APPLICATION NOTE

R8C/35C Group

I^2C bus Interface Using UART2 Special Mode 1
(Slave Transmit/Receive)

REJ05B1351-0102
Rev.1.02
June 1, 2012

1. Abstract

This document describes the slave transmit/receive processes in I^2C bus interface slave communication using the R8C/35C Group serial interface (UART2) special mode 1 (I^2C mode).

For details on UART2 special mode 1, refer to the **M16C Family and R8C Family I^2C bus Interface Using UARTi Special Mode 1** application note.

2. Introduction

The application example described in this document applies to the following microcomputer (MCU) and parameter:

- MCU: R8C/35C Group
- XIN Clock: 20 MHz

The simplified I^2C bus communication is enabled by controlling additional functions for I^2C bus communication added to the UARTi clock synchronous circuit for I^2C bus interface using UARTi special mode 1. The I^2C bus interface using UARTi special mode 1 has more limitations for software processing time and timing than the I^2C bus interface hardware module. Careful verification and evaluation of your system are recommended, including the interaction between the I^2C bus communication program and programs other than the I^2C bus communication program.

3. Application Example

3.1 Program Outline

I^{2}C bus interface slave communication (slave transmission/reception) using the UART2 special mode 1 is processed in the application example. A maximum of 255 bytes of data can be transmitted/received.

This transmission procedure conforms to the I^{2}C bus communication protocol when used under the following conditions:

- Slave address: 7 bits
- Standard-mode and Fast-mode are supported
- Communication data length: 1 to 255 bytes (not including the slave address)
- Restart condition is not supported
- Single master/single slave communication

Figure 3.1 shows the Communication Format, Figure 3.2 shows the Block Diagram, Figure 3.3 shows the Outline Flowchart, and Figure 3.4 to Figure 3.6 show Timing Diagrams.

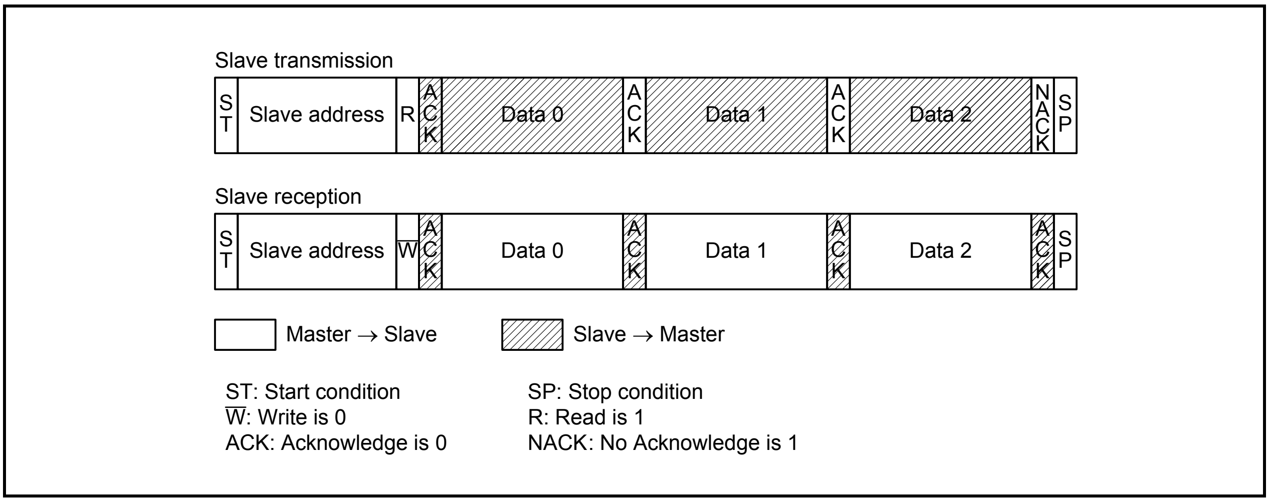


Figure 3.1 Communication Format

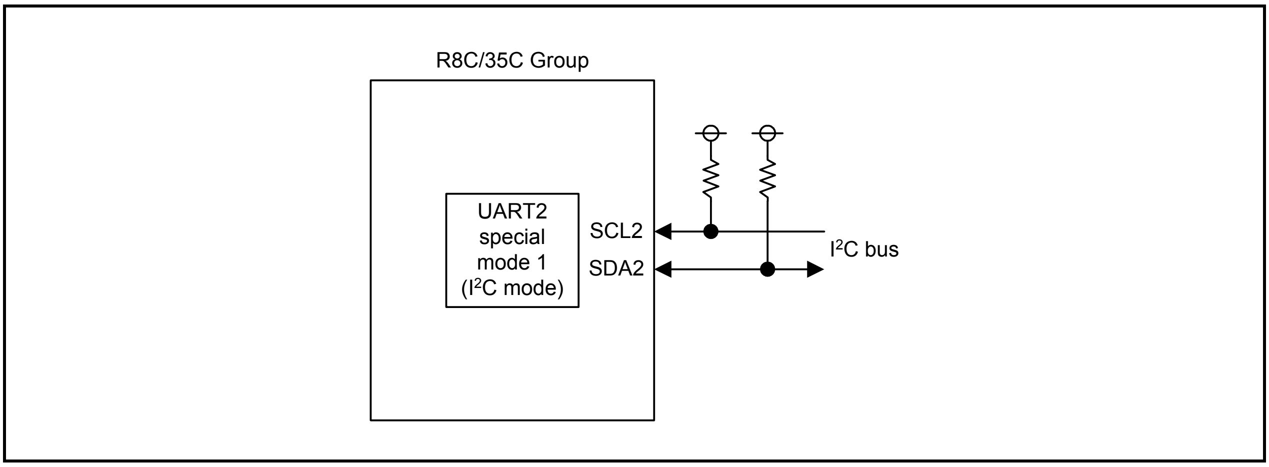


Figure 3.2 Block Diagram

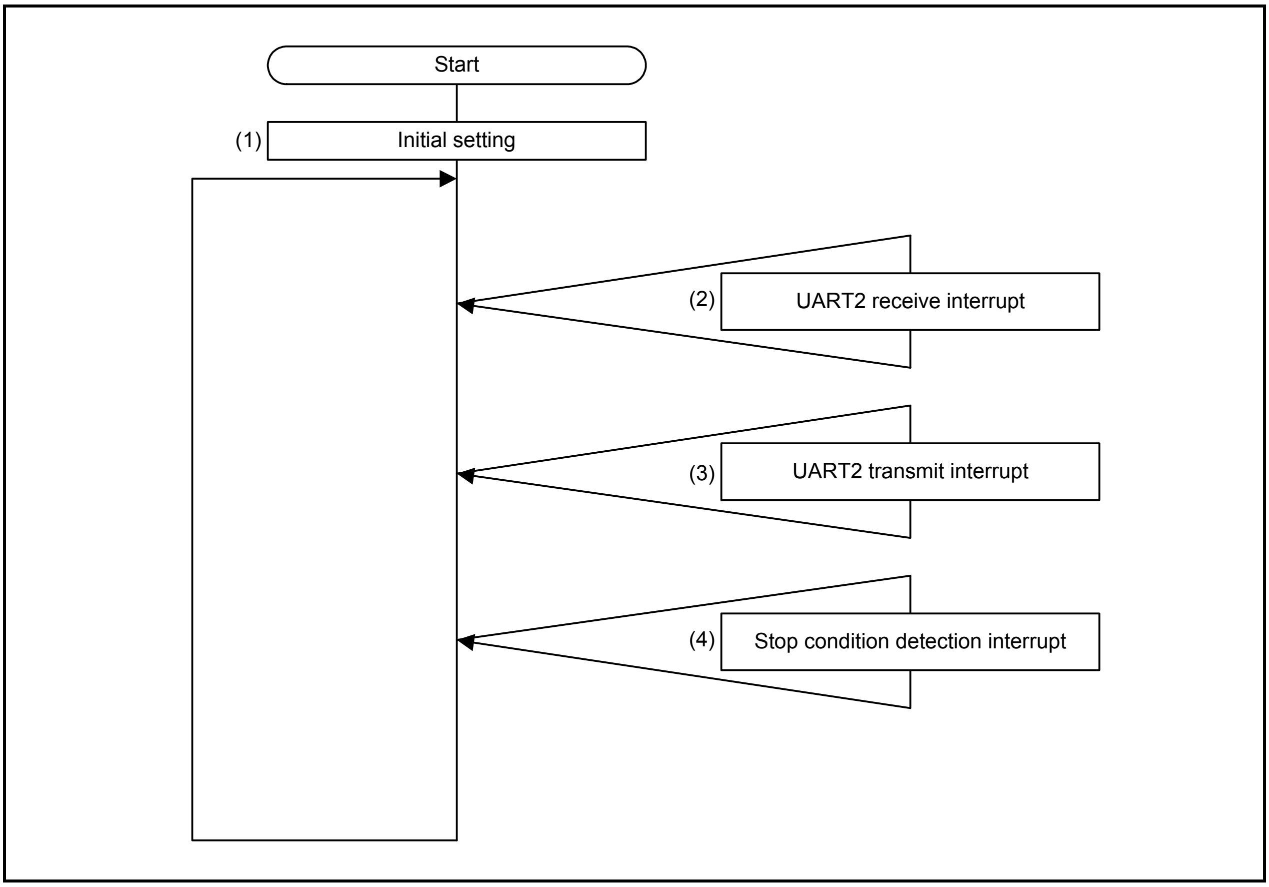


Figure 3.3 Outline Flowchart

The numbers in Figure 3.3 correspond to the numbers indicated in the program processing in the operating timing charts in Figure 3.4 to Figure 3.6.

(1) Initial setting
Initialize the system clock, UART2 associated SFRs, and variables used.

(2) UART2 receive interrupt
When a slave address is received, a UART2 receive interrupt is generated at the falling edge of the eighth bit of the SCL clock. The slave address is determined after reading the U2RB register.

When the slave address is matched:
- Generate an ACK and set the SCL2 pin to low hold at the ninth bit.
- Enable the stop condition detection interrupt and UART2 transmit interrupt. Disable the UART2 receive interrupt.
- Set transmit/receive data to the U2TB register.

When the slave address is not matched:
- Generate a NACK.

After the above processing, release SCL2 pin low hold at the eighth bit.

(3) UART2 transmit interrupt
A UART2 transmit interrupt is generated at the falling edge of the ninth bit of the SCL clock. When the first byte (slave address) is received, ACK output which set in the UART2 receive interrupt handling is released. When transmitting, determine the ACK/NACK and set the next byte transmit data. When receiving, store the receive data and set ACK for the next byte.

(4) Stop condition detection interrupt
When a stop condition is detected, an interrupt is generated. SFR values which changed in mid-communication are returned to the initial settings. Disable the stop condition detection interrupt and UART2 transmit interrupt. Enable the UART2 receive interrupt.

Note:
1. Write data to the U2TB register at slave transmit/receive according to the following procedure.

- When receiving the first byte data (slave address):

(1) Write the second byte data to the U2TB register in the receive interrupt handling.
(2) Write the third byte data to the U2TB register in the transmit interrupt handling.

- When receiving the second byte data onwards

Each time a transmit interrupt handling occurs, write 1-byte data sequentially to the U2TB register starting with the fourth byte

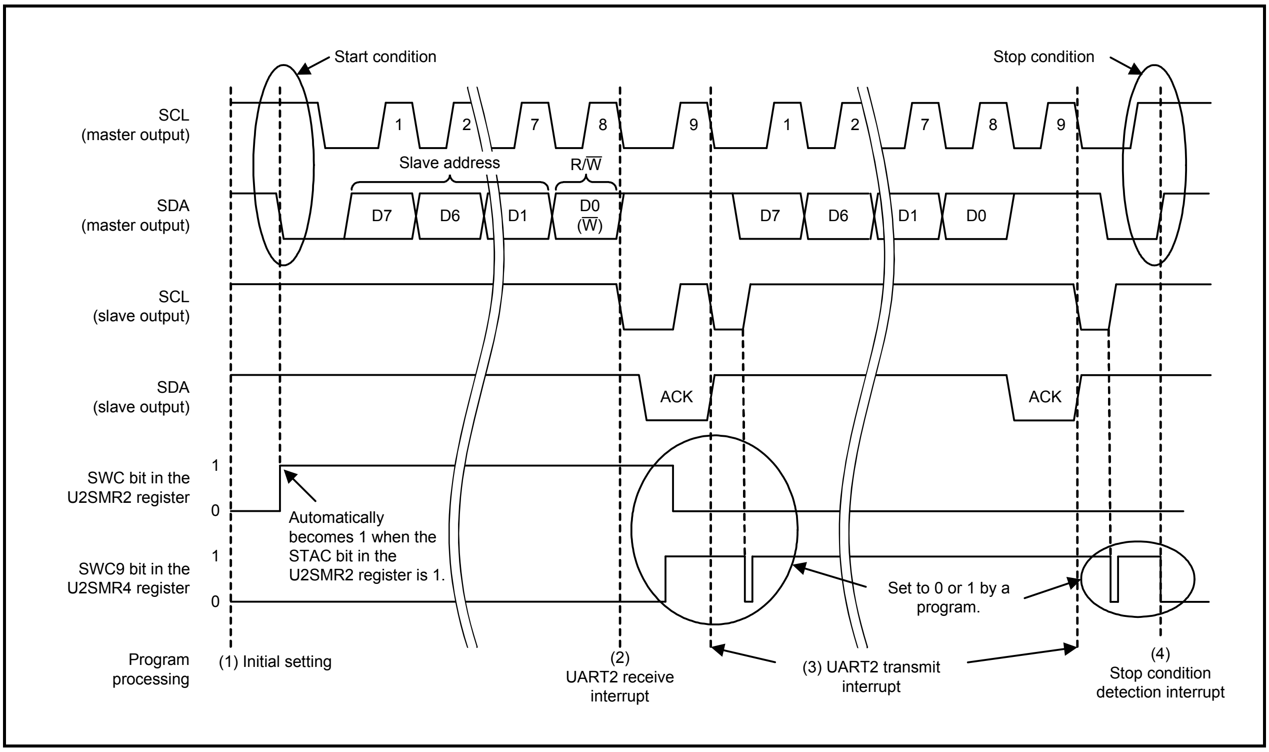


Figure 3.4 Slave Receive Timing

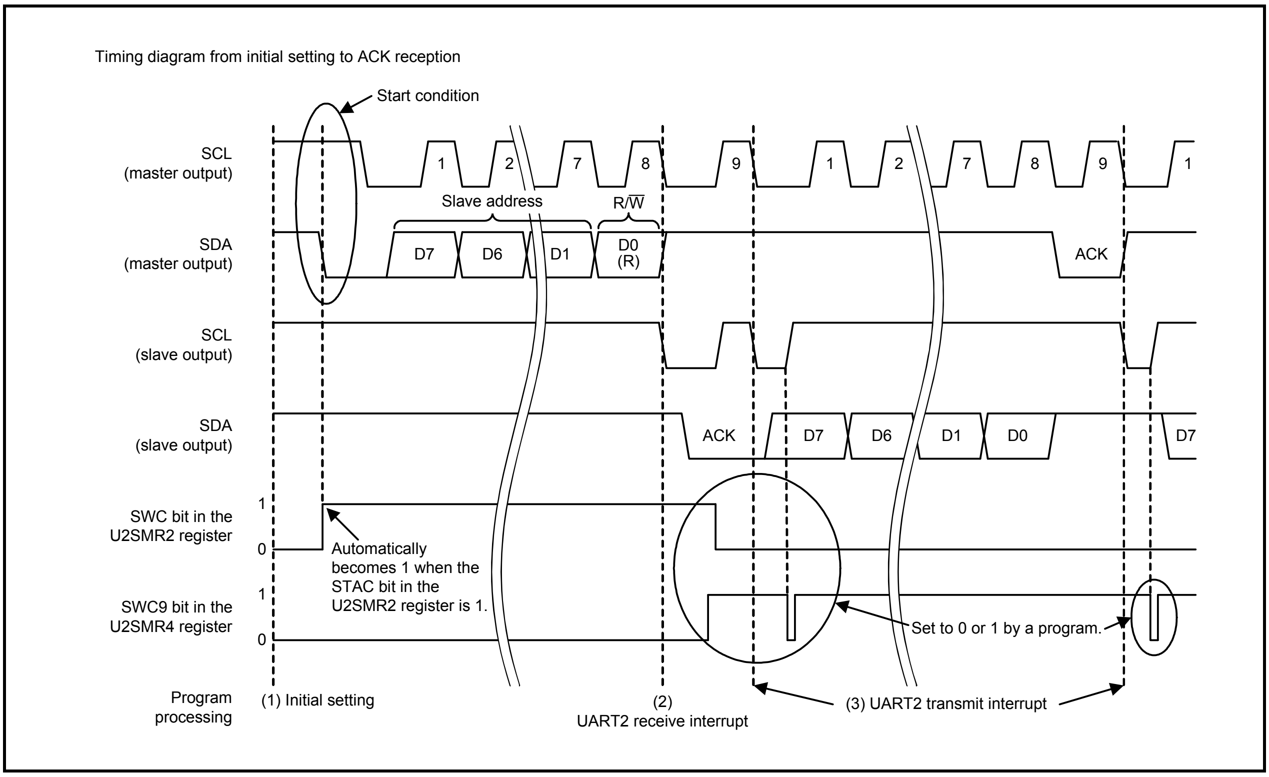

Timing diagram from initial setting to ACK reception
Start condition
SCL (master output)
1 2 7 8 9 1 2 7 8 9 1
Slave address
R/W
SDA (master output)
D7 D6 D1 D0 (R)
ACK
SCL (slave output)
SDA (slave output)
ACK D7 D6 D1 D0 D7
SWC bit in the U2SMR2 register
1 0
Automatically becomes 1 when the STAC bit in the U2SMR2 register is 1.
SWC9 bit in the U2SMR4 register
1 0
Set to 0 or 1 by a program.
Program processing
(1) Initial setting
(2) UART2 receive interrupt
(3) UART2 transmit interrupt

Figure 3.5 Slave Transmit Timing (1)

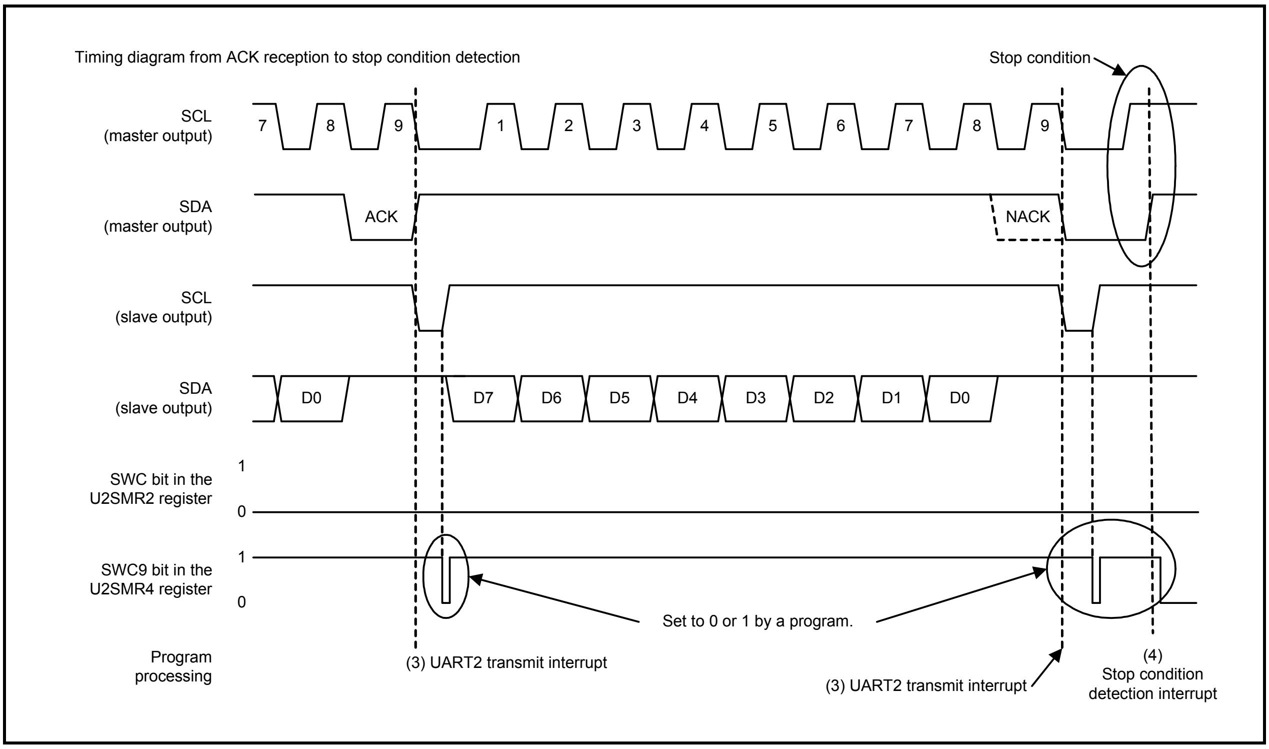

Timing diagram from ACK reception to stop condition detection
Stop condition
SCL (master output)
7 8 9 1 2 3 4 5 6 7 8 9
SDA (master output)
ACK
NACK
SCL (slave output)
SDA (slave output)
D0 D7 D6 D5 D4 D3 D2 D1 D0
SWC bit in the U2SMR2 register
1 0
SWC9 bit in the U2SMR4 register
1 0
Set to 0 or 1 by a program.
Program processing
(3) UART2 transmit interrupt
(3) UART2 transmit interrupt
(4) Stop condition detection interrupt

Figure 3.6 Slave Transmit Timing (2)

3.1.1 Peripheral Functions

Serial interface (UART2) special mode 1 (I^2C mode) is used under the following setting conditions:

- I^2C mode is used.
- Transfer clock is external clock source.
- f1 used as U2BRG count source.
- SDA2 and SCL2 pins are N-channel open-drain output.
- Transfer format is MSB first.
- Transmission completed (TXEPT is 1) is selected as the UART2 transmit interrupt source.
- With clock delay.
- Seven to eight cycles of U2BRG count source is selected as SDA2 digital delay value.
- UART2 initial setting is used.
- Enable SCL2 wait output.
- Disable SCL2 wait output 2.
- Enable SCL2 wait output 3.
- SDA2 output disable function is used.
- Start condition detection interrupt is not used.
- Stop condition detection interrupt is used.
- UART2 transmit interrupt is used.
- UART2 receive interrupt is used.

Table 3.1 Pins Used and Their Function

Pin	I/O	Function
P3_4/SCL2	I/O	I^2C mode clock I/O pin
P3_7/SDA2	I/O	I^2C mode data I/O pin

3.1.2 Notes on Using the Attached Sample Program

Note the following when using the program included with this application note:

(1) Do not use multiple interrupts.

(2) The size of the receive buffer and the transmit buffer are set to 255 bytes. The buffer size is defined by the BUFSIZE macro (1 to 255 bytes). When the number of transmit/receive bytes exceeds the size of the buffer, the slave disregards the communication. Disable the UART2 transmit interrupt, and release pins SCL2 and SDA2.

(3) After the master generates a stop condition, when the slave processing time [1] has passed, start the next transmit/receive (start condition is generated).

Note:

1. The slave processing time indicates the time between detecting a stop condition and enabling I^2C mode in the main processing, and is dependent on the processing of the user program. The maximum processing time for this sample program is approximately 500 µs.

3.2 Memory

Table 3.2 Memory

Memory	Size	Remarks
ROM	694 bytes	In the iic.c module
RAM	4 bytes	In the iic.c module
Maximum user stack	21 bytes	
Maximum interrupt stack	27 bytes	

Usage memory size varies depending on C compiler version and compile options. The above applies under the following conditions:

C compiler: M16C Series, R8C Family Compiler V.5.45 Release 01

C compile options: -c -finfo -dir “$(CONFIGDIR)” -R8C

4. Software

This section shows the program example to set the example described in section 3. Application Example. Refer to the latest **R8C/35C Group hardware user's manual** for details on individual registers.

4.1 Usage Variables

Definition file name: rej05b1351_src.c

Variable Name	Size	Description
unsigned char iic_tx[BUFSIZE]	255 bytes	Transmit buffer
unsigned char iic_rx[BUFSIZE]	255 bytes	Receive buffer
unsigned char rcv_data[BUFSIZE]	255 bytes	Store to receive data read from receive buffer

Definition file name: iic.c

Variable Name		Size /Bit-number	Description
static byte_dt iic_str		-	Structure to store status
Structure member	iic_status	1 byte	All statuses
	iic_rw	b0	R/$\overline{W}$ flag 0: Write ($\overline{W}$) slave receive 1: Read (R) slave transmit
	iic_buf_full	b1	Buffer full flag 0: Within buffer size 1: Buffer full
	iic_end	b2	Communication completed flag 0: Busy (mid-communication) 1: Ready (except for mid-communication)
	-	b7 to b3	Not used (undefined)
static unsigned char far* iic_pointer		2 bytes	Transmit/receive buffer pointer
static unsigned char iic_index		1 byte	Number of transmit/receive bytes

4.2 Function Tables

Declaration	void main (void)		
Outline	Main processing		
Argument	Argument name		Meaning
	None		-
Variable (global)	Variable name		Contents
	unsigned char iic_tx[BUFSIZE]		Transmit buffer
	unsigned char iic_rx[BUFSIZE]		Receive buffer
	unsigned char rcv_data[BUFSIZE]		Store received data
Returned value	Type	Value	Meaning
	None	-	-
Function	After setting the system clock, I2C mode is enabled. Communication status is determined by the returned value of the iic_slave_end function. Each status is processed after communication is completed, and the uart2_init function is called to enable I2C mode.		

Declaration	void mcu_init (void)		
Outline	System clock setting processing		
Argument	Argument name		Meaning
	None		-
Variable (global)	Variable name		Contents
	None		-
Returned value	Type	Value	Meaning
	None	-	-
Function	Called from main function. Perform system clock (XIN clock) setting.		

Declaration	void uart2_init (unsigned char ini)		
Outline	UART2 initial setting		
Argument	Argument name		Meaning
	unsigned char ini		0: I2C mode disabled 1: I2C mode enabled
Variable (global)	Variable name		Contents
	(structure member) iic_status		All statuses
Returned value	Type	Value	Meaning
	None	-	-
Function	Called from main function. Initialize SFR to use UART2 special mode 1 (I2C mode). When I2C mode is enabled, set iic_status to 0x00 (clear all statuses). When executing this function, interrupts are disabled by the I flag.		

Declaration	void _uart2_bcnic (void)		
Outline	Stop condition detection interrupt handling		
Argument	Argument name		Meaning
	None		-
Variable (global)	Variable name		Contents
	None		-
Returned value	Type	Value	Meaning
	None	-	-
Function	An interrupt occurs when a stop condition is detected, and the stp_int function is called.		

Declaration	static void stp_init (void)		
Outline	Stop condition detection processing		
Argument	Argument name		Meaning
	None		-
Variable (global)	Variable name		Contents
	(structure member) iic_end		Communication completed flag
Returned value	Type	Value	Meaning
	None	-	-
Function	Called from stop condition detection interrupt handling. UART2 related SFR values changed mid-communication are returned to their initial values, and the communication completed flag is set to 1.		

Declaration	void _uart2_receive (void)		
Outline	UART2 receive interrupt handling		
Argument	Argument name		Meaning
	None		-
Variable (global)	Variable name		Contents
	unsigned char far* iic_pointer		Transmit/receive buffer pointer
	unsigned char iic_index		Number of transmit/receive bytes
	(structure member) iic_status		All statuses
	(structure member) iic_rw		R/$\overline{W}$ flag
Returned value	Type	Value	Meaning
	None	-	-
Function	An interrupt occurs at the falling edge of the eighth bit of the SCL clock. This function calls the iic_id_check function after reading the U2RB register in the function header. • When the slave address is matched, generate an ACK, and set the SCL2 pin to low hold at the ninth bit. The receive interrupt is disabled, and the transmit interrupt and stop condition detection interrupt are enabled. The number of transmit/receive bytes and all statuses are cleared. When the slave is receiving, set the ACK for the next byte. When the slave is transmitting, set transmit data for the next byte. • When the slave address is not matched, generate a NACK. After the above processing, release SCL2 pin low hold.		

Declaration	unsigned char* iic_id_check (unsigned char id, unsigned char rw)		
Outline	Slave address determine processing		
Argument	Argument name		Meaning
	unsigned char id		Received slave address
	unsigned char rw		R/$\overline{W}$ flag
Variable (global)	Variable name		Contents
	None		-
Returned value	Type	Value	Meaning
	unsigned char*	iic_rx	Receive buffer address
		iic_tx	Transmit buffer address
		NULL	Slave address does not match
Function	Called from UART2 receive interrupt handling. Received slave address is determined. When the slave address is matched, the returned value is the buffer address. When the slave address is not matched, the returned value is NULL.		

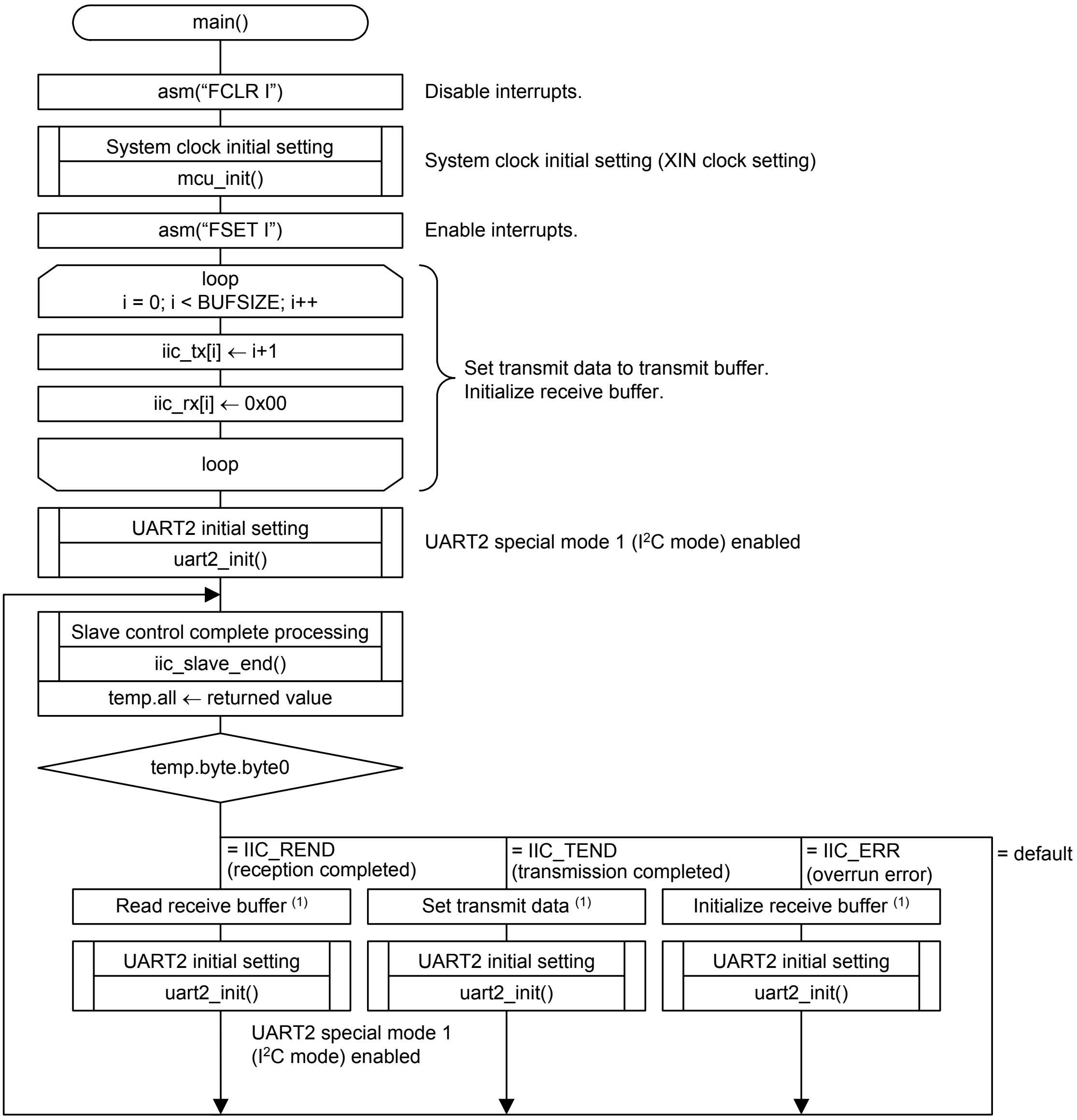
Declaration	void _uart2_trans (void)		
Outline	UART2 transmit interrupt handling		
Argument	Argument name		Meaning
	None		-
Variable (global)	Variable name		Contents
	unsigned char iic_index		Number of transmit/receive bytes
	(structure member) iic_rw		R/$\overline{W}$ flag
Returned value	Type	Value	Meaning
	None	-	-
Function	An interrupt occurs at the falling edge of the ninth bit of the SCL clock. The U2RB register is read in the function header. When the first byte (slave address) is received, disable ACK output set by the receive interrupt handler. After the first byte is received, the slave_rcv_int function is called when the slave is receiving and the slave_trn_int function is called when the slave is transmitting.		

Declaration	static void slave_rcv_int (unsigned char rb_data)		
Outline	Slave receive processing		
Argument	Argument name		Meaning
	unsigned char rb_data		Receive data read from the U2RB register
Variable (global)	Variable name		Contents
	unsigned char iic_index		Number of transmit/receive bytes
	unsigned char far* iic_pointer		Transmit/receive buffer pointer
	(structure member) iic_buf_full		Buffer full flag
Returned value	Type	Value	Meaning
	None	-	-
Function	Called from UART2 transmit interrupt handling. The argument value is stored in the receive buffer (except the slave address). • When the number of received bytes is less than the buffer size, set an ACK for the next byte. Release SCL2 pin low hold, then set the SCL2 pin to low hold for the next byte. • When the number of received bytes is the same as or greater than the buffer size, the buffer full flag is set to 1. Release pins SCL2 and SDA2, and disable the UART2 transmit interrupt.		

Declaration	static void slave_trn_int (unsigned char rb_data)		
Outline	Slave transmit processing		
Argument	Argument name		Meaning
	unsigned char rb_data		ACK/NACK read from the U2RB register
Variable (global)	Variable name		Contents
	unsigned char iic_index		Number of transmit/receive bytes
	unsigned char far* iic_pointer		Transmit/receive buffer pointer
	(structure member) iic_buf_full		Buffer full flag
Returned value	Type	Value	Meaning
	None	-	-
Function	Called from UART2 transmit interrupt handling. • When an ACK is detected and the number of transmit bytes is less than the buffer size, set transmit data for the next byte. Release SCL2 pin low hold, then set the SCL2 pin to low hold for the next byte. • When the number of transmit bytes is the same as or greater than the buffer size, set the buffer full flag to 1. Release pins SCL2 and SDA2, and disable the UART2 transmit interrupt. • When NACK is detected, release pins SCL2 and SDA2, and disable the UART2 transmit interrupt.		

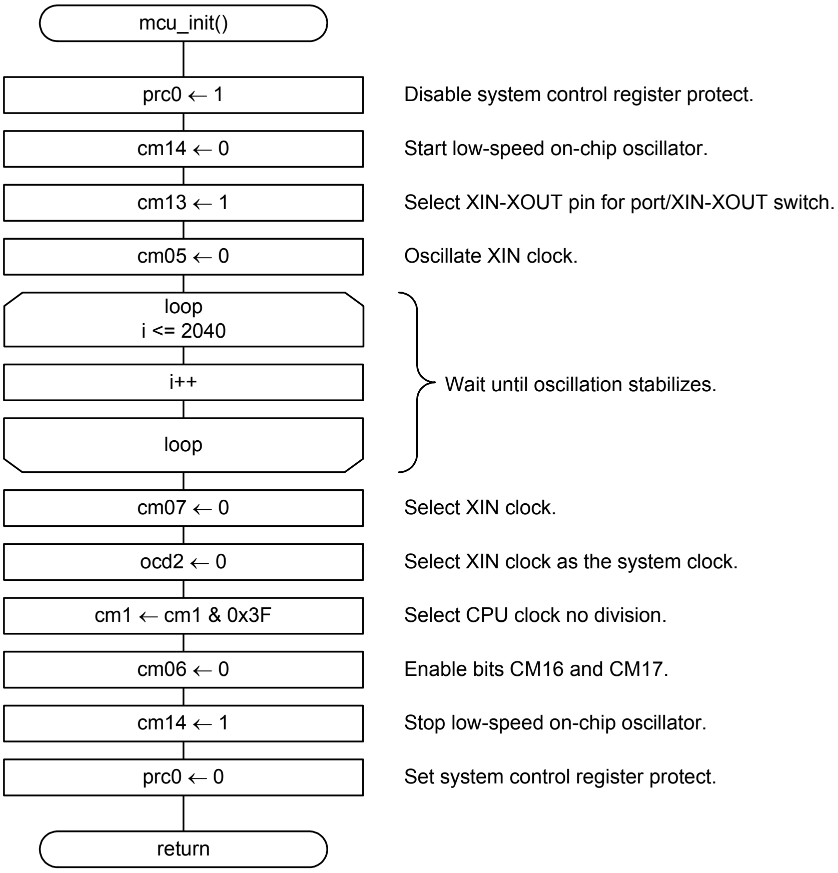
Declaration	unsigned short iic_slave_end (void)			
Outline	Slave control completed processing			
Argument	Argument name			Meaning
	None			-
Variable (global)	Variable name			Contents
	(structure member) iic_end			Communication completed flag
	(structure member) iic_rw			R/$\overline{W}$ flag
	unsigned char iic_index			Number of transmit/receive bytes
Returned value	Type		Value	Meaning
	unsigned short	Lower byte	IIC_BUSY	Mid-communication
			IIC_REND	Reception completed
			IIC_TEND	Transmission completed
			IIC_ERR	Overrun error detected
		Upper byte	1 to 255	Number of transmit/receive bytes
Function	Called from the main processing. It informs the user of the state of slave control completion. When the communication completed flag is 1 and there is transmit/receive data except for the slave address, disable I^{2}C mode. Otherwise, return IIC_BUSY (mid-communication). After disabling I^{2}C mode, when the communication completed flag is 0, the next communication is determined to be started and the IIC_ERR (overrun error detection) function is returned. When the communication completed flag is 1, return IIC_REND (reception completed) or IIC_TEND (transmission completed).			

4.3 Main Processing



Note:
1. Additional processing can be added as needed.

4.4 System Clock Setting



Disable system control register protect.

Start low-speed on-chip oscillator.

Select XIN-XOUT pin for port/XIN-XOUT switch.

Oscillate XIN clock.

Wait until oscillation stabilizes.

Select XIN clock.

Select XIN clock as the system clock.

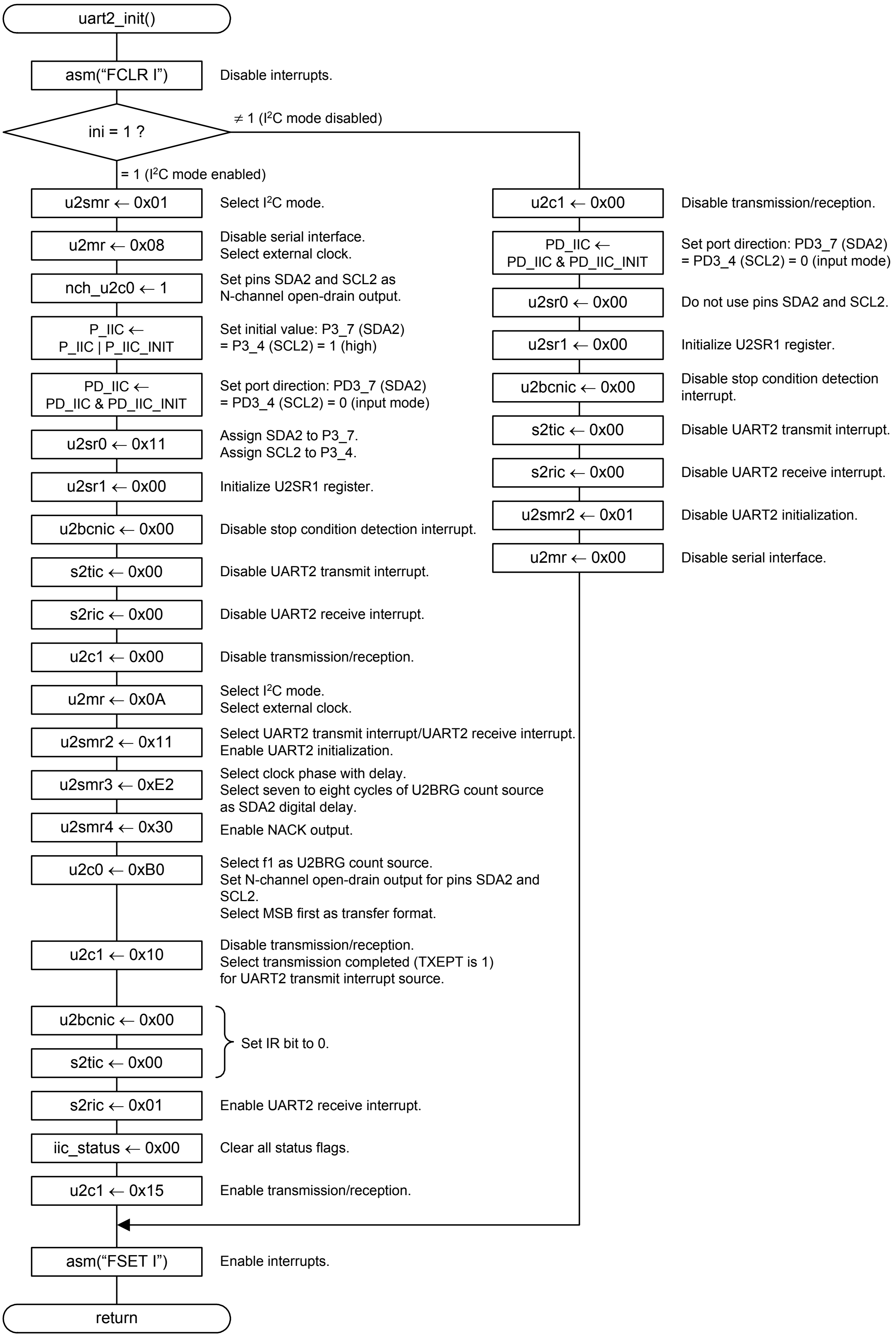
Select CPU clock no division.

Enable bits CM16 and CM17.

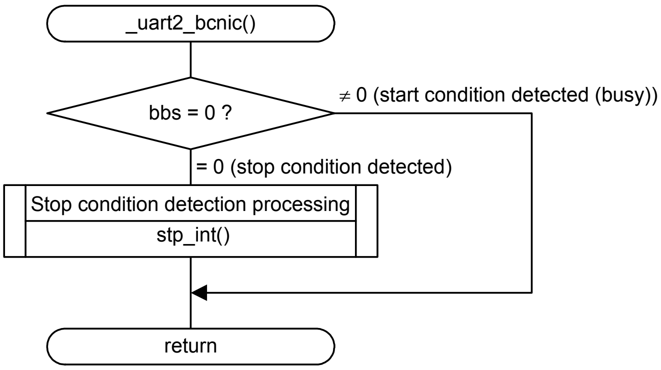
Stop low-speed on-chip oscillator.

Set system control register protect.

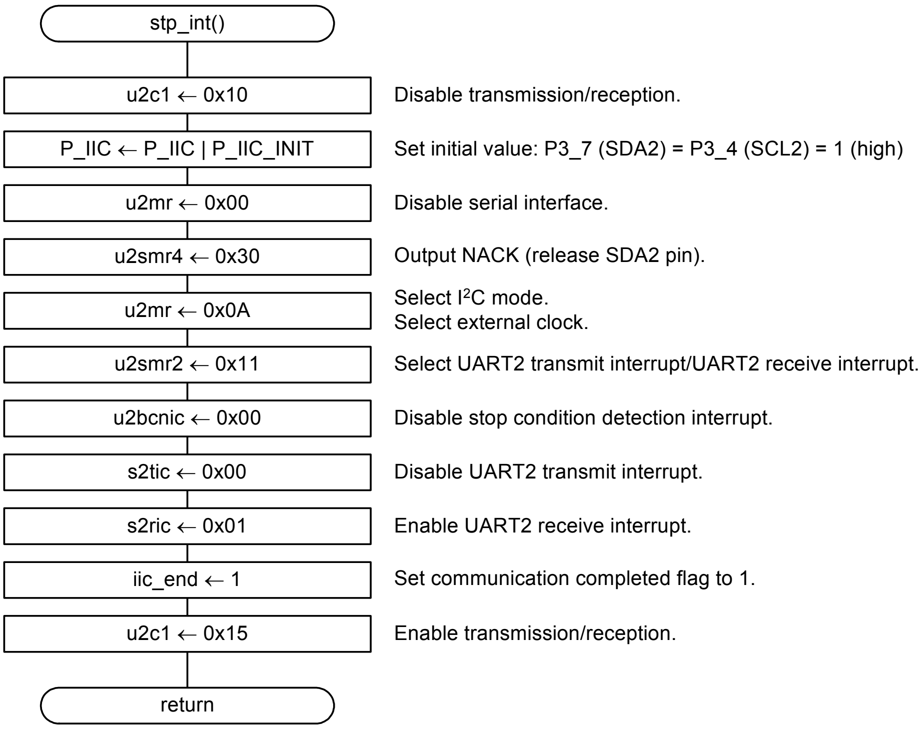
4.5 UART2 Initial Setting


uart2_init()
asm(“FCLR I”)
Disable interrupts.
ini = 1 ?
≠ 1 (I²C mode disabled)
= 1 (I²C mode enabled)
u2smr ← 0x01
Select I²C mode.
u2mr ← 0x08
Disable serial interface.
Select external clock.
nch_u2c0 ← 1
Set pins SDA2 and SCL2 as
N-channel open-drain output.
P_IIC ←
P_IIC | P_IIC_INIT
Set initial value: P3_7 (SDA2)
= P3_4 (SCL2) = 1 (high)
PD_IIC ←
PD_IIC & PD_IIC_INIT
Set port direction: PD3_7 (SDA2)
= PD3_4 (SCL2) = 0 (input mode)
u2sr0 ← 0x11
Assign SDA2 to P3_7.
Assign SCL2 to P3_4.
u2sr1 ← 0x00
Initialize U2SR1 register.
u2bcnic ← 0x00
Disable stop condition detection interrupt.
s2tic ← 0x00
Disable UART2 transmit interrupt.
s2ric ← 0x00
Disable UART2 receive interrupt.
u2c1 ← 0x00
Disable transmission/reception.
u2mr ← 0x0A
Select I²C mode.
Select external clock.
u2smr2 ← 0x11
Select UART2 transmit interrupt/UART2 receive interrupt.
Enable UART2 initialization.
u2smr3 ← 0xE2
Select clock phase with delay.
Select seven to eight cycles of U2BRG count source
as SDA2 digital delay.
u2smr4 ← 0x30
Enable NACK output.
u2c0 ← 0xB0
Select f1 as U2BRG count source.
Set N-channel open-drain output for pins SDA2 and
SCL2.
Select MSB first as transfer format.
u2c1 ← 0x10
Disable transmission/reception.
Select transmission completed (TXEPT is 1)
for UART2 transmit interrupt source.
u2bcnic ← 0x00
s2tic ← 0x00
Set IR bit to 0.
s2ric ← 0x01
Enable UART2 receive interrupt.
iic_status ← 0x00
Clear all status flags.
u2c1 ← 0x15
Enable transmission/reception.
u2c1 ← 0x00
Disable transmission/reception.
PD_IIC ←
PD_IIC & PD_IIC_INIT
Set port direction: PD3_7 (SDA2)
= PD3_4 (SCL2) = 0 (input mode)
u2sr0 ← 0x00
Do not use pins SDA2 and SCL2.
u2sr1 ← 0x00
Initialize U2SR1 register.
u2bcnic ← 0x00
Disable stop condition detection
interrupt.
s2tic ← 0x00
Disable UART2 transmit interrupt.
s2ric ← 0x00
Disable UART2 receive interrupt.
u2smr2 ← 0x01
Disable UART2 initialization.
u2mr ← 0x00
Disable serial interface.
asm(“FSET I”)
Enable interrupts.
return

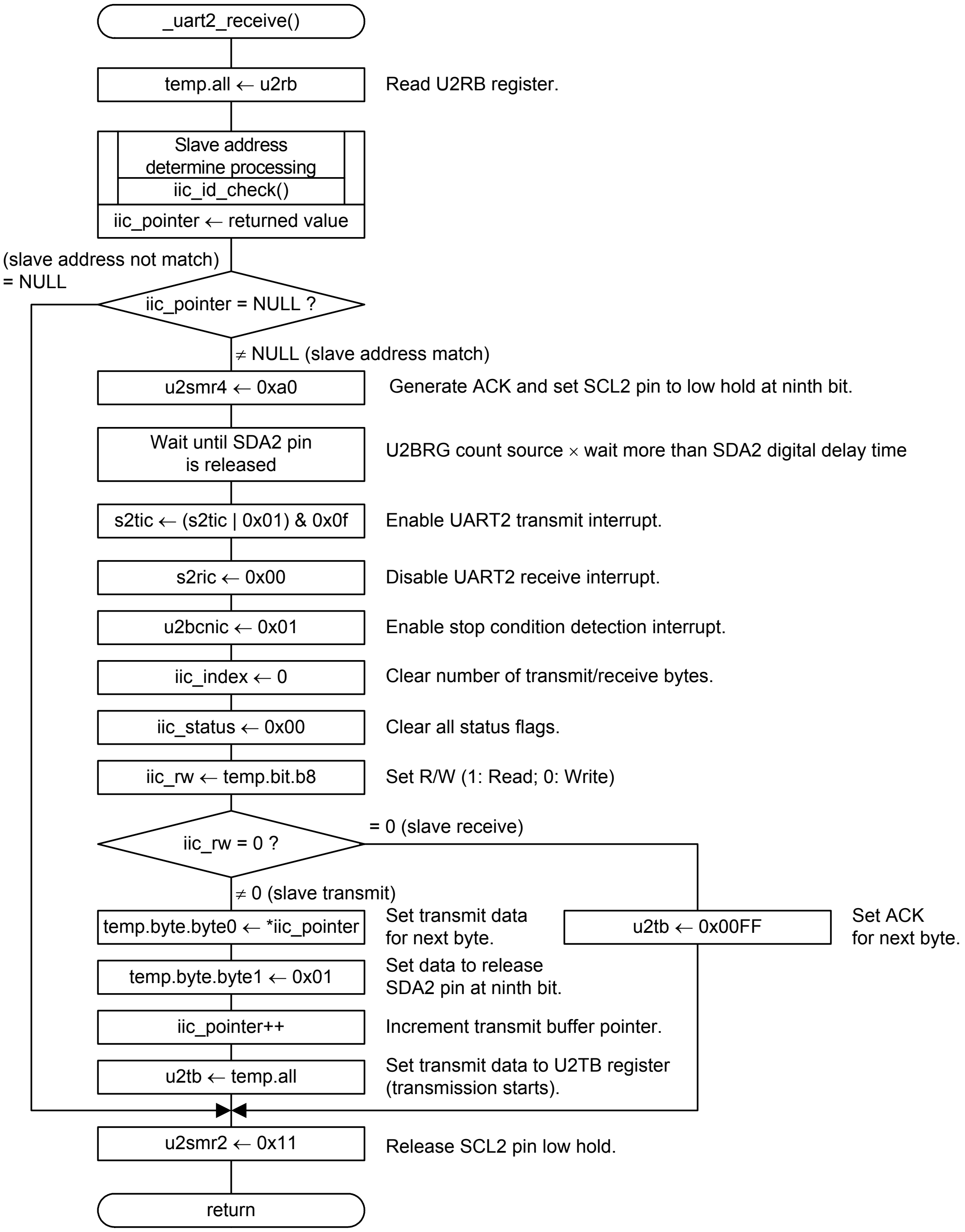
4.6 Stop Condition Detection Interrupt Handling


_uart2_bcnic()
bbs = 0 ?
≠ 0 (start condition detected (busy))
= 0 (stop condition detected)
Stop condition detection processing
stp_int()
return

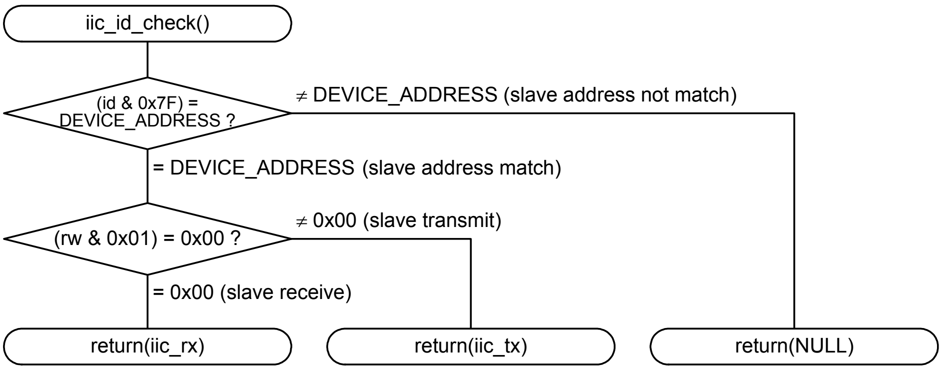
4.7 Stop Condition Detection Processing


stp_int()
u2c1 ← 0x10
Disable transmission/reception.
P_IIC ← P_IIC | P_IIC_INIT
Set initial value: P3_7 (SDA2) = P3_4 (SCL2) = 1 (high)
u2mr ← 0x00
Disable serial interface.
u2smr4 ← 0x30
Output NACK (release SDA2 pin).
u2mr ← 0x0A
Select I²C mode.
Select external clock.
u2smr2 ← 0x11
Select UART2 transmit interrupt/UART2 receive interrupt.
u2bcnic ← 0x00
Disable stop condition detection interrupt.
s2tic ← 0x00
Disable UART2 transmit interrupt.
s2ric ← 0x01
Enable UART2 receive interrupt.
iic_end ← 1
Set communication completed flag to 1.
u2c1 ← 0x15
Enable transmission/reception.
return

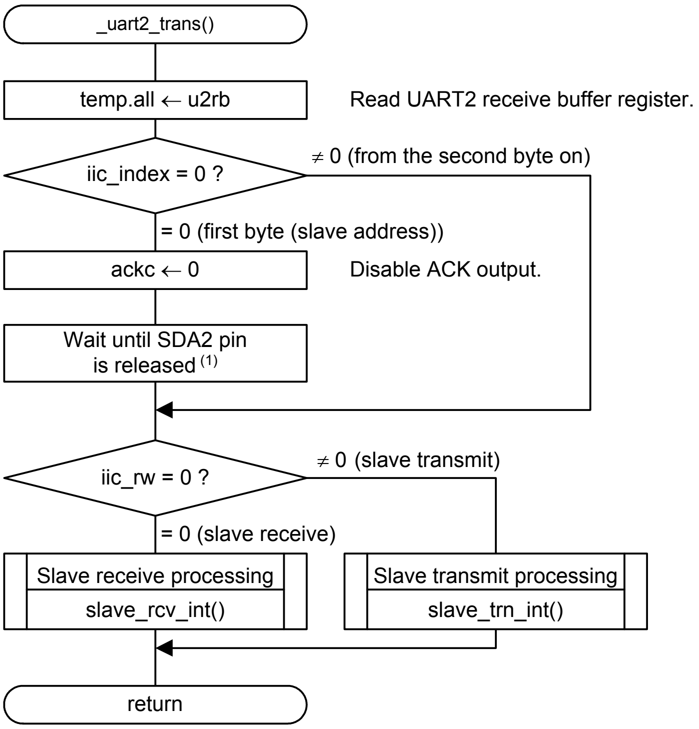
4.8 UART2 Receive Interrupt Handling


_uart2_receive()
temp.all ← u2rb
Read U2RB register.
Slave address
determine processing
iic_id_check()
iic_pointer ← returned value
(slave address not match)
= NULL
iic_pointer = NULL ?
≠ NULL (slave address match)
u2smr4 ← 0xa0
Generate ACK and set SCL2 pin to low hold at ninth bit.
Wait until SDA2 pin
is released
U2BRG count source × wait more than SDA2 digital delay time
s2tic ← (s2tic | 0x01) & 0x0f
Enable UART2 transmit interrupt.
s2ric ← 0x00
Disable UART2 receive interrupt.
u2bcnic ← 0x01
Enable stop condition detection interrupt.
iic_index ← 0
Clear number of transmit/receive bytes.
iic_status ← 0x00
Clear all status flags.
iic_rw ← temp.bit.b8
Set R/W (1: Read; 0: Write)
iic_rw = 0 ?
= 0 (slave receive)
≠ 0 (slave transmit)
temp.byte.byte0 ← *iic_pointer
Set transmit data
for next byte.
u2tb ← 0x00FF
Set ACK
for next byte.
temp.byte.byte1 ← 0x01
Set data to release
SDA2 pin at ninth bit.
iic_pointer++
Increment transmit buffer pointer.
u2tb ← temp.all
Set transmit data to U2TB register
(transmission starts).
u2smr2 ← 0x11
Release SCL2 pin low hold.
return

4.9 Slave Address Determine Processing

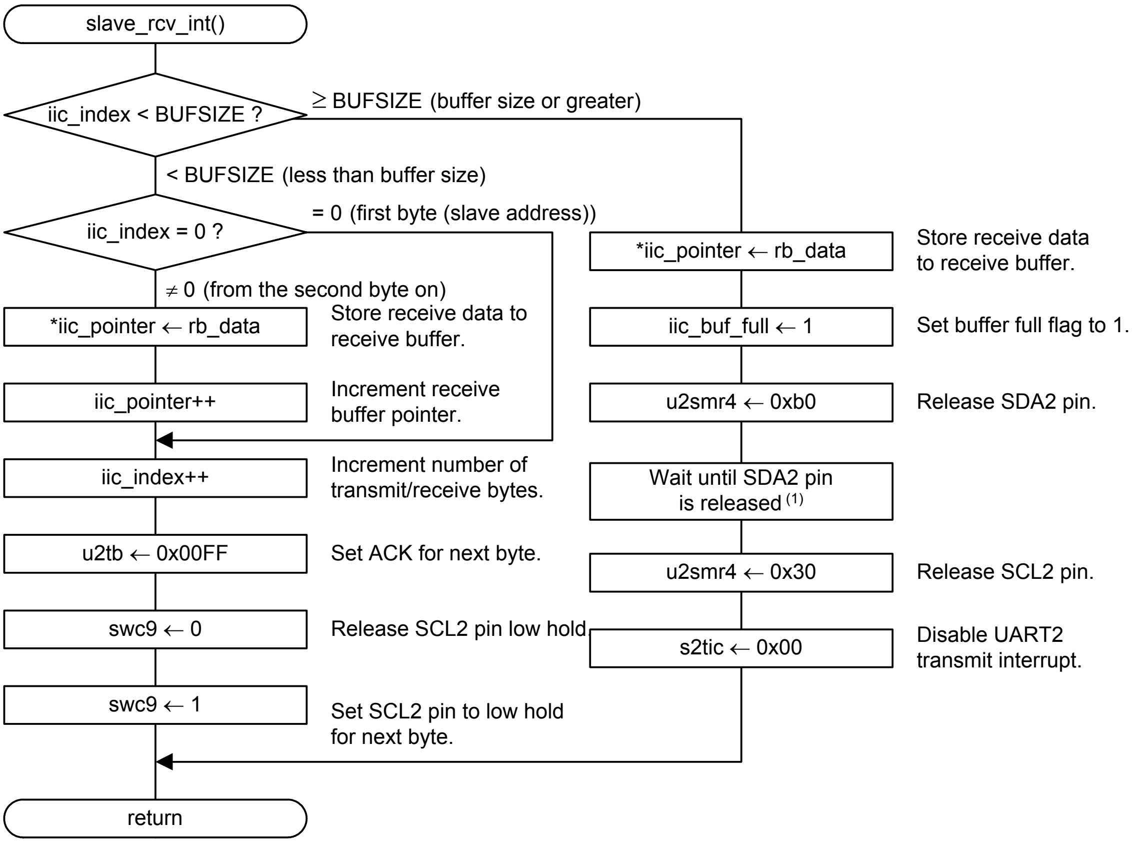


4.10 UART2 Transmit Interrupt Handling



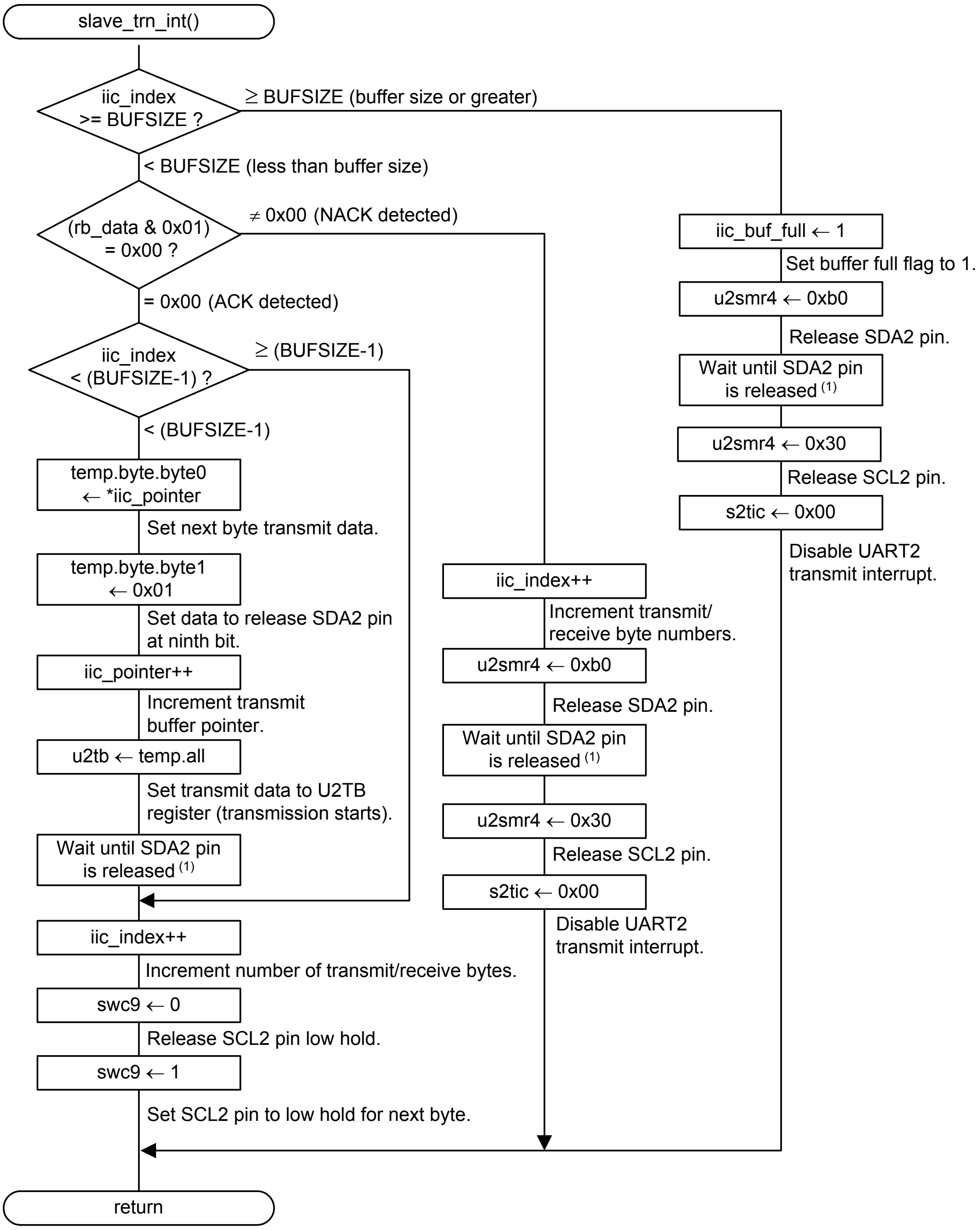
Note:
1. U2BRG count source × wait more than SDA2 digital delay time

4.11 Slave Receive Processing


slave_rcv_int()
iic_index < BUFSIZE ?
≥ BUFSIZE (buffer size or greater)
< BUFSIZE (less than buffer size)
iic_index = 0 ?
= 0 (first byte (slave address))
≠ 0 (from the second byte on)
*iic_pointer ← rb_data
Store receive data to receive buffer.
iic_pointer++
Increment receive buffer pointer.
iic_index++
Increment number of transmit/receive bytes.
u2tb ← 0x00FF
Set ACK for next byte.
swc9 ← 0
Release SCL2 pin low hold
swc9 ← 1
Set SCL2 pin to low hold for next byte.
return
*iic_pointer ← rb_data
Store receive data to receive buffer.
iic_buf_full ← 1
Set buffer full flag to 1.
u2smr4 ← 0xb0
Release SDA2 pin.
Wait until SDA2 pin is released (1)
u2smr4 ← 0x30
Release SCL2 pin.
s2tic ← 0x00
Disable UART2 transmit interrupt.

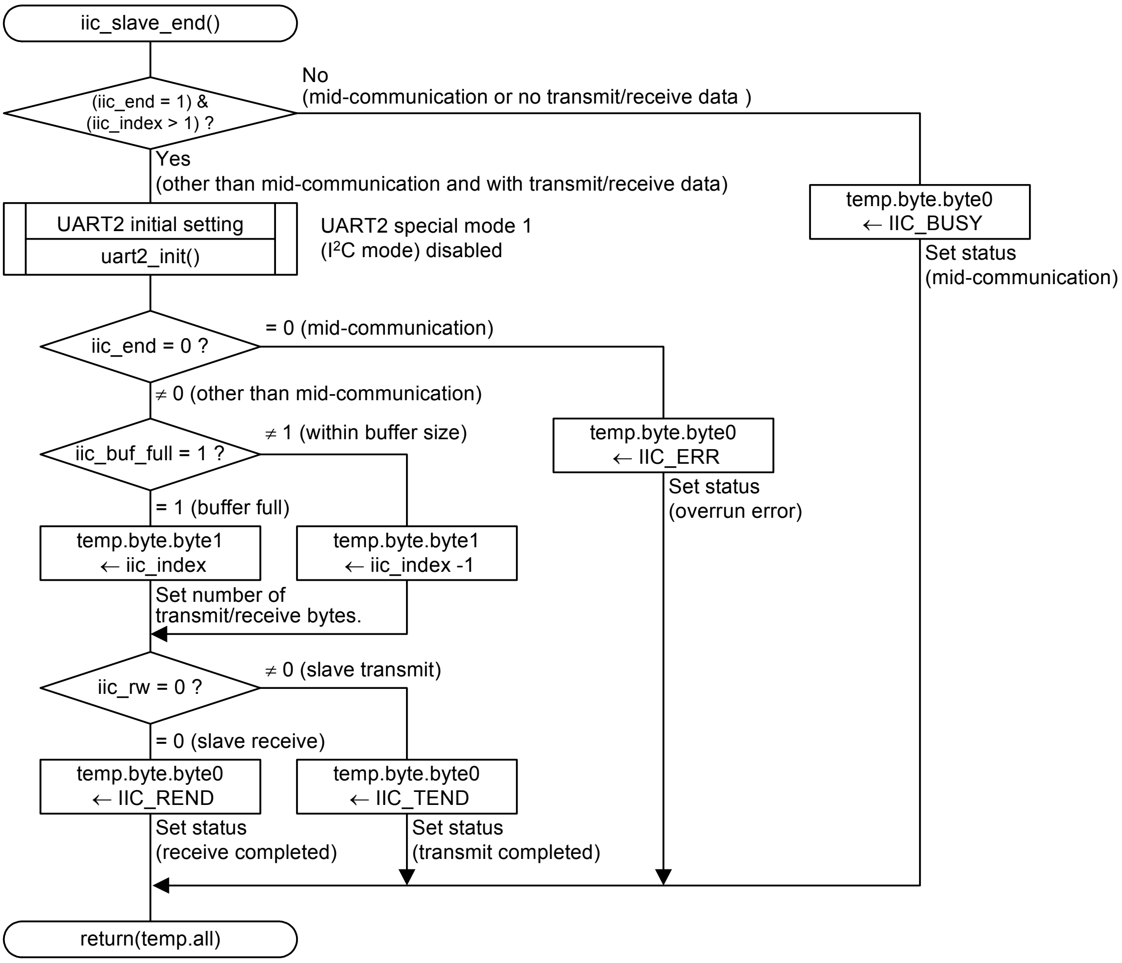
Note:
1. U2BRG count source × wait more than SDA2 digital delay time

4.12 Slave Transmit Processing


slave_trn_int()
iic_index >= BUFSIZE ?
≥ BUFSIZE (buffer size or greater)
< BUFSIZE (less than buffer size)
(rb_data & 0x01) = 0x00 ?
≠ 0x00 (NACK detected)
= 0x00 (ACK detected)
iic_index < (BUFSIZE-1) ?
≥ (BUFSIZE-1)
< (BUFSIZE-1)
temp.byte.byte0 ← *iic_pointer
Set next byte transmit data.
temp.byte.byte1 ← 0x01
Set data to release SDA2 pin at ninth bit.
iic_pointer++
Increment transmit buffer pointer.
u2tb ← temp.all
Set transmit data to U2TB register (transmission starts).
Wait until SDA2 pin is released (1)
iic_index++
Increment number of transmit/receive bytes.
swc9 ← 0
Release SCL2 pin low hold.
swc9 ← 1
Set SCL2 pin to low hold for next byte.
iic_index++
Increment transmit/ receive byte numbers.
u2smr4 ← 0xb0
Release SDA2 pin.
Wait until SDA2 pin is released (1)
u2smr4 ← 0x30
Release SCL2 pin.
s2tic ← 0x00
Disable UART2 transmit interrupt.
iic_buf_full ← 1
Set buffer full flag to 1.
u2smr4 ← 0xb0
Release SDA2 pin.
Wait until SDA2 pin is released (1)
u2smr4 ← 0x30
Release SCL2 pin.
s2tic ← 0x00
Disable UART2 transmit interrupt.
return

Note:
1. U2BRG count source × wait more than SDA2 digital delay time

4.13 Slave Control Completed Processing



5. Sample Program

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

6. Reference Documents

Application Note
M16C Family, R8C Family I^{2}C Bus Interface Using UARTi Special Mode 1 (REJ05B1349)
The latest version can be downloaded from the Renesas Electronics website.

R8C/35C Group User’s Manual: Hardware Rev.1.00
The latest version can be downloaded from the Renesas Electronics website.

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

C Compiler Manual
M16C Series, R8C Family C Compiler Package V.5.45
C Compiler User’s Manual Rev.1.00
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

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

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

Revision History	R8C/35C Group I^{2}C bus Interface Using UART2 Special Mode 1 (Slave Transmit/Receive)

Rev.	Date	Description	
		Page	Summary
1.00	Sep. 1, 2010	—	First edition issued
1.01	Mar. 10, 2011	8	Table 3.2 ROM size, 646 bytes revised as 694 bytes
		19	4.8 UART2 Receive Interrupt Handling, processing of waiting until SDA2 pin is released added
		20	4.10 UART2 Transmit Interrupt Handling, processing of waiting until SDA2 pin is released added
		21	4.11 Slave Receive Processing, processing of U2SMR4 ← 0xb0 and wait until SDA2 pin is released added
		22	4.12 Slave Transmit Processing, processing of U2SMR4 ← 0xb0 and wait until SDA2 pin is released added
1.02	June 1, 2012	2	A condition added to 3.1 Program Outline

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

Notice

1. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "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 etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

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

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

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

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

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

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

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

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

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

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

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

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

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

Colophon 2.0