RL78/G13

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

R01AN2849EJ0100
Rev. 1.00
Jan. 29, 2016

Introduction

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

The 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 CSI.
Port I/O	Displays text on the LCD. Turns on and off the LED. Outputs the BUSY signal. Note

Note: The BUSY signal indicates whether communication is enabled or disabled. When it is set to 0, it indicates communication is enabled. When it is set to 1, it indicates communication is disabled.

1.1 Outline of the Flash Memory Self-Programming Library

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

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

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

1.2 Code Flash Memory

The configuration of the RL78/G13 (R5F100LE) code flash memory is shown below.

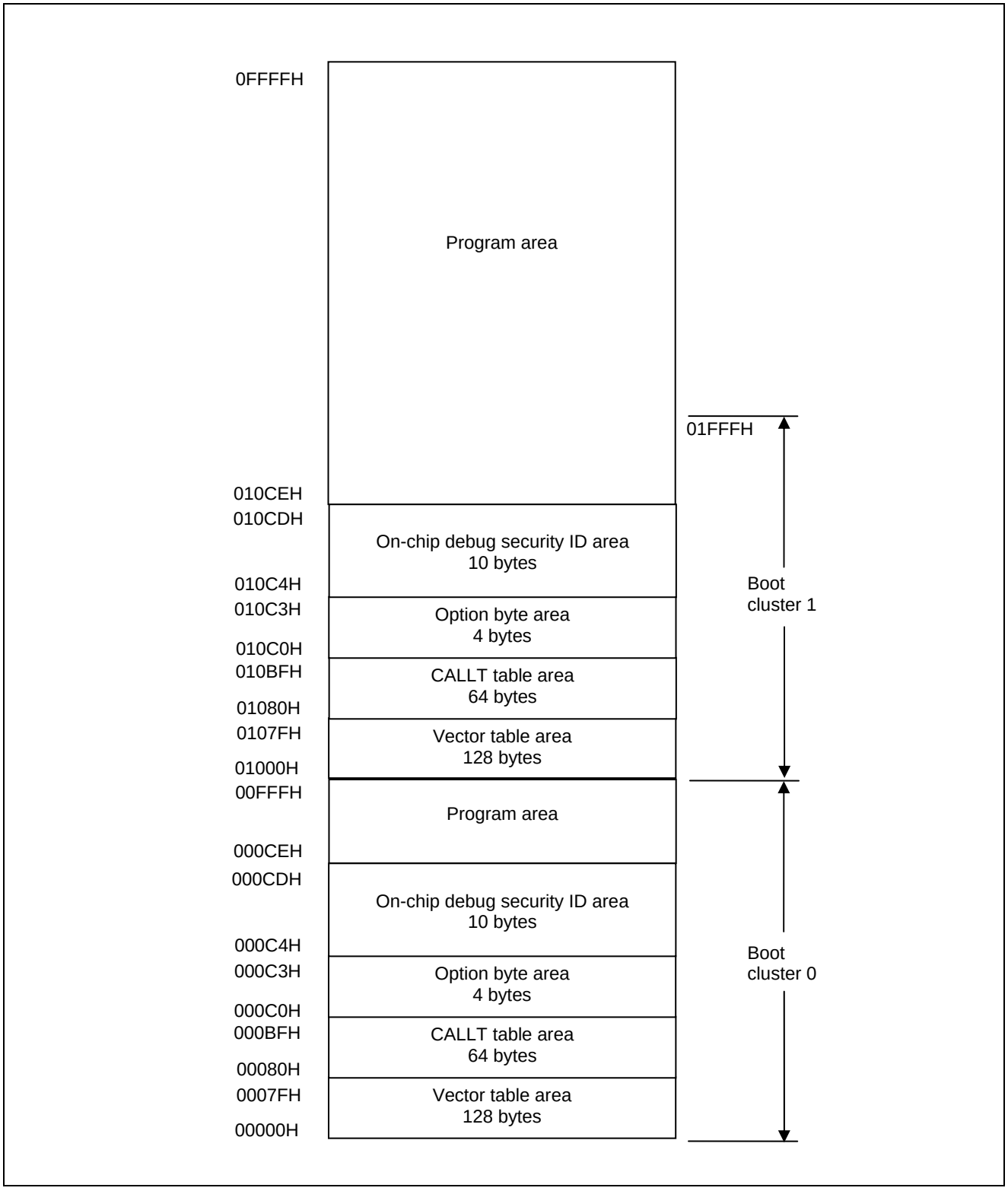


Figure 1.1 Code Flash Memory Configuration

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 memory self-programming library

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

1.3 Flash Memory Self-Programming

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

The flash memory 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 memory self-programming library and the reprogramming program in RAM when erasing or reprogramming the code flash memory or making settings for the security flags. If there is no need to run the user program while the sequencer control is in progress, it is possible to keep the flash memory self-programming library and reprogramming program on ROM (code flash memory) for execution.

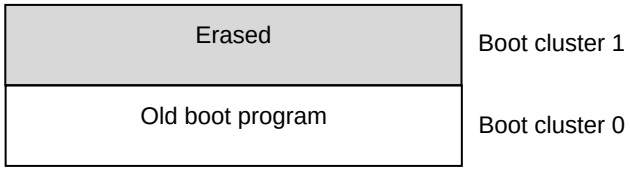
1.3.1 Boot Swap Function

When reprogramming of the area where vector table data, the basic functions of the program, and flash memory self-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.

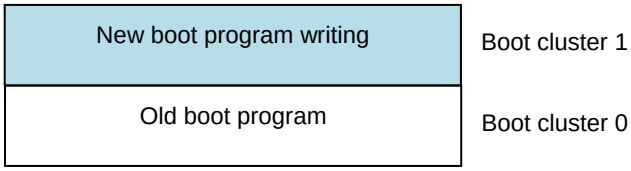
The boot swap function swaps between boot cluster 0 which is the boot program area and boot cluster 1 which is the target of boot swapping. A new program is written into boot cluster 1 before reprogramming is attempted. This boot cluster 1 is swapped with boot cluster 0 and boot cluster 1 is designated 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).

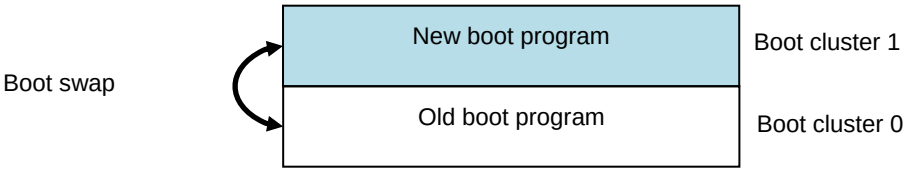


(2) Writing the new boot program into boot cluster 1
Call the FSL_Write function to write the new boot program into boot cluster 1 and call the FSL_IVerify function to verify boot cluster 1.
The steps that have been performed up to here ensure that the programs will run normally even when the programming 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

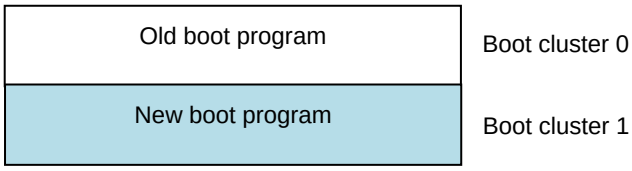


Figure 1.2 Outline of Boot Swapping

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.

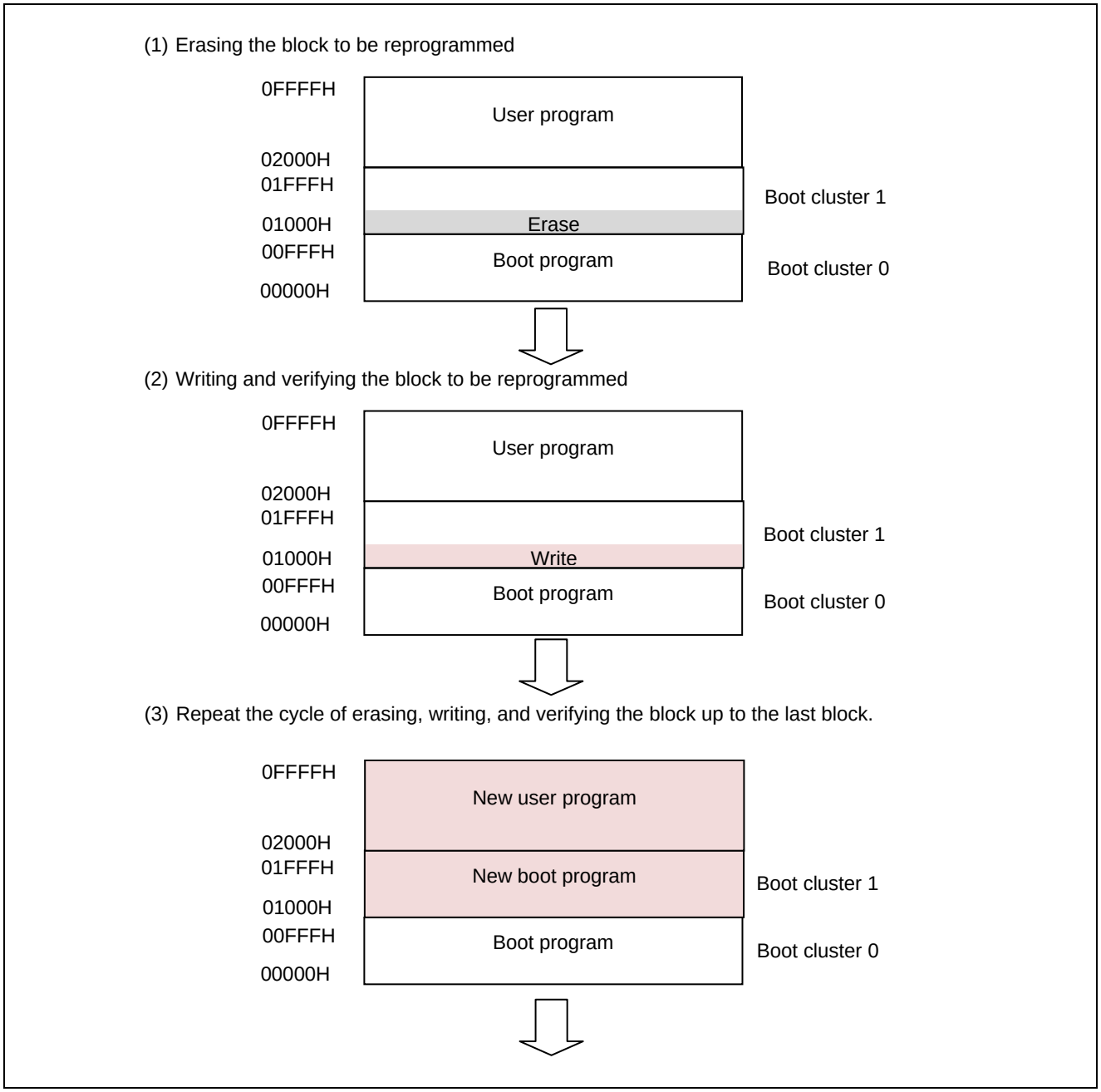


Figure 1.3 Outline of Flash Memory Reprogramming (1/2)

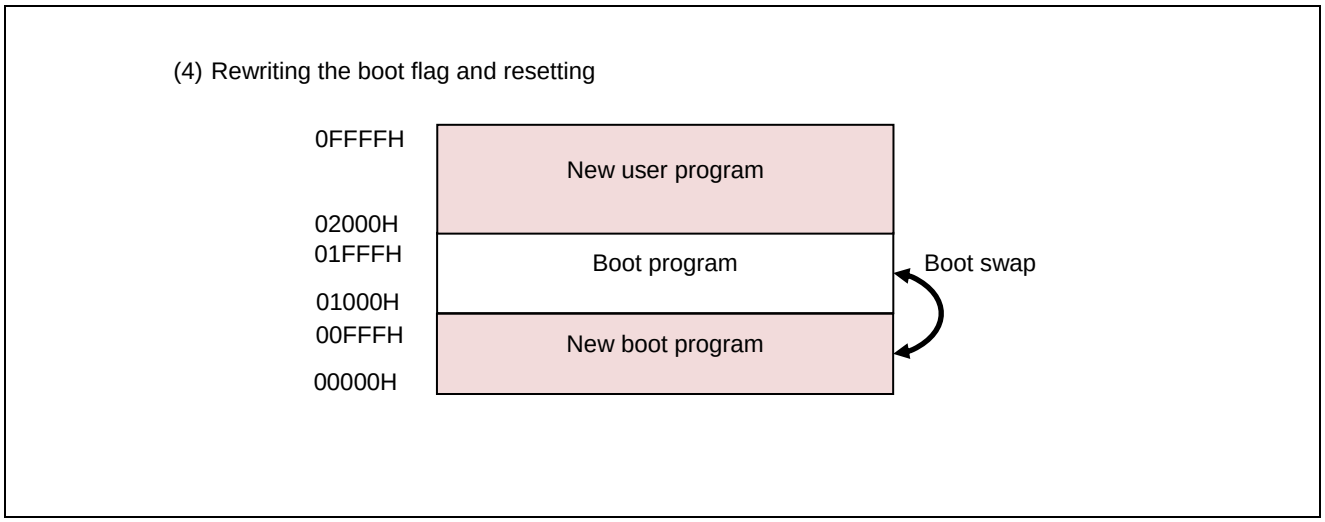


Figure 1.4 Outline of Flash Memory Reprogramming (2/2)

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.

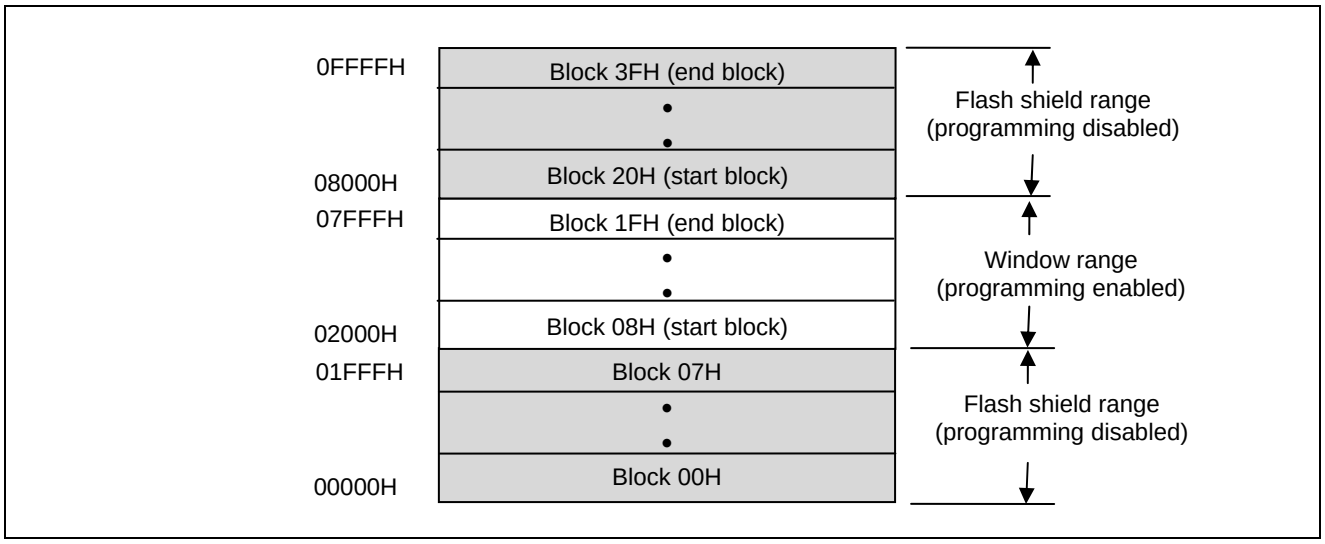


Figure 1.5 Outline of the Flash Shield Window

1.4 How to Get the Flash Memory Self-Programming Library

Before compiling the sample program, please download the latest flash self-programming library and copy the library files to the following folder below “r01an2849_flash”.

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

librl78 folder : fsl.lib

The flash memory 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+ V3.01.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.01.00 from Renesas Electronics Corp.
Integrated development environment (e^2 studio)	e^2 studio V4.0.0.26 from Renesas Electronics Corp.
C compiler (e^2 studio)	CC-RL V1.01.00 from Renesas Electronics Corp.
Board to be used	Renesas Starter Kit for RL78/G13 (R0K50100LS000BE)
Flash memory self-programming library (Type, Ver)	FSLRL78 Type01, Ver 2.21 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 for 3-Wire Serial I/O (Master Transmission/Reception) (R01AN2547E) Application Note
- RL78/G13 Serial Array Unit for 3-Wire Serial I/O (Slave Transmission/Reception) (R01AN2711E) 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.

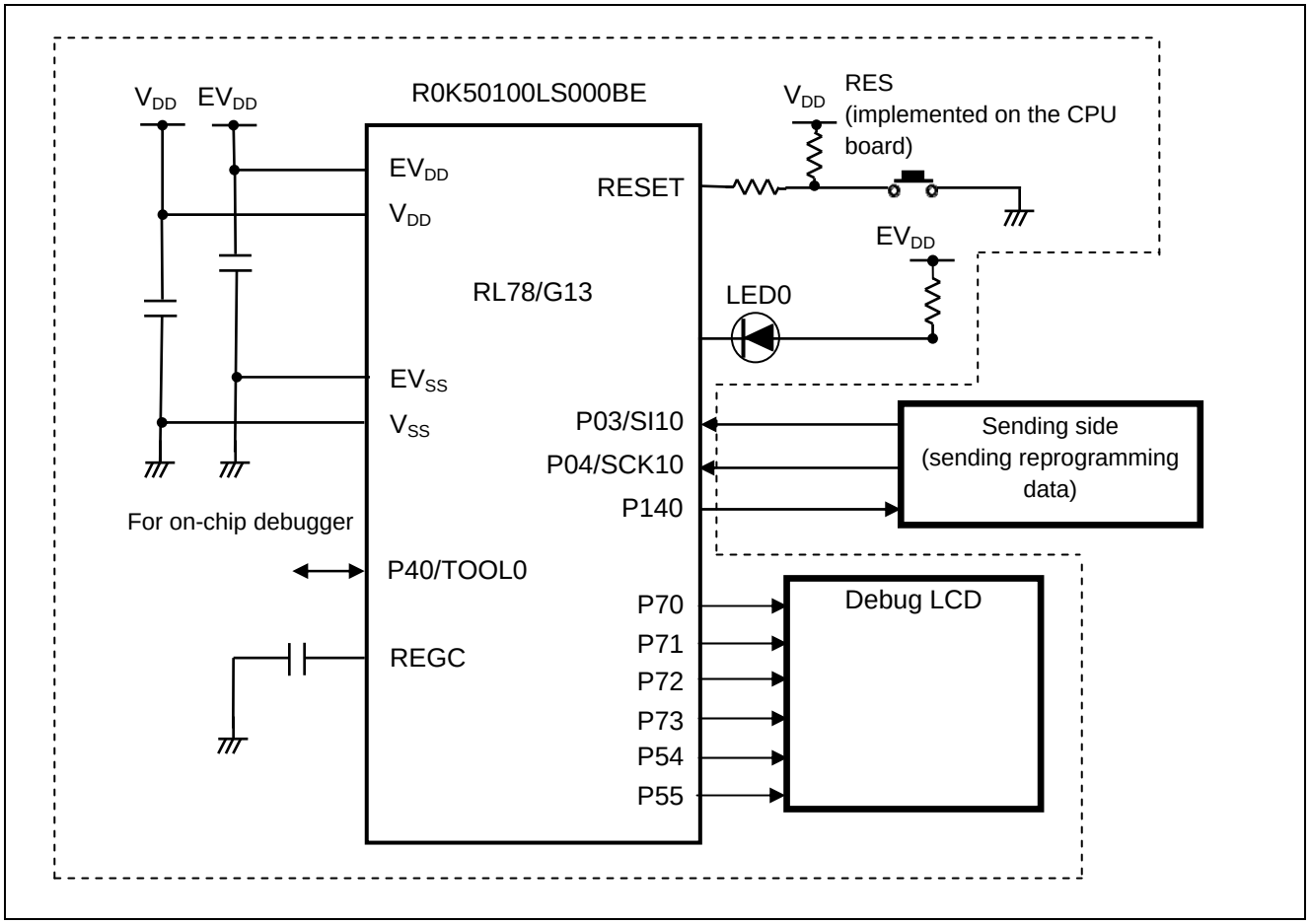


Figure 4.1 Hardware Configuration

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
P03/ANI16/SI10/RxD1/SDA10	Input	CSI serial data receive pin
P04/SCK10/SCL10	Input	CSI serial clock input 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
P140	Output	BUSY signal Note

Note: The BUSY signal indicates whether communication is enabled or disabled. When it is set to 0, it indicates communication is enabled. When it is set to 1, it indicates communication is disabled.

5. Description of the Software

5.1 Communication Specifications

The sample program covered in this application note receives reprogramming data via the CSI 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, sets the BUSY signal to the high level. 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 CSI communication settings and the specifications for the commands.

Table 5.1 CSI Communication Settings

Transfer mode	Single transfer mode
Data bit length [bit]	8
Data transfer direction	MSB first
Data transmission/reception timing	Type 1
Transfer clock	External clock (slave) which operates with a clock from master

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 sets the BUSY signal to the high level. In the case of an abnormal termination, the sample program displays "ERROR!" on the LCD and suppresses the execution of the subsequent operations.

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

5.1.2 WRITE Command

When the sample program receives the WRITE command, it writes the data it received into flash memory, and performs verify processing each time it completes the write of one block. The sample program sets the BUSY signal to the high level on normal termination of the command. In the case of an abnormal termination, the sample program displays "ERROR!" on the LCD and suppresses the execution of the subsequent operations.

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

5.1.3 END Command

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

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

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

5.1.4 Communication Sequence

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

(1) Sample program:

Sets the BUSY signal to the high level and notifies the sending side that command reception is enabled.

(2) Sending side:

Sends the START command.

(3) Sample program:

Turns on LED1 which indicates that flash memory is being accessed, sets the BUSY signal to the low level, and notifies the sending side that command reception is disabled. The program then performs initialization for flash memory self-programming. It sets the BUSY signal to the high level and notifies the sending side that command reception is enabled upon normal termination.

(4) Sending side:

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

(5) Sample program:

Sets the BUSY signal to the low level and notifies the sending side that command reception is disabled. It 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 sets the BUSY signal to the high level and notifies the sending side that command reception is enabled.

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

(7) Sending side:

Sends the END command.

(8) Sample program:

Sets the BUSY signal to the low level and notifies the sending side that command reception is disabled. The program performs verify processing on the block that is currently subjected to reprogramming. It 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 channel 2.

<Setting conditions>

- Uses the SAU0 channel 2 as CSI.
- Uses the P03/SI10 pin for data input.
- Uses the P04/SCK10 pin for the operation clock.
- Selects the single transfer mode as the transfer mode.
- Sets the data length to 8 bits.
- Sets the order of data transfer mode to MSB first.
- Sets the data transmission/reception timing to type 1.
- Selects the external clock as a transfer clock.

(2) Sets up the I/O port.

<Setting conditions>

- LED on/off control port (LED0): Sets P52 for output.
- BUSY signal output port: Sets P140 for output.

(3) Disables interrupts.

(4) Starts the CSI10.

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

(6) Sets P140 (BUSY signal) to the high level and notifies the sending side of the transmission-enabled state.

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

- Switches into the normal operation mode from the HALT mode upon a CSI transfer end interrupt request.

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

- Sets P52 to the low level and turn on LED0 indicating that flash memory is being accessed.
- Sets P140 (BUSY signal) to the low level and notify the sending side of the transmission-disabled state.
- 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.

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

(10) Sets P140 (BUSY signal) to the high level and notifies the sending side of the transmission-enabled state.

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

(12) Sets P140 (BUSY signal) to the low level and notifies the sending side of the transmission-disabled state.

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

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

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

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

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

(18) Sets P140 (BUSY signal) to the high level and notifies the sending side of the transmission-enabled state.

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

(20) Repeats steps (16) to (19) until 1 block (1024 bytes) of programming is completed or an END command (0x04) is received from the sending side. Proceed with the next step when 1 block 1024 bytes) of programming is completed or an END command (0x04) is received from the sending side.

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

(22) Repeats steps (13) to (21) unless an END command (0x04) is received from the sending side. Proceeds with the next step when an END command is received.

(23) 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.

(24) 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_SelfInitialize R_MAIN_SelfExecute R_MAIN_WriteExecute
r_cg_serial_user.c	SAU module	Additional functions: R_CSI10_ReceiveStart
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 setting 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)
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
PORT_LOW	0	Low level of BUSY signal port
PORT_HIGH	1	High level of BUSY signal port

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_CSI10_Start	Starts CSI10.
R_CSI10_ReceiveStart	Receives data via CSI10.
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 self-programming.

5.9 Function Specifications

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

[Function Name] R_CSI10_Start

Synopsis	Start CSI10.	
Header	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h	
Declaration	void R_CSI10_Start(void)	
Explanation	This function clears a CSI10 interrupt request flag (CSIIF10 = 0). It starts the CSI10 after an interrupt is enabled (CSIMK10 = 0).	
Arguments	None	
Return value	None	
Remarks	None	

[Function Name] R_CSI10_ReceiveStart

Synopsis	Receive data via CSI10.	
Header	r_cg_macrodriver.h r_cg_serial.h r_cg_userdefine.h	
Declaration	uint8_t R_CSI10_ReceiveStart(uint16_t *rxlength, uint8_t *rxbuf)	
Explanation	This function stores the receive data in the receive buffer (rxbuf) and the receive data length [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	
	Parameter error (txlength is smaller than 0): 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_WriteExecute(uint32_t WriteAddr)
Explanation	This function rewrites the data in the code flash memory.
Arguments	WriteAddr Write start address
Return value	Normal termination: NORMAL_END Abnormal termination: ERROR
Remarks	None

5.10 Flowcharts

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

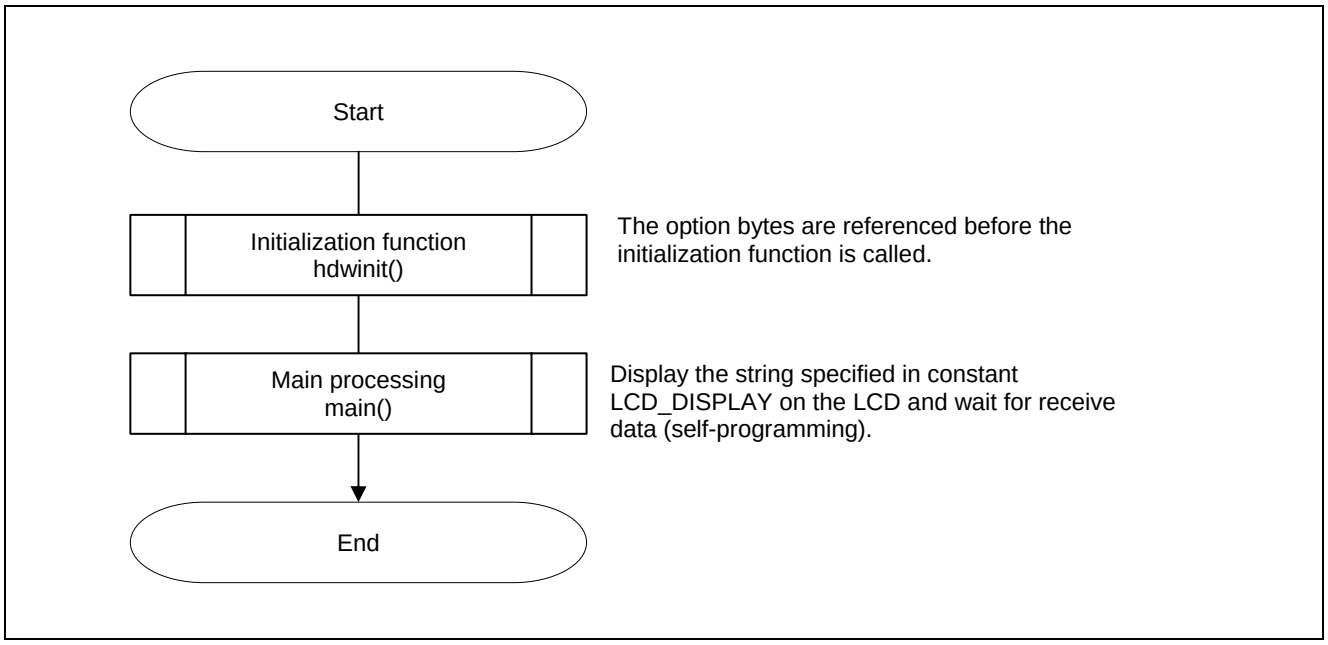


Figure 5.1 Overall Flow

5.10.1 Initialization Function

Figure 5.2 shows the flowchart for the initialization function.

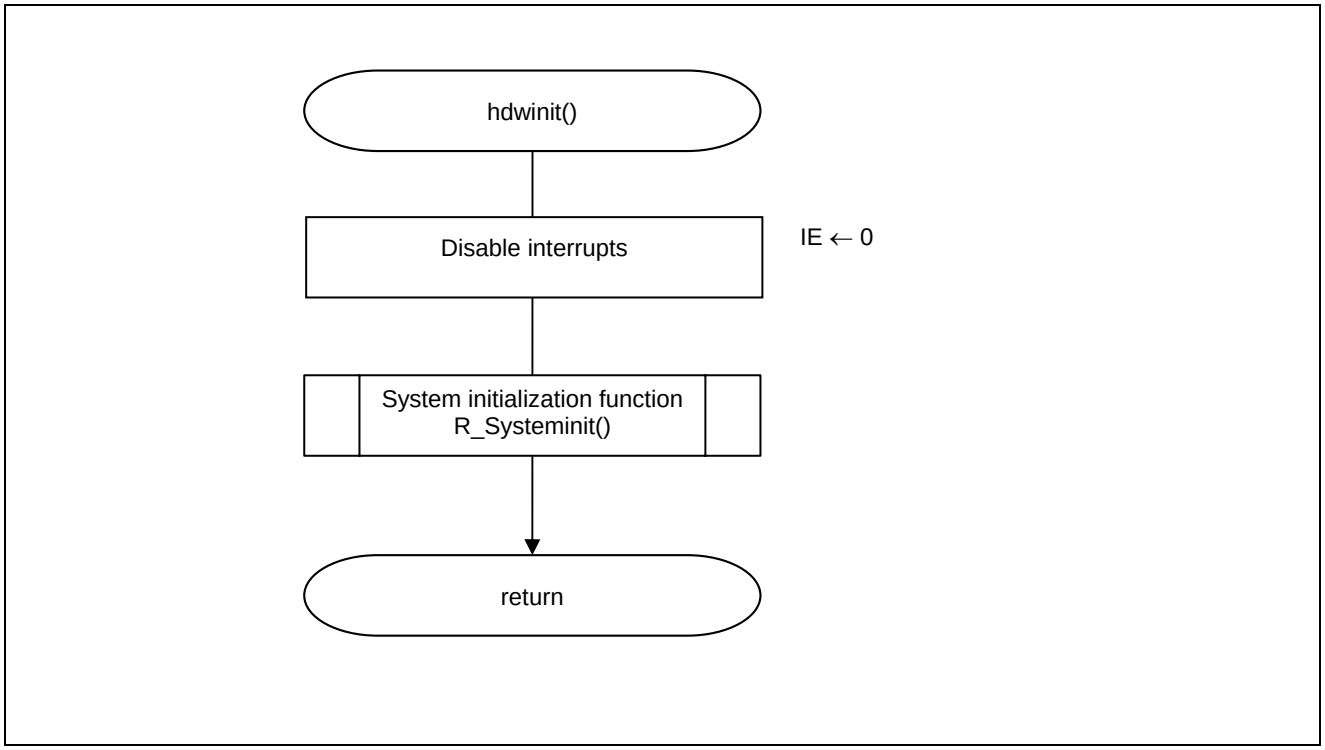


Figure 5.2 Initialization Function

5.10.2 System Initialization Function

Figure 5.3 shows the flowchart for the system initialization function.

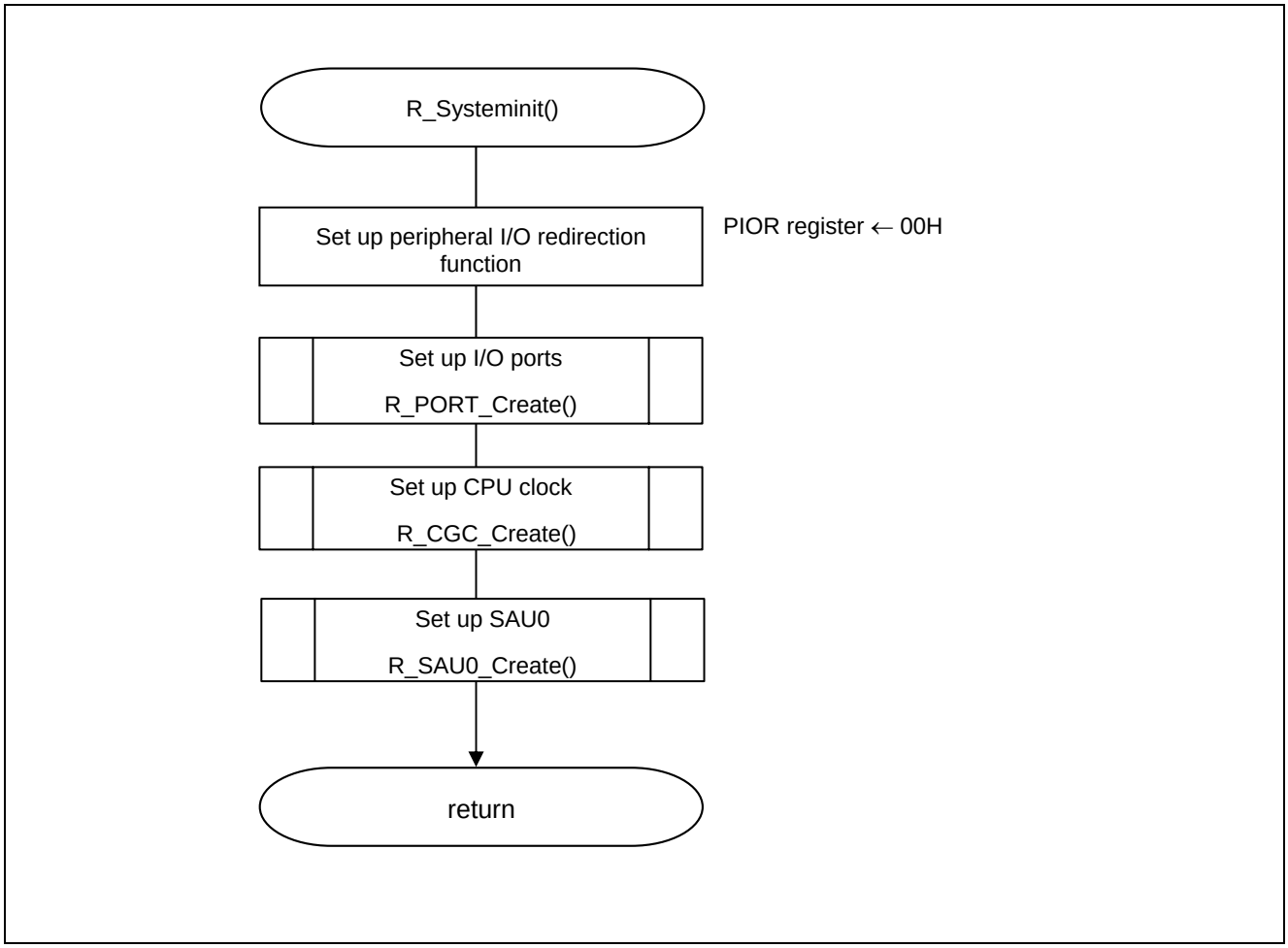


Figure 5.3 System Initialization Function

5.10.3 I/O Port Setup

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

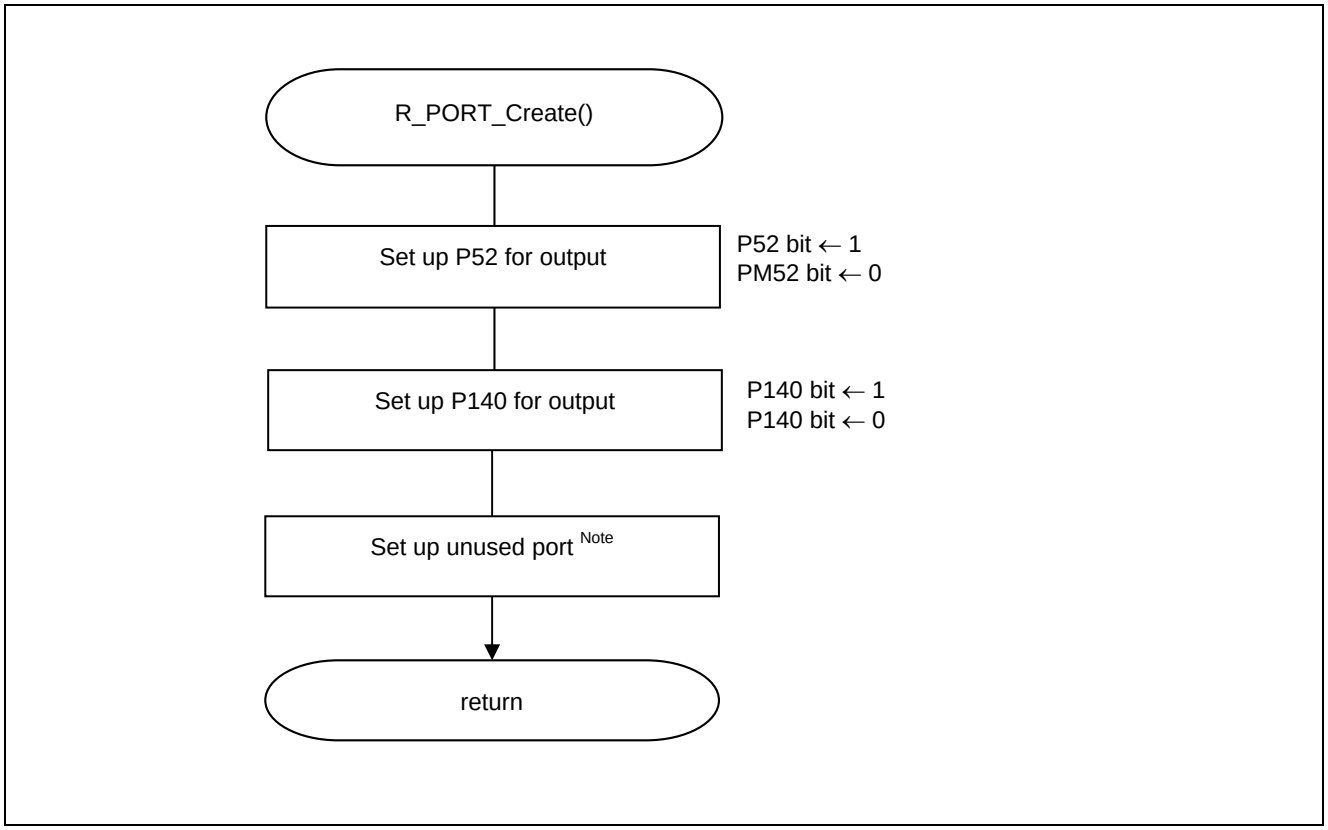


Figure 5.4 I/O Port Setup

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.

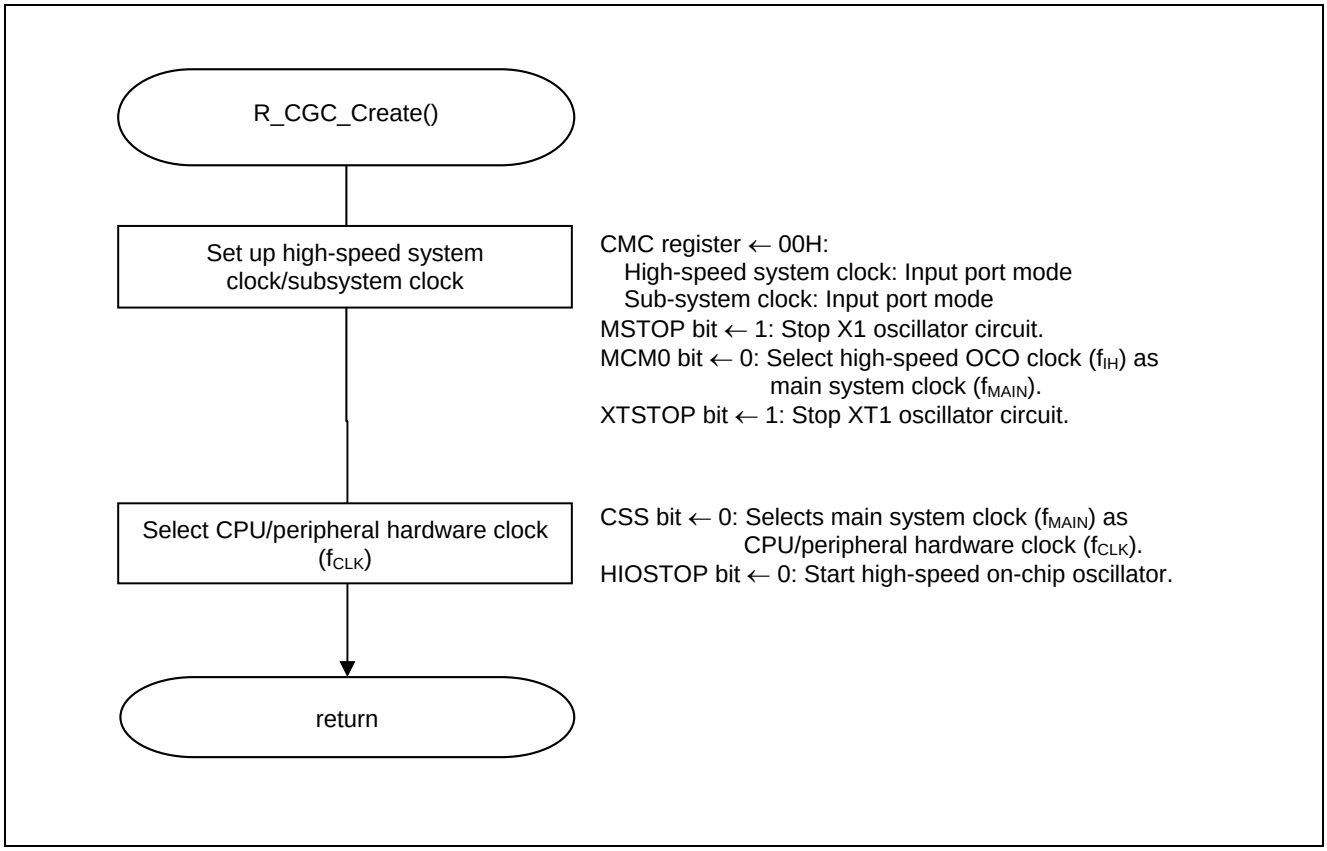


Figure 5.5 CPU Clock Setup

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.

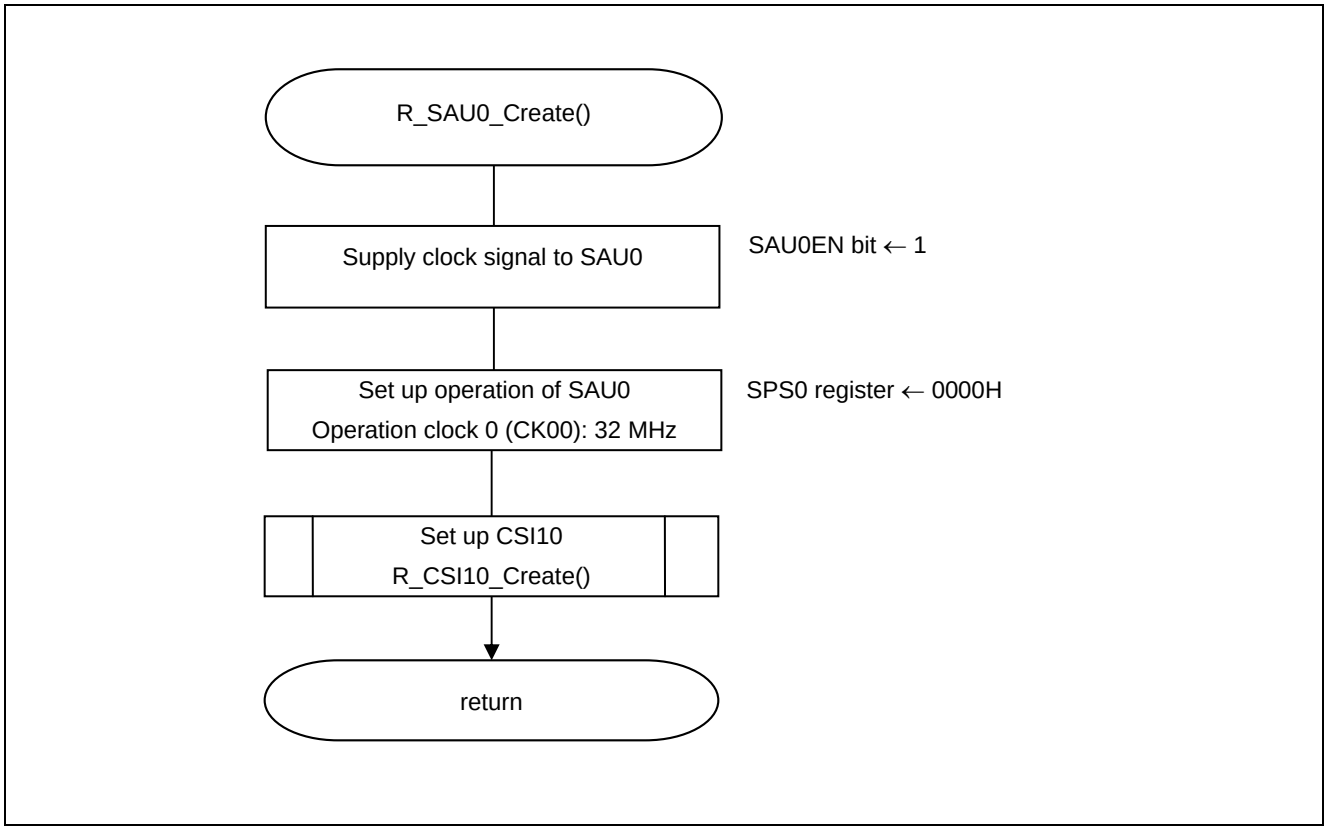


Figure 5.6 SAU0 Setup

5.10.6 CSI Setup

Figure 5.7 shows the flowchart for CSI10 setup (1/2). Figure 5.8 shows the flowchart for CSI10 setup (2/2).

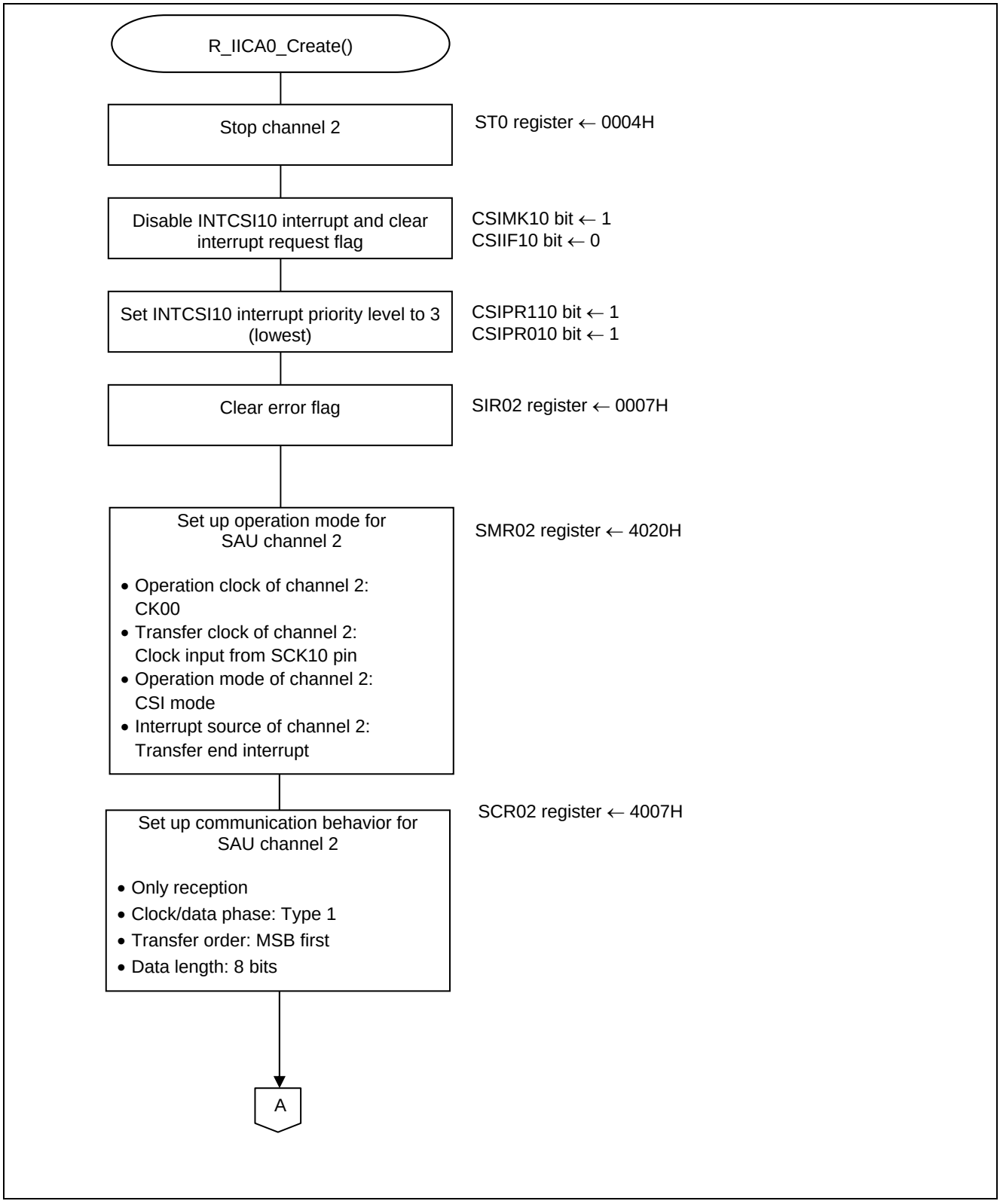


Figure 5.7 CSI10 Setup (1/2)

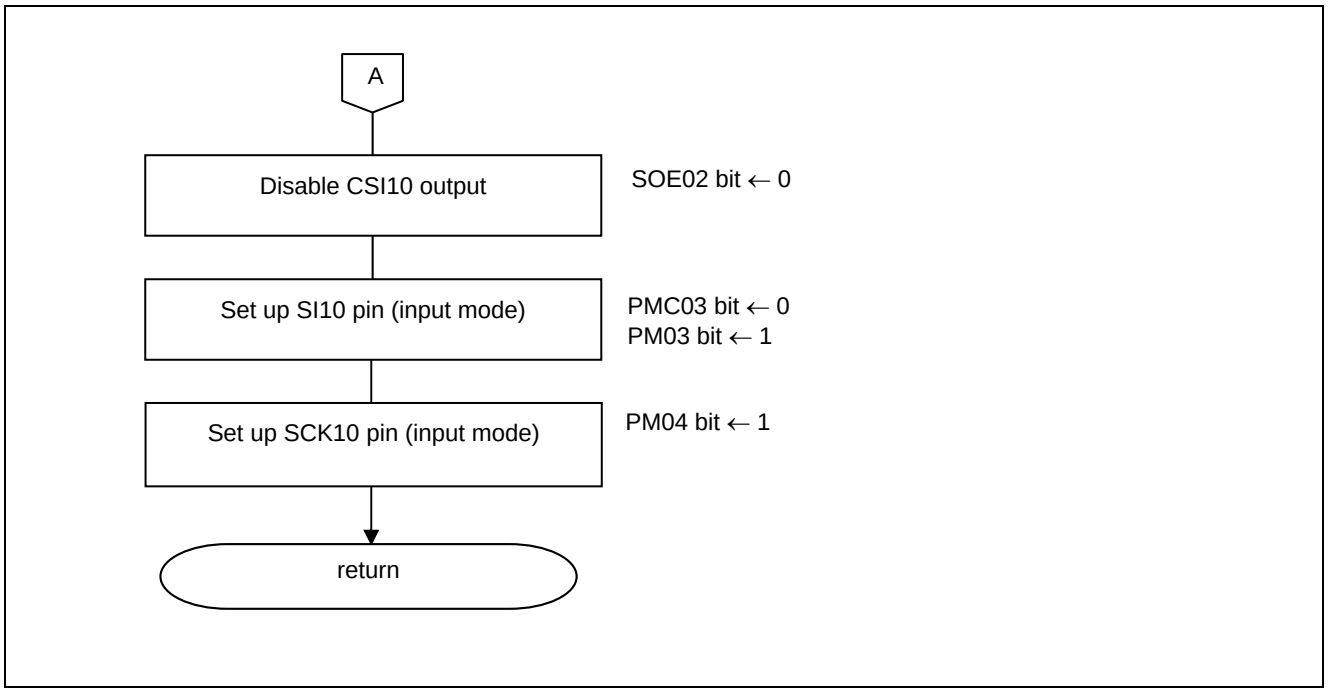


Figure 5.8 CSI10 Setup (2/2)

5.10.7 Main Processing

Figure 5.9 shows the flowchart for main processing (1/2). Figure 5.10 shows the flowchart for main processing (2/2).

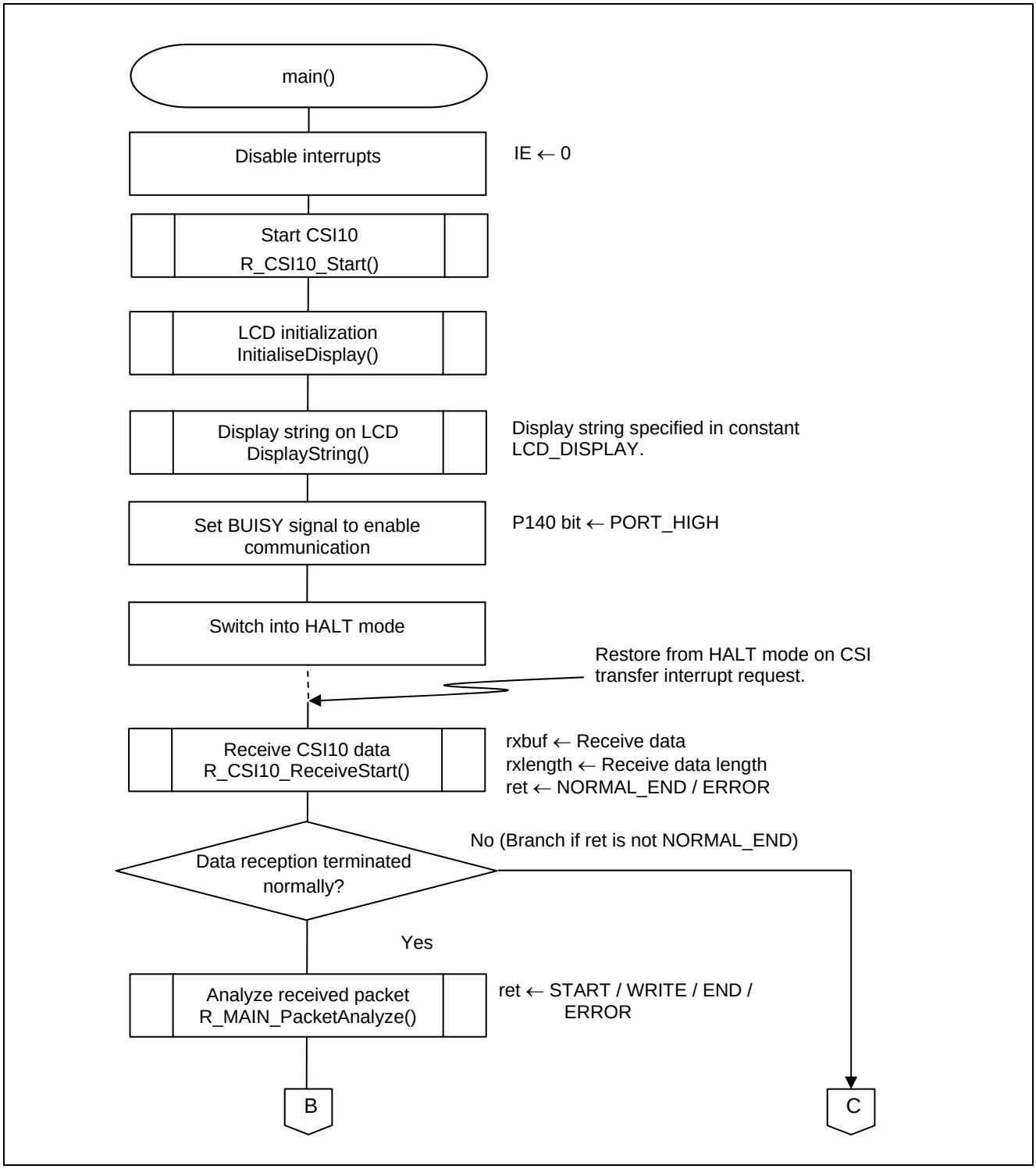


Figure 5.9 Main Processing (1/2)

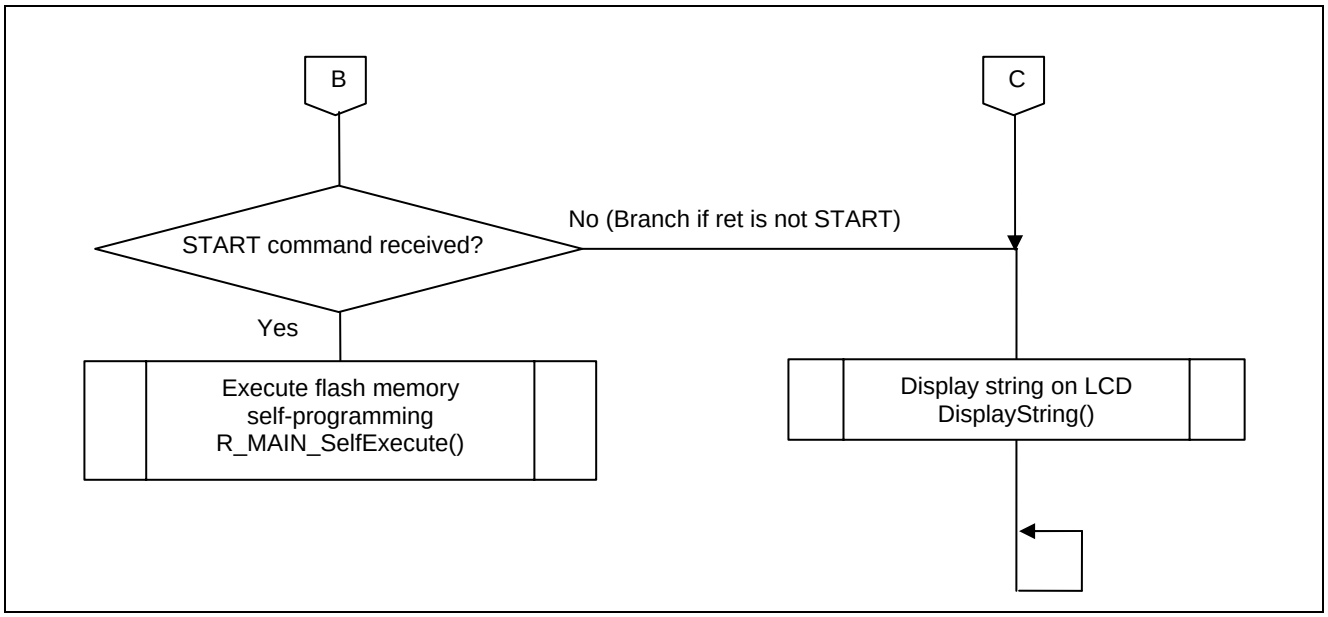


Figure 5.10 Main Processing (2/2)

5.10.8 Starting the CSI10

Figure 5.11 shows the flowchart for starting the CSI10.

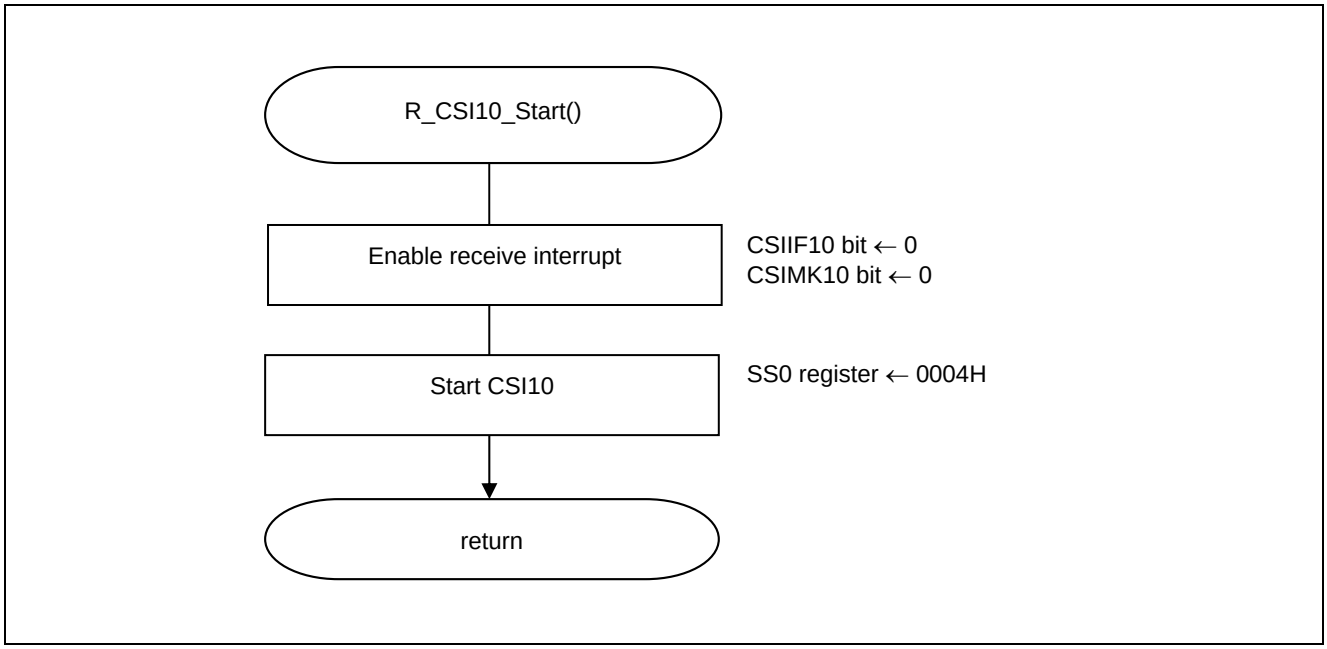


Figure 5.11 Starting the CSI10

5.10.9 Data Reception via CSI10

Figure 5.12 shows the flowchart for data reception via the CSI10 (1/3). Figure 5.13 shows the flowchart for data reception via the CSI10 (2/3). Figure 5.14 shows the flowchart for data reception via the CSI10 (3/3).

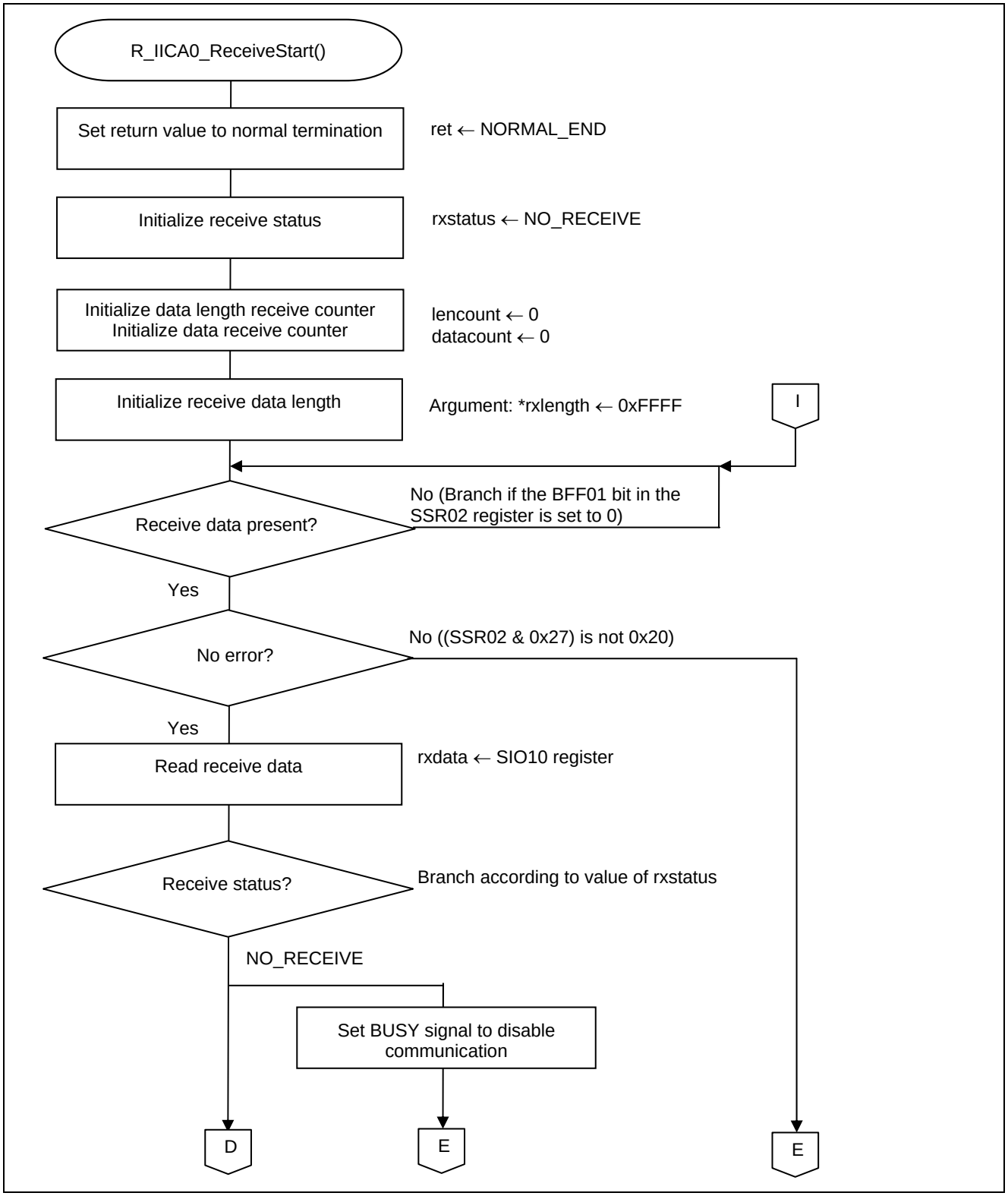


Figure 5.12 Data Reception via CSI10 (1/3)

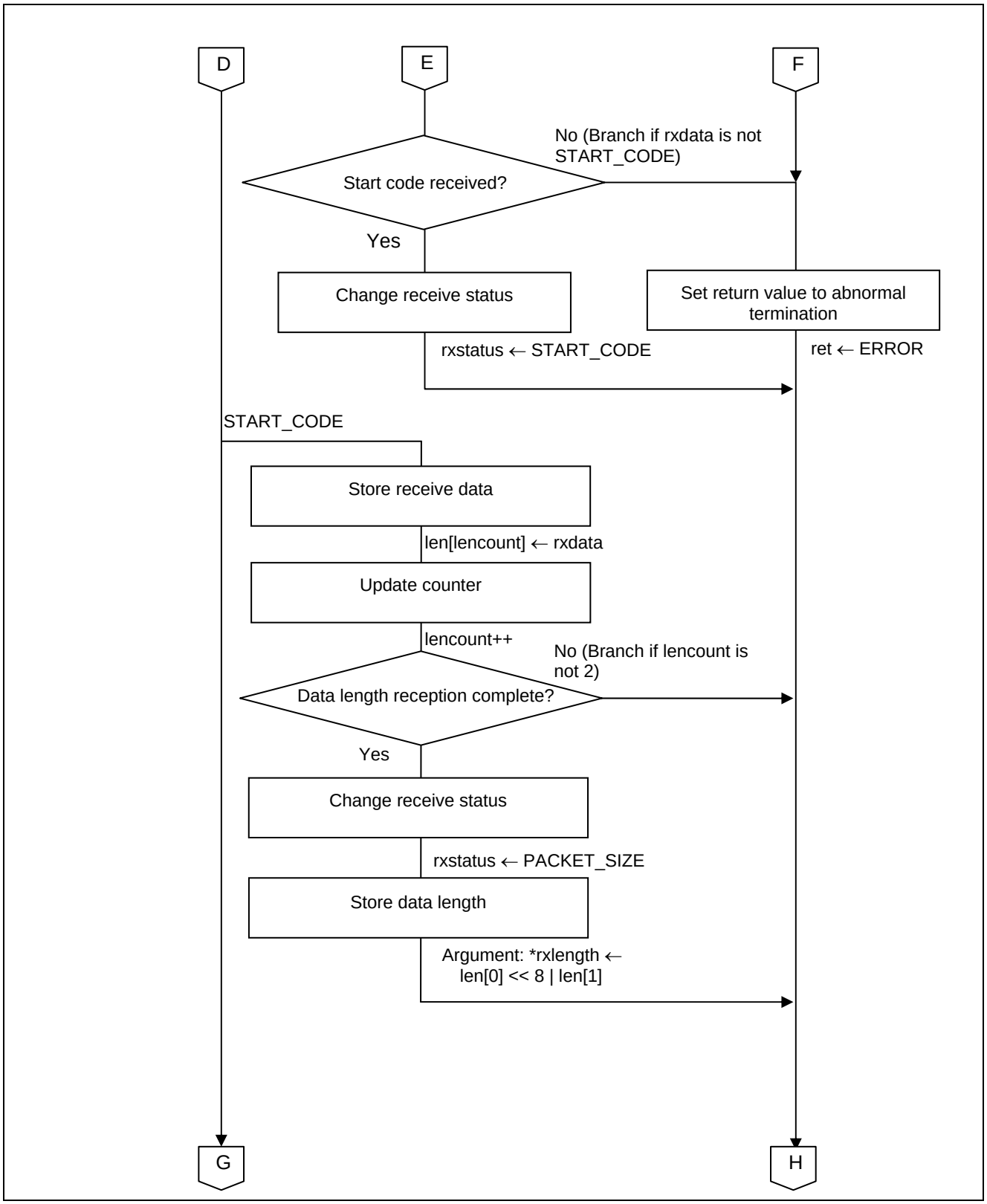


Figure 5.13 Data Reception via CSI10 (2/3)

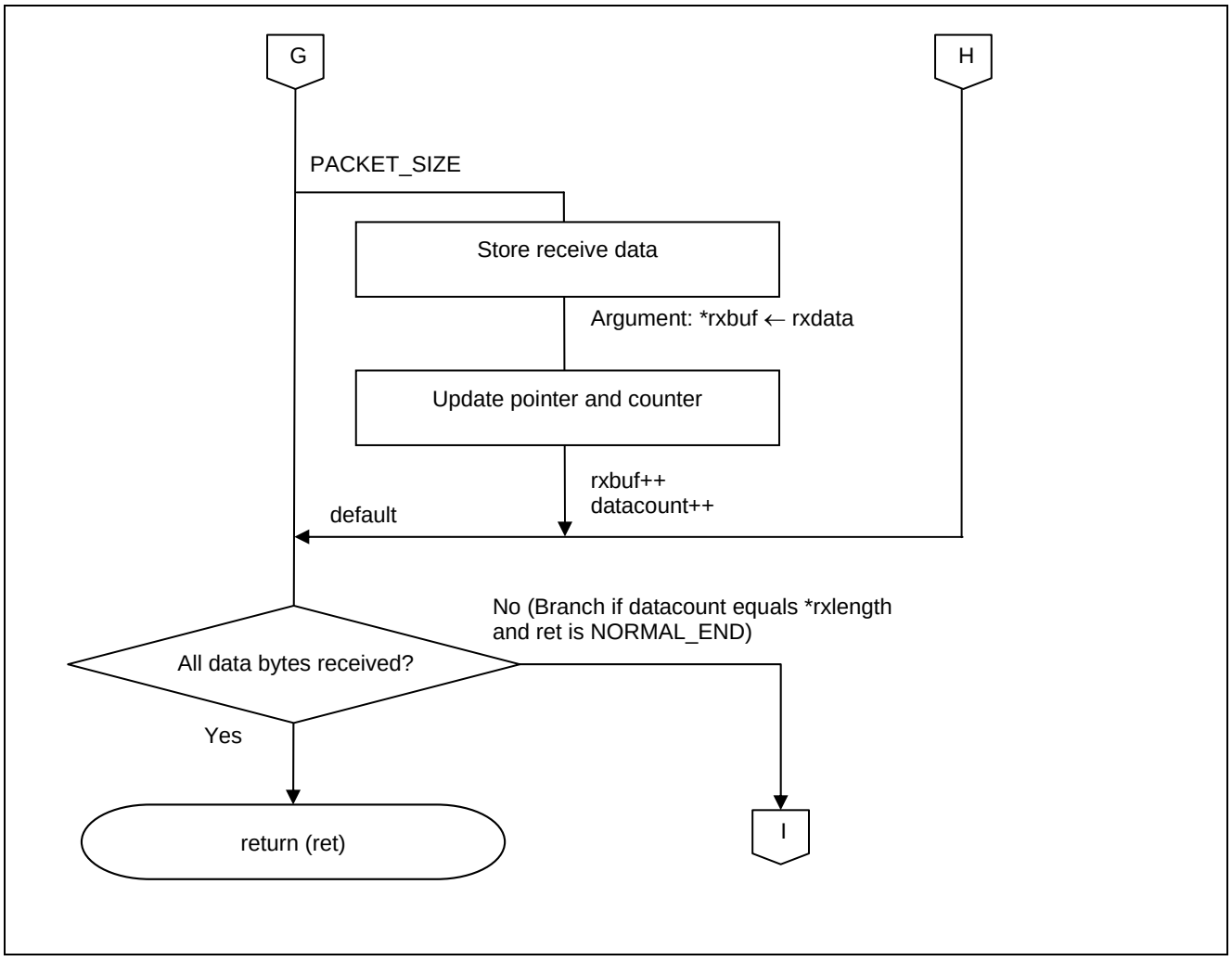


Figure 5.14 Data Reception via CSI10 (3/3)

5.10.10 Receive Packet Analysis

Figure 5.15 shows the flowchart for receive packet analysis.

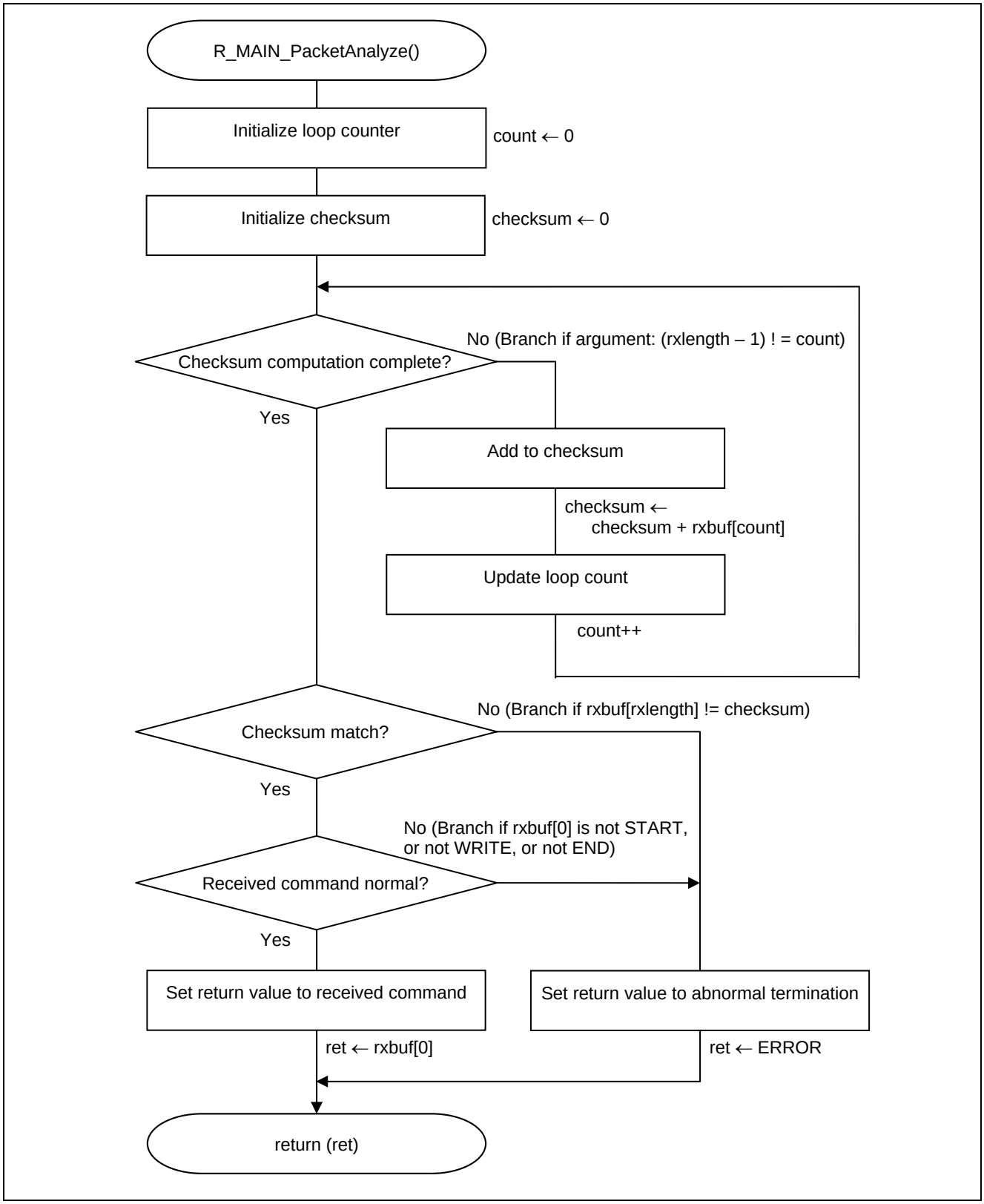


Figure 5.15 Receive Packet Analysis

5.10.11 Flash Memory Self-Programming Execution

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

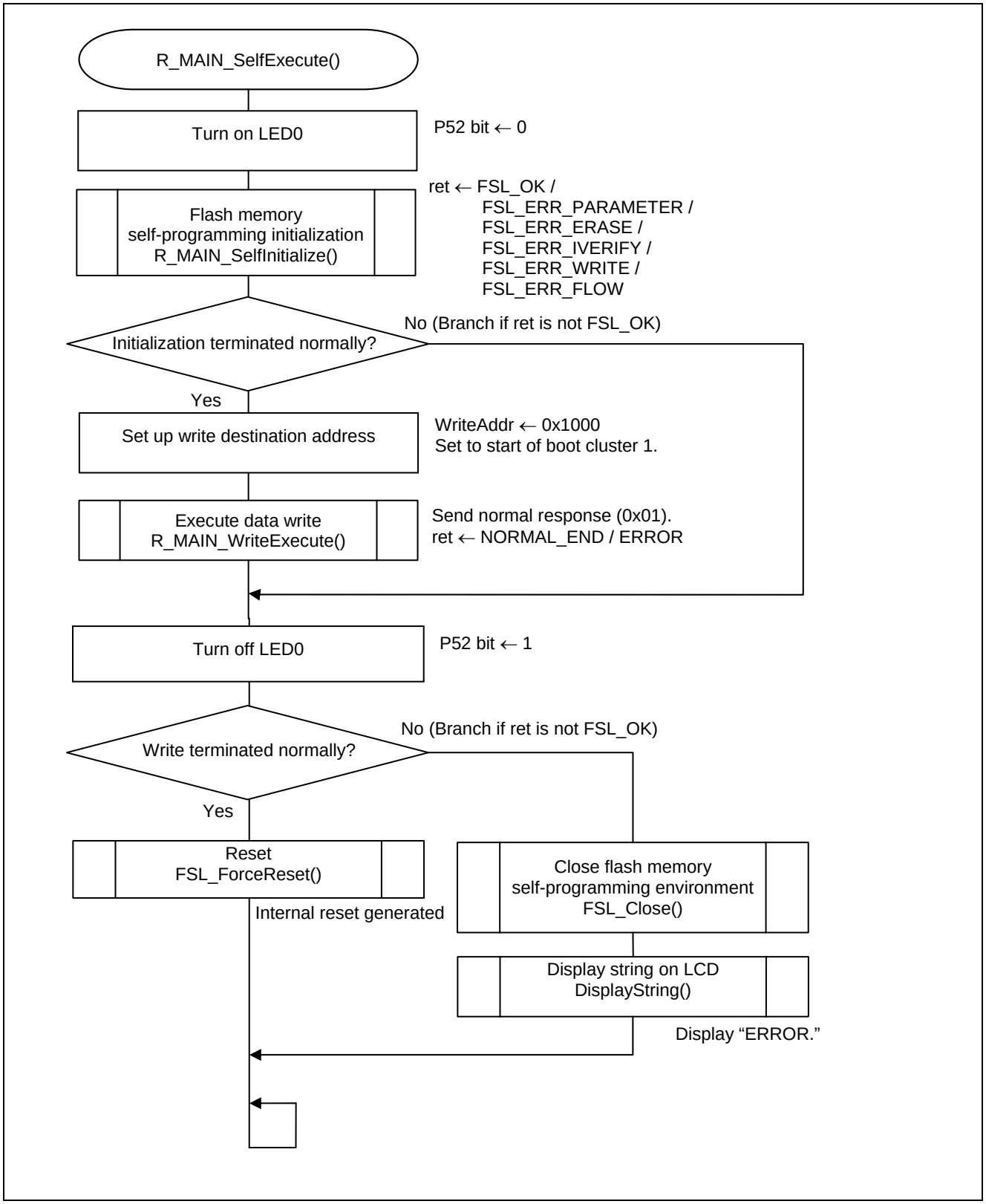


Figure 5.16 Flash Memory Self-Programming Execution

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).

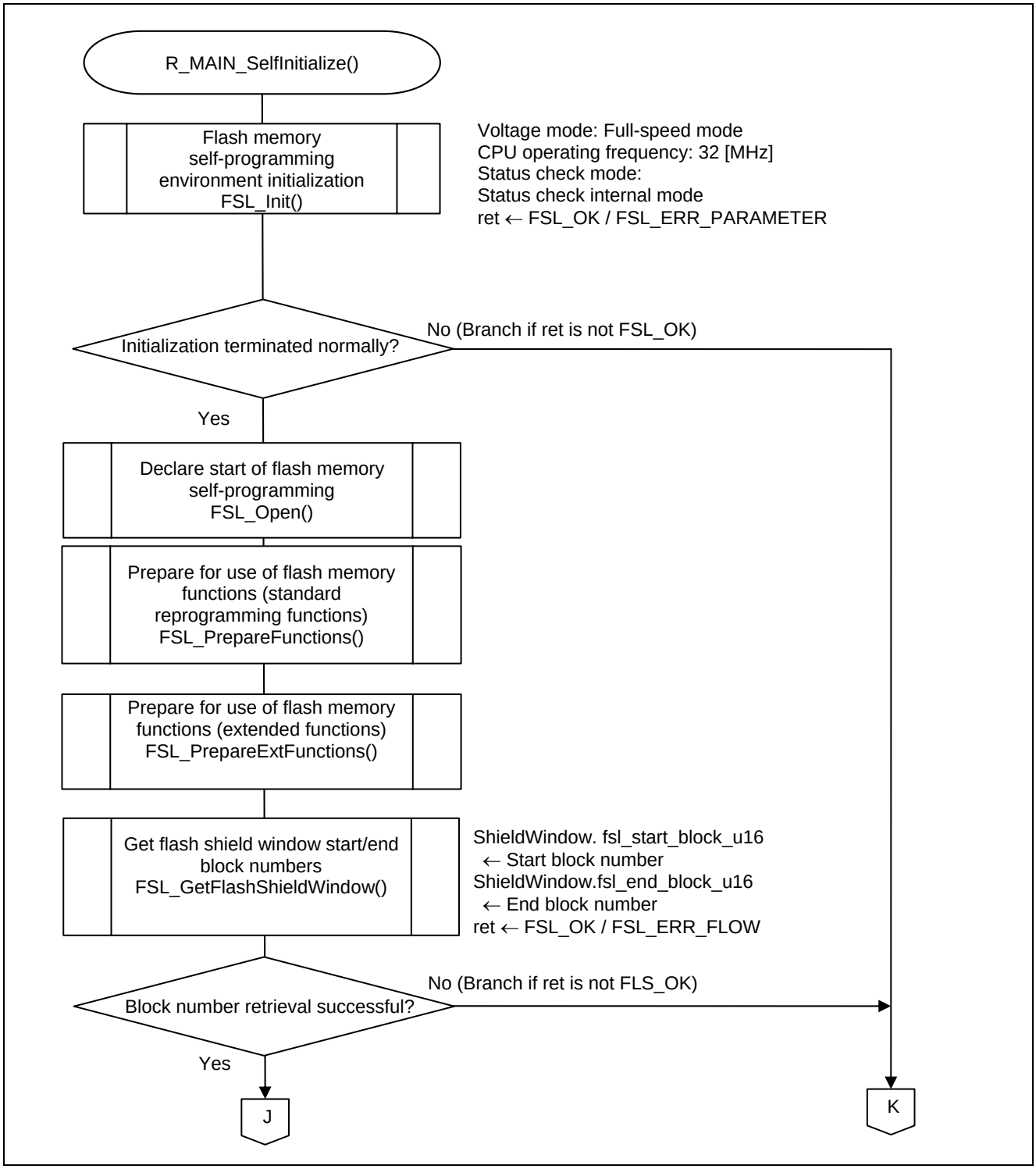


Figure 5.17 Flash Memory Self-Programming Initialization (1/2)

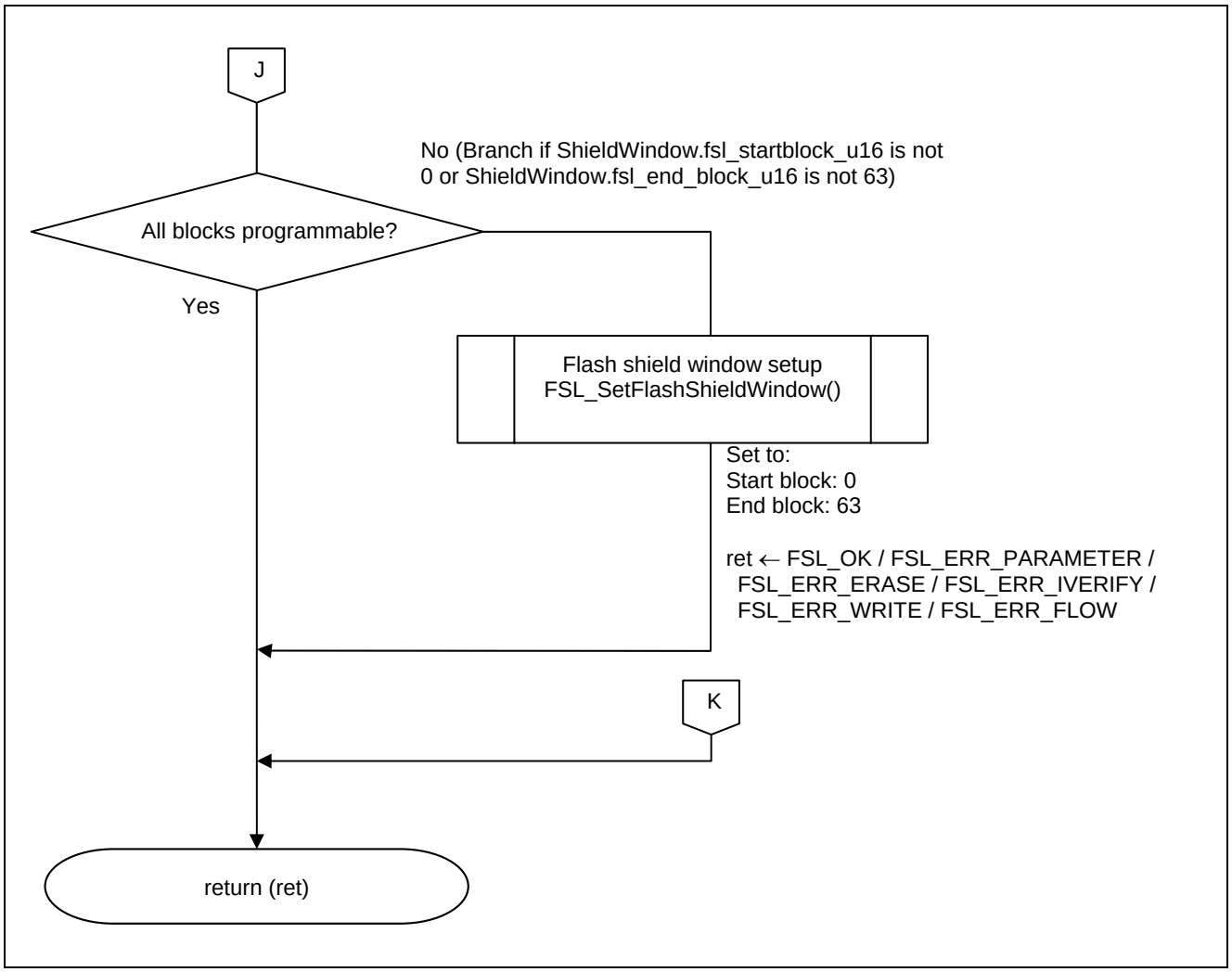


Figure 5.18 Flash Memory Self-Programming Initialization (2/2)

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).

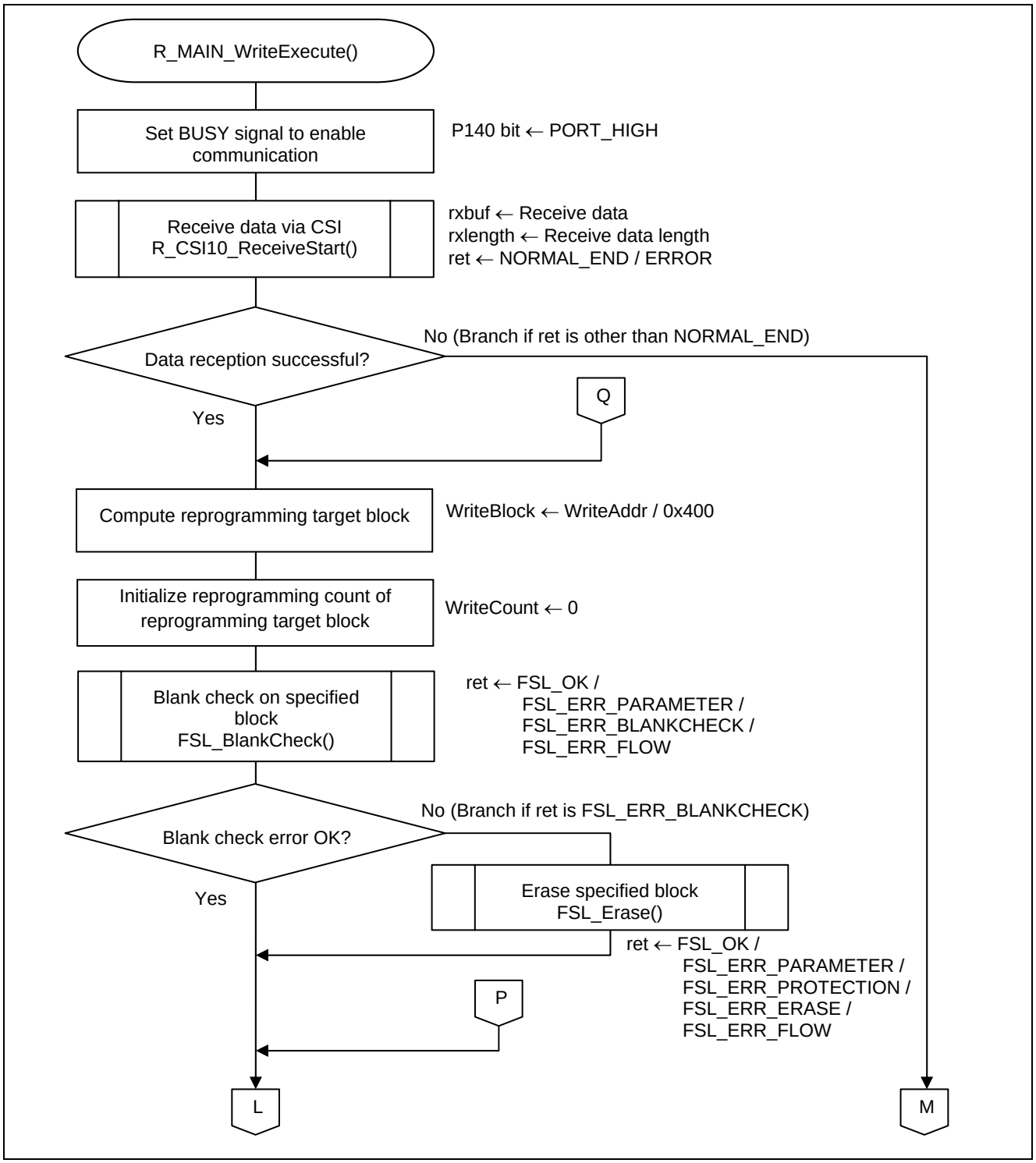


Figure 5.19 Flash Memory Reprogramming Execution (1/3)

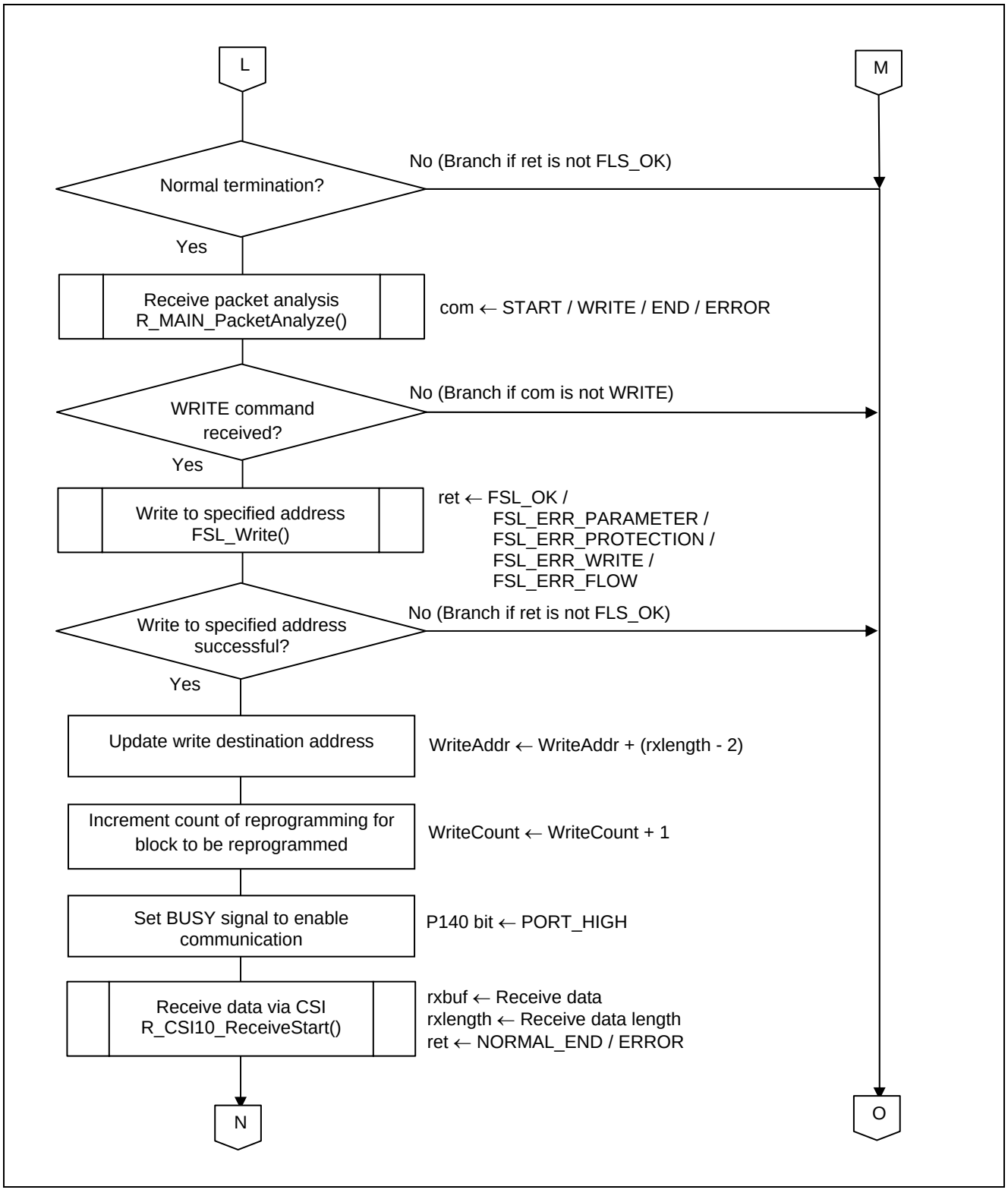


Figure 5.20 Flash Memory Reprogramming Execution (2/3)

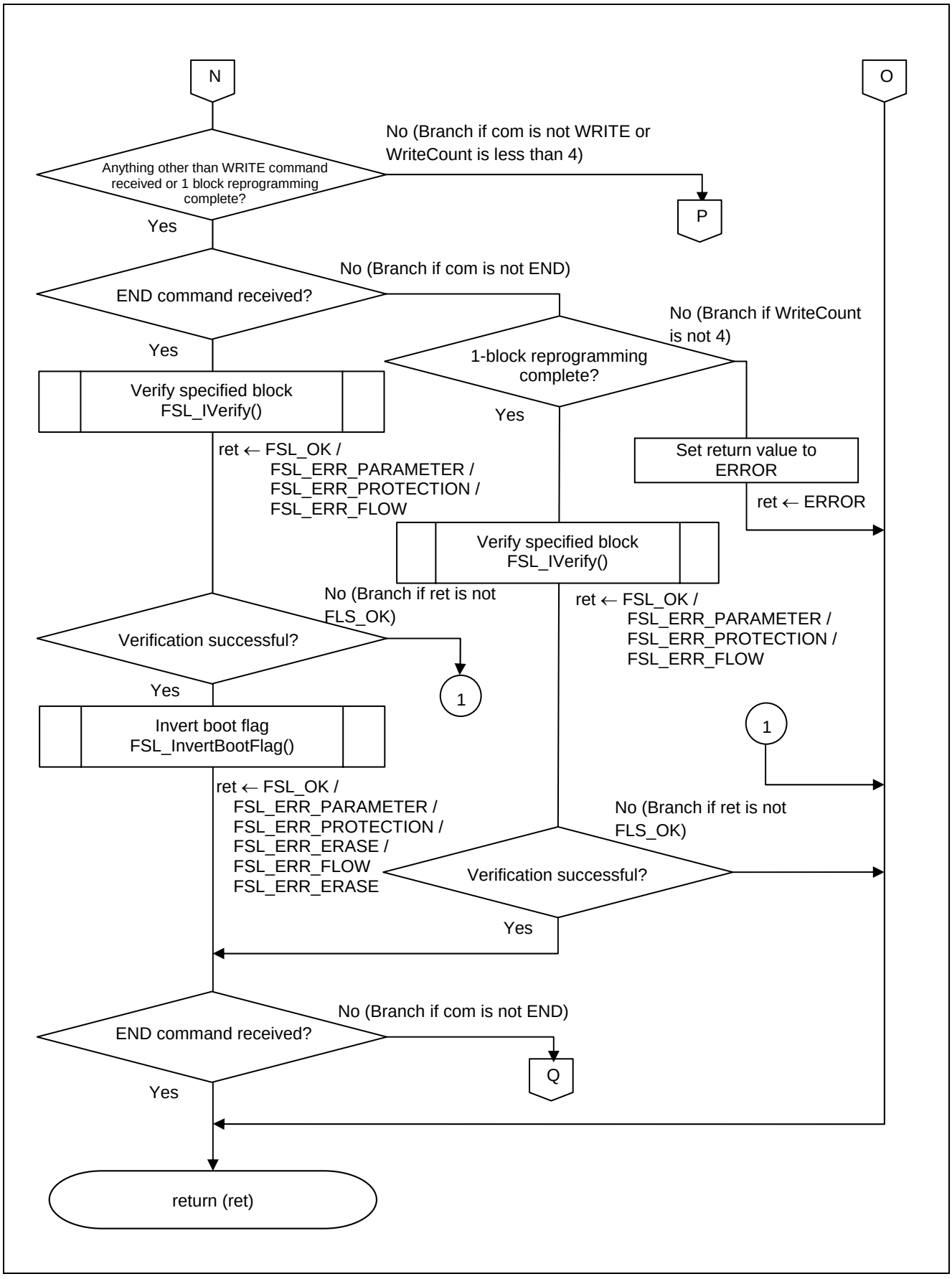


Figure 5.21 Flash Memory Reprogramming Execution (3/3)

5.11 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. 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.

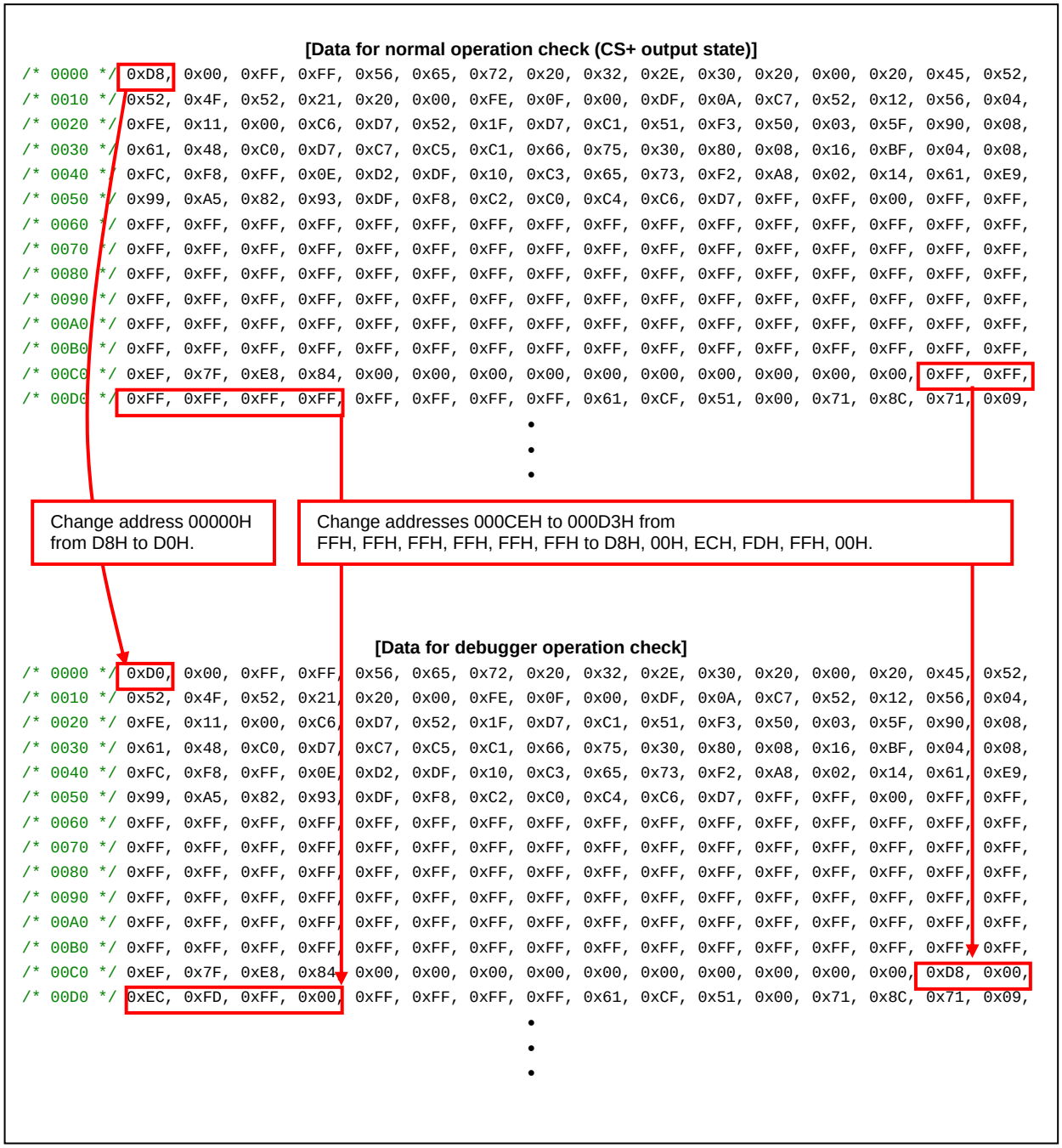
(5) A reset occurs and "Ver 2.0" is displayed on the LCD.

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 (CSI)

Rev.	Date	Description	
		Page	Summary
1.00	Jan. 29, 2016	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

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

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

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

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

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

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

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

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

Colophon 5.0