RL78/G13

Self-Programming (Received Data via UART) CC-RL

R01AN2761EJ0101
Rev. 1.01
Feb. 27, 2024

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 self-programming library Type01.

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/G13

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 displays the information about the current version of the library on the LCD. Subsequently, the program receives data (reprogramming data) from the sending side and, after turning on the LED indicating that it is accessing flash memory, carries out self-programming to rewrite the code flash memory with the reprogramming data. When the rewrite is completed, the sample program turns off the LED and displays the information about the new version on the LCD.

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
Channel 2 of serial array unit 0	Receives data via UART.
Channel 3 of serial array unit 0	Sends data via UART.
Port I/O	Displays text on the LCD. Turns on and off the LED.

1.1 Outline of the Flash Self-Programming Library

The flash 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 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/G13 (R5F100LE) code flash memory is shown below.

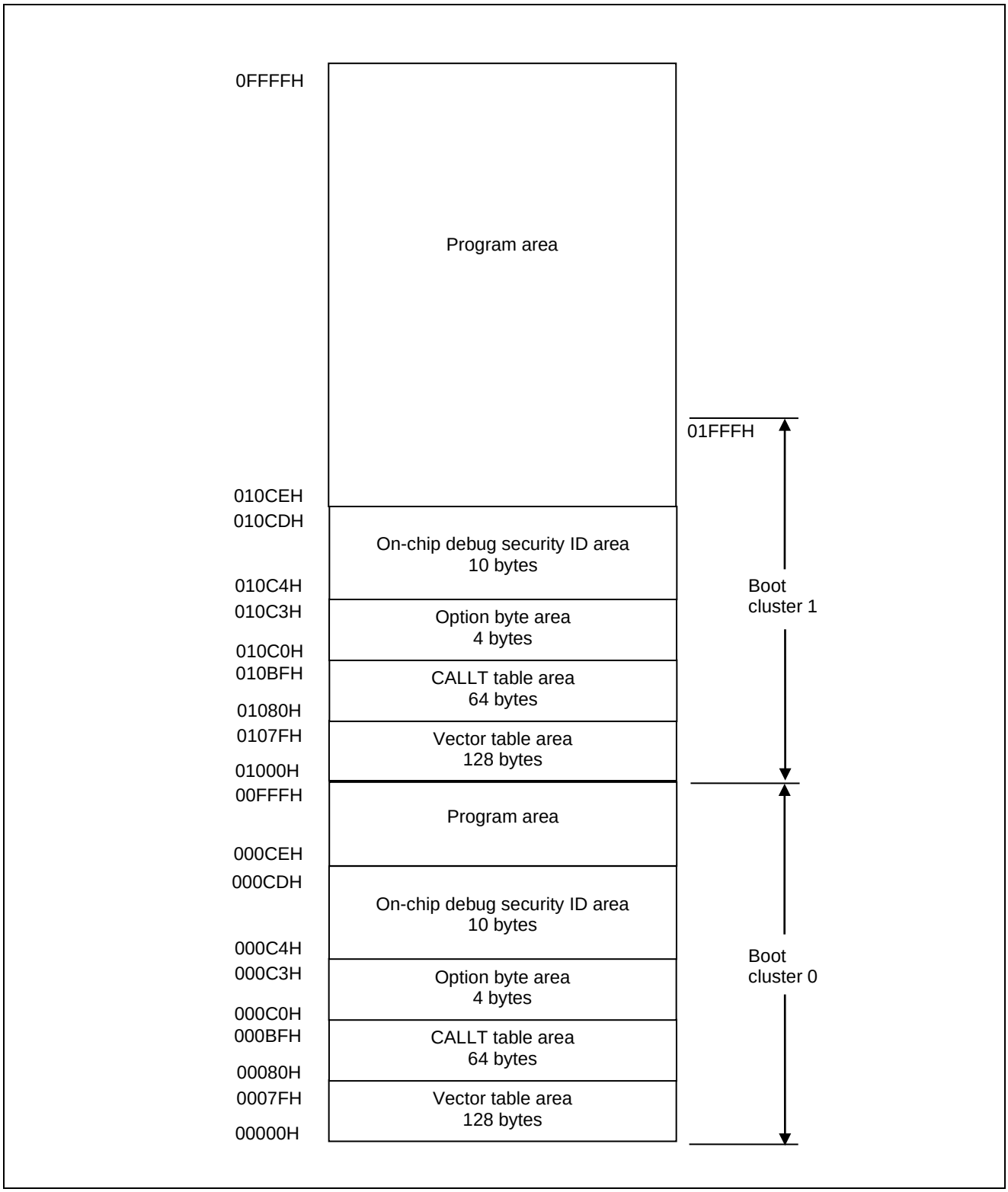


Figure 1.1 Code Flash Memory Configuration

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 (010C0H to 010C3H) in boot cluster 1. Accordingly, place the same values in the area (010C0H to 010C3H) as those in the area (000C0H to 000C3H) when using the boot swap function.

The features of the RL78/G13 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 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/G13 is provided with a library for flash memory self-programming. Flash memory self-programming is accomplished by calling functions of the flash self-programming library from the reprogramming program.

The flash self-programming library for the RL78/G13 controls flash memory reprogramming using a sequencer (a dedicated circuit for controlling flash memory). The code flash memory cannot be referenced while control by the sequencer is in progress. When the user program needs to be run while the sequencer control is in progress, therefore, it is necessary to relocate part of the segments for the flash self-programming library and the reprogramming program in RAM when erasing or reprogramming the code flash memory or making settings for the security flags. If there is no need to run the user program while the sequencer control is in progress, it is possible to keep the flash self-programming library and reprogramming program on ROM (code flash memory) for execution.

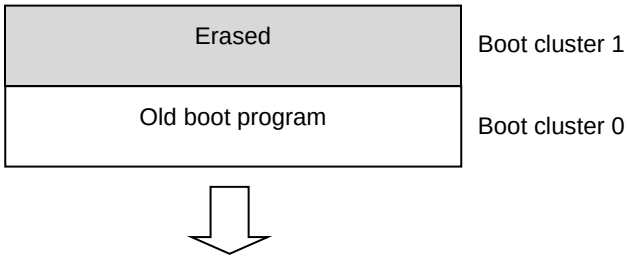
1.3.1 Boot Swap Function

When reprogramming of the area where vector table data, the basic functions of the program, and flash sself-programming library are allocated fails due to a temporary power blackout or a reset caused by an external factor, the data that is being reprogrammed will be corrupted, as the result of which the restarting of the user program or reprogramming cannot be accomplished when a reset is subsequently performed. This problem is be avoided by the introduction of the boot swap function.

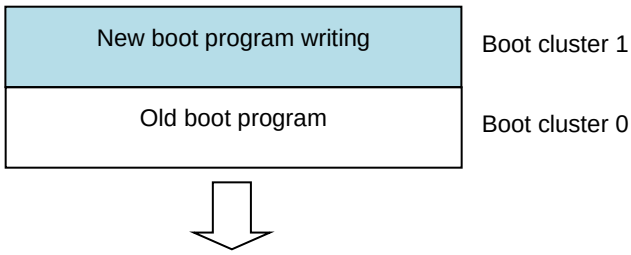
The boot swap function swaps between boot cluster 0 which is the boot program area and boot cluster 1 which is the target of boot swapping. A new program is written into boot cluster 1 before reprogramming is attempted. This boot cluster 1 is swapped with boot cluster 0 and boot cluster 1 is designated as the boot program area. In this configuration, even when a temporary power blackout occurs while the boot program area is being reprogrammed, the system boot will start at boot cluster 1 on the next reset start, thus ensuring the normal execution of the programs.

The outline image of boot swapping is shown in the figure below.

(1) Erasing boot cluster 1
Call the FSL_Erase function to erase boot cluster 1 (blocks 4 to 7).

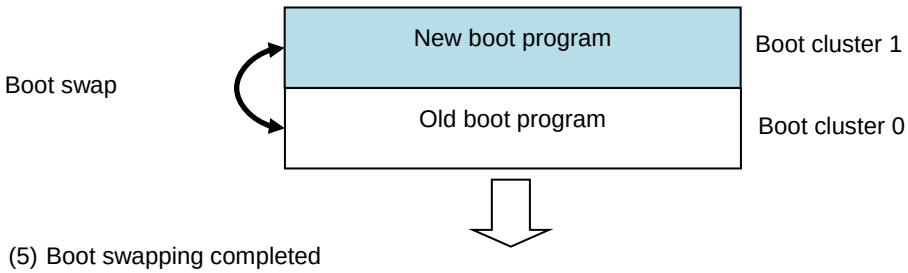


(2) Writing the new boot program into boot cluster 1
Call the FSL_Write function to write the new boot program into boot cluster 1 and call the FSL_IVerify function to verify boot cluster 1.
The steps that have been performed up to here ensure that the programs will run normally even when the programming of the new boot program fails due to a temporary power blackout or reset because the system boot is started by the old boot program.



(3) Setting the boot swap bit
Call the FSL_InvertBootFlag function to invert the state of the boot flag.
When a temporary power blackout or reset occurs after the state of the boot flag is inverted, the programs will run normally because the system boot is started by the new boot program whose reprogramming has been completed.

(4) When a reset occurs
When a reset occurs, boot clusters 0 and 1 are swapped.



(5) Boot swapping completed

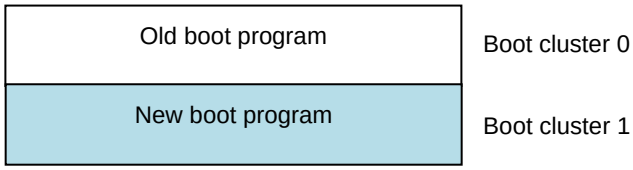


Figure 1.2 Outline of Boot Swapping

1.3.2 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.

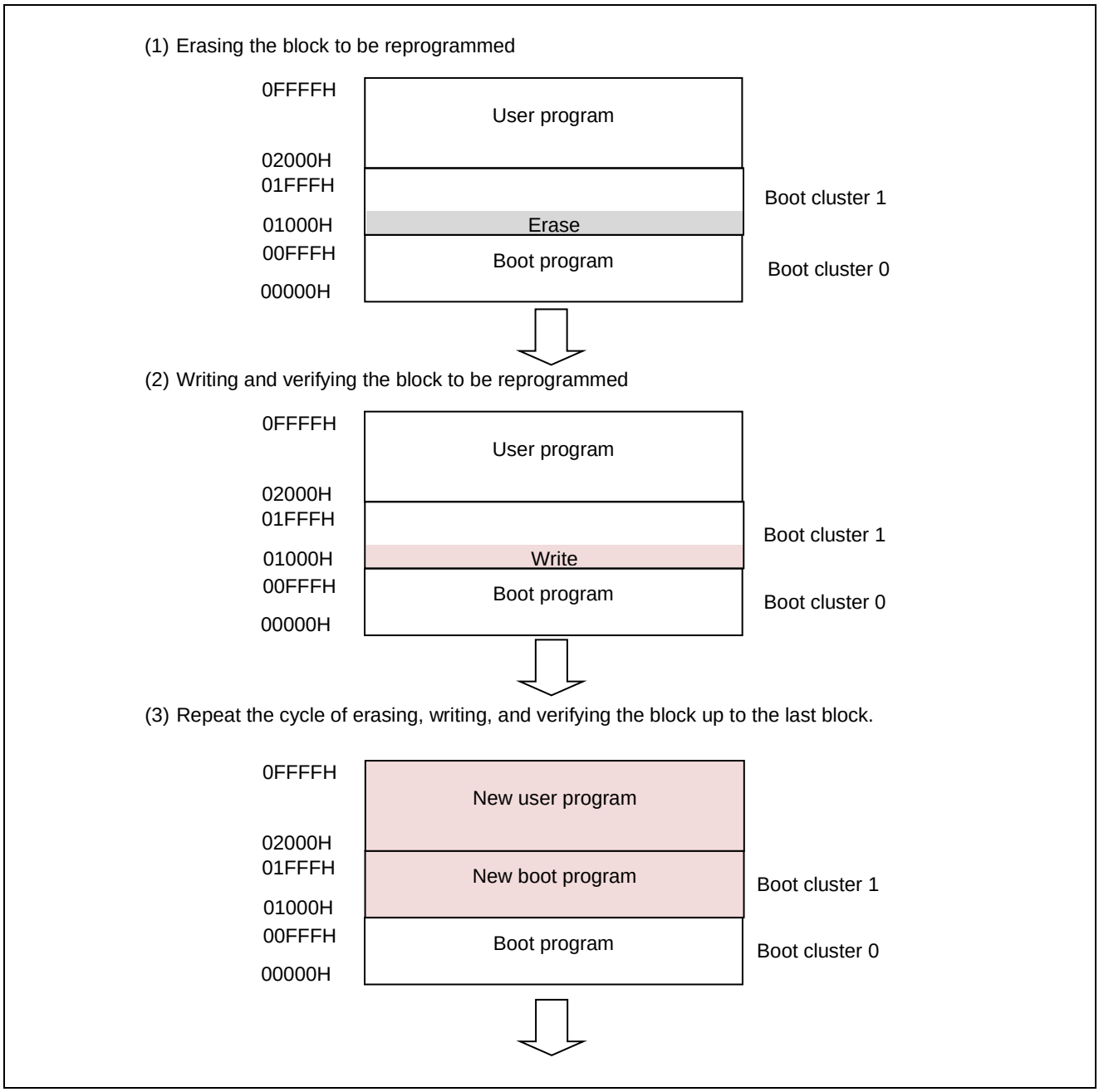


Figure 1.3 Outline of Flash Memory Reprogramming (1/2)

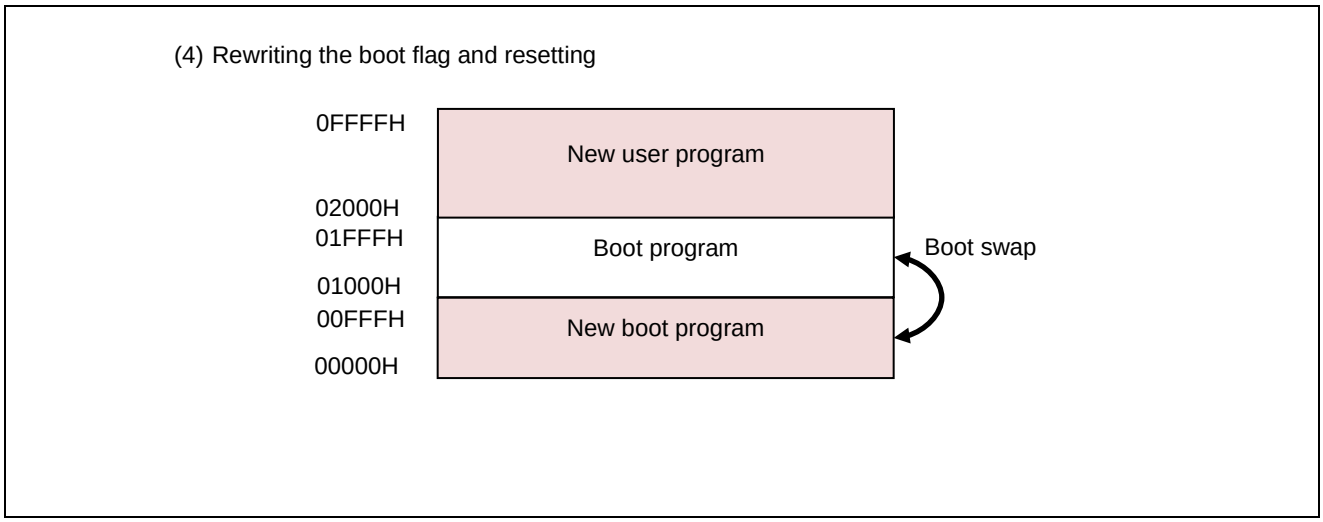


Figure 1.4 Outline of Flash Memory Reprogramming (2/2)

1.3.3 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.

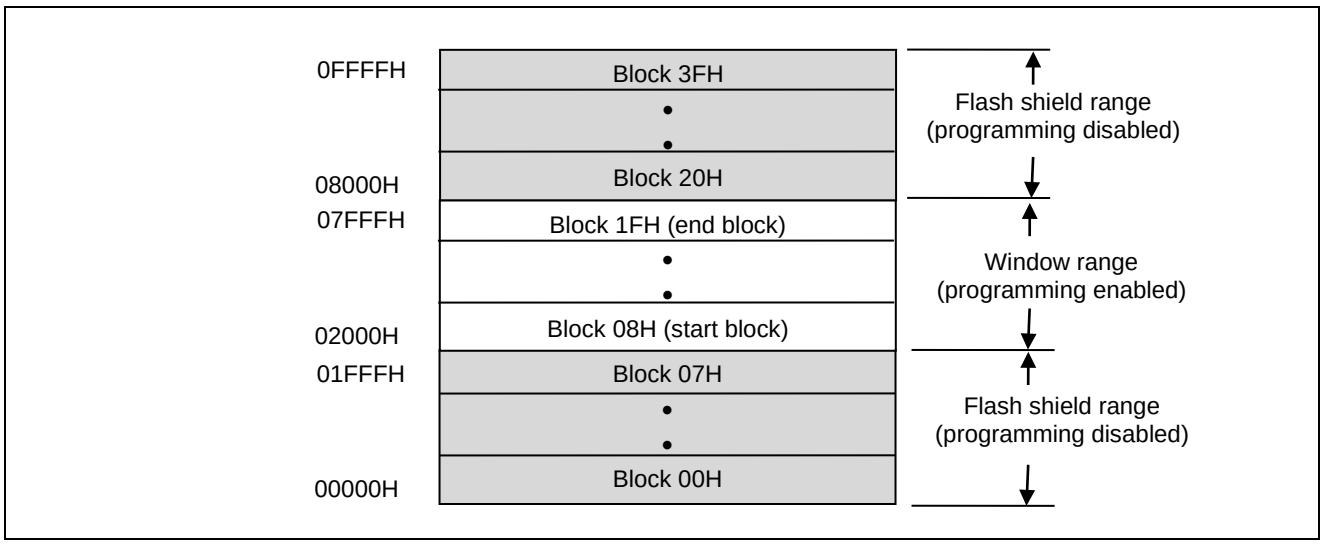


Figure 1.5 Outline of the Flash Shield Window

1.4 How to Get the Flash 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 “r01an2761ej_flash”.

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

librl78 folder : fsl.lib

The flash self-programming library can be obtained from the following URL:

http://www.renesas.com/products/tools/flash_prom_programming/flash_libraries/index.jsp

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/G13 (R5F100LEA)
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+)	CS+ V8.11.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.13.00 from Renesas Electronics Corp.
Integrated development environment (e^2 studio)	e^2 studio V2024-01 (24.1.0) from Renesas Electronics Corp.
C compiler (e^2 studio)	CC-RL V1.13.00 from Renesas Electronics Corp.
Board to be used	Renesas Starter Kit for RL78/G13 (R0K50100LS000BE)
Flash self-programming library (Type, Ver)	FSLRL78 Type01, Ver 3.00 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/G13 Initialization (R01AN2575E) Application Note
- RL78/G13 Serial Array Unit (UART Communication) (R01AN2571E) Application Note

4. Description of the Hardware

4.1 Hardware Configuration Example

Figure 4.1 shows an example of the hardware configuration used for this application note.

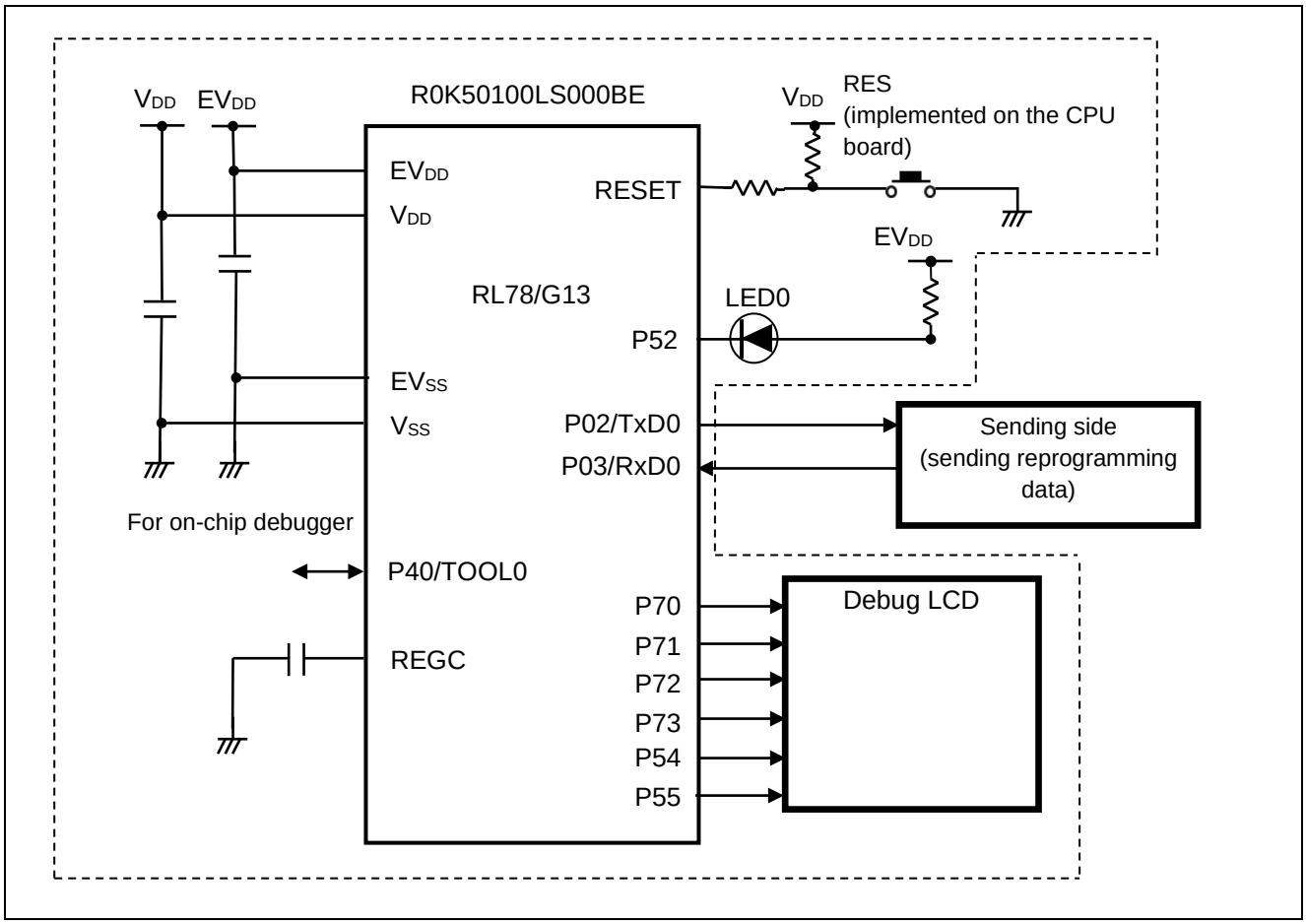


Figure 4.1 Hardware Configuration

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 4.1 lists pins to be used and their functions.

Table 4.1 Pins to be Used and their Functions

Pin Name	I/O	Description
P02/ANI17/SO10/TxD1	Output	UART serial data transmit pin
P03/ANI16/SI10/RxD1/SDA10	Input	UART serial data receive pin
P52	Output	LED0 (indicating flash memory access status) on/off control
P54	Output	Debug LCD control
P55	Output	Debug LCD control
P70/KR0/SCK21/SCL21	Output	Debug LCD control
P71/KR1/SI21/SDA21	Output	Debug LCD control
P72/KR2/SO21	Output	Debug LCD control
P73/KR3/SO01	Output	Debug LCD control

5. Description of the Software

5.1 Communication Specifications

The sample program covered in this application note receives reprogramming data via the UART 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, and, if the command terminates normally, returns a normal response (0x01) to the sending side. If the command terminates abnormally, the program returns no response, displays "ERROR!" on the LCD, and suppresses the execution of the subsequent operations. This section describes the necessary UART communication settings and the specifications for the commands.

Table 5.1 UART Communication Settings

Data bit length [bit]	8
Data transfer direction	LSB first
Parity setting	No parity
Transfer rate [bps]	115200

5.1.1 START Command

When the sample program receives the START command, it performs initialization processing for flash memory self-programming. When the command terminates normally, the program returns a normal response (0x01) to the sending side. In the case of an abnormal termination, the sample program returns no response, 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 writes the data it received into flash memory, and performs verify processing each time it completes the write of one block. The sample program returns a normal response (0x01) to the sending side on normal termination of the command. In the case of an abnormal termination, the sample program returns no response, 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 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. When the sample program receives the END command, it returns no response to the sending side whether the command terminates normally or abnormally.

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:

Turns on LED1 which indicates that flash memory is being accessed, performs initialization for flash memory self-programming, and returns a normal response (0x01) upon normal termination.

(3) Sending side:

Sends the WRITE command and reprogramming data (256 bytes).

(4) Sample program:

Writes the data it received into the code flash memory. The write address starts at 0x1000 (start of boot cluster 1). Subsequently, it is incremented by the receive data size (size of reprogramming data: 256 bytes) each time the sample program receives the WRITE command and reprogramming data.

The program performs verify processing when the rewrite of 1 block (1024 bytes) is completed.

When all of these steps terminate normally, the sample program returns a response (0x01).

(5) Steps (3) and (4) are repeated until the reprogramming of all data is completed.

(6) Sending side:

Sends the END command.

(7) Sample program:

Performs verify processing on the block that is currently subjected to reprogramming. The sample program then inverts the state of the boot flag, and generates a reset after turning off LED0 which indicates that flash memory is 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 displays the information about the current version of the library on the LCD. Subsequently, the program receives data (reprogramming data) from the sending side and, after turning on the LED indicating that it is accessing flash memory, carries out self-programming to rewrite the code flash memory with the reprogramming data. When reprogramming is completed, the sample program turns off the LED and displays the information about the new version on the LCD.

(1) Initializes the SAU0 channels 2 and 3.

<Setting conditions>

- Uses the SAU0 channels 2 and 3 as UART.
- Uses the P02/TxD1 pin for data output and the P03/RxD1 pin for data input.
- Sets the data length to 8 bits.
- Sets the order of data transfer mode to LSB first.
- Sets the parity setting to “No parity”.
- Sets the receive data level to standard.
- Sets the transfer rate to 115200 bps.

(2) Sets up the I/O port.

<Setting conditions>

- LED on/off control port (LED0): Sets P52 for output.

(3) Disables interrupts.

(4) Starts the UART1.

(5) Initializes the LCD and displays on the LCD the string that is set to the constant LCD_STRING.

(6) Enters the HALT mode and waits for data from the sending side.

- Switches into the normal operation mode from the HALT mode upon a UART receive end interrupt request.

(7) Upon receipt of a START command (0x02) from the sending side, performs initialization for self-programming.

- Sets P52 to the low level and turns on LED0 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 : 32 [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_PrepareExtFunctions function to make available the flash memory functions (extended 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 63, calls the FSL_SetFlashShieldWindow function to set the start block of the flash shield window to block 0 and the end block to block 63.

(8) Sets the write destination address to 0x1000 (start of boot cluster 1).

(9) Sends a normal response (0x01) to the sending side.

(10) Receives the WRITE command (0x03) and reprogramming data (256 bytes).

(11) Computes the reprogramming target block from the write destination address.

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

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

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

(15) Adds the write size to the write destination address.

(16) Sends a normal response (0x01) to the sending side.

(17) Receives the WRITE command and reprogramming data (256 bytes) or the END command (0x04).

(18) Repeats steps (14) to (17) until 1 block (1024 bytes) of programming is completed or an END command (0x04) is received from the sending side. Proceeds with the next step when 1 block 1024 bytes) of programming is completed or an END command (0x04) is received from the sending side.

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

(20) Repeats steps (11) to (19) unless an END command (0x04) is received from the sending side. Proceeds with the next step when an END command is received.

(21) Calls the FSL_InvertBootFlag function to invert the state of the boot flag. Boot clusters 0 and 1 will then be swapped at reset time.

(22) Sets P52 to the high level, turns off LED0 indicating that flash memory is being accessed, then calls the FSL_ForceReset function to generate an internal reset.

Caution: When flash memory self-programming could not be terminated normally (error occurring during processing), the sample program displays "ERROR!" on the LCD and suppresses the execution of the subsequent operations.

5.3 File Configuration

Table 5.2 lists the additional functions for files that are automatically generated in the integrated development environment and other additional files.

Table 5.2 List of Additional Functions and Files

File Name	Outline	Remarks
r_main.c	Main module	Additional functions: R_MAIN_PacketAnalyze R_MAIN_SelfExecute R_MAIN_SelfInitialize R_MAIN_WriteExecute
r_cg_serial_user.c	SAU module	Additional functions: R_UART1_ReceiveStart R_UART1_SendStart
lcd.c	DebugLCD module	Controls DebugLCD included in Renesas Starter Kit for RL78/G13.

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/G13 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. When using the boot swap function for self-programming, it is necessary to set the same values that are set in 000C0H to 000C3H also in 010C0H to 010C3H because the bytes in 000C0H to 000C3H are swapped with the bytes in 010C0H to 010C3H.

5.5 On-chip Debug Security ID

The RL78/G13 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 010C4H to 010CDH because bytes in 000C4H to 000CDH are swapped with the bytes in 010C4H to 010CDH.

5.6 Link Option

The –start option, which is one of the link options, is provided for allocating the Flash Self-Programming Library Type01 to a ROM area.

Use the –start option to specify all sections for which settings are required by the Flash Self-Programming Library Type01.

Caution: For details on the link option procedures, refer to RL78 Compiler CC-RL User's Manual (R20UT3123E).

5.7 List of Constants

Table 5.4 lists the constants for the sample program.

Table 5.4 Constants for the Sample Program

Constant	Setting	Description
LCD_DISPLAY	“Ver 1.0 ”	String to be displayed on the LCD (version information) Display up to eight characters on the LCD.
ERR_DISPLAY	“ERROR! “	String to be displayed on the LCD at occurrence of error
NORMAL_END	0x00	Normal termination
ERROR	0xFF	Abnormal termination
NO_RECIEVE	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	0x02	START command
WRITE	0x03	WRITE command
END	0x04	END command
FULL_SPEED_MODE	0x00	Argument to flash memory self-programming library initialization function: Set operation mode to full-speed mode.
FREQUENCY_32M	0x20	Argument to flash memory self-programming library initialization function: RL78/G13's operating frequency = 32 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	0x3F	End block number of flash shield window
BLOCK_SIZE	0x400	One block size of code flash memory (1024 bytes)
TXSIZE	0x01	Size of response data to be sent to the sending side
RXSIZE	0x102	Size of receive buffer

5.8 List of Functions

Table 5.5 lists the functions that are used in this sample program.

Table 5.5 List of Functions

Function Name	Outline
R_UART1_Start	Starts UART1.
R_UART1_ReceiveStart	Receives data via UART1.
R_MAIN_PacketAnalyze	Analyzes receive data.
R_MAIN_SelfExecute	Executes flash memory self-programming.
R_MAIN_SelfInitialize	Executes initialization for flash memory self-programming.
R_MAIN_WriteExecute	Executes flash memory reprogramming.
R_UART1_SendStart	Sends data via UART1.

5.9 Function Specifications

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

[Function Name] R_UART1_Start

Synopsis	Start UART1.	
Header	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h	
Declaration	void R_UART1_Start(void)	
Explanation	This function starts channels 2 and 3 of the serial array unit 0 and places them in communication wait state.	
Arguments	None	
Return value	None	
Remarks	None	

[Function Name] R_UART1_ReceiveStart

Synopsis	Receive data via UART1.	
Header	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h	
Declaration	uint8_t R_UART1_ReceiveStart(uint16_t *rxlength, uint8_t *rxbuf)	
Explanation	This function stores the received data in the receive buffer rxbuf and its data length [in bytes] in rxlength.	
Arguments	rxlength	Address of area storing receive data length [in bytes]
	rxbuf	Address of receive data buffer
Return value	Normal termination: NORMAL_END	
	Abnormal termination: ERROR	
Remarks	None	

[Function Name] R_MAIN_PacketAnalyze

Synopsis	Analyze receive data.	
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_serial.h r_cg_userdefine.h	
Declaration	uint8_t R_MAIN_PacketAnalyze(uint16_t rxlength, uint8_t *rxbuf)	
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	rxlength	Address of area storing receive data length [in bytes]
	rxbuf	Address of receive data buffer
Return value	START command received: START	
	WRITE command received: WRITE	
	END command received: END	
	Command parameter error or checksum error: ERROR	
Remarks	None	

[Function Name] R_MAIN_SelfExecute

Synopsis	Execute flash memory self-programming.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_serial.h r_cg_userdefine.h fsl.h fsl_types.h
Declaration	void R_MAIN_SelfExecute(void)
Explanation	This function executes flash memory self-programming.
Arguments	None
Return value	None
Remarks	None

[Function Name] R_MAIN_SelfInitialize

Synopsis	Execute initialization for flash memory self-programming.
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_serial.h r_cg_userdefine.h fsl.h fsl_types.h
Declaration	uint8_t R_MAIN_SelfExecute(void)
Explanation	This function executes initialization prior to flash memory self-programming.
Arguments	None
Return value	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] R_MAIN_WriteExecute

Synopsis	Execute flash memory reprogramming.	
Header	r_cg_macrodriver.h r_cg_cgc.h r_cg_port.h r_cg_serial.h r_cg_userdefine.h fsl.h fsl_types.h	
Declaration	uint8_t R_MAIN_SelfExecute(uint32_t WriteAddr)	
Explanation	This function reprograms the flash memory.	
Arguments	WriteAddr	Write start address
Return value	Normal termination: NORMAL_END Abnormal termination: ERROR	
Remarks	None	

[Function Name] R_UART1_SendStart

Synopsis	Send data via UART1.	
Header	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h	
Declaration	uint8_t R_UART1_SendStart(uint16_t *txlength, uint8_t *txbuf)	
Explanation	This function transmits the number of data bytes specified in txlength [bytes] from txbuf.	
Arguments	txlength	Transmit data length [in bytes]
	txbuf	Address of transmit data buffer
Return value	Normal termination: NORMAL_END	
	Parameter error (txlength is 0 or less): ERROR	
Remarks	None	

5.10 Flowcharts

Figure 5.1 shows the overall flow of the sample program described in this application note.

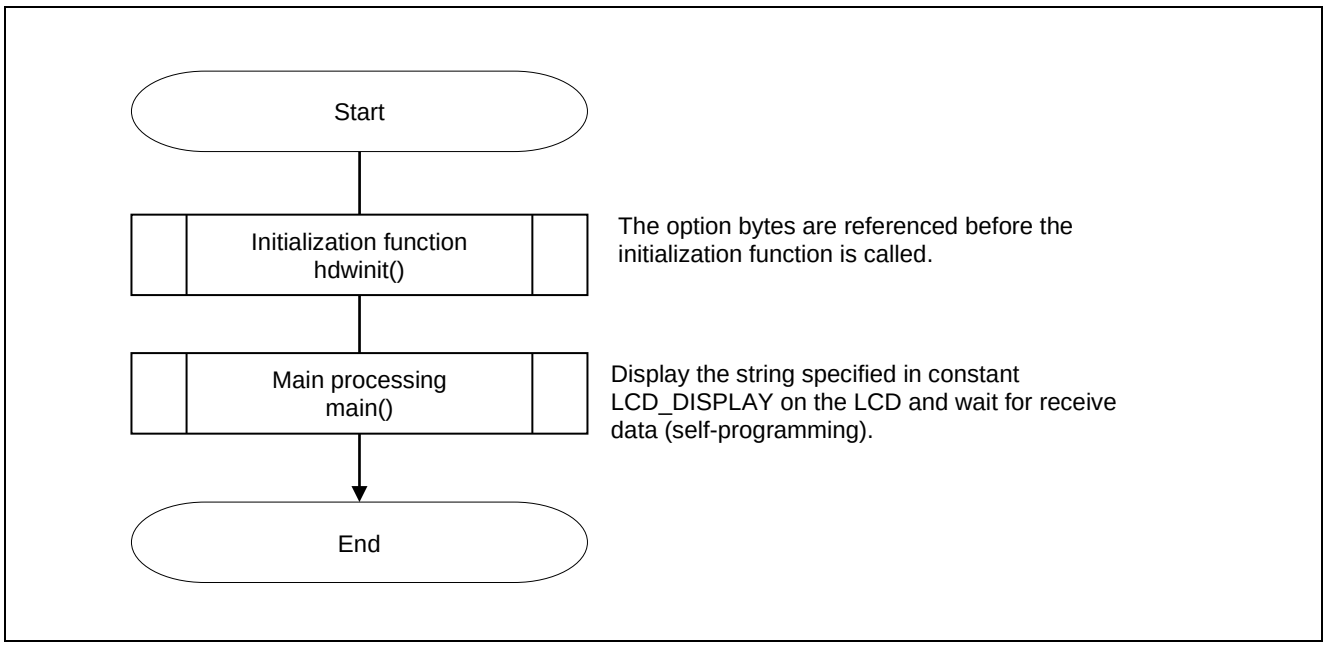


Figure 5.1 Overall Flow

5.10.1 Initialization Function

Figure 5.2 shows the flowchart for the initialization function.

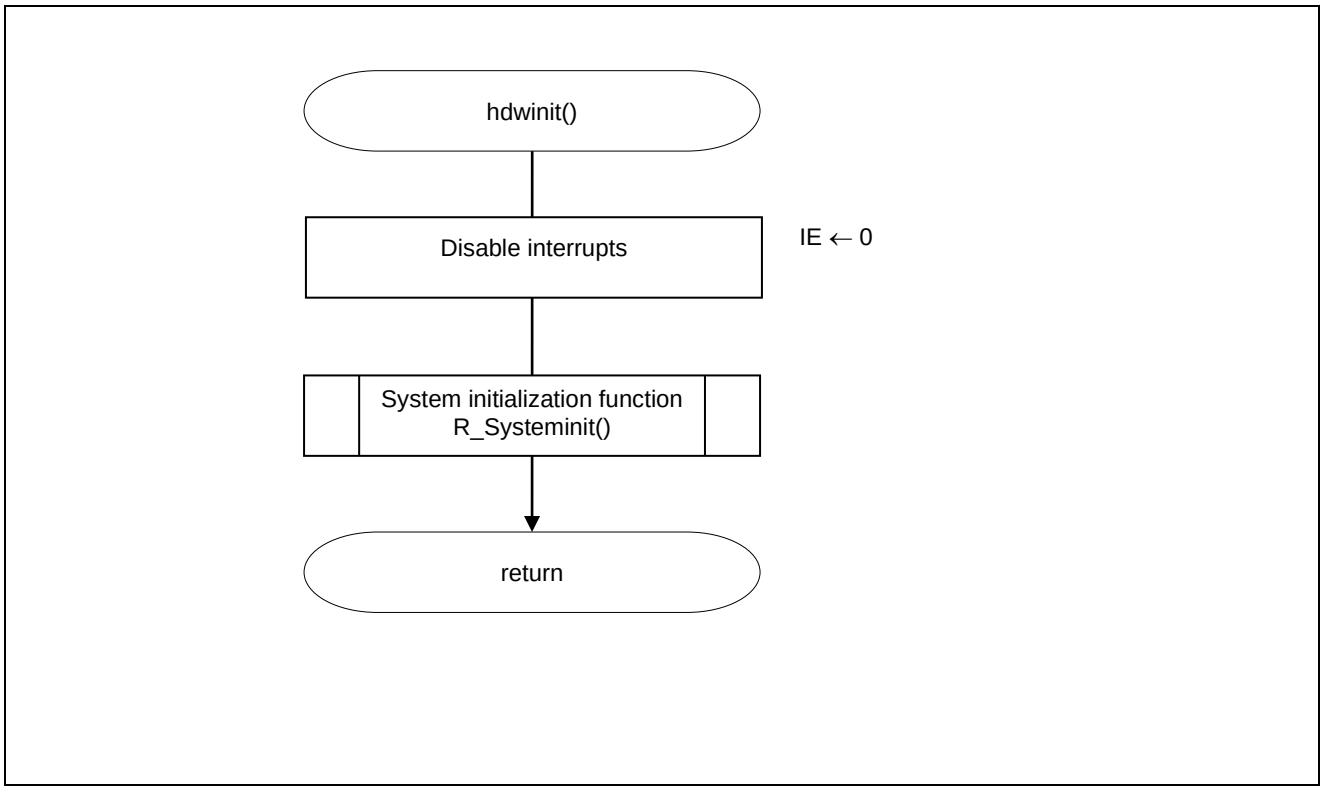


Figure 5.2 Initialization Function

5.10.2 System Initialization Function

Figure 5.3 shows the flowchart for the system initialization function.

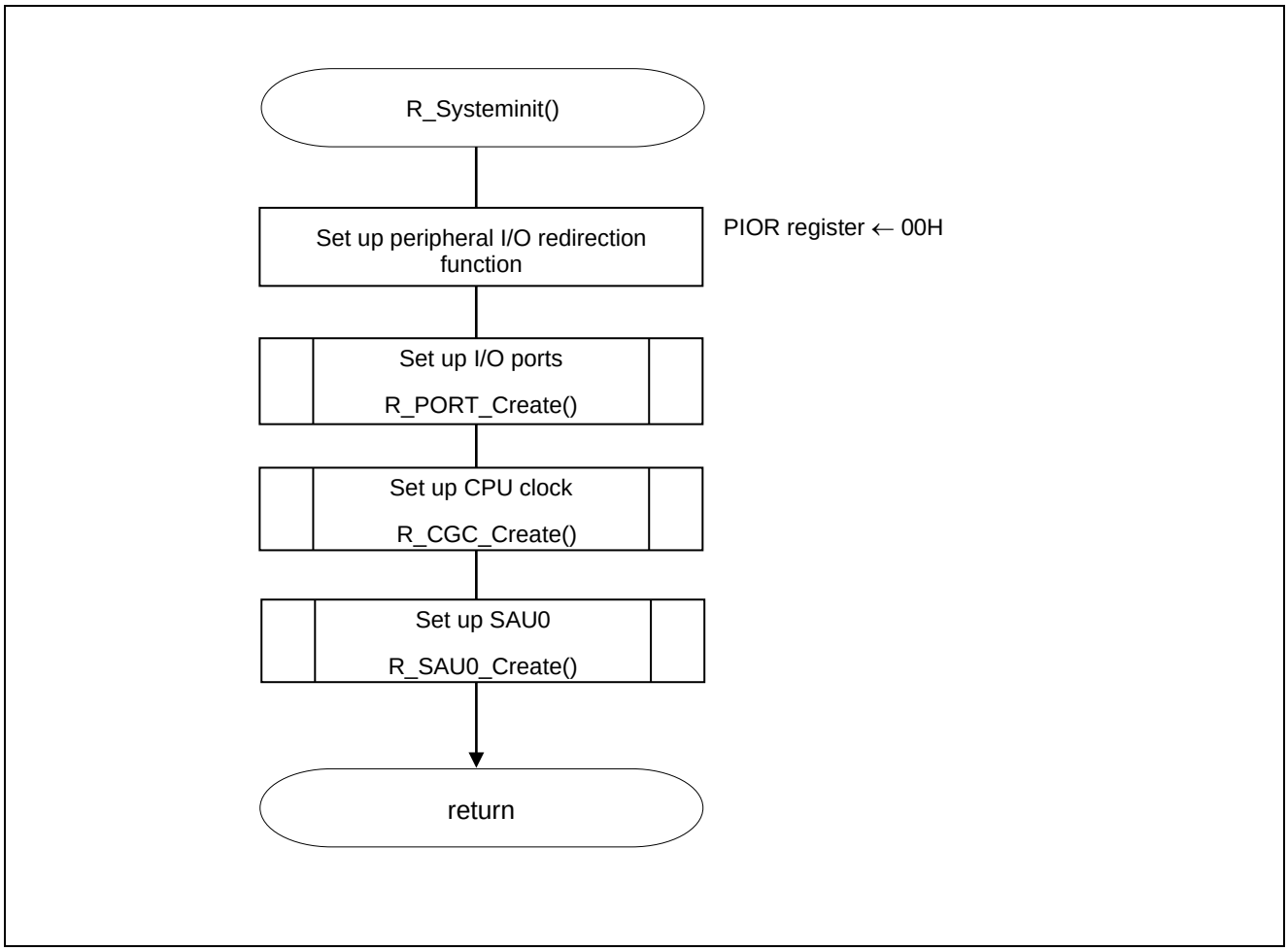


Figure 5.3 System Initialization Function

5.10.3 I/O Port Setup

Figure 5.4 shows the flowchart for I/O port setup.

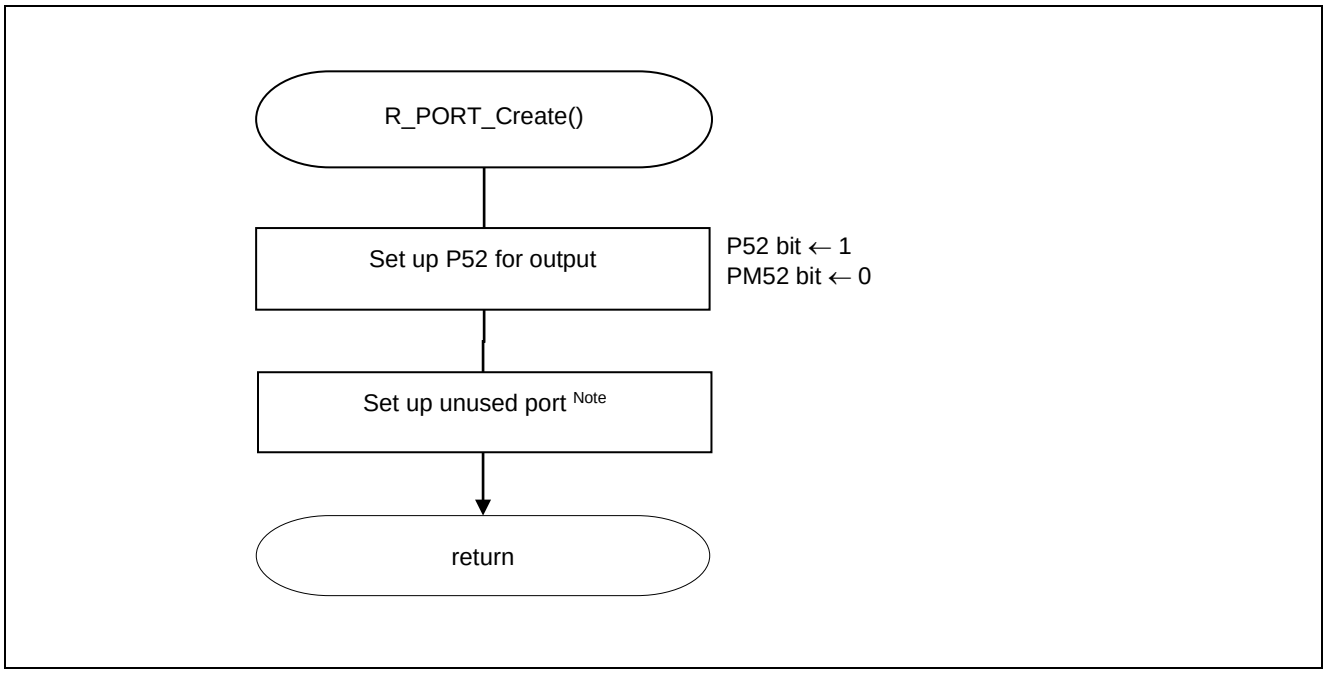


Figure 5.4 I/O Port Setup

Note: Refer to the section entitled "Flowcharts" in RL78/G13 Initialization (R01AN2575E) 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.

5.10.4 CPU Clock Setup

Figure 5.5 shows the flowchart for CPU clock setup.

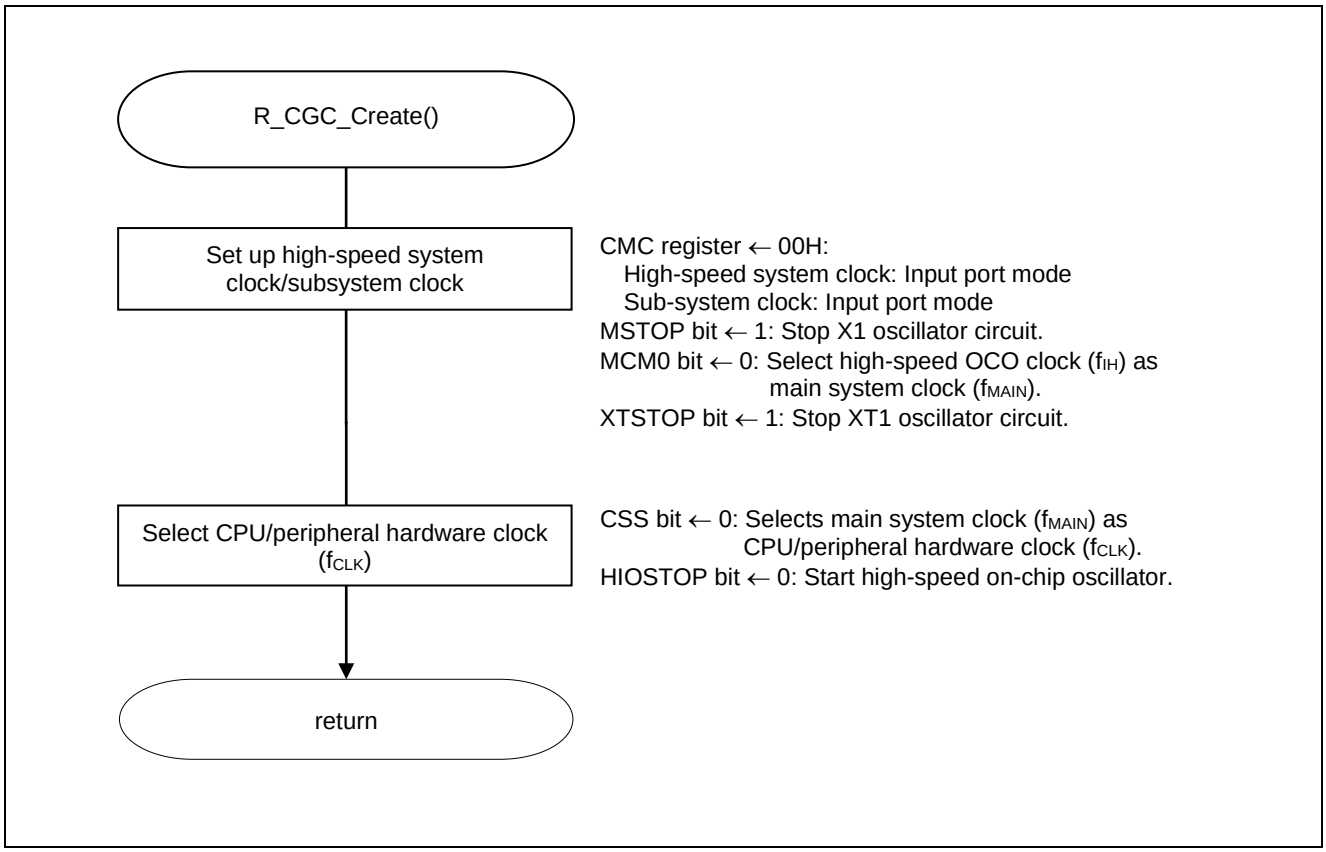


Figure 5.5 CPU Clock Setup

Caution: For details on the procedure for setting up the CPU clock (R_CGC_Create ()), refer to the section entitled "Flowcharts" in RL78/G13 Initialization (R01AN2575E) Application Note.

5.10.5 SAU0 Setup

Figure 5.6 shows the flowchart for SAU0 setup.

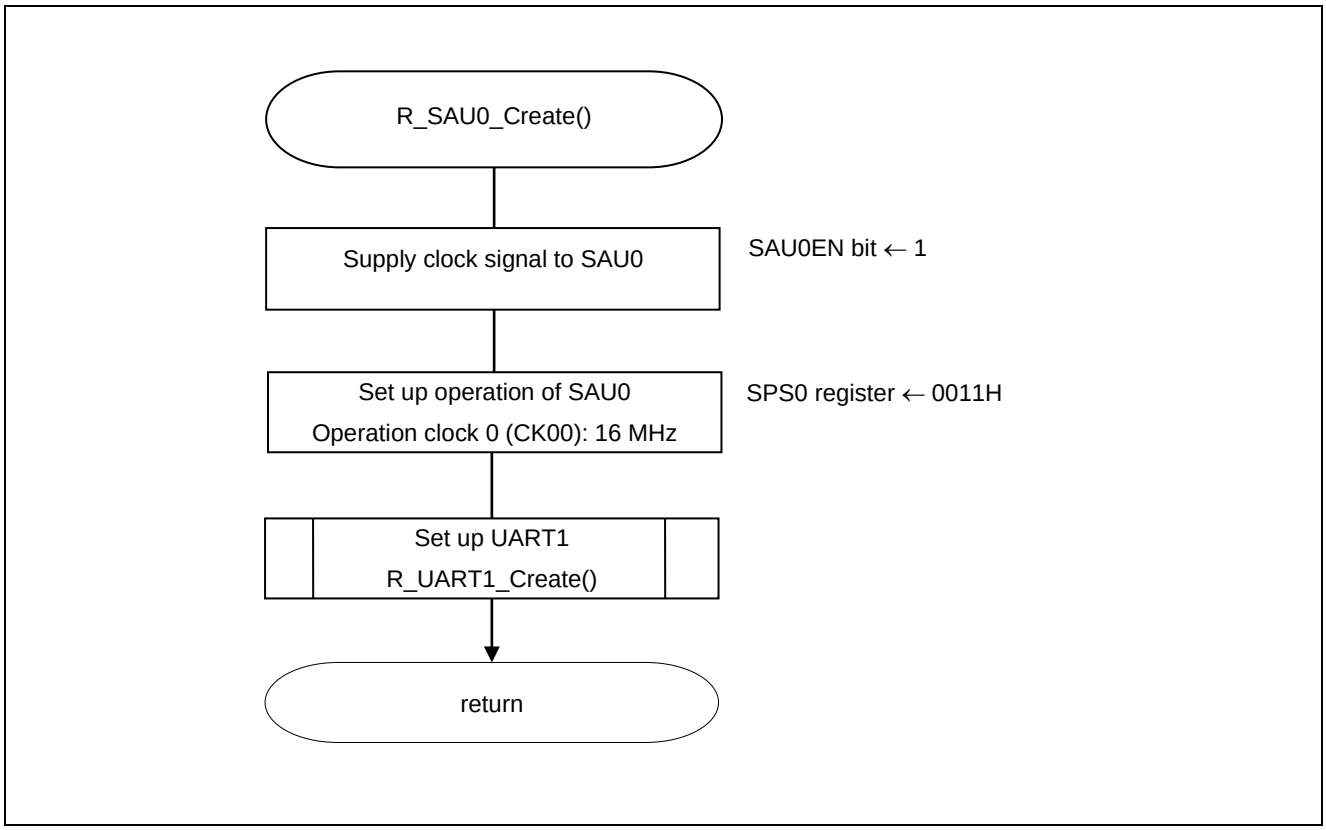


Figure 5.6 SAU0 Setup

5.10.6 UART1 Setup

Figure 5.7 shows the flowchart for UART1 setup (1/3). Figure 5.8 shows the flowchart for UART1 setup (2/3). Figure 5.9 shows the flowchart for UART1 setup (3/3).

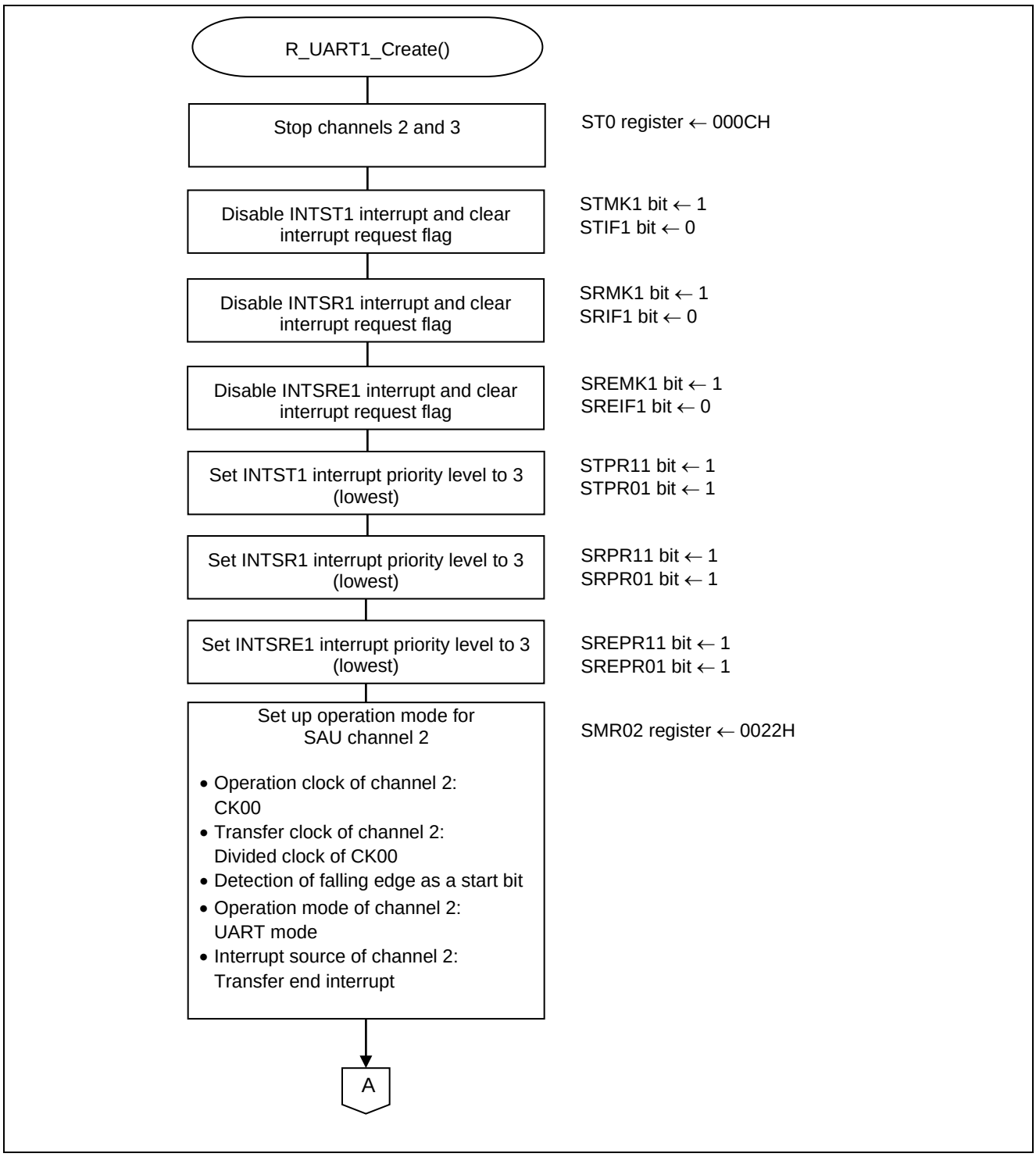


Figure 5.7 UART1 Setup (1/3)

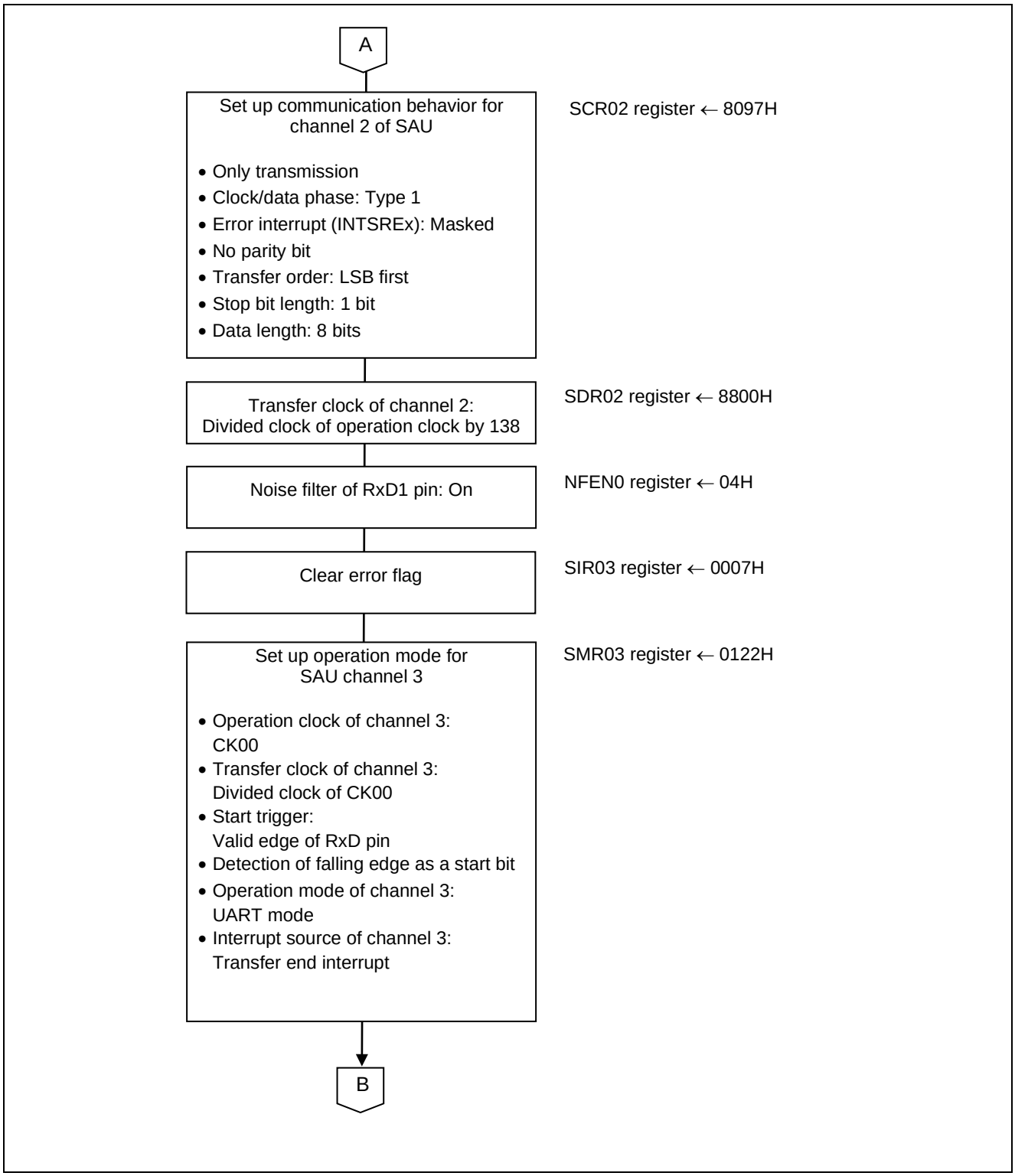


Figure 5.8 UART1 Setup (2/3)

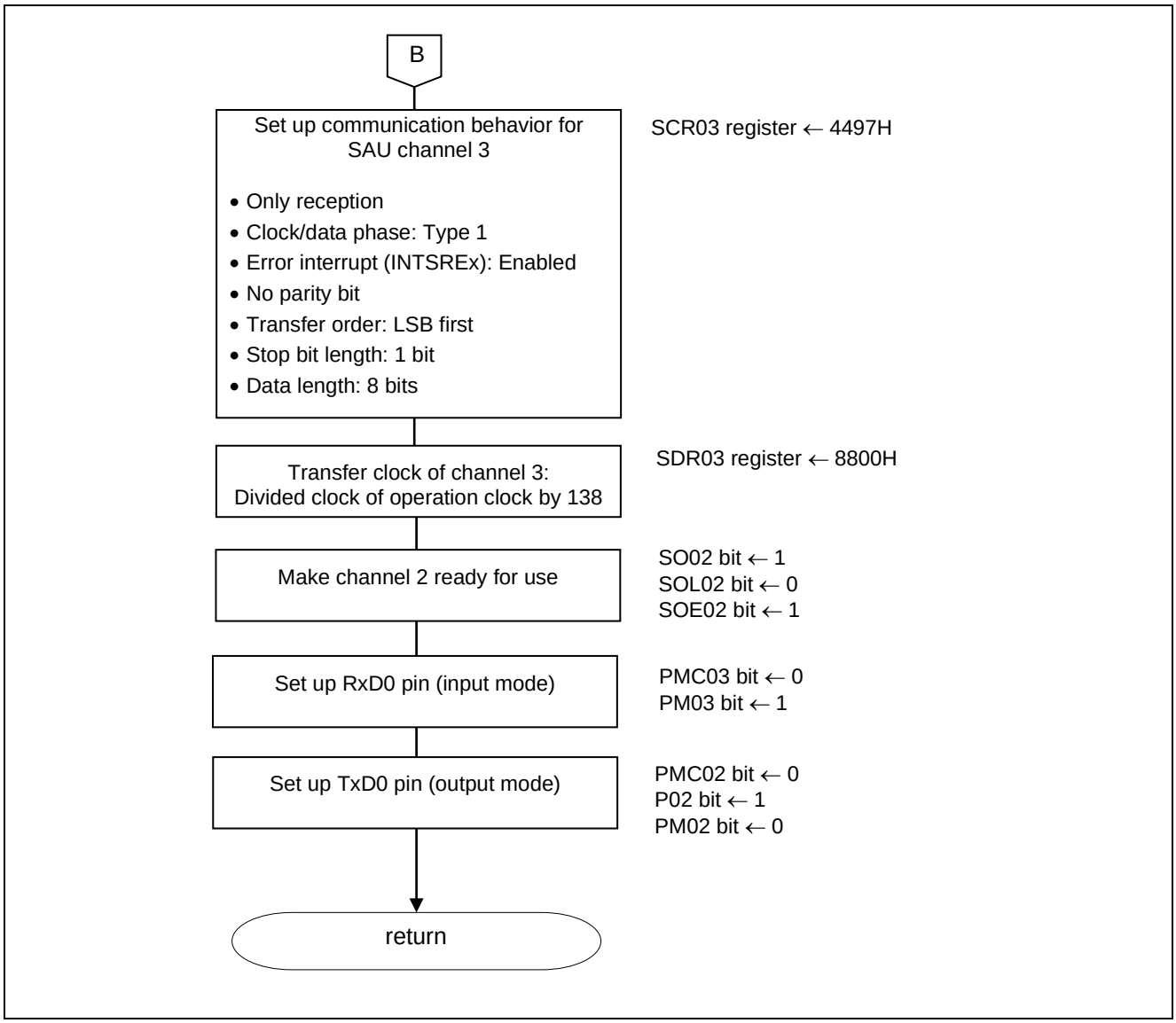


Figure 5.9 UART1 Setup (3/3)

5.10.7 Main Processing

Figure 5.10 shows the flowchart for main processing (1/2). Figure 5.11 shows the flowchart for main processing (2/2).

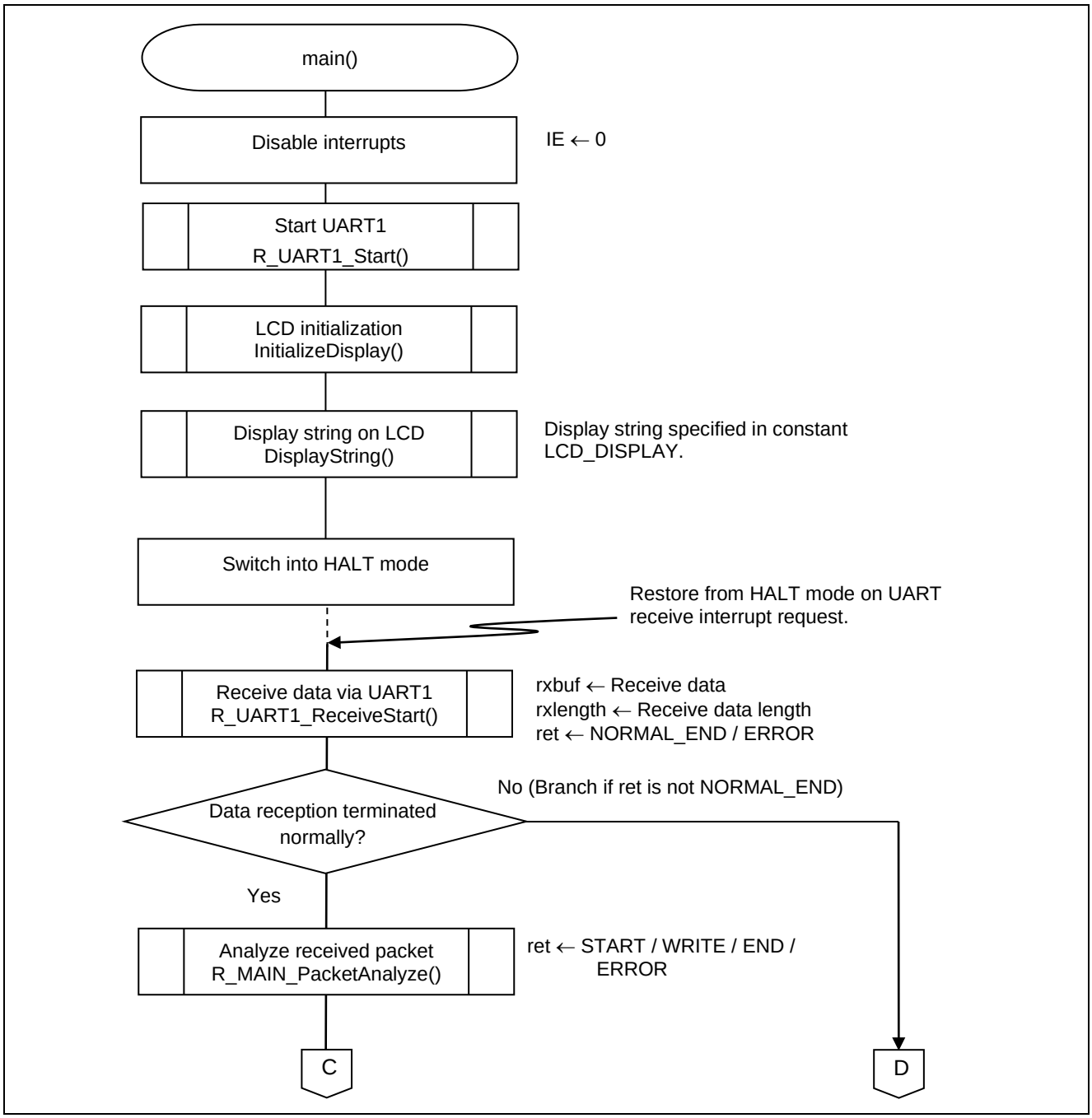


Figure 5.10 Main Processing (1/2)

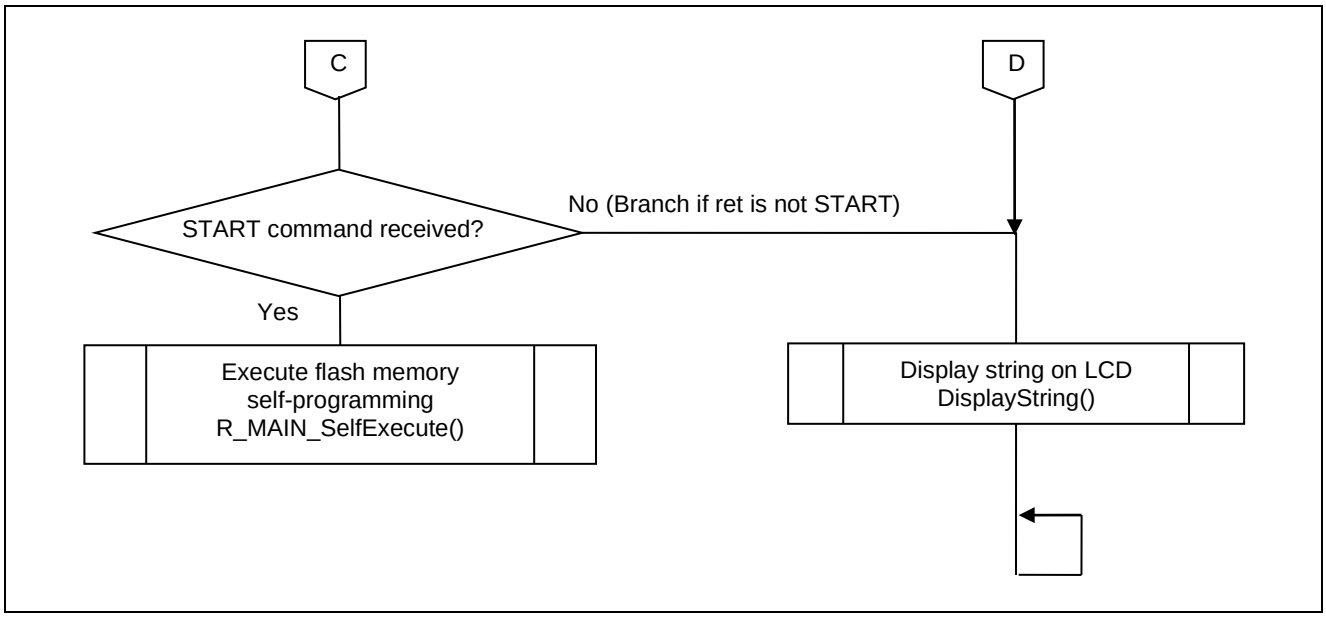


Figure 5.11 Main Processing (2/2)

5.10.8 Starting the UART1

Figure 5.12 shows the flowchart for starting the UART1.

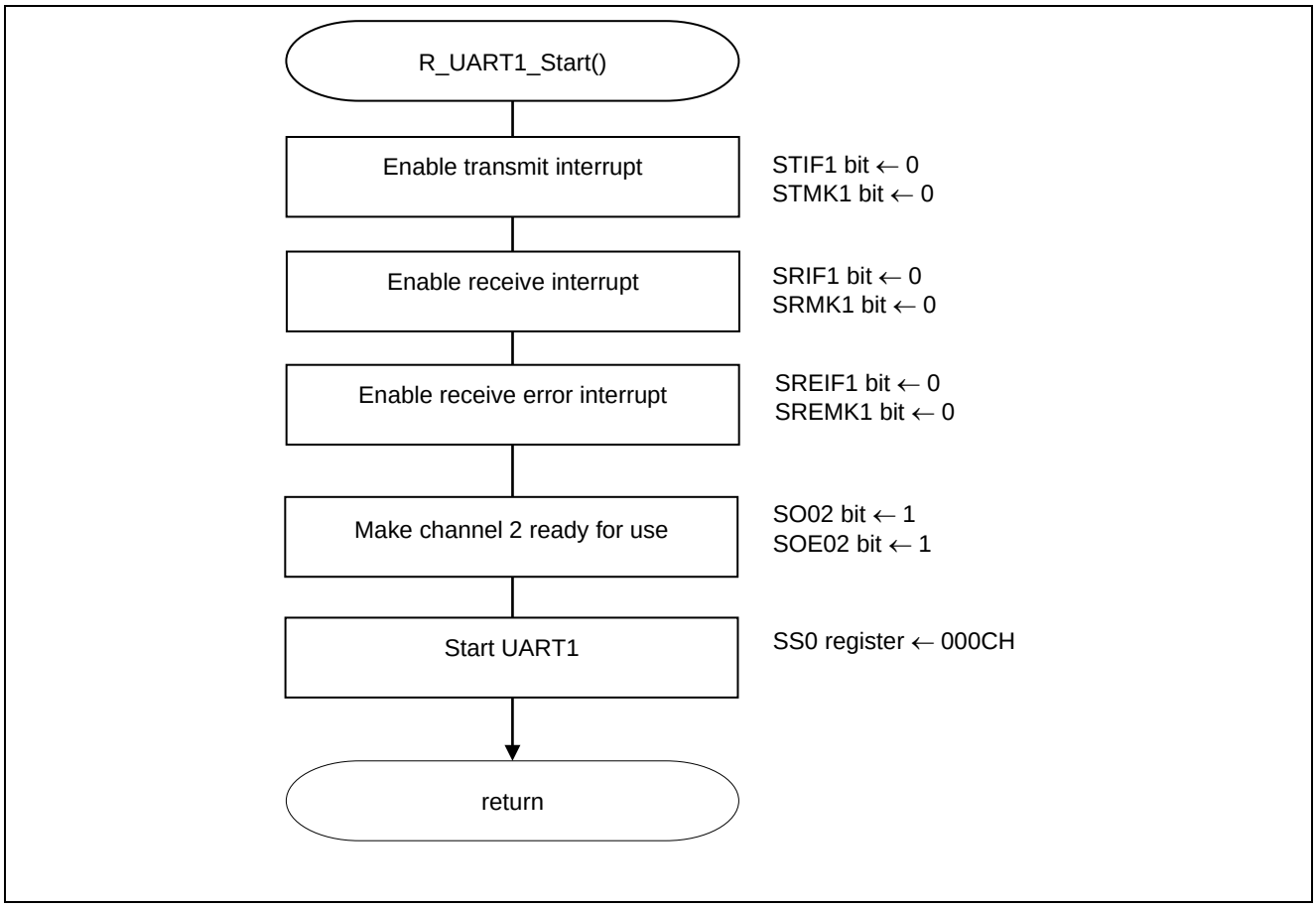


Figure 5.12 Starting the UART1

5.10.9 Data Reception via UART1

Figure 5.13 shows the flowchart for data reception via the UART1 (1/2). Figure 5.14 shows the flowchart for data reception via the UART1 (2/2).

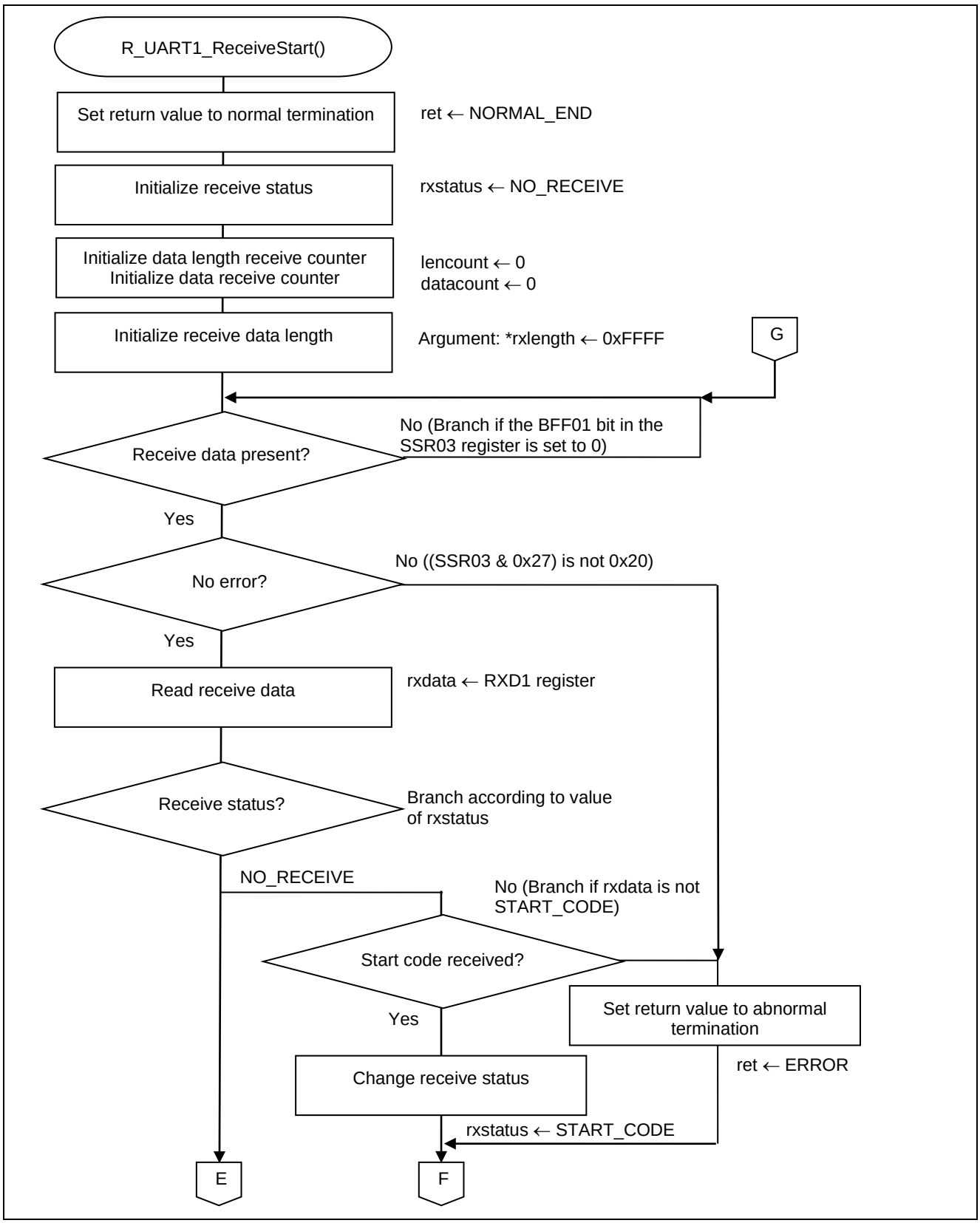


Figure 5.13 Data Reception via UART1 (1/2)

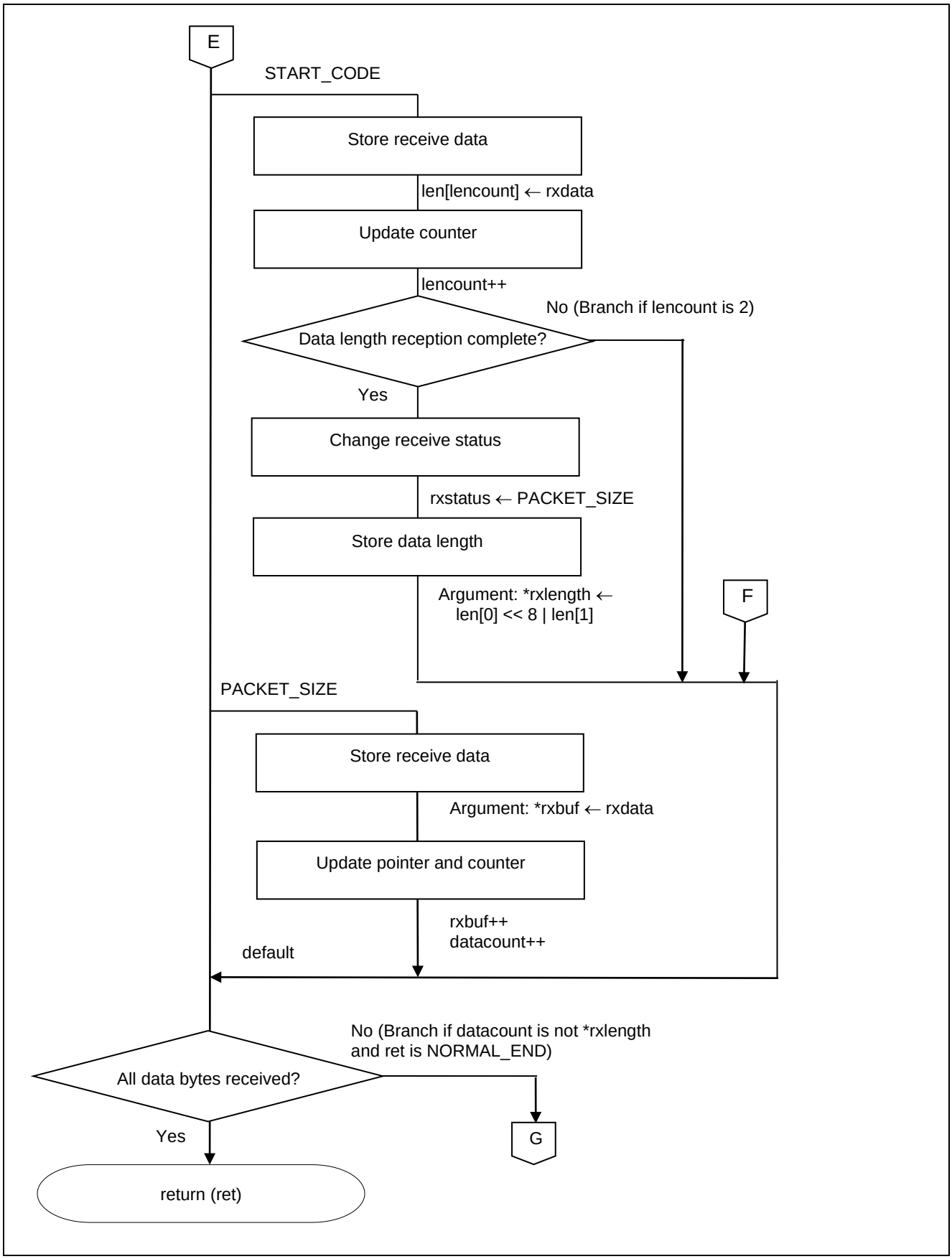


Figure 5.9 Data Reception via UART1 (2/2)

5.10.10 Receive Packet Analysis

Figure 5.15 shows the flowchart for receive packet analysis.

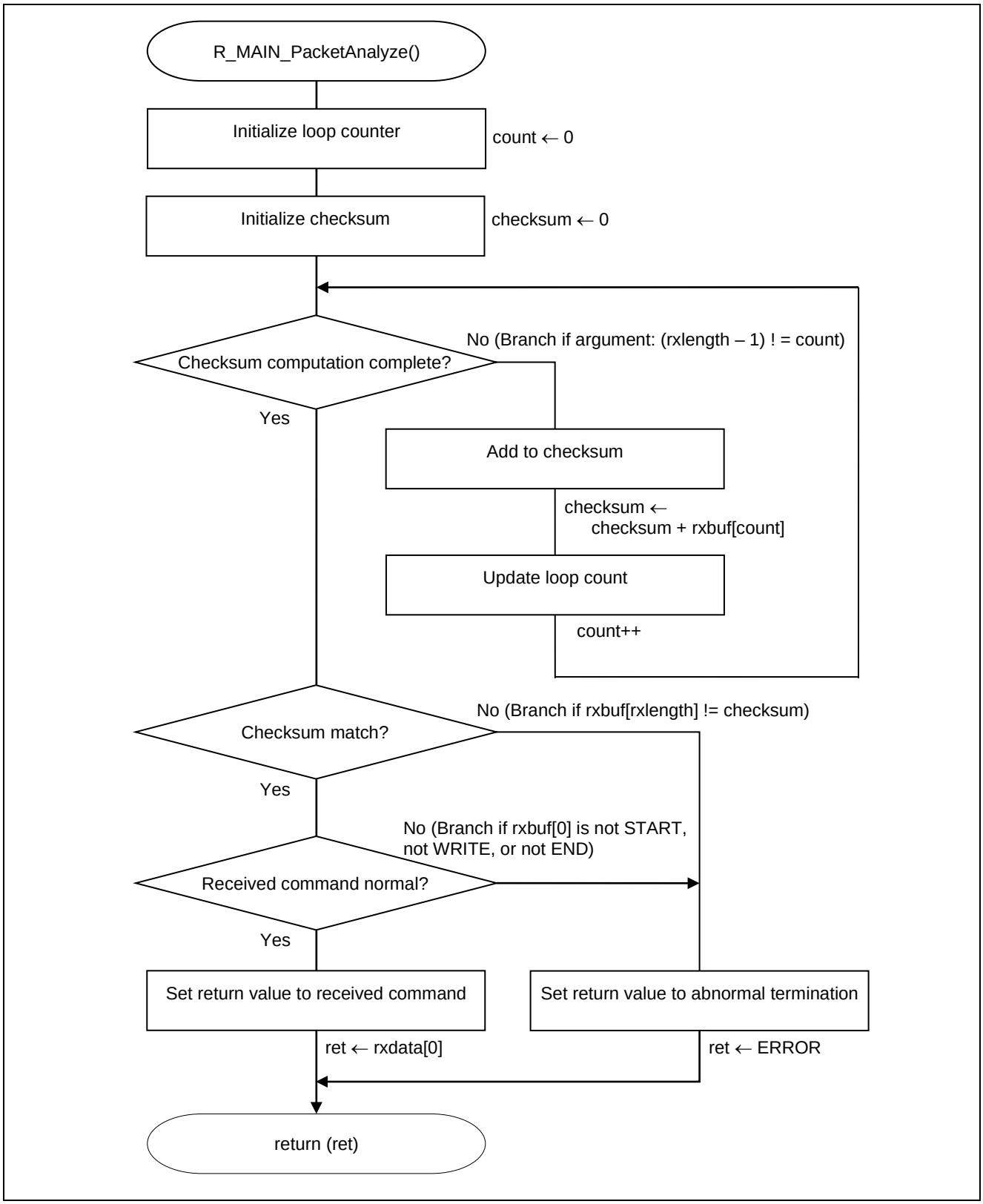


Figure 5.10 Receive Packet Analysis

5.10.11 Flash Memory Self-Programming Execution

Figure 5.16 shows the flowchart for flash memory self-programming execution.

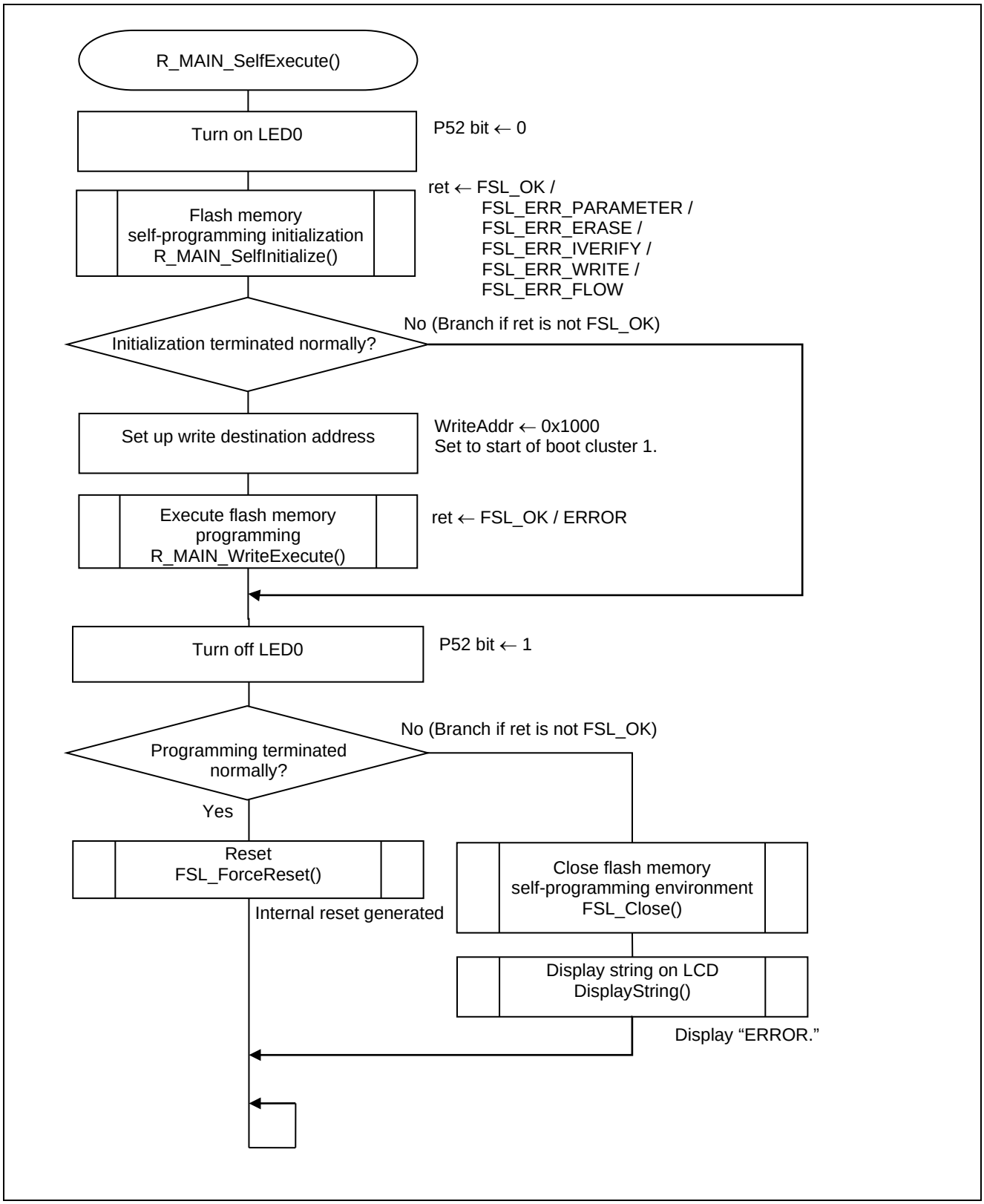


Figure 5.16 Flash Memory Self-Programming Execution

5.10.12 Flash Memory Self-Programming Initialization

Figure 5.17 shows the flowchart for flash memory self-programming initialization (1/2). Figure 5.18 shows the flowchart for flash memory self-programming initialization (2/2).

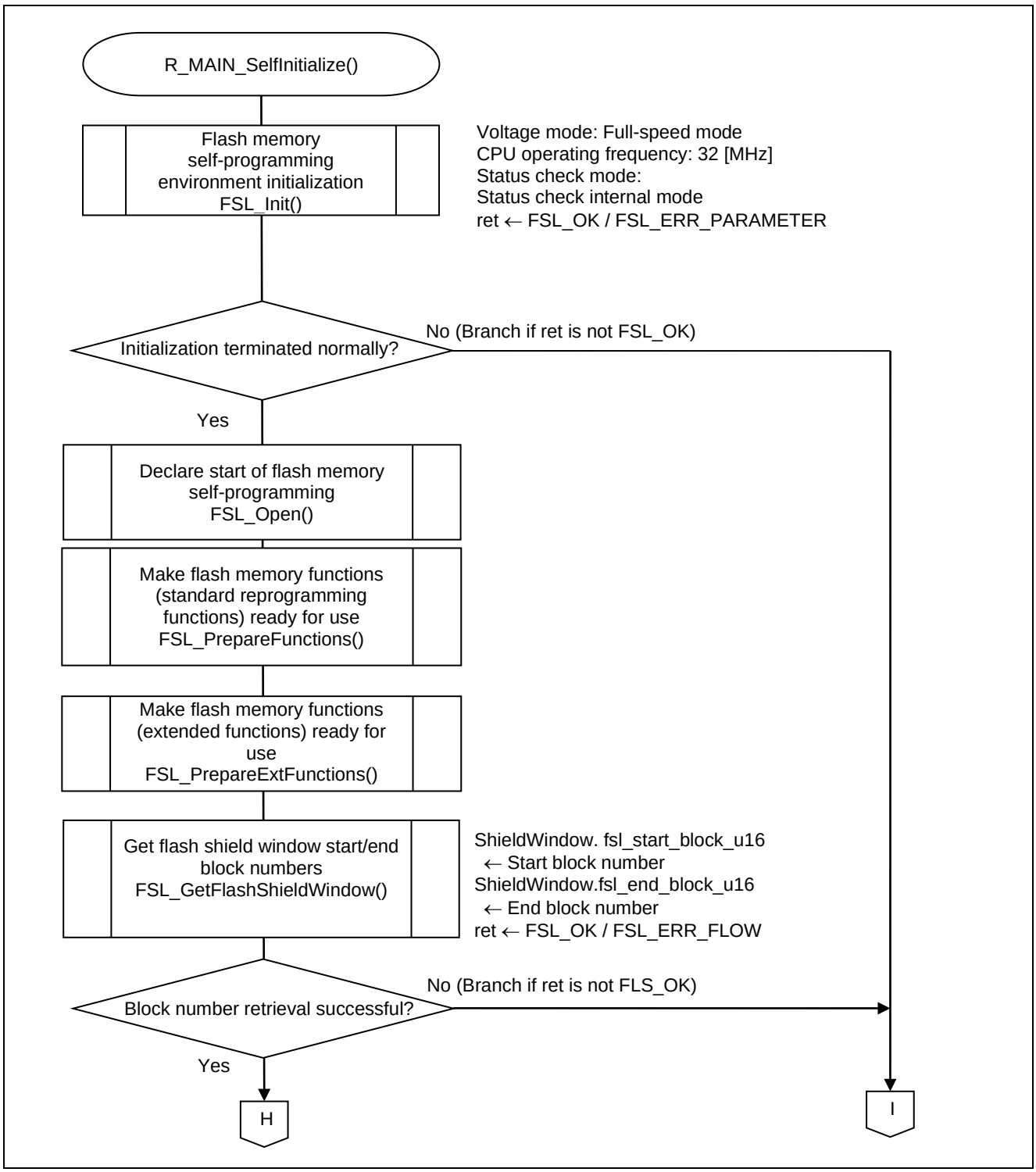


Figure 5.17 Flash Memory Self-Programming Initialization (1/2)

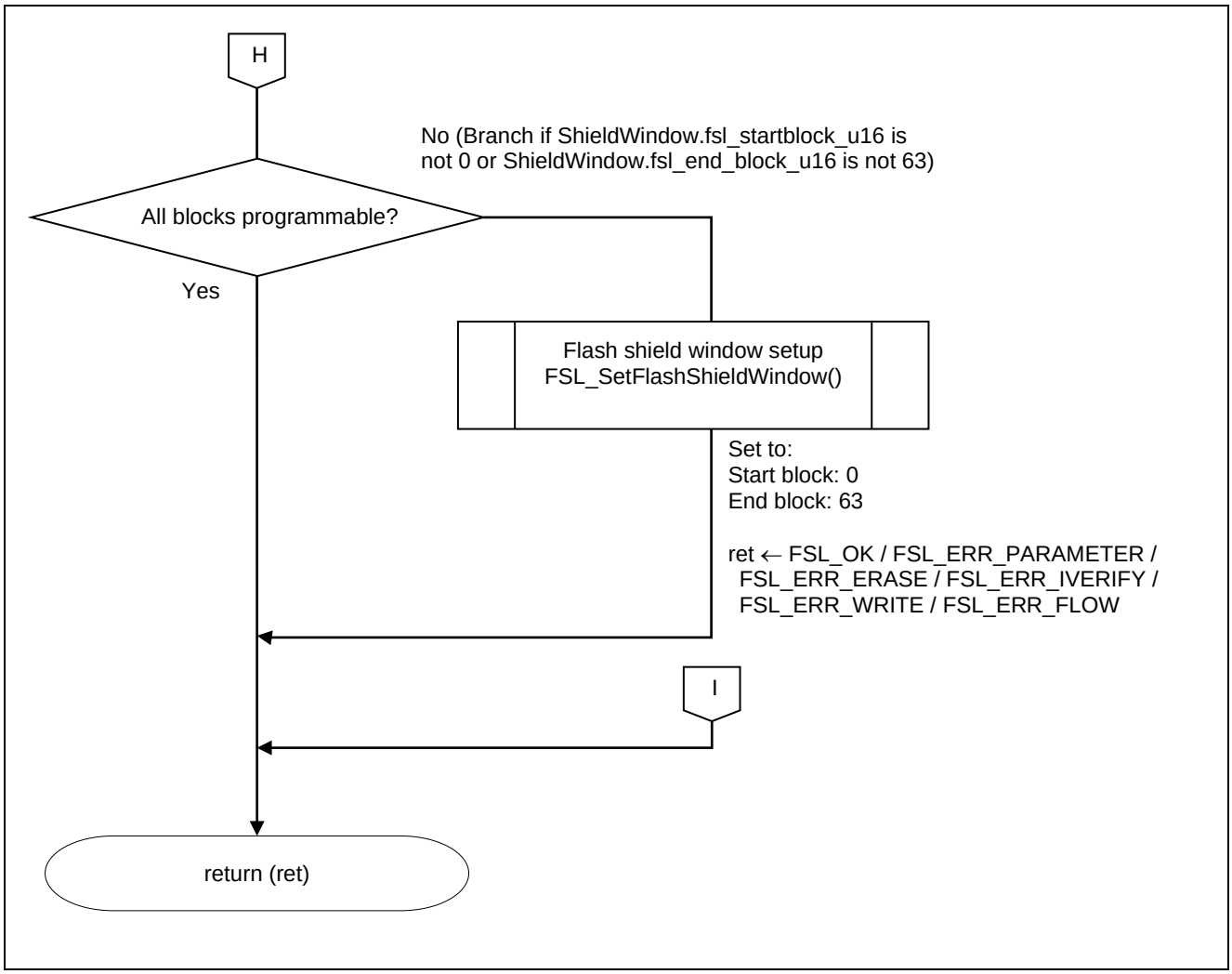


Figure 5.18 Flash Memory Self-Programming Initialization (2/2)

5.10.13 Flash Memory Reprogramming Execution

Figure 5.19 shows the flowchart for flash memory reprogramming execution (1/3). Figure 5.20 shows the flowchart for flash memory reprogramming execution (2/3). Figure 5.21 shows the flowchart for flash memory reprogramming execution (3/3).

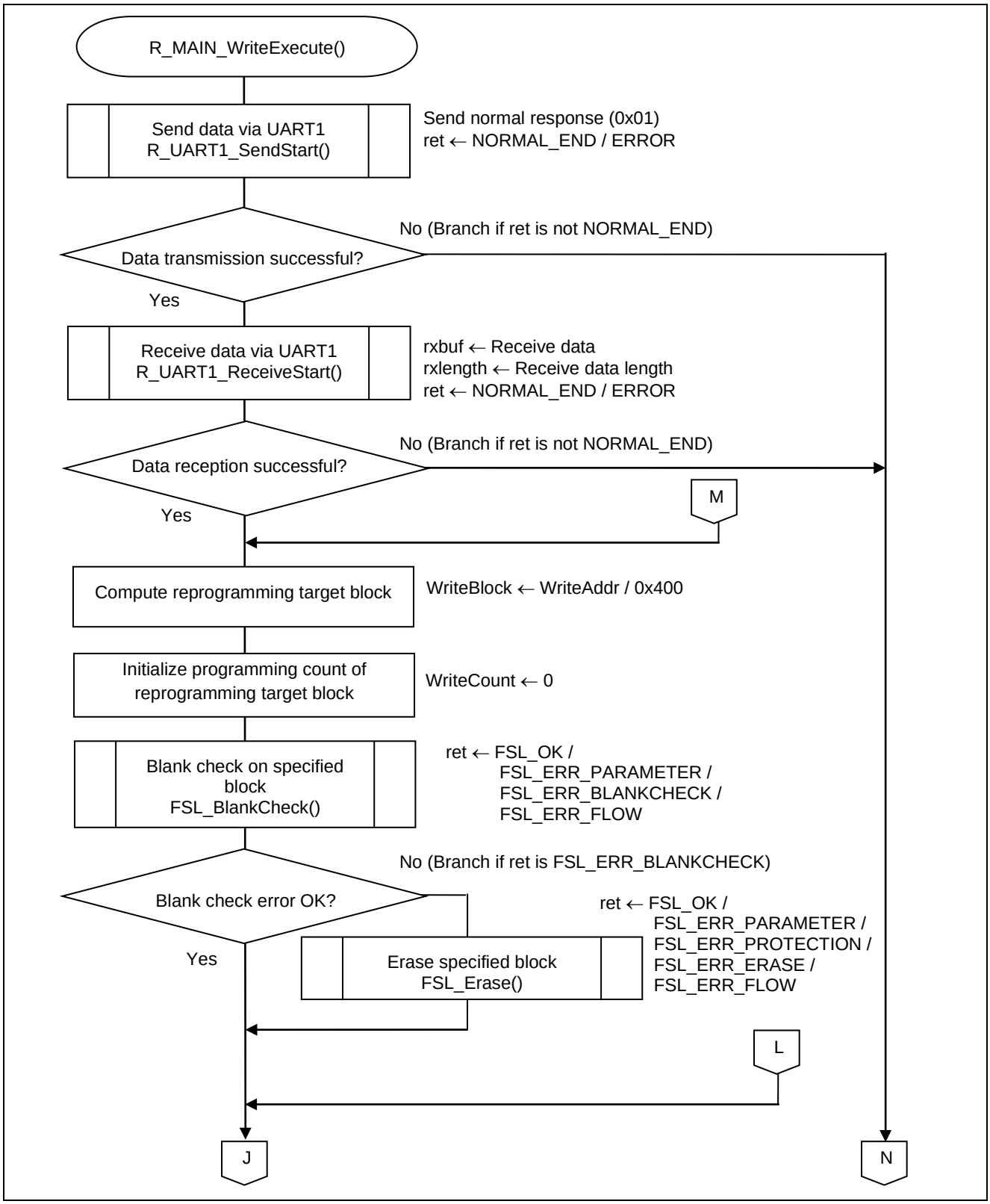


Figure 5.19 Flash Memory Reprogramming Execution (1/3)

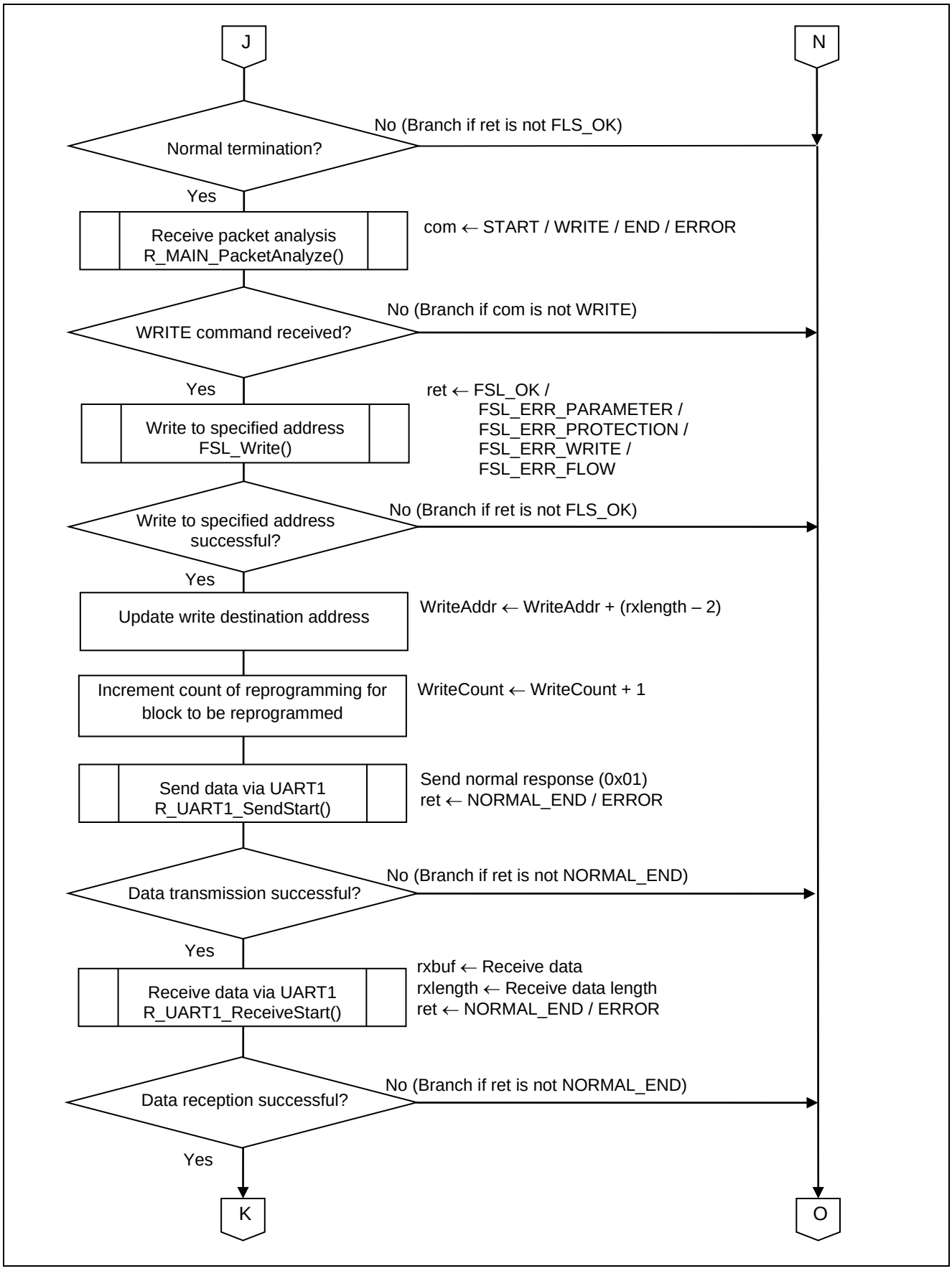


Figure 5.20 Flash Memory Reprogramming Execution (2/3)

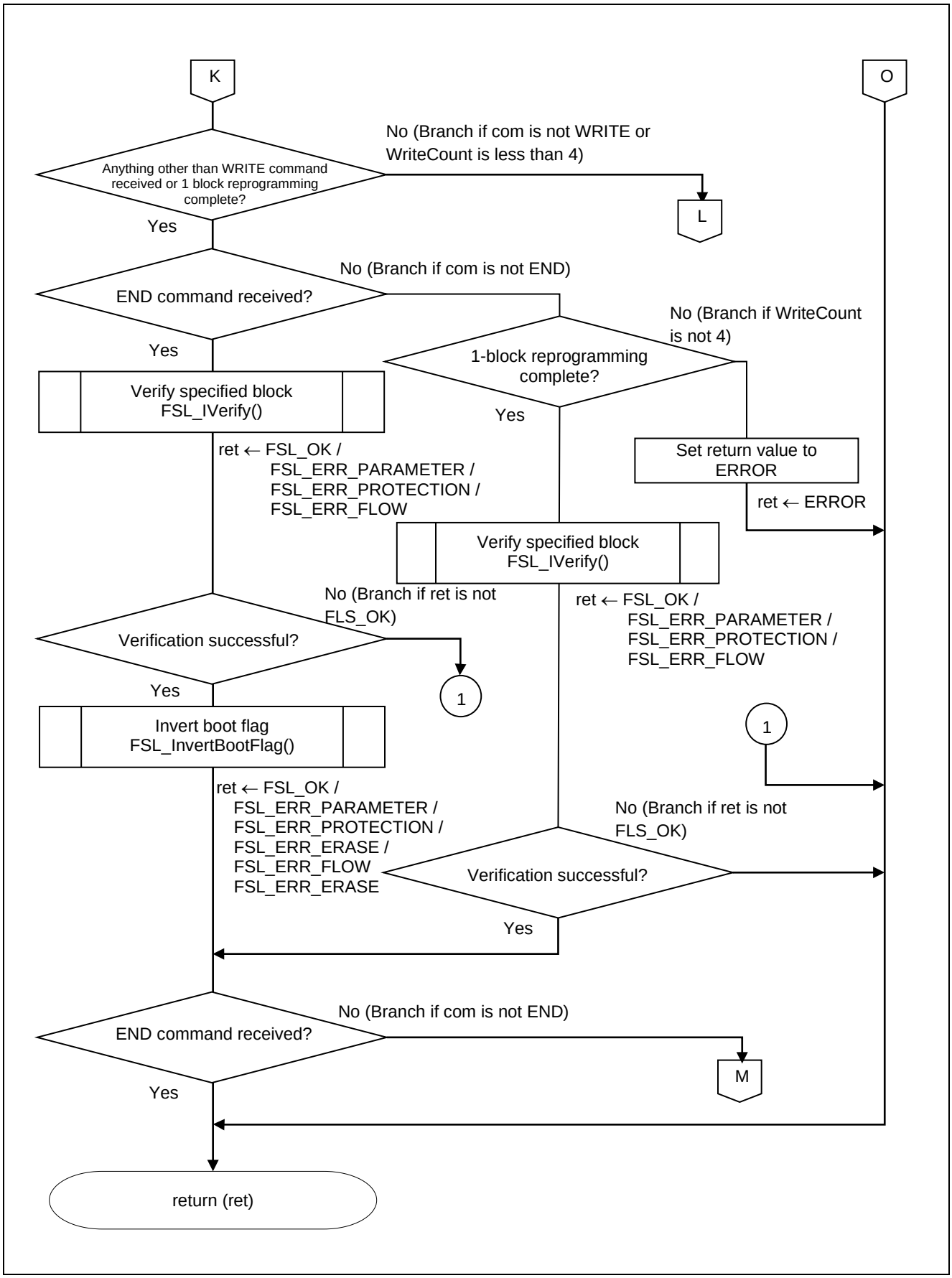


Figure 5.21 Flash Memory Reprogramming Execution (3/3)

5.10.14 Data Transmission via UART1

Figure 5.22 shows the flowchart for data transmission via the UART1.

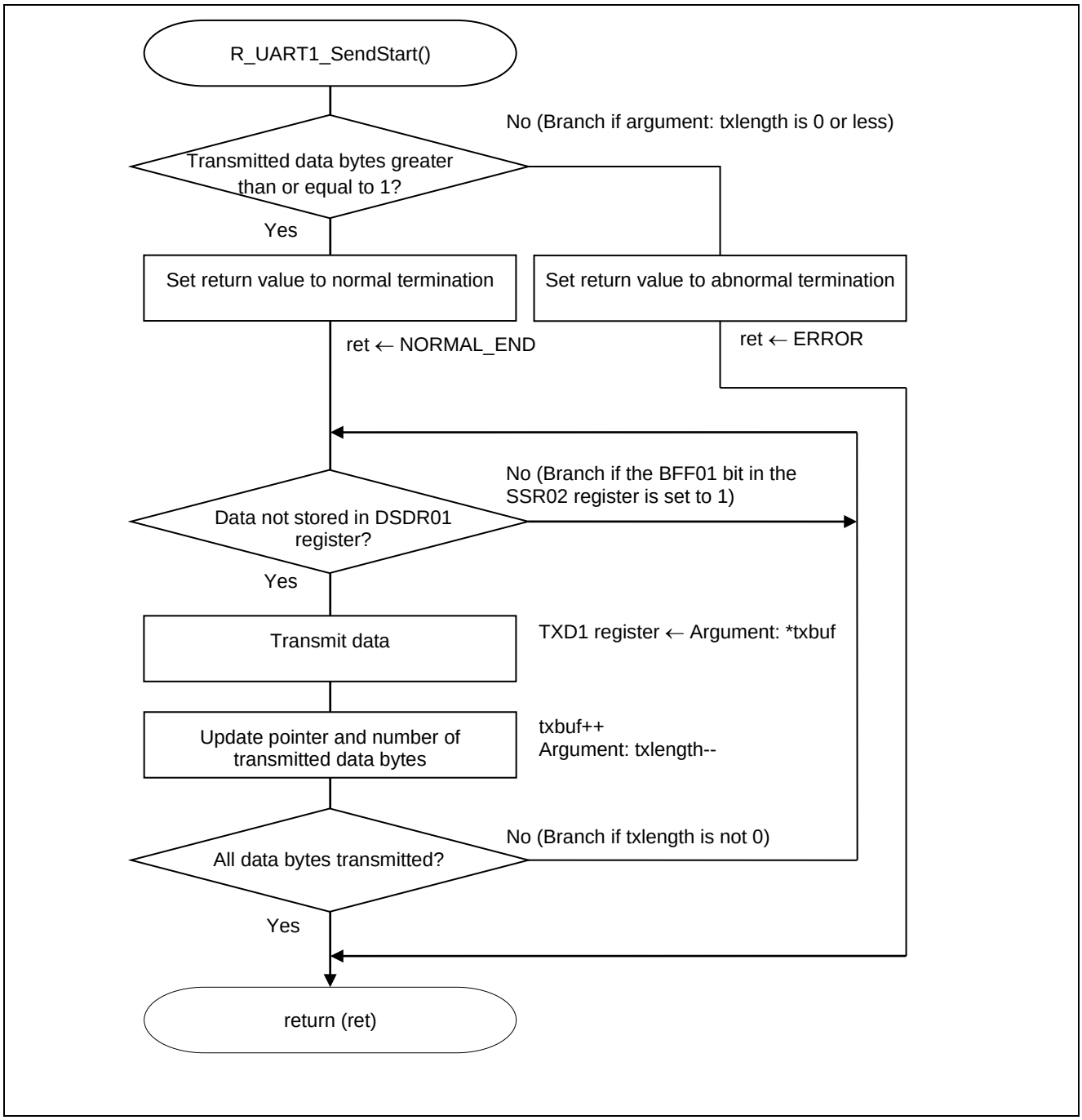


Figure 5.22 Data Transmission via UART1

5.11 Operation Check Procedure

Change the string defined in the constant LCD_DISPLAY that is defined in r_cg_userdefine.h for the sample program and rebuild the project. Set up to eight characters in the constant LCD_DISPLAY. Flash memory self-programming is carried out by sending the HEX file that is generated as the reprogramming data from the sending side. Refer to Section 5.1, Communication Specifications, for the specifications for the communication between the sending side and this sample program.

For example, the operation of the sample program will look like as shown below when the value of the constant LCD_DISPLAY is changed to "Ver 2.0."

(1) "Ver 1.0" is displayed on the LCD.

The constant LCD_DISPLAY is defined as "Ver 1.0" by this sample program.

(2) Send a START command from the sending side to initiate communication.

After the START command is sent, communication between the sending side and this sample program proceeds as specified in Section 5.1, Communication Specifications.

(3) When the sample program receives a WRITE command and reprogramming data and starts flash memory self-programming, LED0 on the RSK board turns on.

(4) LED0 turns off when the sample program receives an END command.

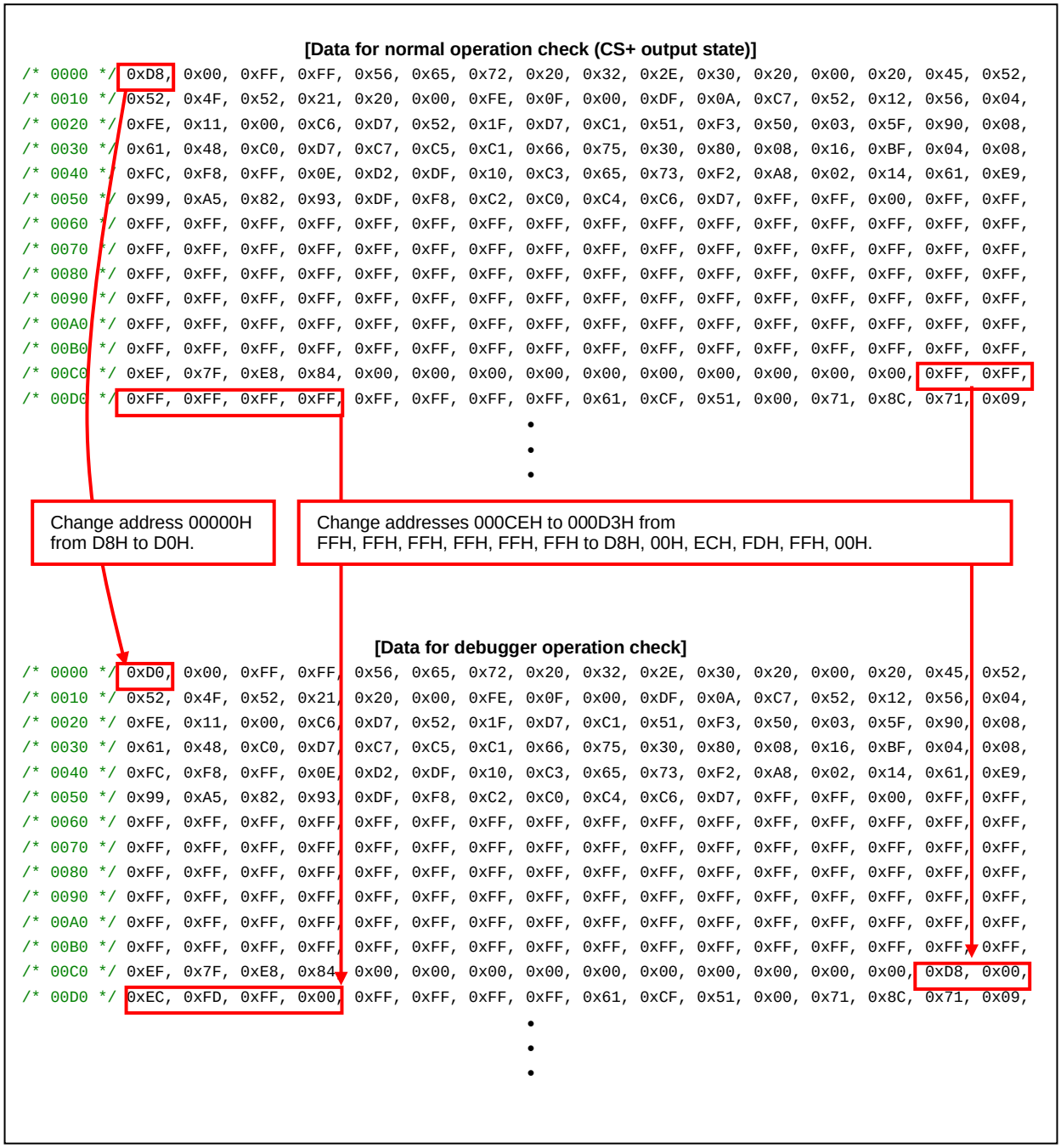
(5) A reset occurs and "Ver 2.0" is displayed on the LCD.

5.11.1 Making Checks with a Debugger

When flash memory self-programming is executed with a debugger (E1 emulator) connected, it becomes unable to check the execution of the program correctly with the debugger after the reprogramming. To check the program execution with the debugger after reprogramming, it is necessary to change the HEX file that is to be used as reprogramming data from the state established immediately when it is generated by CS+.

More specifically, it is necessary to rewrite the reset vector (address 0x00000) to the address where the monitor program is placed and to add changes to a part of the monitor program (addresses 0x000CE to 0x0000D3) as shown below.

Address	CS+ Output State	Change To
0x00000 (Reset vector)	0xD8	0xD0
0x000CE	0xFF	0xD8
0x000CF	0xFF	0x00
0x000D0	0xFF	0xEC
0x000D1	0xFF	0xFD
0x000D2	0xFF	0xFF
0x000D3	0xFF	0x00



This sample program generates a reset by inverting the state of the boot flag and carries out boot swapping after rewriting boot cluster 1. The FSL_ForceReset function of the flash memory self-programming library is used to generate the reset. When this function is executed with a debugger (E1 emulator) connected, a break will occur and processing stop. After the break occurs, it is necessary to manually effect a reset and execute the program again.

6. Sample Code

The sample code is available on the Renesas Electronics Website.

7. Documents for Reference

RL78/G13 User's Manual: Hardware (R01UH0146E)

RL78 Family User's Manual: Software (R01US0015E)

RL78 Family 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/G13 Self-Programming (UART)

Rev.	Date	Description	
		Page	Summary
1.00	May 28, 2015	—	First edition issued
1.01	Feb. 27, 2024	11	Updated operation check conditions

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

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

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

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

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

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

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

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

Trademarks

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

Contact information

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