APPLICATION NOTE

M32C/84, 85, 87, 88 Groups

Example of Rewriting the User ROM Area Using EW1 Mode

REJ05B1424-0100
Rev.1.00
Nov. 15, 2010

1. Abstract

This document describes an example of using EW1 mode in flash memory version.

2. Introduction

The application example described in this document applies to the following microcomputers (MCUs):

MCUs: M32C/84 Group, M32C/85 Group, M32C/87 Group, and M32C/88 Group

This application note can be used with other M32C/80 Series MCUs which have the same special function registers (SFRs) as the above groups. Check the manuals for any modifications to functions. Careful evaluation is recommended before using the program described in this application note.

3. CPU Rewrite Modes

The CPU rewrite mode consists of EW0 mode and EW1 mode.

3.1 EW1 Mode Features

EW1 mode allows the user to rewrite a different block from that of the CPU rewrite program by allocating the CPU rewrite program to any block in the user ROM area, and issuing program and erase commands.

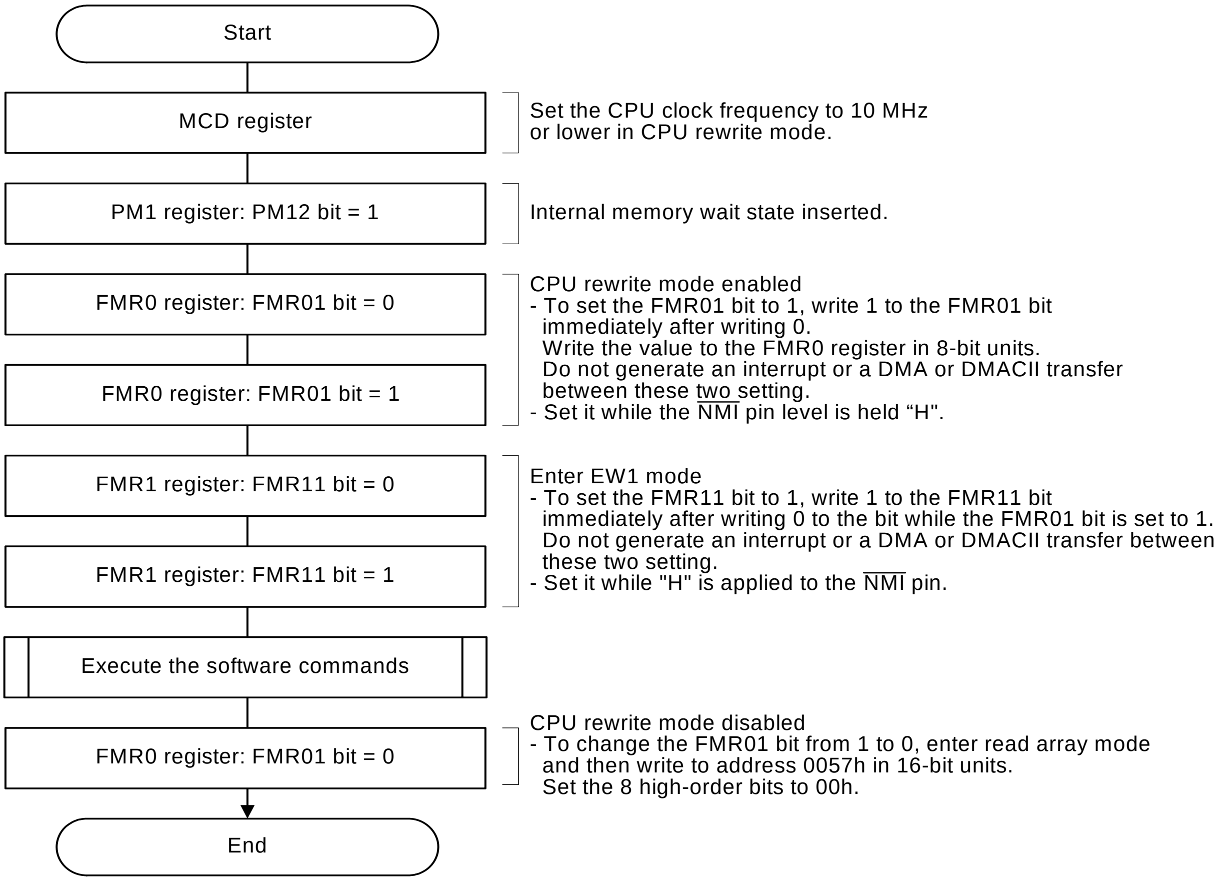
During programming or erasing, peripheral function interrupts, DMA requests, and DMACII requests are not accepted.

3.2 EW1 Mode Settings

After setting the FMR01 bit in the FMR0 register to 1 (CPU rewrite mode enabled), the CPU enters EW1 mode by setting the FMR11 bit in the FMR1 register.

Read the FMR0 register to determine the status of program and erase operations when completed. In EW1 mode, the status register cannot be read.

Figure 3.1 shows the Setting Procedure for EW1 Mode.



Start
MCD register
PM1 register: PM12 bit = 1
FMR0 register: FMR01 bit = 0
FMR0 register: FMR01 bit = 1
FMR1 register: FMR11 bit = 0
FMR1 register: FMR11 bit = 1
Execute the software commands
FMR0 register: FMR01 bit = 0
End

Set the CPU clock frequency to 10 MHz or lower in CPU rewrite mode.

Internal memory wait state inserted.

CPU rewrite mode enabled
- To set the FMR01 bit to 1, write 1 to the FMR01 bit immediately after writing 0.
  Write the value to the FMR0 register in 8-bit units.
  Do not generate an interrupt or a DMA or DMACII transfer between these two setting.
- Set it while the $\overline{\text{NMI}}$ pin level is held “H".

Enter EW1 mode
- To set the FMR11 bit to 1, write 1 to the FMR11 bit immediately after writing 0 to the bit while the FMR01 bit is set to 1.
  Do not generate an interrupt or a DMA or DMACII transfer between these two setting.
- Set it while "H" is applied to the $\overline{\text{NMI}}$ pin.

CPU rewrite mode disabled
- To change the FMR01 bit from 1 to 0, enter read array mode and then write to address 0057h in 16-bit units.
  Set the 8 high-order bits to 00h.

Note:
1. Do not use EW1 mode in memory expansion mode or boot mode.

Figure 3.1 Setting Procedure for EW1 Mode

3.2.1 Memory Map

Figure 3.2 shows the Flash Memory Map for the M32C/87 Group (M32C/87, M32C/87A, and M32C/87B). Refer to the respective hardware user's manuals for details of other MCUs.

The user ROM area has an area to store programs, and another 4-Kbyte area as the block A for data storage.

The user ROM area is divided into blocks, each of which can be protected (locked) from erasing or programming.

The user ROM area can be rewritten in CPU rewrite mode, standard serial I/O mode, or parallel I/O mode.

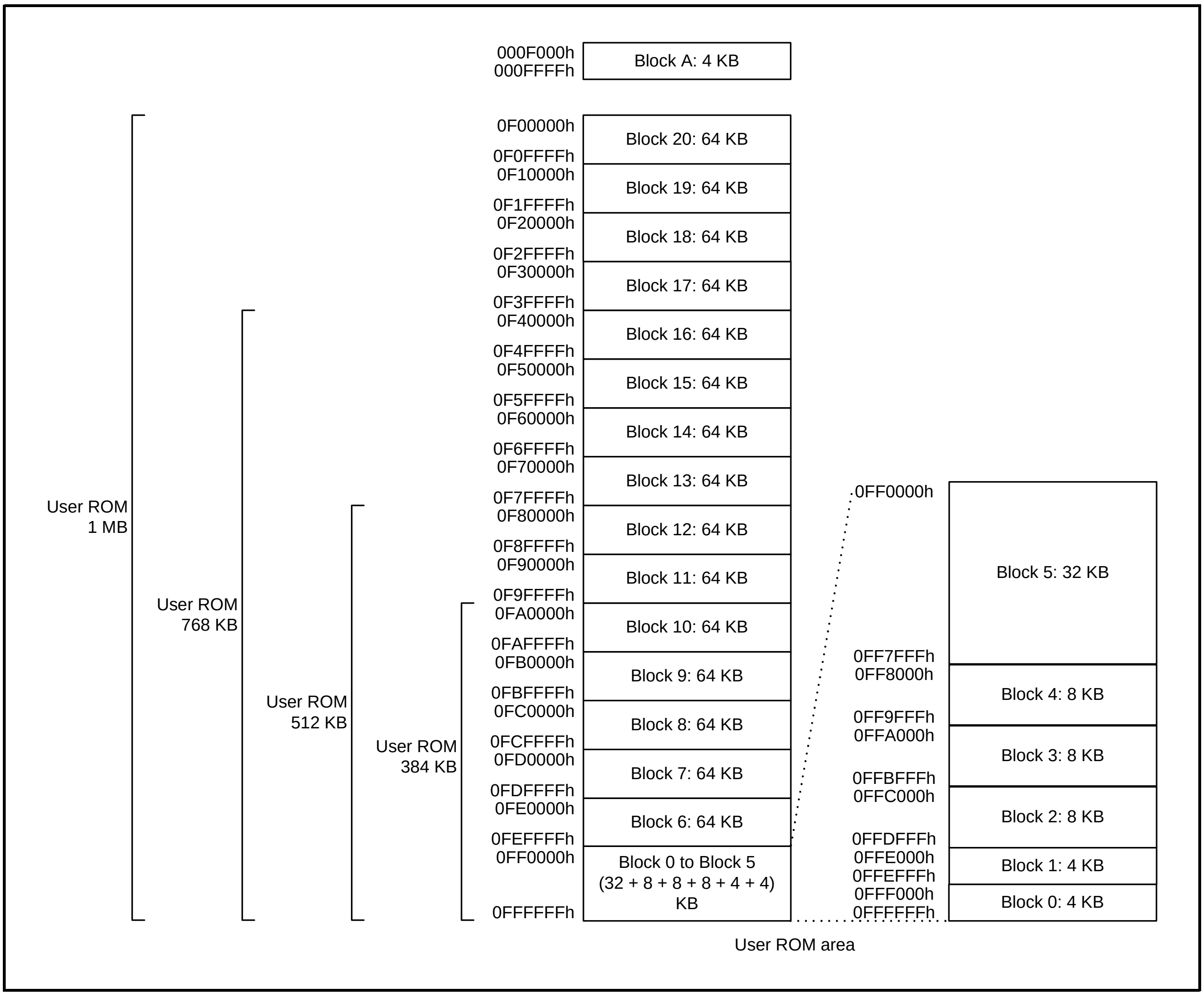


Figure 3.2 Flash Memory Map

3.3 Notes on EW1 Mode

3.3.1 Operating Speed

Prior to entering EW1 mode, set the CPU clock frequency to 10 MHz or lower using
bits MCD4 to MCD0 in the MCD register, and also set the PM12 bit in the PM1 register to 1 (1 wait state).

3.3.2 Interrupts

- When an interrupt request is generated by the peripheral function or watchdog timer (when the PM22 bit in the PM2 register is set to 0) during the erase or program operation, the interrupt is acknowledged after the erase or program operation is completed.
- When an interrupt request is generated by the $\overline{\text{NMI}}$, watchdog timer (when the PM22 bit is set to 1), Vdet4 detection function, or oscillation stop detection function, registers FMR0 and FMR1 are forcibly initialized and the erase or program operation in progress is aborted. Now that the flash memory can be accessed, the interrupt routine will be executed.

3.3.3 How to Access

To set the FMR01 or FMR02 bit in the FMR0 register, or the FMR11 bit in the FMR1 register to 1, write 1 immediately after writing 0 to the bit. Write to the FMR0 or FMR1 register in 8-bit units. Do not generate an interrupt or a DMA or DMACII transfer between these two settings. Also, set these bits while a high-level signal is applied to the $\overline{\text{NMI}}$ pin.
To change the FMR01 bit from 1 to 0, enter read array mode first, and then write into address 0057h in 16-bit units. Set the 8 high-order bits to 00h.

3.3.4 Rewriting User ROM Area

Do not rewrite a block where the rewrite control program is stored.

3.3.5 Writing Command and Data

Write command codes and data to even addresses in the user ROM area.

3.3.6 Block Erase

If an erase operation in progress is aborted due to such as the $\overline{\text{NMI}}$ interrupt, hardware reset, or supply voltage drop, the lock bit of the block which has been erased may become 0 (locked). To erase the same block again, set the FMR02 bit in the FMR0 register to 1 (lock bit disabled) and then execute the block erase command.

3.3.7 Wait Mode

To enter wait mode, set the FMR01 bit in the FMR0 register to 0 (CPU rewrite mode disabled) and then execute the WAIT instruction.

3.3.8 Stop Mode

To enter stop mode, use the following procedure:

- Set the FMR01 bit to 0 (CPU rewrite mode disabled) before setting the CM10 bit to 1 (stop mode).
- Execute the JMP.B instruction right after the instruction to set the CM10 bit in the CM1 register to 1 (stop mode).

```
Example:    BSET 0, CM1; Stop mode
            JMP.B L1
         L1:
            Program after exiting stop mode
```

3.3.9 Low-Power Consumption Mode and On-Chip Oscillator Low-Power Consumption Mode

When the CM05 bit in the CM0 register is set to 1 (main clock stopped), do not execute the following commands:

- Program command
- Block erase command
- Lock bit program command
- Read lock bit status command

4. Description of the Application Example

This application note describes an example of a monitor program where the sample program is received from the master device, and the sample program execute and program ROM area rewrite commands are executed.
Figure 4.1 shows the System Structure Diagram.

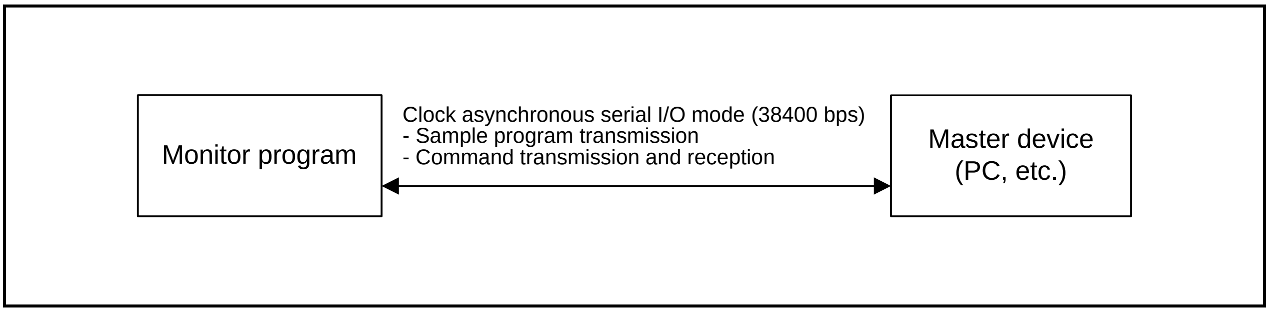

Monitor program
Clock asynchronous serial I/O mode (38400 bps)
- Sample program transmission
- Command transmission and reception
Master device
(PC, etc.)

Figure 4.1 System Structure Diagram

Clocks used in this application note are listed in the following table.

Table 4.1 Clock Conditions

Item	Frequency
Main clock	10 MHz
PLL frequency	30 MHz (multiply by 6 then divide by 2)

Control commands used in this application note are listed in the following table.

Table 4.2 Control Commands

<table>
<tr><th>Control Command Name</th><th>Command Explanation</th><th>1st to 3rd Bytes</th><th>4th to 5th Bytes</th><th colspan="3">After 6th Byte</th></tr>
<tr><td rowspan="2">Program (write) command</td><td rowspan="2">Execute to write the received data</td><td rowspan="2">"prg"</td><td rowspan="2">Size (2 bytes)</td><td>Data (max. 256 bytes)</td><td>SUM value (2 bytes)</td><td>Results (1)</td></tr>
<tr><td colspan="3">The data, SUM value transmission, and received results are repeated up to the program size.</td></tr>
<tr><td>Erase command</td><td>Erases the program ROM area</td><td>"ers"</td><td>Results (1)</td><td colspan="3"></td></tr>
</table>

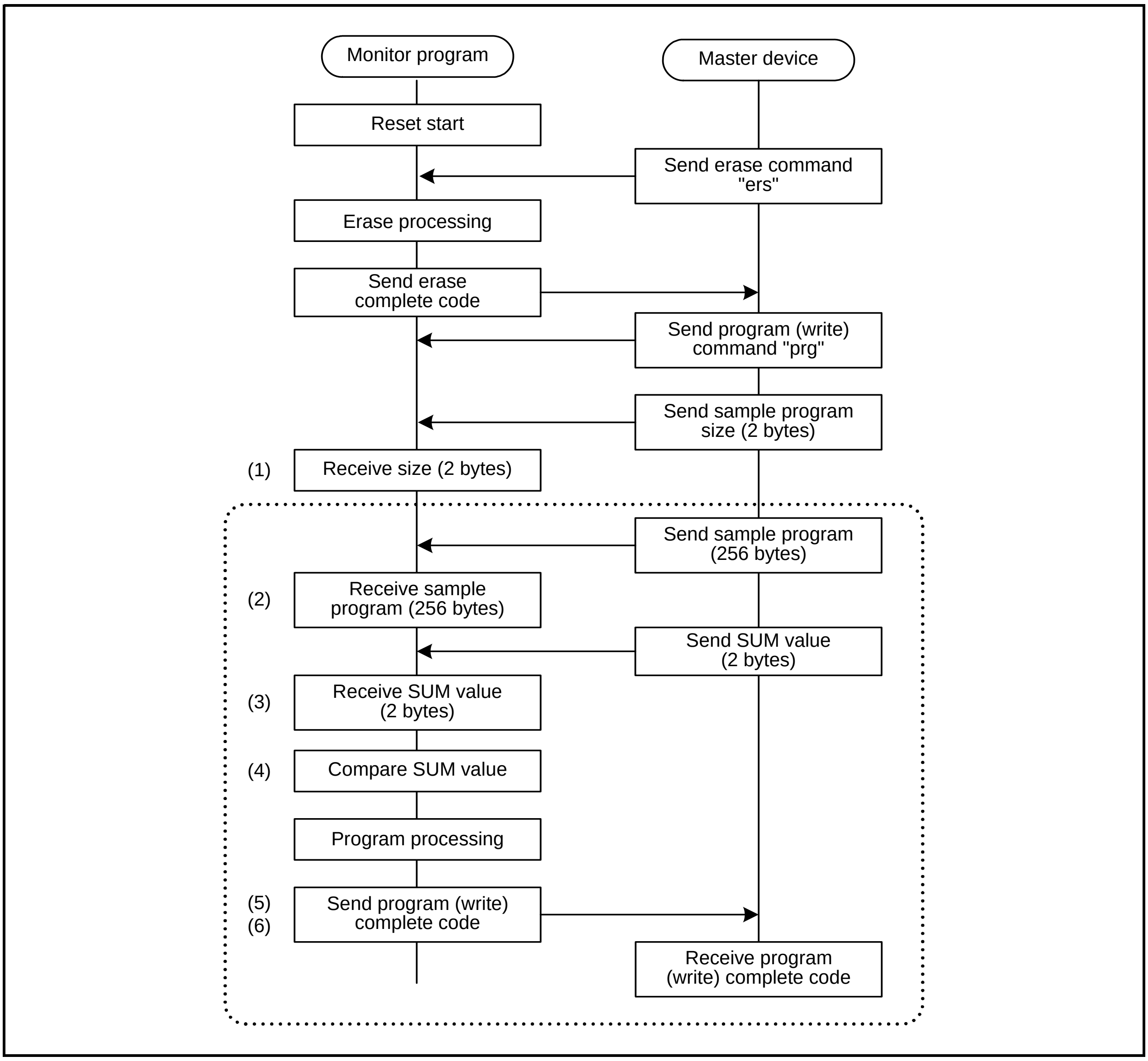
Note:

1. When the program and erase operations are successfully completed, 6FH ('o') is returned. If an error occurs, 65H ('e') is returned.

UART0 clock asynchronous serial I/O mode is used in communication with the master device. The UART0 settings are as follows:

- Mode: Clock asynchronous serial I/O mode
- Communication bit rate: 38400 bps
- CTS/RTS: Not used
- Stop bit: 1 stop bit
- Parity: None
- Data bit length: 8 bits

Figure 4.2 shows a Monitor Program Operation Example.



Monitor program
Master device
Reset start
Send erase command "ers"
Erase processing
Send erase complete code
Send program (write) command "prg"
Send sample program size (2 bytes)
(1)
Receive size (2 bytes)
Send sample program (256 bytes)
(2)
Receive sample program (256 bytes)
Send SUM value (2 bytes)
(3)
Receive SUM value (2 bytes)
(4)
Compare SUM value
Program processing
(5)
(6)
Send program (write) complete code
Receive program (write) complete code

Figure 4.2 Monitor Program Operation Example

The monitor program in this application note is explained below.
Block 12 in the user ROM area is used.

- When the MCU starts up, the monitor program waits to receive the control command.

When the received command is "prg"

(1) Receive the sample size (2-byte data).
(2) Receive one packet (maximum 256 bytes) of program data.
(3) Receive the data SUM value (2-byte data).
(4) Calculate the SUM value for the received one packet data and compare with the received SUM value (2-byte data).
(5) If there is no match, error code is sent to master device.
(6) If the values match, set the CPU clock to 10 MHz or lower so that one packet of data is written to the user ROM area before returning the CPU clock to its original setting.
- When the data has been successfully written, the write complete code is sent to the master device.
- If a write error occurs, an error code is sent to the master device and data reception is stopped.

(7) If an error does not occur, steps (2) through (6) are repeated until receipt of the data is completed.

When the received command is "ers"

(1) Set the CPU clock to 10 MHz or lower and erase the program ROM area before returning the CPU clock to its original setting.
(2) When successfully erased, the erase complete code is sent to the master device.
(3) If an erase error occurs, an error code is sent to the master device.

5. Structure

Declaration	typedef struct buff{
	    unsigned char command[CMD_SIZE];
	    unsigned short size;
	    unsigned char prg_data[RECORD_SIZE];
	    unsigned short rev_sum;
	}REV_BUFF;	

Variable	unsigned char command[CMD_SIZE]	Receive command
	unsigned short size	Receive size
	unsigned char prg_data[RECORD_SIZE]	RECORD_SIZE (256) byte data storage array
	unsigned short rev_sum	SUM value storage variable
Function	Store the received sample program (256 bytes) and the SUM value.	

6. Function Tables

Declaration	void main(void)	
Outline	Main function	
Argument	None	
Variable (global)	Variable name	Content
	REV_BUFF rb	Array for storing received data
		Size data
		Store the SUM value
Returned value	None	
Function	Initialize CPU operating mode and the peripheral functions. Receive data from the master device, and execute the command. Transmit the execution result to the master device.	

Declaration	void mcu_init(void)
Outline	CPU initial setting function
Argument	None
Variable (global)	None
Returned value	None
Function	Select the PLL clock as the CPU clock.

Declaration	void peripheral_init(void)
Outline	Initial setting of peripheral functions
Argument	None
Variable (global)	None
Returned value	None
Function	Set timer A0 to 10 ms, and UART3 transmission/reception.

Declaration	void cpu_slow(void)
Outline	CPU slow down processing function
Argument	None
Variable (global)	None
Returned value	None
Function	Select the CPU clock as main clock.

Declaration	void cpu_fast(void)
Outline	CPU speed up processing function
Argument	None
Variable (global)	None
Returned value	None
Function	Select the CPU clock as the PLL clock.

Declaration	unsigned char rev_byte(unsigned char *rev_data)		
Outline	1-byte command receive function		
Argument	Argument name		Meaning
	unsigned char *rev_data		Address of the array for storing a received command
Variable (global)	None		
Returned value	Type	Value	Meaning
	unsigned char	COMPLETE	Successfully completed
		ERR_URT_TMO	Timeout
		ERR_URT_RCV	Error occurred
Function	Store the received 1-byte data in the array.		

Declaration	unsigned char rev_cmd_check(unsigned char *cmd_buff)		
Outline	Command check function		
Argument	Argument name		Meaning
	unsigned char *cmd_buff		Starting address of the array for storing a received command.
Variable (global)	None		
Returned value	Type	Value	Meaning
	unsigned char	REV_ERASE	Erase command received
		REV_PROGRAM	Program command received
		REV_ERROR	Error occurred
Function	Determine the received character string and return the appropriate command.		

Declaration	unsigned short rev_size(void)	
Outline	Size receive function	
Argument	None	
Variable (global)	None	
Returned value	Type	Meaning
	unsigned short	Received data size
Function	Return the size sent from the master device.	

Declaration	unsigned char rev_data(void)		
Outline	Program data receive function		
Argument	None		
Variable (global)	Variable name		Content
	REV_BUFF rb		Array for storing receive data Size data Store the SUM value
Returned value	Type	Value	Meaning
	unsigned char	COMPLETE	Successfully received
		ERROR	Failed to receive
Function	Receive 256-byte data and the SUM value. Compare the SUM value for the received one packet data and the received SUM value. When the received data is 256 bytes or less, write FFh in the remaining space.		

Declaration	void snd_msg(unsigned char *msg)	
Outline	Message send function	
Argument	Argument name	Meaning
	unsigned char *msg	Starting address of the array for the transmit message
Variable (global)	None	
Returned value	None	
Function	Send a message to the master device.	

Declaration	unsigned char erase(void)		
Outline	Flash memory erase function		
Argument	None		
Variable (global)	None		
Returned value	Type	Value	Meaning
	unsigned char	COMPLETE	Successfully completed
		ERR_CMD_SEQ	Command sequence error
		ERR_ERASE	Erase error
		ERR_PROGRAM	Program write error
Function	Erase the specified block in EW1 mode and execute a full status check.		

Declaration	unsigned char receive_program(void)		
Outline	Flash memory write function		
Argument	None		
Variable (global)	Variable name		Content
	REV_BUFF rb		Array for storing receive data
			Size data
			Store the SUM value
Returned value	Type	Value	Meaning
	unsigned char	COMPLETE	Successfully completed
		ERROR	Write data error
		ERR_ERASE	Erase error
		ERR_PROGRAM	Program write error
		ERR_CMD_SEQ	Command sequence error
Function	Receive the size, data, and SUM value sent from the master device. Write 256 bytes of data from the specified address in EW1 mode. If an error occurred during the write operation, execute the clear status command.		

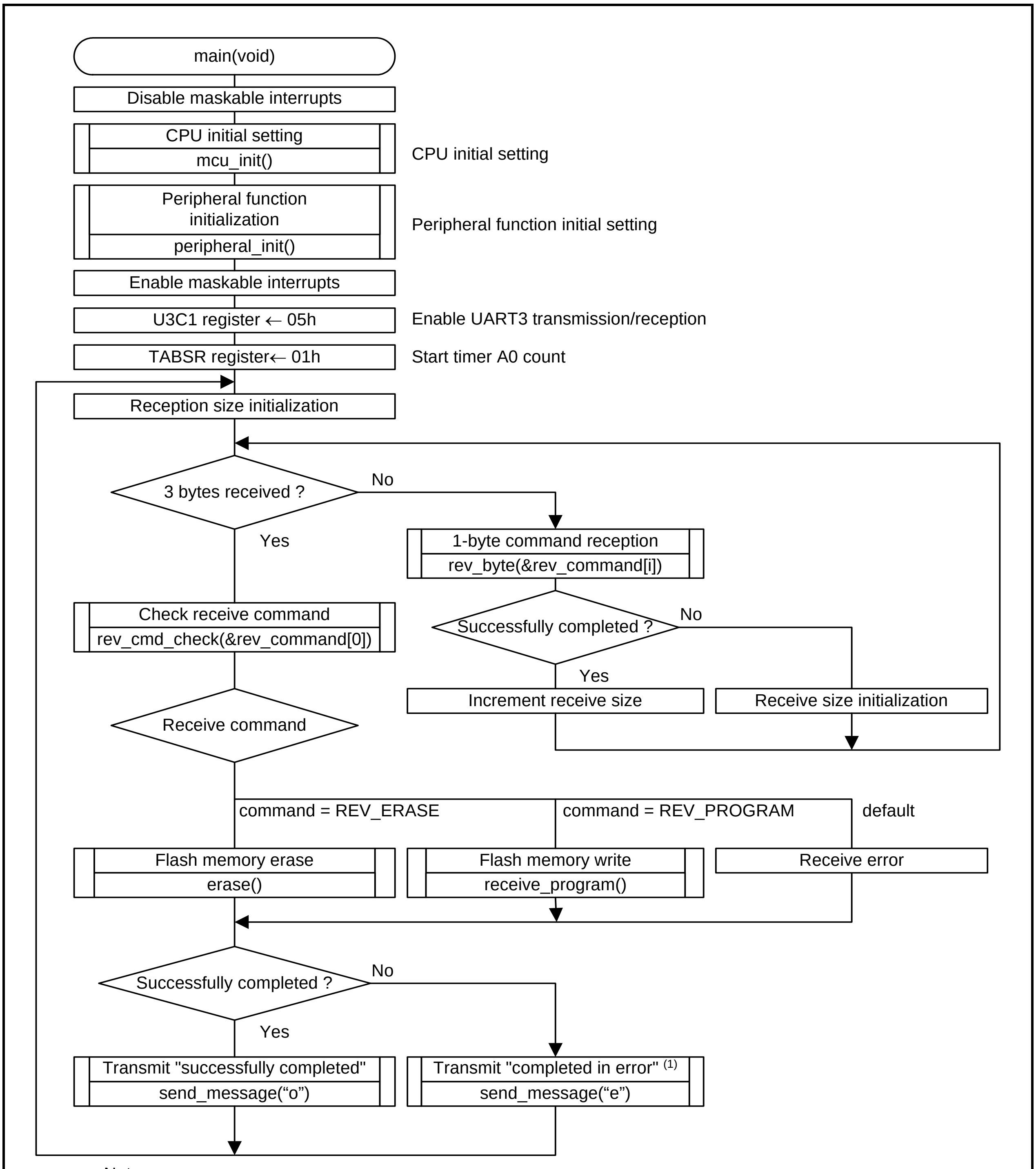
Declaration	unsigned char block_erase_command(unsigned short far* addr)		
Outline	Block erase function		
Argument	Argument name		Meaning
	unsigned short far* addr		Address of block to be erased
Variable (global)	None		
Returned value	Type	Value	Meaning
	unsigned char	COMPLETE	Successfully completed
		ERR_CMD_SEQ	Command sequence error
		ERR_ERASE	Erase error
		ERR_PROGRAM	Program write error
Function	After executing the block erase command to the specified block, execute a full status check.		

Declaration	unsigned char program_command(unsigned short far* addr,unsigned short *buff)		
Outline	Program function		
Argument	Argument name		Meaning
	unsigned short far* addr		Starting address of write destination
	unsigned short *buff		2-byte write data
Variable (global)	None		
Returned value	Type	Value	Meaning
	unsigned char	COMPLETE	Successfully completed
		ERR_CMD_SEQ	Command sequence error
		ERR_ERASE	Erase error
		ERR_PROGRAM	Program write error
Function	After executing the program command to the specified address, execute a full status check.		

Declaration	unsigned char full_status_check(void)		
Outline	Full status check function		
Argument	None		
Variable (global)	None		
Returned value	Type	Value	Meaning
	unsigned char	COMPLETE	Successfully completed
		ERR_CMD_SEQ	Command sequence error
		ERR_ERASE	Erase error
		ERR_PROGRAM	Program write error
Function	Execute a full status check and return the result.		

7. Flowcharts

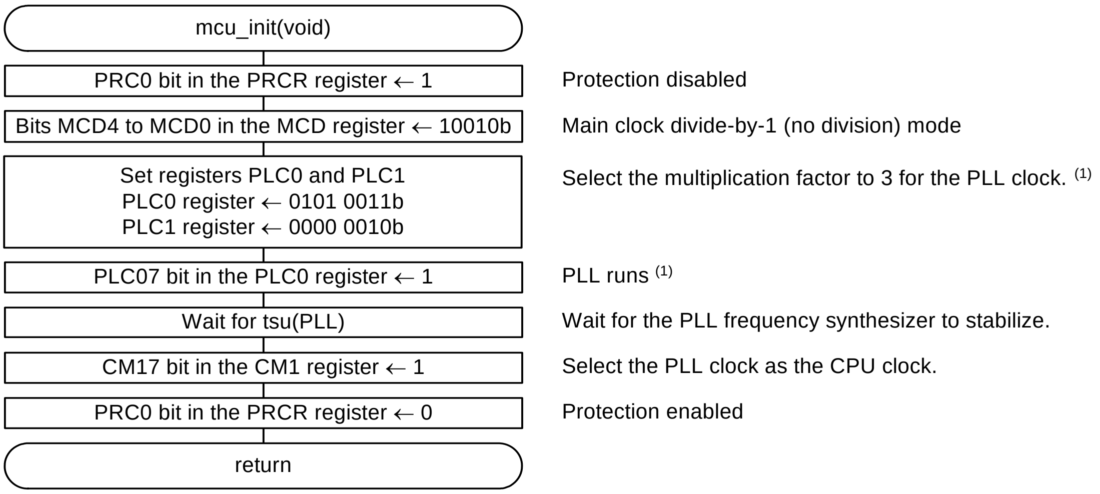
7.1 Main Function



Note:
1. In this application note, an error message is only sent when an error occurs.
Error processing can be added as needed.

Figure 7.1 Main Function

7.2 CPU Initial Setting Function


mcu_init(void)
PRC0 bit in the PRCR register ← 1
Protection disabled
Bits MCD4 to MCD0 in the MCD register ← 10010b
Main clock divide-by-1 (no division) mode
Set registers PLC0 and PLC1
PLC0 register ← 0101 0011b
PLC1 register ← 0000 0010b
Select the multiplication factor to 3 for the PLL clock. (1)
PLC07 bit in the PLC0 register ← 1
PLL runs (1)
Wait for tsu(PLL)
Wait for the PLL frequency synthesizer to stabilize.
CM17 bit in the CM1 register ← 1
Select the PLL clock as the CPU clock.
PRC0 bit in the PRCR register ← 0
Protection enabled
return

Note:
1. Simultaneously set registers PLC0 and PLC1 in 16-bit units.

Figure 7.2 CPU Initial Setting Function

7.3 Peripheral Function Initial Setting Function

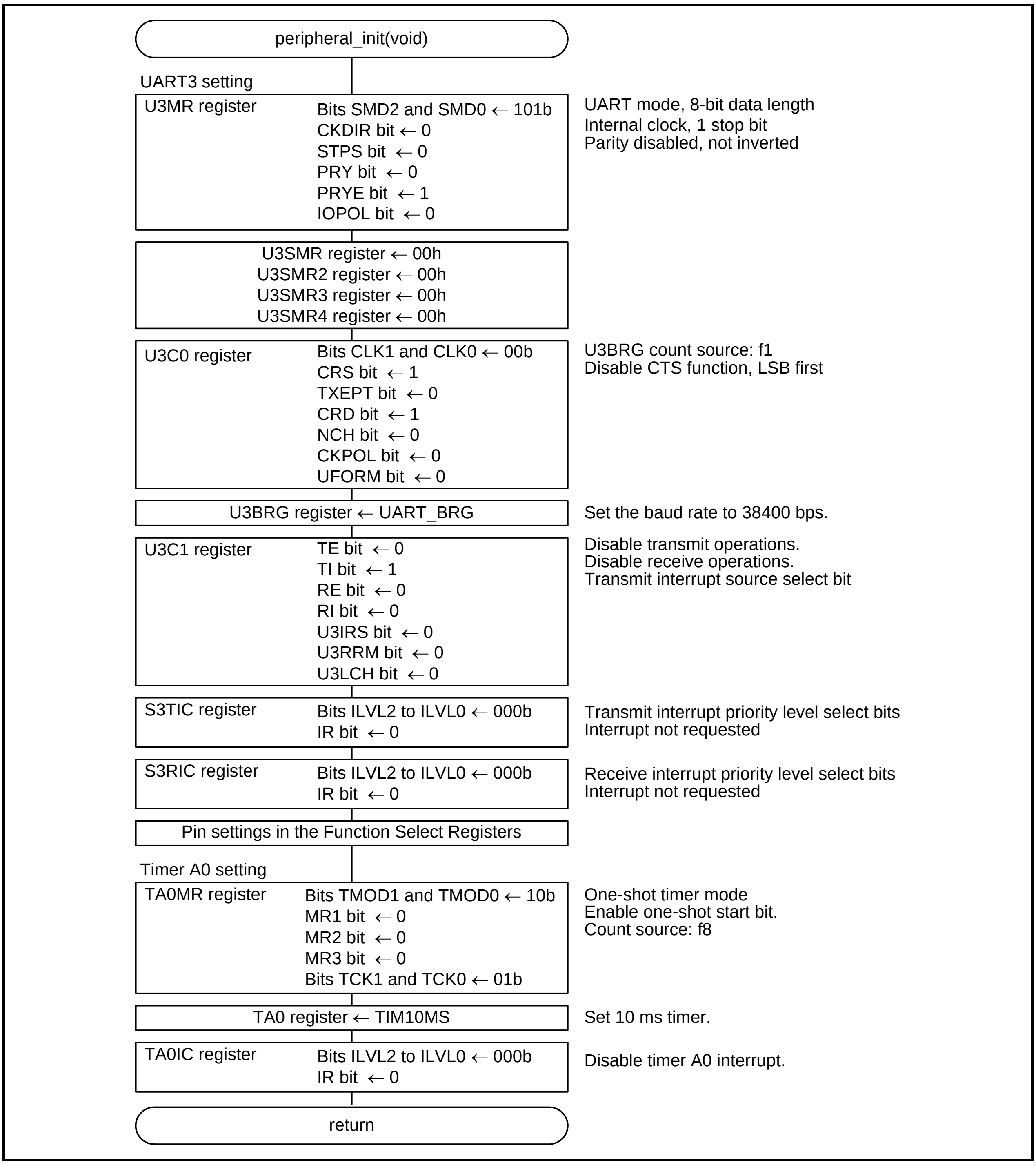
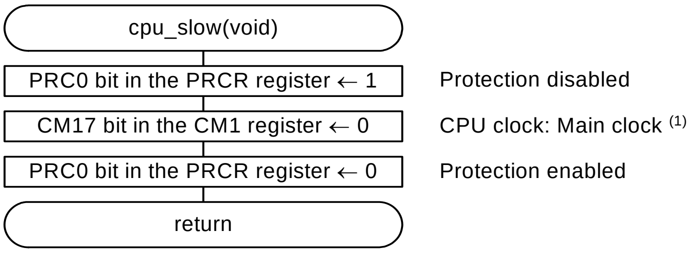


Figure 7.3 Peripheral Function Initial Setting Function

7.4 CPU Slow Down Processing Function


cpu_slow(void)
PRC0 bit in the PRCR register ← 1
CM17 bit in the CM1 register ← 0
PRC0 bit in the PRCR register ← 0
return
Protection disabled
CPU clock: Main clock (1)
Protection enabled

Note:
1. Set the CPU clock to 10 MHz or less.

Figure 7.4 CPU Slow Down Processing Function

7.5 CPU Speed Up Processing Function

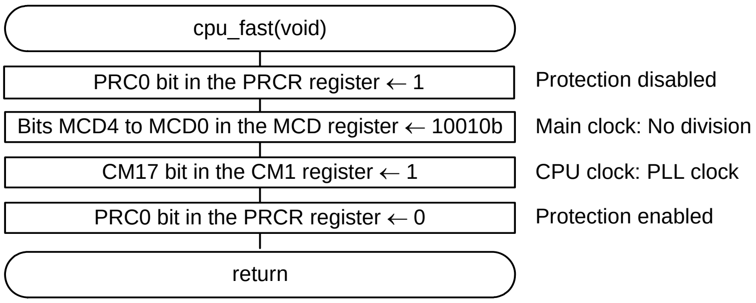
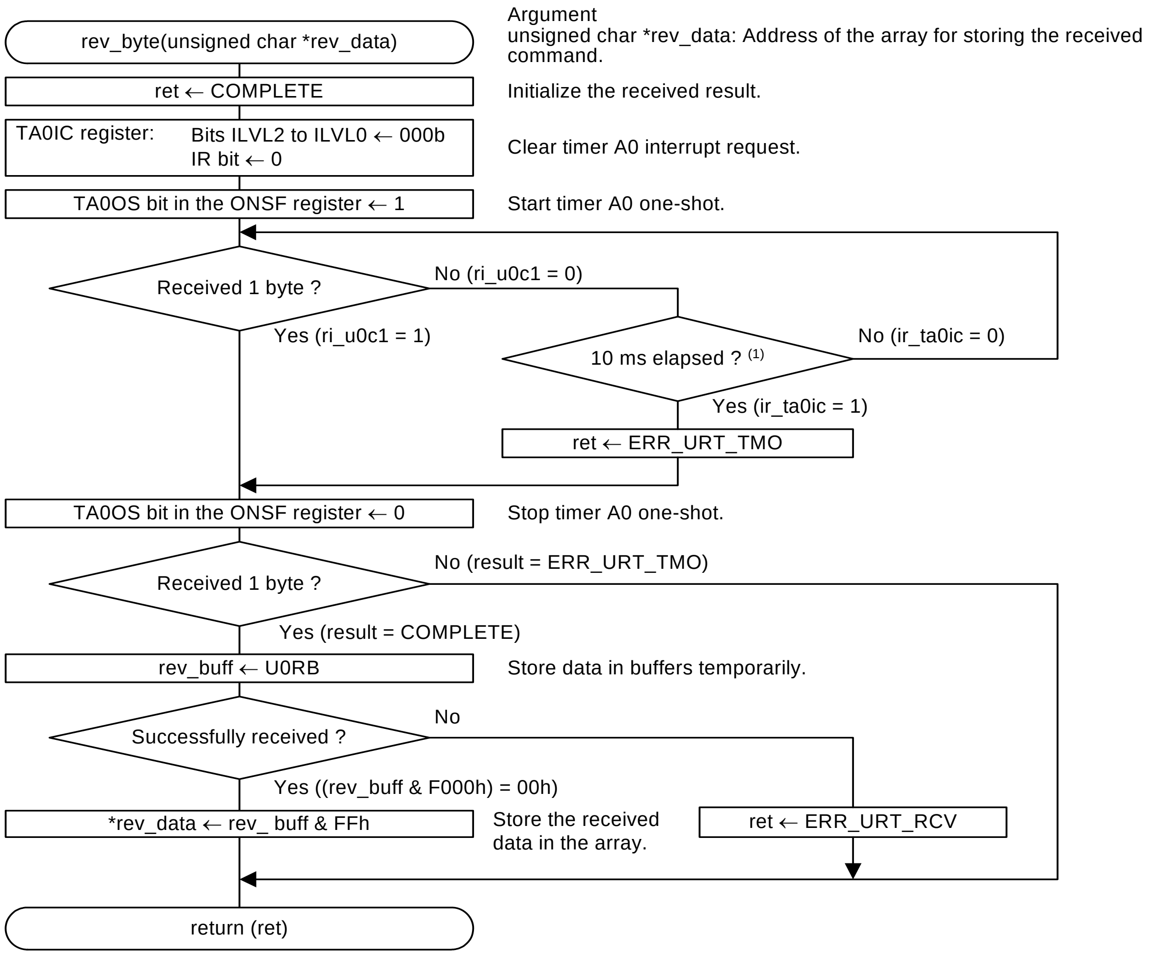

cpu_fast(void)
PRC0 bit in the PRCR register ← 1
Bits MCD4 to MCD0 in the MCD register ← 10010b
CM17 bit in the CM1 register ← 1
PRC0 bit in the PRCR register ← 0
return
Protection disabled
Main clock: No division
CPU clock: PLL clock
Protection enabled

Figure 7.5 CPU Speed Up Processing Function

7.6 1-byte Command Receive Function


Argument
unsigned char *rev_data: Address of the array for storing the received command.
rev_byte(unsigned char *rev_data)
ret ← COMPLETE
Initialize the received result.
TA0IC register: Bits ILVL2 to ILVL0 ← 000b
IR bit ← 0
Clear timer A0 interrupt request.
TA0OS bit in the ONSF register ← 1
Start timer A0 one-shot.
Received 1 byte ?
No (ri_u0c1 = 0)
Yes (ri_u0c1 = 1)
10 ms elapsed ? (1)
No (ir_ta0ic = 0)
Yes (ir_ta0ic = 1)
ret ← ERR_URT_TMO
TA0OS bit in the ONSF register ← 0
Stop timer A0 one-shot.
Received 1 byte ?
No (result = ERR_URT_TMO)
Yes (result = COMPLETE)
rev_buff ← U0RB
Store data in buffers temporarily.
Successfully received ?
No
Yes ((rev_buff & F000h) = 00h)
*rev_data ← rev_ buff & FFh
Store the received data in the array.
ret ← ERR_URT_RCV
return (ret)

Note:

1. Timeout processing is performed when a command is received in this function, while it is not performed in other functions. Timeout processing can be added to other functions when necessary.

Figure 7.6 1-byte Command Receive Function

7.7 Command Check Function

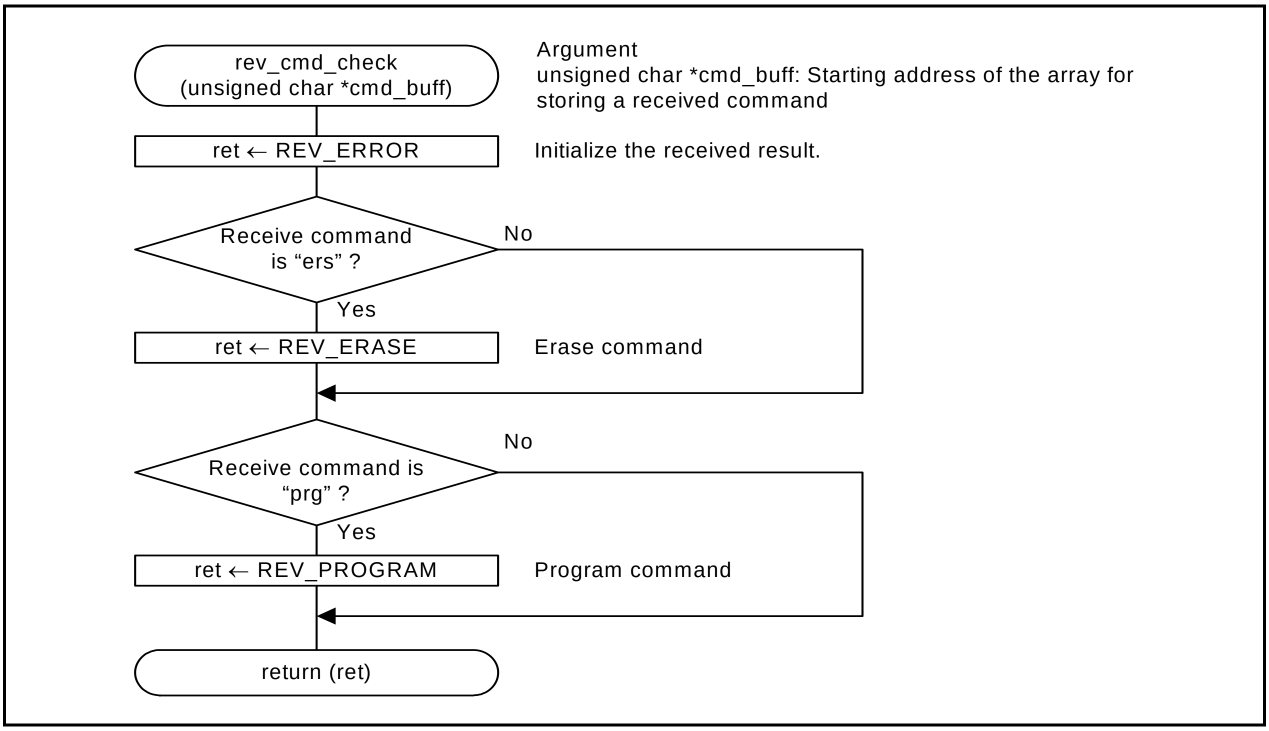

rev_cmd_check
(unsigned char *cmd_buff)
Argument
unsigned char *cmd_buff: Starting address of the array for storing a received command
ret ← REV_ERROR
Initialize the received result.
Receive command is "ers" ?
No
Yes
ret ← REV_ERASE
Erase command
Receive command is "prg" ?
No
Yes
ret ← REV_PROGRAM
Program command
return (ret)

Figure 7.7 Command Check Function

7.8 Size Receive Function

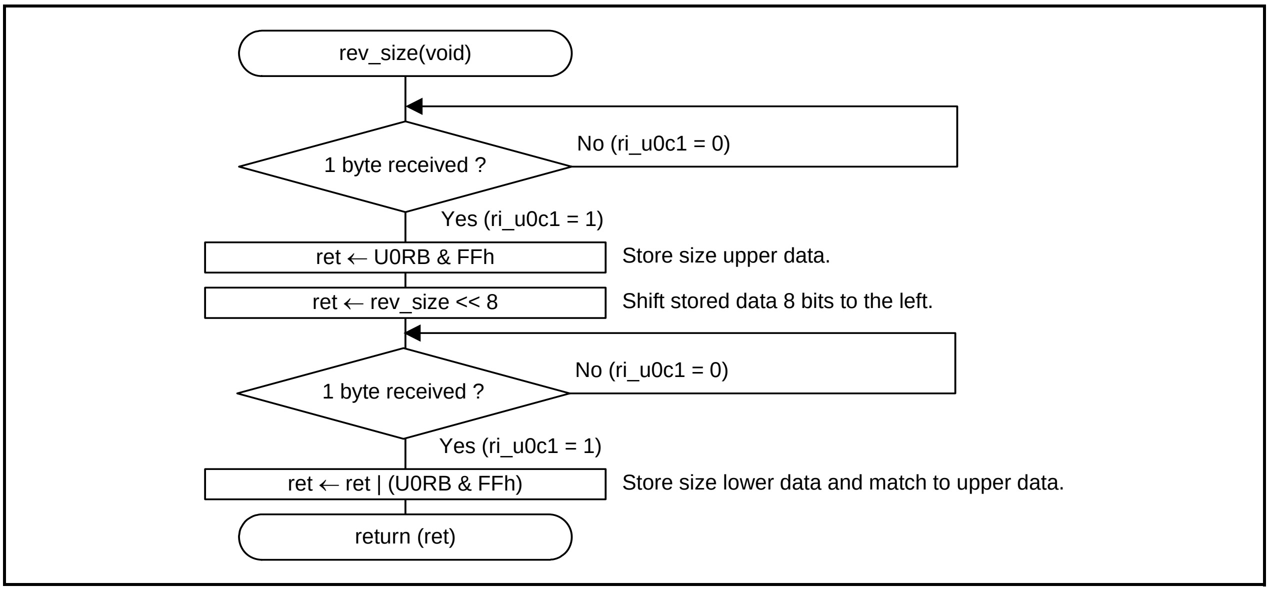
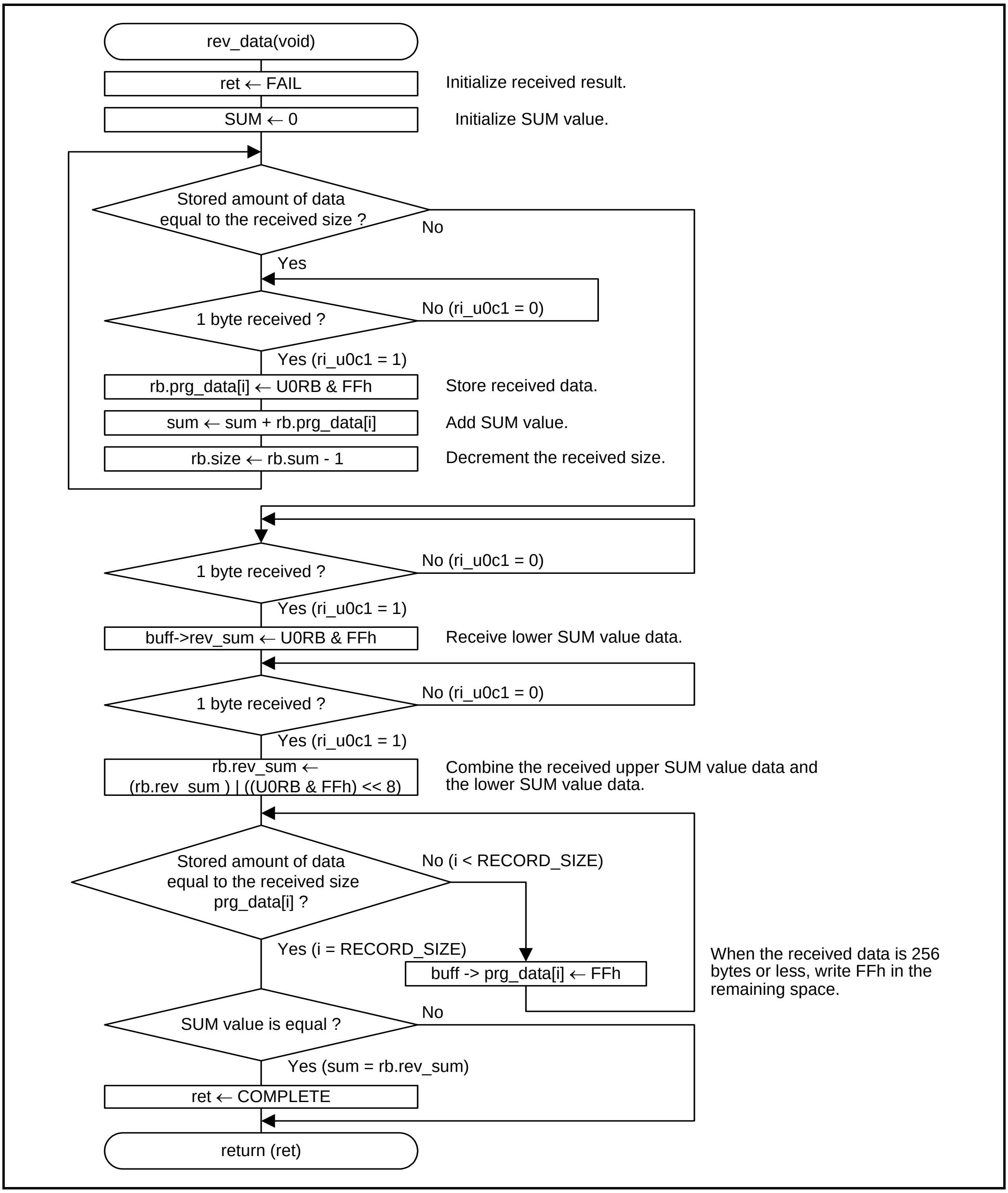


Figure 7.8 Size Receive Function

7.9 Program Data Receive Function



Initialize received result.

Initialize SUM value.

Store received data.

Add SUM value.

Decrement the received size.

Receive lower SUM value data.

Combine the received upper SUM value data and the lower SUM value data.

When the received data is 256 bytes or less, write FFh in the remaining space.

Figure 7.9 Program Data Receive Function

7.10 Message Transmit Function

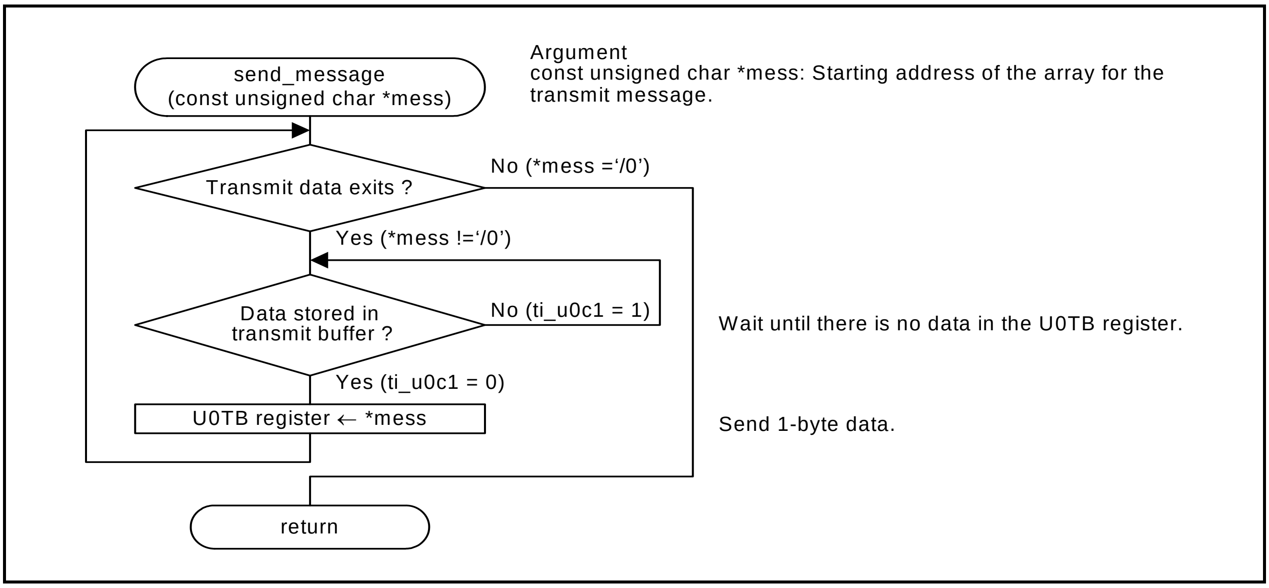
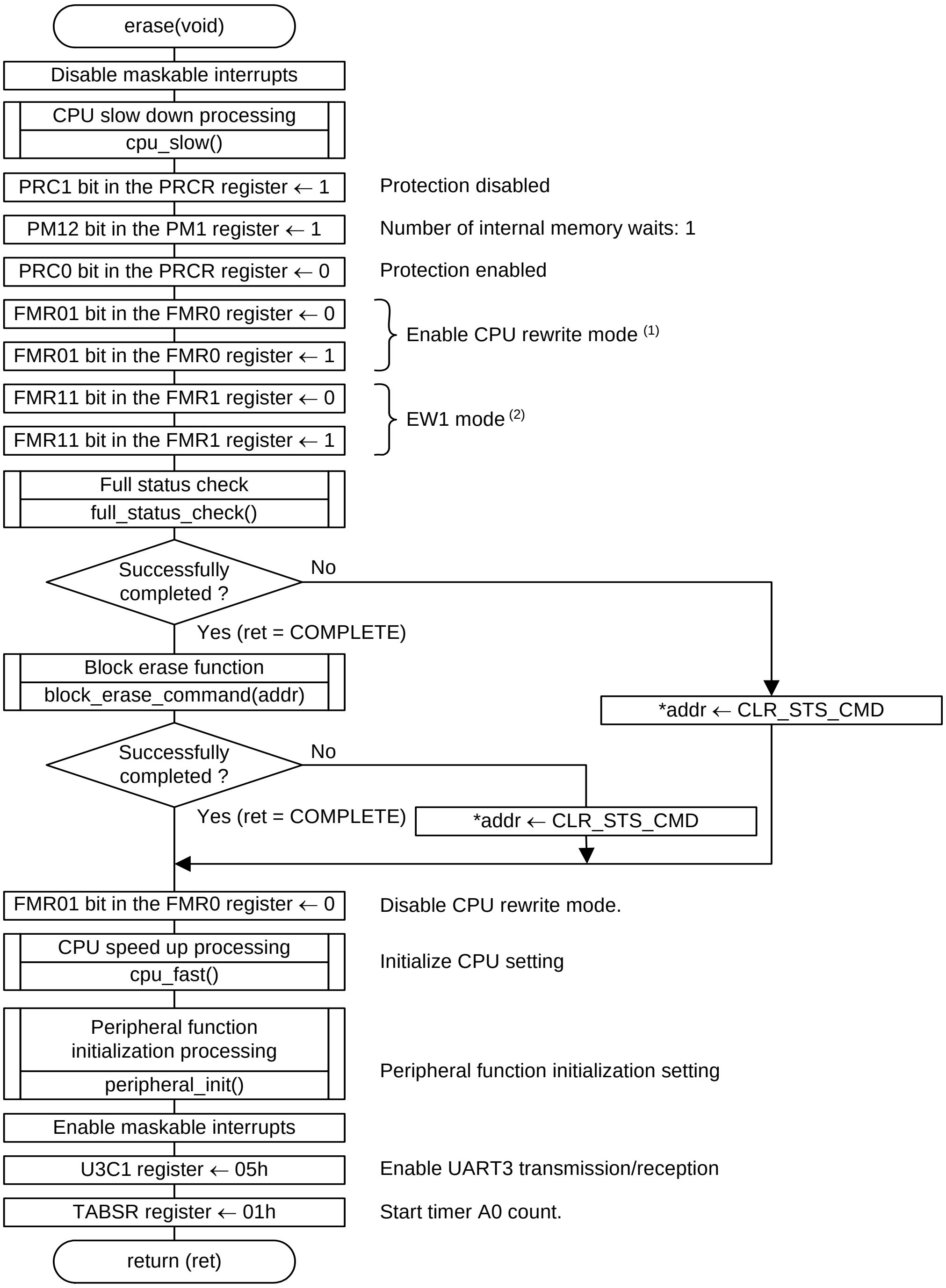

send_message
(const unsigned char *mess)
Argument
const unsigned char *mess: Starting address of the array for the transmit message.
Transmit data exits ?
No (*mess ='/0')
Yes (*mess !='/0')
Data stored in transmit buffer ?
No (ti_u0c1 = 1)
Wait until there is no data in the U0TB register.
Yes (ti_u0c1 = 0)
U0TB register ← *mess
Send 1-byte data.
return

Figure 7.10 Message Transmit Function

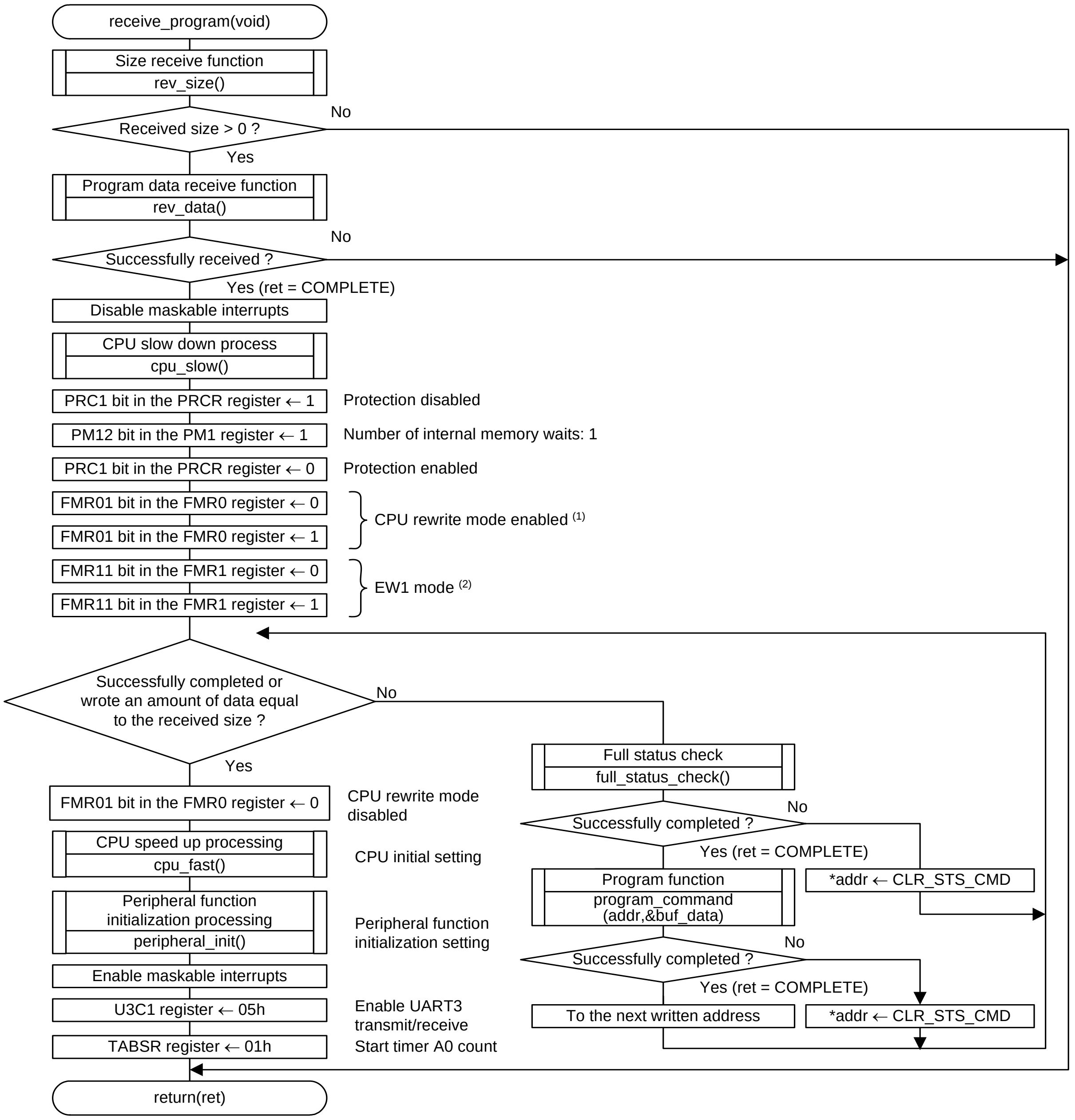
7.11 Flash Memory Erase Function


erase(void)
Disable maskable interrupts
CPU slow down processing
cpu_slow()
PRC1 bit in the PRCR register ← 1
Protection disabled
PM12 bit in the PM1 register ← 1
Number of internal memory waits: 1
PRC0 bit in the PRCR register ← 0
Protection enabled
FMR01 bit in the FMR0 register ← 0
FMR01 bit in the FMR0 register ← 1
Enable CPU rewrite mode (1)
FMR11 bit in the FMR1 register ← 0
FMR11 bit in the FMR1 register ← 1
EW1 mode (2)
Full status check
full_status_check()
Successfully completed ?
No
Yes (ret = COMPLETE)
Block erase function
block_erase_command(addr)
*addr ← CLR_STS_CMD
Successfully completed ?
No
Yes (ret = COMPLETE)
*addr ← CLR_STS_CMD
FMR01 bit in the FMR0 register ← 0
Disable CPU rewrite mode.
CPU speed up processing
cpu_fast()
Initialize CPU setting
Peripheral function initialization processing
peripheral_init()
Peripheral function initialization setting
Enable maskable interrupts
U3C1 register ← 05h
Enable UART3 transmission/reception
TABSR register ← 01h
Start timer A0 count.
return (ret)

Notes:
1. To set the FMR01 bit to 1, write 1 to the FMR01 bit in the FMR0 register immediately after writing 0 to the bit. When setting the FMR01 bit, write to the FMR0 register in 8-bit units. Do not generate an interrupt or a DMA or DMACII transfer between these two settings.
2. To set the FMR11 bit in the FMR1 register to 1, write 1 to the FMR11 bit immediately after writing 0 to the bit while the FMR01 bit in the FMR0 register is set to 1.
When setting the FMR11 bit, write to the FMR1 register in 8-bit units. Do not generate an interrupt or a DMA or DMACII transfer between these two settings.
Set the FMR11 bit while the $\overline{\text{NMI}}$ pin level is held "H".
When setting the FMR01 bit to 0, the FMR11 bit becomes 0.

Figure 7.11 Flash Memory Erase Function

7.12 Flash Memory Write Function


receive_program(void)
Size receive function
rev_size()
Received size > 0 ?
No
Yes
Program data receive function
rev_data()
Successfully received ?
No
Yes (ret = COMPLETE)
Disable maskable interrupts
CPU slow down process
cpu_slow()
PRC1 bit in the PRCR register ← 1
Protection disabled
PM12 bit in the PM1 register ← 1
Number of internal memory waits: 1
PRC1 bit in the PRCR register ← 0
Protection enabled
FMR01 bit in the FMR0 register ← 0
FMR01 bit in the FMR0 register ← 1
CPU rewrite mode enabled (1)
FMR11 bit in the FMR1 register ← 0
FMR11 bit in the FMR1 register ← 1
EW1 mode (2)
Successfully completed or wrote an amount of data equal to the received size ?
No
Yes
FMR01 bit in the FMR0 register ← 0
CPU rewrite mode disabled
CPU speed up processing
cpu_fast()
CPU initial setting
Peripheral function initialization processing
peripheral_init()
Peripheral function initialization setting
Enable maskable interrupts
U3C1 register ← 05h
Enable UART3 transmit/receive
TABSR register ← 01h
Start timer A0 count
return(ret)
Full status check
full_status_check()
Successfully completed ?
No
Yes (ret = COMPLETE)
Program function
program_command (addr,&buf_data)
*addr ← CLR_STS_CMD
Successfully completed ?
No
Yes (ret = COMPLETE)
To the next written address
*addr ← CLR_STS_CMD

Notes:

1. To set the FMR01 bit to 1, write 1 to the FMR01 bit in the FMR0 register immediately after writing 0 to the bit. When setting the FMR01 bit, write to the FMR0 register in 8-bit units. Do not generate an interrupt or a DMA or DMACII transfer between these two settings.
2. To set the FMR11 bit in the FMR1 register to 1, write 1 to the FMR11 bit immediately after writing 0 to the bit while the FMR01 bit in the FMR0 register is set to 1.
   When setting the FMR11 bit, write to the FMR1 register in 8-bit units. Do not generate an interrupt or a DMA or DMACII transfer between these two settings.
   Set the FMR11 bit while the NMI pin level is held "H".
   When setting the FMR01 bit to 0, the FMR11 bit becomes 0.

Figure 7.12 Flash Memory Write Function

7.13 Block Erase Function

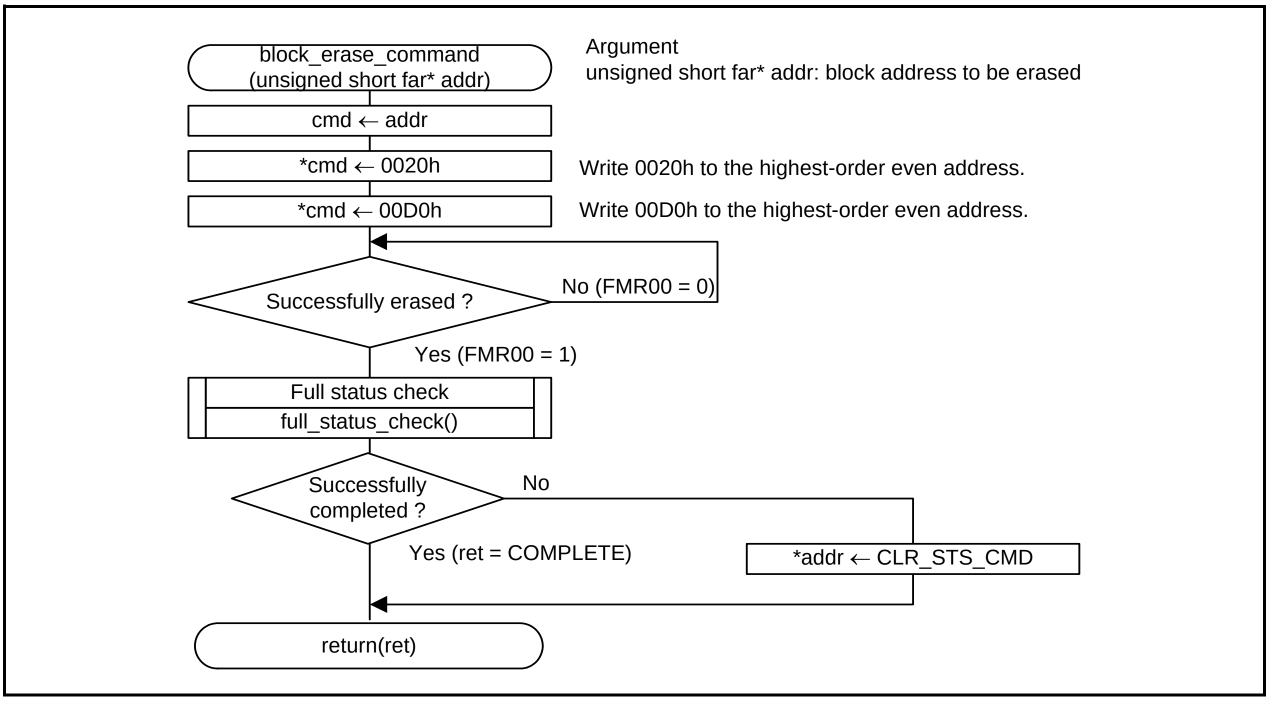


Figure 7.13 Block Erase Function

7.14 Program Function

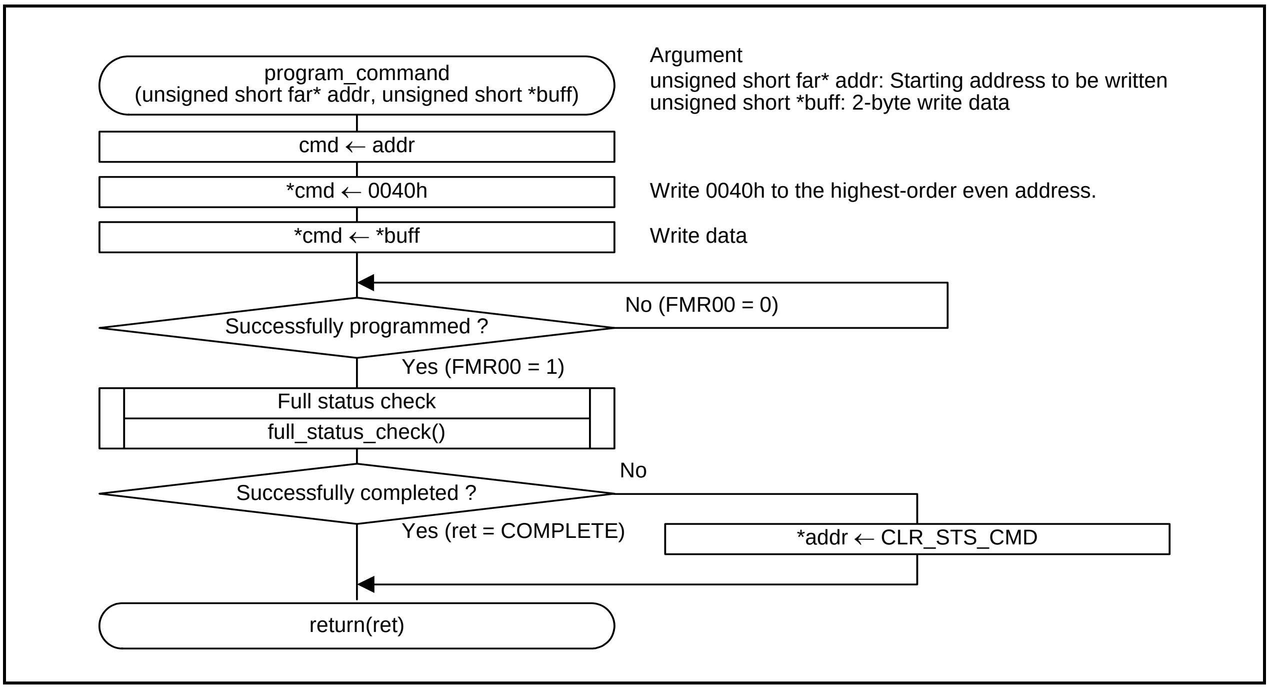

program_command
(unsigned short far* addr, unsigned short *buff)
Argument
unsigned short far* addr: Starting address to be written
unsigned short *buff: 2-byte write data
cmd ← addr
*cmd ← 0040h
Write 0040h to the highest-order even address.
*cmd ← *buff
Write data
Successfully programmed ?
No (FMR00 = 0)
Yes (FMR00 = 1)
Full status check
full_status_check()
Successfully completed ?
No
Yes (ret = COMPLETE)
*addr ← CLR_STS_CMD
return(ret)

Figure 7.14 Program Function

7.15 Full Status Check Function

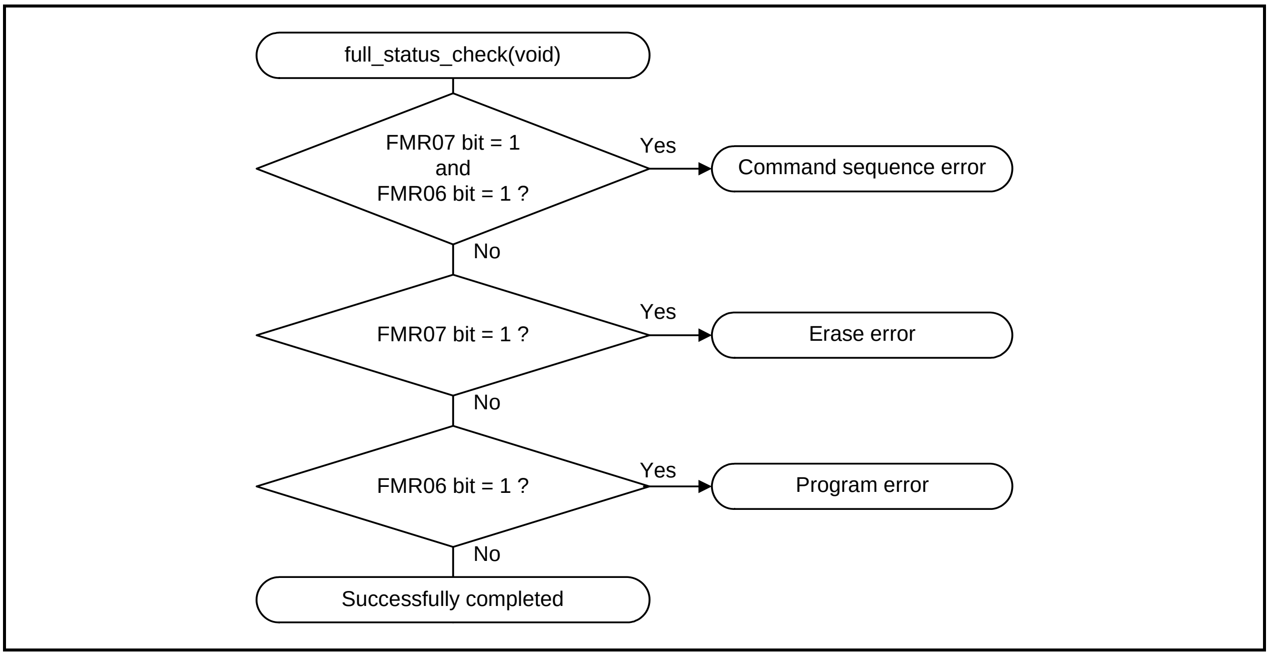


Figure 7.15 Full Status Check Function

8. Sample Program

A sample program can be downloaded from the Renesas Electronics website.

9. Reference Documents

User’s Manuals
R32C/84 Group (M32C/84, M32C/84T) User’s Manual: Hardware Rev.1.01
R32C/85 Group (M32C/85, M32C/85T) User’s Manual: Hardware Rev.1.03
R32C/87 Group (M32C/87, M32C/87A, M32C/87B) User’s Manual: Hardware Rev.1.51
R32C/88 Group (M32C/88T) User’s Manual: Hardware Rev.1.10
The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

C Compiler Manual
M32C/100 Series C Compiler Package V.5.42 Release 00 C Compiler User’s Manual Rev.2.00
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

REVISION HISTORY	M32C/84, 85, 87, 88 Groups Example of Rewriting the User ROM Area Using EW1 Mode

Rev.	Date	Description	
		Page	Summary
1.00	Nov. 15, 2010	-	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to one with a different part number, confirm that the change will not lead to problems.

   — The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or system 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0