RL78/G12

Self-Programming (Received Data via IIC)

R01AN1463EJ0110
Rev. 1.10
June 01, 2016

Introduction

This application note gives the outline of flash memory reprogramming using a self-programming technique. In this application note, flash memory is reprogrammed using the flash memory self-programming library Type01. The reprogramming data is received via IIC.

The sample program described in this application note limits the target of reprogramming to the boot area. For details on the procedures for performing self-programming and for reprogramming the entire area of code flash memory, refer to RL78/G13 Microcontroller Flash Memory Self-Programming Execution (R01AN0718E) Application Note.

Target Device

RL78/G12

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

This application note explains a sample program that performs flash memory reprogramming using a self-programming library.

The sample program reads values from the code flash memory area ranging from addresses 0x3BFC to 0x3BFF and sets the flashing period of the LEDs. Subsequently, the sample program receives data (4 bytes) from the sending side and carries out self-programming to rewrite the values stored in the code flash memory addresses 0x3BFC to 0x3BFF with the received data. When the rewrite is completed, the sample program reads values from the code flash memory addresses 0x3BFC to 0x3BFF again and resets the flashing period of the LEDs with the read value.

Table 1.1 lists the peripheral functions to be used and their uses.

Table 1.1 Peripheral Functions to be Used and their Uses

Peripheral Function	Use
Serial interface IICA	Receives data via IIC.
Port I/O	Displays text on the LCD. Turns on and off the LED.

1.1 Outline of the Flash Memory Self-Programming Library

The flash memory self-programming library is a software product that is used to reprogram the data in the code flash memory using the firmware installed on the RL78 microcontroller.

The contents of the code flash memory can be reprogrammed by calling the flash memory self-programming library from a user program.

To do flash memory self-programming, it is necessary for the user program to perform initialization for flash memory self -programming and to execute the C or assembler functions that correspond to the library functions to be used.

1.2 Code Flash Memory

The configuration of the RL78/G12 (R5F1026A) code flash memory is shown below.

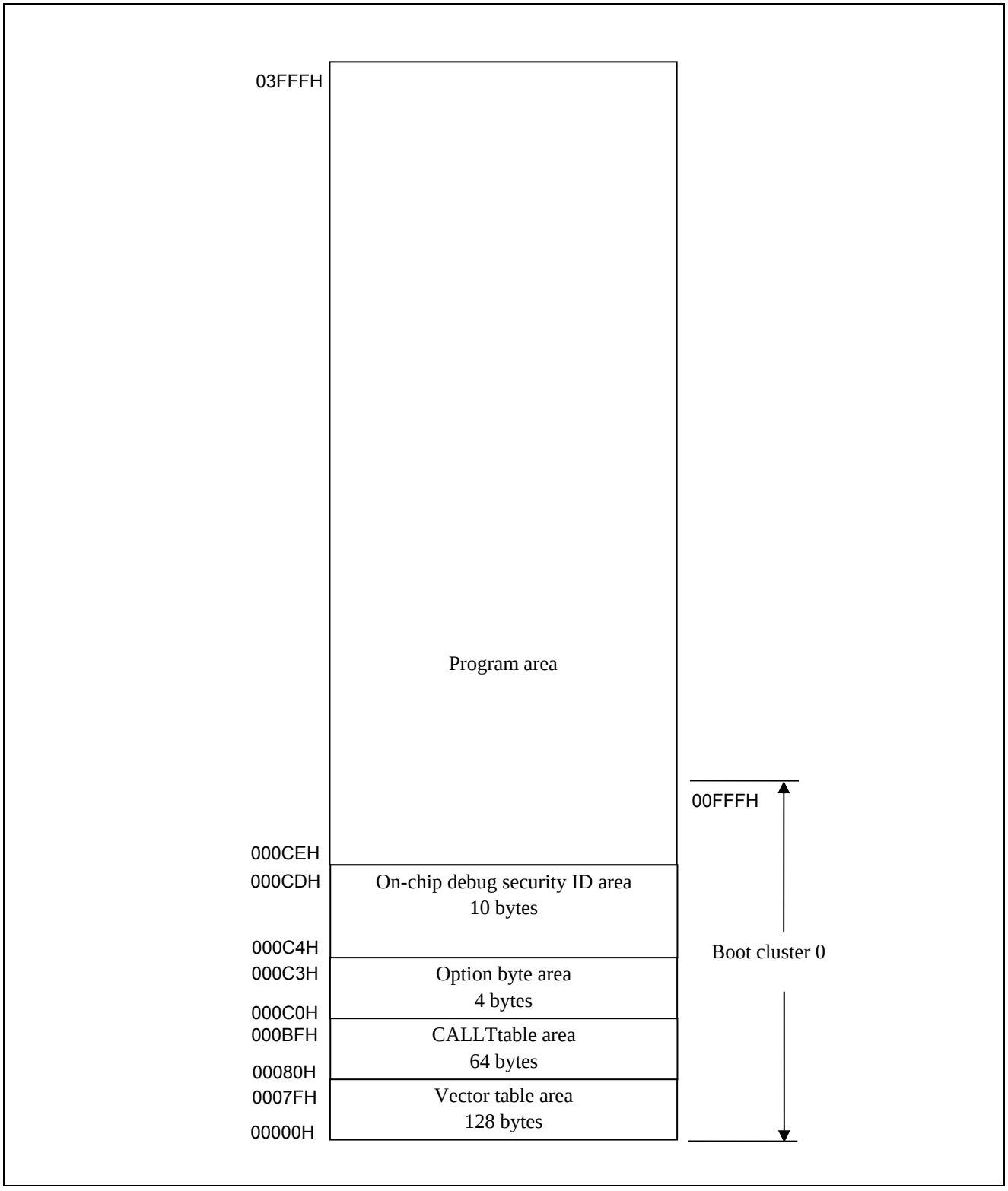

Figure 1.1 Code Flash Memory Configuration

The features of the RL78/G12 code flash memory are summarized below.

Table 1.2 Features of the Code Flash Memory

Item	Description
Minimum unit of erasure and verification	1 block (1024 bytes)
Minimum unit of programming	1 word (4 bytes)
Security functions	Block erasure, programming, and boot area reprogramming protection are supported. (They are enabled at shipment)
	It is possible to disable reprogramming and erasure outside the specified window only at flash memory self-programming time using the flash shield window.
	Security settings programmable using the flash memory self-programming library

Caution: The boot area reprogramming protection setting and the security settings for outside the flash shield window are disabled during flash memory self-programming.

1.3 Flash Memory Self-Programming

The RL78/G12 is provided with a library for flash memory self-programming. Flash memory self-programming is accomplished by calling functions of the flash memory self-programming library from the reprogramming program.

The code flash memory cannot be referenced while flash memory self-programming is in progress. When the user program needs to be run, it is necessary to relocate part of the segments for the flash memory self-programming library and the reprogramming program in RAM when erasing or reprogramming the code flash memory or making settings for the security flags.

1.3.1 Flash Memory Reprogramming

This subsection describes the outline image of reprogramming using the flash memory self-programming technique. The program that performs flash memory self-programming is placed in boot cluster 0.

The sample program covered in this application note limits the target of reprogramming to the boot area. For details on the procedures for perform self-programming and for reprogramming the entire area of code flash memory, refer to RL78/G13 Microcontroller Flash Memory Self-Programming Execution (R01AN0718E) Application Note.

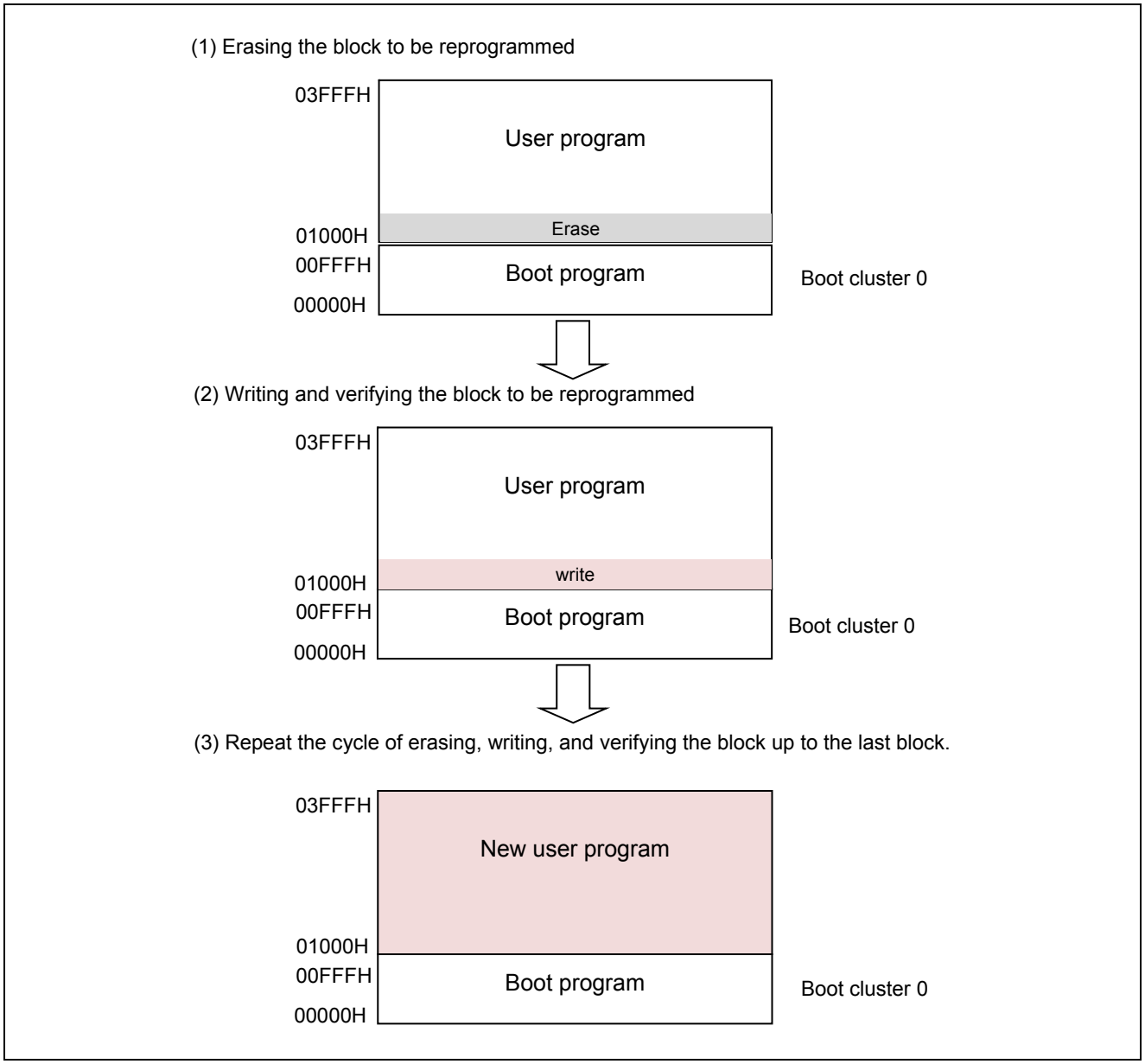

Figure 1.2 Outline of Flash Memory Reprogramming (1/2)

RL78/G12 does not have the boot swap function. Do not make factors that prevent reprogramming, such as shutting down of the power supply or external reset signal, while the boot area is being updated. If reprogramming is prevented on the way, restarting the program by reset or reprogramming may be unavailable ever because of its broken data.

1.3.2 Flash Shield Window

The flash shield window is one of security mechanisms used for flash memory self-programming. It disables the write and erase operations on the areas outside the designated window only during flash memory self-programming.

The figure below shows the outline image of the flash shield window on the area of which the start block is 08H and the end block is 1FH.

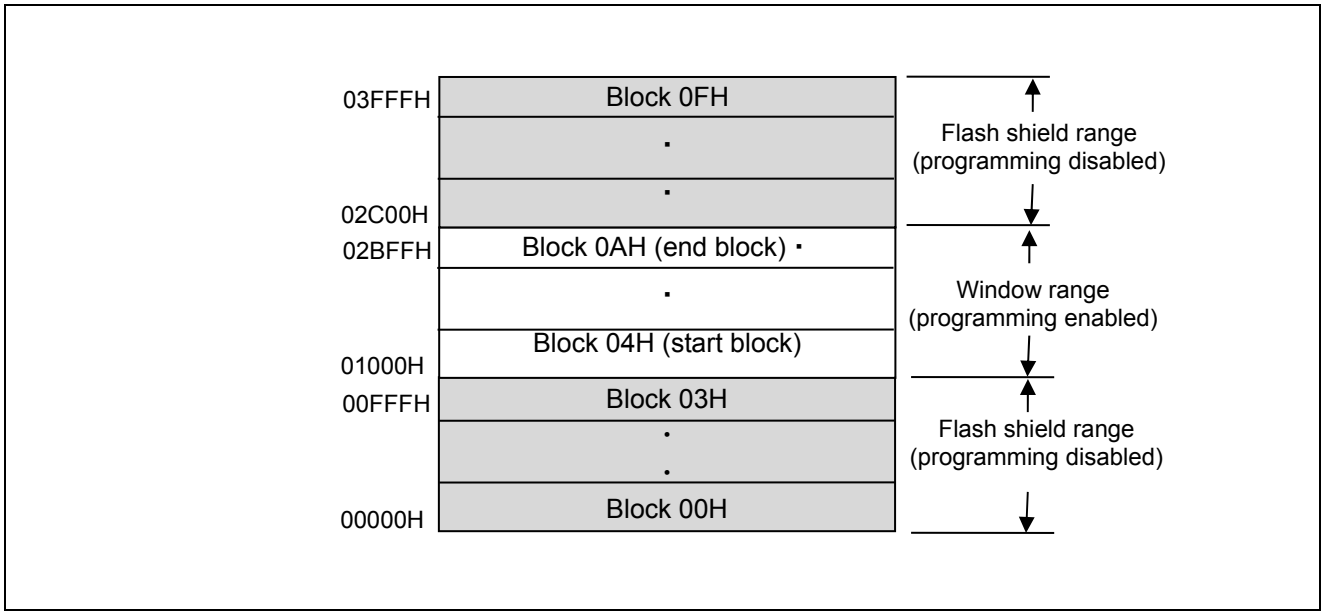

Figure 1.3 Outline of the Flash Shield Window

1.4 How to Get the Flash Memory Self-Programming Library

Before compiling the sample program, please download the latest flash self-programming library and copy the library files to the following folder below “Workspace”.

incrl78 folder : fsl.h, fsl.inc, fsl_types.h

librl78 folder : fsl.lib

The flash memory self-programming library is available on the Renesas Electronics Website.

Please contact your local Renesas Electronics sales office or distributor for more information.

2. Operation Check Conditions

The sample code described in this application note has been checked under the conditions listed in the table below.

Table 2.1 Operation Check Conditions

Item	Description
Microcontroller used	RL78/G12 (R5F1026A)
Operating frequency	• High-speed on-chip oscillator (HOCO) clock: 32 MHz • CPU/peripheral hardware clock: 32 MHz
Operating voltage	5.0 V (Operation is possible over a voltage range of 2.9 V to 5.5 V.) LVD operation (V_{LVD}): Reset mode which uses 2.81 V (2.76 V to 2.87 V)
Integrated development environment	CS+ V3.02.00 from Renesas Electronics Corp.
C compiler	CA78K0R V1.72 from Renesas Electronics Corp.
Board to be used	RL78/G12 target board (QB-R5F1026A-TB)
Flash memory self-programming library (Type, Ver)	FSLRL78 Type01, Ver 2.20 Note

Note: Use and evaluate the latest version.

3. Related Application Notes

The application notes that are related to this application note are listed below for reference.

- RL78/G12 Initialization (R01AN1030E) Application Note
- RL78/G12 Serial Interface IICA (for Master Transmission/Reception) (R01AN1371E) Application Note
- RL78/G12 Serial Interface IICA (for Slave Transmission/Reception) (R01AN1372E) Application Note
- RL78 Microcontroller Flash Memory Self-Programming Library Type01 (R01AN0350E) Application Note
- RL78/G13 Microcontroller Flash Memory Self-Programming Execution (R01AN0718E) Application Note.

4. Description of the Hardware

4.1 Hardware Configuration Example

Figure 3.1 shows an example of the hardware configuration used for this application note.

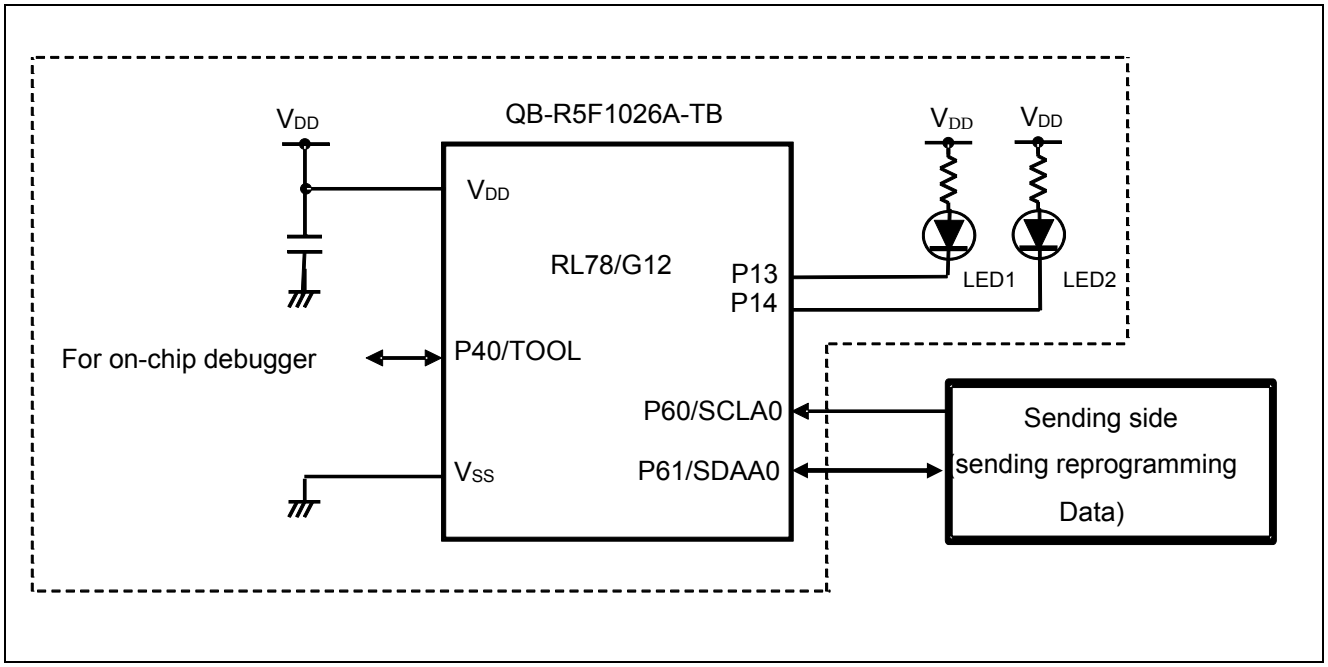

Figure3.1 Hardware Configuration

Cautions: 1. The purpose of this circuit is only to provide the connection outline and the circuit is simplified accordingly. When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to V_{DD} or V_{SS} via a resistor).

2. V_{DD} must be held at not lower than the reset release voltage (V_{LVD}) that is specified as LVD.

4.2 List of Pins to be Used

Table 3.1 lists pins to be used and their functions.

Table 3.1 Pins to be Used and their Functions

Pin name	I/O	Function
P13	Output	LED1 control
P14	Output	LED2 control
P60	Input/Output	IICA0 serial clock I/O pin
P61	Input/Output	IICA0 serial data transmission/reception pin

5. Description of the Software

5.1 Communication Specifications

The sample program covered in this application note receives data via the IIC bus for flash memory self-programming. The sending side sends three commands, i.e., the START, WRITE, and END commands. The sample program takes actions according to the command it received. If the command terminates normally, the sample program releases the IIC bus from the wait state and receives the next command. If the command terminates abnormally, the sample program turns on LED1 and LED2 with the IIC bus placed in the wait state and suppresses the execution of the subsequent operations. This section describes the necessary IIC communication settings and the specifications for the commands.

Table 4.1 IIC Communication Settings

Local address	0xA0
Operation mode	Standard (100 KHz)

5.1.1 START Command

When the sample program receives the START command, it places the IIC bus in the wait state and performs initialization processing for flash memory self-programming. When the command terminates normally, the program releases the IIC bus from the wait state. In the case of an abnormal termination, the sample program displays "ERROR!" on the LCD and suppresses the execution of the subsequent operations.

START code (0x01)	Data length (0x0002)	Command (0x02)	Data (None)	Checksum (1 byte)

5.1.2 WRITE Command

When the sample program receives the WRITE command, it places the IIC bus in the wait state, writes the data it received into flash memory, and performs verify processing each time it completes the write of one block. The sample program releases the IIC bus from the wait state on normal termination of the command. In the case of an abnormal termination, the sample program displays "ERROR!" on the LCD and suppresses the execution of the subsequent operations.

START code (0x01)	Data length (0x0102)	Command (0x03)	Data (256 bytes)	Checksum (1 byte)

5.1.3 END Command

When the sample program receives the END command, it places the IIC bus in the wait state and performs verify processing on the block that is currently being written. If the verification terminates normally, the program inverts the state of the boot flag, then generates a reset for boot swapping. In the case of an abnormal termination, the sample program displays "ERROR!" on the LCD and suppresses the execution of the subsequent operations.

START code (0x01)	Data length (0x0002)	Command (0x04)	Data (None)	Checksum (1 byte)

* The checksum is the sum of the command and data fields in units of bytes.

5.1.4 Communication Sequence

This sample program takes actions according to the sequence described below upon receipt of a command from the sending side.

(1) Sending side:

Sends the START command.

(2) Sample program:

Places the IIC bus in the wait state and turns on LED1 which indicates that flash memory is being accessed. The program then performs initialization for flash memory self-programming and releases the IIC bus from the wait state upon normal termination.

(3) Sending side:

Sends the WRITE command and data (4 bytes).

(4) Sample program:

Places the IIC bus in the wait state and writes the data (4byte) it received into the code flash memory. The write address starts at 0x3BFC and ends at 0x3BFF. When all of these steps terminate normally, the sample program releases the IIC bus from the wait state.

(5) Sending side:

(6) Sample program:

Performs verify processing on the block that is currently subjected to reprogramming with the IIC bus placed in the wait state and turns off LED2 to indicate that flash memory is not being accessed.

5.2 Operation Outline

This application note explains a sample program that performs flash memory reprogramming using a self-programming library.

The sample program reads values from the code flash memory addresses 0x3BFC to 0x3BFF and sets the flashing interval of LED1 with the read value. Subsequently, the program receives data (4 bytes) from the sending side and carries out self-programming to rewrite the values that are stored in code flash memory addresses 0x3BFC to 0x3BFF with the received data. When reprogramming is completed, the sample program reads again the values that are stored in code flash memory addresses 0x3BFC to 0x3BFF and sets the flashing interval of LED1 with the read value.

LED1 flashes at the interval that is equal to the average value of the data (4 bytes) received from the sending side (sum of byte values stored in code flash memory addresses 0x3BFC to 0x3BFF divided by 4) × 10 [ms]. For example, if address 0x3BFC contains a value of "15," address 0x3BFD contains "150," address 0x3BFE contains "100," and address 0x3BFF contains "200," according to the calculation (15 + 150 + 100 + 200) / 4 * 10 = 1162.5, LED1 flashes at intervals of 1162.5 [ms].

LED2 indicates that flash memory is being accessed when it is on.

(1) Sets up the I/O port

<Setting conditions>

- LED on/off control ports (LED1 and LED2): Sets P13 and P14 for output.

(2) Sets up the serial interface IICA

<Setting conditions>

- Sets the operation mode to standard.
- Sets the transfer clock to 100 kHz.
- Sets the local address to 0xA0.
- Sets up interrupts so that an interrupt occurs on every ninth clock.
- Disables interrupt requests to be generated on detection of the stop condition.
- Sets the P60/KR4/SCLA0 pin as transfer clock I/O pin and the P61/KR5/SDAA0 pin as pin for data transmission/reception.

(3) Initializes the TAU0 channel 0

<Setting conditions>

- Sets operation clock 0 (CK00) of the TAU0 to 23.44 [KHz], operation clock 1 (CK01) to 24 [MHz], operation clock 2 (CK02) to 12 [MHz], and operation clock 3 (CK03) to 93.75 [KHz].
- Sets the operation clock to operation clock 0 (CK00).
- Enables only software trigger start as the start trigger.
- Sets the operation mode to the interval timer mode in which no timer interrupt occurs at the beginning of counting.

(4) Enables interrupts.

(5) Reads values from code flash memory addresses 0x3BFC to 0x3BFF, calculates an average of the values in addresses 0x3BFC to 0x3BFF, and turns on LED1.

(6) If the read values are greater than 0, sets the interval time of the TAU0 channel 0 to the average value of values in addresses 0x3BFC to 0x3BFF × 10 [ms] and starts the TAU0 channel 0.

(7) Enters the HALT mode and waits for data from the sending side. The program enters the HALT mode again if it returns from the HALT mode upon a TAU0 channel 0 interrupt request.

(8) Switches into the normal operation mode from the HALT mode upon an IICA transmission end interrupt request.

(9) Disables interrupts.

(10) Upon receipt of an address and transfer direction information from the sending side, places the IIC bus in the wait state and checks the transfer direction.

- When the master sends data to the slave, it clears the receive end interrupt request flag and releases the IIC bus from the wait state.
- When the master receives data from the salve, it turns on LED1 and LED2 and suppresses the execution of the subsequent operations.

(11) If the LED is flashing, this is stopped by stopping the operation of channel 0 in TAU0.

(12) Upon receipt of a START command (0x02) from the sending side, places the IIC bus in the wait state and performs initialization for self-programming.

- Sets P14 to the low level to turn on LED2, indicating that flash memory is being accessed.
- Calls the FSL_Init function to initialize the flash memory self-programming environment and makes the following settings:

  Voltage mode: Full-speed mode

  CPU operating frequency: 24 [MHz]

  Status check mode: Status check internal mode
- Calls the FSL_Open function to start flash memory self-programming (starting the flash memory environment).
- Calls the FSL_PrepareFunctions function to make available the flash memory functions (standard reprogramming functions) that are necessary for the RAM executive.
- Calls the FSL_GetFlashShieldWindow function to get the start and end blocks of the flash shield window.
- If the start block of the flash shield window is a block other than block 0 or if the end block is a block other than block 15, calls the FSL_SetFlashShieldWindow function to set the start block of the flash shield window to block 0 and the end block to block 15.

(13) Releases the IIC bus from the wait state and notifies the sending side of the transmission-enabled state.

(14) Receives the WRITE command (0x03) and reprogramming data (4 bytes).

(15) Places the IIC bus in the wait state and computes the reprogramming target block from the write destination address.

(16) Calls the FSL_BlankCheck function to check whether the reprogramming target block has already been reprogrammed.

(17) If the reprogramming target block is reprogrammed, calls the FSL_Erase function to erase the reprogramming target block.

(18) Calls the FSL_Write function to write the received data at the write destination address.

(19) Releases the IIC bus from the wait state and notifies the sending side of the transmission-enabled state.

(20) Receives an END command (0x04).

(21) Calls the FSL_IVerify function to verify the reprogramming target block.

(22) Calls the FSL_IVerify function to verify the reprogramming target block.

(23) Releases the IIC bus from the wait state and notifies the sending side of the transmission-enabled state.

(24) Returns to step (4).

Caution When flash memory self-programming could not be terminated normally (error occurring during processing), the sample program turns on LED1 and LED2 and suppresses the execution of the subsequent operations.

5.3 File Configuration

Table 4.2 lists the additional functions for files that are automatically generated in the integrated development environment and other additional files.

Table 4.2 List of Additional Functions and Files

File Name	Outline	Remarks
r_main.c	Main module	Additional functions: R_MAIN_PacketAnalyze R_MAIN_SelfInitialize R_MAIN_SelfExecute R_MAIN_WriteExecute
r_cg_serial_user.c	SAU module	Additional functions: R_IICA0_CheckTransferDirection R_IICA0_ReceiveStart
lcd.c	DebugLCD module	Controls DebugLCD included in RL78/G12 target board.

5.4 List of Option Byte Settings

Table 5.3 summarizes the settings of the option bytes.

Table 5.3 Option Byte Settings

Address	Setting	Description
000C0H/010C0H	11101111B	Disables the watchdog timer. (Stops counting after the release from the reset status.)
000C1H/010C1H	01111111B	LVD reset mode 2.81 V (2.76 V to 2.87 V)
000C2H/010C2H	11101000B	HS mode, HOCO: 32 MHz
000C3H/010C3H	10000100B	Enables the on-chip debugger Erases the data in the flash memory when on-chip debug security ID authentication fails.

The option bytes of the RL78/G12 comprise the user option bytes (000C0H to 000C2H) and on-chip debug option byte (000C3H).

The option bytes are automatically referenced, and the specified settings are configured at power-on time or the reset is released.

The option bytes must be specified from "User Option Byte Value" on the "Device" panel in the "Link Options" of CS+. Set "Set up user option bytes" to "Yes (-gb)."

5.5 Link Directive File

The reprogramming program that performs flash memory self-programming and the flash memory self-programming library are allocated to blocks 0 to 3 (boot cluster 0) by the link directive file. It is also used to make configuration so that the RAM area to be used by the flash memory self-programming library will not be used.

The outline of the link directive file that this sample program uses is shown below.

```
;*************************************************************
; Redefined ROM area
;*************************************************************
; ---------------------------------------------------------
; Define new memory entry for boot cluster 0
; ---------------------------------------------------------
MEMORY ROM          : ( 000000H, 001000H )
; ---------------------------------------------------------
; Define new memory entry for write-data area
; ---------------------------------------------------------
MEMORY OCDROM       : ( 00FE00H, 000200H )

;*************************************************************
; Library(fsl.lib) segment
;*************************************************************
; ---------------------------------------------------------
; Merge FSL_FCD segment
; ---------------------------------------------------------
MERGE FSL_FCD    := ROM
; ---------------------------------------------------------
; Merge FSL_FECD segment
; ---------------------------------------------------------
MERGE FSL_FECD := ROM
; ---------------------------------------------------------
; Merge FSL_RCD segment
; ---------------------------------------------------------
MERGE FSL_RCD    := ROM
; ---------------------------------------------------------
; Merge FSL_BCD segment
; ---------------------------------------------------------
MERGE FSL_BCD    := ROM
; ---------------------------------------------------------
; Merge FSL_BECD segment
; ---------------------------------------------------------
MERGE FSL_BECD := ROM

;*************************************************************
; Redefined RAM area
;*************************************************************
; ---------------------------------------------------------
; Define new memory entry for self-RAM
; ---------------------------------------------------------
MEMORY SELFRAM       : ( 0FEF00H, 00040AH )
; ---------------------------------------------------------
; Redefined default data segment RAM
; ---------------------------------------------------------
MEMORY RAM           : ( 0FF30AH, 000B16H )
; ---------------------------------------------------------
; Define new memory entry for saddr area
; ---------------------------------------------------------
MEMORY RAM_SADDR : ( 0FFE20H, 0001E0H )

;*************************************************************
; run-time library segment ( 0000H - FFFFH )
;*************************************************************
; ---------------------------------------------------------
; Merge @@LCODE,@@LCODEL(run-time library) segment
; ---------------------------------------------------------
MERGE @@LCODE    := ROM
MERGE @@LCODEL := ROM
;*************************************************************
; const segment
;*************************************************************
MERGE CNST_SEG := ROM
```

Code area definition
The code is allocated to the boot area.

OCD monitor area definition

Allocates the flash memory self-programming library to the boot area.

Defines so that the area to be used by the flash memory self-programming library will not be used as the standard RAM area.

Standard RAM area definition

Defines so that the area to be used by the flash memory self-programming library will not be used as the standard RAM area.

Allocates the run-time library to the boot area.

Allocates the const variable to the boot area.

5.6 List of Constants

Table 4.4 lists constants for the sample program.

Table 4.4 Constants for the Sample Program

Constant Setting Description	Constant Setting Description	Constant Setting Description
BAUDRATE	100	Transfer speed in units of kbps (100 kbps (normal mode))
FHOCO	24	HOCO oscillation frequency in units of MHz (24 MHz)
OPCLK	FHOCO/2	IICA operation clock in units of MHz (12 MHz)
DIICWL	(47 * OPCLK + 9)/10	Value set in IICWL0 register (57)
RISETIME	100	Signal rise time (100 ns)
FALLTIME	100	Signal fall time (100 ns)
WIDTHHIGH	5300 - (RISETIME + FALLTIME)	SCLA0 high-level width (5100 ns)
DIICWH	(WIDTHHIGH * OPCLK + 999)/1000	Value set in IICWH0 register (62)
CSLFADDR	0xA0	Slave address
BUFSIZE	9	Data buffer size
NORMAL_END	0x00	Normal termination
ERROR	0xFF	Abnormal termination
NO_RECEIVE	0x00	Command reception state: Not received
START_CODE	0x01	Command reception state: START code received
PACKET_SIZE	0x02	Command reception state: Data length received
START_CMD	0x02	START command
WRITE_CMD	0x03	WRITE command
END_CMD	0x04	END command
FULL_SPEED_MODE	0x00	Argument to flash memory self-programming library initialization function: Set operation mode to full-speed mode.
FREQUENCY_24M	0x18	Argument to flash memory self-programming library initialization function: RL78/G12's operating frequency = 24 MHz
INTERNAL_MODE	0x01	Argument to flash memory self-programming library initialization function: Turn on status check internal mode.
START_BLOCK_NUM	0x00	Start block number of flash shield window
END_BLOCK_NUM	0x0F	End block number of flash shield window
BLOCK_SIZE	0x400	One block size of code flash memory (1024 bytes)
WRITESIZE	0x01	Write data size (words)
WRITEADDR	0x3BFC	Write start address
WRITEBLOCK	WRITEADDR/400H	Read start address

Note Change the target address in the range of 0x3800 to 0x3BFC.

5.7 List of Variables

Table 4.5 shows the global variables.

Table 5.5 Global Variables

Type	Variable Name	Contents	Function Used
8 bits × 9 arrays	RRCVBUF	Buffer for data	main SRECVIICA0 SPACKETANALYZE
8 bits × 2 arrays	RLEN	Buffer for receiving the data length	SRECVIICA0
16 bits	RRCVLG	Size of receive data	SRECVIICA0 SPACKETANALYZE
8 bits	RINTIICFLAG	IICA receive interrupt flag	main, IINTIICA0, SCLRIICAFLAG
8 bits	RRECVSTATUS	IICA receive status	SRECVIICA0
8 bits	RLENCOUNT	Data length counter	SRECVIICA0
8 bits	RDATACOUNT	Data counter	SRECVIICA0
8 bits	RRECVDATA	1 byte of received data	SRECVIICA0
8 bits	RREADAVE	Average of the values in code flash memory at addresses 0x3BFC to 0x3BFF	main, SLEDBLINK
8 bits × 8 arrays	RARG	Array for use as FSL subroutine parameter	SFSLINIT, SFSLWRITEEXECUTE

5.8 List of Functions (Subroutines)

Table 4.6 summarizes the functions (subroutines).

Table 4.6 List of Functions (Subroutines)

Function name	Outline
SINIPORT	Makes initial settings of the ports.
SSTARTIICA0	Makes initial settings of IICA0 and puts it in communication standby state.
SINITAU	Makes initial settings of TAU.
SLEDBLINK	Makes the LEDs flash.
SSTARTINTV0	Starts the timer counting operation of TAU0.
SSTOPINTV0	Stops the timer counting operation of TAU0.
IINTTM00	Executes interrupt processing of TAU0.
IINTIICA0	Interrupt processing routine of INTIICA0
SCHKDIRIICA0	Checks direction of communication via IICA0.
SRECVIICA0	Receives data via IICA0.
SCLRIICAFLAG	Clears IICA0 receive end interrupt flag and IICA0 receive error interrupt flag.
SPACKETANALYZE	Analyzes receive data.
SFSLEXECUTE	Executes flash memory self-programming.
SFSLINIT	Executes initialization for flash memory self-programming.
SFSLWRITEEXECUTE	Executes flash memory reprogramming.

5.9 Function Specifications

This section describes the specifications for the functions that are used in the sample program.

[Function Name] SINIPORT

Synopsis	Make initial settings of the I/O ports.
Explanation	This function sets P13 and P14 for output.
Arguments	None
Return Value	None
Remarks	None

[Function Name] SSTARTIICA0

Synopsis	Make initial settings of IICA0.
Explanation	This function Initializes IICA0 to 0xA0 which is the slave address in the normal mode.
Arguments	None
Return Value	None
Remarks	None

[Function Name] IINTIICA0

Synopsis	IICA0 interrupt processing
Explanation	This function performs a routine which accepts and processes INTIICA0. Subroutine SINTIICA0 processes communication.
Arguments	None
Return Value	None
Remarks	None

[Function Name] SCHKDIRIICA0

Synopsis	Check direction of data transfer via IICA0.
Explanation	This function checks the direction in which data is being transferred.
Arguments	None
Return Value	C register ● Normal termination: NORMAL_END ● Transfer direction error (transfer direction is from slave to master): ERROR
Remarks	None

[Function Name] SRECVIICA0

Synopsis	Receive data via IICA0.
Explanation	This function stores the receive data in the receive buffer (RRCVBUF) and the receive data length [bytes] in RRCVLG.
Arguments	None
Return Value	C register ● Normal termination: NORMAL_END ● Abnormal termination: ERROR
Remarks	None

[Function Name] SCLRIICAFLAG

Synopsis	Clear IICA0 receive interrupt flag.
Explanation	This function clears the IICA0 receive end interrupt flag (RINTIICFLAG).
Arguments	None
Return Value	None
Remarks	None

[Function Name] SPACKETANALYZE

Synopsis	Analyze receive data.
Explanation	This function checks the parameters of the command received, and computes and compares the checksum to check whether the received data is correct.
Arguments	None
Return Value	C register ● START command received: START_CMD ● WRITE command received: WRITE_CMD ● END command received: END_CMD ● Abnormal termination: ERROR
Remarks	None

[Function Name] SINITAU

Synopsis	Make initial settings of TAU0.
Explanation	This function makes initial settings of TAU0.
Arguments	None
Return Value	None
Remarks	None

[Function Name] IINTTM00

Synopsis	TAU0 channel 0 interrupt
Explanation	This function inverts the state (ON/OFF) of LED1.
Arguments	None
Return Value	None
Remarks	None

[Function Name] SSTARTINTV0

Synopsis	Start TAU0 channel 0.
Explanation	This function starts the TAU0 channel 0.
Arguments	None
Return Value	None
Remarks	None

[Function Name] SSTOPINTV0

Synopsis	Stop TAU0 channel 0.
Explanation	This function stops the TAU0 channel 0.
Arguments	None
Return Value	None
Remarks	None

[Function Name] SLEDBLINK

Synopsis	Start LED flashing.
Explanation	This function sets the LED1 flashing interval to the value of RREADAVE × 10 [ms] and starts flashing LED1.
Arguments	None
Return Value	None
Remarks	None

[Function Name] SFSLEXECUTE

Synopsis	Execute flash memory self-programming.
Explanation	This function executes flash memory self-programming.
Arguments	None
Return Value	C register ● Normal termination: FSL_OK ● Parameter error: FSL_ERR_PARAMETER ● Erase error: FSL_ERR_ERASE ● Internal verify error: FSL_ERR_IVERIFY ● Write error: FSL_ERR_WRITE ● Flow error: FSL_ERR_FLOW
Remarks	None

[Function Name] SFSLINIT

Synopsis	Execute initialization for flash memory self-programming.
Explanation	This function executes initialization prior to flash memory self-programming.
Arguments	None
Return Value	C register ● Normal termination: FSL_OK ● Parameter error: FSL_ERR_PARAMETER ● Erase error: FSL_ERR_ERASE ● Internal verify error: FSL_ERR_IVERIFY ● Write error: FSL_ERR_WRITE ● Flow error: FSL_ERR_FLOW
Remarks	None

[Function Name] SFSLWRITEEXECUTE

Synopsis	Execute flash memory reprogramming.
Explanation	This function executes flash memory reprogramming.
Arguments	None
Return Value	C register ● Normal termination: NORMAL_END ● Abnormal termination: ERROR
Remarks	None

5.10 Flowcharts

Figure 4.1 shows the overall flow of the sample program described in this application note.

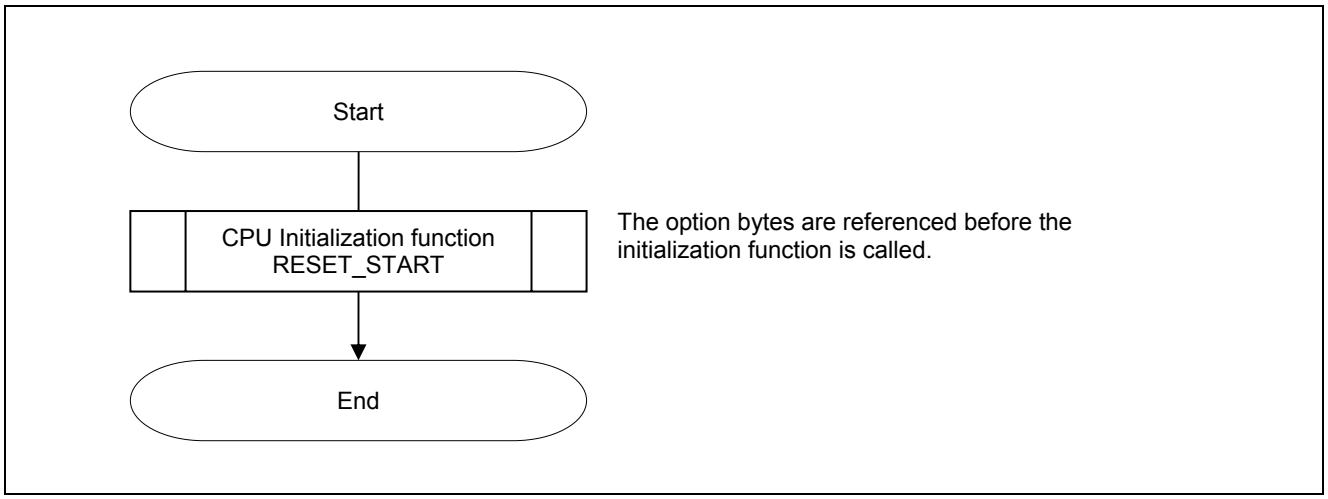

Figure 5.1 Overall Flow

5.10.1 Initialization Function

Figure 4.2 shows the flowchart for the initialization function.

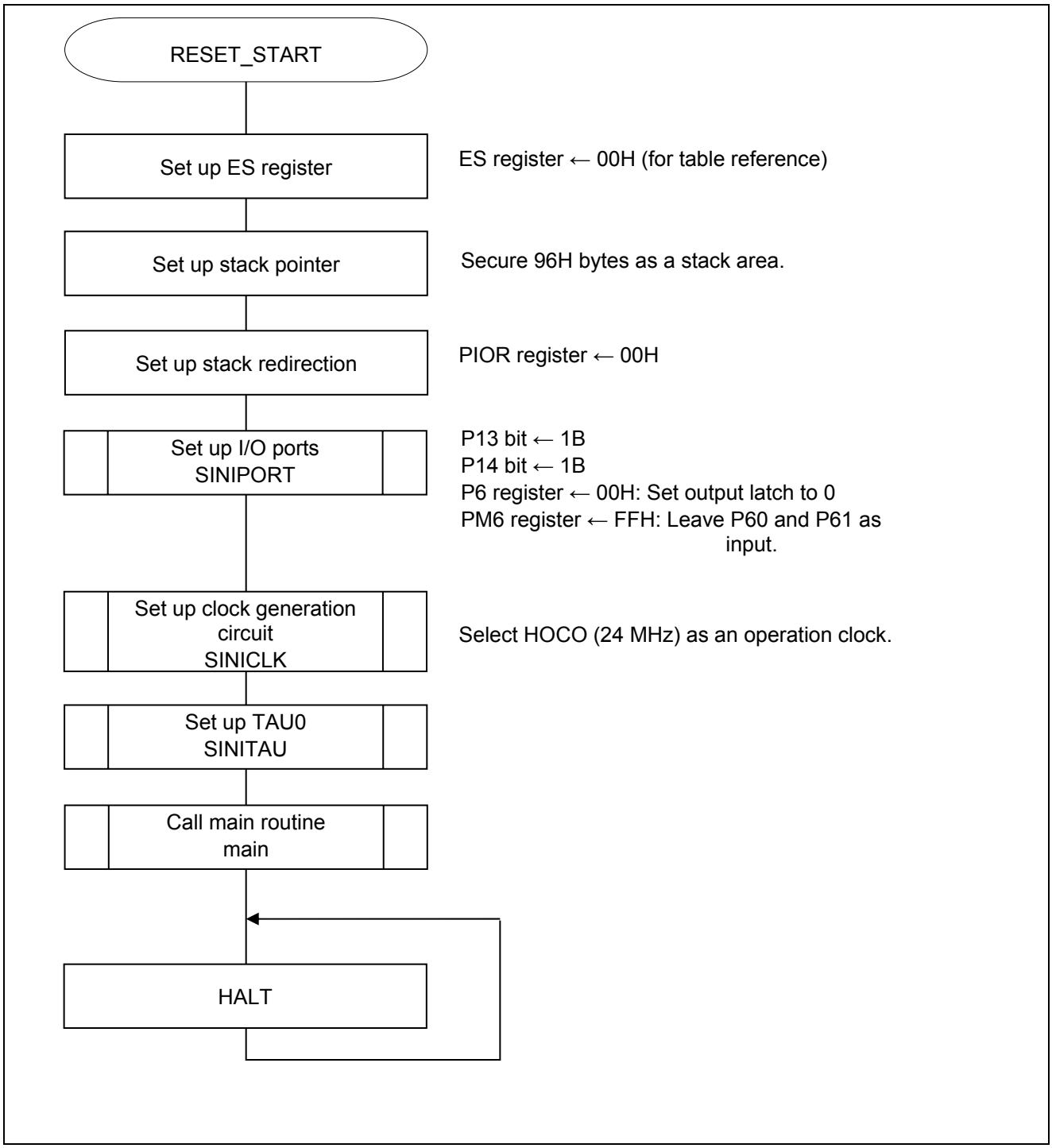

Figure 5.2 Initialization Function

5.10.2 I/O Port Initial Setup

Figure 4.3 shows the flowchart for I/O port initial setup.

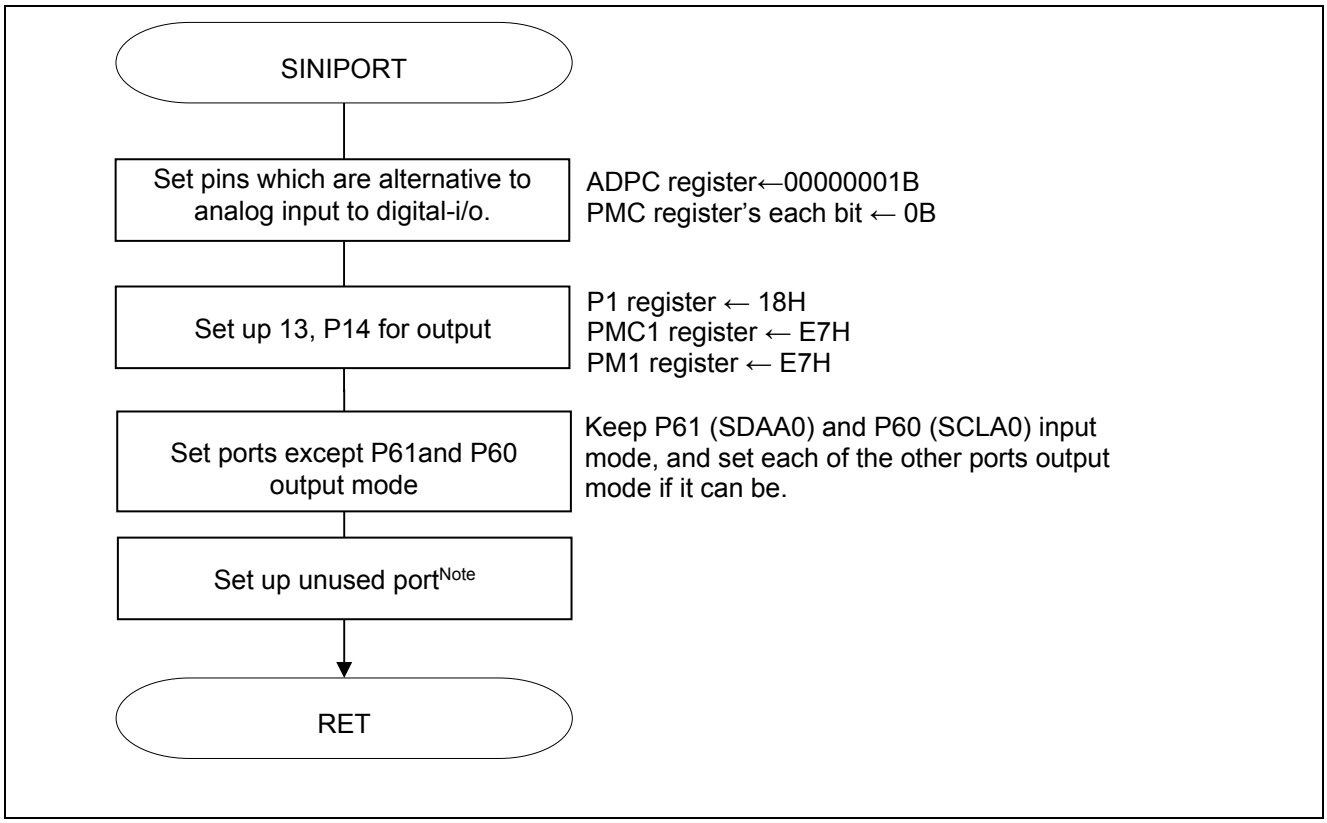

Figure 5.3 I/O Port Initial Setup

Note: Refer to the section entitled "Flowcharts" in RL78/G12 Initialization (R01AN1030E) Application Note for the configuration of the unused ports.

Caution: Provide proper treatment for unused pins so that their electrical specifications are observed. Connect each of any unused input-only ports to V_{DD} or V_{SS} via a separate resistor.

5.10.3 CPU Clock Setup

Figure 4.4 shows the flowchart for CPU clock setup.

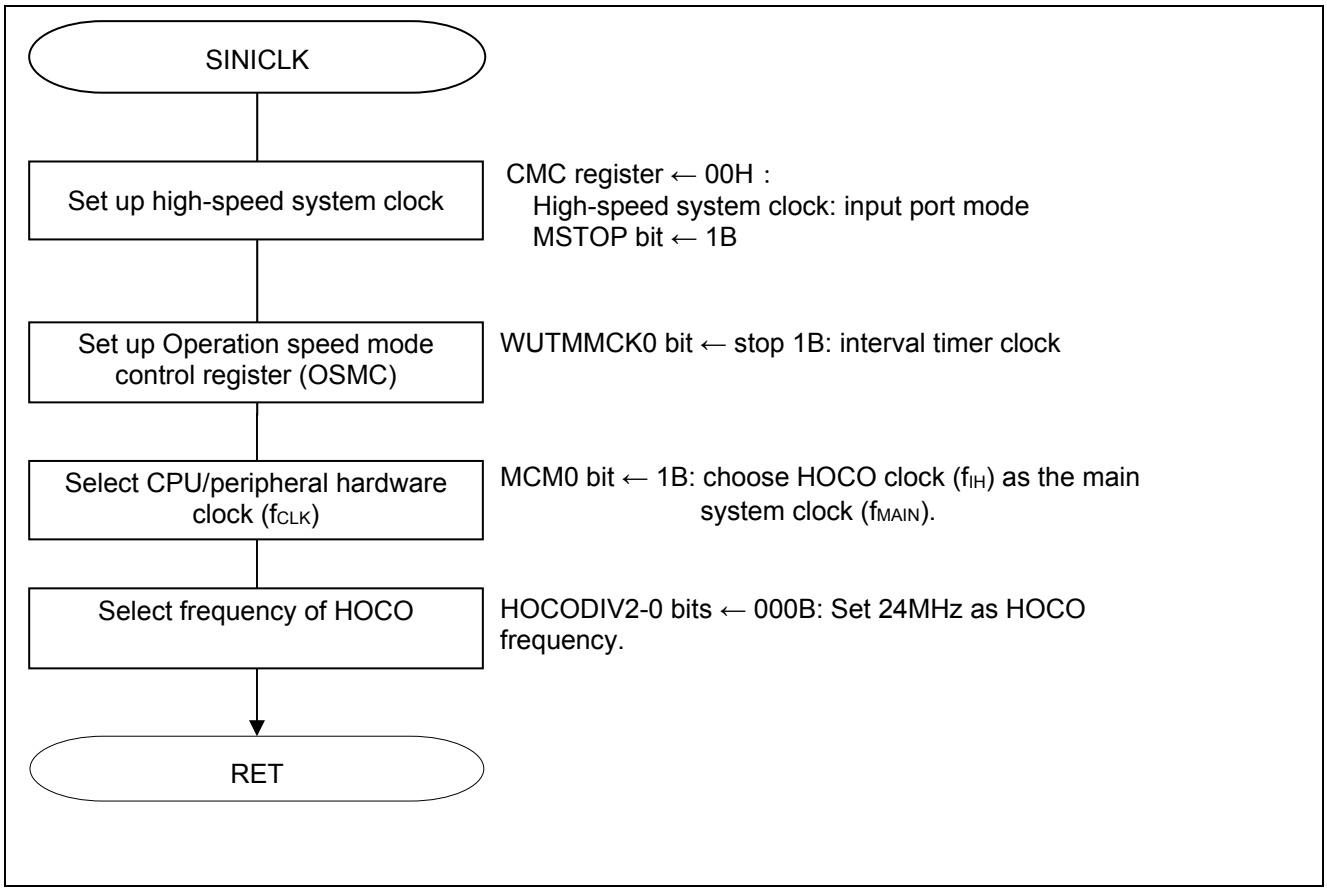

Figure 5.4 CPU Clock Setup

Caution: For details on the procedure for setting up the CPU clock (R_CGC_Create ()), refer to the section entitled "Flowcharts" in RL78/G12 Initialization (R01AN1030E) Application Note.

5.10.4 Serial Interface (IICA) Initial Setup

Figure 5.5 shows the flowchart for serial interface (IICA) setup (1/2). Figure 4.6 shows the flowchart for serial interface (IICA) setup (2/2).

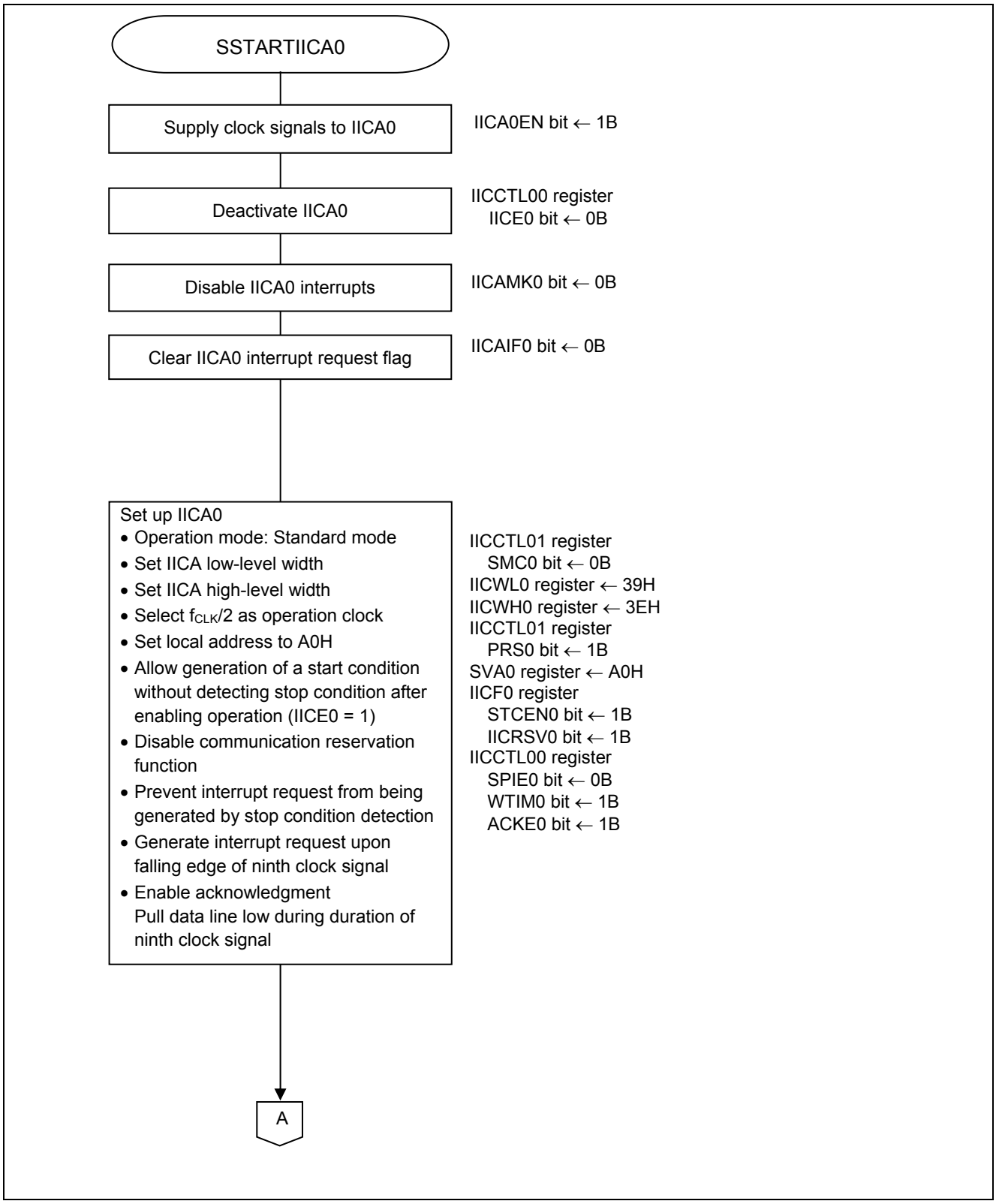

Figure 5.5 Serial Interface (IICA) Initial Setup (1/2)

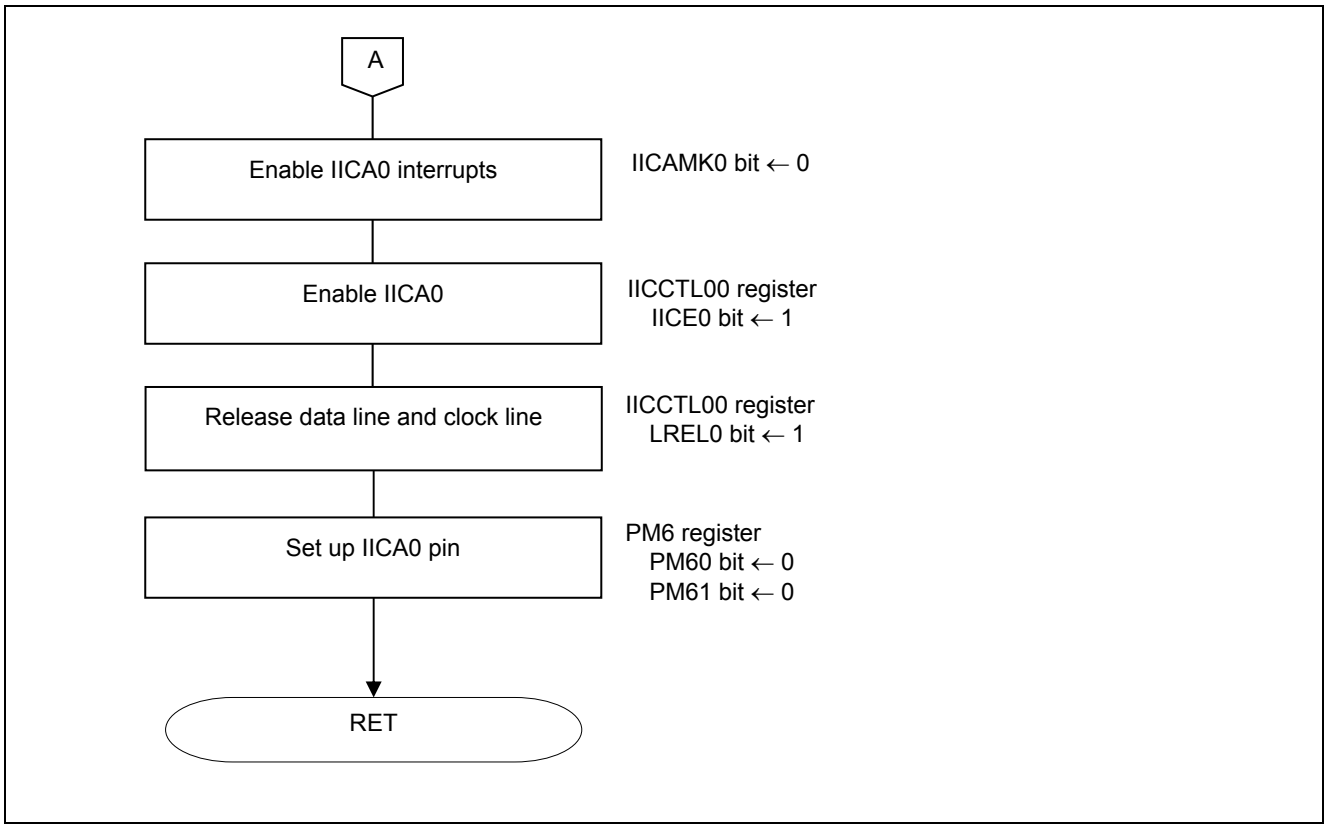

Figure 5.6 Serial Interface (IICA) Initial Setup (2/2)

5.10.5 Timer Array Unit 0 (TAU0) Initial Setup

Figure 4.7 shows the flowchart for timer array unit 0 (TAU0) initial setup.

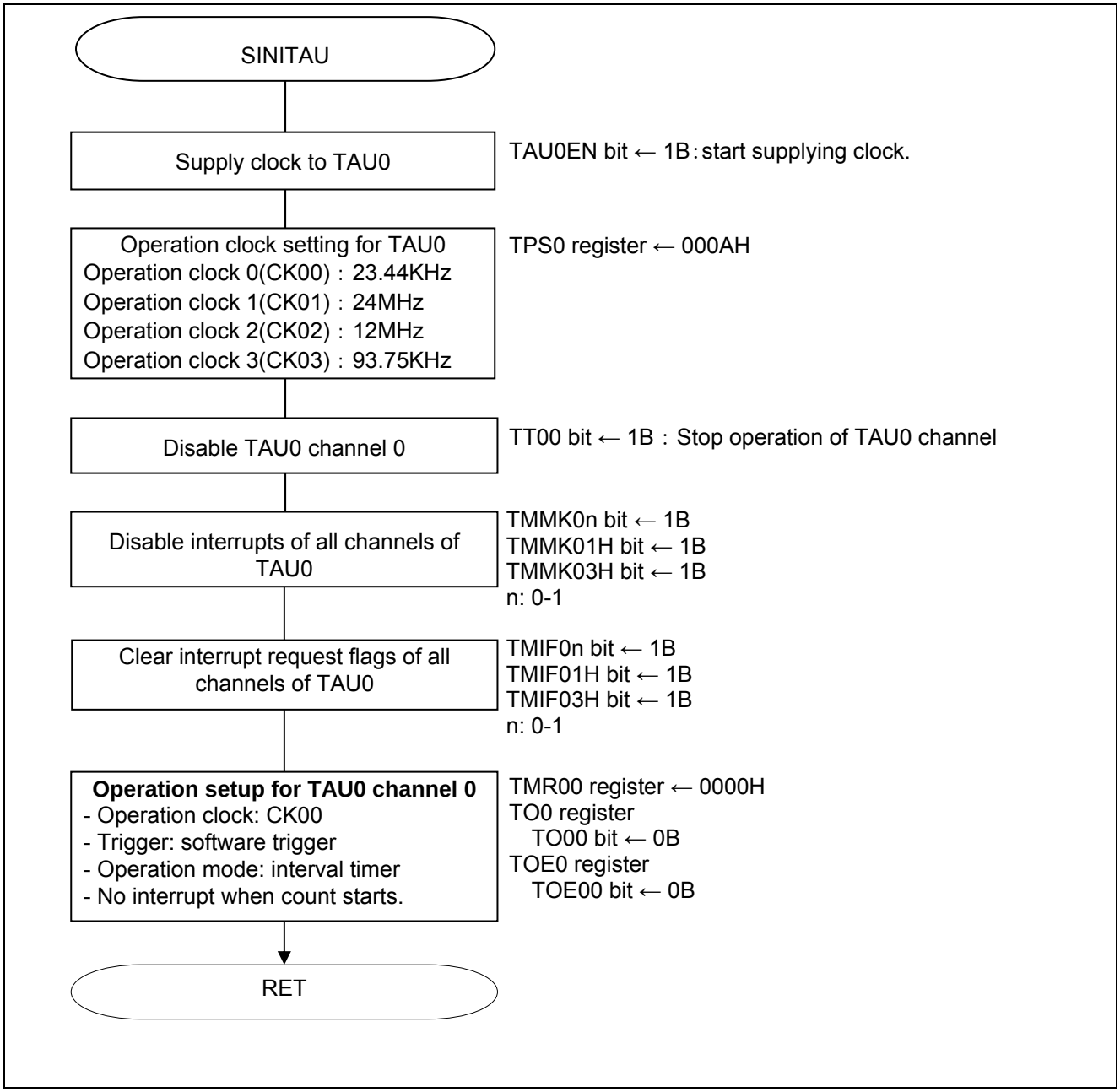

Figure 5.7 Timer Array Unit 0 (TAU0) Initial Setup

5.10.6 Main Processing

Figure 4.8 shows the flowchart for main processing (1/2). Figure 4.9 shows the flowchart for main processing (2/2).

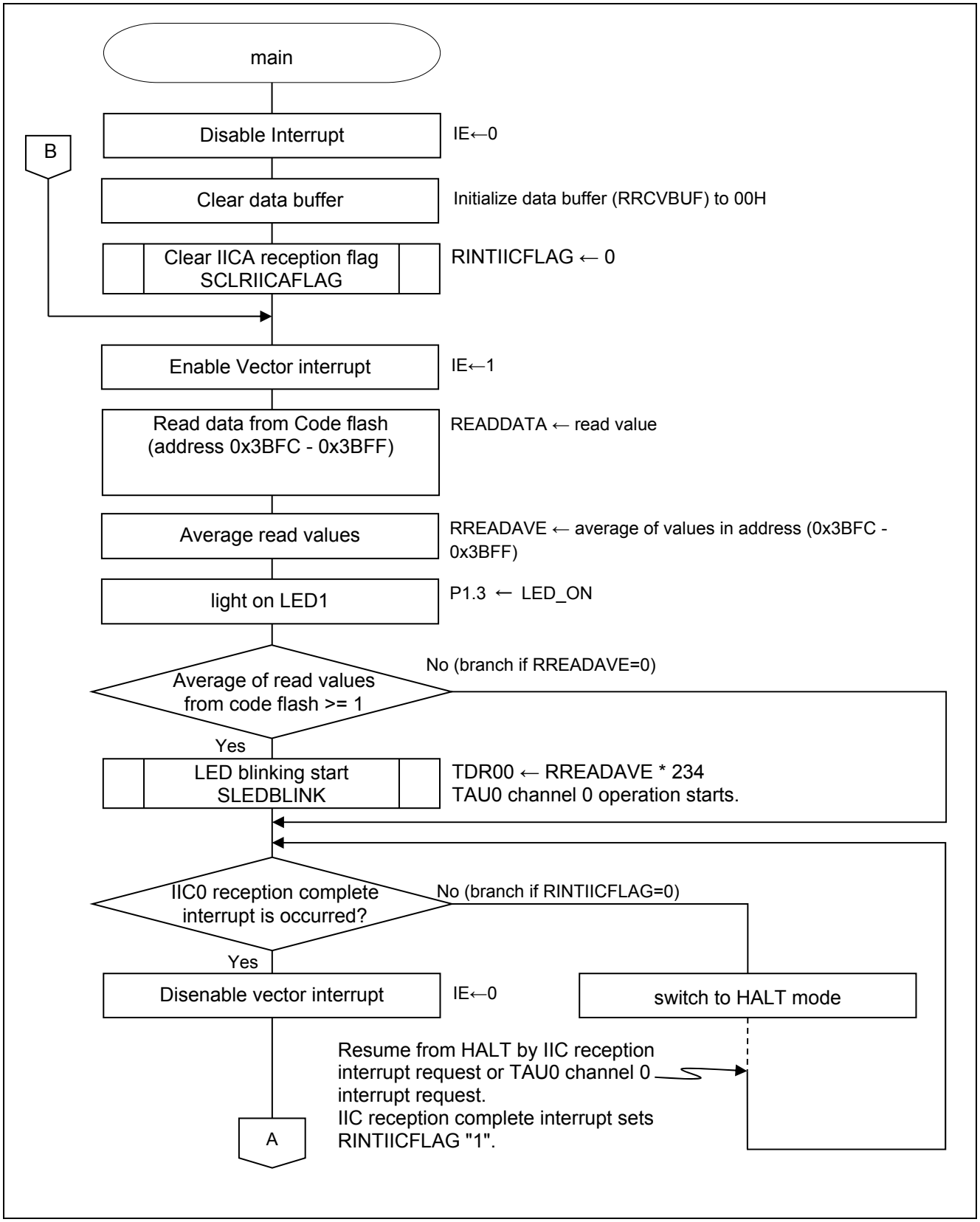

Figure 5.8 Main Processing (1/2)

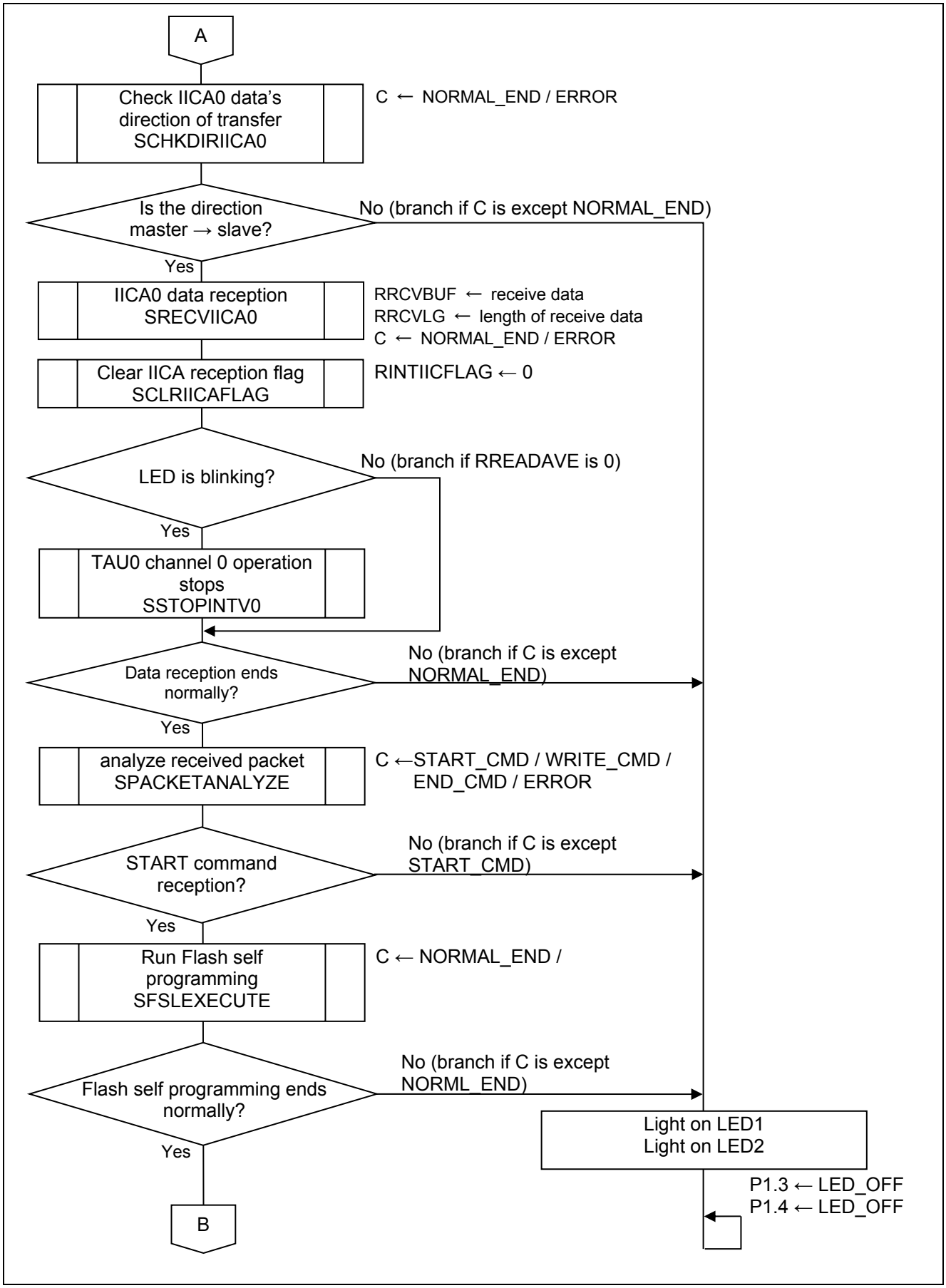

Figure 5.9 Main Processing (2/2)

5.10.7 LED blinking Start

Figure 4.10 shows the flowchart for starting LED blinking process.

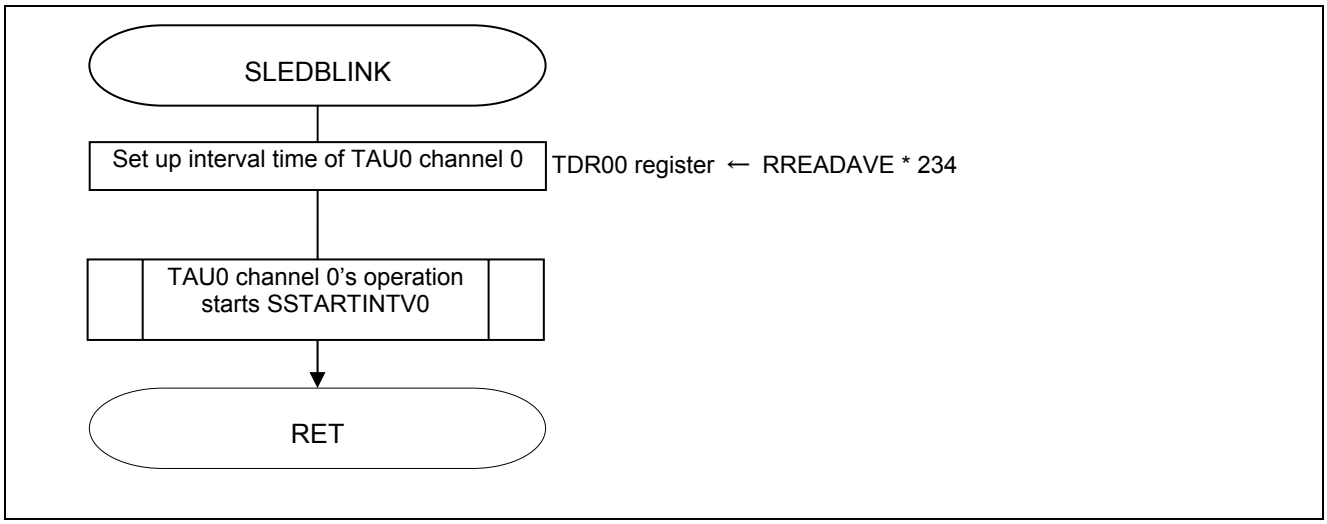

Figure 5.10 LED blinking start

5.10.8 TAU0 channel 0 operation start

Figure 4.11 shows the flowchart for TAU0 channel 0 operation start process.

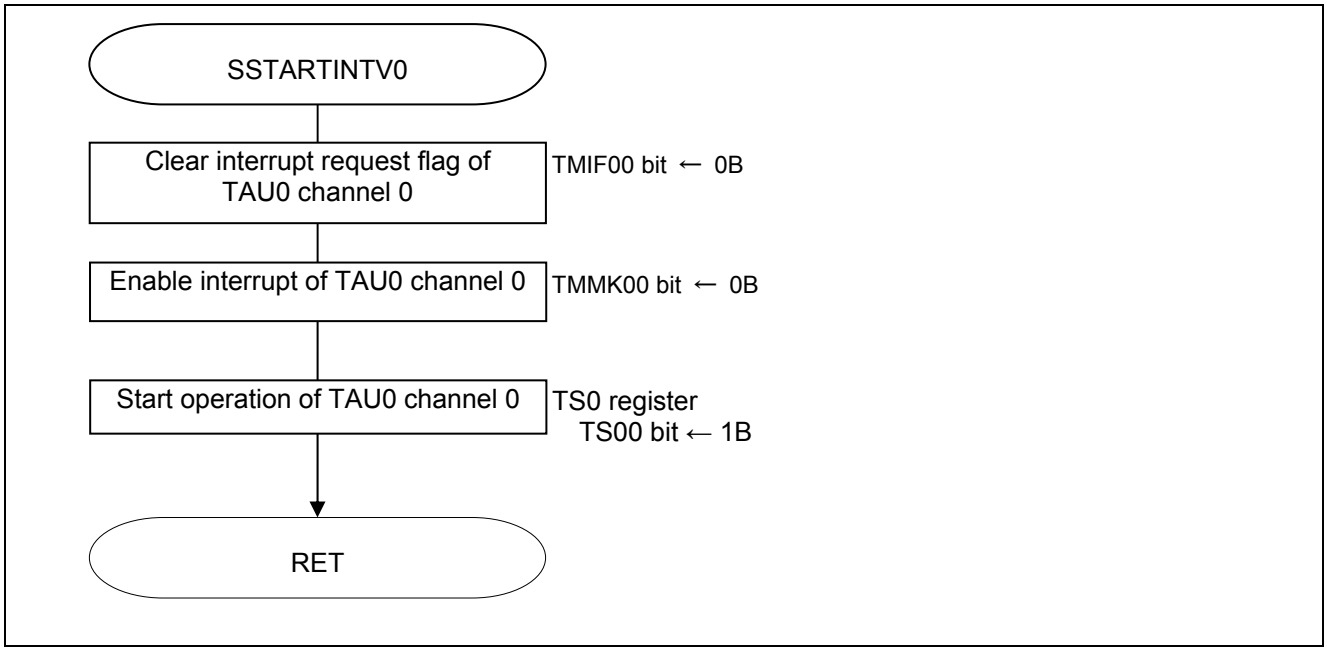

Figure 5.11 TAU0 channel 0 operation start

5.10.9 TAU0 channel 0 operation stop

Figure 4.12 shows the flowchart for TAU0 channel 0 operation stop process.

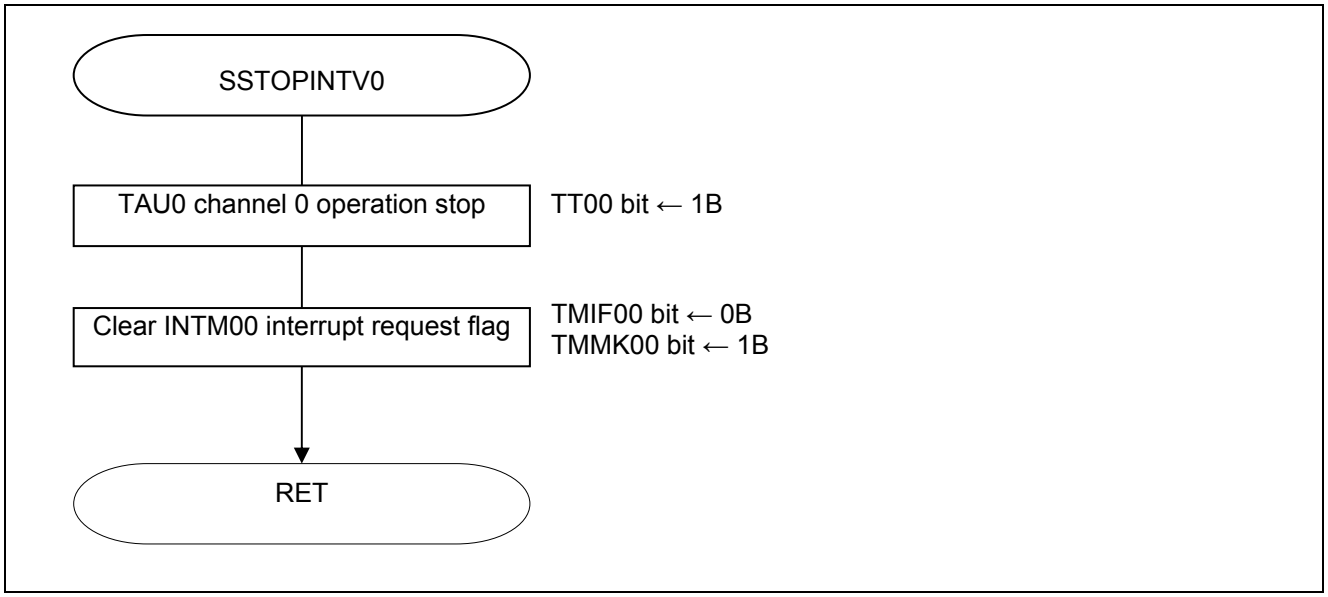

Figure 5.12 TAU0 channel 0 operation stop

5.10.10 TAU0 channel 0 interrupt

Figure 4.13 shows the flowchart of TAU0 channel 0 interrupt process.

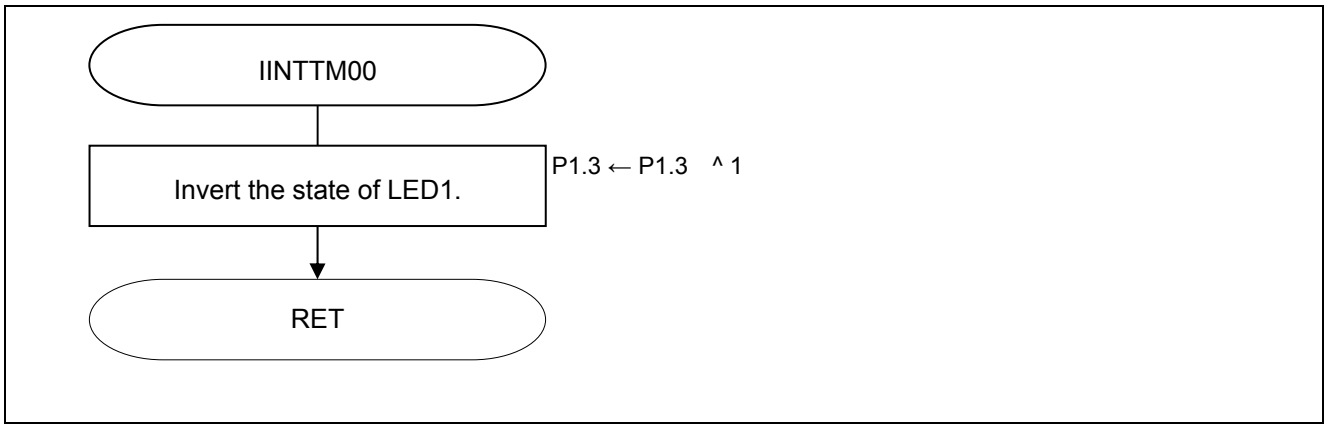

Figure 5.13 TAU0 channel 0 interrupt

5.10.11 IICA0 interrupt

Figure 4.14 shows the flowchart of IICA0 interrupt process.

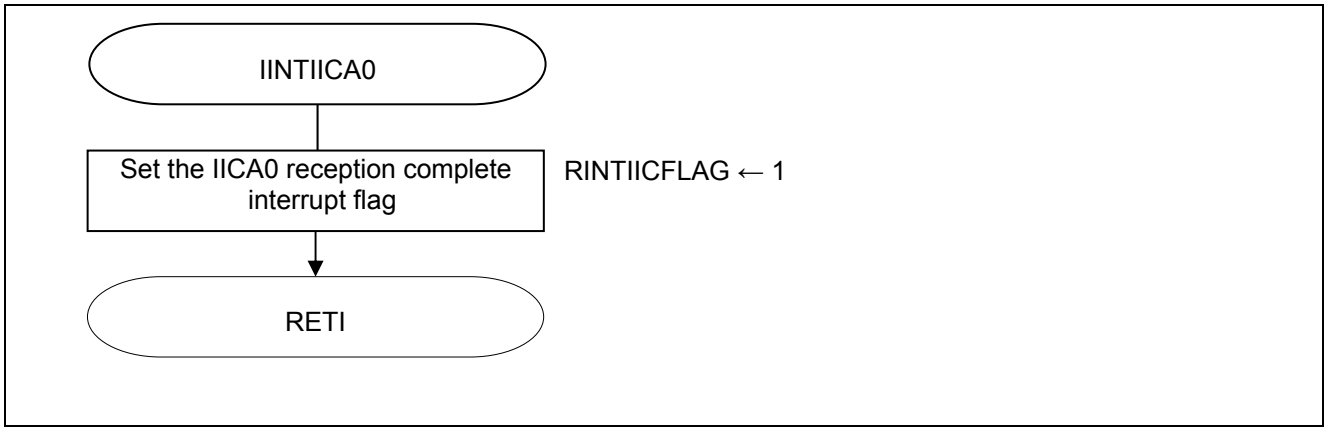

Figure 5.14 IICA0 interrupt

5.10.12 Checking the Direction of Data Transfer via IICA0

Figure 4.15 shows the flowchart for checking the direction of data transfer via IICA0.

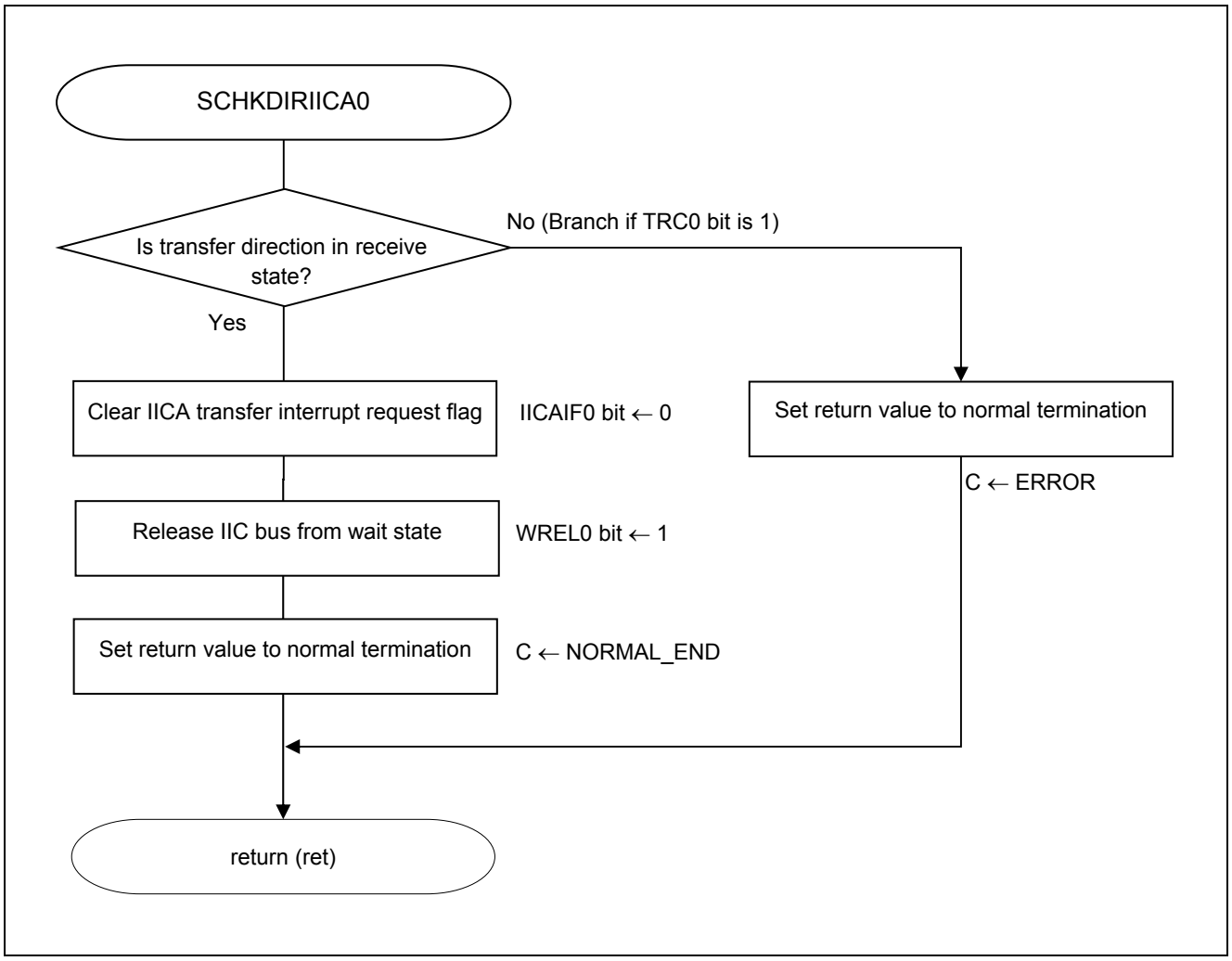

Figure 5.15 Checking the Direction of Data Transfer via IICA0

5.10.13 Data Reception via IICA0

Figure 4.16 shows the flowchart for data reception via the IICA0 (1/3). Figure4.17 shows the flowchart for data reception via the IICA0 (2/3). Figure 4.18 shows the flowchart for data reception via the IICA0 (3/3).

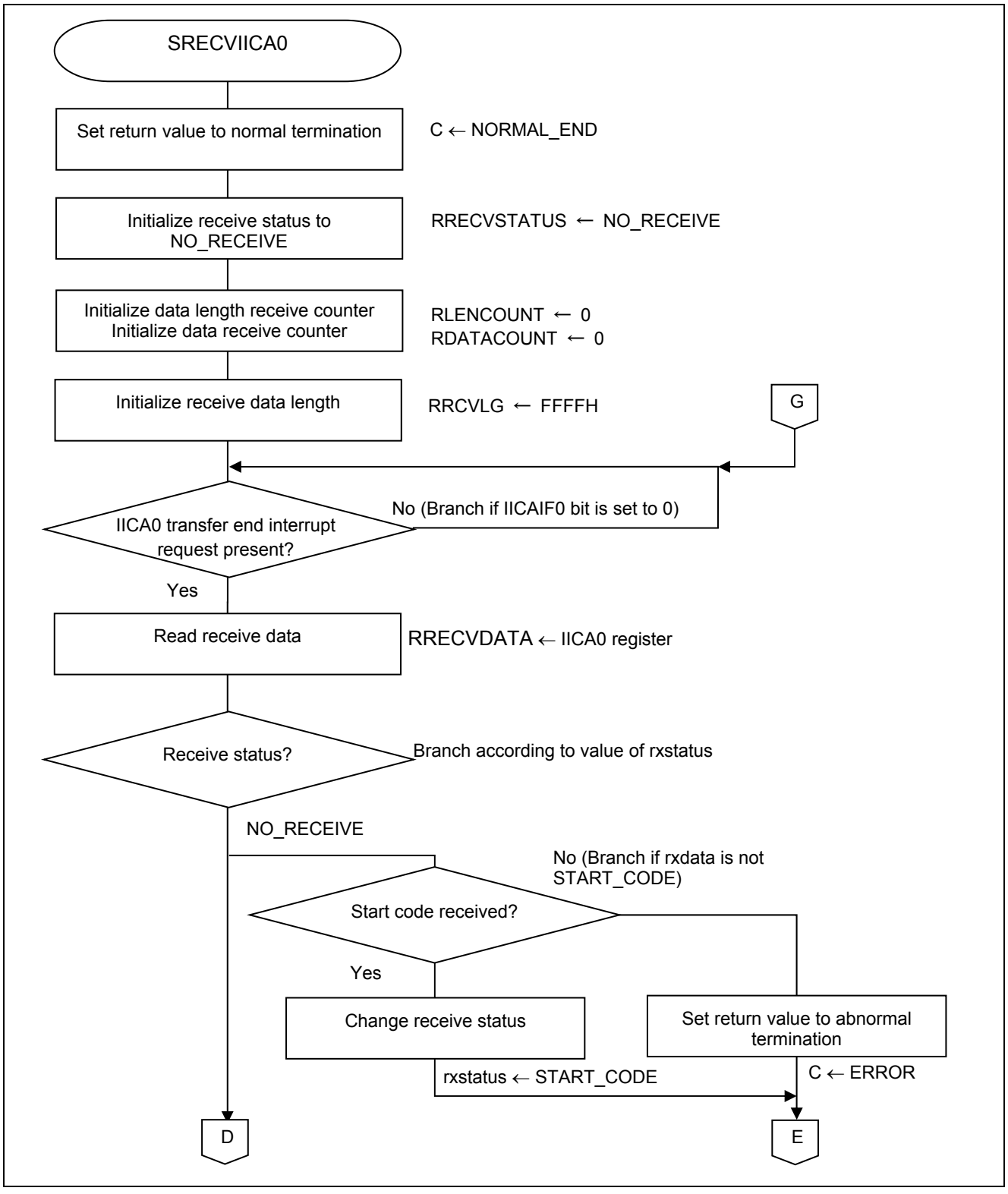

Figure 5.16 Data Reception via IICA0 (1/3)

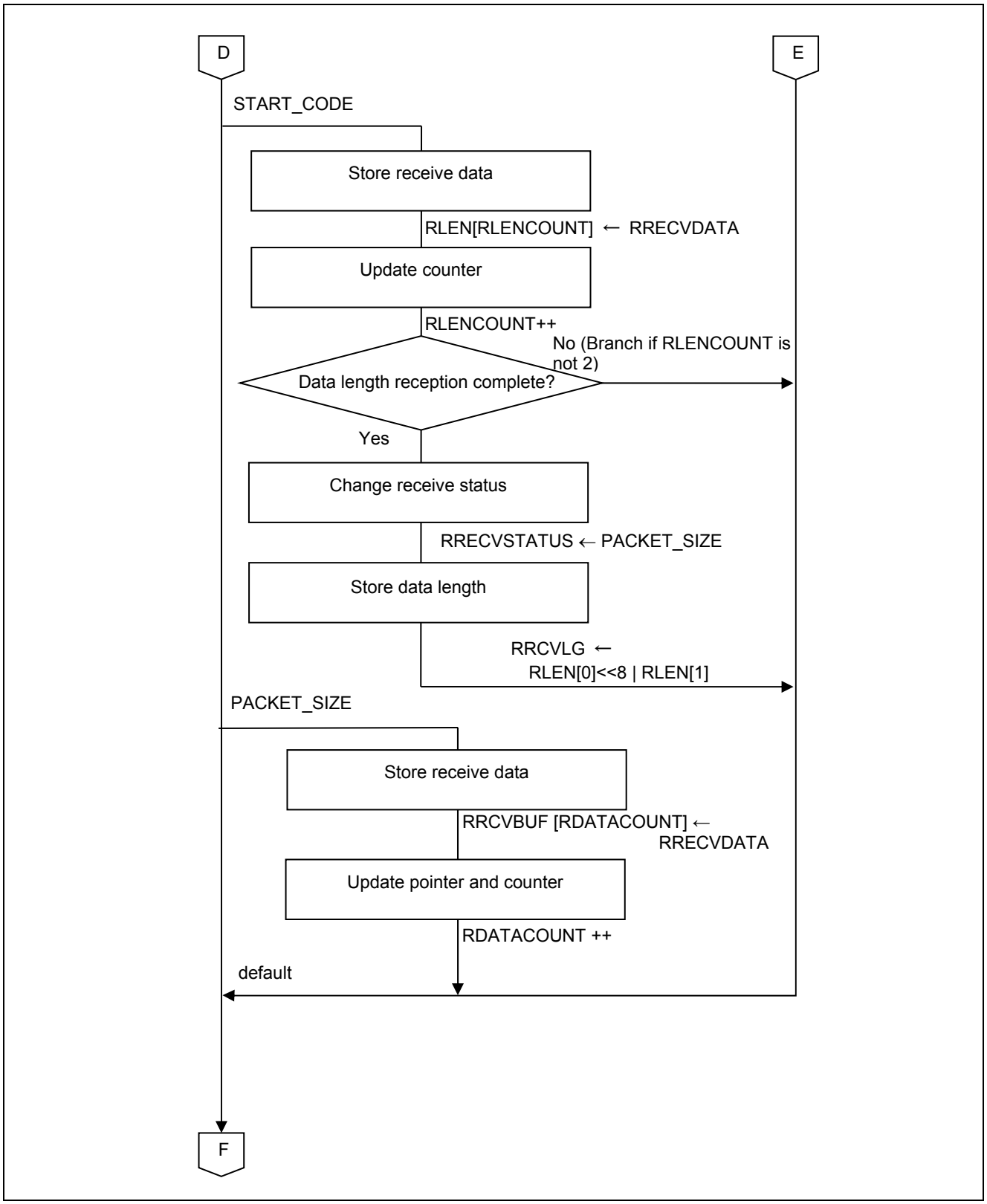

Figure 5.17 Data Reception via IICA0 (2/3)

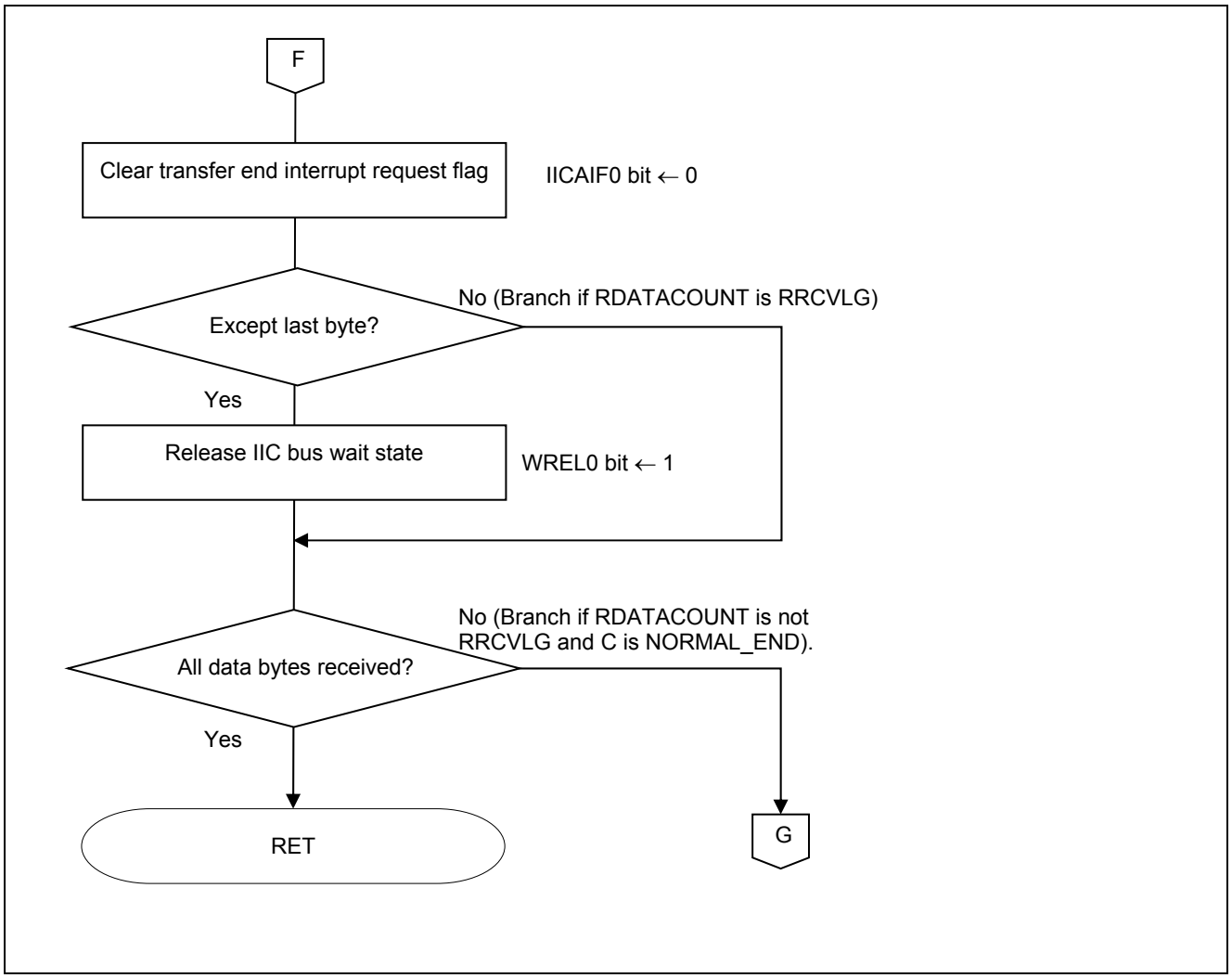

Figure 5.18 Data Reception via IICA0 (3/3)

5.10.14 Clear IICA reception interrupt flag

Figure 5.19 shows the flowchart for clearing the IICA0 reception interrupt flag.

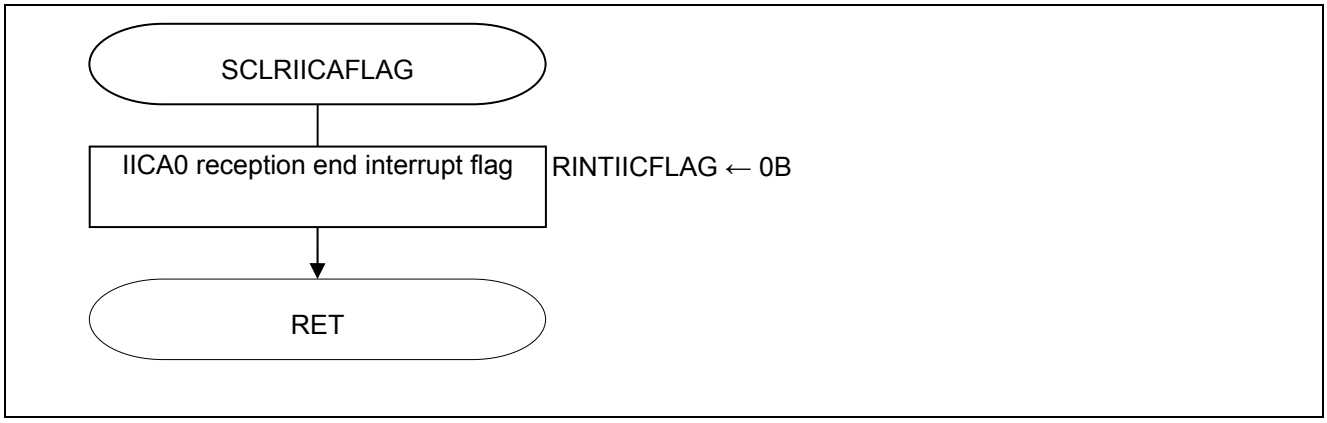

Figure 5.19 Clearing the IICA0 reception interrupt flag

5.10.15 Receive Packet Analysis

Figure 4.20 shows the flowchart for receive packet analysis.

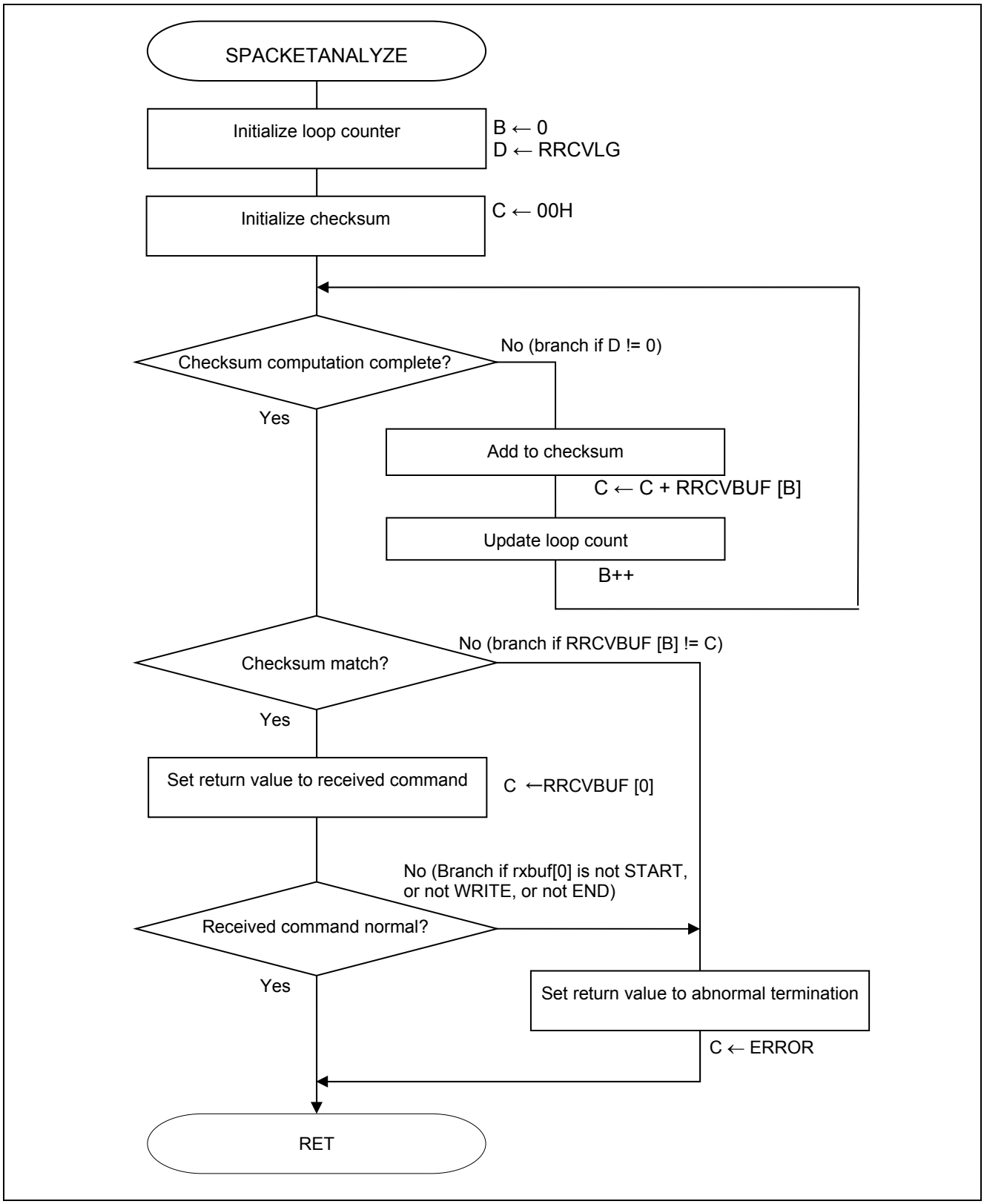

Figure 5.20 Receive Packet Analysis

5.10.16 Flash Memory Self-Programming Execution

Figure 4.21 shows the flowchart for flash memory self-programming execution.

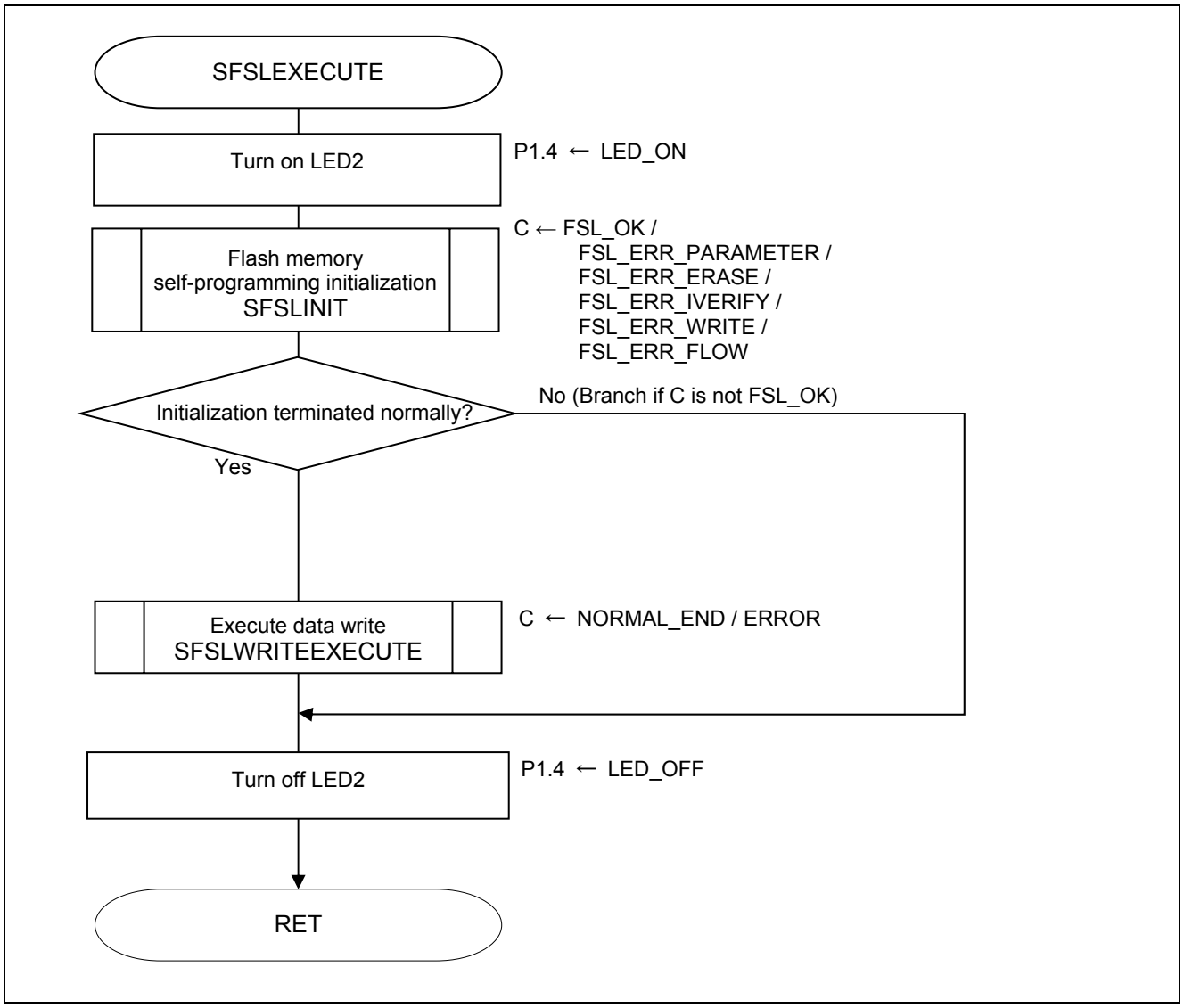

Figure 5.21 Flash Memory Self-Programming Execution

5.10.17 Flash Memory Self-Programming Initialization

Figure 4.22 shows the flowchart for flash memory self-programming initialization (1/2). Figure 4.23 shows the flowchart for flash memory self-programming initialization (2/2).

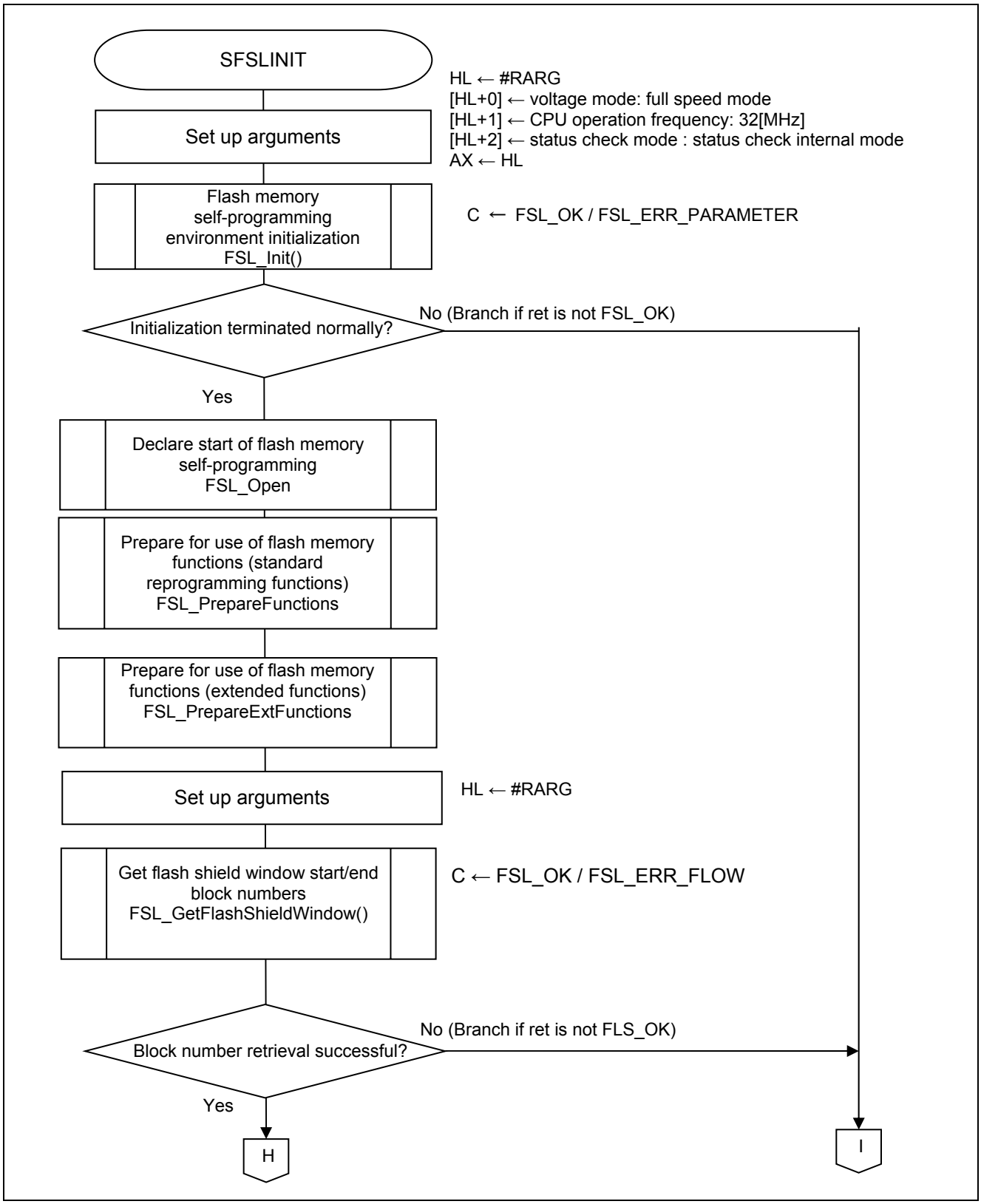

Figure 5.22 Flash Memory Self-Programming Initialization (1/2)

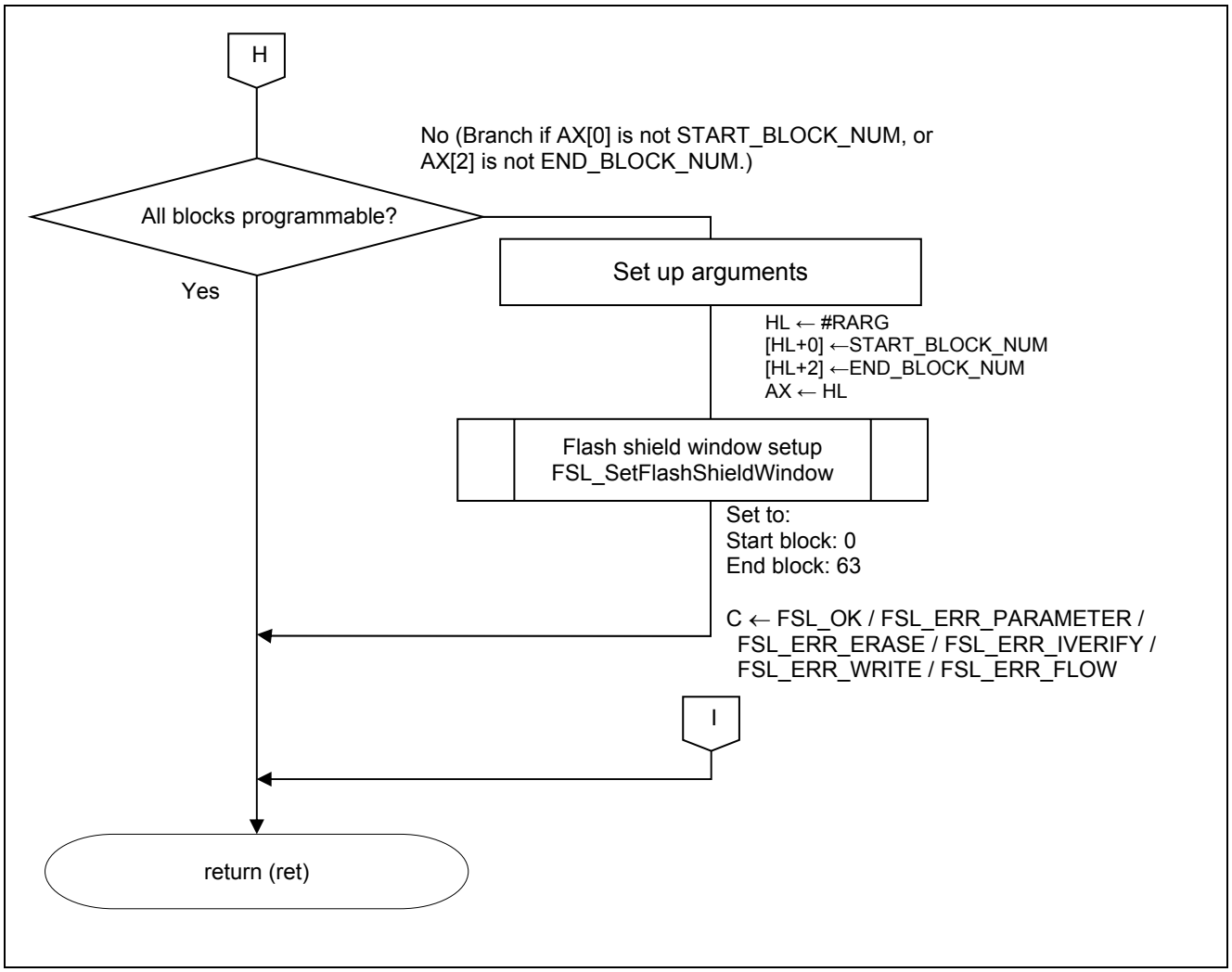

Figure 5.23 Flash Memory Self-Programming Initialization (2/2)

5.10.18 Flash Memory Reprogramming Execution

Figure 4.24 shows the flowchart for flash memory reprogramming execution (1/2). Figure 4.25 shows the flowchart for flash memory reprogramming execution (2/2).

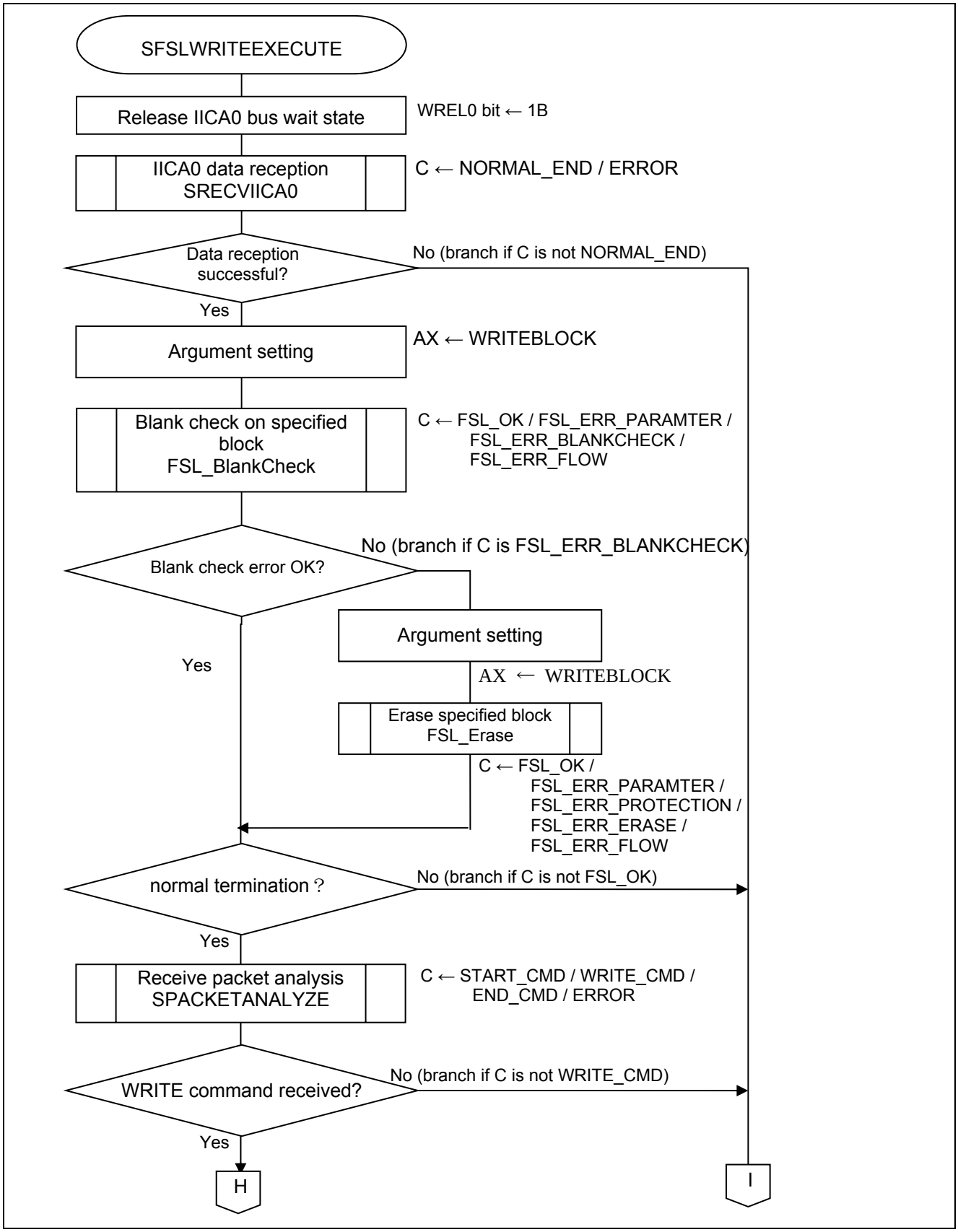

Figure 5.24 Flash Memory Reprogramming Execution (1/2)

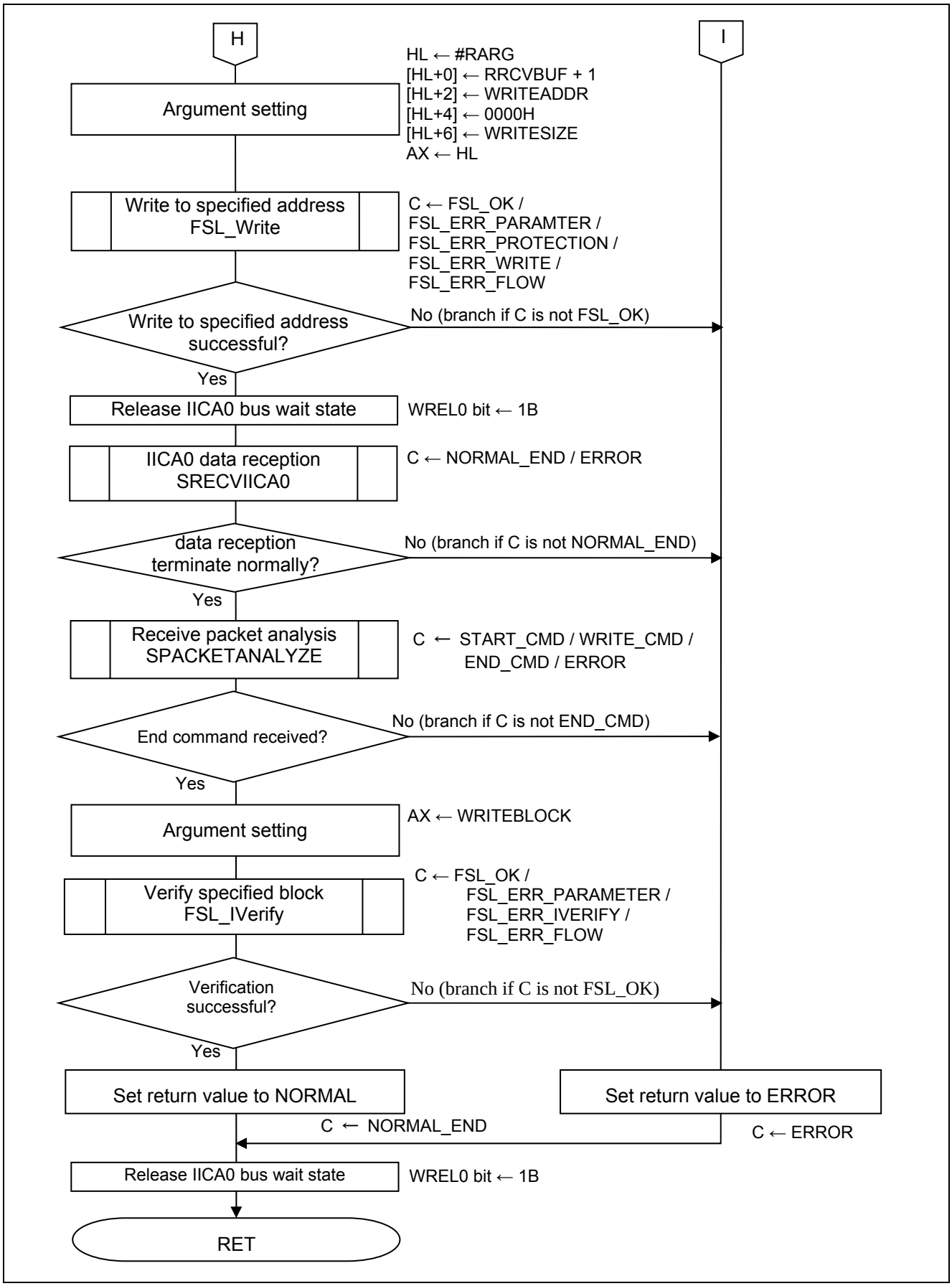

Figure 5.25 Flash Memory Reprogramming Execution (2/2)

6. Sample Code

The sample code is available on the Renesas Electronics Website.

7. Documents for Reference

RL78/G12 User's Manual: Hardware (R01UH0200E)

RL78 Family User's Manual: Software (R01US0015E)

RL78 family’s Flash Self Programming Library Type01 User’s Manual (R01US0050E)

(The latest versions of the documents are available on the Renesas Electronics Website.)

Technical Updates/Technical Brochures

(The latest versions of the documents are available on the Renesas Electronics Website.)

Website and Support

Renesas Electronics Website

- http://www.renesas.com/index.jsp

Inquiries

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

Revision Record	RL78/G12 Self-Programming (IIC)

Rev.	Date	Description	
		Page	Summary
1.00	Sep. 30, 2013	—	First edition issued
1.10	June 01, 2016	10	Modification of 1.4 How to Get the Flash Memory Self-Programming Library.
		50	Addition of Documents for Reference.

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
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

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 1207, 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 India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0