To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

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.

SH7147

Synchronous Serial Communication Unit Master Transmission/Reception (Writing to/Reading from EEPROM via SPI Bus)

1. Abstract

This application note describes master transmission/reception during four-wire serial transmission using the Synchronous Serial Communication Unit module. You can use this application note as reference information for designing user software.

Although the operation of each program in this application note has been checked, make sure that you conduct your own operation checks before actually using.

Note: EEPROM (HN58X2564I) used in this application note is intended for commercial use.

2. Introduction

2.1 Specifications

- Write 1 byte data to EEPROM on SPI bus by Synchronous Serial Communication Unit master transmission.
- Read 1 byte data from EEPROM on SPI bus by Synchronous Serial Communication Unit master reception.

2.2 Function Used

- Synchronous Serial Communication Unit

2.3 Applicable Conditions

- MCU : SH7147(R5F71474AK64FPV)
- Operation Frequency : Internal Clock 64MHz
  : Bus Clock 32MHz
  : Peripheral Clock 32MHz
  : MTU2S Clock 64MHz
  : MTU2 Clock 32MHz
- C Compiler : Renesas Technology product Ver. V.9.1.0.0
  SuperH RISC engine Family C/C++ compiler package
- Compile Option : Default setting by HEW

3. Description of Sample Task

- The Synchronous Serial Communication Unit module of the SH7147 is used to write/read 1-byte data to/from a four -wire serial-transmission EEPROM (HN58X2564I, 64 Kbits, 8 Kwords × 8 bits).
- The connection is a single-master configuration with the SH7147 used as the master device.
- The data transfer clock is set to 2 MHz.
- Figure 1 shows connection between the SH7147 and the EEPROM. Table 1 shows pin functions of Synchronous Serial Communication Unit and EEPROM.
- Table 2 shows the overview of Synchronous Serial Communication Unit settings.
- Table 3 is a list of EEPROM instruction codes used in this sample task.
- Table 4 is a list of EEPROM status registers used in this sample task.

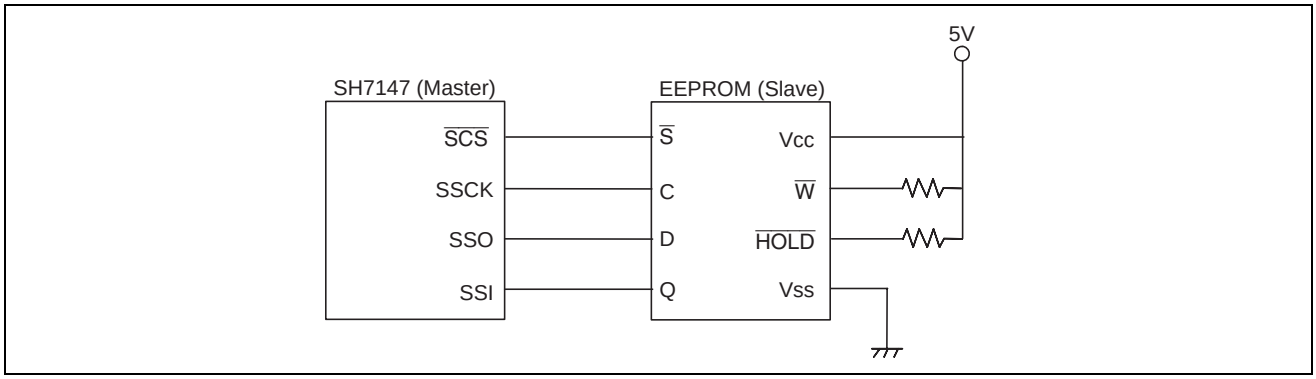


Figure 1 Connection between the SH7147 and EEPROM

Table 1 Pin Functions of Synchronous Serial Communication Unit and EEPROM

Pin Name	Pin Function	Synchronous Serial Communication Unit/EEPROM
$\overline{SCS}$	Synchronous Serial Communication Unit Chip Select Input/Output Pin (Output setting in this sample)	Synchronous Serial Communication Unit Pin
SSCK	Synchronous Serial Communication Unit Clock Input/Output Pin (Output setting in this sample)	
SSI	Synchronous Serial Communication Unit Data Input/Output Pin (Input setting in this sample)	
SSO	Synchronous Serial Communication Unit Data Input/Output Pin (Output setting in this sample)	
$\overline{S}$	EEPROM Chip Select Input Pin	EEPROM Pin
C	EEPROM Clock Input Pin	
D	EEPROM Data Input Pin	
Q	EEPROM Data Output Pin	
Vcc	EEPROM Power Supply Pin	
$\overline{W}$	EEPROM Write Protect Pin (High-fixed in this sample)	
$\overline{HOLD}$	EEPROM Hold Pin (High-fixed in this sample)	
Vss	EEPROM Ground Pin	

Table 2 Overview of Synchronous Serial Communication Unit Settings

Format	Settings
Operation mode	Master mode
Data input pin	Normal mode (uses two pins, data input pin and data output pin)
Transfer clock	2 MHz (Pϕ=32MHz)
Number of data bits	8 bits (when transmitting write code to EEPROM) 16 bits (when reading status flag from EEPROM) 32 bits (when writing data to EEPROM) 32 bits (when reading data from EEPROM)
MSB/LSB first	MSB first
Timing for setting the TEND bit	After the final bit is transmitted

Table 3 EEPROM Instruction Codes

Code Name	Operation	Code Format
WREN	Sets the EEPROM to be writable	0000 0110
WRDI	Sets the EEPROM to be unwritable	0000 0100
RDSR	Reads the EEPROM Status Register	0000 0101
WRSR	Writes to the EEPROM Status Register	0000 0001
READ	Reads stored data from EEPROM	0000 0011
WRITE	Writes stored data to EEPROM	0000 0010

Table 4 EEPROM Status Register

Register Name	Bit Name	Bit	Function
EEPROM Status Register	SRWD	7	Operates in combination with the Write Protect ($\overline{W}$) signal. Depending on the combination with the Write Protect ($\overline{W}$) signal, the device can be placed in Hardware Protected mode. However, as for $\overline{W}$ signal is High-fixed in this sample, Hardware Protected mode cannot be set. (Please refer EEPROM manual for details)
	Reserved	6	—
	Reserved	5	—
	Reserved	4	—
	BP[1:0]	3-2	Sets the software protected area 00: Not software protected 01: Software protected area is H'1800-H'1FFF 10: Software protected area is H'1000-H'1FFF 11: Software protected area is H'0000-H'1FFF
	WEL	1	0: Data write and write to status register are disabled. 1: Data write and write to status register are enabled.
	WIP	0	0: Write operation end 1: Write operation undergoing

3.1 Operation Description of Functions Used

The Synchronous Serial Communication Unit supports a master mode (with clock output from this LSI) and a slave mode (clock input from an external device). In addition, the Synchronous Serial Communication Unit allows synchronous serial communication between devices with different clock polarities and clock phases. Figure 2 shows a block diagram of the Synchronous Serial Communication Unit module. Table 5 shows description of each register.

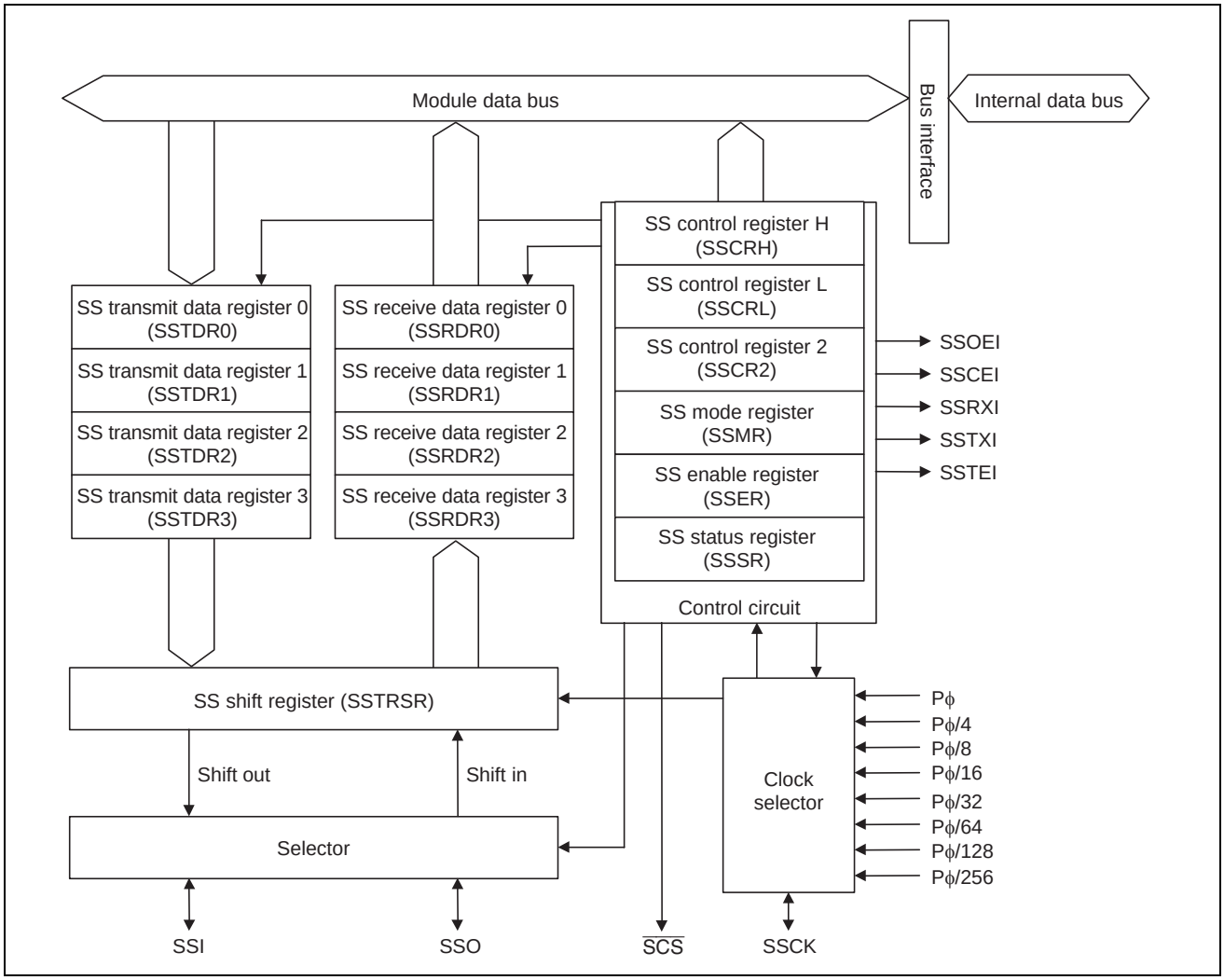


Figure 2 Synchronous Serial Communication Unit Block Diagram

Table 5 Synchronous Serial Communication Unit Registers

Register Name	Abbreviation	Function
SS Control Register H	SSCRH	Selects the master or slave mode Selects the input/output pin mode Selects the SSO pin output value Selects the $\overline{SCS}$ pin function
SS Control Register L	SSCRL	Selects the flag clear mode, operating mode, software reset, and the transmit/receive data length
SS Mode Register	SSMR	Selects MSB first or LSB first, the clock polarity, the clock phase, and the transfer clock rate.
SS Enable Register	SSER	Enables and disables transmission, reception, and interrupts requests.
SS Status Register	SSSR	The status flag register for various interrupts.
SS Control Register 2	SSCR2	Sets the timing for asserting the $\overline{SCS}$ signal, the timing for outputting data from the SSO pin, and the timing for setting the TEND bit.
SS Transmit Data Registers 0 to 3	SSTDR0 to 3	8-bit registers used to store transmit data.
SS Receive Data Registers 0 to 3	SSRDR0 to 3	8-bit registers used to store receive data.
SS Shift Register	SSTRSR	Shift register for serial data transmission and reception.

Note: For details on functions of each register, please refer “Section 14. Synchronous Serial Communication Unit” in the SH7147 Hardware Manual

3.2 Operation Description

(1) Writing data into EEPROM

Procedure 1: Set/Confirm Write Enable state

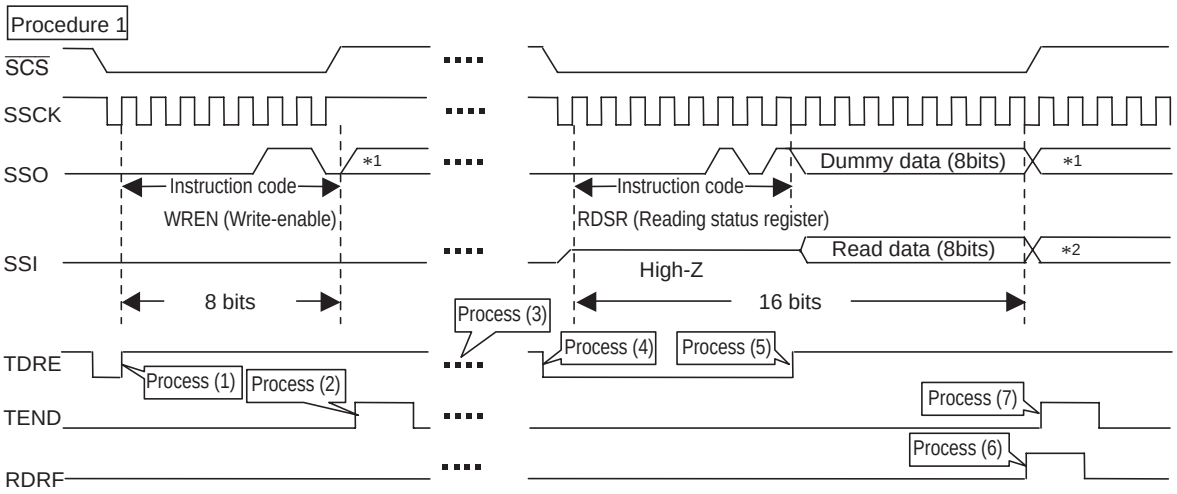
- EEPROM is set to Write Enable state by transmitting WREN (write-enable) code to the EEPROM to set WEL bit of EEPROM status register to 1.
- Confirm that EEPROM is set to Write Enable state by transmitting RDSR (reading status register) code to EEPROM to check that WEL bit of status register has been set 1.
- Figure 3 describes the communication and the operation. Table 6 shows description of software/ Hardware process.

Procedure 2: Data Write

- Write data to EEPROM by transmitting WRITE (write) code, write address (upper 8bits, lower 8bits), and write data to EEPROM.
- Figure 4 describes the communication and the operation. Table 7 shows description of software/ hardware process.

Procedure 3: Data Write End Confirmation

- Confirm that Write process have ended by transmitting RDSR (reading status register) code to confirm WIP bit of status register is cleared 0.
- Figure 5 describes the communication and the operation. Table 8 shows description of software/ hardware process.



Notes: 1. SSO pin, after data transmission, retains the level of the final bit.
2. SSI pin, after data reception, retains the level of the final bit.

Figure 3 Communication and Operation (Set/Confirm Write Enable state)

Table 6 Description of Software/Hardware Process (Set/Confirm Write Enable state)

	Software Process	Hardware Process
Process (1)	• Sets TE bit to 1 for enabling transmission • Confirms that TDRE bit is been set 1 • Sets transmit data (write-enable code) to SSTDR0 register	• Setting transmit data to the SSTDR0 register clears the TDRE bit to 0. • Transmits data from SSTDR0 register to SSTRSR register and then sets TDRE bit to 1
Process (2)	• Confirms that TEND=1 (transmit end) then clears TEND to 0. • Clears TE bit to 0 for prohibiting transmission	• Transmits the final bit while TDRE=1, and then sets TEND bit to 1
Process (3)	• Sets DATS bit to 1 for setting transmit data length to 16bit. • Confirms that TEND = 1 • Waits until 1 bit period elapses • Enables transmission/reception by setting TE bit and RE bit to 1.	• None
Process (4)	• Confirms that TDRE=1 • Sets transmit data (status read code) to SSTDR0 register • Sets transmit data (dummy data) to SSTDR1 register	• Setting transmit data to the SSTDR0 to 1 register clears the TDRE bit to 0.
Process (5)	• None	• Transmits data from SSTDR1 register to SSTRSR register, and then sets TDRE bit to 1
Process (6)	• Confirms that RDRF=1, and then ORER≠1, transmits values of SSRDR0 - 1 register to RAM	• Transmits reception data from SSTRSR register to SSRDR1 register, and then sets RDRF bit to 1 • RDRF bit is cleared to 0 when values of SSRDR0 – 1 register are read by CPU.
Process (7)	• Confirms that TEND=1 (transmit end) then clears TEND to 0. • Waits for one bit period to elapse • Confirms that 1bit of SSRDR1 register (EEPROM/WEL bit) is set to 1 (Write Enable state) • Clears TE bit and RE bit to 0 for prohibiting transmission/reception.	• Transmits the final bit while TDRE=1, and then sets TEND bit to 1

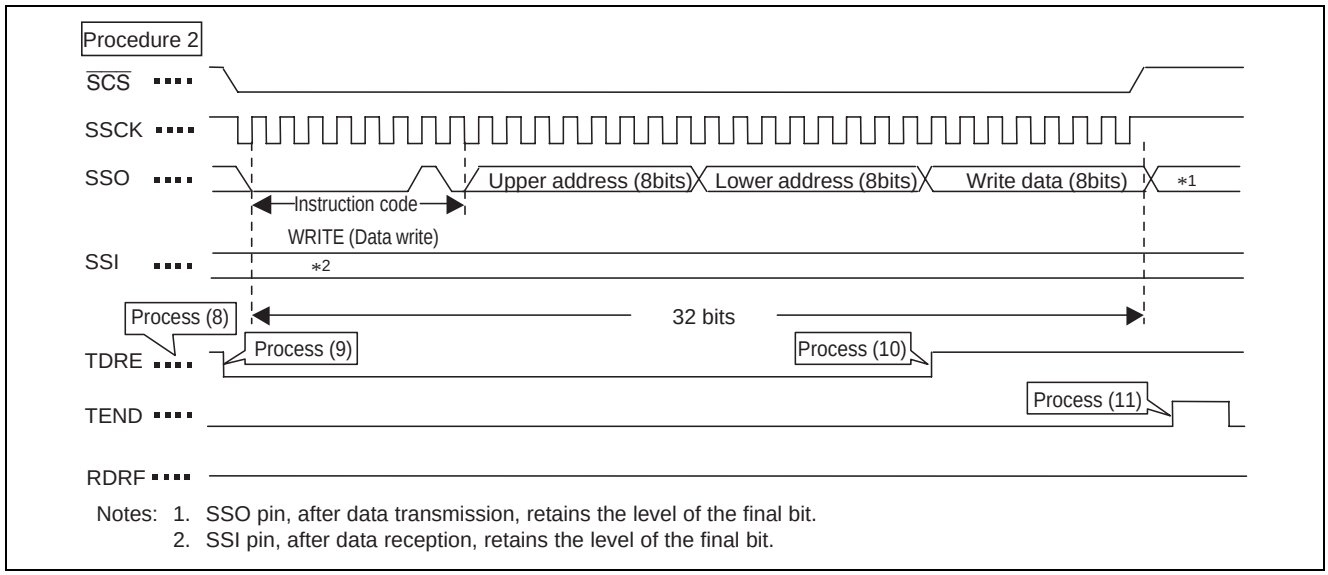
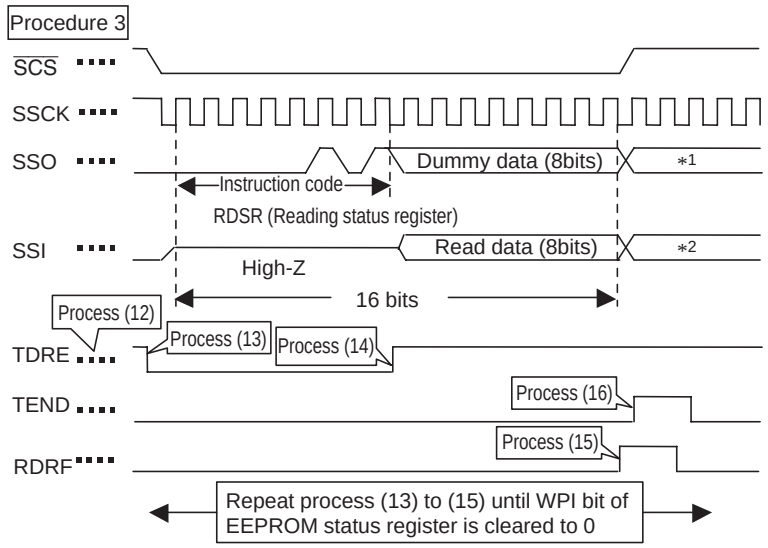


Figure 4 Communication and Operation (Data Write)

Table 7 Description of Software/Hardware Process (Data Write)

	Software Process	Hardware Process
Process (8)	• Sets DATS bit to 2 for setting transmit data length to 32bits. • Sets TE bit to 1 for enabling transmission	• None
Process (9)	• Confirms that TDRE=1 • Sets transmit data (write code) to SSTDR0 register • Sets transmit data (upper address) to SSTDR1 register • Sets transmit data (lower address) to SSTDR2 register • Sets transmit data (write data) to SSTDR3 register	• Clears TDRE bit to 0, when transmit data are set to SSTDR0-3 register
Process (10)	• None	• Transfers data from SSTDR3 register to SSTRSR register and then sets TDRE bit to 1.
Process (11)	• Confirms that TEND=1 (transmission end) and then clears TEND bit to 0. • Confirms that TEND is 0 cleared • Waits for one bit period to elapse • Clears TE bit to 0 for prohibiting transmission	• Transmits the final bit while TDRE=1, and then sets TEND bit to 1



Notes: 1. SSO pin, after data transmission, retains the level of the final bit.
2. SSI pin, after data reception, retains the level of the final bit.

Figure 5 Communication and Operation (Data Write End Confirmation)

Table 8 Description of Software/Hardware Process (Data Write End Confirmation)

	Software Process	Hardware Process
Process (12)	• Sets DATS bit to 1 for setting transmit data length to 16bits. • Sets TE bit and RE bit to 1 for enabling transmission and reception	• None
Process (13)	• Confirms that TDRE=1 • Sets transmit data (status reading code) to SSTDR0 register • Sets transmit data (dummy data) to SSTDR1 register	• Clears TDRE bit to 0 when transmit data are set to SSTDR0 – 1 register.
Process (14)	• None	• Transfers data from SSTDR1 register to SSTRSR register and then sets TDRE bit to 1.
Process (15)	• Confirms that RDRF=1 and ORER≠1, and then transfers values of SSRDR0 – 1 register to RAM	• Transfers reception data from SSTRSR register to SSRDR1 register and then sets RDRE bit to 1. • RDRF bit is cleared to 0 when values of SSRDR0 – 1 register are read by CPU.
Process (16)	• Confirms that TEND=1 (transmission end) and then clears TEND bit to 0 • Waits for one bit period to elapse • If 1bit (EEPROM/WEL bit) of SSRDR2 register is 0 (write end), TE bit and RE bit is cleared to 0 for prohibiting transmission/reception. If 1bit (EEPROM/WEL bit) of SSRDR2 register is not 0 (write not end), returned to process (13)	• Transmits the final bit while TDRE=1, and then sets TEND bit to 1

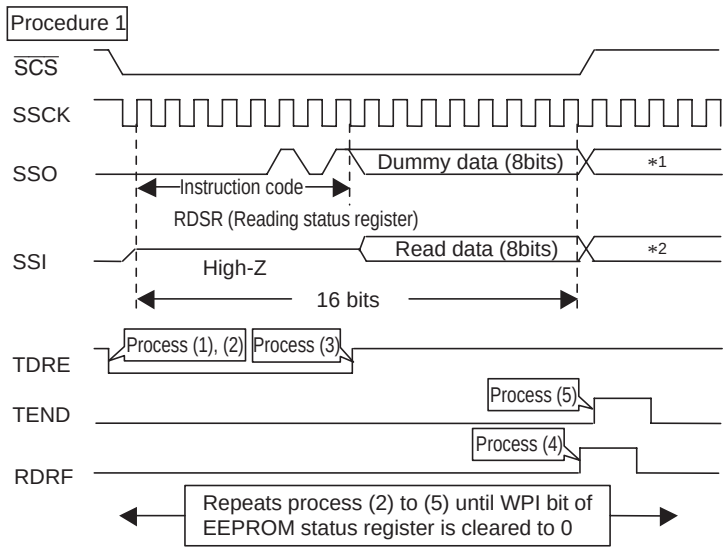
(1) Reading data from EEPROM

Procedure 1: Data Write End Confirmation

- Confirm that Write process have ended by transmitting RDSR (reading status register) code to confirm WIP bit of status register is cleared 0.
- Figure 6 describes the communication and the operation. shows description of software/ hardware process.

Procedure 2: Data Read

- Transfer WREN (Write enable) code to read data from EEPROM. Then, transfer read data to RAM.
- Figure 7 describes the communication and operation. shows description of software/ hardware process.

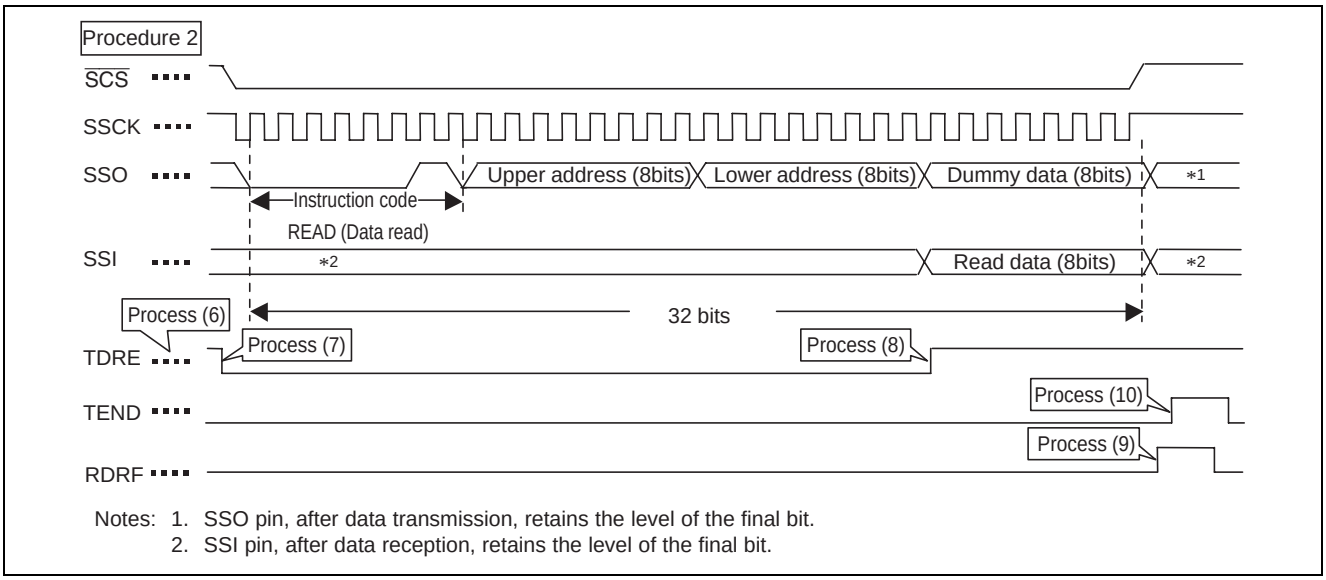


Notes: 1. SSO pin, after data transmission, retains the level of the final bit.
2. SSI pin, after data reception, retains the level of the final bit.

Figure 6 Communication and Operation (Data Write End Confirmation)

Table 9 Description of Software/Hardware Process (Data Write End Confirmation)

	Software Process	Hardware Process
Process (1)	• Sets DATS bit to 1 and transmit data length to 16-bit • Sets TE bit and RE bit for enabling transmission/reception	• None
Process (2)	• Confirms that TDRE=1 • Sets transmit data (status reading code) to SSTDR0 register • Sets transmit data (dummy data) to SSTDR1 register	• TDRE bit is cleared to 0 when transmit data are set to SSTDR0-1 register.
Process (3)	• None	• Transfers data from SSTDR1 register to SSTRSR register, and then sets TDRE bit to 1
Process (4)	• Confirms that RDRF=1 and ORER≠1, and then transfers values of SSRDR0-1 register to RAM • Confirms that RDRF=0	• Transfers receive data from SSTRSR register to SSRDR1 register, and then sets RDRF bit to 1. • RDRF bit is cleared to 0 when values of SSRDR0 – 1 register are read by CPU.
Process (5)	• Confirms that TDRE bit and TEND bit are set 1 (transmission end) and then clear TEND bit to 0 • Waits for one bit period to elapse • If 0bit (EEPROM/WIP bit) of SSRDR2 register is 0 (write end), TE bit and RE bit is cleared to 0 for prohibiting transmission/reception. If 0bit (EEPROM/WIP bit) of SSRDR2 register is not 0 (write not end), returned to process (2)	• Transmits the final bit while TDRE=1, and then sets TEND bit to 1.



Notes: 1. SSO pin, after data transmission, retains the level of the final bit.
2. SSI pin, after data reception, retains the level of the final bit.

Figure 7 Communication and Operation (Data Read)

Table 10 Description of Software/Hardware Process (Data Read)

	Software Process	Hardware Process
Process (6)	• Sets DATS bit to 2 for setting transmit data length to 32bits. • Sets TE bit and RE bit to 1 for enabling transmission/reception	• None
Process (7)	• Confirms that TDRE = 1 • Sets transmit data (read code) to SSTDR0 register • Sets transmit data (upper address) to SSTDR1 register • Sets transmit data (lower address) to SSTDR2 register • Sets transmit data (dummy data) to SSTDR3 register	• TDRE bit is 0 cleared when transmit data are set to SSTDR0-3 register.
Process (8)	• None	• Transfers data from SSTDR3 register to SSTRSR register, and then sets TDRE bit to 1.
Process (9)	• Confirms that RDRF=1 and ORER≠1, and then transfers values of SSRDR0-3 register to RAM	• Transfers receive data from SSTRSR register to SSRDR3 register, and then sets RDRF bit to 1. • RDRF bit is cleared to 0 when values of SSRDR0 – 3 register are read by CPU.
Process (10)	• Confirms that TEND bit are set 1 (transmission end) and then clear TEND bit to 0 • Waits for one bit period to elapse • Clears TE bit and RE bit to 0 for prohibiting transmission/reception.	• Transmits the final bit while TDRE=1, and then sets TEND bit to 1.

3.3 Sample Program Procedure

(1) Following describes register setting used in this sample task (data write to EEPROM/ data read from EEPROM).

Table 11 shows register setting for clock pulse generator (CPG), power-down mode, and PFC.

Table 12 shows register setting for Synchronous Serial Communication Unit.

Table 11 Setting of Used Registers (CPG, Power-down Mode, PFC)

Module	Register Name	Bit Name	Bit	Function	Set Value
CPG	Frequency Control Register (FRQCR)	Reserved	15	—	0
		IFC[2:0]	14-12	Division ratio of the internal clock (Iϕ) frequency 0: ×1, when the input clock is 8MHz Iϕ: 64MHz	0
		BFC[2:0]	11-9	Division ratio of the bus clock (Bϕ) frequency 1: ×1/2, when the input clock is 8MHz Bϕ: 32MHz	1
		PFC[2:0]	8-6	Division ratio of the peripheral clock (Pϕ) frequency 1: ×1/2, when the input clock is 8MHz Pϕ: 32MHz	1
		MIFC[2:0]	5-3	Division ratio of the MTU2S clock (MIϕ) frequency 0: ×1, when the input clock is 8MHz MIϕ: 64MHz	0
		MPFC[2:0]	2-0	Division ratio of the MTU2 clock (MPϕ) frequency 1: ×1/2, when the input clock is 8MHz MPϕ: 32MHz	1
Power-down Mode	Standby Control Register (STBCR3)	Reserved	7-6	—	1
		MSTP13	5	1: Stops clock supply to SCI_2 (SCI_2 is not operating)	1
		MSTP12	4	1: Stops clock supply to SCI_1 (SCI_1 is not operating)	1
		MSTP11	3	1: Stops clock supply to SCI_0 (SCI_0 is not operating)	1
		MSTP10	2	0: SSU in operation	0
		Reserved	1	—	1
		MSTP8	0	1: Stops clock supply to RCAN-ET (RCAN-ET is not operating)	1
PFC	Port D Control Register L3 (PDCRL3)	Reserved	15-11	—	0
		PD10MD[2:0]	10-8	101: Sets SSO input/output (SSU)	101
		Reserved	7	—	0
		PD9MD[2:0]	6-4	101: Sets SSI input/output (SSU)	101
		Reserved	3	—	0
		PD8MD[2:0]	2-0	101: Sets SSCK input/output (SSU)	101
	Port D Control Register L2 (PDCRL2)	Reserved	15	—	0
		PD7MD[2:0]	14-12	101: Sets _SCS input/output (SSU)	101
		Reserved	11	—	0
		PD6MD[2:0]	10-8	0: Sets PD6 input/output (general port)	0
		Reserved	7	—	0
		PD5MD[2:0]	6-4	0: Sets PD5 input/output (general port)	0
		Reserved	3	—	0
		PD4MD[2:0]	2-0	0: Sets PD4 input/output (general port)	0
	Port D I/O Register L (PDIORL)	Reserved	15-11	—	0
		PD10IOR	10	1: PD10 pin is output	1
		PD9IOR	9	1: PD9 pin is input	0
		PD8IOR	8	1: PD8 pin is output	1
		PD7IOR	7	1: PD7 pin is output	1
		PD6IOR	6	1: PD6 pin is input	0
		PD5IOR	5	1: PD5 pin is input	0
		PD4IOR	4	1: PD4 pin is input	0
		PD3IOR	3	1: PD3 pin is input	0
		PD2IOR	2	1: PD2 pin is input	0
		PD1IOR	1	1: PD1 pin is input	0
		PD0IOR	0	1: PD0 pin is input	0

Table 12 Setting of Used Registers (Synchronous Serial Communication Unit)

Module	Register Name	Bit Name	Bit	Function	Set Value
Synchronous Serial Communication Unit	SS Control Register H (SSCRH)	MSS	7	1: Master Mode	1
		BIDE	6	0: Normal Mode	0
		Reserved	5	—	0
		SOL	4	0: Changes the serial data output to a low level.	0
		SOLP	3	1: Changes output level depending on SOL bit value	1
		Reserved	2	—	1
		CSS[1:0]	1-0	11: $\overline{SCS}$ is used as the automatic output function	11
	SS Control Register L (SSCRL)	FCLRM	7	1: Clears the interrupt flag when the register is accessed	1
		SSUMS	6	0: SSU Mode	0
		SRES	5	Setting 1 resets SSU internal sequencer	0
		Reserved	4-2	—	0
		DATS[1:0]	1-0	00: 8-bit data length	00
				01: 16-bit data length	01
				10: 32-bit data length	10
				11: Setting prohibited	
	SS Mode Register (SSMR)	MLS	7	1: MSB-first	1
		CPOS	6	0: Outputs clock polarity high in the idle state and low in the active state.	0
		CPHS	5	0: Changes the data at the first edge of the clock	0
		Reserved	4-3	—	0
		CKS[2:0]	2-0	011: Transfer clock rate → Pϕ/16	011
	SS Enable Register (SSER)	TE	7	0: Disables transmit operation	0
				1: Enables transmit operation	1
		RE	6	0: Disables receive operation	0
				1: Enables receive operation	1
		Reserved	5-4	—	0
		TEIE	3	0: Disables SSTEI (transmit end) interrupt	0
		TIE	2	0: Disables SSTXI (transmit data empty) interrupt	0
		RIE	1	0: Disables SSRXI (receive data full) interrupt and SSOEI (overrun error) interrupt	0
		CEIE	0	0: Disables SSCEI (conflict error) interrupt	0
	SS Status Register (SSSR)	Reserved	7	—	0
		ORER	6	1: Overrun error occurred	0
				[Set condition] Next 1 byte reception ended while RDRF=1	
				[Clear condition] 0 write after reading ORER=1	
		Reserved	5-4	—	0
		TEND	3	1: Transmit end	0
				[Set conditions]	1
				• When TENDSTS=0: when the final bit of transmit data is transmitted, while TDRE=1	
				• When TENDSTS=1: after the final bit of transmit data have been transmitted, while TDRE=1	
				[Clear conditions]	
				• 0 write after reading TEND=1	
				• Write data to SSTDR	

Module	Register Name	Bit Name	Bit	Function	Set Value
Synchronous Serial Communication Unit	SS Status Register (SSSR)	TDRE	2	1: SSTDR data empty [Set conditions] • TE=0 • When SSTDR became data writable by data transfer from SSTDR to SSTRSR [Clear conditions] • Write 0 to TDRE flag after reading TDRE=1 • Write to SSTDR while TE=1 • Write transmit data to SSTDR when DTC is activated by SSTXI interrupt and its DISEL bit is 0.	0 1
	SS Status Register (SSSR)	RDRF	1	1: SSRDR data full [Set condition] Serial reception ended properly and receive data is transferred from SSTRSR to SSRDR [Clear conditions] • 0 write to RDRF flag after reading RDRF=1 • Read receive data from SSRDR • Read receive data from SSRDR when DTC is activated by SSRXI interrupt and its DISEL bit is 0.	0 1
		CE	0	1: Conflict error/ incomplete error occurred	0
	SS Control Register 2 (SSCR2)	Reserved	7-5	—	0
		TENDSTS	4	1: Sets the TEND bit after transmission of the final bit.	1
		SCSATS	3	Selects the timing for asserting $\overline{SCS}$ pin 0: Minimum of t_{LEAD}, t_{LAG} output period is $1/2 \times t_{SUcyc}$ Note: Please refer AC Characteristics section in the hardware manual for t_{LEAD}, t_{LAG}, t_{SUcyc}	0
		SSODTS	2	Selects the timing for outputting data from the SSO pin 0: SSO pin outputs data when BIDE=0, MSS=1, TE=1, or when BIDE=1, TE=1, RE=0	0
		Reserved	1-0	—	0
	SS Transmit Data Register 0 to 3 (SSTDR0 - 3)	—	7-0	This register is an 8-bit register that stores transmit data.	H'00 (initial value)
	SS Receive Data Register 0 to 3 (SSRDR0 - 3)	—	7-0	This register is an 8-bit register that stores receive data.	H'00 (initial value)

(2) Writing Flow

Figure 8 to Figure 13 describes process flow for writing to EEPROM.

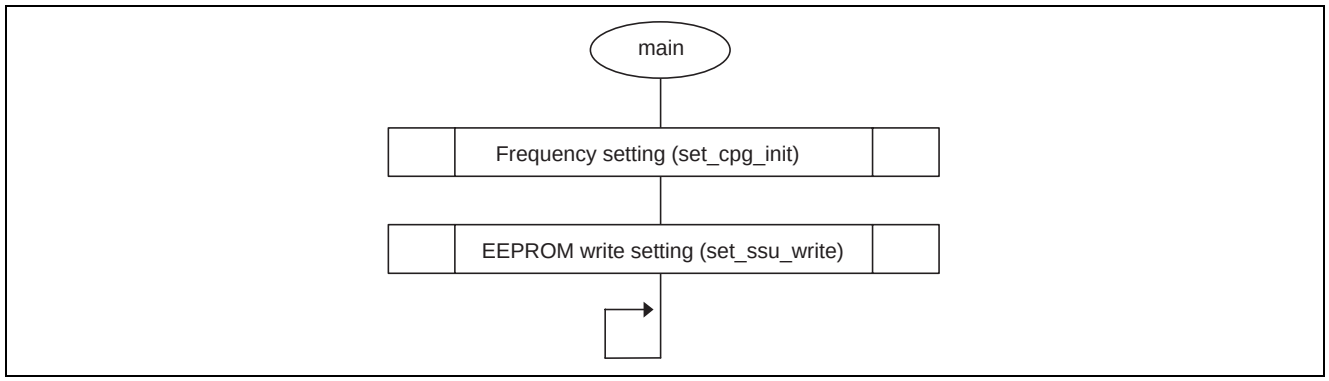


Figure 8 Main Flow (Data Write to EEPROM)

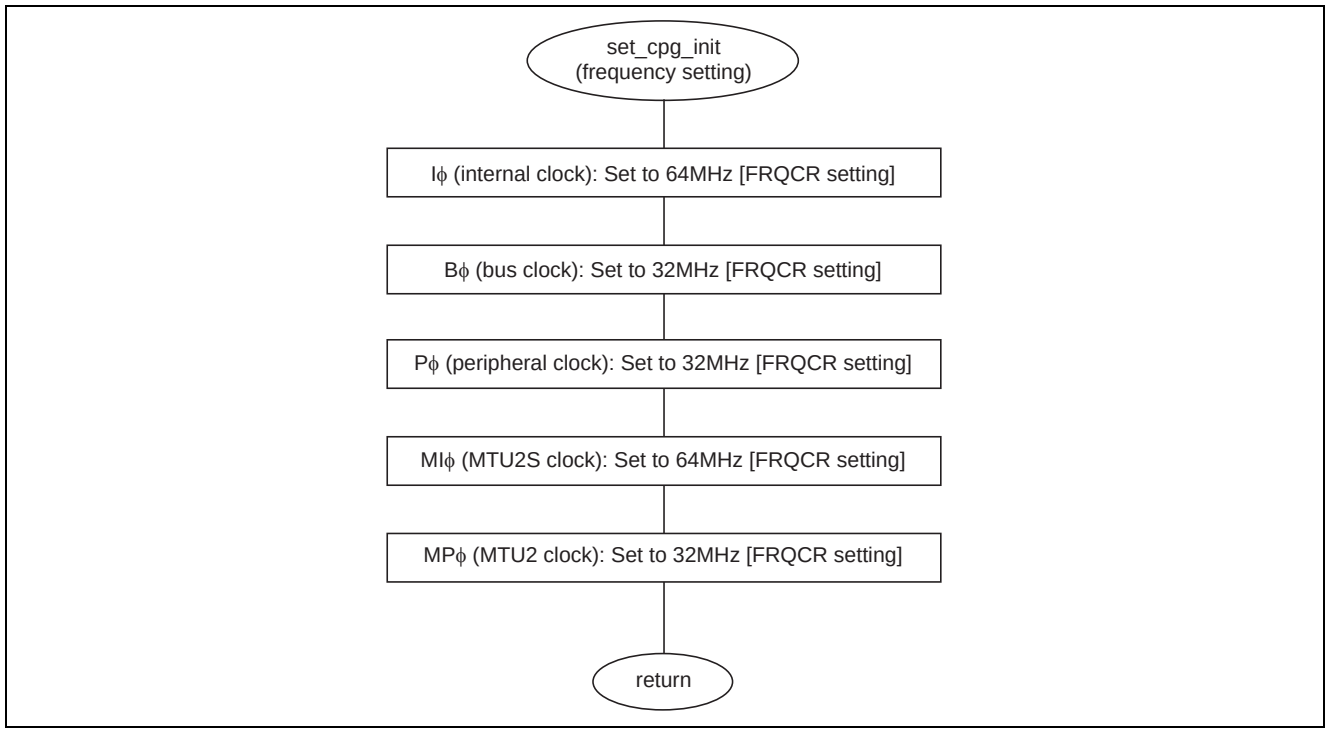


Figure 9 Frequency Setting Flow

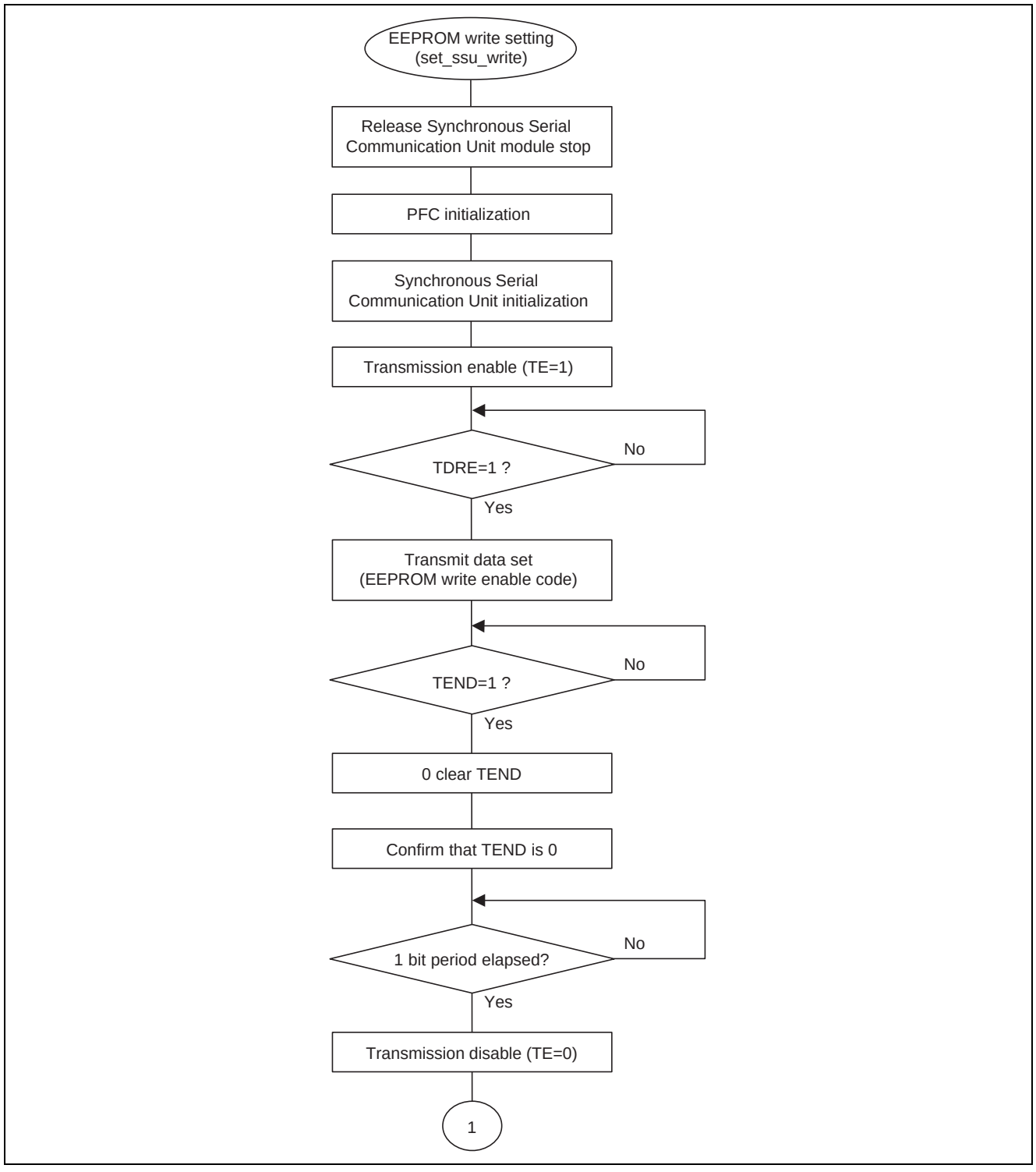


Figure 10 Data Writing Flow (1)

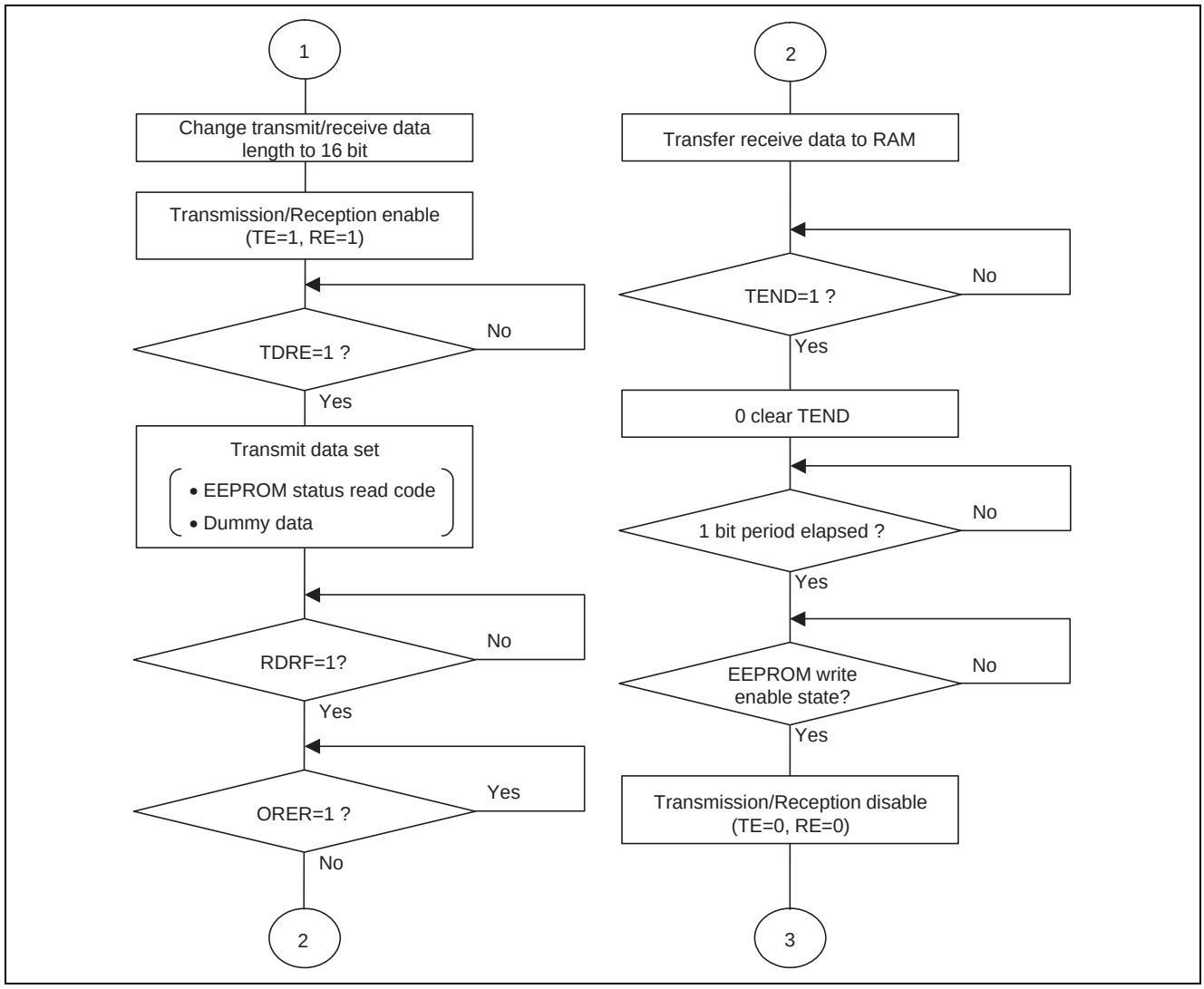


Figure 11 Data Writing Flow (2)

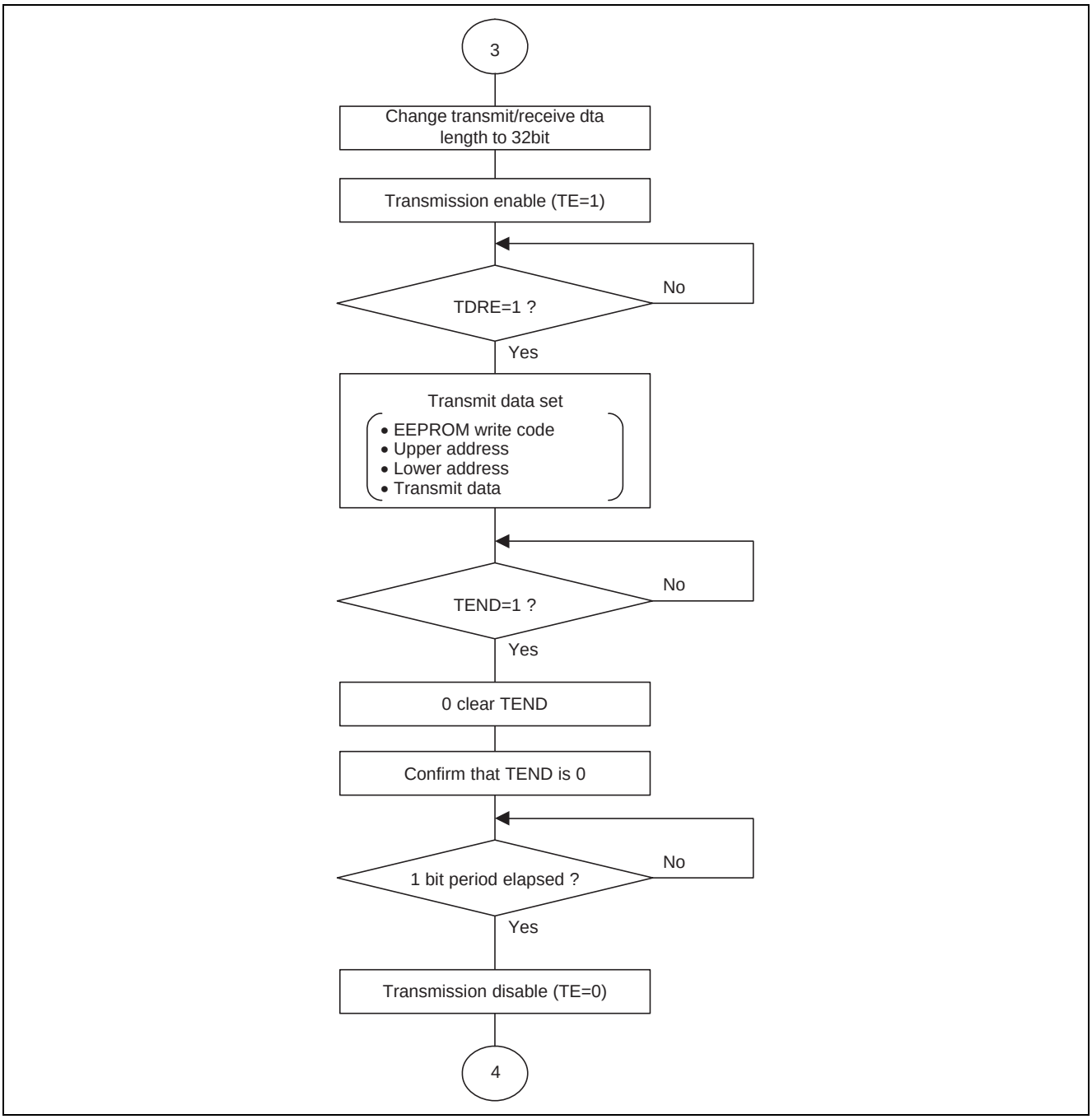


Figure 12 Data Writing Flow (3)

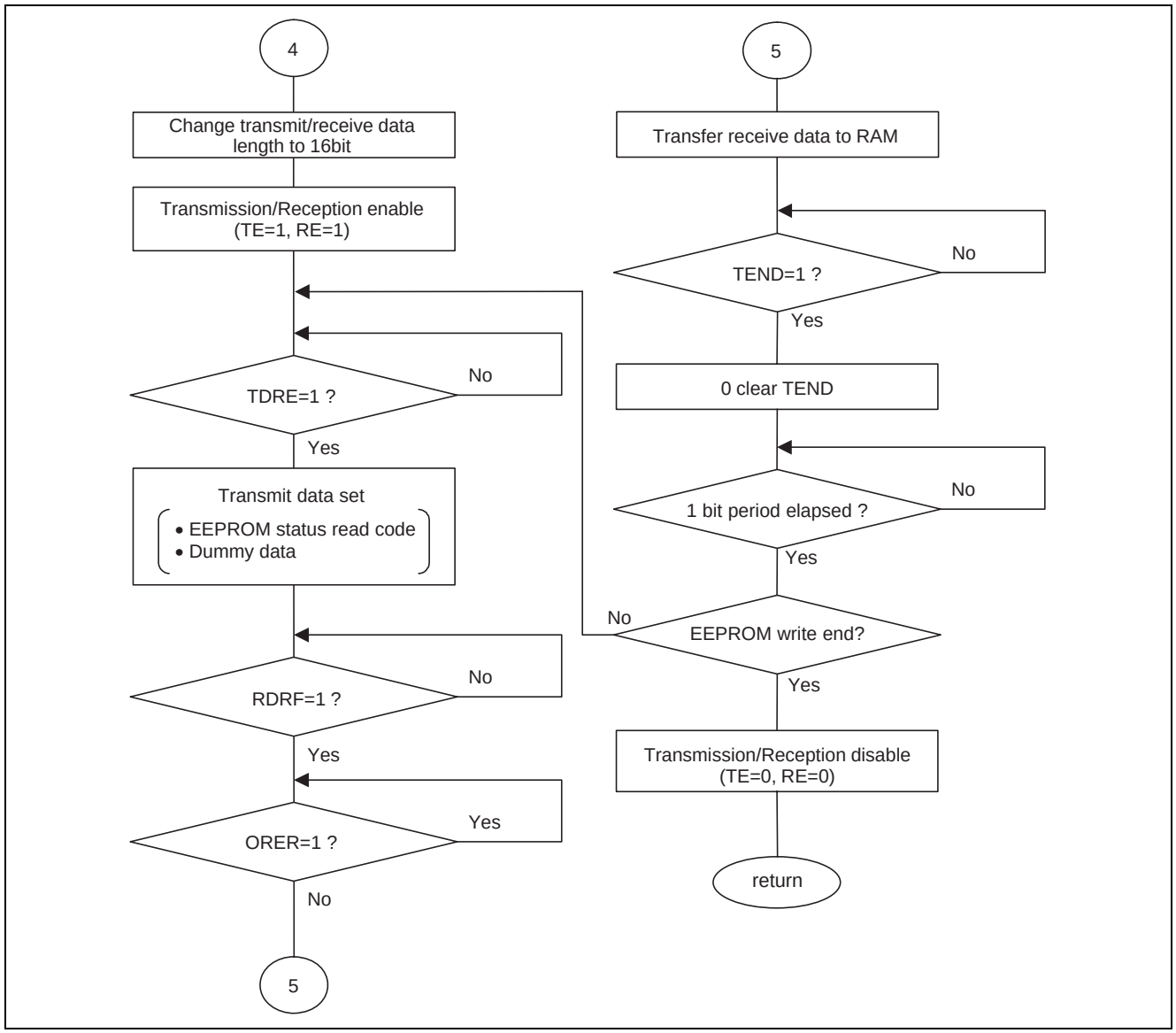


Figure 13 Data Writing Flow (4)

(3) Reading Flow

Figure 14 to Figure 17 describes process flow for reading from EEPROM.

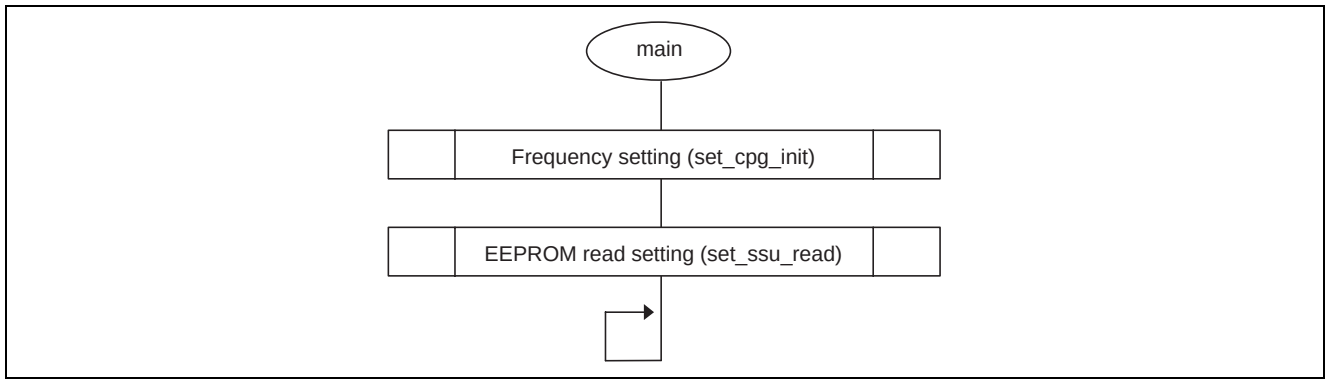


Figure 14 Main Flow (Data Read from EEPROM)

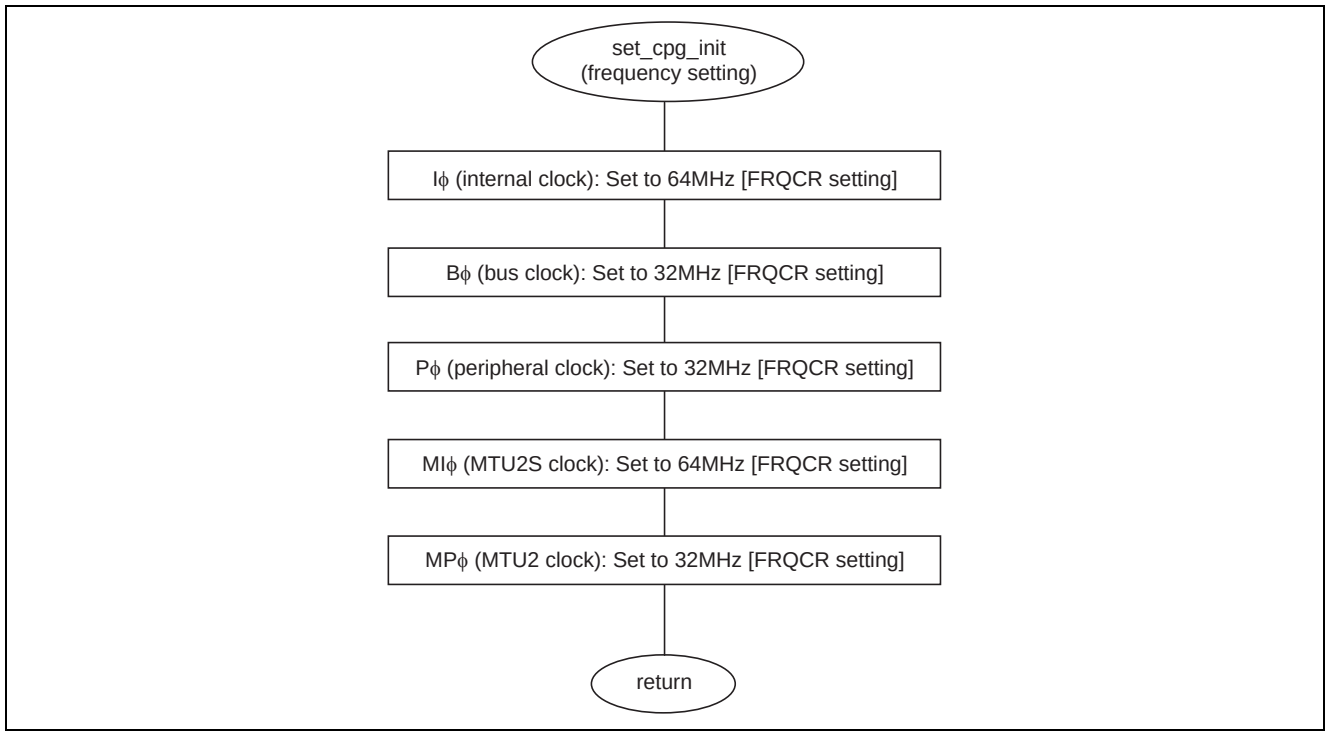


Figure 15 Frequency Setting Flow

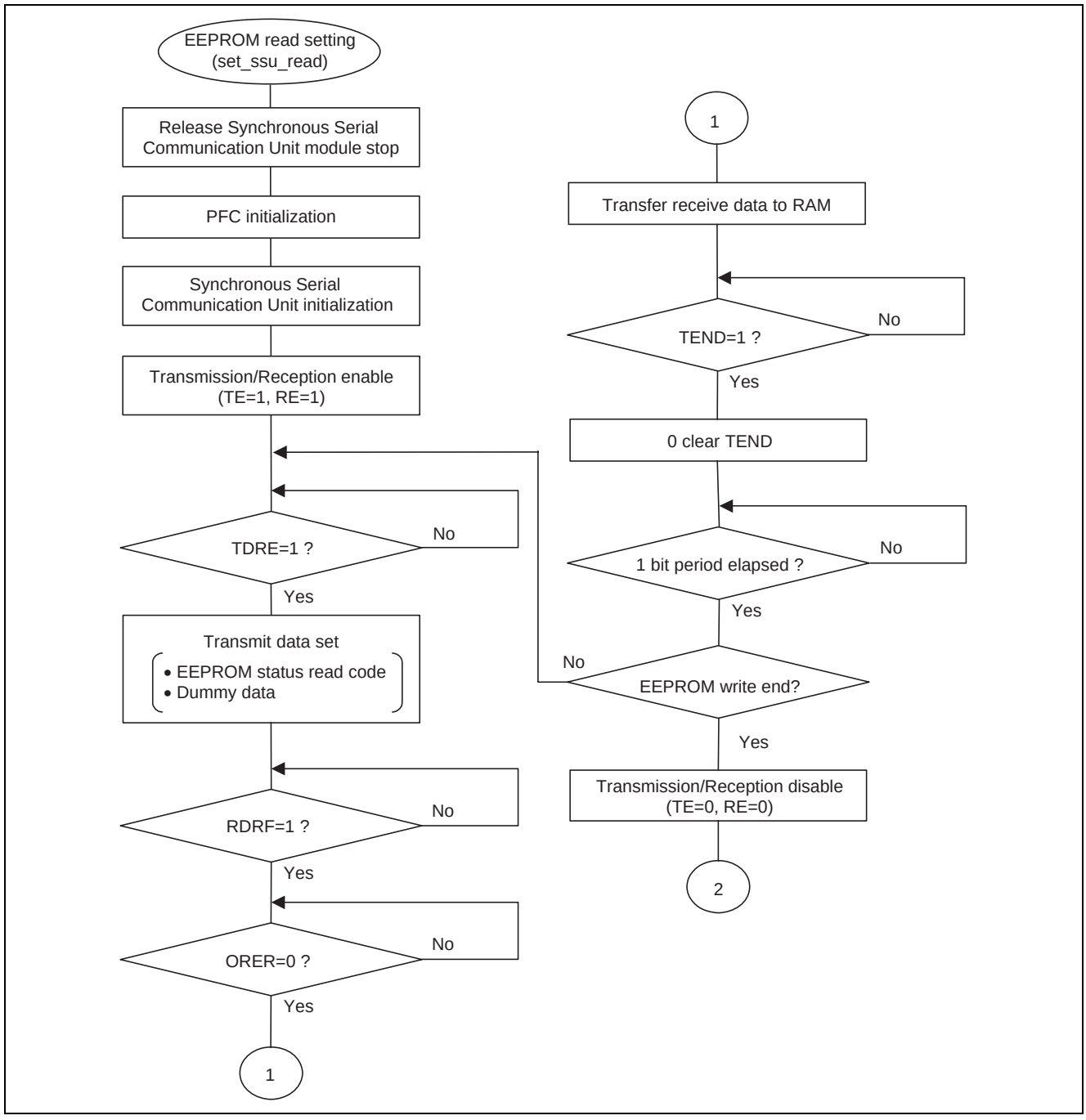


Figure 16 Data Reading Flow (1)

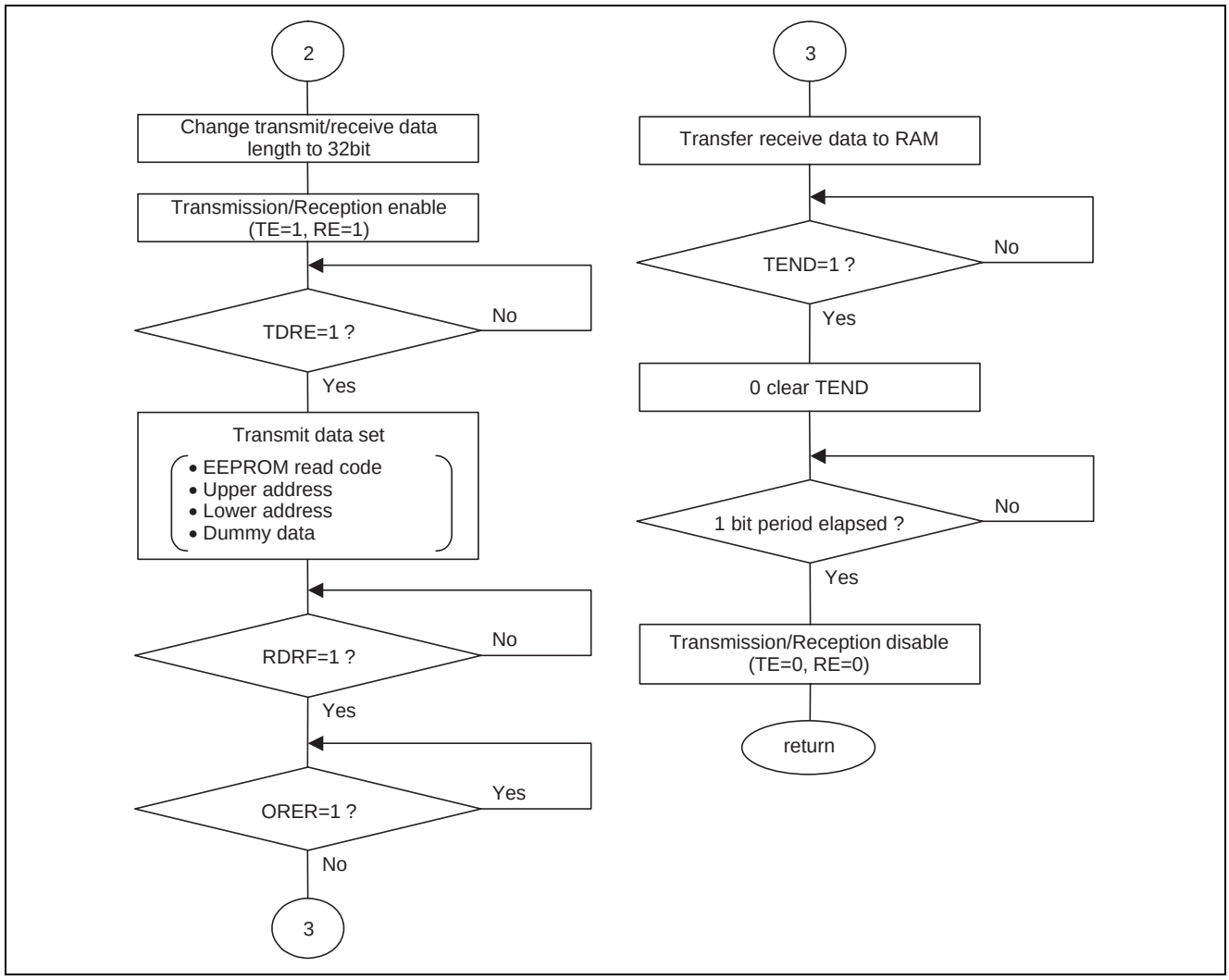


Figure 17 Data Reading Flow (2)

4. Sample Program

(1) Data Write to EEPROM via Synchronous Serial Communication Unit Communication

```
/*********************************************************************/
/*  SH7147                                                           */
/*  EEPROM Write via Synchronous Serial Communication Unit Communication */
/*********************************************************************/

/*------------ Include File-----------------------------------------*/
#include <machine.h>
#include"iodefine.h"

/*********************************************************************/
/*      Function Declaration                                         */
/*********************************************************************/
void main(void);                  /* Main Routine */
void set_cpg_init(void);          /* Each Clock Setting */
void set_ssu_write(void);         /* EEPROM Write Setting */
void wait_1bit(void);             /* Wait 1 Bit Period */

/*********************************************************************/
/*      Main Routine                                                 */
/*********************************************************************/
void main(void){

    set_cpg_init();               /* Each Clock Setting */
    set_ssu_write();              /* EEPROM Write Setting */

    while(1);
}

/*********************************************************************/
/*      Variable Declaration                                         */
/*********************************************************************/

    unsigned char write_data;     /* Write Data */
    unsigned char upper_address;  /* Write Address Upper Address */
    unsigned char under_address;  /* Write Address Lower Address */
    unsigned char dummy_data;     /* Reception Data (Dummy Data) Storing RAM */
    unsigned char status_flag1;   /* Reception Data (Status Register) Storing RAM */
    unsigned char status_flag2;   /* Reception Data (Status Register) Storing RAM */

/*********************************************************************/
/*      Each Clock Setting                                           */
/*      Iφ:Bφ:Pφ:MIφ:MPφ = 64MHz:32MHz:32MHz:64MHz:32MHz             */
/*********************************************************************/
void set_cpg_init(void){

    CPG.FRQCR.BIT.IFC = 0;        /* Iφ=64MHz */
    CPG.FRQCR.BIT.BFC = 1;        /* Bφ=32MHz */
    CPG.FRQCR.BIT._PFC = 1;       /* Pφ=32MHz */
    CPG.FRQCR.BIT.MIFC = 0;       /* MIφ=64MHz */
    CPG.FRQCR.BIT.MPFC = 1;       /* MPφ=32MHz */
```

```
}

/*****************************************************************************/
/*      Set 1 bit period (500 ns) wait.                                      */
/*****************************************************************************/
void wait_1bit(void){

    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
}

/*****************************************************************************/
/*      Synchronous Serial Communication Unit Write Setting Routine          */
/*****************************************************************************/
void set_ssu_write(void){

    /* RAM Initialization*/
    dummy_data = 0x00;
    status_flag1 = 0x00;
    status_flag2 = 0xFF;

    /* Write Data Setting */
    write_data = 0x55;
```

```
/* Write Address Setting */
upper_address = 0x1A;
under_address = 0xAA;

/* Module Standby Releasing */
STB.CR3.BIT._SSU = 0;                   /* 0:active 1:standby */

/* PFC Setting */
PFC.PDCRL3.BIT.PD10MD = 5;              /* Set SSO */
PFC.PDCRL3.BIT.PD9MD = 5;               /* Set SSI */
PFC.PDCRL3.BIT.PD8MD = 5;               /* Set SSCK */
PFC.PDCRL2.BIT.PD7MD = 5;               /* Set SCS */
PFC.PDIORL.BIT.B10 = 1;                 /* PD10(SSO) Output */
PFC.PDIORL.BIT.B9 = 0;                  /* PD9(SSI) Input */
PFC.PDIORL.BIT.B8 = 1;                  /* PD8(SSCK) Output */
PFC.PDIORL.BIT.B7 = 1;                  /* PD7(SCS) Output */

/* Synchronous Serial Communication Unit Module Initialization */
SSU.SSER.BYTE = 0x00;                   /* Transmission/Reception Disable */
SSU.SSCRH.BYTE = 0x8F;
           /* Master Mode, Normal Mode, Low Level Output, Automatic Output */
SSU.SSCRL.BYTE = 0x80;     /* Flag Clear when Register Accessed, SSU Mode
                              Data Length: 8 bit */
SSU.SSMR.BYTE = 0x83;      /* MSB First, Transfer Clock: Pϕ(32MHz)/16 = 2MHz */
SSU.SSCR2.BYTE = 0x10;     /* Set TEND after the final bit is transmitted */

/* EEPROM Write Enable Code Transmit */
SSU.SSER.BIT.TE = 1;                    /* Transmit Enable */
while(SSU.SSSR.BIT.TDRE !=1);           /* Confirm TDRE=1 */
SSU.SSTDR0 = 0x06;                      /* Transmit Data (Write Enable Code) Set */
while(SSU.SSSR.BIT.TEND !=1);           /* Confirm TEND=1 */
SSU.SSSR.BIT.TEND = 0;                  /* Clear TEND */
while(SSU.SSSR.BIT.TEND !=0);           /* Confirm TEND=0 */
wait_1bit();                            /* Wait 1 Bit Period */
SSU.SSER.BIT.TE = 0;                    /* Transmission Disable */


/* EEPROM Status Flag Check (Confirm that it is in Write Enable State)*/
SSU.SSCRL.BIT.DATS = 1;                 /* Data Length: 16bit */
SSU.SSER.BYTE = 0xC0;                   /* Transmission/Reception Enable TE=1.RE=1 */
while(SSU.SSSR.BIT.TDRE !=1);           /* Confirm TDRE=1 */
SSU.SSTDR0 = 0x05;                      /* Transmit Data (Status Read Code) Set */
SSU.SSTDR1 = 0x00;                      /* Transmit (Dummy Data) Set */
while(SSU.SSSR.BIT.RDRF !=1);           /* Confirm RDRF=1 */
while(SSU.SSSR.BIT.ORER ==1);           /* Confirm ORER≠1 */
dummy_data = SSU.SSRDR0;                /* Dummy Data */
status_flag1 = SSU.SSRDR1;              /* Receive Data(Status Register) */
while(SSU.SSSR.BIT.TEND !=1);           /* Confirm TEND=1 */
SSU.SSSR.BIT.TEND = 0;                  /* Clear TEND */
while(SSU.SSSR.BIT.TEND !=0);           /* Confirm TEND=0 */
wait_1bit();                            /* Wait 1 Bit Period */
while((status_flag1 & 0x02) != 0x02);   /* EEPROM Write Enable State? */
SSU.SSER.BYTE = 0x00;                   /* Transmission/Reception Disable */
```

```
    /* EEPROM Write Data Transmit */
    SSU.SSCRL.BIT.DATS = 2;                 /* Data Length: 32 bit */
    SSU.SSER.BIT.TE = 1;                    /* Transmission Enable */
    while((SSU.SSSR.BIT.TDRE & 1) !=1);     /* Confirm TDRE=1 */
    SSU.SSTDR0 = 0x02;                      /* Transmit Data(Write code) Set */
    SSU.SSTDR1 = upper_address;             /* Transmit Data(Upper address) Set */
    SSU.SSTDR2 = under_address;             /* Transmit Data(Lower address) Set */
    SSU.SSTDR3 = write_data;                /* Transmit Data(Write data) Set */
    while(SSU.SSSR.BIT.TEND !=1);           /* Confirm TEND=1 */
    SSU.SSSR.BIT.TEND = 0;                  /* Clear TEND */
    while(SSU.SSSR.BIT.TEND !=0);           /* Confirm TEND=0 */
    wait_1bit();                            /* Wait 1 Bit Period */
    SSU.SSER.BIT.TE = 0;                    /* Transmission Disable */

    /* EEPROM Status Flag Check (Confirm that write ended) */
    SSU.SSCRL.BIT.DATS = 1;                 /* Data Length: 16 bit */
    SSU.SSER.BYTE = 0xC0;                   /* Transmission/Reception Enable TE=1,RE=1 */
    do {
        while(SSU.SSSR.BIT.TDRE !=1);       /* Confirm TDRE=1 */
        SSU.SSTDR0 = 0x05;                  /* Transmit Data(Status read code) Set */
        SSU.SSTDR1 = 0x00;                  /* Transmit Data(Dummy Data) Set */
        while(SSU.SSSR.BIT.RDRF !=1);       /* Confirm RDRF=1 */
        while(SSU.SSSR.BIT.ORER ==1);       /* Confirm ORER≠1 */
        dummy_data = SSU.SSRDR0;            /* Dummy Data */
        status_flag2 = SSU.SSRDR1;          /* Receive Data(Status Register) */
        while(SSU.SSSR.BIT.TEND !=1);       /* Confirm TEND=1 */
        SSU.SSSR.BIT.TEND = 0;              /* Clear TEND */
        while(SSU.SSSR.BIT.TEND !=0);       /* Confirm TEND=0 */
        wait_1bit();                        /* Wait 1 Bit Period */
    } while ((status_flag2 & 0x01) != 0x00);
    SSU.SSER.BYTE = 0x00;                   /* Transmission/Reception Disable */

}
```

(2) Data Read from EEPROM via Synchronous Serial Communication Unit Communication

```
/*********************************************************************/
/*  SH7147                                                           */
/*  EEPROM Read via Synchronous Serial Communication Unit Communication */
/*********************************************************************/

/*------------ Include File-----------------------------------------*/
#include <machine.h>
#include"iodefine.h"

/*********************************************************************/
/*      Function Declaration                                         */
/*********************************************************************/
void main(void);                  /* Main Routine */
void set_cpg_init(void);          /* Each Clock Setting */
void set_ssu_read(void);          /* EEPROM Read Setting */
void wait_1bit(void);             /* Wait 1 Bit Period */

/*********************************************************************/
/*      Variable Declaration                                         */
/*********************************************************************/

unsigned char upper_address;      /* Upper Address for Read */
unsigned char under_address;      /* Lower Address for Read */
unsigned char dummy_data;         /* Receive Data (Dummy Data) Storing RAM */
unsigned char status_flag3;       /* Receive Data (Status Register) Storing RAM */
unsigned char ssu_rx_data0;       /* Receive Data (Dummy Data) Storing RAM */
unsigned char ssu_rx_data1;       /* Receive Data (Dummy Data) Storing RAM */
unsigned char ssu_rx_data2;       /* Receive Data (Dummy Data) Storing RAM */
unsigned char ssu_rx_data3;       /* Receive Data (Read Data) Storing RAM */

/*********************************************************************/
/*      Main Routine                                                 */
/*********************************************************************/
void main(void){

    set_cpg_init();               /* Each Clock Setting */
    set_ssu_read();               /* EEPROM Read Setting */

    while(1);
}

/*********************************************************************/
/*     Each Clock Setting                                            */
/*      Iφ:Bφ:Pφ:MIφ:MPφ = 64MHz:32MHz:32MHz:64MHz:32MHz             */
/*********************************************************************/
void set_cpg_init(void){

    CPG.FRQCR.BIT.IFC = 0;        /* Iφ=64MHz */
    CPG.FRQCR.BIT.BFC = 1;        /* Bφ=32MHz */
    CPG.FRQCR.BIT._PFC = 1;       /* Pφ=32MHz */
    CPG.FRQCR.BIT.MIFC = 0;       /* MIφ=64MHz */
    CPG.FRQCR.BIT.MPFC = 1;       /* MPφ=32MHz */
```

```
}

/*****************************************************************************/
/*      Set 1 bit period (500 ns) wait.                                      */
/*****************************************************************************/
void wait_1bit(void){

    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
    nop();
}

/*****************************************************************************/
/*      Synchronous Serial Communication Unit Read Setting Routine           */
/*****************************************************************************/
void set_ssu_read(void){

    /* RAM Initialization */
    dummy_data = 0x00;          /* Receive Data (Dummy Data) Storing RAM */
    status_flag3 = 0xFF;        /* Receive Data (Status Register) Storing RAM */
    ssu_rx_data0 = 0;           /* Receive Data (Dummy Data) Storing RAM */
    ssu_rx_data1 = 0;           /* Receive Data (Dummy Data) Storing RAM */
    ssu_rx_data2 = 0;           /* Receive Data (Dummy Data) Storing RAM */
    ssu_rx_data3 = 0;           /* Receive Data (Read Data) Storing RAM */
```

```
/* Write Address Setting */
upper_address = 0x1A;
under_address = 0xAA;

/* Module Standby Releasing */
STB.CR3.BIT._SSU = 0;                   /* 0:active 1:standby */

/* PFC Setting */
PFC.PDCRL3.BIT.PD10MD = 5;              /* SSO Set */
PFC.PDCRL3.BIT.PD9MD = 5;               /* SSI Set */
PFC.PDCRL3.BIT.PD8MD = 5;               /* SSCK Set */
PFC.PDCRL2.BIT.PD7MD = 5;               /* SCS Set */
PFC.PDIORL.BIT.B10 = 1;                 /* PD10(SSO) Output */
PFC.PDIORL.BIT.B9 = 0;                  /* PD9(SSI) Input */
PFC.PDIORL.BIT.B8 = 1;                  /* PD8(SSCK) Output */
PFC.PDIORL.BIT.B7 = 1;                  /* PD7(SCS) Output */

/* Synchronous Serial Communication Unit Module Initialization */
SSU.SSER.BYTE = 0x00;                   /* Transmission/Reception Disable */
SSU.SSCRH.BYTE = 0x8F;
           /* Master Mode, Normal Mode, Low Level Output, Automatic Output */
SSU.SSCRL.BYTE = 0x81;        /* Flag Clear when register accessed, SSU Mode,
                                 Data Length: 16 bit */
SSU.SSMR.BYTE = 0x83;         /* MSB First, Transfer Clock: Pϕ(32MHz)/16 = 2MHz */
SSU.SSCR2.BYTE = 0x10;        /* Set TEND bit after the final bit is transmitted */

/* EEPROM Status Flag Check */
SSU.SSER.BYTE = 0xC0;                   /* Transmission/Reception Enable TE=1.RE=1 */
do {
    while(SSU.SSSR.BIT.TDRE !=1);       /* Confirm TDRE=1 */
    SSU.SSTDR0 = 0x05;                  /* Transmit Data(Status read code) Set */
    SSU.SSTDR1 = 0x00;                  /* Transmit Data(Dummy Data) Set */
    while(SSU.SSSR.BIT.RDRF !=1);       /* Confirm RDRF=1 */
    while(SSU.SSSR.BIT.ORER ==1);       /* Confirm ORER≠1 */
    dummy_data = SSU.SSRDR0;            /* Dummy Data */
    status_flag3 = SSU.SSRDR1;          /* Receive Data (Status register) */
    while(SSU.SSSR.BIT.TEND !=1);       /* Confirm TEND=1 */
    SSU.SSSR.BIT.TEND = 0;              /* Clear TEND */
    wait_1bit();                        /* Wait 1 Bit Period */
} while (status_flag3 != 0x00);         /* EEPROM Write ended? */
SSU.SSER.BYTE = 0x00;                   /* Transmission/Reception Disable */

/* EEPROM Data Read */
SSU.SSCRL.BIT.DATS = 2;                 /* Data Length: 32 bit */
SSU.SSER.BYTE = 0xC0;                   /* Transmission/Reception Enable TE=1.RE=1 */
while(SSU.SSSR.BIT.TDRE !=1);           /* Confirm TDRE=1 */
SSU.SSTDR0 = 0x03;                      /* Transmit Data(Read code) Set */
SSU.SSTDR1 = upper_address;             /* Transmit Data(Upper address) Set */
SSU.SSTDR2 = under_address;             /* Transmit Data(Lower address) Set */
SSU.SSTDR3 = 0x00;                      /* Transmit Data(Dummy Data) Set */
while(SSU.SSSR.BIT.RDRF !=1);           /* Confirm RDRF=1 */
while(SSU.SSSR.BIT.ORER ==1);           /* Confirm ORER≠1 */
ssu_rx_data0 = SSU.SSRDR0;              /* Dummy Data */
ssu_rx_data1 = SSU.SSRDR1;              /* Dummy Data */
```

```
    ssu_rx_data2 = SSU.SSRDR2;              /* Dummy Data */
    ssu_rx_data3 = SSU.SSRDR3;              /* Read data from EEPROM */
    while(SSU.SSSR.BIT.TEND !=1);           /* Confirm TEND=1 */
    SSU.SSSR.BIT.TEND = 0;                  /* Clear TEND */
    wait_1bit();                            /* Wait 1 Bit Period */
    SSU.SSER.BYTE = 0x00;                   /* Transmission/Reception Disable */
}
```

5. Reference

Hardware Manual
SH7147 Hardware Manual Rev.2.00

6. Web-site and contact for support

Renesas Web-site
http://www.renesas.com/

Revision History

		Description	
Rev.	Date	Page	Summary
Rev.1.00	2006.10.12	—	First edition issued
Rev.2.00	2007.09.18	—	Second edition issued

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life
   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.