RX210, RX21A, and RX220 Groups

Multi-Master I2C Bus Using RIIC

R01AN1348EJ0101
Rev. 1.01
July 1, 2014

Abstract

This document describes a method of performing multi-muster communication using the I^{2}C bus interface (RIIC) in the RX210, RX21A, and RX220 Groups.

Products

- RX210, RX21A, and RX220 Groups

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

Multi-master communication with I^2C bus is performed using the RIIC.

After a reset, a master transmission and master reception are performed once each. 10-byte data (00h to 09h) is transmitted in a master transmission, then 10-byte data is received in a master reception.

When an arbitration-lost is detected during a master transmission or master reception, a slave operation is performed while other master device communications are prioritized.

- Transfer rate: 100 kbps
- Address format: 7-bit address format
- Master/slave operations: Master transmission, master reception, slave transmission, and slave reception

Refer to the User's Manual: Hardware for the product used and the I^2C bus specifications for details on I^2C bus communication formats.

Table 1.1 lists the Peripheral Function and Its Application and Figure 1.1 shows the Operation Overview.

Table 1.1 Peripheral Function and Its Application

Peripheral Function	Application
RIIC	Multi-master I^2C bus

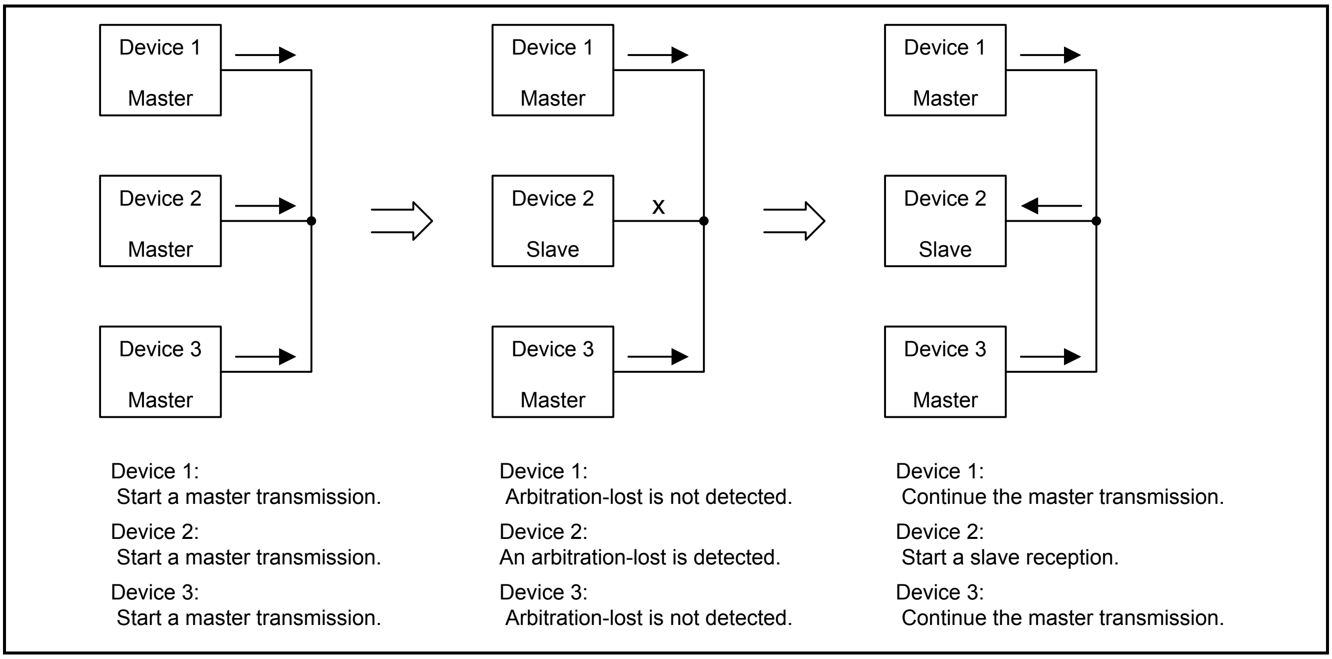

Device 1
Master
Device 2
Master
Device 3
Master
Device 1
Master
Device 2
Slave
x
Device 3
Master
Device 1
Master
Device 2
Slave
Device 3
Master
Device 1:
Start a master transmission.
Device 2:
Start a master transmission.
Device 3:
Start a master transmission.
Device 1:
Arbitration-lost is not detected.
Device 2:
An arbitration-lost is detected.
Device 3:
Arbitration-lost is not detected.
Device 1:
Continue the master transmission.
Device 2:
Start a slave reception.
Device 3:
Continue the master transmission.

Figure 1.1 Operation Overview

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	R5F52108ADFP (RX210 Group)
Operating frequencies	- Main clock: 20 MHz - PLL: 100 MHz (main clock divided by 2 and multiplied by 10) - System clock (ICLK): 50 MHz (PLL divided by 2) - Peripheral module clock B (PCLKB): 25 MHz (PLL divided by 4)
Operating voltage	5.0 V
Integrated development environment	Renesas Electronics Corporation High-performance Embedded Workshop Version 4.09.01
C compiler	Renesas Electronics Corporation C/C++ Compiler Package for RX Family V.1.02 Release 01
	Compile options -cpu=rx200 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug -nologo (The default setting is used in the integrated development environment.)
iodefine.h version	Version 1.2A
Endian	Little endian
Operating mode	Single-chip mode
Processor mode	Supervisor mode
Sample code version	Version 1.00
Board used	Renesas Starter Kit for RX210 (product part no.: R0K505210C000BE)

3. Reference Application Notes

For additional information associated with this document, refer to the following application notes.

- RX210 Group Initial Setting Rev. 2.00 (R01AN1002EJ)
- RX21A Group Initial Setting Rev. 1.10 (R01AN1486EJ)
- RX220 Group Initial Setting Rev. 1.10 (R01AN1494EJ)

The initial setting functions in the reference application notes are used in the sample code in this application note. The revision numbers of the reference application notes are current as of when this application note was made. However the latest version is always recommended. Visit the Renesas Electronics Corporation website to check and download the latest version.

4. Hardware

4.1 Hardware Configuration

Figure 4.1 shows a Connection Example.

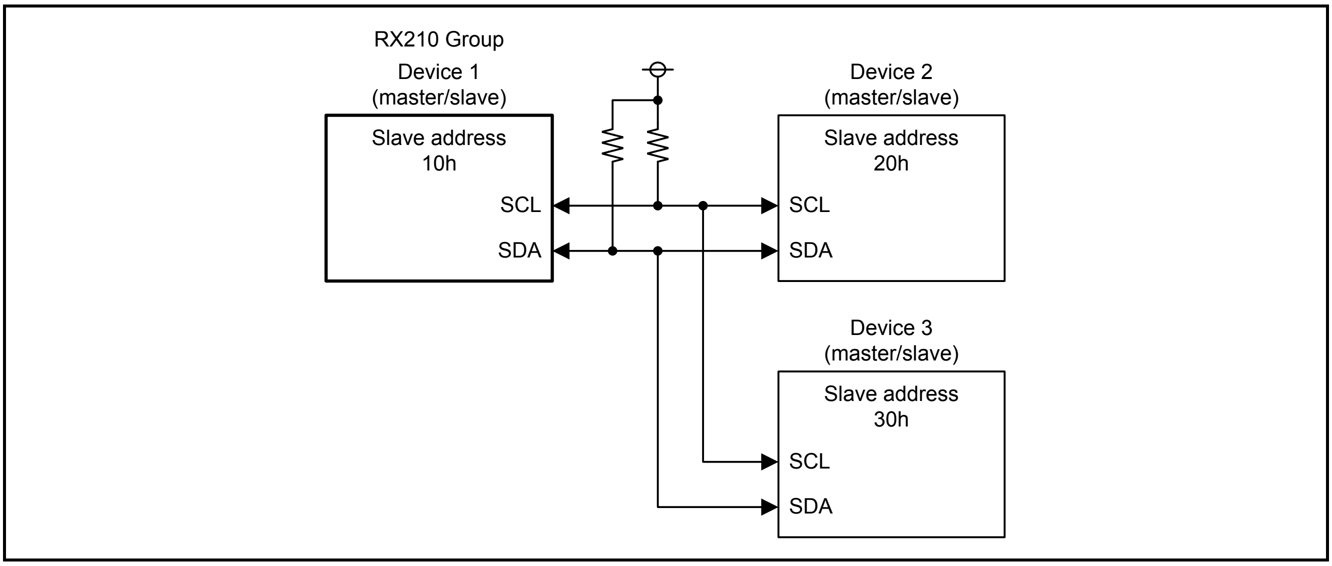


Figure 4.1 Connection Example

4.2 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

The pins described here are for 100-pin products. When the product with less than 100-pin is used, select appropriate pins for the product used.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
P12/SCL	I/O	Serial clock I/O pin
P13/SDA	I/O	Serial data I/O pin

5. Software

After a reset, perform the initialization to enable the RIIC operation and start the slave operation.

After confirming the I^2C bus is in the bus free state, start 10-byte master transmission with 20h as the slave address, 0 as the R/W bit value, and 00h to 09h as the transmit data. When the master transmission is completed, confirm the I^2C bus is in the bus free state, and start 10-byte master reception with 20h as the slave address and 1 as the R/W bit value. When the master reception is completed, confirm the I^2C bus is in the bus free state, and disable the RIIC operation.

When a master arbitration-lost is detected, the operation is automatically switched to slave reception mode. If the slave addresses match, a slave reception or slave transmission is performed according to the R/W bit setting.

When a NACK is detected, a transfer operation is canceled automatically. Then generate a stop condition and terminate the communication.

After a master transmission is started, when the following conditions are met, the callback function is called.

- An arbitration-lost is detected.
- Master transmission is completed.
- Master reception is completed.

Settings for peripheral functions are as follows:

RIIC

- Master/slave operations: Master transmission, master reception, slave reception, and slave transmission
- Address format: 7-bit address format
- Slave address: 10h
- Transfer rate: 100 kbps
- Arbitration-lost detection: Master arbitration-lost detection
- Interrupts: Transmit data empty interrupt (ICTXI) enabled
  Transmit end interrupt (ICTEI) enabled
  Receive data full interrupt (ICRXI) enabled
  NACK reception interrupt (NAKI) enabled
  Stop condition detection interrupt (SPI) enabled
  Arbitration-lost interrupt (ALI) enabled

5.1 Operation Overview

5.1.1 Master Transmission

(1) Start master transmission

Verify the ICCR2.BBSY flag is 0, then set the ICCR2.ST bit to 1 (requests to issue a start condition).

(2) Issue a start condition

When a start condition is issued, the ICSR2.TDRE flag becomes 1, and a TXI0 interrupt request is generated. In the TXI0 interrupt handling, write the slave address and value of the R/W# bit to the ICDRT register.

(3) Transmit data

When data is transferred from the ICDRT register to the ICDRS register, the TDRE flag becomes 1 again, and a TXI0 interrupt request is generated. Write the value of the master transmit buffer to the ICDRT register in the TXI0 interrupt handling. When the last data is written, in the subsequent TXI0 interrupt handling, set the ICIER.TIE bit to 0 (transmit data empty interrupt request (ICTXI) is disabled) and the ICIER.TEIE bit to 1 (transmit end interrupt request (ICTEI) is enabled).

(4) Complete the transmission

When the last data transmission is completed, the ICSR2.TEND flag becomes 1, and a TEI0 interrupt request is generated. Set the ICCR2.SP bit to 1 (requests to issue a stop condition) in the TEI0 interrupt handling.

(5) Issue a stop condition

When a stop condition is issued, the ICSR2.STOP flag becomes 1, and the EEI0 interrupt request is generated. In the EEI0 interrupt handling, set the TIE bit to 1, set the TEIE bit to 0, and call the callback function (completion of master transmission/reception).

Figure 5.1 shows the Timing Diagram of Master Transmission.

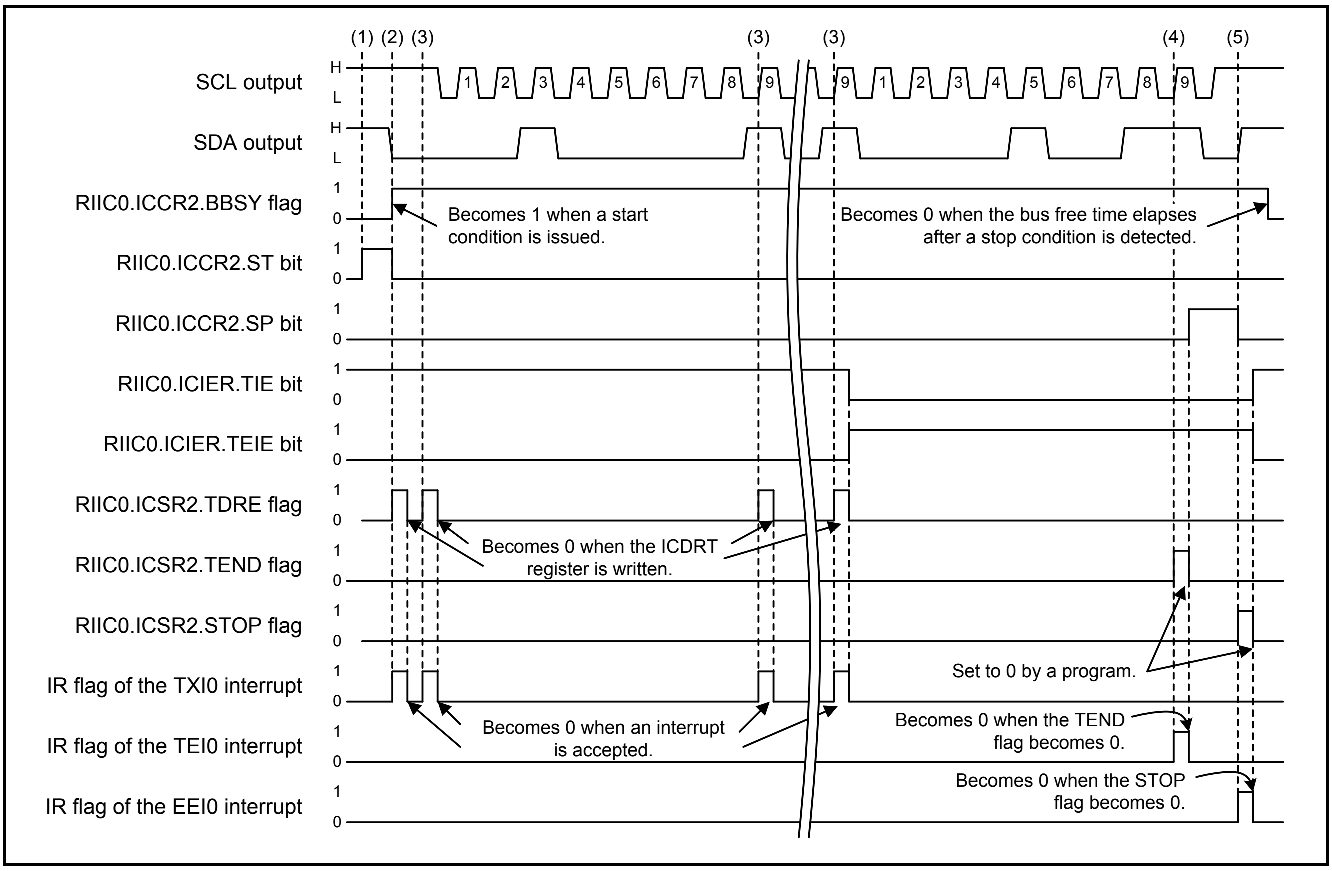

(1) (2) (3)
(3)
(3)
(4)
(5)
SCL output
SDA output
RIIC0.ICCR2.BBSY flag
RIIC0.ICCR2.ST bit
RIIC0.ICCR2.SP bit
RIIC0.ICIER.TIE bit
RIIC0.ICIER.TEIE bit
RIIC0.ICSR2.TDRE flag
RIIC0.ICSR2.TEND flag
RIIC0.ICSR2.STOP flag
IR flag of the TXI0 interrupt
IR flag of the TEI0 interrupt
IR flag of the EEI0 interrupt
H
L
1
0
1 2 3 4 5 6 7 8 9
9 1 2 3 4 5 6 7 8 9
Becomes 1 when a start condition is issued.
Becomes 0 when the bus free time elapses after a stop condition is detected.
Becomes 0 when the ICDRT register is written.
Set to 0 by a program.
Becomes 0 when an interrupt is accepted.
Becomes 0 when the TEND flag becomes 0.
Becomes 0 when the STOP flag becomes 0.

Figure 5.1 Timing Diagram of Master Transmission

5.1.2 Master Reception

(1) Start master reception

Verify the ICCR2.BBSY flag is 0, then set the ICCR2.ST bit to 1 (requests to issue a start condition).

(2) Issue a start condition

When a start condition is issued, the ICSR2.TDRE flag becomes 1, and a TXI0 interrupt request is generated. In the TXI0 interrupt handling, write the slave address and value of the R/W# bit to the ICDRT register. Set the ICIER.TIE bit to 0 (transmit data empty interrupt request (ICTXI) is disabled) and the IR flag of the TXI0 interrupt to 0.

(3) Complete the slave address transmission

When the transmission for the slave address is completed, the ICSR2.RDRF flag becomes 1, and a RXI0 interrupt request is generated. Dummy read the ICDRR register in the RXI0 interrupt handling.

(4) Complete the reception

When the data reception is completed, the RDRF flag becomes 1, and a RXI0 interrupt request is generated. In the RXI0 interrupt handling, the ICDRR register value is stored in the master receive buffer. When the reception for third to last byte of data is completed, set the ICMR3.WAIT bit to 1. When the reception for second to last byte of data is completed, set the ICMR3.RDRFS bit to 1 and the ICMR3.ACKBT bit to 1. When the last data reception is completed, set the ICCR2.SP bit to 1 (requests to issue a stop condition), the ACKBT bit to 1, and the WAIT bit to 0.

(5) Issue a stop condition

When a stop condition is issued, the ICSR2.STOP flag becomes 1, and the EEI0 interrupt request is generated. In the EEI0 interrupt handling, set the TIE bit to 1 and call the callback function (completion of master transmission/ reception).

Figure 5.2 shows the Timing Diagram of Master Reception.

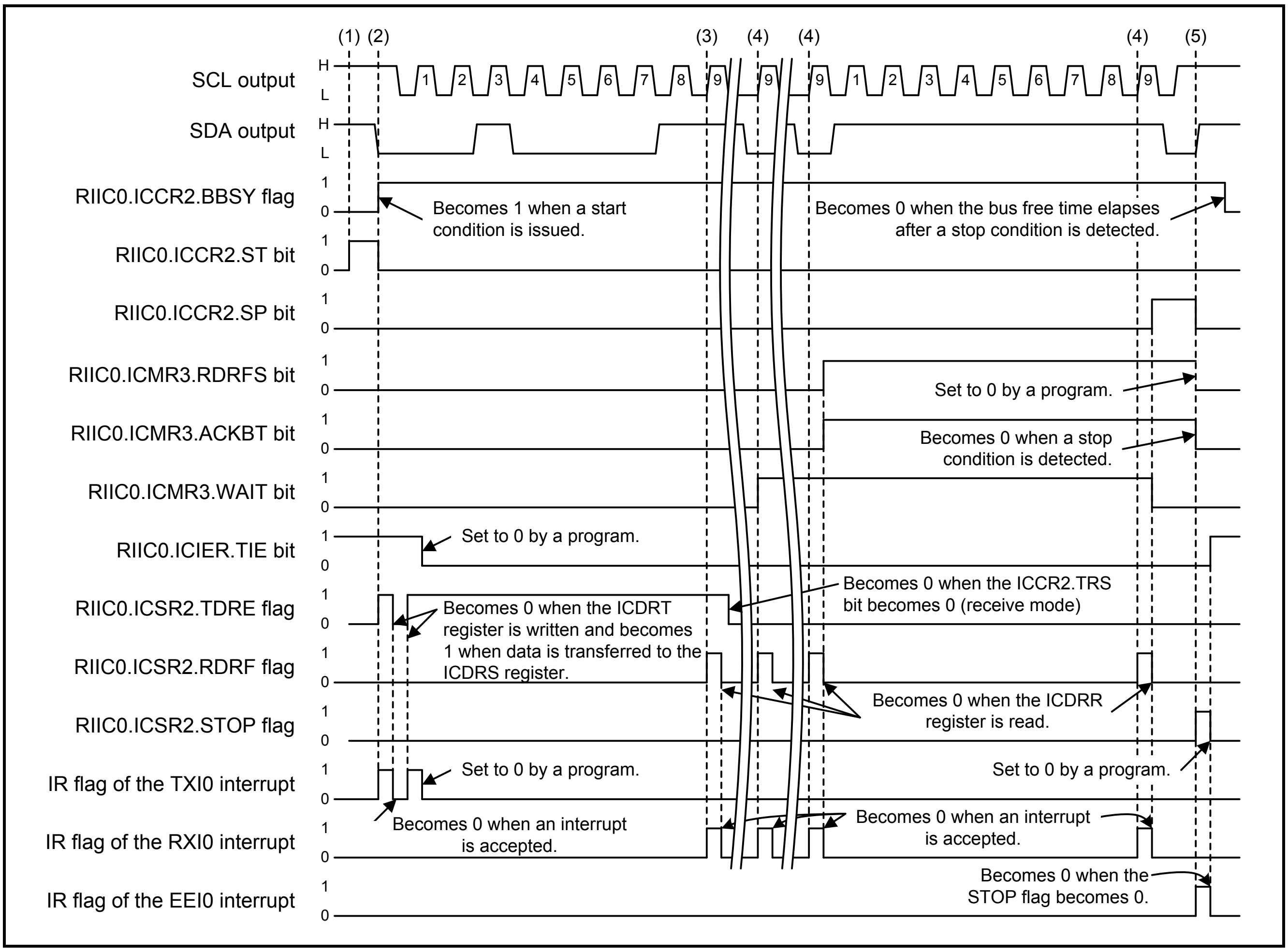

(1) (2)
(3) (4) (4)
(4) (5)
SCL output
SDA output
H
L
1
0
1 2 3 4 5 6 7 8 9 9 9 1 2 3 4 5 6 7 8 9
RIIC0.ICCR2.BBSY flag
RIIC0.ICCR2.ST bit
RIIC0.ICCR2.SP bit
RIIC0.ICMR3.RDRFS bit
RIIC0.ICMR3.ACKBT bit
RIIC0.ICMR3.WAIT bit
RIIC0.ICIER.TIE bit
RIIC0.ICSR2.TDRE flag
RIIC0.ICSR2.RDRF flag
RIIC0.ICSR2.STOP flag
IR flag of the TXI0 interrupt
IR flag of the RXI0 interrupt
IR flag of the EEI0 interrupt
Becomes 1 when a start condition is issued.
Becomes 0 when the bus free time elapses after a stop condition is detected.
Set to 0 by a program.
Becomes 0 when a stop condition is detected.
Set to 0 by a program.
Becomes 0 when the ICCR2.TRS bit becomes 0 (receive mode)
Becomes 0 when the ICDRT register is written and becomes 1 when data is transferred to the ICDRS register.
Becomes 0 when the ICDRR register is read.
Set to 0 by a program.
Set to 0 by a program.
Becomes 0 when an interrupt is accepted.
Becomes 0 when an interrupt is accepted.
Becomes 0 when the STOP flag becomes 0.

Figure 5.2 Timing Diagram of Master Reception

5.1.3 Slave Operation

When slave addresses match during a slave operation, a slave transmission or slave reception is started according to the R/W bit.
When a slave transmission or slave reception is completed, stop the slave operation and call the callback function.
When continuing the slave operation, call the user interface function (start a slave operation) again.
The sample code calls the user interface function (start a slave operation) with the callback function to continue the slave operation.

5.1.3.1 Slave Transmission

- After a slave transmission is started, transmit data in the slave transmit buffer from the top until a NACK is received.
- Transmit FFh when the number of transmissions exceeds the specified byte count of slave transmissions.
- When the transmission is completed, call the callback function. However if a NACK is received while the number of transmissions is less than the specified byte count of slave transmissions, the callback function is not called. When the slave transmission is started again, transmit data in the slave transmit buffer from the top.

5.1.3.2 Slave Reception

- After a slave reception is started, store received data in the slave receive buffer from the top until a stop condition is detected.
- Received data which exceeds the specified byte count of slave receptions is discarded without being stored in the slave receive buffer.
- When a reception is completed, call the callback function. However if a stop condition is detected while the number of receptions is less than the specified byte count of slave receptions, the callback function is not called. When the slave reception is started again, store data in the slave receive buffer from the top (overwrite data if it already exists).

5.2 File Composition

Table 5.1 lists the Files Used in the Sample Code. Files generated by the integrated development environment are not included in this table.

Table 5.1 Files Used in the Sample Code

File Name	Outline	Remarks
main.c	Main processing	
r_init_stop_module.c	Stop processing for active peripheral functions after a reset	
r_init_stop_module.h	Header file for r_init_stop_module.c	
r_init_non_existent_port.c	Nonexistent port initialization	
r_init_non_existent_port.h	Header file for r_init_non_existent_port.c	
r_init_clock.c	Clock initialization	
r_init_clock.h	Header file for r_init_clock.c	
riic.c	RIIC processing	
riic_int.c	RIIC interrupt handling	
riic.h	Header file for riic.c	

5.3 Option-Setting Memory

Table 5.2 lists the Option-Setting Memory Used in the Sample Code. When necessary, set a value suited to the user system.

Table 5.2 Option-Setting Memory Used in the Sample Code

Symbol	Address	Setting Value	Contents
OFS0	FFFF FF8Fh to FFFF FF8Ch	FFFF FFFFh	The IWDT is stopped after a reset. The WDT is stopped after a reset.
OFS1	FFFF FF8Bh to FFFF FF88h	FFFF FFFFh	The voltage monitor 0 reset is disabled after a reset. HOCO oscillation is disabled after a reset.
MDES	FFFF FF83h to FFFF FF80h	FFFF FFFFh	Little endian

5.4 Constants

Table 5.3 lists the Constants Used in the Sample Code.

Table 5.3 Constants Used in the Sample Code

Constant Name	Setting Value	Contents
SELF_ADDRESS	10h	Self-slave address
SLV_ADDRESS	20h	Slave address
MST_DATA_NUM	(10 + 1)	Byte count of master transmissions/receptions (byte count + slave address)
SLV_DATA_NUM	10	Byte count of slave transmissions/receptions
MST_WRITE	00h	Set value of the R/W bit: 0: Write
MST_READ	01h	Set value of the R/W bit: 1: Read
RIIC_OK	0	Return value of the RiicMstStart function
RIIC_NG	1	Return value of the RiicMstStart function
RIIC_BUS_BUSY	2	Return value of the RiicMstStart function
RIIC_ST_MST_STOP	0	Return value of the RiicGetMstState function
RIIC_ST_MST_BUSY	1	Return value of the RiicGetMstState function
RIIC_ST_MST_NACK	2	Return value of the RiicGetMstState function
RIIC_ST_MST_AL	3	Return value of the RiicGetMstState function
RIIC_ST_MST_COMPLETE	4	Return value of the RiicGetMstState function
RIIC_SET	1	Set the flag.
RIIC_CLEAR	0	Clear the flag.
RIIC_ENABLE	1	Enable the RIIC.
RIIC_DISABLE	0	Disable the RIIC.
RIIC_RXI	01h	Argument of the RiicIntEna and RiicIntDis functions
RIIC_TXI	02h	Argument of the RiicIntEna and RiicIntDis functions
RIIC_TEI	04h	Argument of the RiicIntEna and RiicIntDis functions

5.5 Variables

Table 5.4 lists the Global Variables, and Table 5.5 and Table 5.6 list the static Variables.

Table 5.4 Global Variables

Type	Variable Name	Contents	Function Used
uint8_t	MstTrmBuff[256]	Master transmit buffer	main
uint8_t	MstRcvBuff[256]	Master receive buffer	main
uint8_t	SlvTrmBuff[256]	Slave transmit buffer	main
uint8_t	SlvRcvBuff[256]	Slave receive buffer	main

Table 5.5 static Variables (1/2)

Type	Variable Name	Contents	Function Used
static uint8_t	RiicTrmAddr	Slave address	RiicMstStart RiicTDRE
static uint8_t *	RiicMstBuff	Pointer to a master transmit/receive buffer	RiicMstStart RiicRDRF RiicTDRE RiicTEND
static uint32_t	RiicMstCnt	Master transmission/reception coutner	RiicIni RiicMstStart RiicRDRF RiicTDRE RiicTEND
static uint32_t	RiicMstNum	Byte count of master transmissions/receptions	
static uint8_t *	RiicSlvTrmBuff	Pointer to a slave transmit buffer	RiicSlvStart RiicTDRE RiicSTOP
static uint8_t *	RiicSlvTrmStartBuff	Pointer to a slave transmit buffer	RiicSlvStart RiicSTOP
static uint32_t	RiicSlvTrmCnt	Slave transmission counter	RiicIni RiicSlvStart RiicTDRE RiicTEND RiicSTOP
static uint8_t *	RiicSlvRcvBuff	Pointer to a slave receive buffer	RiicSlvStart RiicRDRF RiicSTOP
static uint8_t *	RiicSlvRcvStartBuff	Pointer to a slave receive buffer	RiicSlvStart RiicSTOP
static uint32_t	RiicSlvRcvCnt	Slave reception counter	RiicIni RiicSlvStart RiicRDRF RiicSTOP
static uint32_t	RiicSlvNum	Byte count of slave transmissions/receptions	RiicIni RiicSlvStart RiicRDRF RiicTDRE RiicTEND RiicSTOP

Table 5.6 static Variables (2/2)

Type	Variable Name	Contents	Function Used
static volatile uint8_t	RiicStartFlg	Communication flag: 0: Not communicating 1: Communicating	RiicIni RiicRDRF RiicTDRE RiicSTOP RiicAL
static volatile uint8_t	RiicMstFlg	Master operation flag: 0: Slave operation 1: Master operation	RiicMstStart RiicRDRF RiicTDRE RiicTEND RiicSTOP RiicNACK RiicAL
static volatile uint8_t	RiicMstState	Master state: 0: Stopped 1: Busy (communicating) 2: NACK detected 3: Arbitration-lost detected 4: Communication completed	RiicIni RiicMstStart RiicGetMstState RiicSTOP RiicNACK RiicAL

5.6 Functions

Table 5.7 lists the Functions.

Table 5.7 Functions

Function Name	Outline
main	Main processing
port_init	Port initialization
R_INIT_StopModule	Stop processing for active peripheral functions after a reset
R_INIT_NonExistentPort	Nonexistent port initialization
R_INIT_Clock	Clock initialization
CbSlaveTrm	Callback function (completion of slave transmission)
CbSlaveRcv	Callback function (completion of slave reception)
CbMaster	Callback function (completion of master transmission/reception)
RiicIni	User interface function (RIIC initialization)
RiicSlvStart	User interface function (start a slave operation)
RiicMstStart	User interface function (start a master operation)
RiicGetMstState	User interface function (obtain the master state)
RiicEnable	Enabling the RIIC
RiicDisable	Disabling the RIIC
RiicIntEna	Enabling the RIIC interrupts
RiicIntDis	Disabling the RIIC interrupts
RiicRDRF	Receive data full interrupt
RiicTDRE	Transmit data empty interrupt
RiicTEND	Transmit end interrupt
RiicSTOP	Stop condition detection interrupt
RiicNACK	NACK detection interrupt
RiicAL	Arbitration-lost detection interrupt
RiicTMO	Timeout detection interrupt
RiicSTART	Start condition detection interrupt
Excep_RIIC0_EEI0	RIIC0.EEI0 interrupt handling
Excep_RIIC0_RXI0	RIIC0.RXI0 interrupt handling
Excep_RIIC0_TXI0	RIIC0.TXI0 interrupt handling
Excep_RIIC0_TEI0	RIIC0.TEI0 interrupt handling

5.7 Function Specifications

The following tables list the sample code function specifications.

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	After initialization, perform 10-byte master transmission and 10-byte master reception.
Arguments	None
Return Value	None

port_init	
Outline	Port initialization
Header	None
Declaration	void port_init(void)
Description	Initialize ports.
Arguments	None
Return Value	None

R_INIT_StopModule	
Outline	Stop processing for active peripheral functions after a reset
Header	r_init_stop_module.h
Declaration	void R_INIT_StopModule(void)
Description	Configure the setting to enter the module-stop state.
Arguments	None
Return Value	None
Remarks	Transition to the module-stop state is not performed in the sample code. For details on this function, refer to the Initial Setting application note for the product used.

R_INIT_NonExistentPort	
Outline	Nonexistent port initialization
Header	r_init_non_existent_port.h
Declaration	void R_INIT_NonExistentPort(void)
Description	Initialize port direction registers for ports that do not exist in products with less than 100 pins.
Arguments	None
Return Value	None
Remarks	The number of pins in the sample code is set for the 100-pin package (PIN_SIZE=100). After this function is called, when writing in byte units to the PDR registers or PODR registers which have nonexistent ports, set the corresponding bits for nonexistent ports as follows: set the I/O select bits in the PDR registers to 1 and set the output data store bits in the PODR registers to 0. For details on this function, refer to the Initial Setting application note for the product used.

R_INIT_Clock	
Outline	Clock initialization
Header	r_init_clock.h
Declaration	void R_INIT_Clock(void)
Description	Initialize the clock.
Arguments	None
Return Value	None
Remarks	The sample code selects processing which uses PLL as the system clock without using the sub-clock. For details on this function, refer to the Initial Setting application note for the product used.

CbSlaveTrm	
Outline	Callback function (completion of slave transmission)
Header	None
Declaration	void CbSlaveTrm(void)
Description	This function is called when a slave transmission is completed.
Arguments	None
Return Value	None

CbSlaveRcv	
Outline	Callback function (completion of slave reception)
Header	None
Declaration	void CbSlaveRcv(void)
Description	This function is called when a slave reception is completed.
Arguments	None
Return Value	None

CbMaster	
Outline	Callback function (completion of master transmission/reception)
Header	None
Declaration	void CbMaster(void)
Description	This function is called when the following conditions are met after a master communication is started. - An arbitration-lost is detected. - A master transmission is completed. - A master reception is completed.
Arguments	None
Return Value	None
Remarks	The processing of this function is not included in the sample code. Add a program as required.

RiicIni		
Outline	User interface function (RIIC initialization)	
Header	riic.h	
Declaration	void RiicIni(uint8_t in_SelfAddr, uint8_t in_Enable)	
Description	Initialize the RIIC.	
Arguments	uint8_t in_SelfAddr	Self-address (bit 0 is set to 0)
	uint8_t in_Enable	RIIC enabled/disabled: RIIC_ENABLE: RIIC enabled RIIC_DISABLE: RIIC disabled
Return Value	None	

RiicSlvStart		
Outline	User interface function (start a slave operation)	
Header	riic.h	
Declaration	void RiicSlvStart(uint8_t * in_RcvAddr, uint8_t * in_TrmAddr, uint32_t in_num, CallBackFunc cbTrm, CallBackFunc cbRcv)	
Description	Start a slave operation.	
Arguments	uint8_t * in_RcvAddr	Pointer to a slave receive data storage
	uint8_t * in_TrmAddr	Pointer to a slave transmit data storage
	uint32_t in_num	Byte count of slave transmissions/receptions
	CallBackFunc cbTrm	Callback function (completion of a slave transmission)
	CallBackFunc cbRcv	Callback function (completion of a slave reception)
Return Value	None	

RiicMstStart		
Outline	User interface function (start a master operation)	
Header	riic.h	
Declaration	uint8_t RiicMstStart(uint8_t in_addr, uint8_t * in_buff, uint32_t in_num, CallBackFunc cb)	
Description	Start a master operation.	
Arguments	uint8_t in_addr	Slave address (bit 0 is the R/W bit)
	uint8_t * in_buff	Pointer to a master transmit/receive data storage
	uint32_t in_num	Byte count of master transmissions/receptions
	CallBackFunc cb	Callback function (completion of a master transmission/reception)
Return Value	RIIC_OK: Completed successfully RIIC_NG: Argument error (byte count of transmissions/receptions is less than 2) RIIC_BUS_BUSY: Bus busy	

RiicGetMstState	
Outline	User interface function (obtain the master state)
Header	riic.h
Declaration	uint8_t RiicGetMstState(void)
Description	Return the master state.
Arguments	None
Return Value	RIIC_ST_MST_STOP: Stopped RIIC_ST_MST_BUSY: Busy (communicating) RIIC_ST_MST_NACK: NACK detected RIIC_ST_MST_AL: Arbitration-lost detected RIIC_ST_MST_COMPLETE: Communication completed

RiicEnable		
Outline	Enabling the RIIC	
Header	None	
Declaration	void RiicEnable(uint8_t addr)	
Description	Enable the RIIC operation.	
Arguments	uint8_t addr	Self-address
Return Value	None	

RiicDisable		
Outline	Disabling the RIIC	
Header	None	
Declaration	void RiicDisable(void)	
Description	Disable the RIIC operation.	
Arguments	None	
Return Value	None	

RiicIntEna		
Outline	Enabling the RIIC interrupts	
Header	None	
Declaration	void RiicIntEna(uint8_t req)	
Description	Enable the RIIC interrupt request.	
Arguments	uint8_t req	Requests enabled: RIIC_RXI: Enable the RXI interrupt request RIIC_TXI: Enable the TXI interrupt request RIIC_TEI: Enable the TEI interrupt request
Return Value	None	

RiicIntDis		
Outline	Disabling the RIIC interrupts	
Header	None	
Declaration	void RiicIntDis(uint8_t req)	
Description	Disable the RIIC interrupt request.	
Arguments	uint8_t req	Requests disabled: RIIC_RXI: Disable the RXI interrupt request RIIC_TXI: Disable the TXI interrupt request RIIC_TEI: Disable the TEI interrupt request
Return Value	None	

RiicRDRF		
Outline	Receive data full interrupt	
Header	riic.h	
Declaration	void RiicRDRF(void)	
Description	This function is called from the RIIC0.RXI0 interrupt handling and reads the receive data.	
Arguments	None	
Return Value	None	

RiicTDRE	
Outline	Transmit data empty interrupt
Header	riic.h
Declaration	void RiicTDRE(void)
Description	This function is called from the RIIC0.TXI0 interrupt handling and writes the transmit data.
Arguments	None
Return Value	None

RiicTEND	
Outline	Transmit end interrupt
Header	riic.h
Declaration	void RiicTEND(void)
Description	This function is called from the RIIC0.TEI0 interrupt handling and issues a stop condition.
Arguments	None
Return Value	None

RiicSTOP	
Outline	Stop condition detection interrupt
Header	riic.h
Declaration	void RiicSTOP(void)
Description	This function is called from the RIIC0.EEI0 interrupt handling and calls the callback function.
Arguments	None
Return Value	None

RiicNACK	
Outline	NACK detection interrupt
Header	riic.h
Declaration	void RiicNACK(void)
Description	This function is called from the RIIC0.EEI0 interrupt handling and issues a stop condition.
Arguments	None
Return Value	None

RiicAL	
Outline	Arbitration-lost detection interrupt
Header	riic.h
Declaration	void RiicAL(void)
Description	This function is called from the RIIC0.EEI0 interrupt handling and calls the callback function.
Arguments	None
Return Value	None

RiicTMO	
Outline	Timeout detection interrupt
Header	riic.h
Declaration	void RiicTMO(void)
Description	This function is called from the RIIC0.EEI0 interrupt handling.
Arguments	None
Return Value	None
Remarks	The processing of this function is not included in the sample code. Add a program as required.

RiicSTART	
Outline	Start condition detection interrupt
Header	None
Declaration	void RiicSTART(void)
Description	This function is called from the RIIC0.EEI0 interrupt handling.
Arguments	None
Return Value	None
Remarks	The processing of this function is not included in the sample code. Add a program as required.

Excep_RIIC0_EEI0	
Outline	RIIC0.EEI0 interrupt handling
Header	None
Declaration	void Excep_RIIC0_EEI0(void)
Description	This function performs interrupt handling when a communication error or event occurs.
Arguments	None
Return Value	None

Excep_RIIC0_RXI0	
Outline	RIIC0.RXI0 interrupt handling
Header	None
Declaration	void Excep_RIIC0_RXI0(void)
Description	This function performs receive data full interrupt handling.
Arguments	None
Return Value	None

Excep_RIIC0_TXI0	
Outline	RIIC0.TXI0 interrupt handling
Header	None
Declaration	void Excep_RIIC0_TXI0(void)
Description	This function performs transmit data empty interrupt handling.
Arguments	None
Return Value	None

Excep_RIIC0_TEI0	
Outline	RIIC0.TEI0 interrupt handling
Header	None
Declaration	void Excep_RIIC0_TEI0(void)
Description	This function performs transmit end interrupt handling.
Arguments	None
Return Value	None

5.8 Flowcharts

5.8.1 Main Processing

Figure 5.3 shows the Main Processing.

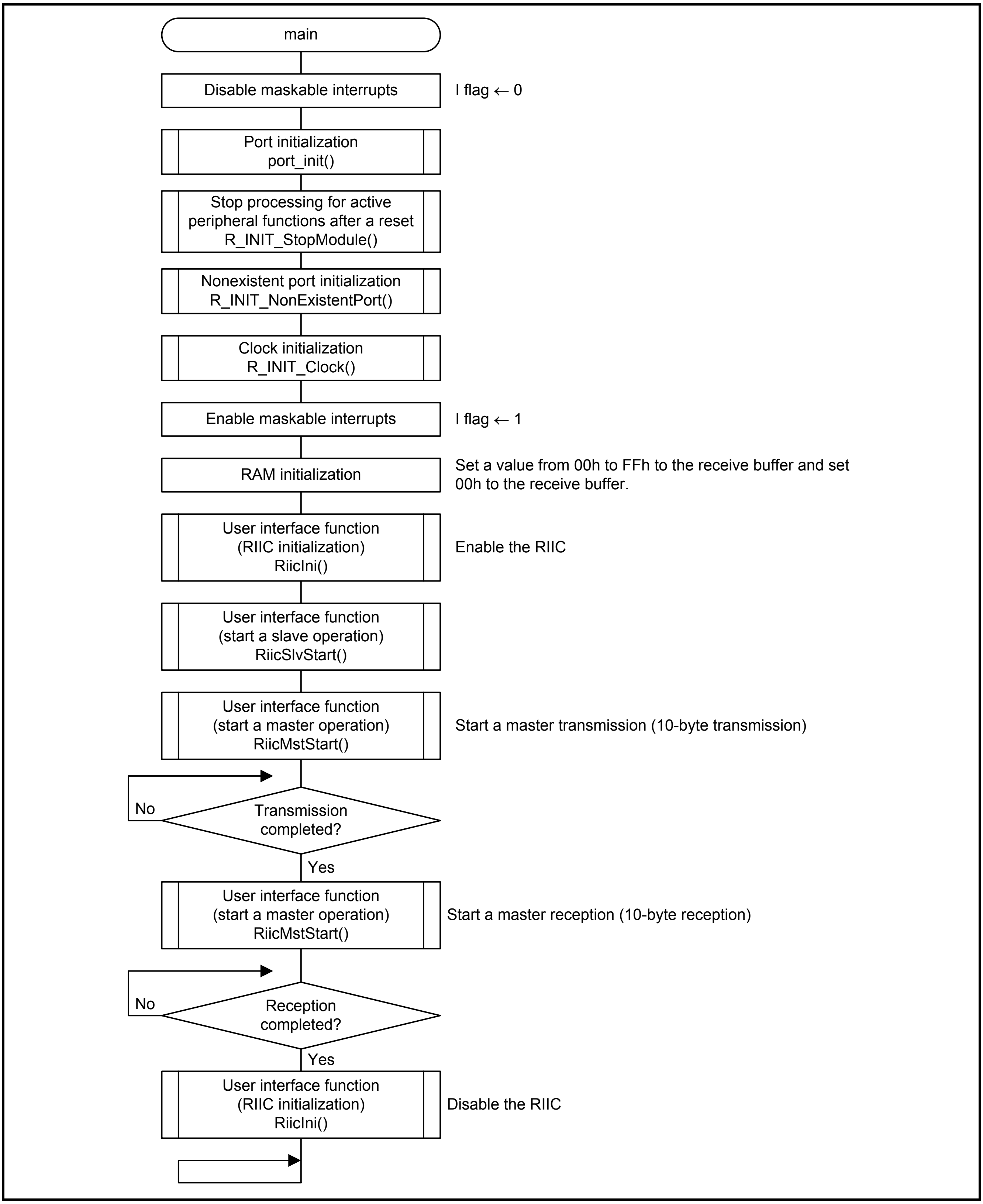


Figure 5.3 Main Processing

5.8.2 Port Initialization

Figure 5.4 shows the Port Initialization.

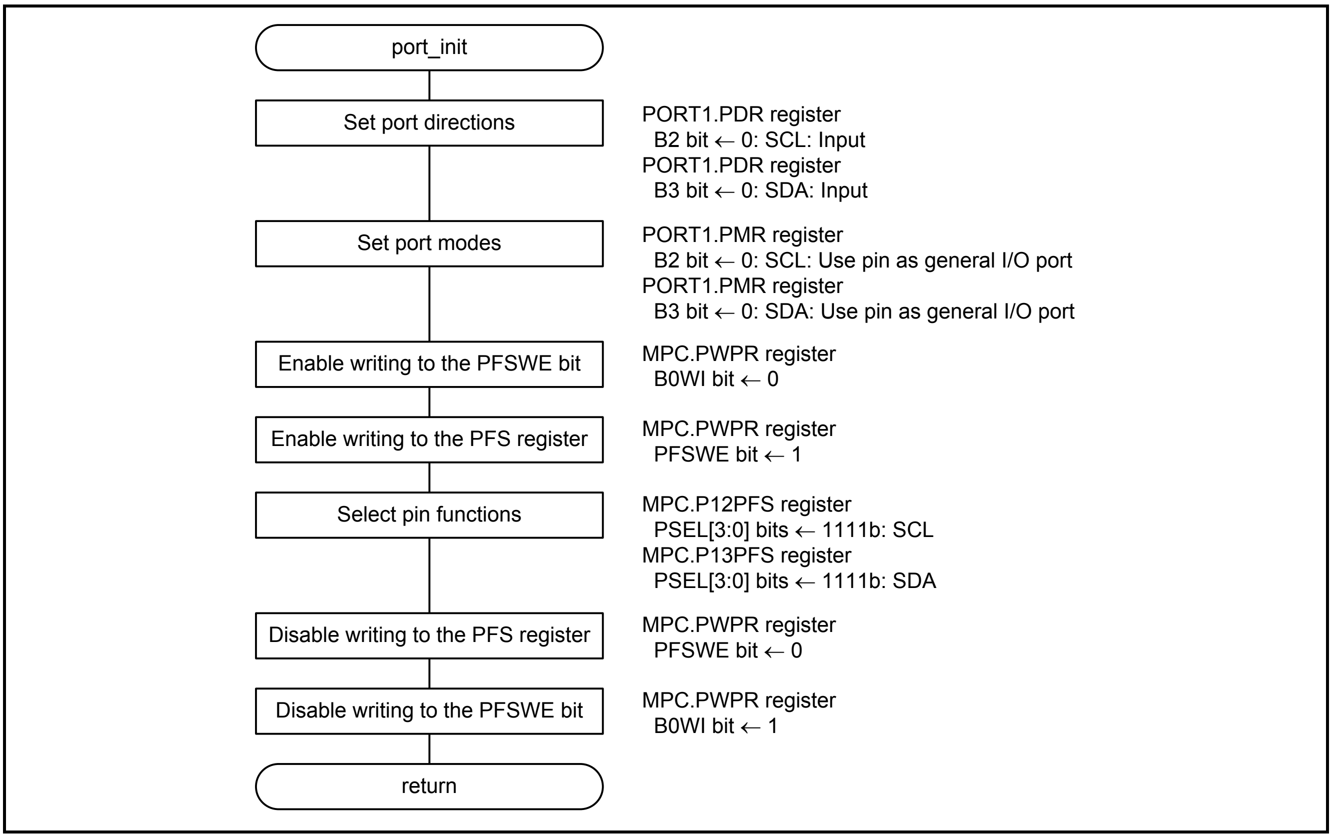


Figure 5.4 Port Initialization

5.8.3 Callback Function (Completion of Slave Transmission)

Figure 5.5 shows the Callback Function (Completion of Slave Transmission).

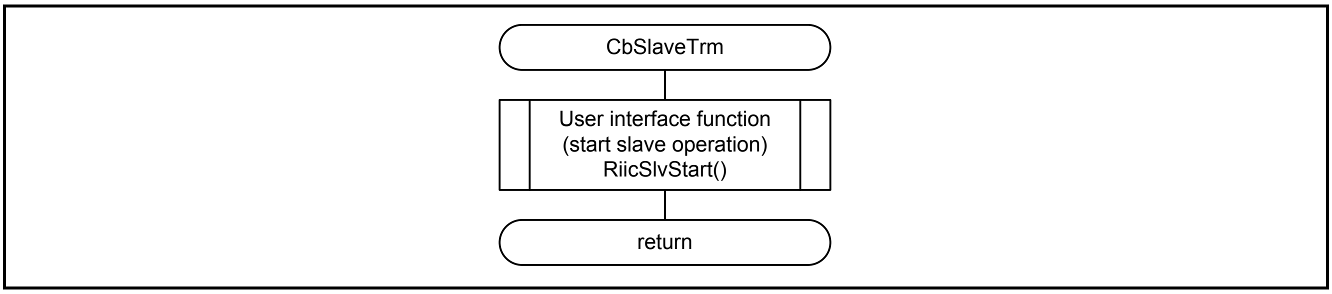


Figure 5.5 Callback Function (Completion of Slave Transmission)

5.8.4 Callback Function (Completion of Slave Reception)

Figure 5.6 shows the Callback Function (Completion of Slave Reception).

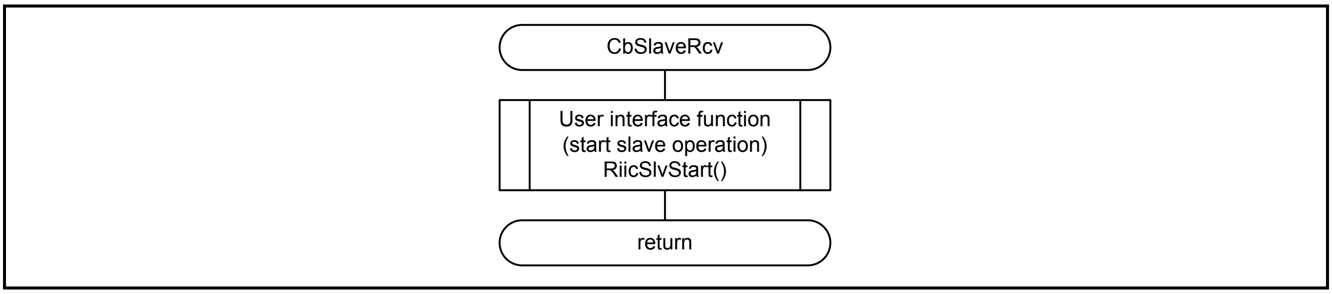
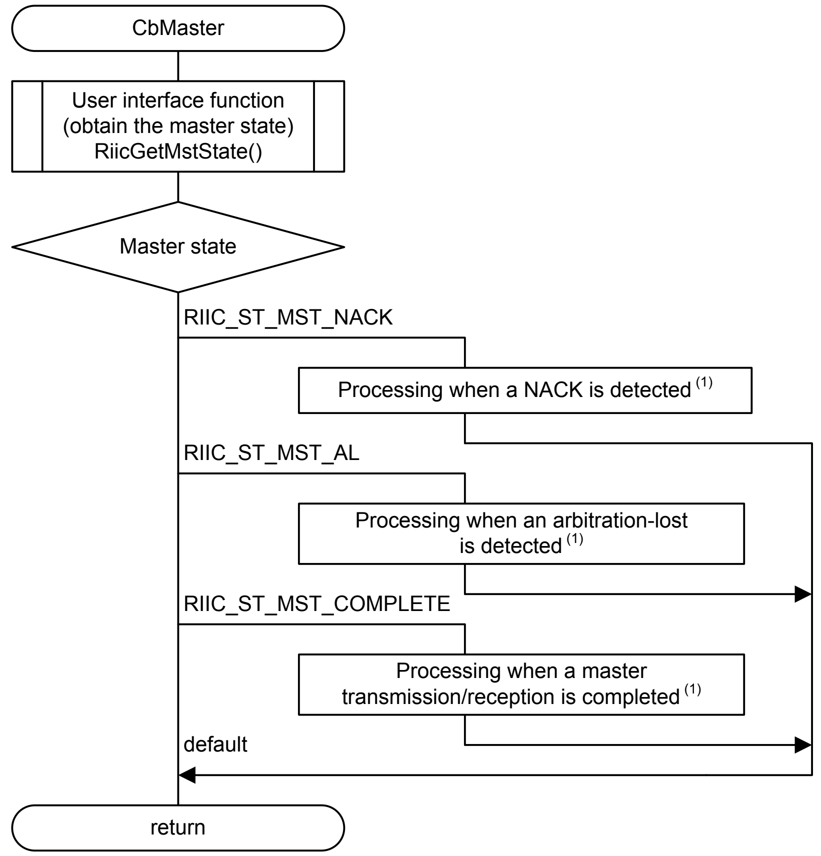


Figure 5.6 Callback Function (Completion of Slave Reception)

5.8.5 Callback Function (Completion of Master Transmission/Reception)

Figure 5.7 shows the Callback Function (Completion of Master Transmission/Reception).


CbMaster
User interface function
(obtain the master state)
RiicGetMstState()
Master state
RIIC_ST_MST_NACK
Processing when a NACK is detected (1)
RIIC_ST_MST_AL
Processing when an arbitration-lost is detected (1)
RIIC_ST_MST_COMPLETE
Processing when a master transmission/reception is completed (1)
default
return

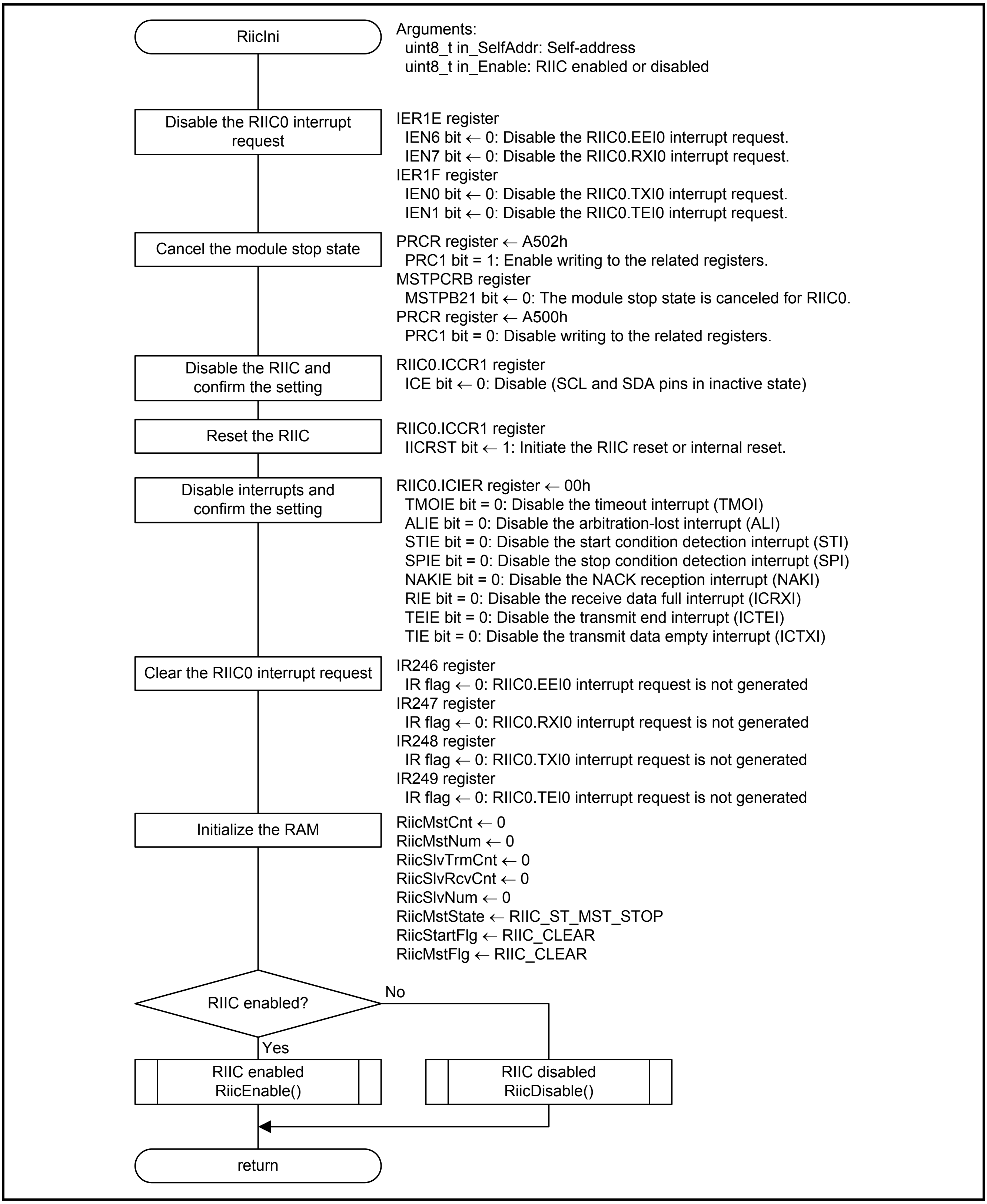
Note:

1. The processing of this function is not included in the sample code. Add a program as required.

Figure 5.7 Callback Function (Completion of Master Transmission/Reception)

5.8.6 User Interface Function (RIIC Initialization)

Figure 5.8 shows the User Interface Function (RIIC Initialization).


RiicIni
Disable the RIIC0 interrupt request
Cancel the module stop state
Disable the RIIC and confirm the setting
Reset the RIIC
Disable interrupts and confirm the setting
Clear the RIIC0 interrupt request
Initialize the RAM
RIIC enabled?
No
Yes
RIIC enabled RiicEnable()
RIIC disabled RiicDisable()
return

Arguments:
  uint8_t in_SelfAddr: Self-address
  uint8_t in_Enable: RIIC enabled or disabled

IER1E register
  IEN6 bit ← 0: Disable the RIIC0.EEI0 interrupt request.
  IEN7 bit ← 0: Disable the RIIC0.RXI0 interrupt request.
IER1F register
  IEN0 bit ← 0: Disable the RIIC0.TXI0 interrupt request.
  IEN1 bit ← 0: Disable the RIIC0.TEI0 interrupt request.

PRCR register ← A502h
  PRC1 bit = 1: Enable writing to the related registers.
MSTPCRB register
  MSTPB21 bit ← 0: The module stop state is canceled for RIIC0.
PRCR register ← A500h
  PRC1 bit = 0: Disable writing to the related registers.

RIIC0.ICCR1 register
  ICE bit ← 0: Disable (SCL and SDA pins in inactive state)

RIIC0.ICCR1 register
  IICRST bit ← 1: Initiate the RIIC reset or internal reset.

RIIC0.ICIER register ← 00h
  TMOIE bit = 0: Disable the timeout interrupt (TMOI)
  ALIE bit = 0: Disable the arbitration-lost interrupt (ALI)
  STIE bit = 0: Disable the start condition detection interrupt (STI)
  SPIE bit = 0: Disable the stop condition detection interrupt (SPI)
  NAKIE bit = 0: Disable the NACK reception interrupt (NAKI)
  RIE bit = 0: Disable the receive data full interrupt (ICRXI)
  TEIE bit = 0: Disable the transmit end interrupt (ICTEI)
  TIE bit = 0: Disable the transmit data empty interrupt (ICTXI)

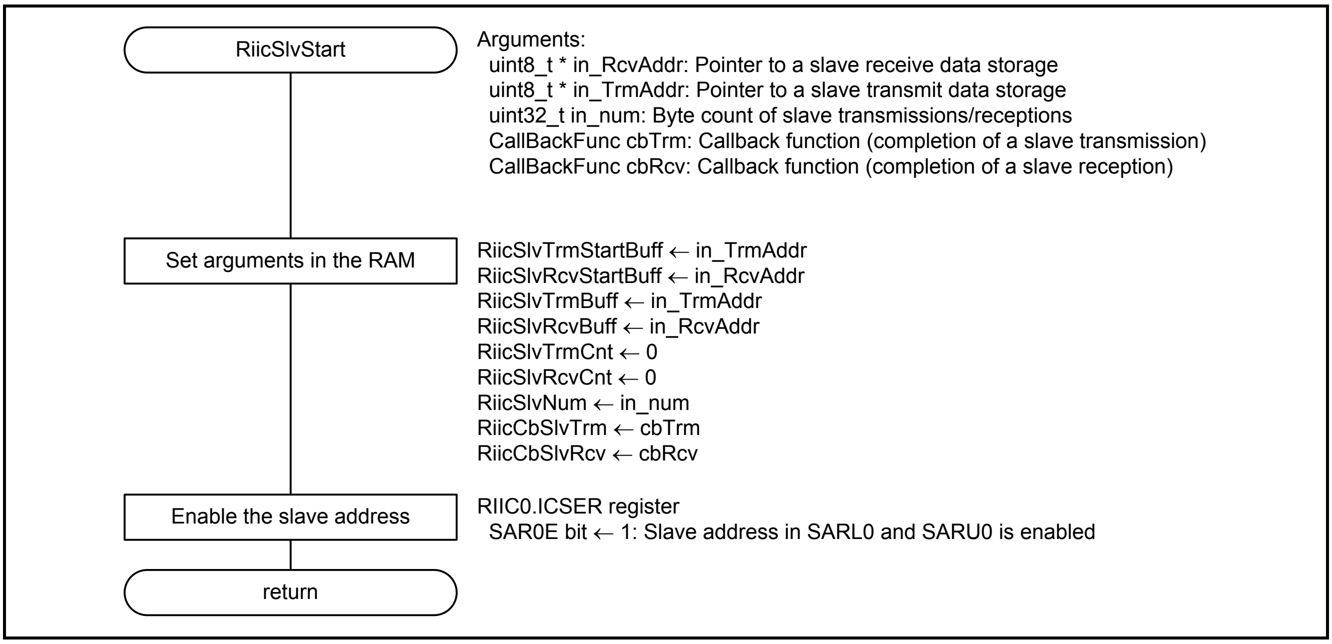
IR246 register
  IR flag ← 0: RIIC0.EEI0 interrupt request is not generated
IR247 register
  IR flag ← 0: RIIC0.RXI0 interrupt request is not generated
IR248 register
  IR flag ← 0: RIIC0.TXI0 interrupt request is not generated
IR249 register
  IR flag ← 0: RIIC0.TEI0 interrupt request is not generated

RiicMstCnt ← 0
RiicMstNum ← 0
RiicSlvTrmCnt ← 0
RiicSlvRcvCnt ← 0
RiicSlvNum ← 0
RiicMstState ← RIIC_ST_MST_STOP
RiicStartFlg ← RIIC_CLEAR
RiicMstFlg ← RIIC_CLEAR

Figure 5.8 User Interface Function (RIIC Initialization)

5.8.7 User Interface Function (Start a Slave Operation)

Figure 5.9 shows the User Interface Function (Start a Slave Operation).



Arguments:
  uint8_t * in_RcvAddr: Pointer to a slave receive data storage
  uint8_t * in_TrmAddr: Pointer to a slave transmit data storage
  uint32_t in_num: Byte count of slave transmissions/receptions
  CallBackFunc cbTrm: Callback function (completion of a slave transmission)
  CallBackFunc cbRcv: Callback function (completion of a slave reception)

RiicSlvTrmStartBuff ← in_TrmAddr
RiicSlvRcvStartBuff ← in_RcvAddr
RiicSlvTrmBuff ← in_TrmAddr
RiicSlvRcvBuff ← in_RcvAddr
RiicSlvTrmCnt ← 0
RiicSlvRcvCnt ← 0
RiicSlvNum ← in_num
RiicCbSlvTrm ← cbTrm
RiicCbSlvRcv ← cbRcv

RIIC0.ICSER register
  SAR0E bit ← 1: Slave address in SARL0 and SARU0 is enabled

Figure 5.9 User Interface Function (Start a Slave Operation)

5.8.8 User Interface Function (Start a Master Operation)

Figure 5.10 shows the User Interface Function (Start a Master Operation).

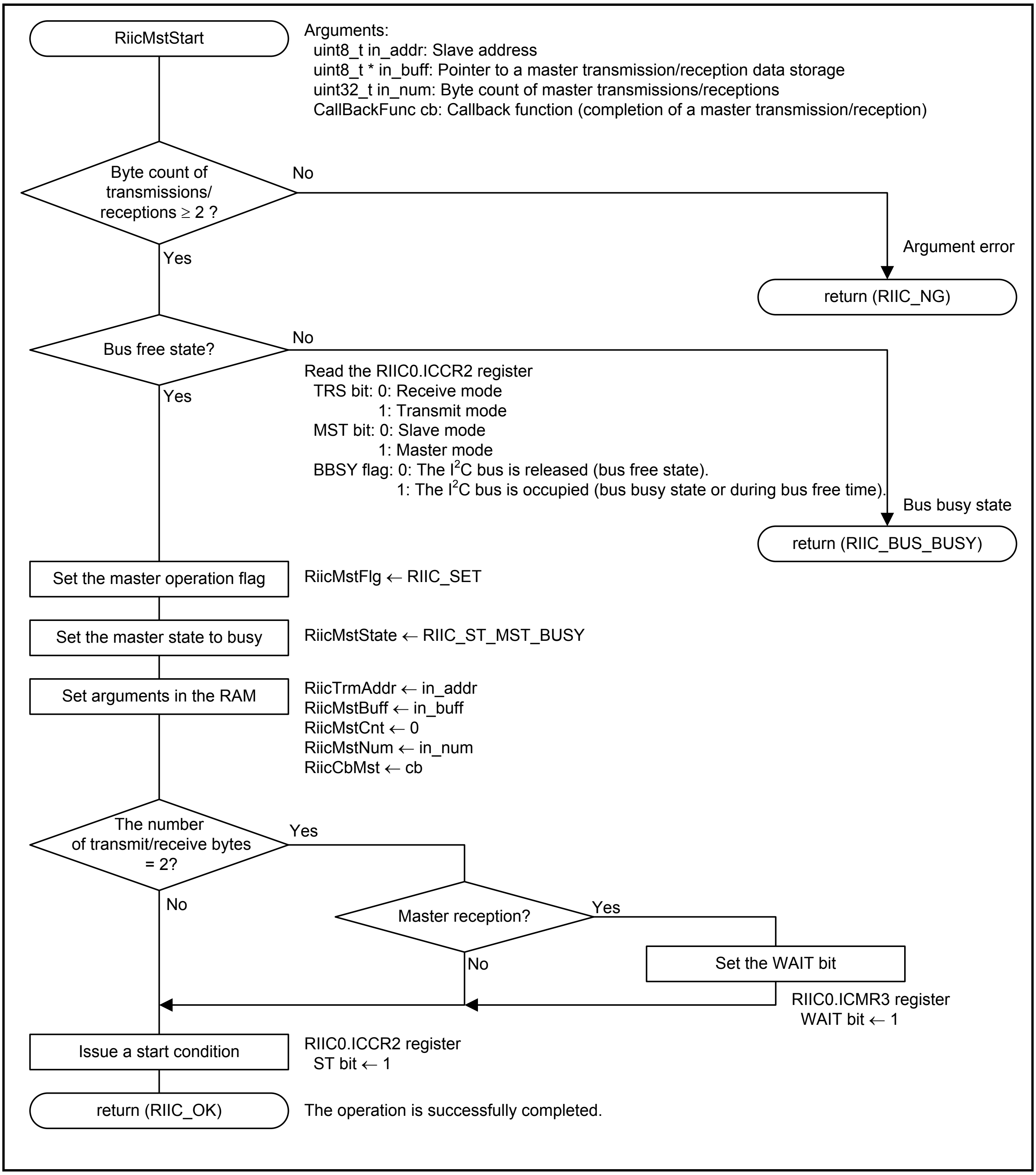

RiicMstStart
Arguments:
uint8_t in_addr: Slave address
uint8_t * in_buff: Pointer to a master transmission/reception data storage
uint32_t in_num: Byte count of master transmissions/receptions
CallBackFunc cb: Callback function (completion of a master transmission/reception)
Byte count of transmissions/ receptions ≥ 2 ?
No
Yes
Argument error
return (RIIC_NG)
Bus free state?
No
Yes
Read the RIIC0.ICCR2 register
TRS bit: 0: Receive mode
1: Transmit mode
MST bit: 0: Slave mode
1: Master mode
BBSY flag: 0: The I2C bus is released (bus free state).
1: The I2C bus is occupied (bus busy state or during bus free time).
Bus busy state
return (RIIC_BUS_BUSY)
Set the master operation flag
RiicMstFlg ← RIIC_SET
Set the master state to busy
RiicMstState ← RIIC_ST_MST_BUSY
Set arguments in the RAM
RiicTrmAddr ← in_addr
RiicMstBuff ← in_buff
RiicMstCnt ← 0
RiicMstNum ← in_num
RiicCbMst ← cb
The number of transmit/receive bytes = 2?
Yes
No
Master reception?
Yes
No
Set the WAIT bit
RIIC0.ICMR3 register
WAIT bit ← 1
Issue a start condition
RIIC0.ICCR2 register
ST bit ← 1
return (RIIC_OK)
The operation is successfully completed.

Figure 5.10 User Interface Function (Start a Master Operation)

5.8.9 User Interface Function (Obtain the Master State)

Figure 5.11 shows the User Interface Function (Obtain the Master State).

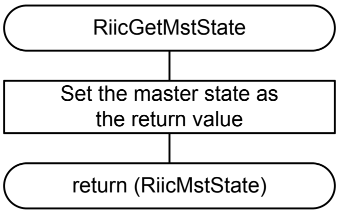


Figure 5.11 User Interface Function (Obtain the Master State)

5.8.10 Enabling the RIIC

Figure 5.12 shows the Enabling the RIIC.

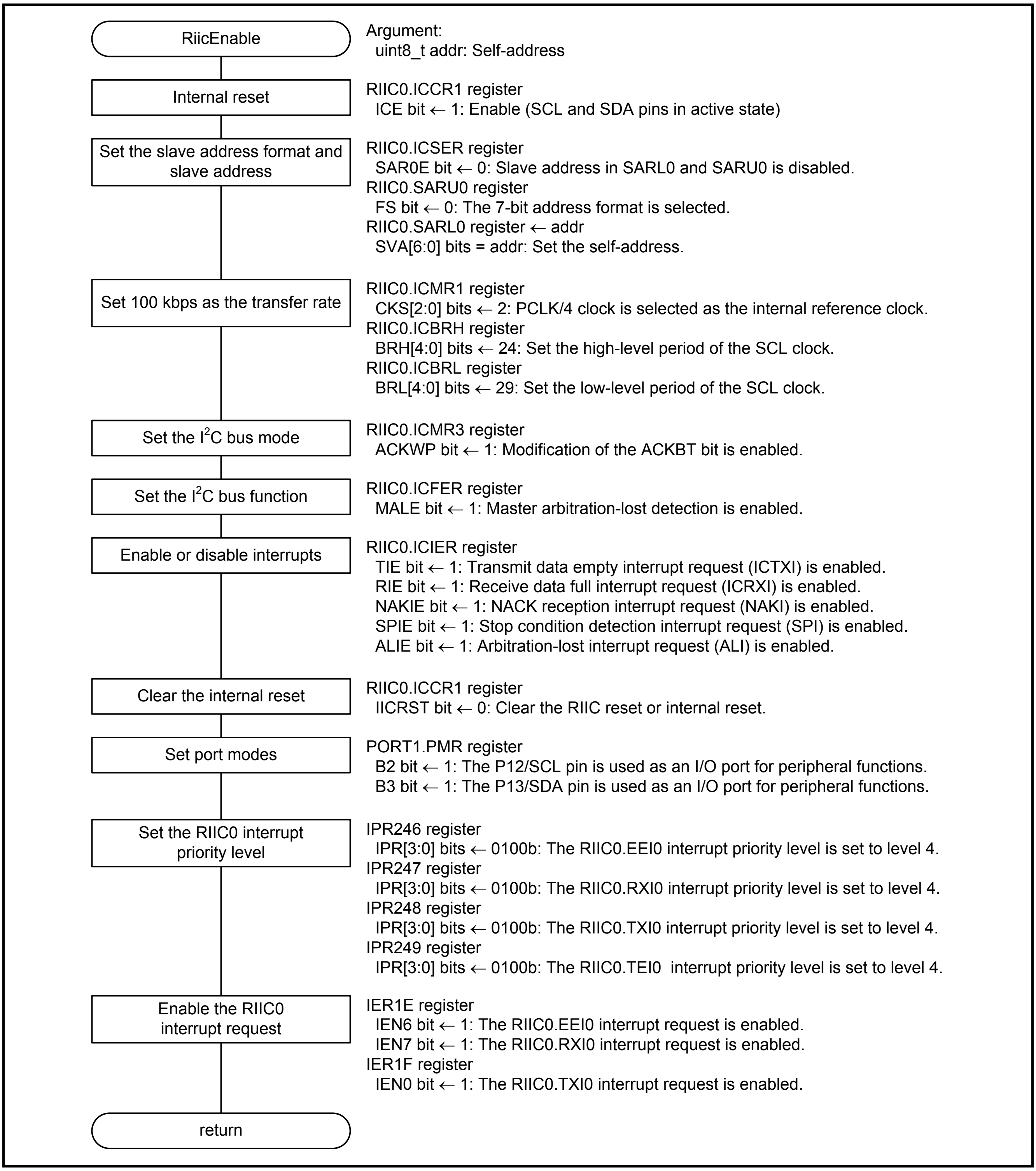
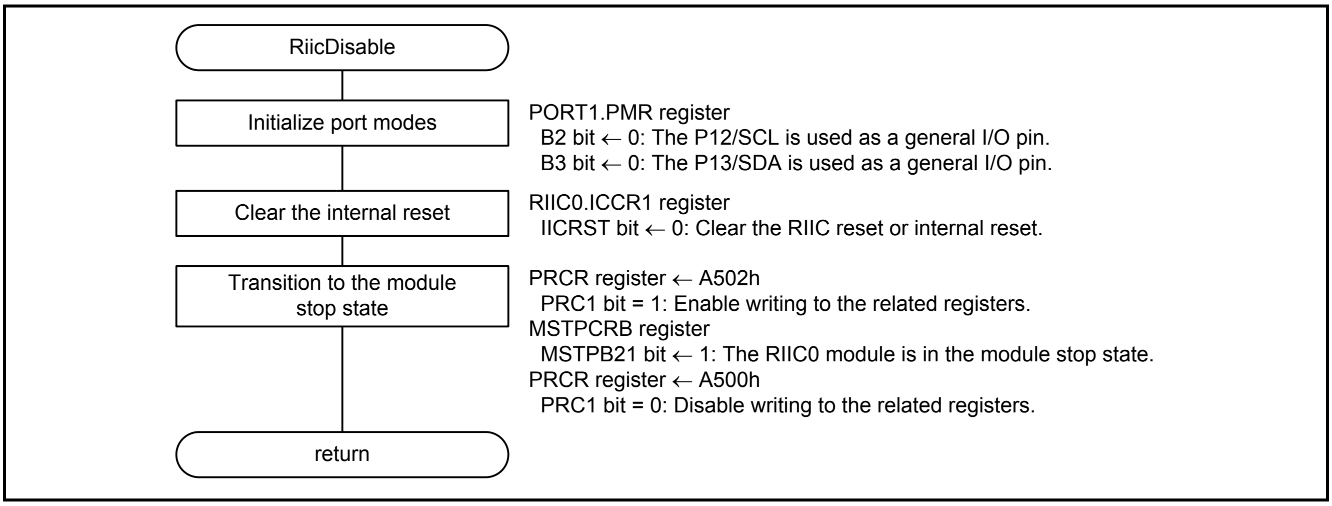

RiicEnable
Argument:
uint8_t addr: Self-address
Internal reset
RIIC0.ICCR1 register
ICE bit ← 1: Enable (SCL and SDA pins in active state)
Set the slave address format and slave address
RIIC0.ICSER register
SAR0E bit ← 0: Slave address in SARL0 and SARU0 is disabled.
RIIC0.SARU0 register
FS bit ← 0: The 7-bit address format is selected.
RIIC0.SARL0 register ← addr
SVA[6:0] bits = addr: Set the self-address.
Set 100 kbps as the transfer rate
RIIC0.ICMR1 register
CKS[2:0] bits ← 2: PCLK/4 clock is selected as the internal reference clock.
RIIC0.ICBRH register
BRH[4:0] bits ← 24: Set the high-level period of the SCL clock.
RIIC0.ICBRL register
BRL[4:0] bits ← 29: Set the low-level period of the SCL clock.
Set the I^{2}C bus mode
RIIC0.ICMR3 register
ACKWP bit ← 1: Modification of the ACKBT bit is enabled.
Set the I^{2}C bus function
RIIC0.ICFER register
MALE bit ← 1: Master arbitration-lost detection is enabled.
Enable or disable interrupts
RIIC0.ICIER register
TIE bit ← 1: Transmit data empty interrupt request (ICTXI) is enabled.
RIE bit ← 1: Receive data full interrupt request (ICRXI) is enabled.
NAKIE bit ← 1: NACK reception interrupt request (NAKI) is enabled.
SPIE bit ← 1: Stop condition detection interrupt request (SPI) is enabled.
ALIE bit ← 1: Arbitration-lost interrupt request (ALI) is enabled.
Clear the internal reset
RIIC0.ICCR1 register
IICRST bit ← 0: Clear the RIIC reset or internal reset.
Set port modes
PORT1.PMR register
B2 bit ← 1: The P12/SCL pin is used as an I/O port for peripheral functions.
B3 bit ← 1: The P13/SDA pin is used as an I/O port for peripheral functions.
Set the RIIC0 interrupt priority level
IPR246 register
IPR[3:0] bits ← 0100b: The RIIC0.EEI0 interrupt priority level is set to level 4.
IPR247 register
IPR[3:0] bits ← 0100b: The RIIC0.RXI0 interrupt priority level is set to level 4.
IPR248 register
IPR[3:0] bits ← 0100b: The RIIC0.TXI0 interrupt priority level is set to level 4.
IPR249 register
IPR[3:0] bits ← 0100b: The RIIC0.TEI0 interrupt priority level is set to level 4.
Enable the RIIC0 interrupt request
IER1E register
IEN6 bit ← 1: The RIIC0.EEI0 interrupt request is enabled.
IEN7 bit ← 1: The RIIC0.RXI0 interrupt request is enabled.
IER1F register
IEN0 bit ← 1: The RIIC0.TXI0 interrupt request is enabled.
return

Figure 5.12 Enabling the RIIC

5.8.11 Disabling the RIIC

Figure 5.13 shows the Disabling the RIIC.



PORT1.PMR register
B2 bit ← 0: The P12/SCL is used as a general I/O pin.
B3 bit ← 0: The P13/SDA is used as a general I/O pin.

RIIC0.ICCR1 register
IICRST bit ← 0: Clear the RIIC reset or internal reset.

PRCR register ← A502h
PRC1 bit = 1: Enable writing to the related registers.
MSTPCRB register
MSTPB21 bit ← 1: The RIIC0 module is in the module stop state.
PRCR register ← A500h
PRC1 bit = 0: Disable writing to the related registers.

Figure 5.13 Disabling the RIIC

5.8.12 Enabling the RIIC Interrupts

Figure 5.14 shows the Enabling the RIIC Interrupts.

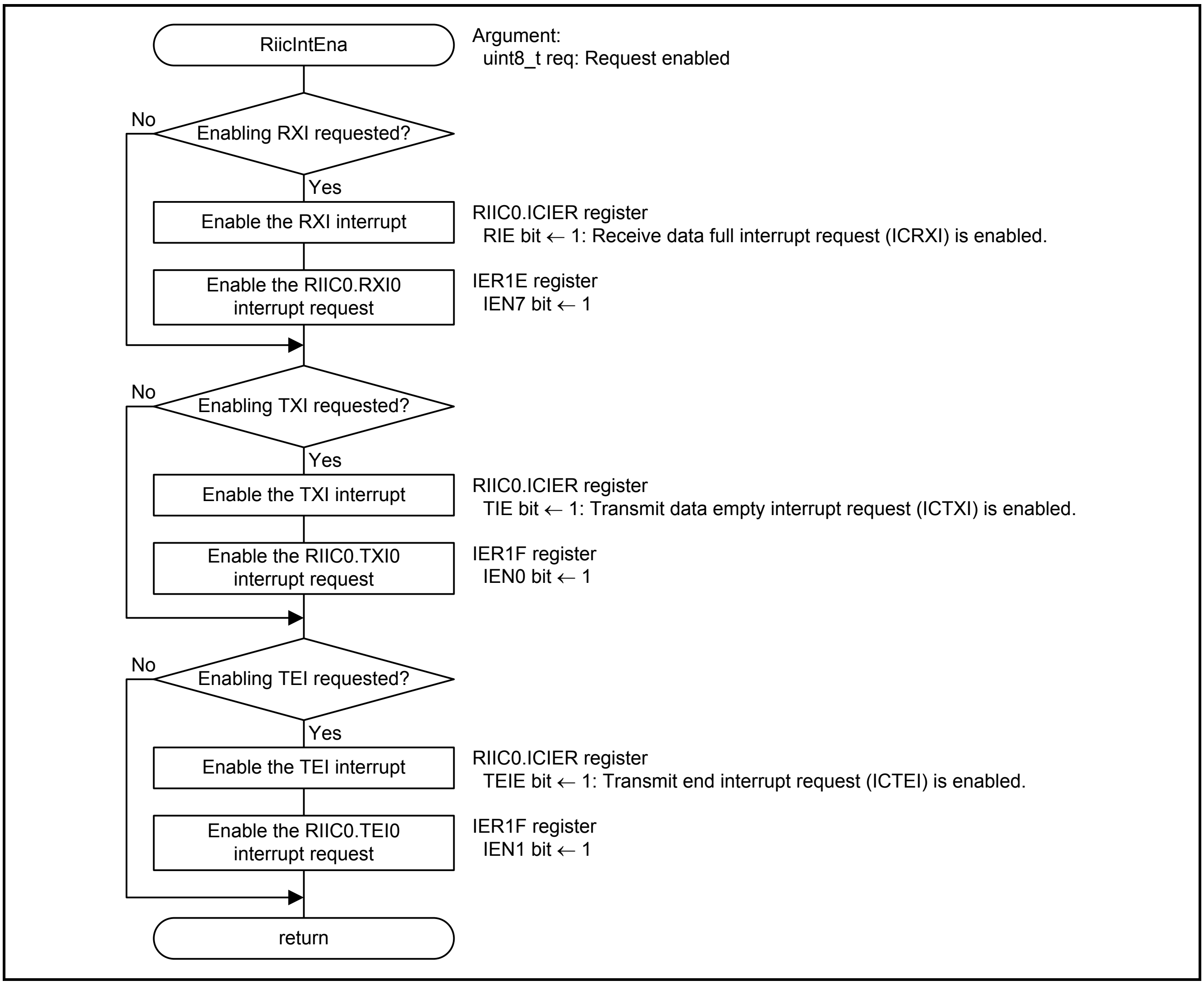

RiicIntEna
Argument:
uint8_t req: Request enabled
No
Enabling RXI requested?
Yes
Enable the RXI interrupt
RIIC0.ICIER register
RIE bit ← 1: Receive data full interrupt request (ICRXI) is enabled.
Enable the RIIC0.RXI0 interrupt request
IER1E register
IEN7 bit ← 1
No
Enabling TXI requested?
Yes
Enable the TXI interrupt
RIIC0.ICIER register
TIE bit ← 1: Transmit data empty interrupt request (ICTXI) is enabled.
Enable the RIIC0.TXI0 interrupt request
IER1F register
IEN0 bit ← 1
No
Enabling TEI requested?
Yes
Enable the TEI interrupt
RIIC0.ICIER register
TEIE bit ← 1: Transmit end interrupt request (ICTEI) is enabled.
Enable the RIIC0.TEI0 interrupt request
IER1F register
IEN1 bit ← 1
return

Figure 5.14 Enabling the RIIC Interrupts

5.8.13 Disabling the RIIC Interrupts

Figure 5.15 shows the Disabling the RIIC Interrupts.

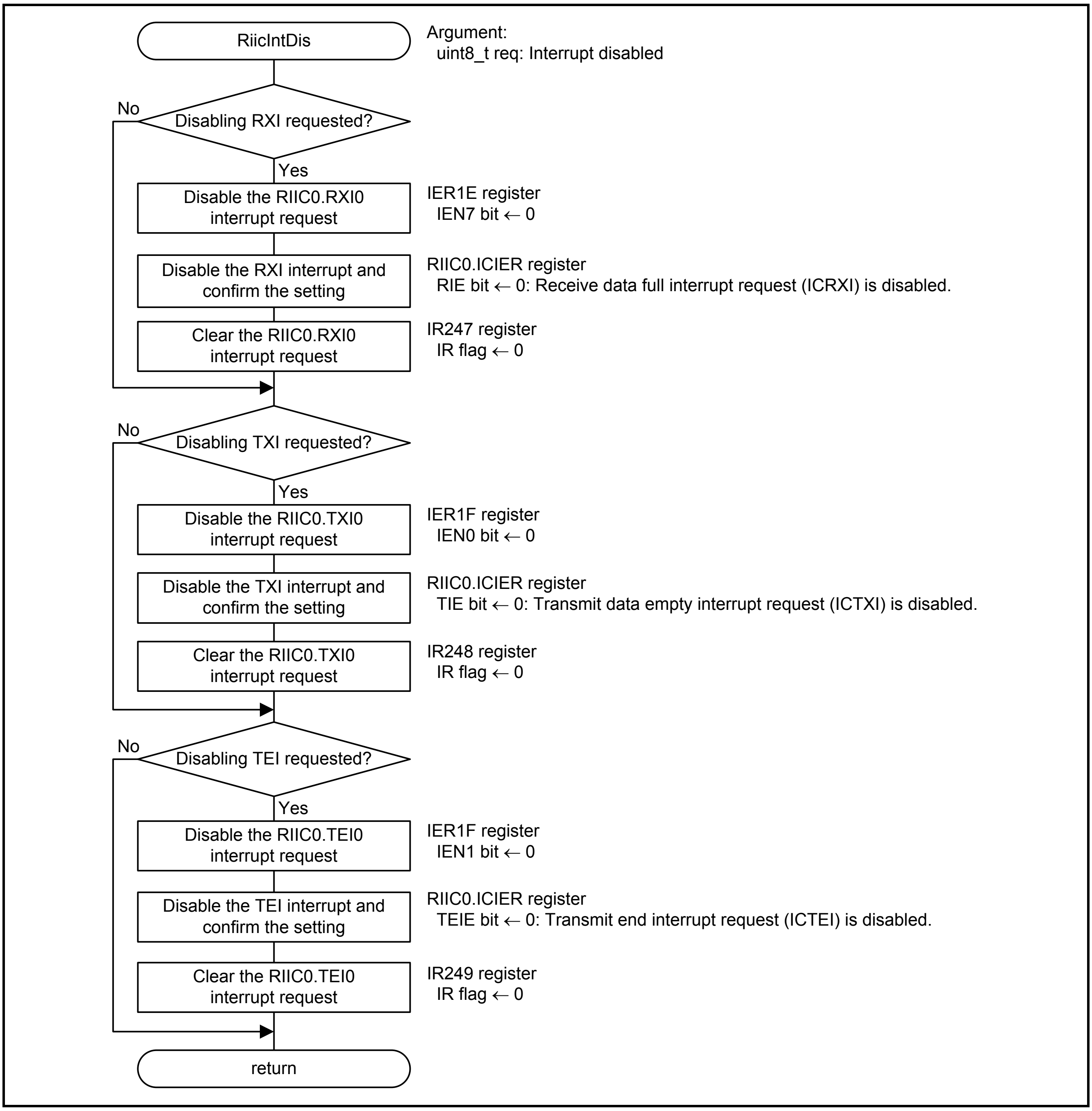

RiicIntDis
Argument:
uint8_t req: Interrupt disabled
Disabling RXI requested?
No
Yes
Disable the RIIC0.RXI0 interrupt request
IER1E register
IEN7 bit ← 0
Disable the RXI interrupt and confirm the setting
RIIC0.ICIER register
RIE bit ← 0: Receive data full interrupt request (ICRXI) is disabled.
Clear the RIIC0.RXI0 interrupt request
IR247 register
IR flag ← 0
Disabling TXI requested?
No
Yes
Disable the RIIC0.TXI0 interrupt request
IER1F register
IEN0 bit ← 0
Disable the TXI interrupt and confirm the setting
RIIC0.ICIER register
TIE bit ← 0: Transmit data empty interrupt request (ICTXI) is disabled.
Clear the RIIC0.TXI0 interrupt request
IR248 register
IR flag ← 0
Disabling TEI requested?
No
Yes
Disable the RIIC0.TEI0 interrupt request
IER1F register
IEN1 bit ← 0
Disable the TEI interrupt and confirm the setting
RIIC0.ICIER register
TEIE bit ← 0: Transmit end interrupt request (ICTEI) is disabled.
Clear the RIIC0.TEI0 interrupt request
IR249 register
IR flag ← 0
return

Figure 5.15 Disabling the RIIC Interrupts

5.8.14 Receive Data Full Interrupt

Figure 5.16 and Figure 5.17 show Receive Data Full Interrupt.

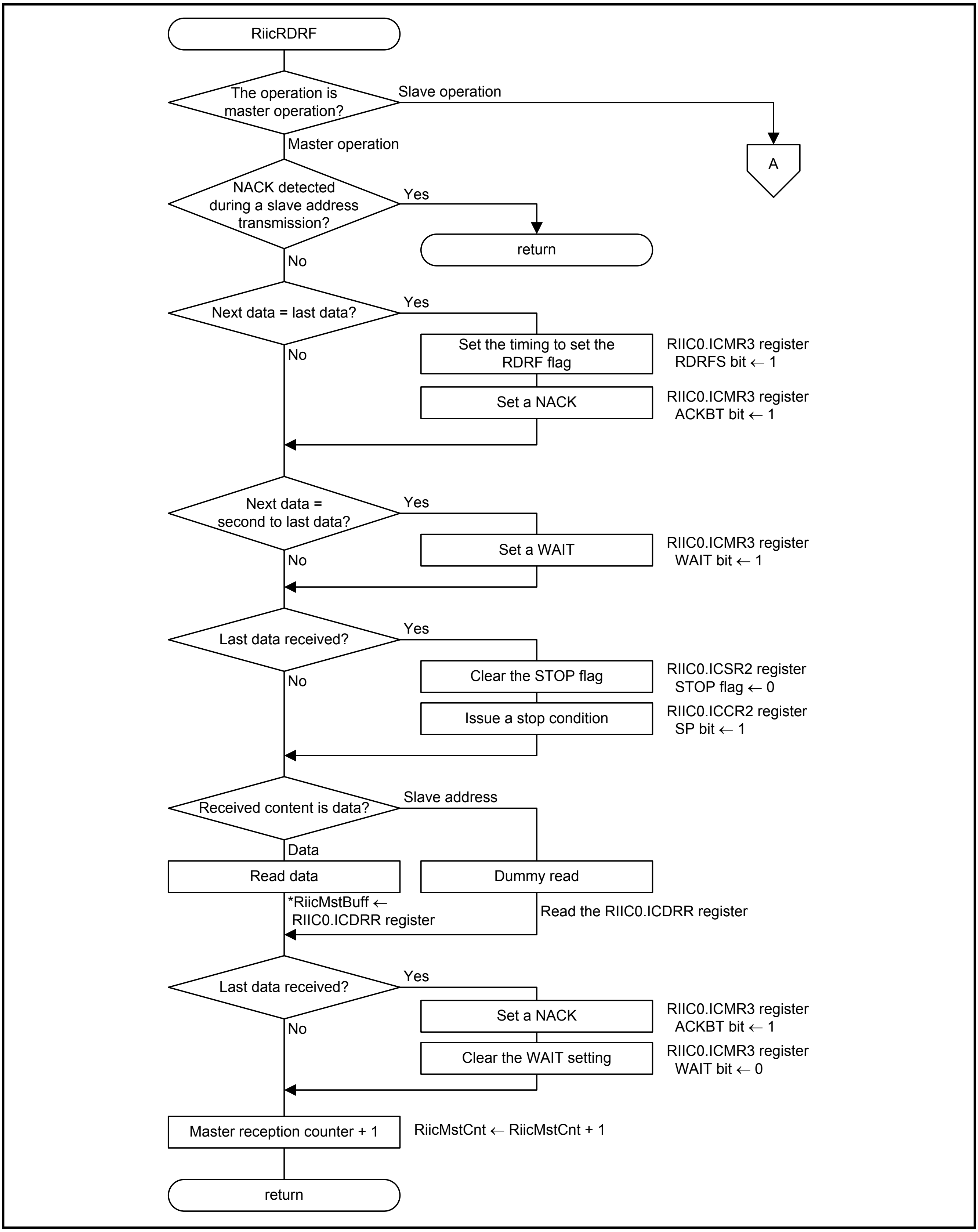

RiicRDRF
The operation is master operation?
Slave operation
A
Master operation
NACK detected during a slave address transmission?
Yes
return
No
Next data = last data?
Yes
Set the timing to set the RDRF flag
RIIC0.ICMR3 register RDRFS bit ← 1
Set a NACK
RIIC0.ICMR3 register ACKBT bit ← 1
No
Next data = second to last data?
Yes
Set a WAIT
RIIC0.ICMR3 register WAIT bit ← 1
No
Last data received?
Yes
Clear the STOP flag
RIIC0.ICSR2 register STOP flag ← 0
Issue a stop condition
RIIC0.ICCR2 register SP bit ← 1
No
Received content is data?
Slave address
Data
Read data
Dummy read
*RiicMstBuff ← RIIC0.ICDRR register
Read the RIIC0.ICDRR register
Last data received?
Yes
Set a NACK
RIIC0.ICMR3 register ACKBT bit ← 1
Clear the WAIT setting
RIIC0.ICMR3 register WAIT bit ← 0
No
Master reception counter + 1
RiicMstCnt ← RiicMstCnt + 1
return

Figure 5.16 Receive Data Full Interrupt (1/2)

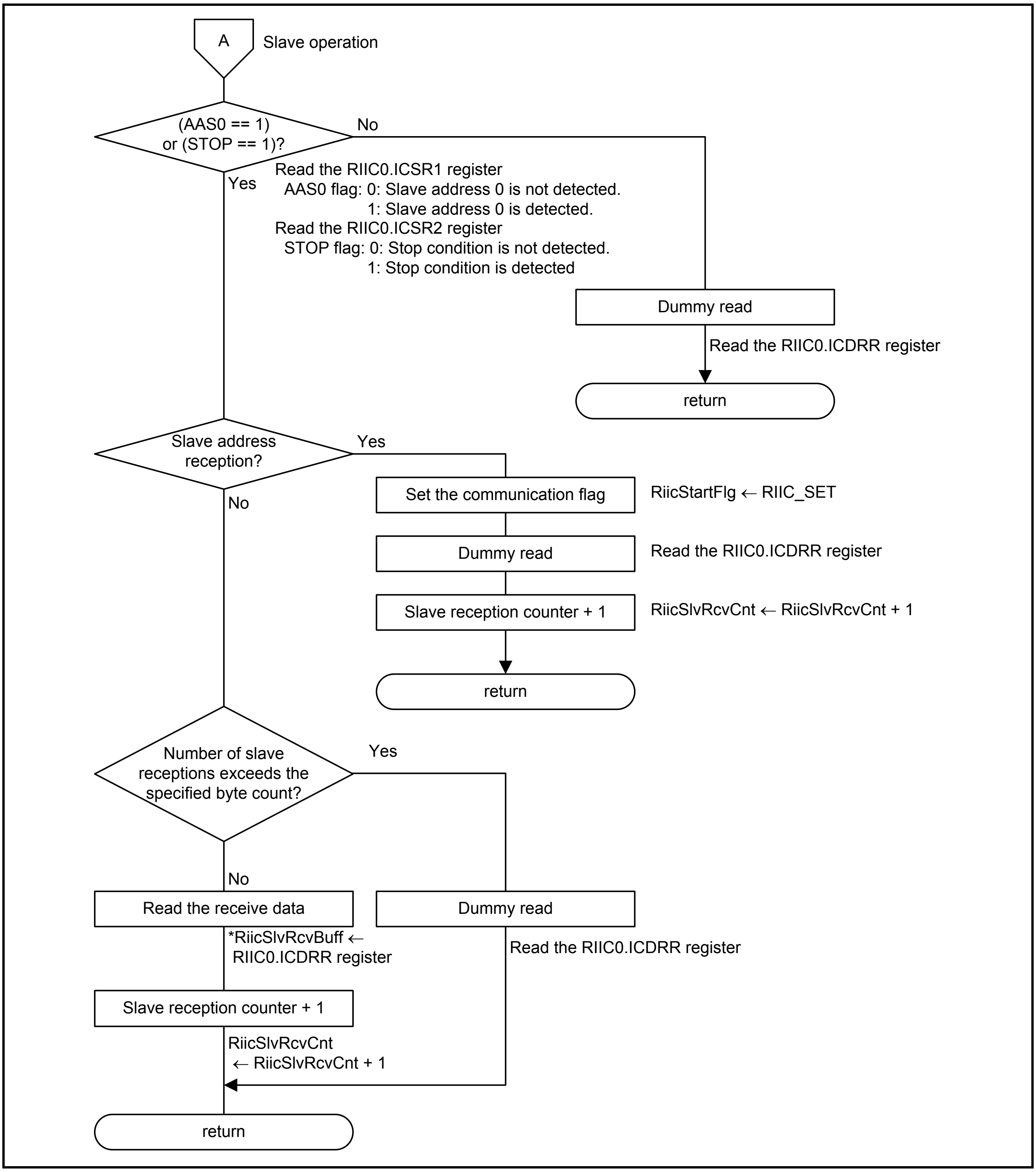

A
Slave operation
(AAS0 == 1)
or (STOP == 1)?
No
Yes
Read the RIIC0.ICSR1 register
AAS0 flag: 0: Slave address 0 is not detected.
1: Slave address 0 is detected.
Read the RIIC0.ICSR2 register
STOP flag: 0: Stop condition is not detected.
1: Stop condition is detected
Dummy read
Read the RIIC0.ICDRR register
return
Slave address
reception?
Yes
No
Set the communication flag
RiicStartFlg ← RIIC_SET
Dummy read
Read the RIIC0.ICDRR register
Slave reception counter + 1
RiicSlvRcvCnt ← RiicSlvRcvCnt + 1
return
Number of slave
receptions exceeds the
specified byte count?
Yes
No
Read the receive data
*RiicSlvRcvBuff ←
RIIC0.ICDRR register
Dummy read
Read the RIIC0.ICDRR register
Slave reception counter + 1
RiicSlvRcvCnt
← RiicSlvRcvCnt + 1
return

Figure 5.17 Receive Data Full Interrupt (2/2)

5.8.15 Transmit Data Empty Interrupt

Figure 5.18 shows the Transmit Data Empty Interrupt.

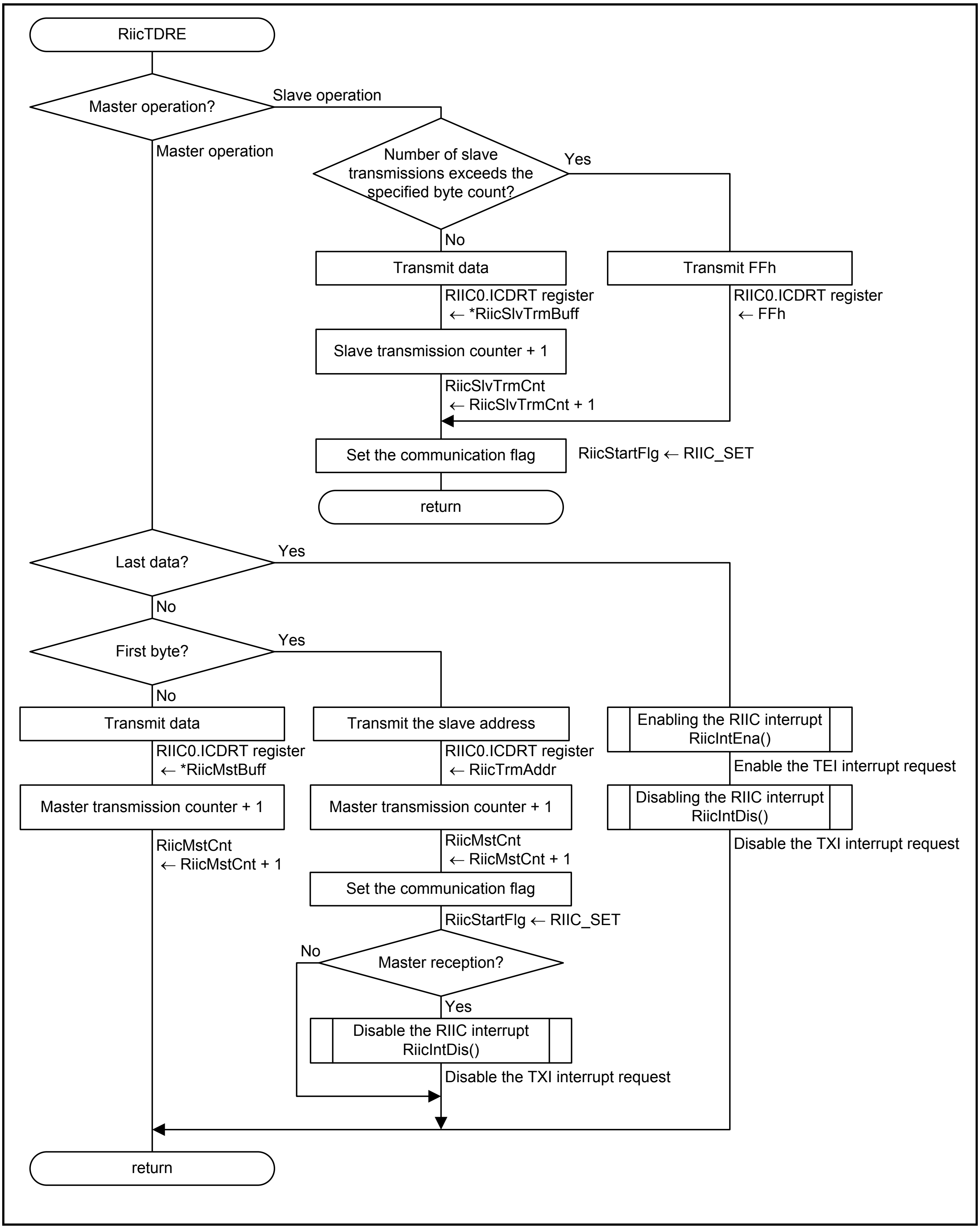

RiicTDRE
Master operation?
Slave operation
Master operation
Number of slave transmissions exceeds the specified byte count?
Yes
No
Transmit data
RIIC0.ICDRT register ← *RiicSlvTrmBuff
Transmit FFh
RIIC0.ICDRT register ← FFh
Slave transmission counter + 1
RiicSlvTrmCnt ← RiicSlvTrmCnt + 1
Set the communication flag
RiicStartFlg ← RIIC_SET
return
Last data?
Yes
No
First byte?
Yes
No
Transmit data
RIIC0.ICDRT register ← *RiicMstBuff
Master transmission counter + 1
RiicMstCnt ← RiicMstCnt + 1
Transmit the slave address
RIIC0.ICDRT register ← RiicTrmAddr
Master transmission counter + 1
RiicMstCnt ← RiicMstCnt + 1
Set the communication flag
RiicStartFlg ← RIIC_SET
Master reception?
No
Yes
Disable the RIIC interrupt RiicIntDis()
Disable the TXI interrupt request
Enabling the RIIC interrupt RiicIntEna()
Enable the TEI interrupt request
Disabling the RIIC interrupt RiicIntDis()
Disable the TXI interrupt request
return

Figure 5.18 Transmit Data Empty Interrupt

5.8.16 Transmit End Interrupt

Figure 5.19 shows the Transmit End Interrupt.

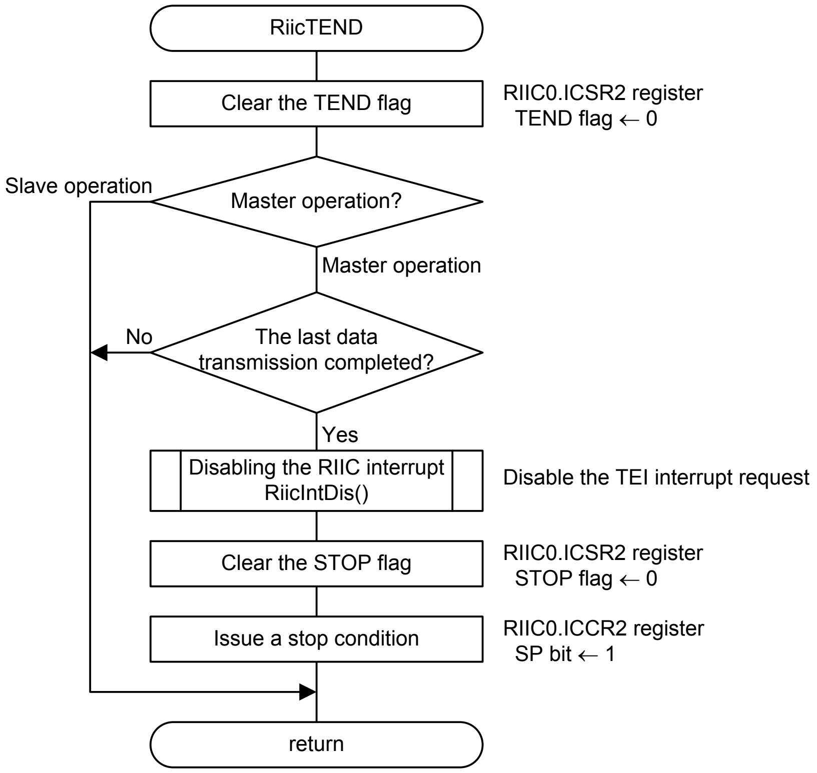


Figure 5.19 Transmit End Interrupt

5.8.17 Stop Condition Detection Interrupt

Figure 5.20 shows the Stop Condition Detection Interrupt.

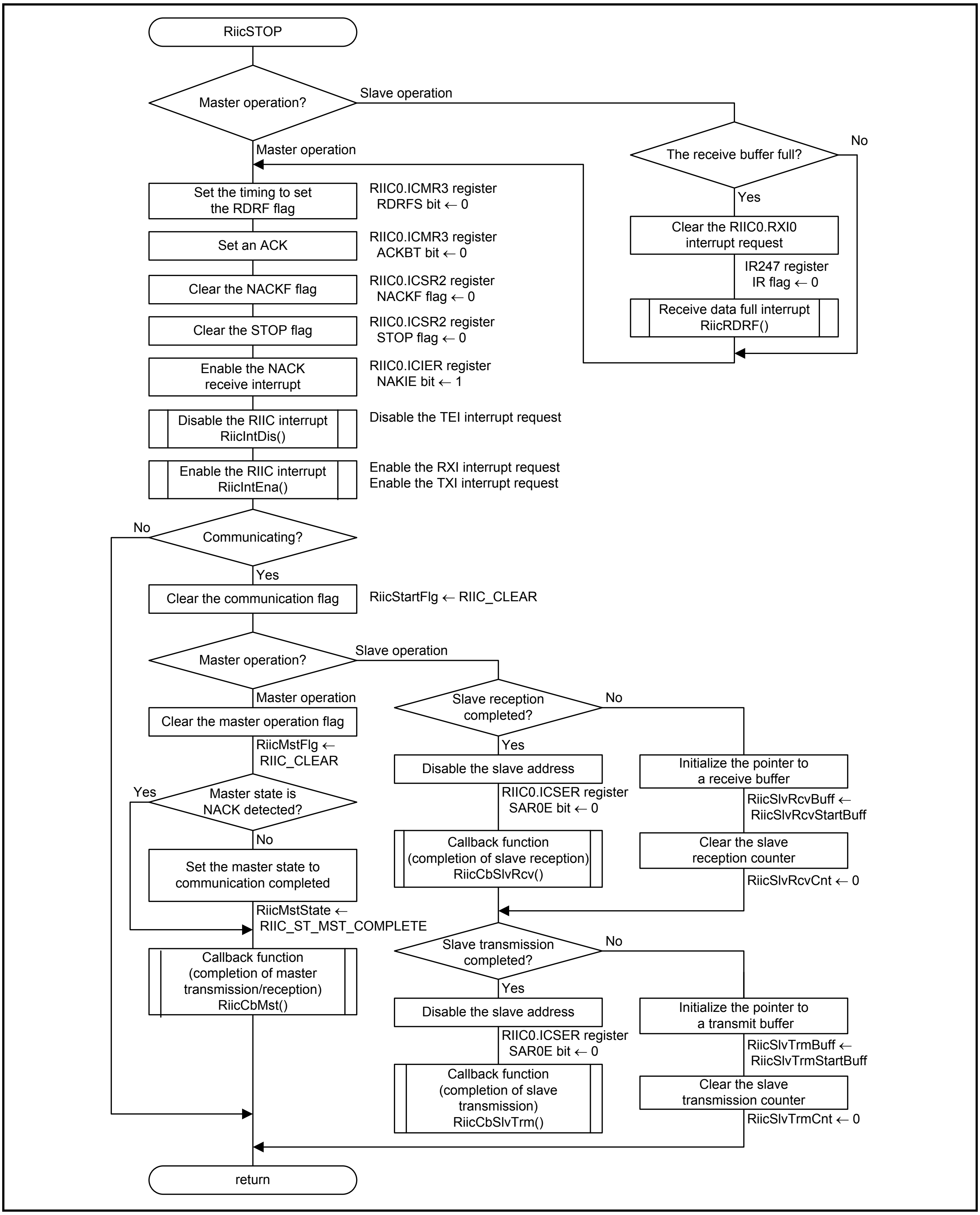

RiicSTOP
Master operation?
Slave operation
Master operation
The receive buffer full?
No
Yes
Set the timing to set the RDRF flag
RIIC0.ICMR3 register RDRFS bit ← 0
Set an ACK
RIIC0.ICMR3 register ACKBT bit ← 0
Clear the NACKF flag
RIIC0.ICSR2 register NACKF flag ← 0
Clear the STOP flag
RIIC0.ICSR2 register STOP flag ← 0
Enable the NACK receive interrupt
RIIC0.ICIER register NAKIE bit ← 1
Clear the RIIC0.RXI0 interrupt request
IR247 register IR flag ← 0
Receive data full interrupt RiicRDRF()
Disable the RIIC interrupt RiicIntDis()
Disable the TEI interrupt request
Enable the RIIC interrupt RiicIntEna()
Enable the RXI interrupt request
Enable the TXI interrupt request
Communicating?
No
Yes
Clear the communication flag
RiicStartFlg ← RIIC_CLEAR
Master operation?
Slave operation
Master operation
Clear the master operation flag
RiicMstFlg ← RIIC_CLEAR
Master state is NACK detected?
Yes
No
Set the master state to communication completed
RiicMstState ← RIIC_ST_MST_COMPLETE
Callback function (completion of master transmission/reception) RiicCbMst()
Slave reception completed?
No
Yes
Disable the slave address
RIIC0.ICSER register SAR0E bit ← 0
Callback function (completion of slave reception) RiicCbSlvRcv()
Initialize the pointer to a receive buffer
RiicSlvRcvBuff ← RiicSlvRcvStartBuff
Clear the slave reception counter
RiicSlvRcvCnt ← 0
Slave transmission completed?
No
Yes
Disable the slave address
RIIC0.ICSER register SAR0E bit ← 0
Callback function (completion of slave transmission) RiicCbSlvTrm()
Initialize the pointer to a transmit buffer
RiicSlvTrmBuff ← RiicSlvTrmStartBuff
Clear the slave transmission counter
RiicSlvTrmCnt ← 0
return

Figure 5.20 Stop Condition Detection Interrupt

5.8.18 NACK Detection Interrupt

Figure 5.21 shows the NACK Detection Interrupt.

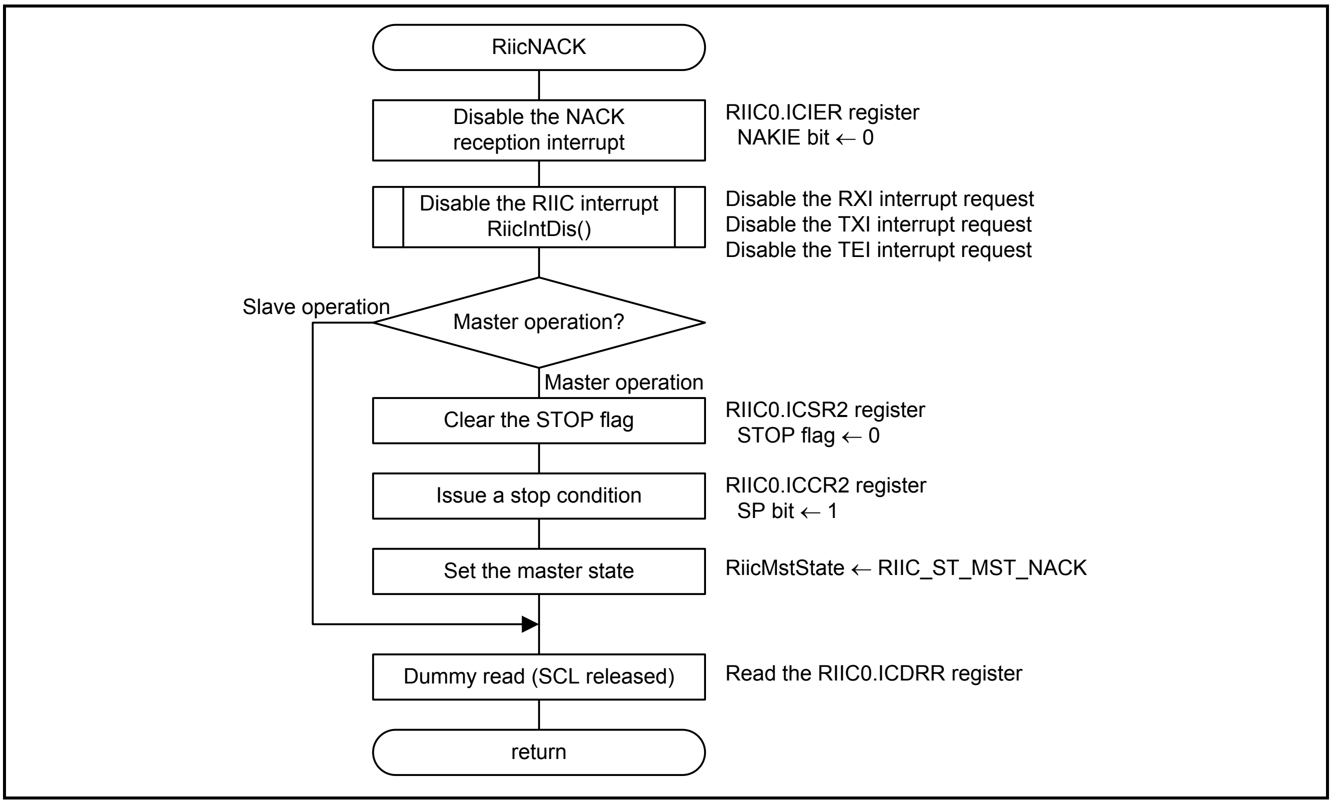

RiicNACK
Disable the NACK reception interrupt
RIIC0.ICIER register NAKIE bit ← 0
Disable the RIIC interrupt RiicIntDis()
Disable the RXI interrupt request
Disable the TXI interrupt request
Disable the TEI interrupt request
Slave operation
Master operation?
Master operation
Clear the STOP flag
RIIC0.ICSR2 register STOP flag ← 0
Issue a stop condition
RIIC0.ICCR2 register SP bit ← 1
Set the master state
RiicMstState ← RIIC_ST_MST_NACK
Dummy read (SCL released)
Read the RIIC0.ICDRR register
return

Figure 5.21 NACK Detection Interrupt

5.8.19 Arbitration-Lost Detection Interrupt

Figure 5.22 shows the Arbitration-Lost Detection Interrupt.

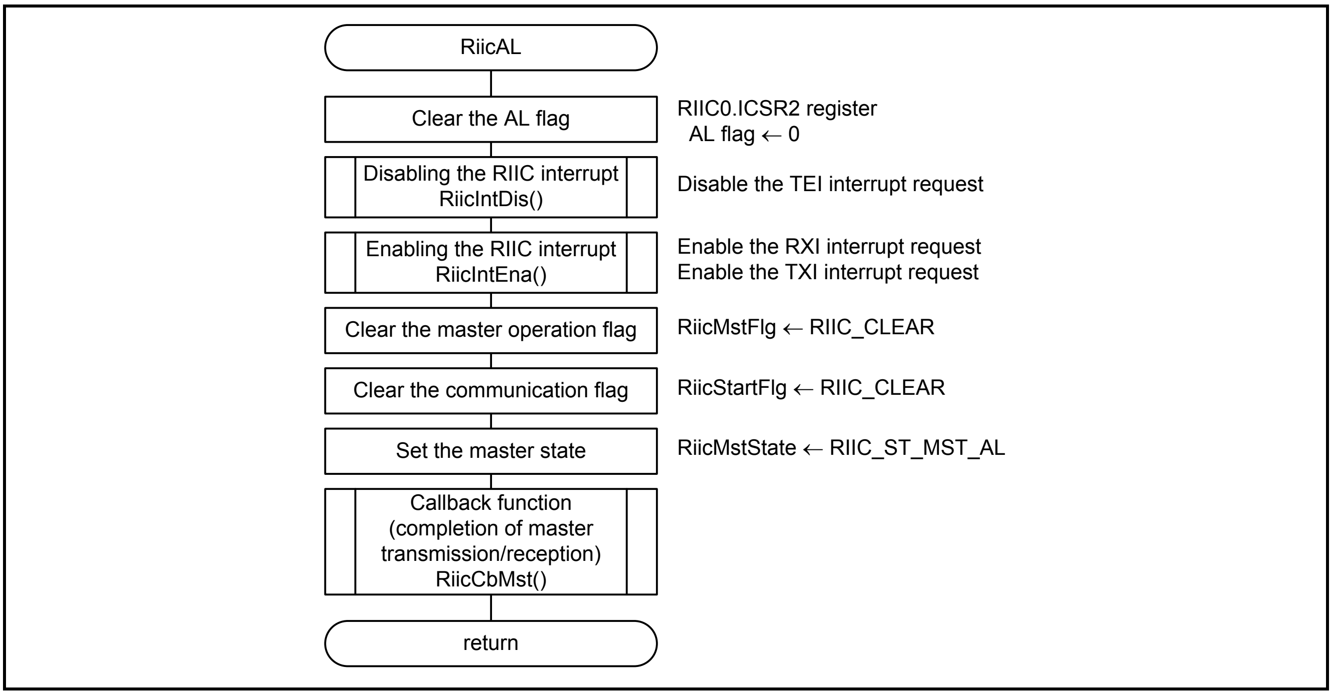
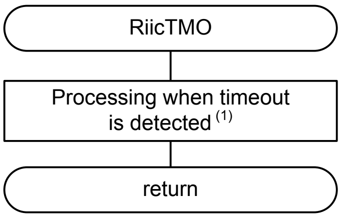

RiicAL
Clear the AL flag
RIIC0.ICSR2 register
AL flag ← 0
Disabling the RIIC interrupt
RiicIntDis()
Disable the TEI interrupt request
Enabling the RIIC interrupt
RiicIntEna()
Enable the RXI interrupt request
Enable the TXI interrupt request
Clear the master operation flag
RiicMstFlg ← RIIC_CLEAR
Clear the communication flag
RiicStartFlg ← RIIC_CLEAR
Set the master state
RiicMstState ← RIIC_ST_MST_AL
Callback function (completion of master transmission/reception)
RiicCbMst()
return

Figure 5.22 Arbitration-Lost Detection Interrupt

5.8.20 Timeout Detection Interrupt

Figure 5.23 shows the Timeout Detection Interrupt.



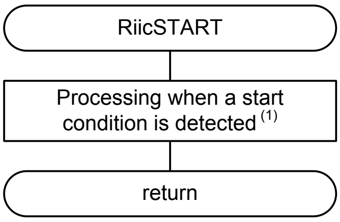
Note:

1. The processing of this function is not included in the sample code. Add a program as required.

Figure 5.23 Timeout Detection Interrupt

5.8.21 Start Condition Detection Interrupt

Figure 5.24 shows the Start Condition Detection Interrupt.



Note:

1. The processing of this function is not included in the sample code. Add a program as required.

Figure 5.24 Start Condition Detection Interrupt

5.8.22 RIIC0.EEI0 Interrupt Handling

Figure 5.25 shows the RIIC0.EEI0 Interrupt Handling.

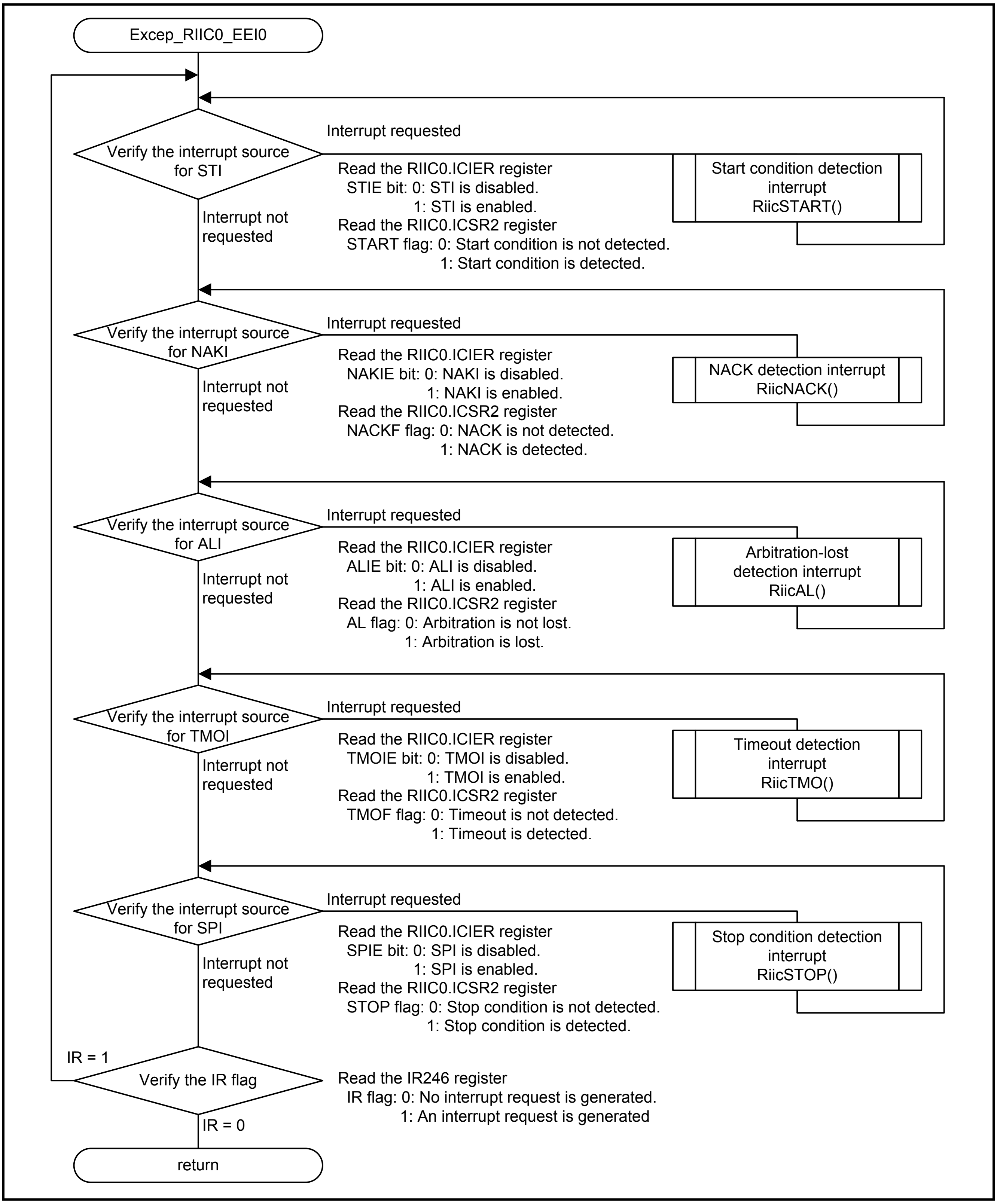

Excep_RIIC0_EEI0
Verify the interrupt source for STI
Interrupt requested
Interrupt not requested
Read the RIIC0.ICIER register
STIE bit: 0: STI is disabled.
1: STI is enabled.
Read the RIIC0.ICSR2 register
START flag: 0: Start condition is not detected.
1: Start condition is detected.
Start condition detection interrupt RiicSTART()
Verify the interrupt source for NAKI
Interrupt requested
Interrupt not requested
Read the RIIC0.ICIER register
NAKIE bit: 0: NAKI is disabled.
1: NAKI is enabled.
Read the RIIC0.ICSR2 register
NACKF flag: 0: NACK is not detected.
1: NACK is detected.
NACK detection interrupt RiicNACK()
Verify the interrupt source for ALI
Interrupt requested
Interrupt not requested
Read the RIIC0.ICIER register
ALIE bit: 0: ALI is disabled.
1: ALI is enabled.
Read the RIIC0.ICSR2 register
AL flag: 0: Arbitration is not lost.
1: Arbitration is lost.
Arbitration-lost detection interrupt RiicAL()
Verify the interrupt source for TMOI
Interrupt requested
Interrupt not requested
Read the RIIC0.ICIER register
TMOIE bit: 0: TMOI is disabled.
1: TMOI is enabled.
Read the RIIC0.ICSR2 register
TMOF flag: 0: Timeout is not detected.
1: Timeout is detected.
Timeout detection interrupt RiicTMO()
Verify the interrupt source for SPI
Interrupt requested
Interrupt not requested
Read the RIIC0.ICIER register
SPIE bit: 0: SPI is disabled.
1: SPI is enabled.
Read the RIIC0.ICSR2 register
STOP flag: 0: Stop condition is not detected.
1: Stop condition is detected.
Stop condition detection interrupt RiicSTOP()
IR = 1
Verify the IR flag
Read the IR246 register
IR flag: 0: No interrupt request is generated.
1: An interrupt request is generated
IR = 0
return

Figure 5.25 RIIC0.EEI0 Interrupt Handling

5.8.23 RIIC0.RXI0 Interrupt Handling

Figure 5.26 shows the RIIC0.RXI0 Interrupt Handling.

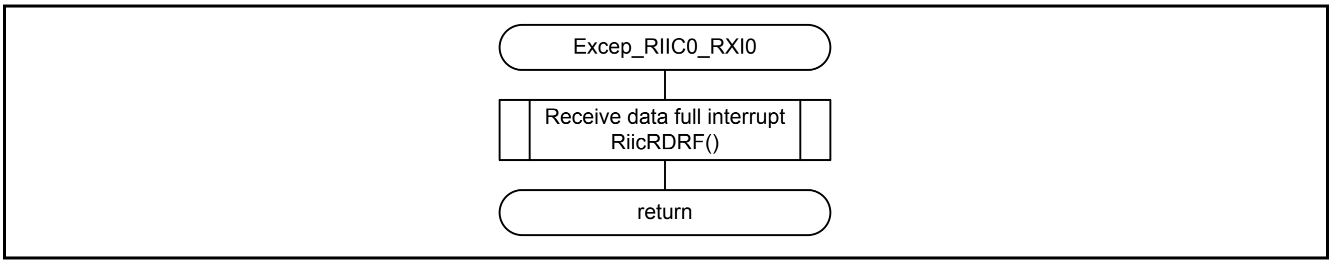


Figure 5.26 RIIC0.RXI0 Interrupt Handling

5.8.24 RIIC0.TXI0 Interrupt Handling

Figure 5.27 shows the RIIC0.TXI0 Interrupt Handling.

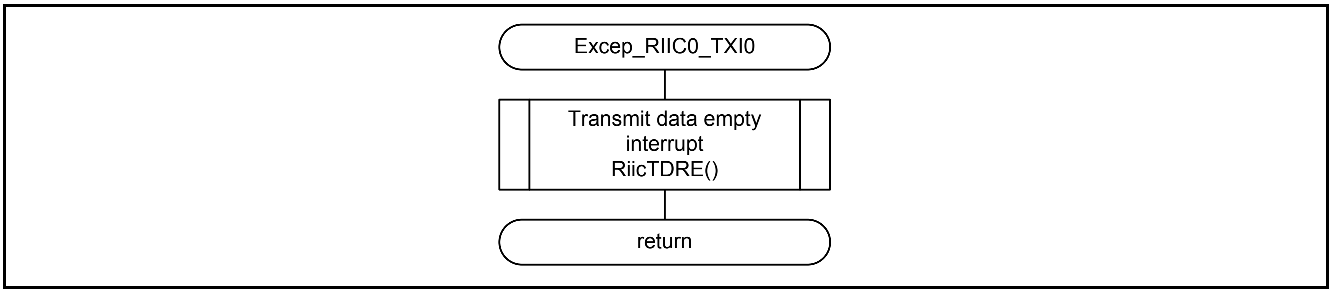


Figure 5.27 RIIC0.TXI0 Interrupt Handling

5.8.25 RIIC0.TEI0 Interrupt Handling

Figure 5.28 shows the RIIC0.TEI0 Interrupt Handling.

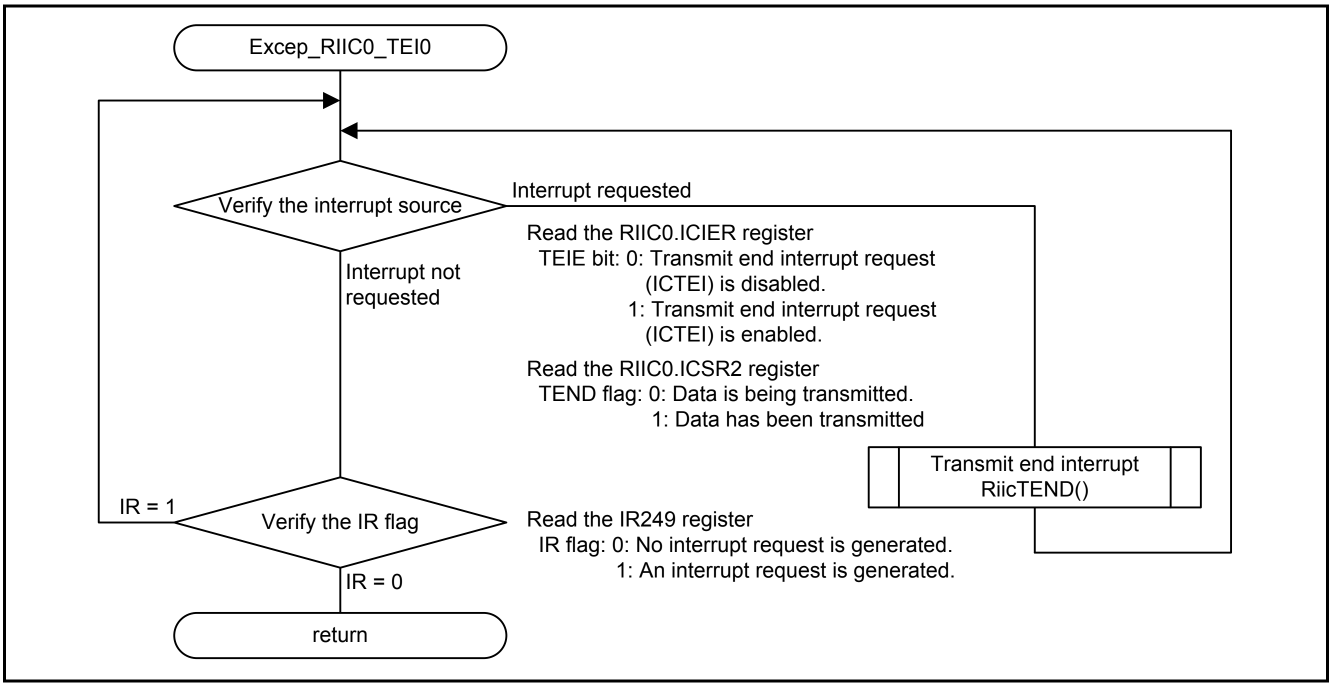

Excep_RIIC0_TEI0
Verify the interrupt source
Interrupt requested
Interrupt not requested
Read the RIIC0.ICIER register
TEIE bit: 0: Transmit end interrupt request (ICTEI) is disabled.
1: Transmit end interrupt request (ICTEI) is enabled.
Read the RIIC0.ICSR2 register
TEND flag: 0: Data is being transmitted.
1: Data has been transmitted
Transmit end interrupt
RiicTEND()
IR = 1
Verify the IR flag
Read the IR249 register
IR flag: 0: No interrupt request is generated.
1: An interrupt request is generated.
IR = 0
return

Figure 5.28 RIIC0.TEI0 Interrupt Handling

6. Applying This Application Note to the RX21A or RX220 Group

The sample code accompanying this application note has been confirmed to operate with the RX210 Group. To make the sample code operate with the RX21A or RX220 Group, use this application note in conjunction with the Initial Setting application note for each group.

For details on using this application note with the RX21A and RX220 Groups, refer to “5. Applying the RX210 Group Application Note to the RX21A Group” in the RX21A Group Initial Setting application note, and “4. Applying the RX210 Group Application Note to the RX220 Group” in the RX220 Group Initial Setting application note.

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

User’s Manual: Hardware

RX210 Group User’s Manual: Hardware Rev.1.50 (R01UH0037EJ)
RX21A Group User’s Manual: Hardware Rev.1.00 (R01UH0251EJ)
RX220 Group User’s Manual: Hardware Rev.1.10 (R01UH0292EJ)
The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User’s Manual: Development Tools

RX Family C/C++ Compiler Package V.1.01 User’s Manual Rev.1.00 (R20UT0570EJ)
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	RX210, RX21A, and RX220 Groups Application Note Multi-Master I2C Bus Using RIIC

Rev.	Date	Description	
		Page	Summary
1.00	Apr. 1, 2013	—	First edition issued
1.01	July 1, 2014	1	Products: Added the RX21A and RX220 Groups.
		5	3. Reference Application Notes: Added the Initial Setting application notes for the RX21A and RX220 Groups.
		16, 17	Modified the description of reference application note in the following functions: R_INIT_StopModule, R_INIT_NonExistentPort, and R_INIT_Clock.
		42	6. Applying This Application Note to the RX21A or RX220 Group: Added.
		43	8. Reference Documents: Added the User’s Manual: Hardware for the RX21A and RX220 Groups.

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

   Handle unused pins in 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 LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. 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.

RENESAS

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, 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: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

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: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

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.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0