RL78/I1A

PMBus (Slave Transmission/Reception)

Purpose

This document describes a sample code that achieves PMBus functionality (slave transmission/reception) by using the IICA serial interface provided on the RL78/I1A. The sample code uses IICA to perform the slave operations (address reception and data transmission/reception) of PMBus in a single master system.

Target Devices

RL78/I1A

When applying this application note to other microcontrollers, make the necessary changes according to the specifications of the microcontroller and verify them thoroughly.

Contents

1. Specifications

This document describes how to perform slave transmission/reception (address reception, data transmission/reception) using the PMBus (Power Management Bus) .

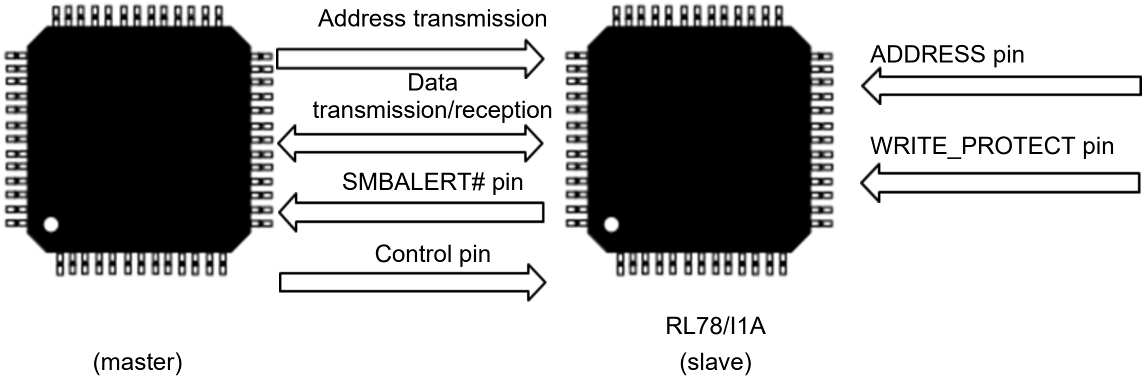
PMBus is a Open Standard Serial interface for power supply. Can make communicating easy between each power supply unit by defining necessary command language. Can connect and control many power supply units on the bus line.

This sample code was only implemented communication part for PMBus, detail process for power control part should be added to the sample code.

Table 1.1 shows the peripheral functions used and their applications, and Figure 1.1 shows an overview of PMBus communication.

Table 1.1 Peripheral Functions Used and Their Applications

Peripheral Function	Application
Serial interface IICA	Used for slave transmission/reception via PMBus. Provides PMBus functionality in a single master system by using the SCLA0 and SDAA0 pins.
Channel 0 of timer array unit	Interval timer mode Used for generating the PMBus timeout.
Watchdog timer (WDT)	Used for restoring the program by reset in case of a program loop.
P200 to P206	Used as the ADDRESS pins for setting the PMBus address.
P02	Used as SMBALERT# pin of PMBus.
P20	Used as the CONTROL pin of PMBus.
P75	Used as the WRITE_PROTECT pin of PMBus.



■ Address reception
Each device connected to PMBus has a unique address. Each device receives the address of the transfer target (slave) from the master. When the slave receives the local address, it generates an acknowledge signal.

■ Data transmission/reception
Transmitting data to the master or receiving data from the master after receiving an address.

■ SMBALERT#
The SMBALERT# pin is a optional pin. It is used as an interrupt line from the slave to the master.

■ Control pin
The CONTROL pin is a optional pin. It can control ON/OFF for each slaves. And it can set disable, enable, and active level for this pin by command.

■ Address pin
The set value of the local address is entered in 7 bits.

■ WRITE_PROTECT pin
The WRITE_PROTECT pin is a optional pin.

Figure 1.1 Overview of PMBus Communication(1/2)

■ Communication format for PMBus
PMBus is defined the communication format depending on communication direction and communication size.
Main type of them are shown as follows.

S: start condition
Sr: repeat start condition
Rd: read(1)
Wr: write(0)
A: acknowledge(ACK: 0, NACK: 1)
P: stop condition
PEC: Packet Error Code
□ : Master-->Slave
▨ : Slave-->Master

- Send byte

1bit	7bit	1bit	1bit	8bit	1bit	8bit	1bit	1bit
S	Slave Address	Wr	▨ A	Data Byte	▨ A	PEC	▨ A	P

- Receive byte

1bit	7bit	1bit	1bit	8bit	1bit	8bit	1bit	1bit
S	Slave Address	Rd	▨ A	▨ Data Byte	A	▨ PEC	A(1)	P

- Write byte

1bit	7bit	1bit	1bit	8bit	1bit	8bit	1bit	8bit	1bit	1bit
S	Slave Address	Wr	▨ A	Command Code	▨ A	Data Byte	▨ A	PEC	▨ A	P

- Read byte

1bit	7bit	1bit	1bit	8bit	1bit	1bit	7bi	1bit	1bit	8bit	1bit	
S	Slave Address	Wr	▨ A	Command Code	▨ A	S	Slave Address	Rd	▨ A	▨ Data Byte	A	···

8bit	1bit	1bit
▨ PEC	A(1)	P

- Write word

1bit	7bit	1bit	1bit	8bit	1bit	8bit	1bit	8bit	1bit	
S	Slave Address	Wr	▨ A	Command Code	▨ A	Data Byte Low	▨ A	Data Byte High	▨ A	···

8bit	1bit	1bit
PEC	▨ A	P

- Read word

1bit	7bit	1bit	1bit	8bit	1bit	1bit	7bit	1bit	1bit	8bit	1bit	
S	Slave Address	Wr	▨ A	Command Code	▨ A	S	Slave Address	Rd	▨ A	▨ Data Byte Low	A	···

8bit	1bit	8bit	1bit	1bit
▨ Data Byte High	A	▨ PEC	A(1)	P

- Block write

1bit	7bit	1bit	1bit	8bit	1bit	8bit	1bit	8bit	1bit	
S	Slave Address	Wr	▨ A	Command Code	▨ A	Byte Count = N	▨ A	Data Byte1	▨ A	···

8bit	1bit		8bit	1bit	8bit	1bit	1bit
Data Byte2	▨ A	···	Data ByteN	▨ A	PEC	▨ A	P

- Block read

1bit	7bit	1bit	1bit	8bit	1bit	1bit	7bit	□bit	1bit	8bit	1bit	
S	Slave Address	Wr	▨ A	Command Code	▨ A	S	Slave Address	Rd	▨ A	▨ Byte Count = N	A	···

8bit	1bit		8bit	1bit	8bit	1bit	1bit
▨ Data Byte2	A	···	▨ Data ByteN	A	▨ PEC	A(1)	P

Figure 1.1 Overview of PMBus Communication(2/2)

2. Conditions Under Which Operation Has Been Verified

The operation of the sample code shown in this application note has been verified under the conditions shown below.

Table 2.1 Conditions Under Which Operation Has Been Verified

Item	Description
Microcontroller used	RL78/I1A (R5F107DE)
Operating frequency	• High-speed on-chip oscillator (HOCO) clock: 32 MHz • CPU/peripheral hardware clock: 32 MHz
Operating voltage	5.0 V (The Micro can operate on 4.0 V to 5.5 V for PMBus bus input level compliance)
Integrated development Environment (CS+)	CS+ V1.02.01 made by Renesas Electronics
C compiler (CS+)	CA78K0R V1.41 made by Renesas Electronics
Integrated development environment (CS+)	CS+ V8.04.00 made by Renesas Electronics
C compiler (CS+)	CC-RL V1.09.00 made by Renesas Electronics
Integrated development environment (e^{2}studio)	e^{2}studio V2020-10 made by Renesas Electronics
C compiler (e^{2}studio)	CC-RL V1.09.00 made by Renesas Electronics
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.20.1
C compiler (IAR)	IAR C/C++ Compiler for Renesas RL78 V4.20.1.2260

3. Related Application Notes

Related application notes are shown below. Also refer to these documents when using this application note.

RL78/I1A User's Manual: Hardware (R01UH0169)

RL78 Family User's Manual: Software (R01US0015)

4. Hardware

4.1 Pins used

Table 4.1 shows the pins used and their features.

Table 4.1 Pins Used And Their Features

Pin Name	I/O	Description
P10/SCLA0	I/O	Serial clock I/O pin for IICA0 (PMBus)
P11/SDAA0	I/O	Serial data transmission/reception pin for IICA0 (PMBus)
P200 to P206	Input	ADDRESS pin
P02	Output	SMBALERT# pin[Note 1]
P20	Input	CONTROL pin
P75	Input	WRITE_PROTECT pin[Note 2]

Notes 1. Be sure to set this pin to the N-ch open drain output (VDD withstand voltage) mode and connect an external pull-up resistor.

2. Be sure to connect an external pull-up resistor to this pin.

Remark The PMBus specifications define SMBALERT#, CONTROL, WRITE_PROTECT as optional signal lines.

5. Software

5.1 Operation overview

The sample code performs the slave transmission/reception (address reception, data transmission/reception) operations of PMBus using the IICA serial interface. The sample code first performs initial setup, and then waits for data to be transmitted from the master. Received data is stored in receive buffer after receiving own address from master. And then analyze received command and call function(user function) for each command. Detail process for power control should be added in user function.

(1) Perform initial setup of the peripheral I/O port.

Setting conditions:

- Set the SMBALERT# port (1 bit) to the output mode, and N-ch open drain output (VDD withstand voltage) mode. The initial value is set to High.
- Set the ADDRESS port (7 bits) to the input mode.
- Set the CONTROL port (1 bit) to the input mode.
- Set the WRITE PROTECT port (1 bit) to the input mode.
- Set the P10/SCLA0 and P11/ADAA0 to TTL input buffer mode and N-ch O.D. output mode.

(2) Perform initial setup of the timer array unit.

Setting conditions:

- Specify f_{CLK} as the operating clock.
- Specify the interval timer mode as the operating mode of TAU.
- Specify 0x7CFF as the compare value (for generating 1 ms reference time)

(3) Perform initial setup of the IICA serial interface.

Setting conditions:

- Set the operation mode to the standard mode.
- Set the transfer clock to 100 kHz.
- Set the local address.
- Set an interrupt to occur at every 9th clock.
- Disable the stop condition interrupt.
- Set the P10/SCLA0 pin to serial clock (I/O), and the P11/SDAA0 pin to data transmission/reception (I/O).

((1) to (3) after the implementation, and then waits for the address sending from the master.)

(4) When a local address or an extension code is received from the master, an IICA0 interrupt occurs, followed by command reception of the PMBus protocol.

(5)Analyze received command, after the stop condition is received.

[Operation after receive the receive commands]
Analyze the received data(command and PEC), store it to storage RAM if receive normally, then call user function.

[Operation after receive the send commands]
Analyze the received data(command) and call user function if receive normally, then make the send data and send it.

(6) During data communication, if 25 ms has elapsed during communication of one packet, it is regarded as a communication error (timeout), and the IICA serial interface is shut down and initialized.

Cautions 1. The operation mode can be set to the standard mode or fast mode.
The transfer clock can be set to 100 kHz or 400 kHz.
*The transfer clock can be changed by changing the comment part in the 76th line in `r_iica0_driver.c`.
2. The peripheral I / O ports can be used for other functions except used ports for PMBus.

5.2 Option byte settings

Table 5.1 shows the option byte settings.

Table 5.1 Option Byte Settings

Address	Setting	Description
000C0H/010C0H	01111110B	Enables the counter operation by the watchdog timer (Counting stops after the reset period ends.)
000C1H/010C1H	11111111B	Disables LVD (Use an external reset)
000C2H/010C2H	11101000B	HS (high-speed main) mode High-speed on-chip oscillator clock frequency: 32 MHz
000C3H/010C3H	10000100B	Enables on-chip debugging.

5.3 Functions

Table 5.2 shows the functions.

Table 5.2 Functions

Function	Description
R_TAU0_Create	Timer array unit initialization processing
R_TAU0_Channel0_Start	Timer array unit operation start processing
R_TAU0_Channel0_Stop	Timer array unit operation stop processing
R_TIMER_Control	1 ms reference timer generation processing
R_PORT_Create	Port initialization processing
R_PORT_Smbalert	SMBALERT# port output setting processing
R_PORT_Get_Address	ADDRESS port input value acquisition processing
R_PORT_Get_Control	CONTROL port input value acquisition processing
R_PORT_Get_Write_Protect	WRITE_PROTECT port input value acquisition processing
R_IICA0_Create	IICA0 initialization processing
R_IICA0_Stop	IICA0 stop processing
R_IICA0_Init	IICA0 driver module initialization
R_IICA0_Get_Receivedata	IICA0 reception data acquisition
R_IICA0_Set_Senddata	Transmission data setting processing
R_PMBUS_Init	PMBus control module initialization
R_PMBUS_Set_Analysis_Sts	Communication status acquisition
R_PMBUS_Timeout_Enable	Timeout timer operation control processing
R_PMBUS_Control	PMBus control module control processing
R_PMBUS_Timeout_Make	Timeout time generation processing
R_PMBUS_Fault_Handler	Error information setting processing
R_PMBUS_User_App	Command target function call processing

5.4 Function specifications

This section shows the specifications of the principal functions used in the sample code.

R_IICA0_Get_Receivedata

Overview	Reception data acquisition processing	
Header	r_iica0_driver.h	
Declaration	uint8_t *R_IICA0_Get_Receivedata(void)	
Description	This function acquires the initial address of the buffer storing the reception data.	
Parameters	None	
Return value	buffer_address	Initial address of the reception buffer
Remark	None	

R_IICA0_Set_Senddata

Overview	Transmission data setting processing	
Header	r_iica0_driver.h	
Declaration	void R_IICA0_Set_Senddata(uint8_t*)	
Description	This function copies the transmission data to the buffer and transmits the first byte of data.	
Parameters	send_buffer	Start address of the buffer storing transmission data
Return value	None	
Remark	None	

R_PMBUS_Set_Analysis_Sts

Overview	Communication status setting processing	
Header	r_pmbus_control.h	
Declaration	void R_PMBUS_Set_Analysis_Sts(uint8_t)	
Description	This function sets the communication status.	
Parameters	analysis_sts	Communication status
Return value	None	
Remark	None	

R_PMBUS_Timeout_Enable

Overview	Timeout timer operation control processing	
Header	r_pmbus_control.h	
Declaration	void R_PMBUS_Timeout_Enable(uint8_t)	
Description	This function enables the operation of the timeout timer.	
Parameters	timer_enable	Timer operation enable/disable flag
Return value	None	
Remark	None	

`R_PMBUS_Timeout_Make`

Overview	Timeout time generation processing
Header	`r_pmbus_control.h`
Declaration	`void R_PMBUS_Timeout_Make(void)`
Description	This function counts the timeout time(25ms) and judges the timeout.
Parameters	None
Return value	None
Remark	None

`R_PMBUS_Fault_Handler`

Overview	Error information setting processing	
Header	`r_pmbus_control.h`	
Declaration	`void R_PMBUS_Fault_Handler( uint8_t )`	
Description	This function sets error information to the register of internal status.	
Parameters	`error_result`	Error information
Return value	None	
Remark	None	

`R_PMBUS_Control`

Overview	PMBus control module control processing
Header	`r_pmbus_control.h`
Declaration	`void R_PMBUS_Control(void)`
Description	This function controls the operation of the PMBus communication.
Parameters	None
Return value	None
Remark	None

`R_PMBUS_User_App`

Overview	Command target function call processing	
Header	`r_pmbus_userapp.h`	
Declaration	`void R_PMBUS_User_App( uint8_t )`	
Description	This function calls the command target function.	
Parameters	`array_index`	Target command
Return value	None	
Remark	None	

5.5 Flowcharts

5.5.1 Overview of processing flow

Figure 5.1 shows an overview of the processing flow used in this sample code.

[Overview]
This sample code sets up the initial setting to using each peripherals and call main() function after calls hdwinit() function.

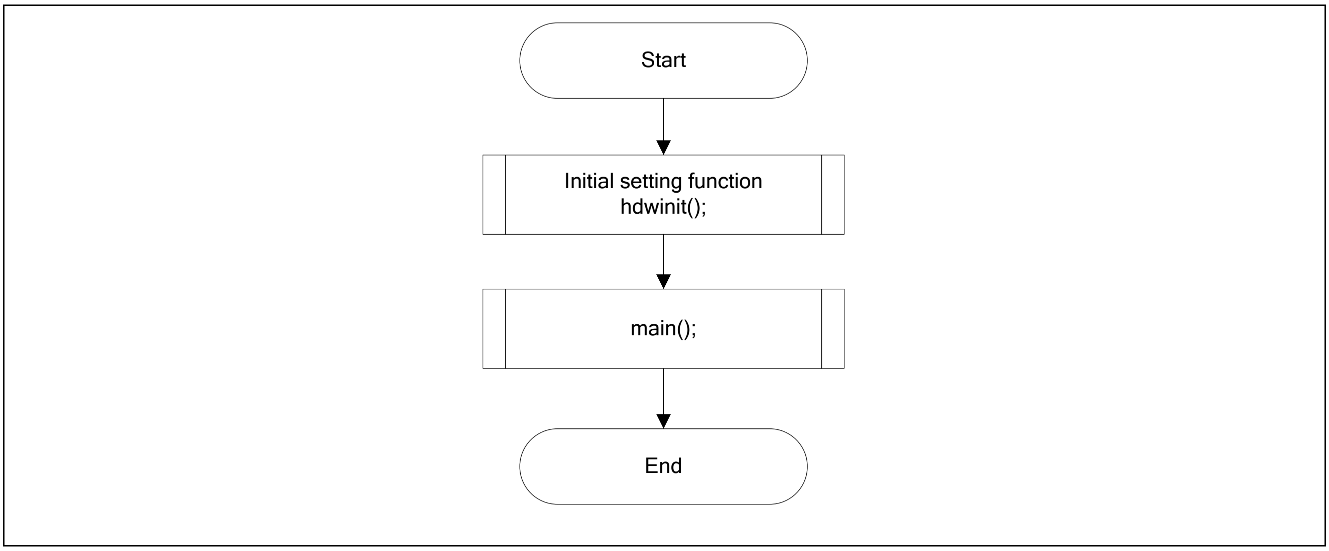


Figure 5.1 Overview of Processing Flow

5.5.2 Initialization function

Figure 5.2 shows the flow of initialization function processing.

[Overview]
hdwinit() function sets up the initial setting to using each peripherals by r_systeminit() function during disabled interrupt.

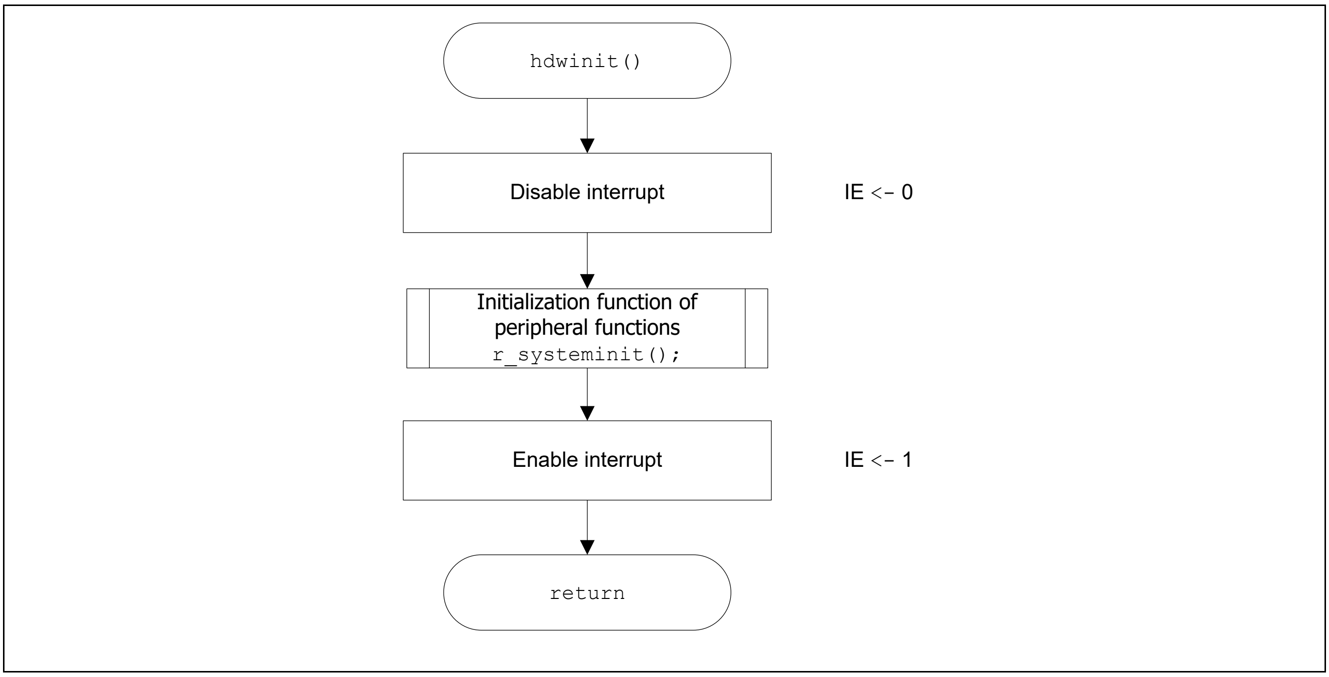


Figure 5.2 Initialization Function

5.5.3 Initialization function of peripheral functions

Figure 5.3 shows the flow of the Initialization function processing.

[Overview]
r_systeminit() function sets up the initial setting to using each peripheral functions.
R_PORT_Create() function sets up the initial setting to using pins.
r_cgc_create() function sets up the initial setting regarding the operation clock.
R_IICA0_Create() function sets up the initial setting to the IICA0 function.
R_TAU0_Create() function sets up the setting creating 1ms to the TAU0.
This is the minimum scale for the timing control of PMBus communication.

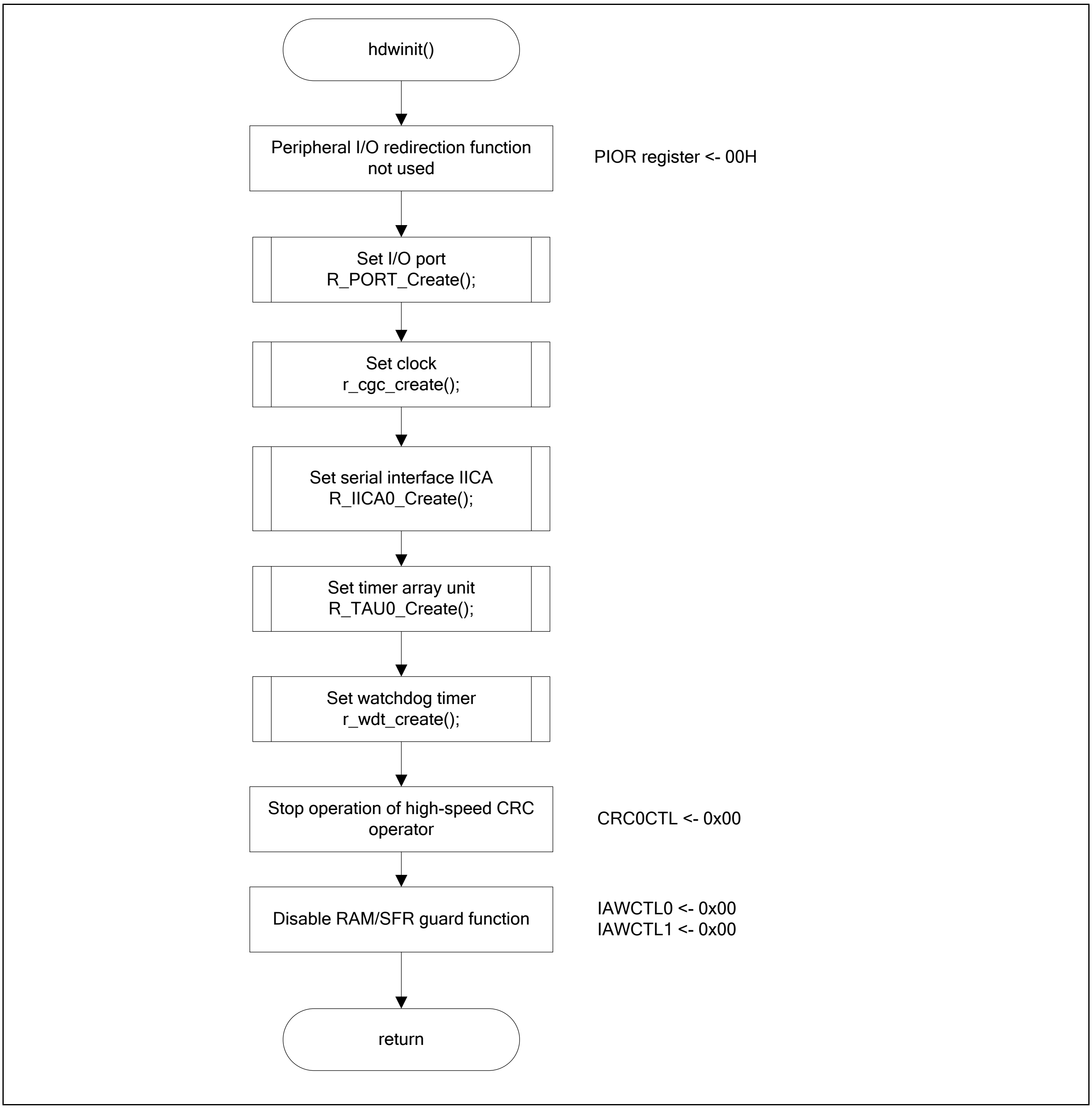


Figure 5.3 System Function

5.5.4 main routine processing

Figure 5.4 shows the flow of main routine processing.

[Overview]
This function runs the main process in this sample code.
moduleinit() function initializes variables of each module.
R_TAU0_Channel0_Start() function starts TAU0 for counting.
r_wdd_restert() function restarts the watch dog timer every period.
R_TIMER_Control() function polls 1ms timing that is minimum scale for the timing control of PMBus communication.
R_PMBUS_Control() function controls the operation of the PMBus.

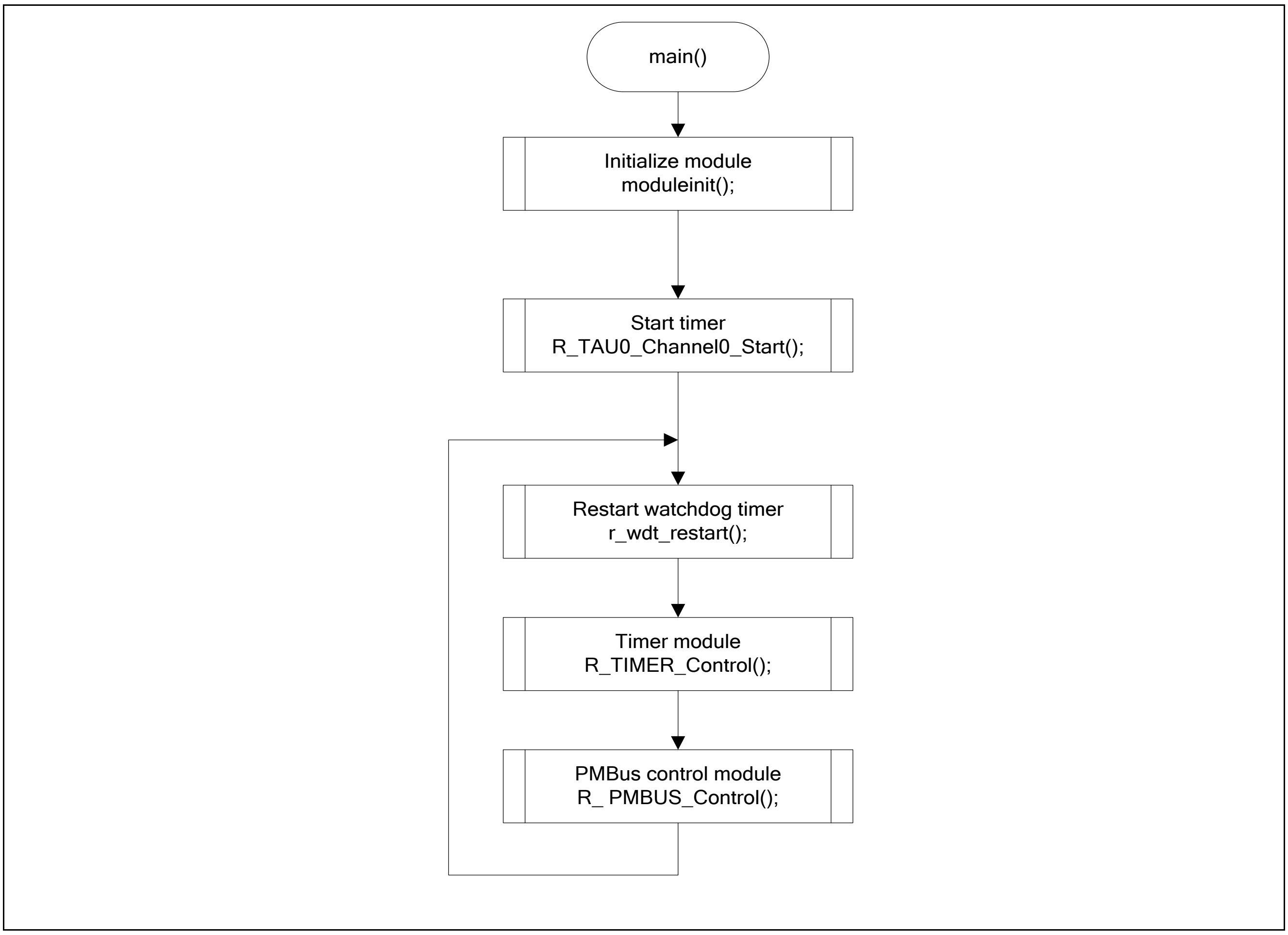


Figure 5.4 main Routine Processing

5.5.5 Timer control processing

Figure 5.5 shows the flow of timer control processing.

[Overview]
R_TIMER_Control() function polls 1ms timing that is minimum scale for the timing control of PMBus communication.

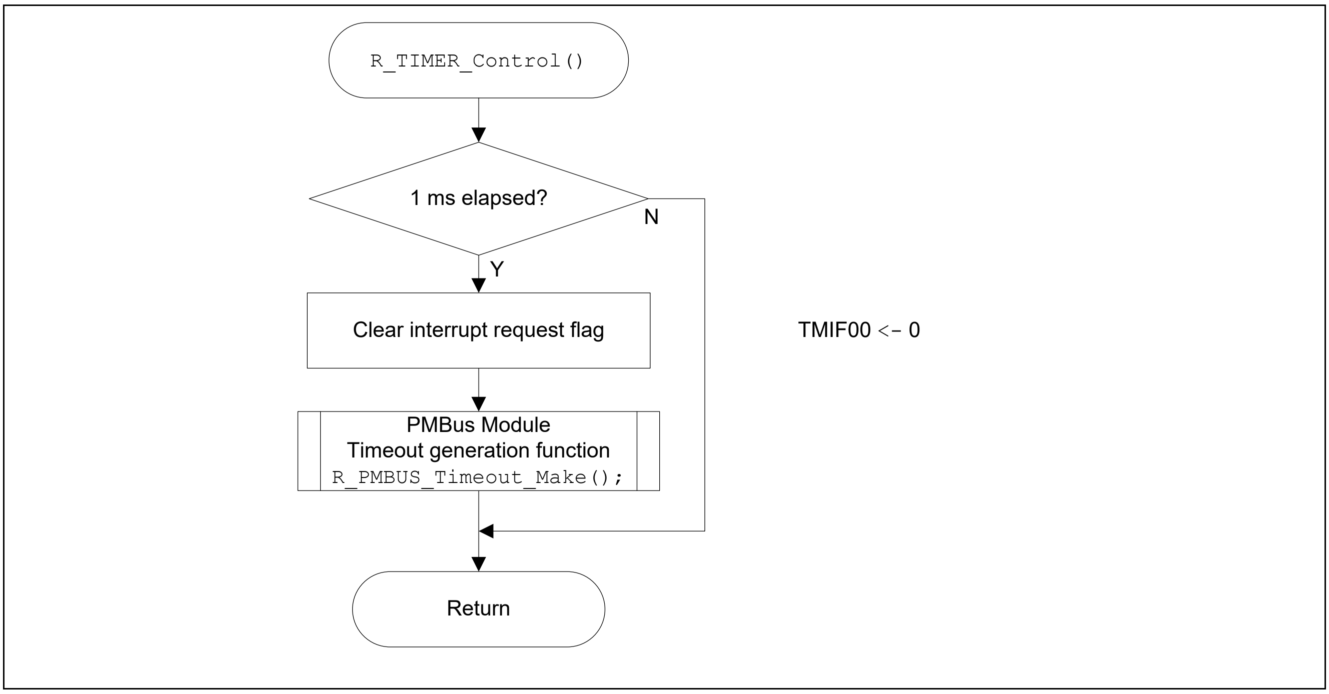


Figure 5.5 Timer Control Processing

5.5.6 IICA0 reception data acquisition

Figure 5.6 shows the flow of IICA0 reception data acquisition.

[Overview]
R_IICA0_Get_Receivedata() function returns the address of received buffers to caller.
This function is called by r_pmbus_receive_check() function.

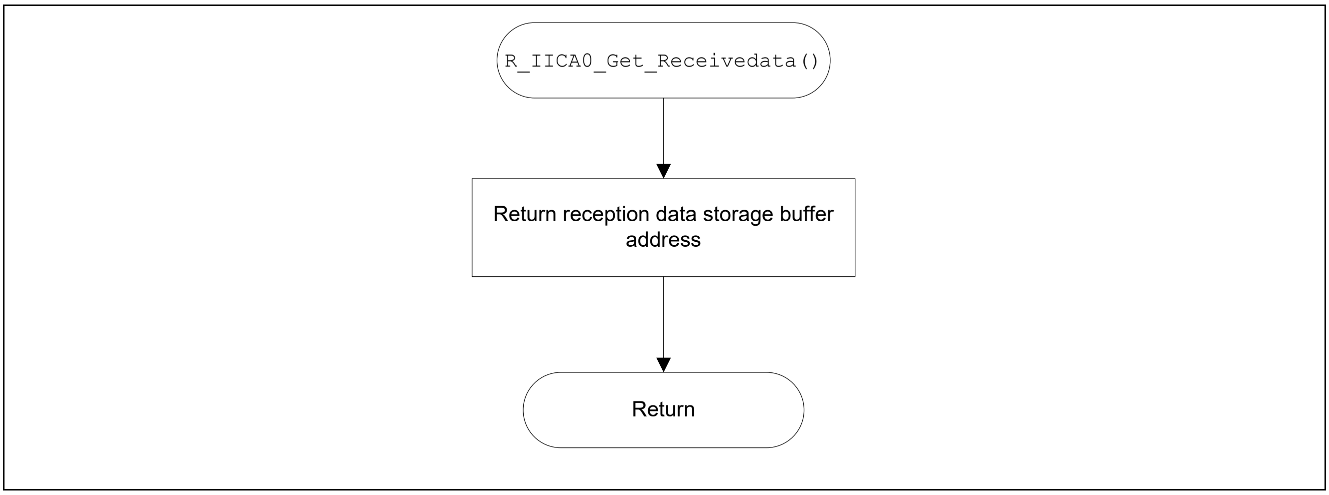

R_IICA0_Get_Receivedata()
Return reception data storage buffer address
Return

Figure 5.6 IICA0 Reception Data Acquisition

5.5.7 IICA0 transmission data acquisition

Figure 5.7 shows the flow of IICA0 transmission data acquisition.

[Overview]
R_IICA0_Set_Senddata() function copies the send data to the buffers of send data and send 1st byte.

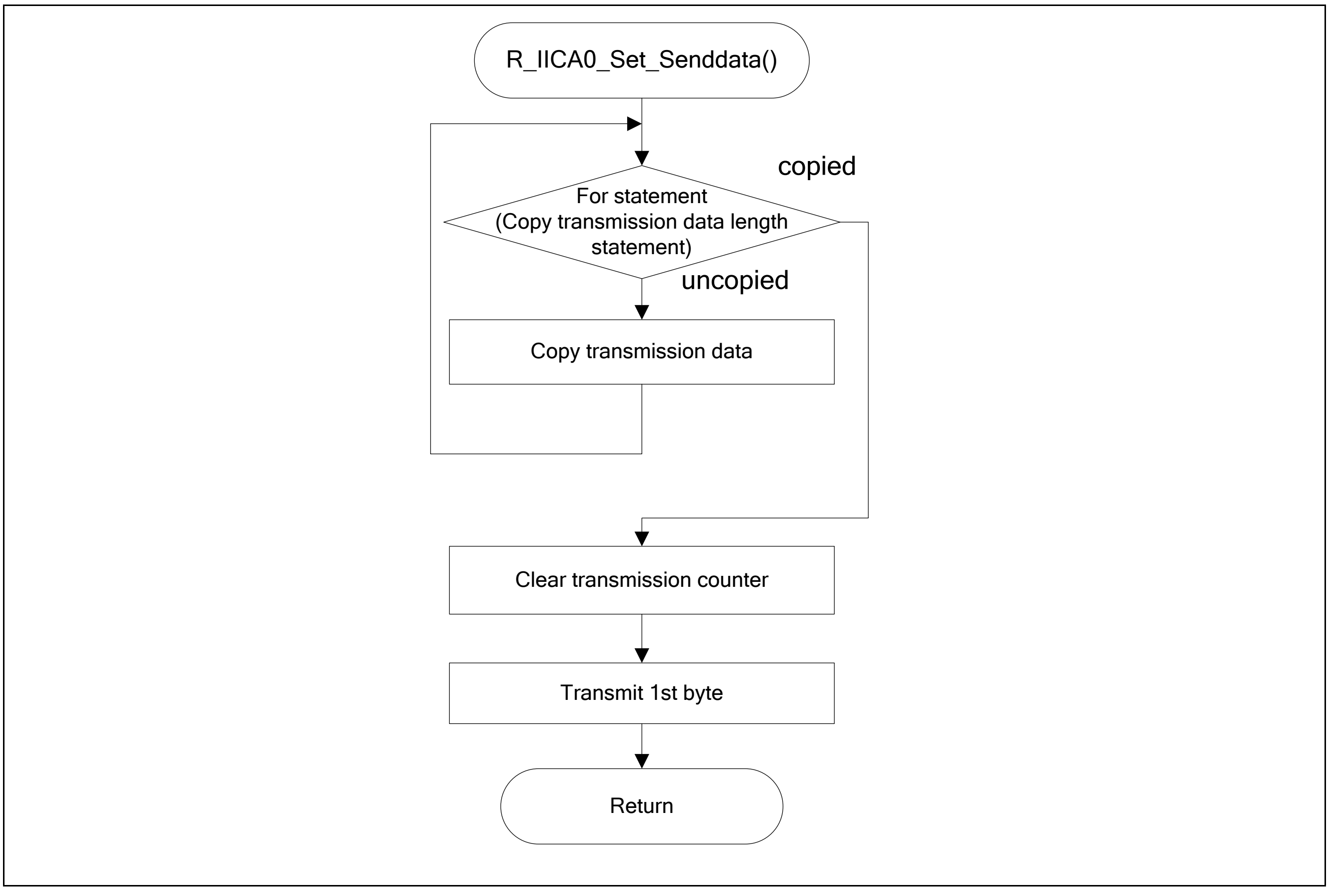


Figure 5.7 IICA0 Transmission Data Acquisition

5.5.8 IICA0 interrupt processing

Figure 5.8 shows the flow of IICA0 interrupt processing.

[Overview]

r_iica0_interrupt() function processes communication end process regarding IICA0 in interrupt process.This function operates some operations depending on the reason of interrupt:the control for timeout timer, the control for receiving data, and the control for wait.

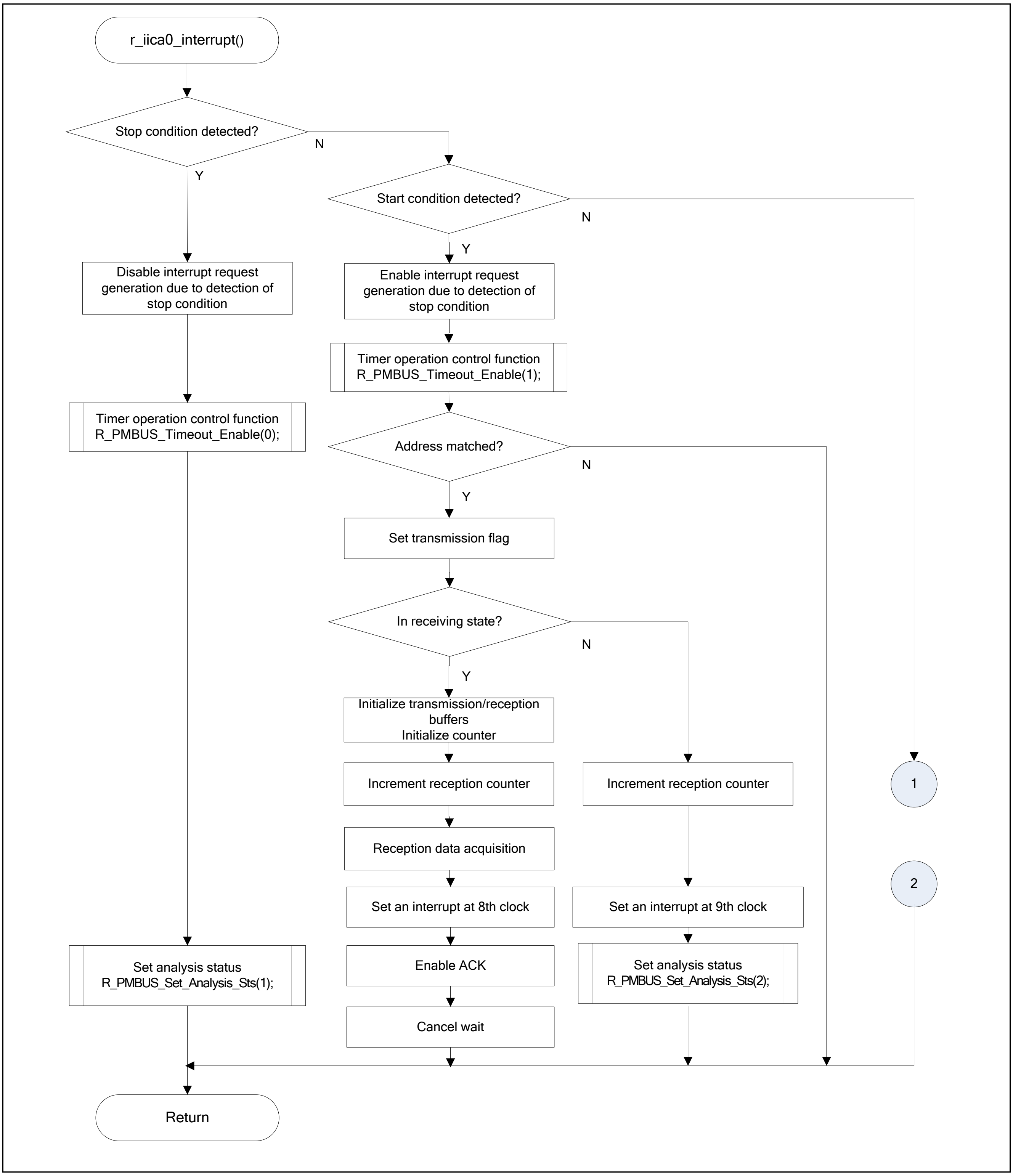

r_iica0_interrupt()
Stop condition detected?
N
Y
Start condition detected?
N
Y
Disable interrupt request generation due to detection of stop condition
Enable interrupt request generation due to detection of stop condition
Timer operation control function R_PMBUS_Timeout_Enable(1);
Timer operation control function R_PMBUS_Timeout_Enable(0);
Address matched?
N
Y
Set transmission flag
In receiving state?
N
Y
Initialize transmission/reception buffers Initialize counter
Increment reception counter
Increment reception counter
1
Reception data acquisition
2
Set an interrupt at 8th clock
Set an interrupt at 9th clock
Set analysis status R_PMBUS_Set_Analysis_Sts(1);
Enable ACK
Set analysis status R_PMBUS_Set_Analysis_Sts(2);
Cancel wait
Return

Figure 5.8 IICA0 Interrupt Processing (1/2)

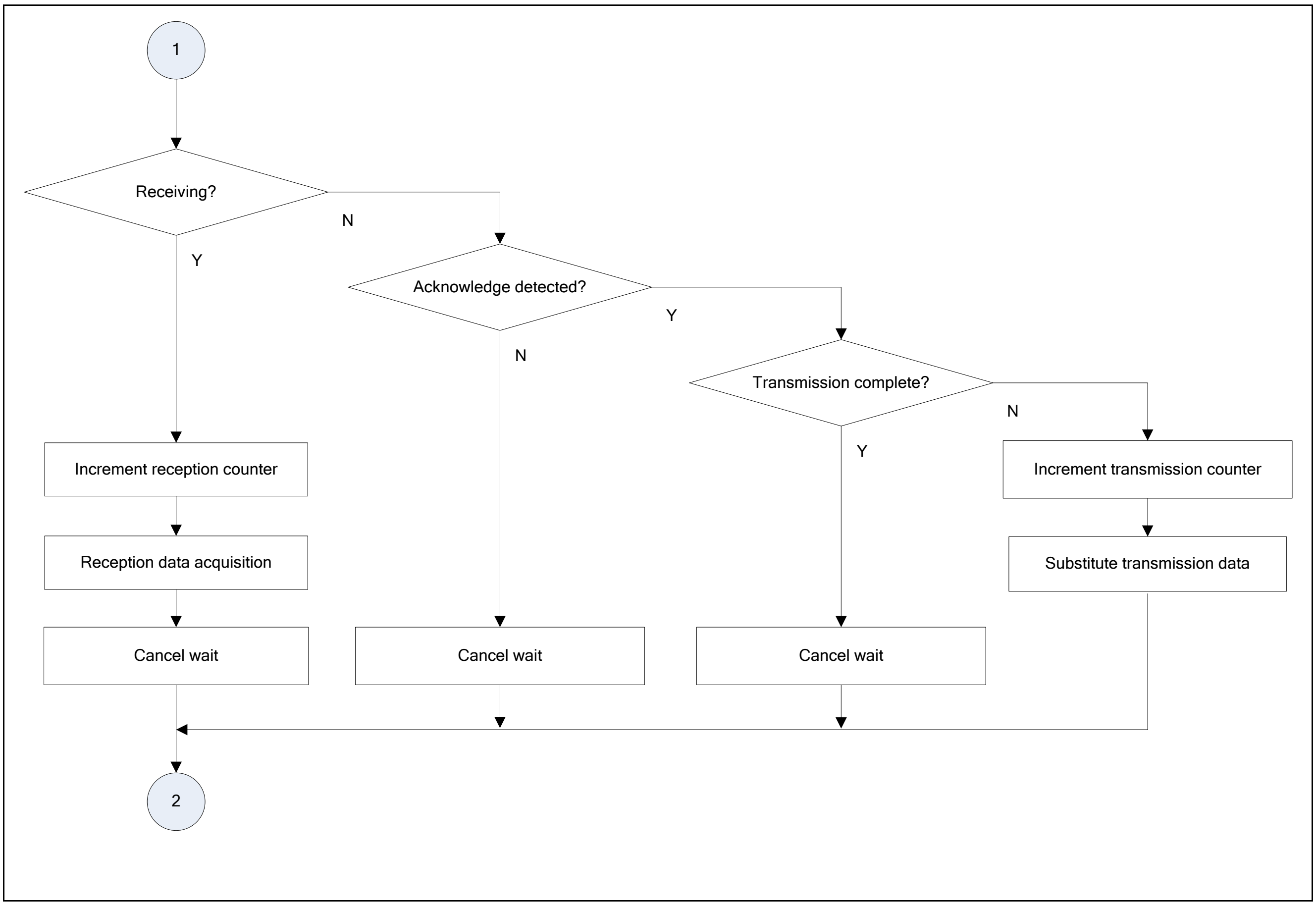

1
Receiving?
N
Y
Acknowledge detected?
Y
N
Transmission complete?
N
Y
Increment reception counter
Reception data acquisition
Cancel wait
Cancel wait
Cancel wait
Increment transmission counter
Substitute transmission data
2

Figure 5.8 IICA0 Interrupt Processing (2/2)

5.5.9 PMBus control processing

Figure 5.9 shows the flow of PMBus control processing.

[Overview]
R_PMBUS_Control() function controls the operation of the PMBus.
This function calls some functions depending on the status of communication:
the Analysis function for receive data, the user function, the function of creating send data.

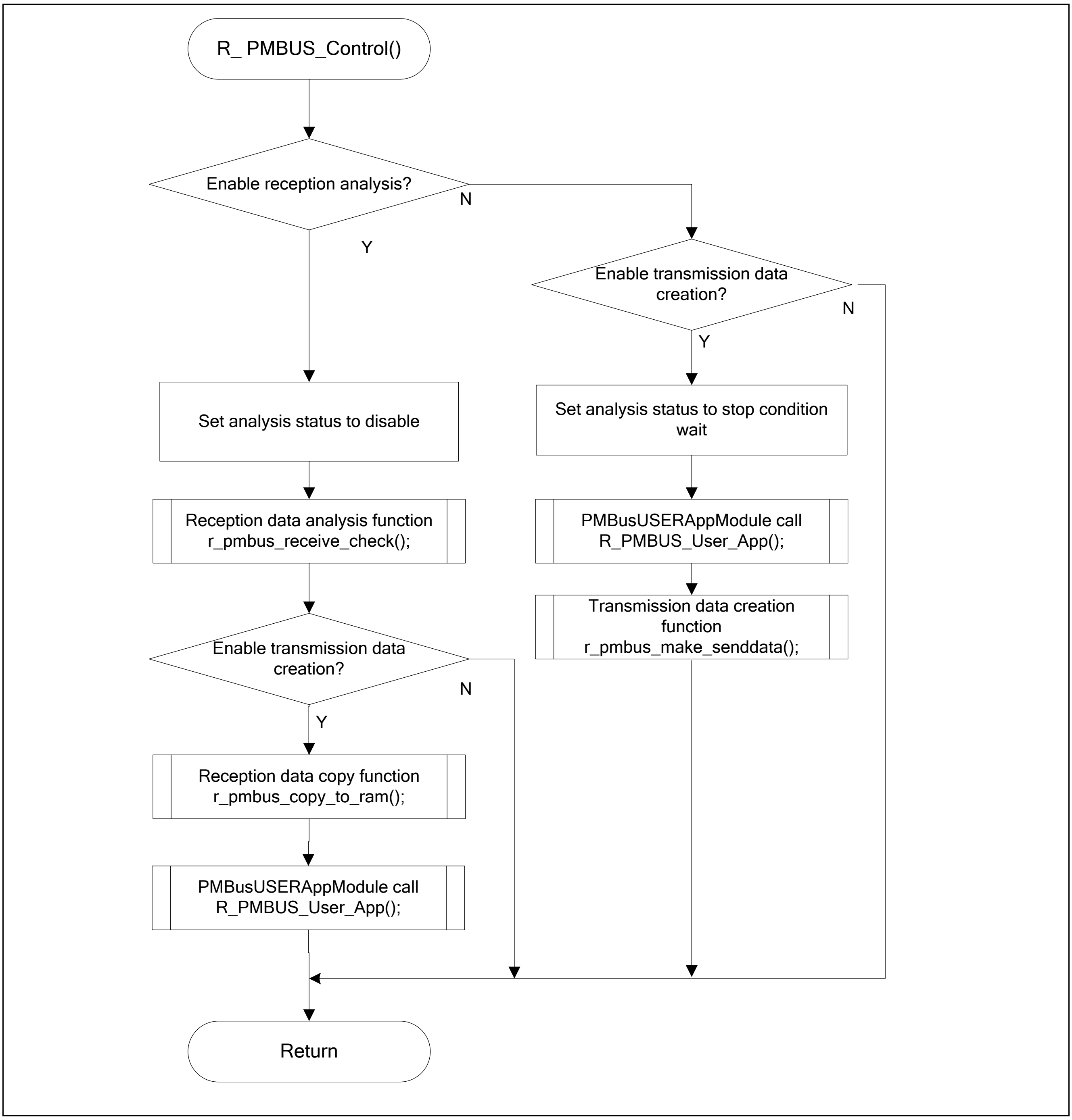

R_ PMBUS_Control()
Enable reception analysis?
N
Y
Enable transmission data creation?
N
Y
Set analysis status to disable
Set analysis status to stop condition wait
Reception data analysis function r_pmbus_receive_check();
PMBusUSERAppModule call R_PMBUS_User_App();
Transmission data creation function r_pmbus_make_senddata();
Enable transmission data creation?
N
Y
Reception data copy function r_pmbus_copy_to_ram();
PMBusUSERAppModule call R_PMBUS_User_App();
Return

Figure 5.9 PMBus Control Processing

5.5.10 Communication status acquisition

Figure 5.10 shows the flow of communication status acquisition.

[Overview]
R_PMBUS_Set_Analysis_Sts() function is called in IICA0 interrupt.
This function returns the status of communication from argument to caller.

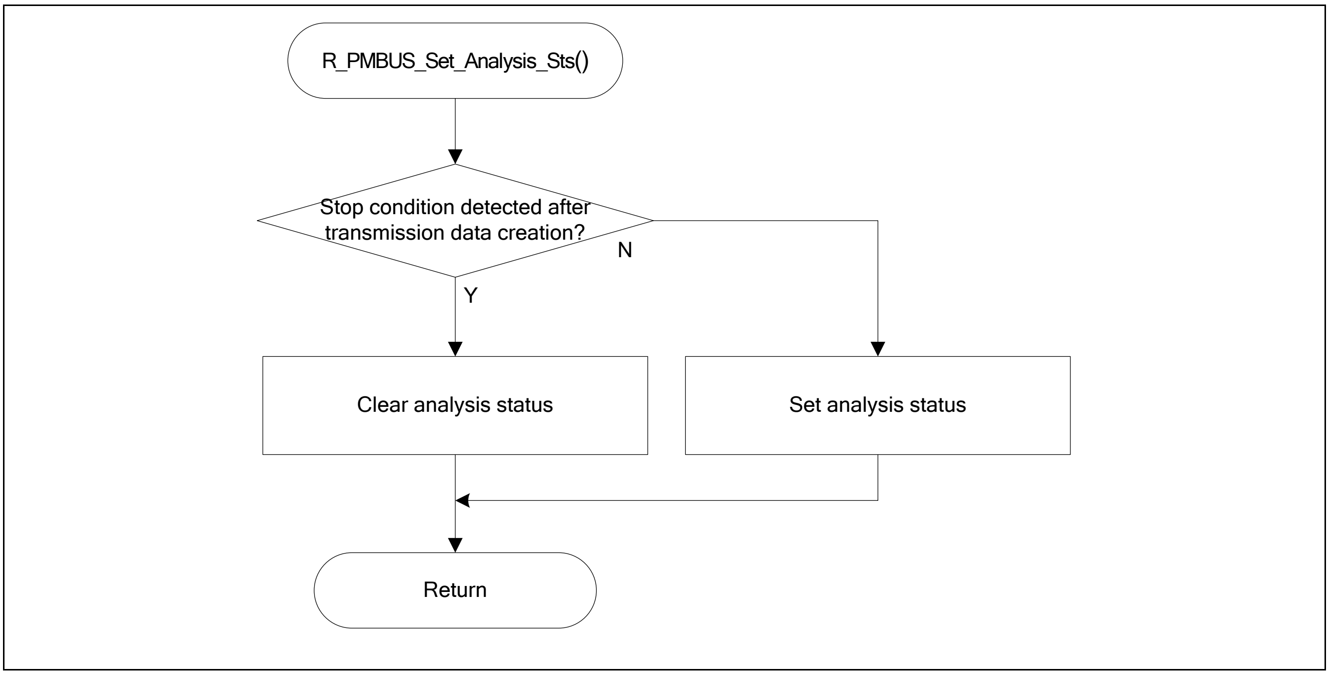

R_PMBUS_Set_Analysis_Sts()
Stop condition detected after transmission data creation?
N
Y
Clear analysis status
Set analysis status
Return

Figure 5.10 Communication Status Acquisition

5.5.11 Timeout operation control processing

Figure 5.11 shows the flow of timeout operation control processing.

[Overview]
R_PMBUS_Timeout_Enable() function controls the process of the time control for PMBus by argument.

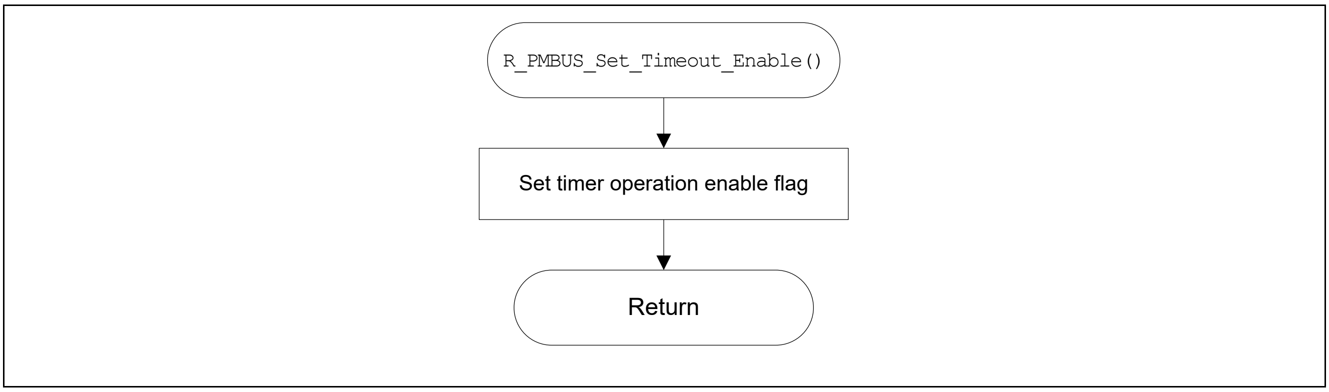


Figure 5.11 Timeout Operation Control Processing

5.5.12 Timeout time generation

Figure 5.12 shows the flow of timeout time generation.

[Overview]
R_PMBUS_Timeout_Make() function creates the timeout timing for PMBus.
If counter is over 25ms for timeout, calls R_IICA0_Stop() function and stop IICA0. And then calls R_IICA0_Create() in order to restart.

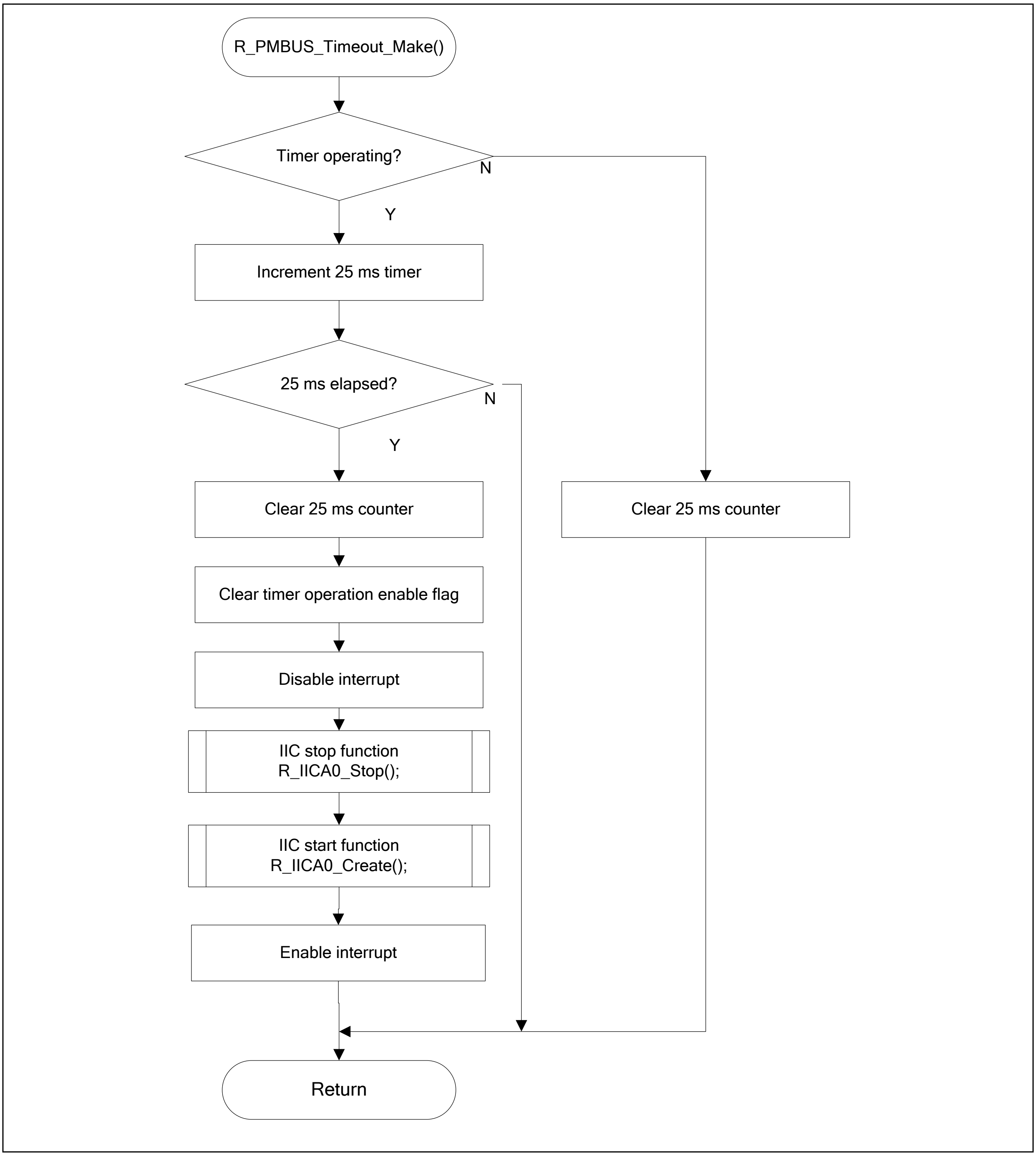


Figure 5.12 Timeout Time Generation

5.5.13 CRC8 calculation

Figure 5.13 shows the flow of CRC8 calculation.

[Overview]
r_pmbus_pec_crc8() function calculates CRC8 for communication data.

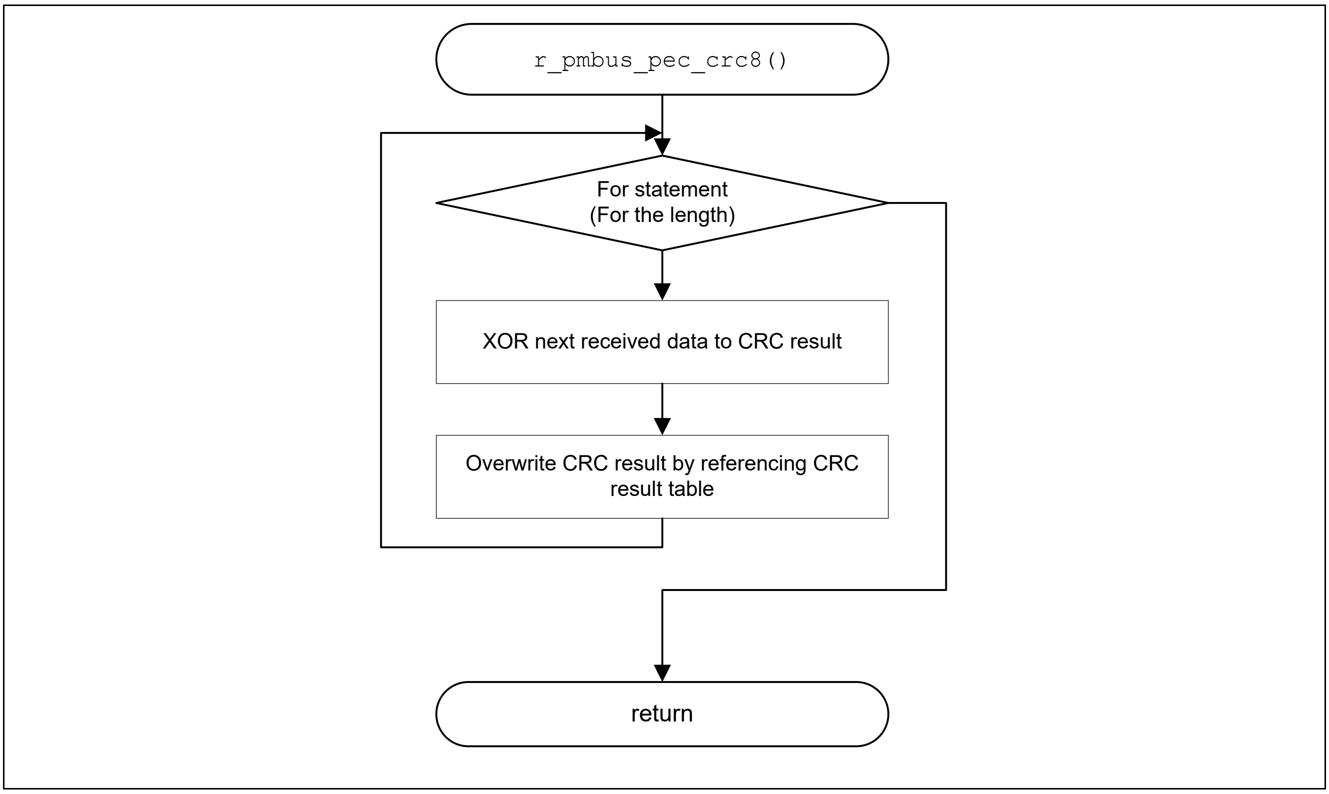

r_pmbus_pec_crc8()
For statement
(For the length)
XOR next received data to CRC result
Overwrite CRC result by referencing CRC result table
return

Figure 5.13 CRC8 Calculation

5.5.14 Reception data analysis

Figure 5.14 shows the flow of reception data analysis.

[Overview]
r_pmbus_receive_check() function analyzes received data.
This function gets the address of received buffer by R_IICA0_Get_Receivedata() function.
And calculates CRC8 by r_pmbus_pec_crc8() function.
If analyzed data is error, calls R_PMBUS_Fault_Handler() function and sets up error status.

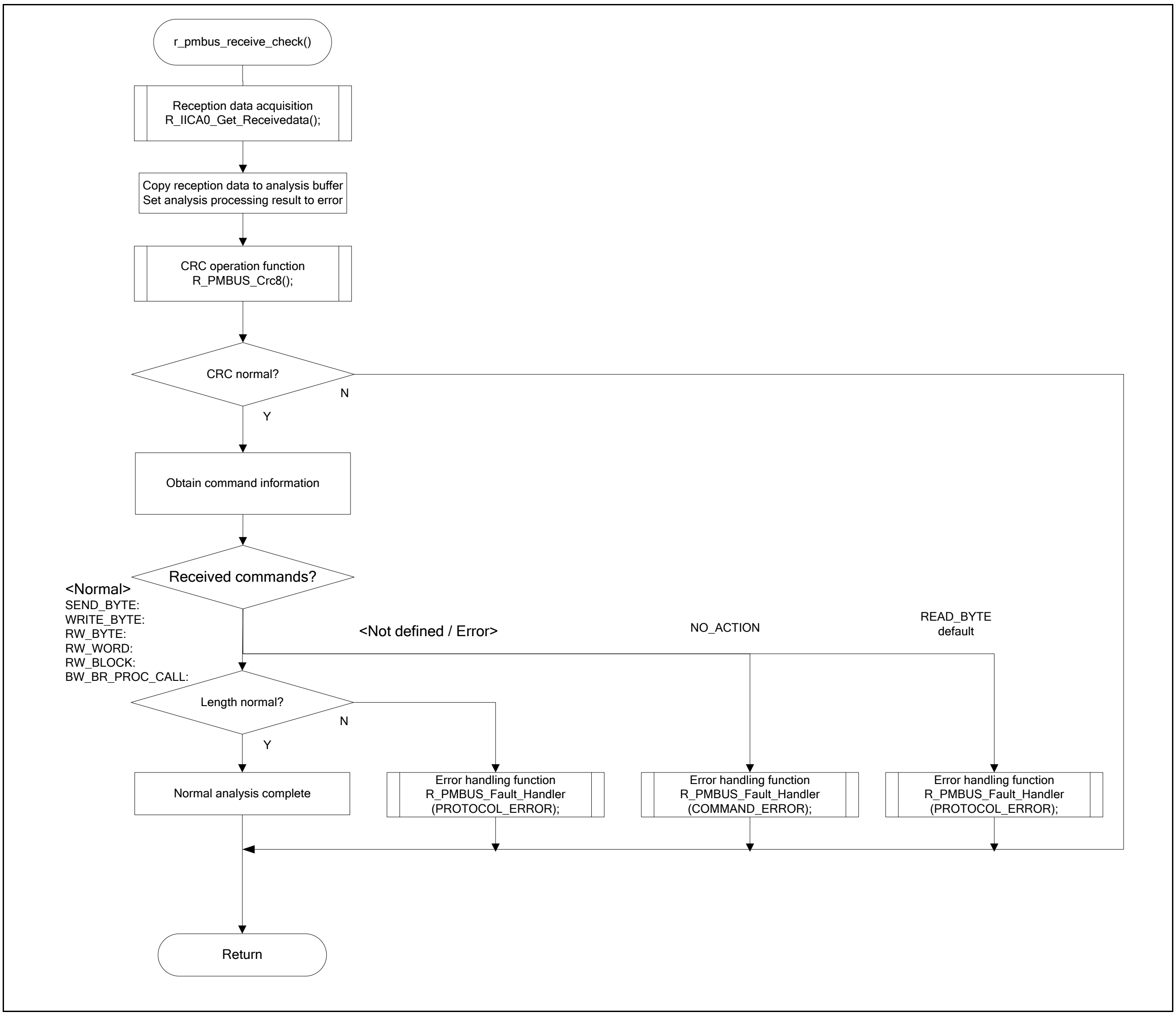


Figure 5.14 Reception Data Analysis

5.5.15 Command analysis

Figure 5.15 shows the flow of command analysis.

[Overview]
r_pmbus_command_check() function analyzes the received command.
This function gets the buffer address for receiving by R_IICA0_Get_Receivedata().
And receive the data that is corresponding to each commands in PMBus.
Then, calls r_pmbus_pec_crc8() function. After calculates CRC8, returns result to caller.

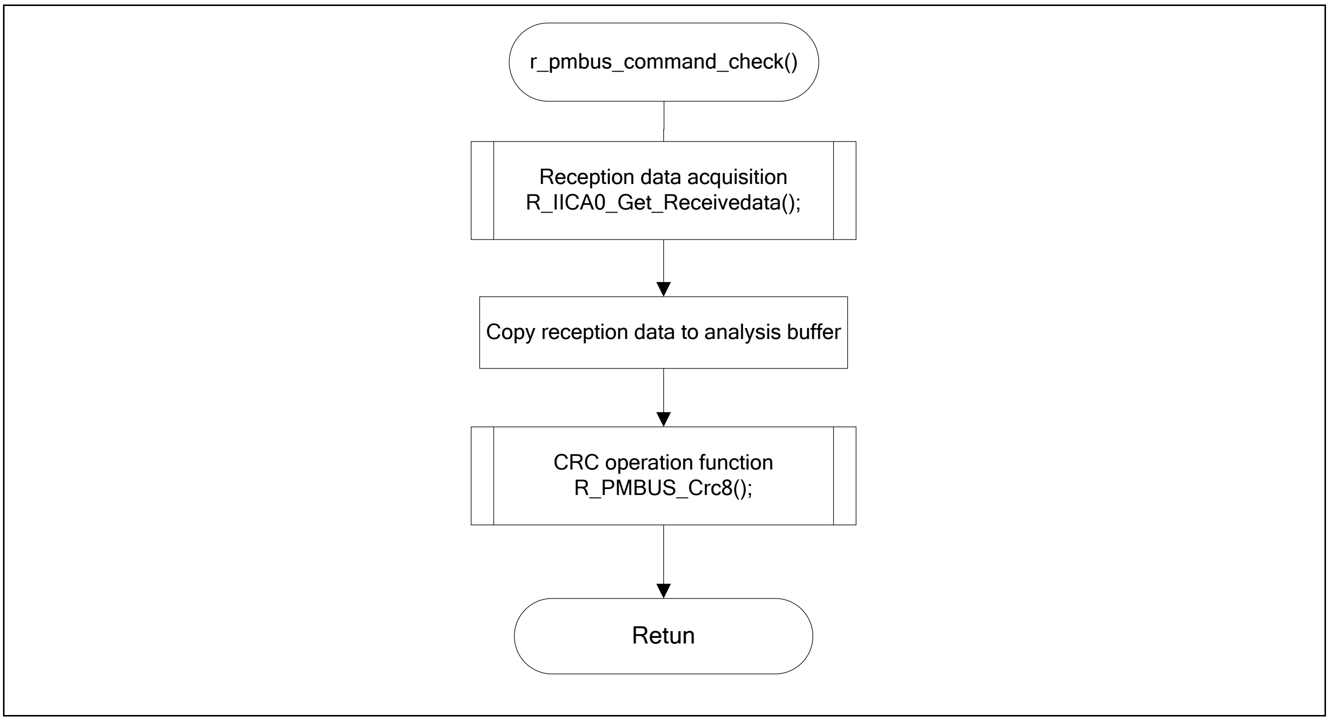

r_pmbus_command_check()
Reception data acquisition
R_IICA0_Get_Receivedata();
Copy reception data to analysis buffer
CRC operation function
R_PMBUS_Crc8();
Retun

Figure 5.15 Command Analysis

5.5.16 Transmission data creation

Figure 5.16 shows the flow of transmission data creation.

[Overview]
r_pmbus_make_senddata() function operates some processes corresponding to received commands.
If received commands is normal, then gets the send data. After that calculates CRC8 by r_pmbus_pec_crc8() function, then copies the send data to the send buffer and send data.
If received command is not defined, this function calculates the error. And then calls R_PMBUS_Fault_Handler() function, sets up error status.

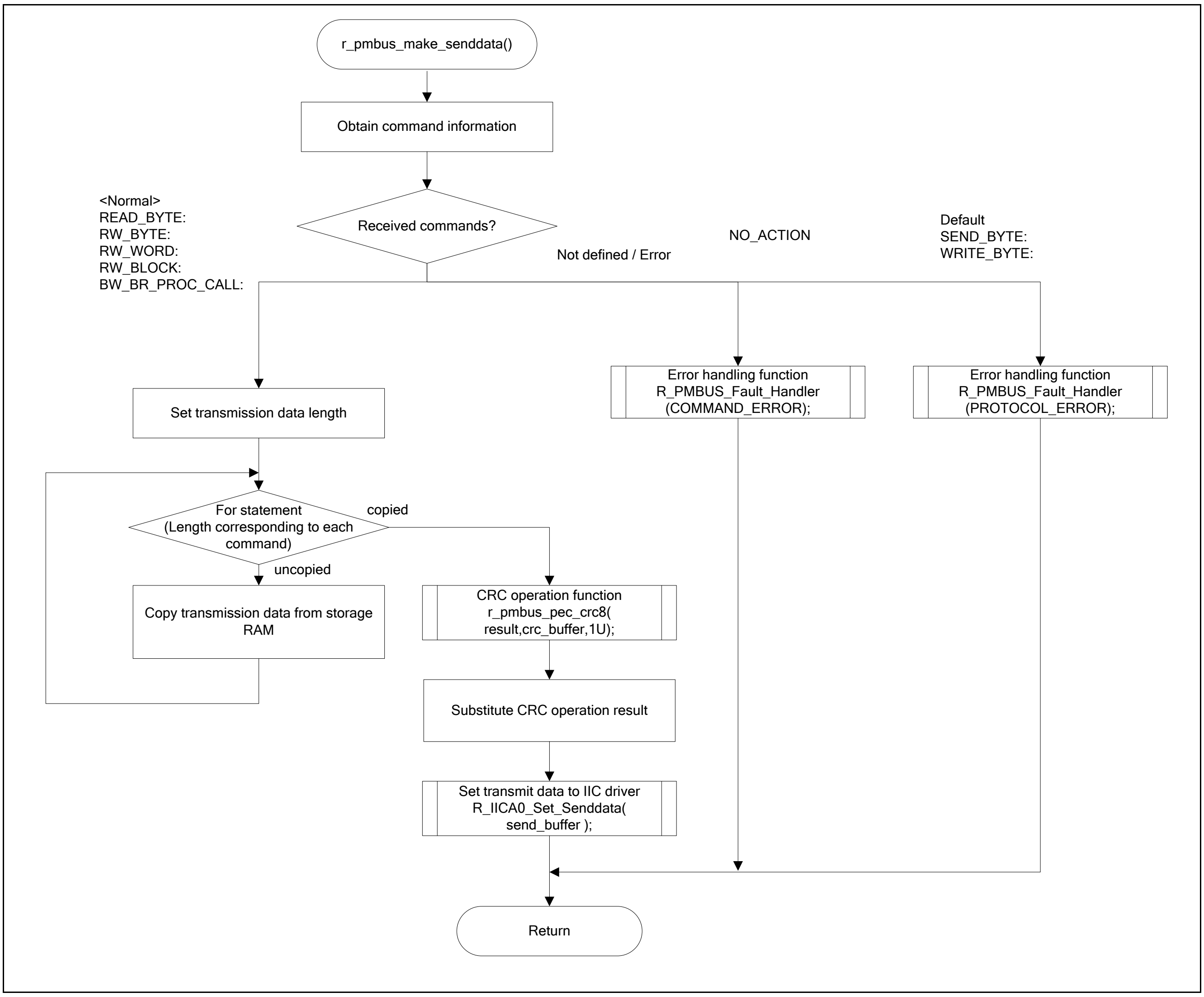


Figure 5.16 Transmission Data Creation

5.5.17 Reception data copy processing

Figure 5.17 shows the flow of reception data copy processing.

[Overview]
r_pmbus_copy_to_ram() function saves received data to ram area of data saving.

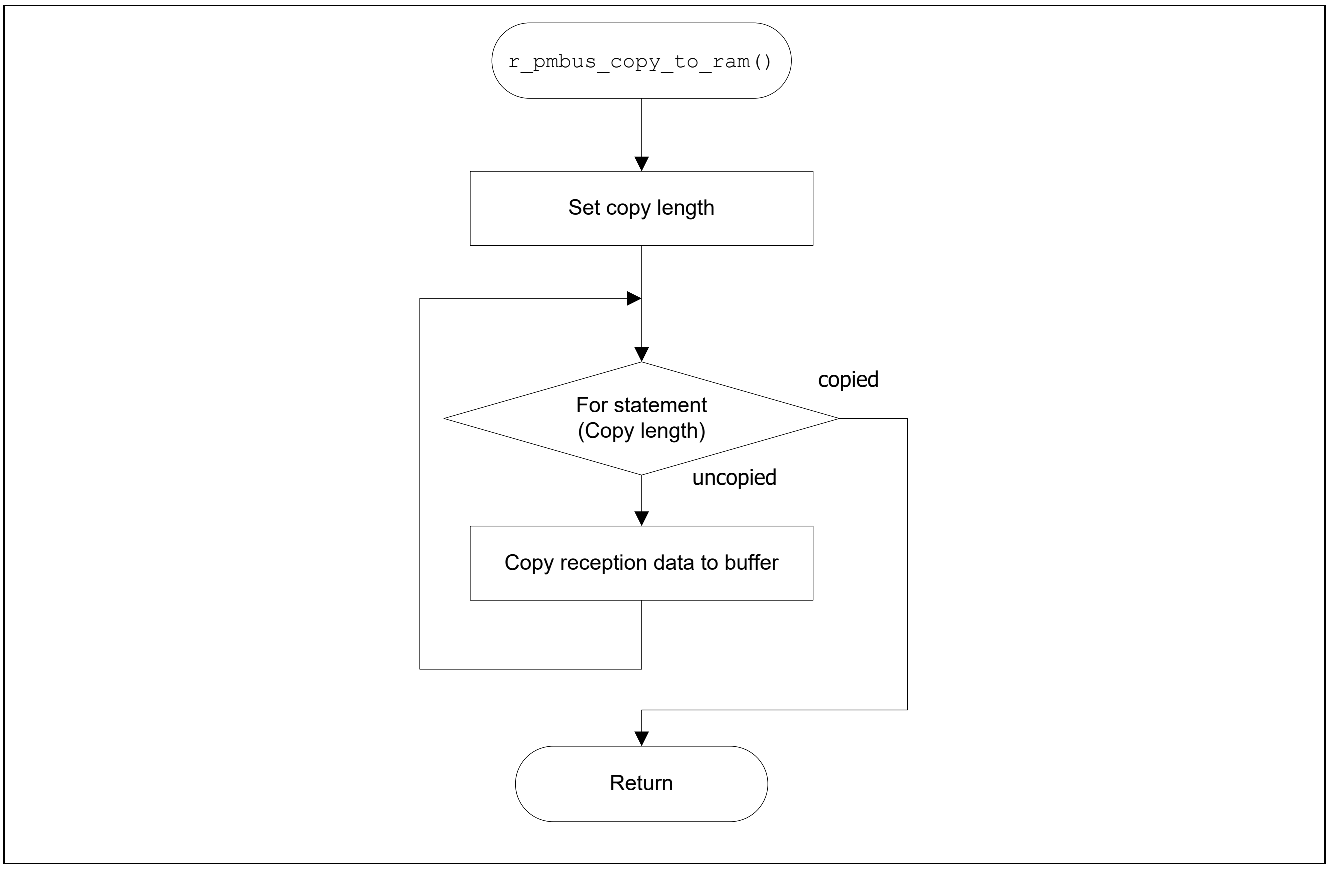


Figure 5.17 Reception Data Copy Processing

5.5.18 Communication error status save processing

Figure 5.18 shows the flow of communication error status save processing.

[Overview]
R_PMBUS_Fault_Handler() function sets up each error status.

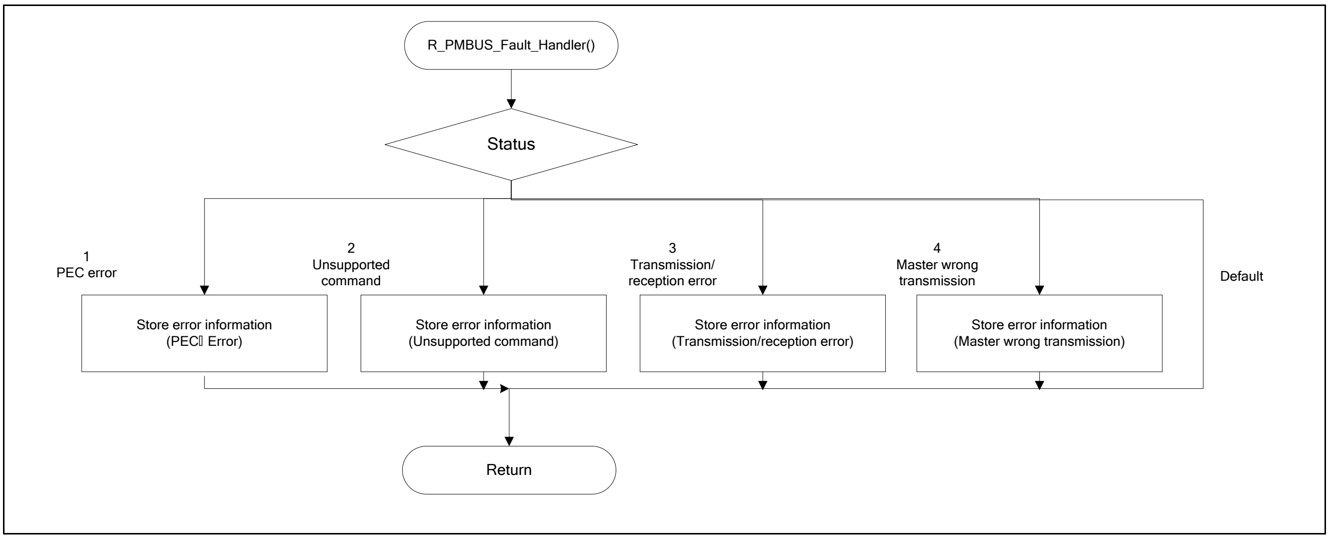


Figure 5.18 Communication Error Status Save Processing

5.5.19 User-specified PMBus function call

Figure 5.19 shows the flow of user-specified PMBus function call processing.

[Overview]
R_PMBUS_User_App() function calls some processes corresponding to each commands.

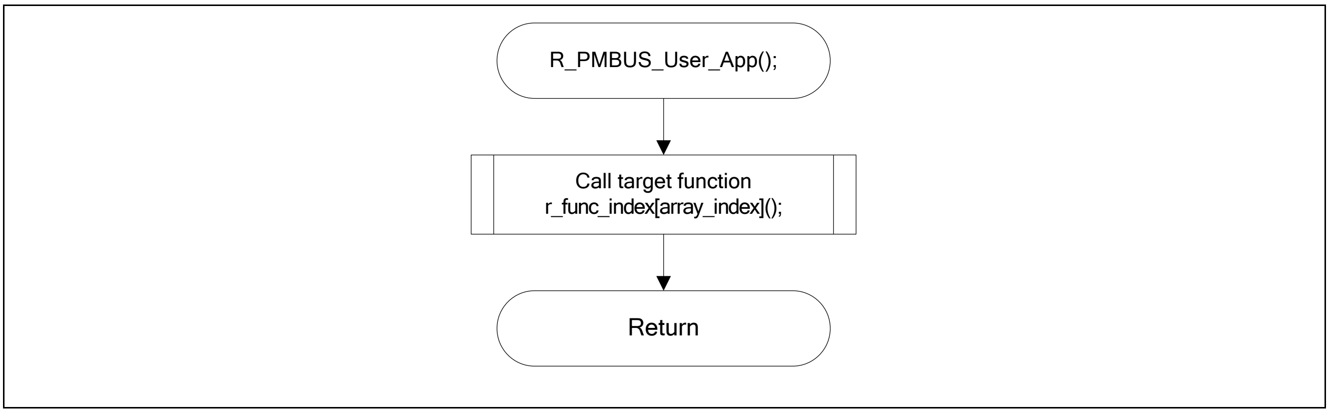


Figure 5.19 User-specified PMBus Function Call Processing

5.6 ROM/RAM Size

The ROM/RAM sizes used in this sample code are shown in Table 5.3. The stack size is the estimated size.

Table 5.3 ROM/RAM Sizes

Environment	ROM Size	RAM Size	Stack Size
CA78K0R Compiler	4.726 Kbytes	0.826 Kbytes	0.063 Kbytes
CC-RL Compiler	4.446 Kbytes	0.438 Kbytes	0.039 Kbytes
IAR Compiler	3.869 Kbytes	0.696 Kbytes	0.063 Kbytes

6. Sample Code

Obtain the sample code from the Renesas Electronics website.

7. Reference Documents

RL78/I1A User's Manual: Hardware (R01UH0169)
RL78 Family User's Manual: Software (R01US0015)

Appendix A Constants

Table A.1 shows the constants used in the sample code.

Table A.1 Constants Used in Sample Code (1/3)

Constant Name	Setting	Description
PEC_ERROR	0x01	Packet error code error (CRC error)
COMMAND_ERROR	0x02	Reception of unsupported command
PROTOCOL_ERROR	0x03	Command protocol error
MASTER_ERROR	0x04	Master protocol error
SMBALERT_HIGH	0x01	For setting the SMBALERT# port output level to "High"
SMBALERT_LOW	0x00	For setting the SMBALERT# port output level to "Low"
UINT8_T_CLR	0x00	For clearing uint8_t variables
UINT16_T_CLR	0x0000	For clearing uint16_t variables
PEC_BYTE	0x01	For packet error code (1 byte) area
PMBUS_RX_ADDRESS	0x00	Location at which to store addresses (Number of reception buffer elements)
PMBUS_RX_COMMAND	0x01	Location at which to store commands (Number of reception buffer elements)
PMBUS_RX_DATA	0x02	Location at which to store data (Number of reception buffer elements)
PMBUS_TX_LENGTH	0x00	Location at which to store length (Number of transmission buffer elements)
PMBUS_TX_ADDRESS	0x01	Location at which to store addresses (Number of transmission buffer elements)
PMBUS_TX_COMMAND	0x02	Location at which to store commands (Number of transmission buffer elements)
IICA0_LENGTH	0x00	Location at which to store length (Number of reception buffer elements)
IICA0_ADDRESS	0x01	Location at which to store addresses (Number of reception buffer elements)
IICA0_COMMAND	0x02	Location at which to store commands (Number of reception buffer elements)
DEFAULT_STS	0x00	Communication status (communication wait state)
RECEIVE_STS	0x01	Communication status (data is being received)
SEND_STS	0x02	Communication status (data is being transmitted)
SEND_ALERT_STS	0x03	Communication status (SMBALERT# is abnormal)
SEND_COMP_STS	0x04	Communication status (data has been transmitted)
TIMER_OFF	0x00	Timer operation disabled
TIMER_ON	0x01	Timer operation enabled
OUTPUT_LOW	0x00	Setting the port output level to "Low"
P0_NCH_ON	0x04	Setting P0 to N-ch open drain
P1_NCH_ON	0x03	Setting P1 to N-ch open drain
PMC02_CLR	0x00	For setting PMC02
ADPC_DI_ON	0x01	For setting ADPC
BIT_OUTPUT_MODE	0x00	For setting the port output mode (in bit units)
BIT_INPUT_MODE	0x01	For setting the port input mode (in bit units)
SMBALERT_PORT_P	P0.2	For setting the SMBALERT# port*
SMBALERT_PORT_PM	PM0.2	For setting the SMBALERT# port mode

Caution The user can change the setting of the constants that specify the port setting.
*Hardware pin prescribed in the PMBus specifications.

Table A.1 Constants Used in Sample Code (2/3)

Constant Name	Setting	Description
SMBALERT_PORT_PMC	PMC0.2	For setting the SMBALERT# port mode control
CONTROL_PORT_P	P2.0	For setting the CONTROL port*
CONTROL_PORT_PM	PM2.0	For setting the CONTROL port mode
WRITE_PROTECT_PORT_P	P7.5	For setting the WRITE_PROTECT port*
WRITE_PROTECT_PORT_PM	PM7.5	Setting the WRITE_PROTECT port mode
ADDRESS_PORT_P	Values input to all ADDRESS ports[Note]	For acquiring the values input to all ADDRESS ports (1st to 7th bits)
ADDRESS_PORT_1_P	P20.0	For setting the ADDRESS port (1st bit)
ADDRESS_PORT_1_PM	PM20.0	For setting the ADDRESS port mode (1st bit)
ADDRESS_PORT_2_P	P20.1	For setting the ADDRESS port (2nd bit)
ADDRESS_PORT_2_PM	PM20.1	For setting the ADDRESS port mode (2nd bit)
ADDRESS_PORT_3_P	P20.2	For setting the ADDRESS port (3rd bit)
ADDRESS_PORT_3_PM	PM20.2	For setting the ADDRESS port mode (3rd bit)
ADDRESS_PORT_4_P	P20.3	For setting the ADDRESS port (4th bit)
ADDRESS_PORT_4_PM	PM20.3	For setting the ADDRESS port mode (4th bit)
ADDRESS_PORT_5_P	P20.4	For setting the ADDRESS port (5th bit)
ADDRESS_PORT_5_PM	PM20.4	For setting the ADDRESS port mode (5th bit)
ADDRESS_PORT_6_P	P20.5	For setting the ADDRESS port (6th bit)
ADDRESS_PORT_6_PM	PM20.5	For setting the ADDRESS port mode (6th bit)
ADDRESS_PORT_7_P	P20.6	For setting the ADDRESS port (7th bit)
ADDRESS_PORT_7_PM	PM20.6	For setting the ADDRESS port mode (7th bit)
TAU_TDR00_VALUE	0x7CFF	Setting value of the compare register for generating 1 ms
TAU_TDR00_START	0x0001	For enabling the timer array unit
TAU_TDR00_STOP	0x0001	For stopping the timer array unit
TMIF00_REQUEST	0x01	For comparison for responding to“1 ms elapsed?” requests
TMIF00_CLR	0x00	For clearing the TMIF00 flag
SEND_BYTE	0x00	Communication protocol (SEND_BYTE)
READ_BYTE	0x01	Communication protocol (READ_BYTE)
WRITE_BYTE	0x02	Communication protocol (WRITE_BYTE)
RW_BYTE	0x03	Communication protocol (RW_BYTE)
READ_WORD	0x04	Communication protocol (READ_WORD)
RW_WORD	0x05	Communication protocol (RW_WORD)
RW_BLOCK	0x06	Communication protocol (RW_BLOCK)
BW_BR_PROC_CALL	0x07	Communication protocol (BW_BR_PROC_CALL)
NO_ACTION	0xFF	Communication protocol (NO_ACTION)
VARIABLE	0x02	Command data length (for user-dependent commands)
VARIABLE_2BYTES_DATA	0x03	Command data length (for user-dependent commands)

Note For ADDRESS_PORT_P, the value must be set as the local address after the input values of ADDRESS_PORT_1_P through ADDRESS_PORT_7_P are converted to 1-byte data.

Caution The user can change the setting of the constants that specify the port setting.
*Hardware pin prescribed in the PMBus specifications.

Table A.1 Constants Used in Sample Code (3/3)

Constant Name	Setting	Description
VARIABLE_3BYTES_DATA	0x04	Command data length (for user-dependent commands)
VARIABLE_4BYTES_DATA	0x05	Command data length (for user-dependent commands)
VARIABLE_5BYTES_DATA	0x06	Command data length (for user-dependent commands)
VARIABLE_6BYTES_DATA	0x07	Command data length (for user-dependent commands)
VARIABLE_7BYTES_DATA	0x08	Command data length (for user-dependent commands)
VARIABLE_8BYTES_DATA	0x09	Command data length (for user-dependent commands)
MFR_DEFINED	0x01	Command data length (for user-dependent commands)
PEC_ERROR_SET_CML	0x20	For setting CML command data (PEC_ERROR information)
COMMAND_ERROR_SET_CML	0x80	For setting CML command data (COMMAND_ERROR information)
PROTOCOL_ERROR_SET_CML	0x40	For setting CML command data (PROTOCOL_ERROR information)
MASTER_ERROR_SET_CML	0x02	For setting CML command data (PEC_ERROR information)
CML_ERROR_SET_STATUS_WORD	0x02	For setting STATUS_WORD command data (CML information)
PMBUS_RX_BUFF_MAX	0x24	Number of elements in the reception buffer for the PMBus control module
PMBUS_TX_BUFF_MAX	0x24	Number of elements in the transmission buffer for the PMBus control module
COPY_TO_RAM_ENABLE	0x01	For enabling copy to RAM
COPY_TO_RAM_DISABLE	0x00	For disabling copy to RAM
TIMER_COUNT_ENABLE	0x01	For enabling timer operation
TIMER_COUNT_DISABLE	0x00	For disabling timer operation
TIMEOUT_COMPARE	0x19	Timeout time elapsed (25 ms)
ALL_COMMAND	0x0100	For initialization of command data (256 commands)
EXCEPT_DATA_BYTE	0x03	For excluding data bytes
ADD_COUNT	0x01	For adding buffer element count
CHECK_NG	0x00	Data check result is abnormal.
CHECK_OK	0x01	Data check result is normal.
DATA_BYTE_LOW	0x00	For storing the error information setting position (lower byte of 1 word)
DATA_BYTE_HIGH	0x01	For storing the error information setting position (upper byte of 1 word)
DATA_BYTE	0x00	For storing the error information setting position (1 byte)

Appendix B Variables

Table B.1 shows the global variables.

Table B.1 Global Variables

Type	Variable Name	Description	Function That Uses This Variable
-	-	-	-

Appendix C The specification of the other functions

This section shows the specifications of the functions used in the sample code.

`R_TAU0_Create`

Overview	Timer array unit initialization processing
Header	`r_timer_module.h`
Declaration	`void R_TAU0_Create(void)`
Description	This function performs the initial settings for using channel 0 of the timer array unit as the interval timer (1 ms).
Parameters	None
Return value	None
Remark	None

`R_TAU0_Channel0_Start`

Overview	Timer array unit operation start processing
Header	`r_timer_module.h`
Declaration	`void R_TAU0_Channel0_Start(void)`
Description	This function starts the operation of channel 0 of the timer array unit.
Parameters	None
Return value	None
Remark	None

`R_TAU0_Channel0_Stop`

Overview	Timer array unit operation stop processing
Header	`r_timer_module.h`
Declaration	`void R_TAU0_Channel0_Stop(void)`
Description	This function stops the operation of channel 0 of the timer array unit.
Parameters	None
Return value	None
Remark	None

`R_TIMER_Control`

Overview	1 ms reference timer generation processing
Header	`r_timer_module.h`
Declaration	`void R_TIMER_Control(void)`
Description	This function generates a 1 ms reference time and clears the TMIF00 flag.
Parameters	None
Return value	None
Remark	None

R_PORT_Create

Overview	Port initialization processing
Header	`r_port_module.h`
Declaration	`void R_PORT_Create(void)`
Description	This function initializes the ports used.
Parameters	None
Return value	None
Remark	None

R_PORT_Smbalert

Overview	SMBALERT# port output setting processing	
Header	`r_port_module.h`	
Declaration	`void R_PORT_Smbalert(uint8_t)`	
Description	This function specifies the SMBALERT# port output setting.	
Parameters	`level_set`	SMBALERT# port output level
Return value	None	
Remark	This sample code provides the functions for the peripheral I/O prescribed in the PMBus specifications. To enable a port, specify the port by using a constant in `r_port_module.h`.	

R_PORT_Get_Address

Overview	ADDRESS port input value acquisition processing	
Header	`r_port_module.h`	
Declaration	`uint8_t R_PORT_Get_Address(void)`	
Description	This function acquires the value set to the local address from the ADDRESS port input.	
Parameters	None	
Return value	`address_result`	Input value (value set as local address)
Remark	This sample code provides the functions for the peripheral I/O prescribed in the PMBus specifications. To enable a port, specify the port by using a constant in `r_port_module.h`.	

R_PORT_Get_Control

Overview	CONTROL port input value acquisition processing	
Header	`r_port_module.h`	
Declaration	`uint8_t R_PORT_Get_Control(void)`	
Description	This function acquires the input value of the CONTROL port.	
Parameters	None	
Return value	`control_result`	Input value (CONTROL port setting value)
Remark	This sample code provides the functions for the peripheral I/O prescribed in the PMBus specifications. To enable a port, specify the port by using a constant in `r_port_module.h`.	

R_PORT_Get_Write_Protect

Overview	WRITE_PROTECT port input value acquisition processing
Header	`r_port_module.h`
Declaration	`uint8_t R_PORT_Get_Write_Protect(void)`
Description	This function acquires the input value of the WRITE_PROTECT port.
Parameters	None
Return value	`write_result` Input value (WRITE_PROTECT port setting value)
Remark	This sample code provides the functions for the peripheral I/O prescribed in the PMBus specifications. To enable a port, specify the port by using a constant in `r_port_module.h`.

R_IICA0_Create

Overview	Serial interface IICA initialization processing
Header	`r_iica0_driver.h`
Declaration	`void R_IICA0_Create(void)`
Description	This function initializes the IICA serial interface.
Parameters	None
Return value	None
Remark	This sample code provides the functions for the peripheral I/O prescribed in the PMBus specifications. For the ports used, the local address setting can be switched to the peripheral I/O or software by switching the compile switch in the `R_IICA0_Create` function. Similarly, the transfer rate can be switched between 100 kbps and 400 kbps.

R_IICA0_Stop

Overview	Serial interface IICA stop processing
Header	`r_iica0_driver.h`
Declaration	`void R_IICA0_Stop(void)`
Description	This function stops the IICA serial interface.
Parameters	None
Return value	None
Remark	None

R_IICA0_Init

Overview	IICA0 driver module initialization
Header	`r_iica0_driver.h`
Declaration	`void R_IICA0_Init(void)`
Description	This function initializes the IICA0 diver module. (Initializes the variables.)
Parameters	None
Return value	None
Remark	None

R_PMBUS_Init

Overview	PMBus control module initialization
Header	r_pmbus_control.h
Declaration	void R_PMBUS_Init(void)
Description	This function initializes the PMBus control module. (Initializes the variables.)
Parameters	None
Return value	None
Remark	None

Appendix D PMBus Commands

This sample code supports all the commands specified in the specifications of PMBus™ (Power Management Bus). Table C.1 shows the commands.

Table C.1 Commands (1/6)

Command Code	Command Name	SMBus Transaction Type	Number of Data Bytes
00h	PAGE	R/W Byte	1
01h	OPERATION	R/W Byte	1
02h	ON_OFF_CONFIG	R/W Byte	1
03h	CLEAR_FAULTS	Send Byte	0
04h	Reserved	-	-
05h	Reserved	-	-
06h	Reserved	-	-
07h	Reserved	-	-
08h	Reserved	-	-
09h	Reserved	-	-
0Ah	Reserved	-	-
0Bh	Reserved	-	-
0Ch	Reserved	-	-
0Dh	Reserved	-	-
0Eh	Reserved	-	-
0Fh	Reserved	-	-
10h	WRITE_PROTECT	R/W Byte	1
11h	STORE_DEFAULT_ALL	Send Byte	0
12h	RESTORE_DEFAULT_ALL	Write Byte	1
13h	STORE_DEFAULT_CODE	Send Byte	0
14h	RESTORE_DEFAULT_CODE	Write Byte	1
15h	STORE_USER_ALL	Send Byte	0
16h	RESTORE_USER_ALL	Write Byte	1
17h	STORE_USER_CODE	Send Byte	0
18h	RESTORE_USER_CODE	Write Byte	1
19h	Reserved	-	-
1Ah	Reserved	-	-
1Bh	Reserved	-	-
1Ch	Reserved	-	-
1Dh	Reserved	-	-
1Eh	Reserved	-	-
1Fh	Reserved	-	-
20h	VOUT_MODE	R/W Byte	1
21h	VOUT_COMMAND	R/W Word	2
22h	VOUT_TRIM	R/W Word	2
23h	VOUT_CAL	R/W Word	2
24h	VOUT_MAX	R/W Word	2
25h	VOUT_MARGIN_HIGH	R/W Word	2
26h	VOUT_MARGIN_LOW	R/W Word	2
27h	VOUT_TRANSITION_RATE	R/W Word	2

Table C.1 Commands (2/6)

Command Code	Command Name	SMBus Transaction Type	Number of Data Bytes
28h	VOUT_DROOP	R/W Word	2
29h	VOLTAGE_SCALE_LOOP	R/W Word	2
2Ah	VOLTAGE_SCALE_MONITOR	R/W Word	2
2Bh	Reserved	-	-
2Ch	Reserved	-	-
2Dh	Reserved	-	-
2Eh	Reserved	-	-
2Fh	Reserved	-	-
30h	COEFFICIENTS	Block R/W Process Call	6 (Includes Byte Count)
31h	POUT_MAX	R/W Word	2
32h	MAX_DUTY	R/W Word	2
33h	FREQUENCY_SWITCH	R/W Word	2
34h	Reserved	-	-
35h	VIN_ON	R/W Word	2
36h	VIN_OFF	R/W Word	2
37h	INTERLEAVE	R/W Word	2
38h	IOUT_SCALE	R/W Word	2
39h	IOUT_CAL_OFFSET	R/W Word	2
3Ah	VFAN_1	R/W Word	2
3Bh	VFAN_2	R/W Word	2
3Ch	Reserved	-	-
3Dh	Reserved	-	-
3Eh	Reserved	-	-
3Fh	Reserved	-	-
40h	VOUT_OV_FAULT_LIMIT	R/W Word	2
41h	VOUT_OV_FAULT_RESPONSE	R/W Byte	1
42h	VOUT_OV_WARN_LIMIT	R/W Word	2
43h	VOUT_UV_WARN_LIMIT	R/W Word	2
44h	VOUT_UV_FAULT_LIMIT	R/W Word	2
45h	VOUT_UV_FAULT_RESPONSE	R/W Byte	1
46h	IOUT_OC_FAULT_LIMIT	R/W Word	2
47h	IOUT_OC_FAULT_RESPONSE	R/W Byte	1
48h	IOUT_OC_LV_FAULT_LIMIT	R/W Word	2
49h	IOUT_OC_LV_FAULT_RESPONSE	R/W Byte	1
4Ah	IOUT_OC_WARN_LIMIT	R/W Word	2
4Bh	IOUT_UC_FAULT_LIMIT	R/W Word	2
4Ch	IOUT_UC_FAULT_RESPONSE	R/W Byte	1
4Dh	Reserved for POUT_FAULT_LIMIT	R/W Word	2
4Eh	Reserved for POUT_MAX_FAULT_RESPONSE	R/W Byte	1
4Fh	OT_FAULT_LIMIT	R/W Word	2
50h	OT_FAULT_RESPONSE	R/W Byte	1
51h	OT_WARN_LIMIT	R/W Word	2
52h	UT_WARN_LIMIT	R/W Word	2
53h	UT_FAULT_LIMIT	R/W Word	2

Table C.1 Commands (3/6)

Command Code	Command Name	SMBus Transaction Type	Number of Data Bytes
54h	UT_FAULT_RESPONSE	R/W Byte	1
55h	VIN_OV_FAULT_LIMIT	R/W Word	2
56h	VIN_OV_FAULT_RESPONSE	R/W Byte	1
57h	VIN_OV_WARN_LIMIT	R/W Word	2
58h	VIN_UV_WARN_LIMIT	R/W Word	2
59h	VIN_UV_FAULT_LIMIT	R/W Word	2
5Ah	VIN_UV_FAULT_RESPONSE	R/W Byte	1
5Bh	IIN_OC_FAULT_LIMIT	R/W Word	2
5Ch	IIN_OC_FAULT_RESPONSE	R/W Byte	1
5Dh	IIN_OC_WARN_LIMIT	R/W Word	2
5Eh	POWER_GOOD_ON	R/W Word	2
5Fh	POWER_GOOD_OFF	R/W Word	2
60h	TON_DELAY	R/W Word	2
61h	TON_RISE	R/W Word	2
62h	TON_MAX_FAULT_LIMIT	R/W Word	2
63h	TON_MAX_FAULT_RESPONSE	R/W Byte	1
64h	TOFF_DELAY	R/W Word	2
65h	TOFF_FALL	R/W Word	2
66h	TOFF_MAX_FAULT_LIMIT	R/W Word	2
67h	TOFF_MAX_FAULT_RESPONSE	R/W Byte	1
68h	Reserved	-	-
69h	Reserved	-	-
6Ah	Reserved	-	-
6Bh	Reserved	-	-
6Ch	Reserved	-	-
6Dh	Reserved	-	-
6Eh	Reserved	-	-
6Fh	Reserved	-	-
70h	Reserved (Test Input Fuse A)	-	-
71h	Reserved (Test Input Fuse B)	-	-
72h	Reserved (Test Input OR-ing A)	-	-
73h	Reserved (Test Input OR-ing B)	-	-
74h	Reserved (Test Output OR-ing)	-	-
75h	Reserved	-	-
76h	Reserved	-	-
77h	Reserved	-	-
78h	STATUS_BYTE	Read Byte	1
79h	STATUS_WORD	Read Word	2
7Ah	STATUS_VOUT	Read Byte	1
7Bh	STATUS_IOUT	Read Byte	1
7Ch	STATUS_INPUT	Read Byte	1
7Dh	STATUS_TEMPERATURE	Read Byte	1
7Eh	STATUS_CML	Read Byte	1
7Fh	STATUS_OTHER	Read Byte	1
80h	STATUS_MFR_SPECIFIC	Read Byte	1
81h	Reserved	-	-
82h	Reserved	-	-

Table C.1 Commands (4/6)

Command Code	Command Name	SMBus Transaction Type	Number of Data Bytes
83h	Reserved	-	-
84h	Reserved	-	-
85h	Reserved	-	-
86h	Reserved	-	-
87h	Reserved	-	-
88h	READ_VIN	Read Word	2
89h	READ_IIN	Read Word	2
8Ah	READ_VCAP	Read Word	2
8Bh	READ_VOUT	Read Word	2
8Ch	READ_IOUT	Read Word	2
8Dh	READ_TEMPERATURE_1	Read Word	2
8Eh	READ_TEMPERATURE_2	Read Word	2
8Fh	READ_TEMPERATURE_3	Read Word	2
90h	READ_FAN_SPEED_1	Read Word	2
91h	READ_FAN_SPEED_2	Read Word	2
92h	READ_VFAN_1	Read Word	2
93h	READ_VFAN_2	Read Word	2
94h	READ_DUTY_CYCLE	Read Word	2
95h	READ_FREQUENCY	Read Word	2
96h	Reserved	-	-
97h	Reserved	-	-
98h	PMBUS_REVISION	Read Byte	1
99h	MFR_ID	R/W Block	Variable
9Ah	MFR_MODEL	R/W Block	Variable
9Bh	MFR_REVISION	R/W Block	Variable
9Ch	MFR_LOCATION	R/W Block	Variable
9Dh	MFR_DATE	R/W Block	Variable
9Eh	MFR_SERIAL	R/W Block	Variable
9Fh	Reserved	-	-
A0h	MFR_VIN_MIN	Read Word	2
A1h	MFR_VIN_MAX	Read Word	2
A2h	MFR_IIN_MAX	Read Word	2
A3h	MFR_PIN_MAX	Read Word	2
A4h	MFR_VOUT_MIN	Read Word	2
A5h	MFR_VOUT_MAX	Read Word	2
A6h	MFR_IOUT_MAX	Read Word	2
A7h	MFR_POUT_MAX	Read Word	2
A8h	MFR_TAMBIENT_MAX	Read Word	2
A9h	MFR_TAMBIENT_MIN	Read Word	2
AAh	Reserved	-	-
ABh	Reserved	-	-
ACh	Reserved	-	-
ADh	Reserved	-	-
AEh	Reserved	-	-
AFh	Reserved	-	-
B0h	USER_DATA_00	Block R/W	Variable
B1h	USER_DATA_01	Block R/W	Variable

Table C.1 Commands (5/6)

Command Code	Command Name	SMBus Transaction Type	Number of Data Bytes
B2h	USER_DATA_02	Block R/W	Variable
B3h	USER_DATA_03	Block R/W	Variable
B4h	USER_DATA_04	Block R/W	Variable
B5h	USER_DATA_05	Block R/W	Variable
B6h	USER_DATA_06	Block R/W	Variable
B7h	USER_DATA_07	Block R/W	Variable
B8h	USER_DATA_08	Block R/W	Variable
B9h	USER_DATA_09	Block R/W	Variable
BAh	USER_DATA_10	Block R/W	Variable
BBh	USER_DATA_11	Block R/W	Variable
BCh	USER_DATA_12	Block R/W	Variable
BDh	USER_DATA_13	Block R/W	Variable
BEh	USER_DATA_14	Block R/W	Variable
BFh	USER_DATA_15	Block R/W	Variable
C0h	Reserved	-	-
C1h	Reserved	-	-
C2h	Reserved	-	-
C3h	Reserved	-	-
C4h	Reserved	-	-
C5h	Reserved	-	-
C6h	Reserved	-	-
C7h	Reserved	-	-
C8h	Reserved	-	-
C9h	Reserved	-	-
CAh	Reserved	-	-
CBh	Reserved	-	-
CCh	Reserved	-	-
CDh	Reserved	-	-
CEh	Reserved	-	-
CFh	Reserved	-	-
D0h	MFR_SPECIFIC_00	Mfr.Defined	Mfr.Defined
D1h	MFR_SPECIFIC_01	Mfr.Defined	Mfr.Defined
D2h	MFR_SPECIFIC_02	Mfr.Defined	Mfr.Defined
D3h	MFR_SPECIFIC_03	Mfr.Defined	Mfr.Defined
D4h	MFR_SPECIFIC_04	Mfr.Defined	Mfr.Defined
D5h	MFR_SPECIFIC_05	Mfr.Defined	Mfr.Defined
D6h	MFR_SPECIFIC_06	Mfr.Defined	Mfr.Defined
D7h	MFR_SPECIFIC_07	Mfr.Defined	Mfr.Defined
D8h	MFR_SPECIFIC_08	Mfr.Defined	Mfr.Defined
D9h	MFR_SPECIFIC_09	Mfr.Defined	Mfr.Defined
DAh	MFR_SPECIFIC_10	Mfr.Defined	Mfr.Defined
DBh	MFR_SPECIFIC_11	Mfr.Defined	Mfr.Defined
DCh	MFR_SPECIFIC_12	Mfr.Defined	Mfr.Defined
DDh	MFR_SPECIFIC_13	Mfr.Defined	Mfr.Defined
DEh	MFR_SPECIFIC_14	Mfr.Defined	Mfr.Defined
DFh	MFR_SPECIFIC_15	Mfr.Defined	Mfr.Defined
E0h	MFR_SPECIFIC_16	Mfr.Defined	Mfr.Defined

Table C.1 Commands (6/6)

Command Code	Command Name	SMBus Transaction Type	Number of Data Bytes
E1h	MFR_SPECIFIC_17	Mfr.Defined	Mfr.Defined
E2h	MFR_SPECIFIC_18	Mfr.Defined	Mfr.Defined
E3h	MFR_SPECIFIC_19	Mfr.Defined	Mfr.Defined
E4h	MFR_SPECIFIC_20	Mfr.Defined	Mfr.Defined
E5h	MFR_SPECIFIC_21	Mfr.Defined	Mfr.Defined
E6h	MFR_SPECIFIC_22	Mfr.Defined	Mfr.Defined
E7h	MFR_SPECIFIC_23	Mfr.Defined	Mfr.Defined
E8h	MFR_SPECIFIC_24	Mfr.Defined	Mfr.Defined
E9h	MFR_SPECIFIC_25	Mfr.Defined	Mfr.Defined
EAh	MFR_SPECIFIC_26	Mfr.Defined	Mfr.Defined
EBh	MFR_SPECIFIC_27	Mfr.Defined	Mfr.Defined
ECh	MFR_SPECIFIC_28	Mfr.Defined	Mfr.Defined
EDh	MFR_SPECIFIC_29	Mfr.Defined	Mfr.Defined
EEh	MFR_SPECIFIC_30	Mfr.Defined	Mfr.Defined
EFh	MFR_SPECIFIC_31	Mfr.Defined	Mfr.Defined
F0h	MFR_SPECIFIC_32	Mfr.Defined	Mfr.Defined
F1h	MFR_SPECIFIC_33	Mfr.Defined	Mfr.Defined
F2h	MFR_SPECIFIC_34	Mfr.Defined	Mfr.Defined
F3h	MFR_SPECIFIC_35	Mfr.Defined	Mfr.Defined
F4h	MFR_SPECIFIC_36	Mfr.Defined	Mfr.Defined
F5h	MFR_SPECIFIC_37	Mfr.Defined	Mfr.Defined
F6h	MFR_SPECIFIC_38	Mfr.Defined	Mfr.Defined
F7h	MFR_SPECIFIC_39	Mfr.Defined	Mfr.Defined
F8h	MFR_SPECIFIC_40	Mfr.Defined	Mfr.Defined
F9h	MFR_SPECIFIC_41	Mfr.Defined	Mfr.Defined
FAh	MFR_SPECIFIC_42	Mfr.Defined	Mfr.Defined
FBh	MFR_SPECIFIC_43	Mfr.Defined	Mfr.Defined
FCh	MFR_SPECIFIC_44	Mfr.Defined	Mfr.Defined
FDh	MFR_SPECIFIC_45	Mfr.Defined	Mfr.Defined
FEh	MFR_SPECIFIC_COMMAND EXT	Extended Command	1 or 2
FFh	PMBUS_COMMAND_EXT	Extended Command	Or 2

Appendix E Adding and Deleting Supported Commands

This sample code allows the user to add or delete supported commands.
The following shows how to add or delete commands.

■ Adding or deleting commands

Step 1: Changing `r_matrix[256]` of the `r_pmbus_control.c` file

`r_matrix[256]` stores the information of each command.

Command information:

- Communication protocol
- Command data length
- Address of the command data storage area

To add or delete supported commands, you must change the information of each command.
To add a supported command, the buffer for storing command data must be reserved.

Example: To add a command to 0x04

Before change

```
/******************************************************

 PMBus Protocol Table (0x00-0xFF)

******************************************************/

static const command  r_matrix[256] = {

/*  Protocol------read/write bytes---Command_ram_address------------Command No  */

{ RW_BYTE,  0x1,         r_pmbus_page         },      /* 0x0 */

{ RW_BYTE,  0x1,         r_pmbus_opration     },      /* 0x1 */

{ RW_BYTE,  0x1  r_pmbus_on_off_config},      /* 0x2 */

{ SEND_BYTE, 0x0,        0x0                  },      /* 0x3 */

{ NO_ACTION, 0x0,        r_pmbus_dummy},      /* 0x4 */

               ...

};
```

After change

Declare the buffer storing the data of the command to add.

```
static uint8_t r_pmbus_user[1];       /* 0x4 */

/******************************************************

 PMBus Protocol Table (0x00-0xFF)

******************************************************/

static const command  r_matrix[256] = {

/*  Protocol------read/write bytes---Command_ram_address------------Command No  */

{ RW_BYTE,  0x1,         r_pmbus_page         },      /* 0x0 */

{ RW_BYTE,  0x1,         r_pmbus_opration     },      /* 0x1 */

{ RW_BYTE,  0x1  r_pmbus_on_off_config},      /* 0x2 */

{ SEND_BYTE, 0x0,        0x0                  }

{ RW_BYTE,  0x1,         r_pmbus_user         },
```

Describe the address of the buffer storing the data of the command to add.
* The buffer is declared by the user.

Add the communication protocol of the command to add.

Add the data length of the command to add.

Example: To delete the command at 0x01

Before change

```
/******************************************************
 PMBus Protocol Table (0x00-0xFF)
******************************************************/
static const command  r_matrix[256] = {
/*  Protocol------read/write bytes---Command_ram_address------------Command No  */
{ RW_BYTE,  0x1,       r_pmbus_page           },      /* 0x0 */
{ RW_BYTE,  0x1,       r_pmbus_opration       },      /* 0x1 */
{ RW_BYTE,  0x1r_pmbus_on_off_config },      /* 0x2 */
{ SEND_BYTE, 0x0,      0x0                    },      /* 0x3 */
{ NO_ACTION, 0x0,      r_pmbus_dummy          },      /* 0x4 */
               ...
};
```

After change

```
/******************************************************
 PMBus Protocol Table (0x00-0xFF)
******************************************************/
static const command  r_matrix[256] = {
/*  Protocol------read/write bytes---Command_ram_address------------Command No  */
{ RW_BYTE,  0x1,       r_pmbus_page           },
{ NO_ACTION, 0x0,      r_pmbus_dummy          },
                                        },      /* 0x2 */
                                              },      /* 0x3 */
                       r_pmbus_dummy          },      /* 0x4 */
               ...
};
```

Change the address of the buffer storing the data of the command to delete to r_pmbus_dummy.

Change the communication protocol of the command to delete to NO_ACTION.

Change the data length of the command to delete to 0.

Step 2: Changing `r_func_index[256]` of the `r_pmbus_userapp.c` file
`r_func_index[256]` is a function pointer array.
It stores the address of the response processing function of each command.
To add or delete supported commands, the function pointer array must be registered or deleted.

Example: To add a command to 0x04

Before change

```
/******************************************************

 PMBus Protocol Table (0x00-0xFF)

*******************************************************/

static void (* const r_func_index[256])( void ) = {

/* User function */

        &r_pmbus_user_page,           /* 0x0 */

        &r_pmbus_user_operation,      /* 0x1 */

        &r_pmbus_user_on_off_config,  /* 0x2 */

        &r_pmbus_user_clear_faults,   /* 0x3 */

        &r_pmbus_user_dummy,          /* 0x4 */

                ...

};
```

After change

```
/******************************************************

 PMBus Protocol Table (0x00-0xFF)

*******************************************************/

static void (* const r_func_index[256])( void ) = {

/* User function */

        &r_pmbus_user_page,           /* 0x0 */

        &r_pmbus_user_operation,      /* 0x1 */

        &r_pmbus_user_on_off_config,  /* 0x2 */

        &r_pmbus_user_clear_faults,   /* 0x3 */

        &r_pmbus_user_add,            /* 0x4 */

                ...

};
```

Register the function of the command to add.

Example: To delete the command at 0x01

Before change

```
/******************************************************
 PMBus Protocol Table (0x00-0xFF)
*******************************************************/
static void (* const r_func_index[256])( void ) = {
/* User function */
        &r_pmbus_user_page,           /* 0x0 */
        &r_pmbus_user_operation,      /* 0x1 */
        &r_pmbus_user_on_off_config,  /* 0x2 */
        &r_pmbus_user_clear_faults,   /* 0x3 */
        &r_pmbus_user_dummy,          /* 0x4 */
                ...
};
```

After change

```
/******************************************************
 PMBus Protocol Table (0x00-0xFF)
*******************************************************/
static void (* const r_func_index[256])( void ) = {
/* User function */
        &r_pmbus_user_page,
        &r_pmbus_user_dummy,          /* 0x1 */
        &r_pmbus_user_on_off_config,  /* 0x2 */
        &r_pmbus_user_clear_faults,   /* 0x3 */
        &r_pmbus_user_dummy,          /* 0x4 */
                ...
};
```

Change the function of the command to delete to r_pmbus_user_dummy

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact

Revision History

Rev.	Date	Description	
		Page	**Summary**
1.00	Mar. 25, 2013	-	First edition issued
1.10	Dec. 10, 2020	6 30	Support CC-RL and IAR compilers Add 5.6 ROM/RAM Size

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/.