RL78/G23

Firmware Upgrade Using External Flash Memory via Simplified SPI (CSI) Communication

Introduction

This application note provides an overview of firmware upgrade using external flash memory.

The RL78/G23 can use simplified SPI (CSI) to communicate with external flash memory to obtain data for firmware upgrade from the memory. Use the Renesas Flash Driver (RFD) to write the obtained data into code flash memory for upgrading firmware.

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.

目次

1. Specifications

1.1 Outline

The sample program described in this application note starts by displaying the current version information of firmware on the LCD module. When a switch is pressed (an INTP0 interrupt occurs), the program lights an LED indicating that flash memory is being accessed. The MCU enters self-programming mode. The program obtains the data for upgrading firmware from the external flash memory and reprograms code flash memory with it. When reprogramming is completed, the program turns off the LED and resets the CPU. When the MCU is restarted, the program displays the new version information on the LCD module.

Table 1-1 Peripheral Functions to be Used and their Uses

Peripheral Function	Use
Serial array unit CSI11	Sends and receives data using simplified SPI (CSI).
Serial interface IICA0	Performs I2C communications with the LCD module.
External interrupt	Switch input

1.1.1 Overview of the Renesas Flash Driver RL78 Type 01

The Renesas Flash Driver (RFD) RL78 Type 01 software rewrites data in flash memory in the RL78/G23.

A user program calls the API functions of the RFD RL78 Type 01 to rewrite the content of code flash memory or data flash memory.

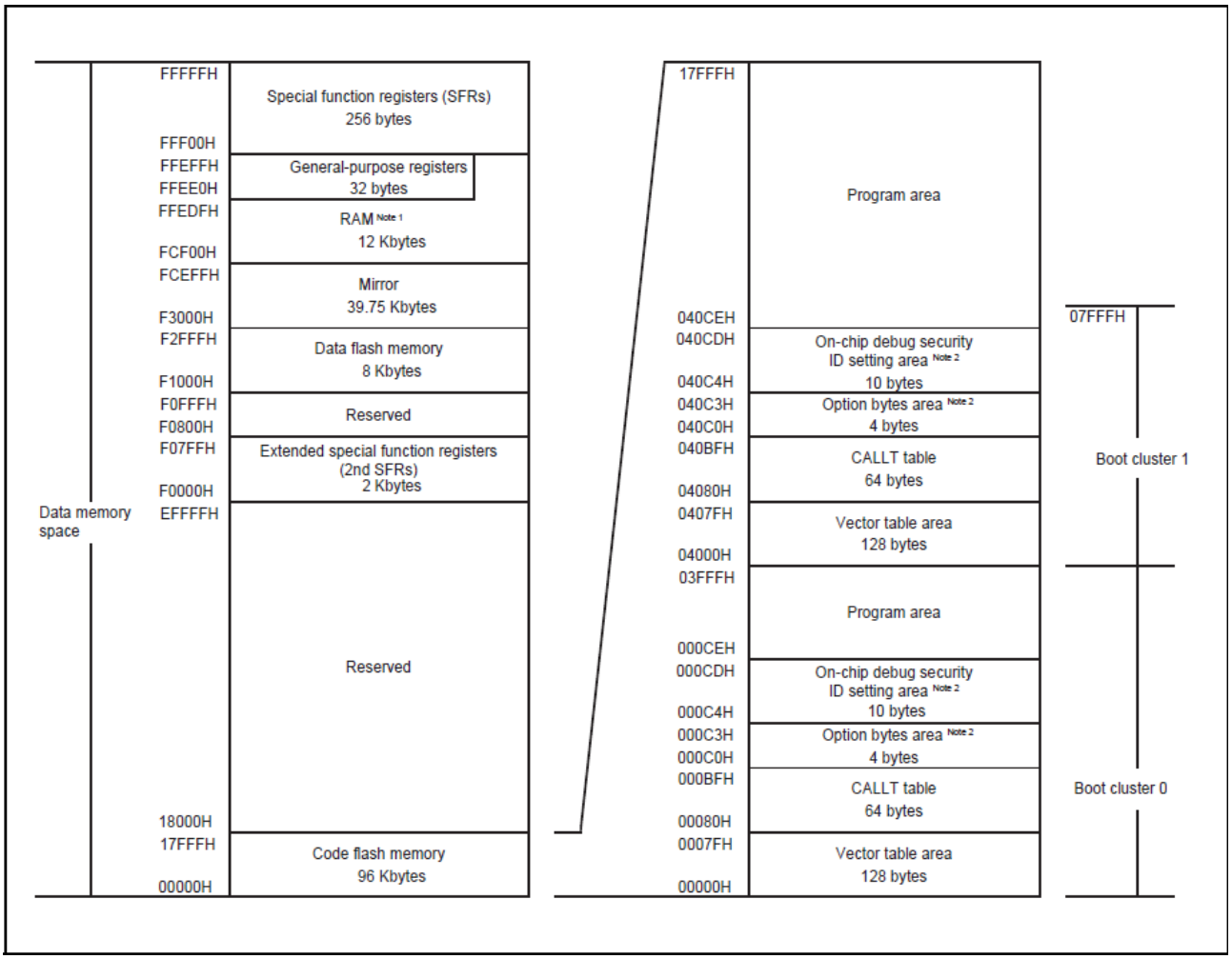
To enable self-programming, the self-programming environment must be configured. During self-programming, the high-speed on-chip oscillator must be running. If the high-speed on-chip oscillator is deactivated, activate it (HIOSTOP = 0), wait for 5 μs, and start self-programming. In addition, deactivate the middle-speed on-chip oscillator (MIOEN = 0) and select the high-speed on-chip oscillator as the main on-chip oscillator clock (fOCO) by setting MCM1 = 0. Do not change the settings of the flash operating mode select register (FLMODE register) while reprogramming flash memory.

Additionally, the user program and reference data to be used during code flash programming mode must be copied from ROM (code flash memory) to RAM in advance for execution and reference in RAM.

1.1.2 Code Flash Memory

The configuration of the RL78/G23 (R7F100GLG) code flash memory is shown below.

Figure 1-1 Code Flash Memory Configuration



Caution: When the boot swap function is used, the option byte area (000C0H to 000C3H) in boot cluster 0 is swapped with the option byte area (040C0H to 040C3H) in boot cluster 1. Accordingly, place the same values in the area (040C0H to 040C3H) as those in the area (000C0H to 000C3H) when using the boot swap function.

The features of the RL78/G23 code flash memory are summarized below.

Table 1-2 Features of the Code Flash Memory

Item	Description
Minimum unit of erasure	1 block (2048 bytes)
Minimum unit of programming	1 word (4 bytes)
Minimum unit of verification	1 byte
Security functions	Block erasure, programming, and boot cluster 0 reprogramming protection are supported. (They are enabled at shipment)
	The flash shield window function allows users to prohibit writing and erasure of areas inside or outside the specified window range only during self-programming.
	Security settings programmable using the flash self-programming code (Renesas Flash Driver RL78 Type01)

Caution: During self-programming, the security settings for prohibiting erasure and writing of data in blocks are disabled. If you want to prohibit erasure and writing of data in blocks during self-programming, use the flash shield window function.

1.1.3 Flash Memory Self-Programming

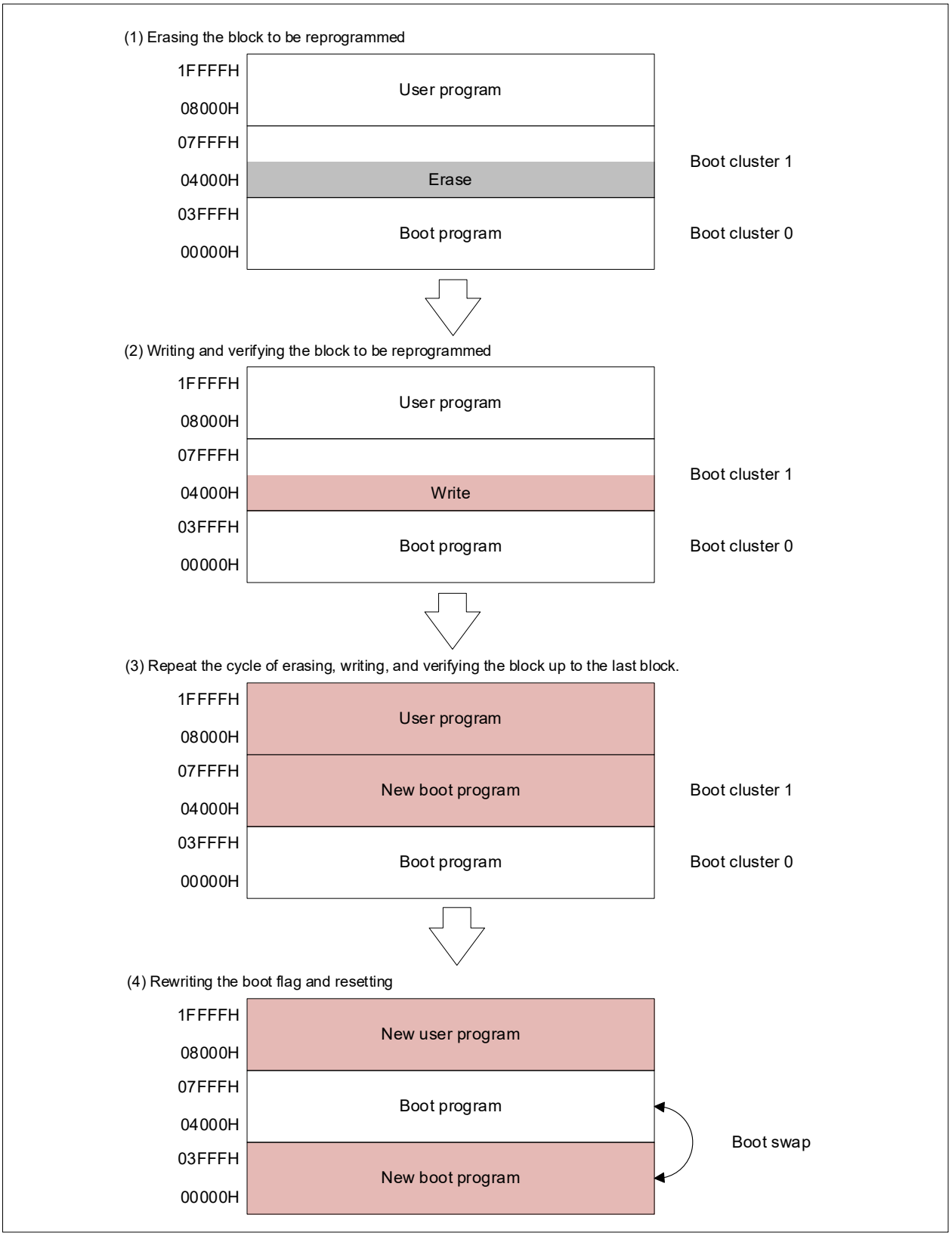
For the RL78/G23, the Renesas Flash Driver (RFD) is provided to enable self-programming. Self-programming is performed by calling the API functions of the RFD from a rewrite program.

For self-programming in the RL78/G23, the flash memory sequencer is used to control reprogramming of flash memory. While the flash memory sequencer is controlling reprogramming, code flash memory cannot be referenced. If a user program in the code flash memory needs to run while the sequencer is working, some segments of the RFD and the rewrite program must be stored in RAM before data is erased from or written to the code flash memory and security flags are set.

1.1.4 Flash Memory Reprogramming

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

Figure 1-3 Outline of Flash Memory Reprogramming

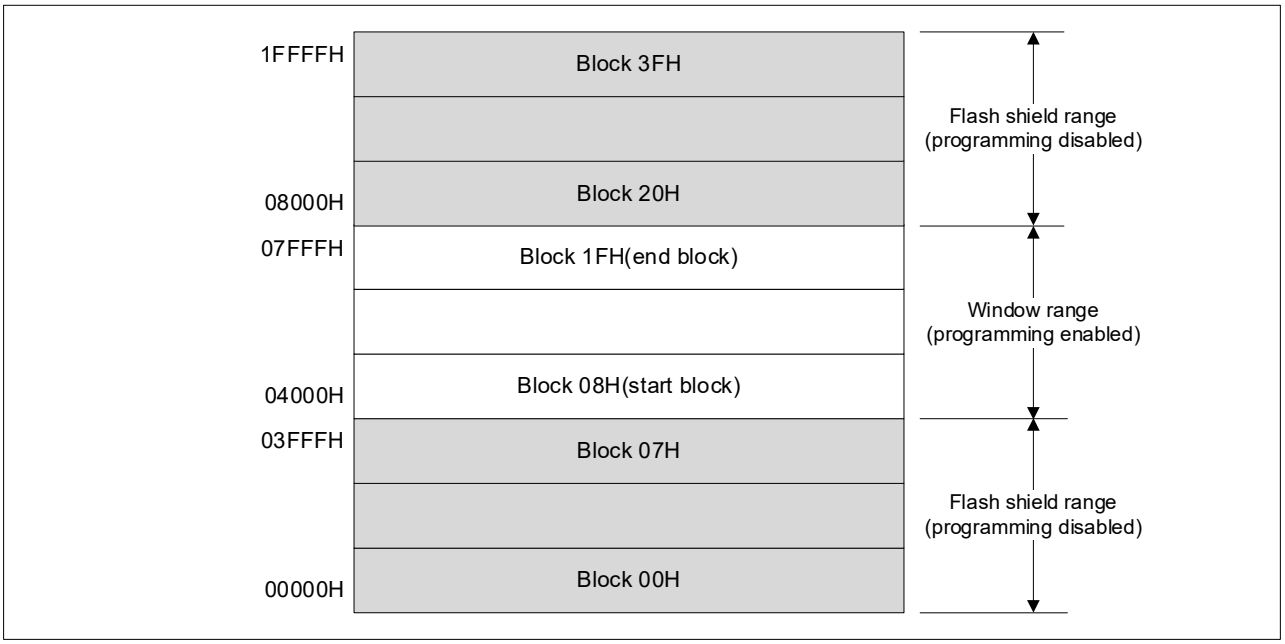


1.1.5 Flash Shield window

The flash shield window function is one of the security functions for self-programming. This function prohibits writing and erasure of areas inside or outside the specified window range only during 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 0FH.

Figure 1-4 Outline of the Flash Shield Window



1.1.6 Communication Specifications

The sample program described in this application note obtains data for firmware upgrade from external flash memory and performs self-programming. If self-programing ends abnormally, the program does not return a response, displays "ERROR!" on the LCD module, and cancels subsequent processing. The following table lists the communication settings for simplified SPI (CSI).

Table 1-2 Simplified SPI (CSI) Communication Settings

Transfer mode	single-transfer mode
Data bit length[bit]	8
Data transfer sequence	MSB first
Data and clock phase	Type1
Transfer rate	153600 bps

1.1.7 How to obtain the Renesas Flash Driver RL78 Type01

Before starting compilation, download the latest version of the Renesas Flash Driver RL78 Type01 and copy the file to the RFD_RL78_TYPE1 folder.

You can obtain the Renesas Flash Driver RL78 Type01 from the following URL:

https://www.renesas.com/jp/ja/document/scd/renesas-flash-driver-rl78-type-01-rl78g23

1.2 Operation Outline

This application note describes how to upgrade firmware using external flash memory.

The rewrite program displays the current version information of firmware on the LCD module. When a switch is pressed, the program lights the LED indicating that flash memory is being accessed. The MCU enters self-programming mode. The rewrite program obtains the data for upgrading firmware from the external flash memory and reprograms code flash memory with it. When reprogramming is complete, the program turns off the LED and displays the new version information on the LCD module.

(1) The program initializes a port.
- Configures the P53 pin as an output port (the initial logic level of the pin is high and LED 1 is turned off).

(2) The program initializes serial array unit CSI11.
- Configures pins for the CSI11 (sets P30 as SCK11, P50 as SI11, and P51 as SO11).
- Sets fCLK as the CSI11 operation clock.
- Sets single transfer mode as the transfer mode.
- Sets 8 bits as the data bit length.
- Sets MSB as the data transfer sequence.
- Sets type 1 as the data and clock phase.
- Sets 153600 bps as the baud rate.
- Enables INTCSI11 interrupts.

(3) The program initializes serial interface IICA.
- Configures pins for the IICA0 (sets P60 as SCLA0 and P61 as SDAA0).
- Sets fCLK/2 as the IICA0 operation clock.
- Sets 10H as the local address.
- Sets standard as the operation mode.
- Sets 80000 bps as the transfer clock.
- Enables INTIICA0 interrupts.

(4) The program initializes external interrupts.
- Sets the falling edge as the valid edge for the INTP0 pin.

(5) The program enables external interrupts.

(6) The program starts the operation of the CSI11.

(7) The program initializes the LCD module and displays the character string specified for constant LCD_STRING on the LCD module.

(8) The program waits for input from the switch.

(9) When switch input is confirmed, the program sets the P53 pin to low level output, lights LED 1 (indicating that flash memory is being accessed), and initializes self-programming.

(10) The program sets 08000H (program area in code flash memory) as the write destination address.

(11) The program calculates the block to be written from the write destination address (block number: 010H).

(12) The program calls the r_CF_EraseBlock function to erase the rewrite target block.

(13) The program obtains write data (256 bytes) from the external flash memory.

(14) The program calls the r_CF_WriteData function to write the obtained data at the write destination address.

(15) The program calls the r_CF_VerifyData function to verify the written data and the block to which received data is written.

(16) The program increments the write destination address by the size of write (256 bytes).

(17) The program repeats steps (10) through (13) until all the data for one block (2048 bytes) is completely written.

(18) After setting the P53 pin to high level output, the program turns off LED 1 (indicating that flash memory is being accessed) and calls the R_RFD_ForceReset function to generate an internal reset.

(19) The program restarts and initializes the MCU. Then the program displays the new character string specified for constant LCD_STRING on the LCD module.

Caution: If self-programming of the flash memory is not successfully ended (an error occurred during processing), the rewrite program displays “ERROR!” on the LCD module and cancels subsequent processing.

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

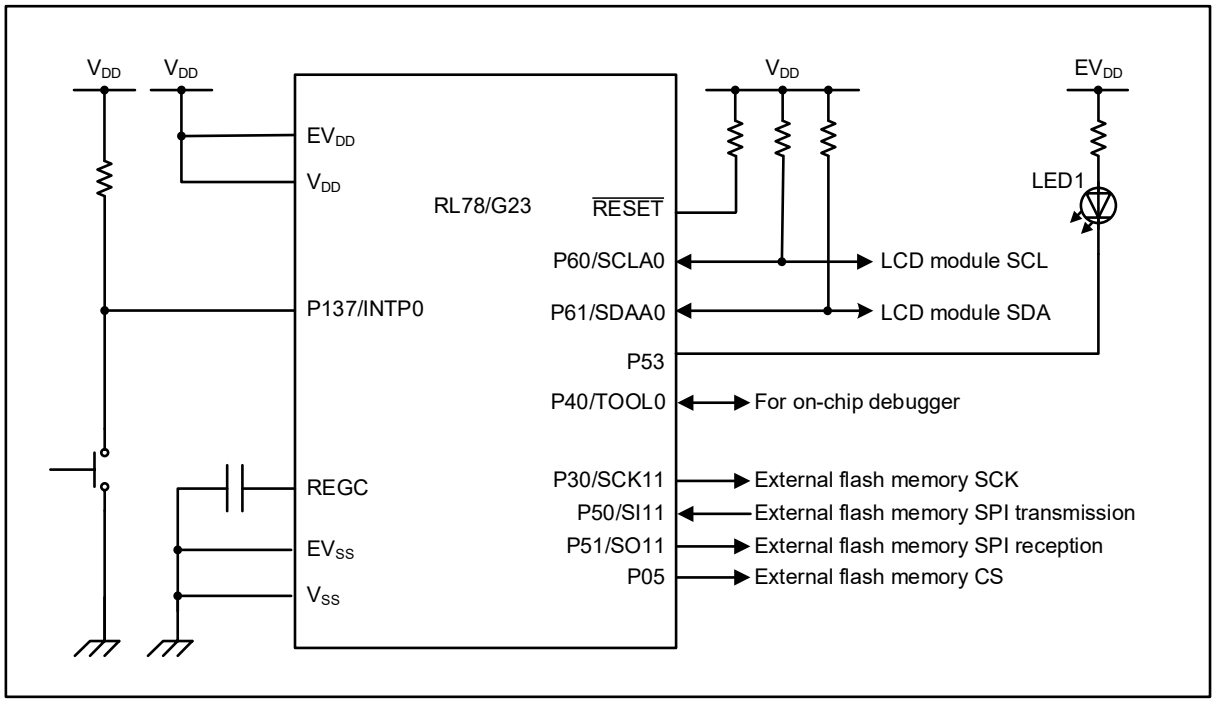
Item	Description
Microcontroller used	RL78/G23 (R7F100GLG)
Operating frequency	High-speed on-chip oscillator (fIH): 32MHz
Operating voltage	3.3 V (can be operated at 3.1 V to 5.5 V) LVD operations (V_{LVD}): Reset mode At rising edge TYP. 1.90 V At falling edge TYP. 1.86 V
Integrated development environment (CS+)	CS+ for CC V8.06.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.10.00 from Renesas Electronics Corp.
Integrated development environment (e^2 studio)	e2studio V2021-07 from Renesas Electronics Corp.
C compiler (e^2 studio)	CC-RL V1.10.00 from Renesas Electronics Corp.
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.21.2 from IAR Systems Corp.
C compiler (IAR)	IAR C/C++ Compiler for Renesas RL78 V4.21.2.2420 from IAR Systems Corp.
Board to be used	RL78/G23-64p Fast Prototyping Board, RTK7RLG230CLG000BJ

3. Description of the Hardware

3.1 Hardware Configuration Example

Figure 3-1 shows an example of the hardware configuration used for this application note.

Figure 3-1 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_{LVD0}) that is specified as LVD0.

3.2 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	Description
P30/SCK11	Output	CSI serial clock input pin
P50/SI11	Input	CSI serial data reception pin
P51/SO11	Output	CSI serial data transmission pin
P05	Output	External flash memory CS pin
P53	Output	Lights or extinguishes LED 1(for indicating that flash memory is being accessed).
P137/INTP0	Input	Trigger for reprogramming
P60/SCLA0、P61/SDAA0	Input/Output	I2C communication with the LCD module

Caution In this application note, only the pins used are processed. When actually creating a circuit, perform pin processing appropriately and design it so that it satisfies the electrical characteristics.

4. Software Explanation

4.1 List of Option Byte Settings

Table 4-1 summarizes the settings of the option bytes.

Table 4-1 Option Byte Settings

Address	Setting	Description
000C0H/040C0H	11101111B	Disables the watchdog timer. (Stops counting after the release from the reset status.)
000C1H/040C1H	11111110B	LVD operations (V_{LVD}): Reset mode At rising edge TYP. 1.90 V At falling edge TYP. 1.86 V
000C2H/040C2H	11101000B	HS mode High-speed on-chip oscillator clock: 32MHz
000C3H/040C3H	10000101B	Enables the on-chip debugger

4.2 Startup routine settings

4.2.1 Definition of the section for the stack area (.stack_bss)

The sample program stores the data for firmware upgrade as local variables. Because local variables are stored in the stack area, modify cstart.asm, secure desired size of stack area, and initialize the stack area.

```
;$IF (__RENESAS_VERSION__ < 0x01010000)                Add ';' to the first line and comment out
;-------------------------------------------------------------------------------
;     stack area
;-------------------------------------------------------------------------------
; !!! [CAUTION] !!!
; Set up stack size suitable for a project.
.SECTION .stack_bss, BSS
_stackend:
      .DS       0x200
_stacktop:
;$ENDIF                                                Add ';' to the first line and comment out
    .
    .
    .
    .
      ;------------------------------------------------
      ; setting the stack pointer
      ;------------------------------------------------
;$IF (__RENESAS_VERSION__ >= 0x01010000)               Add ';' to the first line and comment out
;      MOVW      SP,#LOWW(__STACK_ADDR_START)          Add ';' to the first line and comment out
;$ELSE     ; for CC-RL V1.00                           Add ';' to the first line and comment out
      MOVW      SP,#LOWW(_stacktop)
;$ENDIF                                                Add ';' to the first line and comment out

      ;------------------------------------------------
      ; initializing stack area
      ;------------------------------------------------
;$IF (__RENESAS_VERSION__ >= 0x01010000)               Add ';' to the first line and comment out
;      MOVW      AX,#LOWW(__STACK_ADDR_END)            Add ';' to the first line and comment out
;$ELSE     ; for CC-RL V1.00                           Add ';' to the first line and comment out
      MOVW      AX,#LOWW(_stackend)
;$ENDIF                                                Add ';' to the first line and comment out
      CALL      !!_stkinit
```

4.2.2 Deploying the Rewrite Programs in the RAM Area

Deploy the program used for rewriting firmware in the RAM area.

Table 4-2 lists the sections that store the program used for rewriting firmware.

Table 4-2 Section Information

Section Name	Deployment-destination section name	Item to Be Deployed
RFD_CMN_f	RFD_CMN_fR	Program section for the common flash memory control API function
RFD_CF_f	RFD_CF_fR	Program section for the code flash memory API function
RFD_EX_f	RFD_EX_fR	Program section for the extra area control API function
SMP_CMN_f	SMP_CMN_fR	Program section for the common flash memory control sample function
SMP_CF_f	SMP_CF_fR	Program section for the code flash memory control sample function

To deploy the preceding sections in the RAM area, you need to add processing to "cstart.asm".

In "cstart.asm", add code for the processing after the following lines:

```
;---------------------------------------------
; ROM data copy
;---------------------------------------------
```

The code to be added is as follows:

```
    ; copy .text to RAM (section-name)
    MOV     C,#HIGHW(STARTOF(section-name))
    MOVW    HL,#LOWW(STARTOF(section-name))
    MOVW    DE,#LOWW(STARTOF(Placement section name))
    BR      $.L12_TEXT
.Lm1_TEXT:
    MOV     A,C
    MOV     ES,A
    MOV     A,ES:[HL]
    MOV     [DE],A
    INCW    DE
    INCW    HL
    CLRW    AX
    CMPW    AX,HL
    SKNZ
    INC
.Lm2_TEXT:
    MOVW    AX,HL
    CMPW    AX,#LOWW(STARTOF(section-name) + SIZEOF(section-name))
    BNZ     $.L11_TEXT
```

- In **section-name**, specify the name of the section to be deployed.
- Add the preceding code for each section that needs to be deployed.
- For m, set any number of your choice. Specify a different number for each section.

4.3 On-chip Debug Security ID

The RL78/G23 has the on-chip debug security ID area allocated to addresses 000C4H to 000CDH of flash memory to preclude the memory contents from being sneaked by the unauthorized third party.

When using the boot swap function for self-programming, it is necessary to set the same values that are set in 000C4H to 000CDH also in 040C4H to 040CDH because bytes in 000C4H to 000CDH are swapped with the bytes in 040C4H to 040CDH.

4.4 Resources Used by the Sample Program

4.4.1 List of Sections in the ROM Area

Table 4-3 lists the sections that are deployed in the ROM area and used by the sample program.

Table 4-3 List of Sections in the ROM Area

Section Name	Item to Be Deployed
RFD_CMN_f	Program section for the common flash memory control API function
RFD_CF_f	Program section for the code flash memory control API function
RFD_EX_f	Program section for the extra area control API function
RFD_DF_f	Program section for the data flash memory control API function
SMP_CMN_f	Program section for the common flash memory control sample function
SMP_CF_f	Program section for the code flash memory control sample function

4.4.2 List of Sections in the RAM Area

Table 4-4 lists the sections that are deployed in the RAM area and used by the sample program.

Table 4-4 List of Sections in the RAM Area

Section Name	Items to Be Deployed
RFD_DATA_n	Data section for RFD RL78 Type01
RFD_CMN_fR	Program section for the common flash memory control API function
RFD_CF_fR	Program section for the code flash memory control API function
RFD_EX_fR	Program section for the extra area control API function
SMP_CMN_fR	Program section for the common flash memory control sample function
SMP_CF_fR	Program section for the code flash memory control sample function

4.5 List of Constants

Table 4-5 lists the constants for the sample program.

Table 4-5 Constants for the Sample Program

Constant	Setting	Description
STR_VERSION	“V1.00”	Firmware version information
LED_ON	00H	The LED is turned on.
LED_OFF	01H	The LED is turned off.
WAITCOUNT_32M	8000	5-ms count value when the MCU is operating in HS mode at 32 MHz
WRITE_START_ADDRESS	00080000H	Code flash memory beginning address (except for boot cluster 1)
FLASH_READ_START_ADDRESS	00000000H	External flash memory read beginning address
WRITE_DATA_SIZE	0100H	Code flash memory write size (256 bytes)
CF_BLOCK_SIZE	0800H	Code flash memory block size (2048 bytes)
CPU_FREQUENCY	32	CPU operating frequency
VALUE_U08_MASK1_FSQ_STATUS_ERR_CFDF_SEQUENCER	10H	Error status mask value for the execution results of the flash memory sequencer
VALUE_U08_MASK1_FSQ_STATUS_ERR_EXTRA_SEQUENCER	20H	Error status mask value for the execution results of the flash memory sequencer Bit 5: Extra area sequencer error
VALUE_U08_MASK1_FSQ_STATUS_ERR_ERASE	01H	Error status mask value for the execution results of the flash memory sequencer Bit 0: Erase command error
VALUE_U08_MASK1_FSQ_STATUS_ERR_WRITE	02H	Error status mask value for the execution results of the flash memory sequencer Bit 1: Write command error
VALUE_U08_MASK1_FSQ_STATUS_ERR_BLANKCHECK	08H	Error status mask value for the execution results of the flash memory sequencer Bit 3: Blank check command error
VALUE_U32_DF_BASE_ADDR	000F1000H	Data flash memory start address
VALUE_U08_SHIFT_ADDR_TO_BLOCK_CF	11	Constant for bit shifting used for calculating the block number in code flash memory
VALUE_U08_SHIFT_ADDR_TO_BLOCK_DF	8	Constant for bit shifting used for calculating the block number in data flash memory
VALUE_U01_MASK0_1BIT	0	Fixed to 0.
VALUE_U01_MASK1_1BIT	1	Fixed to 1.
VALUE_U08_MASK0_8BIT	00H	Fixed to 00H.
VALUE_U08_MASK1_8BIT	FFH	Fixed to FFH.

4.6 Enumerated type

Table 4-6 shows the definition of the enumeration used in the sample program.

Table 4-6 enum e_ret (Enumerated variable name: e_ret_t)

Symbol Name	Setting	Description
ENUM_RET_STS_OK	00H	Normal status
ENUM_RET_ERR_CFDF_SEQUENCER	10H	Code/data flash area sequencer error
ENUM_RET_ERR_EXTRA_SEQUENCER	11H	Extra area sequencer error
ENUM_RET_ERR_ERASE	12H	Erase error
ENUM_RET_ERR_WRITE	13H	Write error
ENUM_RET_ERR_BLANKCHECK	14H	Blank error
ENUM_RET_ERR_CHECK_WRITE_DATA	15H	Error in comparison between the written and read values
ENUM_RET_ERR_MODE_MISMATCHED	16H	Mode mismatch error
ENUM_RET_ERR_PARAMETER	17H	Parameter error
ENUM_RET_ERR_CONFIGURATION	18H	Device configuration error

4.7 List of Variables

Table 4-7 shows the definition of the global variables used in the sample program.

Table 4-7 Global Variables

Type	Variable Name	Description	Function Used
uint8_t	g_csi11_end	CSI11 communication completion flag	r_Config_CSI11_callback_receiveend、r_Config_CSI11_callback_sendend
uint8_t	f_update_start	Upgrade start flag	main、r_Config_INTC_intp0_interrupt

4.8 List of Functions

Table 4-8 lists the functions that are used in this sample program.

Table 4-8 List of Functions

Function Name	Outline
r_rfd_initialize	Initialization processing for RFD RL78 Type01
r_CF_BlockControlSequence	Block control processing for code flash memory
r_CF_WriteVerifySequence	Write-and-verify processing for the code flash memory
r_CF_EraseBlock	Block erase processing for the code flash memory
r_CF_WriteData	Write processing for the code flash memory
r_CF_VerifyData	Verify processing for the code flash memory
r_CheckCFDFSequencerEnd	Sequence end processing for the code flash memory
r_Config_CSI11_callback_sendend	Callback processing for CSI11 transmission completion interrupts
r_Config_CSI11_callback_receiveend	Callback processing for CSI11 reception completion interrupts
r_CSI11_GetData	Processing for obtaining data from external flash memory
r_Config_IICA0_callback_master_sendend	Callback processing for IICA0 transmission completion interrupts
r_Config_IICA0_callback_master_error	Callback processing for IICA0 transmission error interrupts
r_LCM_init	Processing to initialize the LCD module
r_LCM_clear	Processing to clear display for the LCD module
r_LCM_send_string	Processing to send strings to the LCD module
r_LCM_send_command	Command sending processing for the LCD module
r_LCM_send_data	Processing to send data to the LCD module
r_LCM_turn_sendend_on	Communication end flag setting for the LCD module
r_LCM_wait_sendend	Communication end wait processing for the LCD module
r_Config_INTC_intp0_interrupt	External interrupt processing
R_RFD_Init[Note]	Initialization processing for RFD RL78 Type01
R_RFD_SetFlashMemoryMode[Note]	Flash memory control mode change processing
R_RFD_CheckCFDFSeqEndStep1[Note]	Processing to confirm that the extra area sequencer has terminated
R_RFD_CheckCFDFSeqEndStep2[Note]	Processing to check whether the command was terminated by clearing the extra area sequencer control register
R_RFD_ClearSeqRegister[Note]	Processing to clear the register that controls the code/data flash area sequencer or extra area sequencer
R_RFD_GetSeqErrorStatus[Note]	Processing to obtain error information generated by the code/data flash area sequencer command or extra area sequencer command
R_RFD_ForceReset[Note]	Internal CPU reset request
R_RFD_EraseCodeFlashReq[Note]	Code flash memory erase processing
R_RFD_WriteCodeFlashReq[Note]	Code flash memory write processing

Note: This is an API function defined for the flash self-programming code. For details about the API function, see the "RL78 Family Renesas Flash Driver RL78 Type01 User's Manual".

4.9 Function Specifications

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

r_rfd_initialize	
Synopsis	Initialization processing for RFD RL78 Type01
Header	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_rfd_initialize(void);
Explanation	This function initializes RFD RL78 Type01.
Arguments	None
Return value	ENUM_RET_STS_OK: Normal status ENUM_RET_ERR_CONFIGURATION: Device configuration error ENUM_RET_ERR_PARAMETER: Parameter error

r_CF_BlockControlSequence	
Synopsis	Block control processing for code flash memory
Header	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_BlockControlSequence(uint32_t write_start_addr, int32_t flash_read_addr);
Explanation	This function erases data from, writes data to, and verifies data in code flash memory. The unit of control is 1 block from the address specified for one of the arguments.
Arguments	uint32_t write_start_addr: Start address for erasure, writing, and verification int32_t flash_read_addr: External flash memory read beginning address
Return value	ENUM_RET_STS_OK: Normal end ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_ERASE: Erasure error

r_CF_WriteVerifySequence	
Synopsis	Write-and-verify processing for the code flash memory
Header	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_WriteVerifySequence(uint32_t start_addr, uint16_t write_data_length, uint8_t __near * write_data);
Explanation	This function writes data to the code flash memory and verifies the written data.
Arguments	uint32_t i_u32_start_addr: Write start address uint16_t i_u16_write_data_length: Write size uint8_t __near * inp_u08_write_data: Write data
Return value	ENUM_RET_STS_OK: Normal status ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_WRITE: Write error

r_CF_EraseBlock	
Synopsis	Block erase processing for the code flash memory
Header	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_EraseBlock(uint32_t i_u32_start_addr);
Explanation	This function erases data in the code flash memory. A block of data is erased. The block that includes the address specified for an argument will be erased.
Arguments	uint32_t start_addr: Erase start address
Return value	ENUM_RET_STS_OK: Normal status ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_ERASE: Erase error

r_CF_WriteData	
Synopsis	Write processing for the code flash memory
Header	r_rfd_common_api.h、r_rfd_code_flash_api.h、r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_WriteData(uint32_t i_u32_start_addr, uint16_t i_u16_write_data_length, uint8_t __near * inp_u08_write_data);
Explanation	This function writes data to the code flash memory.
Arguments	uint32_t i_u32_start_addr: Write start address uint16_t i_u16_write_data_length: Write size uint8_t __near * inp_u08_write_data: Write data
Return value	ENUM_RET_STS_OK: Normal status ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_WRITE: Write error

r_CF_VerifyData	
Synopsis	Verify processing for the code flash memory
Header	r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_VerifyData(uint32_t start_addr, uint16_t data_length, uint8_t __near * write_data);
Explanation	This function verifies the data written to the code flash memory.
Arguments	uint32_t start_addr: Verify start address uint16_t data_length: Data size uint8_t __near * write_data: Comparison data
Return value	ENUM_RET_STS_OK: Normal status (match) ENUM_RET_ERR_CHECK_WRITE_DATA: Error in comparison between the written and read values (Mismatch)

r_CheckCFDFSequencerEnd	
Synopsis	Sequence end processing for the code flash memory
Header	r_rfd_common_api.h、r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CheckCFDFSequencerEnd(void);
Explanation	This function confirms that the code flash memory sequence has terminated.
Arguments	None
Return value	ENUM_RET_STS_OK: Normal status ENUM_RET_ERR_CFDF_SEQUENCER: Code/data flash area sequencer error ENUM_RET_ERR_ERASE: Erase error ENUM_RET_ERR_WRITE: Write error ENUM_RET_ERR_BLANKCHECK: Blank error

r_Config_CSI11_callback_sendend	
Synopsis	Callback function for CSI11 transmission completion interrupts
Header	E2PROM_driver.h
Declaration	static void r_Config_CSI11_callback_sendend(void)
Explanation	This callback function is called when a CSI11 transmission completion interrupt is generated.
Arguments	None
Return value	None

r_Config_CSI11_callback_receiveend	
Synopsis	Callback function for CSI11 reception completion interrupts
Header	E2PROM_driver.h
Declaration	static void r_Config_CSI11_callback_receiveend(void)
Explanation	This callback function is called when a CSI11 reception completion interrupt is generated.
Arguments	None
Return value	None

r_CSI11_GetData	
Synopsis	Processing for obtaining data from external flash memory
Header	Config_CSI11.h、FlashMemory_driver.h
Declaration	void r_CSI11_GetData(uint32_t addr, uint16_t rx_len, uint8_t * rx_buf)
Explanation	This function obtains data for firmware upgrade from external flash memory.
Arguments	uint32_t addr: Start address for obtaining data from external flash memory uint8_t * rx_buf: Pointer to the buffer storing data to be obtained uint16_t rx_len: Size of data to be obtained
Return value	None

r_Config_IICA0_callback_master_sendend	
Synopsis	Processing for IICA0 transmission completion interrupts
Header	r_cg_macrodriver.h、Config_IICA0.h、LCM_driver.h
Declaration	static void r_Config_IICA0_callback_master_receiveend(void);
Explanation	This callback function is called when an IICA0 transmission completion interrupt is generated.
Arguments	None
Return value	None

r_Config_IICA0_callback_master_error	
Synopsis	IICA0 transmission error handling
Header	r_cg_macrodriver.h、Config_IICA0.h、LCM_driver.h
Declaration	static void r_Config_IICA0_callback_master_error(MD_STATUS flag);
Explanation	This callback function is called when an IICA0 transmission error interrupt is generated.
Arguments	MD_STATUS flag: Error type
Return value	None

r_LCM_init	
Synopsis	Processing to initialize the LCD module
Header	LCM_driver.h、Config_IICA0.h
Declaration	void r_LCM_init(void);
Explanation	This function initializes the LCD module.
Arguments	None
Return value	None

r_LCM_clear	
Synopsis	Processing to clear display for the LCD module
Header	LCM_driver.h、Config_IICA0.h
Declaration	void r_LCM_clear(void);
Explanation	This function sends the Clear Display command to the LCD module.
Arguments	None
Return value	None

r_LCM_send_string	
Synopsis	Processing to send strings to the LCD module
Header	LCM_driver.h、Config_IICA0.h
Declaration	void r_LCM_send_string(uint8_t * const str, lcm_position_t pos);
Explanation	This function displays the character string passed by using the "str" argument on the LCD module. A line can also be displayed by using the "pos" argument.
Arguments	uint8_t * const str: Character string to be displayed lcm_position_t pos: Displayed at the top with LCM_POSITION_TOP Displayed at the bottom with LCM_POSITION_BOTTOM.
Return value	None

r_LCM_send_command	
Synopsis	Command sending processing for the LCD module
Header	LCM_driver.h、Config_IICA0.h
Declaration	void r_LCM_send_command(uint8_t command);
Explanation	This function sends the command passed by using the "command" argument to the LCD module.
Arguments	uint8_t command: Command to send to LCD module
Return value	None

r_LCM_send_data	
Synopsis	Processing to send data to the LCD module
Header	LCM_driver.h、Config_IICA0.h
Declaration	void r_LCM_send_data(uint8_t data);
Explanation	This function sends the data passed by using the "data" argument to the LCD module.
Arguments	uint8_t data: Data to be sent to the LCD module
Return value	None

r_LCM_turn_sendend_on	
Synopsis	Communication end flag setting for the LCD module
Header	LCM_driver.h、Config_IICA0.h
Declaration	void r_LCM_turn_sendend_on(void);
Explanation	This function sets (for g_LCM_is_sendend) the flag that indicates the end of IIC communication with the LCD module.
Arguments	None
Return value	None

r_LCM_wait_sendend	
Synopsis	Communication end wait processing for the LCD module
Header	LCM_driver.h、Config_IICA0.h
Declaration	static void r_LCM_wait_sendend(void);
Explanation	This function waits until IIC communication with the LCD module ends, and then waits for the command execution wait time (5 ms).
Arguments	None
Return value	None

r_Config_INTC_intp0_interrupt	
Synopsis	External interrupt processing
Header	r_cg_macrodriver.h、r_cg_userdefine.h、Config_INTC.h
Declaration	static void __near r_Config_INTC_intp0_interrupt (void)
Explanation	This function sets the upgrade start flag to 1 when a switch is pressed.
Arguments	None
Return value	None

4.10 Flowcharts

4.10.1 Main Processing

Figure 4-1 to Figure 4-2 shows the flowchart for main processing.

Figure 4-1 Main Processing (1/2)

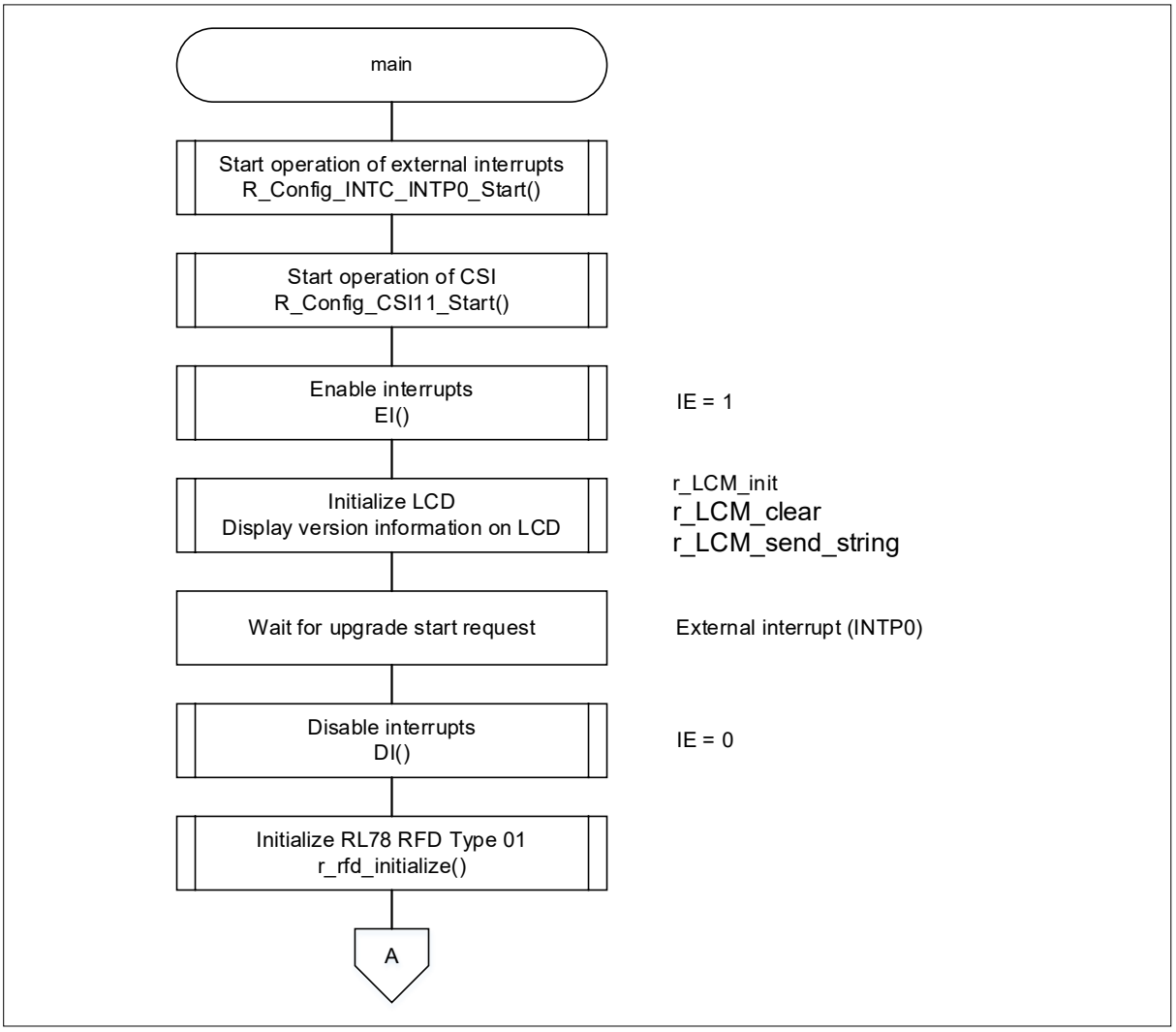
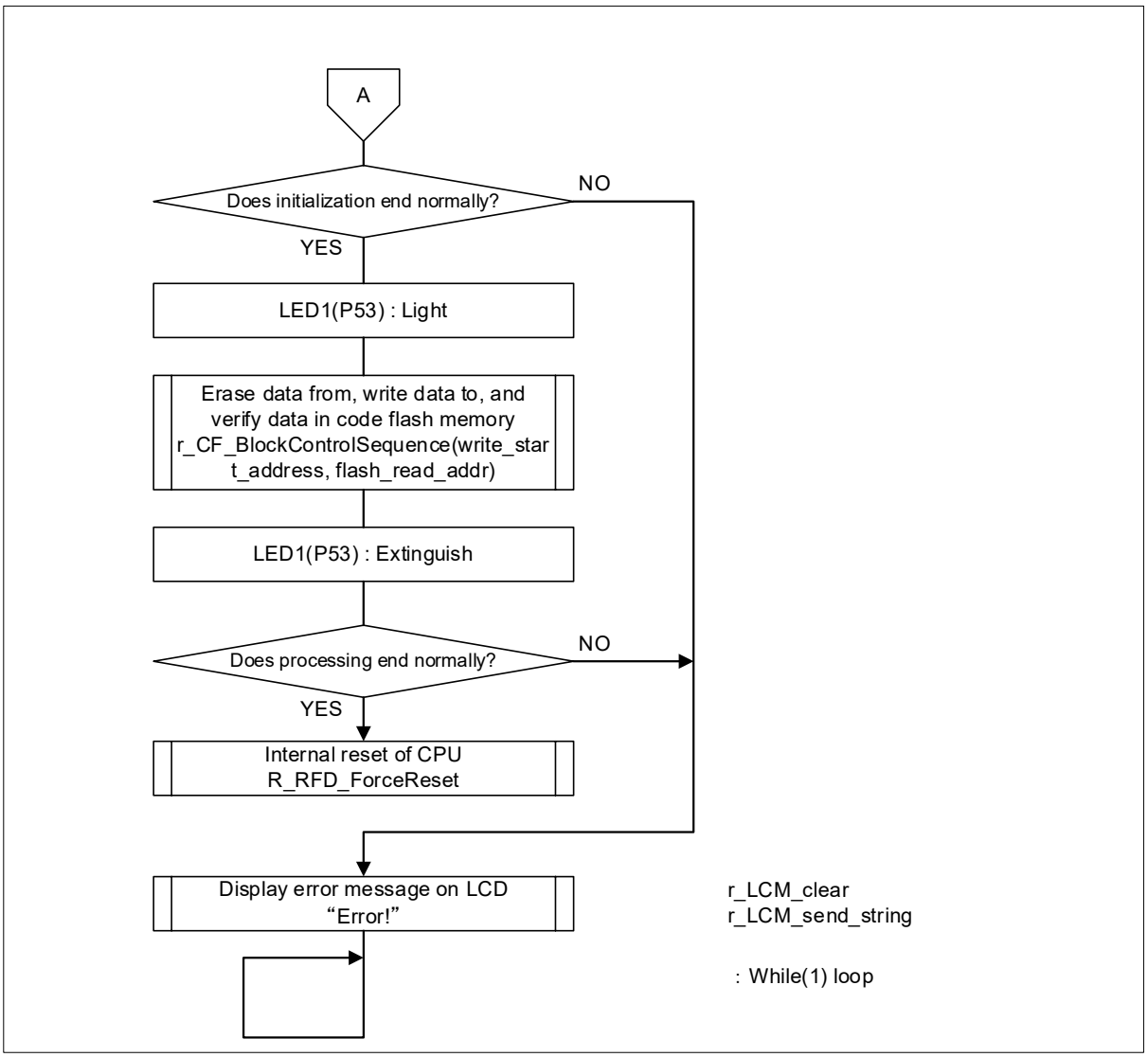


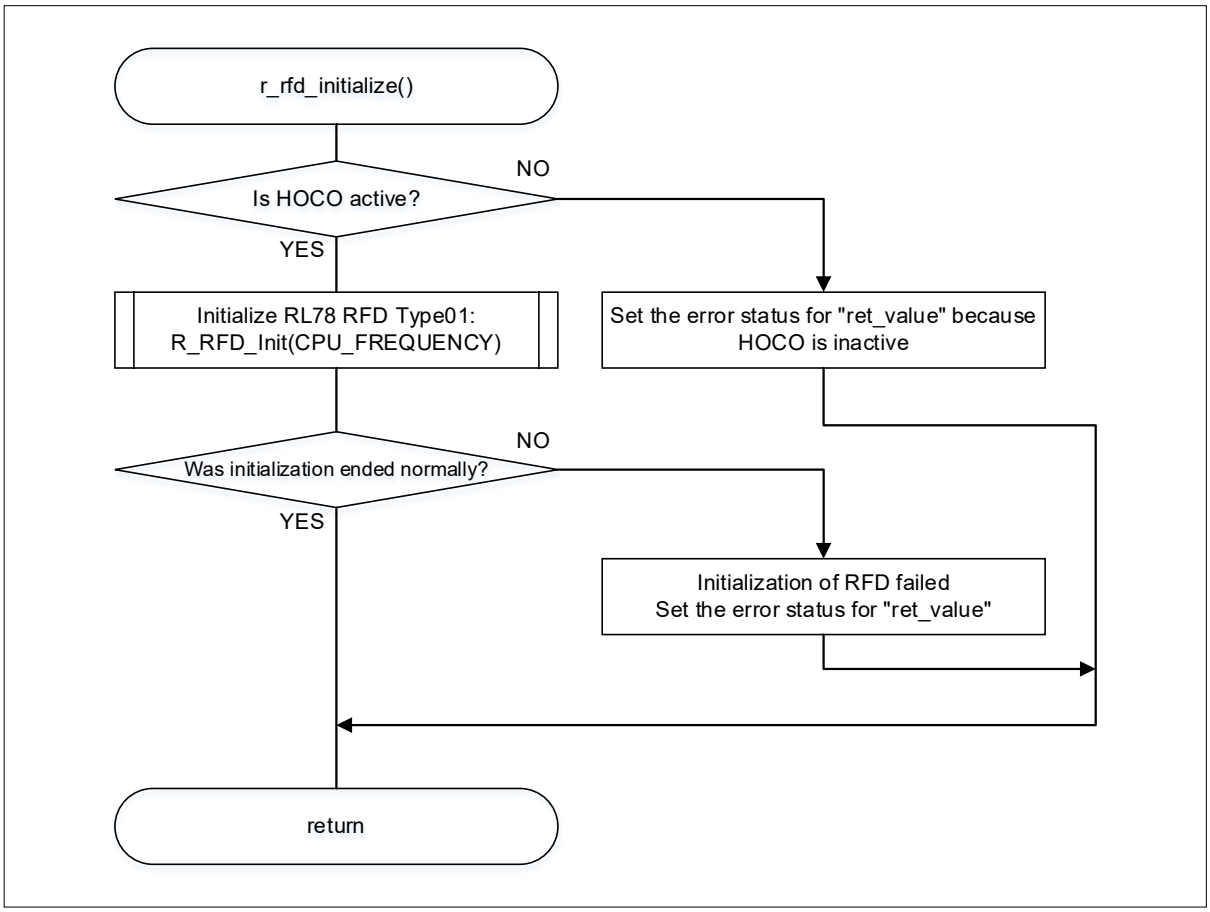
Figure 4-2 Main Processing (2/2)



4.10.2 Initialization Processing for RFD RL78 Type01

Figure 4-3 shows the flowchart for initialization processing for RFD RL78 Type01.

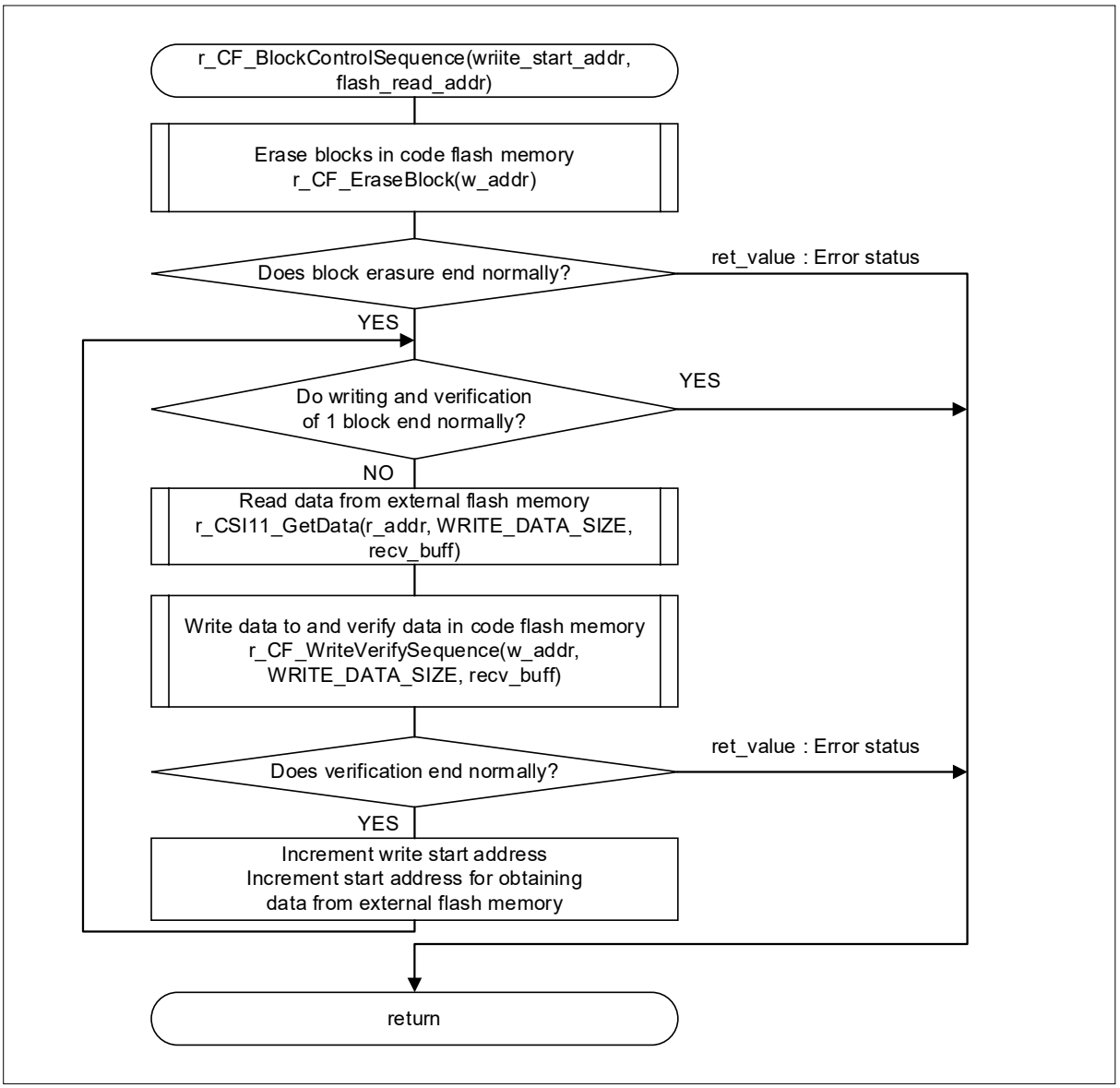
Figure 4-3 Initialization Processing for RFD RL78 Type01



4.10.3 Block Control Processing for Code Flash Memory

Figure 4-4 shows the flowchart for block control processing for code flash memory.

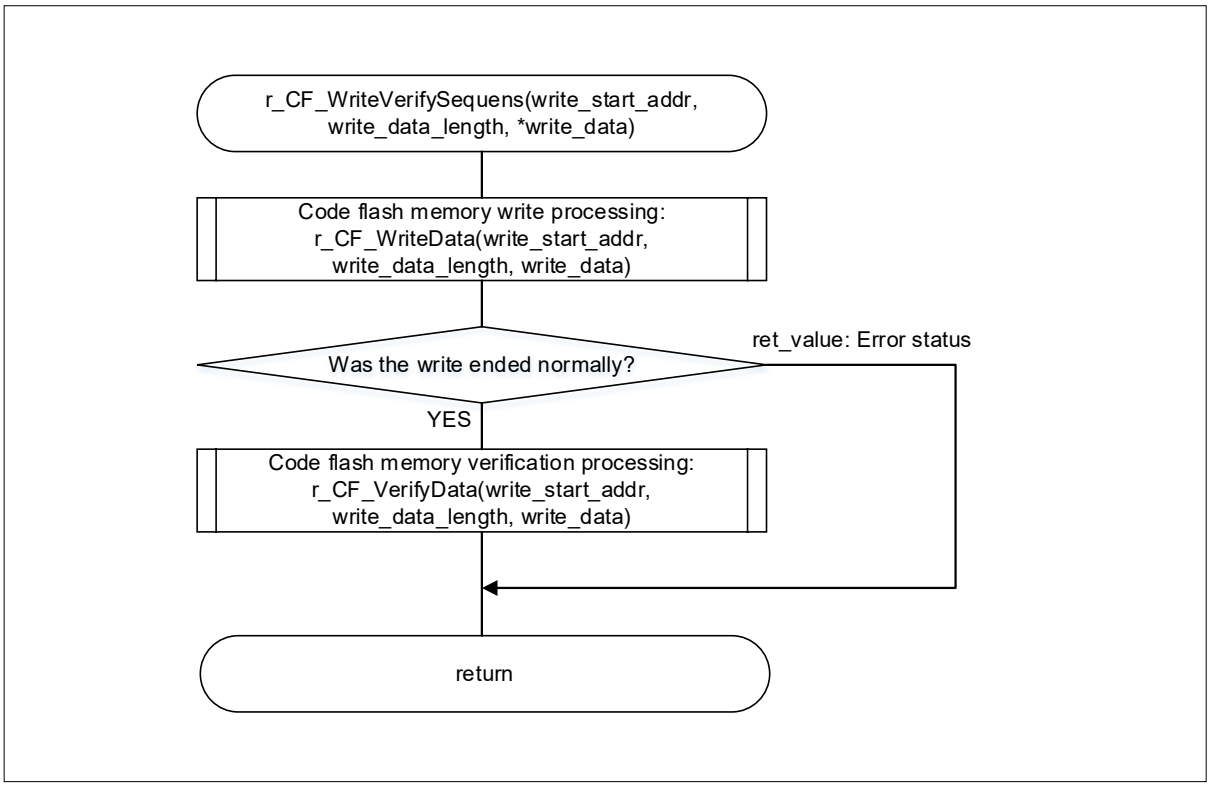
Figure 4-4 Block Control Processing for Code Flash Memory



4.10.4 Write-and-verify Processing for the Code Flash Memory

Figure 4-5 shows the flowchart for write-and-verify processing for the code flash memory.

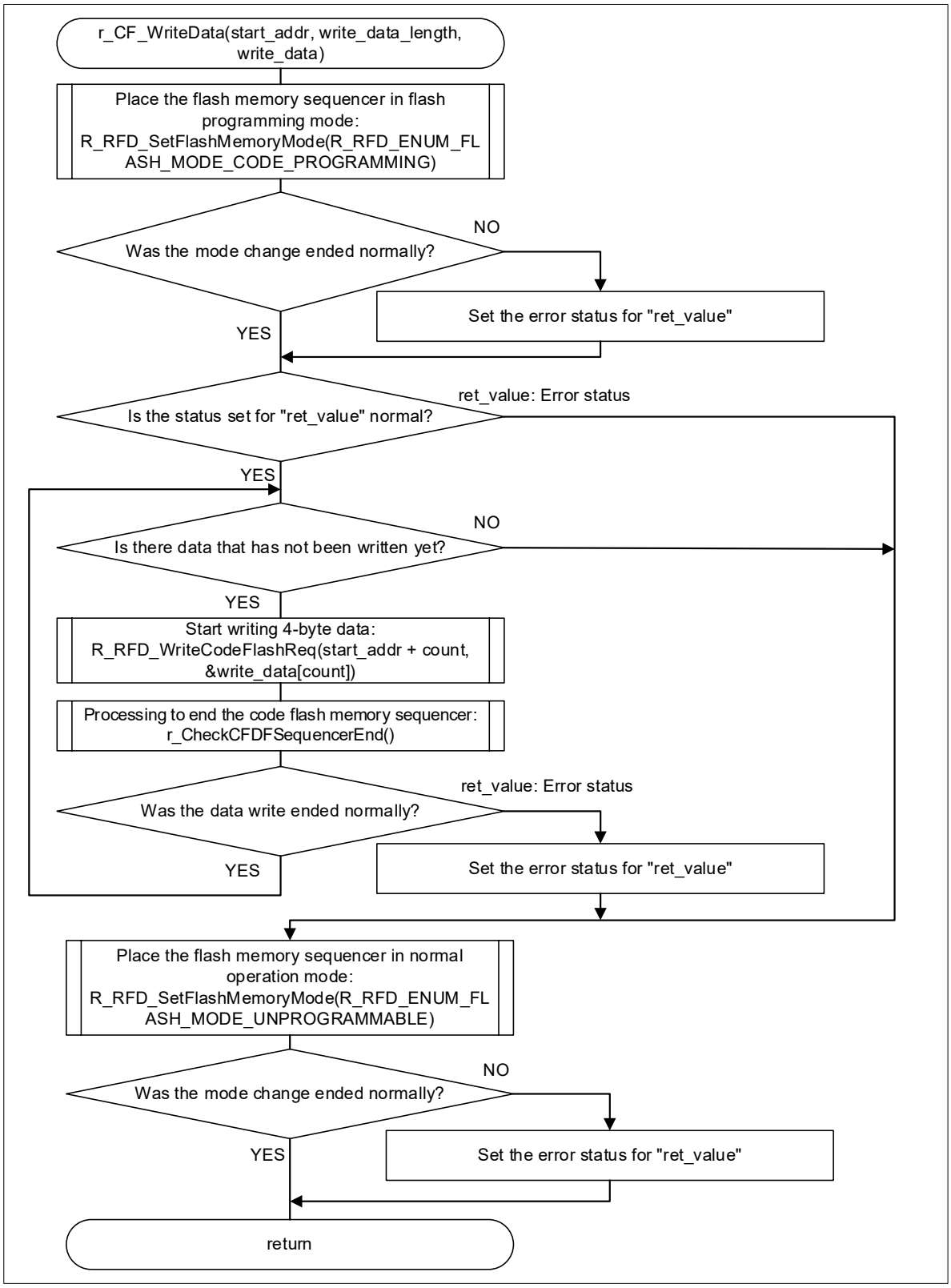
Figure 4-5 Write-and-verify Processing for the Code Flash Memory



4.10.5 Write Processing for the Code Flash Memory

Figure 4-6 shows the flowchart for write processing for the code flash memory.

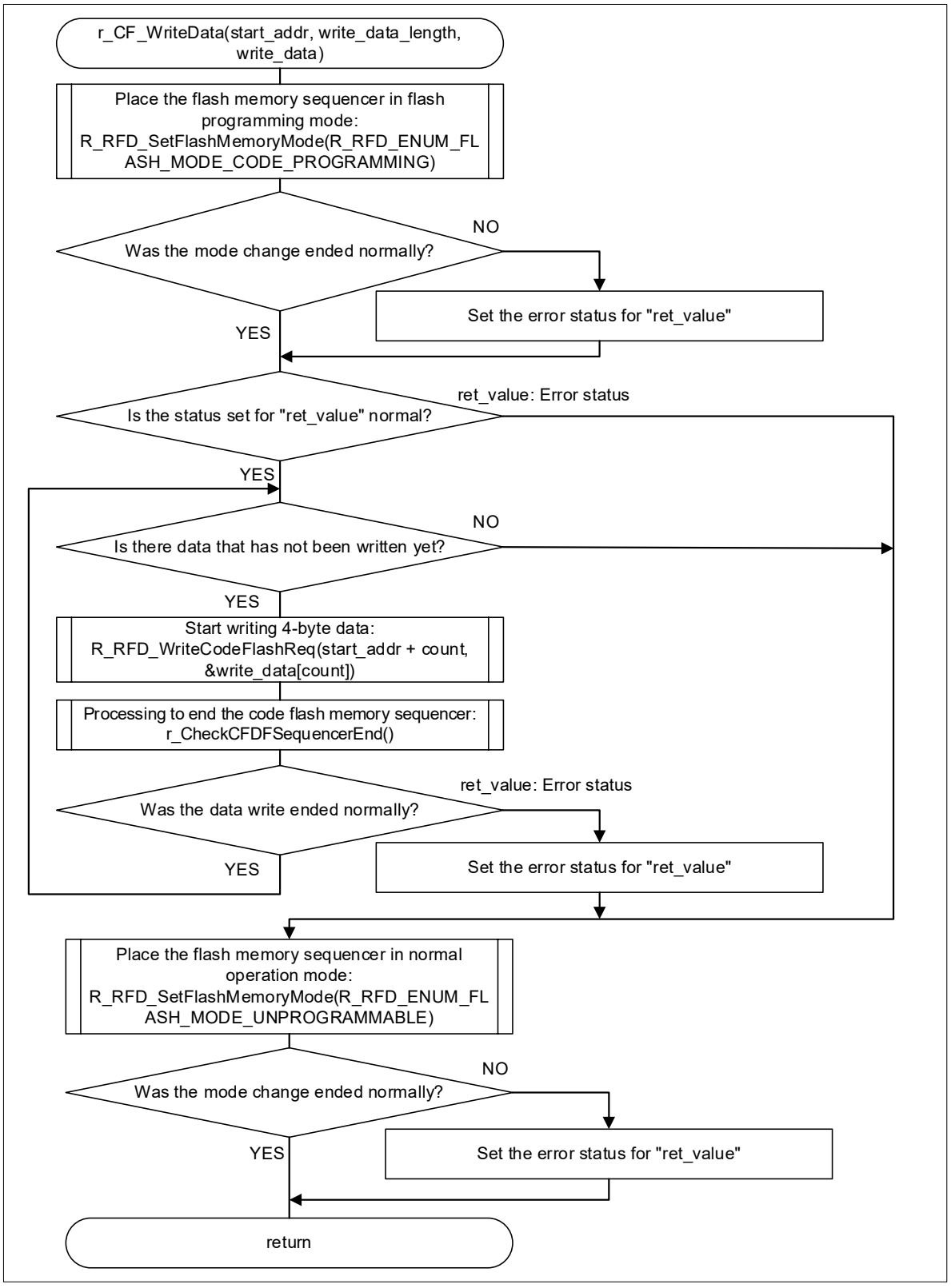
Figure 4-6 Write processing for the Code Flash Memory



4.10.6 Write Processing for the Code Flash Memory

Figure 4-7 shows the flowchart for write processing for the code flash memory.

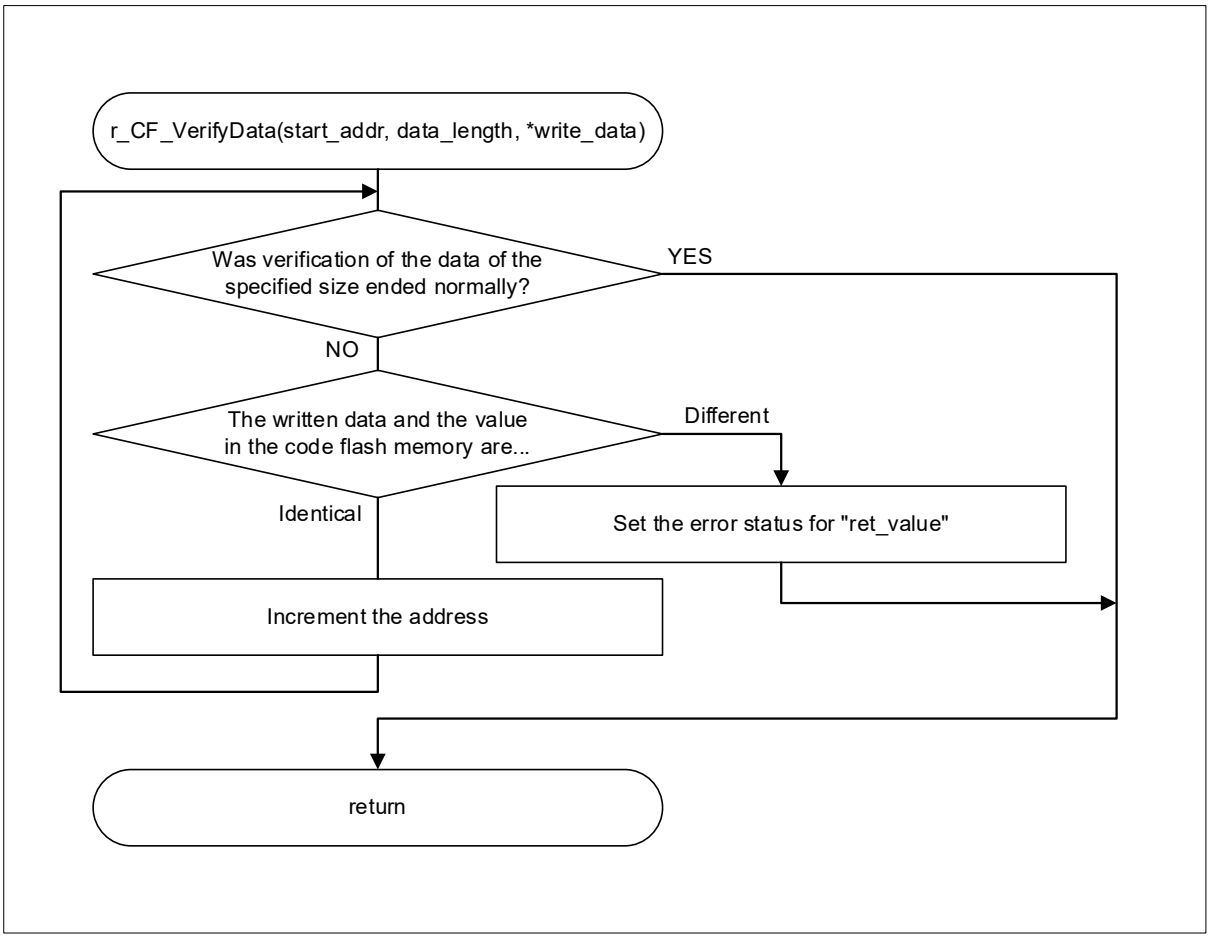
Figure 4-7 Write processing for the Code Flash Memory



4.10.7 Verify Processing for the Code Flash Memory

Figure 4-8 shows the flowchart for verify processing for the code flash memory.

Figure 4-8 Verify Processing for the Code Flash Memory



4.10.8 Sequence End Processing for the Code Flash Memory

Figure 4-9 to Figure 4-10 shows the flowchart for sequence end processing for the code flash memory.

Figure 4-9 Sequence End Processing for the Code Flash Memory (1/2)

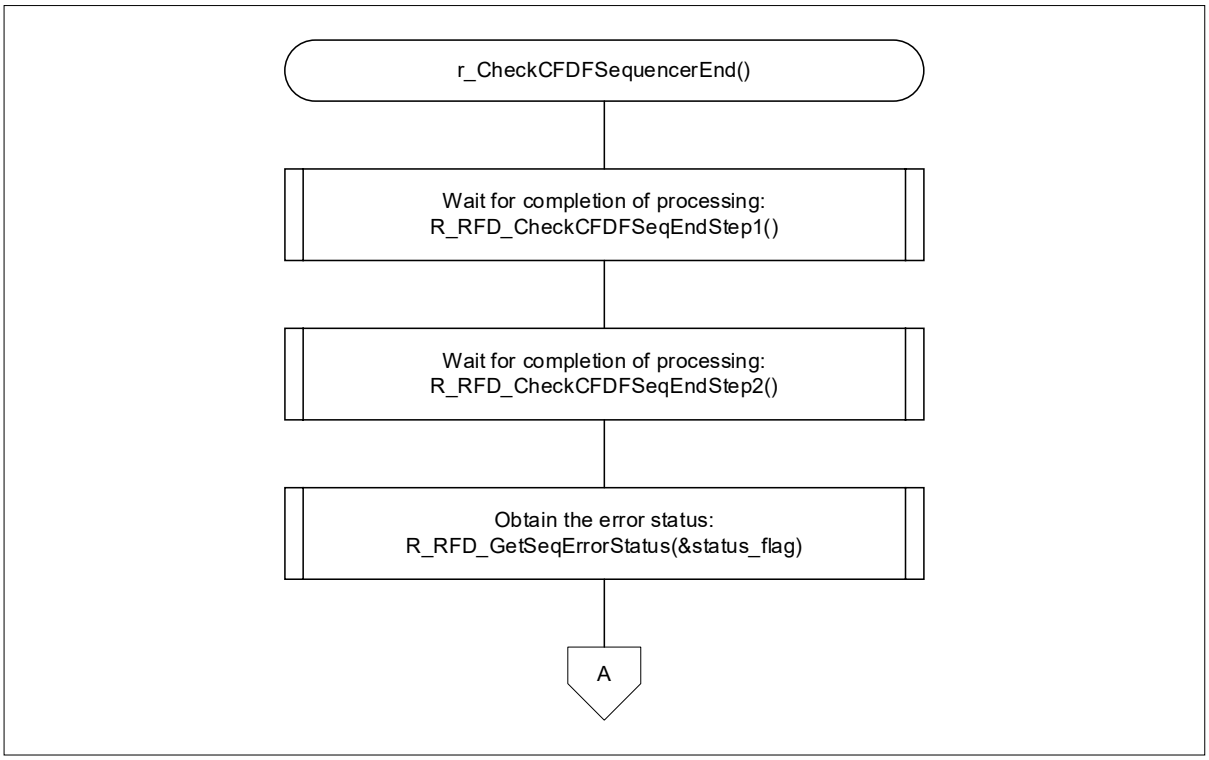
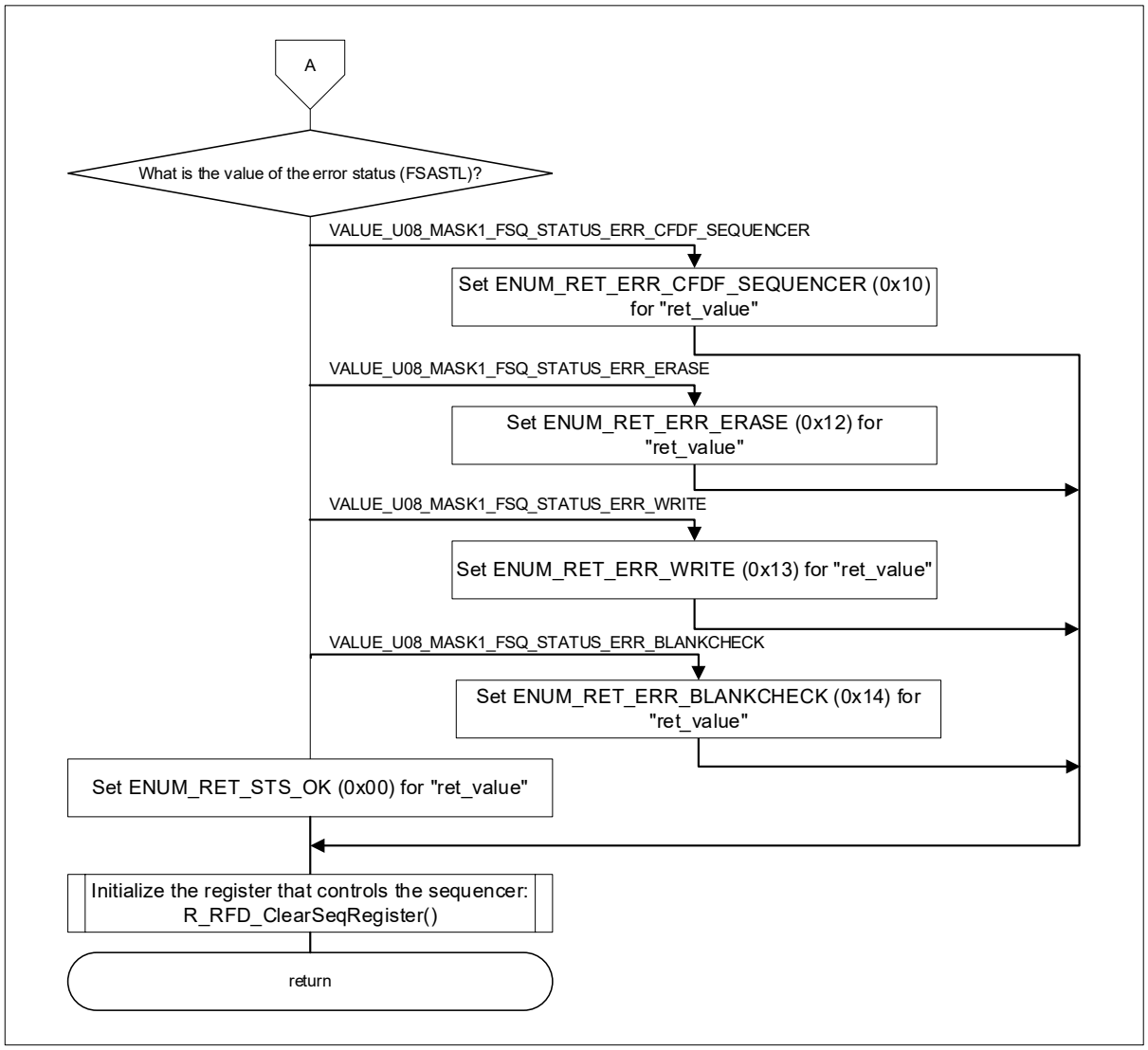


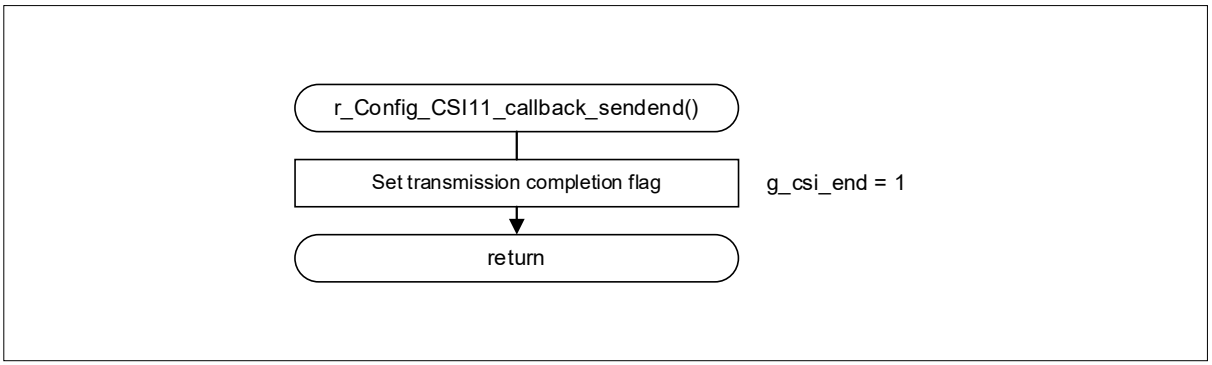
Figure 4-10 Sequence End Processing for the Code Flash Memory (2/2)



4.10.9 Processing for CSI11 Transmission Completion Interrupts

Figure 4-11 shows the flowchart for processing for CSI11 transmission completion interrupts.

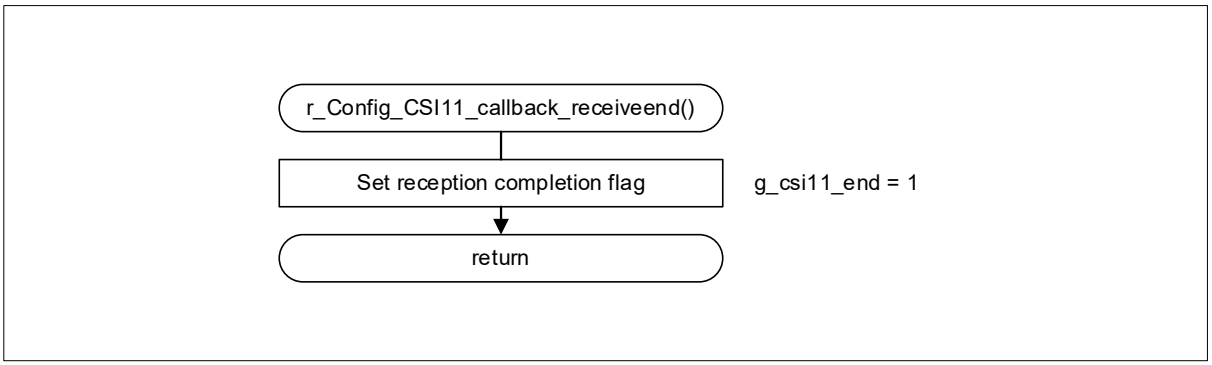
Figure 4-11 Processing for CSI11 Transmission Completion Interrupts



4.10.10 Processing for CSI11 Reception Completion Interrupts

Figure 4-12 shows the flowchart for processing for CSI11 reception completion interrupts.

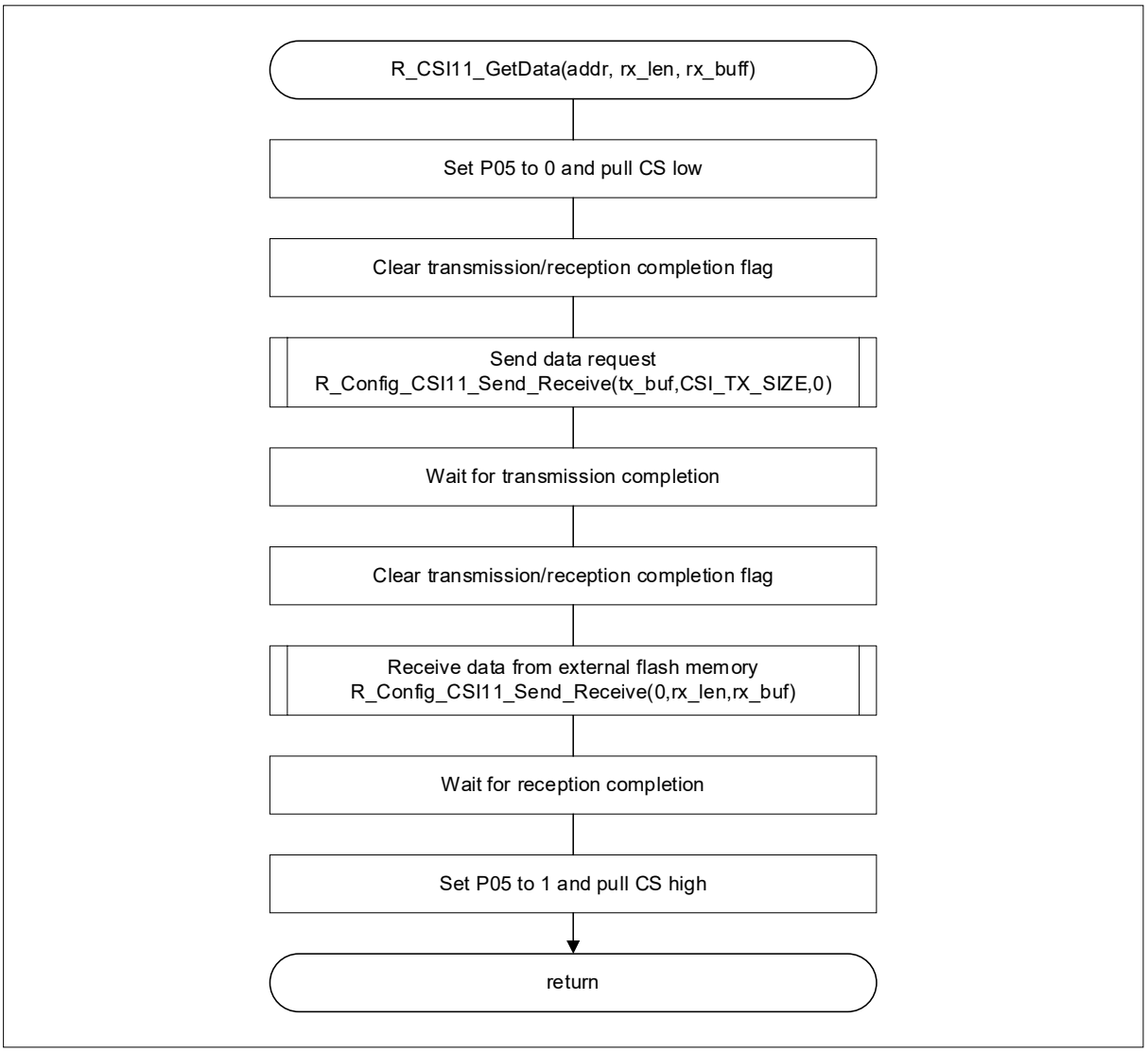
Figure 4-12 Processing for CSI11 Reception Completion Interrupts



4.10.11 Processing for Obtaining Data from External Flash Memory

Figure 4-13 shows the flowchart for obtaining data from external flash memory.

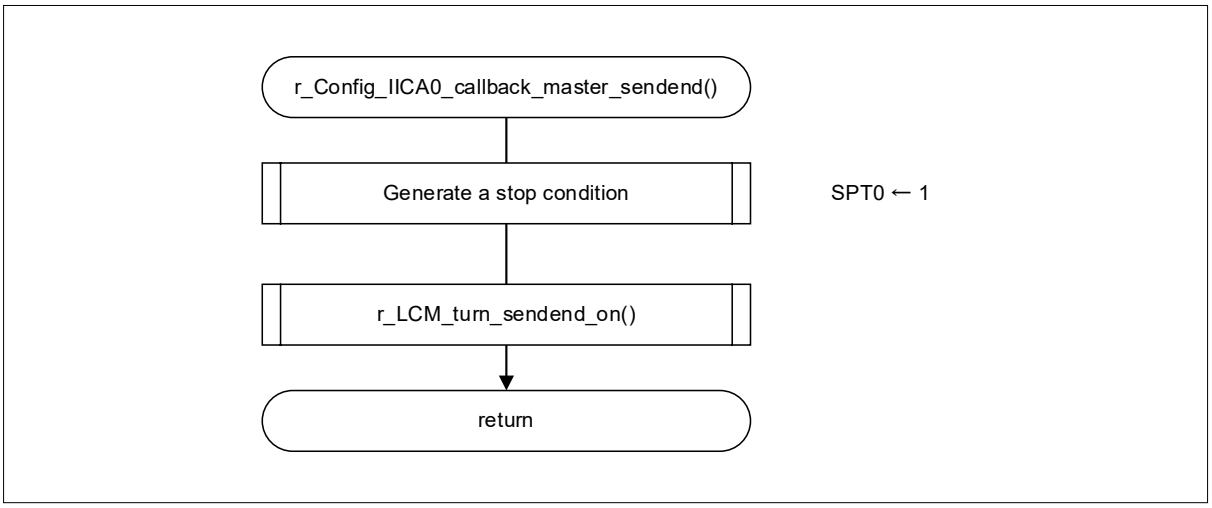
Figure 4-13 Processing for Obtaining Data from External Flash Memory



4.10.12 Processing for IICA0 Transmission Completion Interrupts

Figure 4-14 shows the flowchart for processing for IICA0 transmission completion interrupts.

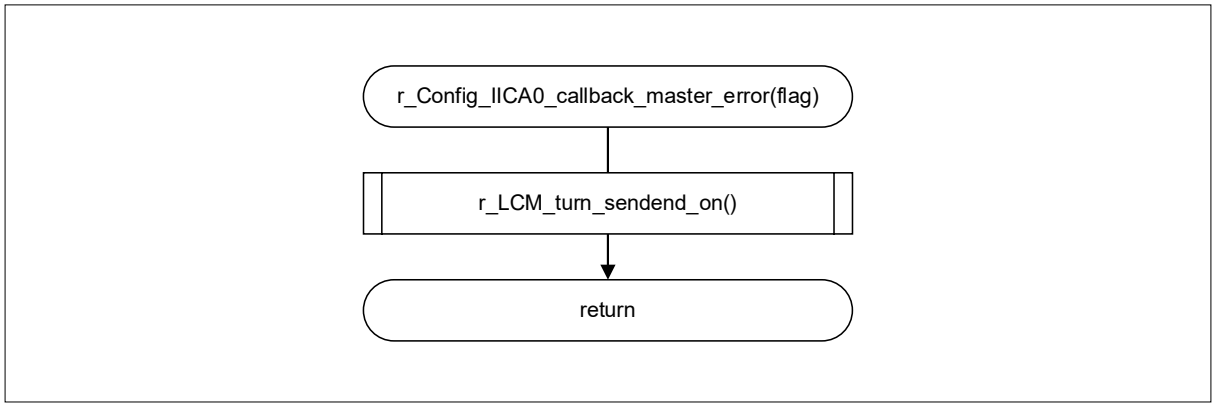
Figure 4-14 Processing for IICA0 Transmission Completion Interrupts



4.10.13 IICA0 Transmission Error Handling

Figure 4-15 shows the flowchart for IICA0 transmission error handling.

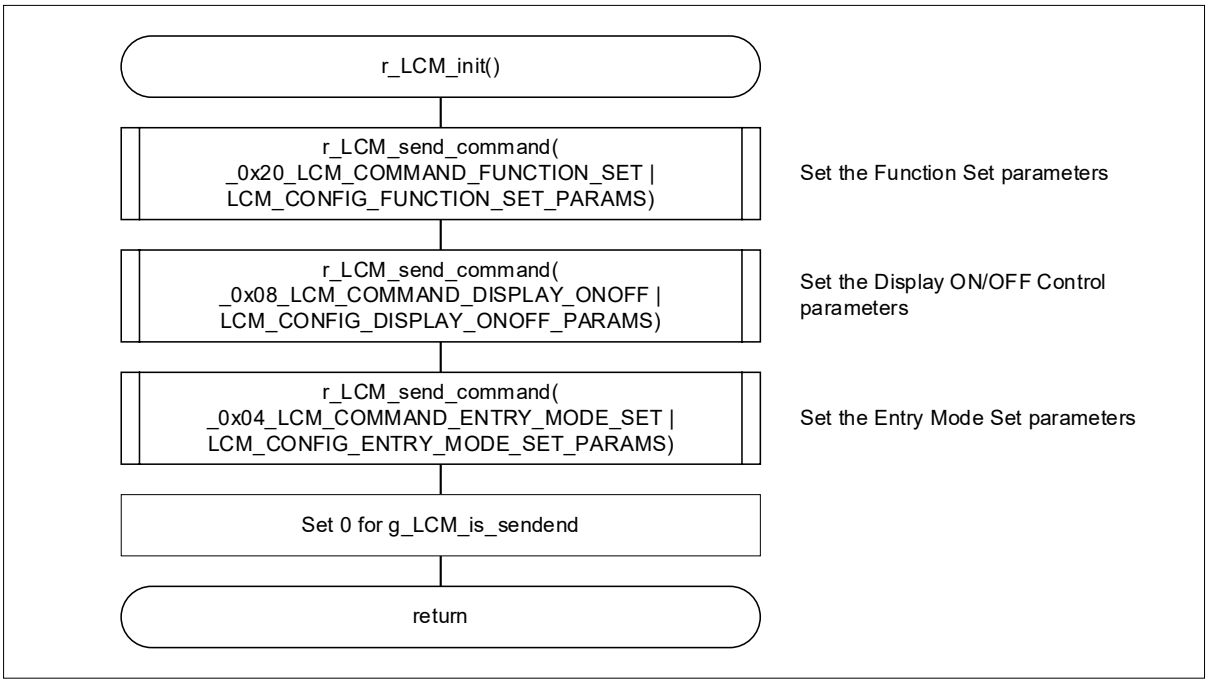
Figure 4-15 IICA0 Transmission Error Handling



4.10.14 Processing to Initialize the LCD Module

Figure 4-16 shows the flowchart for processing to initialize the LCD module.

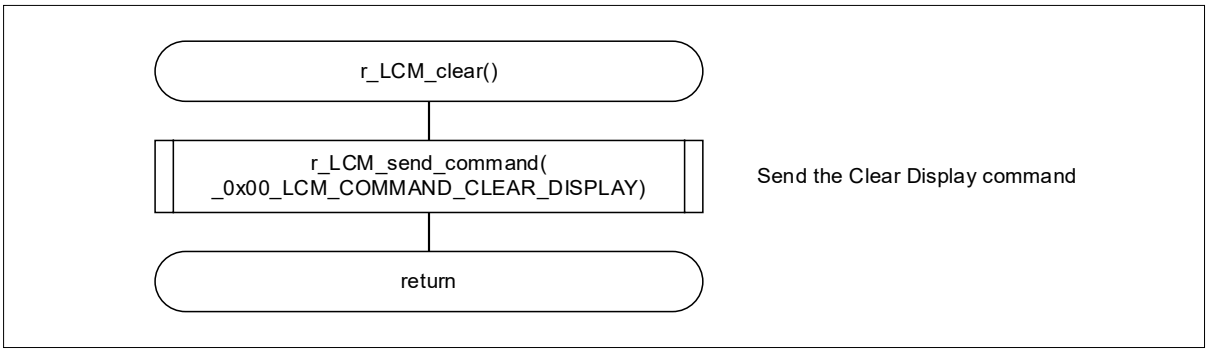
Figure 4-16 Processing to Initialize the LCD Module



4.10.15 Processing to Clear Display for the LCD Module

Figure 4-17 shows the flowchart for processing to clear display for the LCD module.

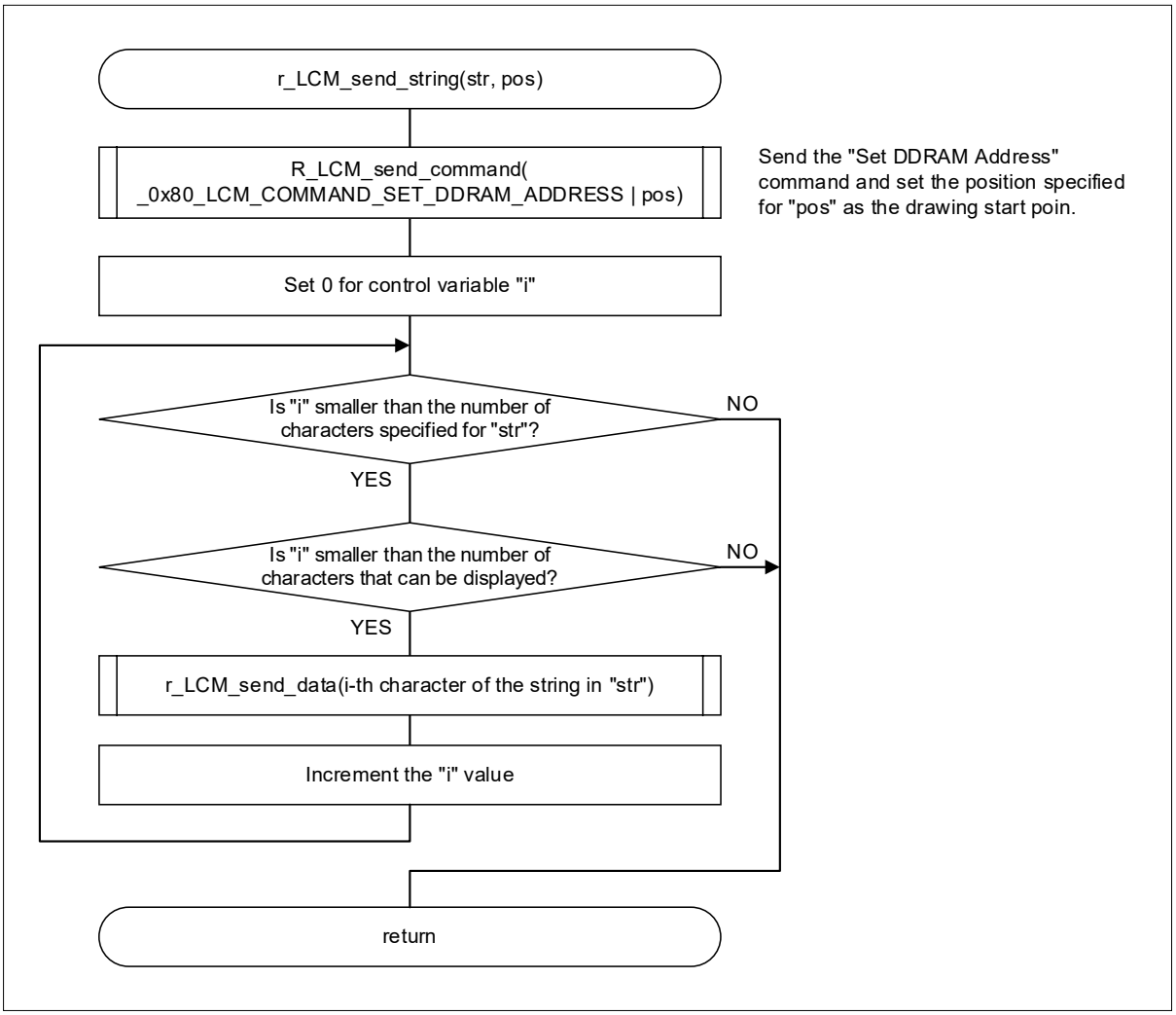
Figure 4-17 Processing to Clear Display for the LCD Module



4.10.16 Processing to Send Strings to the LCD Module

Figure 4-18 shows the flowchart for processing to send strings to the LCD module.

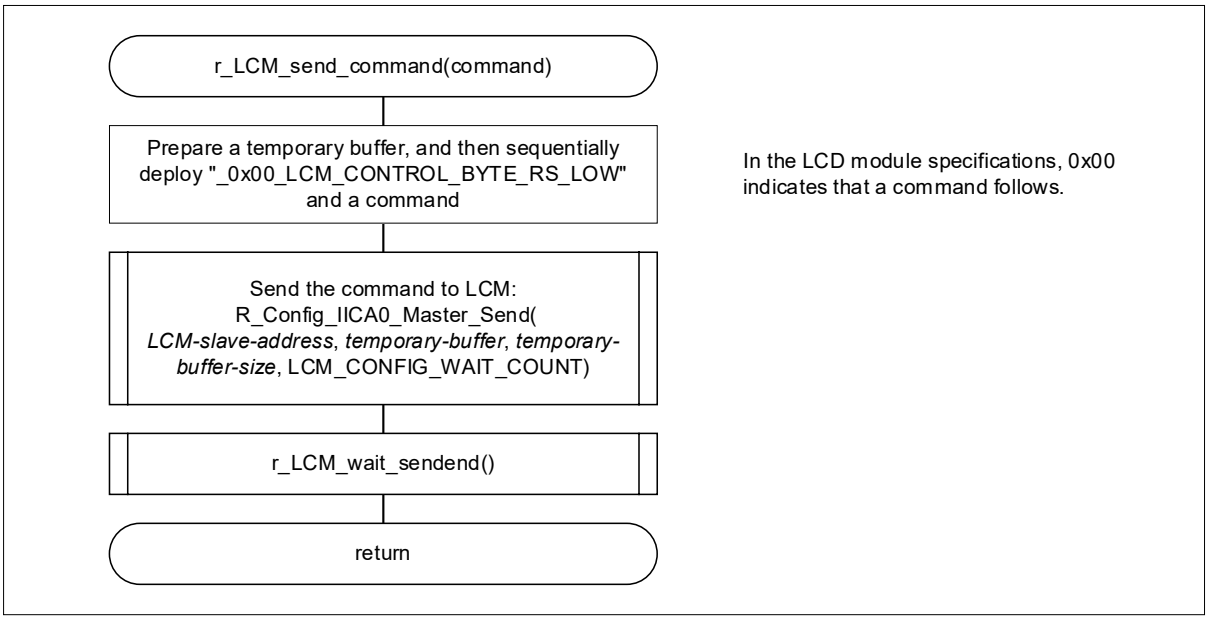
Figure 4-18 Processing to Send Strings to the LCD Module



4.10.17 Command Sending Processing for the LCD Module

Figure 4-19 shows the flowchart for command sending processing for the LCD module.

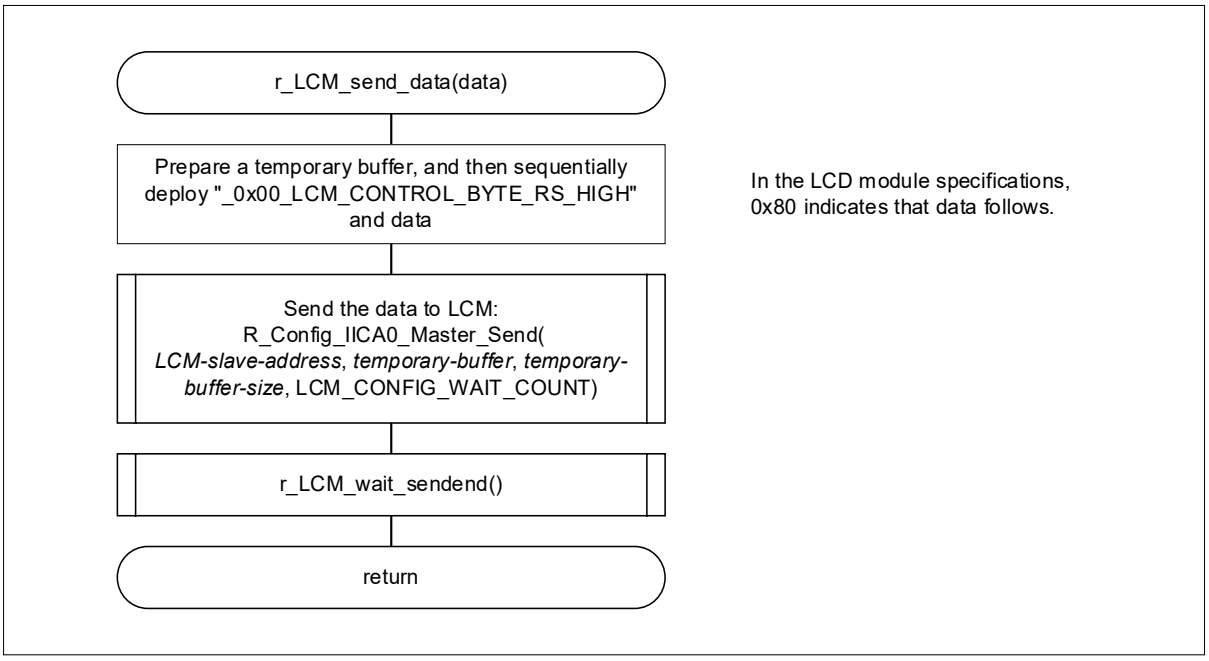
Figure 4-19 Command Sending Processing for the LCD Module



4.10.18 Processing to Send Data to the LCD Module

Figure 4-20 shows the flowchart for processing to send data to the LCD module.

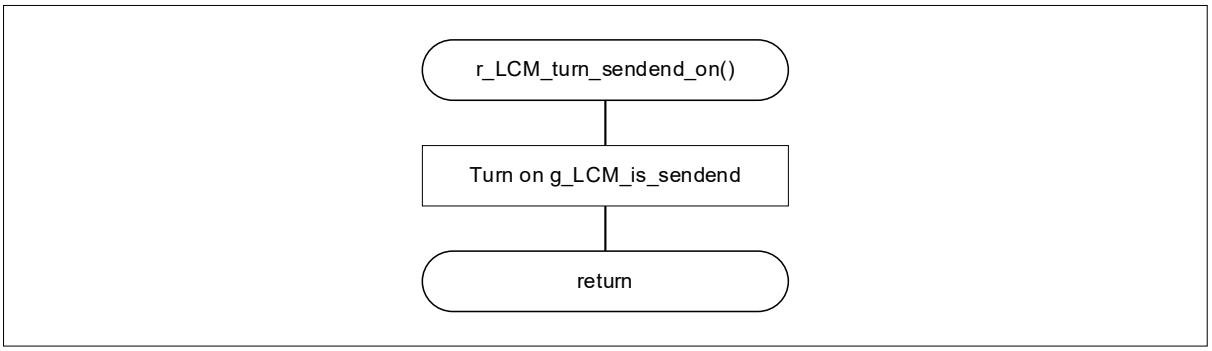
Figure 4-20 Processing to Send Data to the LCD Module



4.10.19 Communication End Flag Setting for the LCD Module

Figure 4-21 shows the flowchart for communication end flag setting for the LCD module.

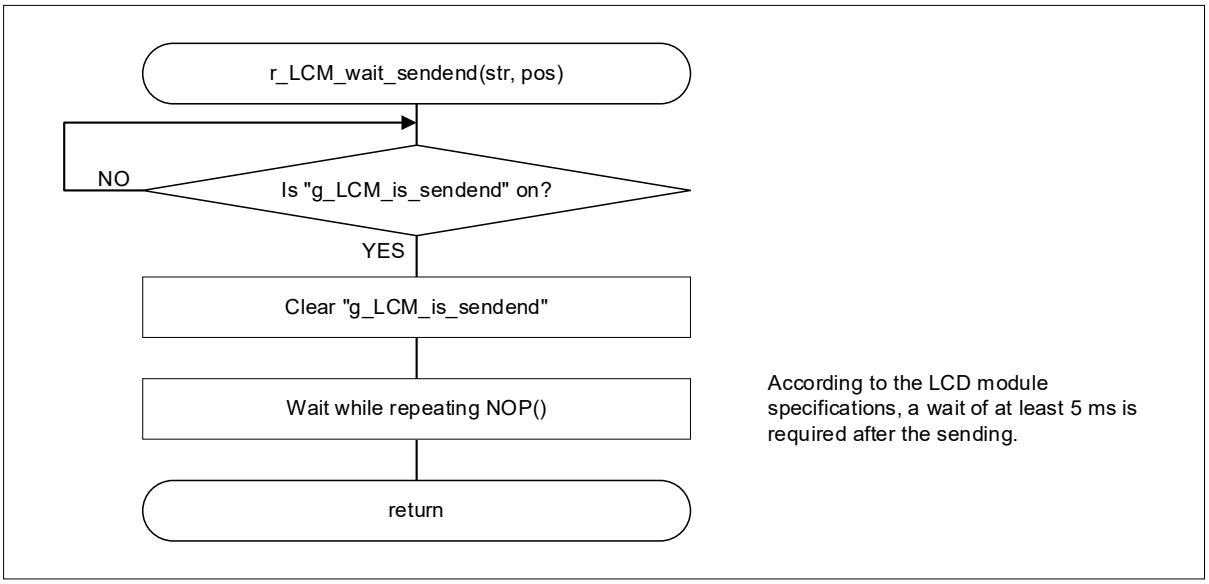
Figure 4-21 Communication End Flag Setting for the LCD Module



4.10.20 Communication End Wait Processing for the LCD Module

Figure 4-22 shows the flowchart for communication end wait processing for the LCD module.

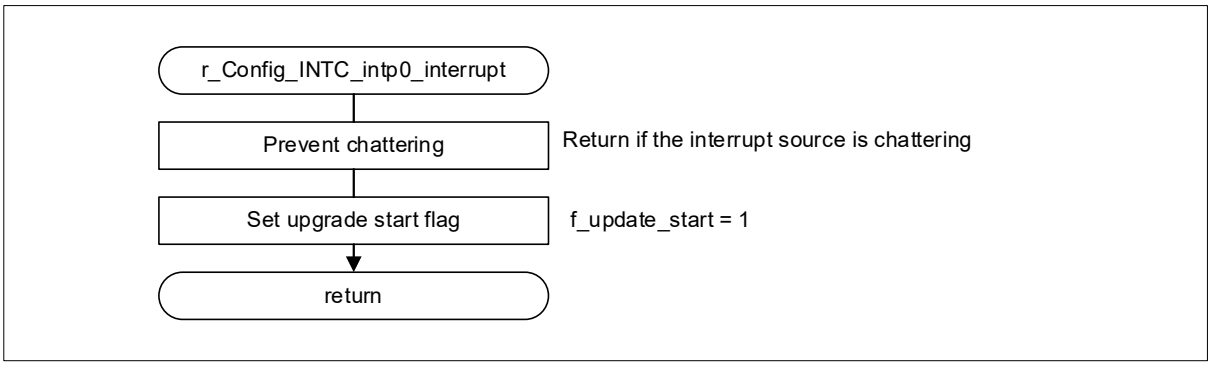
Figure 4-22 Communication End Wait Processing for the LCD Module



4.10.21 External Interrupt (INTP0) Processing

Figure 4-23 shows the flowchart for external interrupt (INTP0) processing.

Figure 4-23 External Interrupt (INTP0) Processing



5. Sample code

Sample code can be downloaded from the Renesas Electronics website.

6. Reference Documents

RL78/G23 User’s Manual: Hardware (R01UH0896J)
RL78 family user's manual software (R01US0015J)
The latest versions can be downloaded from the Renesas Electronics website.

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar. 18. 22	—	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

Contact information

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

Trademarks

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