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.

H8/300H Tiny Series

Examples of Communications Operation Using the Built-in SSU

Introduction

This application note describes two examples of communications operation using the synchronous serial communication unit (SSU) incorporated in the H8/36057.

Target Device

H8/36057

Contents

1. SSU Communication System Summary 2
2. Description of Functions 3
3. Description of Operation 5
4. Application Example (Specification 1: Standard Mode) 9
5. Application Example (Specification 2: Bi-Directional Mode) 21

1. SSU Communication System Summary

The synchronous serial communication unit (SSU) performs communication using four buses: chip select ($\overline{SCS}$ pin), clock line (SSCK pin) and data lines (SSI and SSO pins).

1.1 Standard Mode

- In standard mode, the master device transmits the chip select signal via the $\overline{SCS}$ pin, transfer clock via the SSCK pin, and data via the SSO pin to the slave device. The master device uses the SSI pin to receive data from the slave device.
- In standard mode, the slave device receives the chip select signal via the $\overline{SCS}$ pin, transfer clock via the SSCK pin, and data via the SSO pin from the master device. The slave device transmits data via the SSI pin synchronously with the transfer clock provided from the master device.

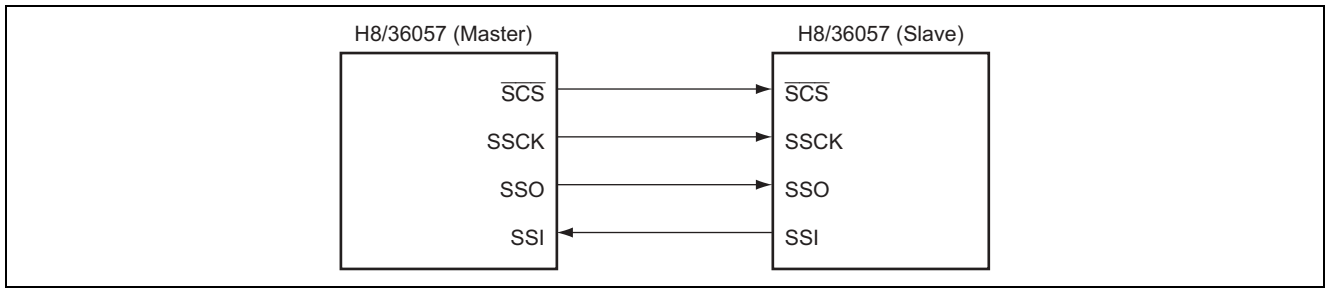


Figure 1 Block Diagram of the SSU Communications between H8/36057 LSIs (Master and Slave in Standard Mode)

1.2 Bi-Directional Mode

- In bi-directional mode, the master device transmits the chip select signal via the $\overline{SCS}$ pin and transfer clock via the SSCK pin, and receives data via the SSO pin.
- In bi-directional mode, the slave device receives the chip select signal and transfer clock from the master device via the $\overline{SCS}$ pin and the SSCK pin, respectively, and transmits and receives data via the SSO pin synchronously with the transfer clock provided from the master device.

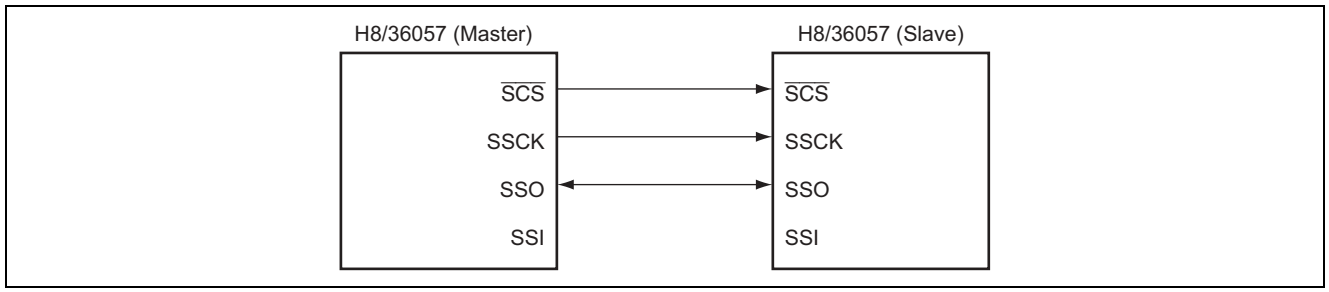


Figure 2 Block Diagram of the SSU Communications between H8/36057 LSIs (Mater and Slave in Bi-Directional Mode)

2. Description of Functions

2.1 Transmitter

Figure 3 shows a block diagram of the SSU transmission functions used in this sample task.

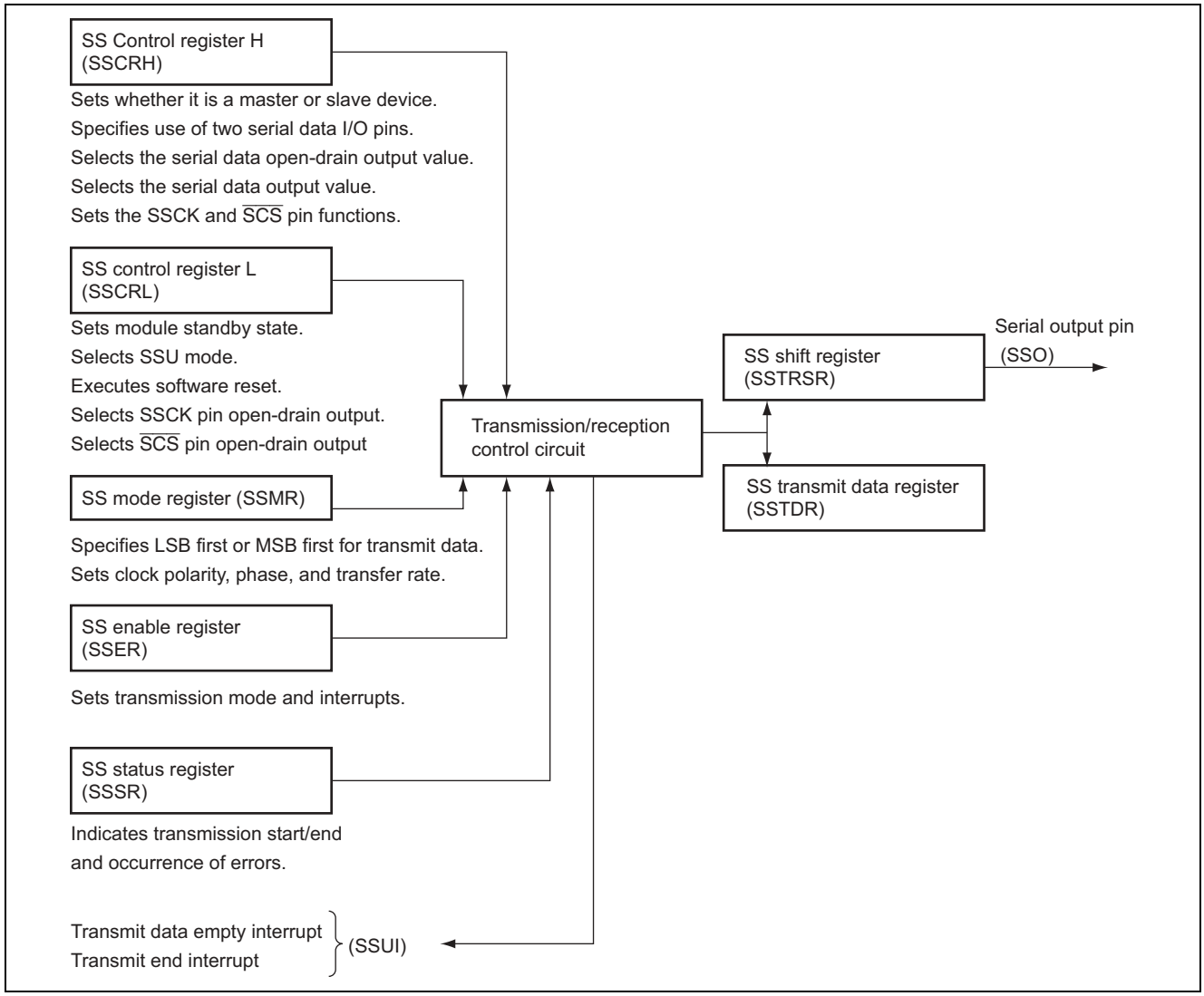


Figure 3 SSU Transmission Function Block Diagram

2.2 Receiver

Figure 4 shows a block diagram of the SSU reception functions used in this sample task.

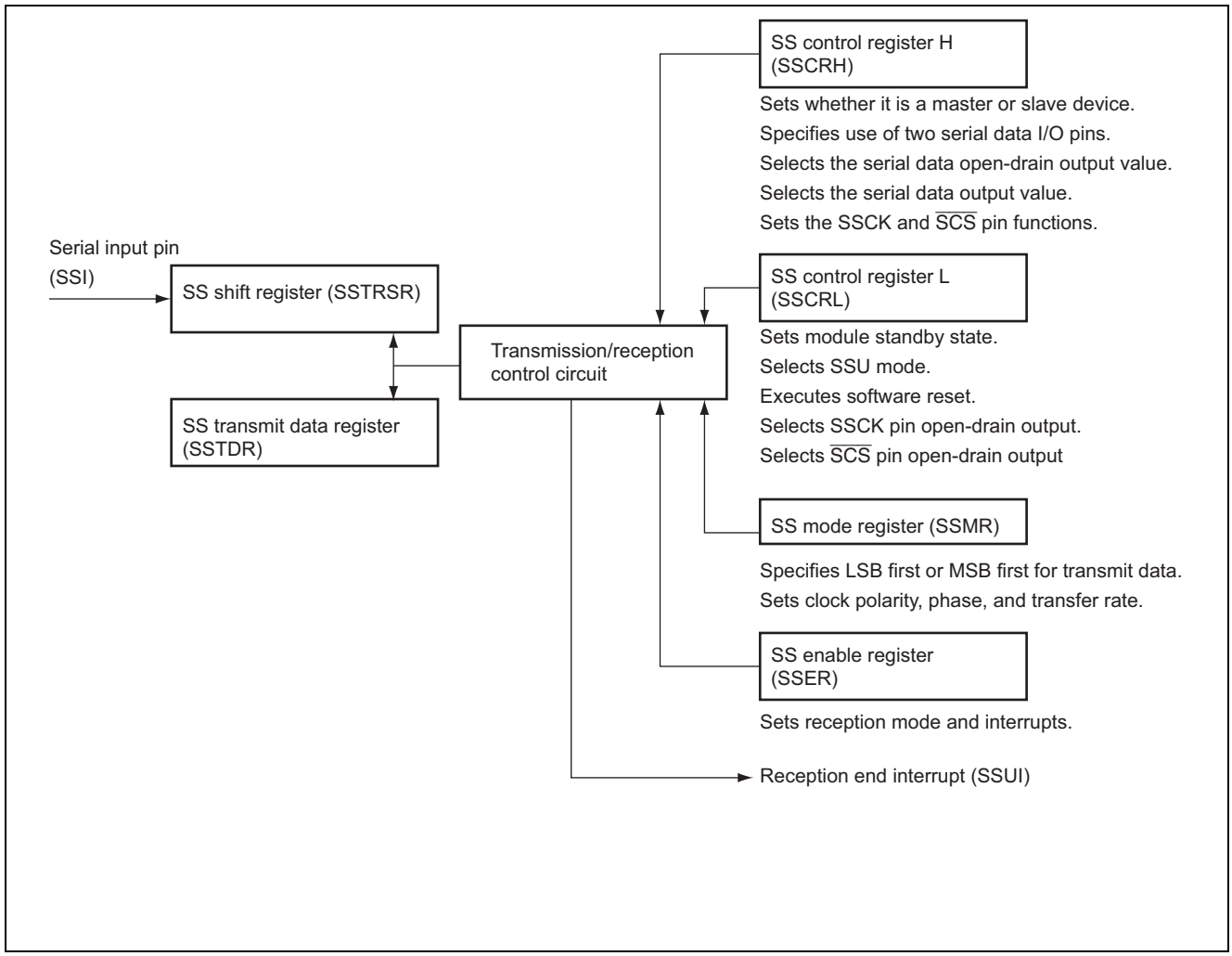


Figure 4 SSU Reception Function Block Diagram

3. Description of Operation

3.1 Transmit Operation

1. Figure 5 is a block diagram of the case when the master H8/36057 transmits data to a slave H8/36057.

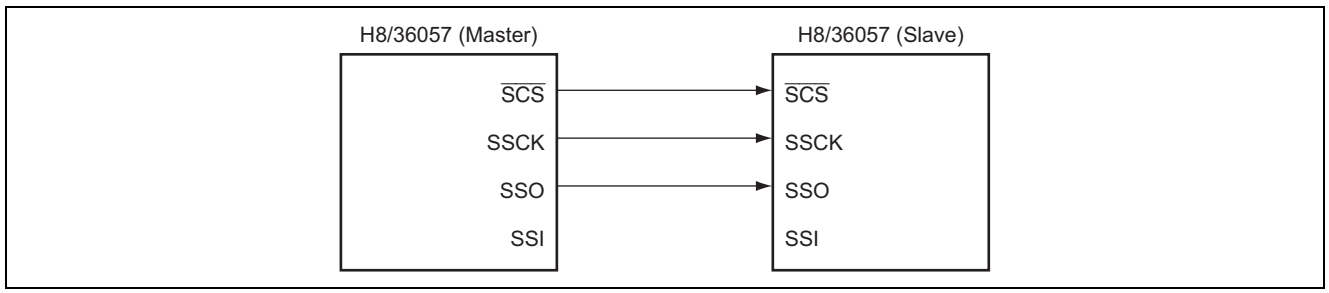


Figure 5 Block Diagram of the SSU Communications between H8/36057 LSIs (When the master device transmits data)

2. Figure 6 shows the transmit operation in this sample task. Hardware and software processing is performed with the timing shown in figure 6 to implement communications between the H8/36057 LSIs.
   Figure 6 shows an example of transmit operation in which the polarity of the SSCK clock is specified to output high during an idle state and output low during an active state (CPOS = 0), the SSCK clock phase is specified to change data at the first edge (CPHS = 0), and 8-bit data (H'AA) is transmitted with LSB first.

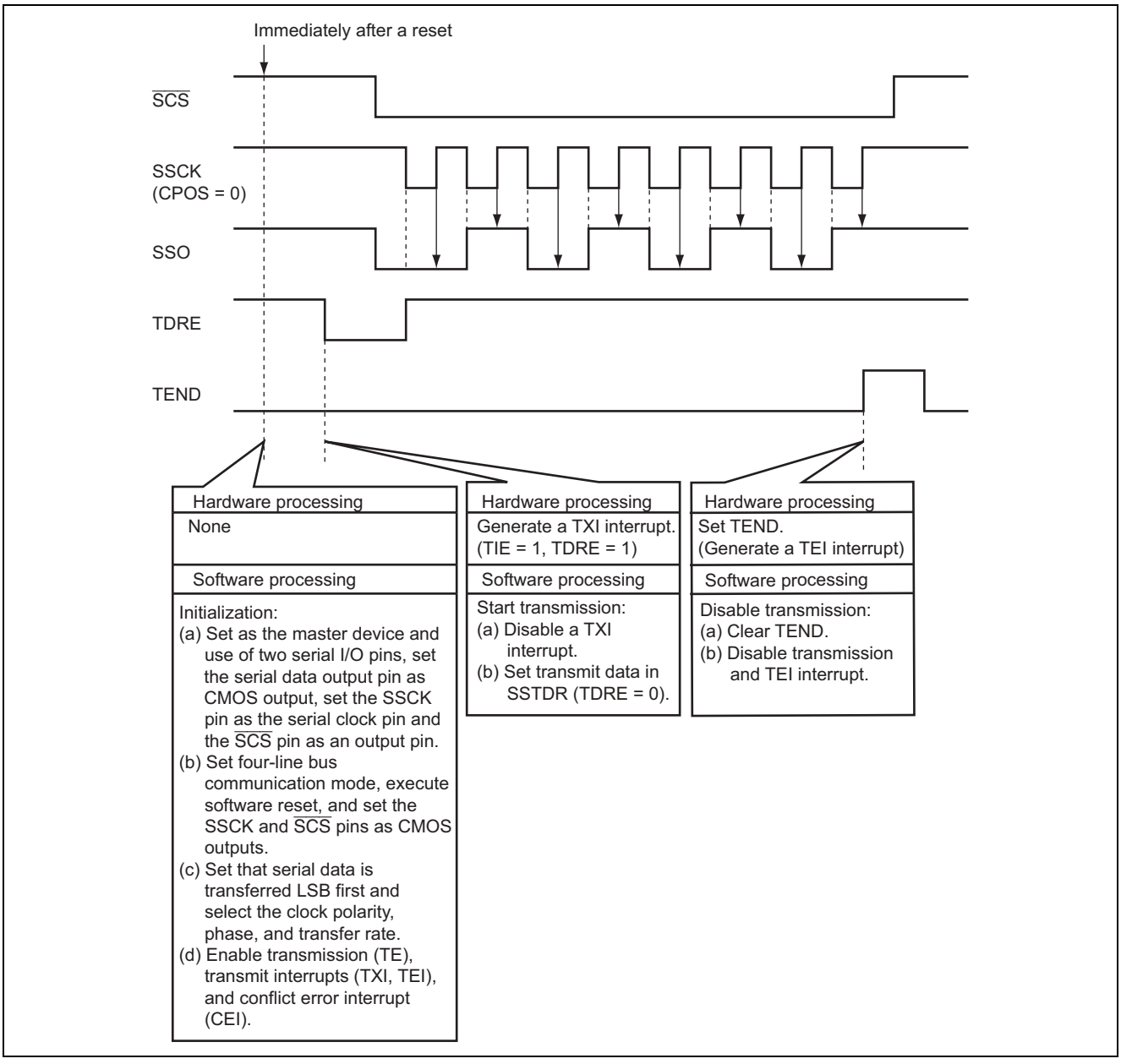


Figure 6 Transmit Operation

3.2 Receive Operation

1. Figure 7 shows a block diagram of the case when the slave H8/36057 receives data from the master H8/36057.

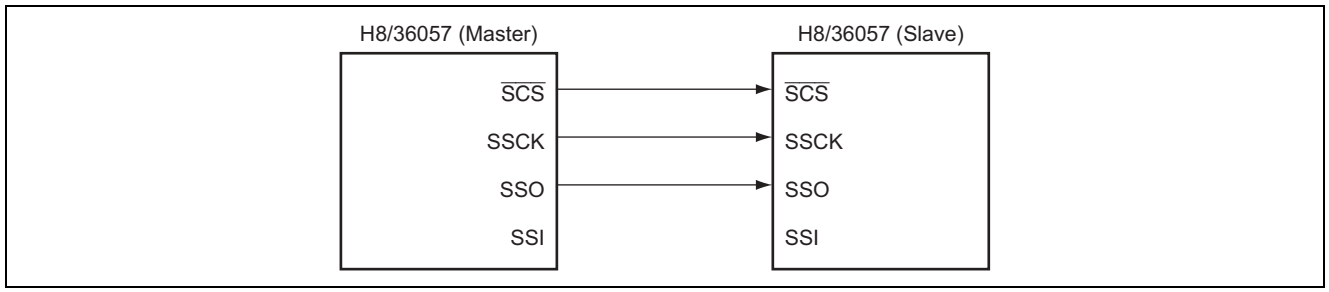


Figure 7 Block Diagram of the SSU Communications between H8/36057 LSIs (When the slave device receives data)

2. Figure 8 shows the receive operation in this sample task. Hardware and software processing is performed with the timing shown in figure 8 to implement communications between the H8/36057 LSIs.
   Figure 8 shows an example of transmit operation in which the polarity of the SSCK clock is specified to output high during an idle state and output low during an active state (CPOS = 0), the SSCK clock phase is specified to change data at the first edge (CPHS = 0), and 8-bit data (H'AA) is received with LSB first.

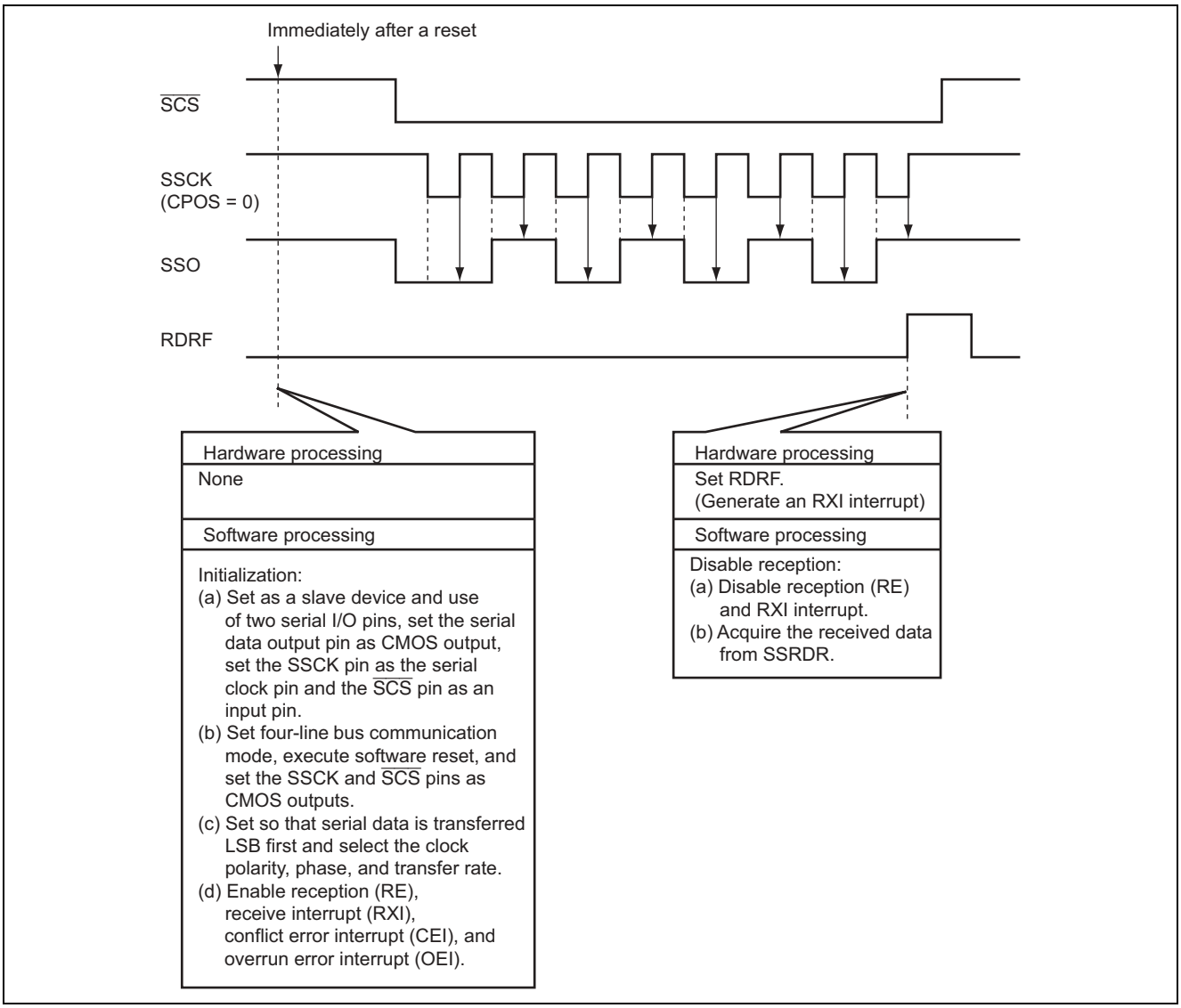


Figure 8 Receive Operation

4. Application Example (Specification 1: Standard Mode)

As shown in figure 9, data transmission and reception are performed between H8/36057 LSIs (ϕ = 20 MHz) in standard mode (master: transmission and reception; slave: transmission and reception).

- The transfer format is as follows: data length is 8 bits (data transmitted by the master = H'AA, data transmitted by the slave = H'CC), data is transmitted with LSB first, and transfer rate is 1.25 Mbps.
- Two pins (SSO and SSI) are used as serial data I/O pins (standard mode).
- The clock polarity of the SSCK pin is specified to output high during an idle state and output low during an active state (CPOS = 0). The SSCK clock phase is specified to change data at the first edge (CPHS = 0).

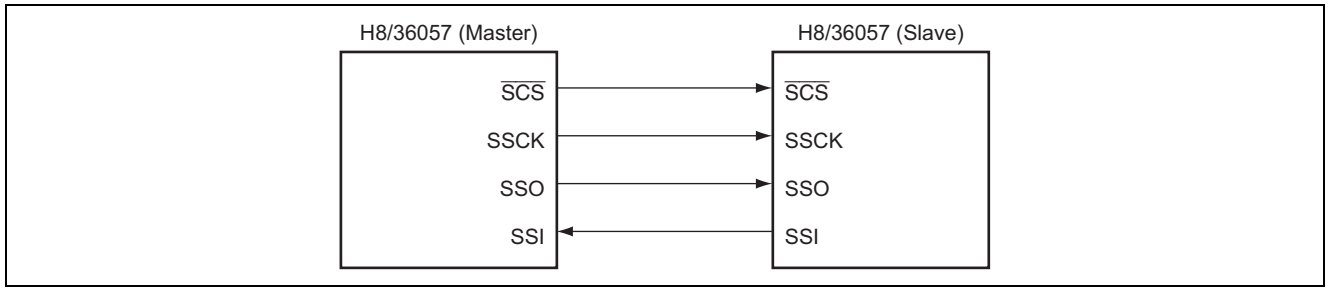


Figure 9 Block Diagram of the SSU Communications between H8/36057 LSIs (Master and slave devices transmits/receives data in standard mode)

4.1 Function Assignment

4.1.1 Master

Table 1 shows the assignment of functions in this sample task.

In this sample task, master functions are assigned to the SSU of the H8/36057 to perform data transmission and reception simultaneously.

Table 1 Assignment of Functions (Master Function)

Element	Description
$\overline{\text{SCS}}$ pin	Transmits a chip select signal.
SSCK pin	Transmits a transfer clock.
SSO pin	Transmits data to the H8/36057 (slave).
SSI pin	Receives data from the H8/36057 (slave).
SSCRH	Sets the master mode and use of two serial data I/O pins, sets that serial data is output as CMOS output, sets the $\overline{\text{SCS}}$ pin as an output pin and the SSCK pin as the serial clock pin.
SSCRL	Sets four-line bus communication mode, sets the SSCK and $\overline{\text{SCS}}$ pin s as CMOS outputs, and executes software reset.
SSMR	Sets LSB first serial data transfer, clock polarity, phase, and transfer rate.
SSER	Sets transmission/reception mode, transmission/reception interrupts, and error interrupts.
SSSR	Indicates the transmission/reception statuses via the TDRE, TEND, and RDRF bits, a conflict error occurrence via the CE bit, and an overrun error occurrence via the ORER bit.
SSTDR	Sets data to be transmitted to the H8/36057 (slave).
SSRDR	Stores data received from the H8/36057 (slave).

4.1.2 Slave

Table 2 shows the assignment of functions in this sample task.

In this sample task, slave functions are assigned to the SSU of the H8/36057 to perform data transmission and reception simultaneously.

Table 2 Assignment of Functions (Slave Function)

Element	Description
$\overline{\text{SCS}}$ pin	Receives a chip select signal.
SSCK pin	Receives a transfer clock.
SSO pin	Receives data from the H8/36057 (master).
SSI pin	Transmits data to the H8/36057 (master).
SSCRH	Sets the slave mode and use of two serial data I/O pins, sets that serial data is output as CMOS output, sets the $\overline{\text{SCS}}$ pin as an input pin and the SSCK pin as the serial clock pin.
SSCRL	Sets four-line bus communication mode, sets the SSCK and $\overline{\text{SCS}}$ pin s as CMOS outputs, and executes software reset.
SSMR	Sets LSB first serial data transfer, clock polarity, phase, and transfer rate.
SSER	Sets transmission/reception mode, transmission/reception interrupts, and error interrupts.
SSSR	Indicates the transmission/reception statuses via the TDRE, TEND, and RDRF bits, a conflict error occurrence via the CE bit, and an overrun error occurrence via the ORER bit.
SSTDR	Sets data to be transmitted to the H8/36057 (master).
SSRDR	Stores data received from the H8/36057 (master).

4.2 Description of Operation (Data Transmission and Reception)

Figure 10 shows a block diagram of the case when data transmission and reception are performed between H8/36057s (master and slave) (master: transmits and receives; slave: transmits and receives).

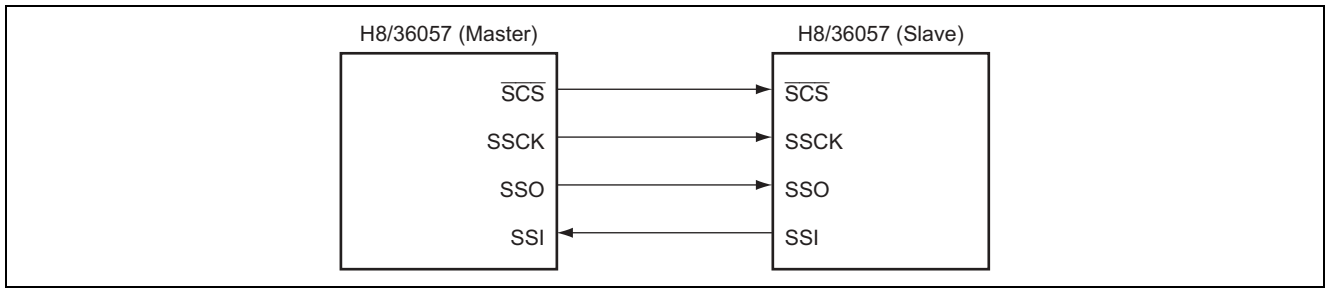


Figure 10 Block Diagram of the SSU Communications between H8/36057 LSIs (Master and slave devices transmits/receives data in standard mode)

Figure 11 shows the transmit and receive operations in this sample task. Hardware and software processing is performed with the timing shown in figure 11 to implement the SSU communications between H8/36057 LSIs.

4.2.1 Master and Slave Operations

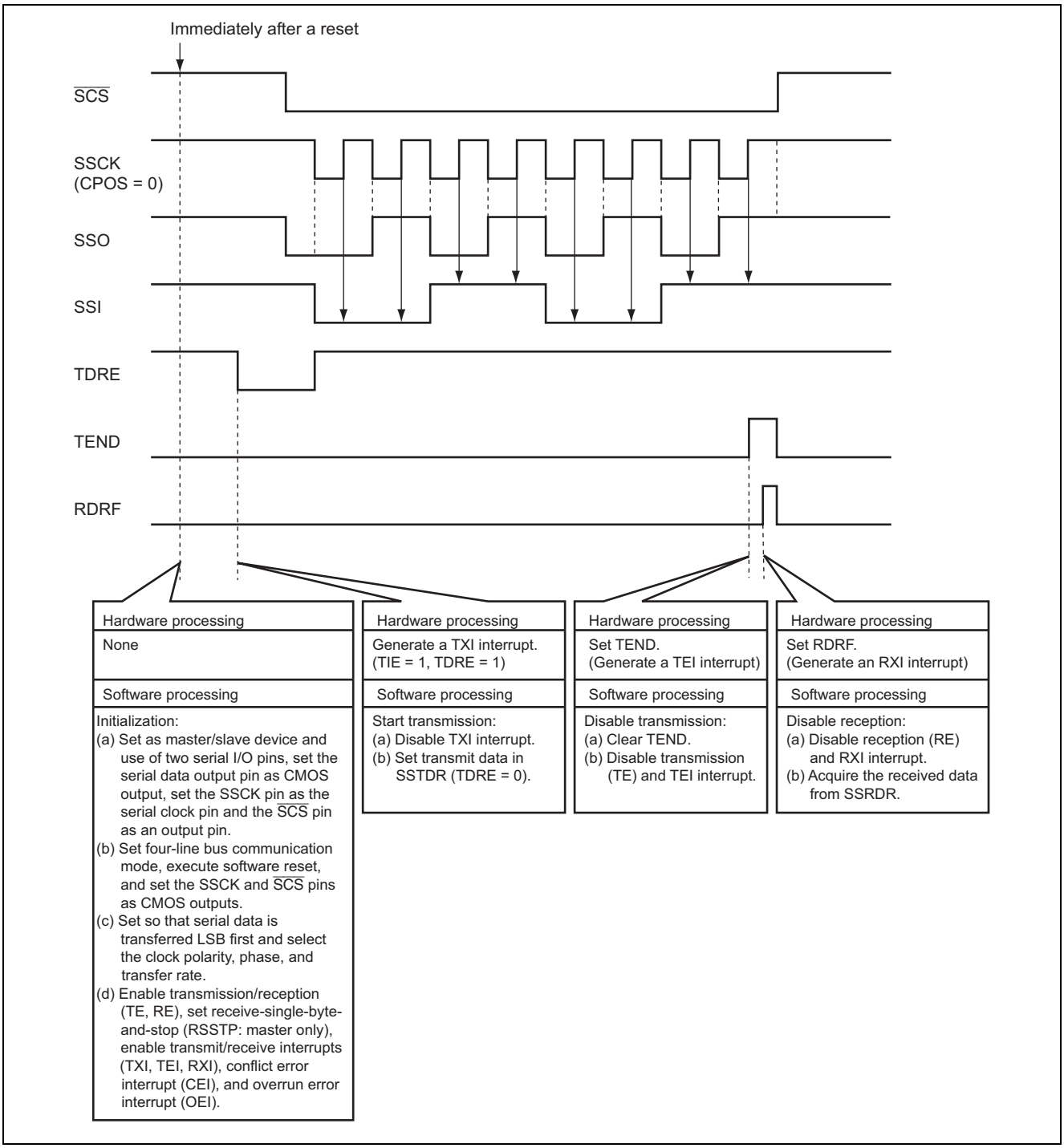


Figure 11 Transmit and Receive Operations

4.3 Description of Software (Master/Slave: Transmission and Reception Functions)

4.3.1 Master

The modules used in this sample task are described below.

(1) **Description of Modules**

Module Name	Label Name	Function
Main routine	main	Sets mask levels.
Initialization	SSU_init	Initializes the SSU.
Error processing	SSUI_int	Started by a CEI interrupt and performs error processing.
		Started by an OEI interrupt and performs error processing.
Data transmission		Started by a TXI interrupt and performs data transmission.
Transmission end		Started by a TEI interrupt and ends data transmission.
Data reception		Started by an RXI interrupt and performs data reception.

(2) **Arguments**

This sample task uses no arguments.

(3) **Internal Registers**

The following table describes the internal registers used in this sample task.

Register Name	Function	Setting	Used in
SSCRH	Sets the H8/36057 as the master device and use of two serial data I/O pins and sets serial data output pin, SSCK pin, and $\overline{SCS}$ pin functions.	H'8E	Initialization routine
SSCRL	Sets bus communication mode, executes software reset, and sets the SSCK and $\overline{SCS}$ pin functions.	H'60	Initialization routine
SSMR	Sets serial data transmission with LSB first, clock polarity, phase, and transfer rate (ϕ clock input).	H'04	Initialization routine
SSER	Enables interrupts (CEI, OEI, RXI, TEI, and TXI) and enables or disables the SSU transmission and reception.	H'EF	Initialization routine
SSTDR	Sets data to be transmitted to the receiver H8/36057.	H'AA	Data transmission
SSRDR	Stores data received from the transmitter H8/36057.		Data reception
SSSR	Detects an error.		Error processing

(4) **RAM Usage**

Label Name	Function	Data Size	Used in
Rxdata	Stores received data.	unsigned char	Data reception

4.3.2 Slave

(1) **Description of Modules**

Module Name	Label Name	Function
Main routine	main	Sets mask levels.
Initialization	SSU_init	Initializes the SSU.
Error processing	SSUI_int	Started by a CEI interrupt and performs error processing.
		Started by an OEI interrupt and performs error processing.
Data reception		Started by an RXI interrupt and performs data reception.
Transmission end		Started by a TEI interrupt and ends data transmission.
Data transmission		Started by a TXI interrupt and performs data transmission.

(2) **Arguments**

This sample task uses no arguments.

(3) **Internal Registers**

The following table describes the internal registers used in this sample task.

Label Name	Function	Setting	Used in
SSCRH	Sets the H8/36057 as a slave device and use of two serial data I/O pins, and sets serial data output pin, SSCK pin, and $\overline{SCS}$ pin functions.	H'0D	Initialization routine
SSCRL	Sets bus communication mode, executes software reset, set the SSCK and $\overline{SCS}$ pin functions.	H'60	Initialization routine
SSMR	Sets serial data transmission with LSB first, clock polarity, phase, and transfer rate (ϕ clock input).	H'04	Initialization routine
SSER	Enables interrupts (CEI, OEI, RXI, TEI, and TXI) and enables or disables the SSU transmission and reception.	H'CF	Initialization routine
SSRDR	Stores data received from the transmitter H8/36057.		Data reception
SSTDR	Sets data to be transmitted the receiver H8/36057.	H'CC	Data transmission
SSSR	Detects an error.		Error processing

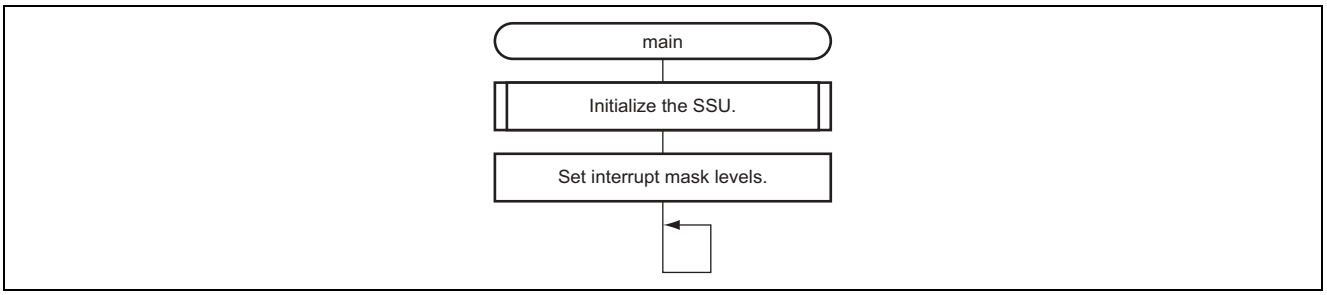
(4) **RAM Usage**

Label Name	Function	Data Size	Used in
Rxdata	Stores received data.	unsigned char	Data reception

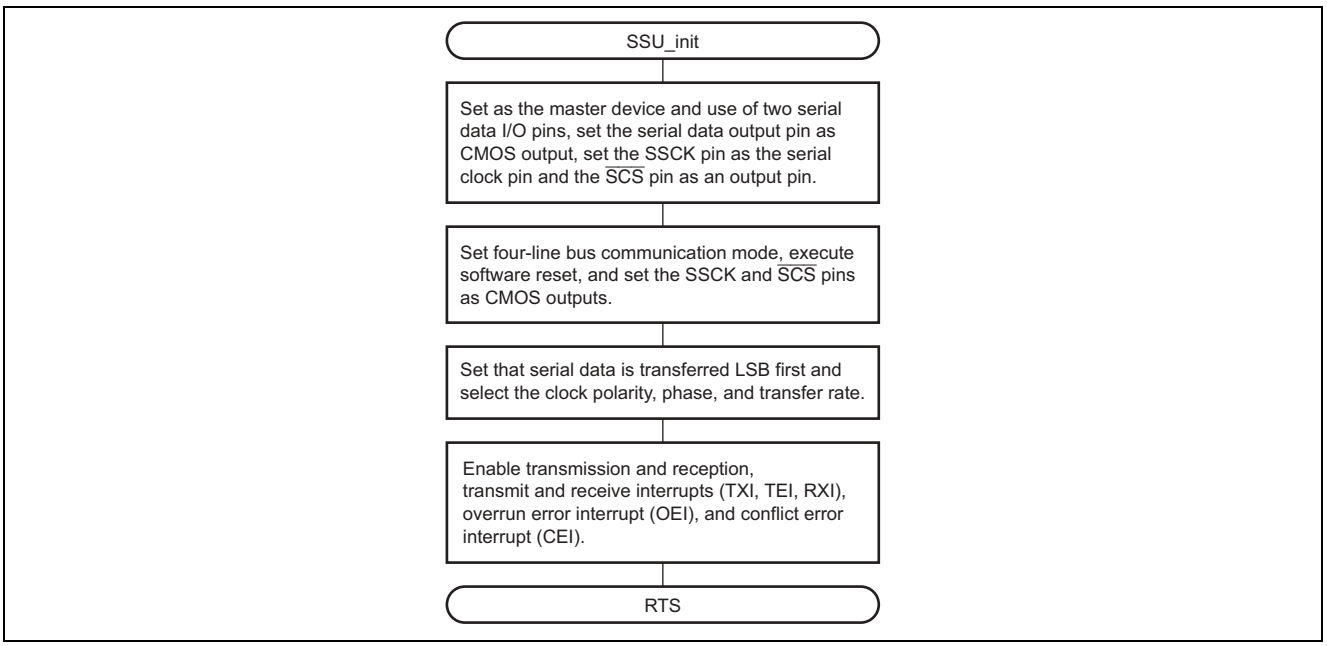
4.4 Flowchart

4.4.1 Master

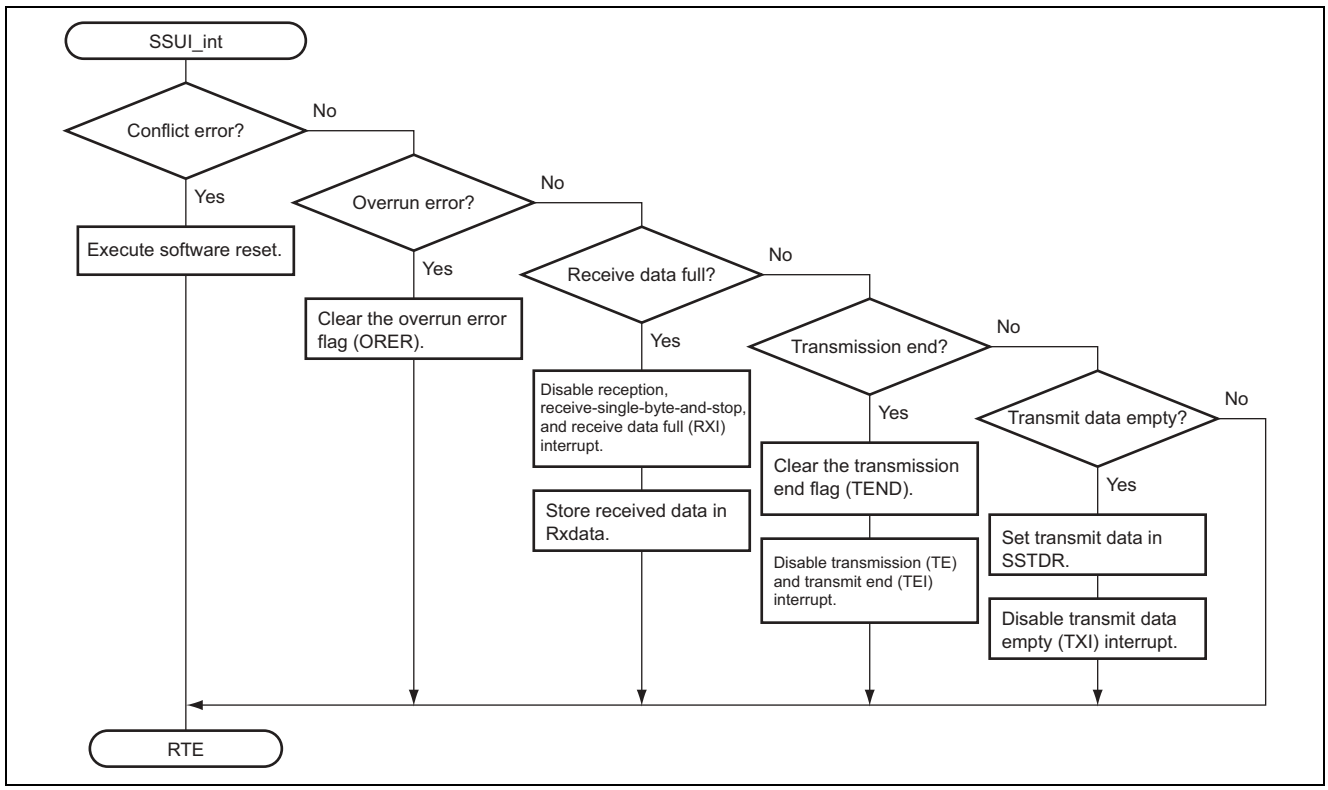
(1) **Main Routine**



(2) **Initialization**

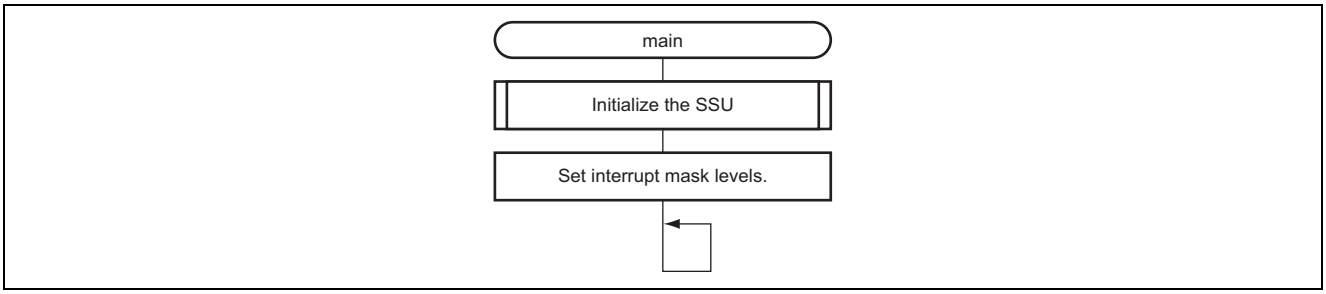


(3) **Error processing, data reception, and data transmission**

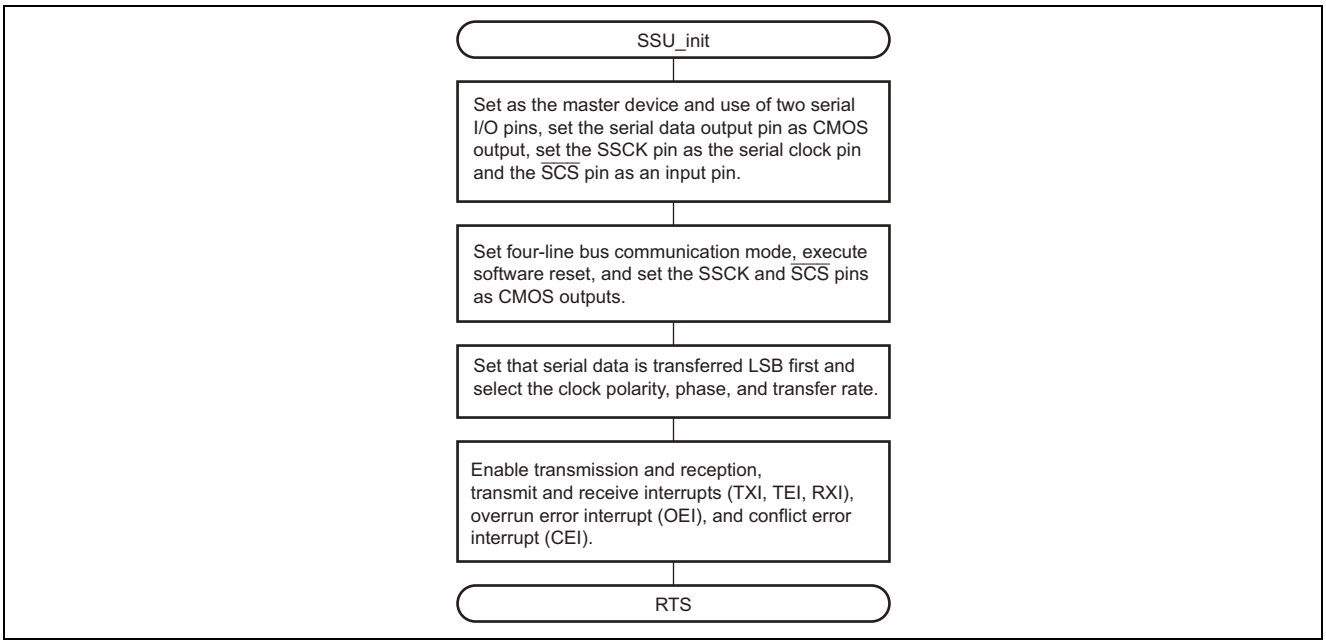


4.4.2 Slave

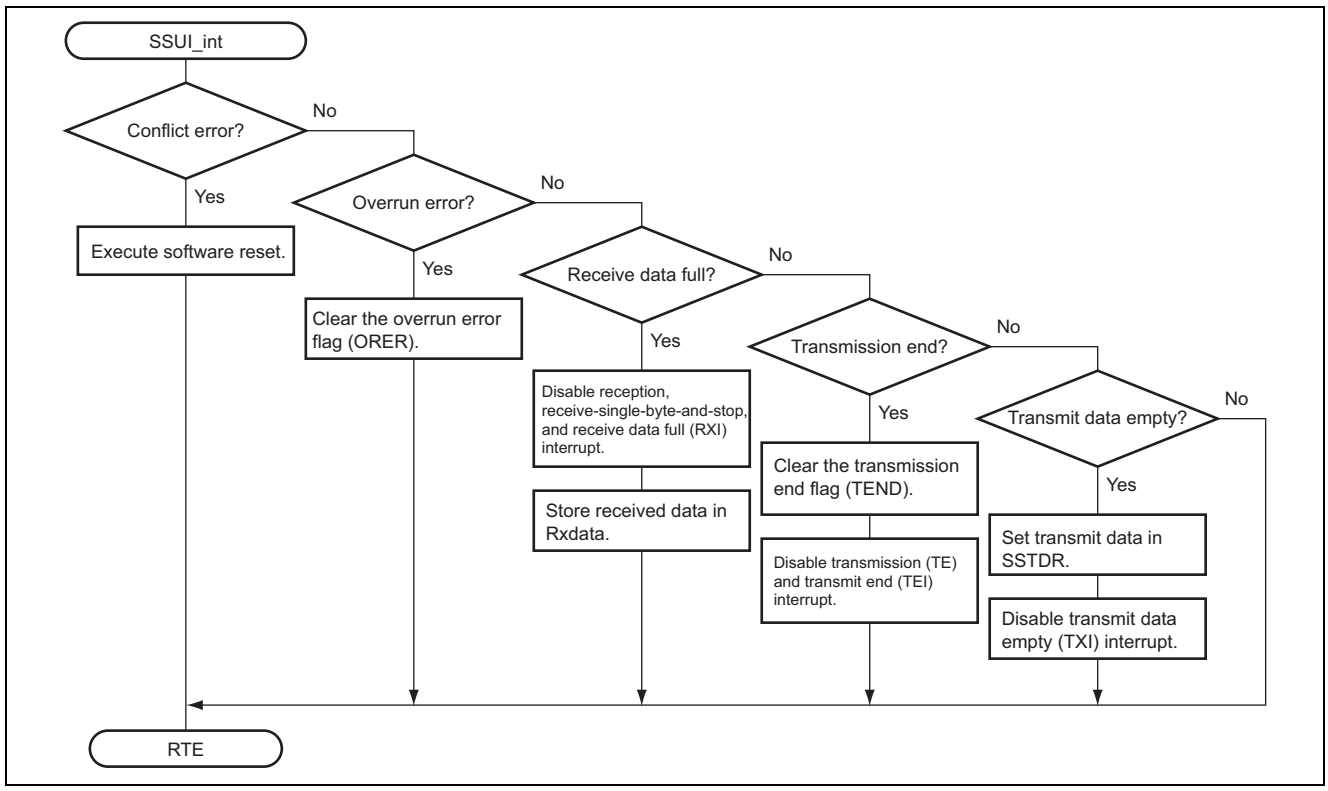
(1) **Main Routine**



(2) **Initialization**



(3) **Error processing, data reception, and data transmission**



4.5 Program Listing

4.5.1 Master

```
#include <machine.h>
#include "H8_36057.h"
void main (void);
void SSU_init (void);
unsigned char Rxdata;
/************************************************************************************************/
/*  Main Routine                                                                                */
/************************************************************************************************/
void main (void)
#pragma asm
    mov.l #H'FFFFFF7C,SP
#pragma endasm
{
    SSU_init();
    set_imask_ccr(0);
    while(1);
}
/************************************************************************************************/
/*  SSU Module Initialize Routine                                                               */
/************************************************************************************************/
void SSU_init(void)
{
    SSU.SSER.BYTE  = 0x00;
    SSU.SSCRL.BYTE = 0x00;
    SSU.SSCRH.BYTE = 0x8E;
    SSU.SSCRL.BYTE = 0x60;
    SSU.SSMR.BYTE  = 0x04;
    SSU.SSSR.BYTE &= 0x04;
    SSU.SSER.BYTE  = 0xEF;
}
/************************************************************************************************/
#pragma interrupt (SSUI_int)       /* Error, reception, and transmission interrupts            */
/************************************************************************************************/
void SSUI_int(void)
{
    if(SSU.SSSR.BIT.CE && SSU.SSER.BIT.CEIE){
           SSU.SSSR.BIT.CE = 0;
           SSU.SSCRL.BIT.SRES = 1;
    }
    else if(SSU.SSSR.BIT.ORER && SSU.SSER.BIT.RIE){
            SSU.SSSR.BIT.ORER = 0;
    }
    else if(SSU.SSSR.BIT.RDRF && SSU.SSER.BIT.RIE){
            SSU.SSER.BYTE &= 0x8D;
            Rxdata = SSU.SSRDR;
    }
    else if(SSU.SSSR.BIT.TEND && SSU.SSER.BIT.TEIE){
            SSU.SSSR.BIT.TEND = 0;
            SSU.SSER.BYTE &= 0x63;
    }
    else if(SSU.SSSR.BIT.TDRE && SSU.SSER.BIT.TIE){
            SSU.SSTDR = 0xAA;
            SSU.SSER.BIT.TIE = 0;
    }
}
```

4.5.2 Slave

```
#include <machine.h>
#include "H8_36057.h"
void main (void);
void SSU_init (void);
unsigned char Rxdata;
/************************************************************************************************/
/*  Main Routine                                                                                */
/************************************************************************************************/
void main (void)
#pragma asm
    mov.l #H'FFFFFF7C,SP
#pragma endasm
{
    SSU_init();
    set_imask_ccr(0);
    while(1);
}
/************************************************************************************************/
/* SSU Module Initialize Routine                                                                */
/************************************************************************************************/
void SSU_init(void)
{
    SSU.SSER.BYTE  = 0x00;
    SSU.SSCRL.BYTE = 0x00;
    SSU.SSCRH.BYTE = 0x0D;
    SSU.SSCRL.BYTE = 0x60;
    SSU.SSMR.BYTE  = 0x04;
    SSU.SSSR.BYTE &= 0x04;
    SSU.SSER.BYTE  = 0xCF;
}
/************************************************************************************************/
#pragma interrupt (SSUI_int)      /* Error, reception, and transmission interrupts              */
/************************************************************************************************/
void SSUI_int(void)
{
    if(SSU.SSSR.BIT.CE && SSU.SSER.BIT.CEIE){
           SSU.SSSR.BIT.CE = 0;
           SSU.SSCRL.BIT.SRES = 1;
    }
    else if(SSU.SSSR.BIT.ORER && SSU.SSER.BIT.RIE){
            SSU.SSSR.BIT.ORER = 0;
    }
    else if(SSU.SSSR.BIT.RDRF && SSU.SSER.BIT.RIE){
            SSU.SSER.BYTE &= 0x8D;
            Rxdata = SSU.SSRDR;
    }
    else if(SSU.SSSR.BIT.TEND && SSU.SSER.BIT.TEIE){
            SSU.SSSR.BIT.TEND = 0;
            SSU.SSER.BYTE &= 0x63;
    }
    else if(SSU.SSSR.BIT.TDRE && SSU.SSER.BIT.TIE){
            SSU.SSTDR = 0xCC;
            SSU.SSER.BIT.TIE = 0;
    }
}
```

4.6 Program when the Master Transmits Data and the Slave Receives Data

4.6.1 Settings for the Master Device

A program for the master device to transmit 8-bit data only once should use the routines with the following label names, as are shown in the master program listing in the preceding subsection: main, SSU_init (change the SSER setting to H'8D), and SSUI_int (CEI, TEI, and TXI).

4.6.2 Settings for the Slave Device

A program for the slave device to receive 8-bit data only once should use the routines with the following label names, as are shown in the slave program listing in the preceding subsection: main, SSU_init (change the SSER setting to H'43), and SSUI_int (CEI, OEI, and RXI).

4.7 Program when the Master Transmits $\overline{SCS}$ and SSCK and Receives Data, and the Slave Transmits Data

4.7.1 Settings for the Master Device

A program for the master device to transmit only SSCK and $\overline{SCS}$ (for receiving 8-bit data once) should use the routines with the following label names, as are shown in the master program listing in the preceding subsection: main, SSU_init (change the SSER setting to H'63), and SSUI_int (CEI, OEI, and RXI).

4.7.2 Settings for the Slave Device

A program for the slave device to only transmit data (8-bit data) should use the routines with the following label names, as are shown in the slave program listing in the preceding subsection: main, SSU_init (change the SSER setting to H'8D), and SSUI_int (CEI, TEI, and TXI).

5. Application Example (Specification 2: Bi-Directional Mode)

As shown in figure 12, data transmission and reception are performed between H8/36057 LSIs (ϕ = 20 MHz) in bi-directional mode (master: transmission and reception; slave: transmission and reception).

- The transfer format is as follows: data length is 16 bits (data transmitted by the master/slave = H'AA, H'55), data is transmitted with MSB first, and transfer rate is 0.08 Mbps.
- A single pin (SSO) is used for serial data I/O (bi-directional mode).
- The clock polarity of the SSCK pin is specified to output low during an idle state and output high during an active state (CPOS = 1). The SSCK clock phase is specified to latch data at the first edge (CPHS = 1).

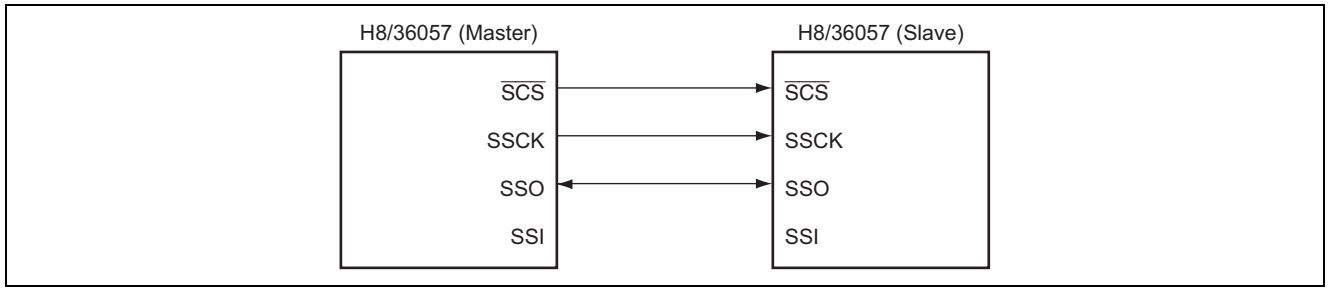


Figure 12 Block Diagram of the SSU Communications between H8/36057 LSIs (Master and slave devices transmits/receives data in bi-directional mode)

5.1 Function Assignment

5.1.1 Master

Table 3 shows the assignment of functions in this sample task.

In this sample task, master functions are assigned to the SSU of the H8/36057 to perform data transmission and reception.

Table 3 Assignment of Functions (Master Function)

Element	Description
$\overline{\text{SCS}}$ pin	Transmits a chip select signal.
SSCK pin	Transmits a transfer clock.
SSO pin	Transmits/receives data to/from the H8/36057 (slave).
SSCRH	Sets the master mode and use of one serial data I/O pin, sets that serial data is output as CMOS output, sets the $\overline{\text{SCS}}$ pin as an output pin and the SSCK pin as the serial clock pin.
SSCRL	Sets four-line bus communication mode, sets the SSCK and $\overline{\text{SCS}}$ pin s as CMOS outputs, and executes software reset.
SSMR	Sets MSB first serial data transfer, clock polarity, phase, and transfer rate.
SSER	Sets transmission/reception mode, transmission/reception interrupts, and error interrupts.
SSSR	Indicates the transmission/reception statuses via the TDRE, TEND, and RDRF bits, a conflict error occurrence via the CE bit, and an overrun error occurrence via the RER bit.
SSTDR	Sets data to be transmitted to the H8/36057 (slave).
SSRDR	Stores data received from the H8/36057 (slave).

5.1.2 Slave

Table 4 shows the assignment of functions in this sample task.

In this sample task, slave functions are assigned to the SSU of the H8/36057 to perform data transmission and reception.

Table 4 Assignment of Functions (Slave Function)

Element	Description
$\overline{\text{SCS}}$ pin	Receives a chip select signal.
SSCK pin	Receives a transfer clock.
SSO pin	Transmits/receives data to/from the H8/36057 (master).
SSCRH	Sets the slave mode and use of one serial data I/O pin, sets that serial data is output as CMOS output, sets the $\overline{\text{SCS}}$ pin as an input pin and the SSCK pin as the serial clock pin.
SSCRL	Sets four-line bus communication mode, sets the SSCK and $\overline{\text{SCS}}$ pin s as CMOS outputs, and executes software reset.
SSMR	Sets MSB first serial data transfer, clock polarity, phase, and transfer rate.
SSER	Sets transmission/reception mode, transmission/reception interrupts, and error interrupts.
SSSR	Indicates the transmission/reception statuses via the TDRE, TEND, and RDRF bits, a conflict error occurrence via the CE bit, and an overrun error occurrence via the ORER bit.
SSTDR	Sets data to be transmitted to the H8/36057 (master).
SSRDR	Stores data received from the H8/36057 (master).

5.2 Description of Operation (Data Transmission and Reception)

Figure 13 shows a block diagram of the case when data transmission and reception are performed between H8/36057 LSIs (master and slave) (master: transmits and receives; slave: transmits and receives).

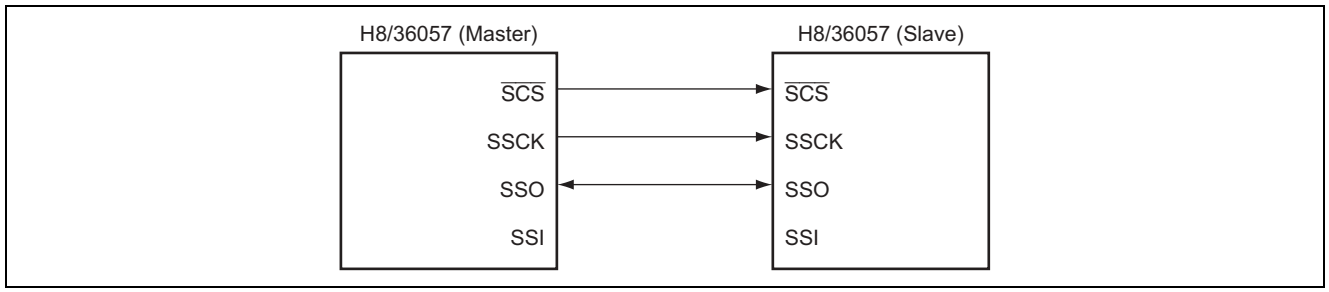


Figure 13 Block Diagram of the SSU Communications between H8/36057 LSIs (Master and slave devices transmits/receives data in bi-directional mode)

Figures 14 and 15 show the transmit and receive operations in this sample task. Hardware and software processing is performed with the timings shown in figures 14 and 15 to implement the SSU communications between H8/36057 LSIs.

5.2.1 Master Operation (*Processing in Slave Device)

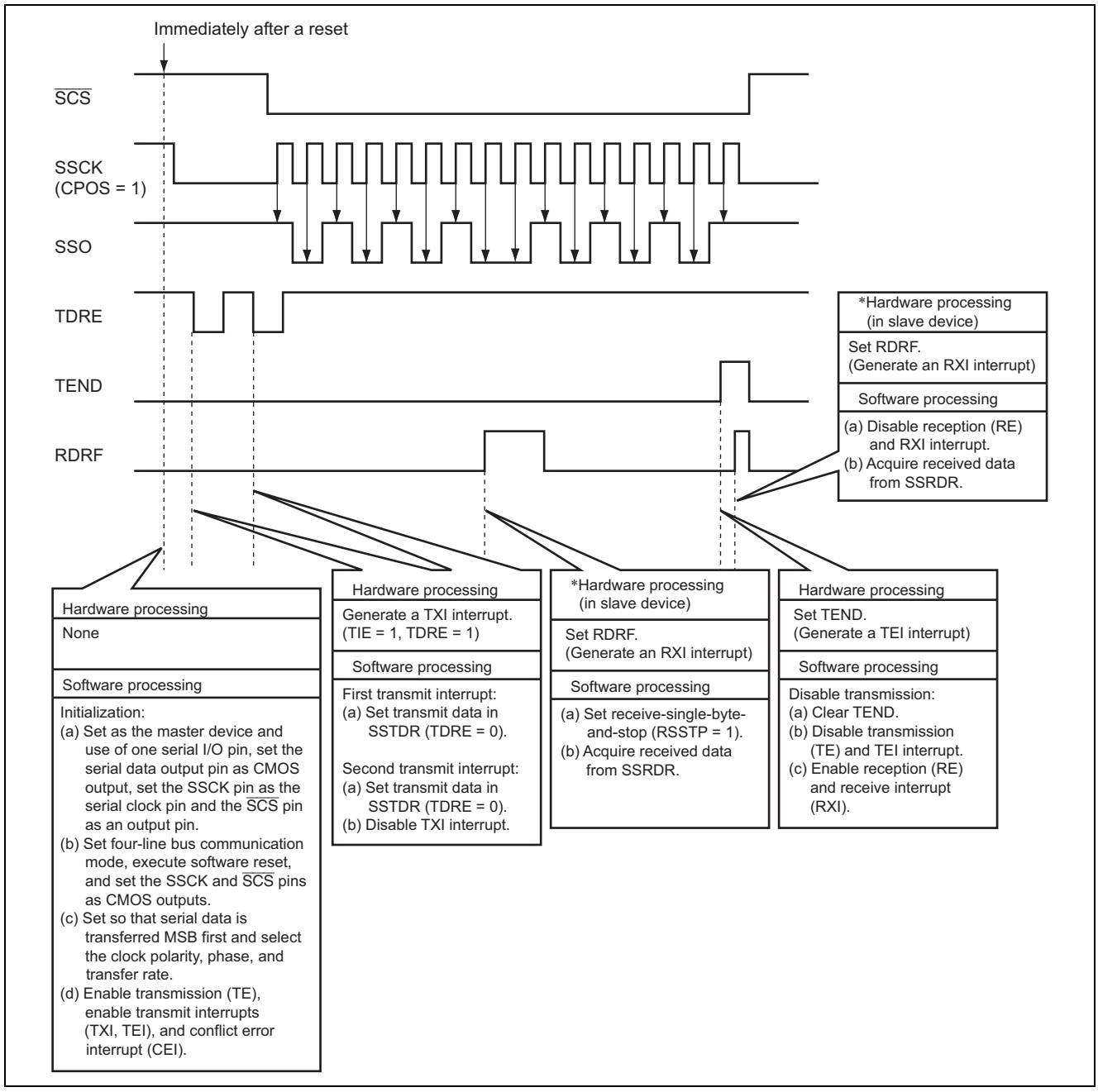


Figure 14 Transmit and Receive Operations in Master Device (Bi-Directional Mode)

5.2.2 Slave Operation (*Processing in Master Device)

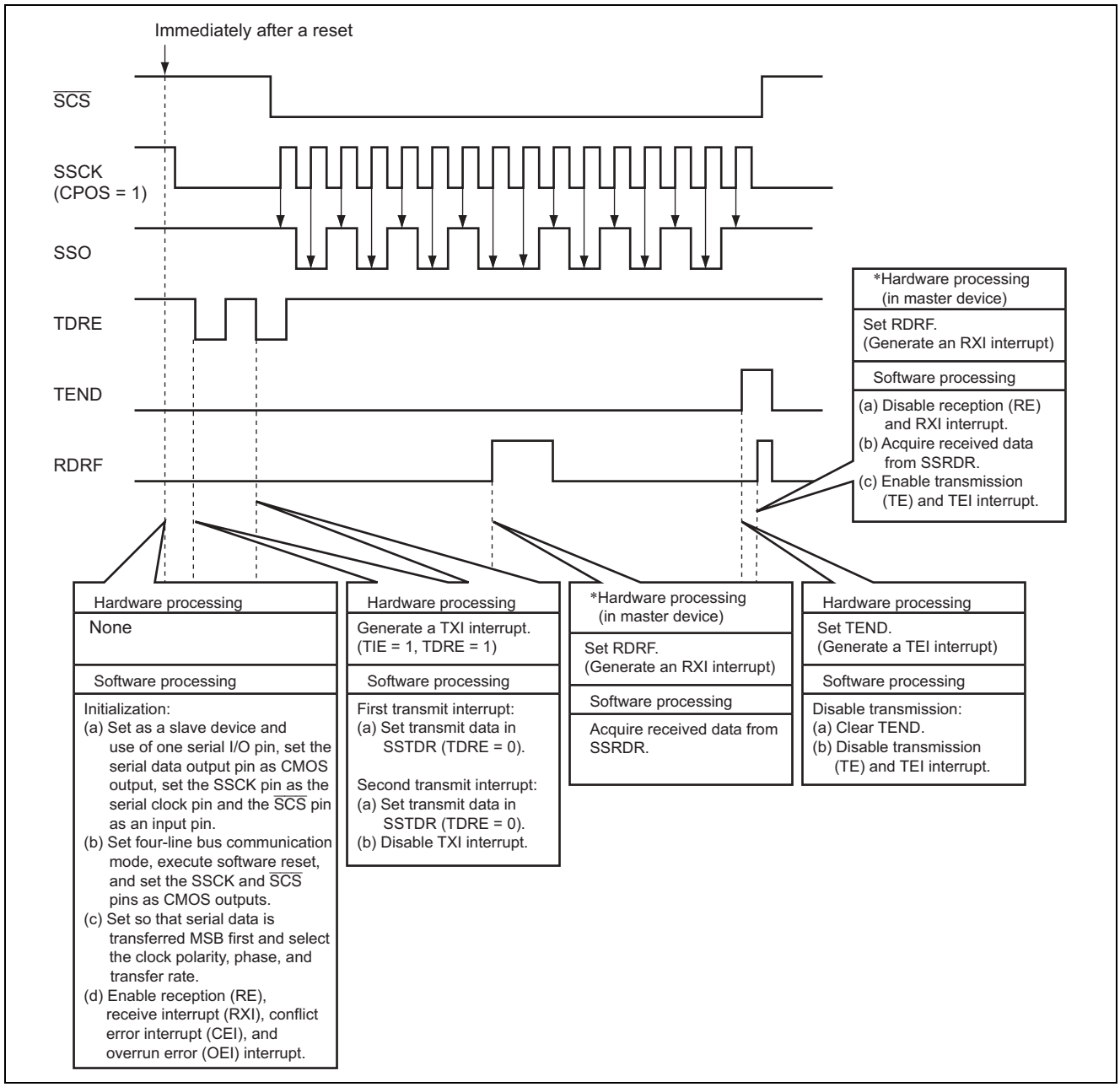


Figure 15 Transmit and Receive Operations in Slave Device (Bi-Directional Mode)

5.3 Description of Software (Master/Slave: Transmission and Reception Functions)

5.3.1 Master

The modules used in this sample task are described below.

(1) **Description of Modules**

Module Name	Label Name	Function
Main routine	main	Sets mask levels.
Initialization	SSU_init	Initializes the SSU.
Variable initialization	SSU_counter_init	Counts the length of transmit/receive data.
Error processing	SSUI_int	Started by a CEI interrupt and performs error processing.
		Started by an OEI interrupt and performs error processing.
Data reception		Started by an RXI interrupt and performs data reception.
Transmission end		Started by a TEI interrupt and ends data transmission.
Data transmission		Started by a TXI interrupt and performs data transmission.

(2) **Arguments**

This sample task uses no arguments.

(3) **Internal Registers**

The following table describes the internal registers used in this sample task.

Label Name	Function	Setting	Used in
SSCRH	Sets the H8/36057 as the master device and use of one serial data I/O pin and sets serial data output pin, SSCK pin, and $\overline{SCS}$ pin functions.	H'CE	Initialization routine
SSCRL	Sets bus communication mode, executes software reset, and sets the SSCK and $\overline{SCS}$ pin functions.	H'60	Initialization routine
SSMR	Sets serial data transmission with MSB first, clock polarity, phase, and transfer rate (ϕ clock input).	H'E0	Initialization routine
SSER	Enables interrupts (CEI, OEI, RXI, TEI, and TXI) and enables or disables the SSU transmission and reception.	H'8D	Initialization routine
SSTDR	Sets data to be transmitted to the receiver H8/36057.	H'AA, H'55	Data transmission
SSRDR	Stores data received from the transmitter H8/36057.		Data reception
SSSR	Detects an error.		Error processing

(4) **RAM Usage**

Label Name	Function	Data Size	Used in
Rxdata [0,1]	Stores received data.	unsigned char	Data reception
T_counter	Counts the length of transmit data.	unsigned char	Data transmission
R_counter	Counts the length of received data.	unsigned char	Data reception

5.3.2 Slave

(1) **Description of Modules**

Module Name	Label Name	Function
Main routine	main	Sets mask levels.
Initialization	SSU_init	Initializes the SSU.
Variable initialization	SSU_ counter_init	Counts the length of transmit/receive data.
Error processing	SSUI_int	Started by a CEI interrupt and performs error processing.
		Started by an OEI interrupt and performs error processing.
Data reception		Started by an RXI interrupt and performs data reception.
Transmission end		Started by a TEI interrupt and ends data transmission.
Data transmission		Started by a TXI interrupt and performs data transmission.

(2) **Arguments**

This sample task uses no arguments.

(3) **Internal Registers**

The following table describes the internal registers used in this sample task.

Label Name	Function	Setting	Used in
SSCRH	Sets the H8/36057 as a slave device and use of one serial data I/O pin and sets serial data output pin, SSCK pin, and $\overline{SCS}$ pin functions.	H'4D	Initialization routine
SSCRL	Sets bus communication mode, executes software reset, and sets the SSCK and $\overline{SCS}$ pin functions.	H'60	Initialization routine
SSMR	Sets serial data transmission with MSB first, clock polarity, phase, and transfer rate (ϕ clock input).	H'E0	Initialization routine
SSER	Enables interrupts (CEI, OEI, and RXI) and enables or disables the SSU transmission and reception.	H'43	Initialization routine
SSRDR	Sets data received from the transmitter H8/36057.		Data reception
SSTDR	Sets data to be transmitted the receiver H8/36057.	H'AA, H'55	Data transmission
SSSR	Detects an error.		Error processing

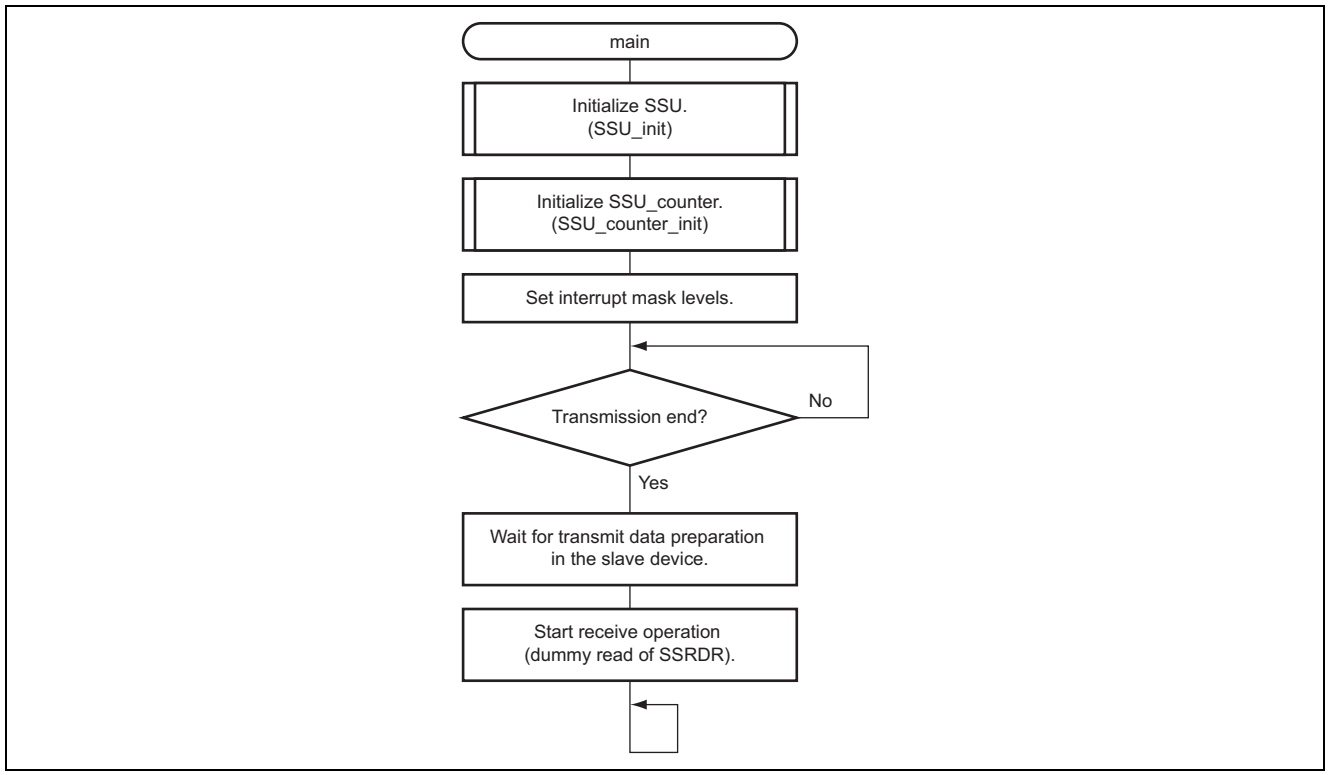
(4) **RAM Usage**

Label Name	Function	Data Size	Used in
Rxdata [0,1]	Stores received data.	unsigned char	Data reception
T_counter	Counts the length of transmit data.	unsigned char	Data transmission
R_counter	Counts the length of receive data.	unsigned char	Data reception

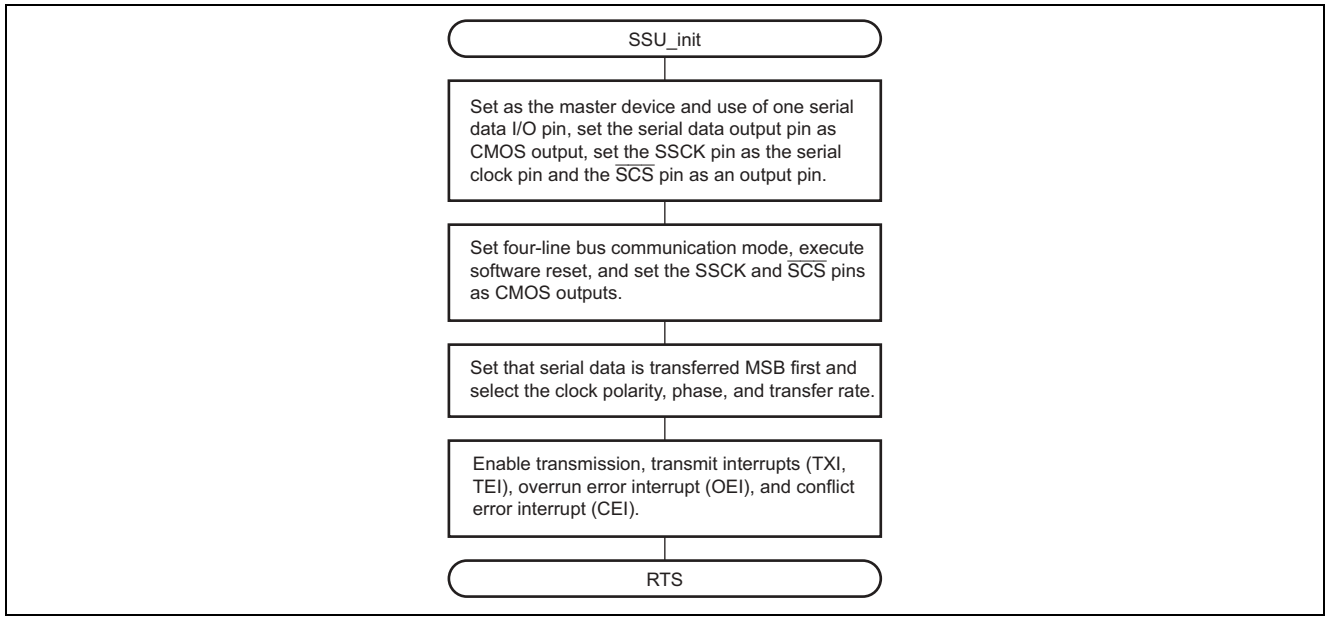
5.4 Flowchart

5.4.1 Master

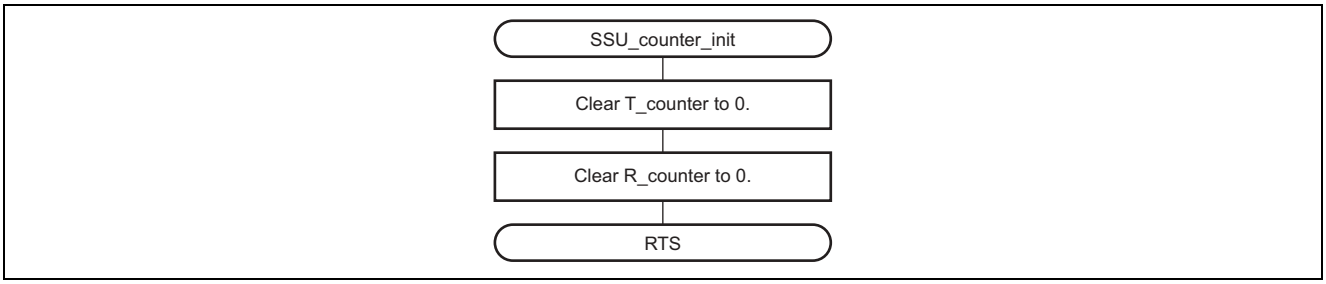
(1) **Main Routine**



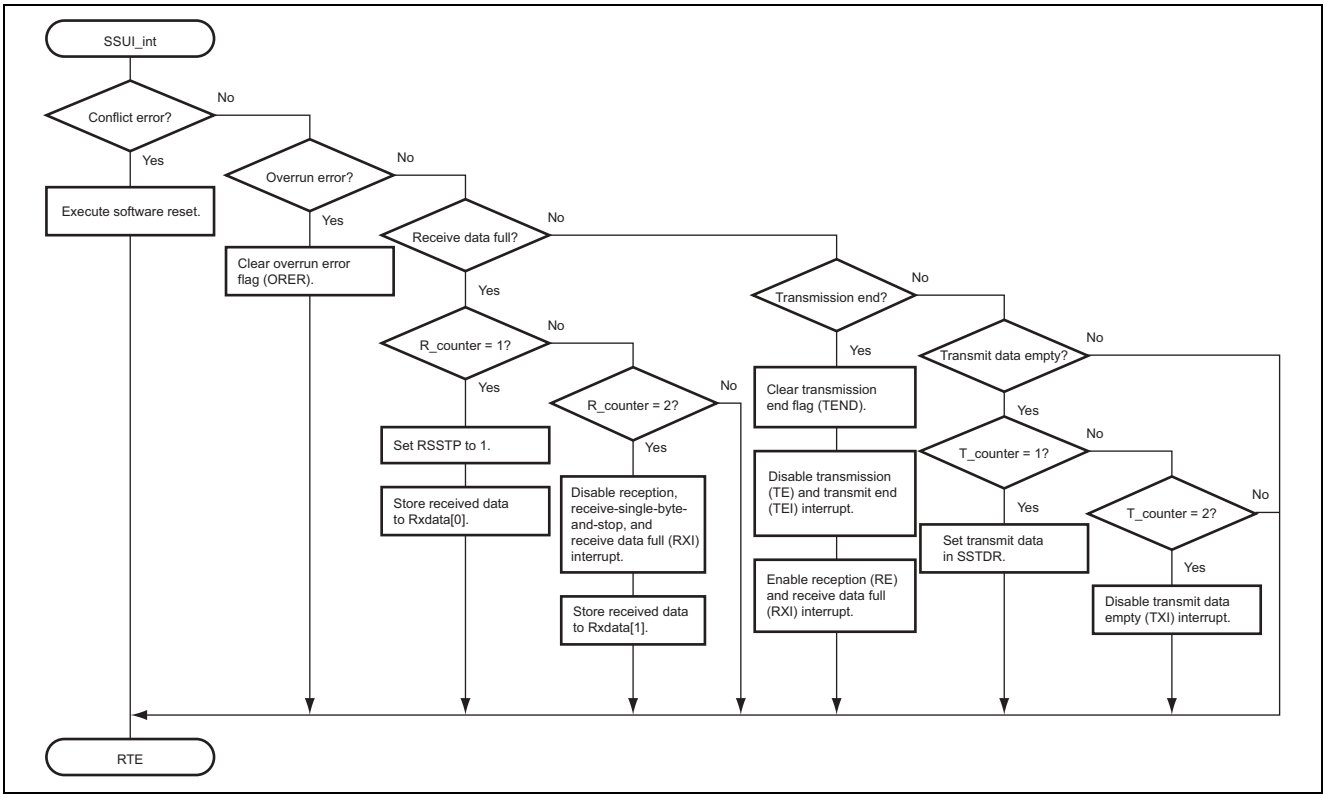
(2) **Initialization**



(3) Variable initialization

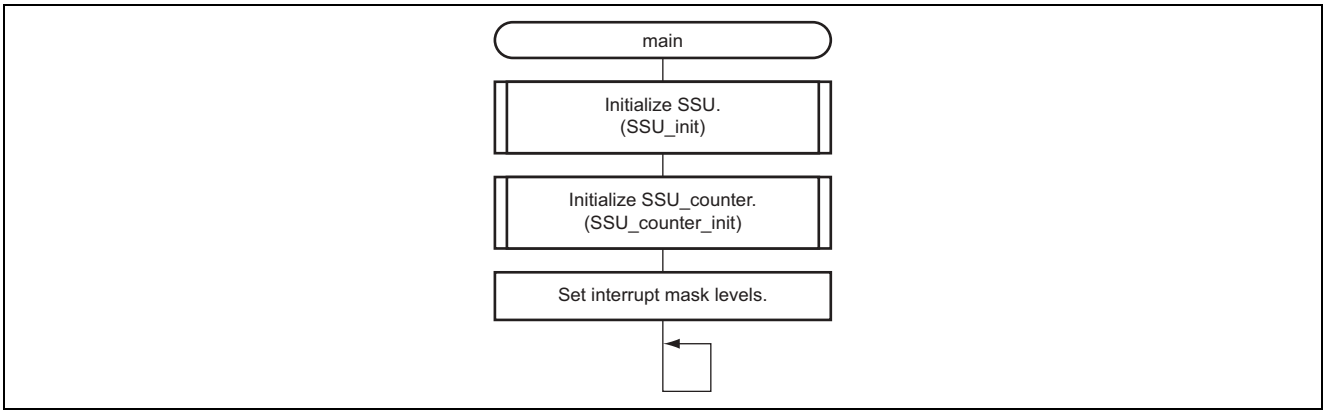


(4) Error processing, data reception, and data transmission

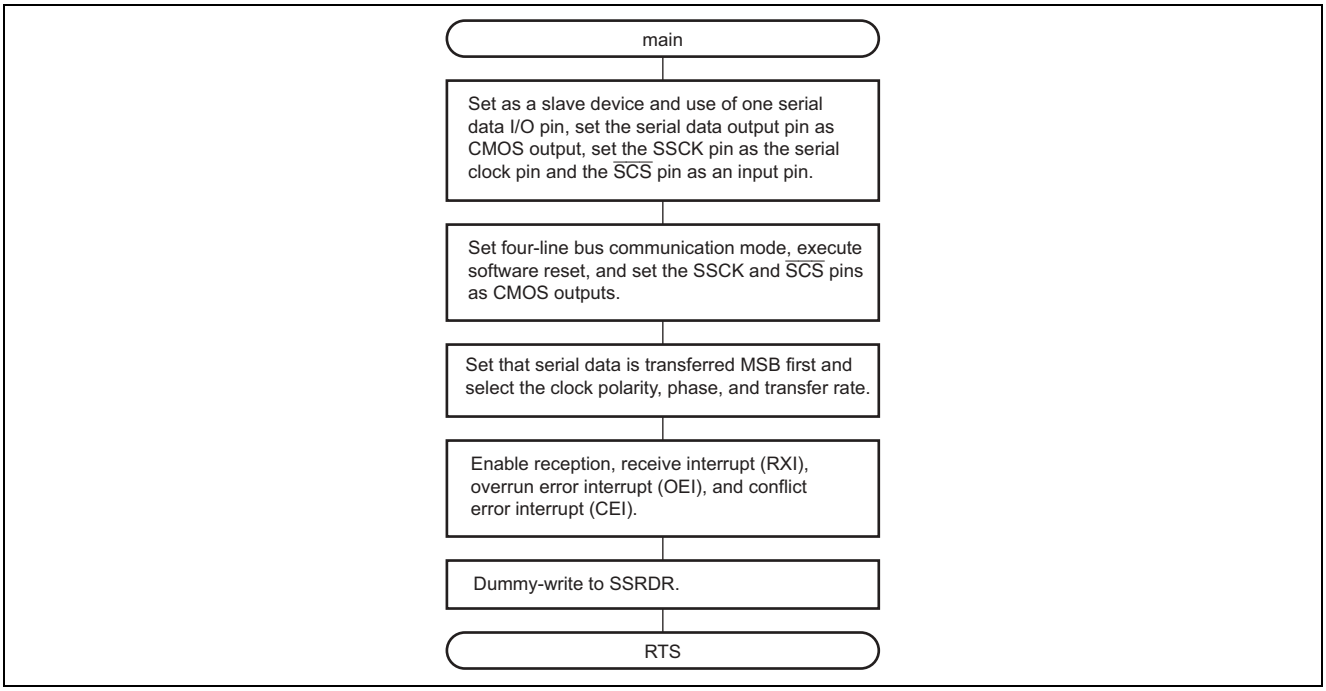


5.4.2 Slave

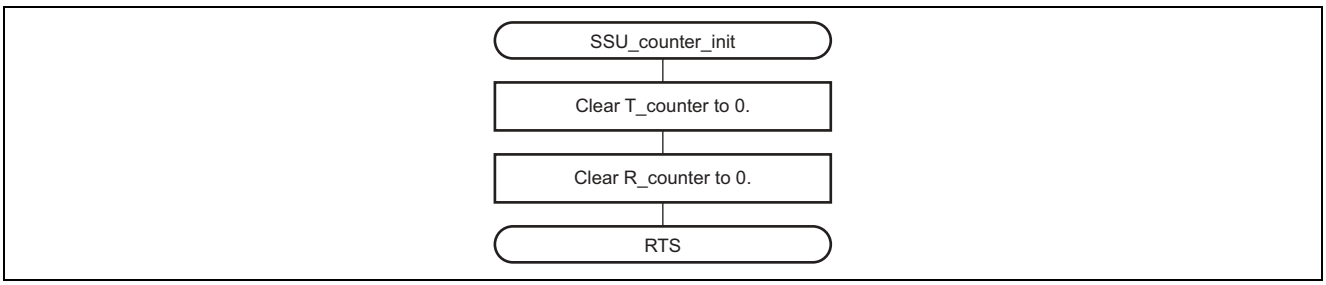
(1) **Main Routine**



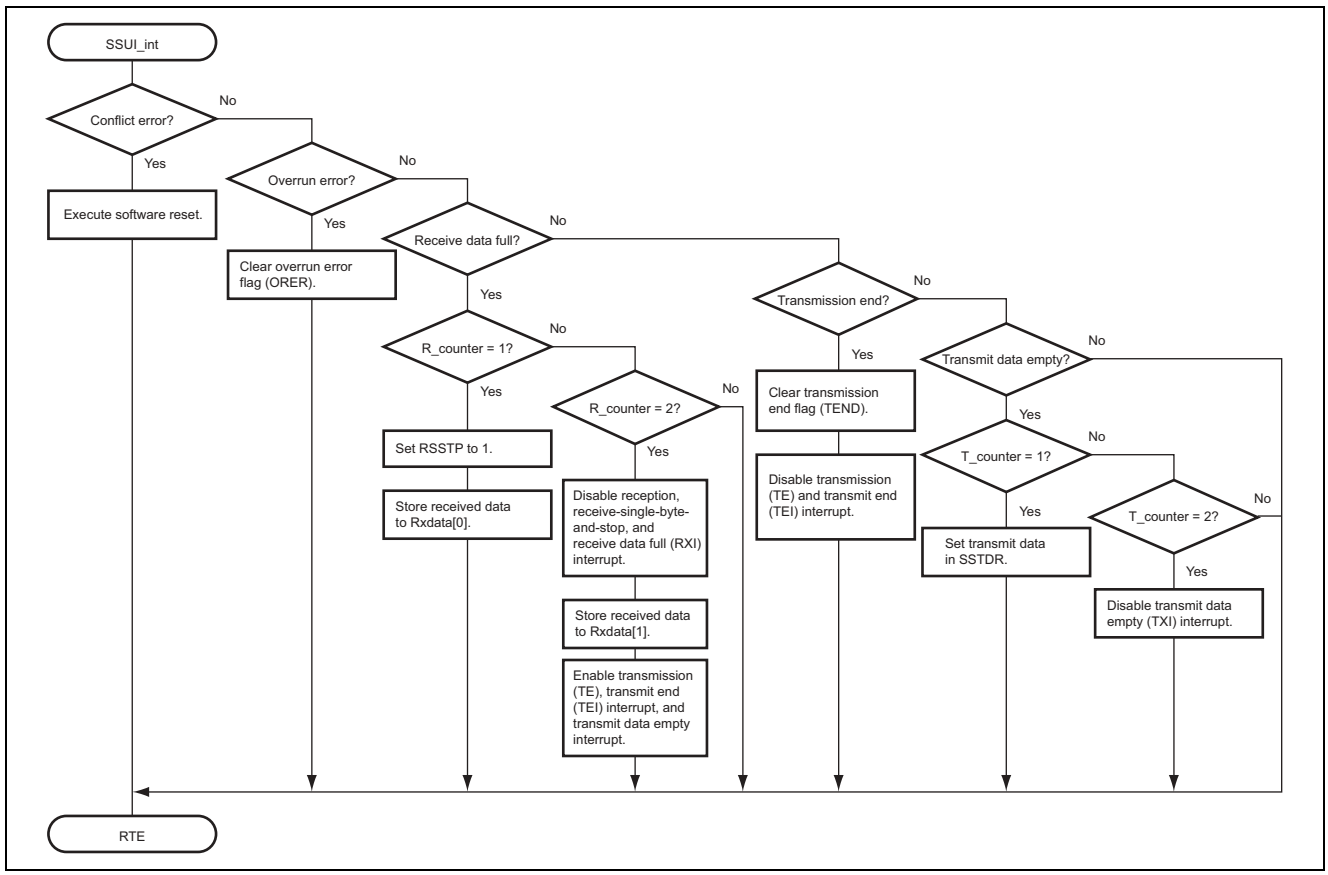
(2) **Initialization**



(3) **Variable initialization**



(4) **Error processing, data reception, and data transmission**



5.5 Program Listing

5.5.1 Master

```
#include <machine.h>
#include "H8_36057.h"

void main (void);
void SSU_init (void);
void SSU_counter_init(void);

unsigned char T_counter;
unsigned char R_counter;
unsigned char Rxdata[2];
/************************************************************************************************/
/*  Main Routine                                                                                */
/************************************************************************************************/
void main (void)
#pragma asm
    mov.l #H'FFFFFF7C,SP
#pragma endasm
{
    signed int lp;
    signed int A;
    SSU_init();
    SSU_counter_init();
    set_imask_ccr(0);

    while(SSU.SSER.BIT.TE);
    for(lp=50;lp>0;lp--);
    A = SSU.SSRDR;
    while(1);
}
/************************************************************************************************/
/*  SSU_counter Initialize Routine                                                              */
/************************************************************************************************/
void SSU_counter_init(void)
{
    T_counter = 0;
    R_counter = 0;
}
/************************************************************************************************/
/*  SSU Module Initialize Routine                                                               */
/************************************************************************************************/
void SSU_init(void)
{
    SSU.SSER.BYTE  = 0x00;
    SSU.SSCRL.BYTE = 0x00;
    SSU.SSCRH.BYTE = 0xCE;
    SSU.SSCRL.BYTE = 0x63;
    SSU.SSMR.BYTE  = 0xE0;
    SSU.SSSR.BYTE &= 0x04;
    SSU.SSER.BYTE  = 0x8D;
}
```

```
/*****************************************************************************************************/
#pragma interrupt (SSUI_int)    /* Error, reception, and transmission interrupts                */
/*****************************************************************************************************/
void SSUI_int(void)
{
    if(SSU.SSSR.BIT.CE && SSU.SSER.BIT.CEIE){
           SSU.SSSR.BIT.CE = 0;
           SSU.SSCRL.BIT.SRES = 1;
    }
    else if(SSU.SSSR.BIT.ORER && SSU.SSER.BIT.RIE){
            SSU.SSSR.BIT.ORER = 0;
    }
    if(SSU.SSSR.BIT.RDRF && SSU.SSER.BIT.RIE){
           R_counter++;
           if(R_counter == 1){
                  SSU.SSER.BIT.RSSTP = 1;
                  Rxdata[0] = SSU.SSRDR;
        }
        else if(R_counter == 2){
                SSU.SSER.BYTE &= 0x8D;
                Rxdata[1] = SSU.SSRDR;
        }
    }
    else if(SSU.SSSR.BIT.TEND && SSU.SSER.BIT.TEIE){
            SSU.SSSR.BIT.TEND = 0;
            SSU.SSER.BYTE &= 0x63;
            SSU.SSER.BYTE = 0x43;
    }
    else if(SSU.SSSR.BIT.TDRE && SSU.SSER.BIT.TIE){
            T_counter++;
            if(T_counter == 1){
                   SSU.SSTDR = 0xAA;
        }
        else if(T_counter == 2){
                SSU.SSTDR = 0x55;
                SSU.SSER.BIT.TIE = 0;
        }
    }
}
```

5.5.2 Slave

```
#include <machine.h>
#include "H8_36057.h"

void main (void);
void SSU_init (void);
void SSU_counter_init(void);

unsigned char T_counter;
unsigned char R_counter;
unsigned char Rxdata[2];
/*****************************************************************************************/
/*  Main Routine                                                                         */
/*****************************************************************************************/
void main (void)
#pragma asm
mov.l #H'FFFFFF7C,SP
#pragma endasm
{
    SSU_init();
    SSU_counter_init();
    set_imask_ccr(0);
    while(1);
}
/*****************************************************************************************/
/*  SSU_counter Initialize Routine                                                       */
/*****************************************************************************************/
void SSU_counter_init(void)
{
    R_counter = 0;
    T_counter = 0;
}
/*****************************************************************************************/
/*  SSU Module Initialize Routine                                                        */
/*****************************************************************************************/
void SSU_init(void)
{
    unsigned char A;
    SSU.SSER.BYTE  = 0x00;
    SSU.SSCRL.BYTE = 0x00;
    SSU.SSCRH.BYTE = 0x4D;
    SSU.SSCRL.BYTE = 0x63;
    SSU.SSMR.BYTE  = 0xE0;
    SSU.SSSR.BYTE &= 0x04;
    SSU.SSER.BYTE  = 0x43;
    A = SSU.SSRDR;
}
```

```
/*****************************************************************************************************/
#pragma interrupt (SSUI_int)      /* Error, reception, and transmission interrupts             */
/*****************************************************************************************************/
void SSUI_int(void)
{
    if(SSU.SSSR.BIT.CE && SSU.SSER.BIT.CEIE){
           SSU.SSSR.BIT.CE = 0;
           SSU.SSCRL.BIT.SRES = 1;
    }
    else if(SSU.SSSR.BIT.ORER && SSU.SSER.BIT.RIE){
            SSU.SSSR.BIT.ORER = 0;
    }
    else if(SSU.SSSR.BIT.RDRF && SSU.SSER.BIT.RIE){
            R_counter++;
            if(R_counter == 1){
                    Rxdata[0] = SSU.SSRDR;
        }
        else if(R_counter == 2){
                SSU.SSER.BYTE &= 0x8D;
                Rxdata[1] = SSU.SSRDR;
                SSU.SSER.BYTE = 0x8D;
        }
    }
    else if(SSU.SSSR.BIT.TEND && SSU.SSER.BIT.TEIE){
            SSU.SSSR.BIT.TEND = 0;
            SSU.SSER.BYTE &= 0x63;
    }
    else if(SSU.SSSR.BIT.TDRE && SSU.SSER.BIT.TIE){
            T_counter++;
            if(T_counter == 1){
                   SSU.SSTDR = 0xAA;
        }
        else if(T_counter == 2){
                SSU.SSTDR = 0x55;
                SSU.SSER.BIT.TIE = 0;
        }
    }
}
```

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jul.28.04	—	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.