# RL78/G23

## Encoder Signal Counting Function Using ELCL

### Introduction

In this application note, the circuit that outputs count pulses from quadrature encoder signals is designed using Logic & Event Link Controller (ELCL). The circuit for each of the clockwise and counterclockwise are configured and the signal generated by the ELCL is connected to the timer input. The motor speed is calculated based on the timer count value and the result is output on Tera Term via USB on the RL78/G23-64p Fast Prototyping Board.

### Target Device

| | |
|---|---|
| Evaluation Board | : RL78/G23-64p Fast Prototyping Board (hereafter FPB) |
| Motor Driver | : AE-DRV8835-S |
| DC motor with Quadrature Encoder | : 30:1 Metal Gearmotor 37Dx68L mm 12V with 64 CPR Encoder (Helical Pinion) |

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.

Contents

# 1. Specifications

This application note uses the ELCL to realize a counting function for quadrature encoder signals.

Figure 1-1 shows the configuration diagram of this system.

Figure 1-1 System Configuration Diagram

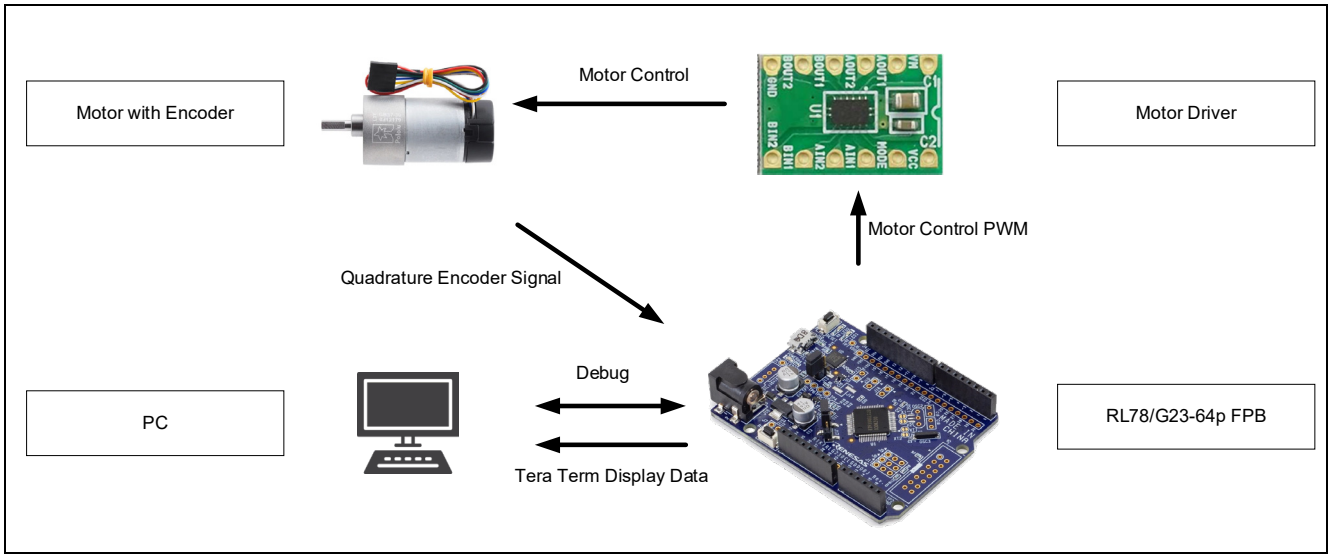
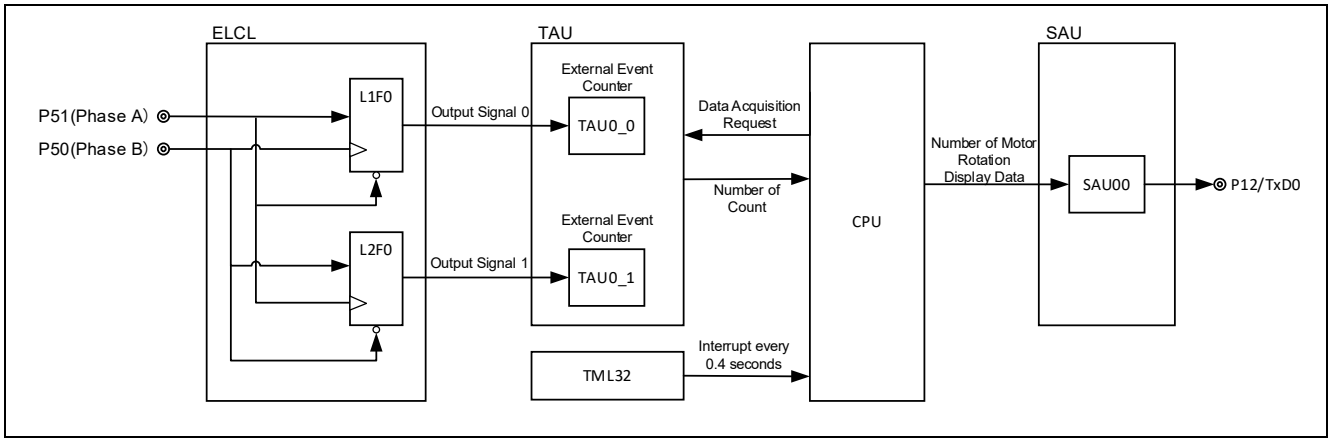


Figure 1-2 shows a configuration that uses ELCL to realize a counting function for quadrature encoder signals. Output of quadrature encoder (Phase A, Phase B) signals are taken into the ELCL by P51 and P50 and pass through two flip-flops (L1F0, L2F0) in the ELCL to determine clockwise and counterclockwise rotation of the motor. Output from L1F0 is a clockwise rotation signal (output signal 0). By counting the rising edge of this rotation signal using the external event counter function of the timer array unit (TAU), the rotation position and speed of the motor can be calculated. Similarly, by using the counterclockwise rotation signal (output signal 1), which is the output from L2F0, the rotational position and speed of the motor can be calculated.

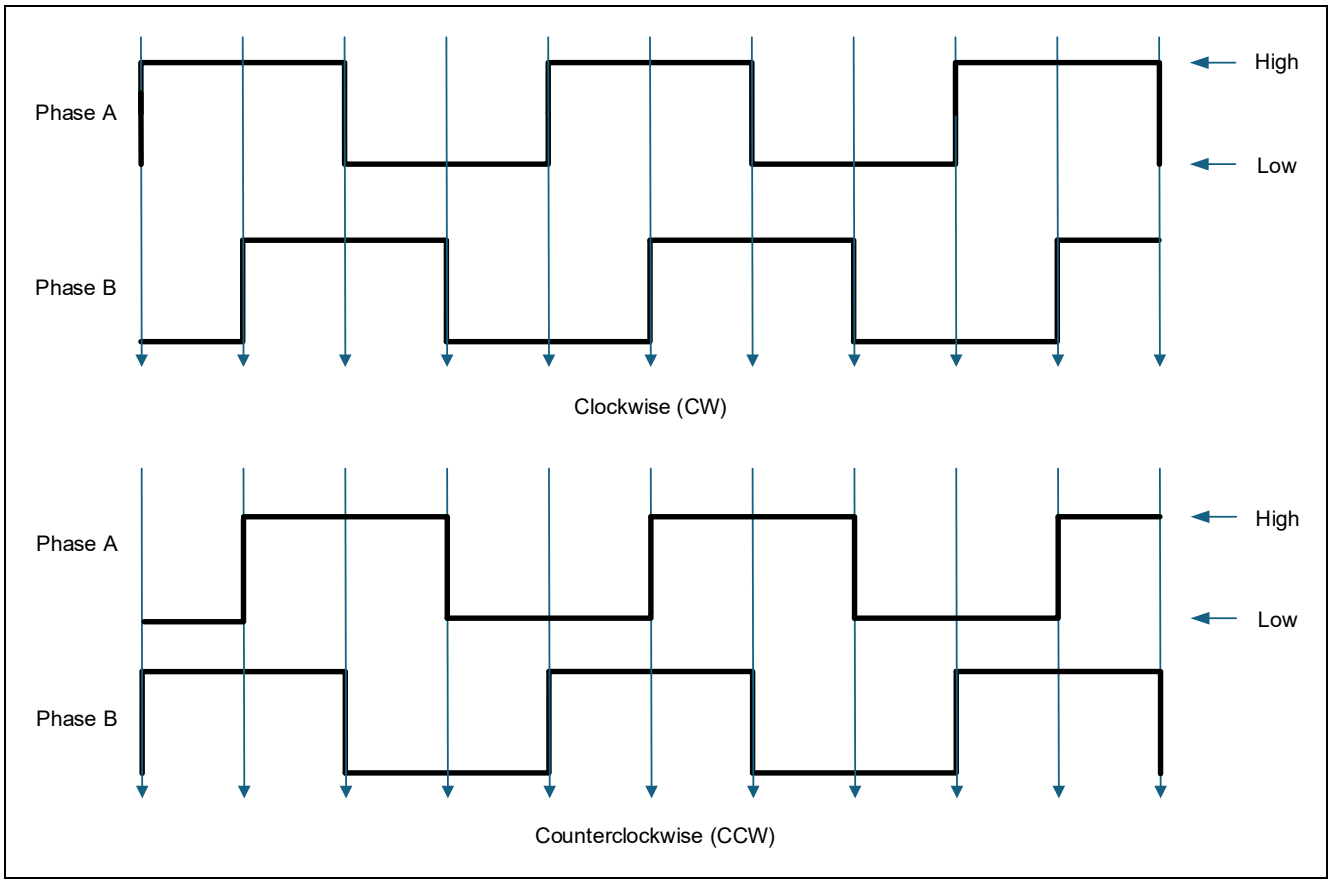
In this system, pressing the user switch on the board switches the motor rotation direction and stop, and the LED lighting pattern is also switched according to the mode. The output signal counts are converted to revolutions per minute, converted to Tera Term display data, and sent to the PC (Tera Term) via UART from the TxD0 terminal.

Figure 1-2 Configuration of a Counting Function For Quadrature Encoder Signals



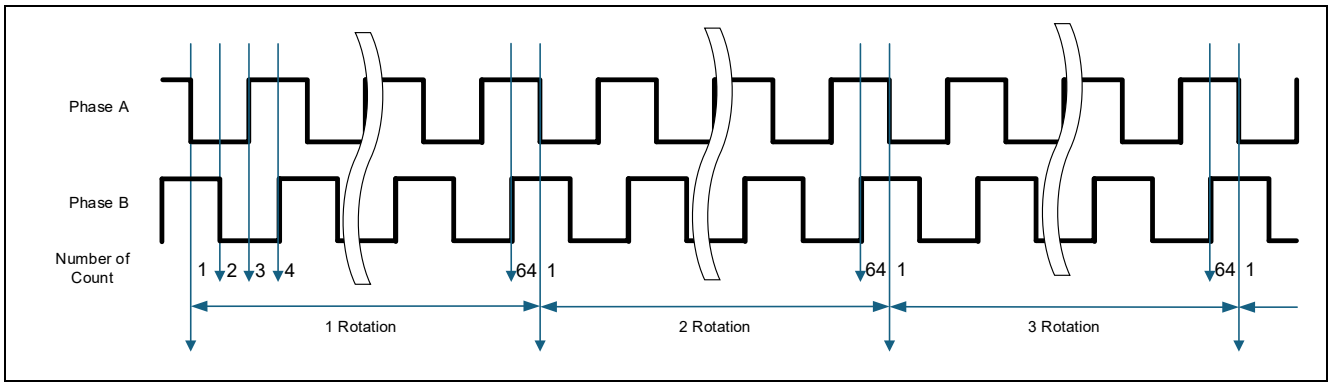
An encoder is an electronic component that detects the direction, amount, and angle of movement of various devices/equipment that rotate or move horizontally, and outputs electrical signals. A rotating disk with a rectangular hole (slit) is illuminated and the light passing through the slit is converted into a signal to output two square waves with a 90-degree phase difference. Figure 1-3 shows the waveforms of the quadrature encoder signals for clockwise and counterclockwise rotation.

Figure 1-3 Quadrature Encoder Signals



The encoder resolution used in this application note is 64. The resolution here is the number of counts of both edges of both phase A and phase B waveforms in one rotation. Figure 1-4 shows an example of counting in clockwise rotation.

Figure 1-4 Example of Counting



The timing chart for counting encoder signals using ELCL for clockwise rotation is shown in Figure 1-5, and for counterclockwise rotation in Figure 1-6. In this application note, counting is performed only at the rising edge of one of the waveforms, which is 1/4 of the resolution.

(1) Input the encoder signal phases A and B from P51 and P50 to ELCL.

(2) Output signal 0 is output from L1F0 and output signal 1 from L2F0.

Figure 1-5 Timing Chart of Clockwise Rotation

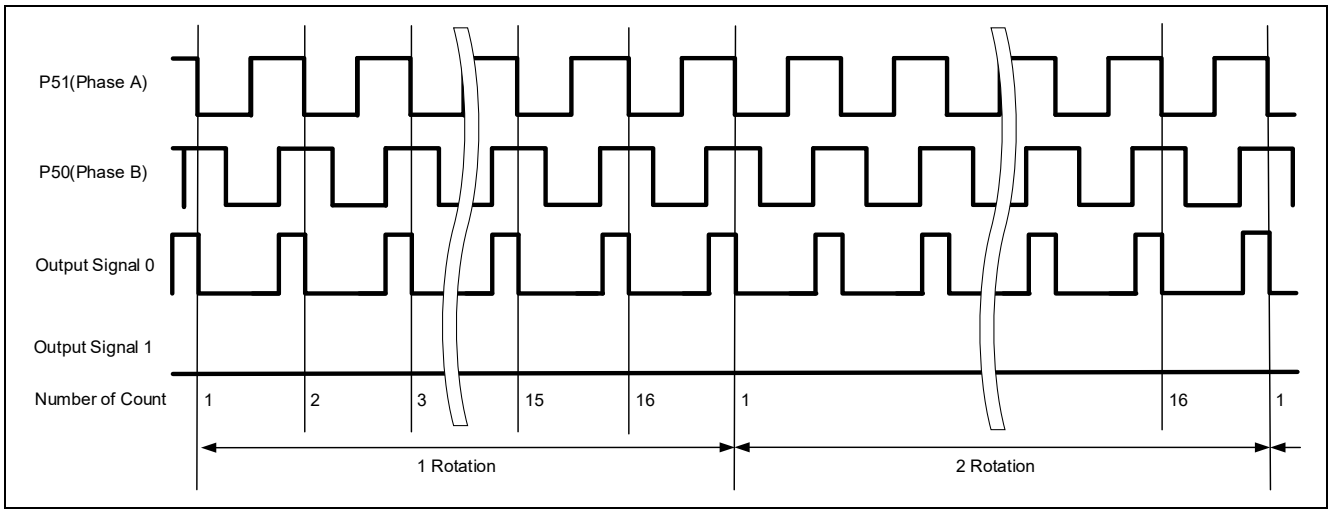
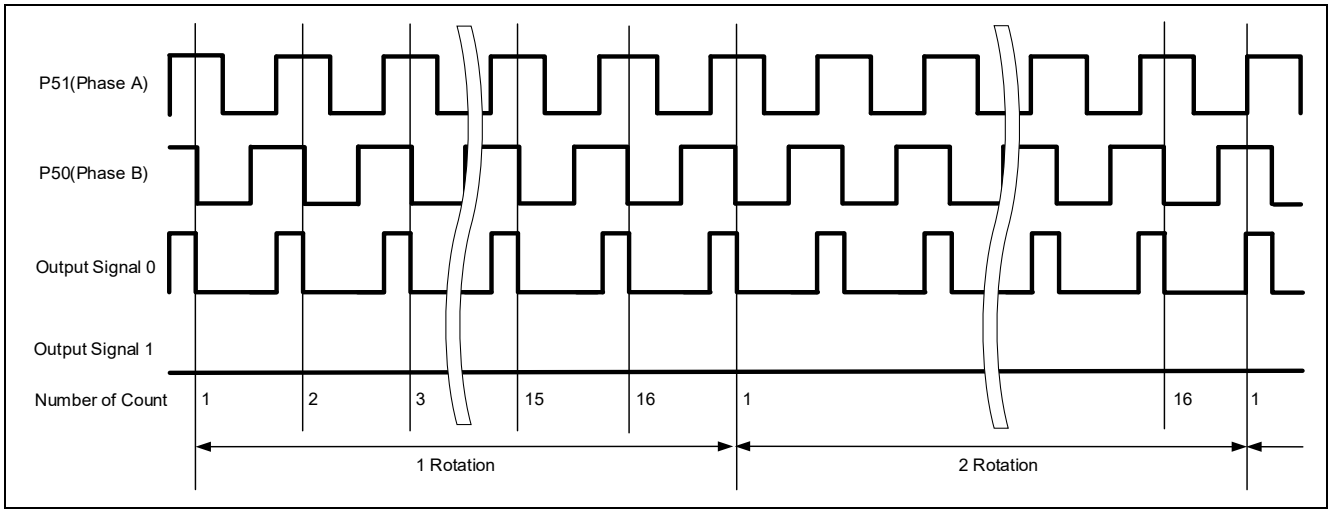



Figure 1-6 Timing Chart of Counterclockwise Rotation



## 2. Operation Confirmation Conditions

The sample code described in this application note has been confirmed under the following conditions.

Table 2-1 Operation Confirmation Conditions

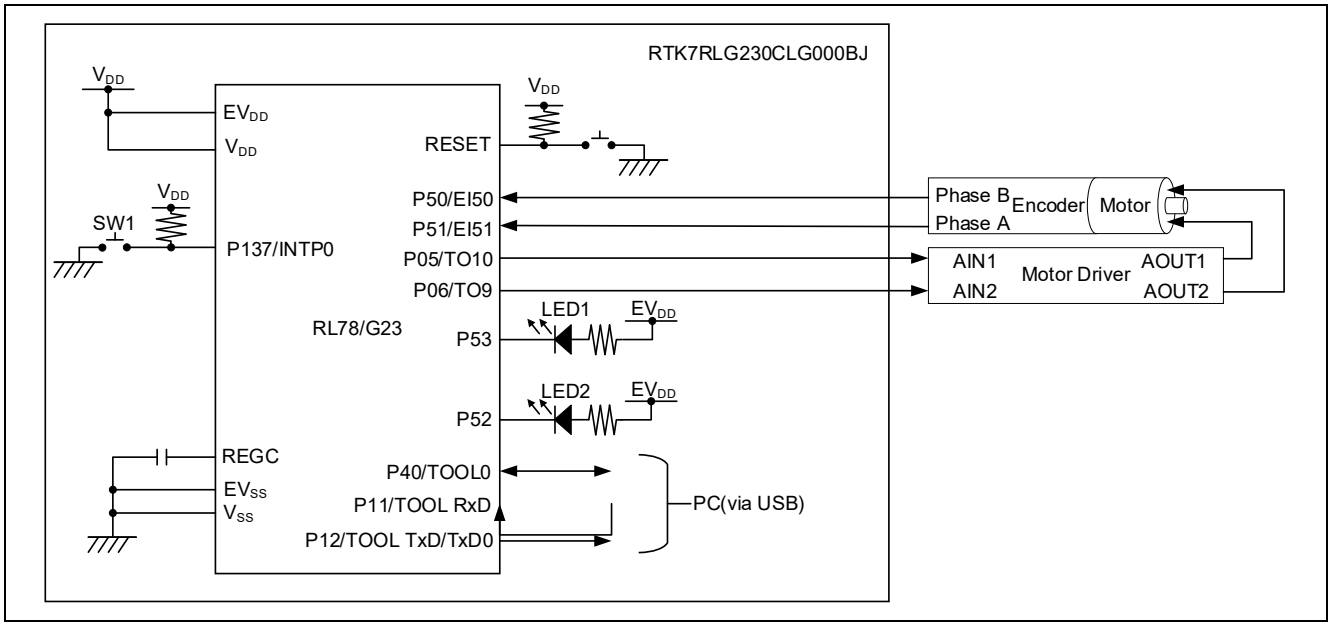
| Item | Descriptions |
|---|---|
| Board used | RL78/G23-64p Fast Prototyping Board |
| Operating frequency | • High-speed On-Chip Oscillator: 32MHz<br>• CPU/Peripheral Hardware Clock: 32MHz |
| Operating voltage | • 5.0V |
| Smart Configurator | v1.13.0 |
| Integrated development environment ($e^2$ studio) | $e^2$ studio 2025-04 (25.4.0) Manufactured by Renesas Electronics |
| C compiler ($e^2$ studio) | CC-RL V1.15.00 Manufactured by Renesas Electronics |
| Integrated development environment (IAR) | IAR Embedded Workbench for Renesas RL78 V 5.20.1 Manufactured by IAR Systems |
| C compiler (IAR) | |
| Integrated development environment (CS+) | CS+ V8.13.00 Manufactured by Renesas Electronics |
| Board Support Package (r_bsp) | V.1.90 |
| Emulator | $e^2$ studio: COM port<br>CS+: COM port<br>IAR: E2 Emulator Lite |
| Motor Driver | AE-DRV8835-S |
| DC motor with Quadrature Encoder | 30:1 Metal Gearmotor 37Dx68L mm 12V with<br>64 CPR Encoder (Helical Pinion) |

# 3. Hardware Description

## 3.1 Example of Hardware Configuration

Figure 3-1 shows an example of the hardware configuration.

Figure 3-1 Example of Hardware Configuration




Note 1. This simplified circuit diagram was created to show an overview of connections only. When designing your circuit, make sure the design includes appropriate pin handling and meets electrical characteristic requirements (connect each input-only port to $V_{DD}$ or $V_{SS}$ through a resistor).

Note 2. Connect any pins whose name begins with $EV_{SS}$ to $V_{SS}$, and any pins whose name begins with $EV_{DD}$ to $V_{DD}$, respectively.

Note 3. $V_{DD}$ must not be lower than the reset release voltage ($V_{LVD0}$) that is specified for the LVD0.

## 3.2 List of used Pins

Table 3-1 shows the pins used and their functions.

Table 3-1 Pins used and Their Functions

| Pin Name | I/O | Function |
|---|---|---|
| P05/TO10 | Output | Motor operation and speed control (High Active) |
| P06/TO9 | Output | Motor operation and speed control (High Active) |
| P51 | Input | ELCL input signal (Quadrature encoder signal phase A) |
| P50 | Input | ELCL input signal (Quadrature encoder signal phase B) |
| P137/INTP0 | Input | SW1 input |
| P52 | Output | LED2 lighting (Low Active) |
| P53 | Output | LED1 lighting (Low Active) |
| P12/TxD0 | Output | UART sending |

Caution: In this application note, only the used pins are processed. When designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.

# 4. Software Description

## 4.1 Operation Overview

This sample code generates count pulse signals for the P50 and P51 input signals using the following two ELCL modules. The generated signals are output to TAU0_0 and TAU0_1.

Figure 4-1 shows the count pulse signal generation system configuration.

Set P51 as the L1L0 input signal, P50 as the Clock input, and the negative logic of P51 as the Reset input. Output signal 0 to TAU0_0.

Set P50 as the L2L0 input signal, P51 as the Clock input, and the negative logic of P50 as the Reset input. Output signal 1 to TAU0_1.

Count the rising edges at TAU0_0 and TAU0_1, calculate the number of revolutions per minute from the count value, and display it on Tera Term via UART.

Figure 4-1 Encoder Signal Count Pulse Generation System Configuration

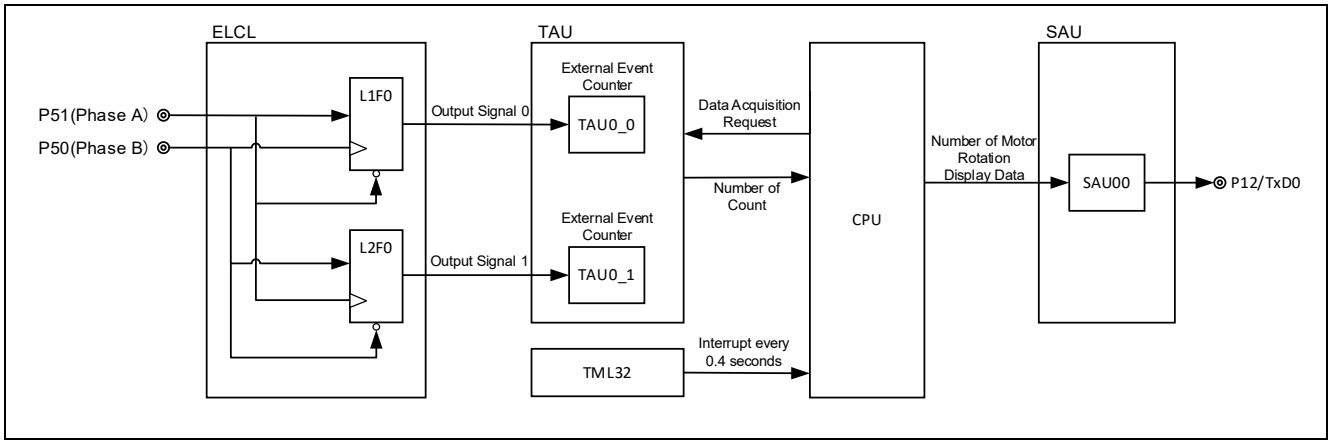
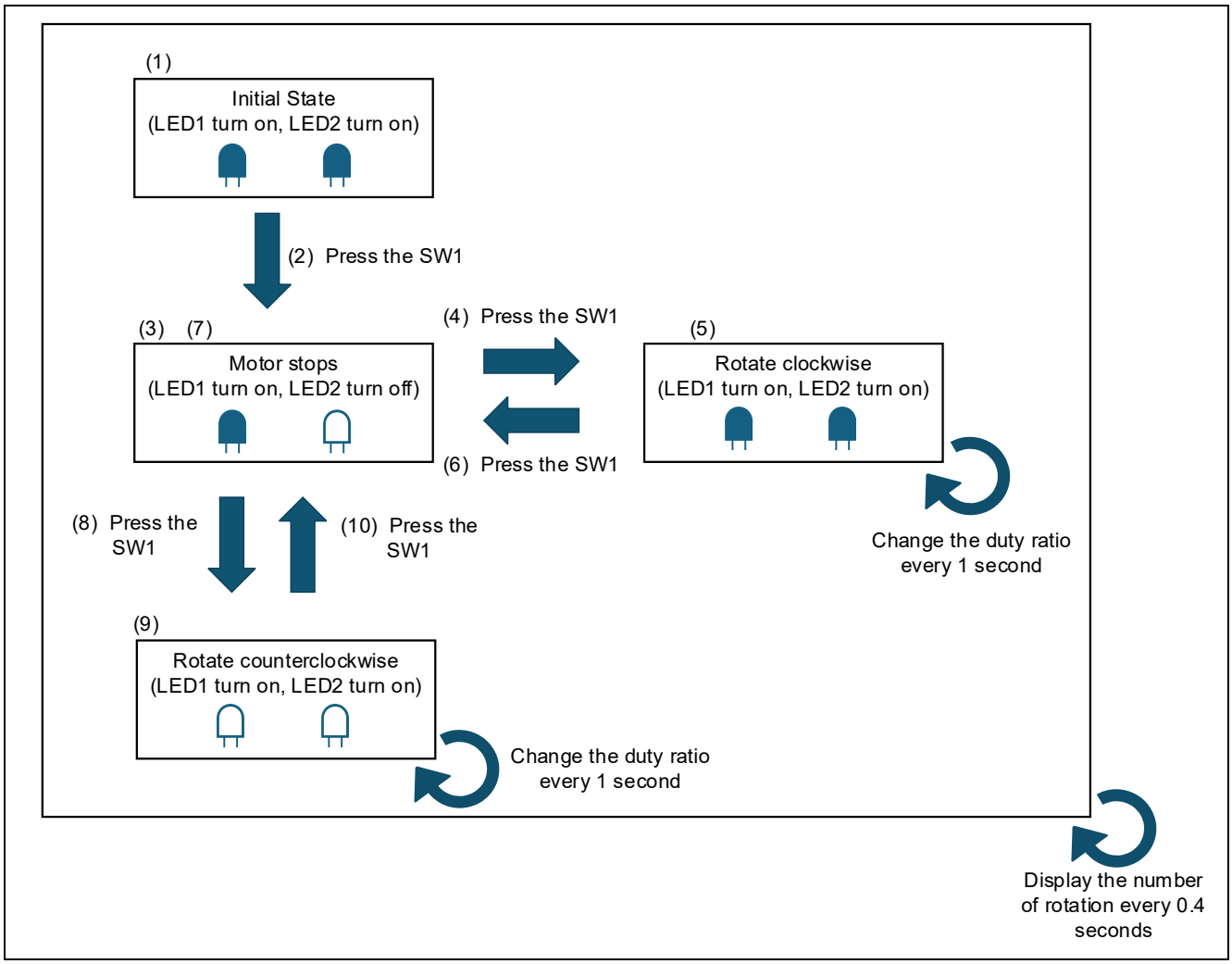


Figure 4-2 shows the operation overview.

(1) When the operation starts, LED1 and LED2 turn on.
(2) Press the SW1.
(3) Motor stops, LED1 turns on and LED2 turns off.
(4) Press the SW1.
(5) Motor rotates clockwise and LED1 and LED2 turn on.
(6) Press the SW1.
(7) Motor stops, LED1 turns on and LED2 turns off.
(8) Press the SW1.
(9) Motor rotates counterclockwise and LED1 and LED2 turn off.
(10) Move on (3).

Figure 4-2 Operation Overview



## 4.2 Folder Structure

Table 4-1, Table 4-2 shows the structure of the source files/header files used in the sample code.
Note that files automatically generated by the integrated development environment and files from the bsp environment are excluded.

Table 4-1 Folder Structure (1/2)

| Folder/File Name | Description | Generated by Smart Configurator |
|---|---|---|
| ¥CS+/e2studio/IARDIR> | Sample code folder | |
| ¥src<DIR> | Program storage folder | |
| main.c | Sample code source file | |
| main.h | Sample code header file | |
| ¥smc_gen<DIR> | Smart configurator generated folder | √ |
| ¥Config_ELCL<DIR> | ELCL program storage folder | √ |
| Config_ELCL.c | ELCL source file | √ |
| Config_ELCL.h | ELCL header file | √ |
| Config_ELCL_user.c | ELCL interrupt source file | √NOTE 1 |
| ¥Config_INTC<DIR> | INTC program storage folder | √ |
| Config_INTC.c | INTC source file | √ |
| Config_INTC.h | INTC header file | √ |
| Config_INTC_user.c | INTC interrupt source file | √ |
| ¥Config_ITL000_ITL001<DIR> | ITL000_ITL001 program storage folder | √ |
| Config_ ITL000_ITL001.c | ITL000_ITL001 source file | √ |
| Config_ ITL000_ITL001.h | ITL000_ITL001 header file | √ |
| Config_ ITL000_ITL001.c | ITL000_ITL001 interrupt source file | √ |
| ¥Config_PORT<DIR> | PORT program storage file | √ |
| Config_PORT.c | PORT source file | √ |
| Config_PORT.h | PORT header file | √ |
| Config_PORT_user.c | PORT interrupt source file | √NOTE 1 |

Table 4-2 Folder Structure (2/2)

| Folder/File Name | Description | Generated by Smart Configurator |
|---|---|---|
| ¥CS+/e2studio/IAR<DIR> | Sample code folder | |
| ¥src<DIR> | Program storage folder | |
| main.c | Sample code source file | |
| main.h | Sample code header file | |
| ¥smc_gen<DIR> | Smart configurator generated folder | √ |
| ¥Config_TAU0_0 <DIR> | TAU0_0 program storage folder | √ |
| Config_ TAU0_0.c | TAU0_0 source file | √ |
| Config_ TAU0_0.h | TAU0_0 header file | √ |
| Config_ TAU0_0_user.c | TAU0_0 interrupt source file | √ |
| ¥Config_ TAU0_1<DIR> | TAU0_1 program storage folder | √ |
| Config_ TAU0_1.c | TAU0_1 source file | √ |
| Config_ TAU0_1.h | TAU0_1 header file | √ |
| Config_ TAU0_1_user.c | TAU0_1 interrupt source file | √ |
| ¥Config_ TAU0_2<DIR> | TAU0_2 program storage folder | √ |
| Config_ TAU0_2.c | TAU0_2 source file | √ |
| Config_ TAU0_2.h | TAU0_2 header file | √ |
| Config_ TAU0_2_user.c | TAU0_2 interrupt source file | √ |
| ¥Config_UART0<DIR> | UART0 program storage folder | √ |
| Config_UART0.c | UART0 source file | √ |
| Config_UART0.h | UART0 header file | √ |
| Config_UART0_user.c | UART0 interrupt source file | √ |

Note “<DIR>” indicates a directory.
Note 1. This sample code does not use it.

## 4.3 List of Option Byte Settings

Table 4-3 shows the option byte settings.

Table 4-3 Option Byte Settings

| Address | Setting Value | Description |
|---|---|---|
| 000C0H/040C0H | 1110 1111B (EFH) | Watchdog Timer stopped operation<br>(Count stops after reset release) |
| 000C1H/040C1H | 0011 1010B (3AH) | LVD0 Off |
| 000C2H/040C2H | 1110 1000B (E8H) | Flash operation mode: High-speed main mode<br>High-speed on-chip oscillator frequency: 32MHz |
| 000C3H/040C3H | 1000 0100B (84H) | On-chip debug operation allowed |

## 4.4 List of Constants

Table 4-4 shows the list of constants used in the sample code.

Table 4-4 Constants used in the sample code

| Constant Name | Setting Value | Contents |
|---|---|---|
| SCALE_FACTOR | 0x00000064 | Scale correction value for overflow suppression |
| AIN1_HIGH | 0x20 | Motor control signal (AIN1 set to HIGH) |
| AIN2_HIGH | 0x40 | Motor control signal (AIN2 set to HIGH) |
| AIN1_LOW | 0xDF | Motor control signal (AIN1 set to LOW) |
| AIN2_LOW | 0xBF | Motor control signal (AIN2 set to LOW) |
| LED1_ON | 0xF7 | LED1 set to on |
| LED2_ON | 0xFB | LED2 set to on |
| LED1_OFF | 0x08 | LED1 set to off |
| LED2_OFF | 0x04 | LED2 set to off |
| BRAKE1 | 0 | Motor control mode (brake 1) |
| CW | 1 | Motor control mode (clockwise) |
| BRAKE2 | 2 | Motor control mode (brake 2) |
| CWW | 3 | Motor control mode (counterclockwise) |
| DUTY_10 | 0x1900 | Duty cycle for motor control (10%) |
| DUTY_90 | 0xE100 | Duty cycle for motor control (90%) |

## 4.5 List of Variables

Table 4-5 shows the list of global variables used in the sample code. However, variables generated by the Smart Configurator that have not been modified are excluded.

Table 4-5 Global Variables

| Type | Variable Name | Contents | Function that uses the variable |
|---|---|---|---|
| uint8_t | g_flag_itl_periodic | Flag for a certain period of the time elapsed in the interval timer | main<br>R_Config_ITL000_ITL001_Callback_Shared_Interrupt |
| uint8_t | g_flag_switch | Switch-pressure flag | main、__near r_Config_INTC_intp0_interrupt |
| uint8_t | g_flag_pwm_periodic | Flag for a certain period elapsed in the PWM function of the timer | main、__near r_config_TAU0_2_channel5_interrupt |
| MD_STATUS | g_uart0_tx_end | Flag indicating completion of UART0 transmission | main<br>r_Config_UART0_callback_sendend<br>__near<br>r_Config_UART0_interrupt_send |
| uint16_t | g_max_count | Maximum count | main、__near<br>R_Config_TAU0_0_Create_UserInit<br>R_Config_TAU0_1_Create_UserInit |

## 4.6 Function List

Table 4-6 shows the functions used in the sample code. However, functions generated by the Smart Configurator that have not been modified are excluded.

Table 4-6 Function List

| Function Name | Overview |
|---|---|
| main () | Main processing |
| calc_rpm () | Calculate the number of rotations based on the pulse count value |
| send_rpm () | UART transmission of calculated CW and CWW number of rotations |
| change_motor_action () | Change the state of the motor operation |
| set_motor_dutycycle () | Set the duty ratio of the motor |
| R_Config_ITL000_ITL001_Callback_Shared_Interrupt () | Interval timer interrupt |
| r_Config_INTC_intp0_interrupt () | Switch external interrupt |
| r_Config_TAU0_2_channel5_interrupt () | Timer interrupt of PWM function |
| r_Config_UART0_interrupt_send () | UART0transmit complete interrupt |

## 4.7 Function Specifications

The following describes the function specifications of the sample code.

[Function name] main

| | |
|---|---|
| Overview | Main processing |
| Headers | Stdint.h、 stdio.h、string.h、 r_smc_entry.h、 main.h |
| Declaration | int main (void); |
| Description | This function starts the operation of the ELCL, Interval timer, Timer and INTC.<br>It checks the flags and calls the required functions. |
| Arguments | None |
| Return values | None |
| Remarks | None |

[Function name] calc_rpm

| | |
|---|---|
| Overview | Calculate the rotational speed of the motor in the CW and CWW directions |
| Headers | Stdint.h、 stdio.h、string.h、 r_smc_entry.h、 main.h |
| Declaration | void calc_rpm(uint16_t * rpm_cw, uint16_t * rpm_cww); |
| Description | This function estimates the rotational speed per minute from a given variable |
| Arguments | rpm_cw, rpm_cww |
| Return values | None |
| Remarks | None |

[Function name] send_rpm

| | |
|---|---|
| Overview | UART transmission of calculated clockwise and counterclockwise rotation speeds |
| Headers | Stdint.h、 stdio.h、string.h、 r_smc_entry.h、 main.h |
| Declaration | void send_rpm(uint16_t rpm_cw, uint16_t rpm_cww); |
| Description | This function sends the specified variable via UART. |
| Arguments | rps_cw, rpm_cww |
| Return values | None |
| Remarks | None |

[Function name] change_motor_action

| | |
|---|---|
| Overview | Change the state of the motor operation |
| Headers | Stdint.h、 stdio.h、string.h、 r_smc_entry.h、 main.h |
| Declaration | void change_motor_action(void); |
| Description | This function changes the state of the motor operation. |
| Arguments | None |
| Return values | None |
| Remarks | None |

[Function name] set_motor_dutycycle

| | |
|---|---|
| Overview | Set the duty ratio of the motor |
| Headers | Stdint.h、 stdio.h、string.h、 r_smc_entry.h、 main.h |
| Declaration | void set_motor_dutycycle(void); |
| Description | This function sets the duty ratio of the motor. |
| Arguments | None |
| Return values | None |
| Remarks | None |

[Function name] R_Config_ITL000_ITL001_Callback_Shared_Interrupt

| | |
|---|---|
| Overview | Common interrupt processing for ITL000 and ITL001 channels |
| Headers | r_cg_macrodriver.h、r_cg_userdefine.h、Config_ITL000_ITL001.h |
| Declaration | void R_Config_ITL000_ITL001_Callback_Shared_Interrupt(void); |
| Description | This function clears the compare match detection flag to ”0”.<br>It sets the interval timer flag to ”1” when a certain period time has elapsed. |
| Arguments | None |
| Return values | None |
| Remarks | None |

[Function name] r_Config_INTC_intp0_interrupt

| | |
|---|---|
| Overview | Interrupt service routine (ISR) for INTP0 (external interrupt 0) |
| Headers | r_cg_macrodriver.h、r_cg_userdefine.h、Config_INTC.h |
| Declaration | static void __near r_Config_INTC_intp0_interrupt(void); |
| Description | This function sets the switch-pressure flag to ”1” by pressing down the switch. |
| Arguments | None |
| Return values | None |
| Remarks | None |

| [Function name] r_Config_TAU0_2_channel5_interrupt | |
|---|---|
| Overview | Interrupt service routine (ISR) for INTTM05 (channel 5) |
| Headers | r_cg_macrodriver.h、r_cg_userdefine.h、Config_TAU0_2.h |
| Declaration | static void __near r_Config_TAU0_2_channel5_interrupt(void); |
| Description | This function sets the flag for a certain period elapsed in the PWM function of the timer to “1”. |
| Arguments | None |
| Return values | None |
| Remarks | None |

| [Function name] r_Config_UART0_interrupt_send | |
|---|---|
| Overview | Transmission interrupt service routine (ISR) for UART0 |
| Headers | r_cg_macrodriver.h、r_cg_userdefine.h、Config_UART0.h |
| Declaration | static void __near r_Config_UART0_interrupt_send(void); |
| Description | This function sets the transmission completion flag to “1”. |
| Arguments | None |
| Return values | None |
| Remarks | None |

## 4.8 Flowchart

### 4.8.1 Main Process

Figure 4-3, Figure 4-4 shows the flowchart for the main process.

Figure 4-3 Main Process (1/2)

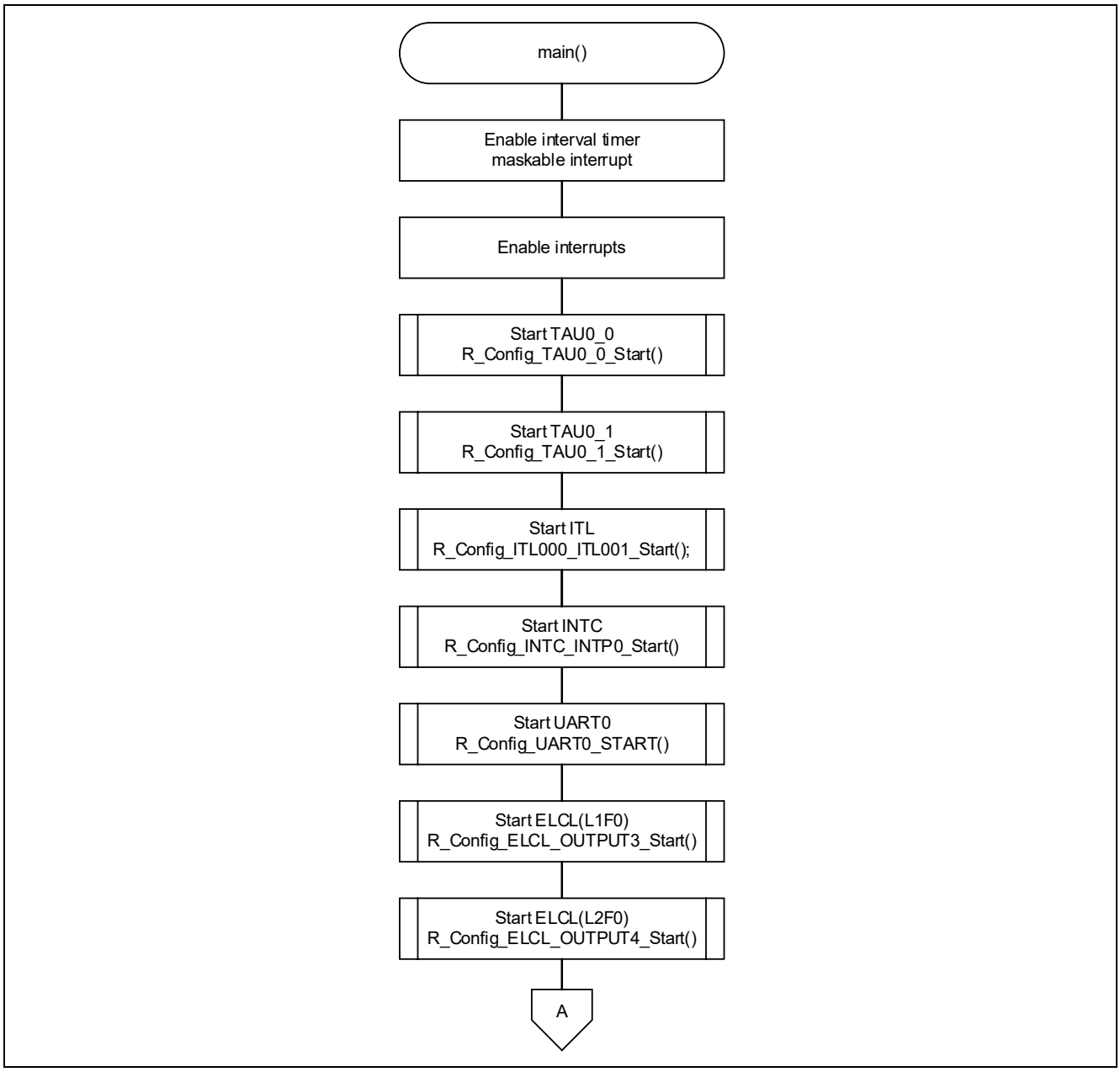
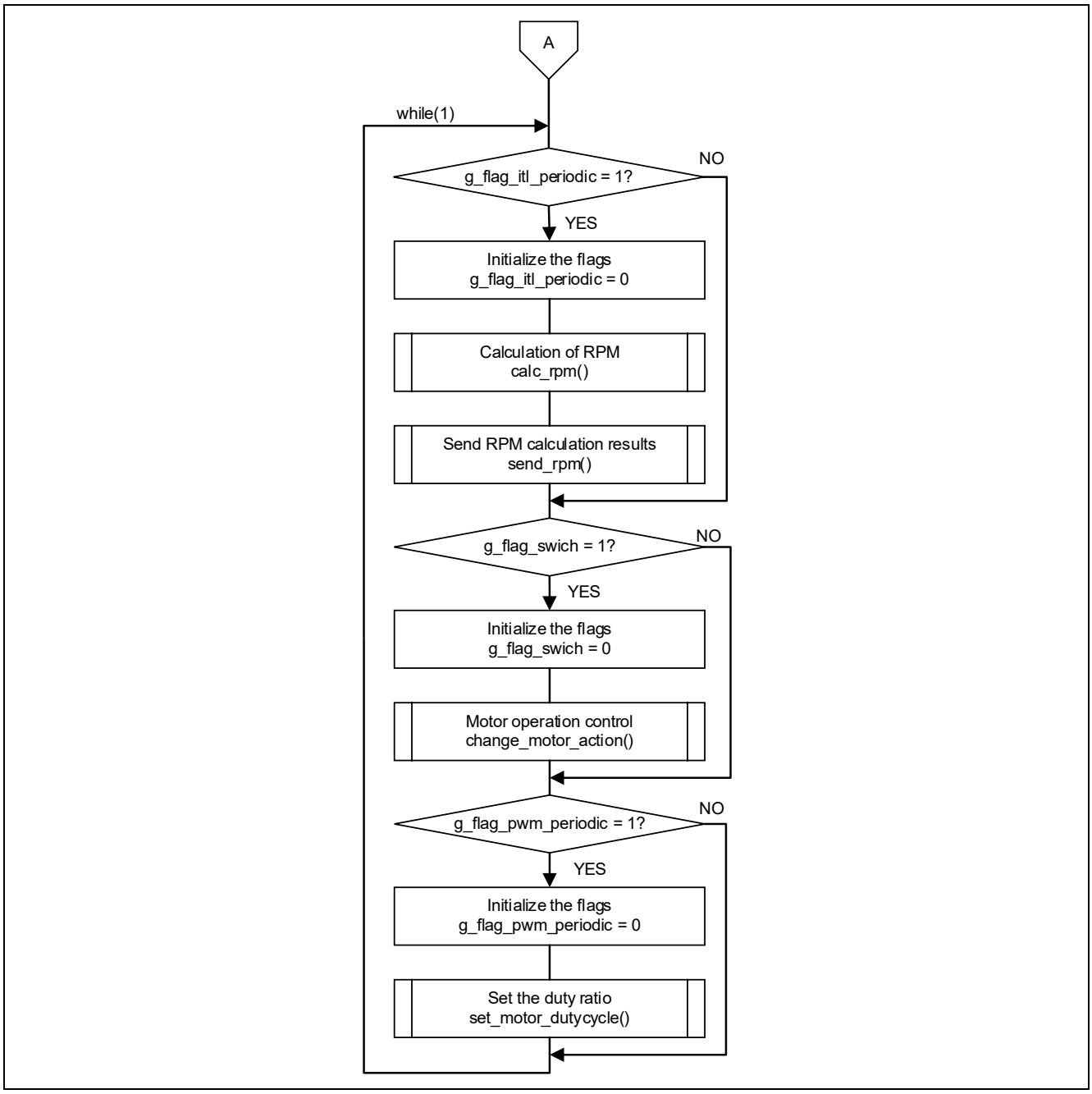


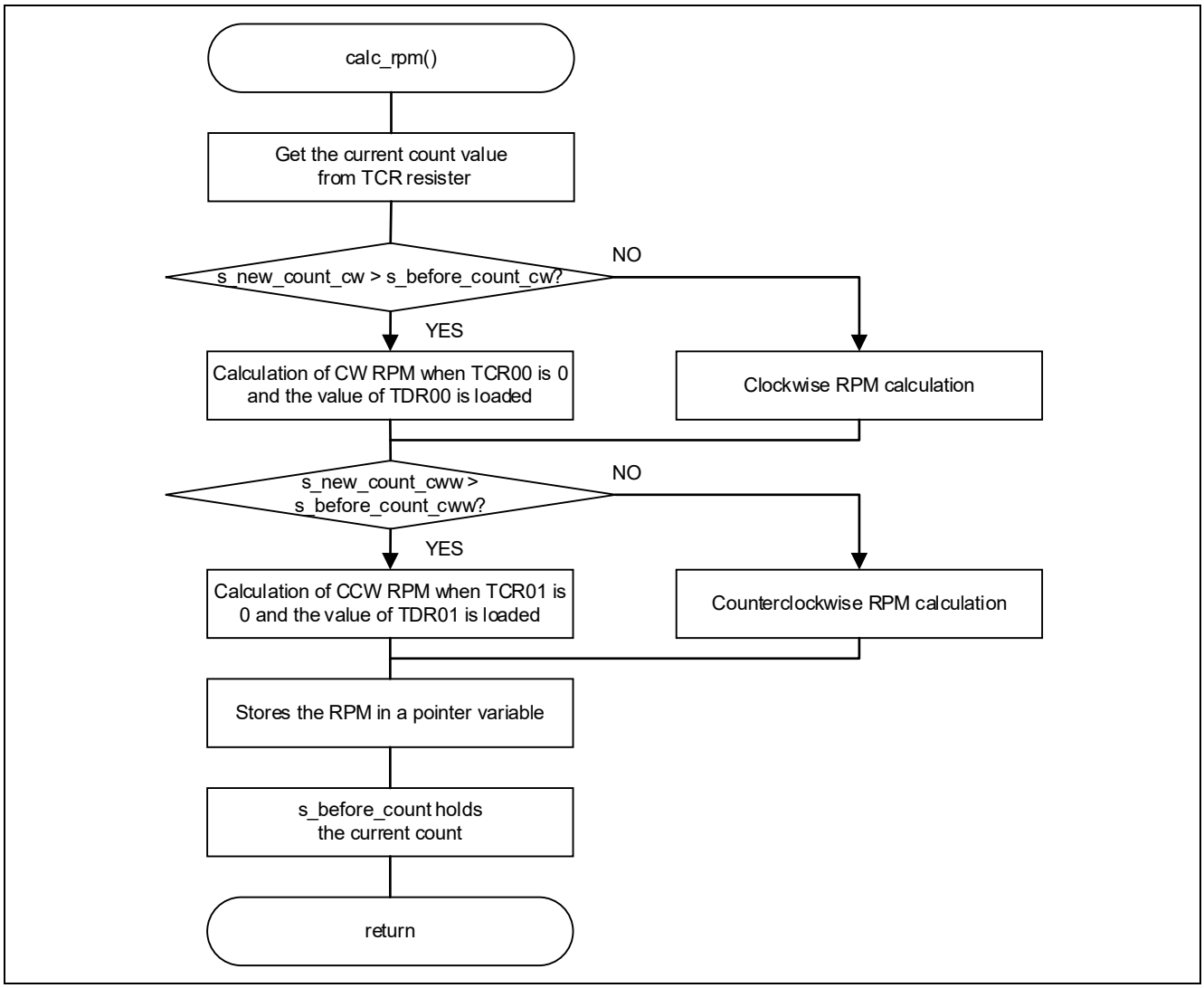
Figure 4-4 Main Process (2/2)



### 4.8.2 Approximate Number of Revolutions

Figure 4-5 shows the flowchart of the approximate number of revolutions.

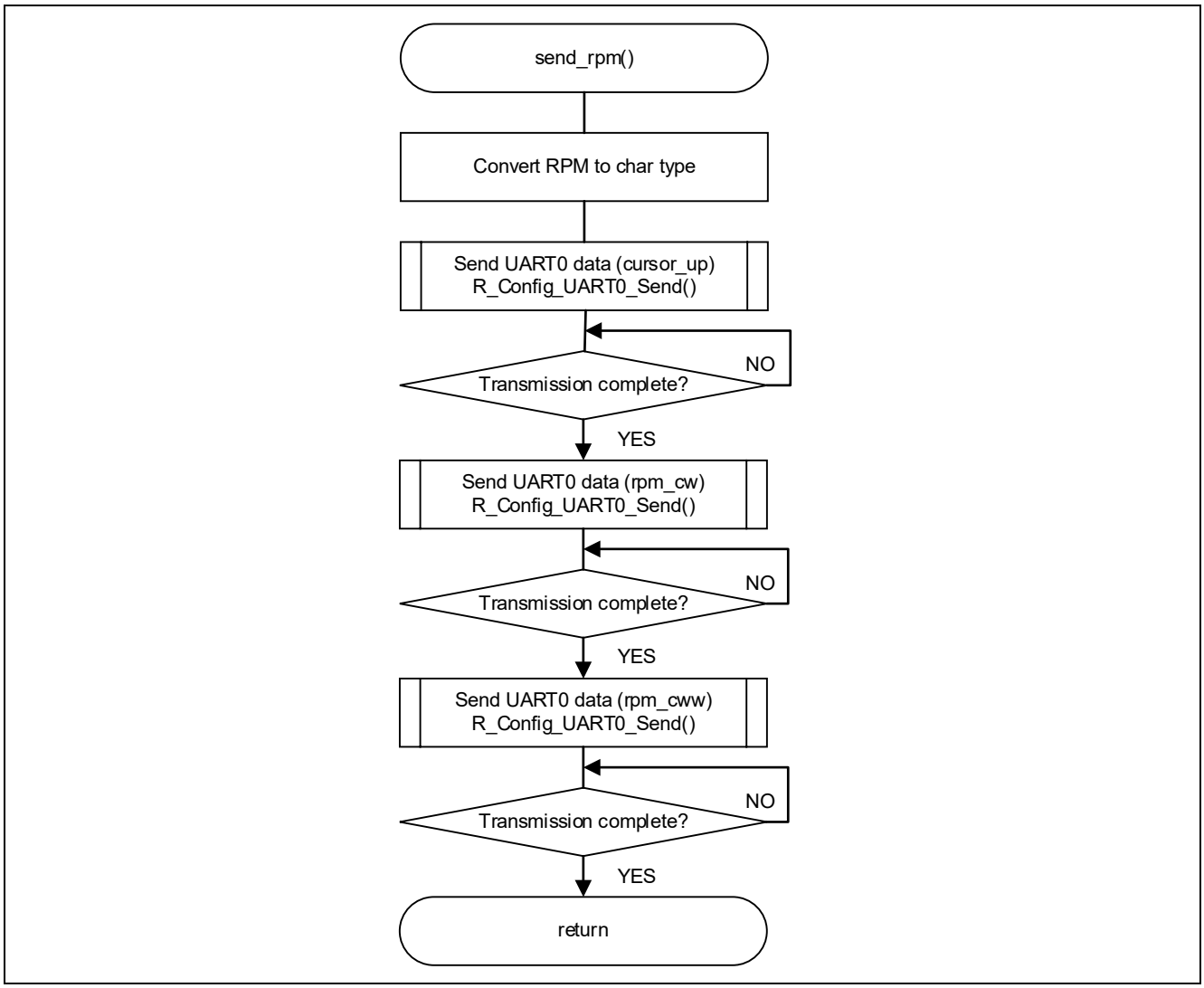
Figure 4-5 Approximate Number of Revolutions



### 4.8.3 Transmit Number of Rotations

Figure 4-6 shows the flowchart of transmitting the number of rotations.

Figure 4-6 Transmit Number of Rotations



### 4.8.4 Change the Motor Operation State

Figure 4-7, Figure 4-8, Figure 4-9, Figure 4-10, Figure 4-11 shows the flowchart of changing the motor operation state.

Figure 4-7 Change the Motor Operation State (1/5)

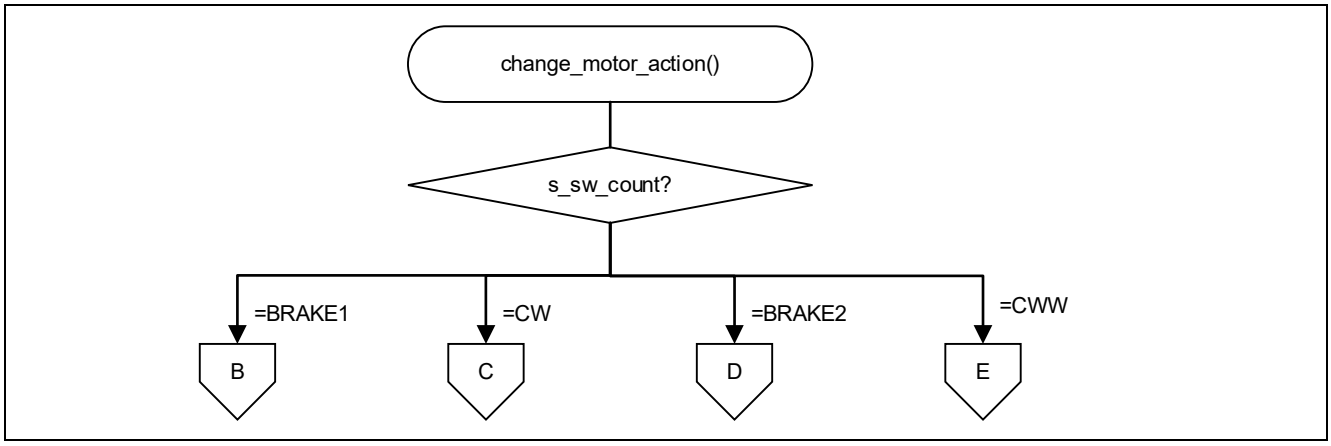



Figure 4-8 Change the Motor Operation State (2/5)

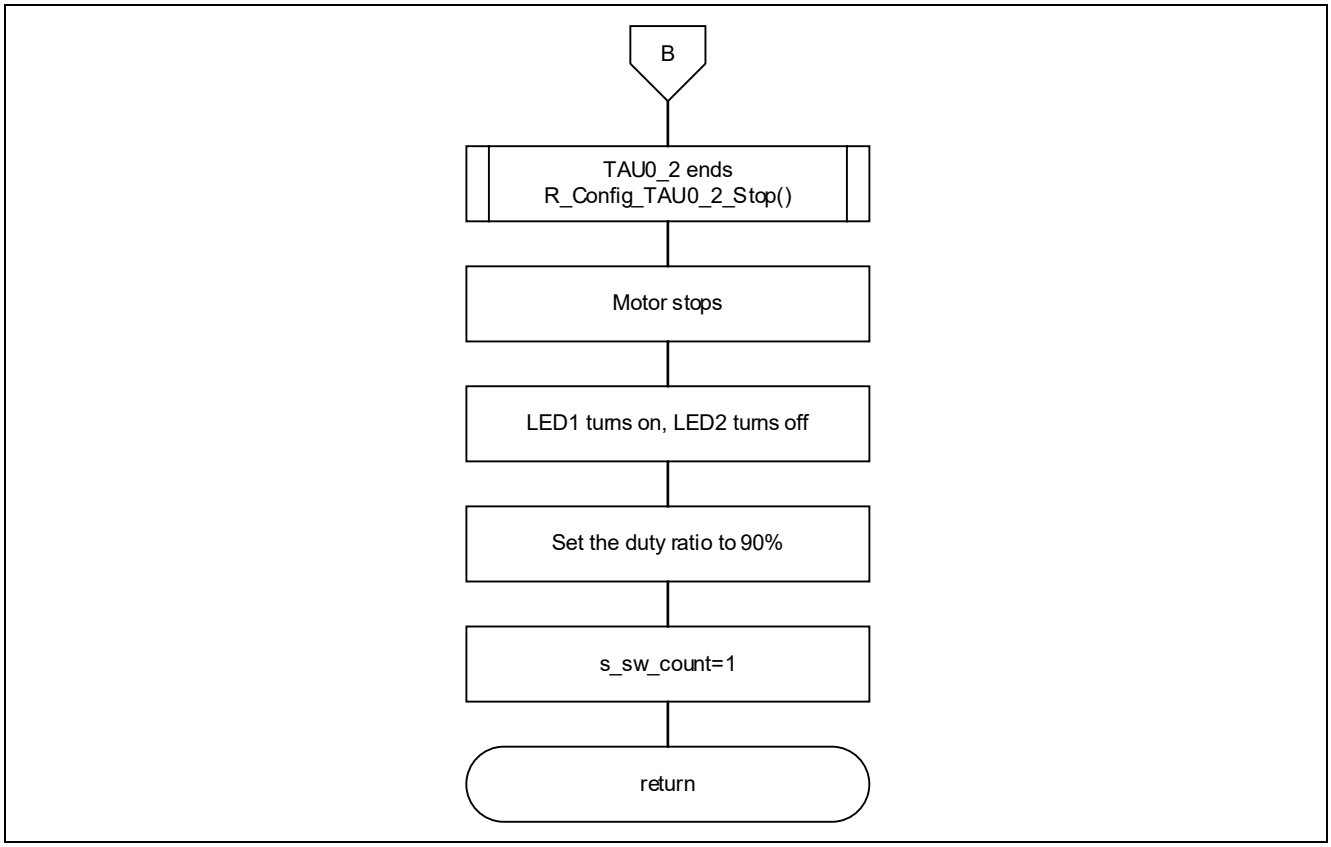


Figure 4-9 Change the Motor Operation State (3/5)

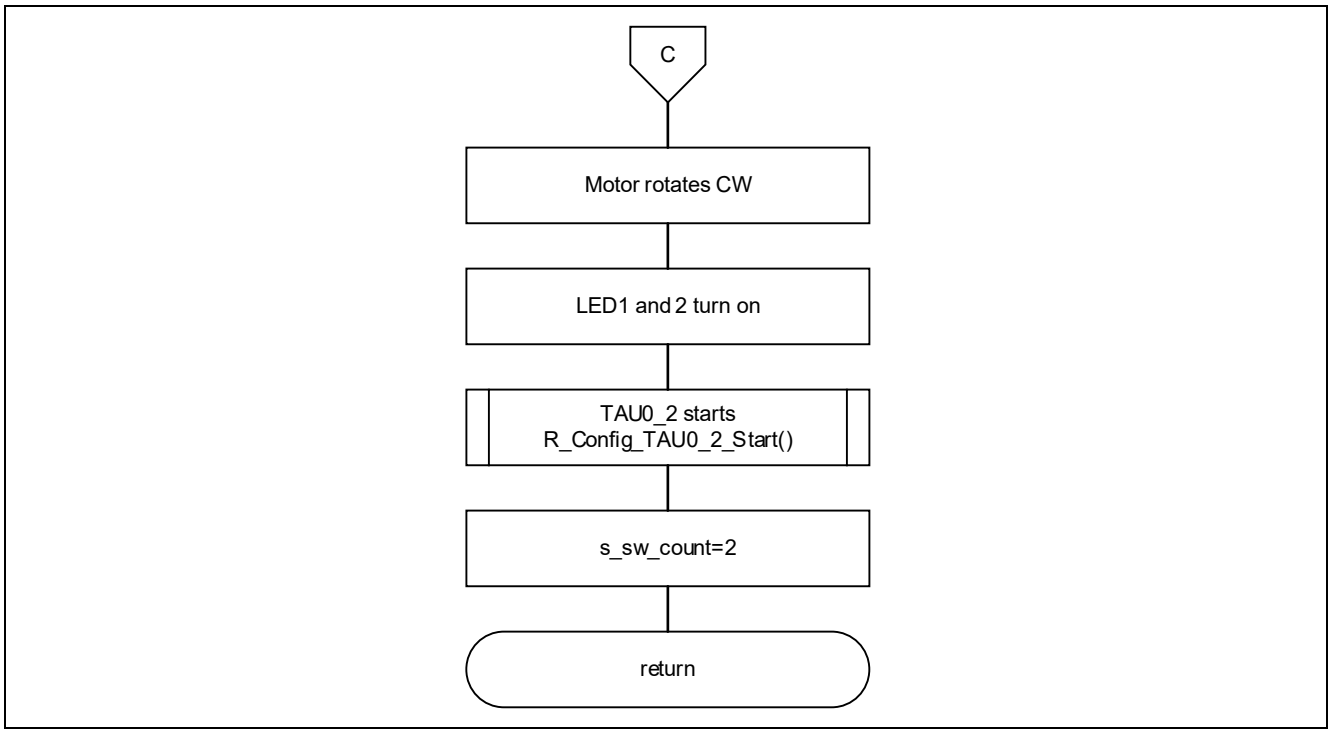



Figure 4-10 Change the Motor Operation State (4/5)

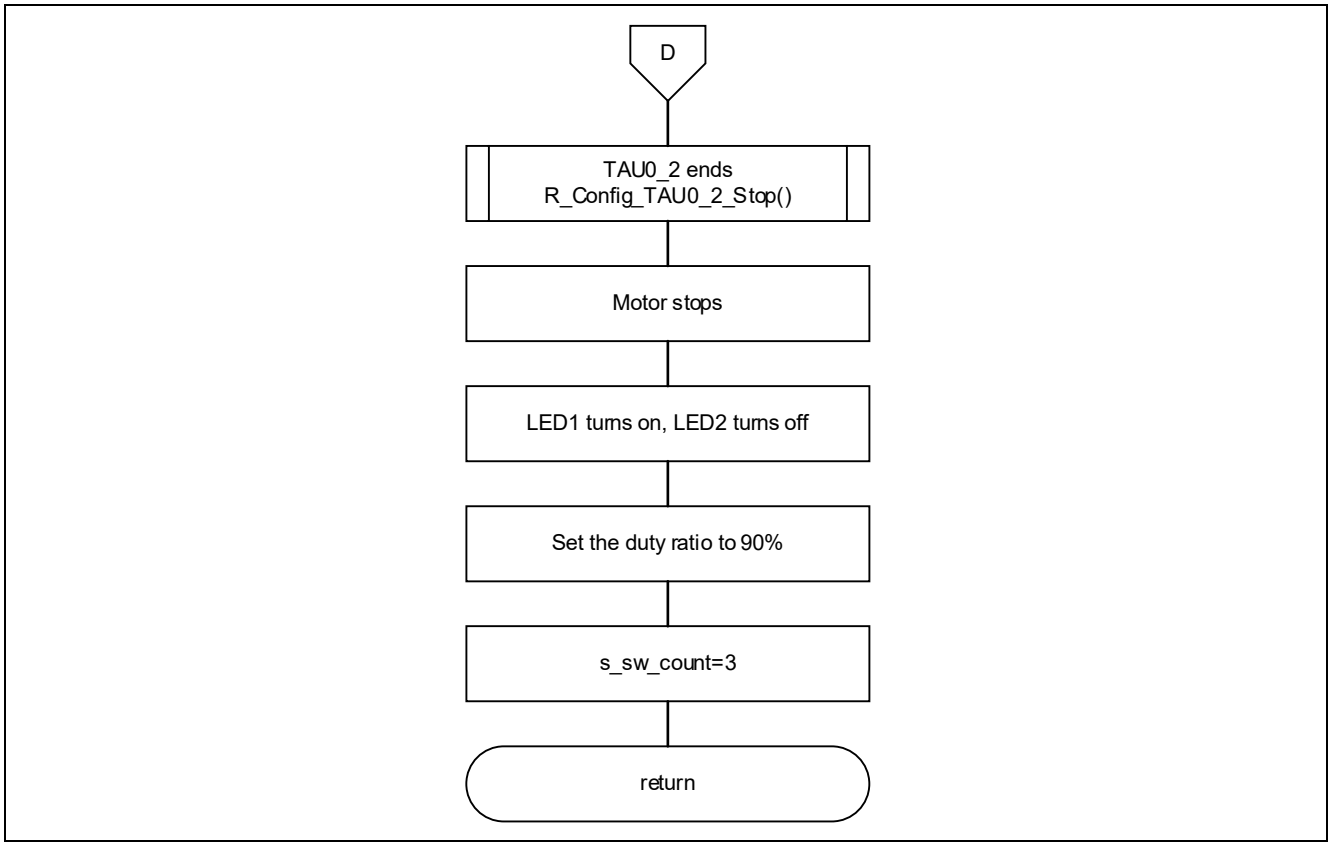
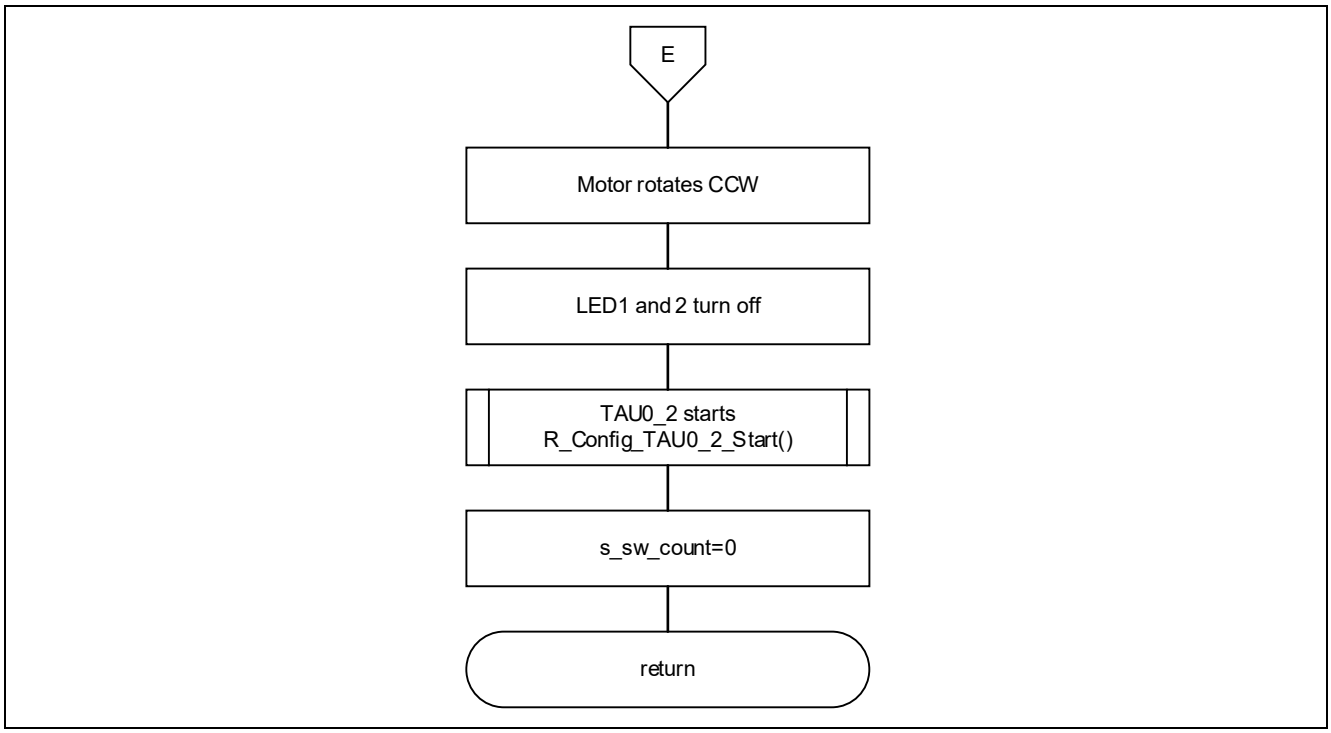


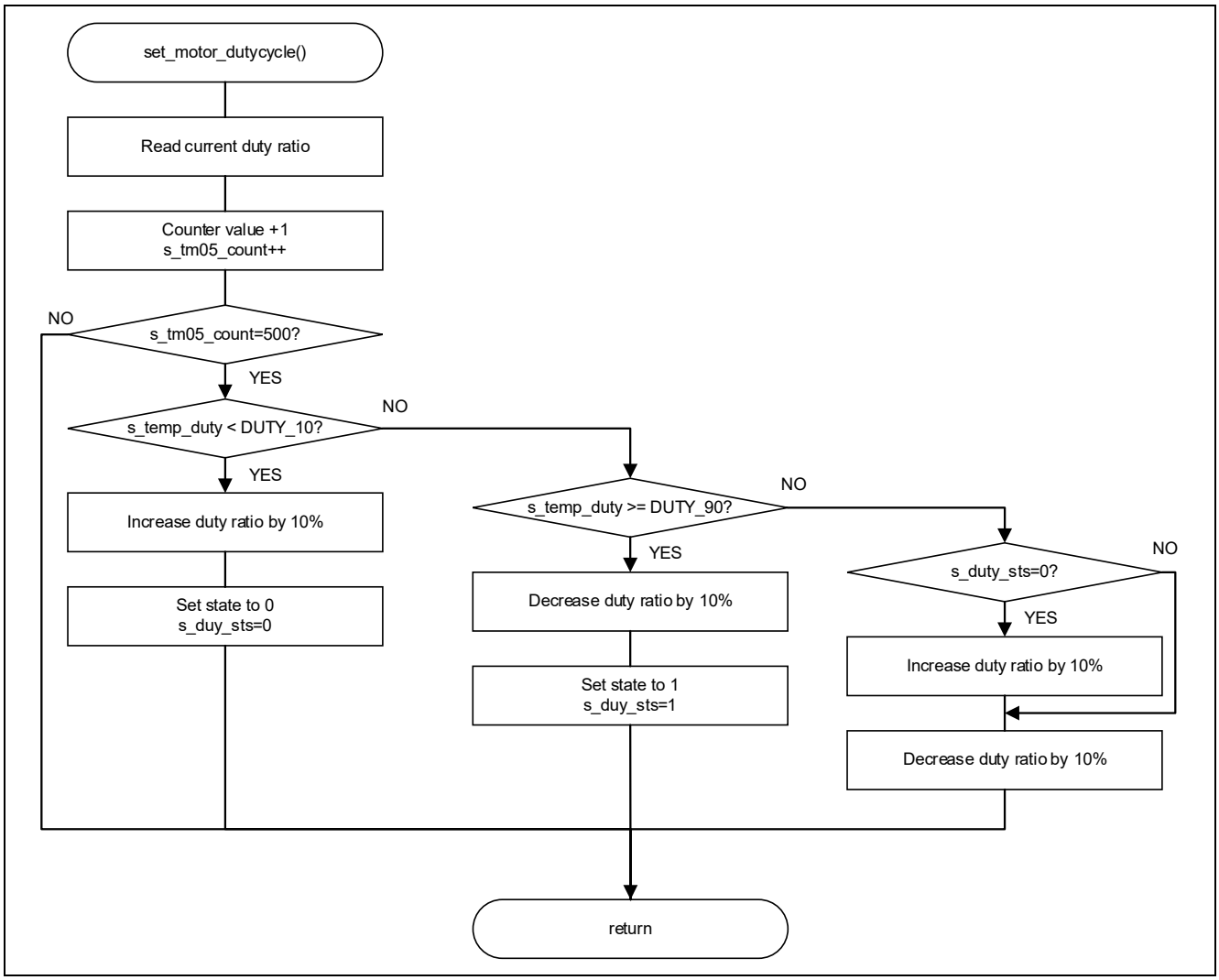
Figure 4-11 Change the Motor Operation State (5/5)



### 4.8.5 Set the Duty Ratio of the Motor

Figure 4-12 shows the flowchart of setting the duty ratio of the motor.

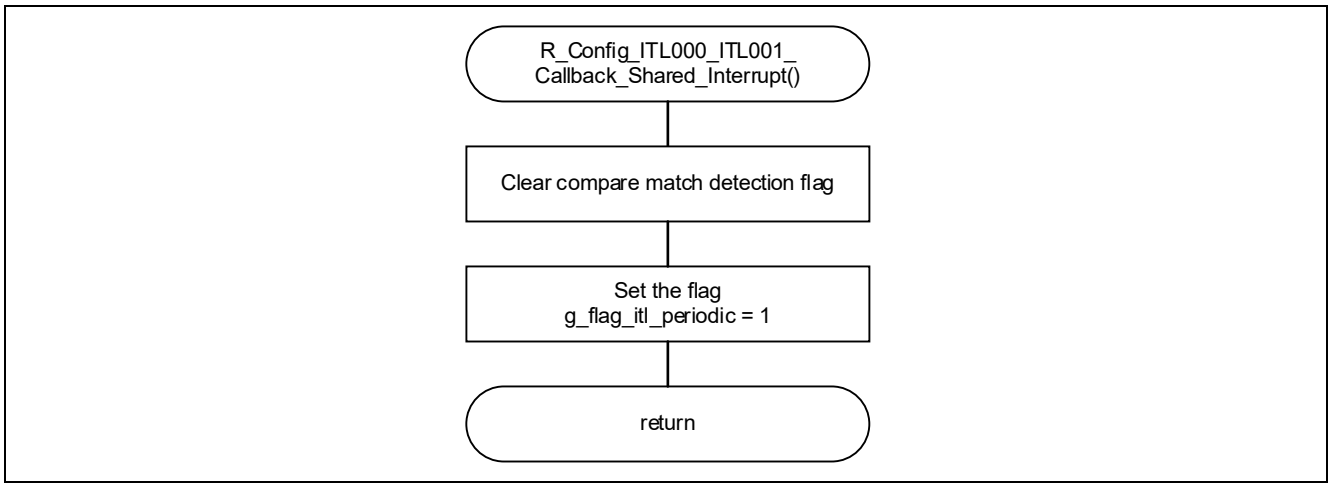
Figure 4-12 Set the Duty Ratio of the Motor



### 4.8.6 Interval Timer Interrupt

Figure 4-13 shows the flowchart of the interval timer interrupt.

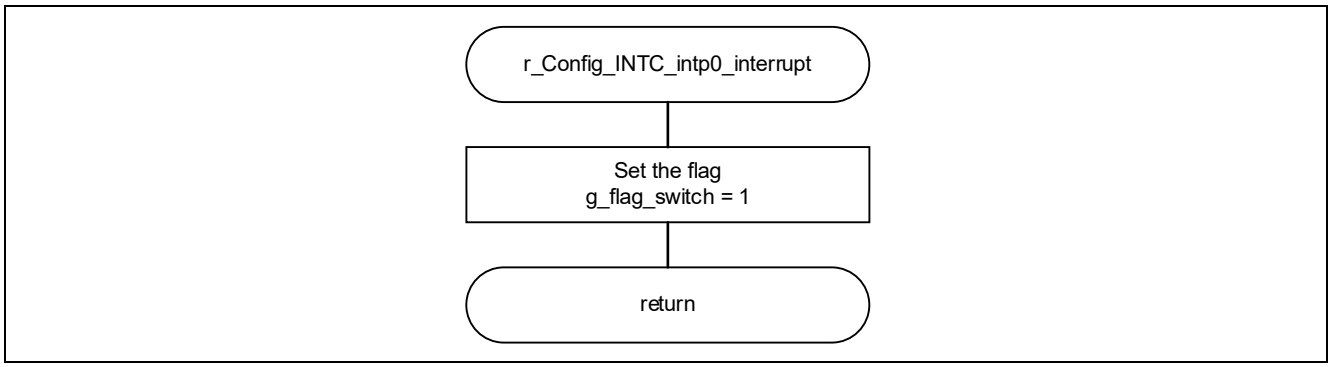
Figure 4-13 Interval Timer Interrupt




### 4.8.7 Switch Press Detection Interrupt

Figure 4-14 shows the flowchart of the switch press detection interrupt.

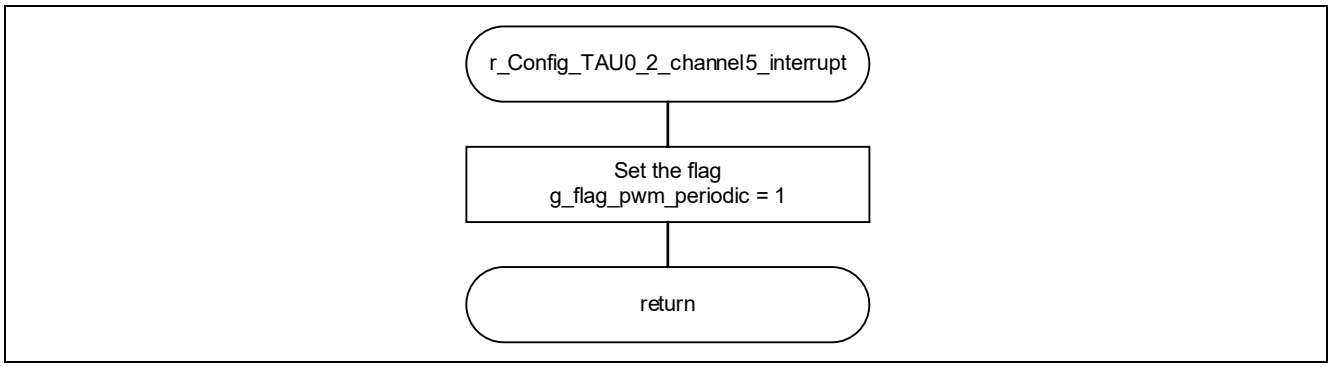
Figure 4-14 Switch Press Detection Interrupt



### 4.8.8 Timer Interrupt for PWM Functions

Figure 4-15 shows the flowchart of the timer interrupt for PWM functions.

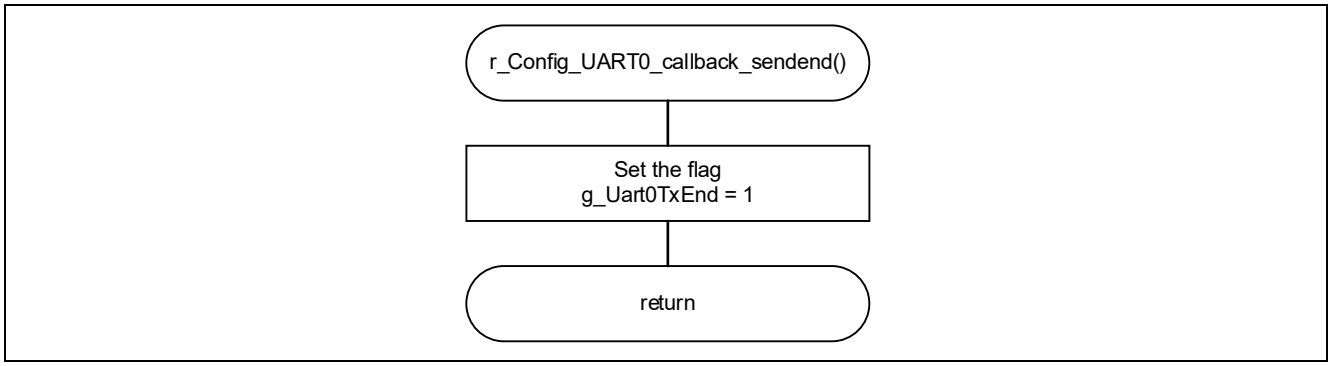
Figure 4-15 Timer Interrupt for PWM Functions




### 4.8.9 UART0 Transmission Complete

Figure 4-16 shows the flowchart of UART0 transmission complete.

Figure 4-16 UART0 Transmission Complete



## 4.9 Motor Control

Table 4-7 shows the pins and operation to control the motor.

Table 4-7 Pins and Operation to Control the Motor

| P05(AIN1) | P06(AIN2) | AOUT1 | AOUT2 | Operation |
|---|---|---|---|---|
| 0 | 0 | Hiz | HiZ | Motor racing |
| 0 | 1 | L | H | Backward rotation |
| 1 | 0 | H | L | Forward rotation |
| 1 | 1 | L | L | Brake |

# 5. Setting Up the Smart Configurator

This application note contains the following Smart Configurator configuration file in addition to the sample code.

r01an7635_elcl.scfg

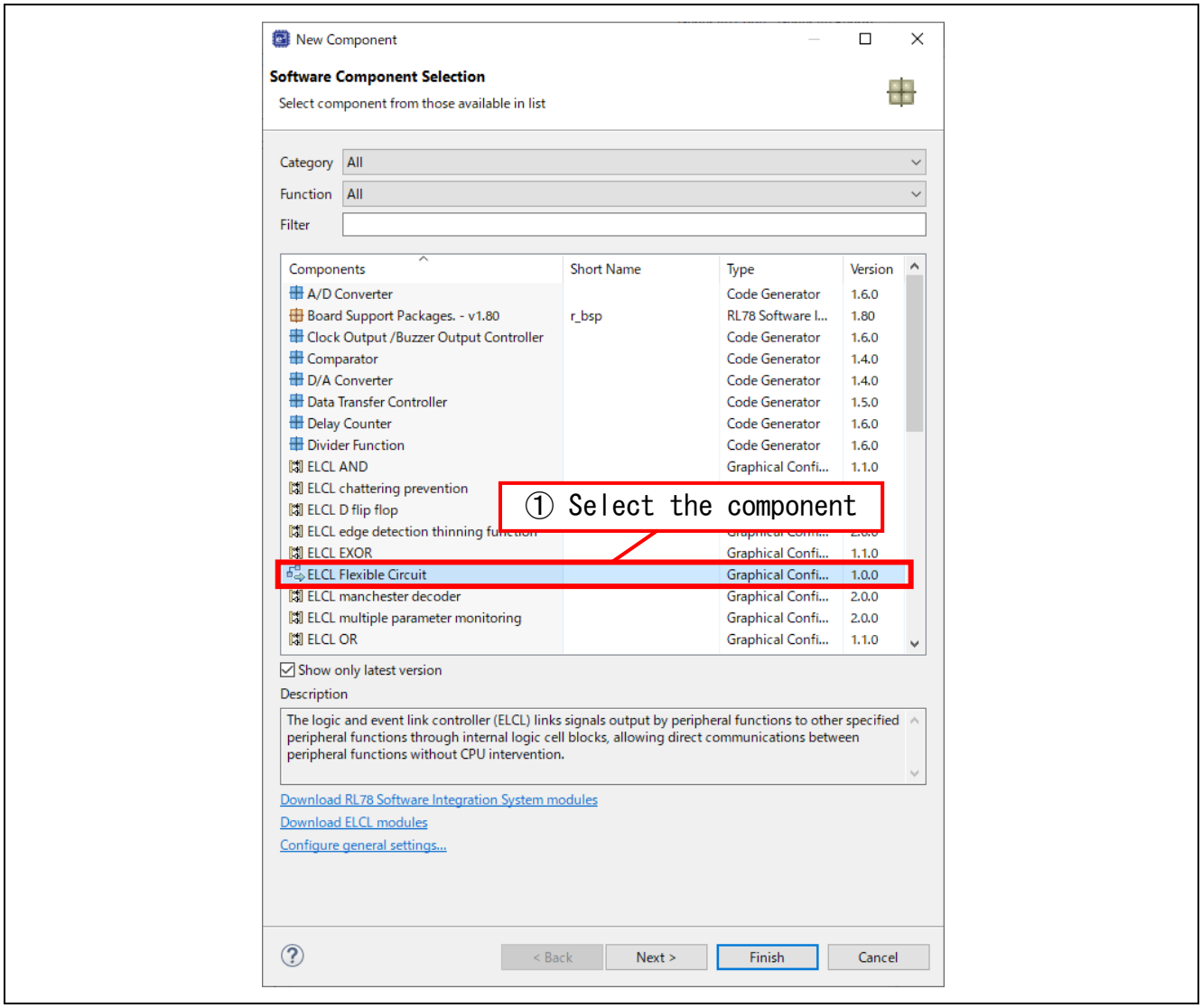
The following describes the file and provides setting examples and precautions for use.

## 5.1 Setting the ELCL Component

The following describes how to set the ELCL component.

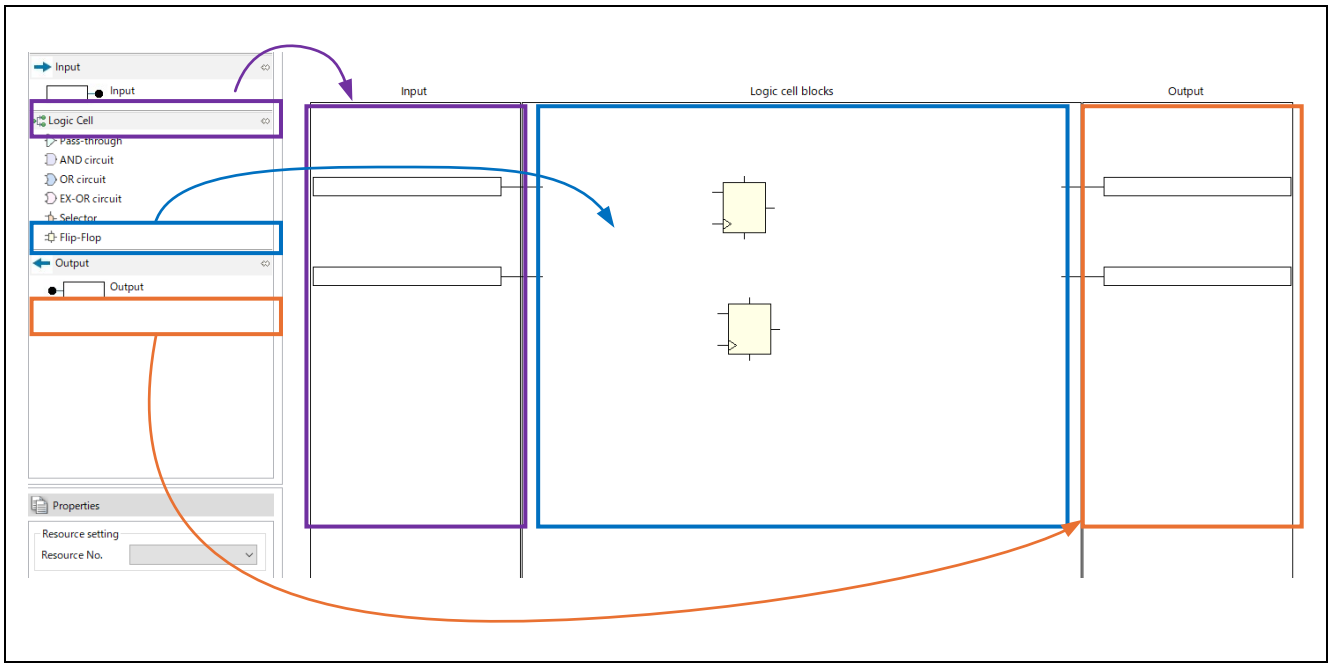
1. Select “ELCL Flexible Circuit”.

Figure 5-1 Selecting the Component



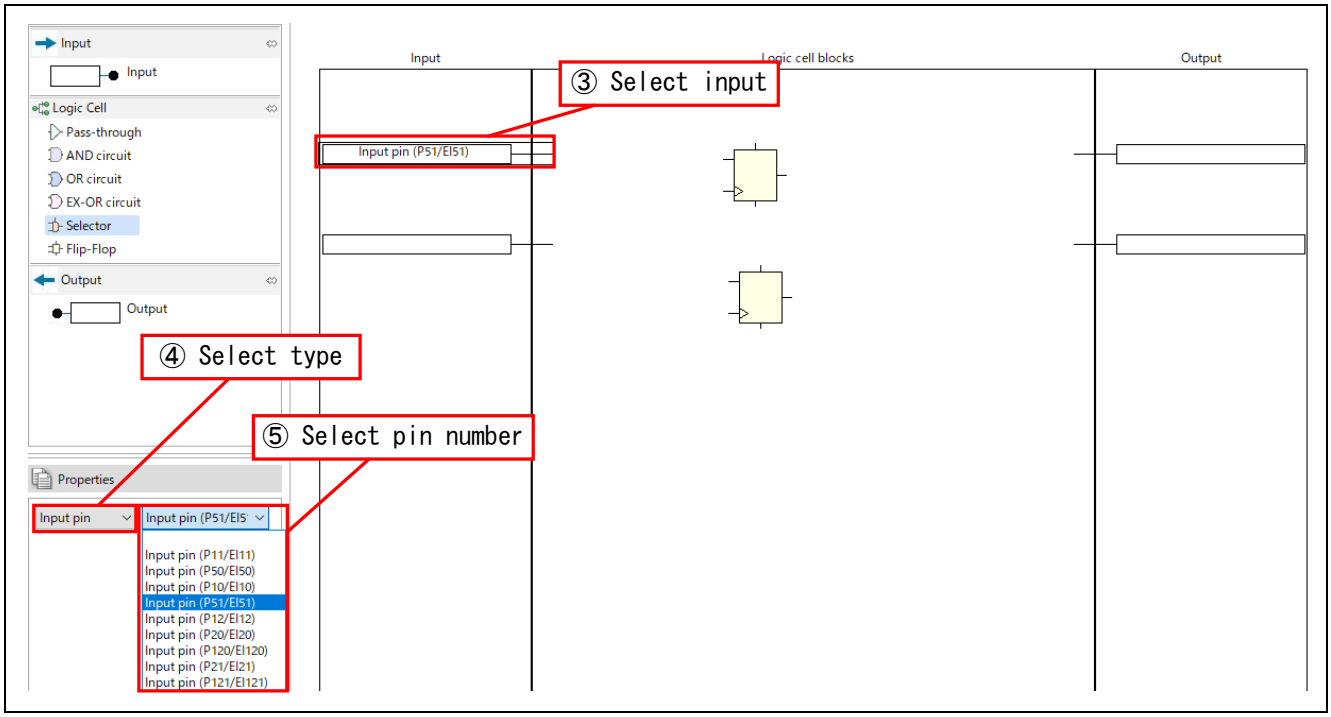
2. Drag and drop the inputs, flip-flops and outputs on the left side to the corresponding areas on the right side.

Figure 5-2 Selecting the blocks




3. Click and select the required input in the input area.
4. Select the type of input.
5. Select the pin number of input.

Figure 5-3 Property Settings of the Input Area (Table 5-5 Property Setting Value for details)



6. Click and select the required flip-flop in the logic cell block area.
7. Select the resources.
8. Check the negative logic output only for reset of the event signal output level setting.

Figure 5-4 Property Settings of the Logic Cell Block Area (1/2) (Table 5-5 Property Setting Value for details)

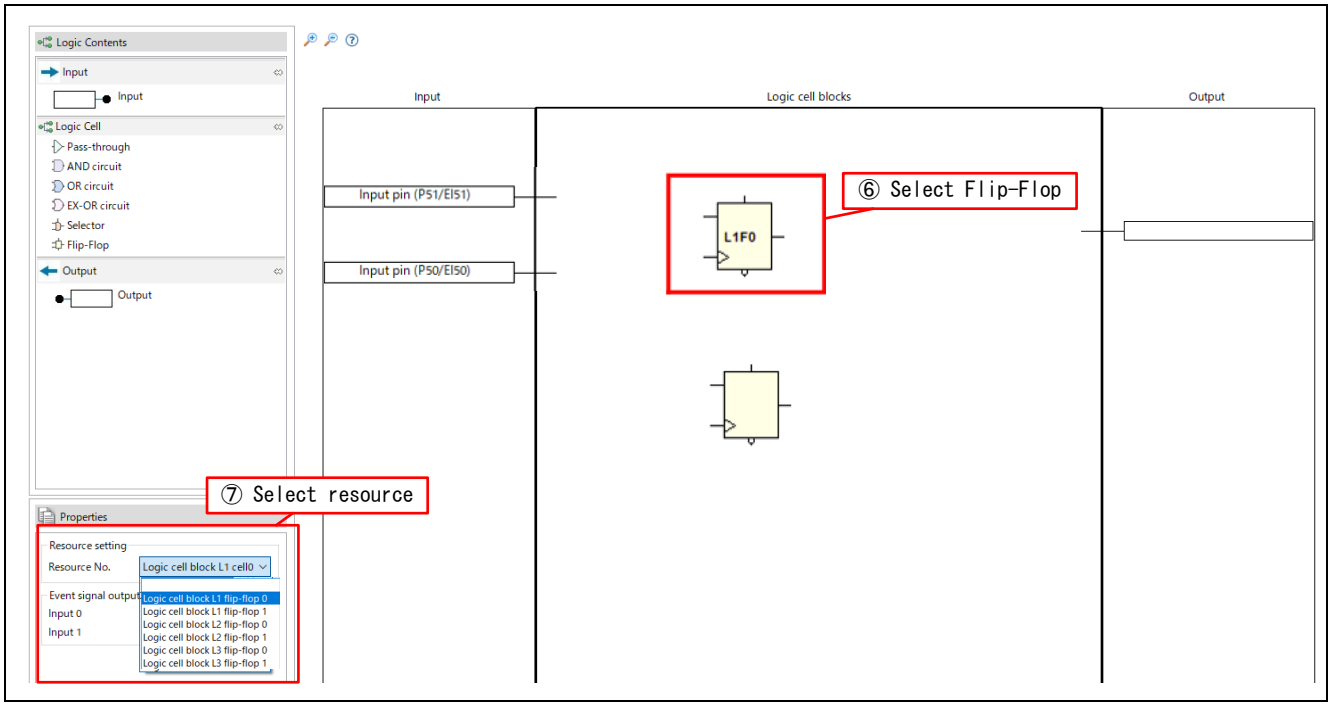
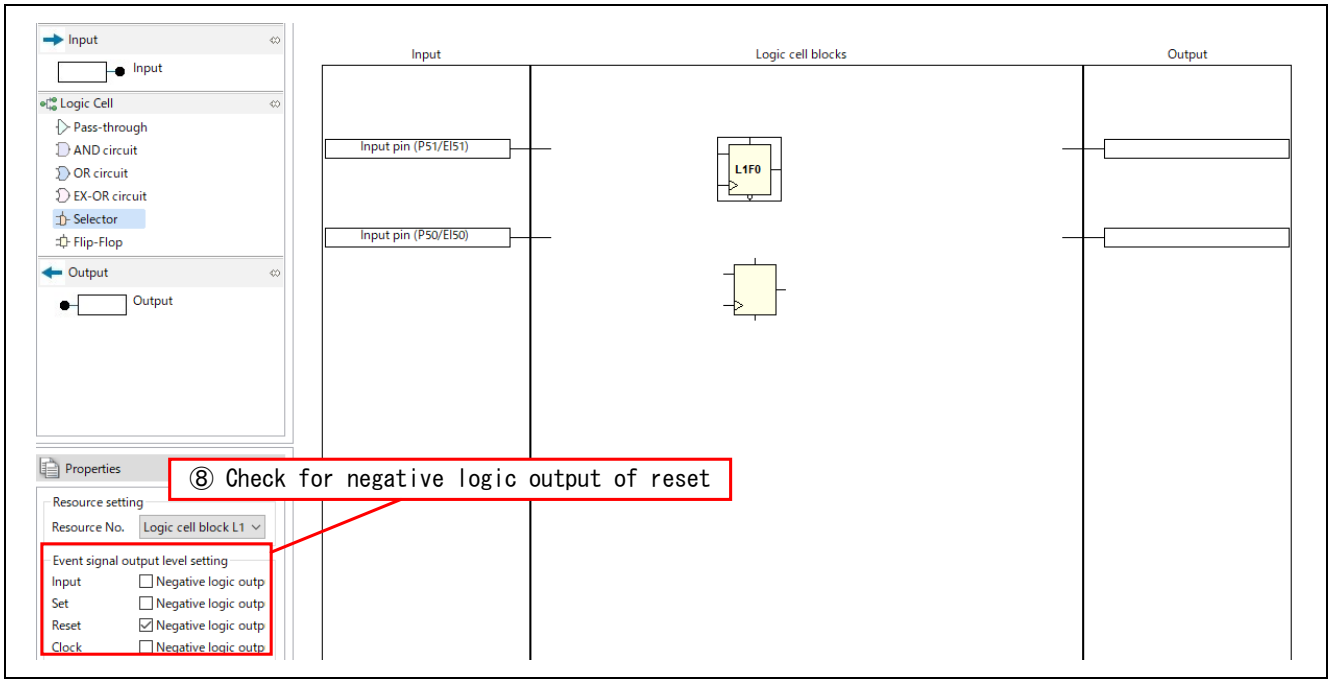
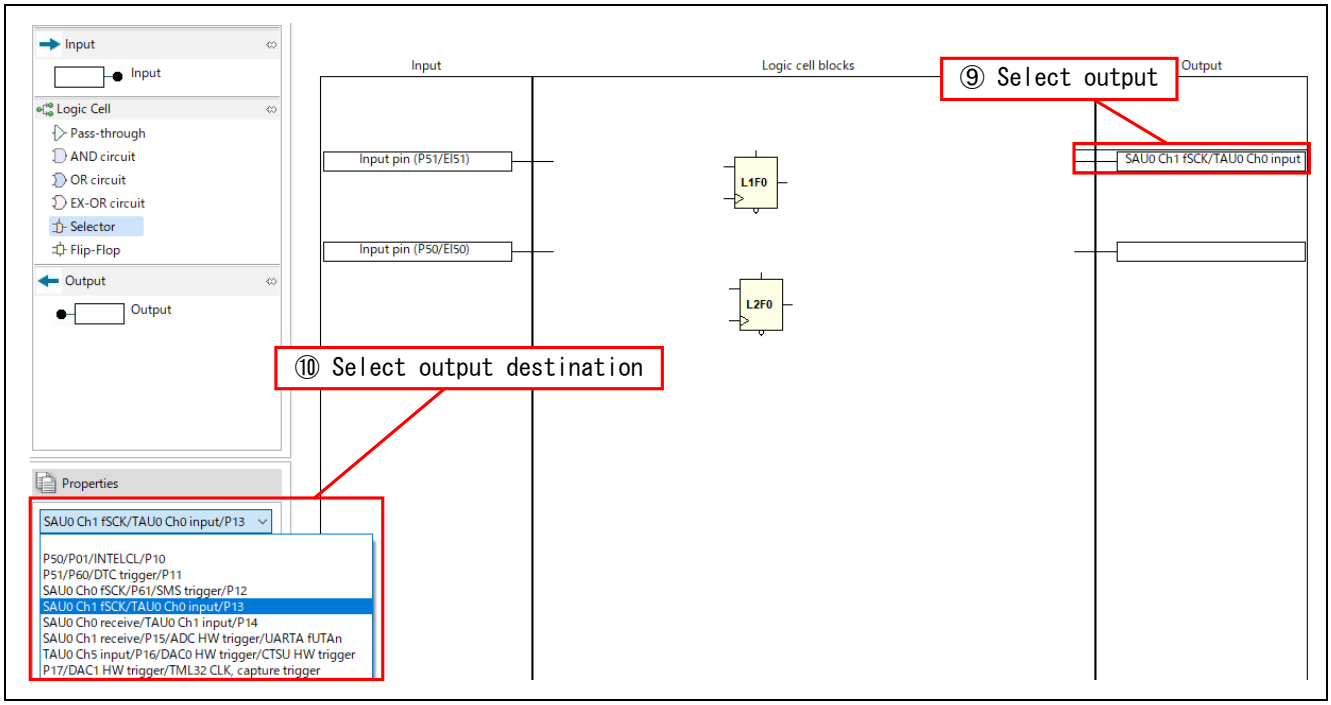



Figure 5-5 Property Settings of the Logic Cell Block Area (2/2)



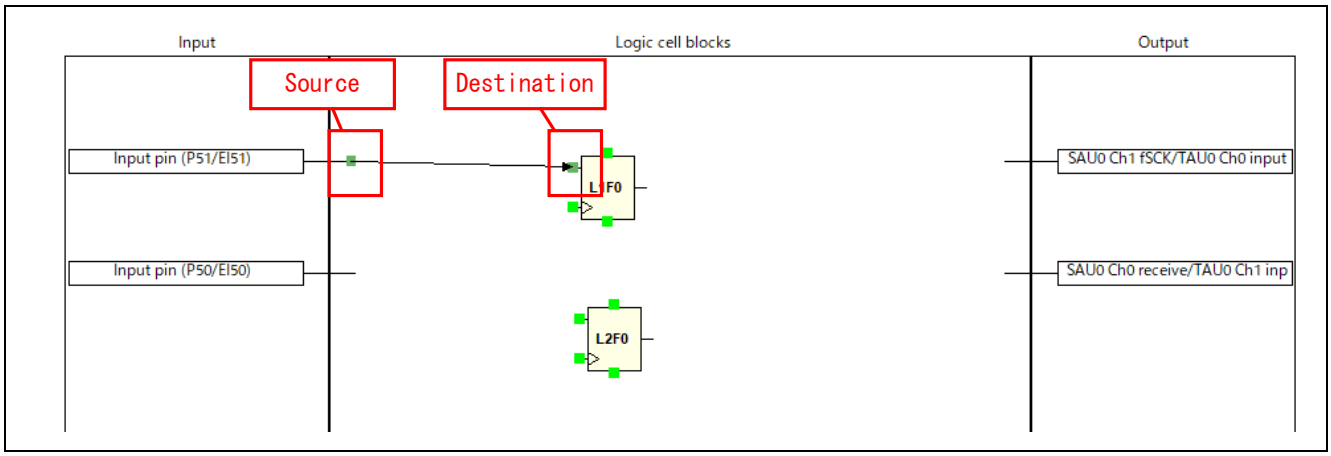
9. Click and select the required output in the output area.
10. Select output destination.

Figure 5-6 Property Settings of the Output Area (Table 5-5 Property Setting Value for details)



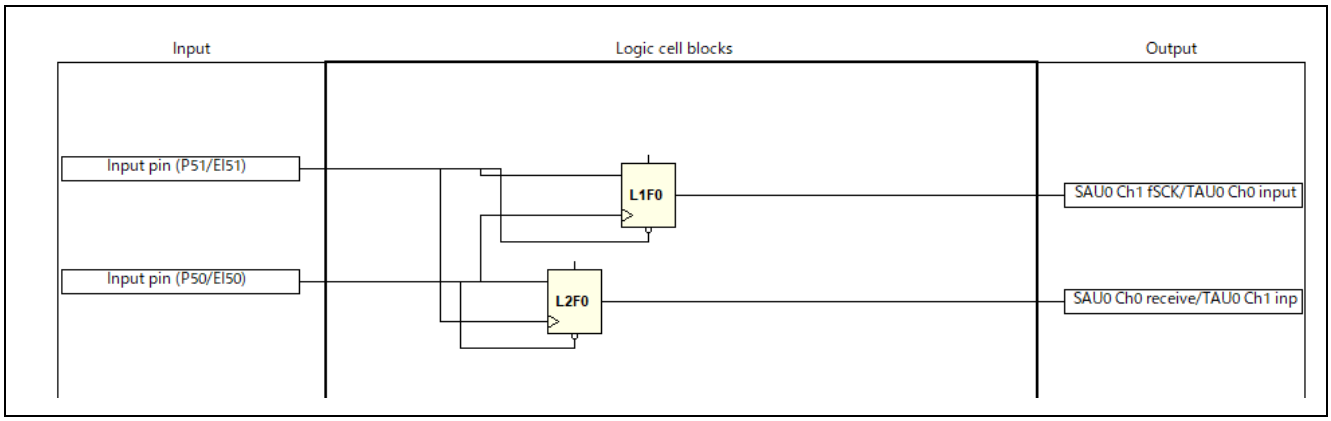

11. Connect the inputs, flip-flops and outputs. Move the cursor to the source of the connection and a green dot will appear. Click on the green dot and pull the cursor to the connection destination and release.

Figure 5-7 Connect the Blocks (Table 5-6 Connection Settings for Each Block for details)



12. Figure 5-8 shows the image of setting completion in this application note.

Figure 5-8 Image of Setting Completion



## 5.2 r01an7635_elcl.scfg

This is the Smart Configurator configuration file used in the sample code. It contains all the functions configured in the Smart Configurator. The sample code settings are as follows.

Table 5-1 Smart Configurator Settings (1/4)

| Tag Name | Component | Content |
|---|---|---|
| Clock | - | Operation mode: High-speed main mode 4.0 (V) to 5.5 (V)<br>$EV_{DD}$ setting: 4.0V ≤ $EV_{DD0}$ < 5.5V<br>High-speed on-chip oscillator: 32MHz<br>$f_{IHP}$ : 32MHz<br>$f_{CLK}$ : 32000kHz (High-speed on-chip oscillator)<br>$f_{SXP}$ : 32.768kHz (Low-speed on-chip oscillator)<br><br>(XT1 oscillation circuit)<br>Operation mode: XT1 oscillation<br>Frequency: 32.768kHz<br>XT1 oscillation mode: Low consumption oscillation 1<br>Power supply mode: Power supply enabled in STOP.HALT mode |
| System | - | On-chip debug operation setting: COM port[Note]<br>Pseudo-RRM/DMM function setting: Used<br>Start/Stop function setting: Unused<br>Trace function setting: Used<br>Security ID setting: Set<br>Security ID: 0x00000000000000000000<br>Security ID authentication failure setting: Erase flash memory data |

Note. Specify the settings as follows when using IAR.
On-chip debug operation setting: Use emulator
Emulator setting: E2 emulator Lite

Table 5-2 Smart Configurator Settings (2/4)

| Tag Name | Component | Content |
|---|---|---|
| Component | r_bsp | Start up select: Enable (use BSP startup)<br>Control of illicit memory access detection (IAWEN): Disable<br>Protected area in the RAM (GRAM0-1): Disabled<br>Protection of the port control registers (GPORT): Disabled<br>Protection of the interrupt control registers (GINT): Disabled<br>Protection of the clock, voltage detector, and RAM parity error detection control registers (GCSC): Disabled<br>Data flash memory area/extra area access control (DFLEN): Disables<br>Initialization of peripheral functions by Code Generator/Smart Configurator: Enable<br>API functions disable (R_BSP_StartClock, R_BSP_StopClock): Disable<br>API functions disable (R_BSP_GetFclkFreqHz): Enable<br>API functions disable (R_BSP_SetClockSource): Disable<br>API functions disable (R_BSP_ChangeClockSetting): Disable<br>API functions disable (R_BSP_SoftwareDelay): Disable<br>Parameter check enable: Enable<br>Enable user warm start callback (PRE): Unused<br>Enable user warm start callback (POST): Unused<br>Watchdog Timer refresh enable: Unused |
| | Config_INTC | Component: Interrupt controller<br>Resource: INTC<br>INTP0 setting: INTP0<br>Valid edge: Rising edge<br>Priority: Level 3 (low priority) |
| | Config_ITL000_ ITL001 | Component: Interval timer<br>Operation: 16-bit counter mode<br>Resource: ITL000_ITL001<br>Operation clock($f_{ITL0}$): $f_{SXP}$<br>Clock source: $f_{ITL0}$<br>Interval value: 400 ms<br>Interrupt setting: Used<br>Priority: Level 3 (low priority) |
| | Config_TAU0_0 | Component: External event counter<br>Resource: TAU0_0<br>Operation clock: CK00<br>Clock source: $f_{CLK}$<br>Input source setting: ELCL<br>External event edger selection (TI00): Rising edge<br>Count value: 1<br>Interrupt setting: Unused |

Table 5-3 Smart Configurator Settings (3/4)

| Tag Name | Component | Content |
|---|---|---|
| | Config_TAU0_1 | Component: External event counter<br>Resource: TAU0_1<br>Operation clock: CK00<br>Clock source: $f_{CLK}$<br>Input source setting: ELCL<br>Operation mode: 16 bits<br>External event edger selection (TI01): Rising edge<br>Count value: 1<br>Interrupt setting: Unused |
| Component | Config_TAU0_2 | Component: PWM output<br>Resource: TAU0_2<br>Operation clock: CK00<br>Clock source: $f_{CLK}$<br>Cycle setting: 2 ms (Decided from the data sheet of the motor)<br>Interrupt setting: Unused<br>PWM slave selection setting: Channel 5 slave, channel 6 slave<br><br>(Slave 5)<br>Duty: 90%<br>Initial output value: 0<br>Output level: Active High<br>Interrupt setting: Used<br>Priority: Level 3 (low priority)<br><br>(Slave 6)<br>Duty: 90%<br>Initial output value: 0<br>Output level: Active High<br>Interrupt setting: Unused |
| | Config_PORT | Component: Port<br>Resource: PORT<br>Port selection: PORT5<br>P52: Output (Output 0)<br>P53: Output (Output 0)<br>Port mode setting: Read the resister value of Pmn |

Table 5-4 Smart Configurator Settings (4/4)

| Tag Name | Component | Content |
|---|---|---|
| Component | Config_UART0[Note] | Component: UART communication<br>Resource: UART0<br>Operation: Transmission<br>Operation clock: CK00<br>Clock source: $f_{CLK}$<br>Transfer mode setting: Single-transfer mode<br>Data bit length setting: 8 bits<br>Data transfer direction setting: LSB<br>Parity setting: No parity bit<br>Stop bit length: 1 bit<br>Transfer rate setting: 153600 bps<br>Interrupt setting: Level 3 (low priority)<br>Callback function setting: Transmission complete |
| | Config_ELCL | Details are shown in 5.2.10 Config_ELCL |

Note. In the CS+ sample project and e2 studio sample project, COM port is specified as the connection of the debugger. Therefore, the communication line for debugging and the communication line for serial communication (UART0) compete, and an "X" is displayed on the Smart Configurator. In this sample, there is no problem if you do not use debugging and serial communication at the same time. (Be sure to disconnect Tera Term when debugging via USB-to-serial conversion. Also, be sure to exit debugging and disconnect the debugger when displaying the motor speed via Tera Term.)

### 5.2.1 Clocks

Set the clocks used in the sample code.

### 5.2.2 System

Specify the on-chip debug setting of the sample code.

The settings of “On-chip debug operation setting” and “Security ID authentication failure setting” affect “On-chip debugging enabled” in Table 4-3 Option Byte Setting. If you change the settings, confirm that no problems will occur.

### 5.2.3 r_bsp

Set the startup of the sample code.

### 5.2.4 Config_TAU0_0

Set TAU00 of the sample code.

In the sample code, external event counter function is used as the pulse count function. No interrupts are used.

### 5.2.5 Config _TAU0_1

Set TAU01 of the sample code.

In the sample code, external event counter function is used as the pulse count function. No interrupts are used.

### 5.2.6 Config_TAU0_2

Set TAU02 of the sample code.

In the sample code, two PWM outputs are set for the motor speed control. The flag that triggers the process of timing the duty ratio change is set in the interrupt vector processing.

### 5.2.7 Config_INTC

Set INTP0 of the sample code.

In the sample code, switch pressing is detected at the rising edge. The interrupt vector process sets a flag that triggers processing when a switch is pressed.

### 5.2.8 Config_ITL000_ITL001

Set TML32 of the sample code.

In the sample code, an interval timer of 400 ms is set to generate the timing for estimating the number of rotates per minute and sending the estimated results. In the interrupt vector processing, a flag is set to trigger processing at regular intervals.

### 5.2.9 Config_UART0

Set SAU00 of the sample code.

In this sample code, the data is sent to Tera Term via COM port by UART communication.

### 5.2.10 Config_ELCL

Set ELCL of the sample code.

In this sample code, L1F0 and L2F0 are used, and the quadrature encoder signal is selected as the input signal. Phase A of the quadrature encoder signal is set as input signal 0 and phase B of the quadrature encoder signal is set as input signal 1.

Figure 5-9 ELCL Setting

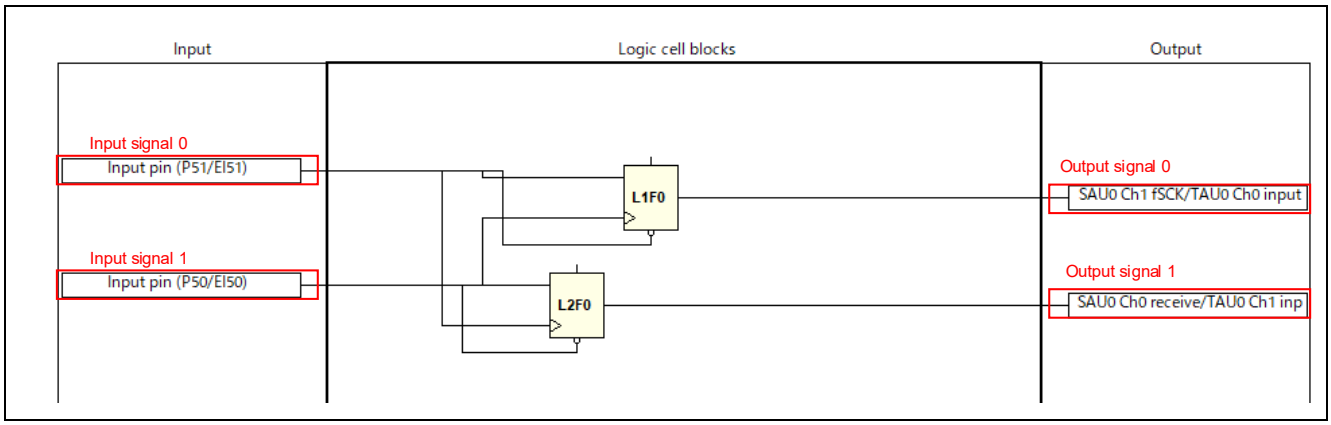


Table 5-5 Property Setting Value (Setting methods are shown in Figure 5-3 to Figure 5-6)

| Items | Property |
|---|---|
| Input signal 0 | P51 |
| Input signal 1 | P50 |
| L1F0 | Reset: Negative logic output |
| L2F0 | Reset: Negative logic output |
| Output signal 0 | TAU0 Ch0 Input |
| Output signal 1 | TAU0 Ch1 Input |

Table 5-6 Connection Settings for Each Block (Setting method is shown in Figure 5-7)

| Source of Connection | Destination of Connection |
|---|---|
| Input signal 0 | Input of L1F0 |
| | Reset of L1F0 |
| | Clock of L2F0 |
| Input signal 1 | Clock of L1F0 |
| | Input of L2F0 |
| | Reset of L2F0 |
| L1F0 | Output signal 0 |
| L2F0 | Output signal 1 |

## 5.3 Changing the Encoder and Motor

The calc_rpm function that approximates the number of rotations per minute from the counts considers the encoder resolution and the gear ratio of the motor. If the number of counts per 0.4 seconds is large, the upper limit of the counts must be changed.

This sample code has the following settings.

If you change the encoder and motor, change the values of the following constants and variables.

Table 5-7 Settings of the Encoder and Motor

| Variable Name | Value | Content |
|---|---|---|
| RESOLUTION[NOTE] | 0x00000040 | 1/4 of the encoder resolution |
| GEAR_RATIO[NOTE] | 0x0000001E | Gear ratio of the motor |
| g_max_count | 0x2710 | Maximum value of the counts |

NOTE. The value should be 32-bit to prevent overflow.

# 6. How to Use Tera Term When Checking Operation

This Application note uses Tera Term to check the count results.
An example of settings and precautions for using Tera Term are shown below.

## 6.1 Tera Term Setting

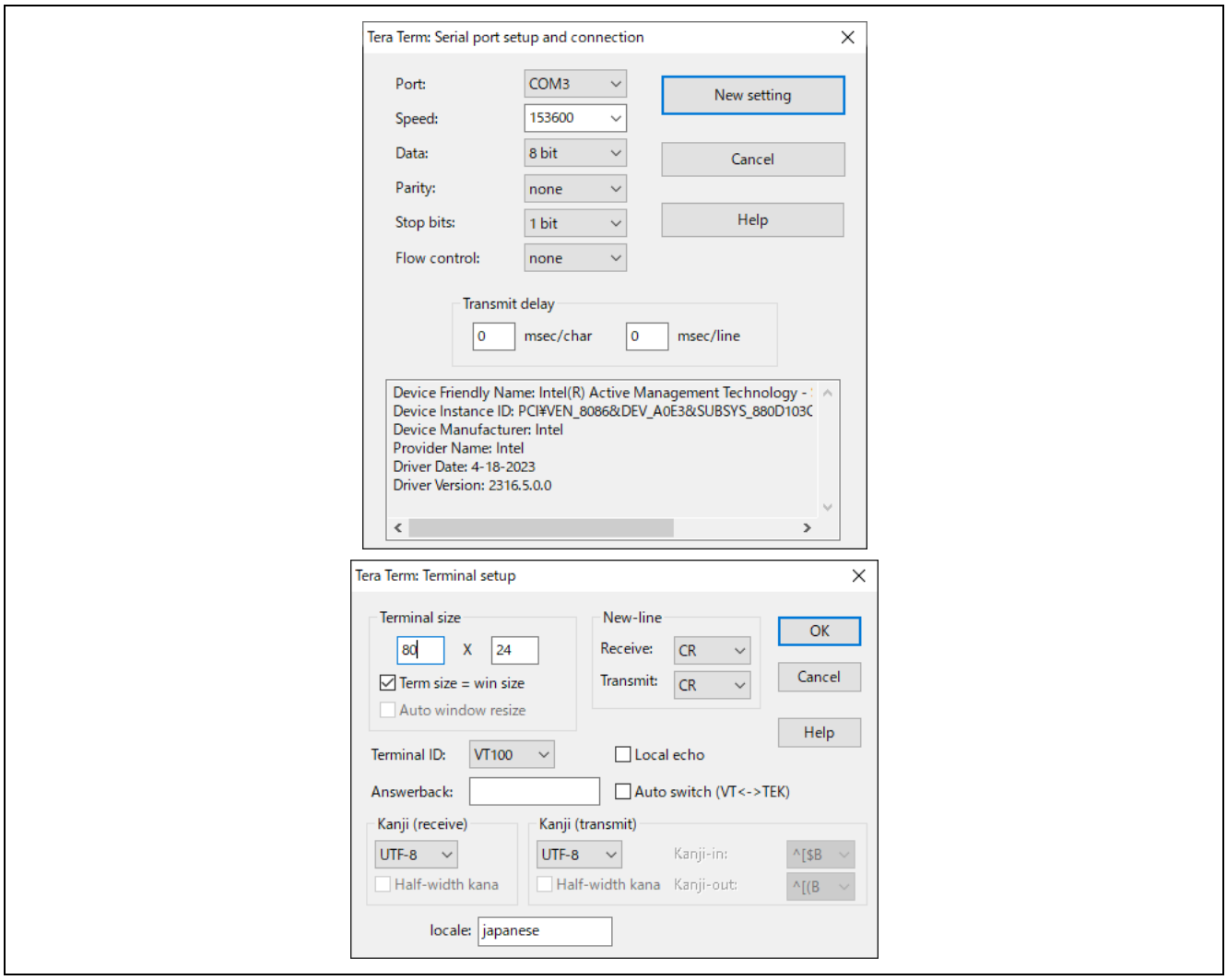
The following shows an example of settings for using Tera Term.

Table 6-1 Tera Term Settings

| Name | Setting Value |
|---|---|
| Port | Change to the number of the COM port using. |
| Speed | 153600 |
| Data | 8bit |
| Parity | none |
| Stop bit | 1bit |
| Flow control | none |
| Newline code (receoved) | CR |

Note. The USB-to-serial converter in FPB competes with the communication line used for debugging and serial communication (UART0).
Therefore, be sure to disconnect Tera Term when debugging via USB-to-serial conversion.
Also, be sure to exit debugging and disconnect the debugger when displaying the motor speed via Tera Term.

Figure 6-1 Tera Term Setup Screen



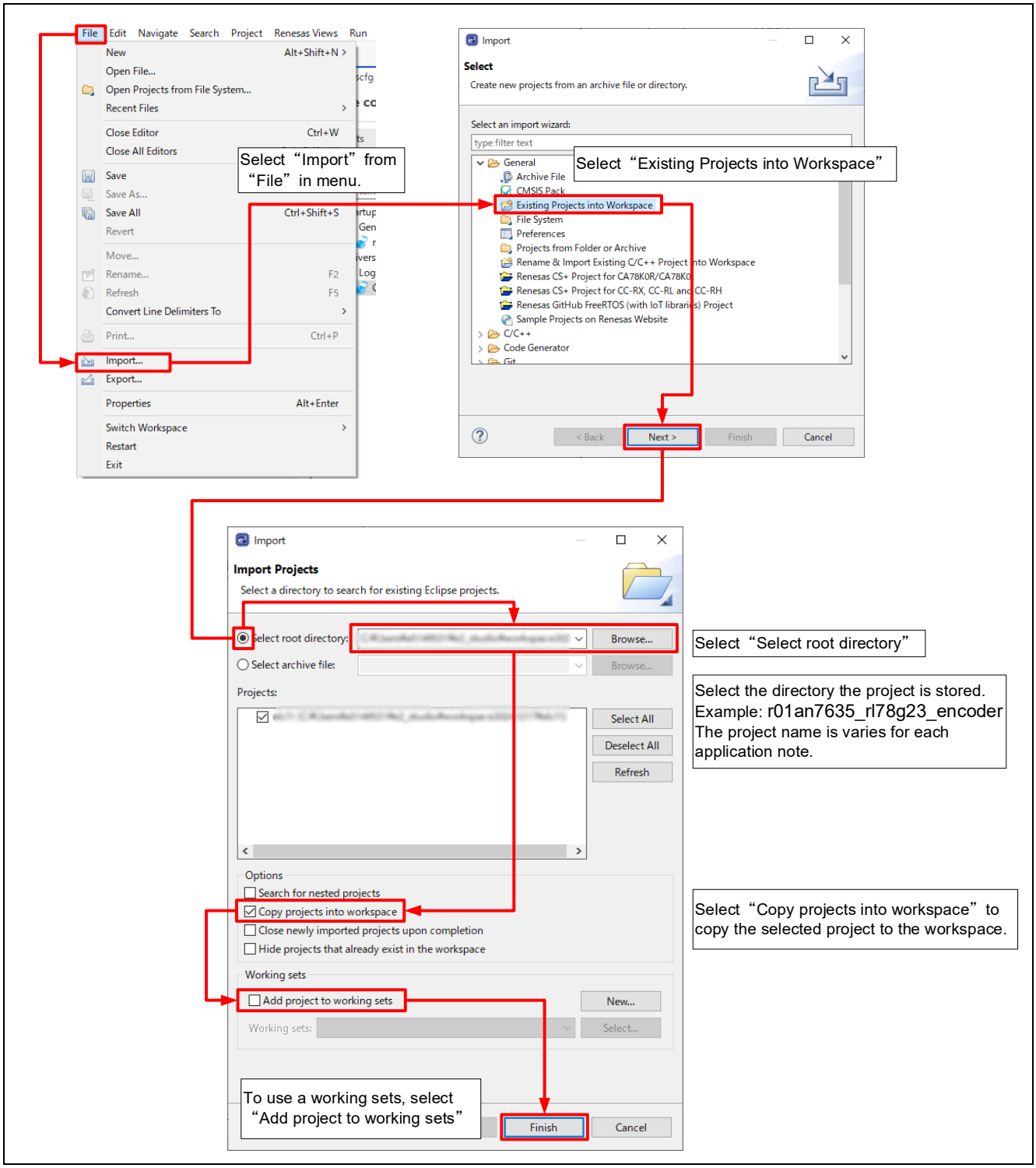
# 7. How to Import the Project

## 7.1 Procedure in e2 studio

To use the software in e² studio, please follow the procedure below to import the software into e² studio. In addition to spaces, the folder name of the project managed by e² studio and the file path to that folder must not contain any single-byte kana characters, double-byte characters, or single-byte symbols (especially '$', '#' and '%').

(The screen may differ depending on the version of e² studio used.)

Figure 7-1 How to Import the Project to e² studio



## 8. Sample code

Sample code can be downloaded from the Renesas Electronics website.

## 9. Documents for Reference

RL78/G23 User’s Manual: Hardware (R01UH0896)

RL78 family user's manual software (R01US0015)

RL78 Smart Configurator User's Guide: CS+ (R20AN0580)

RL78 Smart Configurator User's Guide: e2 studio (R20AN0579)

RL78 Smart Configurator User's Guide: IAR (R20AN0581)

The latest versions can be downloaded from the Renesas Electronics website.

Technical update

The latest versions can be downloaded from the Renesas Electronics website

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Jun,9.25 | - | First Edition |

# 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. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. 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 the 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.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

## Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.