RL78/G23

I2C Supporting Multiple Slave Address (Master)

Introduction

This application note describes how to use the master function of the I2C bus by using the IICA serial interface. In the procedure described, you will operate four serial memory areas (256 bytes x 4) specified by different slave addresses.

This application note is associated with the application note of "I2C Supporting Multiple Slave Addresses (Slave)".

Target Device

RL78/G23

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

Contents

1. Specifications

1.1 Basic Specifications of the I2C Bus (Master)

The following lists the basic specifications of the I2C bus.

・I2C bus to be connected: Fast mode (up to 400 kbps)

Target slave (4 serial memories of 256 bytes)

・Slave address 1: 0010000B (First serial memory)

・Slave address 2: 0100100B (Second serial memory)

・Slave address 3: 1011010B (Third serial memory)

・Slave address 4: 1101011B (Fourth serial memory)

Note: In the RL78 family, the own address (7 bits) is expressed by the upper 7 bits of the SVA0 register.

The least significant bit (LSB) of the SVA0 register is always 0.

For address transmission, the slave address together with the transfer direction (R/W) is written as 8-bit data to IICA shift register 0 (IICA0).

Table 1-1 lists peripheral functions to be used.

Table 1-1 Peripheral Function and Use

Peripheral Function	Use
IICA0	Master function of the I2C bus
TM01	50 µs interval timer interrupt
TM03	500 ms interval timer interrupt
TM07	10 ms interval timer interrupt
P52	Drives the operation status display LED.
P53	Drives the error display LED.
P137	Input to the operation start switch

The following describes the main settings.

(1) Initial settings for IICA0

Table 1-2 lists Initial Settings for IICA0.

Table 1-2 Initial Settings for IICA0

Register Name	Setting Value	Description
PM6	03H	The pin used for both SCL and SDA signals is set for input during the initial setup period.
P6	00H	The pin used for both SCL and SDA signals is set to 0.
IICCTL01	0DH	Fast mode and digital filter enabled for the IICA0 operating clock f_{CLK}/2
IICWL0	15H	SCLA0 low width
IICWH0	14H	SCLA0 high width
SVA0	10H	Sets the slave address to 10H.
IICF0	03H	In the initial status, the bus is idle and reservation for communication is prohibited.
IICCTL00	8CH	SPD interrupt disabled, wait at the 9th clock cycle, ACK response enabled

(2) Initial settings for TM0

Table 1-3 lists the initial settings for TM01.

Table 1-3 Initial Settings for TM01

Register Name	Setting Value	Description
TPS0	0018H	CK00: 125 kHz, CK01/CK02/CK03: 32 MHz
TT0	0002H	Channel 1 disabled
TMPR001	0	Interrupt priority: Level 2
TMR01	8000H	TM01 count clock: CK01 (32 MHz)
TDR01	063FH	Interval time: 50 µs
TOM0	00H	TM01: Master mode
TOL0	00H	TM01 output: Positive logic
TO0	00H	TM01 output: 0
TOE0	00H	TM01 output: Disabled

(3) Initial settings for TM03

Table 1-3 lists the initial settings for TM03.

Table 1-4 Initial Settings for TM03

Register Name	Setting Value	Description
TPS0	0018H	CK00: 125 kHz, CK01/CK02/CK03: 32 MHz
TT0	0008H	Channel 3 disabled
TMR03	8000H	TM01 count clock: CK00 (125 kHz)
TDR03	F423H	Interval time: 500 ms
TOM0	00H	TM03: Master mode
TOL0	00H	TM03 output: Positive logic
TO0	00H	TM03 output: 0
TOE0	00H	TM03 output: Disabled

(4) Initial settings for TM07

Table 1-3 lists the initial settings for TM07.

Table 1-5 Initial Settings for TM07

Register Name	Setting Value	Description
TPS0	0008H	CK00: 125 kHz, CK01/CK02/CK03: 32 MHz
TT0	0080H	Channel 7 disabled
TMPR007	0	Interrupt priority: Level 2
TMR07	0000H	TM07 count clock: CK00 (125 kHz)
TDR07	04E1H	Interval time: 10 ms
TOM0	00H	TM07: Master mode
TOL0	00H	TM07 output: Positive logic
TO0	00H	TM07 output: 0
TOE0	00H	TM07 output: Disabled

1.2 Outline of Operation

The I2C bus sends data to, or receives data from, four serial memory areas, each of which has a 1-byte register address that indicates a slave internal address.

The I2C bus performs the following operations sequentially in the following order for the four serial memory areas:

- Reads 256 bytes sequentially.

- Checks whether the data that was read is equal to the expected value (for the second and the following cycles).

- Writes data (0x00 to 0xFF) in units of 16 bytes.

- Reads 16-byte data 16 times.

- Checks whether the data that was read is equal to the expected value.

- Writes data (0xFF to 0x00) in units of 256 bytes.

1.3 I2C Bus Control

Table 1-1 shows the relationship between the status of the I2C bus and the value of g_status.

Table 1-1 Relationship Between the Status of the I2C Bus and the Value of g_status

I2C Bus Status	g_status	Remarks
The I2C bus is idle.	0x00	IICBSY0 = 0
The I2C bus is being used by another master.	0x0F	IICBSY0 = 1, MSTS0 = 0
The I2C bus is being used in master operation.	0x20	MSTS0 = 1
The I2C bus is sending a slave address in master operation.	0x18	MSTS0 = 1
The I2C bus is sending a slave register address in master operation.	0x16	MSTS0 = 1
The I2C bus is receiving data in master operation.	0x14	MSTS0 = 1
The I2C bus is sending data in master operation.	0x12	MSTS0 = 1
Data reception ended normally.	0x24	MSTS0 = 1
Data transmission ended normally.	0x22	MSTS0 = 1
Processing of IICA0 is in progress or the I2C bus is being used by another master.	0x8F	
NACK was detected for the slave address.	0x80	
NACK from the slave was detected.	0xC0	

1.4 Library for Communication to the Slave

This application note supports the following functions:

① Function that starts writing data with the specified size (in bytes) to the specified slave

② Function that writes data with the size (in bytes) specified from the specified address of the specified slave

③ Function that starts reading the specified data from the specified slave

④ Function that reads data with the size (in bytes) specified from the specified address of the specified slave

⑤ Function that performs polling to check whether the communication started by function ① or ③ has ended

⑥ Function that waits for completion of the communication started by function ① or ③

⑦ Function that generates a stop condition

1.5 Serial Memory Control

The I2C bus can write data to, or read data from, the address of the serial memory area specified by the combination of the slave address and the next 1-byte data (register address).
Figure 1-1 shows the sequence of writing data in succession to a slave serial memory area by specifying register addresses.
The master first sends a start condition (ST) followed by an 8-bit slave address that consists of a 7-bit slave address and the transfer direction (W). The master then sends a register address that specifies the internal address of the serial memory area. After that, the master sequentially sends pieces of write data. After sending the final data piece, the master generates a stop condition (SP) to complete the communication.

Figure 1-1 Continuous Data Write to Designated Register Addresses

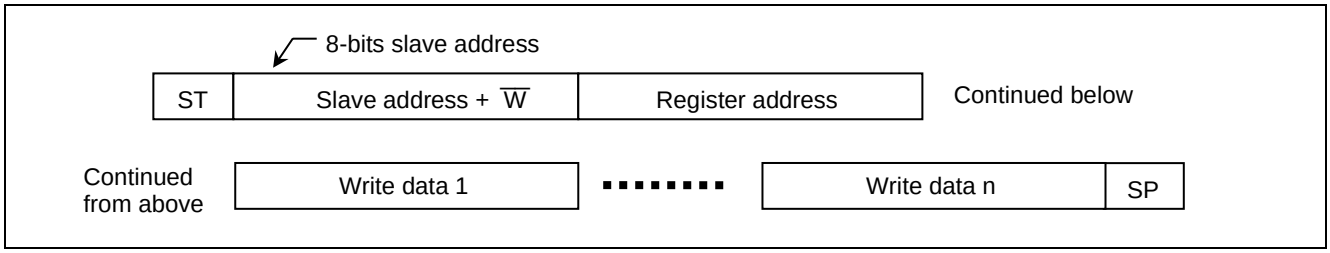
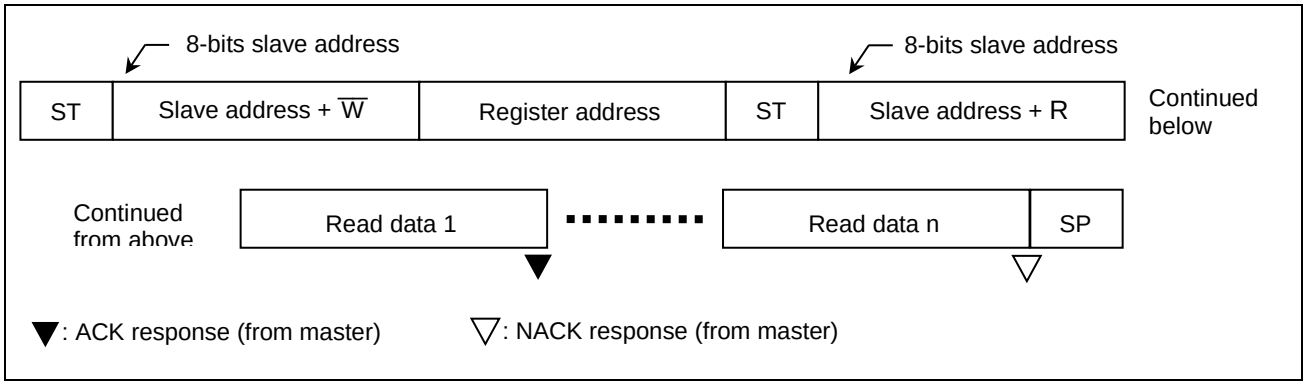


Figure 1-2 shows the sequence of reading data in succession from a slave serial memory area by specifying the register address.
The master first sends a start condition (ST) followed by an 8-bit slave address that consists of a 7-bit slave address and the transfer direction (W). The master then sends a register address that specifies the internal address of the serial memory area. Next, the master sends a restart condition (ST) followed by an 8-bit slave address that consists of a 7-bit slave address and the transfer direction (R). After that, the master sequentially sends pieces of data from the specified register address (sequential reads). When the master returns NACK in response to the received data, the slave stops transmission. Finally, the master generates a stop condition (SP) to complete the communication.

Figure 1-2 Continuous Data Read from Designated Register Addresses



2. Operation Confirmation Conditions

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

Table 2-1 Operation Confirmation Conditions

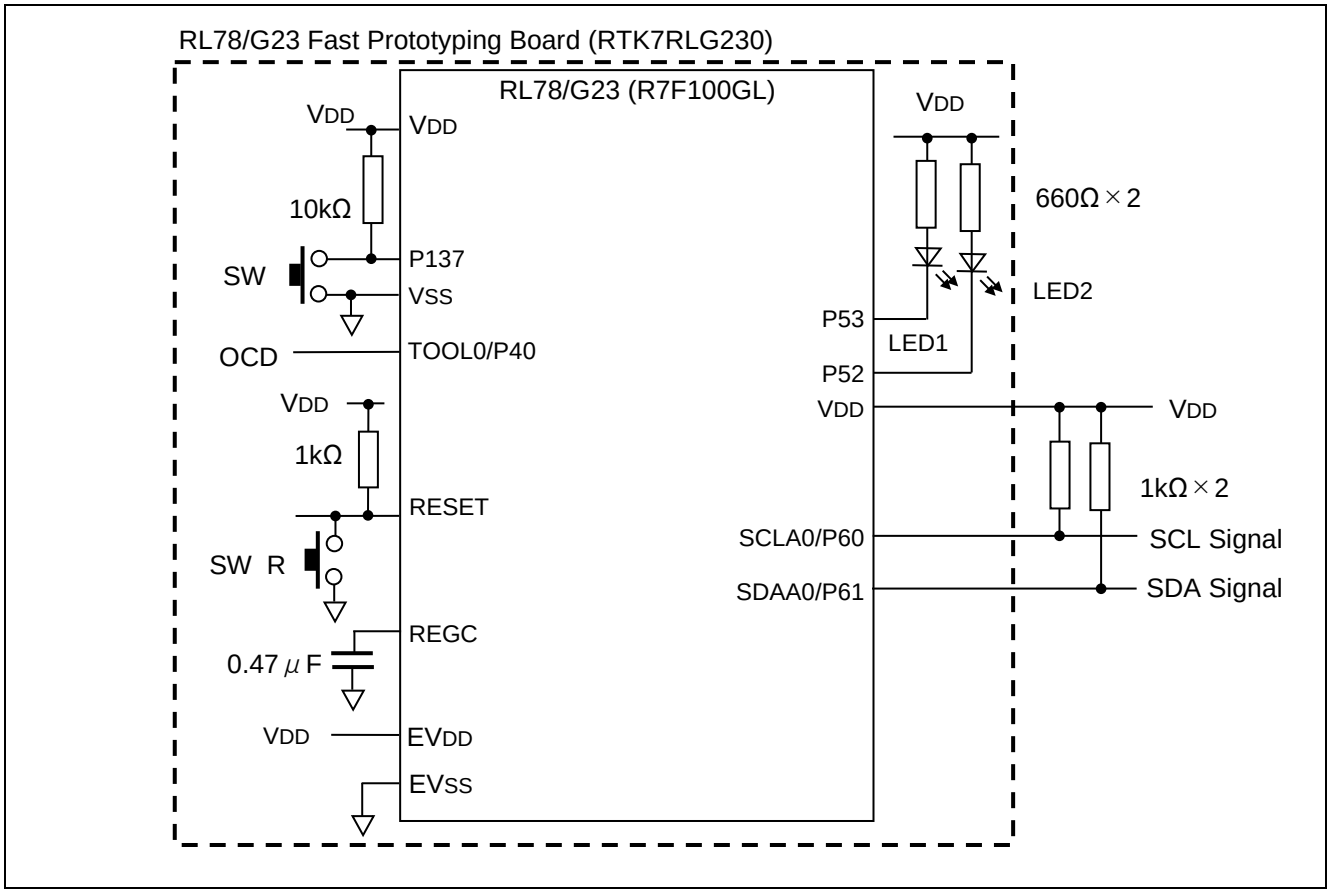
Item	Description
MCU used	RL78/G23 (R7F100GLG)
Board used	RL78/G23 Fast Prototyping Board (RTK7RLG230CLG000BJ)
Operating frequency	High-speed on-chip oscillator clock: 32 MHz CPU/peripheral hardware clock: 32 MHz
Operating voltage	5.0 V (can be operated at 4.0 V to 5.5 V) LVD0 detection voltage: Reset mode At rising edge TYP. 3.96 V (3.84 V to 4.08 V) At falling edge TYP. 3.88 V (3.76 V to 4.00 V)
Integrated development environment (CS+)	CS+ V8.05.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.10.00 from Renesas Electronics Corp.
Integrated development environment (e2studio)	e2 studio V2021-04 (21.4.0) from Renesas Electronics Corp.
C compiler (e2studio)	CC-RL V1.10.00 from Renesas Electronics Corp.
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.21.1 from IAR Systems Corp.
C compiler (IAR)	IAR C/C++ Compiler for Renesas RL78 V4.21.1 from IAR Systems Corp.
Smart configurator (SC)	V1.0.1 from Renesas Electronics Corp.
Board support package (BSP)	V1.00 from Renesas Electronics Corp.

3. Hardware Descriptions

3.1 Example of Hardware Configuration

Figure 3-1 Hardware Configuration shows an example of the hardware configuration used in the application note.

Figure 3-1 Hardware Configuration



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

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

Note 3. Set V_{DD} to a voltage not less than the reset release voltage (V_{LVD0}) set by the LVD.

3.2 List of Pins to be Used

Table 3-1 lists the pins to be used and their functions.

Table 3-1 Pins to be Used and Their Functions

Pin Name	I/O	Function
P60 (D15)	Input / Output	SCL signal
P61 (D14)	Input / Output	SDA signal
P53 (D16)	Output	Drives LED1 (in operation)
P52 (D17)	Output	Drives LED2 (in error)
P137 (D18)	Input	Switch (SW) input

4. Software Explanation

4.1 Setting of Option Byte

Table 4-1 shows the option byte settings.

Table 4-1 Option Byte Settings

Address	Setting Value	Description
000C0H / 010C0H	11101111B	Disables the watchdog timer. (Counting stopped after reset)
000C1H / 010C1H	11111010B	LVD0 detection voltage: reset mode At rising edge TYP. 3.96 V At falling edge TYP. 3.88 V
000C2H / 010C2H	11101000B	HS mode, High-speed on-chip oscillator clock (f_{IH}): 32 MHz
000C3H / 010C3H	10000100B	Enables on-chip debugging

4.2 List of Constants

Table 4-2 lists the constants that are used in the sample code.

Table 4-2 Constants

Constant Name	Setting Value	Description
LED1_pin	P5_bit.no3	Port corresponding to the D16 pin
LED2_pin	P5_bit.no2	Port corresponding to the D17 pin
SW_IN	P13_bit.no7	Port corresponding to the D18 pin. Input pin for the on-board switch.
LED_ON	0	Value of common data for turning the LED on
LED_OFF	1	Value of common data for turning the LED off
INC_DATA[16][16]	0x00 to 0xFF	Write data 1 for RAM areas
RAM1	0010000b	7-bit slave address for RAM1
RAM2	0100100b	7-bit slave address for RAM2
RAM3	1011010b	7-bit slave address for RAM3
RAM4	1101011b	7-bit slave address for RAM4
PAGE_SIZE	256	RAM page size
HIGH	0x01	High level
LOW	0x00	Low level
WAITTIME	1000	Timeout time (number of loops) for detection of a start condition, etc.
BUS_FREE	0x00	The I2C bus is idle.
BUS_BUSY	0x0F	The I2C bus is being used by another master.
BUS_HOLD	0x20	The I2C bus is being used in master operation.
TX_SADDR	0x18	The I2C bus is sending a slave address in master operation.
TX_ADDR_REG	0x16	The I2C bus is sending the value of a register address in master operation.
RX_MODE	0x14	I2C reception is in progress in master operation.
TX_MODE	0x12	I2C transmission is in progress in master operation.
OP_END	0x20	I2C communication was completed in master operation.
RX_END	0x24	I2C reception was completed in master operation.
TX_END	0x22	I2C transmission was completed in master operation.
SUCCESS	0x00	Normal operation
COM_ERROR	0xFF	Error occurred due to a command parameter.
BUS_ERROR	0x8F	Error occurred because the I2C bus was busy.
NO_SLAVE	0x80	NACK was detected for a slave address.
NO_ACK	0xC0	NACK was detected for send data.

4.3 List of Variables

Table 4-3 lists global variables.

Table 4-3 Global Variables

Type	Variable Name	Description	Function Used
uint8_t	g_Tx_buff[]	257-byte send buffer	main()
uint8_t	g_Rx_buff[]	256-byte receive buffer	main()
uint8_t	g_sl_addr_T[]	RAM1, RAM2, RAM3, RAM4	main()
uint8_t	g_status	Current communication status: 0x00: The I2C bus is idle. 0x0F: The I2C bus is being used by another master. 0x12: Sending data. 0x14: Receiving data. 0x16: Sending a register address. 0x18: Sending a slave address. 0x20: Operating as a master. 0x22: Data transmission ended normally. 0x24: Data reception ended normally. 0x8F: Processing of IICA0 is in progress or the bus is busy. 0x80: NACK was detected for a slave address. 0xC0: NACK was detected.	r_IICA0_master_handler(), R_IICA0_check_comstate(), R_IICA0_poll(), R_IICA0_StartCondition(), R_IICA0_bus_check(),
uint8_t *	gp_ rx_address	Pointer to the storage area of received data	r_IICA0_master_handler(), R_IICA0_Master_Receive(), R_IICA0_Rx()
uint16_t	g_ rx_len	Number of bytes to be received	r_IICA0_master_handler(), R_IICA0_Master_Receive(), R_IICA0_Rx()
uint16_t	g_ rx_cnt	Number of received bytes	r_IICA0_master_handler(), R_IICA0_Master_Receive(), R_IICA0_Rx()
uint8_t*	gp _tx_address	Pointer to the data to be sent	r_IICA0_master_handler(), R_IICA0_Master_Send(), R_IICA0_Tx()
uint16_t	g_ tx_cnt	Number of bytes to be sent	r_IICA0_master_handler(), R_IICA0_Master_Send(), R_IICA0_Tx()
uint8_t	g_sl_addr	8-bit slave address	r_IICA0_master_handler(), R_IICA0_Tx(),R_IICA0_Rx()
uint8_t	g_adr_reg	Register address area	r_IICA0_master_handler(), R_IICA0_Tx(),R_IICA0_Rx()
uint8_t	g_adr_flag	Flag indicating whether the register address is enabled: 0x00: Disabled. Data is being sent or received. 0x01: Enabled. Data is being sent. 0x10: Enabled. Data is being received.	r_IICA0_master_handler(), R_IICA0_Tx(),R_IICA0_Rx()
uint8_t	g_SW_state	Switch status	main(), r_Config_TAU0_7_interrupt()

4.4 List of Functions

Table 4-4 shows a list of functions.

Table 4-4 Functions

Function Name	Outline
R_Config_IICA0_Create()	Performs IICA0 initialization
r_Config_IICA0_interrupt()	Performs an IICA0 interrupt
r_IICA0_master_handler()	Performs an IICA0 master interrupt
R_IICA0_StartCondition()	Generates a start condition
R_IICA0_StopCondition()	Generates a stop condition
R_ IICA0_Master_Send()	Starts data transmission to the slave
R_IICA0_Tx()	Sends data to the specified slave address
R_ IICA0_Master_Receive()	Starts data reception from the slave
R_IICA0_Rx()	Receives data from the specified slave address
R_IICA0_poll()	Performs polling for the communication status
wait_time()	Waits for 50 us
R_IICA0_wait_comend()	Waits for completion of communication
R_IICA0_check_comstate()	Checks the IICA0 communication status
R_IICA0_bus_check()	Checks the status of the I2C bus
r_Config_TAU0_7_interrupt()	Performs a 10-ms interval timer interrupt

4.5 Specification of Functions

The function specifications of the sample code are shown below.

R_Config_IICA0_Create()	
Outline	IICA0 initialization
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h
Declaration	void R_ Config_IICA0_Create(void)
Description	Specifies the IICA0 initial settings (fast mode, master).
Argument	None
Return Value	None

r_Config_IICA0_interrupt()	
Outline	IICA0 interrupt
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h
Declaration	static void __near r_iica0_interrupt(void)
Description	Accepts an IICA0 interrupt request.
Argument	None
Return Value	None

r_IICA0_master_handler()	
Outline	Master processing function for IICA0 interrupts
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h
Declaration	static void r_ IICA0_master_handler(void)
Description	Performs an IICA0 interrupt while the I2C bus is operating as the master. If NACK is detected for a slave address, the I2C bus generates a stop condition and ends communication. In transmission mode, if ACK is detected for a slave address, the I2C bus sends the register address or data. In reception mode, the I2C bus receives and stores data.
Argument	None
Return Value	None

R_IICA0_StartCondition()		
Outline	Generating a start condition	
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h	
Declaration	uint8_t R_IICA0_StartCondition(void)	
Description	Generates a start condition and checks its result.	
Argument	None	
Return Value	uint8_t	0x00 (SUCCESS): Generation of a start condition was completed. 0x8F (BUS_ERROR): An error occurred.

R_IICA0_StopCondition()		
Outline	Generating a stop condition	
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h	
Declaration	uint8_t R_IICA0_StopCondition(void)	
Description	Generates a stop condition and detects its result.	
Argument	None	
Return Value	uint8_t	0x00 (SUCCESS): A stop condition was generated and communication was completed. 0x8F (BUS_ERROR): An error occurred.

R_IICA0_Master_Send()		
Outline	Starting data transmission to the slave	
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h	
Declaration	uint8_t R_IICA0_Master_Send(uint8_t sladr8, uint8_t * const tx_buf, uint16_t tx_num)	
Description	If data transmission to IICA0 or data reception from IICA0 has ended normally, this function performs processing as follows: - If the I2C bus is idle, the function generates a start condition. - IICA0 is operating as the master, the function copies the transmission parameters and sends the slave address. In other cases, the function returns an error.	
Argument	uint8_t sladr8	Slave address
	uint8_t * const tx_buf	Pointer to the storage area of send data
	uint16_t tx_num	Number of bytes to be sent
Return Value	uint8_t	Status: 0x00 (SUCCESS): Communication started normally. 0x8F (BUS_ERROR): Processing of IICA0 is in progress or the I2C bus is busy.

R_IICA0_Tx()		
Outline	Sending data to the specified slave address	
Header	r_cg_macrodriver.h, Config_IICA0.h,, r_cg_userdefine.h	
Declaration	uint8_t R_IICA0_Tx(uint8_t sladr7, uint8_t adr, uint8_t * const tx_buf, uint16_t tx_num)	
Description	① This function checks the status of the I2C bus. If the I2C bus is idle, this function generates a start condition. If the I2C bus is busy, this function returns an error. ② After generating a start condition, the function enables the register address (g_adr_flag = 0x01). ③ The function converts the slave address from a 7-bit format to an 8-bit format, and starts sending data. ④ The function waits for completion of communication.	
Argument	uint8_t sladr7	7-bit slave address
	uint8_t adr	Value to be set for the register address
	uint8_t * const tx_buf	Pointer to the storage area of send data
	uint16_t tx_num	Number of bytes to be sent
Return Value	uint8_t	Result: 0x00 (SUCCESS): Transmission was completed. 0x8F (BUS_ERROR): Processing of IICA0 is in progress or the I2C bus is busy.

R_IICA0_Master_Receive()		
Outline	Starting sending of data to the specified slave	
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h	
Declaration	uint8_t R_IICA0_Master_Receive (uint8_t sladr8, uint8_t * const tx_buf, uint16_t tx_num)	
Description	If data transmission to IICA0 or data reception from IICA0 has ended normally, this function performs processing as follows: - If the I2C bus is idle, the function generates a start condition. - If IICA0 is operating as the master, the function copies transmission parameters and sends the slave address. In other cases, the function returns an error.	
Argument	uint8_t sladr8	Slave address
	uint8_t * const rx_buf	Pointer to the storage area of received data
	uint16_t rx_num	Number of bytes to be received
Return Value	uint8_t	Status: 0x00 (SUCCESS): Communication started normally. 0x8F (BUS_ERROR): Processing of IICA0 is in progress or the I2C bus is busy. 0xFF (COM_ERROR): Error in the number of received bytes

R_IICA0_Rx()		
Outline	Receiving data from the specified slave address	
Header	r_cg_macrodriver.h, Config_IICA0.h,, r_cg_userdefine.h	
Declaration	uint8_t R_IICA0_Rx(uint8_t sladr7, uint8_t adr, uint8_t * const tx_buf, uint16_t tx_num)	
Description	① This function checks the status of the I2C bus. If the I2C bus is idle, the function generates a start condition. If the I2C bus is busy, the function returns an error. ② After generating a start condition, the function sets communication parameters. ③ The function enables the register address (g_adr_flag = 0x10). ④ The function converts the slave address from a 7-bit format to an 8-bit format, and starts receiving data. ⑤ The function waits for completion of communication.	
Argument	uint8_t sladr7	7-bit slave address
	uint8_t adr	Value to be set for the register address
	uint8_t * const tx_buf	Pointer to the storage area of send data
	uint16_t tx_num	Number of bytes to be sent
Return Value	uint8_t	Result: 0x00 (SUCCESS): Reception was completed. 0x8F (BUS_ERROR): Processing of IICA0 is in progress or the I2C bus is busy. 0xFF (COM_ERROR): Error in the number of received bytes

R_IICA0_poll()		
Outline	Performing polling for the IICA0 communication status	
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h	
Declaration	uint8_t R_IICA0_poll(void)	
Description	This function checks the status of the communication that is in progress.	
Argument	None	
Return Value	uint8_t	Communication status: 0x00 (SUCCESS): Communication is complete (successful) 0x01 (ON_COMMU): Communication is in progress 0x8F (BUS_ERROR): Processing of IICA0 is in progress or the I2C bus is busy. 0x80 (NO_SLAVE): NACK was detected for the slave address. 0xC0 (NACK): NACK was detected.

wait_time()	
Outline	Waiting for 50 us
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h
Declaration	void wait_time(void)
Description	Waits for 50 us.
Argument	None
Return Value	None

R_IICA0_wait_comend()		
Outline	Waiting for completion of IICA0 communication	
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h	
Declaration	uint8_t R_IICA_wait_comend(uint8_t stop)	
Description	This function waits for completion of the communication that is in progress. If an error is detected after communication is completed, the function generates a stop condition and ends communication.	
Argument	uint8_t stop	0x00: Generate a stop condition in the case of normal end. 0x01: Generate a stop condition when communication is completed.
Return Value	uint8_t	Communication result: 0x00 (SUCCESS): Communication is complete (successful). 0x8F (BUS_ERROR): Processing of IICA0 is in progress or the I2C bus is busy. 0x80 (NO_SLAVE): NACK was detected for the slave address. 0xC0 (NACK): NACK was detected.

R_IICA0_check_comstate()		
Outline	Checking the IICA0 communication status	
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h	
Declaration	uint8_t R_IICA0_check_comstate(void)	
Description	This function checks the status of IICA0 (by reading g_status).	
Argument	None	
Return Value	uint8_t	Value of g_status

R_IICA0_bus_check()		
Outline	Checking the status of the I2C bus	
Header	r_cg_macrodriver.h, Config_IICA0.h, r_cg_userdefine.h	
Declaration	uint8_t R_IICA0_bus_check(void)	
Description	This function checks the status of the I2C bus. If IICA0 is processing data, the function returns an error. In the case where the I2C bus is secured but communication has ended or where the I2C bus is idle, the function generates a start condition.	
Argument	None	
Return Value	uint8_t	Result: 0x00 (SUCCESS): Generation of a start condition was completed. 0x8F (BUS_ERROR): Processing of IICA0 is in progress or the I2C bus is busy.

r_tau0_channel7_interrupt()	
Outline	Performs a 10-ms interval interrupt
Header	Config_TAU0_7.h、r_cg_macrodriver.h、r_cg_userdefine.h
Declaration	static void r_tau0_channel7_interrupt(void)
Description	This function samples the status of a switch.
Argument	None
Return Value	None

4.6 Flowcharts

4.6.1 Main Processing

Figure 4-1 to Figure 4-3 shows the flowchart of the main processing.

Figure 4-1 Main Processing (1/3)

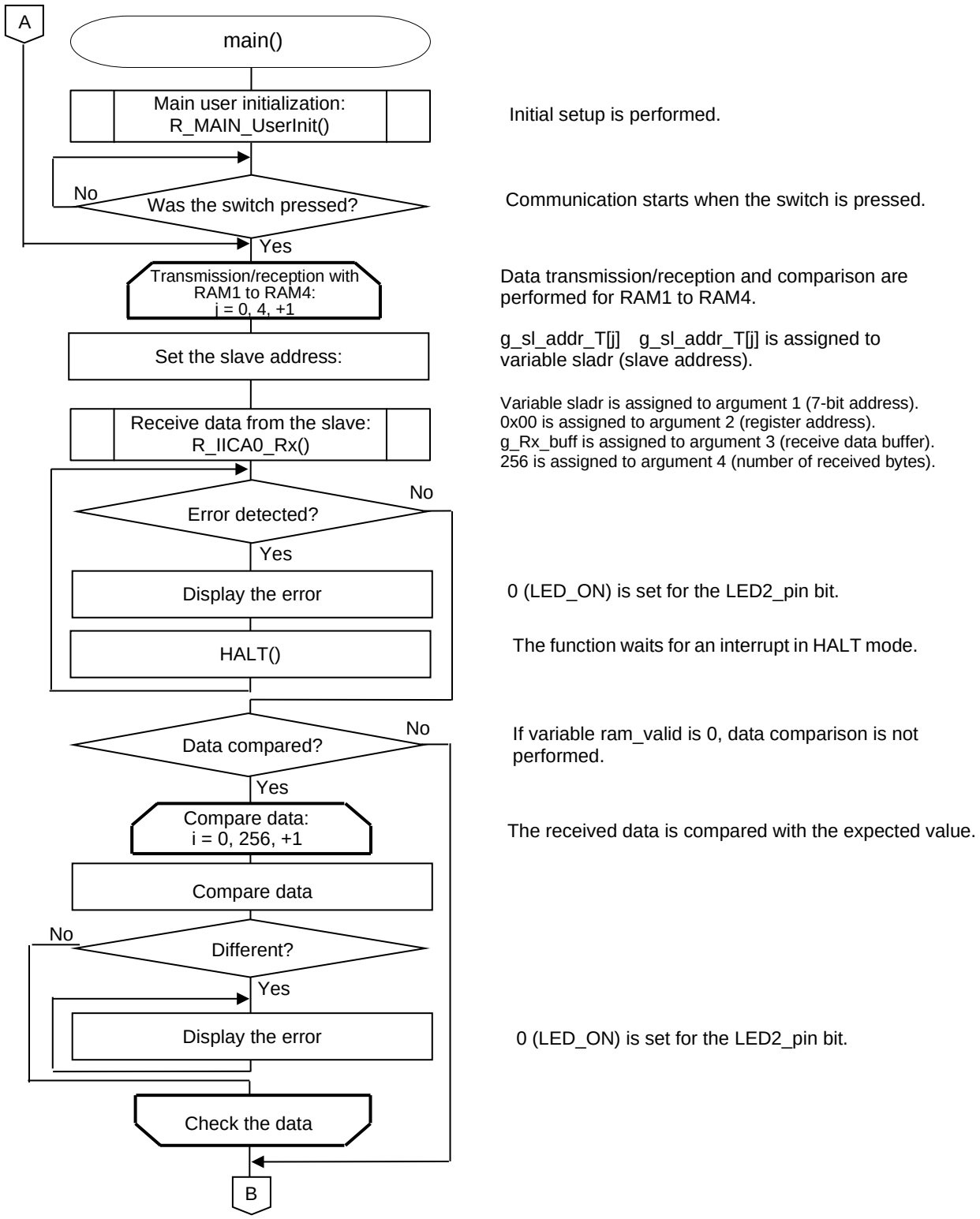
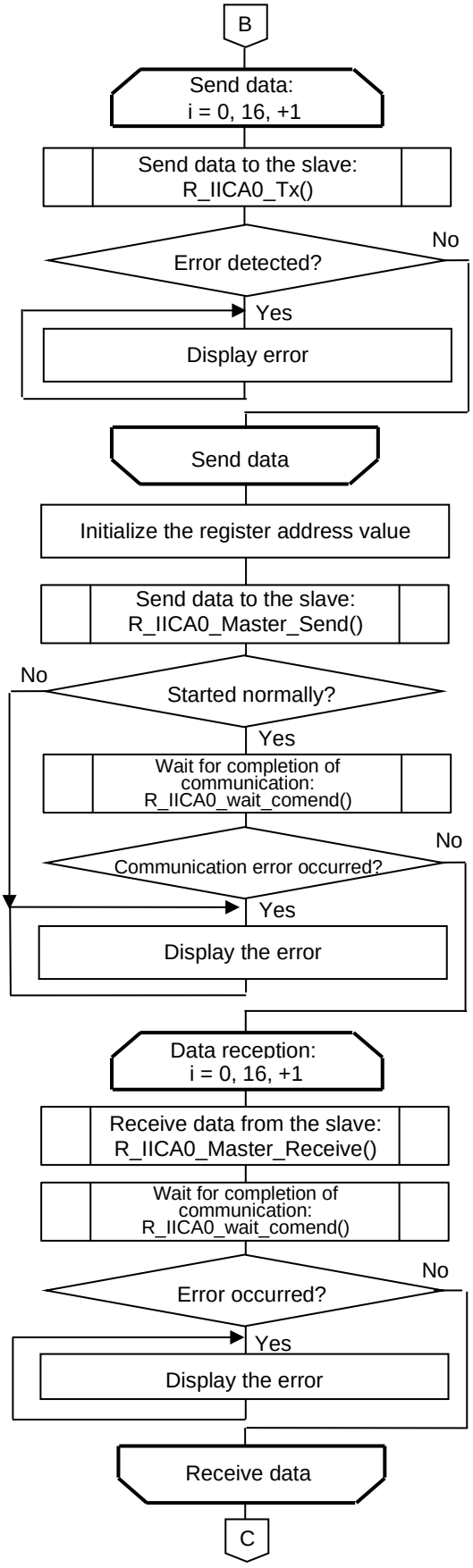


Figure 4-2 Main Processing (2/3)



sladr is assigned to argument 1 (slave address).
i×16 is assigned to argument 2 (register address)
&INC_DATA[i][0] is assigned to argument 3 (buffer).
16 is assigned to argument 4 (number of bytes to be sent)

The communication error that occurred is handled.

0 (LED_ON) is set for the LED2_pin bit.

0x00 is assigned to variable g_Tx_buff[0].

Register address settings:
(g_sl_addr[i] << 1) is assigned to argument 1 (slave address).
g_Tx_buff is assigned to argument 2 (register address)
1 is assigned to argument 3 (number of bytes to be sent).

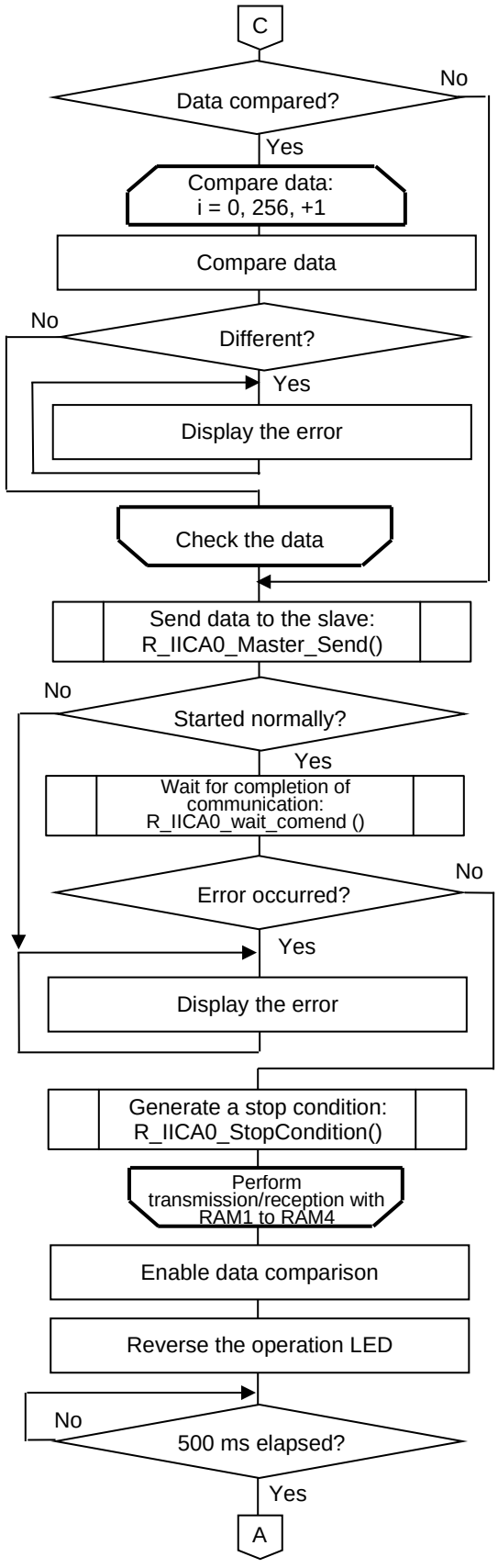
The function waits for completion of transmission.

0 (LED_ON) is set for the LED2_pin bit.

Data reception:
(g_sl_addr[i] << 1) is assigned to argument 1 (slave address)
&g_Rx_buff[i*16] is assigned to argument 2 (buffer address).
16 is assigned to argument 3 (number of received bytes).

0 (LED_ON) is set for the LED2_pin bit.

Figure 4-3 Main Processing (3/3)



If variable ram_valid is 0, comparison is not performed.

The received data is compared with the expected value.

0 (LED_ON) is set for the LED2_pin bit.

Transmission parameters:
(g_sl_addr[i] << 1) is assigned to argument 1 (slave address).
g_Tx_buff is assigned to argument 2 (register address).
257 is assigned to argument 3 (number of bytes to be sent).

0 (LED_ON) is set for the LED2_pin bit.

Variable ram_valid is set.

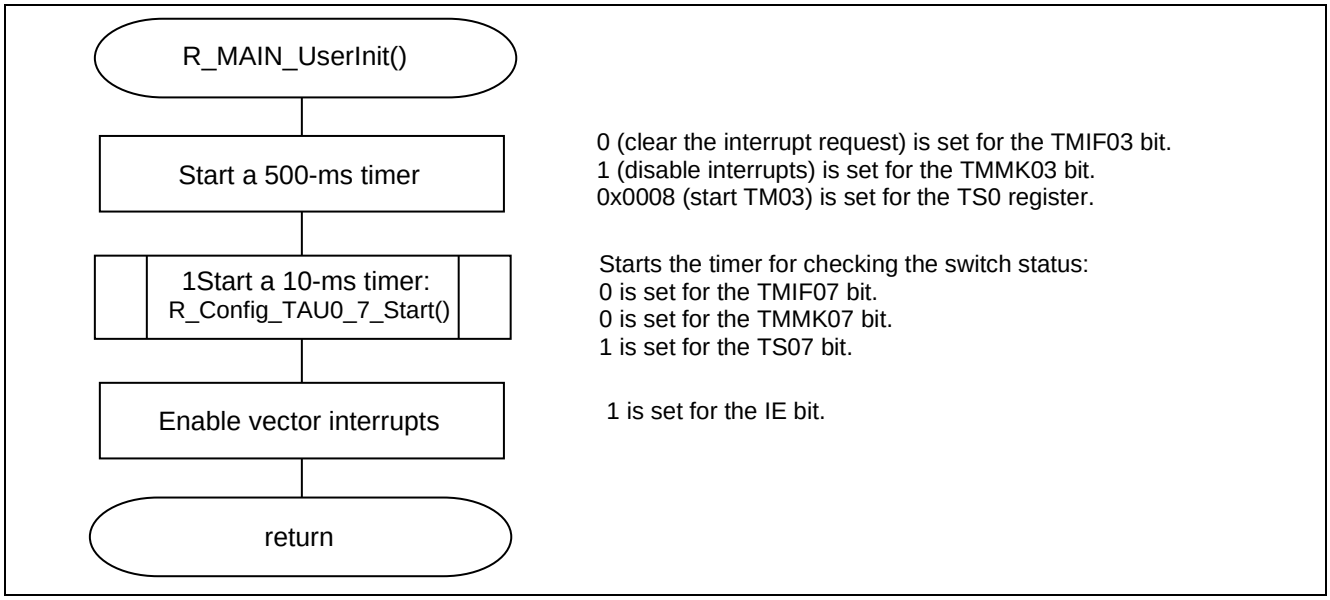
The LED1_pin bit is reversed.

The function waits until the TMIF03 bit is set to 1.

4.6.2 Variables Initialization Processing

Figure 4-4 shows the flowchart of the variables initialization function.

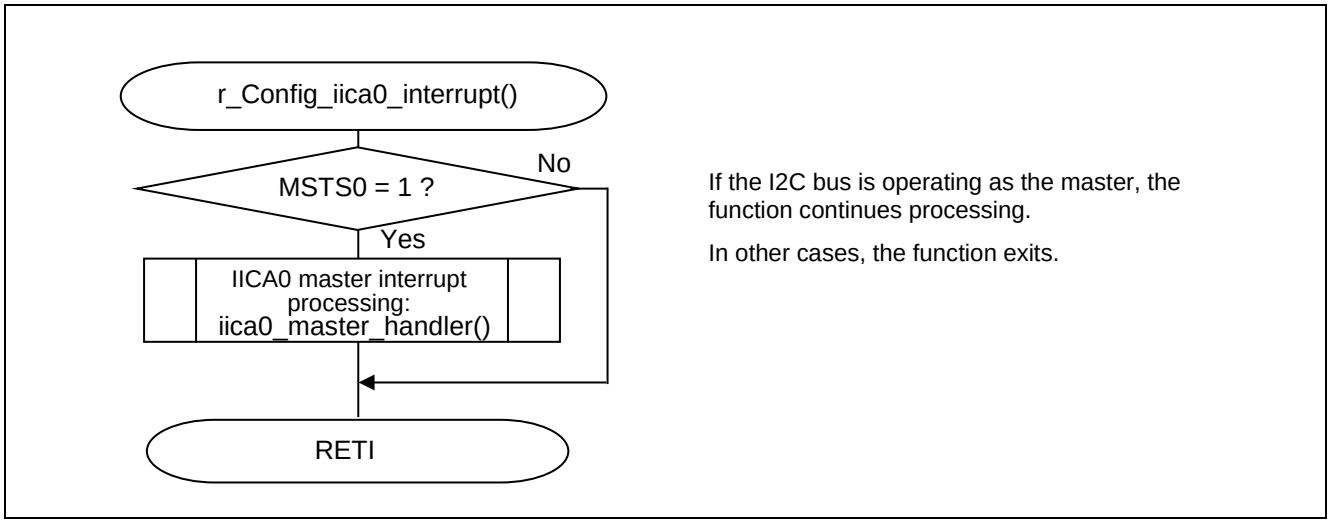
Figure 4-4 Variables Initialization Function



4.6.3 IICA0 Interrupt Processing Function

Figure 4-5 shows the flowchart of the IICA0 interrupt processing function.

Figure 4-5 IICA0 Interrupt Processing Function



4.6.4 IICA0 Master Interrupt Processing Function

Figure 4-6 to Figure 4-9 show the flowchart of the IICA0 master interrupt processing function.

Figure 4-6 IICA0 Master Interrupt Processing Function (1/4)

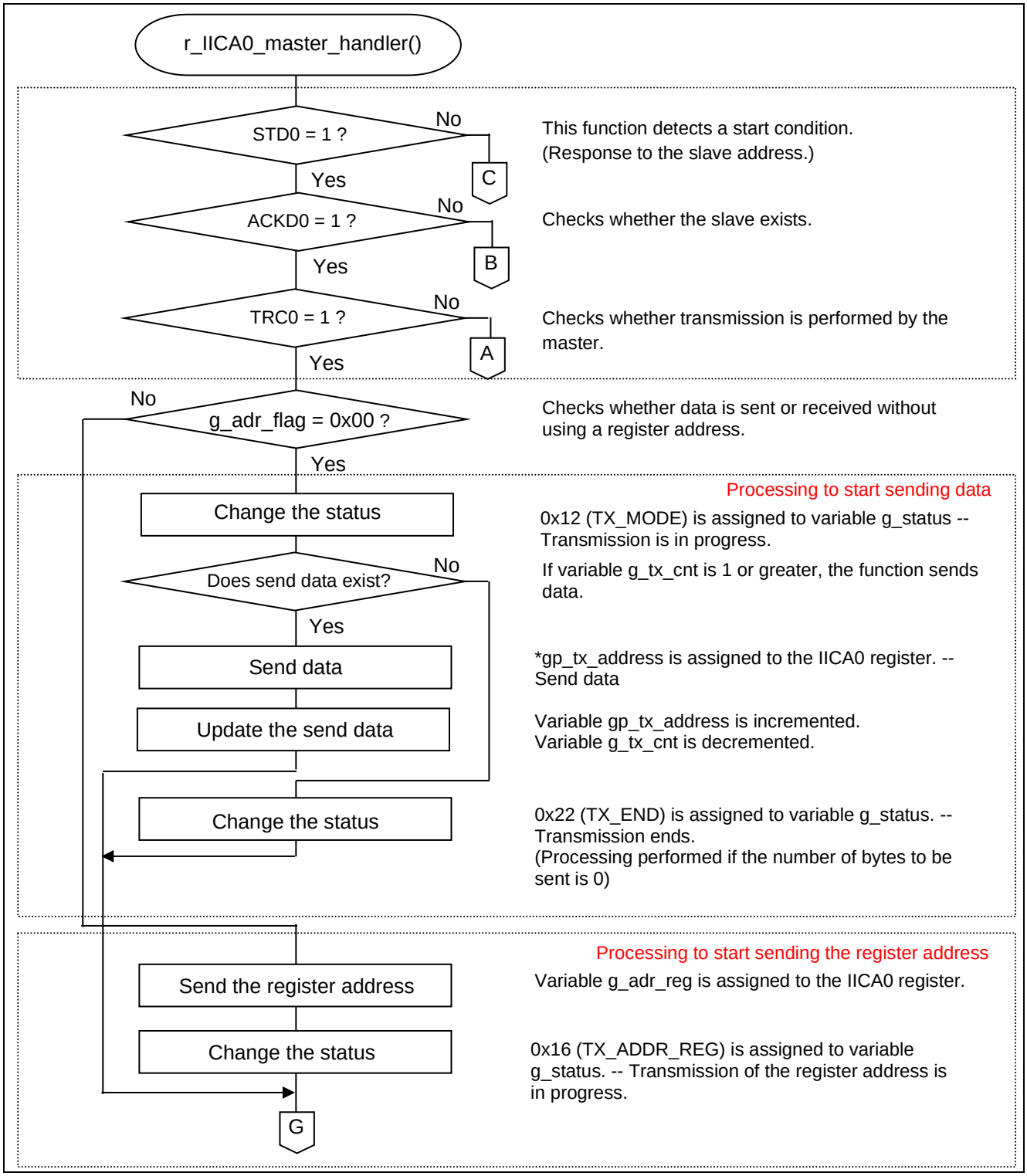


Figure 4-7 IICA0 Master Interrupt Processing Function (2/4)

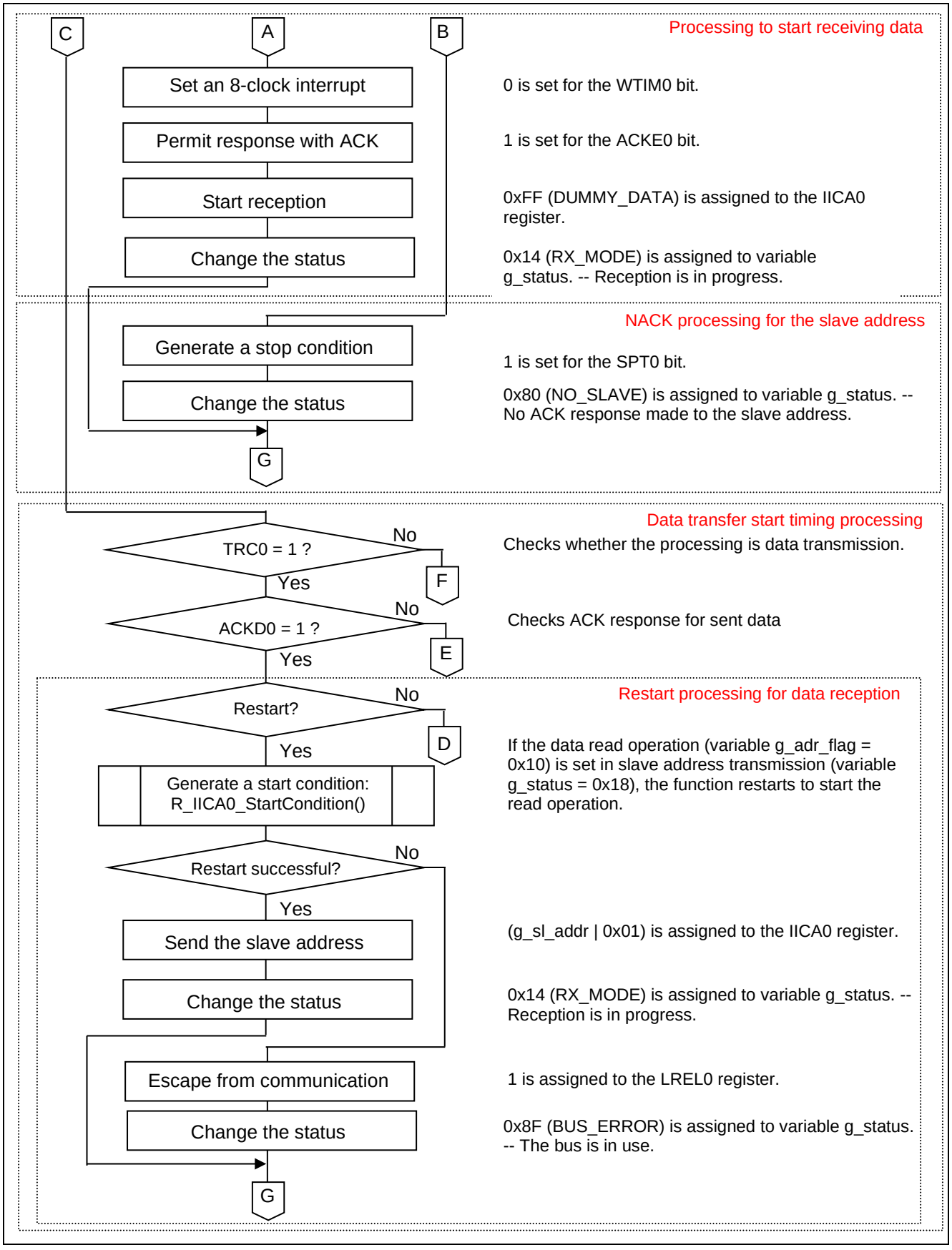


Figure 4-8 IICA0 Master Interrupt Processing Function (3/4)

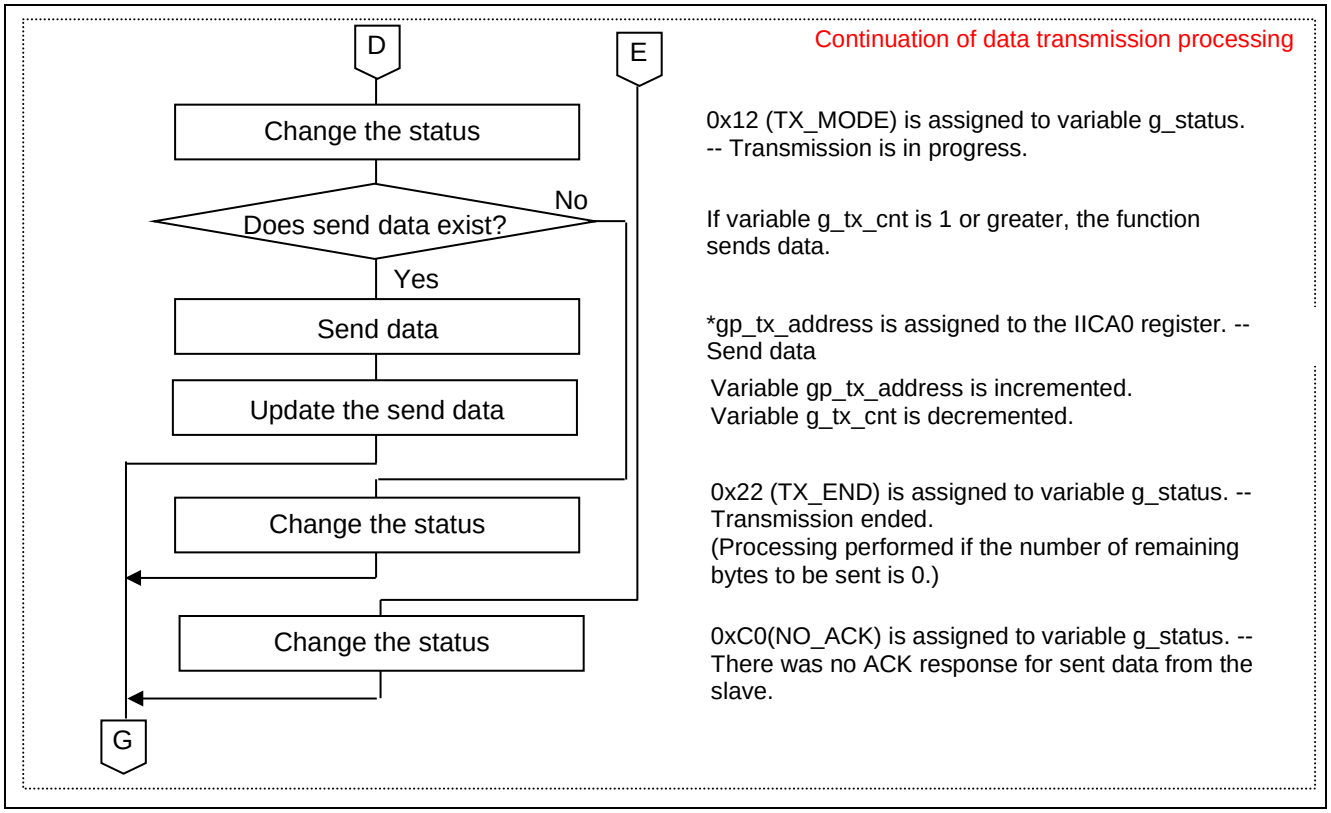
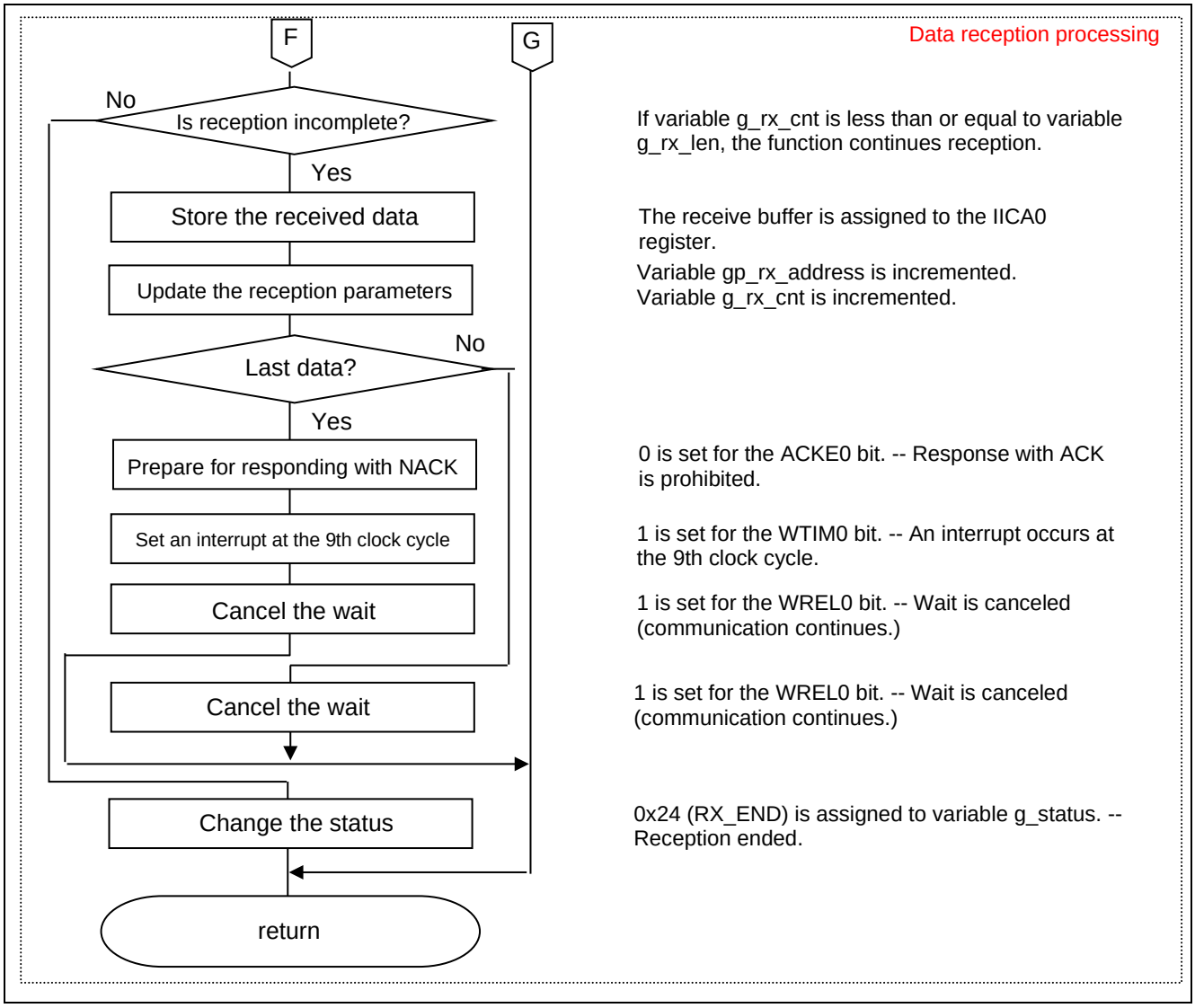


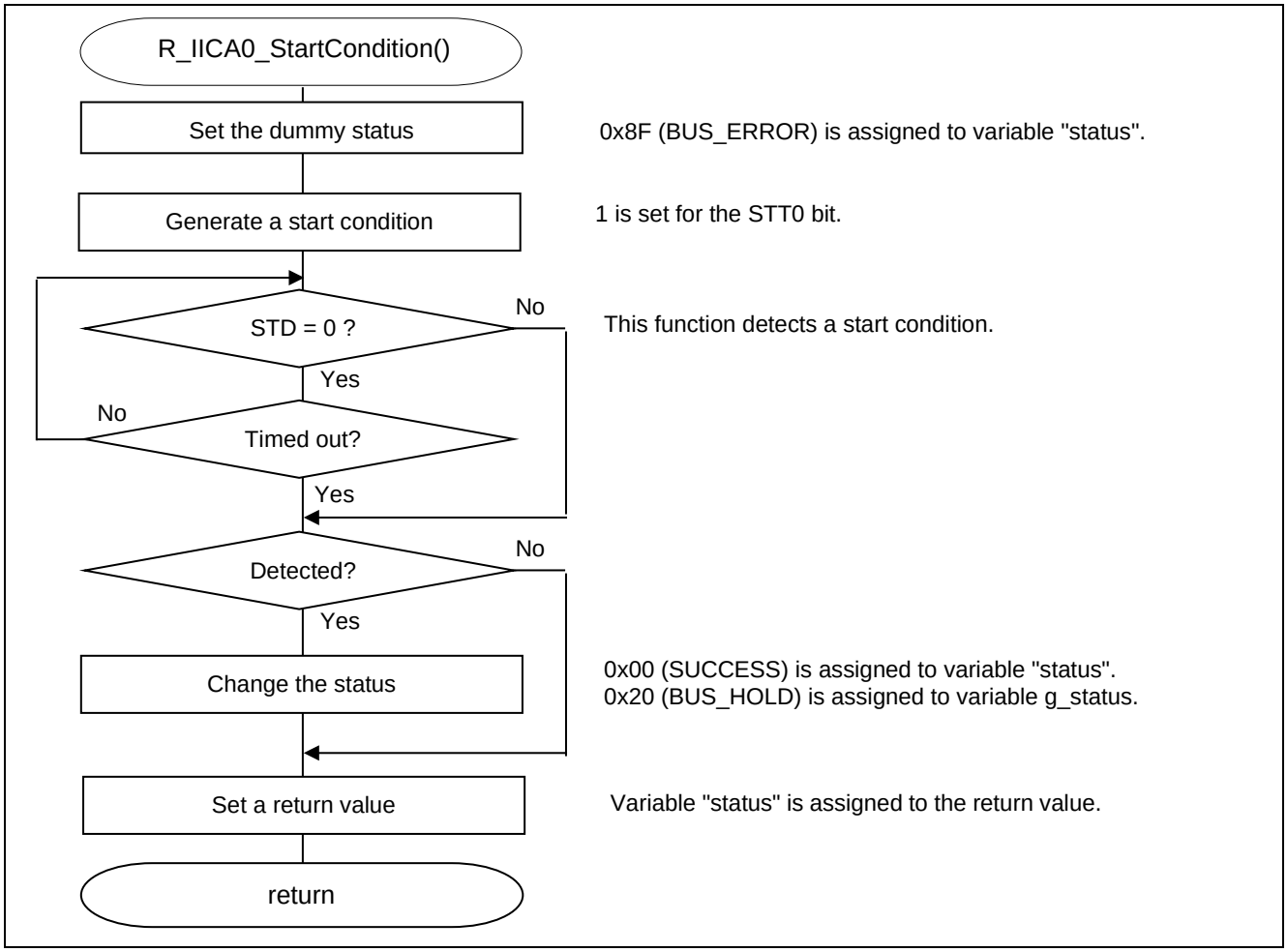
Figure 4-9 IICA0 Master Interrupt Processing Function (4/4)



4.6.5 Function That Generates an IICA0 Start Condition

Figure 4-1 shows a flowchart of the function that generates an IICA0 start condition.

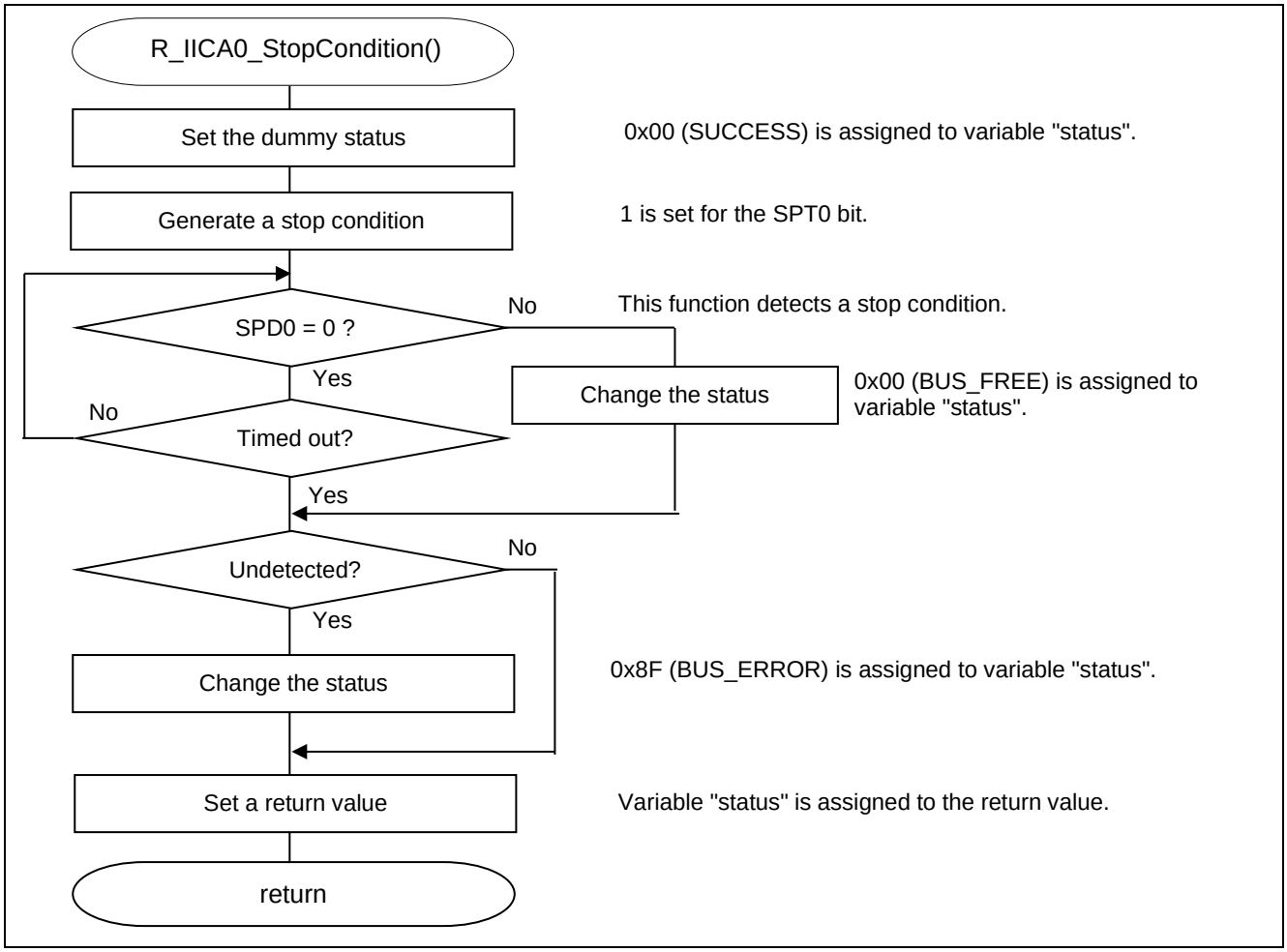
Figure 4-1 Function That Generates an IICA0 Start Condition



4.6.6 Function that Generates an IICA0 Stop Condition

Figure 4-2 shows the flowchart of the function that generates an IICA0 stop condition.

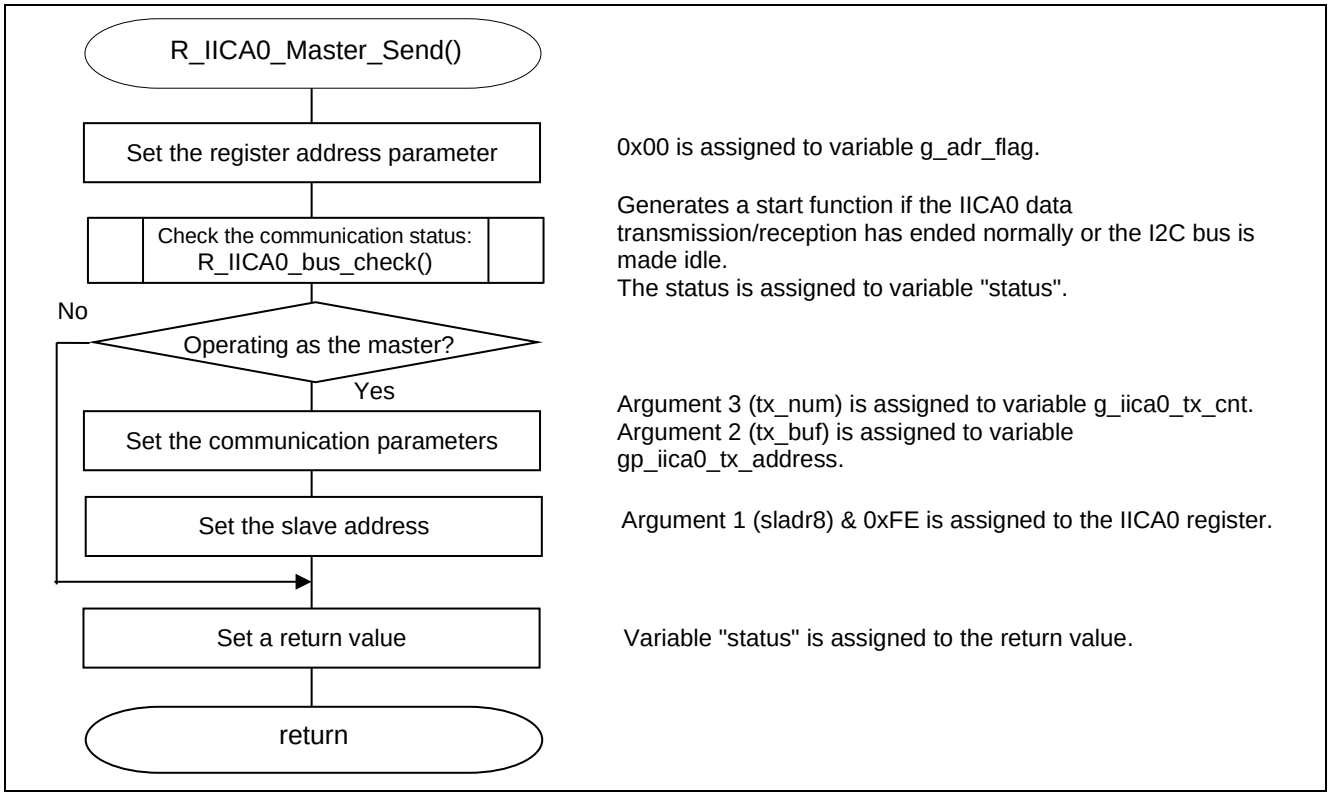
Figure 4-2 Function That Generates an IICA0 Stop Condition



4.6.7 IICA0 Master Transmission Start Function

Figure 4-3 shows the flowchart of the IICA0 master transmission start function.

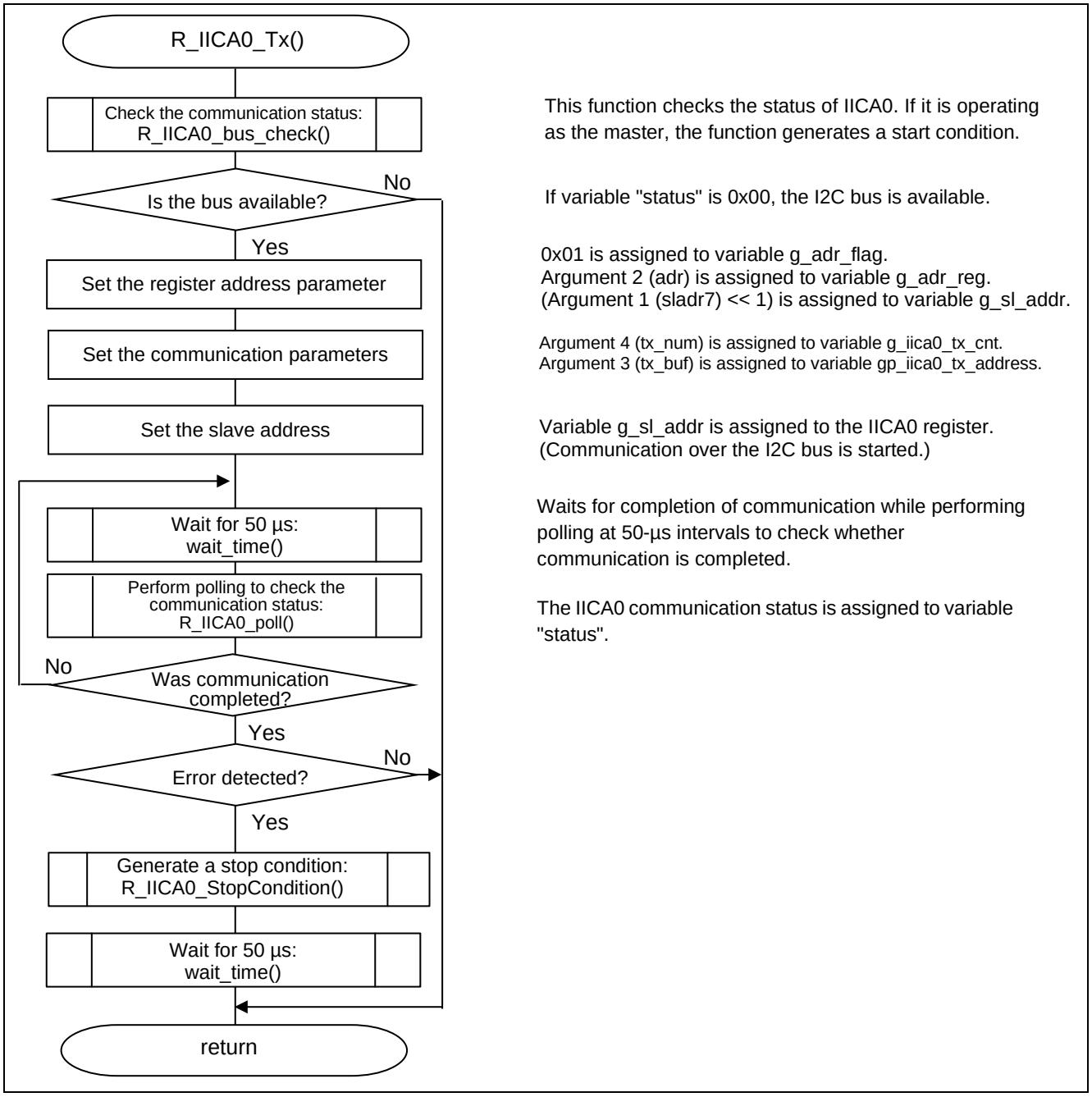
Figure 4-3 IICA0 Master Transmission Start Function



4.6.8 IICA0 Data Transmission Processing Function

Figure 4-4 shows the flowchart of the IICA0 data transmission processing function.

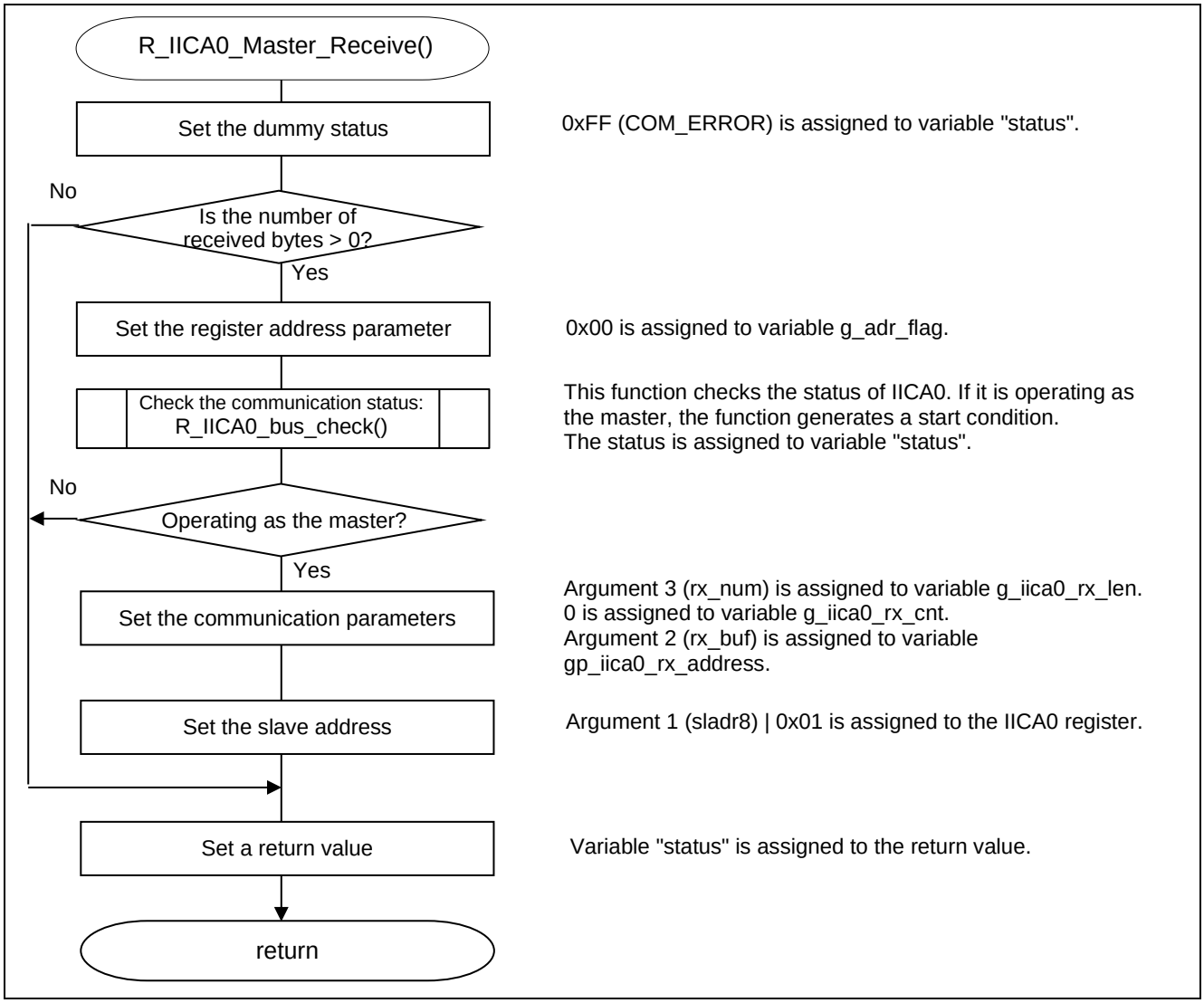
Figure 4-4 IICA0 Data Transmission Processing Function



4.6.9 IICA0 Master Reception Start Function

Figure 4-5 shows the flowchart of the IICA0 master reception start function.

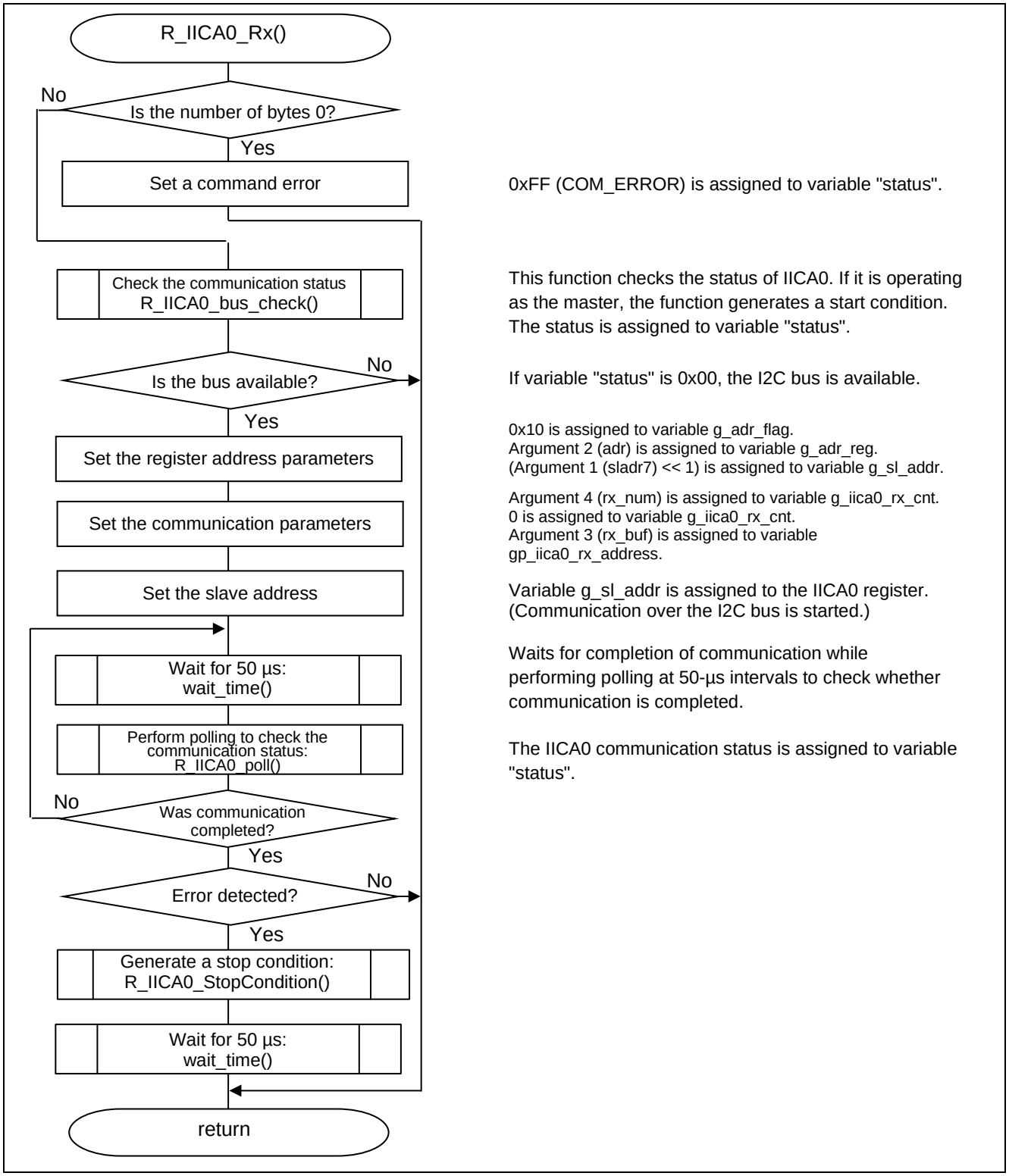
Figure 4-5 IICA0 Master Reception Start Function



4.6.10 IICA0 Data Reception Processing Function

Figure 4-6 shows the flowchart of the IICA0 data reception processing function.

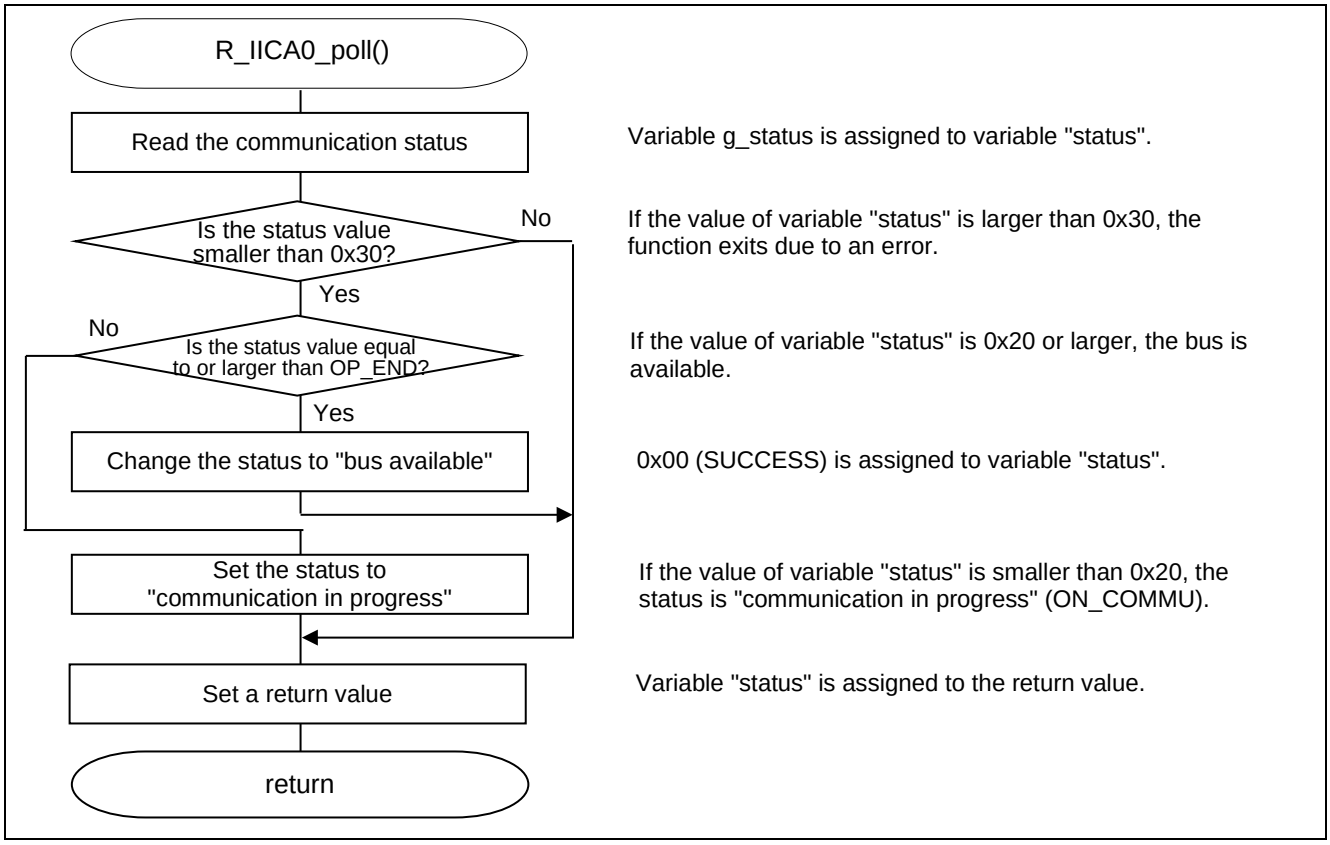
Figure 4-6 IICA0 Data Reception Processing Function



4.6.11 Communication Status Polling Function

Figure 4-7 shows the flowchart of the communication status polling function.

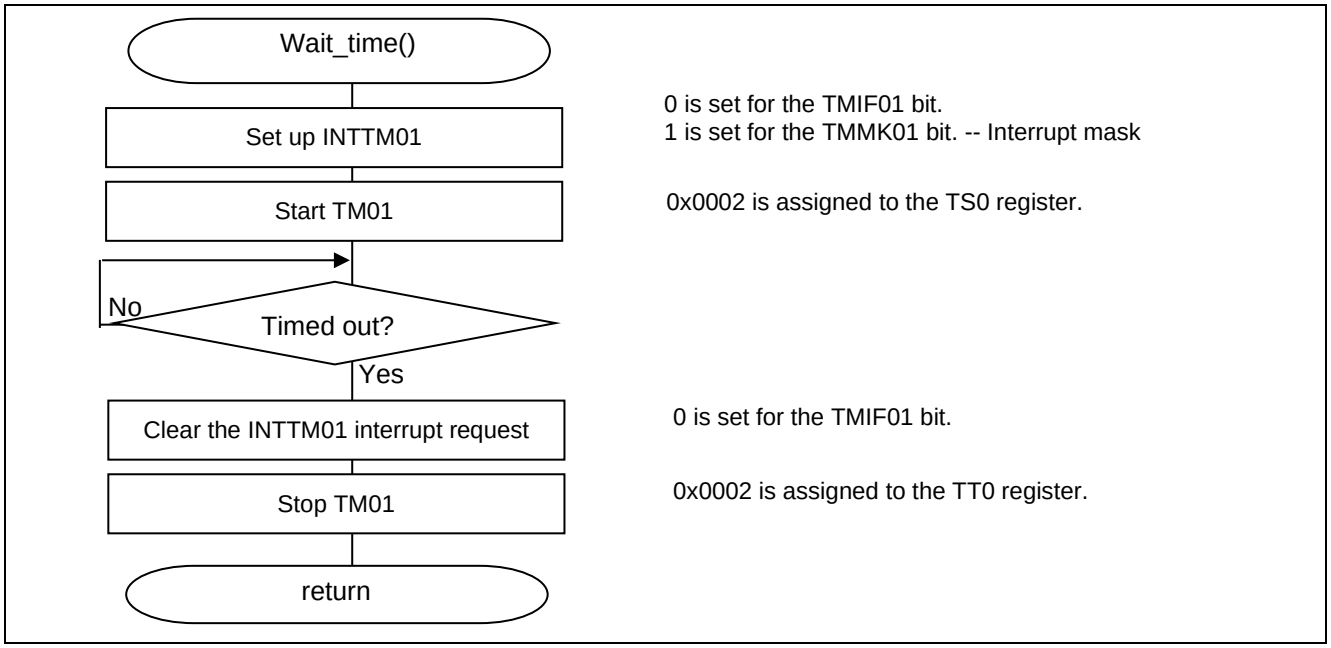
Figure 4-7 Communication Status Polling Function



4.6.12 50-µs Wait Function

Figure 4-8 shows the flowchart of the 50-µs wait function.

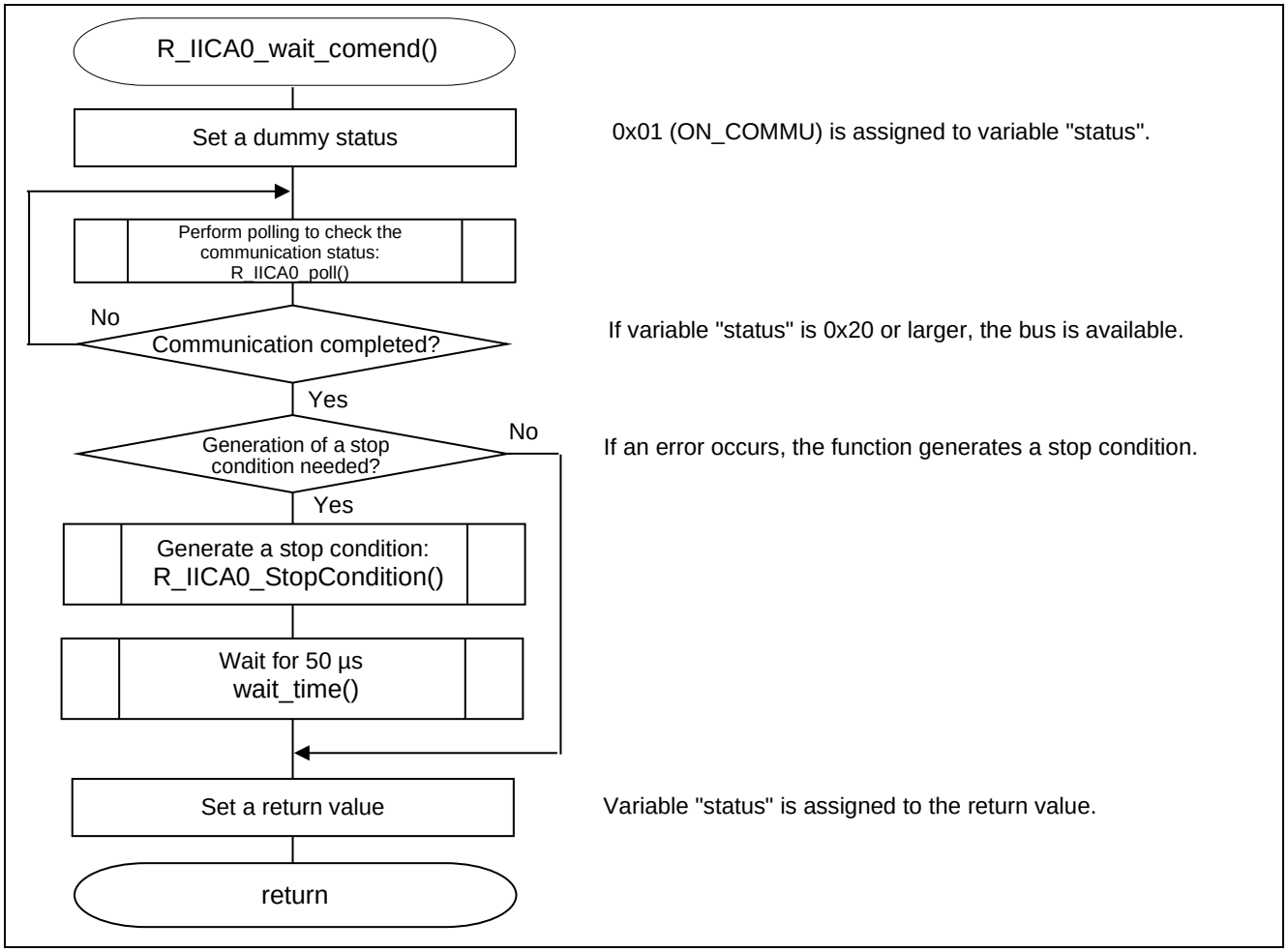
Figure 4-8 50-µs Wait Function



4.6.13 Communication Completion Wait Function

Figure 4-9 shows the flowchart of the communication completion wait function.

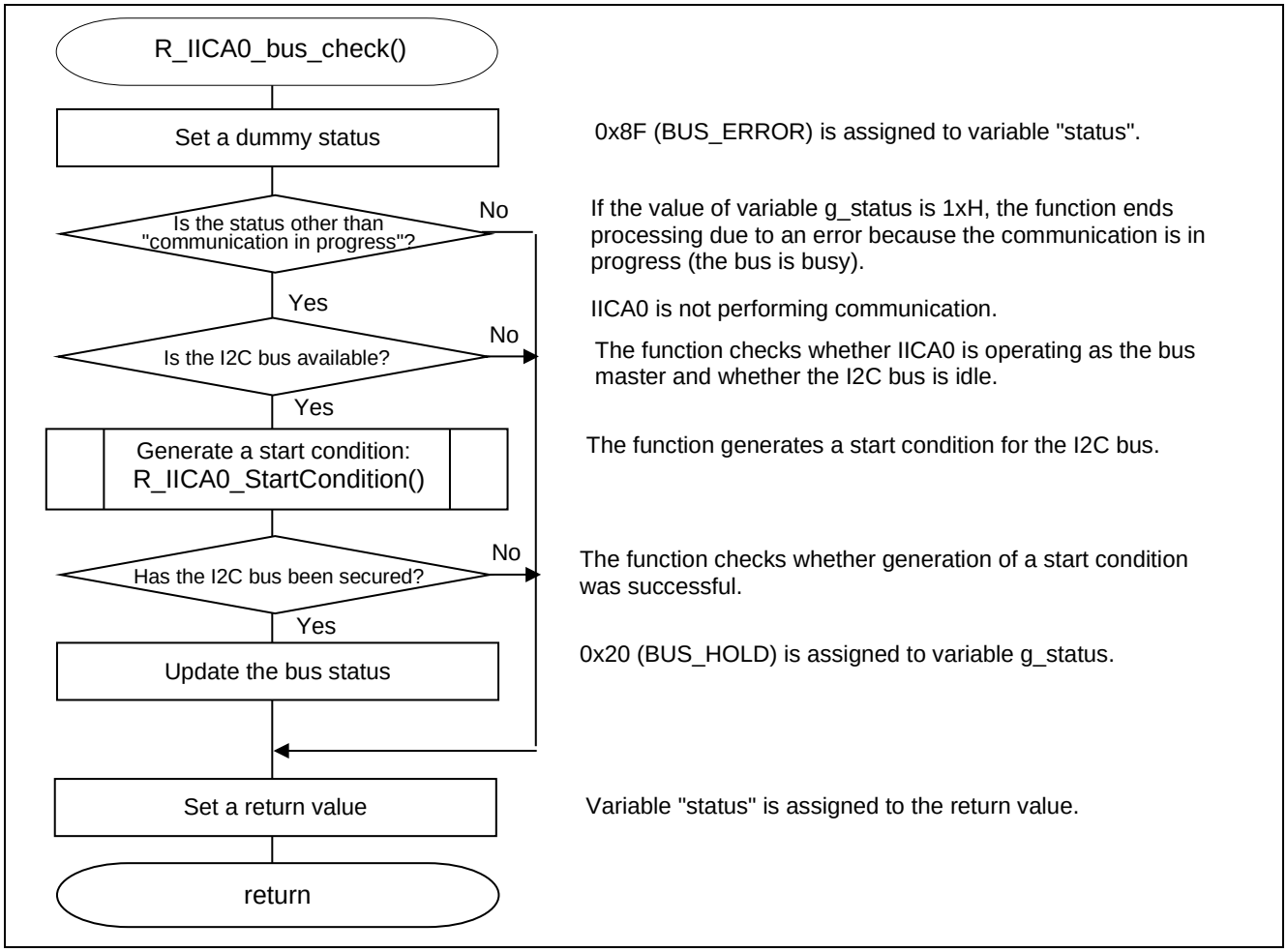
Figure 4-9 Communication Completion Wait Function



4.6.14 I2C Bus Status Check Function

Figure 4-10 shows the flowchart of the I2C bus status check function.

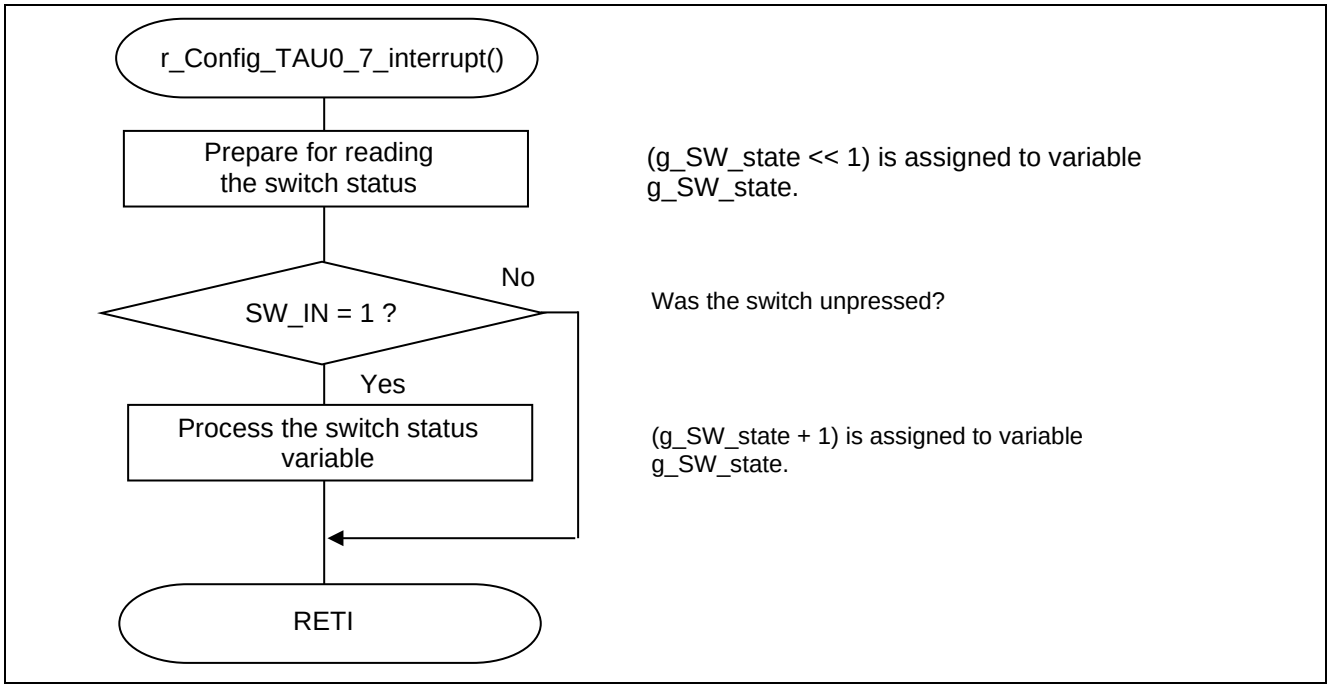
Figure 4-10 I2C Bus Status Check Function



4.6.15 Function That Handles a 10-ms Interval Timer Interrupt

Figure 4-11 shows the flowchart of the function that handles a 10-ms interval timer interrupt.

Figure 4-11 Function That Handles a 10-ms Interval timer Interrupt



5. Sample code

Sample code can be downloaded from the Renesas Electronics website.

6. Reference Documents

RL78/G23 User's Manual: Hardware (R01UH0896)
RL78 family user's manual software (R01US0015)
The latest versions can be downloaded from the Renesas Electronics website.

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

Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Jul.01.21	—	First Edition

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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