RL78/G13

Handshake-based SPI Slave Transmission/Reception

Introduction

This application note describes how the serial array unit (SAU) performs slave transmission/reception by the simple SPI (CSI). The slave selected by the chip select ($\overline{\text{CS}}$) signal performs single transmission/reception, single transmission, or single reception according to the processing. The SAU also performs handshake processing using the BUSY signal.

Target Device

RL78/G13

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Contents

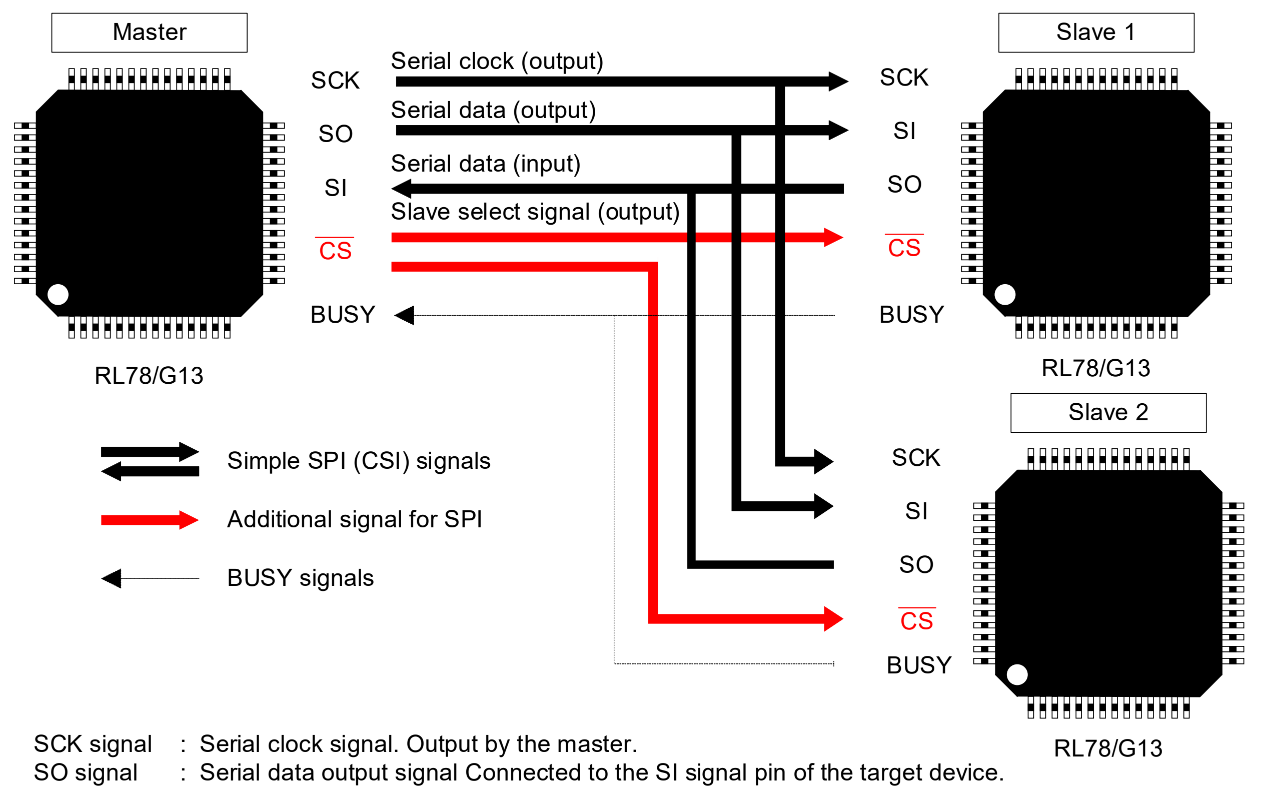
1. Specifications

The serial array unit (SAU) described in this application note performs CSI slave communication. A slave is selected by the chip select ($\overline{CS}$) signal from the master. The SAU performs handshake processing using the BUSY signal and performs slave communication.

1.1 Outline of CSI Communication

CSI communication is clock-synchronous serial communication using three signal lines, namely, serial clock (SCK), serial data input (SI), and serial data output (SO). SPI (Serial Peripheral Interface) uses an additional chip select ($\overline{CS}$) signal to select the slave device. The relationship among these signals is shown in Figure 1-1.

Figure 1-1 Outline of CSI Communication



SCK signal	:	Serial clock signal. Output by the master.
SO signal	:	Serial data output signal Connected to the SI signal pin of the target device.
SI signal	:	Serial data input signal. Connected to the SO signal pin of the target device.
$\overline{CS}$ signal	:	Used by the master device to select the target slave device.
BUSY signal	:	Handshake signal used in this application note.

Slaves wait until they are selected by the $\overline{CS}$ signal. When a slave is selected by the $\overline{CS}$ signal, that slave synchronizes with the SCK signal output from the master, outputs data to the SO signal line, and inputs data from the SI signal line.

In SPI/CSI communication, the slave needs to become ready for communication by the time the master starts communication (sending the SCK signals). In this application note, the slave notifies the master that it is ready for communication by setting the BUSY signal low.

1.2 Outline of Communication

In this application note, a command and communication for the command are performed at intervals of 1 ms. A set of a command and communication for the command is defined as a slot. Figure 1-2 shows an outline of slot processing and Table 1-1 lists the commands to be used.

Figure 1-2 Outline of Slots

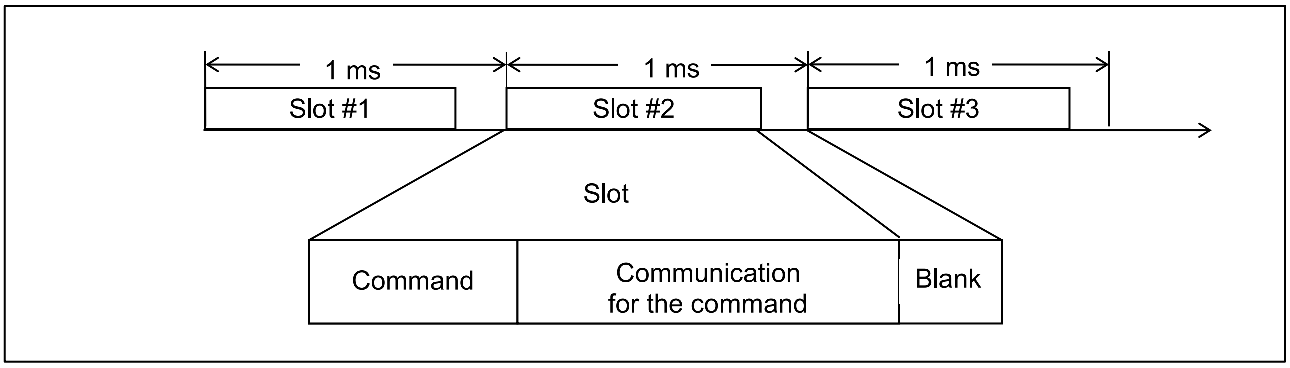


Table 1-1 Commands to be Used

Command	Outline of Operation
Status check	Checks the number of data characters that the slave can transmit or receive.
Receive	Receives data from the slave.
Transmit	Transmits data to the slave.
Transmit/receive	Transmits and receives data to and from the slave.

Table 1-2 lists the peripheral functions and their uses. Figure 1-3 and Figure 1-4 show the CSI communication operations.

Table 1-2 Peripheral Functions and Their Uses

Peripheral Function	Use
Serial array unit 0	Performs CSI slave communication using the SCK00 signal (clock output), SI00 signal (receive data), and SO00 signal (transmit data).
Port	Uses P00 to output the BUSY signal.
External interrupt	Uses INTP0/P137 to detect the $\overline{CS}$ signal.

Figure 1-3 Timing chart for status check commands

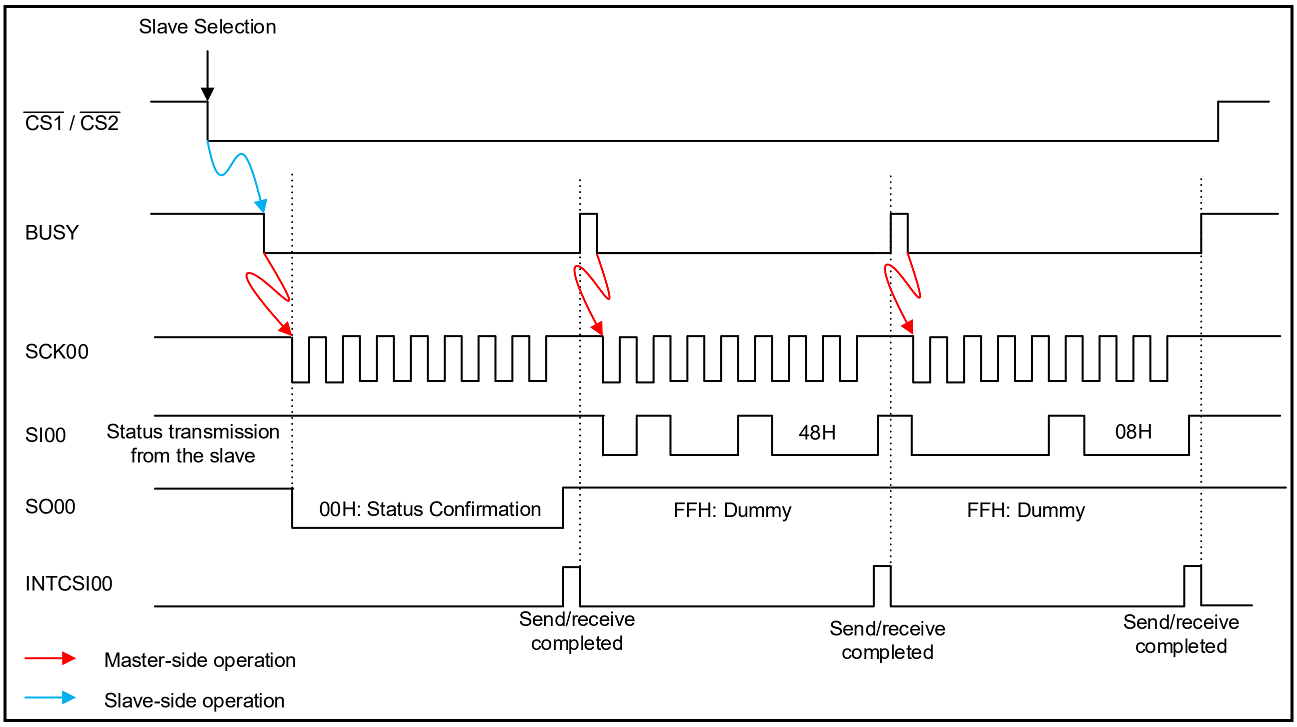
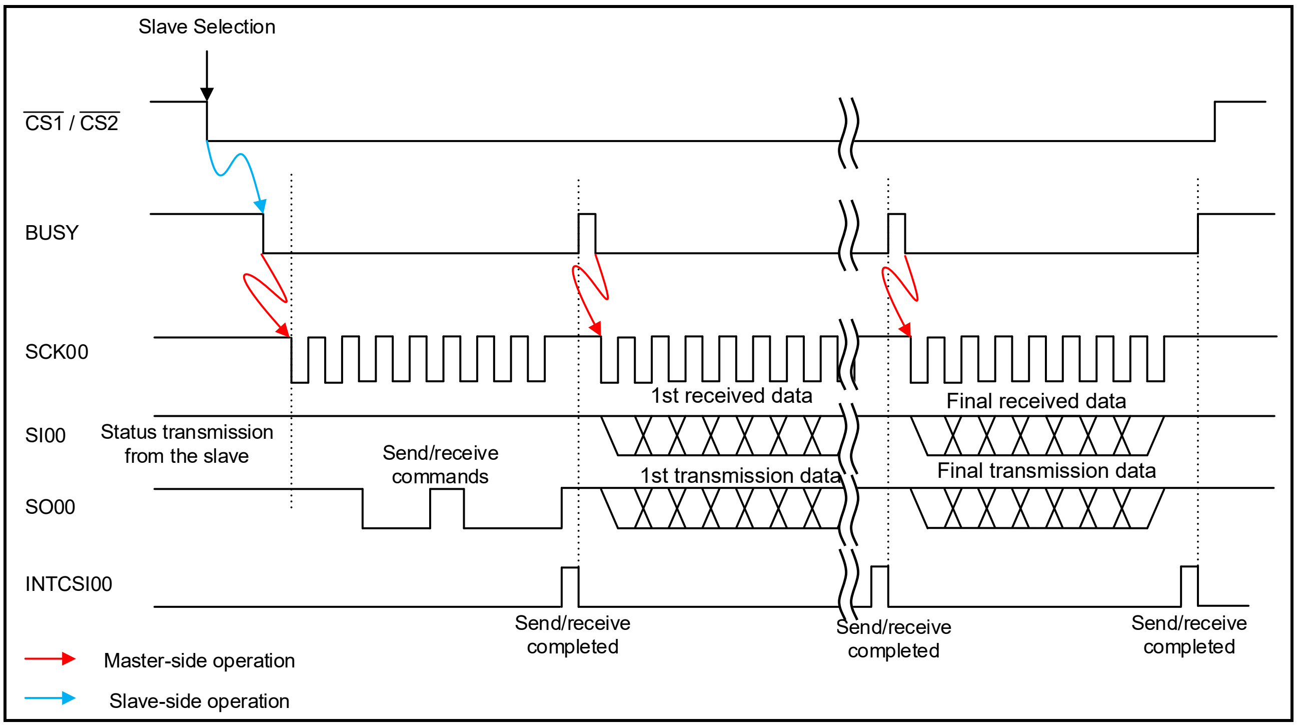


Figure 1-4 Timing chart of commands sent and received



1.3 Communication Format

Table 1-3 lists the characteristics of the CSI communication format that is used in the sample code.

Table 1-3 Communication Format

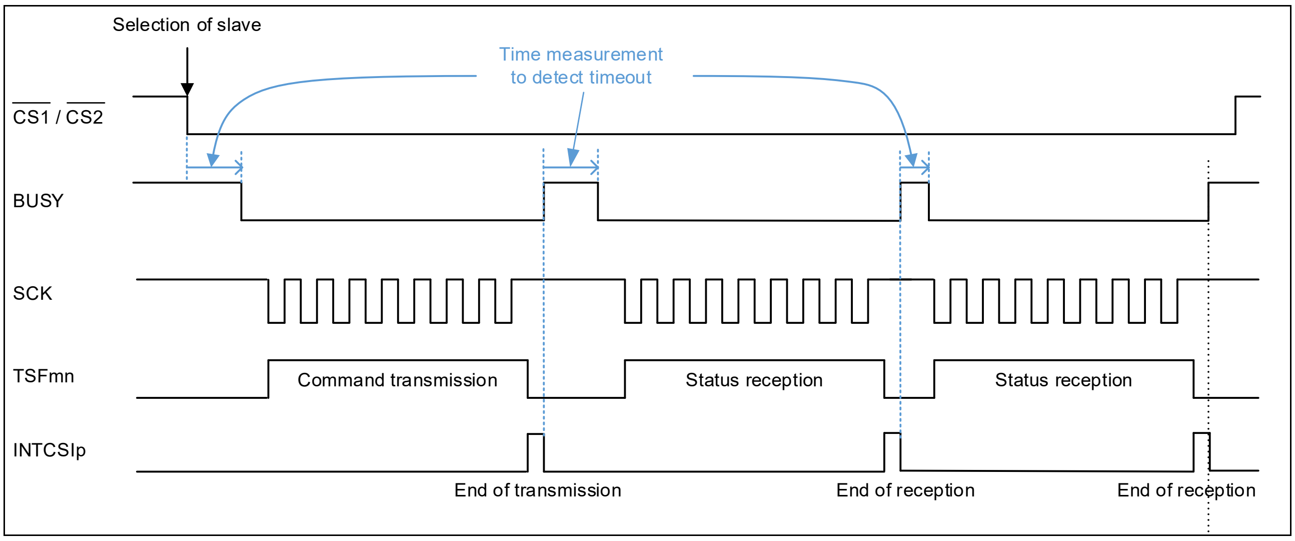
Item	Specification	Remarks
Communication speed	1 Mbps	About 200 kbps at minimum
Data bit length	8 bits/character	
Transfer order	MSB first	
Communication type	Type 1	
Communication mode	Single transfer	
Communication direction	Receive/transmit/transmit and receive	
Maximum number of characters transferred	63 characters/slot	32 characters by default

1.4 Handshake

Generally, SPI communication-dedicated slave devices, such as EEPROM, are always communication-ready and therefore do not require the BUSY signal. However, when using a general-purpose device, such as an MCU, to perform slave communication, processing time required by the software must be secured. In this application note, handshaking is performed using the BUSY signal so that SPI communication can be established even when the load exceeds the processing capability of a general-purpose device. Additionally, a timeout of 16 μs is set up so that the system does not stop even when no response using the BUSY signal is returned.

Figure 1-5 shows an example of handshaking for status checking. The slave selected by the $\overline{CS}$ signal sets the BUSY signal low after completing the preparation of slave communication. The slave sets the BUSY signal high when the command reception from the master is completed. Then, the slave sets the BUSY signal low again after completing the preparation of slave communication according to the command.

Figure 1-5 Handshaking Example



1.5 Specification Details

This sample code, after completion of initialization, transitions to standby mode.$\overline{CS}$ The slave selected by the $\overline{CS}$ signal sets the BUSY signal low after completing the preparation of communication. The slave sets the BUSY signal high when the command reception from the master is completed. After communication is completed, the slave sets the BUSY signal high and prepares for the next communication. While the slave is selected by the $\overline{CS}$ signal, it controls data transmission/reception and the BUSY signal.

(1) Initialize the port.

<Conditions for setting the port>

- Use P00 controlling the BUSY signal in N-ch open drain output mode.
- Use P137/INTP0 detecting the $\overline{CS}$ signal as an input port.

(2) Initialize the external interrupt function.

<Conditions for setting external interrupt>

- Use INTP0 for both edges.
- Set the priority of INTP0 to the lowest level (3, by default).

(3) Initialize the CSI.

<Conditions for setting the CSI>

- Use SAU0 channel 0 as CSI00.
- Use the external clock as the transfer clock.
- Assign the clock input to the P10/SCK00 pin, the data input to the P11/SI00 pin, and the data output to the P12/SO00 pin.
- Use single transfer mode as the transfer mode.
- Set the data length to 8 bits.
- Set the phase between the data and clock to type 1.
- Set the order of data transfer mode to MSB first.
- Use transmission end and reception end interrupts as the interrupt (INTCSI00).
- Set the priority of the interrupt (INTCSI00) to the lowest level (3, by default).

(4) After initialization is completed, the slave performs communication as shown in the following steps.

① The slave waits in standby state (HALT mode) until it is selected by the $\overline{CS}$ signal.

② An external interrupt INTP0 occurs by the falling edge of the $\overline{CS}$ signal, and the slave is released from HALT mode.

③ The slave enables CSI00 for operation, enables SO00 for output, sets the BUSY signal low, and waits for the start of command reception processing.

④ The slave separates the receive data into the upper 2 bits containing the command from the master, and the remaining lower 6 bits.

⑤ The slave branches into the master command processing, status check, data reception, transmission, or transmission/reception.

⑥ The slaves executes the processing specified by the command.

⑦ After the processing is completed, the slave waits in standby state (HALT mode) until the $\overline{CS}$ signal goes high.

⑧ An external interrupt INTP0 occurs by the rising edge of the $\overline{CS}$ signal, and the slave is released from HALT mode. After confirming that the $\overline{CS}$ signal is high, disable CSI00 for operation and disable SO00 for output. These steps are subsequently repeated from step 1.

(5) Command reception
Each communication operation begins with the reception of a 1-byte command. Table 1-4 lists the command formats.

Table 1-4 Command Formats

Command Code		Command Outline
Status check	00000000B	Checks the number of data characters that the slave can transmit or receive. The following responses can be made by the slave: 01xxxxxxB: The number of characters that the slave can transmit is xxxxxxB. 00xxxxxxB: The number of characters that the slave can receive is xxxxxxB.
Reception	01xxxxxxB	The master receives xxxxxxB bytes of data.
Transmission	10xxxxxxB	The master transmits xxxxxxB bytes of data.
Transmission/reception	11xxxxxxB	The master transmits and receives xxxxxxB bytes of data.

2. Operation Confirmation Conditions

The operation of the sample code provided with this application note has been tested under the following conditions.

Table 2-1 Operation Confirmation Conditions

Item Description	Item Description
MCU used	RL78/G13（R5F100LE）
Operating frequency	● High-speed on-chip oscillator clock (fIH): 32 MHz ● CPU/peripheral hardware clock: 32 MHz
Operating voltage	During VDD operation: 5.0 V（2.7V～5.5V） LVD0 detection voltage: Reset mode At rising edge TYP. 1.88 V (1.84 V to 1.91 V) At falling edge TYP. 1.84 V (1.80 V to 1.87 V)
Integrated development environment (CS+)	CS+ V8.09.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.12.00 from Renesas Electronics Corp
Integrated development environment (e2studio)	e2 studio Version: 2023-01 (23.1.0) from Renesas Electronics Corp.
C compiler (e2studio)	CC-RL V1.12.00 from Renesas Electronics Corp.
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.21.2 from IAR Systems Corp.
C compiler (IAR)	IAR C/C++ Compiler for Renesas RL78 V4.21.2.2420 from IAR Systems Corp.
Board used	RL78/G13 (R5F100LE) Target Board（QB-R5F100LE-TB）

3. Related Application Notes

See also the following application notes, which are related to this application note:

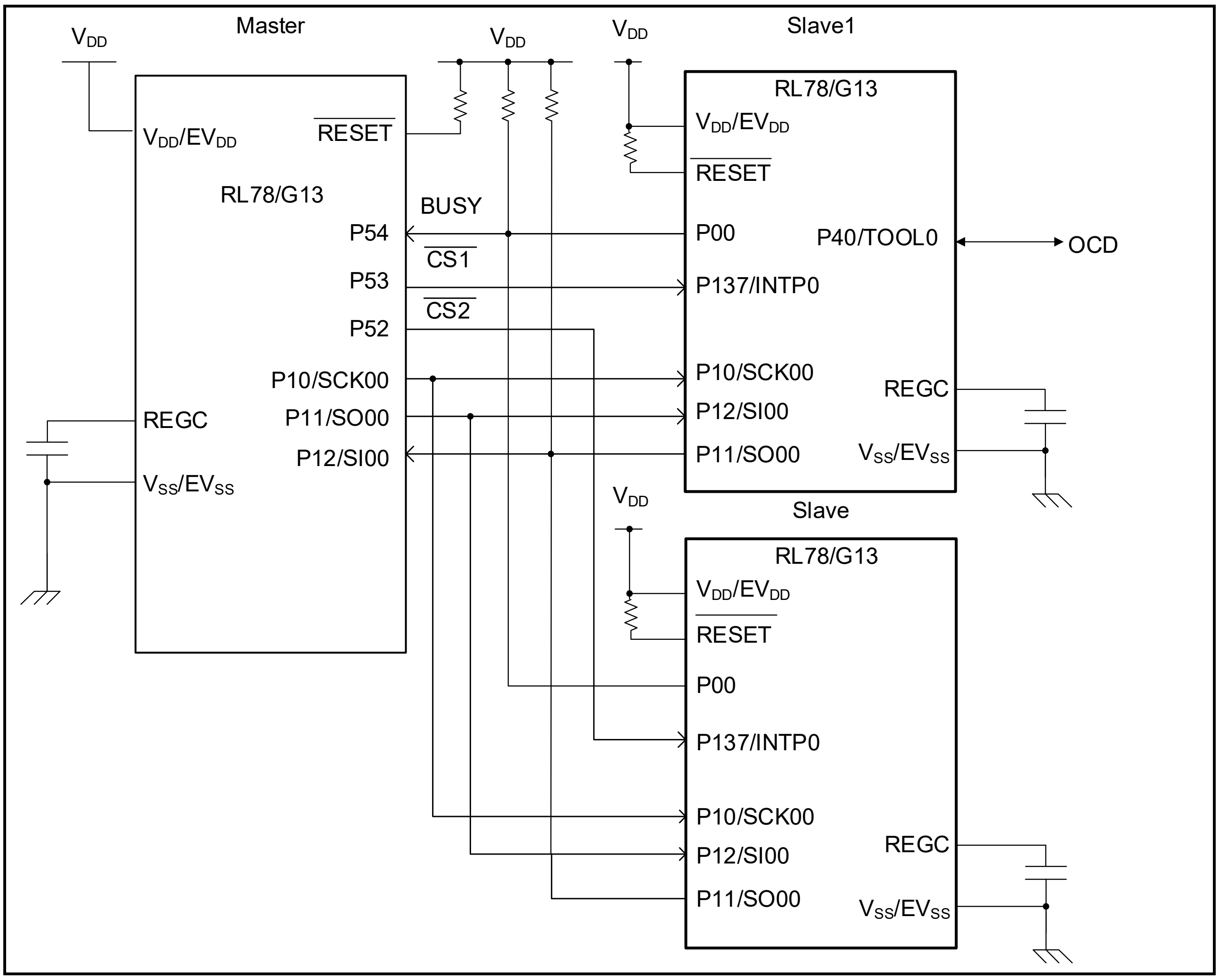
RL78/G13 Handshake-based SPI Master Transmission/Reception (R01AN6883E) APPLICATION NOTE

4. Hardware Descriptions

4.1 Example of Hardware Configuration

Figure 4-1shows an example of the hardware configuration used in the application note.

Figure 4-1 Hardware Configuration



Note 1. This schematic circuit diagram is simplified to show the outline of connections. When creating circuits, design them so that they meet electrical characteristics by properly performing pin processing. (Connect input-only ports to V_{DD} or V_{SS} individually through a resistor.)

Note 2. Connect pins (with a name beginning with EV_{SS}), if any, to V_{SS}, and connect pins (with a name beginning with EV_{DD}), if any, to V_{DD}.

4.2 List of Pins to be Used

Table 4-1 lists the pins to be used and their functions.

Table 4-1 Pins to be Used and Their Functions

Pin Name	I/O	Description
P10/EI10/EO10/SCK00/SCL00/(TI07)/(TO07)	Output	Serial clock output pin
P11/EI11/EO11/SI00/RxD0/TOOLRxD/SDA00/(TI06)/(TO06)	Input	Data reception pin
P12/EI12/EO12/SO00/TxD0/TOOLTxD/(INTP5)/(TI05)/(TO05)	Output	Data transmission pin
P00/TS26/EI00/TI00	Output	BUSY signal output to the master
P137/EI137/INTP0	Input	$\overline{CS}$ signal detection

Caution In this application note, only the used pins are processed. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.

5. Description of the Software

5.1 List of Option Byte Settings

Table 5-1 summarizes the settings of the option bytes.

Table 5-1 Option Byte Settings

Address	Setting Value	Contents
000C0H	1110 1111B (EFH)	Stops the watchdog timer operation. (Stops counting after the release of the reset state.)
000C1H	0011 1111B (3FH)	LVD reset mode Detection Voltage: On the rising edge: TYP. 1.88V On the falling edge: TYP. 1.84V
000C2H	11101000B (E8H)	HS mode, HOCO: 32 MHz
000C3H	10000101B (85H)	Enables the on-chip debugging function.

5.2 List of Constants

Table 5-2 lists the constants that are used in the sample code.

Table 5-2 Constants Used in the Sample Code

Constant Name	Definition location	Setting Value	Contents
CS_pin	r_cg_userdefine.h	P13_bit.no7	Port register for the $\overline{CS}$ signal
PM_SO00	r_cg_userdefine.h	PM1_bit.no2	Port mode register for the SO00 signal
PM_BUSY	r_cg_userdefine.h	PM0_bit.no0	Port mode register for the BUSY signal
BUSYOUT	r_cg_userdefine.h	P0_bit.no0	Port register for the BUSY signal
TX_NUM	r_main.c	32	Number of data characters that can be transmitted
RX_NUM	r_main.c	32	Number of data characters that can be received
data_length	r_main.c	1	Data length
TX_DATA[]	r_main.c	*1	Stores 63 characters of transmit data, the maximum number of characters transferred.

Note: 1. In this application note, ASCII codes from 0x40 to 0x7F are stored.

5.3 List of Variables

Table 5-3 lists the global variables that are used in this sample code.

Table 5-3 Global Variables Used in the Sample Code

Type	Variable Name	Contents	Function Used
uint8_t	g_tx_data	Buffer for transmit data	r_main.c
uint8_t	g_rx_data	Buffer for receive data	r_main.c
uint8_t	g_mode_check	Mode check flag	r_main.c
uint8_t	g_num	Number of characters to be transmitted/received requested by the master	r_main.c
uint8_t	g_rx_data_stored[]	Stores receive data.	r_main.c

5.4 List of Functions

Table 5-4 lists the functions that are used in the sample code.

Table 5-4 Functions

Function Name	Outline	Source file
R_Systeminit	Initialization process	r_systeminit.c
R_CGC_Get_ResetSource	Reset source reading	r_cg_cgc_user.c
R_CGC_Create	CPU clock Configuration	r_cg_cgc.c
R_PORT_Create	Port Configuration	r_cg_port.c
R_SAU0_Create	SAU Configuration	r_cg_serial.c
R_CSI00_Create	CSI00 Configuration	r_cg_serial.c
R_INTC_Create	Interruption Configuration	r_cg_timer.c
main	Main processing	r_main.c
R_MAIN_UserInit	User Main Initialization	r_main.c
R_INTC0_Start	INTP0 start operating	r_cg_intc.c
CSI00_Status_check	CSI status check	r_main.c
CSI00_Send_Receive	CSI transmission/reception	r_main.c
CSI00_Send	CSI transmission (master receives data)	r_main.c
CSI00_Receive	CSI reception (master transmits data)	r_main.c
R_CSI00_Send_Receive	CSI00 Transmit/Receive Processing	r_cg_serial.c
r_csi00_interrupt	INICSI00 interrupt processing	r_cg_serial_user.c
r_CSI00_callback_sendend	Setting BUSY signal high	r_cg_serial_user.c

5.5 Function Specifications

The following tables list the sample code function specifications.

[Function Name] R_Systeminit

Synopsis	Initialization process
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_serial.h r_cg_timer.h r_cg_userdefine.h
Declaration	void R_Systeminit (void)
Explanation	Initialize each peripheral function.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] R_CGC_Get_ResetSource

Synopsis	Reset source reading
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_userdefine.h
Declaration	void R_CGC_Get_ResetSource(void)
Explanation	Reads the reset source.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] R_CGC_Create

Synopsis	CPU clock Configuration
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_userdefine.h
Declaration	void R_CGC_Create (void)
Explanation	Set the CPU clock.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] R_PORT_Create

Synopsis	Port Configuration
Header	r_cg_macrodriver.h r_cg_port.h r_cg_userdefine.h
Declaration	void R_CGC_Create (void)
Explanation	Configure the port settings.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] R_SAU0_Create

Synopsis	SAU Configuration
Header	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h
Declaration	void R_SAU_Create (void)
Explanation	Configure SAU settings.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] R_CSI00_Create

Synopsis	CSI00 Configuration
Header	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h
Declaration	void R_CSI00_Create (void)
Explanation	Configure CSI00 settings.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] R_INTC_Create

Synopsis	Interruption Configuration
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_userdefine.h
Declaration	void R_INTC_Create (void)
Explanation	Configure INTP settings.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] main

Synopsis	Main processing
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_serial.h r_cg_userdefine.h
Declaration	void main(void)
Explanation	Starts the operation of INTP0. Starts CSI communication and receives a command from the master when INTP0 occurs. Branch into the processing corresponding to the received command.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] R_MAIN_UserInit

Synopsis	User Main Initialization
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_serial.h r_cg_userdefine.h
Declaration	void R_MAIN_UserInit(void)
Explanation	Main user initial settings.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] R_INTC0_Start

Synopsis	INTP0 start operating
Header	r_cg_macrodriver.h r_cg_intc.h r_cg_userdefine.h
Declaration	void R_INTC0_Start (void)
Explanation	Starts operation of INTP0.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] CSI00_Status_check

Synopsis	CSI status check
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_serial.h r_cg_userdefine.h
Declaration	void CSI00_Status_check (void)
Explanation	Checks the status of the slave.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] CSI00_Send_Receive

Synopsis	CSI transmission/reception
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_serial.h r_cg_userdefine.h
Declaration	void CSI00_Send_Receive (void)
Explanation	Performs the slave transmission/reception processing.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] CSI00_Send

Synopsis	CSI transmission
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_serial.h r_cg_userdefine.h
Declaration	void CSI00_Send (void)
Explanation	Performs the slave transmission processing.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] CSI00_Receive

Synopsis	CSI reception
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_intc.h r_cg_serial.h r_cg_userdefine.h
Declaration	void CSI00_Receive (void)
Explanation	Performs the slave reception processing.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] R_CSI00_Send_Receive

Synopsis	CSI00 Transmit/Receive Processing
Header	r_cg_macrodriver.h、r_cg_serial.h、r_cg_userdefine.h
Declaration	MD_STATUS R_CSI00_Send_Receive(uint8_t * const tx_buf、uint16_t tx_num、uint8_t * const rx_buf)
Explanation	Configure CSI00 data sending/receiving settings.
Arguments	• uint8_t * const tx_buf : [Address of sent data buffer] • uint16_t tx_num : [Sent data buffer size] • uint8_t * const rx_buf : [Address of received data buffer]
Return value	• [MD_OK]：Sending/receiving settings complete • [MD_ARGERROR]：Sending/receiving settings failure
Remarks	None

[Function Name] r_csi00_interrupt

Synopsis	INICSI00 interrupt processing
Header	r_cg_macrodriver.h、r_cg_serial.h、r_cg_userdefine.h
Declaration	static void __near r_csi00_interrupt(void)
Explanation	If there is unsent data, reads the received data and starts sending the unsent data. If there is no unsent data, reads the received data.
Arguments	• None
Return value	• None
Remarks	None

[Function Name] r _CSI00_callback_sendend

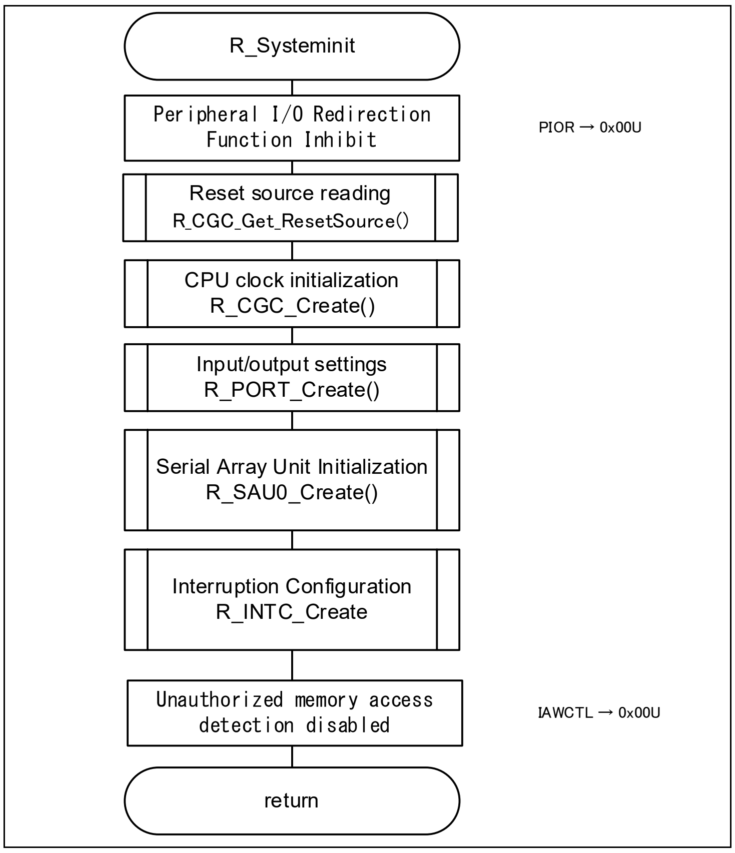
Synopsis	Setting BUSY signal high
Header	r_cg_macrodriver.h、 r_cg_serial.h、 r_cg_userdefine.h
Declaration	static void r _CSI00_callback_sendend(void)
Explanation	Sets the BUSY signal high.
Arguments	• None
Return value	• None
Remarks	None

5.6 Flowcharts

5.6.1 Initialization process

Figure 5-1 shows the flow of Initialization process.

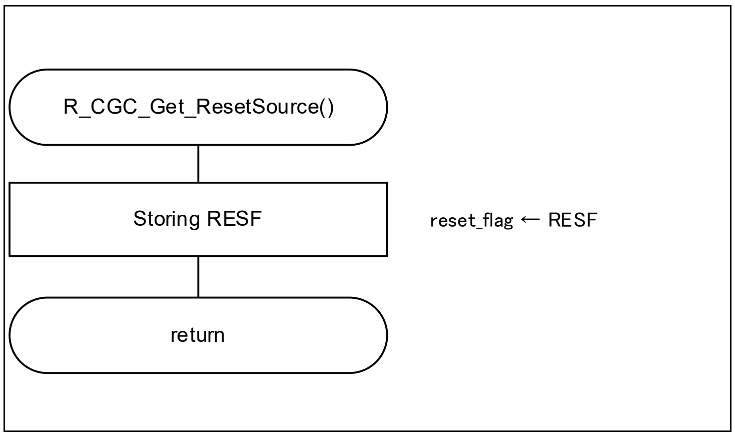
Figure 5-1 Initialization process


R_Systeminit
Peripheral I/O Redirection Function Inhibit
PIOR → 0x00U
Reset source reading R_CGC_Get_ResetSource()
CPU clock initialization R_CGC_Create()
Input/output settings R_PORT_Create()
Serial Array Unit Initialization R_SAU0_Create()
Interruption Configuration R_INTC_Create
Unauthorized memory access detection disabled
IAWCTL → 0x00U
return

5.6.2 Reset source reading

Figure 5-2 shows the flow of Reset source reading

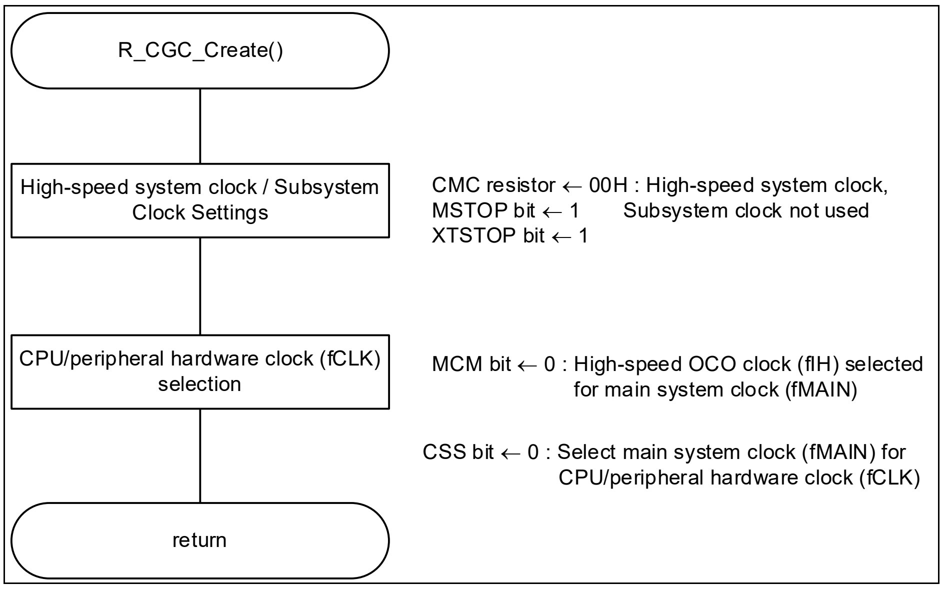
Figure 5-2 Reset source reading



5.6.3 CPU clock Configuration

Figure 5-3 shows the flow of CPU clock Configuration.

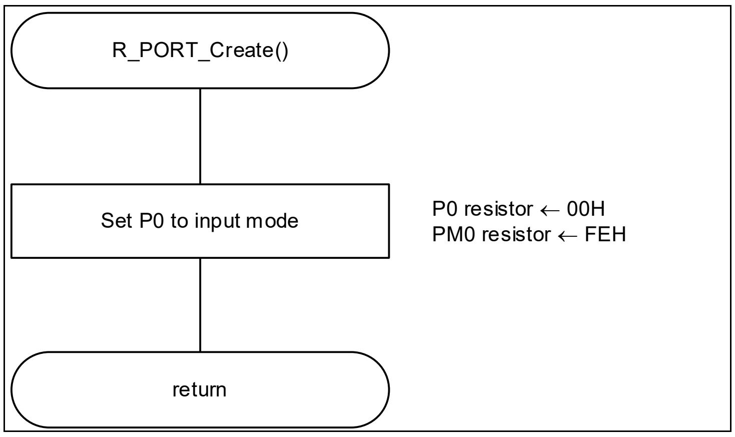
Figure 5-3 CPU clock Configuration


R_CGC_Create()
High-speed system clock / Subsystem Clock Settings
CMC resistor ← 00H : High-speed system clock,
MSTOP bit ← 1 Subsystem clock not used
XTSTOP bit ← 1
CPU/peripheral hardware clock (fCLK) selection
MCM bit ← 0 : High-speed OCO clock (fIH) selected for main system clock (fMAIN)
CSS bit ← 0 : Select main system clock (fMAIN) for CPU/peripheral hardware clock (fCLK)
return

5.6.4 Port Configuration

Figure 5-4 shows the flow of Port Configuration.

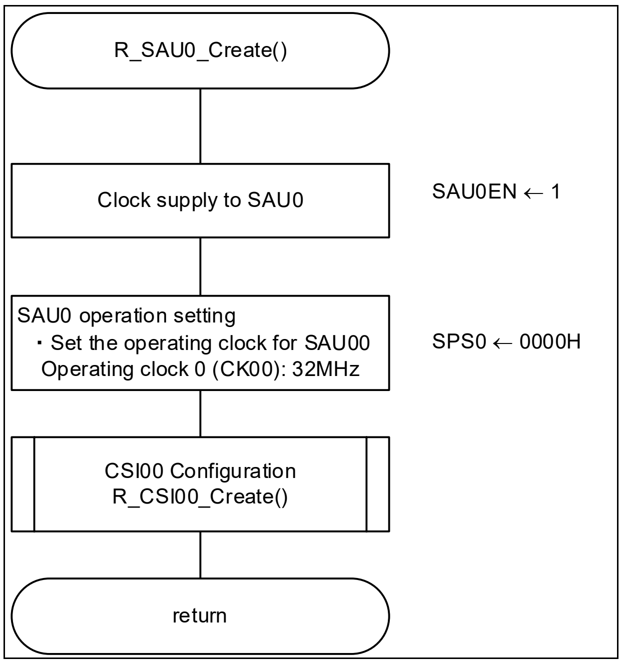
Figure 5-4 Port Configuration



5.6.5 SAU Configuration

Figure 5-5 shows the flow of SAU Configuration.

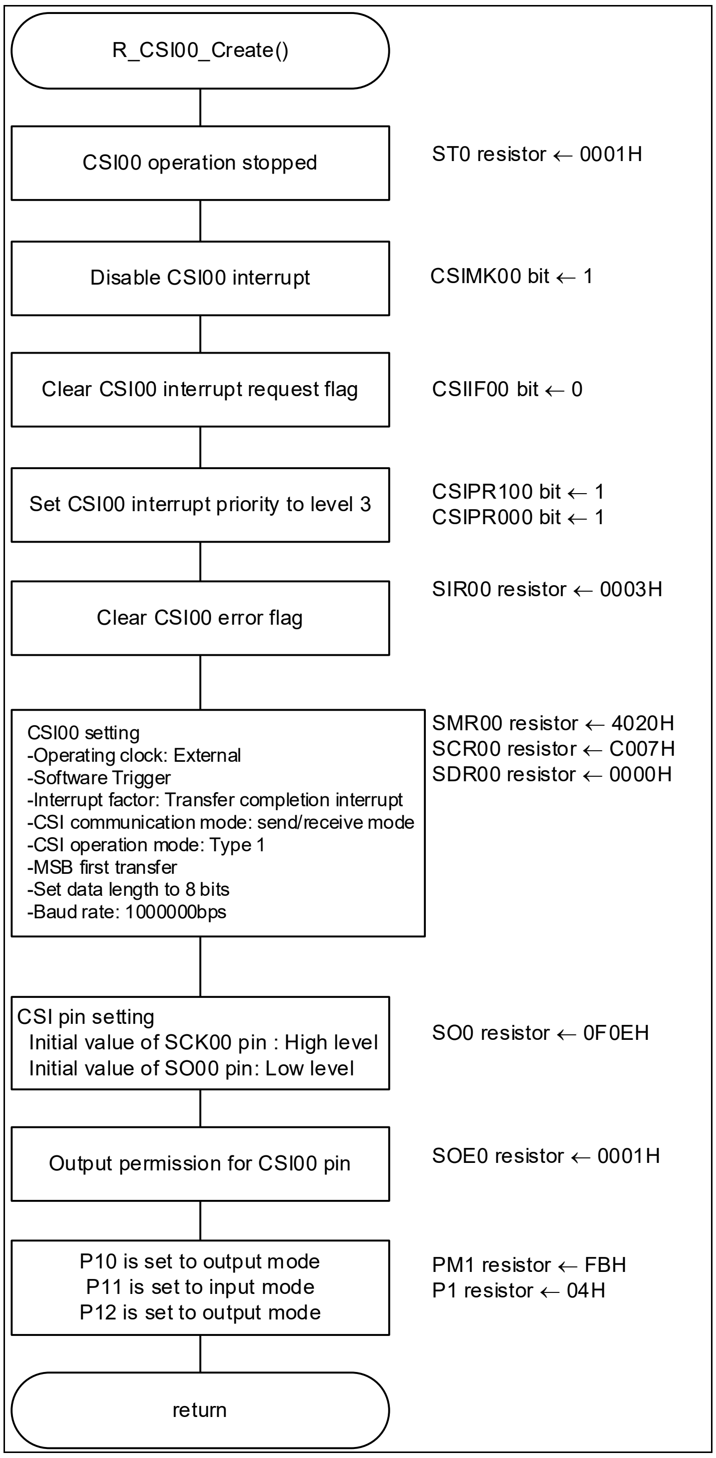
Figure 5-5 SAU Configuration



5.6.6 CSI00 Configuration

Figure 5-6 shows the flow of CSI00 Configuration.

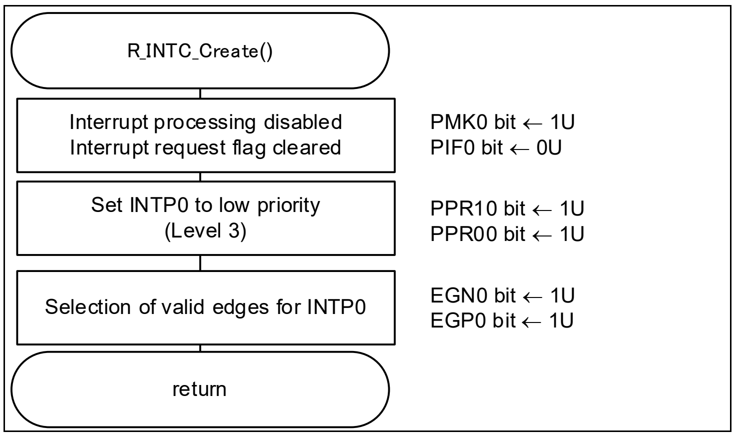
Figure 5-6 CSI00 Configuration



5.6.7 Interruption Configuration

Figure 5-7 shows the flow of TAU0 Configuration.

Figure 5-7 Interruption Configuration


R_INTC_Create()
Interrupt processing disabled
Interrupt request flag cleared
PMK0 bit ← 1U
PIF0 bit ← 0U
Set INTP0 to low priority (Level 3)
PPR10 bit ← 1U
PPR00 bit ← 1U
Selection of valid edges for INTP0
EGN0 bit ← 1U
EGP0 bit ← 1U
return

5.6.8 Flowchart of Main Processing

Figure 5-8 to Figure 5-9 show the overall flow of processing in this application note.

Figure 5-8 Main Processing 1/2

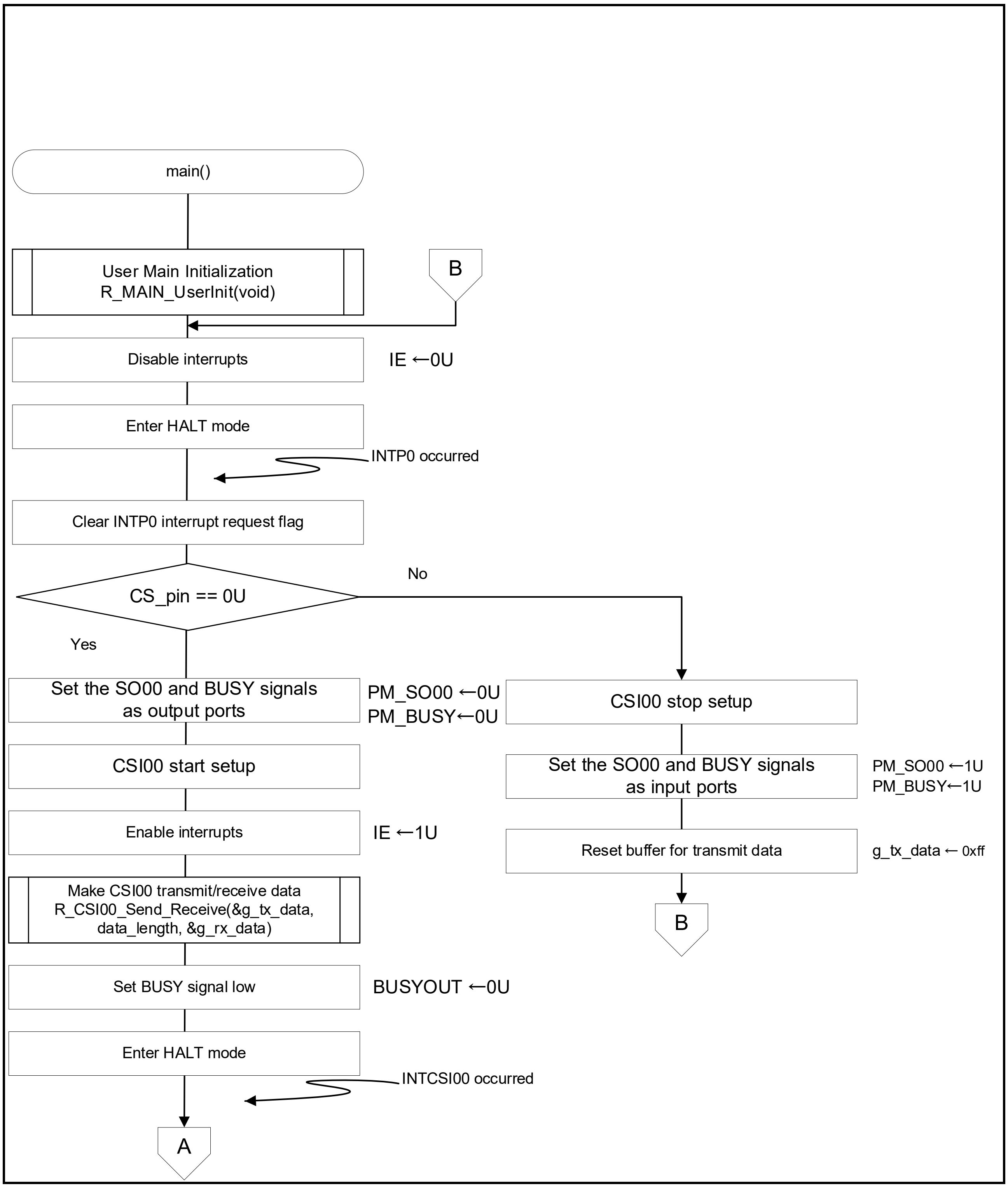
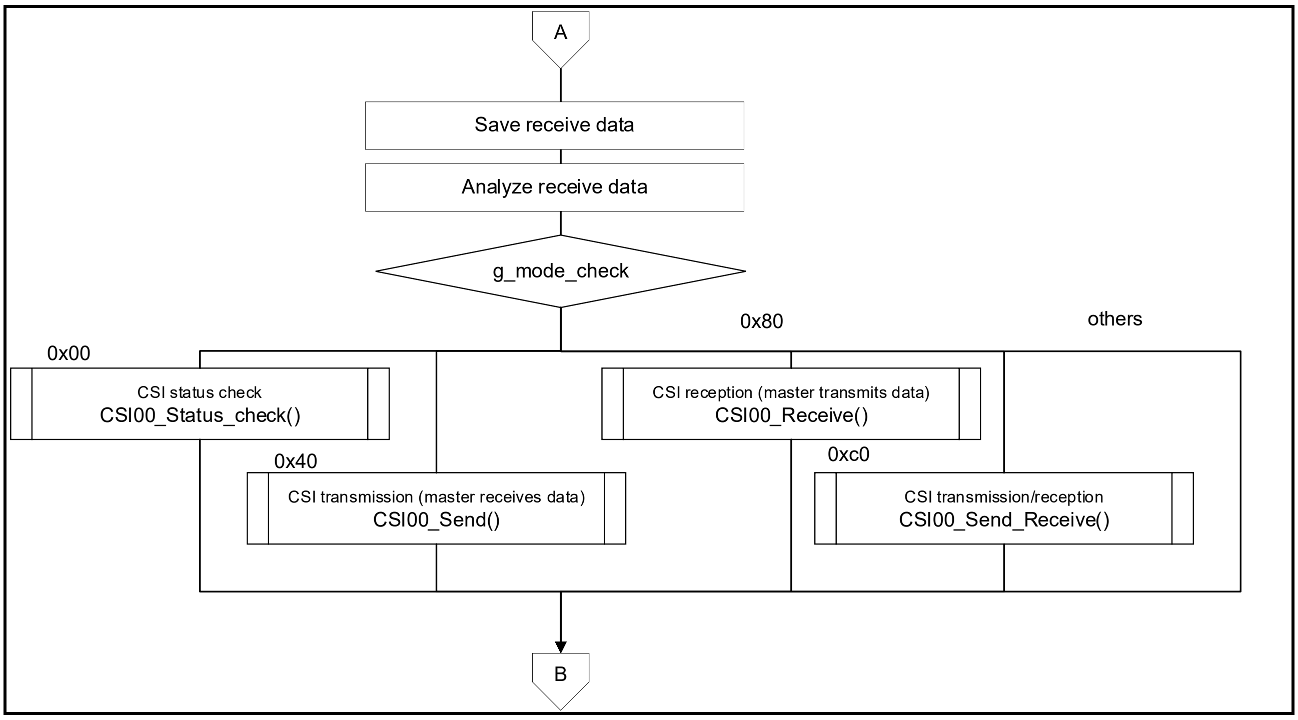


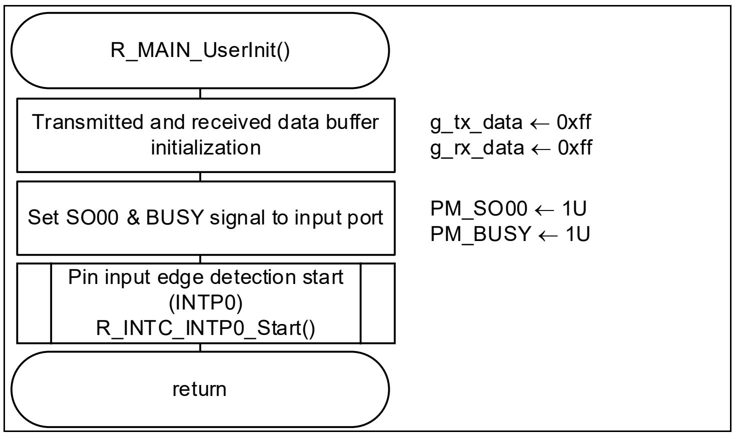
Figure 5-9 Main Processing 2/2



5.6.9 User Main Initialization

Figure 5-10 shows the flow of User Main Initialization.

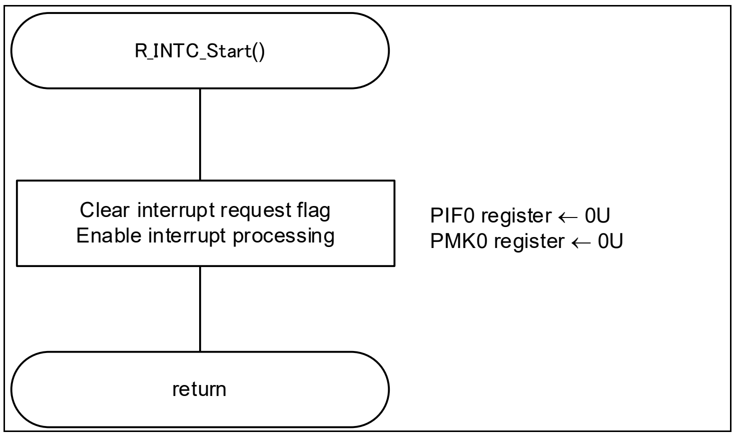
Figure 5-10 User Main Initialization


R_MAIN_UserInit()
Transmitted and received data buffer initialization
g_tx_data ← 0xff
g_rx_data ← 0xff
Set SO00 & BUSY signal to input port
PM_SO00 ← 1U
PM_BUSY ← 1U
Pin input edge detection start (INTP0)
R_INTC_INTP0_Start()
return

5.6.10 INTP0 start operating

Figure 5-11 shows the flow of User Main Initialization.

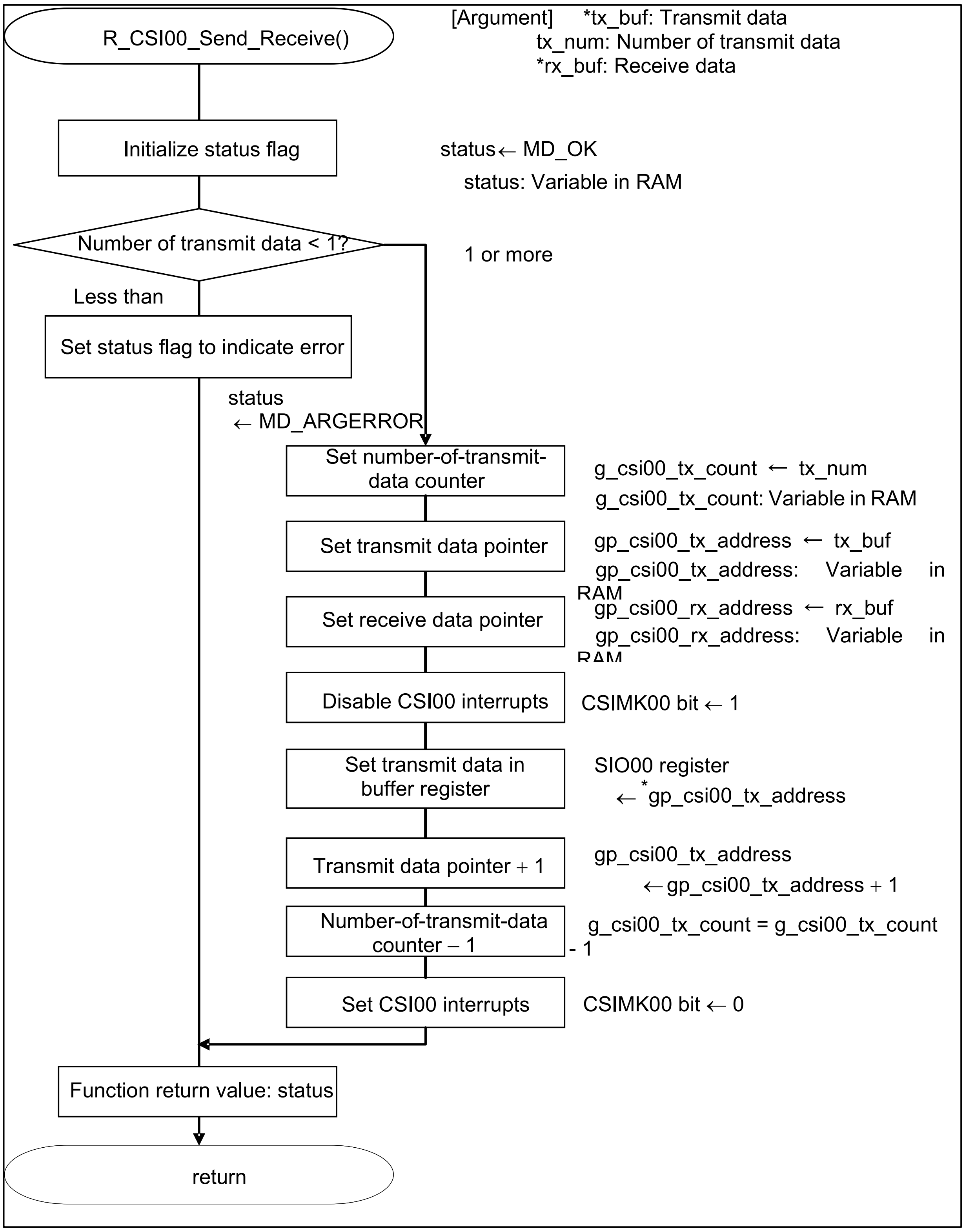
Figure 5-11 INTP0 start operating



5.6.11 CSI00 Transmit/Receive Processing

Figure 5-12 shows the flow of CSI00 Transmit/Receive Processing.

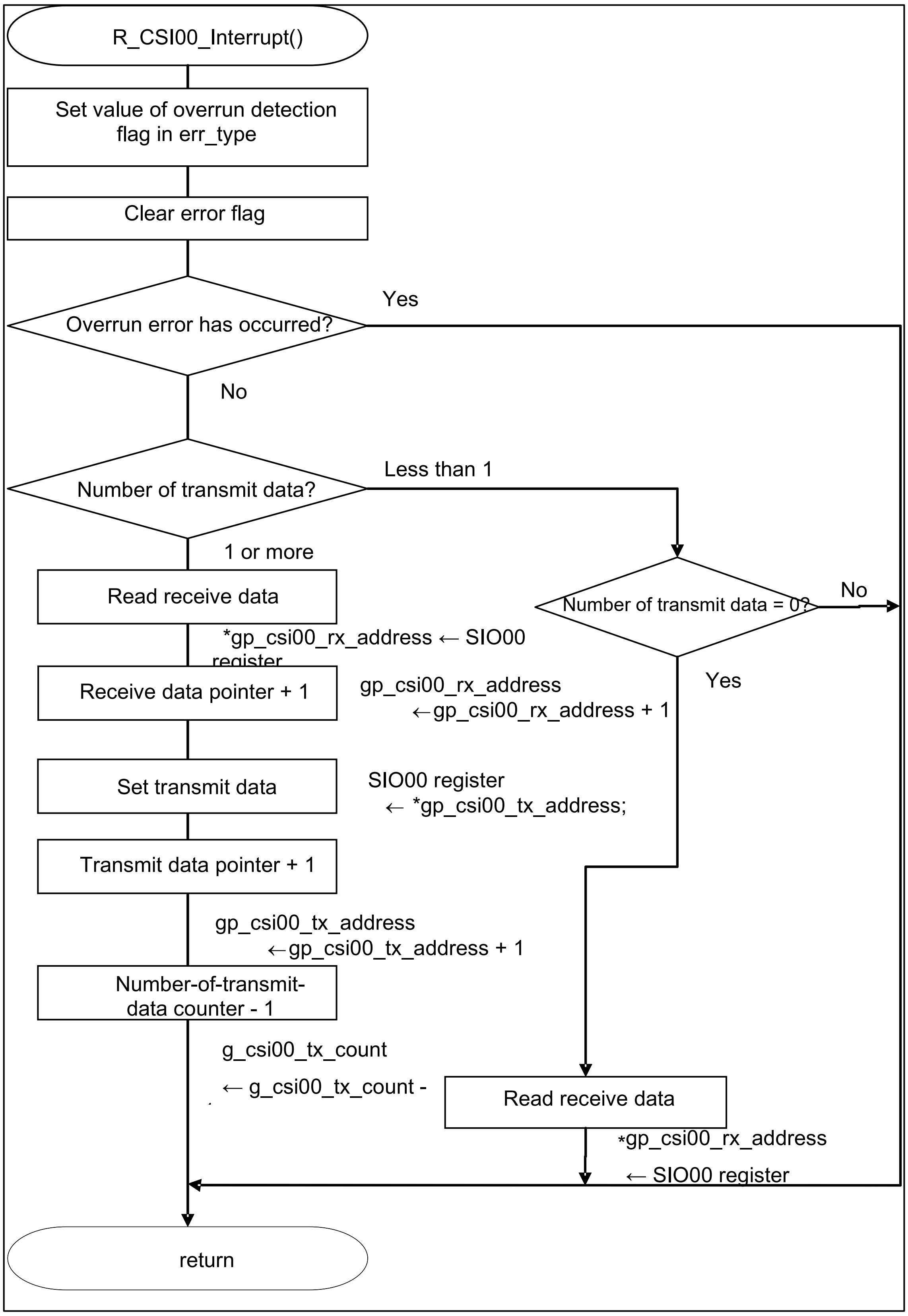
Figure 5-12 CSI00 Transmit/Receive Processing


R_CSI00_Send_Receive()
[Argument] *tx_buf: Transmit data
tx_num: Number of transmit data
*rx_buf: Receive data
Initialize status flag
status ← MD_OK
status: Variable in RAM
Number of transmit data < 1?
1 or more
Less than
Set status flag to indicate error
status ← MD_ARGERROR
Set number-of-transmit-data counter
g_csi00_tx_count ← tx_num
g_csi00_tx_count: Variable in RAM
Set transmit data pointer
gp_csi00_tx_address ← tx_buf
gp_csi00_tx_address: Variable in RAM
Set receive data pointer
gp_csi00_rx_address ← rx_buf
gp_csi00_rx_address: Variable in RAM
Disable CSI00 interrupts
CSIMK00 bit ← 1
Set transmit data in buffer register
SIO00 register ← *gp_csi00_tx_address
Transmit data pointer + 1
gp_csi00_tx_address ← gp_csi00_tx_address + 1
Number-of-transmit-data counter – 1
g_csi00_tx_count = g_csi00_tx_count - 1
Set CSI00 interrupts
CSIMK00 bit ← 0
Function return value: status
return

5.6.12 INICSI00 interrupt processing

Figure 5-13 shows the flow of INICSI00 interrupt processing.

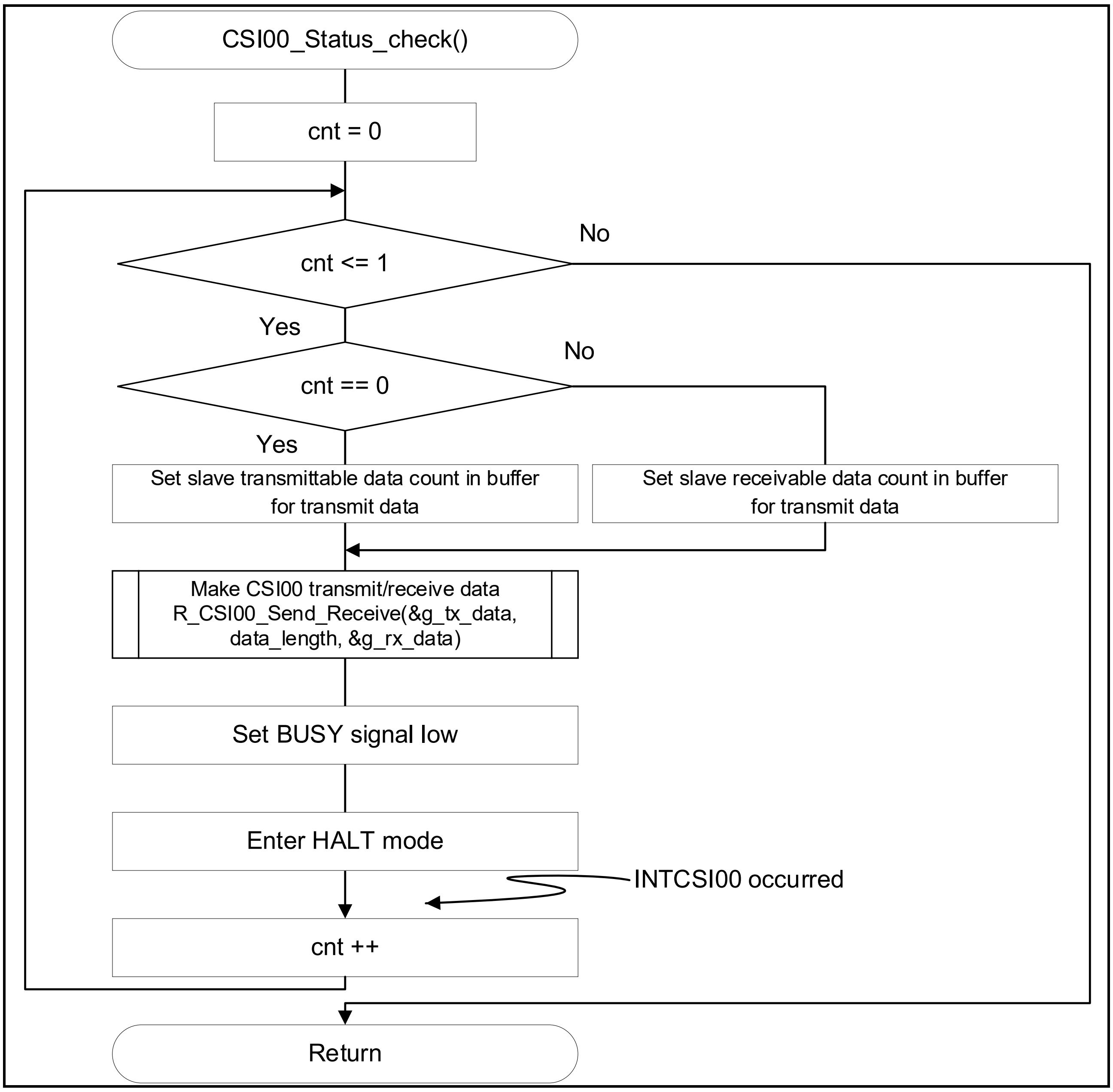
Figure 5-13 INICSI00 interrupt processing


R_CSI00_Interrupt()
Set value of overrun detection flag in err_type
Clear error flag
Overrun error has occurred?
Yes
No
Number of transmit data?
Less than 1
1 or more
Read receive data
*gp_csi00_rx_address ← SIO00 register
Receive data pointer + 1
gp_csi00_rx_address ←gp_csi00_rx_address + 1
Set transmit data
SIO00 register ← *gp_csi00_tx_address;
Transmit data pointer + 1
gp_csi00_tx_address ←gp_csi00_tx_address + 1
Number-of-transmit-data counter - 1
g_csi00_tx_count ← g_csi00_tx_count -
Number of transmit data = 0?
No
Yes
Read receive data
*gp_csi00_rx_address ← SIO00 register
return

5.6.13 Flowchart of CSI Status Check

Figure 5-14 shows the flow of CSI status checking.

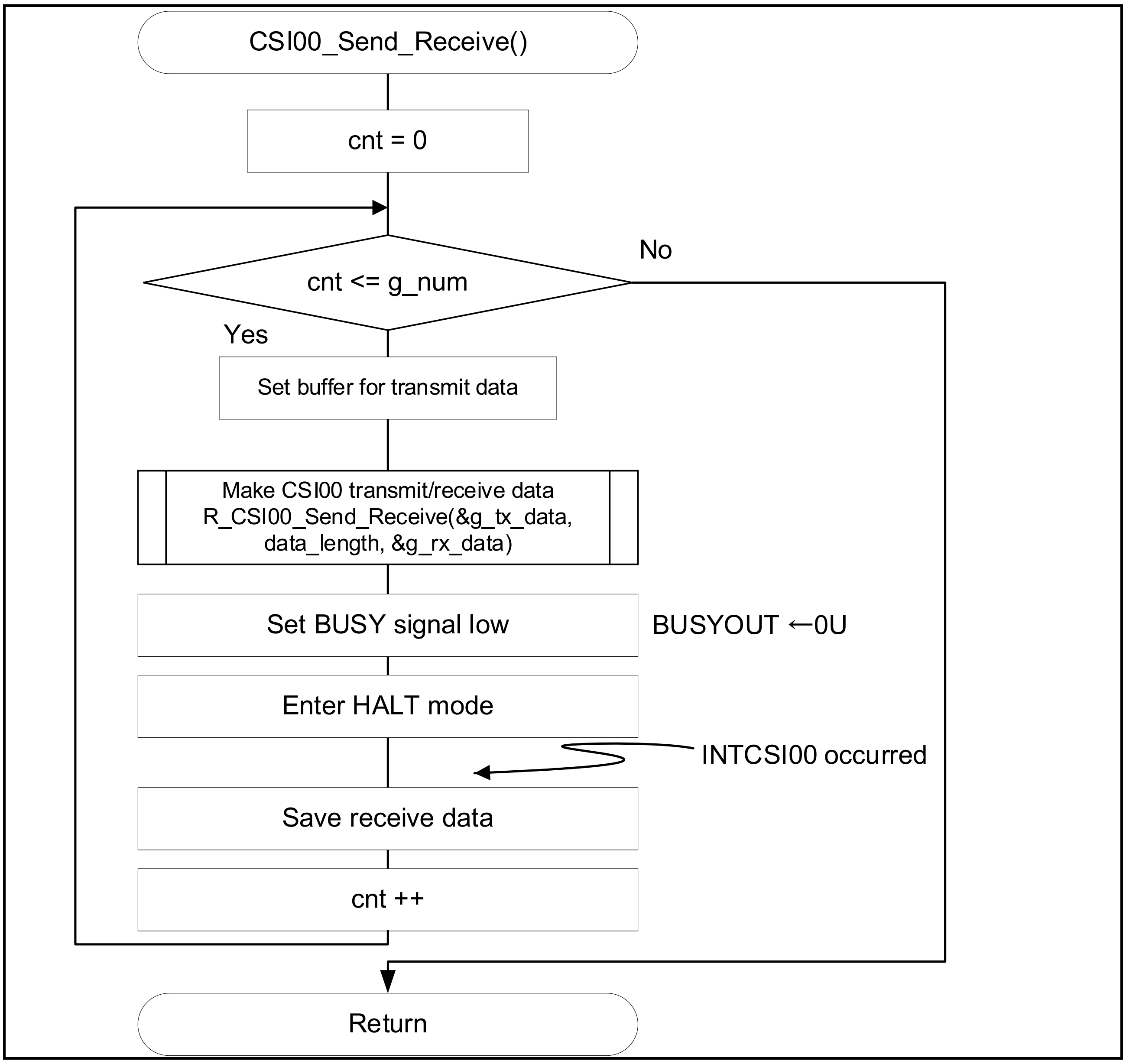
Figure 5-14 CSI Status Check


CSI00_Status_check()
cnt = 0
cnt <= 1
No
Yes
cnt == 0
No
Yes
Set slave transmittable data count in buffer for transmit data
Set slave receivable data count in buffer for transmit data
Make CSI00 transmit/receive data R_CSI00_Send_Receive(&g_tx_data, data_length, &g_rx_data)
Set BUSY signal low
Enter HALT mode
INTCSI00 occurred
cnt ++
Return

5.6.14 Flowchart of CSI Transmission/Reception

Figure 5-15 shows the flow of CSI transmission and reception.

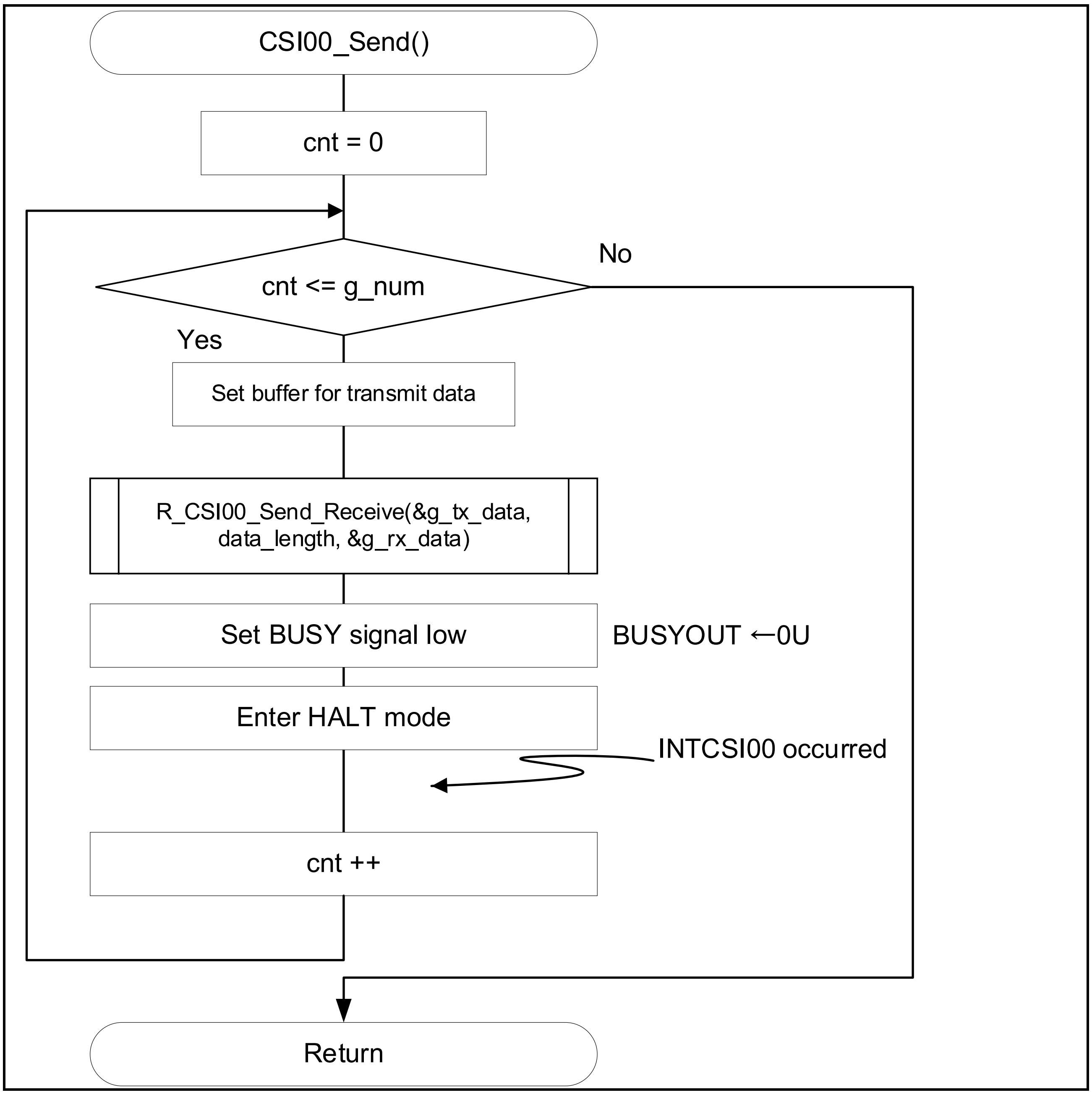
Figure 5-15 CSI Transmission/Reception



5.6.15 Flowchart of CSI Transmission

Figure 5-16 shows the flow of CSI transmission.

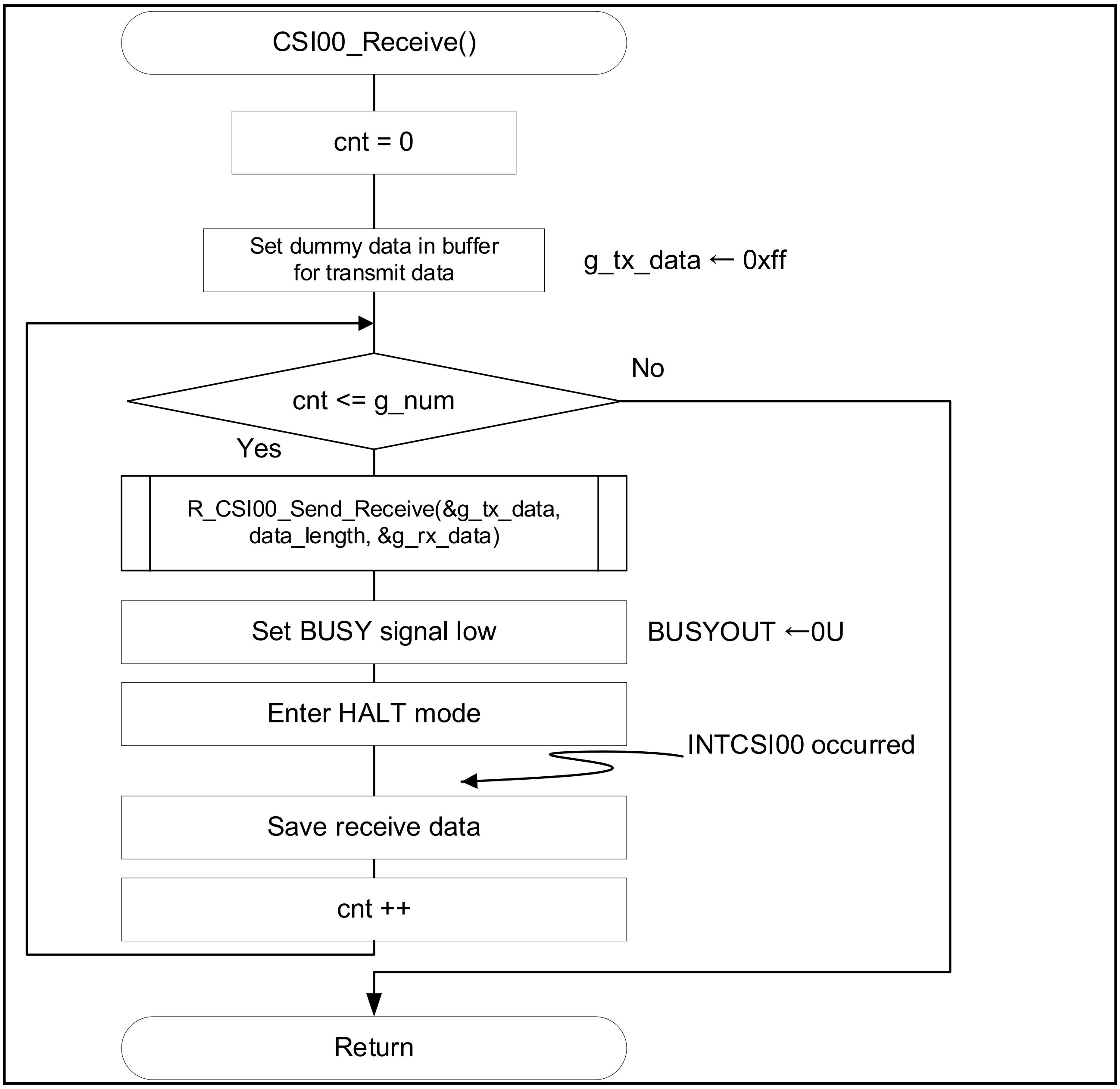
Figure 5-16 CSI Transmission



5.6.16 Flowchart of CSI Reception

Figure 5-17 shows the flow of CSI reception.

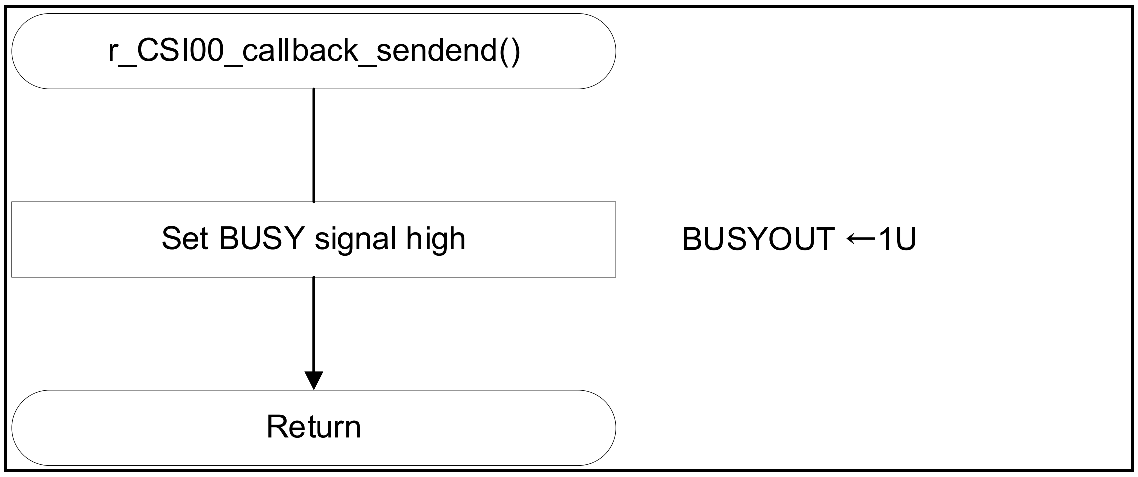
Figure 5-17 CSI Reception



5.6.17 Flowchart of Setting BUSY Signal High

Figure 5-18 shows the flow of setting the BUSY signal high.

Figure 5-18 Setting BUSY Signal High



6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

RL78/G13 User’s Manual: Hardware (R01UH0146E)
RL78 family user's manual software (R01US0015)
The latest versions can be downloaded from the Renesas Electronics website.

Technical update
The latest versions can be downloaded from the Renesas Electronics website.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	2023.6.15	-	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.