RL78/G23

SMS Moving Average Calculation

Introduction

This application note describes how to use the SNOOZE mode sequencer to calculate the moving average of an analog input signal.

Since the moving average can be calculated with the CPU operating clock stopped, lower power consumption can be achieved than before.

Target Device

RL78/G23

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 shows how to calculate the moving average and output the calculation result by processing in SNOOZE mode sequencer(SMS).

Calculates the moving average of the voltage applied to the analog input terminals.

Sets processing that A/D conversion and average calculation processing to SMS in advance. After shifting to STOP mode, SMS is started by the interrupt request (INTITL) of the 32-bit interval timer (TML32). When SMS is started, A/D conversion and moving average calculation are performed, and the calculation result is stored in RAM.

In addition, when INTP0 is occurred by pressing SW1, the CPU returns from STOP mode and the moving average value stored in RAM is transmitted by UART0.

Figure 1-1 shows the system configuration, and Figure 1-2 shows the flow chart of the entire system.

Figure 1-1 System Configuration

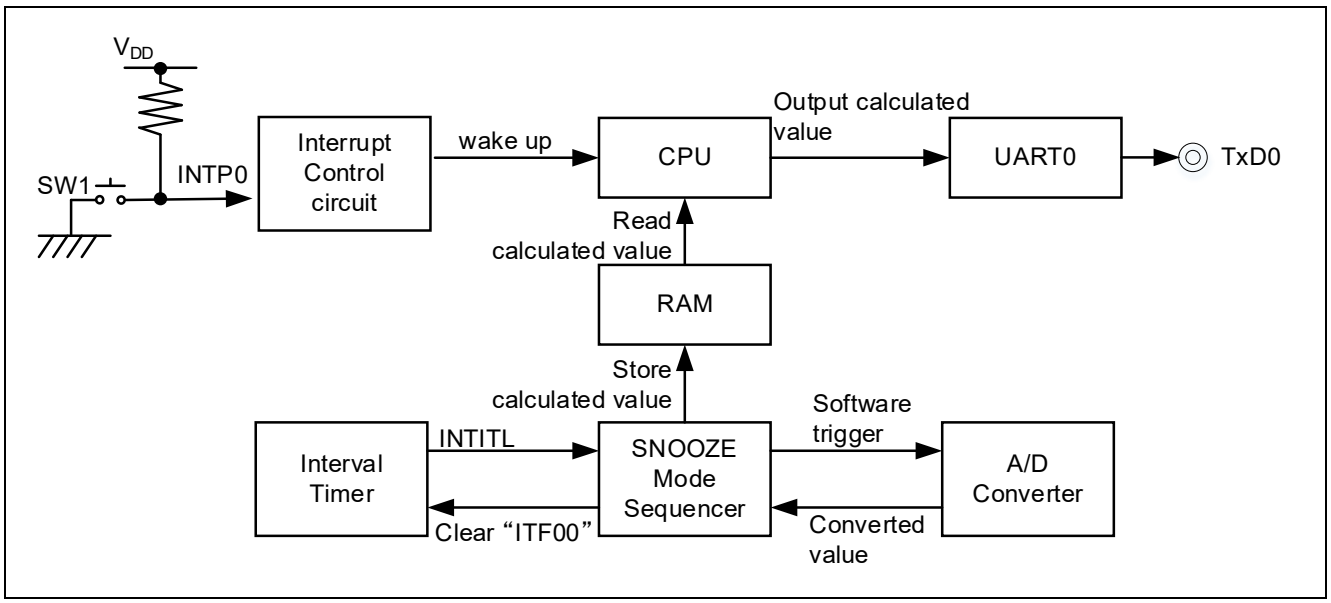
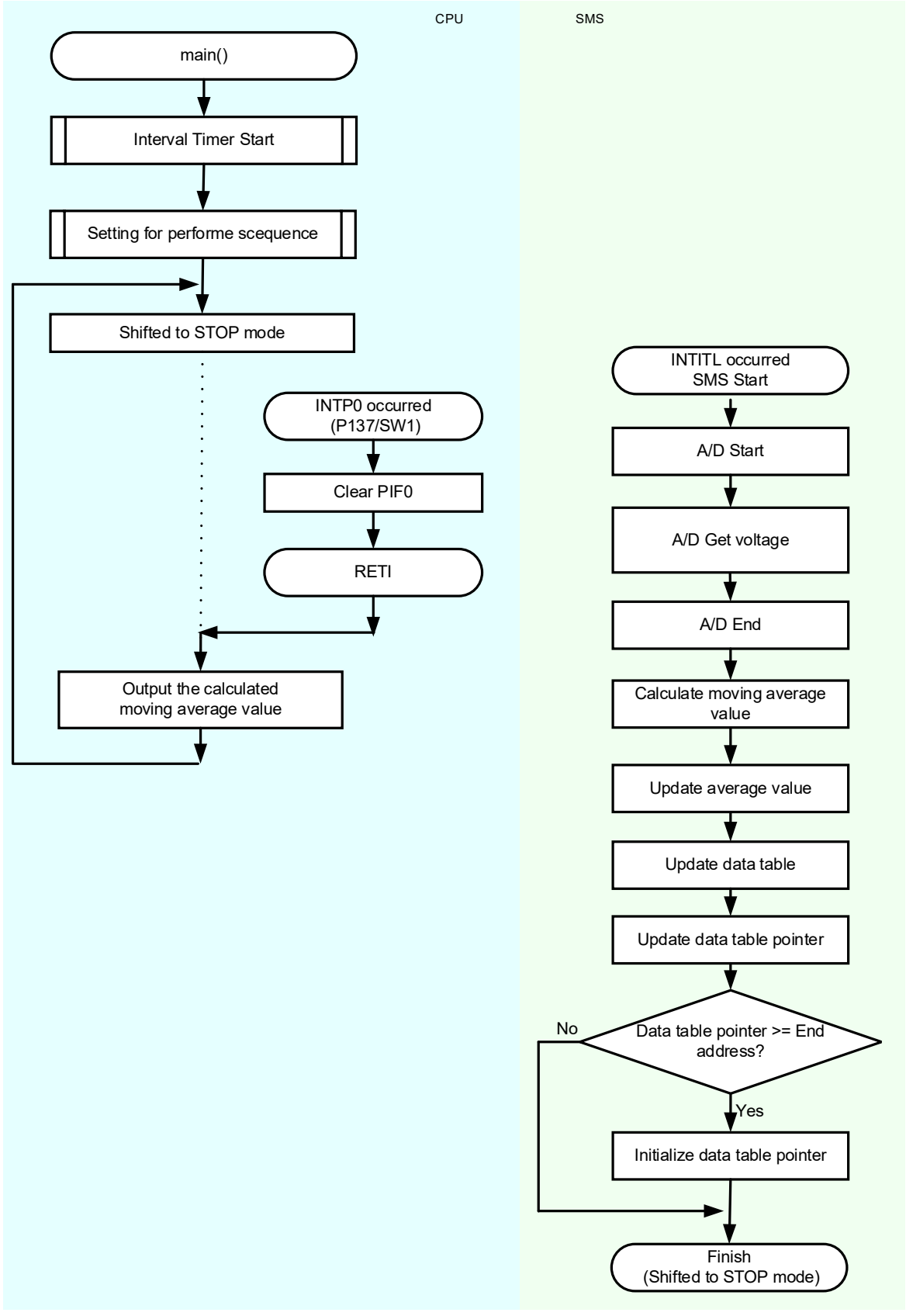


Figure 1-2 Entire Flowchart



2. Conditions for Operation Confirmation Test

The sample code with this application note runs properly under the condition below.

Table 2-1 Operation Confirmation Conditions

Items	Contents
MCU	RL78/G23 (R7F100GLG)
Operating frequencies	• High-speed on-chip oscillator clock: 32 MHz • CPU/peripheral hardware clock: 32 MHz
Operating voltage	• 3.3V • LVD0 operations (V_{LVD0}) : Reset mode Rising edge TYP.1.875V Falling edge TYP.1.835V
Integrated development environment (CS+)	CS+ for CC E8.05.00f from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.09.00 from Renesas Electronics Corp.
Integrated development environment (e^2 studio)	e^2 studio 2021-01 (21.01.0) from Renesas Electronics Corp.
C compiler (e^2 studio)	CC-RL V1.09.00 from Renesas Electronics Corp.
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 v4.20.1 from IAR Systems
C compiler (IAR)	
Smart Configurator	V.1.0.0
Board support package (r_bsp)	V.1.00
Emulator	E2 Emulator Lite
Board	RL78/G23 Fast Prototyping Board (RTK7RLG230CLG000BJ)

3. Related application note

The following application note is related to this application note.

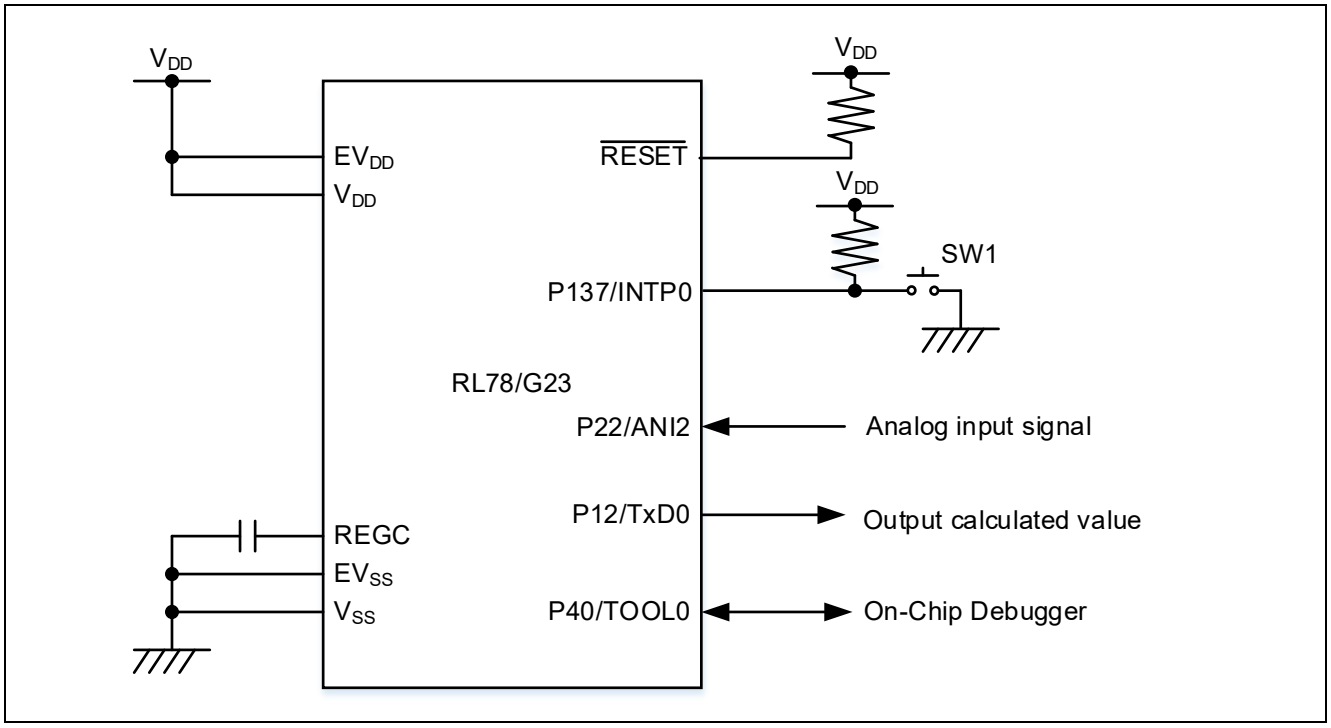
Please refer to them as well.

4. Hardware

4.1 Example of Hardware Configuration

Figure 4-1 shows an example of the hardware configuration in this application.

Figure 4-1 Hardware Configuration



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

Caution 2. Connect the EV_{SS} pin to V_{SS} and the EV_{DD} pin to V_{DD}.

Caution 3. V_{DD} must be held at not lower than the reset release voltage (V_{LVD0}) that is specified as LVD.

4.2 Used Pins

Table 4-1 shows list of used pins and assigned functions.

Table 4-1 List of Pins and Functions

Pin Name	Input/Output	Function
P22/ANI2	Input	Input for A/D conversion
P12/TxD0	Output	UART0 data transmission
P137/INTP0	Input	SW1 (Low Active)

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

5. Software

5.1 Overview of the sample program

In this sample code, the CPU shifts from STOP mode to SNOOZE mode with the interrupt request (INTITL) of 32-bit interval timer (TML32), and SMS calculates the A/D conversion and moving average, and stores the result in RAM. The moving average is calculated based on the last 8 times A/D conversion results.

In addition, when SW1 is pressed, it returns from STOP mode, and the last 8 times A/D conversion result and moving average stored in RAM are sent by UART0. It is possible to display the A/D conversion result and moving average by using the terminal software when the board (RTK7RLG230CLG000BJ) is connected to PC via USB. The displayed value is a decimal number.

The outline of the processing performed by this sample code is shown below.

(1) Starts counting TML32.
(2) Shifts to STOP mode.
(3) Shifts to SNOOZE mode with TML32 compare match.
(4) Performs A/D conversion.
(5) Calculates the moving average based on the last 8 times A/D conversion results.
(6) Stores A/D conversion result and moving average value in RAM.
(7) Returns to (2).

(4) to (7) are processed by SMS.

[Moving average UART0 transmission processing]

(1) INTP0 occurs when SW1 is pressed.
(2) Sends the last 8 times A/D conversion results and moving averages stored in RAM by UART0.
(3) Shifts to STOP mode.

Table 5-1 shows an example of the terminal software settings when using the sample code.

Table 5-1 Terminal software setting example

Item	Setting value
Baud rate（bps)	153600
Data	8 bit
Parity	none
Stop bit	1 bit
Flow control	none

5.2 Folder Configuration

Table 5-2 shows folder configuration of source file and header files using by sample code except the files generated by integrated development environment and the files in the bsp environment.

Table 5-2 Folder configuration

Folder/File configuration	Outline	Created by Smart configurator
¥rr01an5610_sms_average<DIR>	Root folder of this sample code	
¥src<DIR>	Folder for program source	
main.c	Sample code source file	
¥smc_gen<DIR>	Folder created by Smart Configurator	√
¥Config_ADC<DIR	Folder for ADC program	√
Config_ADC.c	Source file for ADC	√
Config_ADC.h	Header file for ADC	√
Config_ADC_user.c	Interrupt source file for ADC	√Note 1
¥Config_INTC<DIR>	Folder for interrupt program	√
Config_INTC.c	Source file for INTP0 (SW1)	√
Config_INTC.h	Header file for INTP0	√
Config_INTC_user.c	Interrupt source file for INTP0	√Note 2
¥Config_ITL000_ITL001<DIR>	Folder for TML32 program	√
Config_ITL000_ITL001.c	Source file for TML32	√
Config_ITL000_ITL001.h	Header file for TML32	√
Config_ITL000_ITL001_user.c	Interrupt source file for TML32	√Note 1
¥Config_SMS<DIR>	Folder for SMS program	√
Config_SMS.c	Source file for SMS	√
Config_SMS.h	Header file for SMS	√
Config_SMS_ASM.smsasm	ASM source file for SMS	√
Config_SMS_user.c	Interrupt source file for SMS	√
¥Config_TAU0_7<DIR>	Folder for TAU program	√
Config_TAU0_7.c	Source file for TAU	√
Config_TAU0_7.h	Header file for TAU	√
Config_TAU0_7_user.c	Interrupt source file for TAU	√Note 2
¥Config_UART0<DIR>	Folder for UART program	√
Config_UART0.c	Source file for UART	√Note 3
Config_UART0.h	Header file for UART	√
Config_UART0_user.c	Interrupt source file for UART	√Note 1
¥general<DIR>	Folder for initialize or common program	√
¥r_bsp<DIR>	Folder for BSP program	√
¥r_config<DIR>	Folder for BSP_CFG program	√

Note. <DIR> means directory.

Note 1. Not used in this sample code.

Note 2. Added the interrupt handling routine to the file generated by the Smart Configurator.

Note 3. Added the LED1 ON/OFF process to the file generated by the Smart Configurator.

5.3 Option Byte Settings

Table 5-3 shows the option byte settings.

Table 5-3 Option Byte Settings

Address	Setting Value	Contents
000C0H/040C0H	1110 1111B (EFH)	Operation of Watchdog timer is stopped (counting is stopped after reset)
000C1H/040C1H	1111 1110B (FEH)	LVD0 operating mode: reset mode Detection voltage: Rising edge 1.875V Falling edge 1.835V
000C2H/040C2H	1110 1000B (E8H)	Flash operating mode: HS mode High-speed on-chip oscillator clock: 32MHz
000C3H/040C3H	1000 0101B (85H)	On-chip debugging is enabled

5.4 Constants

Table 5-4 shows the constants that are used in this sample code.

Table 5-4 Constants used in the sample code

Constant Name	Setting Value	Contents	File
NUMBER_OF_CALC	8	Number of data to average	main.c
CHATTA_WAIT	200	Chattering prevention time (200ms)	Config_INTC_user.c

5.5 Variables

Table 5-5 shows the global variables used in this sample code.

Table 5-5 Global variables used in the sample code

Type	Variable name	contents	Functions used in
volatile uint16_t	g_ms_timer	Count value of the wait process	r_ms_delay, r_Config_TAU0_7_interrupt

5.6 Functions

Table 5-6 shows the functions used in the sample code. However, the unchanged functions generated by the Smart Configurator are excluded.

Table 5-6 Functions

Function name	Outline	Source file
Main	Main process	main.c
r_Config_INTC_intp0_interrupt	INTP0 interrupt process	Config_INTC_user.c
r_ms_delay	Wait process to prevent chattering	Config_TAU0_7_user.c
r_Config_TAU0_7_interrupt	TAU0 channel 7 interrupt process (For chattering prevention)	Config_TAU0_7_user.c
Putchar	UART0 string output process	Config_UART0.c
Send	UART0 transmit data setting process	Config_UART0.c

5.7 Function Specifications

This part describes function specifications of the sample code.

[Function name] main

Outline	Main process
Header	e^2 studio, CS+ : r_smc_entry.h IAR : ior7f100g.h, ior7f100g_ext.h, r_cg_macrodriver.h, Config_TAU0_7.h, Config_UART0.h, Config_SMS.h, Config_ITL000_ITL001.h, Config_INTC.h
Declaration	void main(void)
Description	This function starts the operation of UART0, INTP0, SMS and TML32 and shifts to STOP mode. It returns from STOP mode by INTP0 that occurs for SW1 pressing, outputs the last 8 times A/D conversion results and moving average values stored in RAM with the printf() function, and the mode shifts to STOP mode again.
Arguments	None
Return value	None
Remarks	None

[Function name] r_Config_INTC_intp0_interrupt

Outline	INTP0 interrupt process
Header	r_cg_macrodriver.h, r_cg_userdefine.h, Config_INTC.h, Config_TAU0_7.h
Declaration	#pragma interrupt r_Config_INTC_intp0_interrupt(vect=INTP0)
Description	This function is an interrupt process by INTP0 that occurs when SW1 is pressed. The wait process is executed to prevent chattering when SW1 is pressed, and the INTP0 interrupt flag is cleared.
Arguments	None
Return value	None
Remarks	None

[Fnction name] r_ms_delay

Outline	Wait process
Header	r_cg_macrodriver.h, r_cg_userdefine.h, Config_TAU0_7.h
Declaration	void r_ms_delay(uint16_t msec)
Description	This function waits for the time (ms) specified by the argument msec. This function counts using channel 7. Polls if g_ms_timer is less than CHATTA_WAIT, completes wait process if more than CHATTA_WAIT.
Arguments	msec
Return value	None
Remarks	None

[Function name] r_Config_TAU0_7_interrupt

Outline	TAU0 channel 7 interrupt process
Header	r_cg_macrodriver.h, r_cg_userdefine.h, Config_TAU0_7.h
Declaration	#pragma interrupt r_Config_TAU0_7_interrupt(vect=INTTM07)
Description	This function is an interrupt process by INTTM07 on TAU0 channel 7. Counts up g_ms_timer.
Arguments	None
Return value	None
Remarks	None

[Function name] putchar

Outline	UART0 string output process
Header	r_cg_macrodriver.h, r_cg_userdefine.h, Config_UART0.h
Declaration	int putchar (int ch)
Description	This function outputs the character string specified by the printf() function by one character. The character to be output is specified for the argument “ch”. In this application note, added the send() function to the standard library process of this function in order to output characters by UART0 transmission. The argument “ch” is set in the UART0 transmit buffer by executing the send() function.
Arguments	ch
Return value	0
Remarks	None

[Function name] send

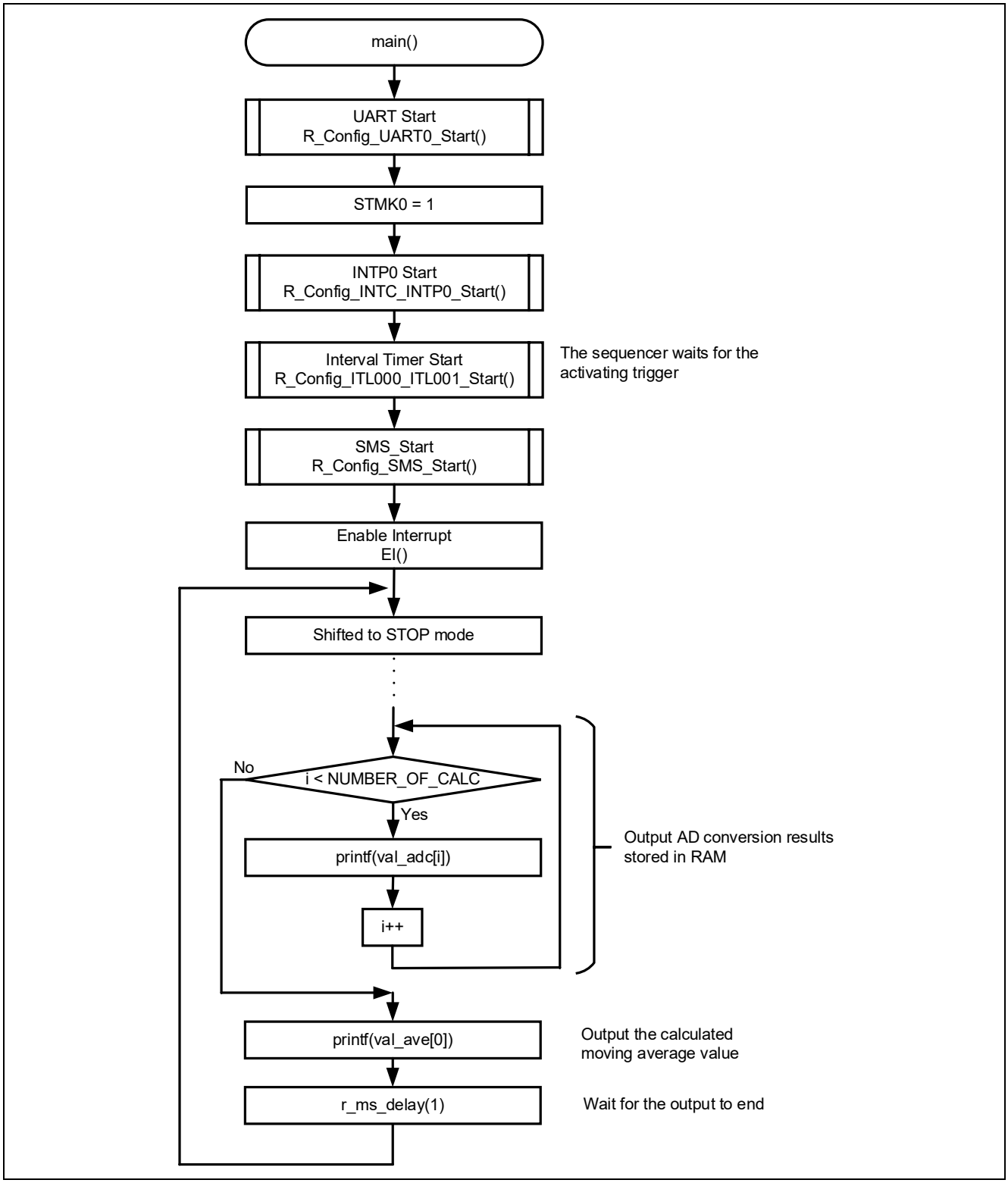
Outline	UART0 string output process
Header	r_cg_macrodriver.h, r_cg_userdefine.h, Config_UART0.h
Declaration	int putchar (int ch)
Description	This function sets an argument "ch" in the UART0 transmit buffer. A character string specified at printf() is set to argument "ch" via putchar() by one character at once.
Arguments	ch
Return value	0
Remarks	None

5.8 Flow Charts

5.8.1 Main Process

Figure 5-1 shows flowchart of main process.

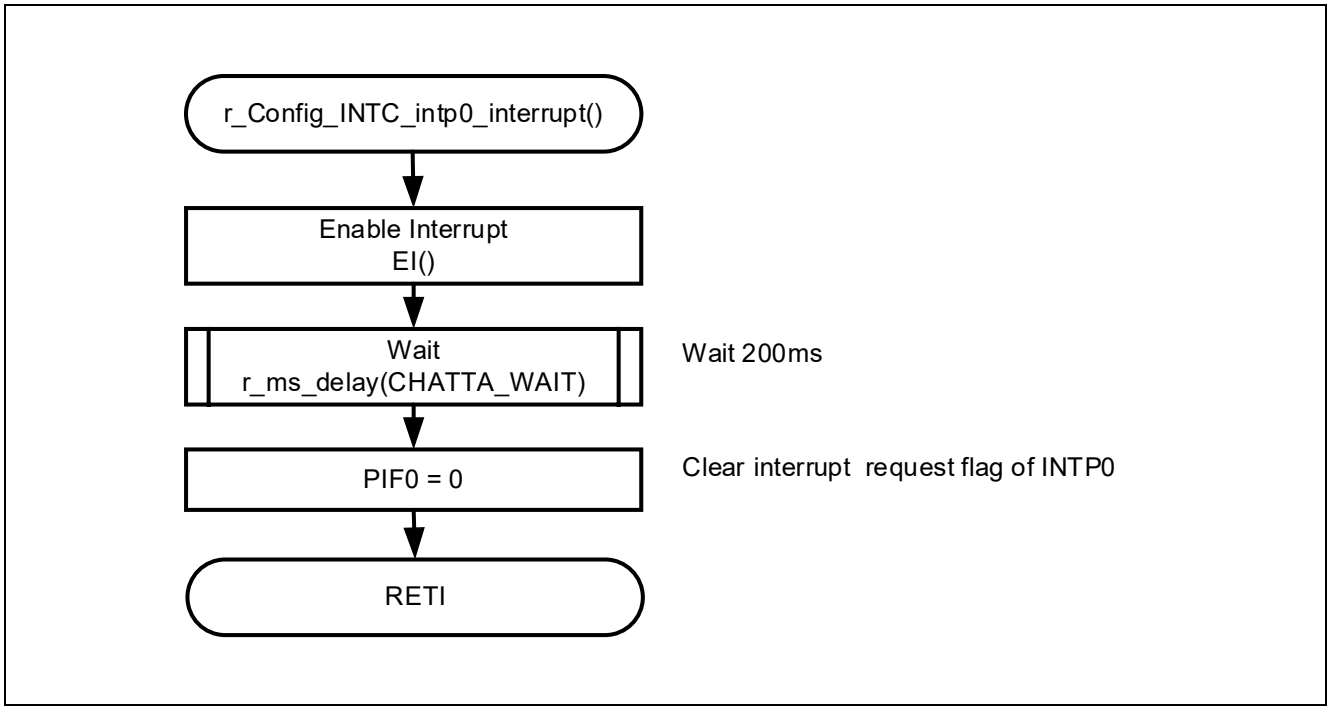
Figure 5-1 Main process



5.8.2 INTP0 interrupt process

Figure 5-2 shows flowchart of INTP0 interrupt process.

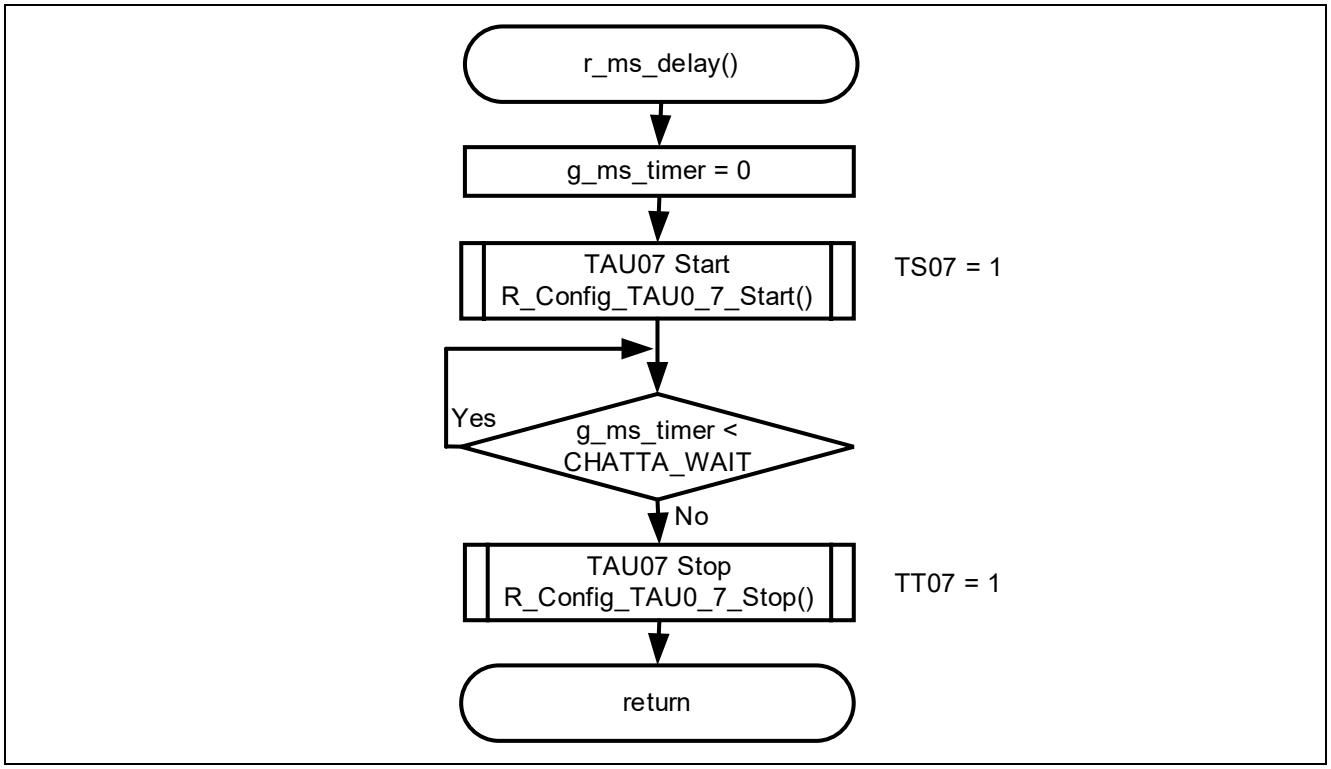
Figure 5-2 INTP0 interrupt process



5.8.3 Wait process

Figure 5-3 shows flowchart of Wait process.

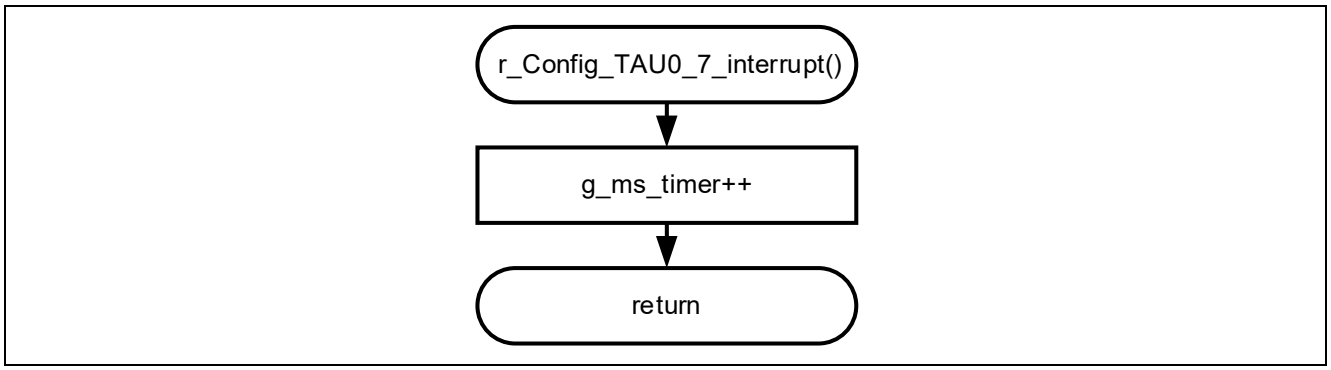
Figure 5-3 Wait process



5.8.4 TAU0 channel 7 interrupt process

Figure 5-4 shows flowchart of TAU0 channel 7 interrupt process.

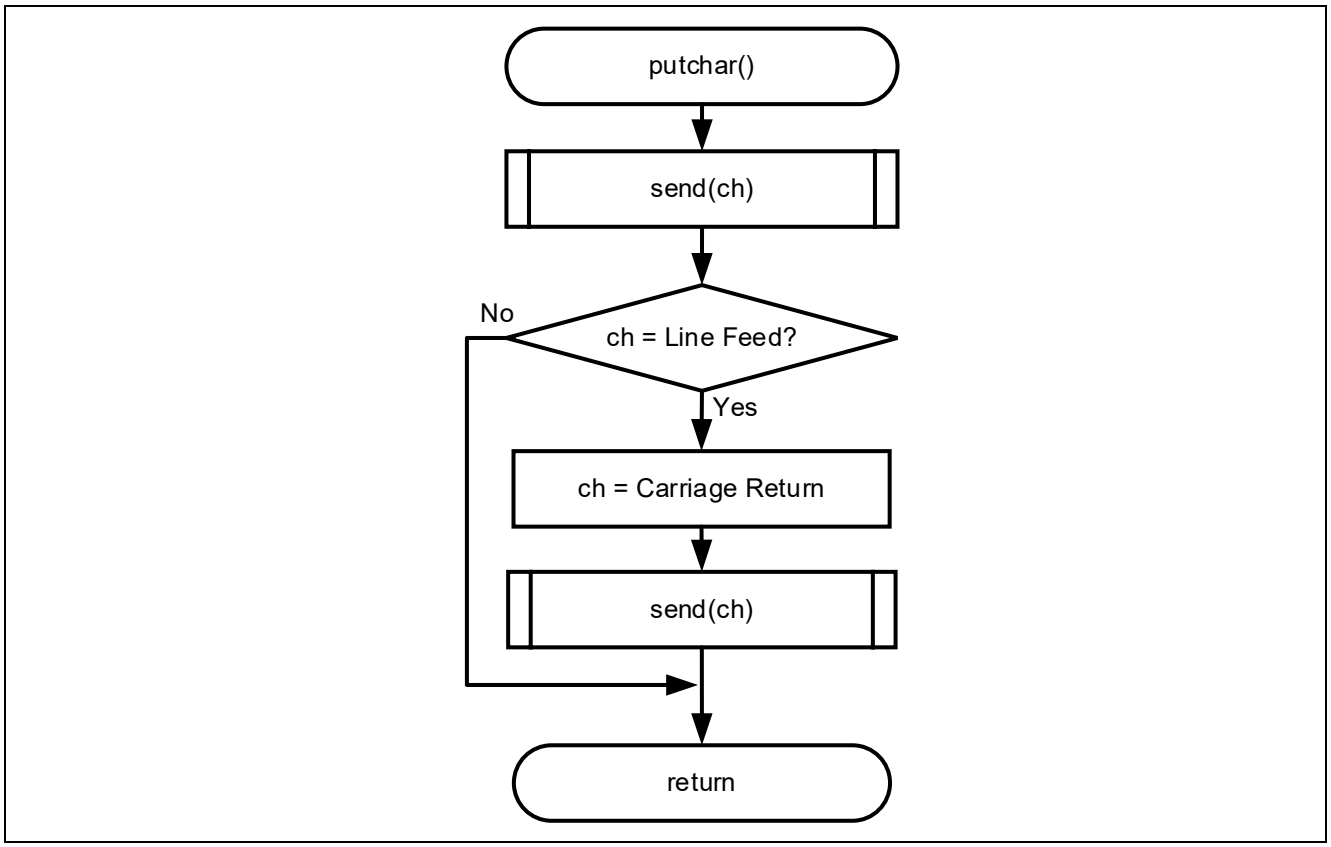
Figure 5-4 TAU0 channel 7 interrupt process



5.8.5 UART0 string output process

Figure 5-5 shows flowchart of UART0 string output process.

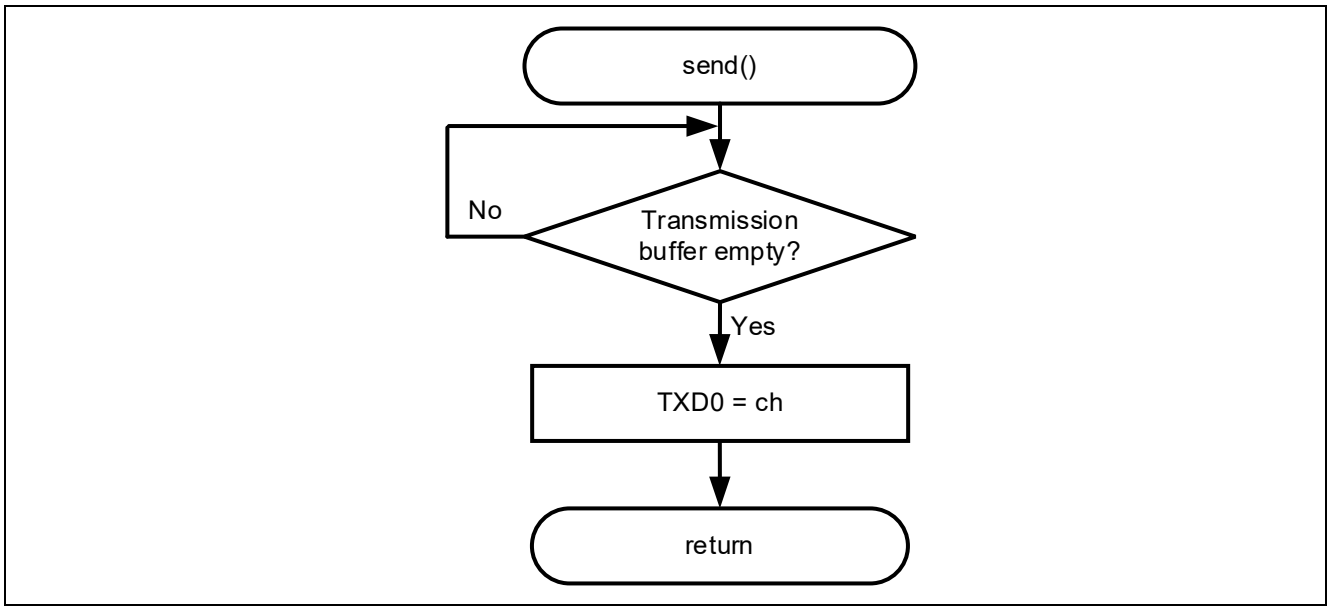
Figure 5-5 UART0 string output process



5.8.6 UART0 transmit data setting process

Figure 5-6 shows flowchart of UART0 transmit data setting process.

Figure 5-6 UART0 transmit data setting process



5.9 SNOOZE Mode Sequencer settings

When the event set in the start trigger occurs, SMS executes the processing commands stored in the sequencer instruction register (SMSI0-31) in order. When executing a processing command, the Sequencer general-purpose register (SMSG0-15) is used to store the source address, destination address, calculated data, and so on.

SMSI0-31 and SMSG0-15 are set by writing the SMS program (.SMSASM file) in assembly language. The SMS program can also be created by combining processing blocks using the SNOOZE mode sequencer component of the Smart Configurator. The created SMS program is converted to a C language file by the SMS assembler and incorporated into the program.

The specifications of SMS processing executed by the sample code are shown below.

Outline	SMS process
Description	SMS is started by a TML32 interrupt, performs A/D conversion. The moving average is calculated from the A/D conversion result, and the result is stored in RAM.
Arguments Note1	val_ram_avg, val_ram_d, val_ram_s, val_ram_e
Return value	None
Remarks	None

Note1. Argument to be specified in the R_Config_SMS_Start function setting. For details, refer to 6.2.1 and 6.2.15.

Figure 5-7 shows flowchart of SMS process.
Table 5-7, Table 5-8 show the register settings that control the SNOOZE mode sequencer.

Figure 5-7 SMS process

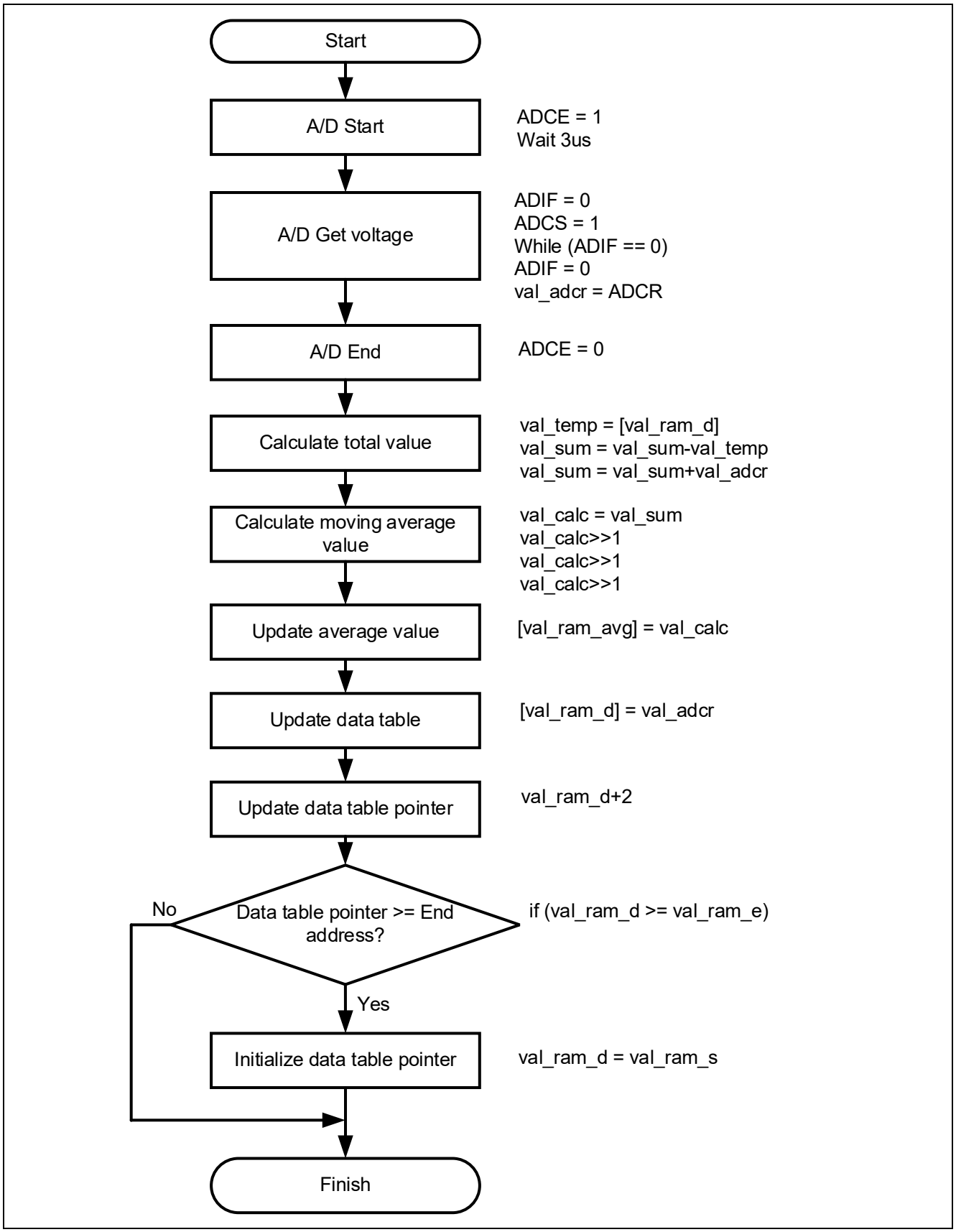


Table 5-7 Sequencer general-purpose registers 0-15

Register Symbol	Setting	Remark
SMSG0	0000H	Remark
SMSG1	0000H	Variable for storing ADCR register: val_adcr
SMSG2	0000H	Variable of sum value: val_sum
SMSG3	0000H	Data table pointer: val_ram_d
SMSG4	0000H	Variable for calculation: val_calc
SMSG5	0000H	Address to store the average value: val_ram_avg
SMSG6	0000H	Data table end address: val_ram_e
SMSG7	0000H	Data table start address: val_ram_s
SMSG8	&ADCR0	ADCR0 address
SMSG9	&ITLS0	ITLS0 address
SMSG10	&SMSG3	SMSG3 register address
SMSG11	&ADM0	ADM0 register address
SMSG12	&IF1H	IF1H register address
SMSG13	2	fixed value: 2
SMSG14	0000H	SMS internal variable
SMSG15	FFFFH	fixed value: FFFFH

Table 5-8 Sequencer instruction registers 0-31

Register Symbol	Setting	Remark
SMSI0	0900H	MOV [SMSG9+0], SMSG0
SMSI1	4B00H	SET1 [SMSG11+0].0
SMSI2	9600H	WAIT 96, 0
SMSI3	5C00H	CLR1 [SMSG12+0].0
SMSI4	4B70H	SET1 [SMSG11+0].7
SMSI5	BC00H	WHILE0 [SMSG12+0].0
SMSI6	5C00H	CLR1 [SMSG12+0].0
SMSI7	3810H	MOVW SMSG1, [SMSG8+0]
SMSI8	5B00H	CLR1 [SMSG11+0].0
SMSI9	33E0H	MOVW SMSG14, [SMSG3+0]
SMSI10	72E1H	SUBW SMSG2, SMSG14
SMSI11	7210H	ADDW SMSG2, SMSG1
SMSI12	2A22H	MOVW [SMSG10+2], SMSG2
SMSI13	7403H	SHRW SMSG4
SMSI14	7403H	SHRW SMSG4
SMSI15	7403H	SHRW SMSG4
SMSI16	2540H	MOVW [SMSG5+0], SMSG4
SMSI17	2310H	MOVW [SMSG3+0], SMSG1
SMSI18	73D0H	ADDW SMSG3, SMSG13
SMSI19	7362H	CMPW SMSG3, SMSG6
SMSI20	8020H	BC $2
SMSI21	2A70H	MOVW [SMSG10+0], SMSG7
SMSI22	F000H	FINISH
SMSI23-31	0000H	unused

6. Application example

In addition to the sample code, this application note stores the following Smart Configurator configuration files.

r01an5610_sms_average.scfg

r01an5610_sms_average.sms

The following is a description of the file and setting examples and precautions for use.

6.1 r01an5605_sms_power_monitoring.scfg

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

Table 6-1 Parameters of Smart Configurator

Tag name	Components	Contents
Clocks	-	Operation mod: High-speed main mode 2.4 (V)~5.5 (V) EV_{DD} setting: $1.8V \leqq EV_{DD0} < 5.5V$ High-speed on-chip oscillator: 32MHz f_{IHP}: 32MHz f_{CLK}: 32000kHz (High-speed on-chip oscillator) f_{SXL}: 32.768kHz (Low-speed on-chip oscillator)
System	-	On-chip debug operation setting: Use emulator Emulator setting: E2 Lite Pseudo-RRM/DMM function setting: Used Start/Stop function setting: Unused Trace function setting: Used Security ID setting: Use security ID Security ID：0x00000000000000000000 Security ID authentication failure setting: Do not erase flash memory data
Components	r_bsp	Start up select : Enable (use BSP startup) Control of invalid memory access detection : Disable RAM guard space (GRAM0-1) : Disabled Guard of control registers of port function (GPORT) : Disabled Guard of registers of interrupt function (GINT) : Disabled Guard of control registers of clock control function, voltage detector, and RAM parity error detection function (GCSC) : Disabled Data flash access control (DFLEN) : Disables Initialization of peripheral functions by Code Generator/Smart Configurator : Enable API functions disable : Enable Parameter check enable : Enable Setting for starting the high-speed on-chip oscillator at the times of release from STOP mode and of transitions to SNOOZE mode : High-speed Enable user warm start callback (PRE) : Unused Enable user warm start callback (POST) : Unused Watchdog Timer refresh enable : Unused
	Config_LVD0	Operation mode setting: Reset mode Voltage detection setting: Reset generation level (V_{LVD0}): 1.835 (V)

Table 6-2 Parameters of Smart Configurator

Tag name	Components	Contents
Components	Config_INTC	INTP0 setting: use Valid edge: Falling edge Priority: Level 3
	Config_ITL000 _ITL001	Components: Interval timer Operation mode: 16 bit count mode Resource: ITL000_ITL001 Operation clock: f_{SXL} Clock source: f_{ITL0}/128 Interval value: 1000 ms Interrupt setting: unused
	Config_TAU0_7	Components: Interval timer Operating mode: 16 bit count mode Resource: TAU0_7 Operation clock: CK00 Clock source: f_{CLK} Interval value: 1 ms Interrupt setting: use Priority: Level 2
	Config_ADC	Components: A/D Converter Comparator operation setting: Stop Resolution setting: 12 bits VREF(+) setting: VDD VREF(-) setting: VSS Trigger mode setting: Software trigger no-wait mode Operation mode setting: One-shot select mode A/D channel selection: ANI2 Conversion time mode: Normal 1 Conversion time: 66/f_{CLK} Conversion result upper/lower bound value setting: Generates an interrupt request (INTAD) when ADLL≦ADCRn≦ADUL Upper bound (ADUL) value: 255 Lower bound (ADLL) value: 0 Interrupt setting: unused
	Config_SMS	Components: SNOOZE Mode Sequencer Start trigger: Interval detection interrupt (INTITL)
	Config_UART0	Components: UART Interface Operation mode: Transmission Resource: UART0 Operation cock: CK00 Clock source: f_{CLK} Transfer mode setting: Single transfer mode Data length setting: 8 bits Transfer direction setting: LSB Parity setting: None Stop bit length setting: 1 bit Transfer data level setting: Non-reverse Transfer rate setting: 153,600bps Priority: Level 3 Callback function setting: unused
Pins	-	Pin Function: Serial Array Unit (SAU00) TxD0: Enabled (P12)

6.1.1 Clocks

Set the clock used in the sample code.

In this sample code, 32000KHz is set for f_{CLK} and the conversion time mode is set to "Standard 1 (2.4 V ≤ VDD ≤ 5.5 V)" with Config_ADC, so the operation mode is "High-speed main mode 2.4 (V) ~ 5.5 (V)". Note that changing the settings.

6.1.2 System

Set the on-chip debug of the sample code.

"Control of on-chip debug operation" and "Security ID authentication failure setting" affect "On-chip debugging is enabled" in "Table 5-3 Option Byte Settings". Note that changing the settings.

6.1.3 r_bsp

Set the startup of the sample code.

6.1.4 Config_LVD0

Set the power management of the sample code.

Affects "Setting of LVD0" in "Table 5-3 Option Byte Settings". Note that changing the settings

6.1.5 Config_INTC

Set the interrupt used in the sample code.

In the sample code, set an external maskable interrupt (INTP0). When the INTP0 is not used, delete it.

6.1.6 Config_IT000_ITL001

Initialize the interval timer for the sample code.

The interval timer interrupt (INTITL) is used to start the SMS in the sample code. Therefore, "Interrupt setting" is set to "Not used". "Interrupt Settings" can also be changed to "Use".

Since INTITL is masked by the R_Config_SMS_Start function, the CPU will not start even if INTITL is generated during STOP or SNOOZE mode. After returning from STOP mode and SNOOZE mode, INTITL is in a masked state, so unmask INTITL if necessary.

6.1.7 Config_TAU0_7

Set TAU07 of the sample code.

In the sample code, it is used as a chattering prevention of INTP0. When the INTP0 is not used, or chattering prevention is not needed, delate it.

6.1.8 Config_ADC

Initialize the ADC for the sample code.

In the sample code, "VREF(+) setting" is set to VDD and "A/D channel selection" is set to ANI0. It is also possible to change "A/D channel selection" to another ANI pin. And "the internal reference voltage" or "the temperature sensor output voltage" can be selected too. However, the A/D converter reference voltage current and temperature sensor operating current will flow during STOP mode in this case.

In the sample code, A/D conversion is not performed when the device is not in SOOZE mode, so "Interrupt Settings" is set to "Not Used". "Interrupt Settings" can also be changed to "Use". Since INTAD is masked by the R_Config_SMS_Start function, the CPU will not start even if INTAD is generated during STOP or SNOOZE mode. After returning from STOP mode and SNOOZE mode, INTAD is in a masked state, so unmask INTAD if necessary.

6.1.9 Config_SMS

Set the sample code SMS.

For details, refer to "6.2 r01an5610_sms_average.sms".

6.1.10 Config_UART0

Configure the UART settings in the sample code.

The UART setting in the smart configurator does not allow the setting "Do not use interrupt".

The sample code does not use the "UART0 transmission complete, free buffer interrupt", so the interrupt mask flag is set (STMK0 = 1) in the main function.

6.1.11 Pins

Set the pin functions.

In the sample code, we will use the UART0 transmit function, but the default setting for the TxD0 pin is P17. However, we need to assign the TxD0 function to P12 due to the configuration of the board, so we will change it.

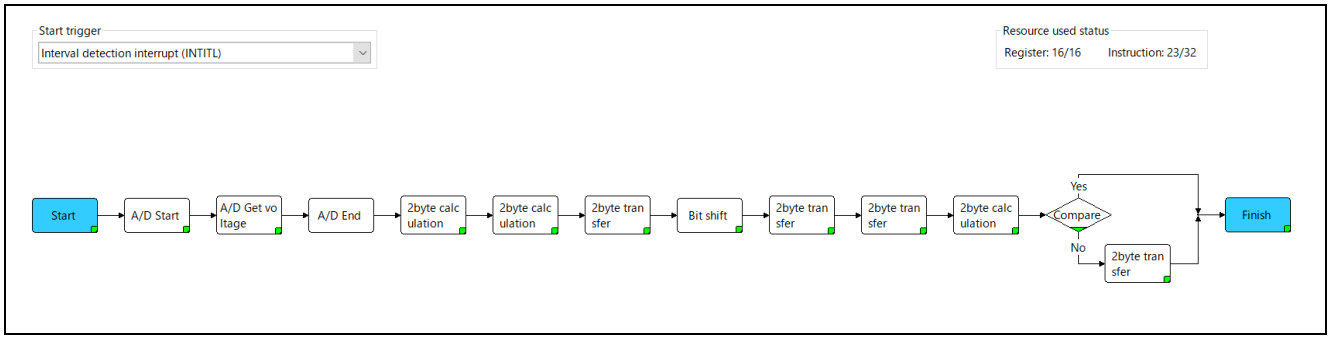
6.2 r01an5610_sms_average.sms

This is the data for Config_SMS alone. In the sample code, the interrupt of the interval timer is used to start SMS, and A/D is used in the operation of SMS. Note that it is necessary to set the interval timer and A/D separately.

The r01an5610_sms_average.sms can also be imported into the Smart Configurator of another project. After setting up the SMS component in another project, go to [Import SMS Sequence] -> [Browse] and select " r01an5610_sms_average.sms " to import it.

When imported into the smart configurator, the flow chart will be as shown in Figure 6-1. This flow chart is the same as "Figure 5-7 SMS process".

Figure 6-1 Config_SMS flow chart

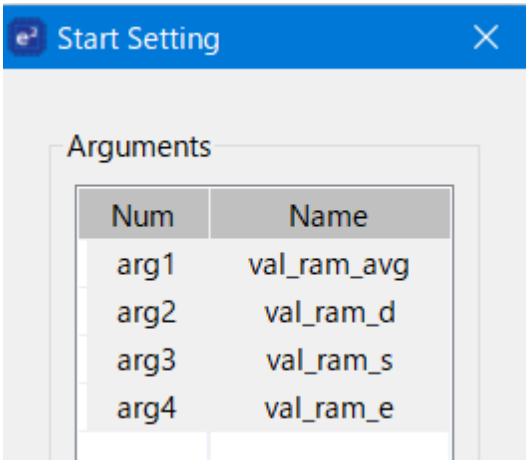


A description of each block is shown below.

6.2.1 Start

When the SMS starts, the values passed as arguments in the SMS start function (R_Config_SMS_Start function) are set to val_ram_avg (average value storage address), val_ram_d (data table pointer), val_ram_s (data table start address), and val_ram_e (data table final address).

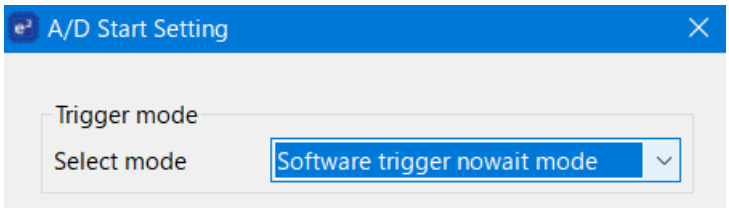
Figure 6-2 Start Setting



6.2.2 A/D Start

Set the A/D trigger mode. Waiting time is automatically added according to the mode.

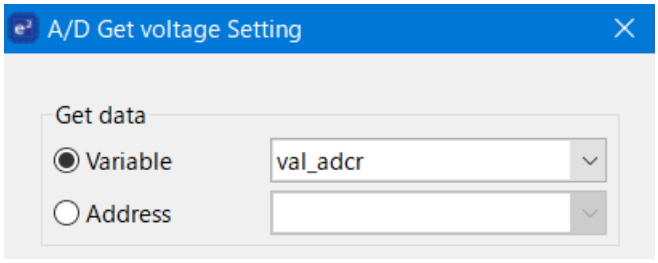
Figure 6-3 Start Setting



6.2.3 A/D Get voltage

Converts A/D and stores the value of the A/D conversion result (ADCR0) in the variable val_adcr.

Figure 6-4 A/D Get voltage Setting



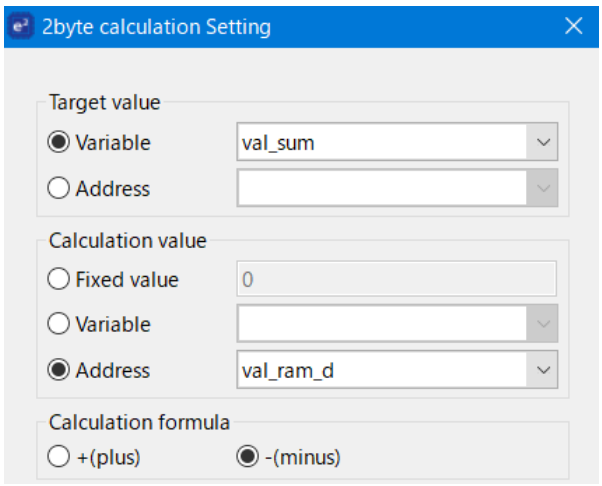
6.2.4 A/D End

End the A/D conversion.

6.2.5 2byte calculation

Subtract the value of val_ram_d (data table pointer), where the oldest A/D conversion result is stored, from the total value of val_sum until the last time, and store the result in val_sum.

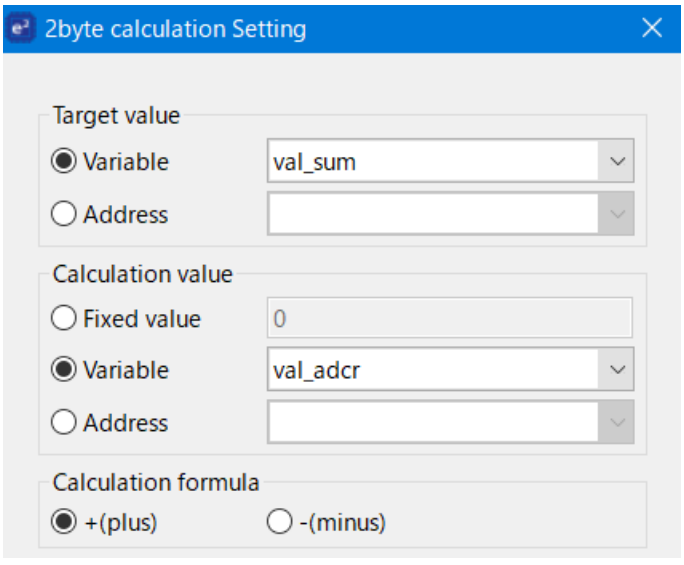
Figure 6-5 2byte calculation Setting



6.2.6 2byte calculation

Add the value of the acquired A/D conversion result val_adcr (variable for storing the ADCR register) to val_sum, and store the result in val_sum as the latest total value.

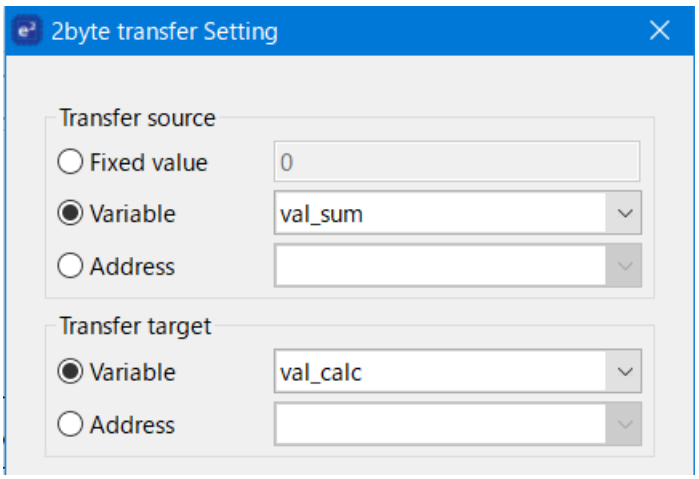
Figure 6-6 2byte calculation Setting



6.2.7 2byte transfer

Store the value of val_sum in val_calc (a variable for calculation).

Figure 6-7 2byte transfer Setting

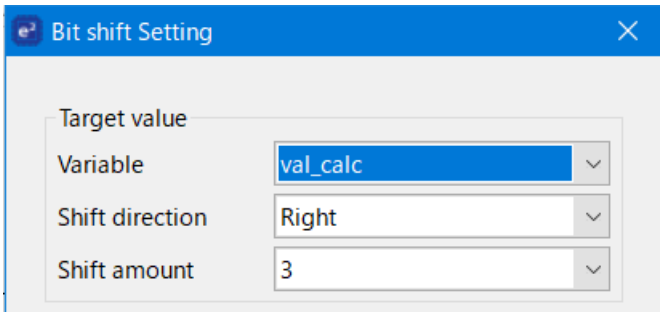


6.2.8 Bit shift

Perform a "divide by 8" by shifting val_calc, which contains the latest total value, by 3 bits to the right.

The result is stored in val_calc as the average value of the 8 pieces of data.

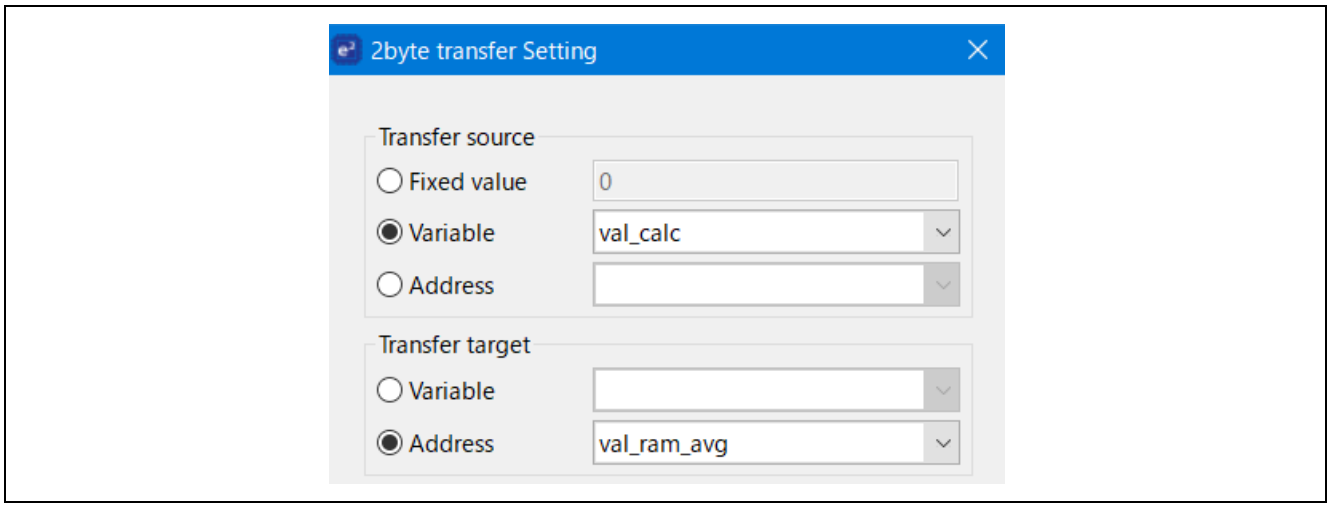
Figure 6-8 Bit shift Setting



6.2.9 2byte transfer

Store the value of val_calc in val_ram_avg (average value storage address).

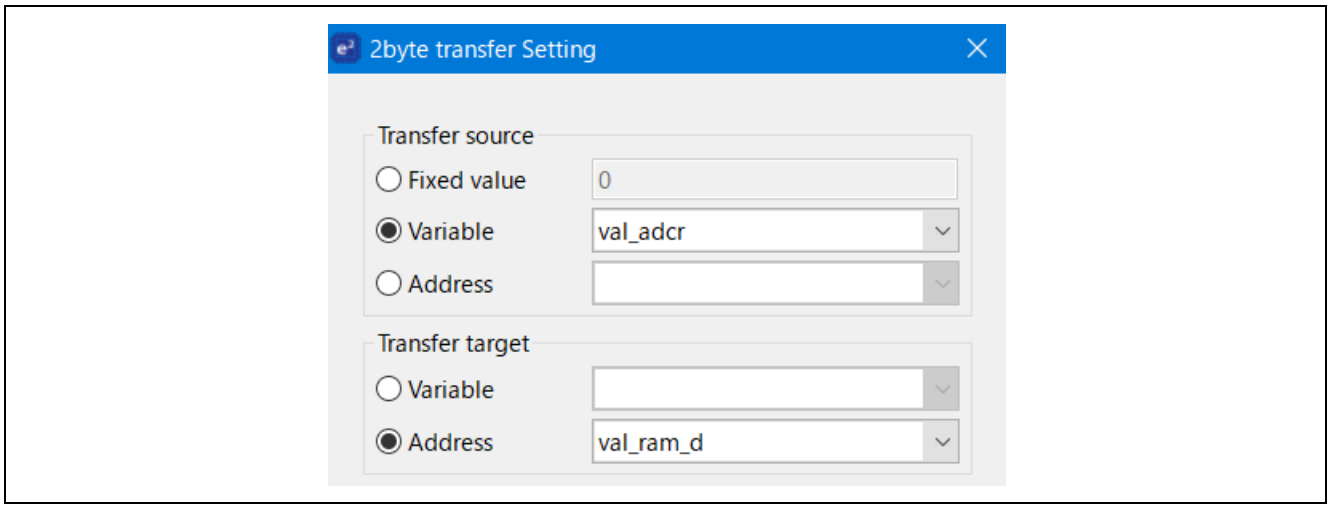
Figure 6-9 2byte transfer Setting



6.2.10 2byte transfer

Store the value of val_adcr in val_ram_d (data table pointer).

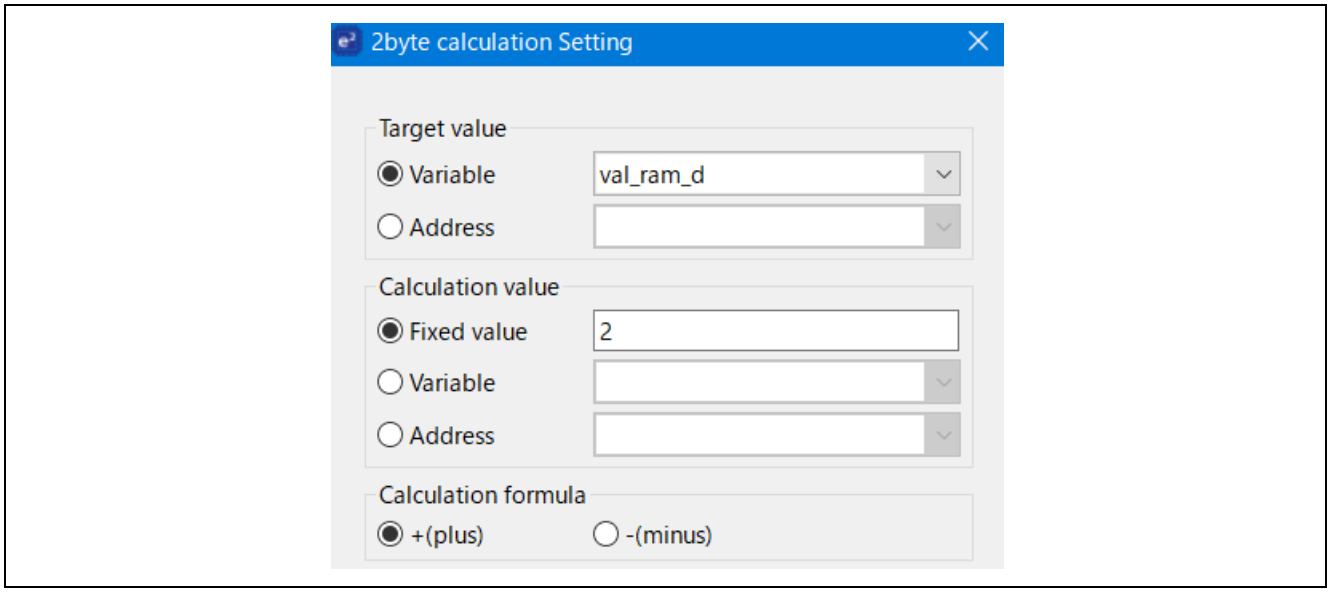
Figure 6-10 2byte transfer Setting



6.2.11 2byte calculation

Add "2" to val_ram_d (data table pointer) where the latest A/D conversion value is stored, and store the result in val_ram_d.

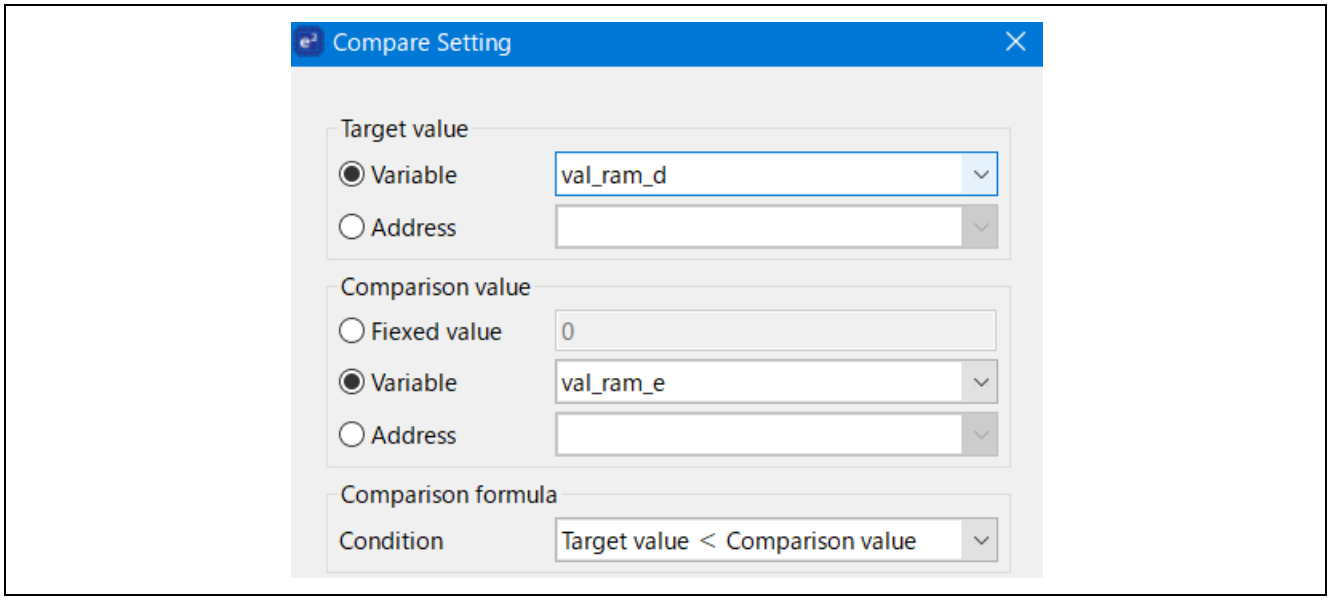
Figure 6-11 2byte calculation Setting



6.2.12 Compare

Compare whether the data table pointer stored in val_ram_d is smaller than the threshold value stored in val_ram_e (the last address of the data table). If val_ram_d is smaller than the threshold value, it will shift to STOP mode. If val_ram_d is larger than the threshold value, it performs the next 2-byte transfer and updates val_ram_d.

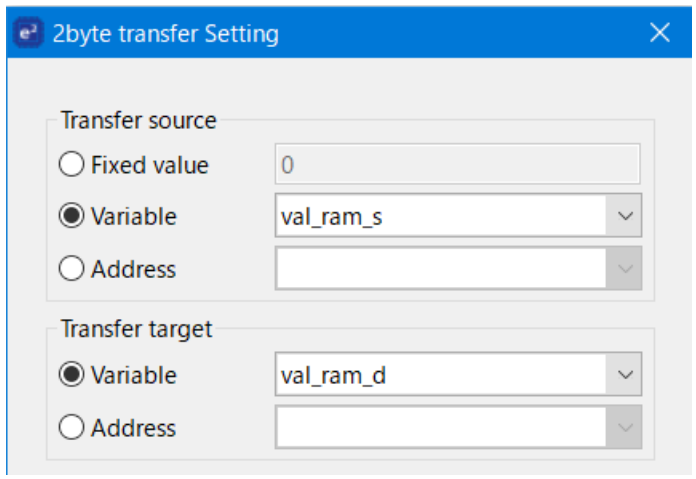
Figure 6-12 Compare Setting



6.2.13 2byte transfer

Store the value of val_ram_s (data table start address) in the transfer target val_ram_d.

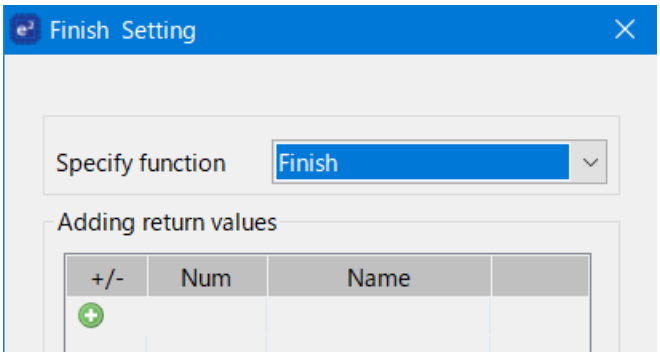
Figure 6-13 2byte transfer Setting



6.2.14 Finish

It shifts to STOP mode. In the sample code, the return value is not used.

Figure 6-14 Finish Setting



6.2.15 Variable Setting

The settings of the variables used in SMS are shown below.

Table 6-3 Variables used in SMS

Data name	Initialization mode	Initial value	Description
val_adcr	No initialization	-	Stores the A/D conversion result.
val_calc	No initialization	-	Stores a variable for calculation.
val_sum	No initialization	-	Stores the total value variable.
val_ram_avg	Pass argument via SMS start function	-	Stores the address where the average value is stored. The address of val_avg is set as an argument in the R_Config_SMS_Start function.
val_ram_d	Pass argument via SMS start function	-	Stores the pointer to the data table. The pointer to val_adc[0] is set as an argument in the R_Config_SMS_Start function.
val_ram_s	Pass argument via SMS start function	-	Stores the start address of the data table. The address of val_adc[0] is set as an argument in the R_Config_SMS_Start function.
val_ram_e	Pass argument via SMS start function	-	Stores the last address of the data table. The address of "val_adc[0] + (NUMBER_OF_CALC * 2)" is set as an argument in the R_Config_SMS_Start function.

6.3 How to change the number of data to be averaged

In the sample code, the number of data to be averaged is set to 8. This section explains how to change the number of data to be averaged.

Change the constant "NUMBER_OF_CALC" and the "Shift Amount" of the "Bit Shift Setting" shown in Figure 6-15 to the settings in Table 6-4. In this sample code, do not set any values other than those in Table 6-4.

```
#define NUMBER_OF_CALC      (8U)
```

Figure 6-15 Bit shift Setting

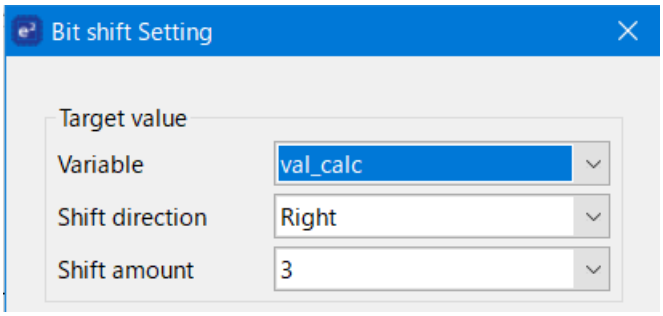


Table 6-4 Relationship between the number of data and the set value

Number of data	NUMBER_OF_CALC	Shit volume
2	2	1
4	4	2
8	8	3
16	16	4

In the sample code, the number of data to be averaged is 8, so "NUMBER_OF_CALC" is set to 8 and the shift amount is set to 3.

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference

RL78/G23 User's Manual: Hardware (R01UH0896E)

RL78 Family User's Manual: Software (R01US0015E)

SMS assembler User's Manual【Preliminary version】(R20UT4792J)

(The latest version can be downloaded from the Renesas Electronics website.)

Technical Update / Technical News

(The latest version can be downloaded from the Renesas Electronics website.)

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr.13.21	-	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/.