

## RX630 Group

### Communication with Flash Memory Using the Serial Communications Interface (Simple SPI)

R01AN0595EJ0100  
Rev.1.00  
Dec 13, 2011

#### Introduction

This application note describes communication with a connected serial flash memory using the simple SPI function of the RX630 Group SCI (serial communications interface) module.

#### Target Device

RX630 Group

This application note can also be used with other RX family microcontrollers that have the same I/O registers (peripheral unit control registers) as the RX630 Group products. Note, however, that since there are changes between devices, such as additional functionality in certain functions, these operations must be verified with the manual for the device actually used. When using the methods described in this application note, full testing in the actual user system is required.

#### Contents

|   |    |
|---|----|
| 1. Specifications .....                                       | 2  |
| 2. Conditions of Checking the Operation of the Software ..... | 3  |
| 3. Software Description .....                                 | 4  |
| 4. Sample Programs.....                                       | 28 |
| 5. Reference Documents.....                                   | 28 |

## 1. Specifications

This sample program communicates with a serial flash memory and writes 8 bytes of data to that serial flash memory. It then reads back that 8 bytes of data and compares the write data with the data read back to confirm that the data was written correctly.

Table 1.1 lists the peripheral functions used and their uses.

**Table 1.1 Peripheral Functions and their Uses**

| <b>Peripheral Function</b>            | <b>Use</b>  |
|---------------------------------------|---|
| SCI (serial communications interface) | <ul style="list-style-type: none"><li>• Simple SPI function</li><li>• Clock frequency: 6.25 MHz</li><li>• Erase size:</li><li>• Program size:</li><li>• Protection units:</li></ul> |
| General-purpose I/O ports             | SS signal output  |

## 2. Conditions of Checking the Operation of the Software

The sample code described in this application note has been confirmed to run normally under the operating conditions given below.

**Table 2.1** Operating Conditions

| <b>Item</b>                        | <b>Description</b>  |
|------------------------------------|---|
| MCU                                | RX630 (R5F5630EDDFP)  |
| Memory used for evaluation         | Xin clock: 12 MHz<br>Subclock: 32.768 kHz   |
| Operating voltage                  | 3.3 V   |
| Integrated development environment | Version 4.09.00.007   |
| C compiler                         | RX Standard Toolchain (V.1.0.1.0)<br><br>C/C++ compiler package for RX family V.1.01.00<br>Compiler options:<br>The default settings for the integrated development environment are used. |
| Operating mode                     | Single chip mode  |
| Version of the sample code         | Version 1.40  |
| Board used                         | R0K505630C001BR   |

### 3. Software Description

#### 3.1 Operation Overview

The following pins are used in SCI simple SPI mode: SMOSIn (master out slave in), SMISOn (master in slave out), SSn# (slave select), and SCKn (SPI clock). Since the transmit block and receive block are independent within the SCI module, full-duplex transmission can be implemented by sharing the clock signal. Furthermore, the transmit and receive block both have a double buffered structure and thus the next transmit data can be written during transmission and the receive data can be read during reception. This makes it possible to perform consecutive transmit and receive operations.

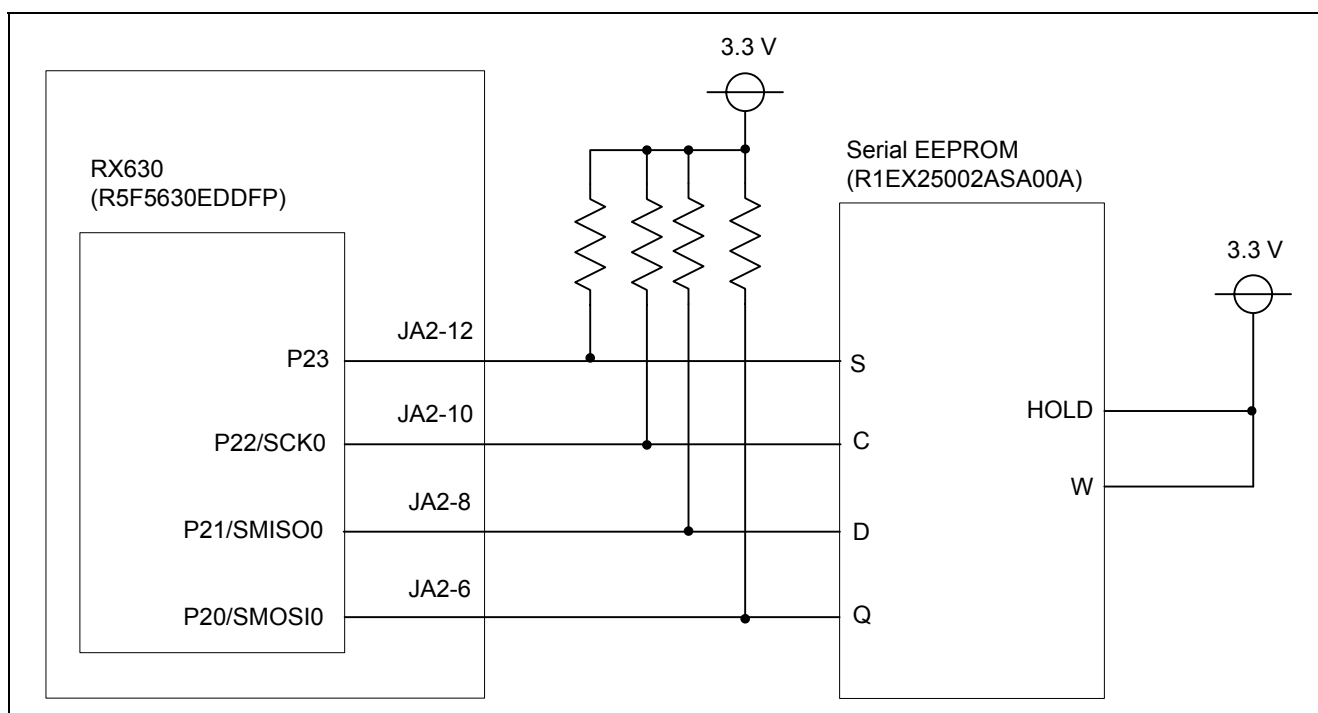
As in clock synchronous mode, in simple SPI mode data is transmitted or received in synchronization with a clock signal. Each character transmitted consists of 8 bits of data, but no parity bit may be added. The transmit data or receive data can be inverted by setting the SCMR.SINV bit to 1.

Although both the SCI c and SCI d SCI modules support simple SPI mode, this application note uses channel 0 in SCI c.

#### 3.1.1 Specifications

The sample code in this application note controls a Renesas Electronics Corporation R1EX25xxx Series SPI serial EEPROM.

Figure 3.1 shows the connections used for this application note.



**Figure 3.1 Connection Diagram**

Table 3.1 lists the SCI communication function settings and conditions used in this application note.

**Table 3.1 SCI Settings and Conditions**

|                    |                          |
|--------------------|--------------------------|
| Channels used      | SCI0                     |
| Communication mode | Simple SPI mode          |
| Interrupts         | Unused                   |
| Transfer rate      | 6.25 MHz (PCLK = 50 MHz) |
| Data length        | 8 bits of data           |

### 3.2 File Structure

Table 3.2 lists the files used in the sample code.

**Table 3.2 File Structure**

| File Name    | Function   | Notes   |
|--------------|--|---|
| main_spi.c   | <ul style="list-style-type: none"> <li>The main processing</li> <li>Interrupt grouping control related processing</li> <li>SCI interrupt processing</li> </ul> |   |
| dbstc.c      | B and R section settings   | File automatically generated by the IDE   |
| intprg.c     | Interrupt handling<br>(The SCI interrupt handler, which is used by this program, has been removed from this file.)   | File automatically generated by the IDE   |
| resetprg.c   | Reset handling   | File automatically generated by the IDE   |
| sbrk.c       | sbrk() function  | File automatically generated by the IDE   |
| vecttbl.c    | Vector table related processing  | File automatically generated by the IDE to which option and memory settings have been added |
| iodefine.h   | I/O register related header file   |   |
| lowsrc.h     | I/O streams related header file  | File automatically generated by the IDE   |
| sbrk.h       | sbrk() function header file  | File automatically generated by the IDE   |
| stacksct.h   | Stack area header file   | File automatically generated by the IDE   |
| typedefine.h | Integer type definitions header file   | File automatically generated by the IDE   |
| vect.h       | Vector table related header file   | File automatically generated by the IDE   |

### 3.3 Constants

Table 3.3 lists the constants used by the sample code.

**Table 3.3 Constants Used in the Sample Code**

| Constant                | Set Value | Description                               |
|-------------------------|-----------|---|
| TOP_ADDRESS             | 0         | Serial flash memory start address         |
| SF_PAGE_SIZE            | 256       | Serial flash memory page size             |
| SF_SECTOR_SIZE          | 0x8000    | Sector size = 32 KB                       |
| SF_NUM_OF_SECTOR        | 32        | Number of sectors: 32                     |
| SFLASHCMD_CHIP_ERASE    | 0xc7      | Serial flash command: Chip erase          |
| SFLASHCMD_SECTOR_ERASE  | 0xd8      | Serial flash command: Sector erase        |
| SFLASHCMD_BYTE_PROGRAM  | 0x02      | Serial flash command: Data write          |
| SFLASHCMD_BYTE_READ     | 0x0B      | Serial flash command: Data read           |
| SFLASHCMD_BYTE_READ_LOW | 0x03      | Serial flash command: Data read           |
| SFLASHCMD_WRITE_ENABLE  | 0x06      | Serial flash command: Write enable        |
| SFLASHCMD_WRITE_DISABLE | 0x04      | Serial flash command: Write disable       |
| SFLASHCMD_READ_STATUS   | 0x05      | Serial flash command: Status read         |
| SFLASHCMD_WRITE_STATUS  | 0x01      | Serial flash command: Status write        |
| UNPROTECT_WR_STATUS     | 0x00      | Serial flash command: Clear protect state |
| PROTECT_WR_STATUS       | 0x3C      | Serial flash command: Protect             |

### 3.4 Variables

Table 3.4 lists the global variables.

**Table 3.4 Global Variables**

| Type          | Name                  | Description                        | Functions Where Used    |
|---------------|-----------------------|------------------------------------|-------------------------|
| unsigned char | data[SF_SECTOR_SIZE]; | Array that holds the transmit data | main, io_cmd_exe        |
| unsigned char | rbuf[SF_SECTOR_SIZE]; | Array that holds the receive data  | main, io_cmd_exe_rdmode |

### 3.5 Functions

Table 3.5 lists the functions defined in the sample code.

**Table 3.5 Functions**

| <b>Function Name</b> | <b>Overview</b>  |
|----------------------|--|
| mcu_init             | CPU initialization   |
| clock_setting        | CPU clock settings   |
| peripheral_init      | Peripheral function initialization                           |
| SCI_init             | Serial communications interface (SCI) related initialization |
| sf_protect_ctrl      | Manipulates the protection state                             |
| sf_chip_erase        | Chip erase   |
| sf_byte_program      | Data write   |
| sf_byte_read         | Data read  |
| write_enable         | Write enable   |
| write_disable        | Write disable  |
| busy_wait            | Wait while busy  |
| read_status          | Status read  |
| write_status         | Status write   |
| io_cmd_exe           | Execute command (no data read)                               |
| io_cmd_exe_rdmode    | Execute command (with data read)                             |
| io_wait_tx_end       | Wait for transmit complete                                   |
| eep_cs_init          | CS signal initialization                                     |
| eep_cs_start         | Starts the CS signal (low output)                            |
| eep_cs_end           | Ends the CS signal (high output)                             |

### 3.6 Function Specifications

This section lists the specifications of the functions in the sample code.

|               |                     |
|---------------|---------------------|
| Name          | mcu_init            |
| Overview      | CPU initialization  |
| Header        | lodefine.h          |
| Declaration   | void mcu_init(void) |
| Description   | Sets the CPU clock. |
| Arguments     | None                |
| Return values | None                |
| Notes         |                     |

|               |   |
|---------------|---|
| Name          | clock_setting   |
| Overview      | Sets the CPU clock.   |
| Header        | lodefine.h  |
| Declaration   | void clock_setting(void)  |
| Description   | <ul style="list-style-type: none"> <li>• Stops the subclock oscillator.</li> <li>• Stops the high-speed clock oscillator.</li> <li>• Sets the oscillator stabilization time for the main clock oscillator to 131,072 cycles.</li> <li>• Sets the PLL oscillator stabilization time to 4,194,304 cycles.</li> <li>• Sets the PLL frequency multiplier to 16x.</li> <li>• Sets the main clock oscillator to the operating state.</li> <li>• Sets the PLL circuit to the operating state.</li> <li>• Sets the system clock to divided by 2, the FlashIF clock to divided by 4, the external bus clock to divided by 4, and the peripheral module clock to divided by 4.</li> <li>• Sets the clock source to be the PLL circuit.</li> </ul> |
| Arguments     | None  |
| Return values | None  |
| Notes         |   |

|               |                                    |
|---------------|------------------------------------|
| Name          | peripheral_init                    |
| Overview      | Peripheral function initialization |
| Header        | lodefine.h                         |
| Declaration   | void peripheral_init(void)         |
| Description   | Initializes SCI related items.     |
| Arguments     | None                               |
| Return values | None                               |
| Notes         |                                    |



## RX630 Group Communication with Flash Memory Using the Serial Communications Interface (Simple SPI)

|               |  |
|---------------|--|
| Name          | SCI_init   |
| Overview      | Serial communications interface (SCI) related initialization   |
| Header        | lodefine.h   |
| Declaration   | void SCI_init(void)  |
| Description   | <ul style="list-style-type: none"><li>• Clears the SCI0 module stop function.</li><li>• Sets pins P21 (RxD), P20 (TxD), P22 (SCK), and P23 (CTS#) to be used as SCI0 pins.</li><li>• Sets SCI0 to serial reception operation (RE) and serial transmission operation (TE) stopped.</li><li>• Sets SCI0 to use the PCLK clock and to clock synchronous mode.</li><li>• Sets SCI0 to interface mode.</li><li>• Sets SCI0 SPI mode to SS pin enabled, no clock phase, and no clock delay.</li><li>• Sets the SCI0 smart card mode select to serial communications interface mode.</li><li>• Sets the SCI0 baud rate to 6.25 MHz.</li><li>• Sets the SCI0 interrupt level to level 1.</li><li>• Clears the SCI0 interrupt request.</li><li>• Sets the SCI0 interrupt request enable to interrupts disabled.</li></ul> |
| Arguments     | None   |
| Return values | None   |
| Notes         |  |

|               |   |
|---------------|---|
| Name          | sf_protect_ctrl   |
| Overview      | Manipulates the protection state  |
| Header        | None  |
| Declaration   | void sf_protect_ctrl(enum sf_req req)   |
| Description   | <ul style="list-style-type: none"><li>• Sets or clears serial flash memory protection.</li><li>• The setting desired is specified with the req argument. The initial protection state and the method for clearing protection depends on the specifications of the serial flash memory used.</li></ul> |
| Arguments     | sf_req req: SF_REQ_UNPROTECT → Clears the all sectors protected state<br>SF_REQ_PROTECT → Protects all sectors  |
| Return values | None  |
| Notes         |   |

|               |  |
|---------------|--|
| Name          | sf_chip_erase  |
| Overview      | Chip erase   |
| Header        | None   |
| Declaration   | void sf_chip_erase(void)   |
| Description   | <ul style="list-style-type: none"><li>• Erases all bits in the serial flash memory.</li><li>• It is necessary to issue a write enable command before erasing or programming. Also, applications must check the serial flash memory status to verify that the busy state has been cleared after erasing or programming.</li></ul> |
| Arguments     | None   |
| Return values | None   |
| Notes         |  |

## RX630 Group Communication with Flash Memory Using the Serial Communications Interface (Simple SPI)

|               |  |
|---------------|--|
| Name          | sf_byte_program  |
| Overview      | Data write   |
| Header        | None   |
| Declaration   | void sf_byte_program(unsigned long addr, unsigned char *buf, int size)   |
| Description   | <ul style="list-style-type: none"><li>• Programs the serial flash memory with the specified data.</li><li>• It is necessary to issue a write enable command before erasing or programming. Also, applications must check the serial flash memory status to verify that the busy state has been cleared after erasing or programming.</li><li>• The maximum write data size depends on the device used.</li></ul> |
| Arguments     | unsigned long addr: Address in the serial flash memory to write<br>unsigned char *buf: Address of buffer that holds the write data<br>int size: Number of bytes to write   |
| Return values | None   |
| Notes         |  |

|               |  |
|---------------|--|
| Name          | sf_byte_read   |
| Overview      | Data read  |
| Header        | None   |
| Declaration   | void sf_byte_read(unsigned long addr, unsigned char *buf, int size)  |
| Description   | Reads the specified number of bytes from the serial flash memory.  |
| Arguments     | unsigned long addr: Address in serial flash memory to read<br>unsigned char *buf: Address of buffer to hold read data<br>int size: Number of bytes to read |
| Return values | None   |
| Notes         |  |

|               |   |
|---------------|---|
| Name          | write_enable  |
| Overview      | Write enable  |
| Header        | None  |
| Declaration   | void write_enable(void)   |
| Description   | Issues a write enable command to make it possible to erase or program the flash memory. |
| Arguments     | None  |
| Return values | None  |
| Notes         |   |

|               |   |
|---------------|---|
| Name          | write_disable   |
| Overview      | Write disable   |
| Header        | None  |
| Declaration   | void write_disable(void)  |
| Description   | Issues a write disable command to prevent erasing or programming the serial flash memory. |
| Arguments     | None  |
| Return values | None  |
| Notes         |   |

## RX630 Group Communication with Flash Memory Using the Serial Communications Interface (Simple SPI)

|               |  |
|---------------|--|
| Name          | busy_wait  |
| Overview      | Wait while busy  |
| Header        | None   |
| Declaration   | void busy_wait(void)   |
| Description   | Loops internally while the serial flash memory status is busy. |
| Arguments     | None   |
| Return values | None   |
| Notes         |  |

|               |  |
|---------------|--|
| Name          | read_status                                  |
| Overview      | Status read                                  |
| Header        | None   |
| Declaration   | unsigned char read_status(void)              |
| Description   | Reads the status of the serial flash memory. |
| Arguments     | None   |
| Return values | Value of the status register                 |
| Notes         |  |

|               |  |
|---------------|--|
| Name          | write_status                                       |
| Overview      | Status write                                       |
| Header        | None   |
| Declaration   | void write_status(unsigned char status)            |
| Description   | Write the status of the serial flash memory.       |
| Arguments     | unsigned char status: Value of the status register |
| Return values | None   |
| Notes         |  |

|               |  |
|---------------|--|
| Name          | io_cmd_exe   |
| Overview      | Send command (for write)   |
| Header        | lodefine.h   |
| Declaration   | void io_cmd_exe(unsigned char *ope, int ope_sz, unsigned char *data, int data_sz)  |
| Description   | <ul style="list-style-type: none"><li>• Executes the specified command.</li><li>• After transmitting the ope argument, transmits the data argument. Received data is discarded.</li><li>• Specify a value in the range 0 to 8 for ope_sz.</li><li>• Specify a value in the range 0 to 256 for data_sz.</li></ul> |
| Arguments     | unsigned char *ope: Start address of the opcode block and address block to send.<br>int ope_sz: Number of bytes in the opcode block and address block.<br>unsigned char *data: Start address of the data block to transmit<br>int data_sz: Number of bytes in the data block                                     |
| Return values | None   |
| Notes         |  |

## RX630 Group Communication with Flash Memory Using the Serial Communications Interface (Simple SPI)

|               |   |
|---------------|---|
| Name          | io_cmd_exe_rdmode   |
| Overview      | Send command (for read)   |
| Header        | lodefine.h  |
| Declaration   | void io_cmd_exe_rdmode(unsigned char *ope, int ope_sz, unsigned char *rd, int rd_sz)  |
| Description   | <ul style="list-style-type: none"><li>Executes the specified command. After transmitting the ope argument, it receives data in the rd argument.</li><li>Specify a value in the range 0 to 8 for ope_sz.</li><li>Specify a value 0 or larger for rd_sz.</li></ul>      |
| Arguments     | unsigned char *ope: Start address of the opcode block and address block to send.<br>int ope_sz: Number of bytes in the opcode block and address block.<br>unsigned char *rd: Address buffer to hold the received data<br>int rd_sz: Number of bytes in the data block |
| Return values | None  |
| Notes         |   |

|               |   |
|---------------|---|
| Name          | io_wait_tx_end  |
| Overview      | Wait for transmit complete  |
| Header        | lodefine.h  |
| Declaration   | void io_wait_tx_end(void)   |
| Description   | Loops internally until it can verify that transmission has completed. |
| Arguments     | None  |
| Return values | None  |
| Notes         |   |

|               |  |
|---------------|--|
| Name          | eep_cs_init                                |
| Overview      | CS signal initialization                   |
| Header        | lodefine.h                                 |
| Declaration   | void eep_cs_init(void)                     |
| Description   | Initializes the CS signal (to high output) |
| Arguments     | None                                       |
| Return values | None                                       |
| Notes         |  |

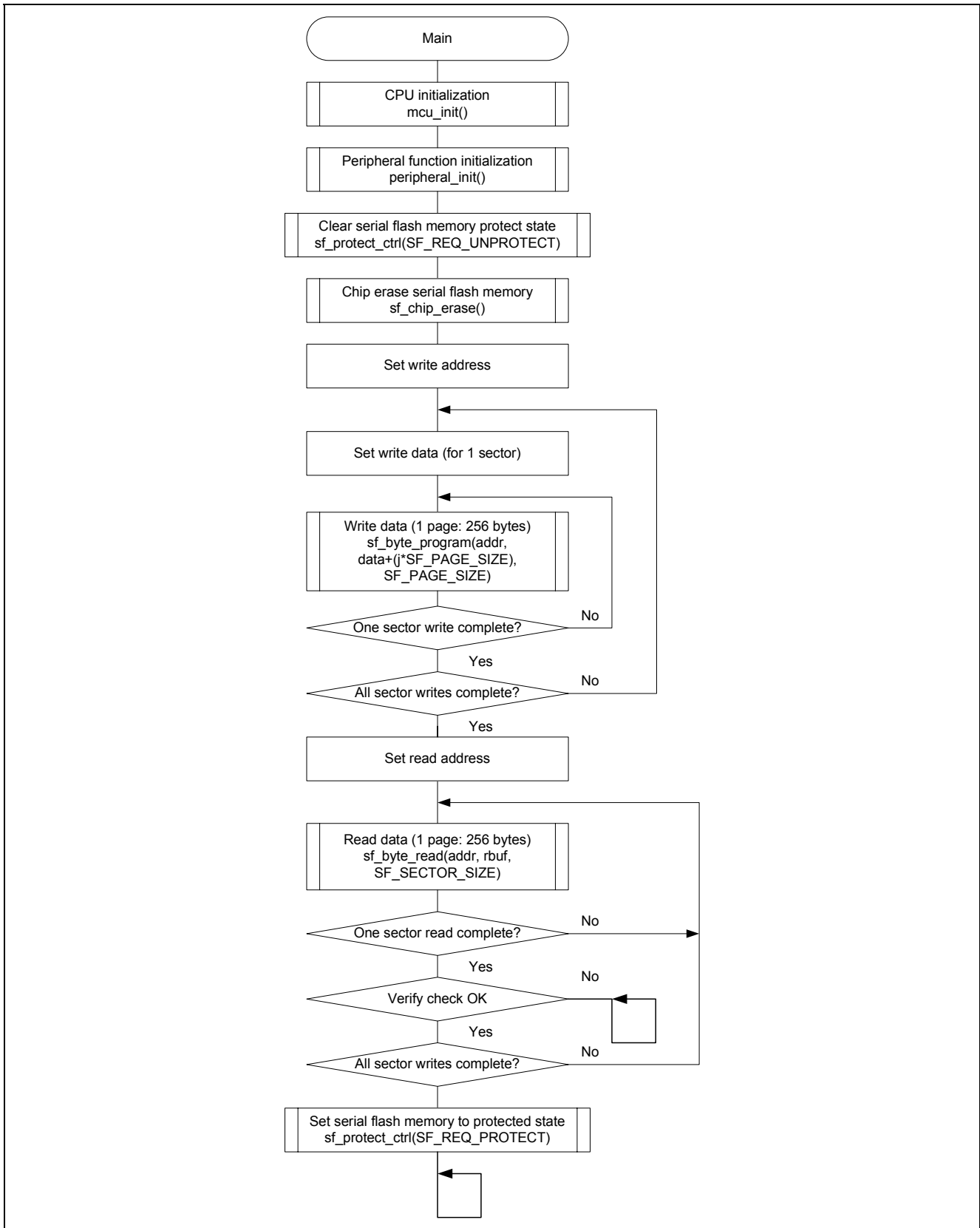
|               |                                   |
|---------------|-----------------------------------|
| Name          | eep_cs_start                      |
| Overview      | Starts the CS signal (low output) |
| Header        | lodefine.h                        |
| Declaration   | void eep_cs_start(void)           |
| Description   | Starts the CS signal (low output) |
| Arguments     | None                              |
| Return values | None                              |
| Notes         |                                   |

|               |                                  |
|---------------|----------------------------------|
| Name          | eep_cs_end                       |
| Overview      | Ends the CS signal (high output) |
| Header        | lodefine.h                       |
| Declaration   | void eep_cs_end(void)            |
| Description   | Ends the CS signal (high output) |
| Arguments     | None                             |
| Return values | None                             |
| Notes         |                                  |

### 3.7 Flowcharts

#### 3.7.1 Main Processing

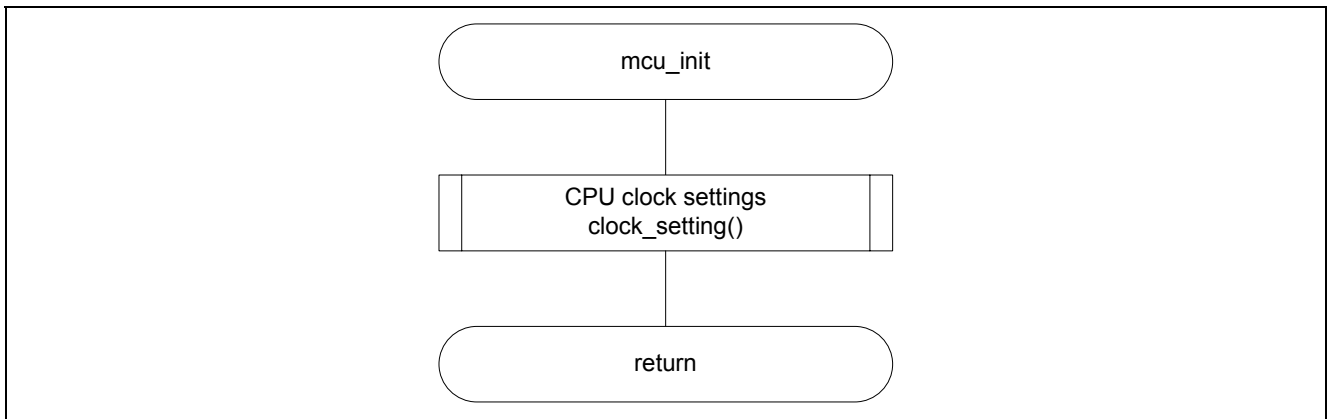
Figure 3.2 shows the main processing flowchart.



**Figure 3.2 Main Processing**

### 3.7.2 CPU Initialization

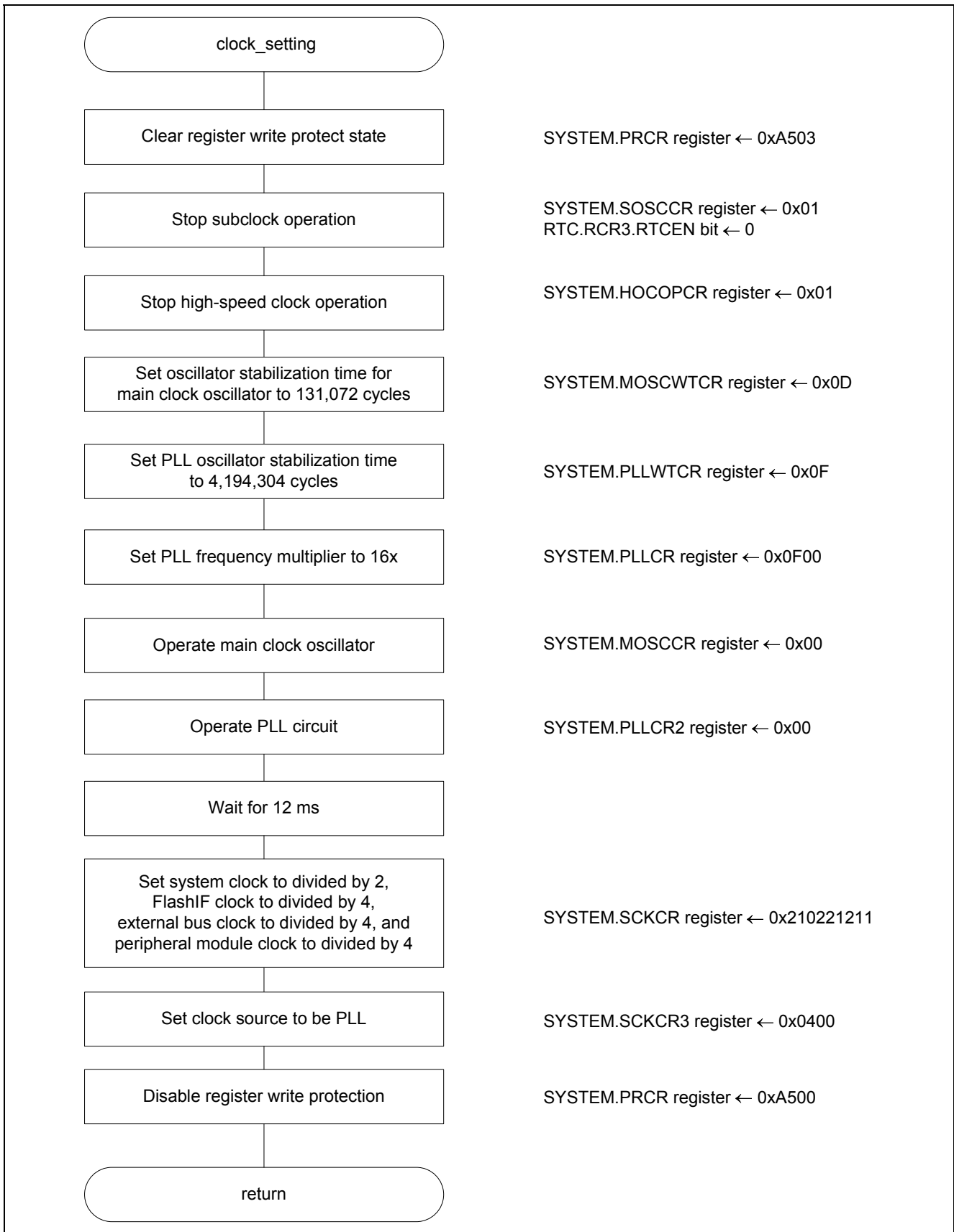
Figure 3.3 shows the flowchart for CPU initialization.



**Figure 3.3 CPU Initialization**

**3.7.3 CPU Clock Settings**

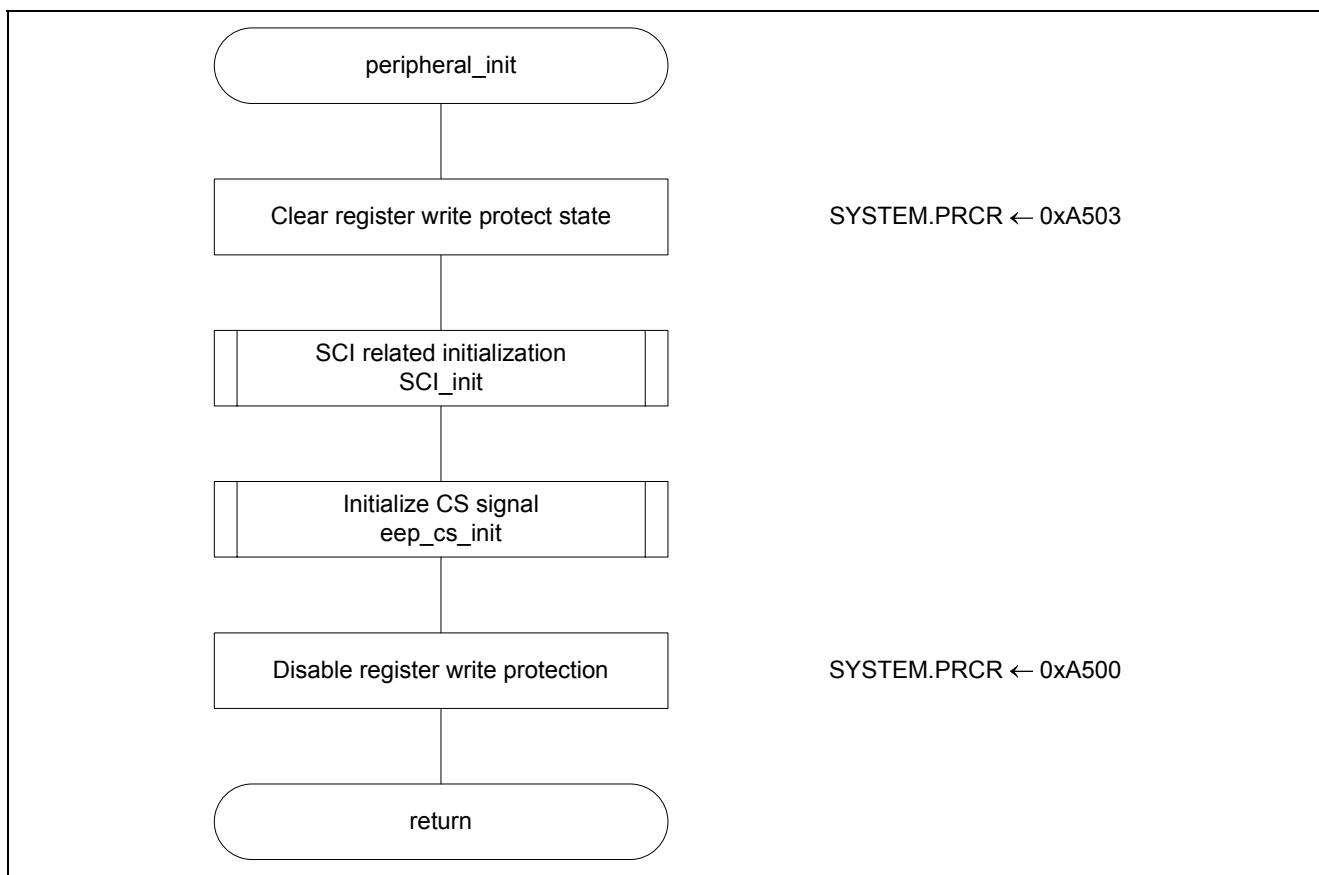
Figure 3.4 shows the flowchart for the CPU clock settings.



**Figure 3.4 CPU Clock Settings**

### 3.7.4 Peripheral Function Initialization

Figure 3.5 shows the flowchart for peripheral function initialization.

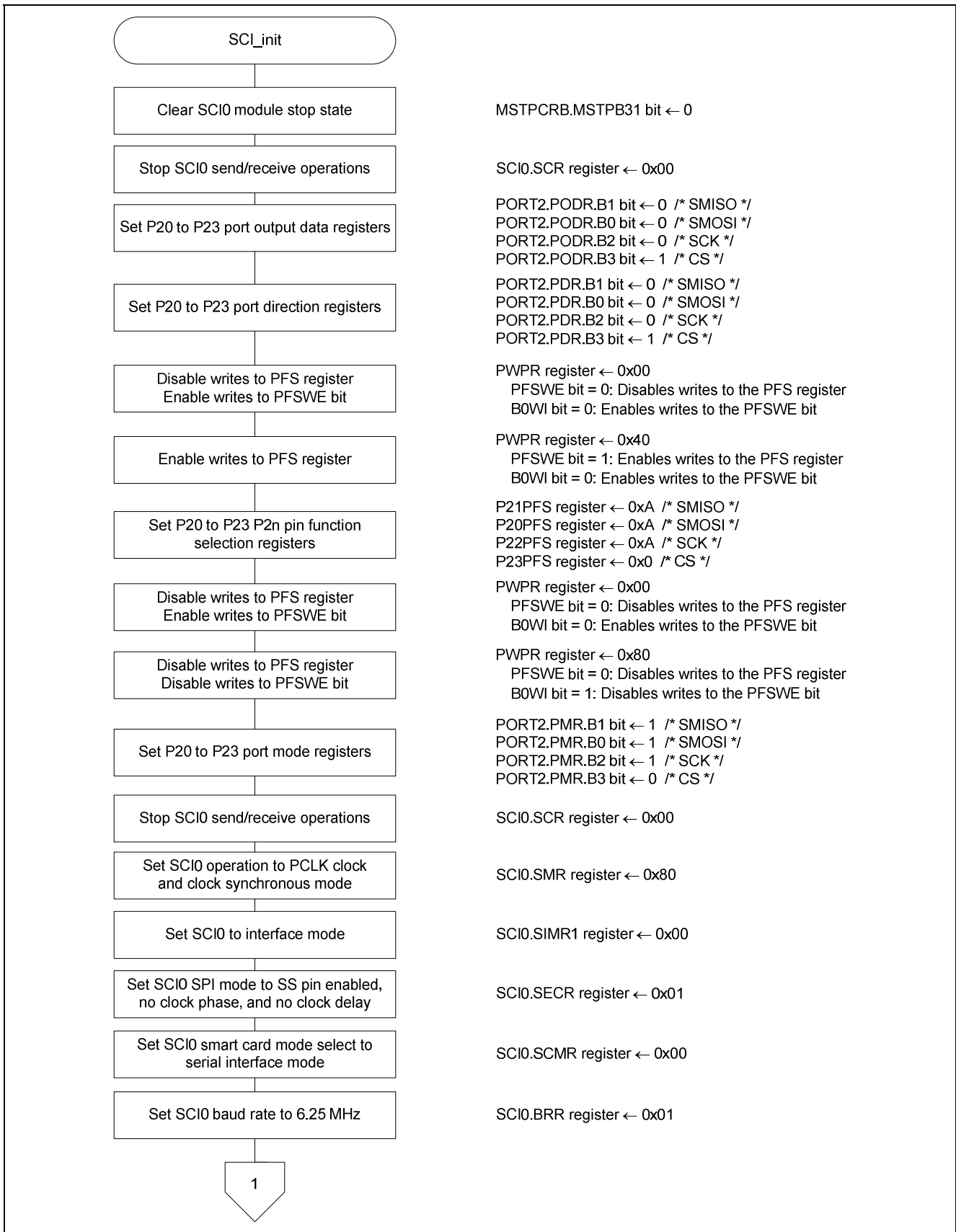


**Figure 3.5 Peripheral Function Initialization**

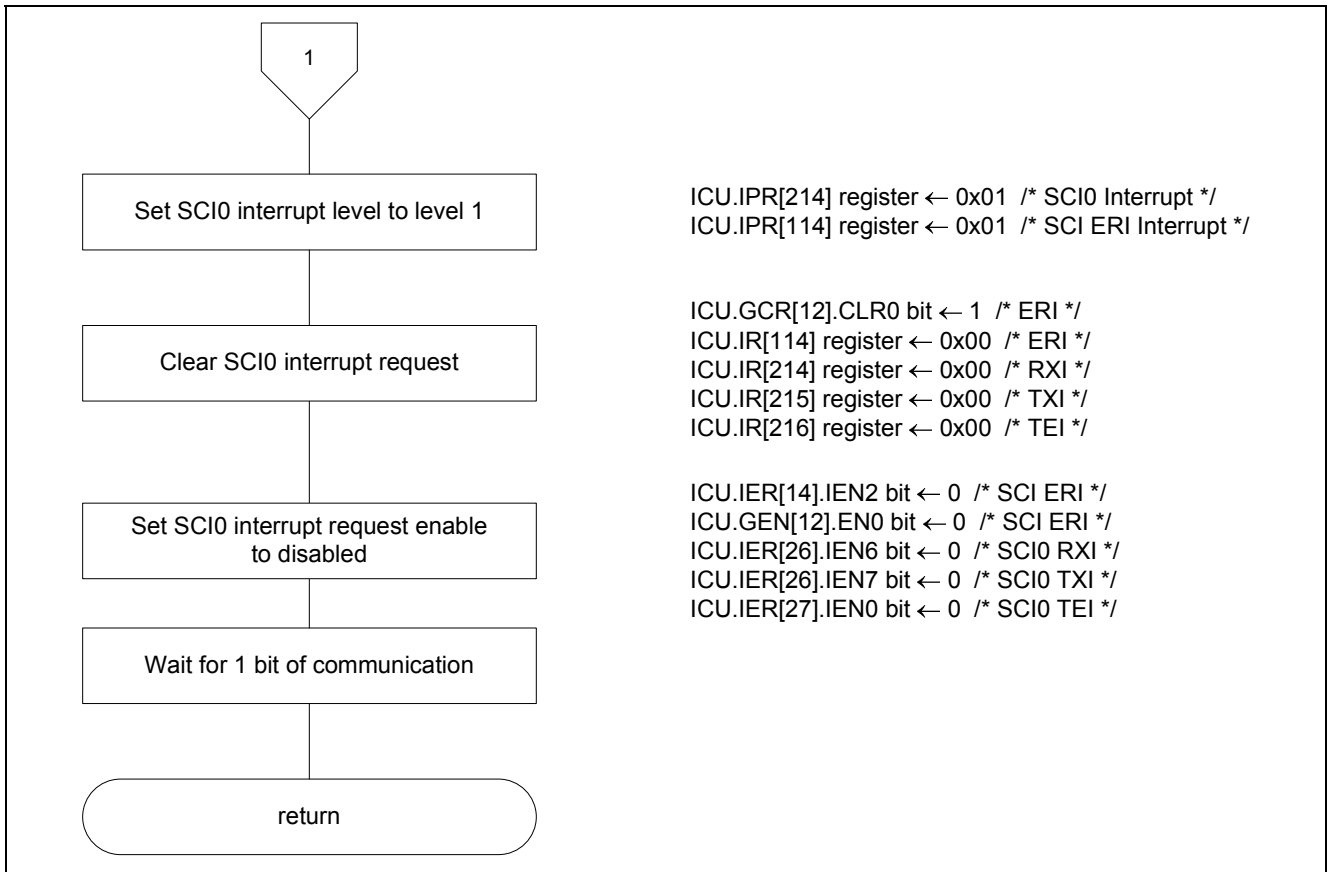


**3.7.5 Serial Communications Interface (SCI) Related Initialization**

Figures 3.6 and 3.7 show the flowcharts for the serial communications interface (SCI) related initialization.



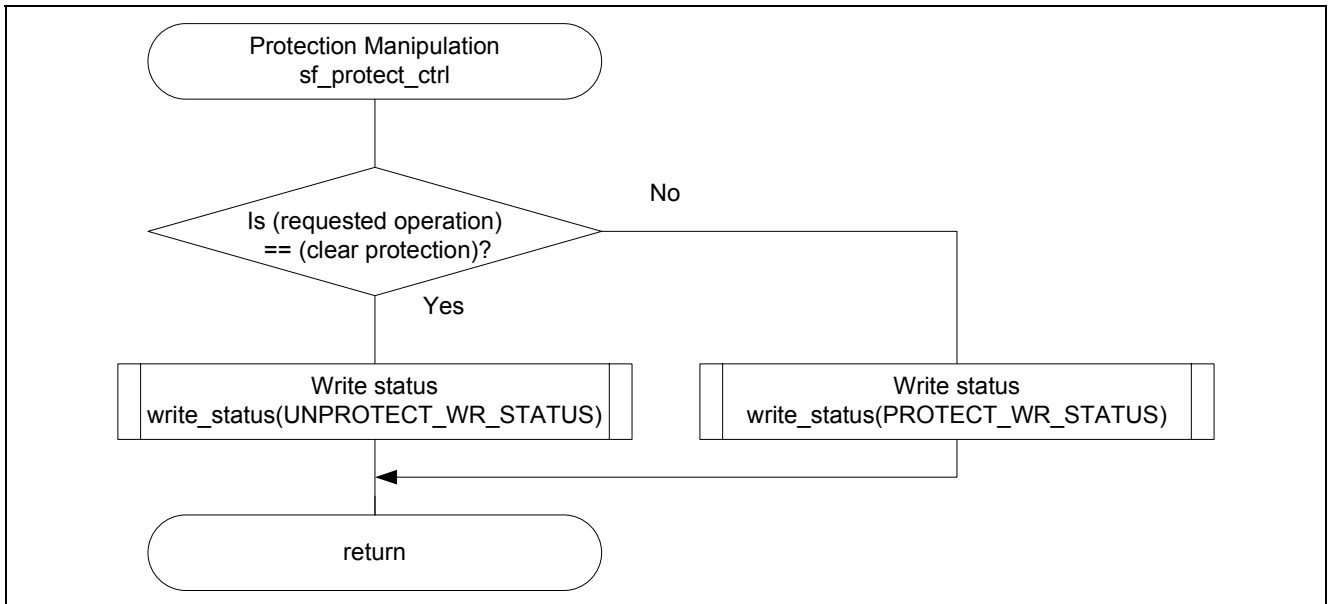
**Figure 3.6 Serial Communications Interface (SCI) Related Initialization (1)**



**Figure 3.7 Serial Communications Interface (SCI) Related Initialization (2)**

**3.7.6 Protection State Manipulation**

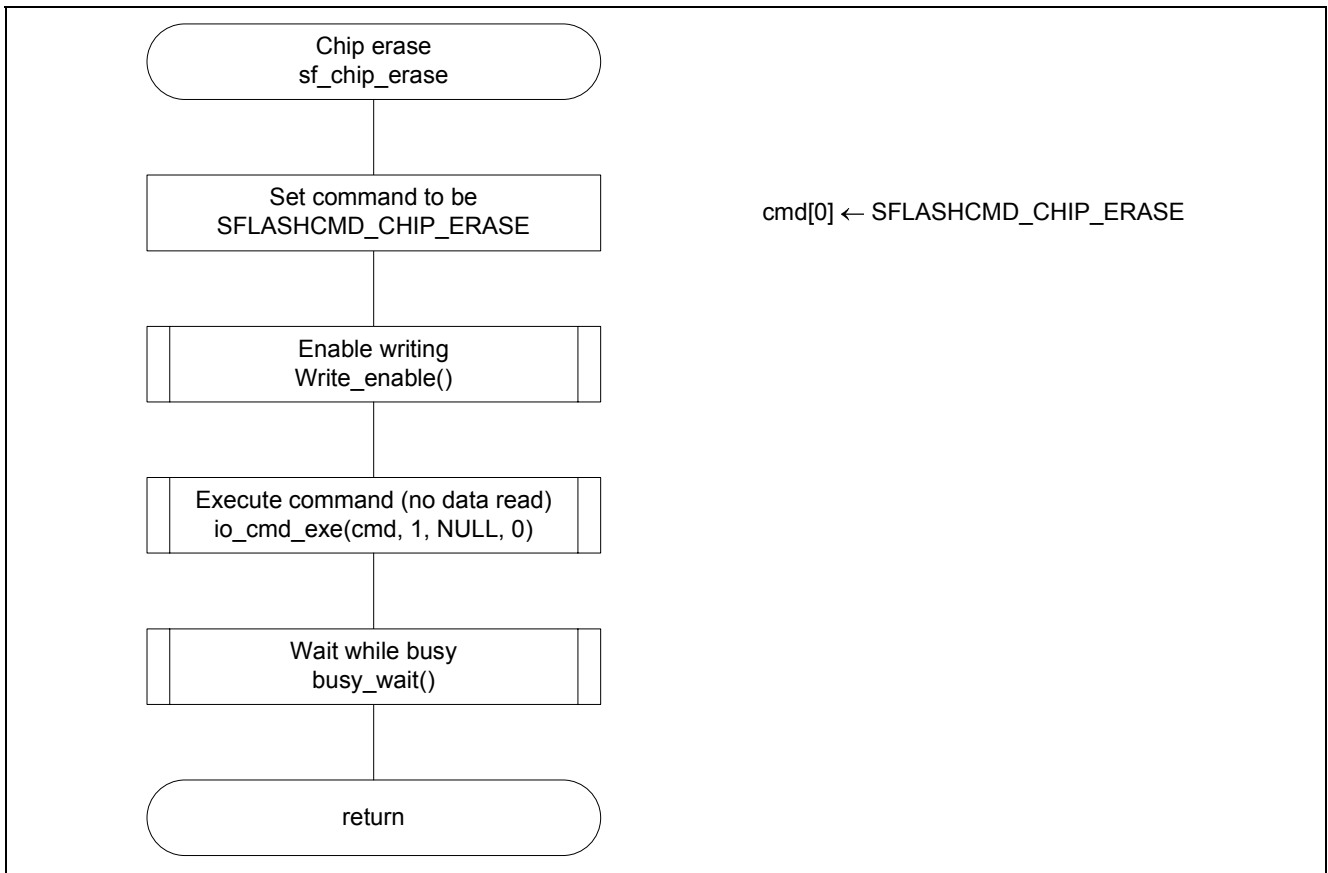
Figure 3.8 shows the flowchart for manipulating the protection state.



**Figure 3.8 Protection State Manipulation**

### 3.7.7 Chip Erase Processing

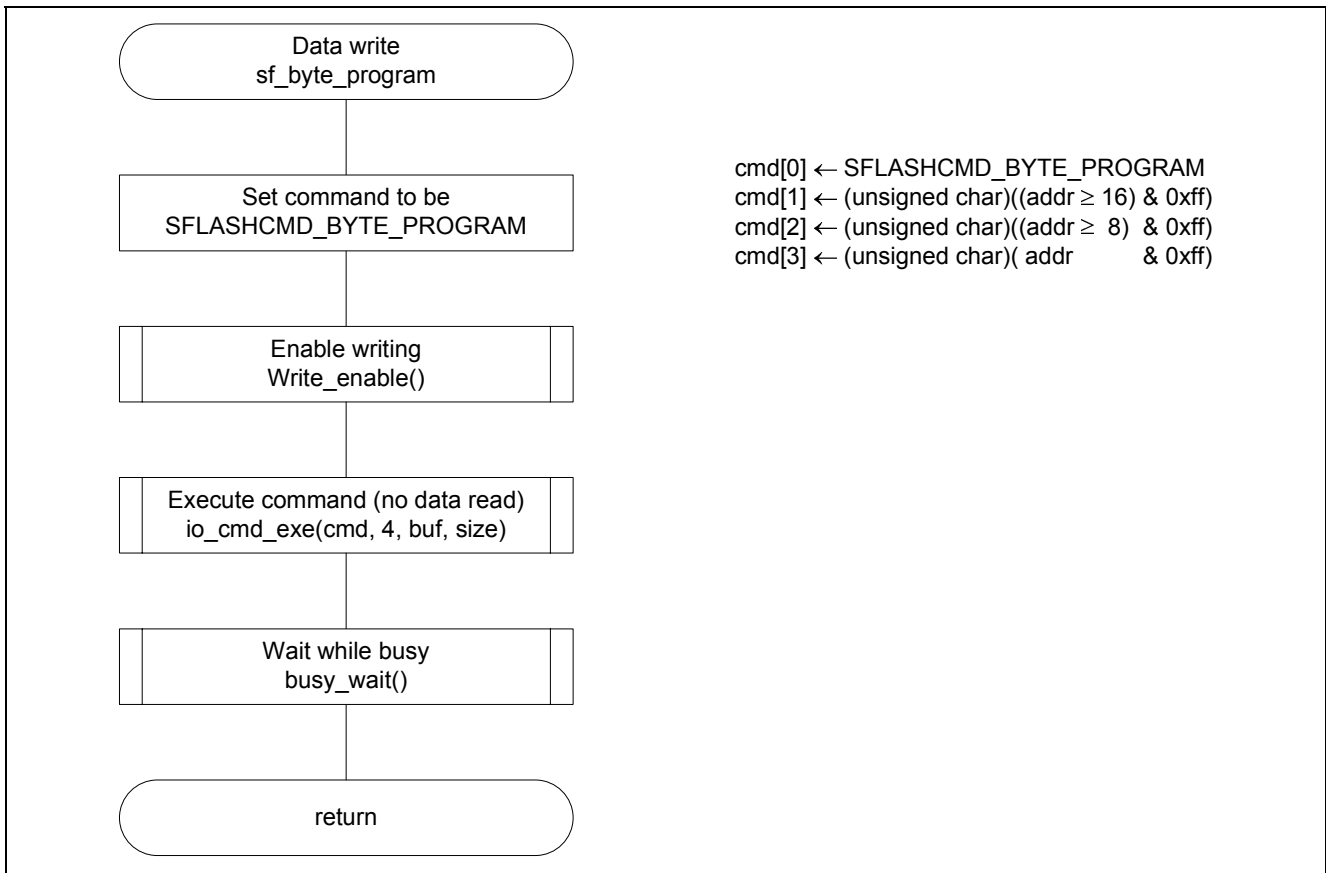
Figure 3.9 shows the flowchart for the chip erase operation.



**Figure 3.9** Chip Erase Processing

### 3.7.8 Data Write Processing

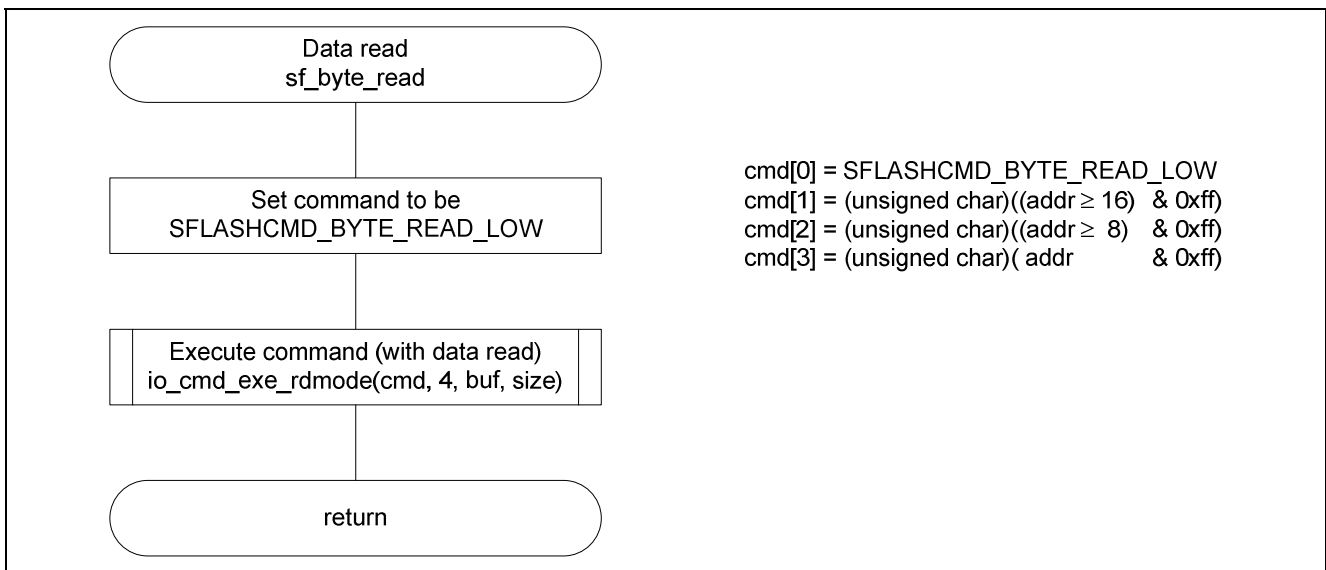
Figure 3.10 shows the flowchart for the data write operation.



**Figure 3.10 Data Write Processing**

### 3.7.9 Data Read Processing

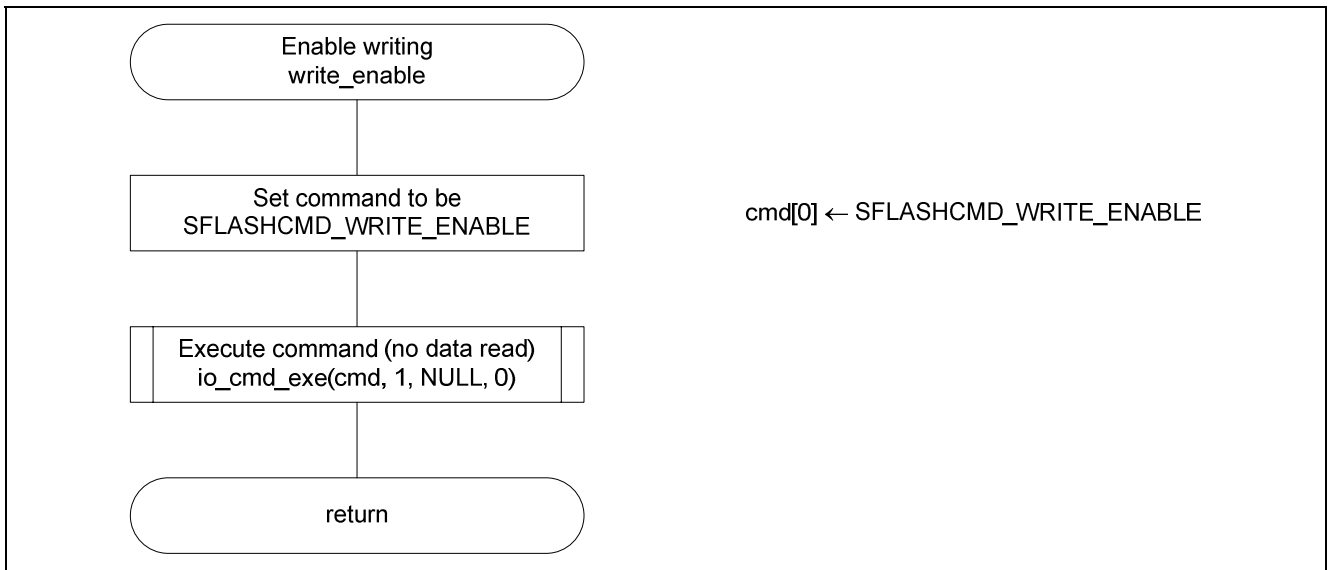
Figure 3.11 shows the flowchart for the data read operation.



**Figure 3.11 Data Read Processing**

### 3.7.10 Write Enable Processing

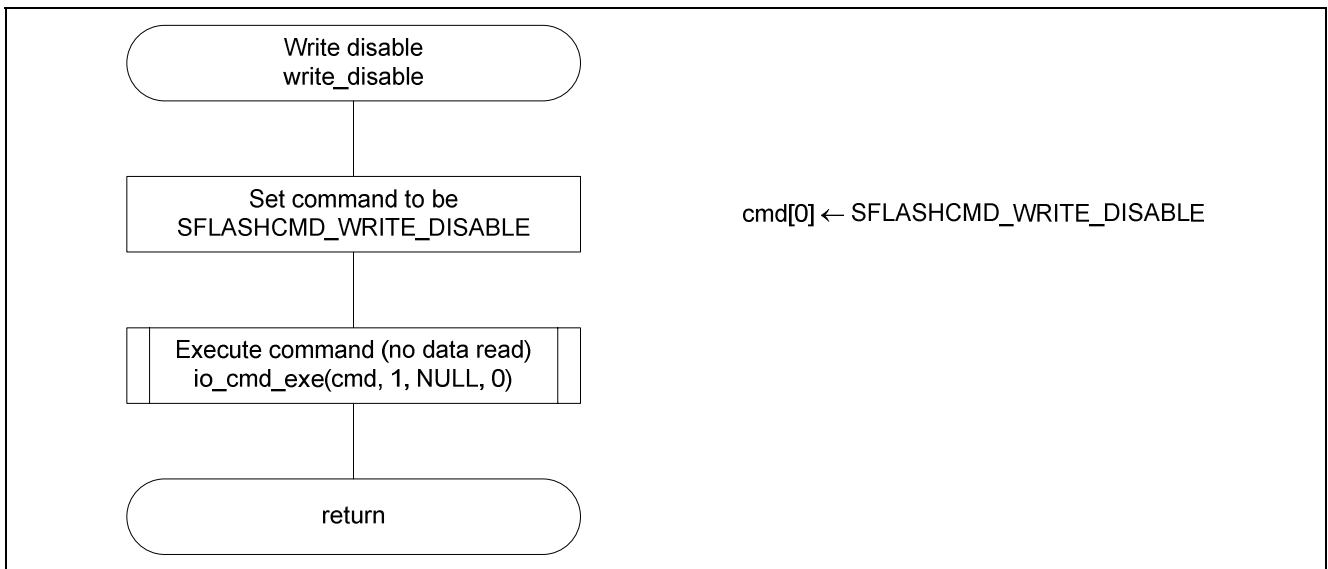
Figure 3.12 shows the flowchart for the write enable operation.



**Figure 3.12 Write Enable Processing**

### 3.7.11 Write Disable Processing

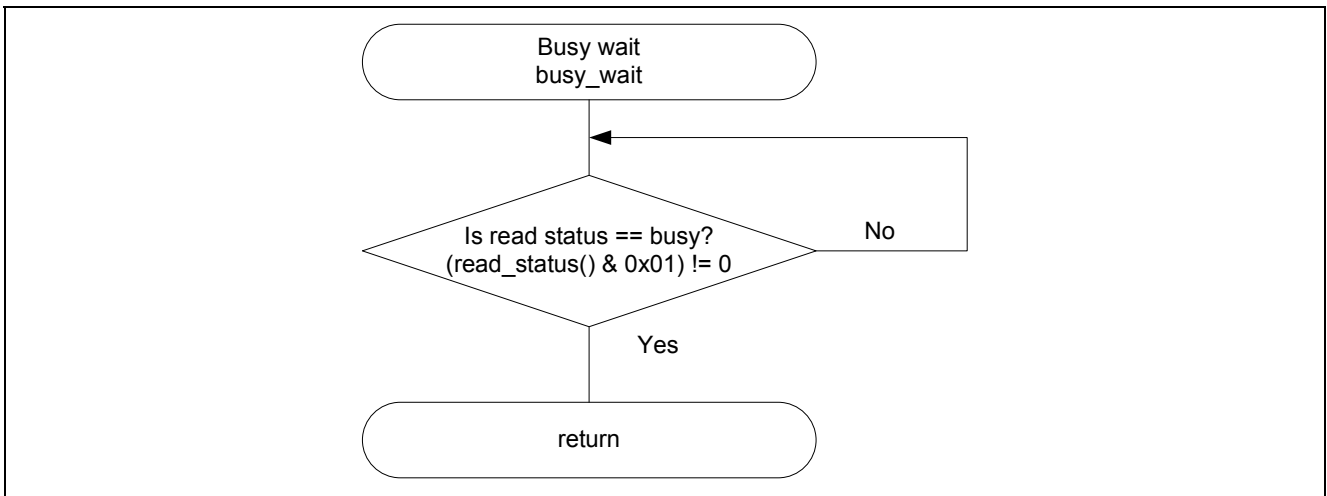
Figure 3.13 shows the flowchart for the write disable operation.



**Figure 3.13 Write Disable Processing**

**3.7.12 Busy Wait Processing**

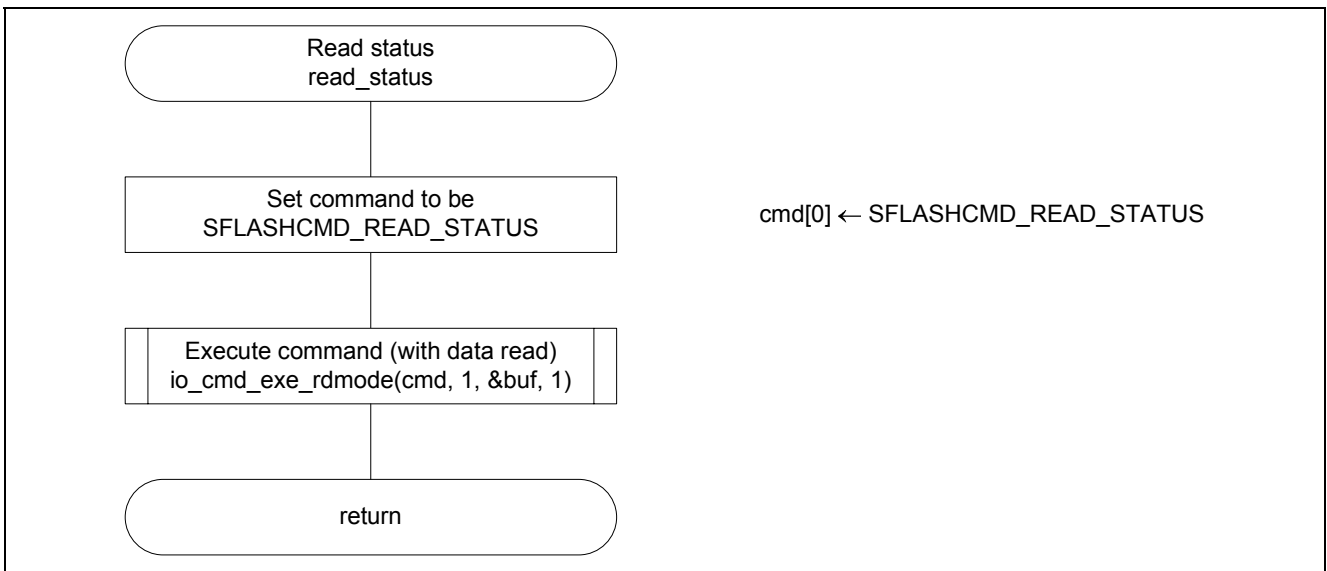
Figure 3.14 shows the flowchart for the busy wait operation.



**Figure 3.14 Busy Wait Processing**

**3.7.13 Status Read Processing**

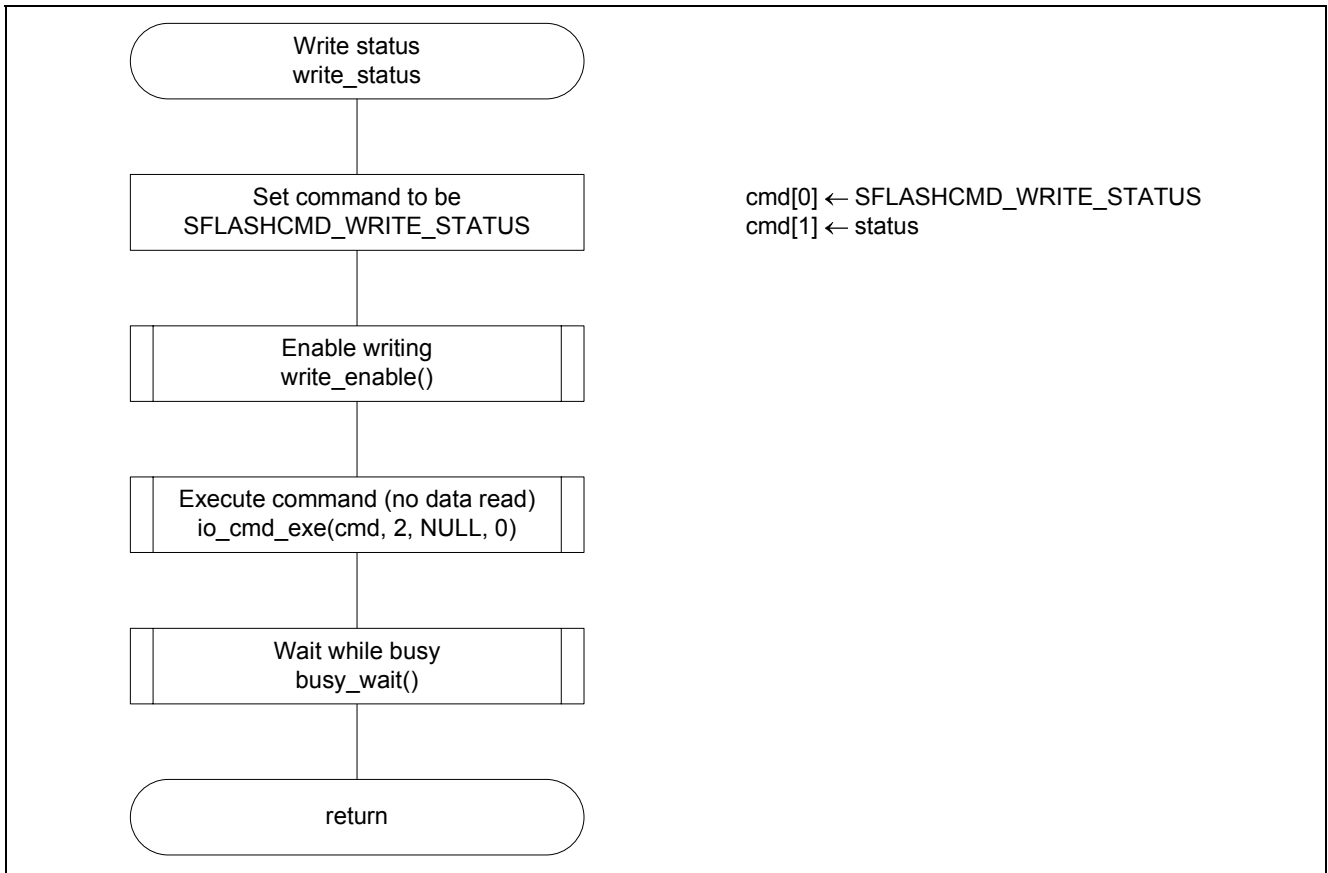
Figure 3.15 shows the flowchart for the status read operation.



**Figure 3.15 Status Read Processing**

### 3.7.14 Status Write Processing

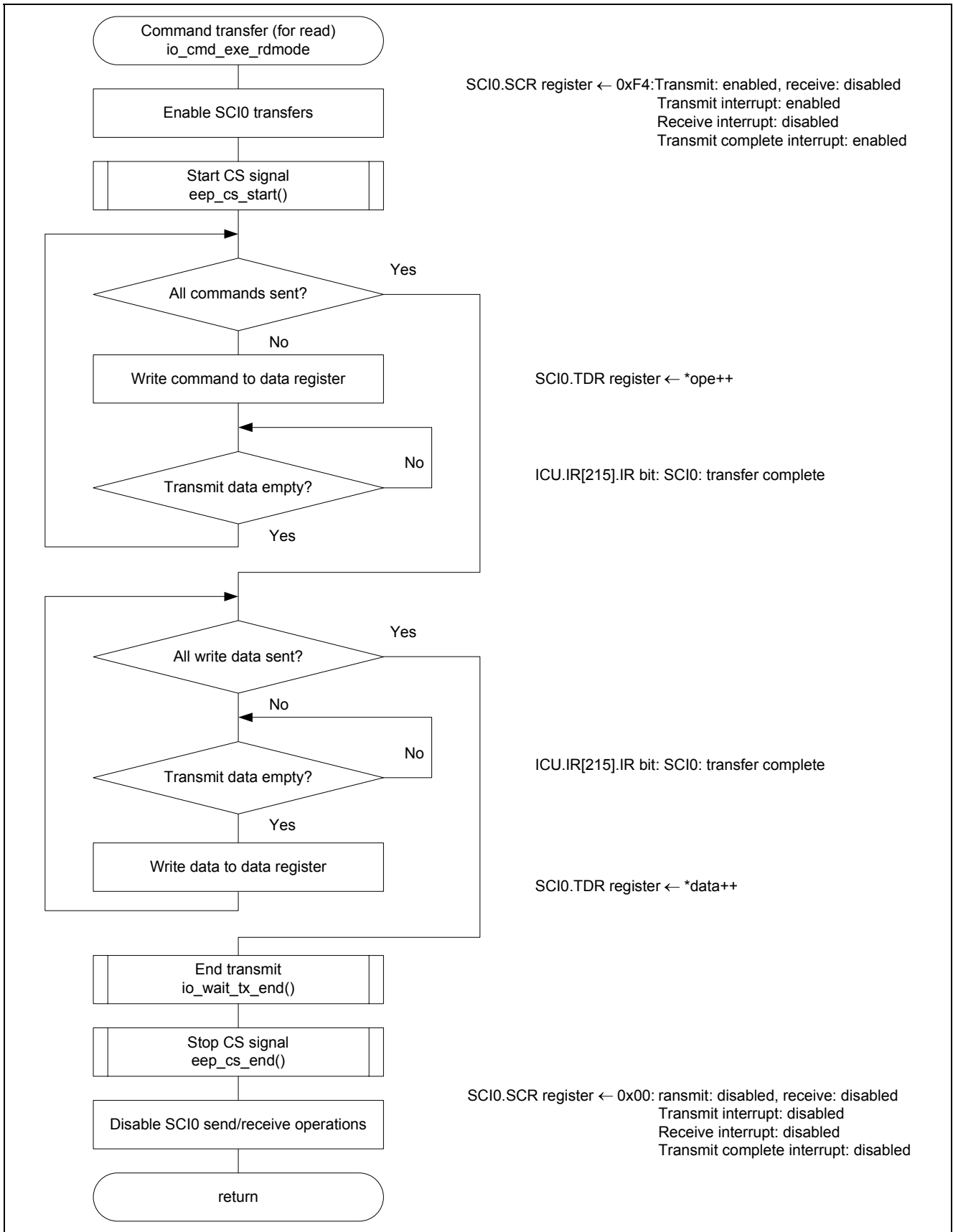
Figure 3.16 shows the flowchart for the status write operation.



**Figure 3.16 Status Write Processing**

**3.7.15 Command Transfer Processing (for write)**

Figure 3.17 shows the flowchart for the command transfer (for write) operation.

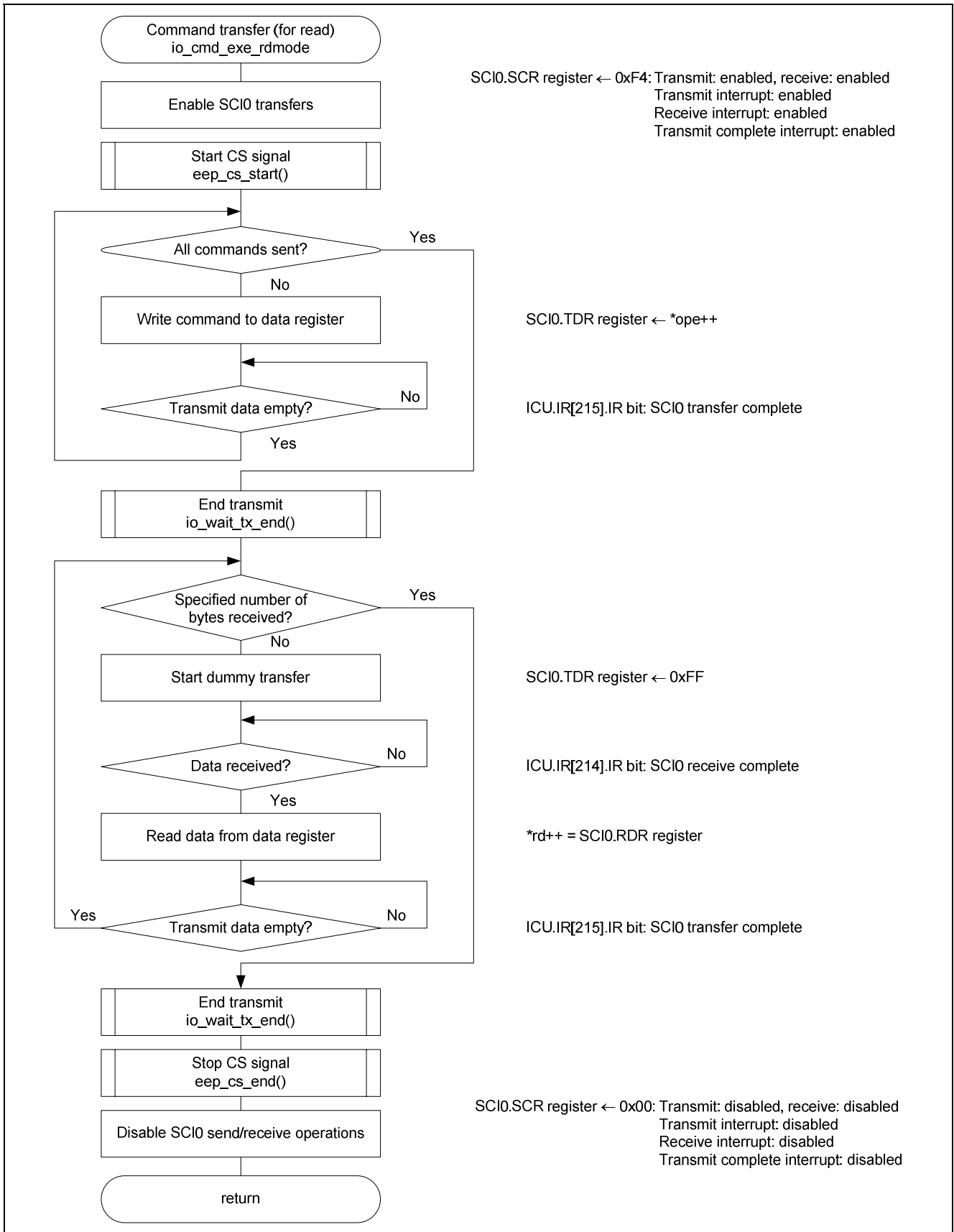


**Figure 3.17 Command Transfer Processing (for write)**



**3.7.16 Command Transfer Processing (for write)**

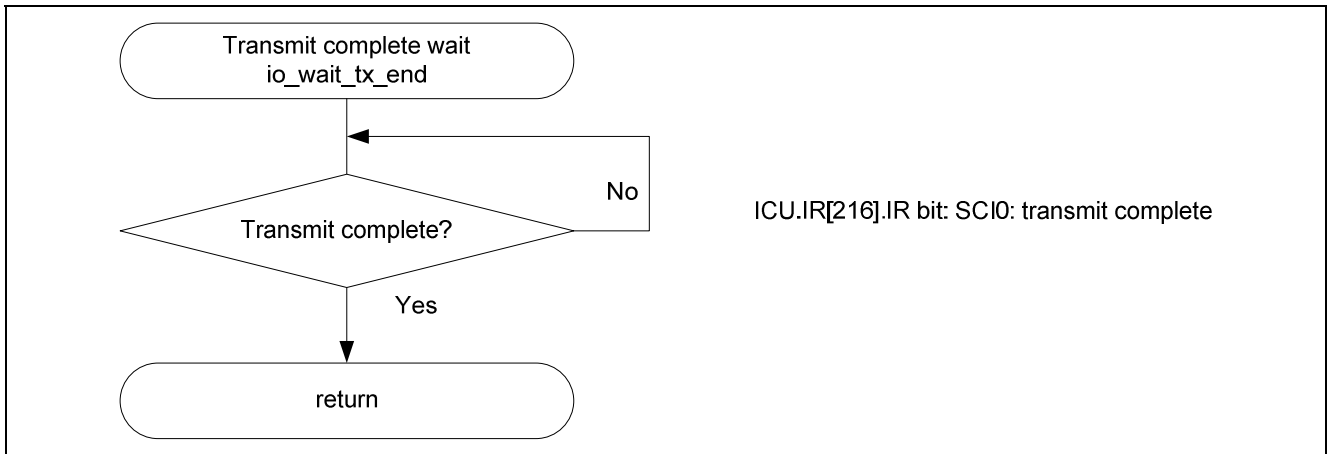
Figure 3.18 shows the flowchart for the command transfer (for read) operation.



**Figure 3.18 Command Transfer Processing (for write)**

**3.7.17 Transmit Complete Wait Processing**

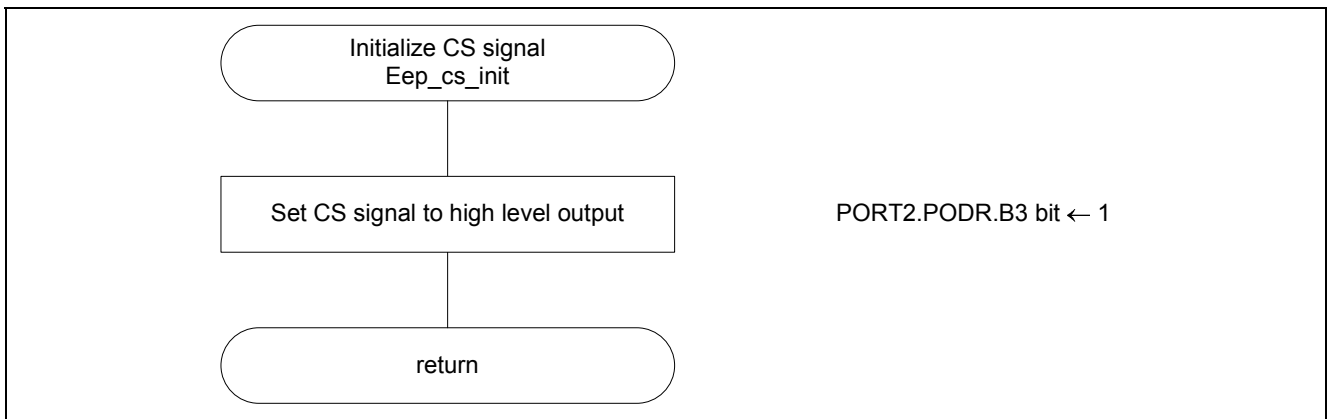
Figure 3.19 shows the flowchart for transmit complete wait processing.



**Figure 3.19 Transmit Complete Wait Processing**

**3.7.18 CS Signal Initialization**

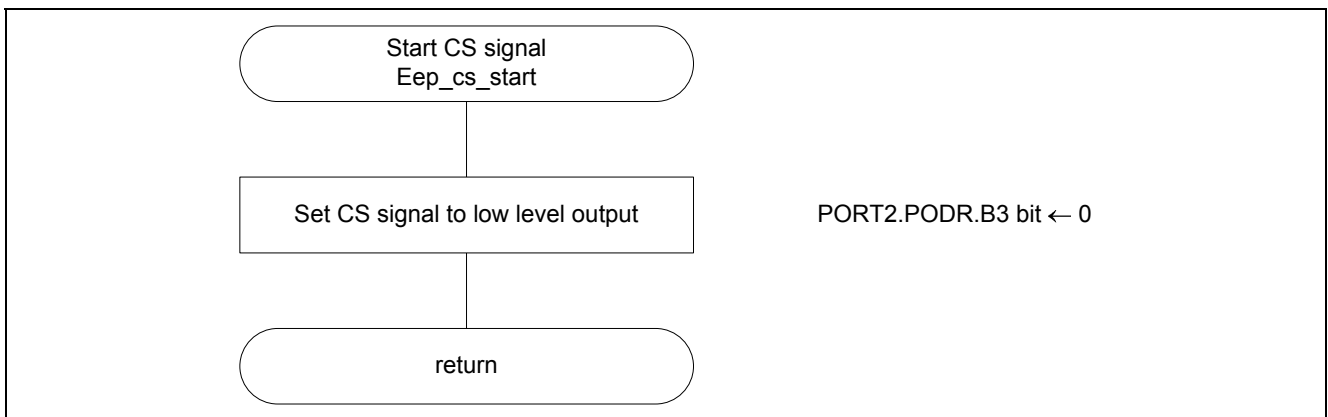
Figure 3.20 shows the flowchart for CS signal initialization.



**Figure 3.20 CS Signal Initialization**

**3.7.19 Start CS Signal**

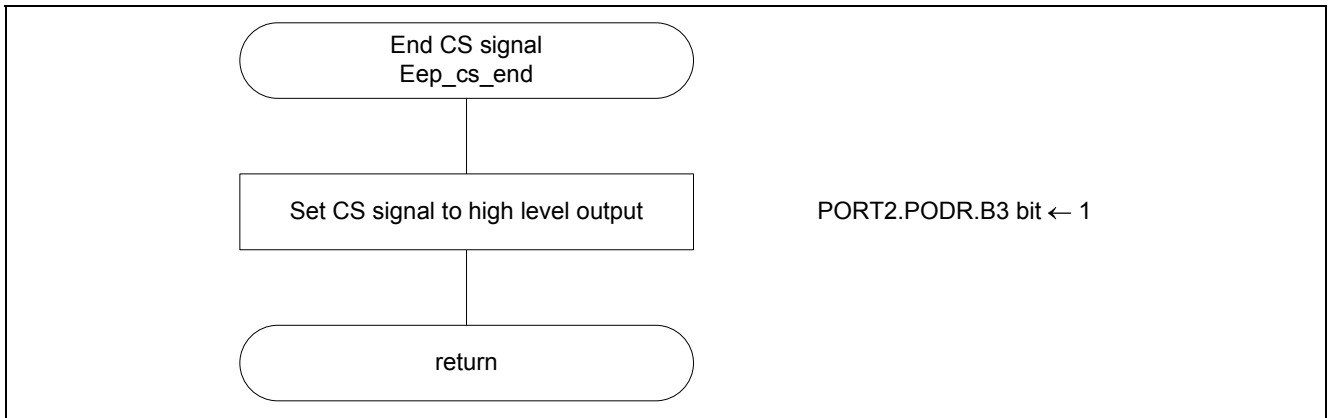
Figure 3.21 shows the flowchart for the CS signal start operation.



**Figure 3.21 CS Signal Start Operation**

### 3.7.20 End CS Signal

Figure 3.22 shows the flowchart for the CS signal end operation.



**Figure 3.22 CS Signal End Operation**

#### **4. Sample Programs**

The sample program can be downloaded from the Renesas Electronics Web site.

#### **5. Reference Documents**

- RX630 Group User's Manual: Hardware, Rev.1.00  
(The latest version can be downloaded from the Renesas Electronics Web site.)
- Technical Updates/Technical News  
(The latest information can be downloaded from the Renesas Electronics Web site.)
- C Compiler Manual  
RX Family C/C++ Compiler Package User's Manual V.1.0.1.0  
(The latest version can be downloaded from the Renesas Electronics Web site.)

## Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

All trademarks and registered trademarks are the property of their respective owners.

## Revision Record

| Rev. | Date      | Description |                      |
|------|-----------|-------------|----------------------|
|      |           | Page        | Summary              |
| 1.00 | Dec.13.11 | —           | 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 document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

### 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 a product with a different part number, confirm that the change will not lead to problems.

- The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the 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. 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.

13F, 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