RL78/G23

Dynamically Controlling the Display of a 7-Segment LED Indicator by Using the SMS

Introduction

This application note describes a method to dynamically control the display of a four-digit 7-segment LED indicator by using the SNOOZE mode sequencer (SMS). The method described uses a cathode-common 7-segment LED indicator.

Target Device

RL78/G23

When using this application note with other Renesas's MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

In the method described in this application note, first, the CPU reads the minute and second data from the RTC and converts the data into the corresponding segment data. Then, the SMS reads the segment data of each digit every two milliseconds without CPU intervention to display minutes and seconds on a 7-segment LED indicator.

This method makes it possible to dynamically control the display of a 7-segment LED indicator even while the CPU is not operating. Because the SNOOZE mode sequencer (SMS) can operate at a lower current than the CPU, the power consumption of the system can be reduced by using the SMS as substitute for the CPU.

1.1 Configuration

Table 1-1 shows the peripherals used in the method and their purposes.

Table 1-1 Peripherals Used and Their Purposes

Peripheral	Purpose
32-bit interval timer	2-ms interval timer (SMS activation trigger)
P1	Port used to drive segment signals
P7	Port used to drive a signal for selecting the digit (P73, P72, P71, or P70)
RTC	Clock used to measure the time

The following describes the major settings that need to be specified.

① Initial settings for the 32-bit interval timer

Table 1-2 shows the initial settings for the ITL000.

Table 1-2 Initial Settings for the ITL000

Register Name	Set Value	Description
TML32EN	1	Supplies a clock to the 32-bit interval timer.
ITLCTL0	40H	Disables operation as a 16-bit counter.
ITLMKF0	1	Masks the compare match status for the channel (0).
ITLS0	00H	Clears the compare match signal for the channel (0).
ITLIF	0	Clears the interrupt request.
ITLCSEL0	01H	Selects fIHP for the counter clock (fITL0).
ITLFDIV00	00H	Uses fITL0 without dividing the frequency.
ITLCMP00	0F9FFH	Sets the interval time to 2 ms.

② Initial settings for the RTC

Table 1-3 shows the initial settings for the RTC.

Table 1-3 Initial Settings for the RTC

Register Name	Set Value	Description
OSMC	0x00	Specifies that the RTC operates as a subsystem clock.
RTCWEN	1	Disables operation as a 16-bit counter.
RTCMK	1	Masks the RTC interrupt.
RTCIF	0	Clears the RTC interrupt request.
RTCC0	0x02	Places the RTC in 12-hour mode and generates an interrupt at one-second intervals.
RTCC1	0x00	Disables generation of alarm interrupts.

1.2 Overview of Operation

This section provides an overview of operation of dynamically controlling the display of a 7-segment LED indicator.

The CPU is activated from the standby mode by an RTC periodic interrupt generated at one-second intervals and reads time information from the RTC. Then, the CPU creates the minute and second display data (segment data) from the time information and stores the created data in memory.

The SMS is activated by an interval detection interrupt generated at 2-ms intervals for the 32-bit interval timer and reads the segment data for the display-target digit from the memory and outputs the data to the port to which a 7-segment LED indicator is connected.

- The SMS is activated by an interval detection interrupt (INTITL) that is generated at 2-ms intervals.
- When the SMS is activated, it clears the compare match detection flag.
- The SMS turns off all digit selection signals.
- The SMS waits for the display blanking period to elapse.
- The SMS reads the digit display data and outputs the data to the port to which the 7-segment LED indicator is connected.
- The SMS waits for the data to be stable.
- The SMS outputs a digit selection signal.
- The SMS updates the display data address.
- The SMS shifts the digit selection signal and updates the address.
- When the processing for all digits is complete, the SMS initializes the display data storage address and digit selection signals.
- The SMS terminates processing and waits for the next interval detection interrupt (INTITL).

1.3 LED Control

A cathode-common 7-segment LED indicator is directly controlled by using a port without using a driver IC. The P10 to P17 pins are used to control segment signals and the P70 to P73 pins are used to control common signals. 0xFFF7 is set for port 7 as the initial value of the common signals.

2. SMS Usage Example

Operation of the sample code covered in this application note is verified under the conditions shown in the following table.

Table 2-1 Conditions Under Which Operation Was Verified

Item	Description
MCU used	RL78/G23(R7F100GLG)
Operating frequency	• High-speed on-chip oscillator clock (fIH): 32MHz • CPU/Peripheral Hardware Clock: 32MHz
Operating voltage	• During VDD operation:3.3V • LVD0 detection voltage: reset mode At rising edge TYP. 1.875V At falling edge TYP. 1.835V
Integrated development environment (CS+)	CS+ for CC V8.07.00 from Renesas Electronics Corp
C compiler (CS+)	CC-RL V1.11 from Renesas Electronics Corp
Integrated development environment (e2 studio)	e2 studio 2021-04 (21.4.0) from Renesas Electronics Corp
C compiler (e2 studio)	CC-RL V1.11 from Renesas Electronics Corp
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.21.1 from IAR Systems
C compiler (IAR)	
Smart Configurator	V.1.0.1
Board Support Package(r_bsp)	V.1.10
Emulator	CS+, e2 studio: E2 emulator Lite IAR: E2 emulator Lite
Board used	RL78/G23-64 Fast Prototyping Board (RTK7RLG230CLG000BJ)

3. Related Application Notes

See also the following application notes, which are related to this application note:

RL78 Smart Configurator User's Guide: CS+(R20AN0580E)
RL78 Smart Configurator User's Guide: e² studio(R20AN0579E)
RL78 Smart Configurator User's Guide: IAREW(R20AN0581E)

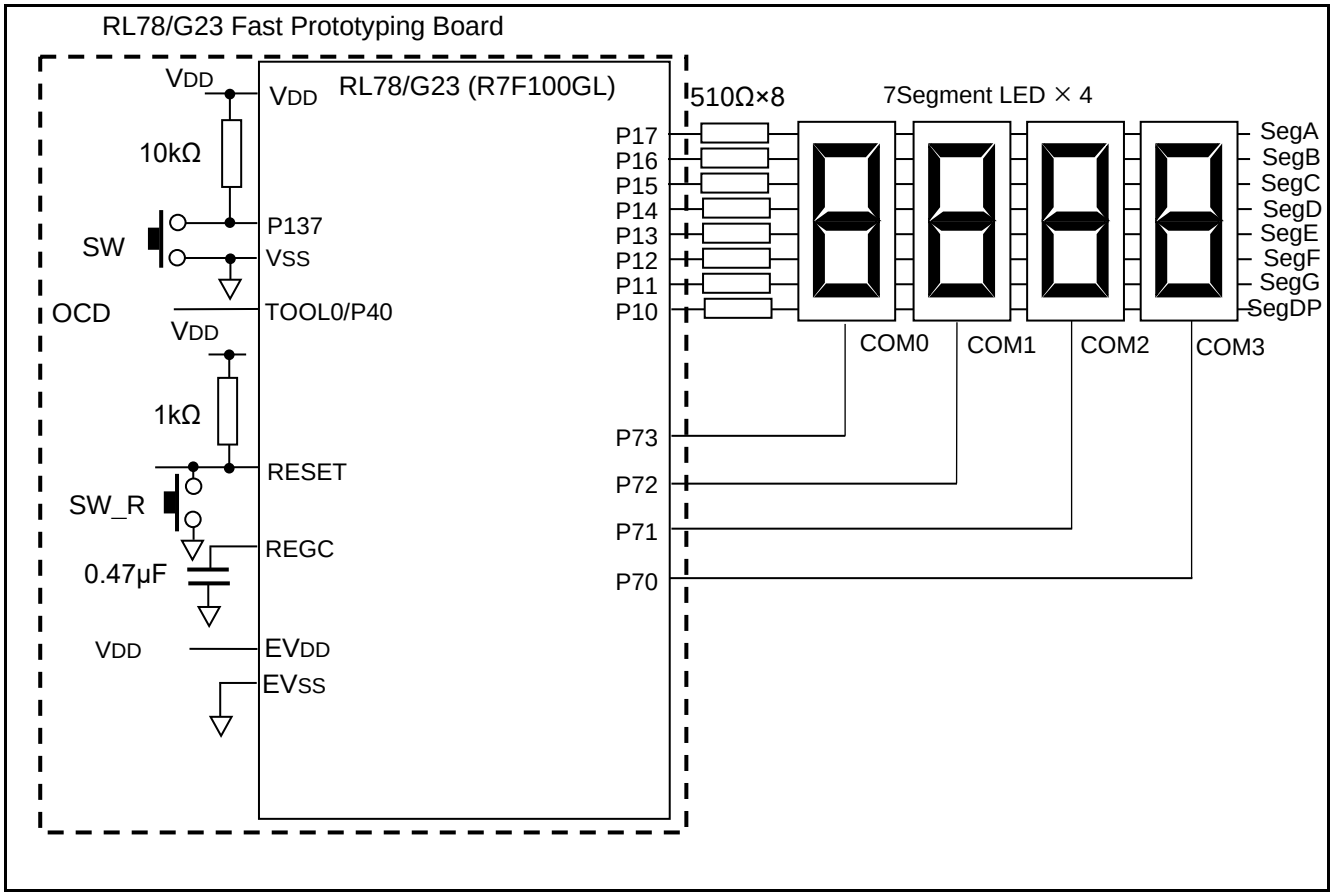
4. Description of the Hardware

4.1 Hardware Configuration Example

This section shows a case in which the SMS performs some processing (such as peripheral control, arithmetic operation, and judgment) instead of the CPU.

Figure 4-1 shows an example hardware configuration used in this application note.

Figure 4-1 Hardware Configuration



Notes: 1. The purpose of this circuit is only to provide the connection outline and the circuit is simplified accordingly.
When designing and implementing an actual circuit, provide proper pin treatment and make sure that the
hardware's electrical specifications are met (connect the input-dedicated ports separately to V_{DD} or V_{SS} via a resistor).

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

4.2 List of Pins to be Used

Table 4-1 shows the pins used and their functions.

Table 4-1 Pins Used and Their Functions

Pin Name	I/O	Function
P17	Output	Segment A control signal
P16	Output	Segment B control signal
P15	Output	Segment C control signal
P14	Output	Segment D control signal
P13	Output	Segment E control signal
P12	Output	Segment F control signal
P11	Output	Segment G control signal
P10	Output	Segment DP control signal
P73	Output	Digit 0 (10-minute digit) selection signal
P72	Output	Digit 1 (1-minute digit) selection signal
P71	Output	Digit 2 (10-second digit) selection signal
P70	Output	Digit 3 (1-second digit) selection signal

5. Description of the Software

5.1 Operation Outline

The CPU exits from the standby mode in response to an RTC periodic interrupt generated at one-second intervals and reads time information from the RTC. Then, the CPU creates the minute and second display data (segment data) from the time information and stores the created data in memory.

The SMS is activated by an interval detection interrupt generated at 2-ms intervals for the 32-bit interval timer. Each time the SMS is activated, it reads the segment data for each display digit from the memory and outputs the data to the port to which a 7-segment LED indicator is connected.

Table 5-1 shows an overview of the operation of this sample code.

Table 5-1 Overview of Operation

No.	Operation of Each Component		
	CPU	SMS	RTC
(1)	Stores the SMS processing commands in a register.	--	--
(2)	Turns on the RTC.	--	Runs a time counter.
(3)	Turns on the SMS.	Waits for an activation trigger.	"
(4)	Turns on the 32-bit interval timer.	"	"
(5)	Enters HALT mode.	Activates the SMS upon detecting a compare match for TML32.	"
(6)	"	Clears the compare match detection flag for TML32 (ITLS0 register).	"
(7)	"	Turns off a digit selection signal to turn off the corresponding digit of the LED indicator.	"
(8)	"	Waits for 1 µs.	"
(9)	"	Reads the segment data for the next digit from the memory and outputs the data from the P1 port.	"
(10)	"	Waits for 1 µs.	"
(11)	"	Turns on the digit selection signal to turn on the corresponding digit of the LED indicator.	"
(12)	"	Updates the memory address to the address of the next digit.	"
(13)	"	Updates the digit selection signal.	"
(14)	"	Goes to step (17) if the selected digit is not the last digit, or goes to step (15) if the selected digit is the last digit.	"
(15)	"	Moves the memory address to the beginning.	"
(16)	"	Resets the digit data to the initial value.	"
(17)	"	Enters a wait state and goes to step (5).	"
(18)	Reads RTC data.	--	Periodically generates an interrupt.
(19)	Creates segment data.	--	Runs the time counter.
(20)	Goes back to step (5) and enters HALT mode.	--	"

": Same as above

--: No operation is performed.

5.2 Folder Structure

Table 5-2 shows the structure of the source files and header files that are used by the sample code. Note that the table does not include files that are automatically generated in the integrated development environment and files for the BSP environment.

Table 5-2 Folder Structure

Folder/File Name	Description	Remarks
¥r01an6429jj0100_sms_dynamic<DIR>	Folder for the sample code	
main.c	Source file of the sample code	
¥src<DIR>	Folder storing programs	
¥smc_gen<DIR>*2	Folder generated by Smart Configurator	
¥Config_ITL000_ITL001<DIR>	Folder storing the programs for the TML32	
Config_ITL000_ITL001.c	Source file for the TML32	
Config_ITL000_ITL001.h	Header file for the TML32	
Config_ITL000_ITL001_user.c	Interrupt source file for the TML32	Unused
¥Config_PORT<DIR>	Folder storing the programs for ports	
Config_PORT.c	Source file for ports	
Config_PORT.h	Header file for ports	
Config_PORT_user.c	User processing source file for ports	Unused
¥Config_RTC<DIR>	Folder storing the programs for the RTC	
Config_RTC.c	Source file for the RTC	
Config_RTC.h	Header file for the RTC	
Config_RTC_user.c	Interrupt source file for the RTC	
¥Config_SMS<DIR>	Folder storing the programs for the SMS	
Config_SMS.c	Source program for the SMS	
Config_SMS.h	Header file for the SMS	
Config_SMS_ASM.smsasm	ASM source file for the SMS	
Config_SMS_user.c	Interrupt source file for the SMS	Unused
¥general<DIR>	Folder storing the initialization and other common programs	
¥r_bsp<DIR>	Folder storing the BSP program	
¥r_config<DIR>	Folder storing the BSP_CFG program	

In the preceding table, the symbol <DIR> indicates that the item is a directory.

*2: The folder structure for the IAR version of sample code is different from the structure shown in the preceding table. For details about the folder structure for the IAR version of sample code, see the IAR version of sample code. The folder structure for the IAR version of sample code includes r01an6429jj0110_sms_dynamic.ipcf. For details, see the "RL78 Smart Configurator User Guide: IAR (R20AN0581)".

5.3 List of Option Byte Settings

Table 5-3 lists the option byte settings.

Table 5-3 Option Byte Settings

Address	Value	Description
000C0H / 010C0H	11101111B	Stops operation of the watchdog timer. (The counter stops after a reset is canceled.)
000C1H / 010C1H	11111011B	3.3 V (operable in the range from 2.82 to 5.5 V) LVD0 detection voltage: Reset mode At a rising edge: TYP. 2.97 V (2.88 to 3.06 V) At a falling edge: TYP. 2.91 V (2.82 to 3.00 V)
000C2H / 010C2H	11101000B	HS mode; high-speed on-chip oscillator at 32 MHz
000C3H / 010C3H	10000100B	Enables on-chip debugging.

5.4 List of Constants

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

Table 5-4 Constant Used in the Sample Code

Constant	Value	Description	Function Used
SEG_TABLE[]		Segment data conversion table	main.c
INT_READY	0x01	An RTC periodic interrupt occurred.	Config_RTC_user.c
NOT_INT_READY	0x00	No RTC periodic interrupt occurred.	main.c

5.5 List of Variables

Table 5-5 lists the global variables that are used in the sample code.

Table 5-5 Global Variables

Type	Variable Name	Description	Function Used
st_rtc_counter_value_t	g_RTC_init_data	RTC initialization data	main
st_rtc_counter_value_t	g_RTC_read_data	Data read from the RTC	main
uint8_t	g_RTC_INT_flag	Flag for whether 1 second has elapsed	main, r_Config_RTC_callback_constperiod
uint8_t	g_disp_data[4]	Segment data	main, (SMS)

5.6 List of Functions

Table 5-6 lists the functions that are used in the sample code. For functions that are generated by Smart Configurator, this table includes only the ones that are modified.

Table 5-6 Functions

Function Name	Outline	Function Used
main	Main processing	main.c
R_MAIN_UserInit	Initializes the startup variables for the peripherals used.	main.c
r_conv_2digit	Converts 8-bit BCD data to 2-digit segment data.	main.c
r_conv_SEG	Converts time information into the display-target segment data.	main.c
r_Config_RTC_callback_constperiod	Generates an RTC periodic interrupt.	Config_RTC_user.c

5.7 Specification of Functions

The function specifications of the sample code are shown below.

[Function Name]	main()
Outline	This function main function.
Header	r_smc_entry.h (Including r_cg_macrodriver.h, Config_RTC.h,r_cg_userdefine.h, etc.)
Declaration	void main(void)
Description	This function turns on the necessary peripherals, and then waits in HALT mode for an interrupt from the SMS or RTC. In response to an RTC periodic interrupt that is generated at 1-second intervals, this function reads time information from the RTC and converts the time information into segment data.
Argument	None
Return Value	None

[Function Name]	R_MAIN_UserInit ()
Outline	This function execute user code to start hardware before main function.
Header	r_smc_entry.h (Including r_cg_macrodriver.h, Config_RTC.h,r_cg_userdefine.h, etc.)
Declaration	void R_MAIN_UserInit (void)
Description	This function is called at the beginning of the main function and turns on the necessary peripherals. After turning on the RTC, this function writes time data. It then turns on the SMS and the timer that is used as an SMS activation trigger.
Argument	None
Return Value	None

[Function Name]	r_conv_2digit()	
Outline	This function get segment data of 2 digit	
Header	r_smc_entry.h (Including r_cg_macrodriver.h, Config_RTC.h,r_cg_userdefine.h, etc.)	
Declaration	void r_conv_2digit(uint8_t data,uint8_t digit)	
Description	This function converts the BCD data specified in the first argument into segment data and stores the conversion results in the segment data storage buffer (g_disp_data) element that corresponds to the digit specified in the second argument.	
Argument	uint8_t data	BCD data
	uint8_t digit	Pointer to the buffer element that stores the converted segment data
Return Value	None	

[Function Name]	r_conv_SEG()	
Outline	This function convert low 4 bit to segment data	
Header	r_smc_entry.h (Including r_cg_macrodriver.h, Config_RTC.h,r_cg_userdefine.h, etc.)	
Declaration	uint8_t r_conv_SEG(uint8_t data)	
Description	This function converts the lowest four bits of the BCD data specified in the argument into the corresponding segment data.	
Argument	uint8_t data	BCD data
Return Value	uint8_t	Segment data value

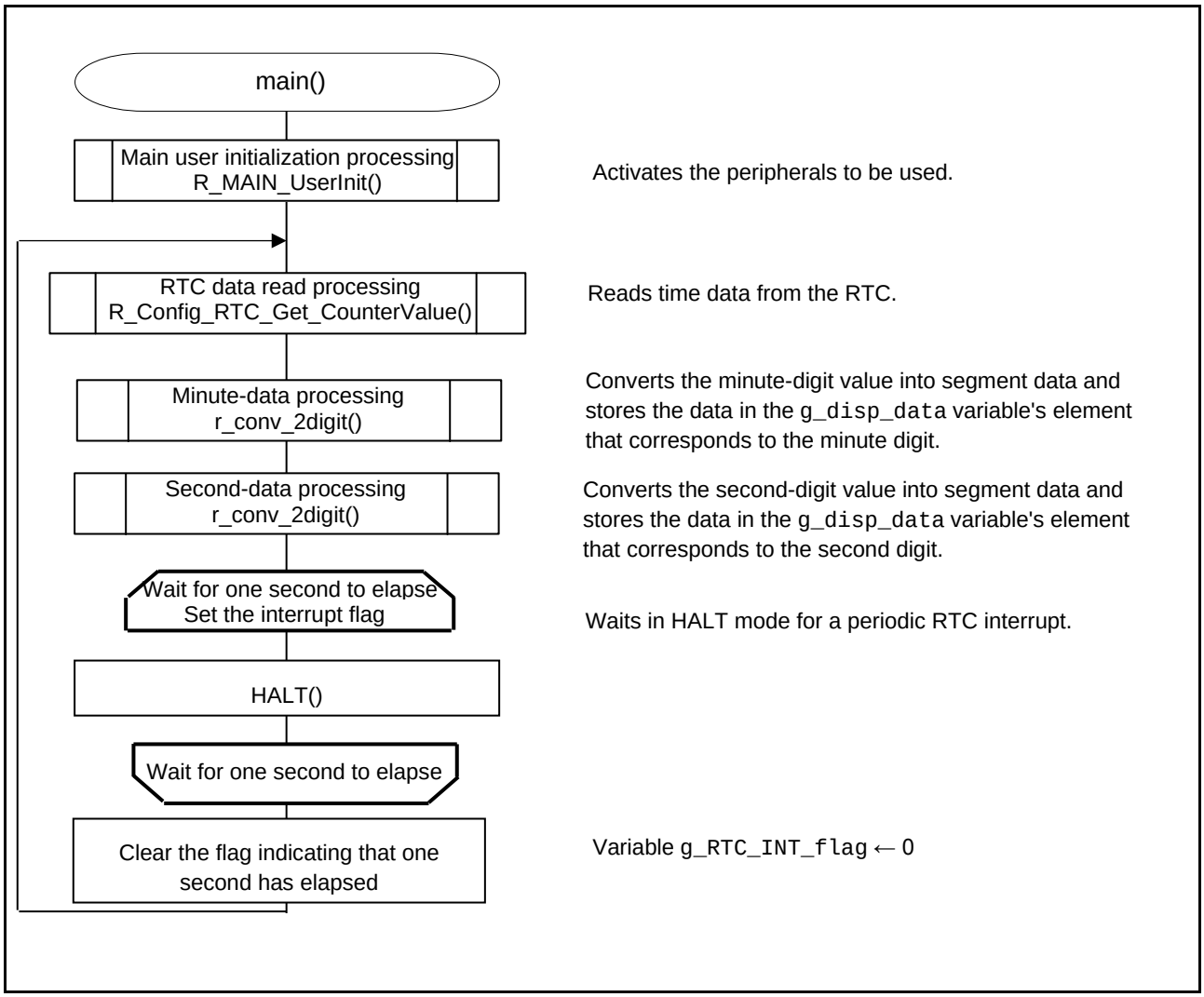
[Function Name]	r_Config_RTC_callback_constperiod()
Outline	This function is real-time clock constant-period interrupt service handler.
Header	r_cg_macrodriver.h, r_cg_userdefine.h, Config_RTC.h
Declaration	static void r_Config_RTC_callback_constperiod(void)
Description	This function sets a flag (g_RTC_INT_flag) in response to an RTC periodic interrupt.
Argument	None
Return Value	None

5.8 Flowcharts

5.8.1 Main Processing

Figure 5-1 shows a flowchart of the main processing.

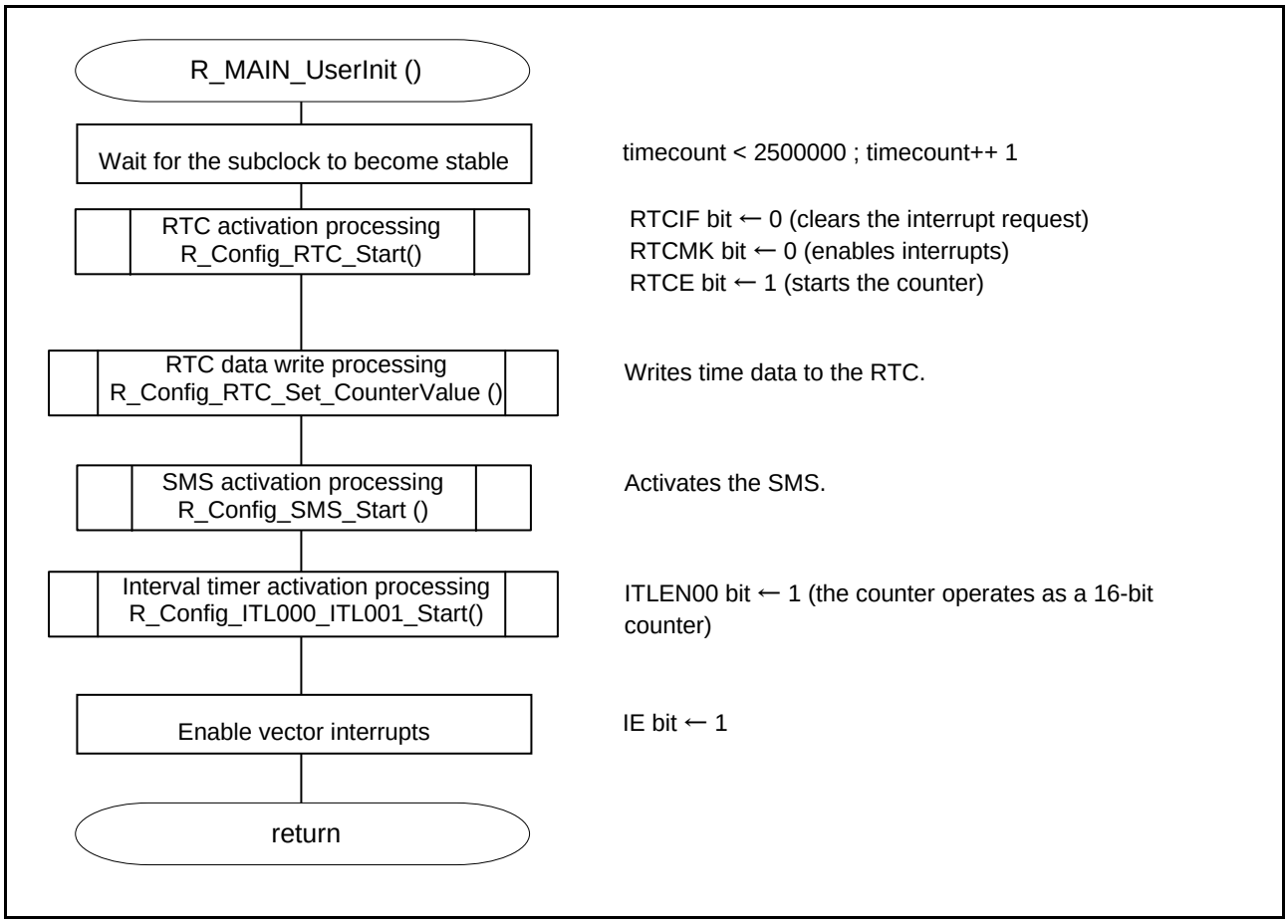
Figure 5-1 Main Processing



5.8.2 Main User Initialization Processing

Figure 5-2 shows a flowchart for the main user initialization processing.

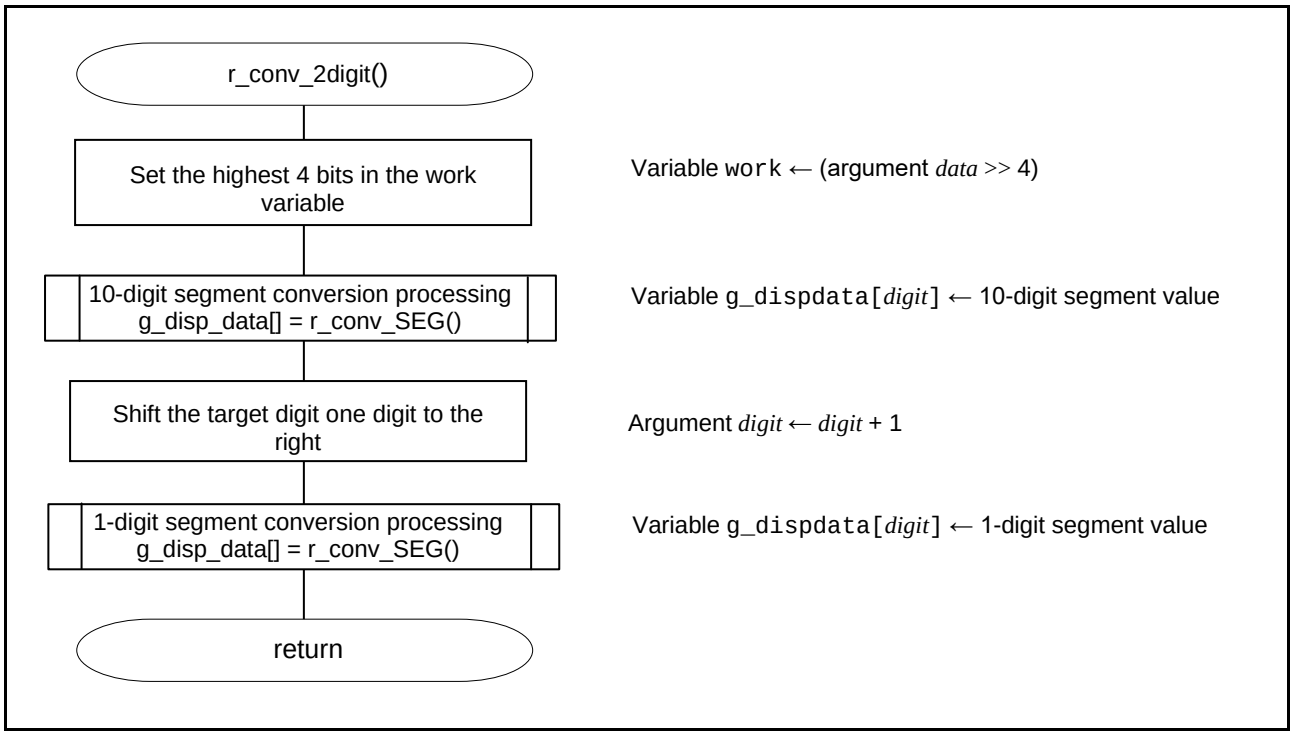
Figure 5-2 Main User Initialization Processing



5.8.3 2-Digit Minute Segment Conversion Processing

Figure 5-3 shows a flowchart for the 2-digit minute segment conversion processing.

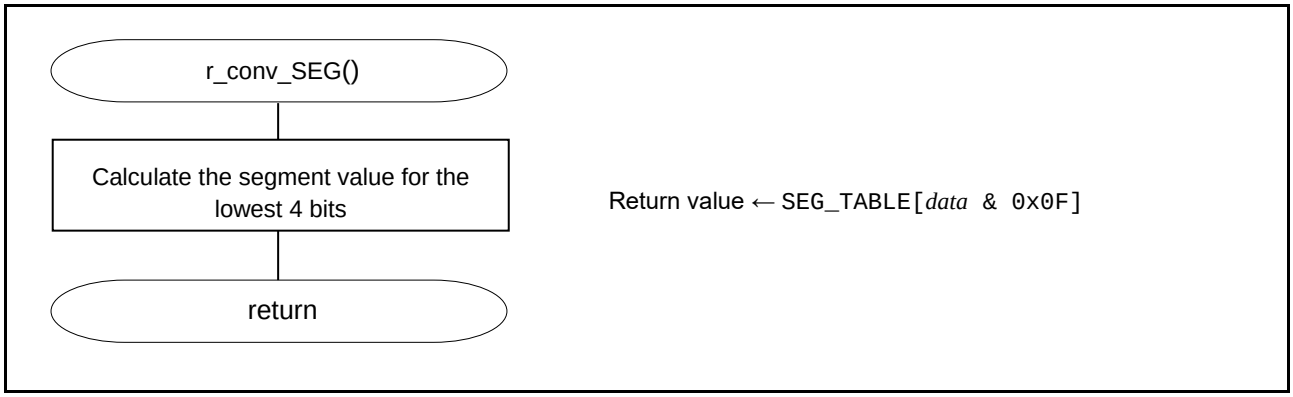
Figure 5-3 2-Digit Minute Segment Conversion Processing



5.8.4 Segment Data Conversion Processing

Figure 5-4 shows a flowchart for the segment data conversion processing.

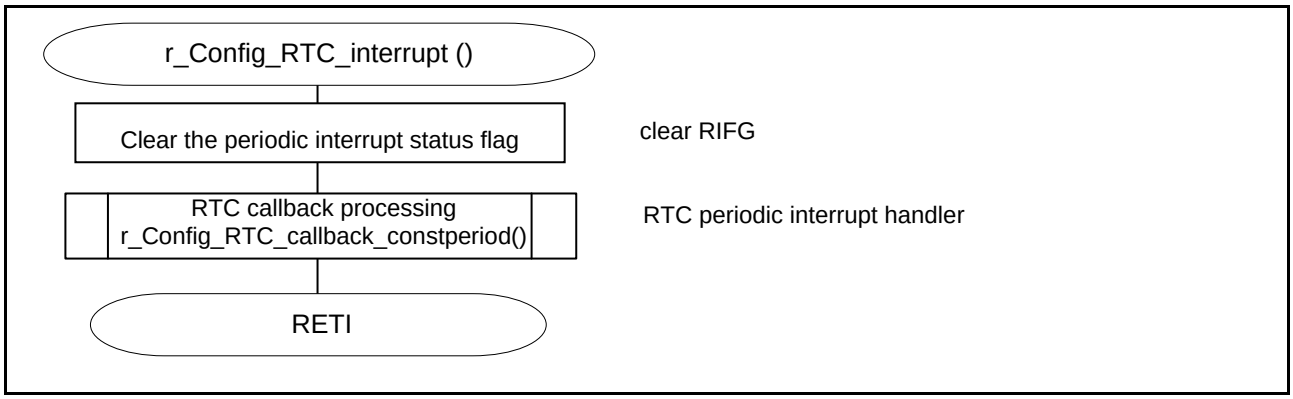
Figure 5-4 Segment Data Conversion Processing



5.8.5 RTC 1-Second Interrupt Processing

Figure 5-5 shows a flowchart for the RTC 1-second interrupt processing.

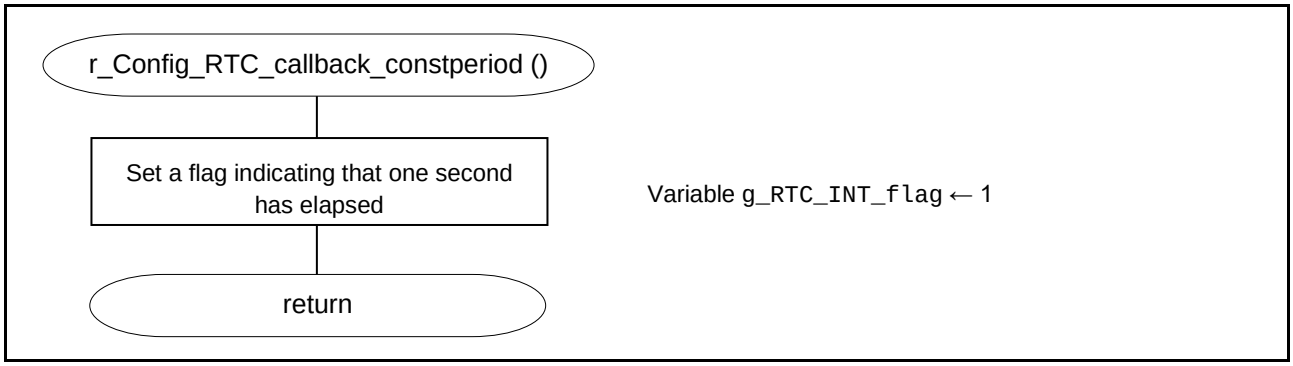
Figure 5-5 RTC 1-Second Interrupt Processing



5.8.6 RTC Callback Processing

Figure 5-6 shows a flowchart for the RTC callback processing.

Figure 5-6 RTC Callback Processing



5.9 Settings for the SNOOZE Mode Sequencer

When the SNOOZE mode sequencer (SMS) is started by occurrence of a triggering event, it sequentially executes the processing commands that are stored in the sequencer instruction register (SMSI0-31). During execution of these commands, the sequencer general-purpose register (SMSG0-15) is used to store the source address, destination address, arithmetic data, and other data.

The SMSI0-31 and SMSG0-15 registers are set by coding an SMS program (.SMSASM file) in assembly language. You can also use the SNOOZE Mode Sequencer component of Smart Configurator to create an SMS program by combining processing blocks. The created SMS program is converted to C language by the assembler for the SMS and then incorporated into the application program.

The following shows the specifications of the SMS processing that is performed by the sample code.

Synopsis	SMS processing
Explanation	Each time a TML32 interval detection interrupt (INTITL) occurs, the SMS is activated and displays a four-digit numeric value on a 7-segment LED indicator. At this time, the SMS drives the four digits on a one-by-one basis by controlling the data set in memory (by the main function) and ports (P1 and P7).
Argument*	DATAPOINTER, pointer
Return value	None
Remarks	None

* These arguments are specified in the R_Config_SMS_Start function. For details, see 6.2.1.

Figure 5-7 shows a flowchart for the SMS processing.
Table 5-7 to Table 5-8 show the registers and values that control the SNOOZE mode sequencer.

Figure 5-7 SMS Processing

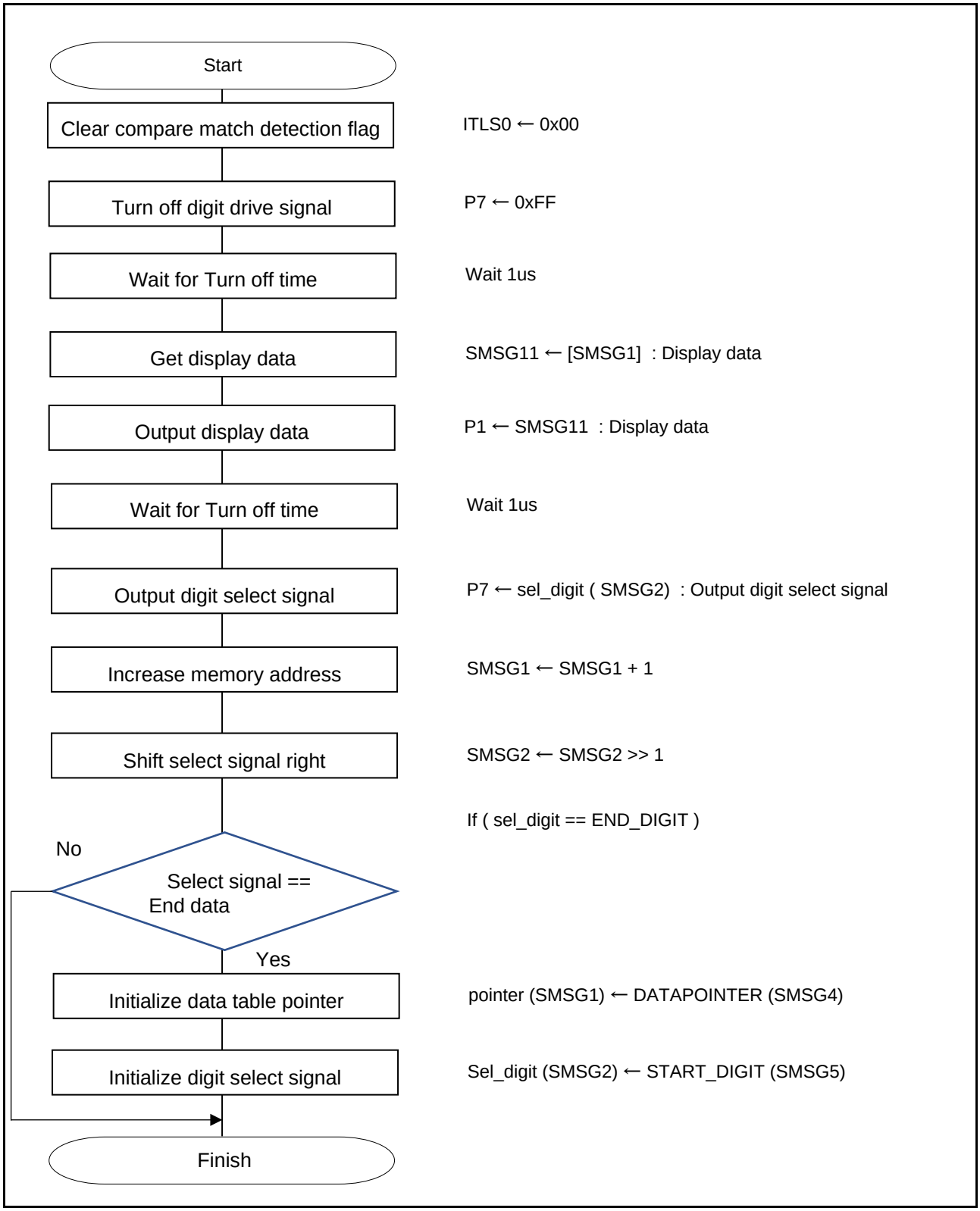


Table 5-7 Sequencer General-Purpose Register 0-15

Register Name	Setting Value	Description
SMSG0	0x0000	Fixed value: 0000H
SMSG1	0x0000	Display data pointer
SMSG2	0xFFF7	Digit selection data
SMSG3	0x0FFF	Comparison data for the last digit
SMSG4	0x0000	Initial value of the display data pointer
SMSG5	0xFFF7	Initial value of the digit selection data
SMSG6	0x036B	Address of the ITLS0 register
SMSG7	0x03C1	Address of the SMSG1 register
SMSG8	0xFF01	Address of the P1 register
SMSG9	0x00FF	Data for clearing digit selection
SMSG10	0x0001	Display data update width (1 byte)
SMSG11	0x0000	Unused
SMSG12	0x0000	Unused
SMSG13	0x0000	Unused
SMSG14	0x0000	Unused
SMSG15	0xFFFF	Fixed value: FFFFH

Table 5-8 Sequencer Instruction Register 0-31

Register Name	Setting Value	Description
SMSI0	0x0600	MOV [SMSG6+0], SMSG0
SMSI1	0x0896	MOV [SMSG8+6], SMSG9
SMSI2	0x9200	WAIT 32, 0
SMSI3	0x11B0	MOV SMSG11, [SMSG1+0]
SMSI4	0x08B0	MOV [SMSG8+0], SMSG11
SMSI5	0x9200	WAIT 32, 0
SMSI6	0x0826	MOV [SMSG8+6], SMSG2
SMSI7	0x71A0	ADDW SMSG1, SMSG10
SMSI8	0x7203	SHRW SMSG2
SMSI9	0x7232	CMPW SMSG2, SMSG3
SMSI10	0x8033	BNZ $3
SMSI11	0x2740	MOVW [SMSG7+0], SMSG4
SMSI12	0x0752	MOV [SMSG7+2], SMSG5
SMSI13	0xF000	FINISH
SMSI14-31	0x0000	Unused

6. Application Examples

This application note also covers the following Smart Configurator settings files in addition to the sample code:

r01an6429_sms_dynamic.scfg

r01an6429_sms_dynamic.sms

The following sections describe the preceding files, show setting examples, and provide notes.

6.1 r01an6429_sms_dynamic.scfg

The r01an6429_sms_dynamic.scfg file is the Smart Configurator settings file used in the sample code. This file covers all Smart Configurator settings. In the sample code, these settings are specified as shown in the following table.

Table 6-1 Smart Configurator Settings

Tab Name	Component	Description
Clocks	--	Operation mode: High-speed main mode at 2.7 to 5.5 V EVDD setting: 2.7 V ≤ EVDD0 < 5.5 V High-speed on-chip oscillator: 32 MHz fIHP: 32 MHz fCLK: 32,000 kHz (high-speed on-chip oscillator) fSXP: 32.768 kHz (Subsystem Clock)
System	--	Operation of on-chip debugging: Emulator is used Pseudo RRM/DMM function: Unused Start and Stop functions: Unused Trace function: Unused Whether to set the security ID: Yes Security ID: 0x00000000000000000000 Action to be taken if authentication by the security ID fails: Erase data from the flash memory.
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: Reset mode Voltage detection: Voltage at which to cause a reset (VLVD0): 2.91 V

Table 6-2 Smart Configurator Setup Values

Tab Name	Component	Description
Component	Config_ITL000_ITL001	Component: Interval timer Operation mode: 16-bit counter mode Resource: ITL000_ITL001 Operation clock: fIHP Clock source: fITL0 Interval time: 2 ms Interrupt settings: Unused
	Config_RTC	Component: Real time clock Clock source: Subsystem clock XR (fSXR) Time mode (12-hour/24-hour): 12-hour Initial RTC setting: 2022-04-01 12:00:00 Setting to enable output from the RTC1HZ pin (1 Hz): Unused Alarm detection function: Unused Interrupt settings: Periodic interrupt function (INTRTC): Once per second Priority level: Level 3 (low priority level)
	Config_SMS	Component: SNOOZE mode sequencer Activation trigger: Interval detection interrupt (INTITL)
	Config_PORT	Port selection: PORT1 or PORT7 Port mode: Reading the value of the P*mn* register PORT1: All pins are set for output. PORT7: Pins P70 to P73 are set for output with a value of "1".

6.1.1 Clocks

The clocks to be used in the sample code can be set.

In the sample code, the fCLK clock is set to 32,000 kHz for use by a 7-segment dot-matrix LED indicator. Therefore, the operation mode is set to "high-speed main mode at 2.7 to 5.5 V". Changing the settings requires prior consideration of whether the change is appropriate.

6.1.2 System

The on-chip debug settings for the sample code can be specified.

The settings for the operation of on-chip debugging and action to be taken if authentication by the security ID fails affect the setting that enables on-chip debugging in "Table 5-3 Option Byte Settings". Changing the settings requires prior consideration of whether the change is appropriate.

6.1.3 r_bsp

The sample code startup settings can be specified.

6.1.4 Config_LVD0

The power management settings for the sample code can be specified.

These settings affect the LVD0 detection voltage setting in "Table 5-3 Option Byte Settings". Changing the settings requires prior consideration of whether the change is appropriate.

6.1.5 Config_IT000_ITL001

The initial settings for the interval timer of the sample code can be specified.

In the sample code, an interval detection interrupt (INTITL) is used to activate the SMS. Therefore, the interrupt settings are disabled (Unused). The interrupt settings can also be enabled. Because INTITL is masked by the R_Config_SMS_Start function, the CPU in HALT mode does not start even if INTITL occurs. After the CPU exits from HALT mode, unmask INTITL when necessary.

6.1.6 Config_RTC

The RTC settings for the sample code can be specified.

In the sample code, an INTRTC interrupt generated at one-second intervals and a time counter are used to obtain the minute and second information. The CPU uses an INTRTC interrupt to exit from HALT mode.

6.1.7 Config_SMS

The SMS settings for the sample code can be specified.

For details, see "6.2 r01an6429_sms_dynamic.sms".

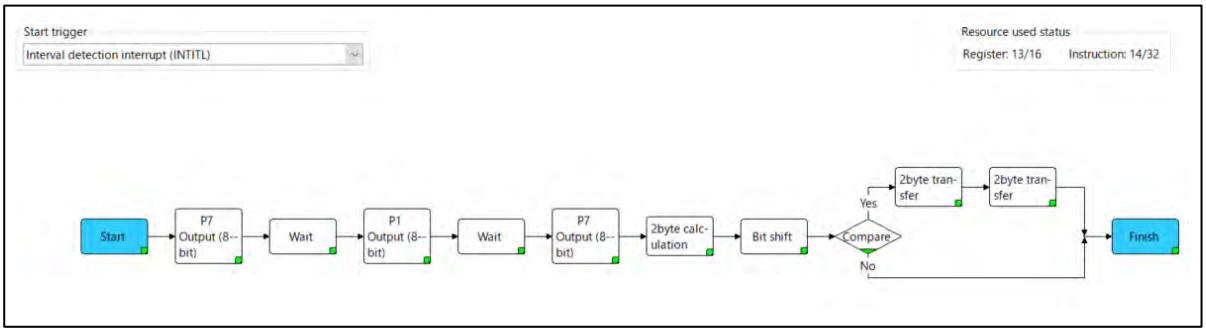
6.2 r01an6429_sms_dynamic.sms

The r01an6429_sms_dynamic.sms file contains the data for only the Config_SMS component. In the sample code, an interval detection interrupt (INTITL) is used to activate the SMS. Also, the RTC and ports are used for CPU activation and processing. Therefore, the interval timer, RTC, and port settings must be specified additionally.

The r01an6429_sms_dynamic.sms file can be imported into the Smart Configurator of another project. To import the file, after you have set up the SMS component in the other (destination) project, select [Import] > [Browse], and then select "r01an6429_sms_dynamic.sms".

When the file is imported into Smart Configurator, the flowchart shown in Figure 6-1 is displayed. The processing of this flowchart is the same as the processing of the flowchart shown in "Figure 5-7 SMS Processing ".

Figure 6-1 Flowchart Displayed in Config_SMS

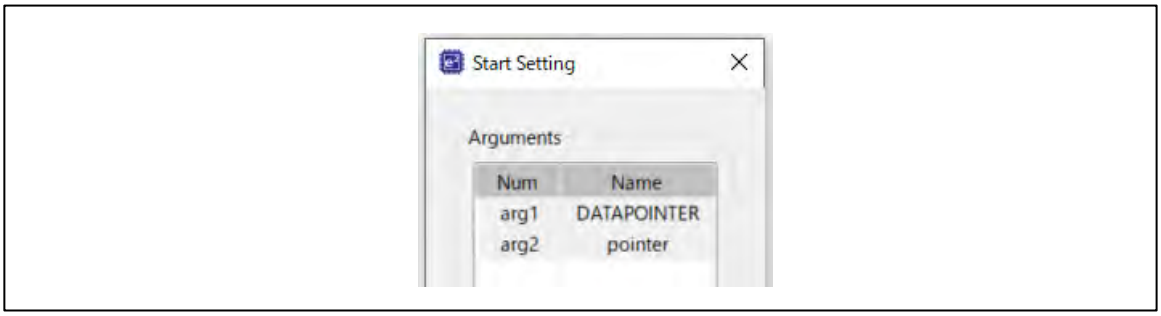


The blocks in the preceding figure are described in the following subsections.

6.2.1 Start Settings

When the SMS is activated, the SMS start function (R_Config_SMS_Start) uses the value passed via the DATAPOINTER argument to set the start address of the display data table, and uses the value passed via the "pointer" argument to set the address from which to read data in the data table.

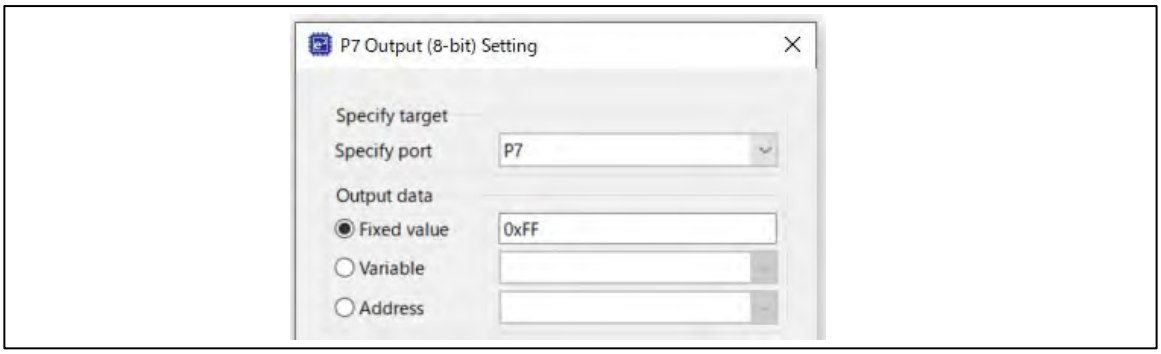
Figure 6-2 Start Settings



6.2.2 P7 Output (8-bit) Settings

To turn off the 7-segment LED indicator, 0xFF is output to P7 (digit selection signal) to stop driving the common signals.

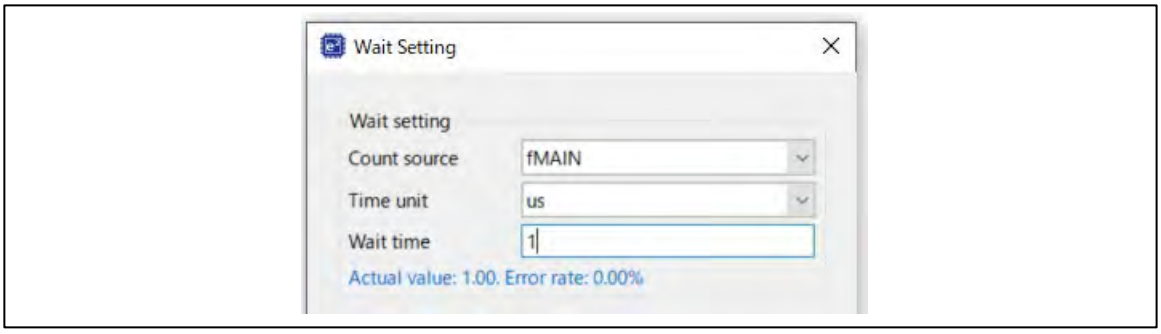
Figure 6-3 P7 Output (8-bit) Settings



6.2.3 Wait Settings for the Common Signals

The SMS waits for 1 µs as a time required before the digit selection signal becomes stable. If an external driver IC is added, increase the wait time in consideration of the delay time for the driver IC.

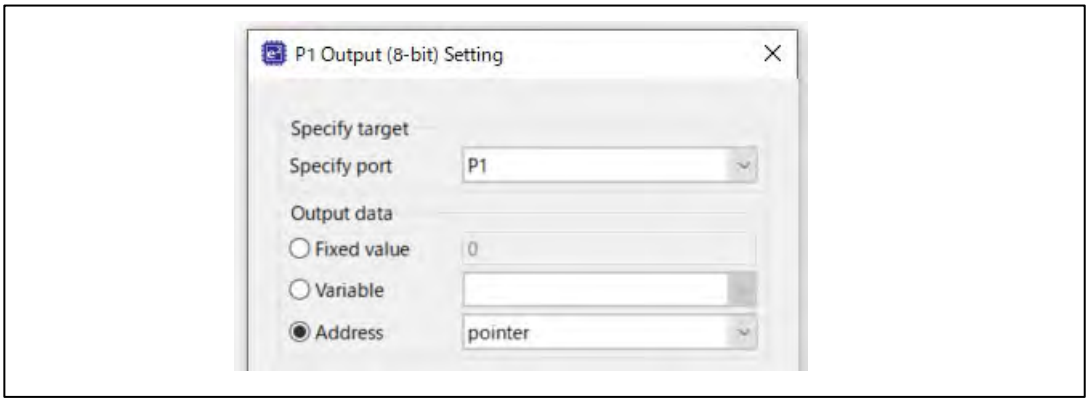
Figure 6-4 Wait Settings for the Common Signals



6.2.4 P1 Output (8-bit) Settings

The segment data corresponding to the data to be displayed is output to the P1 port.

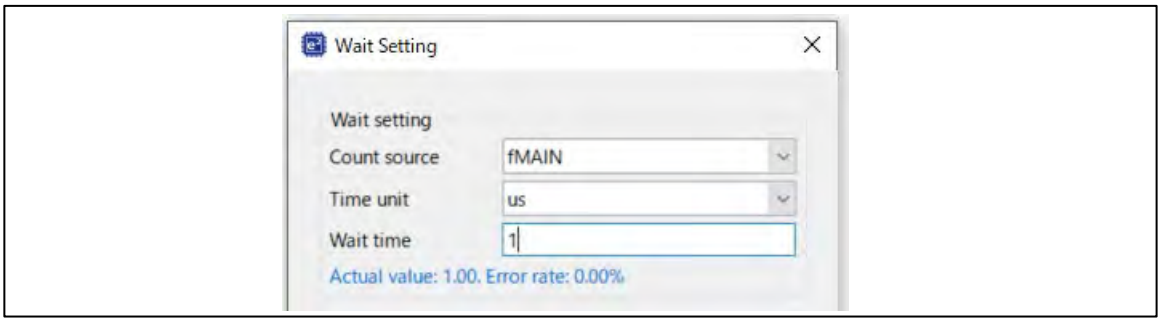
Figure 6-5 P1 Output (8-bit) Settings



6.2.5 Wait Settings for the Segment Signals

The SMS waits for 1 µs as a time required before the segment signals become stable. If an external driver IC is added, increase the wait time in consideration of the delay time for the driver IC.

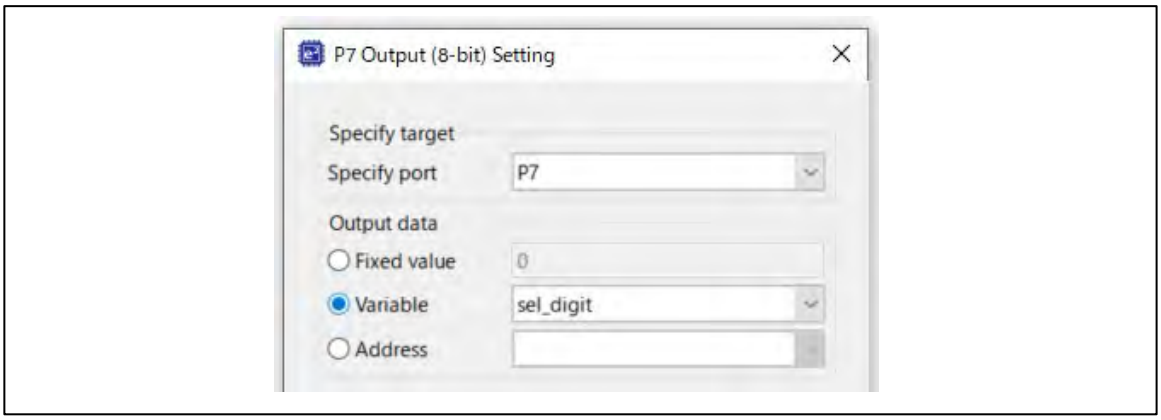
Figure 6-6 Wait Settings for the Segment Signals



6.2.6 P7 Output (8-bit) Settings

The SMS turns on the common signal that corresponds to the digit to be displayed.

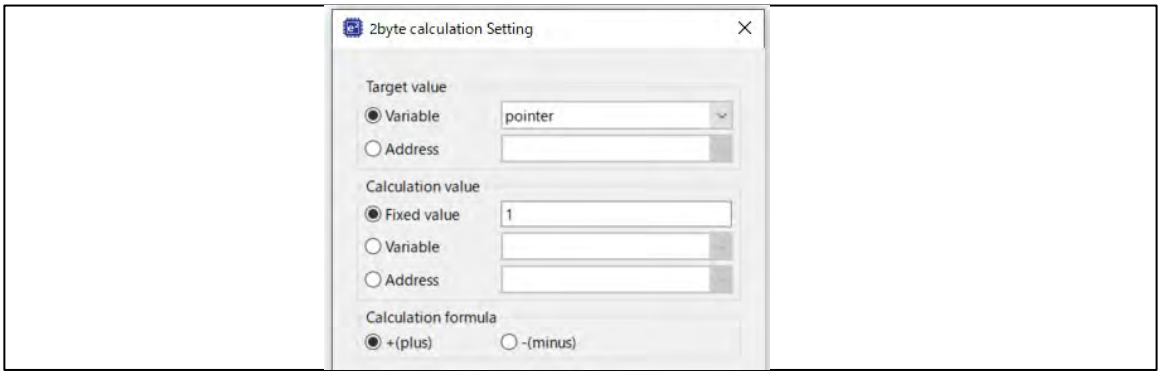
Figure 6-7 P7 Output (8-bit) Settings



6.2.7 2-Byte Calculation Settings

The pointer to the data to be displayed is moved to the next data.

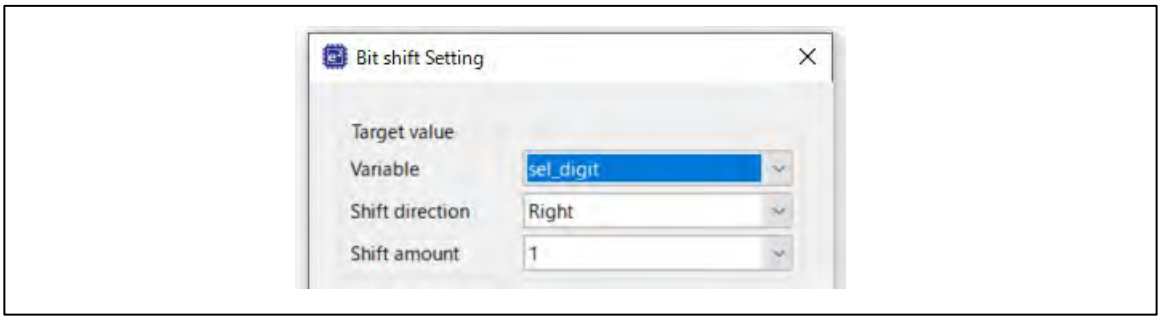
Figure 6-8 2-Byte Calculation Settings



6.2.8 Bit Shifting Settings

The digit selection signal (sel_digit) is shifted one bit to the right so that the next digit is displayed.

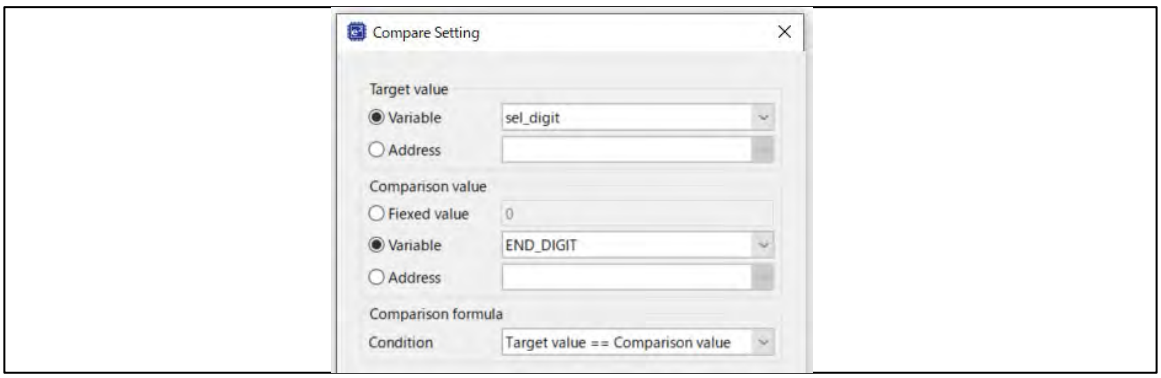
Figure 6-9 Bit Shifting Settings



6.2.9 Comparison Settings

The SMS compares the digit selection signal (sel_digit) with the termination condition (END_DIGIT) to judge whether the selected digit has reached to the last digit.

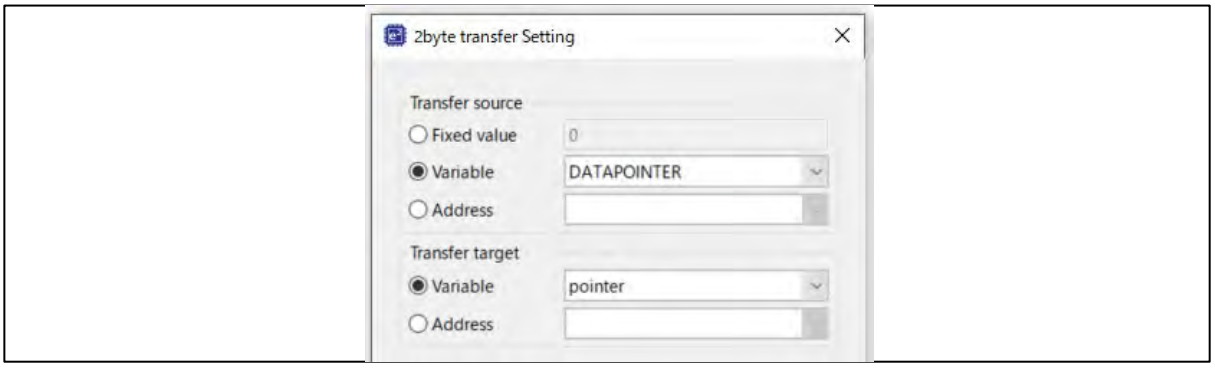
Figure 6-10 Comparison Settings



6.2.10 2-Byte Transfer for Pointer Initialization

The value of the "pointer" variable (the value of the pointer to the point at which to start reading segment data) is reset to the initial value (DATAPOINTER).

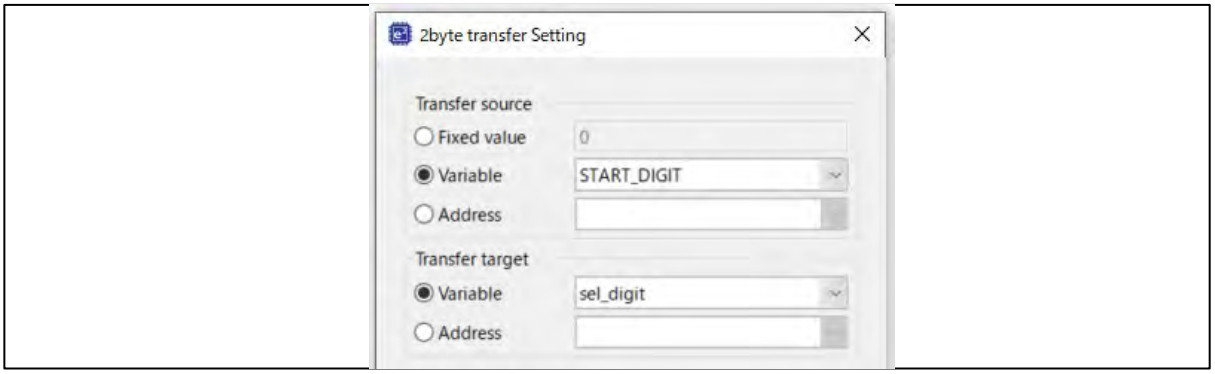
Figure 6-11 2-Byte Transfer for Pointer Initialization



6.2.11 2-Byte Transfer for Initialization of the Digit Selection Signal

The value of the "sel_digit" variable (the value for selecting the next digit) is reset to the initial value (STAR_DIGIT).

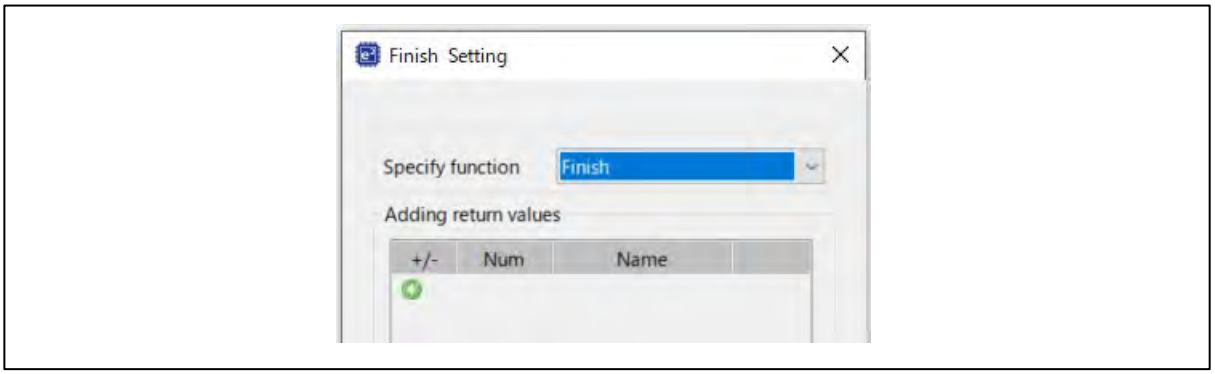
Figure 6-12 2-Byte Transfer for Initialization of the Digit Selection Signal



6.2.12 Finish Settings

The CPU enters HALT mode. In the sample code, no return code is used.

Figure 6-13 Finish Settings



6.2.13 Variable Settings

The following table shows the variables and their settings used for the SMS.

Table 6-3 Variables Used for the SMS

Data Name	Initialized When
pointer	The argument is passed to the SMS start function.
sel_digit	The SMS start function sets 0xFFF7.

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

Furthermore, if you regenerate the code with Smart Configurator, please modify the R_Config_SMS_Start function as indicated in the red frame below.

```
void R_Config_SMS_Start(uint16_t DATAPOINTER, uint16_t pointer)
{
    /* Set the sms data from arguments */
    SMSG4 = DATAPOINTER;
    SMSG1 = pointer;
    /* Initialize SMS data */
    SMSG2 = 65527U;
    SMSG5 = 65527U;
    SMSG3 = 4095U;
    /* Disable related interrupts */
    ITLMK = 1U;
    /* Start sms */
    //SMSEIF = 0U; /* clear INTSMSE interrupt flag */
    //SMSEMK = 0U; /* enable INTSMSE interrupt */
    SMSEMK = 1U; /* disable INTSMSE interrupt */
    SMSEIF = 1U; /* set INTSMSE interrupt flag */
    g_sms_wakeup_flag = 0U;
    ITLS0 = _00_INTITL_CLEAR;
    SMSSTART = 1U;
}
```

8. Reference Documents

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

RL78 family user's manual software (R01US0015)

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	July.22.11	—	First Edition
1.10	Jan.9.24	—	Changed the flowchart for SMS processing

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/.