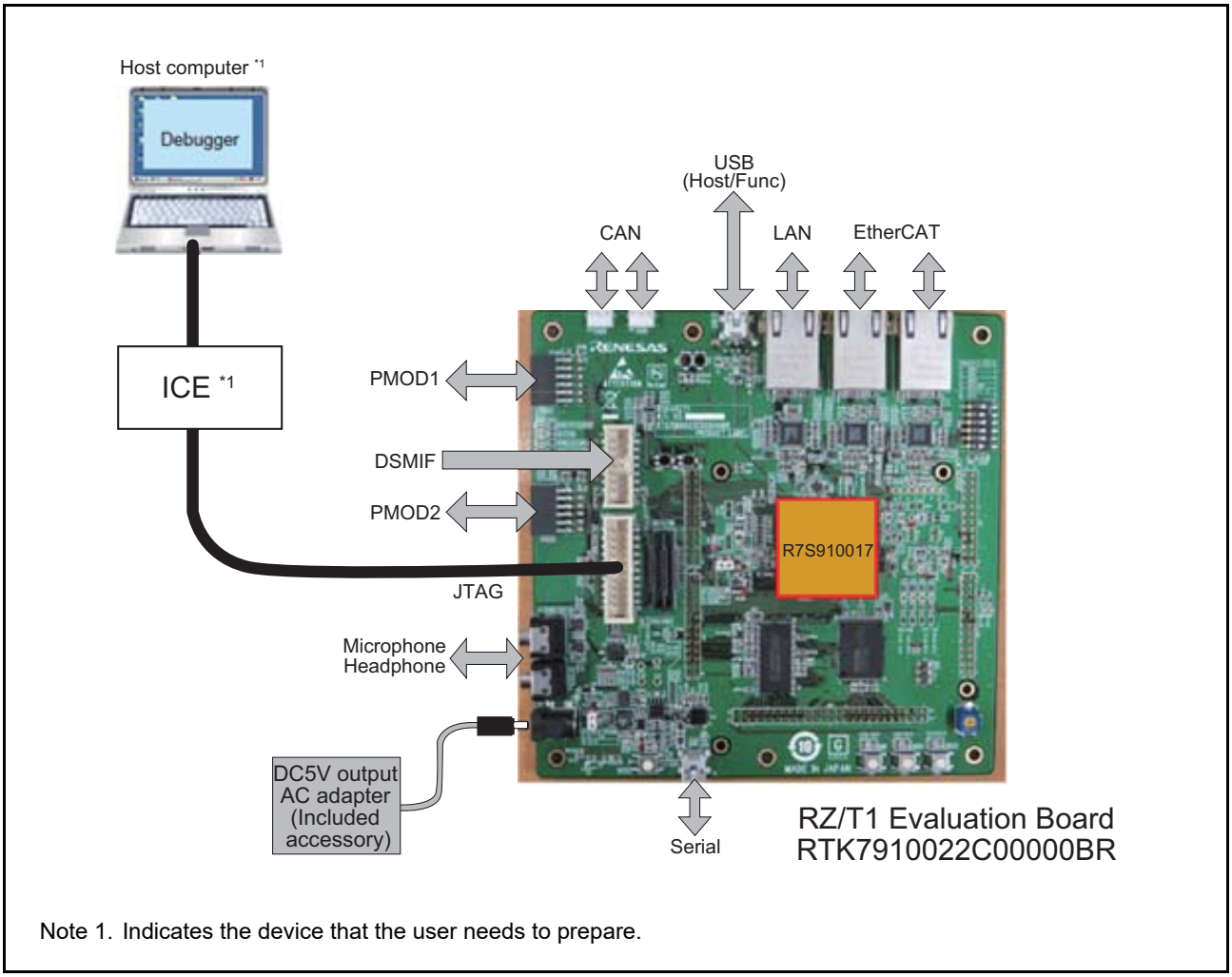
# Table of Contents

# 1. Specifications

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

**Table 1.1 Peripheral Functions and Applications**

| Peripheral Function | Application |
|---|---|
| RZ/T1 internal 16 bit timer pulse unit (TPUa) | Timer control using the compare match function (channel 0) |
| RZ/T1 internal programmable pulse generator (PPG) | Output pin control synchronized with TPUa (unit 1) |
| RZ/T1 internal I/O port<br>RZ/T1 internal multi-function pin controller (MPC) | PO23 (PS6) (Pin number: R19)<br>PO24 (PS7) (Pin number: R20)<br>PO25 (PT0) (Pin number: P19)<br>PO26 (PT1) (Pin number: P20)<br>PO27 (PT2) (Pin number: N19)<br>PO28 (PT3) (Pin number: N20) |
| RZ/T1 internal interrupt controller (ICUA) | Interrupt control (TPUa TPU0)<br>• Compare match A interrupt source (TGI0A), vector number 216<br>• Overflow interrupt source (TGI0V), vector number 220 |
| RZ/T1 internal clock generation circuit | Controls clock supply to TPUa/PPG (PCLKD: 75 MHz) |
| RZ/T1 internal power consumption reduction function | Power consumption reduction of TPUa (TPUa unit 0)<br>Power consumption reduction of PPG (PPG unit 1) |




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

**Figure 1.1 Operating Environment**

# 2. Operating Environment

The sample code covered in this application note 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<br>Embedded Workbench® for Arm Version 8.20.2<br>Manufactured by Arm<br>DS-5™ 5.26.2<br>Manufactured by RENESAS<br>e2studio 6.1.0 |
| Operating mode | SPI boot mode<br>16-bit bus boot mode |
| Board | RZ/T1 Evaluation Board<br>(RTK7910022C00000BR) |
| Device<br>(functions to be used on the board) | • NOR flash memory (connected to CS0 and CS1 spaces)<br>Manufacturer: Macronix International Co., Ltd., Model: MX29GL512FLT2I-10Q<br>• SDRAM (connected to CS2 and CS3 spaces)<br>Manufacturer: Integrated Silicon Solution Inc, Model: IS42S16320D-7TL<br>• Serial flash memory<br>Manufacturer: Macronix International Co., Ltd., Model: MX25L51245G |

# 3. Related Application Note

The application note related to this application note is listed below for reference.

- RZ/T1 Group: Application Note Initial Settings (R01AN2554EJ)

Note: For registers that are not described in this application note, use the values set in the RZ/T1 Group: Application Note Initial Settings as is.

# 4. Peripheral Functions

The basics of the operation modes, 16-bit timer pulse unit (TPUa), programmable pulse generator (PPG), I/O ports, multi-function pin controller (MPC), interrupt controller (ICUA), clock generation circuit, and the power consumption reduction function are described in RZ/T1 Group User's Manual: Hardware.

# 5. Hardware

## 5.1 Hardware Configuration Example

The following figure provides a hardware configuration example for TPUa/PPG (six pins from PT3 to PT0, PS7 and PS6 enclosed by an orange rectangular are used).

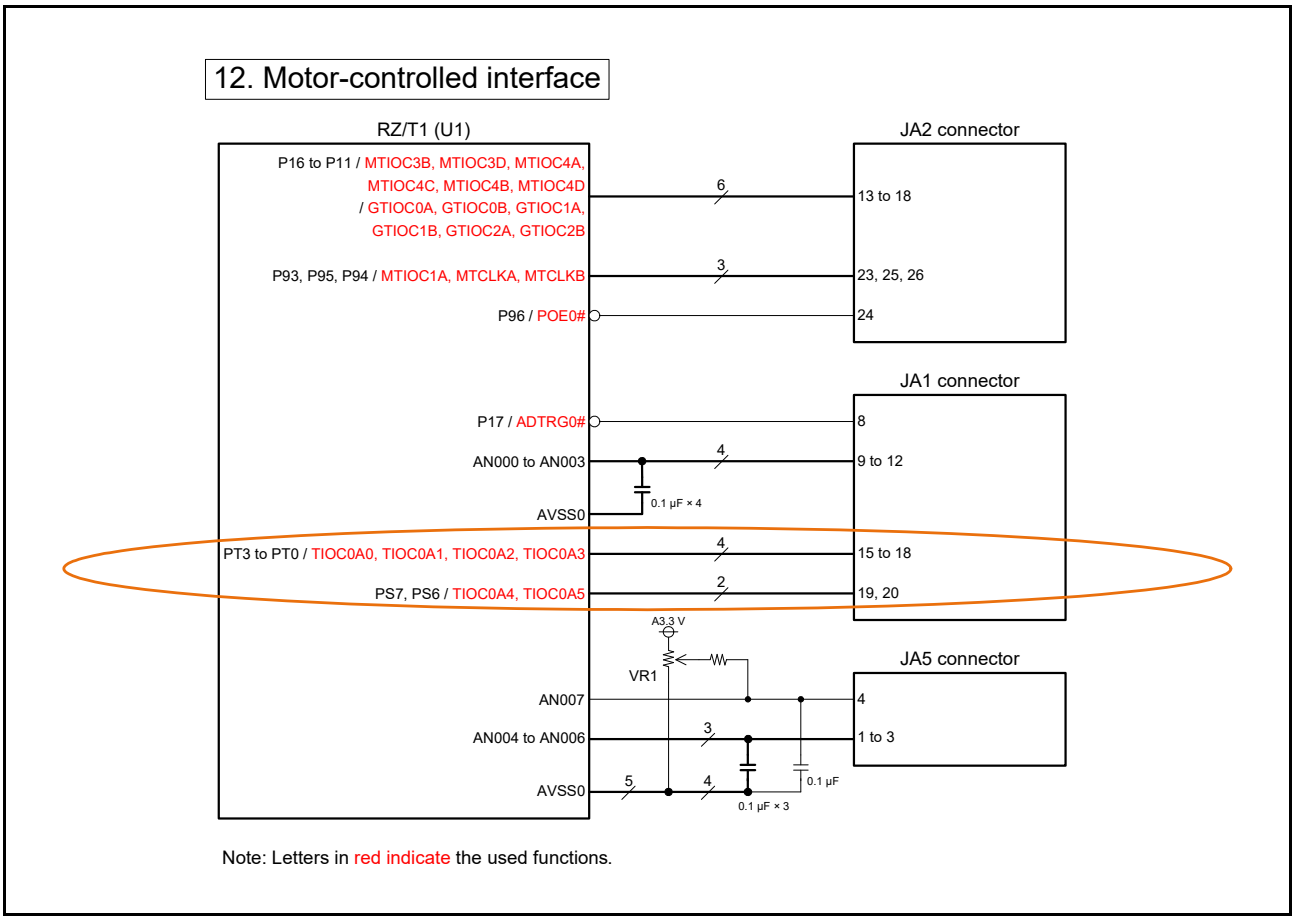



**Figure 5.1 TPUa/PPG Hardware Configuration Example**

## 5.2 Pins

Table 5.1 lists the used pins.

**Table 5.1 Used Pins and Functions**

| Pin Name | I/O | Description |
|---|---|---|
| PO23 (PS6:R19) | Output | PPG unit 1 output signal |
| PO24 (PS7:R20) | Output | PPG unit 1 output signal |
| PO25 (PT0:P19) | Output | PPG unit 1 output signal |
| PO26 (PT1:P20) | Output | PPG unit 1 output signal |
| PO27 (PT2:N19) | Output | PPG unit 1 output signal |
| PO28 (PT3:N20) | Output | PPG unit 1 output signal |

# 6. Software

## 6.1 Operation Outline

Table 6.1 Operation Outline lists the functional outlines of the TPUa and PPG sample program. Figure 6.1 shows the system block diagram.

**Table 6.1 Operation Outline (1 / 2)**

| Function | Outline |
|---|---|
| TPu settings | • Channel 0 on unit 0 (TPU0) is used<br>• Timer prescaler: PCLKD/64 (*PCLKD = 75 MHz)<br>TPU0.TCR.TPSC = 001b<br>• Input clock edge: Counted at falling edges<br>TPU0.TCR.CKEG = 00b<br>• Counter clear source: The TCNT counter is cleared during a compare match of the TGRA register<br>TPU0.TCR.CCLR = 001b<br>• Mode selection: Normal operation<br>TPU0.TMDR.MD = 0000b<br>• Compare match operation: Output disabled<br>TPU0.TIORH.IOA = 0000b |
| TPUa callback | Registers the callback function of compare matches for channel 0 on unit 0 of TPUa, and uses the call-back function as the trigger for switching pulse output waveform. |
| PPG settings | • Select channel 0 on unit 0 of TPUa as the PPG1 trigger.<br>PPG1.PTRSLR.PTRSL = 1b<br>PPG1.PCR.G0CMS = 00b (Group 0: Compare matches of TPU0)<br>PPG1.PCR.G1CMS = 00b (Group 1: Compare matches of TPU0)<br>PPG1.PCR.G2CMS = 00b (Group 2: Compare matches of TPU0)<br>PPG1.PCR.G3CMS = 00b (Group 3: Compare matches of TPU0)<br>PPG1.PMR.G0NOV = 0b (Group 0: Normal operation)<br>PPG1.PMR.G1NOV = 0b (Group 1: Normal operation)<br>PPG1.PMR.G2NOV = 0b (Group 2: Normal operation)<br>PPG1.PMR.G3NOV = 0b (Group 3: Normal operation)<br>PPG1.PMR.G0INV = 1b (Group 0: Direct output)<br>PPG1.PMR.G1INV = 1b (Group 1: Direct output)<br>PPG1.PMR.G2INV = 1b (Group 2: Direct output)<br>PPG1.PMR.G3INV = 1b (Group 3: Direct output) |
| Pins to be used | The PPG function uses pins from PT0 to 3, PS6, and PS7.<br>• PT0<br>⇒ PORTT.PMR.B0 = 1 (Used as a peripheral function)<br>• PT1<br>⇒ PORTT.PMR.B1 = 1 (Used as a peripheral function)<br>• PT2<br>⇒ PORTT.PMR.B2 = 1 (Used as a peripheral function)<br>• PT3<br>⇒ PORTT.PMR.B3 = 1 (Used as a peripheral function)<br>• PS6<br>⇒ PORTS.PMR.B6 = 1 (Used as a peripheral function)<br>• PS7<br>⇒ PORTS.PMR.B7 = 1 (Used as a peripheral function)<br><br>Uses pins PO2 to 28.<br>• PO23 (PS6/TIOCA5/TIOCB5) (Pin number: R19)<br>→ PS6PFS = 0x06<br>• PO24 (PS7/TIOCA4/TIOCB4) (Pin number: R20)<br>→ PS7PFS = 0x06<br>• PO25 (PT0/TIOCA3/TIOCB3) (Pin number: P19)<br>→ PT0PFS = 0x06<br>• PO26 (PT1/TIOCA2/TIOCB2) (Pin number: P20)<br>→ PT1PFS = 0x06<br>• PO27 (PT2/TIOCA1/TIOCB1) (Pin number: N19)<br>→ PT2PFS = 0x06<br>• PO28 (PT3/TIOCA0/TIOCB0) (Pin number: N20)<br>→ PT3PFS = 0x06 |

**Table 6.1 Operation Outline (2 / 2)**

| Function | Outline |
|---|---|
| Processing overview | • Outputs a pulse waveform by changing the level of PPG output from the PO23, PO24, PO25, PO26, PO27, and PO28 pins at a specified compare match interval for TPUa channel 0 (Sets the output value from a pin via the R_PPG_Control API). |

Figure 6.1 shows the system block diagram.

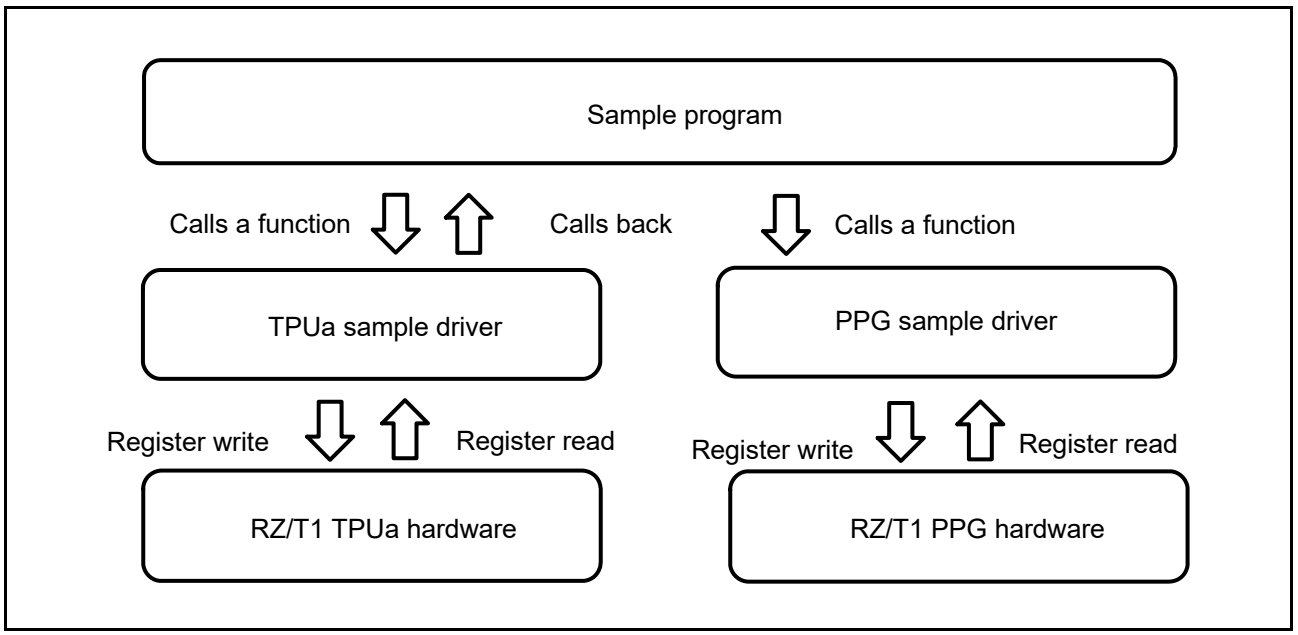



**Figure 6.1 System Block Diagram**

### 6.1.1 Project Settings

For information about project settings used for the development environment, EWARM, see the RZ/T1 Group: Application Note Initial Settings.

### 6.1.2 Preparation for Use

No preparation is required to execute this sample program.

## 6.2 Memory Map

For information about RZ/T1 group address spaces and memory mapping of the RZ/T1 evaluation board, see the RZ/T1 Group: Application Note Initial Settings.

### 6.2.1 Section Assignment

For information about the sections to be used by the sample program, the initial section assignment (load view) of the sample program, the section assignment (execution view) after the scatter loading function is used, refer to the RZ/T1 Group: Application Note Initial Settings.

### 6.2.2 MPU Settings

For information about the MPU settings, refer to the RZ/T1 Group: Application Note Initial Settings.

### 6.2.3 Exception Handling Vector Table

For information about the exception handling vector table, see the RZ/T1 Group: Application Note Initial Settings.

## 6.3 Interrupts

Table 6.2 lists interrupts for the sample program.

**Table 6.2 Interrupts for the Sample Program**

| Interrupt (Source ID) | Priority | Process Outline |
|---|---|---|
| Input capture/compare match interrupt of TPU0.TGRA (TGI0A) | 3 | Timer compare match processing |
| Overflow interrupt of TPU0.TCNT (TGI0V) | 3 | Overflow processing |

## 6.4 Fixed-Width Integer Types

Table 6.3 lists the fixed-width integers for the sample program.

**Table 6.3 Fixed-width Integers for the Sample Program**

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

## 6.5 Constants/Error Codes

Table 6.4 lists the constants to be used in the TPUa and PPG sample programs, Table 6.5 and Table 6.6 list the constants to be used in the TPUa sample driver, and Table 6.7 and Table 6.8 list the constants to be used in the PPG sample driver.

**Table 6.4 Constants for the TPUa and PPG Sample Program**

| Constant Name | Setting Value | Description |
|---|---|---|
| TPUA_TIME_10MS | 0x2DC6 | 10 ms timer count value<br>(Prescaler setting of the timer: PCLKD/64) |
| PPG_OUTPUTDATA_TBL_WORDSIZE | size of (ppg_outputdata) / 2 | Word size value of the ppg_outputdata table data |

**Table 6.5 Constants for the TPUa Sample Driver (Global) (1 / 3)**

| Constant Name | Setting Value | Description |
|---|---|---|
| TPUA_TCR_REG | 0x00000001 | Information indicating the TCR register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TMDR_REG | 0x00000002 | Information indicating the TMDR register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TIORH_REG | 0x00000004 | Information indicating the TIORH register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TIOR_REG | 0x00000008 | Information indicating the TIOR register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TIORL_REG | 0x00000010 | Information indicating the TIORL register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TIER_REG | 0x00000020 | Information indicating the TIER register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TSR_REG | 0x00000040 | Information indicating the TSR register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TCNT_REG | 0x00000080 | Information indicating the TCNT register that is used as parameter of the argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TGRA_REG | 0x00000100 | Information indicating the TGRA register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TGRB_REG | 0x00000200 | Information indicating the TGRB register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TGRC_REG | 0x00000400 | Information indicating the TGRC register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TGRD_REG | 0x00000800 | Information indicating the TGRD register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TSTRA_REG | 0x00001000 | Information indicating the TSTRA register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TSTRB_REG | 0x00002000 | Information indicating the TSTRB register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TSYRA_REG | 0x00004000 | Information indicating the TSYRA register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_TSYRB_REG | 0x00008000 | Information indicating the TSYRB register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_NFCR_REG | 0x00010000 | Information indicating the NFCR register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_PWMFBSLR_REG | 0x00020000 | Information indicating the PWMFBSLR register that is used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |
| TPUA_ALL_REG | 0x0003FFFF | Information indicating all registers that are used as a parameter of an argument for R_TPUA_Control (see 6.11.5, 6.11.6) |

**Table 6.5 Constants for the TPUa Sample Driver (Global) (2 / 3)**

| Constant Name | Setting Value | Description |
|---|---|---|
| TPUA_CFG_PARAM_CHECKING_ENABLE | 1 | Selects whether the API argument parameter check is enabled or disabled.<br>0: Disabled, 1: Enabled |
| TPUA_IR_PRIORITY_TPU0_A | 3 | Interrupt priority for TPU0A |
| TPUA_IR_PRIORITY_TPU0_B | 3 | Interrupt priority for TPU0B |
| TPUA_IR_PRIORITY_TPU0_C | 3 | Interrupt priority for TPU0C |
| TPUA_IR_PRIORITY_TPU0_D | 3 | Interrupt priority for TPU0D |
| TPUA_IR_PRIORITY_TPU0_V | 3 | Interrupt priority for TPU0V |
| TPUA_IR_PRIORITY_TPU1_A | 3 | Interrupt priority for TPU1A |
| TPUA_IR_PRIORITY_TPU1_B | 3 | Interrupt priority for TPU1B |
| TPUA_IR_PRIORITY_TPU1_V | 3 | Interrupt priority for TPU1V |
| TPUA_IR_PRIORITY_TPU1_U | 3 | Interrupt priority for TPU1U |
| TPUA_IR_PRIORITY_TPU2_A | 3 | Interrupt priority for TPU2A |
| TPUA_IR_PRIORITY_TPU2_B | 3 | Interrupt priority for TPU2B |
| TPUA_IR_PRIORITY_TPU2_V | 3 | Interrupt priority for TPU2V |
| TPUA_IR_PRIORITY_TPU2_U | 3 | Interrupt priority for TPU2U |
| TPUA_IR_PRIORITY_TPU3_A | 3 | Interrupt priority for TPU3A |
| TPUA_IR_PRIORITY_TPU3_B | 3 | Interrupt priority for TPU3B |
| TPUA_IR_PRIORITY_TPU3_C | 3 | Interrupt priority for TPU3C |
| TPUA_IR_PRIORITY_TPU3_D | 3 | Interrupt priority for TPU3D |
| TPUA_IR_PRIORITY_TPU3_V | 3 | Interrupt priority for TPU3V |
| TPUA_IR_PRIORITY_TPU4_A | 3 | Interrupt priority for TPU4A |
| TPUA_IR_PRIORITY_TPU4_B | 3 | Interrupt priority for TPU4B |
| TPUA_IR_PRIORITY_TPU4_V | 3 | Interrupt priority for TPU4V |
| TPUA_IR_PRIORITY_TPU4_U | 3 | Interrupt priority for TPU4U |
| TPUA_IR_PRIORITY_TPU5_A | 3 | Interrupt priority for TPU5A |
| TPUA_IR_PRIORITY_TPU5_B | 3 | Interrupt priority for TPU5B |
| TPUA_IR_PRIORITY_TPU5_V | 3 | Interrupt priority for TPU5V |
| TPUA_IR_PRIORITY_TPU5_U | 3 | Interrupt priority for TPU5U |
| TPUA_IR_PRIORITY_TPU6_A | 3 | Interrupt priority for TPU6A |
| TPUA_IR_PRIORITY_TPU6_B | 3 | Interrupt priority for TPU6B |
| TPUA_IR_PRIORITY_TPU6_C | 3 | Interrupt priority for TPU6C |
| TPUA_IR_PRIORITY_TPU6_D | 3 | Interrupt priority for TPU6D |
| TPUA_IR_PRIORITY_TPU6_V | 3 | Interrupt priority for TPU6V |
| TPUA_IR_PRIORITY_TPU7_A | 3 | Interrupt priority for TPU7A |
| TPUA_IR_PRIORITY_TPU7_B | 3 | Interrupt priority for TPU7B |
| TPUA_IR_PRIORITY_TPU7_V | 3 | Interrupt priority for TPU7V |
| TPUA_IR_PRIORITY_TPU7_U | 3 | Interrupt priority for TPU7U |
| TPUA_IR_PRIORITY_TPU8_A | 3 | Interrupt priority for TPU8A |
| TPUA_IR_PRIORITY_TPU8_B | 3 | Interrupt priority for TPU8B |
| TPUA_IR_PRIORITY_TPU8_V | 3 | Interrupt priority for TPU8V |
| TPUA_IR_PRIORITY_TPU8_U | 3 | Interrupt priority for TPU8U |
| TPUA_IR_PRIORITY_TPU9_A | 3 | Interrupt priority for TPU9A |
| TPUA_IR_PRIORITY_TPU9_B | 3 | Interrupt priority for TPU9B |
| TPUA_IR_PRIORITY_TPU9_C | 3 | Interrupt priority for TPU9C |
| TPUA_IR_PRIORITY_TPU9_D | 3 | Interrupt priority for TPU9D |

**Table 6.5 Constants for the TPUa Sample Driver (Global) (3 / 3)**

| Constant Name | Setting Value | Description |
|---|---|---|
| TPUA_IR_PRIORITY_TPU9_V | 3 | Interrupt priority for TPU9V |
| TPUA_IR_PRIORITY_TPU10_A | 3 | Interrupt priority for TPU10A |
| TPUA_IR_PRIORITY_TPU10_B | 3 | Interrupt priority for TPU10B |
| TPUA_IR_PRIORITY_TPU10_V | 3 | Interrupt priority for TPU10V |
| TPUA_IR_PRIORITY_TPU10_U | 3 | Interrupt priority for TPU10U |
| TPUA_IR_PRIORITY_TPU11_A | 3 | Interrupt priority for TPU11A |
| TPUA_IR_PRIORITY_TPU11_B | 3 | Interrupt priority for TPU11B |
| TPUA_IR_PRIORITY_TPU11_V | 3 | Interrupt priority for TPU11V |
| TPUA_IR_PRIORITY_TPU11_U | 3 | Interrupt priority for TPU11U |

**Table 6.6 Constants for the TPUa Sample Driver (Local)**

| Constant Name | Setting Value | Description |
|---|---|---|
| TPUA_NUM_CHANNELS | 12 | Total number of TPUa channels |
| TPUA_HVA_WRITE_DATA | 0x00000000u | Data to be written to HVA |
| TPUA_CHANNEL0 | 0 | Channel number 0 |
| TPUA_CHANNEL1 | 1 | Channel number 1 |
| TPUA_CHANNEL2 | 2 | Channel number 2 |
| TPUA_CHANNEL3 | 3 | Channel number 3 |
| TPUA_CHANNEL4 | 4 | Channel number 4 |
| TPUA_CHANNEL5 | 5 | Channel number 5 |
| TPUA_CHANNEL6 | 6 | Channel number 6 |
| TPUA_CHANNEL7 | 7 | Channel number 7 |
| TPUA_CHANNEL8 | 8 | Channel number 8 |
| TPUA_CHANNEL9 | 9 | Channel number 9 |
| TPUA_CHANNEL10 | 10 | Channel number 10 |
| TPUA_CHANNEL11 | 11 | Channel number 11 |
| TPUA_TIER_BIT_INVALID | 0x00 | Invalid bit location |
| TPUA_TIER_BIT_TGIEA | 0x01 | TGIEA bit location of the TIER register |
| TPUA_TIER_BIT_TGIEB | 0x02 | TGIEB bit location of the TIER register |
| TPUA_TIER_BIT_TGIEC | 0x04 | TGIEC bit location of the TIER register |
| TPUA_TIER_BIT_TGIED | 0x08 | TGIED bit location of the TIER register |
| TPUA_TIER_BIT_TGIEV | 0x10 | TCIEV bit location of the TIER register |
| TPUA_TIER_BIT_TGIEU | 0x20 | TCIEU bit location of the TIER register |
| BITSHIFT_16 | 16u | Number of 16-bit shifts |
| NULL | 0 | NULL |
| TPUA_RZT1_VERSION_MAJOR | 1 | Major Version |
| TPUA_RZT1_VERSION_MINOR | 0 | Minor Version |

**Table 6.7 Constants for the PPG Sample Driver (Global)**

| Constant Name | Setting Value | Description |
|---|---|---|
| PPG_PTRSLR_REG | 0x00000001 | Information indicating the PTRSLR register that is used as a parameter of an argument for R_TPUA_Control (see 6.12.1, 6.12.2) |
| PPG_NDER_REG | 0x00000002 | Information indicating the NDERH/L that is used as a parameter of an argument for R_TPUA_Control (see 6.12.1, 6.12.2) |
| PPG_PODR_REG | 0x00000004 | Information indicating the PODRH/L register that is used as a parameter of an argument for R_TPUA_Control (see 6.12.1, 6.12.2) |
| PPG_NDR_REG | 0x00000008 | Information indicating the NDRH/L register that is used as a parameter of an argument for R_TPUA_Control (see 6.12.1, 6.12.2) |
| PPG_PCR_REG | 0x00000010 | Information indicating the PCR register that is used as a parameter of an argument for R_TPUA_Control (see 6.12.1, 6.12.2) |
| PPG_PMR_REG | 0x00000020 | Information indicating the PMR register that is used as a parameter of an argument for R_TPUA_Control (see 6.12.1, 6.12.2) |
| PPG_ALL_REG | 0x0000003F | Information indicating all registers that are used as a parameter of an argument for R_TPUA_Control (see 6.12.1, 6.12.2) |
| PPG_CFG_PARAM_CHECKING_ENABLE | 1 | Selects whether the parameter check processing for the API argument is enabled or disabled.<br>0: Disabled, 1: Enabled |

**Table 6.8 Constants for the PPG Sample Driver (Local)**

| Constant Name | Setting Value | Description |
|---|---|---|
| PPG_NUM_UNIT | 2u | Total number of PPG units |
| BITSHIFT_8 | 8u | Number of 8-bit shifts |
| BITSHIFT_16 | 16u | Number of 16-bit shifts |
| NULL | 0 | NULL |
| PPG_RZT1_VERSION_MAJOR | 1 | Major Version |
| PPG_RZT1_VERSION_MINOR | 0 | Minor Version |

**Table 6.9 TPUa Sample Driver (Error Codes)**

| Enumerated Name | Setting Value | Description |
|---|---|---|
| TPUA_SUCCESS | 0 | Normal termination |
| TPUA_ERR_BAD_CHAN | 1 | Channel number error |
| TPUA_ERR_CH_NOT_OPENED | 2 | Open error (Not opened) |
| TPUA_ERR_CH_NOT_CLOSED | 3 | Close error (Not closed) |
| TPUA_ERR_UNKNOWN_CMD | 4 | Command error |
| TPUA_ERR_NULL_PTR | 5 | Null pointer |

**Table 6.10 PPG Sample Driver (Error Codes)**

| Enumerated Name | Setting Value | Description |
|---|---|---|
| PPG_SUCCESS | 0 | Normal termination |
| PPG_ERR_BAD_UNIT | 1 | Unit number error |
| PPG_ERR_UN_NOT_OPENED | 2 | Open error (Not opened) |
| PPG_ERR_UN_NOT_CLOSED | 3 | Close error (Not closed) |
| PPG_ERR_UNKNOWN_CMD | 4 | Command error |
| PPG_ERR_NULL_PTR | 5 | Null pointer |

## 6.6 Structures/Unions/Enumerated Types

The following tables list the structures, unions, and enumerated types that are used for the TPUa/PPG sample driver.

**Table 6.11 Structures/Unions (TPUa Sample Driver)**

| Structure/Union Definition | Overview | Definition File |
|---|---|---|
| tpua_callback_t | TPUa callback function information | r_tpua_rzt1_if.h |
| tpua_handle_t | TPUa handle information (channels, unit information, etc.) | |
| tpua_reg_t | TPUa read/write register information | |
| tpua_err_t | Error information on function's return values | |
| tpua_cmd_t | Command codes of the second argument for the R_TPUA_Control function | |

**Table 6.12 Structures/Unions (PPG Sample Driver)**

| Structure/Union Definition | Overview | Definition File |
|---|---|---|
| ppg_handle_t | PPG handle information (channels, unit information, etc.) | r_ppg_rzt1_if.h |
| ppg_reg_t | PPG read/write register information | |
| ppg_err_t | Error information on function's return values | |
| ppg_cmd_t | Command codes of the second argument for the R_PPG_Control function | |

**Table 6.13 TPUa Structure/Union Members (1 / 2)**

| Structure/Union Definition | Member | Description |
|---|---|---|
| tpua_callback_t | void (*pintr_a)(void) | Callback function for the TPU*1A interrupt |
| | void (*pintr_b)(void) | Callback function for the TPU*1B interrupt |
| | void (*pintr_c)(void) | Callback function for the TPU*2C interrupt |
| | void (*pintr_d)(void) | Callback function for the TPU*2D interrupt |
| | void (*pintr_v)(void) | Callback function for the TPU*1V interrupt |
| | void (*pintr_u)(void) | Callback function for the TPU*3U interrupt |
| tpua_handle_t | uint8_t channel | Channel information for TPUa (0 to 11) |
| | uint8_t unit | Unit information for TPUa (0 to 3) |
| | bool tpua_channel_opened | Open status of a channel<br>true: Opened<br>false: Not opened |
| | tpua_callback_t tpua_callback | See the description for tpua_callback_t. |

**Table 6.13 TPUa Structure/Union Members (2 / 2)**

| Structure/Union Definition | Member | Description |
|---|---|---|
| tpua_reg_t | uint32_t reg_flag | Information on the parameter which is an argument for the R_TPUA_Control function (see 6.11.5, 6.11.6, Table 6.5). |
| | uint8_t tcr_reg | TCR register information*4 |
| | uint8_t tmdr_reg | TMDR register information*4 |
| | uint8_t tiorh_reg | TIORH register information*4 |
| | uint8_t tior_reg | TIOR register information*4 |
| | uint8_t tiorl_reg | TIORL register information*4 |
| | uint8_t tier_reg | TIER register information*4 |
| | uint8_t tsr_reg | TSR register information*4 |
| | uint16_t tcnt_reg | TCNT register information*4 |
| | uint16_t tgra_reg | TGRA register information*4 |
| | uint16_t tgrb_reg | TGRB register information*4 |
| | uint16_t tgrc_reg | TGRC register information*4 |
| | uint16_t tgrd_reg | TGRD register information*4 |
| tpua_reg_t | uint8_t tstra_reg | TSTRA register information*4 |
| | uint8_t tstrb_reg | TSTRB register information*4 |
| | uint8_t tsyra_reg | TSYRA register information*4 |
| | uint8_t tsyrb_reg | TSYRB register information*4 |
| | uint8_t nfcr_reg | NFCL register information*4 |
| | uint32_t pwmfbslr_reg | PWMFBSLR register information*4 |

Note 1. 0 to 11
Note 2. 0, 3, 6, 9
Note 3. 1, 2, 4, 5, 7, 8, 10, 11
Note 4. For details about registers, see the RZ/T1 Group User's Manual: Hardware (R01UH0483EJ).

**Table 6.14 PPG Structure/Union Members**

| Structure/Union Definition | Member | Description |
|---|---|---|
| ppg_handle_t | uint8_t unit | Unit |
| | bool ppg_unit_opened | Open status of a unit<br>true: Opened<br>false: Not opened |
| ppg_reg_t | uint32_t reg_flag | Information on a parameter which is an argument of the R_PPG_Control function (see 6.12.1, 6.12.2, Table 6.7). |
| | uint8_t ptrslr_reg | PTRSLR register information*1 |
| | uint16_t nder_reg | NDERH/L register information*1 |
| | uint16_t podr_reg | PODRH/L register information*1 |
| | uint16_t ndr_reg | NDRH/L register information*1 |
| | uint8_t pcr_reg | PCR register information*1 |
| | uint8_t pmr_reg | PMR register information*1 |

Note 1. For details about registers, see the RZ/T1 Group User's Manual: Hardware (R01UH0483EJ).

**Table 6.15 TPUa Enumerated Types**

| Enumerated Type Definition | List | Description |
|---|---|---|
| tpua_err_t | | TPUa API execution result |
| | TPUA_SUCCESS | 0: Normal termination |
| | TPUA_ERR_BAD_CHAN | 1: Channel number error |
| | TPUA_ERR_CH_NOT_OPENED | 2: Open error (Not opened) |
| | TPUA_ERR_CH_NOT_CLOSED | 3: Close error (Not closed) |
| | TPUA_ERR_UNKNOWN_CMD | 4: Command error |
| | TPUA_ERR_NULL_PTR | 5: Null pointer |
| tpua_cmd_t | | TPUa API command |
| | TPUA_CMD_TIMER_START | 0: Timer start |
| | TPUA_CMD_TIMER_STOP | 1: Timer stop |
| | TPUA_CMD_TIMER_SYNC | 2: Timer synchronization |
| | TPUA_CMD_TIMER_ASYNC | 3: Time asynchronous |
| | TPUA_CMD_REG_READ | 4: Register read |
| | TPUA_CMD_REG_WRITE | 5: Register write |
| | TPUA_CMD_INTR_A_ENABLE | 6: TGRA interrupt enabled |
| | TPUA_CMD_INTR_A_DISABLE | 7: TGRA interrupt disabled |
| | TPUA_CMD_INTR_B_ENABLE | 8: TGRB interrupt enabled |
| | TPUA_CMD_INTR_B_DISABLE | 9: TGRB interrupt disabled |
| | TPUA_CMD_INTR_C_ENABLE | 10: TGRC interrupt enabled |
| | TPUA_CMD_INTR_C_DISABLE | 11: TGRC interrupt disabled |
| | TPUA_CMD_INTR_D_ENABLE | 12: TGRD interrupt enabled |
| | TPUA_CMD_INTR_D_DISABLE | 13: TGRD interrupt disabled |
| | TPUA_CMD_INTR_V_ENABLE | 14: TGRV interrupt enabled |
| | TPUA_CMD_INTR_V_DISABLE | 15: TGRV interrupt disabled |
| | TPUA_CMD_INTR_U_ENABLE | 16: TGRU interrupt enabled |
| | TPUA_CMD_INTR_U_DISABLE | 17: TGRU interrupt disabled |

**Table 6.16 PPG Enumerated Types**

| Enumerated Type Definition | List | Description |
|---|---|---|
| ppg_err_t | | PPG API execution result |
| | PPG_SUCCESS | 0: Normal termination |
| | PPG_ERR_BAD_UNIT | 1: Unit number error |
| | PPG_ERR_UN_NOT_OPENED | 2: Open error (Not opened) |
| | PPG_ERR_UN_NOT_CLOSED | 3: Close error (Not closed) |
| | PPG_ERR_UNKNOWN_CMD | 4: Command error |
| | PPG_ERR_NULL_PTR | 5: Null pointer |
| ppg_cmd_t | | PPG API command |
| | PPG_CMD_REG_READ | 0: Register read |
| | PPG_CMD_REG_WRITE | 1: Register write |

## 6.7 Global Variables

Table 6.17 lists static/const type variables for the TPUa/PPG sample program and drivers.

**Table 6.17 Global Variables**

| Type | Variable Name | Description | Function to be Used |
|---|---|---|---|
| static tpua_handle_t | gb_tpua_handles[12] | Handle of the TPUa channel.<br>Reserves 12 handles for the 12 channels as an array because one handle is needed for one channel of TPUa. | R_TPUA_Open<br>R_TPUA_Control<br>R_TPUA_Close |
| static void (* const tpua_reg_read_tbl[TPUA_NUM_CHANNELS])(const uint8_t unit, tpua_reg_t * const prddat) | tpua_reg_read_tbl | Register read function table for each channel of TPUa.<br>Table where the register read functions for TPU0/6, 1/7, 2/8, 3/9, 4/10, and 5/11 are listed. | R_TPUA_Control |
| static void (* const tpua_reg_write_tbl[TPUA_NUM_CHANNELS])(const uint8_t unit, tpua_reg_t * const pwrdat) | tpua_reg_write_tbl | Register write function table for each channel of TPUa.<br>Table where the register write functions for TPU0/6, 1/7, 2/8, 3/9,4/10, and 5/11 are listed. | R_TPUA_Control |
| static const tpua_channel_adr_t | tpua_channel_adr_tbl | Register address table for each channel of TPUa.<br>Table where register addresses for TPU0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and 11 are listed. | R_TPUA_Control |
| static ppg_handle_t | gb_ppg_handles[2] | Handle of the PPG unit.<br>Reserves two handles for two units as an array because one handle is required for one unit of PPG. | R_PPG_Open<br>R_PPG_Control<br>R_PPG_Close |
| ppg_unit_adr_t | ppg_unit_adr_tbl | Register address table for each unit of PPG.<br>Table where register addresses of PPG0 and 1 are listed. | R_PPG_Control |
| volatile static bool | tpua_intr_a_flag | Compare match interrupt occurrence flag | main<br>tpu0_a_cmpmatch_callback |
| volatile static bool | tpua_intr_v_flag | Overflow interrupt occurrence flag | main<br>tpu0_v_callback |
| static bool | tpua_main_exit_flag | Main exit flag | main |
| static const uint16_t | ppg_outputdata[] | Table data of the PPG output pulse waveform.<br>Regards levels of one pin as 1-bit data, and assumes levels to be output to the PO31-16 pin as 16-bit data, and then lists the data in the following table in chronological order:<br><br>const uint16_t ppg_outputdata[] =<br>{<br>0x0380,<br>0x0700,<br>0x0E00,<br>0x1C00,<br>0x1880,<br>0x1180<br>} | |

## 6.8 Functions

Table 6.18 lists the functions to be used.

**Table 6.18 Functions**

| Function Name | Page Number |
|---|---|
| main | 23 |
| tpu0_a_cmpmatch_callback | 25 |
| tpu0_v_callback | 25 |
| R_TPUA_Open | 25 |
| R_TPUA_Control | 27 |
| R_TPUA_Close | 28 |
| R_TPUA_GetVersion | 28 |
| R_PPG_Open | 29 |
| R_PPG_Control | 30 |
| R_PPG_Close | 31 |
| R_PPG_GetVersion | 31 |

## 6.9 Specifications of Functions

The following tables list function specifications of the sample codes.

### 6.9.1 main

| main | |
|---|---|
| Synopsis | Main processing of a sample program |
| Header | – |
| Declaration | void main(void) |
| Description | By using the Open/Close/Control function for TPUa or PPG, outputs the pulse waveform pattern from an output pin.<br>Major processing<br>• Output pin settings (PO23, 24, 25, 26, 27, and 28)<br>- Settings to use pins PT0 to PT3, PS6, and PS7 as the PPG function.<br>PORTT.PMR.B0 = 1 (Use the PT0 pin as a peripheral function)<br>PORTT.PMR.B1 = 1 (Use the PT1 pin as a peripheral function)<br>PORTT.PMR.B2 = 1 (Use the PT2 pin as a peripheral function)<br>PORTT.PMR.B3 = 1 (Use the PT3 pin as a peripheral function)<br>PORTS.PMR.B6 = 1 (Use the PS6 pin as a peripheral function)<br>PORTS.PMR.B7 = 1 (Use the PS7 pin as a peripheral function)<br>- Settings for assigning pins PO23 to 28 to pins PT0 to 3, PS6, and PS7.<br>PS6PFS = 0x06 (Assign PO23 to the PS6 pin)<br>PS7PFS = 0x06 (Assign PO24 to the PS7 pin)<br>PT0PFS = 0x06 (Assign PO25 to the PT0 pin)<br>PT1PFS = 0x06 (Assign PO26 to the PT1 pin)<br>PT2PFS = 0x06 (Assign PO27 to the PT2 pin)<br>PT3PFS = 0x06 (Assign PO28 to the PT3 pin)<br>• TPUa channel 0 open<br>Callback registration for TGRA compare match interrupts and overflow interrupts<br>• TPUa channel 0 settings<br>- Channel 0 selection: 0 channel<br>- Prescaler setting: PCLKD/64 (*PCLKD = 75 MHz)<br>- Input clock edge setting: Falling edge<br>- Counter clear source setting: The TCNT counter is cleared during a compare match of the TGRA register.<br>- Mode setting: Normal operation<br>- Compare match/input capture operation settings: Compare match output is disabled.<br>- Compare match timer is set to 10 ms. |

- PPG unit 1 open
  - PPG unit 1 settings
  - Select channel 0 of TPUa unit 0 for a trigger of PPG1.
    PPG1.PTRSLR.PTRSL = 1b (Trigger of PPG1 is TPU0-3)
    PPG1.PCR.G0CMS = 00b (Group 0: Compare match of TPU0)
    PPG1.PCR.G1CMS = 00b (Group 1: Compare match of TPU0)
    PPG1.PCR.G2CMS = 00b (Group 2: Compare match of TPU0)
    PPG1.PCR.G3CMS = 00b (Group 3: Compare match of TPU0)
    PPG1.PMR.G0NOV = 0b (Group 0: Normal operation)
    PPG1.PMR.G1NOV = 0b (Group 1: Normal operation)
    PPG1.PMR.G2NOV = 0b (Group 2: Normal operation)
    PPG1.PMR.G3NOV = 0b (Group 3: Normal operation)
    PPG1.PMR.G0INV = 1b (Group 0: Direct output)
    PPG1.PMR.G1INV = 1b (Group 1: Normal operation)
    PPG1.PMR.G2INV = 1b (Group 2: Normal operation)
    PPG1.PMR.G3INV = 1b (Group 3: Normal operation)
  - Settings for using PO23-28 pins
- By using the TPUa channel 0 compare match A interrupt (callback) as a trigger, switches the output pattern table (Table 6.17) for every 10 ms-interrupt interval. By doing so, the pulse waveform shown in Figure 6.2 is output.
- When the TPUa channel 0 overflow interrupt (callback) is detected, TPUa channel 0, PPG unit 1 is closed as an error.

Arguments None

Return values None

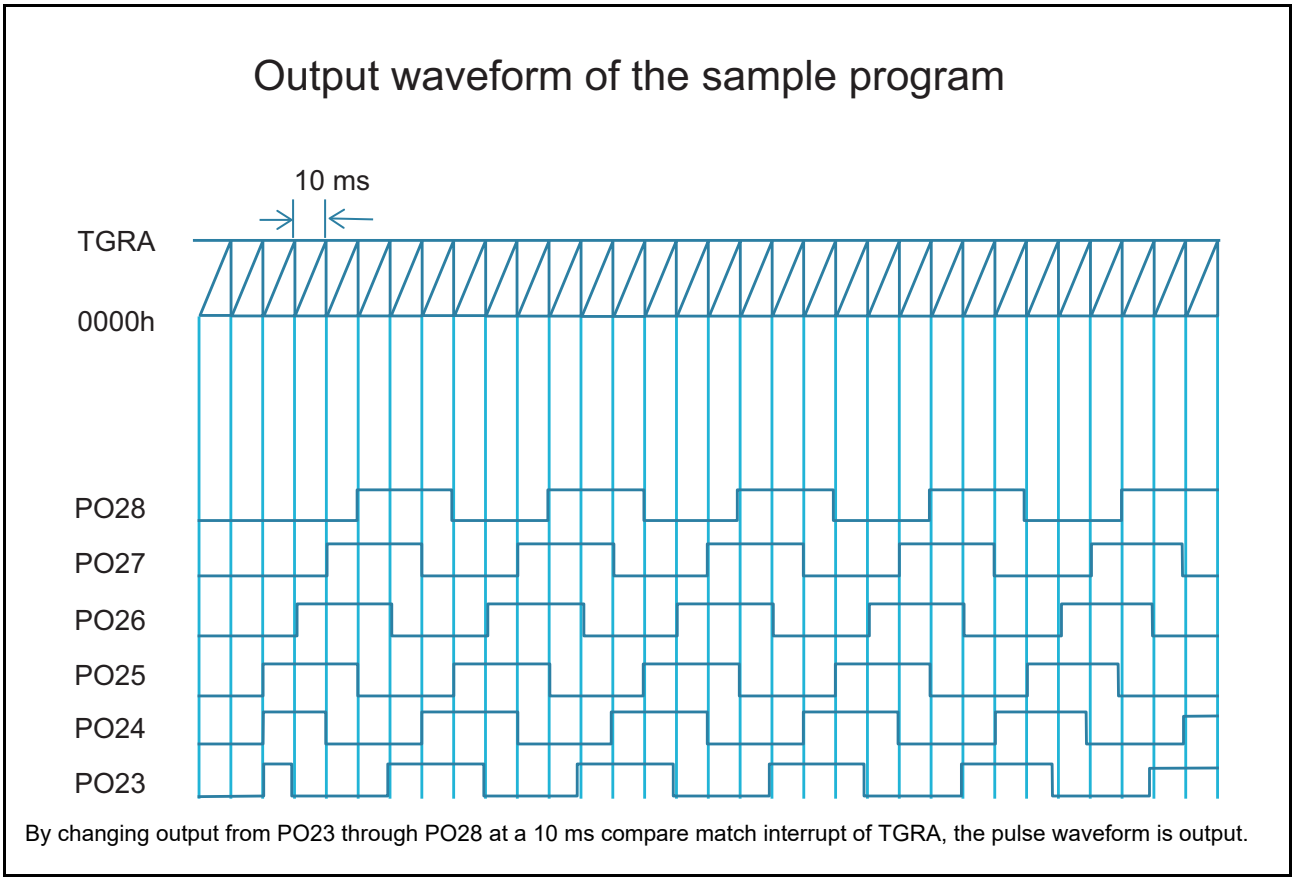



**Figure 6.2 Output Waveform**

### 6.9.2 tpu0_a_cmpmatch_callback

| tpu0_a_cmpmatch_callback | |
|---|---|
| Synopsis | Callback function for TPU0A (compare match) interrupts |
| Header | – |
| Declaration | static void tpu0_a_cmpmatch_callback(void) |
| Description | Sets the compare match interrupt occurrence flag (Table 6.17), and uses main to monitor this flag for detecting a compare match interrupt. |
| Arguments | None |
| Return values | None |

### 6.9.3 tpu0_v_callback

| tpu0_v_callback | |
|---|---|
| Synopsis | Callback function for TPU0V (overflow) interrupts |
| Header | – |
| Declaration | static void tpu0_v_callback(void) |
| Description | Sets the overflow occurrence flag (Table 6.17), and uses main to monitor this flag for detecting an overflow interrupt. |
| Arguments | None |
| Return values | None |

### 6.9.4 R_TPUA_Open

| R_TPUA_Open | |
|---|---|
| Synopsis | Opening TPUa modules |
| Header | r_tpua_rzt1_if.h |
| Declaration | tpua_err_t R_TPUA_Open(const uint8_t channel,<br>const tpua_callback_t * const pcallback,<br>tpua_handle_t * const phandle) |
| Description | For the specification channel (first argument) of TPUa, this function opens the TPUa module, and registers the callback function to be specified for the parameter (second argument), and returns the opened channel information to the handle (third argument) after setting it. The execution result of the function is returned as the return value.<br>Major processing<br>• Checking arguments<br>- Checking the specified range of the channel (first argument)<br>- Pointer null check of the parameter (second argument)<br>- Pointer null check of the handle (third argument)<br>- Checking the channel open status<br>If the channel is already opened, an error occurs.<br>(Already opened channels cannot be opened. To reopen a channel, use the R_TPUA_Close function to close it first.)<br>• Setting the open status<br>• Registering the callback function<br>• Cancelling the stop status of the 0/1 on the TPUa unit (Turning off the power consumption reduction function)<br>- Enabling interrupts through interrupt vector settings of the specified channel (first argument) and ICU |

Registering the callback function
• When registering the callback_a function for TGRA interrupts, and the callback_b function for TGRV interrupts
⇒ For TPUa channel 0, set the callback_a function for pintr_a of the structure member for the second argument (Table 6.13), callbacktbl, and the callback_v function for pintr_v (Specify null for the unused callback function).

```
callbacktbl.pintr_a = &callback_a;
callbacktbl.pintr_b = NULL;
callbacktbl.pintr_c = NULL;
callbacktbl.pintr_d = NULL;
callbacktbl.pintr_v = &callback_v;
  R_TPUA_Open(0, &callbacktabl, &handle)
```

| | | |
|---|---|---|
| Arguments | const uint8_t channel | : Specify a channel.<br>Specify a value from 0 to 11 for a TPUa channel.<br>Unit 0: TPUa channels 0 to 5<br>Unit 1: TPUa channels 6 to 11 |
| | const tpua_callback_t * const pcallback | : Specify a callback function.<br>Specify a callback function to be executed when a TGRA/TGRB/TGRC/TGRD/TGRV/TGRU interrupt occurs. |
| | tpua_handle_t * const phandle | : Specify a pointer for the handle of TPUa.<br>Returns channel information of TPUa which was opened for the area specified by the pointer of phandle.<br>The area specified by the pointer of phandle must be reserved while calling R_TPUA_Open. |
| Return values | TPUA_SUCCESS | : Success: Opening the TPUa module is successful. |
| | TPUA_ERR_BAD_CHAN | : Failure: Invalid channel |
| | TPUA_ERR_CH_NOT_CLOSED | : Failure: Already opened. |
| | TPUA_ERR_NULL_PTR | : Failure: No parameter is specified. |

### 6.9.5 R_TPUA_Control

| R_TPUA_Control | |
|---|---|
| Synopsis | Executing commands of TPUa modules |
| Header | r_tpua_rzt1_if.h |
| Declaration | tpua_err_t R_TPUA_Control(const tpua_handle_t * const phandle,<br>const tpua_cmd_t cmd,<br>tpua_reg_t * const pregdat) |
| Description | For the channel specified by the handle (first argument), executes the specified command (second argument). Depending on the command to be executed, usage of the parameter (third argument) varies. There are cases when data is referenced only, or it is set and returned. The execution result of the function is returned as the return value.<br>Major processing<br>• Checking arguments<br>- Pointer null check of the specified handle (first argument)<br>- Checking the channel open status<br>If the channel is not opened yet, an error occurs.<br>(The R_TPUA_Control function must be executed in advance by using the R_TPUA_Open function when the channel is opened.)<br>• Command processing<br>Uses the command specified by the second argument to perform processing.<br>An error is returned as an unknown command for the command that is not applicable.<br>For details about the command, refer to Section 6.11, R_TPUA_Control Commands. |
| Arguments | const tpua_handle_t * const phandle : Specifies the pointer of the handle for TPUa. Uses R_TPUA_Open to specify the opened handle.<br>const tpua_cmd_t cmd : Specifies the command to execute (For details, refer to Section 6.11).<br>tpua_reg_t * const pregdat : Specifies the pointer for register data. When the register return command is executed, the read value of the register is stored and returned to this argument. When the register write command is executed, the value set for this argument is written to the register. (For details, refer to Section 6.11.) |
| Return values | TPUA_SUCCESS : Success: A command for the TPUa module is successfully executed.<br>TPUA_ERR_CH_NOT_OPENED : Failure: The channel is not opened.<br>TPUA_ERR_NULL_PTR : Failure: No parameter is specified.<br>TPUA_ERR_UNKNOWN_CMD : Failure: An unknown command was specified. |

### 6.9.6 R_TPUA_Close

| R_TPUA_Close | | |
|---|---|---|
| Synopsis | Closing TPUa modules | |
| Header | r_tpua_rzt1_if.h | |
| Declaration | tpua_err_t R_TPUA_Close(const tpua_handle_t * const phandle) | |
| Description | Closes the channel specified by the handle (first argument).<br>Returns the execution result of the function as the return value.<br>Major processing<br>• Checking arguments<br>- Pointer null check of the specified handle (first argument)<br>- Checking the channel open status<br>If the channel is not opened yet, an error occurs.<br>(Already closed channels cannot be closed. Only the channels opened by using the R_TPUA_Open function can be closed.)<br>• ICU disables interrupts on the specified channel (first argument).<br>• Releasing the open status<br>• Transition to the stop state of the 0/1 module in the TPUa unit (Turning on the power consumption reduction function).<br>Only when all channels in the unit are closed, the power consumption reduction function is turned on. If at least one channel in the unit is opened, the function is not turned on. | |
| Arguments | const tpua_handle_t * const phandle | : Specifies the pointer of a handle for TPUa.<br>Uses R_TPUA_Open to specify an opened handle. |
| Return values | TPUA_SUCCESS<br>TPUA_ERR_CH_NOT_OPENED<br>TPUA_ERR_NULL_PTR | : Success: Closing the TPUa module is successful.<br>: Failure: The channel is not opened.<br>: Failure: No parameter is specified. |

### 6.9.7 R_TPUA_GetVersion

| R_TPUA_GetVersion | |
|---|---|
| Synopsis | Acquiring version information of TPUa modules |
| Header | r_tpua_rzt1_if.h |
| Declaration | uint32_t R_TPUA_GetVersion(void) |
| Description | Returns the version information of a TPUa module as the return value. |
| Arguments | None |
| Return values | Version information of the TPUa sample driver (32 bit)<br>16-31bit: Major Version<br>0-15bit: Minor Version |

### 6.9.8 R_PPG_Open

| R_PPG_Open | |
|---|---|
| Synopsis | Opening PPG modules |
| Header | r_ppg_rzt1_if.h |
| Declaration | ppg_err_t R_PPG_Open(const uint8_t unit,<br>ppg_handle_t * const phandle) |
| Description | Opens a PPG module for the specified unit (first argument) of PPG, and sets and returns the opened unit information to the handle (second argument). Returns the execution result of the function as the return value.<br>Major processing<br>• Checking arguments<br>- Checking the specified range for the unit (first argument)<br>- Pointer null check of the handle (second argument)<br>- Checking the open status of the unit<br>An error occurs if the unit is already opened.<br>(An already opened unit cannot be opened. To reopen it, you must close it first by using the R_PPG_Close function.)<br>• Setting the open status<br>• Releasing the stop state of the 0/1 module in the PPG unit (turning off the power consumption reduction function). |
| Arguments | const uint8_t unit : Specifies units.<br>Uses 0 or 1 to specify the PPG unit.<br>ppg_handle_t * const phandle : Specifies the pointer of PPG handles.<br>Returns the unit information of the opened PPG to the area specified by the pointer for phandle.<br>The area specified by the pointer for phandle must be reserved by calling R_PPG_Open. |
| Return values | PPG_SUCCESS : Success: Opening the PPG module is successful.<br>PPG_ERR_BAD_UNIT : Failure: Invalid channel<br>PPG_ERR_UN_NOT_CLOSED : Failure: Already opened.<br>PPG_ERR_NULL_PTR : Failure: No parameter is specified. |

### 6.9.9 R_PPG_Control

| R_PPG_Control | |
|---|---|
| Synopsis | Executing commands of PPG modules |
| Header | r_ppg_rzt1_if.h |
| Declaration | ppg_err_t R_PPG_Control(const ppg_handle_t * const phandle,<br>const ppg_cmd_t cmd,<br>ppg_reg_t * const pregdat) |
| Description | Executes the specified command (second argument) for the unit specified by the handle (first argument). Depending on the command to be executed, usage of parameters (third argument) varies. There are cases when data is referenced only, or it is set and returned. The execution result of the function is returned as the return value.<br>Major processing<br>• Checking arguments (Common processing regardless of arguments)<br>- Pointer null check of the specified handle (first argument)<br>- Checking the open status of a unit<br>An error occurs if the unit is not opened.<br>(The R_PPG_Control function must be executed in advance by using the R_PPG_Open function when the unit is opened.)<br>• Command processing<br>The applicable processing is performed according to the command specified by the second argument.<br>For the command that is not applicable, an error is returned as an unknown command.<br>For details about commands, refer to Section 6.12, R_PPG_Control Commands. |
| Arguments | const ppg_handle_t * const phandle : Specifies the pointer of the handle for PPG. Uses R_PPG_Open to specify the opened handle.<br>const ppg_cmd_t cmd : Specifies the command to execute (For details, refer to Section 6.12).<br>ppg_reg_t * const pregdat : When the register read command is executed, the read value of the register is stored and returned to this argument. When the register write command is executed, the value set for this argument is written to the register (For details, refer to Section 6.12). |
| Return values | PPG_SUCCESS : Success: A command of the PPG module is successfully executed.<br>PPG_ERR_UN_NOT_OPENED : Failure: The unit is not opened.<br>PPG_ERR_NULL_PTR : Failure: No parameter is specified.<br>PPG_ERR_UNKNOWN_CMD : Failure: An unknown command was specified. |

### 6.9.10 R_PPG_Close

| R_PPG_Close | |
|---|---|
| Synopsis | Closing PPG modules |
| Header | r_ppg_rzt1_if.h |
| Declaration | ppg_err_t R_PPG_Close(const ppg_handle_t * const phandle) |
| Description | Closes the unit specified by the specified handle (first argument).<br>Returns the execution result of the function as the return value.<br>Major processing<br>• Checking arguments<br>- Pointer null check of the specified handle (first argument)<br>- Checking the open status of a unit<br>An error occurs if the unit is not opened.<br>(Already closed units cannot be closed. Only the unit opened by the R_PPG_Open function can be closed.)<br>• Releasing the open status<br>• Transition to the stop state of the 0/1 module of the PPG unit (Turning on the power consumption reduction function). |
| Arguments | const ppg_handle_t * const phandle : Specifies the pointer of a handle for PPG.<br>Uses R_PPG_Open to specify an opened handle. |
| Return values | PPG_SUCCESS : Success: Closing the PPG module is successful.<br>PPG_ERR_UN_NOT_OPENED : Failure: The unit is not opened.<br>PPG_ERR_NULL_PTR : Failure: No parameter is specified. |

### 6.9.11 R_PPG_GetVersion

| R_PPG_GetVersion | |
|---|---|
| Synopsis | Acquiring version information of PPG modules |
| Header | r_ppg_rzt1_if.h |
| Declaration | uint32_t R_PPG_GetVersion(void) |
| Description | Returns the version information of a PPG module as the return value. |
| Arguments | None |
| Return values | Version information of the PPG sample driver (32 bit)<br>16-31bit: Major Version<br>0-15bit: Minor Version |

## 6.10 Flowchart

### 6.10.1 main Processing

Figure 6.3 shows the flowchart of the main processing.

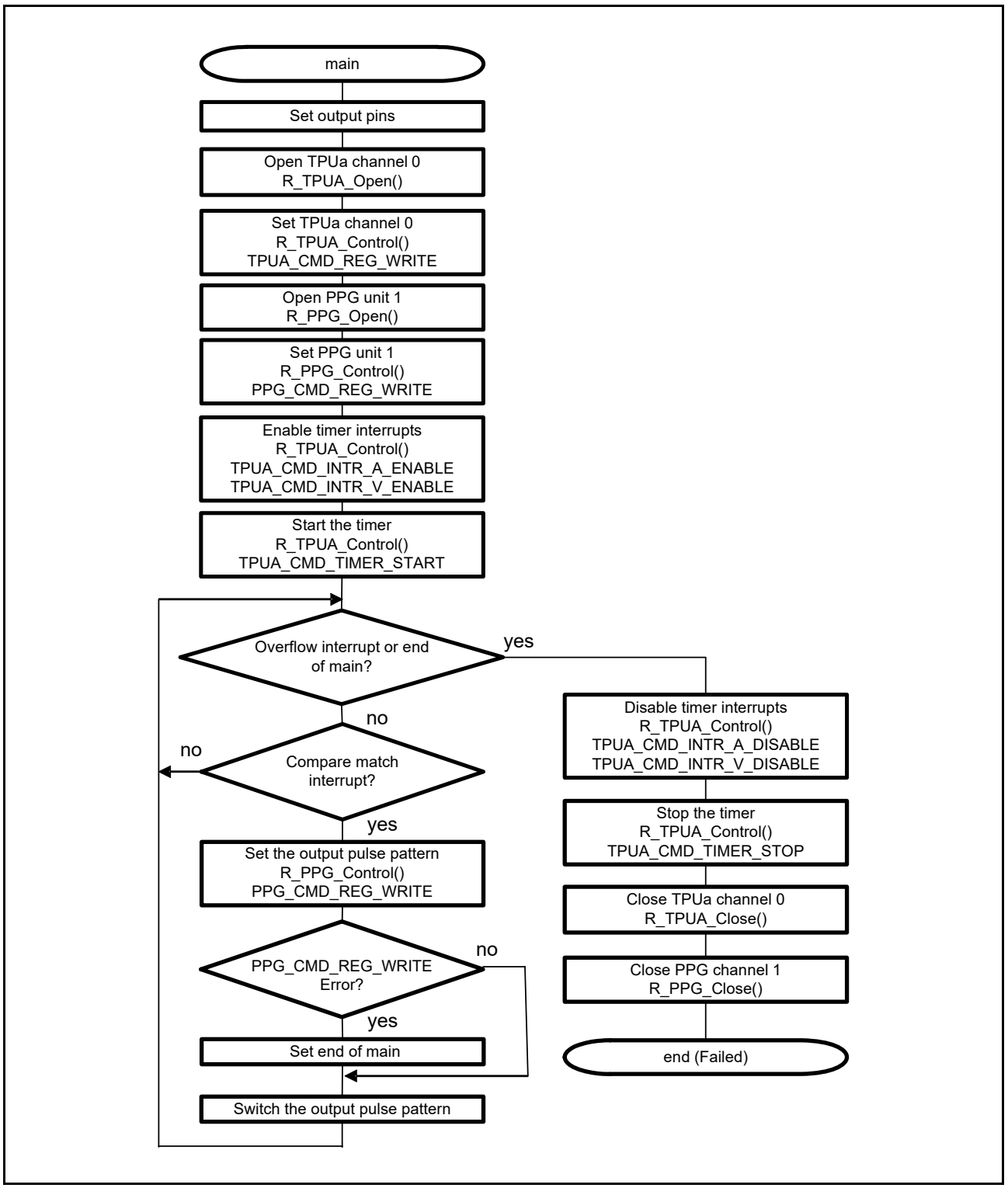



**Figure 6.3 Flowchart of the main Processing**

### 6.10.2 tpu0_a_cmpmatch_callback Processing

Figure 6.4 shows the flowchart of the tpu0_a_cmpmatch_callback processing.

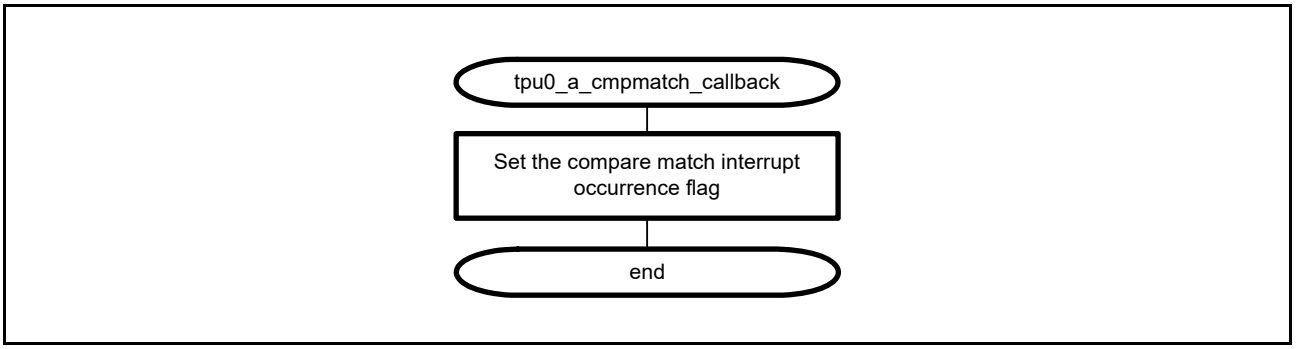



**Figure 6.4 Flowchart of the tpu0_a_cmpmatch_callback Processing**

### 6.10.3 tpu0_v_callback Processing

Figure 6.5 shows the flowchart of the tpu0_v_callback processing.

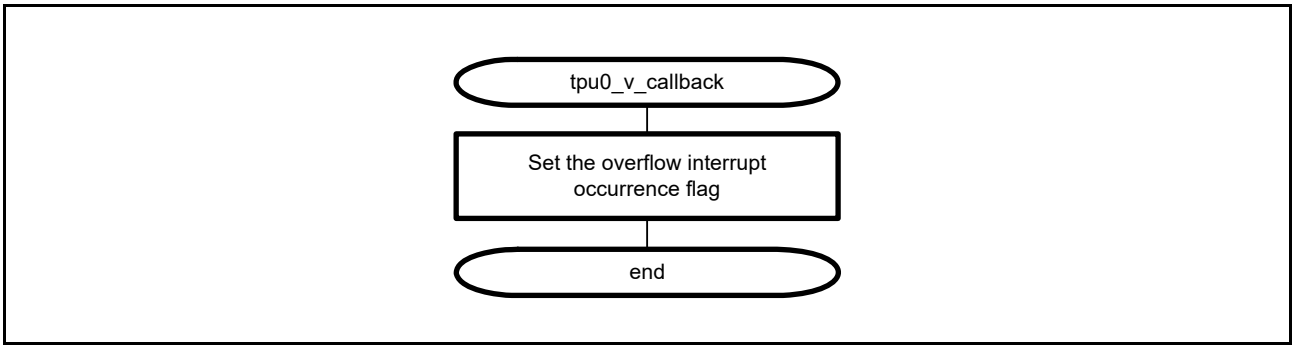



**Figure 6.5 Flowchart of the tpu0_v_callback Processing**

### 6.10.4 R_TPUA_Open Processing

Figure 6.6 shows the flowchart of the R_TPUA_Open processing.

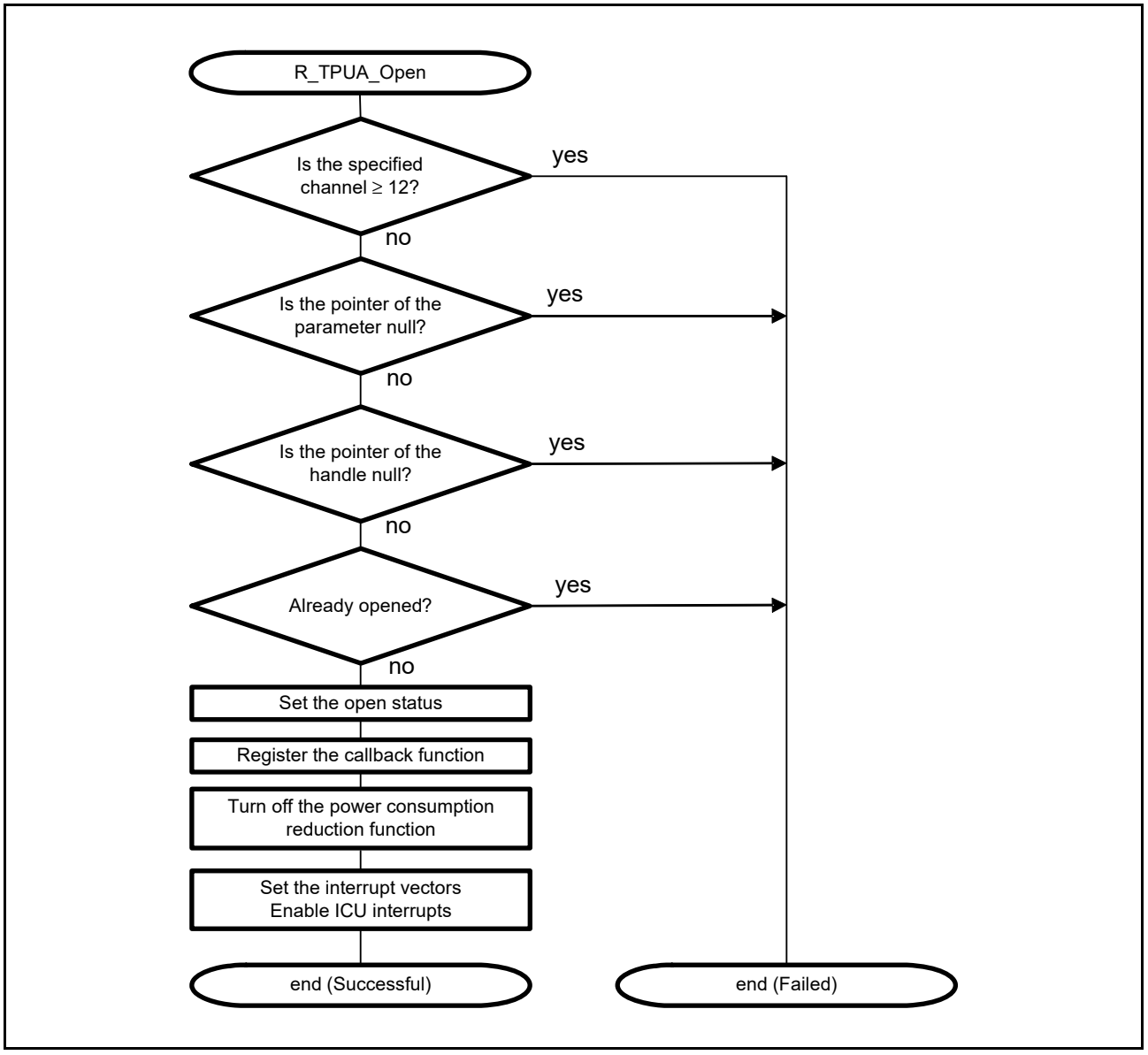



**Figure 6.6 Flowchart of the R_TPUA_Open Processing**

### 6.10.5 R_TPUA_Control Processing

Figure 6.7 and Figure 6.8 show flowcharts of the R_TPUA_Control processing.

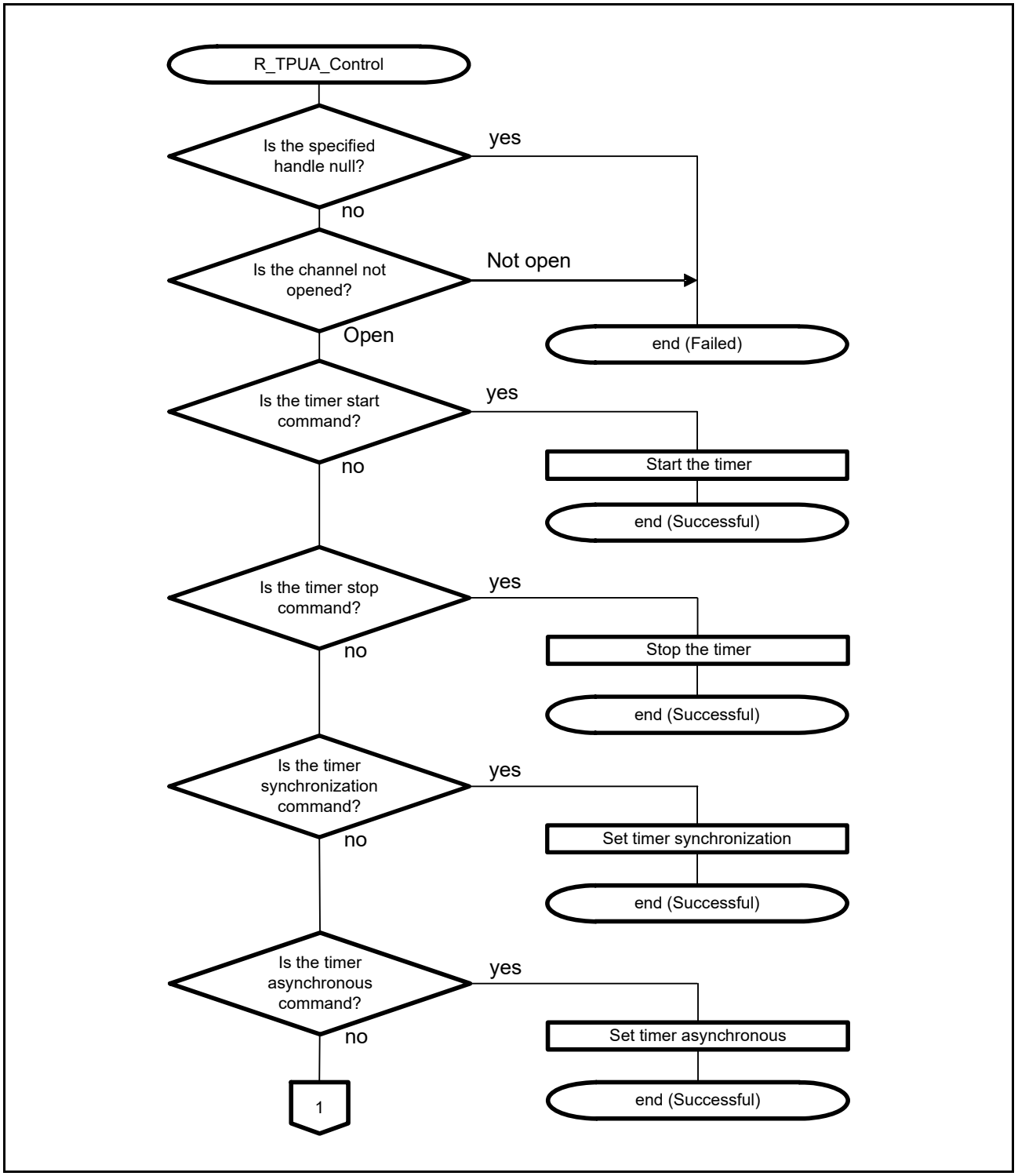



**Figure 6.7 Flowchart of the R_TPUA_Control Processing**

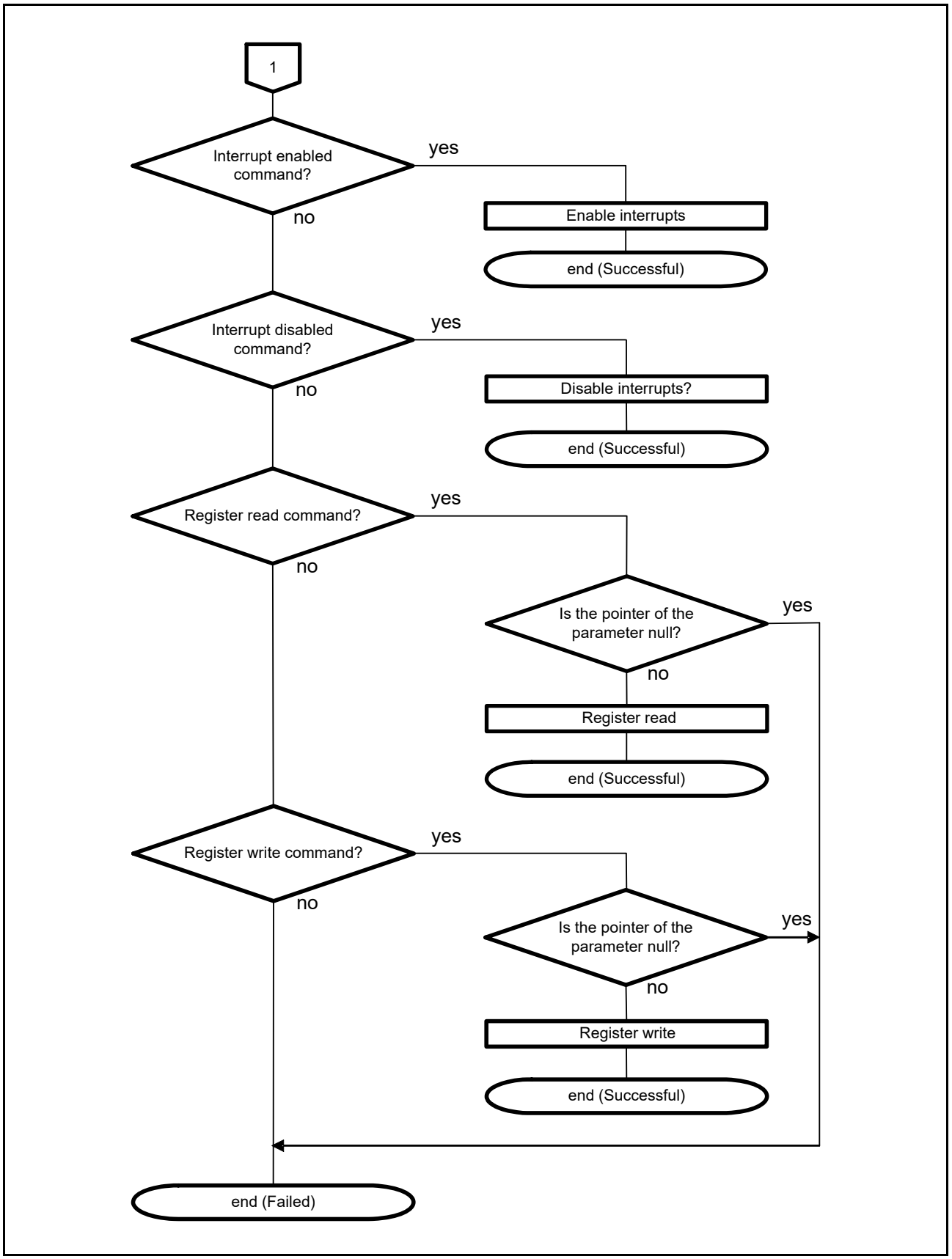



**Figure 6.8 Flowchart of the R_TPUA_Control Processing**

### 6.10.6 R_TPUA_Close Processing

Figure 6.9 shows the flowchart of the R_TPUA_Close processing.

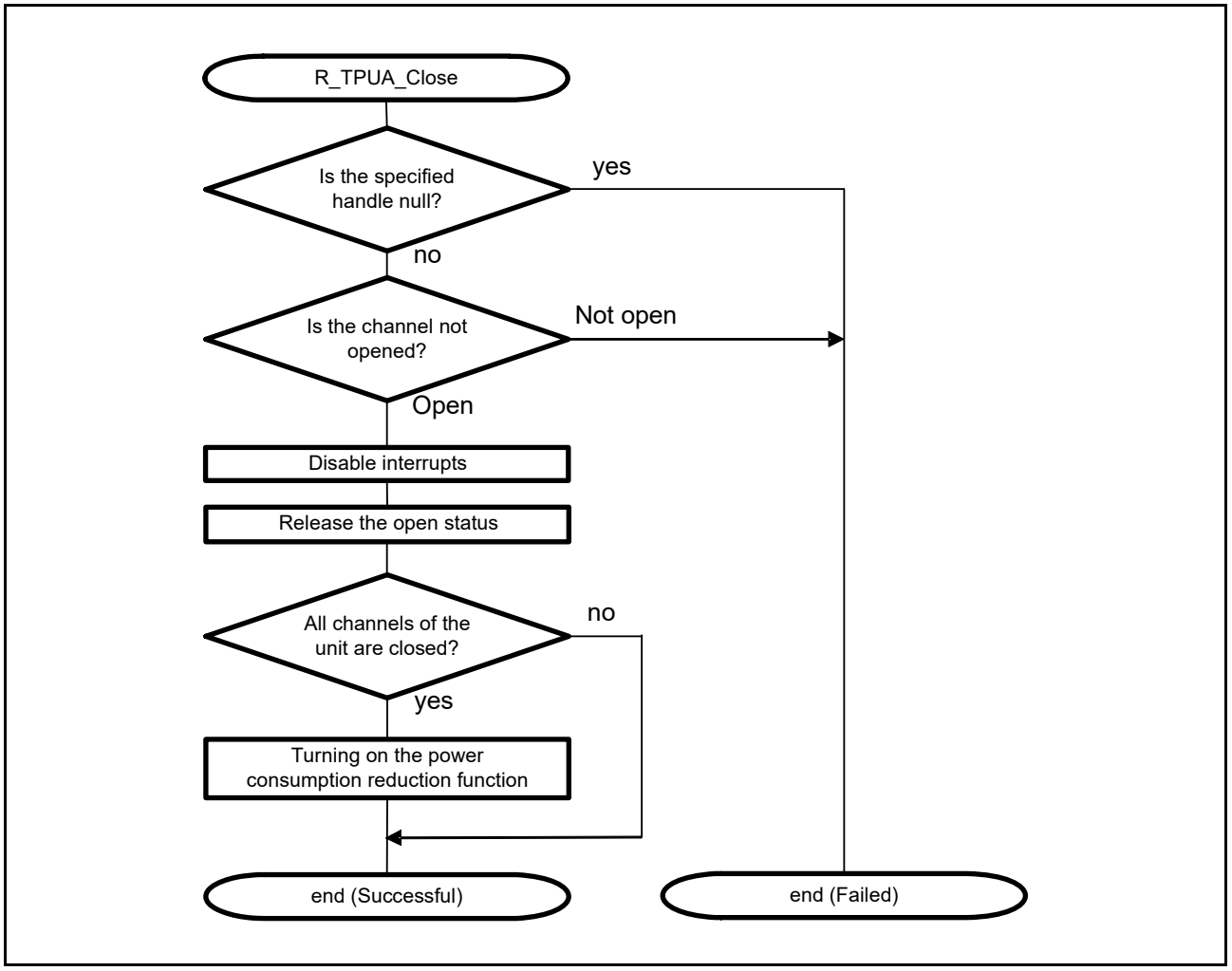



**Figure 6.9 Flowchart of the R_TPUA_Close Processing**

### 6.10.7 R_TPUA_GetVersion Processing

Figure 6.10 shows the flowchart of the R_TPUA_GetVersion processing.

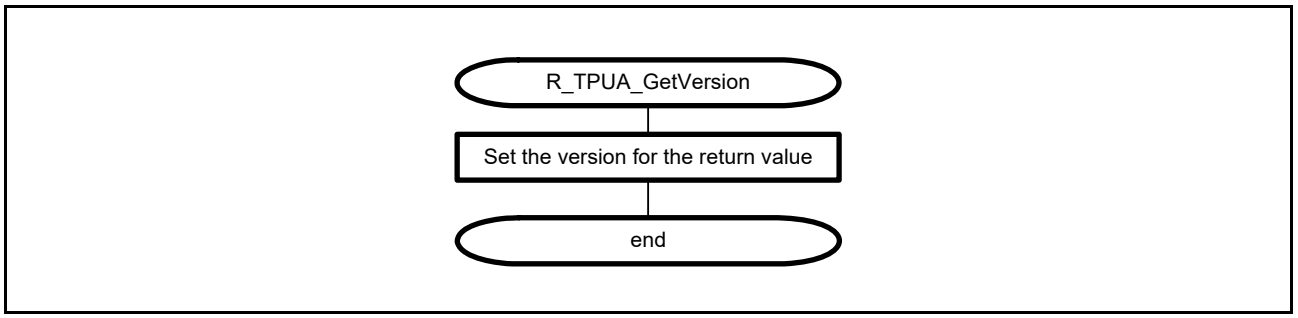



**Figure 6.10 Flowchart of the R_TPUA_GetVersion Processing**

### 6.10.8 R_PPG_Open Processing

Figure 6.11 shows the flowchart of the R_PPG_Open processing.

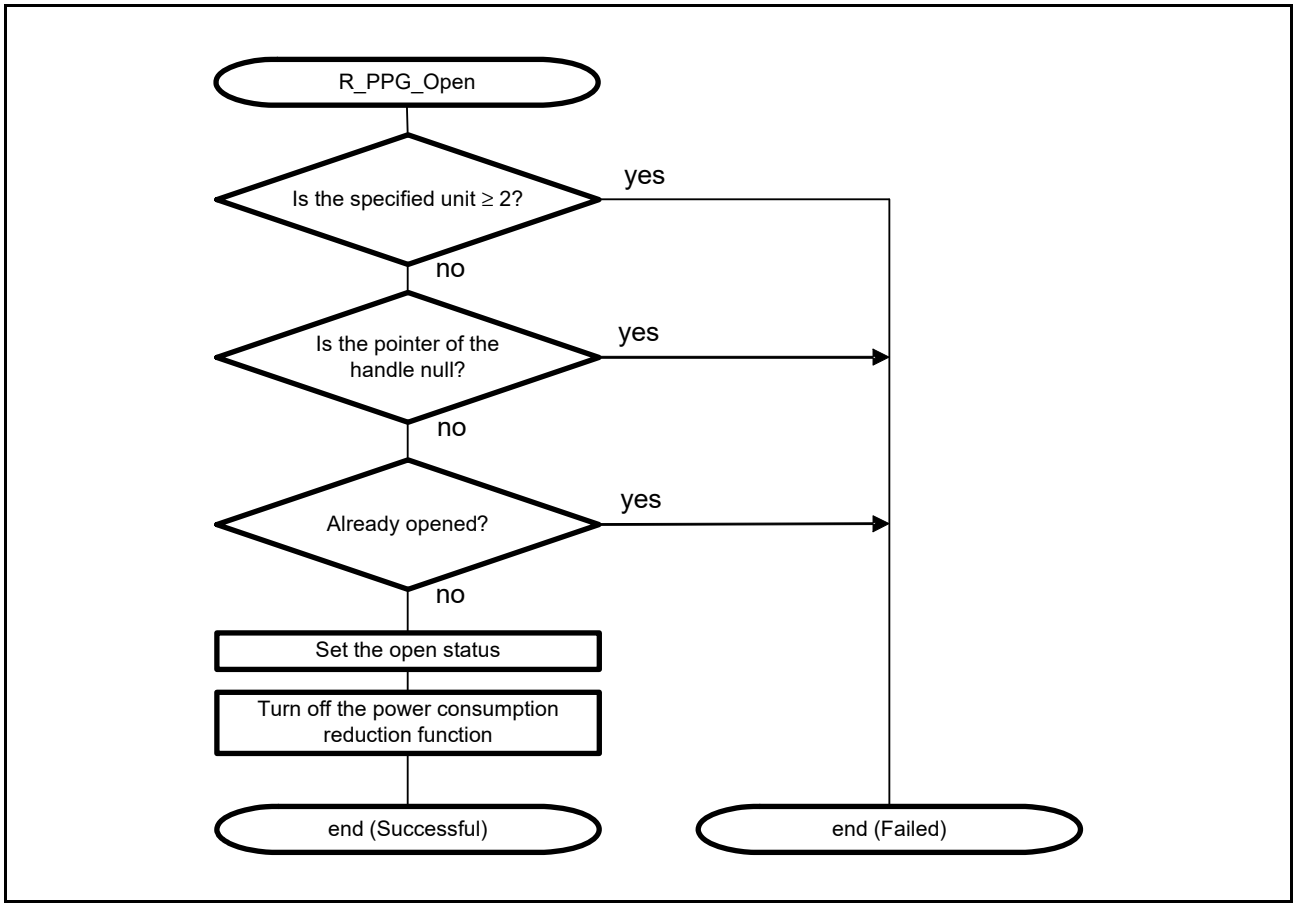



**Figure 6.11 Flowchart of the R_PPG_Open Processing**

### 6.10.9 R_PPG_Control Processing

Figure 6.12 shows the flowchart of the R_PPG_Control processing.

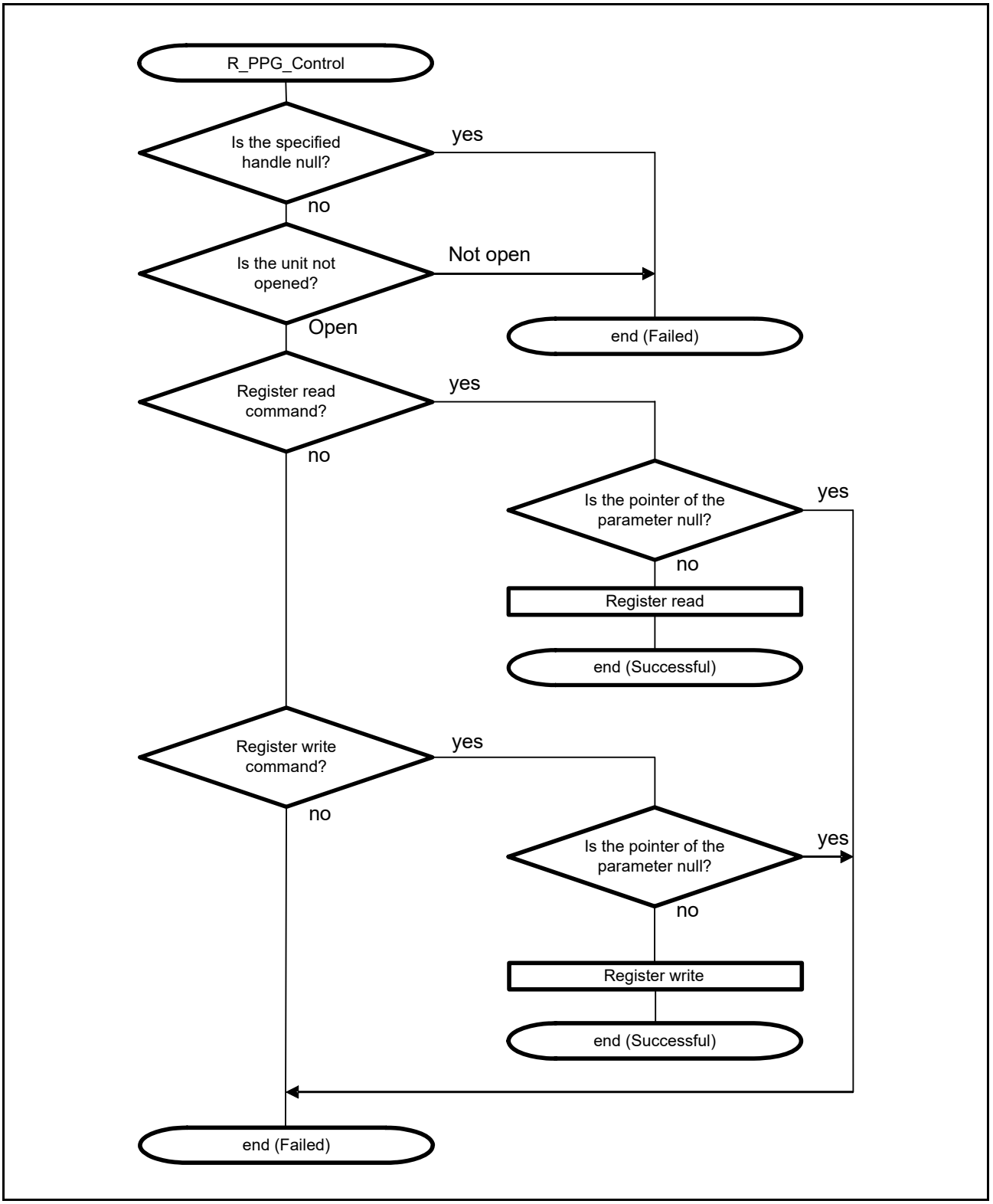



**Figure 6.12 Flowchart of the R_PPG_Control Processing**

### 6.10.10 R_PPG_Close Processing

Figure 6.13 shows the flowchart of the R_PPG_Close processing.

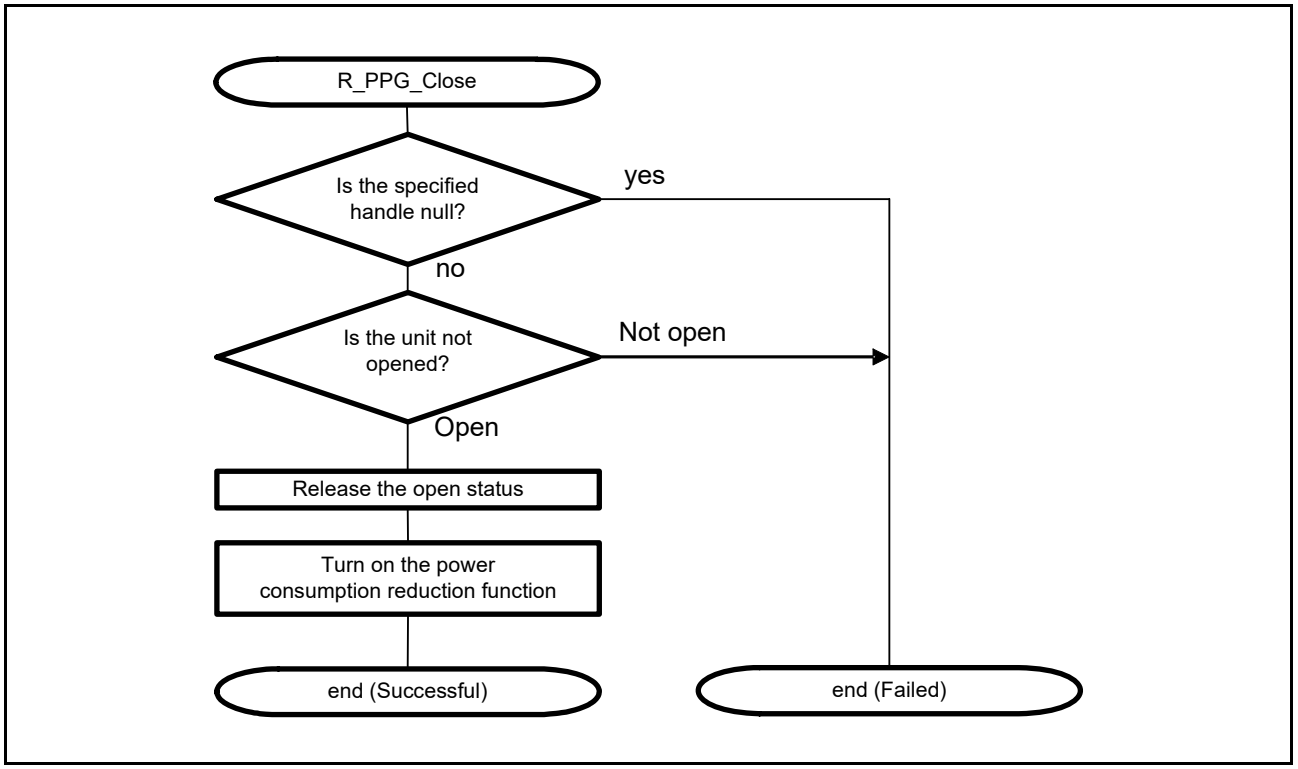



**Figure 6.13 Flowchart of the R_PPG_Close Processing**

### 6.10.11 R_PPG_GetVersion Processing

Figure 6.14 shows the flowchart of the R_PPG_GetVersion processing.

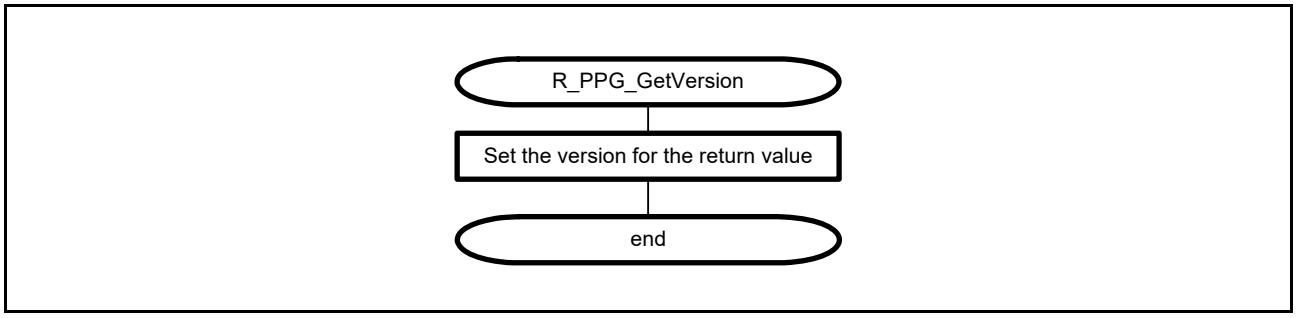



**Figure 6.14 Flowchart of the R_PPG_GetVersion Processing**

## 6.11 R_TPUA_Control Commands

The following table lists the commands that are used by the R_TPUA_Control function.

**Table 6.19 R_TPUA_Control Commands**

| Command | Description |
|---|---|
| TPUA_CMD_TIMER_START | Starts the timer of the specified channel.*1 |
| TPUA_CMD_TIMER_STOP | Stops the timer of the specified channel.*1 |
| TPUA_CMD_TIMER_SYNC | Specifies the timer of the specified channel as synchronous operation.*1 |
| TPUA_CMD_TIMER_ASYNC | Specifies the timer of the specified channel as independent (asynchronous) operation.*1 |
| TPUA_CMD_REG_READ | Reads the register value of the specified channel.*2 |
| TPUA_CMD_REG_WRITE | Writes the specified value to the register of the specified channel.*3 |
| TPUA_CMD_INTR_A_ENABLE | Enables the TGRA interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_A_DISABLE | Disables the TGRA interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_B_ENABLE | Enables the TGRB interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_B_DISABLE | Disables the TGRB interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_C_ENABLE | Enables the TGRC interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_C_DISABLE | Disables the TGRC interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_D_ENABLE | Enables the TGRD interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_D_DISABLE | Disables the TGRD interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_V_ENABLE | Enables the overflow interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_V_DISABLE | Disables the overflow interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_U_ENABLE | Enables the underflow interrupt of the specified channel.*1 |
| TPUA_CMD_INTR_U_DISABLE | Disables the underflow interrupt of the specified channel.*1 |

Note 1. Use the first argument of the R_TPUA_Control function to specify the channel.
The same operation can be specified by using the TPUA_CMD_REG_WRITE command.
Note 2. Use the first argument of the R_TPUA_Control function to specify the channel.
The read value is stored in the third argument of the R_TPUA_Control function.
Note 3. Use the first argument of the R_TPUA_Control function to specify the channel.
The specified value to be written is set by using the third argument of the R_TPUA_Control function.

### 6.11.1 TPUA_CMD_TIMER_START

| TPUA_CMD_TIMER_START | |
|---|---|
| Synopsis | Starting count operation of the TPUa timer. |
| Header | r_tpua_rzt1_if.h |
| Description | Starts count operation of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must use the TPUA_CMD_REG_WRITE command to operate the register directly.

### 6.11.2 TPUA_CMD_TIMER_STOP

| TPUA_CMD_TIMER_STOP | |
|---|---|
| Synopsis | Stopping count operation of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Stops count operation of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must use the TPUA_CMD_REG_WRITE command to operate the register directly.

### 6.11.3 TPUA_CMD_TIMER_SYNC

| TPUA_CMD_TIMER_SYNC | |
|---|---|
| Synopsis | Specifying synchronous operation of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Specifies synchronous operation of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must use the TPUA_CMD_REG_WRITE command to operate the register directly.

### 6.11.4 TPUA_CMD_TIMER_ASYNC

| TPUA_CMD_TIMER_ASYNC | |
|---|---|
| Synopsis | Specifying independent (asynchronous) operation of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Specifies independent operation of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must use the TPUA_CMD_REG_WRITE command to operate the register directly.

### 6.11.5 TPUA_CMD_REG_READ

TPUA_CMD_REG_READ

Synopsis Reading the TPUa register

Header r_tpua_rzt1_if.h

Description Stores the read value of the register specified by reg_flag of the tpua_reg_t structure member in the member of the tpua_reg_t structure.
Before executing the TPUA_CMD_REG_READ command, use reg_flag to specify the register to read. After executing the command, you can read the register value by referencing the member of tpua_reg_t.

Parameters

| | | |
|---|---|---|
| tpua_reg_t | uint32_t reg_flag | Specifies the register to read by using the any of following parameters:<br>• TPUA_TCR_REG: Specifies the TCR register.<br>• TPUA_TMDR_REG: Specifies the TMDR register.<br>• TPUA_TIORH_REG: Specifies the TIORH register.<br>• TPUA_TIOR_REG: Specifies the TIOR register.<br>• TPUA_TIORL_REG: Specifies the TIORL register.<br>• TPUA_TIER_REG: Specifies the TIER register.<br>• TPUA_TSR_REG: Specifies the TSR register.<br>• TPUA_TCNT_REG: Specifies the TCNT register.<br>• TPUA_TGRA_REG: Specifies the TGRA register.<br>• TPUA_TGRB_REG: Specifies the TGRB register.<br>• TPUA_TGRC_REG: Specifies the TGRC register.<br>• TPUA_TGRD_REG: Specifies the TGRD register.<br>• TPUA_TSTRA_REG: Specifies the TSTRA register.<br>• TPUA_TSTRB_REG: Specifies the TSTRB register.<br>• TPUA_TSYRA_REG: Specifies the TSYRA register.<br>• TPUA_TSYRB_REG: Specifies the TSYRB register.<br>• TPUA_NFCR_REG: Specifies the NFCR register.<br>• TPUA_PWMFBSLR_REG: Specifies the PWMFBSLR register.<br>• TPUA_ALL_REG: Specifies all registers.<br>Note: To specify multiple registers, separate parameters with OR. |

Return values TPUA_SUCCESS : Success: Reading the register is successful.

Remarks For details about registers, see the RZ/T1 Group User's Manual: Hardware (R01UH0483EJ).

Example of reading registers
• When reading the TSR, and TCNT registers
⇒ Use a parameter to specify the register you want to read for regdata.reg_flag of the third argument for the R_TPUA_Control function. After executing the command, the read value is saved in the regdata member.

```
regdata.reg_flag = TPUA_TSR_REG | TPUA_TCNT_REG;
R_TPUA_Control(handle, TPUA_CMD_REG_READ, &regdata)
```

### 6.11.6 TPUA_CMD_REG_WRITE

TPUA_CMD_REG_WRITE

Synopsis: Reading the TPUa register

Header: r_tpua_rzt1_if.h

Description: Writes the value specified by the tpua_reg_t structure member for the register specified by reg_flag of the tpua_reg_t structure member.
Before executing the TPUA_CMD_REG_WRITE command, specify the register to be written to reg_flag. At the same time, set the value to be written to the member of tpua_reg_t, and then execute the command. By doing so, the value can be written to the register.

Parameters

| | | |
|---|---|---|
| tpua_reg_t | uint32_t reg_flag | Use the following parameters to specify the register to be written:<br>• TPUA_TCR_REG: Specifies the TCR register.<br>• TPUA_TMDR_REG: Specifies the TMDR register.<br>• TPUA_TIORH_REG: Specifies the TIORH register.<br>• TPUA_TIOR_REG: Specifies the TIOR register.<br>• TPUA_TIORL_REG: Specifies the TIORL register.<br>• TPUA_TIER_REG: Specifies the TIER register.<br>• TPUA_TSR_REG: Specifies the TSR register.<br>• TPUA_TCNT_REG: Specifies the TCNT register.<br>• TPUA_TGRA_REG: Specifies the TGRA register.<br>• TPUA_TGRB_REG: Specifies the TGRB register.<br>• TPUA_TGRC_REG: Specifies the TGRC register.<br>• TPUA_TGRD_REG: Specifies the TGRD register.<br>• TPUA_TSTRA_REG: Specifies the TSTRA register.<br>• TPUA_TSTRB_REG: Specifies the TSTRB register.<br>• TPUA_TSYRA_REG: Specifies the TSYRA register.<br>• TPUA_TSYRB_REG: Specifies the TSYRB register.<br>• TPUA_NFCR_REG: Specifies the NFCR register.<br>• TPUA_PWMFBSLR_REG: Specifies the PWMFBSLR register.<br>• TPUA_ALL_REG: Specifies all registers.<br>Note: To specify multiple registers, separate parameters with OR. |
| | uint8_t tcr_reg | Sets the value to be written to the TCR register. |
| | uint8_t tmdr_reg | Sets the value to be written to the TMDR register. |
| | uint8_t tiorh_reg | Sets the value to be written to the TIORH register. |
| | uint8_t tior_reg | Sets the value to be written to the TIOR register. |
| | uint8_t tiorl_reg | Sets the value to be written to the TIORL register. |
| | uint8_t tier_reg | Sets the value to be written to the TIER register. |
| | uint8_t tsr_reg | Sets the value to be written to the TSR register. |
| | uint16_t tcnt_reg | Sets the value to be written to the TCNT register. |
| | uint16_t tgra_reg | Sets the value to be written to the TGRA register. |
| | uint16_t tgrb_reg | Sets the value to be written to the TGRB register. |
| | uint16_t tgrc_reg | Sets the value to be written to the TGRC register. |
| | uint16_t tgrd_reg | Sets the value to be written to the TGRD register. |
| | uint8_t tstra_reg | Sets the value to be written to the TSTRA register. |
| | uint8_t tstrb_reg | Sets the value to be written to the TSTRB register. |
| | uint8_t tsyra_reg | Sets the value to be written to the TSYRA register. |
| | uint8_t tsyrb_reg | Sets the value to be written to the TSYRB register. |
| | uint8_t nfcr_reg | Sets the value to be written to the TNFCR register. |

| | | |
|---|---|---|
| | uint32_t pwmfbslr_reg | Sets the value to be written to the PWMFBSLR register. |
| Return values | TPUA_SUCCESS | : Success: Register write is successful. |

Remarks For details about registers, see the RZ/T1 Group User's Manual: Hardware (R01UH0483EJ).

Example of specifying register write settings
• When writing 0x01 to the TCR register, and 0x02 to the TMDR register
⇒ Use a parameter to specify the register to write for regdata.reg_flag of the third argument of the R_TPUA_Control function, and the value you want to write for the regdata member.

```
regdata.reg_flag = TPUA_TCR_REG | TPUA_TMDR_REG;
regdata. tcr_reg = 0x01;
regdata. tmdr_reg = 0x02;
R_TPUA_Control(handle, TPUA_CMD_REG_WRITE, &regdata)
```

## 6.11.7 TPUA_CMD_INTR_A_ENABLE

TPUA_CMD_INTR_A_ENABLE

| | |
|---|---|
| Synopsis | Enabling TGRA interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Enables TGRA interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

## 6.11.8 TPUA_CMD_INTR_A_DISABLE

TPUA_CMD_INTR_A_DISABLE

| | |
|---|---|
| Synopsis | Disabling TGRA interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Disables TGRA interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.9 TPUA_CMD_INTR_B_ENABLE

| TPUA_CMD_INTR_B_ENABLE | |
|---|---|
| Synopsis | Enabling TGRB interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Enables TGRB interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.10 TPUA_CMD_INTR_B_DISABLE

| TPUA_CMD_INTR_B_DISABLE | |
|---|---|
| Synopsis | Disabling TGRB interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Disables TGRB interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.11 TPUA_CMD_INTR_C_ENABLE

| TPUA_CMD_INTR_C_ENABLE | |
|---|---|
| Synopsis | Enabling TGRC interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Enables TGRC interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.12 TPUA_CMD_INTR_C_DISABLE

| TPUA_CMD_INTR_C_DISABLE | |
|---|---|
| Synopsis | Disabling TGRC interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Disables TGRC interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.13 TPUA_CMD_INTR_D_ENABLE

| TPUA_CMD_INTR_D_ENABLE | |
|---|---|
| Synopsis | Enabling TGRD interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Enables TGRD interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.14 TPUA_CMD_INTR_D_DISABLE

| TPUA_CMD_INTR_D_DISABLE | |
|---|---|
| Synopsis | Disabling TGRD interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Disables TGRD interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.15 TPUA_CMD_INTR_V_ENABLE

| TPUA_CMD_INTR_V_ENABLE | |
|---|---|
| Synopsis | Enabling overflow interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Enables overflow interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.16 TPUA_CMD_INTR_V_DISABLE

| TPUA_CMD_INTR_V_DISABLE | |
|---|---|
| Synopsis | Disabling overflow interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Disables overflow interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.17 TPUA_CMD_INTR_U_ENABLE

| TPUA_CMD_INTR_U_ENABLE | |
|---|---|
| Synopsis | Enabling underflow interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Enables underflow interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

### 6.11.18 TPUA_CMD_INTR_U_DISABLE

| TPUA_CMD_INTR_U_DISABLE | |
|---|---|
| Synopsis | Disabling underflow interrupts of the TPUa timer |
| Header | r_tpua_rzt1_if.h |
| Description | Disables underflow interrupts of the TPUa timer.*1 |
| Parameters | None |
| Return values | None |
| Remarks | – |

Note 1. Because each command functions for the channel specified by the first argument, only operation for each channel can be specified.
Because of this, if you want to function multiple channels concurrently, such as cascade operation, you must operate the register directly by using the TPUA_CMD_REG_WRITE command.

## 6.12 R_PPG_Control Commands

The following table lists the commands to be used by the R_PPG_Control function.

**Table 6.20 R_PPG_Control Commands**

| Enumerated Type Definition Name | Description |
|---|---|
| PPG_CMD_REG_READ | Reads the register value of the specified channel.*1 |
| PPG_CMD_REG_WRITE | Writes the specified value to the register of the specified channel.*2 |

Note 1. Use the first argument of the R_PPG_Control function to specify the unit.
The read value is saved in the third argument of the R_PPG_Control function.
Note 2. Use the first argument of the R_PPG_Control function to specify the channel.
Set the specified value to be written in the third argument of the R_PPG_Control function.

### 6.12.1 PPG_CMD_REG_READ

| PPG_CMD_REG_READ | | | |
|---|---|---|---|
| Synopsis | Reading the PPG register | | |
| Header | r_ppg_rzt1_if.h | | |
| Description | Stores the read value of the specified register by using reg_flag of the ppg_reg_t structure to the member of the ppg_reg_t structure.<br>The register value can be read by specifying the register to be read by using reg_flag before executing the PPG_CMD_REG_READ command, and then referencing the member of ppg_reg_t after executing the command. | | |
| Parameters | ppg_reg_t | uint32_t reg_flag | Use any of the following parameters to specify the register to read:<br><br>• PPG_PTRSLR_REG: Specifies the PTRSLR register.<br>• PPG_NDER_REG: Specifies the NDER register.<br>• PPG_PODR_REG: Specifies the PODR register.<br>• PPG_NDR_REG: Specifies the NDR register.<br>• PPG_PCR_REG: Specifies the PCR register.<br>• PPG_PMR_REG: Specifies the PMR register.<br>• PPG_ALL_REG: Specifies all registers.<br><br>Note: To specify multiple registers, separate parameters with OR. |
| Return values | PPG_SUCCESS | | : Success: The register is read successfully. |
| Remarks | For details about registers, see the RZ/T1 Group User's Manual: Hardware (R01UH0483EJ).<br><br>Example of reading a register<br>• When reading the PTRSLR and PODR registers<br>⇒ Use a parameter to specify the register you want to read for regdata.reg_flag of the third argument for the R_PPG_Control function. After the command is executed, the read value is stored in the regdata member.<br>regdata.reg_flag = PPG_PTRSLR_REG \| PPG_PODR_REG;<br>R_PPG_Control(handle, PPG_CMD_REG_READ, &regdata) | | |

### 6.12.2 PPG_CMD_REG_WRITE

| PPG_CMD_REG_WRITE | | | |
|---|---|---|---|
| Synopsis | Reading the PPG register | | |
| Header | r_ppg_rzt1_if.h | | |
| Description | Writes the value specified by the ppg_reg_t structure member to the register specified by reg_flag of the ppg_reg_t structure member.<br>Before executing the PPG_CMD_REG_WRITE command, specify the register to be written to reg_flag, and set the value to be written to the member of ppg_reg_t. By doing so, the value can be written to the register. | | |
| Parameters | ppg_reg_t | uint32_t reg_flag | Use any of any of the following parameters to specify the register to be written:<br><br>• PPG_PTRSLR_REG: Specifies the PTRSLR register.<br>• PPG_NDER_REG: Specifies the NDER register.<br>• PPG_PODR_REG: Specifies the PODR register.<br>• PPG_NDR_REG: Specifies the NDR register.<br>• PPG_PCR_REG: Specifies the PCR register.<br>• PPG_PMR_REG: Specifies the PMR register.<br>• PPG_ALL_REG: Specifies all registers.<br><br>Note: To specify multiple registers, separate parameters with OR. |
| | | uint8_t ptrslr_reg | Sets the value to be written to the PTRSLR register. |
| | | uint16_t nder_reg | Sets the value to be written to the NDERH/L register. |
| | | uint16_t podr_reg | Sets the value to be written to the PODRH/L register. |
| | | uint16_t ndr_reg | Sets the value to be written to the NDRH/L register. |
| | | uint8_t pcr_reg | Sets the value to be written to the PCR register. |
| | | uint8_t pmr_reg | Sets the value to be written to the PMR register. |
| Return values | PPG_SUCCESS | | : Success: The value is written to the register successfully. |
| Remarks | For details about registers, see the RZ/T1 Group User's Manual: Hardware (R01UH0483EJ).<br><br>Specifying the register to read or to be written<br>• When writing 0x01 to the PTRSLR register, and 0x0002 to the NDR register<br>⇒ Use a parameter to specify the register to be written for regdata.reg_flag of the third argument for the R_PPG_Control function, and set the value to write in the regdata member.<br>regdata.reg_flag = PPG_PTRSLR_REG \| PPG_NDR_REG;<br>regdata. tcr_reg = 0x01;<br>regdata. tmdr_reg = 0x0002;<br>R_PPG_Control(handle, PPG_CMD_REG_WRITE, &regdata) | | |

# 7. Sample Codes

Obtain the sample codes from the Renesas Electronics website.

# 8. Related Documents

- User's Manual: Hardware
  RZ/T1 Group User's Manual: Hardware
  (Obtain the latest version from the Renesas Electronics website.)

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

- Technical Update and Technical News
  (Obtain the latest information from the Renesas Electronics website.)

- User's Manual: Development Environment
  For details about the IAR integrated development environment (IAR Embedded Workbench® for Arm), visit the IAR website.
  (Obtain the latest version from the IAR website.)

## Website and Support

Renesas Electronics website

http://www.renesas.com/

Inquiries

http://www.renesas.com/inquiry

| Revision History | Application Note: TPUa/PPG Sample Program |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 0.10 | Apr. 02, 2015 | — | First Edition issued |
| 1.00 | Apr. 10, 2015 | — | Only the revision number was changed to be posted on a website. |
| 1.10 | Aug. 03, 2015 | 2. Operating Environment | |
| | | 5 | Table 2.1 Operating Environment: Description added to Integrated Development Environment |
| | | 6. Software | |
| | | 11 | 6.2.4 Required Memory Size: Description and reference added |
| | | 11 | Table 6.2: Table title and size description were partially amended |
| | | 11 | Table 6.2 Required Memory Size: Description on the Note and Size, changed |
| | | 12 | Table 6.3 added |
| | | 12 | Table 6.4 added |
| 1.20 | Dec. 04, 2015 | 2. Operating Environment | |
| | | 5 | Table 2.1 Operating Environment: Integrated Development Environment, information partially amended |
| 1.30 | Apr. 05, 2017 | 2. Operating Environment | |
| | | 5 | 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 | |
| | | 5 | Table 2.1 Operating Environment: The description on the integrated development environment, modified |
| | | 8. Related Documents | |
| | | 55 | The name of IAR Embedded Workbench, 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