RL78/G23

Updating Firmware by Using UART Communication and Boot Swapping

Introduction

This application note describes how to update firmware in code flash memory by using an update program that remains in the code flash memory.

In this method, the code flash memory is divided into two areas: the Execute area and the Temporary area.

Renesas Flash Driver RL78 Type01 is used to reprogram the flash memory and perform boot swapping.

Target Device

RL78/G23

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

RL78/G23

使用UART通信和启动交换更新固件

Introduction

本应用笔记介绍了如何使用代码闪存中保留的更新程序来更新代码闪存中的固件。

在该方法中，代码闪存被分成两个区域：执行区域和临时区域。

瑞萨闪存驱动器RL78001用于重新编程闪存并执行引导交换。

目标设备

RL78/G23

当将本申请说明中涵盖的示例程序应用于另一微型计算机时，根据针对目标微型计算机的规范修改该程序，并对修改后的程序进行广泛的评估。

Contents

Contents

1. Specifications

1.1 Overview of Specifications

The sample program covered in this application note updates the firmware in the code flash memory.

The boot area is reprogrammed by using the boot swapping function. The other areas are reprogrammed by using temporary areas in which the reprogramming data is temporarily saved. This method allows the firmware to be updated while the user program (application) is running.

The firmware is updated via UART communication by using four commands: START, WRITE_BOOT1, WRITE_TEMP, and END.

The execution status of the application and commands is indicated by LEDs.

Two sample projects are included in this application note, each can be replaced by firmware updates.

If you use a product with ROM size other than 128 KB or 768 KB, please refer to “1.2.8 Copy Flag” and “4.3 Setting the ROM Size Specification Constant” and modify the sample programs.

Table 1-1 Directory of Sample Project

workspace			Description
\workspace			
	\CS+ \e2studio \IAR		
		\128KB	Project for 128KB products
		\LED1	Sample project 1 (Blinks LED1)
		\LED8	Sample project 2 (Blinks LED8)
		\768KB	Project for 768KB products
		\LED1	Sample project 1 (Blinks LED1)
		\LED8	Sample project 2 (Blinks LED8)

LED output port assign differ between the project for 128KB and the project for 768KB. In this application note, in the case of using the project for 128 KB is explained as an example. When using the project for 768 KB, please read the port numbers as shown in the table below.

Table 1-2 Assigned port for LED output

LED no.	Project for 128KB products	Project for 768KB products
LED1	P03	P33
LED2	P02	P34
LED3	P43	P145
LED4	P42	P106
LED5	P77	P105
LED6	P41	P104
LED7	P31	P103
LED8	P76	P46

1. Specifications

1.1规格概述

本应用笔记中介绍的示例程序更新代码闪存中的固件。

通过使用引导交换功能重新编程引导区域。其他区域通过使用其中临时保存重编程数据的临时区域来重编程。该方法允许在用户程序（应用程序）运行时更新固件。

固件通过uart通信使用四个命令进行更新：START，WRITE_BOOT1
WRITE_TEMP，并结束。

应用程序和命令的执行状态由Led指示。

本应用笔记中包含两个示例项目，每个项目都可以通过固件更新进行替换。

如果您使用的ROM大小不是128KB或768KB的产品，请参阅"1.2.8复制标志"和"4.3设置ROM大小规格常数"并修改示例程序。

表1-1示例项目目录

workspace			Description
\workspace			
	\CS+ \e2studio \IAR		
		\128KB	128kb产品项目
		\LED1	示例项目1（闪烁LED1）
		\LED8	示例项目2（闪烁LED8）
		\768KB	768kb产品项目
		\LED1	示例项目1（闪烁LED1）
		\LED8	示例项目2（闪烁LED8）

LED输出端口在128kb的项目和768KB的项目之间分配不同。在本应用笔记中，以使用128KB的项目为例进行说明。当使用768KB的项目时，请阅读下表所示的端口号。

表1-2Led输出的分配端口

LED号。	128kb产品项目	768kb产品项目
LED1	P03	P33
LED2	P02	P34
LED3	P43	P145
LED4	P42	P106
LED5	P77	P105
LED6	P41	P104
LED7	P31	P103
LED8	P76	P46

Table 1-3 Peripheral Function and Use

Peripheral Function	Use
Serial Array Unit UART0	Data communication
P03, P02, P43, P42, P77, P41, P31, P76	Digital output controlling LED1 to LED8

Table 1-4 Application operating state and indication on LED1 to LED8. (Updating sample project 1 to 2)

Application operating state	Indication on LED1 to LED8	Operating firmware
Application before updated is running	LED1 blinks	Sample project 1
START command received	LED2 lights up	
WRITE_BOOT1 command received	LED3 lights up	
WRITE_TEMP command received	LED4 lights up	
END command received	LED5 lights up	
Temporary area being copied	LED6 lights up	
Error termination	Only LED7 lights up	
Application after updated is running	LED8 blinks	Sample project 2

1.1.1 Overview of Renesas Flash Driver RL78 Type01

Renesas Flash Driver RL78 Type01 is software that reprograms the firmware in the code flash memory installed on an RL78 microcontroller.

The content of the code flash memory can be reprogrammed by calling Renesas Flash Driver RL78 Type01 from the user program.

To perform flash memory self-programming, the user program needs to perform the necessary initialization processing and run the functions that correspond to the necessary operations in C or assembly language.

表1-3外设功能和用途

外围功能	
串行阵列单元UART0	数据通讯
P03, P02, P43, P42, P77, P41, P31, P76	数字输出控制LED1至LED8

表1-4LED1至LED8上的应用运行状态和指示。（更新示例项目1至2）

应用程序运行状态	Led1至LED8指示	操作固件
更新前的应用程序正在运行	LED1 blinks	样本项目1
收到启动命令	LED2亮起	
收到WRITE_BOOT1命令	LED3亮起	
收到的WRITE_TEMP命令	LED4亮起	
接收到的结束命令	LED5亮起	
正在复制的临时区域	LED6亮起	
错误终止	只有LED7亮起	
更新后的应用程序正在运行	LED8 blinks	样本项目2

1.1.1瑞萨闪存驱动器RL78Type01概述

瑞萨闪存驱动器RL78Type01是一种软件，可对安装在RL78微控制器上的代码闪存中的固件进行重新编程。

代码闪存的内容可以通过调用瑞萨闪存驱动器RL78重新编程Type01来自用户程序。

要执行闪存自编程，用户程序需要执行必要的初始化处理，并以C或汇编语言运行与必要操作相对应的功能。

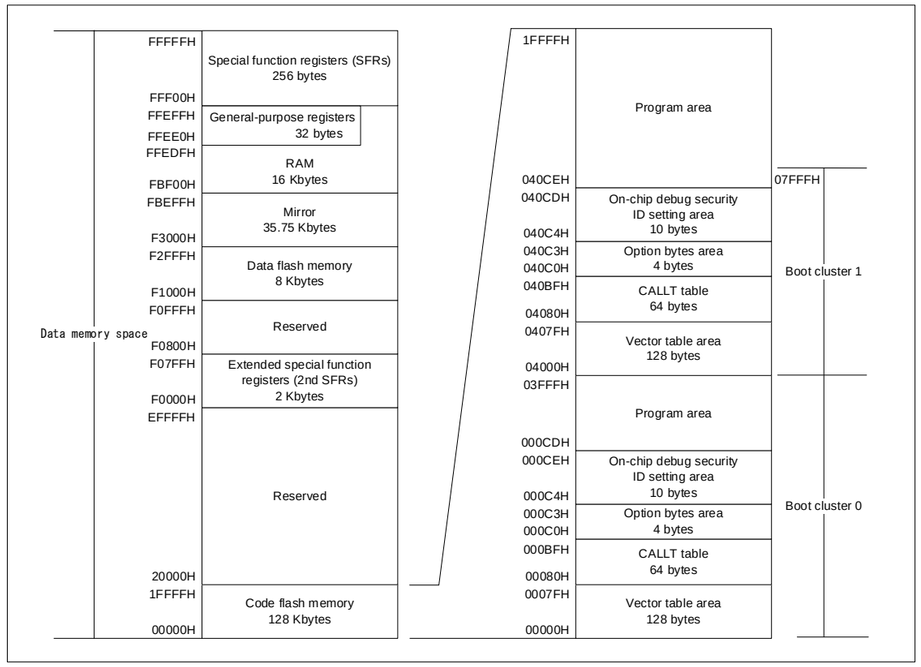
1.1.2 Code Flash Memory

The program area (from address 08000H to the last address) is divided into two areas and the sample program covered in this application note uses these two areas. The first area (from address 08000H to the boundary) is called the Execute area and the second area (from the boundary to the last address) is called the Temporary area. The address of the boundary and the last address differ depending on the size of the ROM. The update program is written in boot cluster 1 and the Temporary area. Therefore, if you write a user program, make sure that it is stored within boot cluster 0 and the Execute area.

Table 1-5 Start and End Addresses of the Two Areas According to ROM Size

ROM size	Execute area	Temporary area
96KB	08000H to FFFFH	10000H to 17FFFH
128KB	08000H to 13FFFH	14000H to 1FFFFH
192KB	08000H to 1BFFFH	1C000H to 2FFFFH
256KB	08000H to 23FFFH	24000H to 3FFFFH
384KB	08000H to 33FFFH	34000H to 5FFFFH
512KB	08000H to 43FFFH	44000H to 7FFFFH
768KB	08000H to 63FFFH	64000H to BFFFFH

Figure 1-1 Memory Map



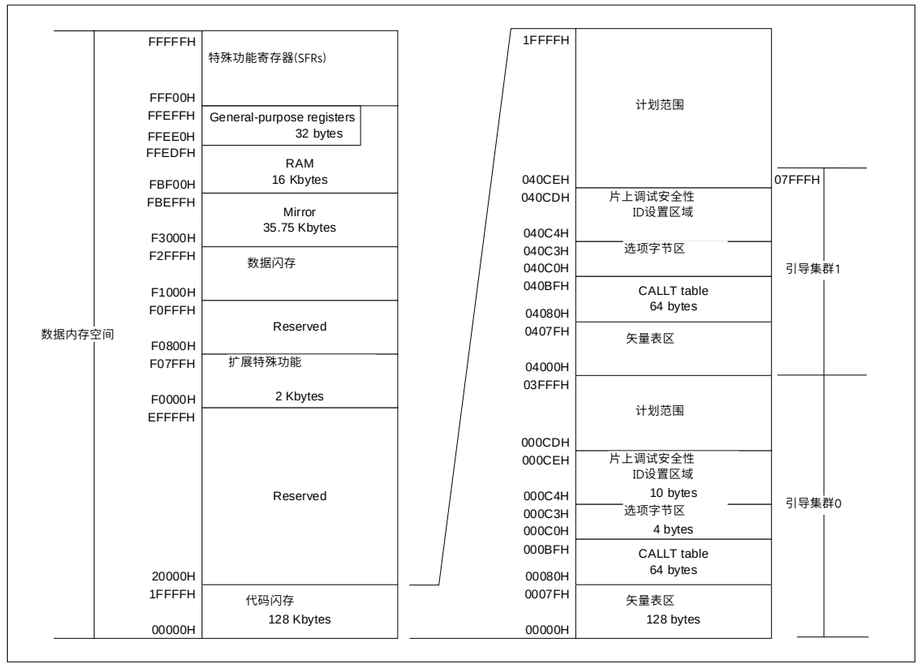
Caution: If you use the boot swap function, make sure that the same value that is set in the option byte area in boot cluster 0 (000C0H to 000C3H) is also set in the option byte area in boot cluster 1 (010C0H to 010C3H) because these areas are swapped by the function.

1.1.2代码闪存程序区域（从地址08000H到最后一个地址）分为两个区域，本应用笔记中涉及的示例程序使用这两个区域。第一个区域（从地址08000H到边界）称为执行区域，第二个区域（从边界到最后一个地址）称为临时区域。边界的地址和最后的地址根据ROM的大小而不同。更新程序写在引导群集1和临时区域中。因此，如果您编写用户程序，请确保它存储在bootcluster0和Execute区域内。

表1-5根据ROM大小的两个区域的开始和结束地址

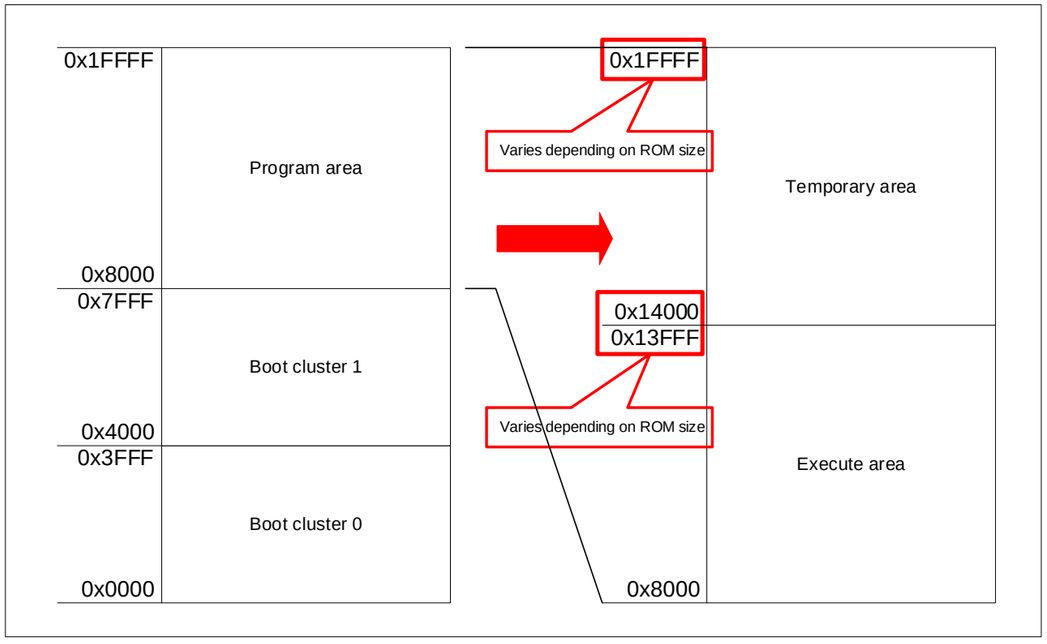
ROM大小	执行区域	临时区域
96KB	08000H to FFFFH	10000H to 17FFFH
128KB	08000H to 13FFFH	14000H to 1FFFFH
192KB	08000H to 1BFFFH	1C000H to 2FFFFH
256KB	08000H to 23FFFH	24000H to 3FFFFH
384KB	08000H to 33FFFH	34000H to 5FFFFH
512KB	08000H to 43FFFH	44000H to 7FFFFH
768KB	08000H to 63FFFH	64000H to BFFFFH

图1-1内存映射



注意：如果使用引导交换功能，请确保在引导群集0（000c0h到000c3h）的选项字节区域中设置的相同值也设置在引导群集1（010c0h到010c3h）的选项字节区域中，因为这些区域是由该功能交换的。

Figure 1-2 Code Flash Memory Map



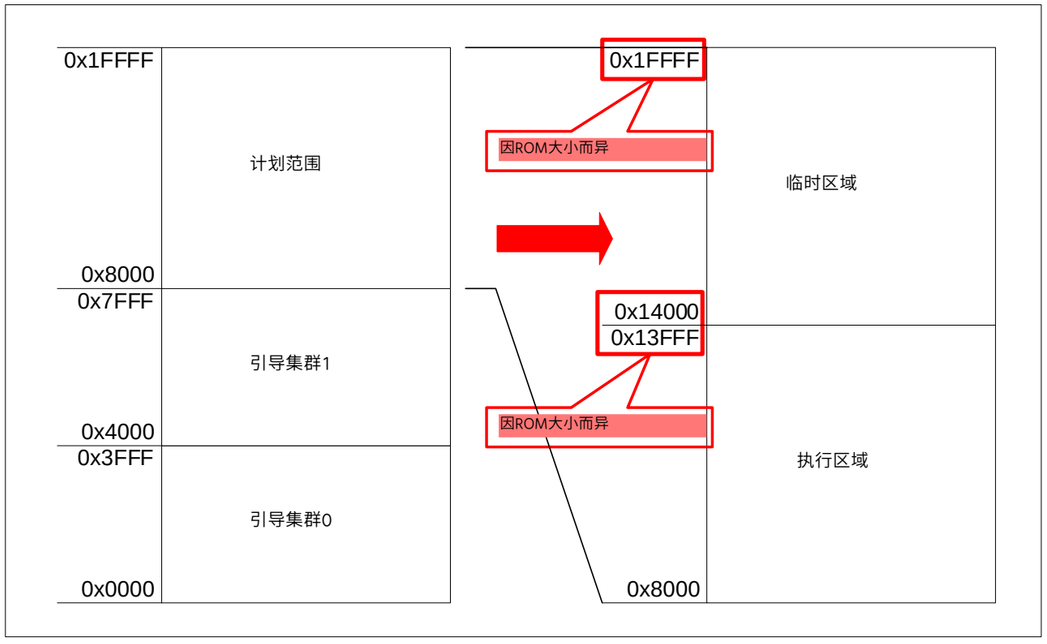
The following table summarizes the features of the code flash memory of the RL78/G23 microcontroller.

Table 1-6 Features of the Code Flash Memory

Item	Description
Minimum unit of erasure	1 block (2,048 bytes)
Minimum unit of writing	1 word (4 bytes)
Minimum unit of verification	1 byte
Security functions	The functions for protection against erasure of blocks, writing to blocks, and reprogramming of the boot area are provided. (All these functions are disabled in the factory settings.)
	The flash shield window is provided, which can protect all area except the specified window range from write and erasure operations during flash memory self-programming only.
	Renesas Flash Driver RL78 Type01 can be used to change the security settings.

Caution: The security functions that are available during flash memory self-programming are only protection against reprogramming of the boot area and the flash shield window.

图1-2代码闪存映射



下表总结了rl78G23微控制器代码闪存的特性。

表1-6代码闪存的特性

Item	Description
擦除的最小单位	1 block (2,048 bytes)
最小写作单位	1 word (4 bytes)
最小核证单位	1 byte
安全功能	提供了防止块擦除、块写入和引导区重新编程的功能。（所有这些功能都在出厂设置中禁用。)提供了闪存屏蔽窗口，它可以保护除指定窗口范围以外的所有区域免受仅在闪存自编程期间的写入和擦除操作。瑞萨闪存驱动器RL78001可用于更改安全设置。

警告：在闪存自编程期间可用的安全功能仅是防止引导区域和闪存屏蔽窗口重新编程的保护。

1.1.3 Flash Memory Self-Programming

The RL78/G23 microcontroller is provided with a library required for performing flash memory self-programming. Flash memory self-programming can be performed by calling functions of Renesas Flash Driver RL78 Type01 from the reprogramming program.

The RL78/G23 microcontroller has a sequencer, which is a circuit that only controls the flash memory. The flash memory self-programming in the RL78/G23 microcontroller uses the sequencer to control the reprogramming of the flash memory. Note that the code flash memory cannot be read while it is being controlled by the sequencer. However, the user program may need to operate while the sequencer is controlling the code flash memory. In such a case, when erasure and write operations are performed and security flags are set for the code flash memory, certain Renesas Flash Driver RL78 Type01 segments or the reprogramming program must be relocated to the RAM. If the user program does not need to run while the sequencer is controlling the code flash memory, Renesas Flash Driver RL78 Type01 and the reprogramming program located on the ROM (code flash memory) can run without relocation.

1.1.4 Boot Swap Function

If the reprogramming of the area in which any of following items are located fails for reasons such as a temporary blackout or reset due an external factor, the data being reprogrammed is corrupted: vector table data, basic program functions, and Renesas Flash Driver RL78 Type01. If data corruption occurs, the user program can no longer be restarted or reloaded by performing a reset. This problem can be prevented by using the boot swap function.

The boot swap function swaps the boot program area (boot cluster 0) with the swap area (boot cluster 1). Before reprogramming starts, the boot swap function writes a new boot program boot cluster 1. The function then swaps boot cluster 0 with boot cluster 1, causing boot cluster 1 to become the boot program area. This ensures that the boot program can normally be started when a reset is performed the next time even if a temporary blackout occurs while the boot program area is being reprogrammed because boot cluster 1 is used to boot the program.

1.1.3闪存自编程

RL78g23微控制器提供了执行闪存自编程所需的库。闪存自编程可通过从重编程程序调用瑞萨闪存驱动器RL78001的函数来执行。

RL78G23微控制器具有定序器，这是一个仅控制闪存的电路。Rl78G23微控制器中的闪存自编程使用序列发生器来控制闪存的重新编程。请注意，代码闪存在序列器控制时无法读取。然而，用户程序可能需要在定序器正在控制代码闪存的同时进行操作。在这种情况下，当执行擦除和写入操作并为代码闪存设置安全标志时，某些瑞萨闪存驱动器RL78001段或重编程程序必须重新定位到RAM中。如果用户程序在定序器控制代码闪存时不需要运行，则瑞萨闪存驱动器RL78001和位于ROM（代码闪存）上的重编程程序可以在不重定位的情况下运行。

1.1.4BootSwap功能

如果由于外部因素导致临时停电或重置等原因，对以下任何项目所在区域的重新编程失败，则重新编程的数据已损坏：矢量表数据、基本程序功能和瑞萨闪存驱动程序RL78Type01。如果发生数据损坏，则不能再通过执行重置来重新启动或重新加载用户程序。使用引导交换功能可以防止此问题。

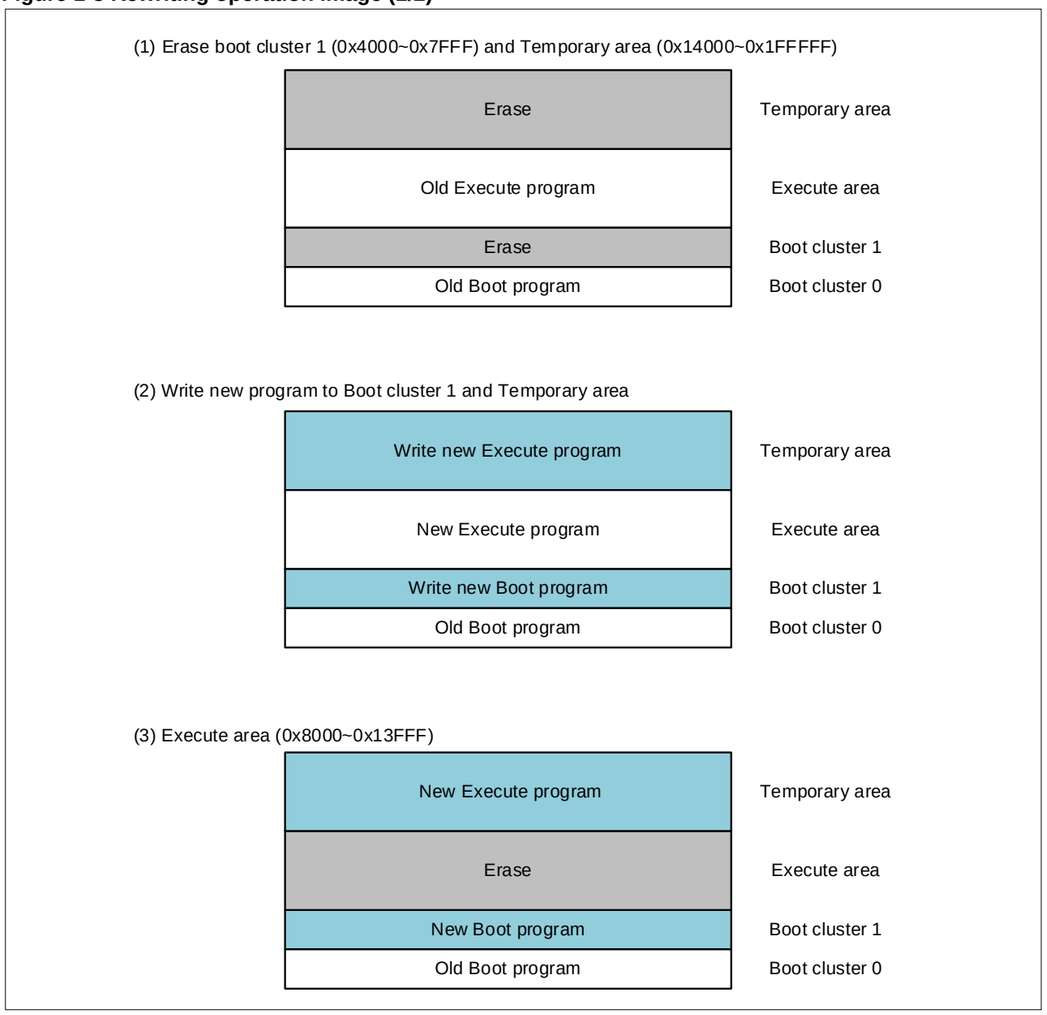
引导交换功能将引导程序区域（引导群集0）与交换区域（引导群集1）交换。
在重新编程开始之前，引导交换功能写入新的引导程序引导集群1。该函数然后将引导群集0与引导群集1交换，导致引导群集1成为引导程序区域。这确保了在下次执行复位时引导程序可以正常启动，即使在引导程序区域正在重新编程时发生临时停电，因为引导群集1用于引导程序。

1.1.5 Updating the Firmware

The following shows an overview of how a program is rewritten by flash memory self-programming. The program that performs flash memory self-programming is deployed in boot cluster 0.

The sample program covered in this application note is designed to reprogram the boot area and program area.

Figure 1-3 Rewriting operation image (1/2)



1.1.5更新固件

下面概述了如何通过闪存自编程重写程序。执行闪存自编程的程序部署在bootcluster0中。

本应用笔记中涵盖的示例程序旨在重新编程引导区和程序区。

图1-3重写操作图像(12)

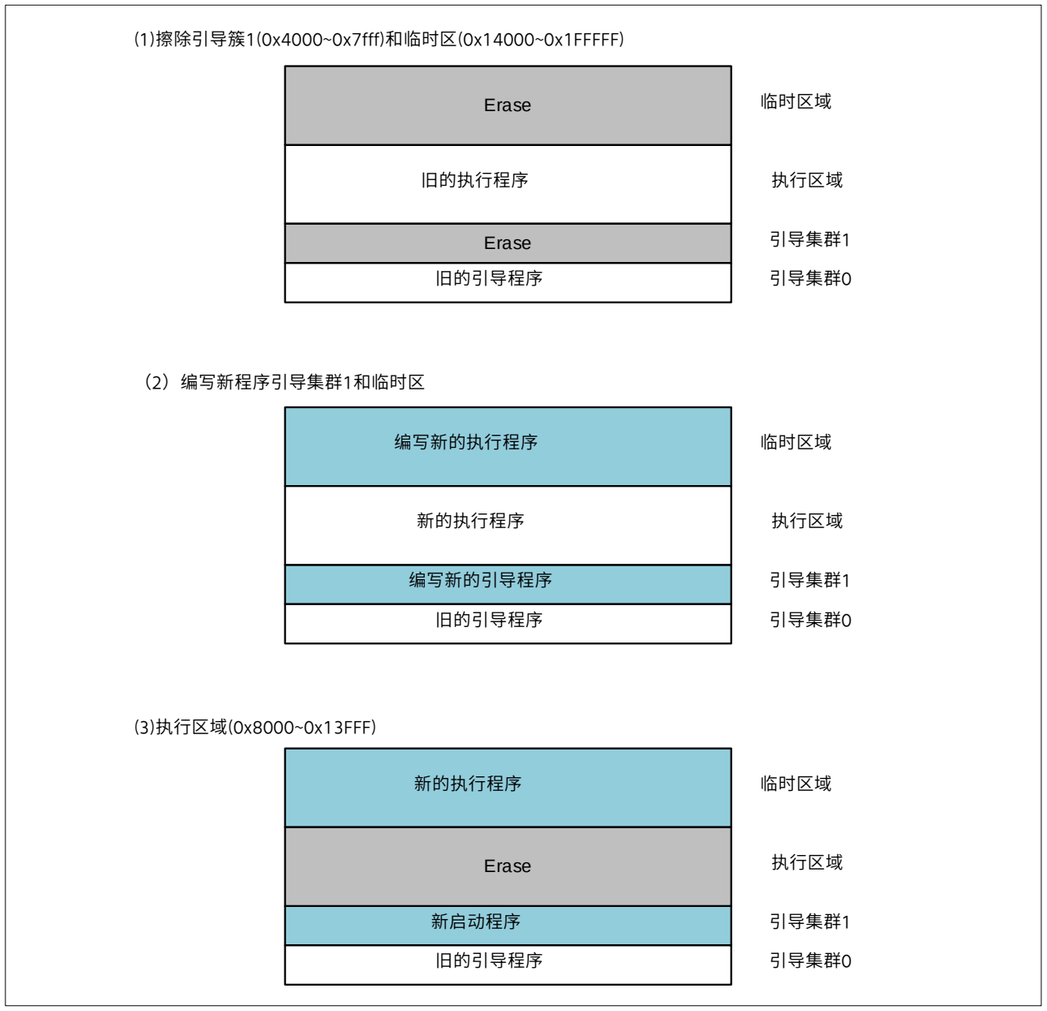


Figure 1-4 Rewriting operation image (2/2)

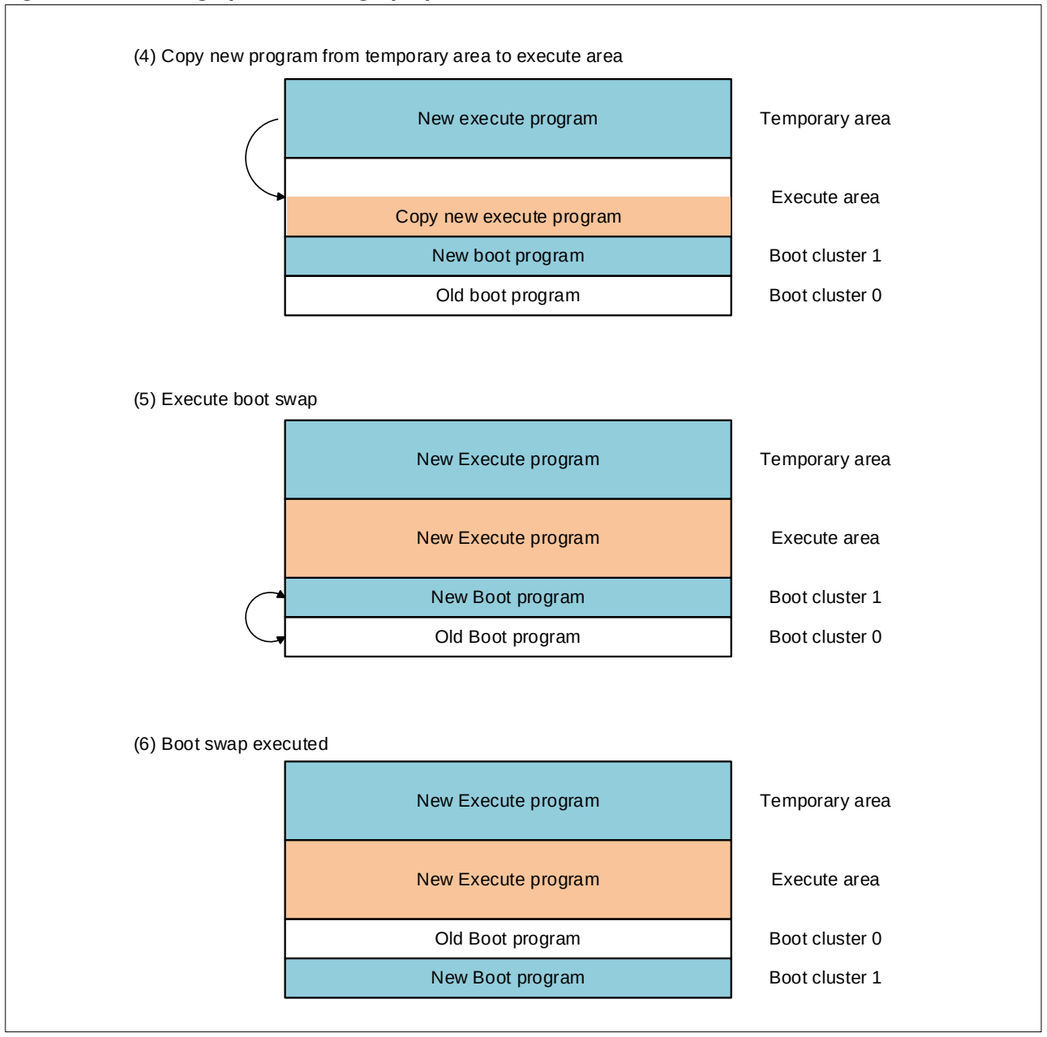
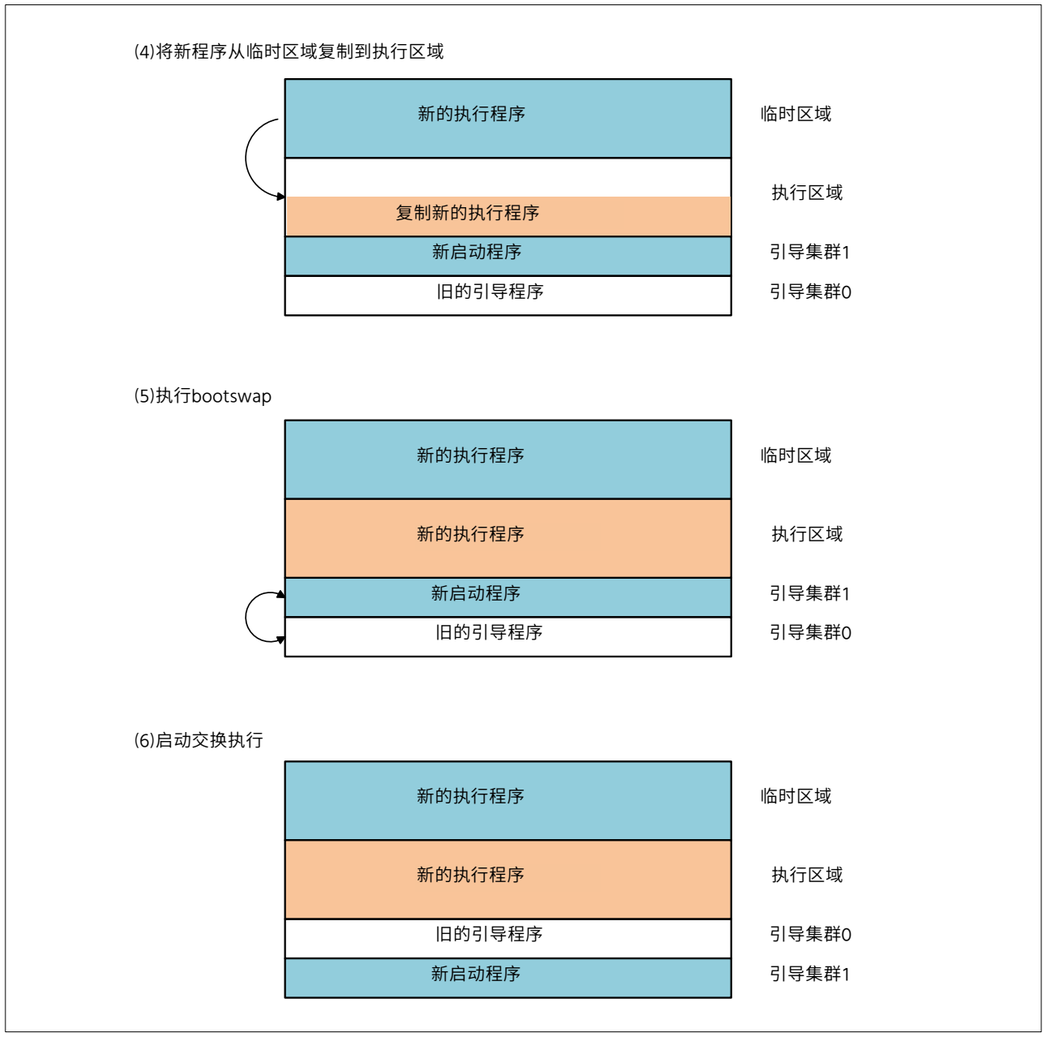


图1-4重写操作图像(2/2)

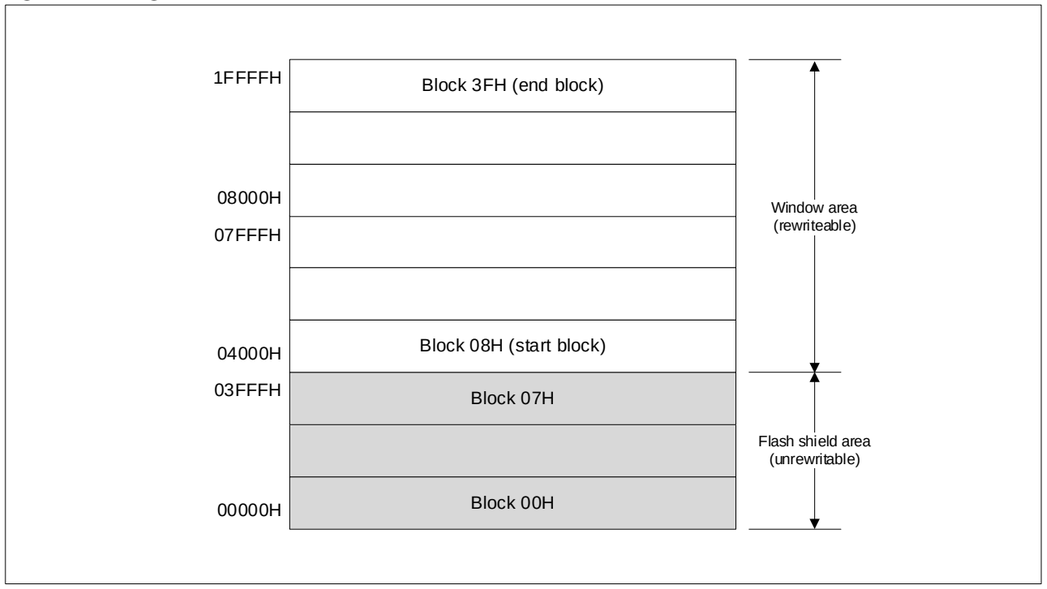


1.1.6 Flash Shield Window

The flash shield window is a security function available during flash memory self-programming. This function protects all areas except the specified window range from the write and erase operations during flash memory self-programming only.

The following figure is an overview of the flash shield window when the start block is 08H and the end block is 1FH.

Figure 1-5 Image of a flush shield window



1.1.7 Obtaining Renesas Flash Driver RL78 Type01

Before you compile the sample program, download the latest version of flash memory self-programming code (Renesas Flash Driver RL78 Type01), and then copy it to the RFD folder.

workspace				Description
r01an6255jj0100-rl78g23-flash				
	\src			
		\RFD		
			\include	Place the downloaded Renesas Flash Driver RL78 Type01
			\source	
			\userown	

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

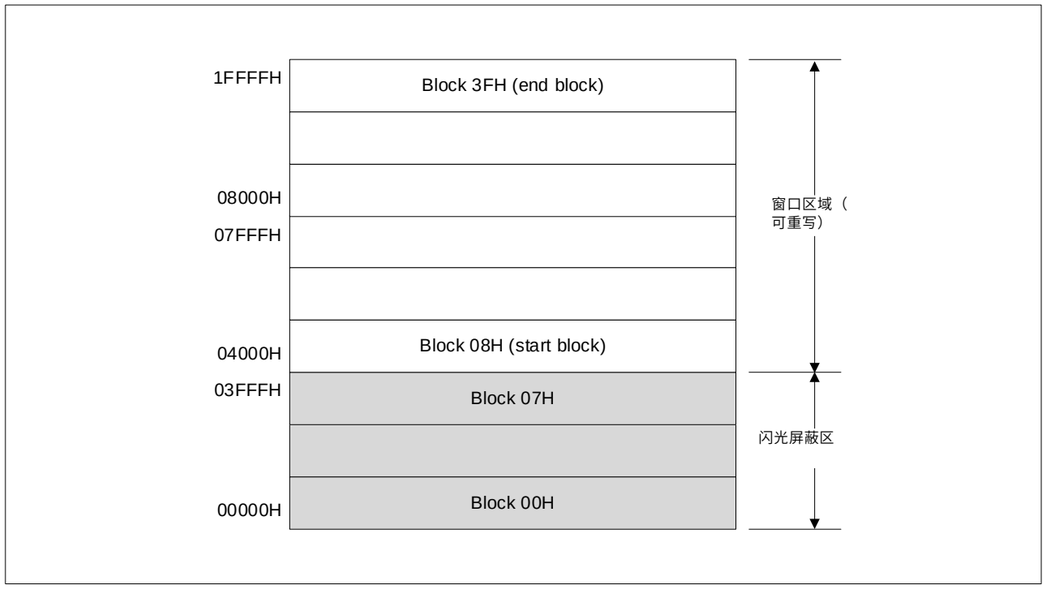
https://www.renesas.com/jp/ja/document/scd/renesas-flash-driver-rl78-type-01-rl78g23

1.1.6闪光屏蔽窗口

Flashshield窗口是闪存自编程期间可用的安全功能。此功能仅在闪存自编程期间保护除指定窗口范围之外的所有区域免受写入和擦除操作的影响。

下图是开始块为08h，结束块为1fh时flashshield窗口的概述。

图1-5冲洗屏蔽窗口的图像



1.1.7获取瑞萨闪存驱动程序RL78Type01在编译示例程序之前，下载最新版本的闪存自编程代码（瑞萨闪存驱动程序RL78Type01），然后将其复制到RFD文件夹。

workspace				Description
r01an6255jj0100-rl78g23-flash				
	\src			
		\RFD		
			\include	放置下载的瑞萨闪存驱动程序
			\source	
			\userown	

您可以从以下网址获取瑞萨闪存驱动程序RL78Type01：

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

1.2 Overview of Operation

(1) Perform initial setup for pins.

・Set the P03, P02, P43, P42, P77, P41, P31, and P76 pins to output mode.

(2) Perform initial setup for the serial array unit.

- Use the UART0 serial array unit (set TXD0 for P12 and RXD0 for P11).
- Set CK00 for the operation clock and fCLK/2 for the clock source.
- Set the clock source for the transfer mode settings.
- Set 8 bits for the data bit length settings.
- Set LSB for the data transfer direction settings.
- Set "no parity" for the parity settings.
- Set 1 bit for the stop bit length settings.
- Set "standard" for the send data level settings.
- Set 115,200 bps for the baud rate settings.

(3) Use command communication to reprogram the data in boot cluster 1 and the program area, and then perform boot swapping.

1.2.1 Communication Specifications

The sample program covered in this application note receives the reprogramming data via UART and performs flash memory self-programming. The sample program then receives the START, WRITE_BOOT1, WRITE_TEMP, or END command. The sample program then performs the processing according to the received command. If the processing terminates normally, the sample program returns "01H" (normal) to the command sender. If the processing terminates abnormally, the sample program turns on the LED that indicates abnormal termination (without returning a response) and performs no subsequent processing. The following shows the UART communication settings and the specifications of the commands.

Table 1-7 UART Communication Settings

Data bit length (bits)	8
Data transfer direction	LSB first
Parity setting	No parity
Transfer rate (bps)	115,200

1.2操作概述

(1)执行引脚的初始设置。

–将P03、P02、P43、P42、P77、P41、P31和P76引脚设置为输出模式。

(2)对串行阵列单元进行初始设置。

- 使用UART0串行阵列单元（p12设置TXD0，P11设置RXD0）。
- 为操作时钟设置CK00，为时钟源设置fCLK2。
- 为传输模式设置设置时钟源。
- 为数据位长度设置设置8位。
- 为数据传输方向设置LSB。
- 为奇偶校验设置设置"无奇偶校验"。
- 为停止位长度设置设置1位。
- 为发送数据级别设置设置"标准"。
- 为波特率设置设置115 200bps。

(3)使用命令通信对引导集群1和程序区中的数据进行重新编程，然后执行引导交换。

1.2.1通信规范

本应用笔记中涉及的示例程序通过UART接收重新编程数据并执行闪存自编程。然后，示例程序接收START、WRITE_BOOT1、WRITE_TEMP或END命令。示例程序然后根据接收到的命令执行处理。如果处理正常终止，则示例程序向命令发送者返回"01h"(正常)。如果处理异常终止，则示例程序打开指示异常终止的LED(不返回响应)并且不执行后续处理。下面显示了UART通信设置和命令的规格。

表1–7UART通信设置

数据位长度(bits)	8
数据传输方向	
奇偶校验设置	没有奇偶校验
传输速率(bps)	

1.2.2 START Command

When the sample program receives the START command, it performs initial setup for flash memory self-programming and erases the Temporary area in boot cluster 1. If the processing terminates normally, the sample program returns "01H" (normal). If the processing terminates abnormally, the sample program turns on the LED that indicates abnormal termination (without returning a response) and performs no subsequent processing.

START code (01H)	Data length (0002H)	Command (02H)	Data (empty)	Checksum (1 byte)

1.2.3 WRITE_BOOT1 Command

When the sample program receives the WRITE_BOOT1 command, it writes the received data to the boot cluster 1 area (4000H to 7FFFH) while verifying the written data for each 256 bytes. If the processing terminates normally, the sample program increments the write destination address by 256 bytes and returns "01H" (normal) to the command sender. If the processing terminates abnormally, the sample program turns on the LED that indicates abnormal termination (without returning a response) and performs no subsequent processing.

START code (01H)	Data length (0102H)	Command (03H)	Data (256 bytes)	Checksum (1 byte)

1.2.4 WRITE_TEMP Command

When the sample program receives the WRITE_TEMP command, it writes the received data to the Temporary area while verifying the written data for each 256 bytes. If the processing terminates normally, the sample program increments the write destination address by 256 bytes and returns "01H" (normal) to the command sender. If the processing terminates abnormally, the sample program turns on the LED that indicates abnormal termination (without returning a response) and performs no subsequent processing.

(The write destination address of the Temporary area differs depending on the product used.)

START code (01H)	Data length (0102H)	Command (04H)	Data (256 bytes)	Checksum (1 byte)

1.2.5 END Command

When the sample program receives the END command, it erases the Execute area. If erasure terminates normally, the sample program copies data from the Temporary area to the Execute area. If copy terminates normally, the sample program returns "01H" (normal) to the command sender. The sample program then reverses the boot flag to cause a reset to occur and performs boot swapping.

START code (01H)	Data length (0002H)	Command (05H)	Data (empty)	Checksum (1 byte)

1.2.6 Checksum Calculation Method

For checksum calculation, the 32-bit addition method is used. This method uses as a checksum the last 8 bits of the result of adding a 1-byte value from address 00000000H for the command or data.

1.2.2启动命令

当示例程序接收到启动命令时，它执行闪存自编程的初始设置并擦除引导群集1中的临时区域。如果处理正常终止，则示例程序返回"01h"（正常）。如果处理异常终止，则示例程序打开指示异常终止的LED(不返回响应)并且不执行后续处理。

启动代码	数据长度	Command (02H)	Data (empty)	Checksum (1 byte)

1.2.3WRITE_BOOT1命令当示例程序接收到WRITE_BOOT1命令时，它将接收到的数据写入引导集群1区域（4000h到7FFFH），同时验证每个256字节的写入数据。如果处理正常终止，则示例程序将写入目标地址递增256字节，并向命令发送方返回"01h"（正常）。如果处理异常终止，则示例程序打开指示异常终止的LED(不返回响应)并且不执行后续处理。

启动代码	数据长度	Command (03H)	Data (256 bytes)	Checksum (1 byte)

1.2.4 WRITE_TEMP Command

当示例程序接收到WRITE_TEMP命令时，它将接收到的数据写入
临时区同时验证每256字节的写入数据。如果处理正常终止，则示例程序将写入目标地址递增256字节，并向命令发送方返回"01h"（正常）。如果处理异常终止，则示例程序打开指示异常终止的LED(不返回响应)并且不执行后续处理。

（临时区域的写入目的地地址因使用的产品而异。）

启动代码	数据长度	Command (04H)	Data (256 bytes)	Checksum (1 byte)

1.2.5结束命令

当示例程序接收到结束命令时，它擦除执行区域。如果擦除正常终止，则示例程序将数据从临时区域复制到执行区域。如果复制正常终止，则示例程序向命令发送者返回"01h"（正常）。然后，示例程序反转引导标志以导致重置发生并执行引导交换。

启动代码	数据长度	Command (05H)	Data (empty)	Checksum (1 byte)

1.2.6校验和计算方法

对于校验和计算，使用32位加法方法。此方法使用从地址00000000h为命令或数据添加1字节值的结果的最后8位作为校验和。

1.2.7 Operation of the Sample Program

The following shows the operation of this sample program:

(1) Set up the input and output ports.

(2) Perform initial setup for SAU0 channel 0.

(3) Wait for data to be sent from the command sender.

(4) Upon receiving the START command, perform initial setup for self-programming.

(5) Set the P02 pin for high-level output to turn on LED2, which indicates that the START command was received.

(6) Call the r_CF_EraseBlock function to erase boot cluster 1.

(7) Call the r_CF_EraseBlock function to erase the data in the Temporary area.

(8) Send "01H" (normal) to the command sender.

(9) Set the P02 pin for low-level output to turn off LED2, which indicates that the START command was received.

(10) Receive the WRITE_BOOT1 command (03H) and write data (256 bytes).

(11) Set the P43 pin for high-level output to turn on LED3, which indicates that the WRITE_BOOT1 command was received.

(12) Call the r_CF_WriteData function to write the received data to the write destination address (local variable for writing to boot cluster 1). The initial value of the local variable for writing to boot cluster 1 is the start address of boot cluster 1.

(13) Call the r_CF_VerifyData function to verify the written data against the received data.

(14) Add a 256-byte checksum to the write destination address (local variable for writing to boot cluster 1).

(15) Send "01H" (normal) to the command sender.

(16) Set the P43 pin for low-level output to turn off LED3, which indicates that the WRITE_BOOT1 command was received.

(17) Repeat steps (11) to (17) until receiving the WRITE_TEMP command (04H).

(18) Receive the WRITE_TEMP command (04H) and write data (256 bytes).

(19) Set the P42 pin for high-level output to turn on LED4, which indicates that the WRITE_BOOT1 command was received.

(20) Call the r_CF_WriteData function to write the received data to the write destination address (local variable for writing to the Temporary area). The initial value of the local variable for writing to the Temporary area is the start address of the Temporary area.

(21) Call the r_CF_VerifyData function to verify the written data against the received data.

(22) Add a 256-byte checksum to the write destination address (local variable for writing to the Temporary area).

(23) Send "01H" (normal) to the command sender.

(24) Set the P42 pin for low-level output to turn off LED4, which indicates that the WRITE_TEMP command was received.

(25) Repeat steps (19) to (25) until receiving the END command (05H).

(26) Perform the following processing if receiving the END command:

(27) Set the P77 pin for high-level output to turn on LED5, which indicates that the END command was received.

(28) Call the r_CF_EraseBlock function to erase the data in the Execute area.

(29) Set the P41 pin for high-level output to turn on LED6, which indicates that copy to the Temporary area is in progress.

1.2.7示例程序的操作以下显示此示例程序的操作:

(1)设置输入输出端口。

(2)执行SAU0通道0的初始设置。

(3)等待数据从命令发送方发送。

(4)在接收到启动命令时，执行自编程的初始设置。

(5)将高电平输出的P02引脚置为导通LED2，表示接收到启动命令。

(6)调用r_cf_eraseblock函数擦除引导集群1。

(7)调用r_CF_EraseBlock函数擦除临时区域中的数据。

(8)向命令发送者发送"01h"(正常)。

(9)设置低电平输出的P02引脚关闭LED2，表示收到启动命令。

(10)接收WRITE_BOOT1命令(03H)并写入数据(256字节)。

(11)设置高电平输出的P43引脚开启LED3，表示收到WRITE_BOOT1命令。

(12)调用r_CF_WriteData函数将接收到的数据写入写入目的地址(local
变量，用于写入引导集群1)。用于写入引导集群1的局部变量的初始值是引导集群1的开始地址。

(13)调用r_CF_VerifyData函数对接收到的数据验证写入的数据。

(14)将256字节的校验和添加到写入目标地址（用于写入引导集群1的本地变量）。

(15)向命令发送者发送"01h"(正常)。

(16)设置低电平输出的P43引脚关闭LED3，表示收到WRITE_BOOT1命令。

(17)重复步骤(11)至(17)，直到接收到WRITE_TEMP命令(04H)。

(18)接收WRITE_TEMP命令(04H)并写入数据(256字节)。

(19)设置高电平输出的P42引脚开启LED4，表示收到WRITE_BOOT1命令。

(20)调用r_CF_WriteData函数，将接收到的数据写入写入目标地址（用于写入临时区域的局部变量）。
用于写入临时区域的局部变量的初始值是临时区域的起始地址。

(21)调用r_CF_VerifyData函数，对照接收到的数据验证写入的数据。

(22)向写入目的地址(用于写入临时区的局部变量)添加256字节的校验和。

(23)向命令发送者发送"01h"(正常)。

(24)设置低电平输出的P42引脚关闭LED4，表示收到WRITE_TEMP命令。

(25)重复步骤(19)至(25)直到接收到结束命令(05H)。

(26)如果接收到结束命令，则执行以下处理:

(27)设置高电平输出的P77引脚导通LED5，表示接收到结束命令。

(28)调用r_CF_EraseBlock函数擦除执行区域中的数据。

(29)将高电平输出的P41引脚设置为打开LED6，这表明复制到临时区域正在进行中。

(30) Call the r_temp_copy function to copy data from the Temporary area to the Execute area.*

(31) Set the P41 pin for low-level output to turn off LED6, which indicates that copy to the Temporary area is in progress.

(32) Send "01H" (normal) to the command sender.

(33) Call the r_RequestBootSwap function to reverse the value of the boot flag so that boot clusters 0 and 1 are swapped when a reset occurs. Cause an internal reset to occur.

* If a reset occurs (due to a temporary blackout, for example) while data is being copied from the Temporary area to the Execute area, the r_temp_copy function is called again. The r_RequestBootSwap function is called after the copy is complete.

Caution: If the sample program receives the END command (05H) in steps (10) to (17), the sample program copies data from the Temporary area to the Execute area unless there is an error. Then, the sample program sends "01H" (normal), calls the r_RequestBootSwap function, and performs boot swapping. If the sample program receives the END command (05H) while boot cluster 1 is being reprogrammed, the sample program performs boot swapping before the reprogramming ends normally. In this case, the sample program can no longer start after the boot area is swapped.

Caution: If flash memory self-programming does not end normally, the sample program only turns on LED6 and performs no subsequent processing.

(30)调用r_temp_copy函数将数据从临时区域复制到执行区域。*

(31)将低电平输出的P41引脚设置为关闭LED6，这表明复制到临时区域正在进行中。

(32)向命令发送者发送"01h"(正常)。

(33)调用r_RequestBootSwap函数反转引导标志的值，以便在发生复位时交换引导簇0和1。导致发生内部复位。

*如果在从
临时区到执行区，再次调用r_temp_copy函数。复制完成后调用r_RequestBootSwap函数。

注意:如果示例程序在步骤(10)至(17)中接收到结束命令(05H)，则示例程序将数据从临时区域复制到执行区域，除非出现错误。然后，示例程序发送"01h"（正常），调用r_RequestBootSwap函数，并执行引导交换。如果示例程序在重新编程引导群集1时收到结束命令(05H)，则示例程序在重新编程正常结束之前执行引导交换。在这种情况下，在引导区被交换后，示例程序不能再启动。

注意：如果闪存自编程没有正常结束，示例程序只打开LED6，不执行后续处理。

1.2.8 Copy Flag

The sample program covered in this application note uses a 4-byte area at the end of the Execute area as the copy flag section in which to set a copy flag.

If a program is normally written, this copy flag is set to AAAA5555H. The copy flag is initialized when the Execute area is erased immediately before data is copied from the Temporary area to the Execute area. If a reset occurs (due to a temporary blackout) while data is being written, the copy flag is set to a value other than AAAA5555H because the write processing does not terminate normally.

When the sample program starts, it checks the copy flag. If the value of the copy flag is not AAAA5555H, the sample program writes data and then performs swapping.

The following table shows the start and end addresses of the Execute area and the address of the copy flag section according to the ROM size.

Table 1-8 Location of the Copy Flag Section According to the ROM Size

ROM Size	Execute Area	Address of the Copy Flag Section
96KB	08000H to FFFFH	FFFCH
128KB	08000H to 13FFFH	13FFCH
192KB	08000H to 1BFFFH	1BFFCH
256KB	08000H to 23FFFH	23FFCH
384KB	08000H to 33FFFH	33FFCH
512KB	08000H to 43FFFH	43FFCH
768KB	08000H to 63FFFH	63FFCH

1.2.8复制标志本应用笔记中涉及的示例程序使用执行区域末尾的4字节区域作为设置复制标志的复制标志部分。

如果程序正常写入，则此复制标志设置为AAAA5555H。复制标志在
执行区域在数据从临时区域复制到执行区域之前立即被擦除。如果在写入数据时发生复位（由于临时停电），则复制标志被设置为AAAA5555H以外的值，因为写入处理不会正常终止。

当示例程序启动时，它会检查复制标志。如果复制标志的值不是AAAA5555H，则示例程序写入数据，然后执行交换。

下表根据ROM大小示出了执行区域的开始和结束地址以及复制标志部分的地址。

表1-8根据ROM大小复制标志部分的位置

ROM大小	执行区域	复制标志部分的地址
96KB	08000H to FFFFH	FFFCH
128KB	08000H to 13FFFH	13FFCH
192KB	08000H to 1BFFFH	1BFFCH
256KB	08000H to 23FFFH	23FFCH
384KB	08000H to 33FFFH	33FFCH
512KB	08000H to 43FFFH	43FFCH
768KB	08000H to 63FFFH	63FFCH

2. Operation Confirmation Conditions

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

Table 2-1 Operation Confirmation Conditions

Item	Description
MCU used	RL78/G23 (R7F100GLG)
Board used	[Project for 128KB products] RL78/G23-64p Fast Prototyping Board (RTK7RLG230CLG000BJ) [Project for 768KB products] RL78/G23-128p Fast Prototyping Board (RTK7RLG230CSN000BJ)
Operation frequency	High-speed on-chip oscillator clock (fIH): 32MHz
Operating voltage	3.3V (can be operated at 3.1V to 3.5V) LVD operation (V_{LVD}): Reset mode At rising edge TYP. 1.90 V (1.84 V to 1.95 V) At falling edge TYP. 1.86 V (1.80 V to 1.91 V)
Integrated development environment (CS+)	CS+ for CC V8.06.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.10.00 from Renesas Electronics Corp.
Integrated development environment (e2studio)	e2studio V2021-10 from Renesas Electronics Corp.
C compiler (e2studio)	CC-RL V1.10.00 from Renesas Electronics Corp.
Integrated development environment (IAR)	IAR Embedded Workbench for Renesas RL78 V4.21.1 from IAR Systems Corp.
C compiler (IAR)	IAR C/C++ Compiler for Renesas RL78 V 4.21.1.2409 from IAR Systems Corp.

2.操作确认条件

本应用笔记提供的示例代码的操作已在
以下条件。

表2-1操作确认条件

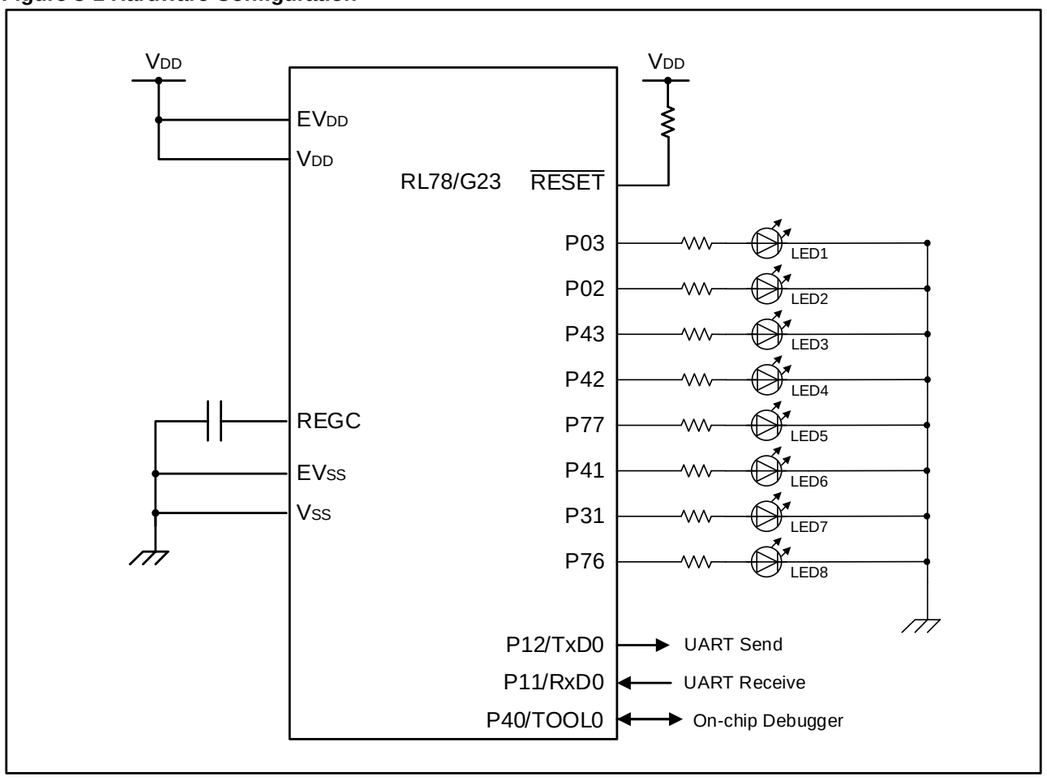
Item	Description
MCU used	RL78/G23 (R7F100GLG)
使用的板	[Project for 128KB products] RL78G23-64p快速原型板(RTK7RLG230CLG000BJ)[768KB产品项目] RL78G23-128P快速原型板(RTK7RLG230CSN000BJ)
操作频率	高速片上振荡器时钟(fIH)：32MHz
工作电压	3.3V（可在3.1v至3.5V下操作）LVD操作 （VLVD）：复位模式 在上升沿TYP。1.90V(1.84V至1.95V)在下降沿典 型值。1.86V(1.80v至1.91V)
集成开发环境(CS+)	瑞萨电子公司的CS+CCV8.06.00。
C compiler (CS+)	RENESASElectronicsCorp.的CC-RLV1.10.00。
集成开发环境(e2studio)	来自瑞萨电子公司的e2studioV2021-10。
C compiler (e2studio)	RENESASElectronicsCorp.的CC-RLV1.10.00。
集成开发环境(iar)	用于瑞萨Rl78V4.21.1的Iar嵌入式工作台
C compiler (IAR)	用于瑞萨Rl78V4.21.1.2409的IarCC++编译器来自IarSystems

3. Hardware Descriptions

3.1 Example of Hardware Configuration

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

Figure 3-1 Hardware Configuration



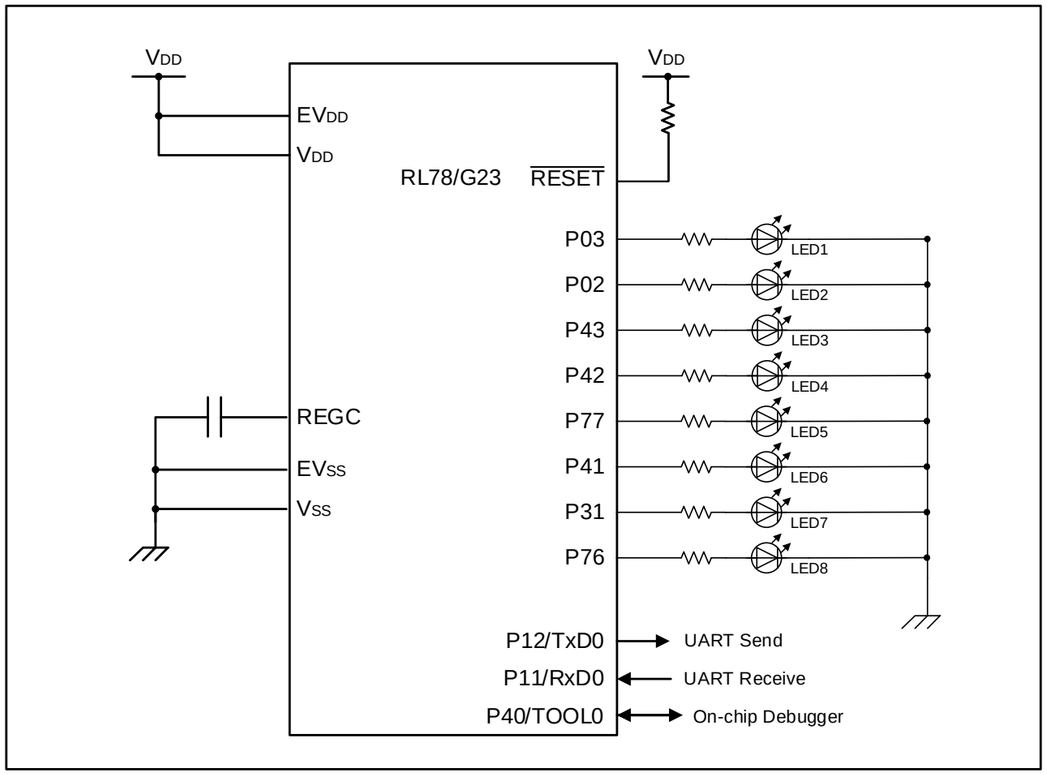
Cautions:1. The purpose of this circuit is only to provide the connection outline and the circuit is simplified accordingly. When designing and implementing an actual circuit, provide proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to V_{DD} or V_{SS} via a resistor).
2. Connect any pins whose name begins with EV_{SS} to V_{SS} and any pins whose name begins with EV_{DD} to V_{DD}, respectively.
3. V_{DD} must be held at not lower than the reset release voltage (V_{LVD}) that is specified as LVD.

3.硬件描述

3.1硬件配置示例

图3-1显示了应用笔记中使用的硬件配置示例。

图3-1硬件配置



注意事项：1。这种电路的目的仅仅是提供连接轮廓并且电路相应地被简化。在设计和实现实际电路时，提供适当的引脚处理并确保满足硬件的电气规格（通过电阻将仅输入端口单独连接到VDD或VSS）。

2.将名称以EVSS开头的任何引脚分别连接到VSS，并将名称以EVDD开头的任何引脚分别连接到VDD。

3.VDD必须保持在不低于被指定为LVD的复位释放电压(VLVD)。

3.2 List of Pins to be Used

Table 3.1 lists the pins to be used and their functions.

Table 3-1 Pins to be Used and their Functions

Pin	Input/Output	Description
P12//TxD0	Output	UART serial data transmit pin
P11/ RxD0	Input	UART serial data receive pin
P03, P02, P43, P42, P77, P41, P31, P76	Output	LED1-LED8 control pins

Caution: In this application note, only the used pins are processed. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.

4. Software Explanation

4.1 Setting of Option Byte

Table 4-1 shows the option byte settings.

Table 4-1 Option Byte Settings

Address	Setting Value	Description
000C0H/040C0H	11101111B	Disables the watchdog timer. (Counting stopped after reset)
000C1H/040C1H	11111110B	LVD operation (V_{LVD}): Reset mode At rising edge TYP. 1.90 V (1.84 V to 1.95 V) At falling edge TYP. 1.86 V (1.80 V to 1.91 V)
000C2H/040C2H	11101000B	HS mode, High-speed on-chip oscillator clock (f_{IH}): 32 MHz
000C3H/040C3H	10000101B	Enables on-chip debugging

The option bytes of the RL78/G23 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 040C0H to 040C3H because the bytes in 000C0H to 000C3H are swapped with the bytes in 040C0H to 040C3H.

3.2要使用的引脚列表

表3.1列出了要使用的引脚及其功能。

表3-1要使用的引脚及其功能

Pin	Input/Output	Description
P12//TxD0	Output	Uart串行数据传输引脚
P11/ RxD0	Input	Uart串行数据接收引脚
P03, P02, P43, P42, P77, P41, P31, P76	Output	LED1-LED8控制引脚

Caution: 在本应用笔记中，只处理使用过的引脚。在实际设计电路时，请确保设计包括足够的引脚处理并满足电气特性要求。

4.软件说明

4.1选项字节的设置

表4-1显示了选项字节设置。

表4-1选项字节设置

Address	设置值	Description
000C0H/040C0H	11101111B	禁用看门狗定时器。（复位后停止计数）
000C1H/040C1H	11111110B	LVD操作(VLVD)：复位模式 在上升沿TYP。1.90V(1.84V至1.95V)在下降沿典型值。1.86V(1.80v至1.91V)
000C2H/040C2H	11101000B	高速片上振荡器时钟(fIH)：32MHz
000C3H/040C3H	10000101B	启用片上调试

Rl78G23的选项字节包括用户选项字节(000c0h至000c2h)和片上调试选项字节(000c3h)。

选项字节被自动引用，并在开机时配置指定的设置或复位被释放。当使用引导交换功能进行自编程时，必须设置在000C0H到000C3H以及在040c0h到040c3h中设置的相同值，因为000c0h到000c3h中的字节与040c0h到040c3h中的字节交换。

4.2 Setting Up the Startup Routine

4.2.1 Defining the Stack Area Section (.stack_bss)

Define the stack area section (.stack_bss).

In the startup routine configuration file (cstart.asm), change the settings as follows:

```
;$IF (__RENESAS_VERSION__ < 0x01010000)
;-----------------------------------------------------------------------
;
;-----------------------------------------------------------------------
; !!! [CAUTION] !!!
; Set up stack size suitable for a project.
.SECTION .stack_bss, BSS
_stackend:
.DS     0x800

_stacktop:
;$ENDIF


;-----------------------------------------------------
; setting the stack pointer
;-----------------------------------------------------
$IF (__RENESAS_VERSION__ >= 0x01010000)
;MOVW    SP,#LOWW(__STACK_ADDR_START)

;$ELSE   ; for CC-RL V1.00
MOVW    SP,#LOWW(_stacktop)
$ENDIF

;-----------------------------------------------------
; initializing stack area
;-----------------------------------------------------
$IF (__RENESAS_VERSION__ >= 0x01010000)
;MOVW    AX,#LOWW(__STACK_ADDR_END)

;$ELSE   ; for CC-RL V1.00

MOVW    AX,#LOWW(_stackend)
$ENDIF
CALL    !!_stkinit
```

Comment out the line by prefixing a semicolon (;).

Specify any stack size of your choice by using a hexadecimal number.

Comment out the line by prefixing a semicolon (;).

Comment out the line by prefixing a semicolon (;).

Comment out the line by prefixing a semicolon (;).

Comment out the line by prefixing a semicolon (;).

4.2设置启动例程

4.2.1定义堆栈区域部分 （。stack_bss)定义堆栈区段(。stack_bss)。

在启动例程配置文件（cstart.asm），更改设置如下:

```
;$IF (__RENESAS_VERSION__ < 0x01010000)
;-----------------------------------------------------------------------
;
;-----------------------------------------------------------------------

;设置适合项目的堆栈大小。

_stackend:
.DS     0x800

_stacktop:
;$ENDIF


                    ----------
;设置堆栈指针
                    ----------
$IF (__RENESAS_VERSION__ >= 0x01010000)
;MOVW    SP,#LOWW(__STACK_ADDR_START)

;$ELSE   ; for CC-RL V1.00
MOVW    SP,#LOWW(_stacktop)
$ENDIF

                    --------------------
;初始化堆栈区域
                    --------------------
$IF (__RENESAS_VERSION__ >= 0x01010000)
;MOVW    AX,#LOWW(__STACK_ADDR_END)

;$ELSE   ; for CC-RL V1.00

MOVW    AX,#LOWW(_stackend)
$ENDIF
CALL    !!_stkinit
```

通过在该行前加上分号(;)来注释该行。

使用十六进制数指定您选择的任何堆栈大小。

通过在该行前加上分号(;)来注释该行。

通过在该行前加上分号(;)来注释该行。

通过在该行前加上分号(;)来注释该行。

通过在该行前加上分号(;)来注释该行。

4.2.2 Deploying the Reprogramming Program in the RAM Area

Confirm the programs that are used to reprogram the firmware and deploy them to the RAM area.

Table4-2 shows the sections where the programs used to reprogram the firmware exist and the sections in which the programs are to be deployed.

Table4-2 Section Information

Section	Destination Section	Description
RFD_CMN_f	RFD_CMN_fR	Program section for the API functions that control the common flash memory
RFD_CF_f	RFD_CF_fR	Program section for the API functions that control the code flash memory
RFD_EX_f	RFD_EX_fR	Program section for the API functions that control the extra area
SMP_CMN_f	SMP_CMN_fR	Program section for the sample functions that control the common flash memory
SMP_CF_f	SMP_CF_fR	Program section for the sample functions that control the code flash memory

To deploy the preceding sections in the RAM area, you must add the necessary processing to the cstart.asm file.

In the cstart.asm file, locate the following lines, and then add the necessary processing after these lines:

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

The following are the details to be added.

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

Note 1. For ***section-name***, specify the name of the section to be deployed.

Note 2. Add the preceding set of entries for each section to be deployed.

Note 3. For m, specify any numeric value of your choice. Make sure that you specify a different value for each section.

4.2.2在RAM区域部署重新编程程序确认用于重新编程固件的程序并将其部署到RAM区域。

表4-2显示了用于重新编程固件的程序存在的部分和程序要部署的部分。

表4-2部分资料

Section	目的地组	
RFD_CMN_f		控制通用闪存的API函数的程序部分
RFD_CF_f	RFD_CF_fR	控制代码闪存的API函数的程序部分
RFD_EX_f	RFD_EX_fR	控制额外区域的API函数的程序部分
SMP_CMN_f	SMP_CMN_fR	用于控制公共闪存的示例函数的程序部分
SMP_CF_f	SMP_CF_fR	控制代码闪存的示例函数的程序部分

要在RAM区域中部署前面的部分，必须向cstart添加必要的处理。asm文件。

在cstart中。asm文件，找到以下行，然后在这些行后面添加必要的处理:

```
;---------------------------------------------------
---------;ROM数据拷贝;-------------------
-----------------------------
```

以下是要添加的详细信息。

```
;复制。texttoRAM(section-name)MOVC #HIGH
W(STARTOF(section-name))
                                                  ))
                                (STARTOF(destination-section-name))
BR$。Lm2_TEXT.Lm1_
TEXT:MOVA C

MOVES，A
                    [HL]
                   A
INCWDEINCW
HLCLRW斧头

                   HL
   SKNZ
   INC
.Lm2_TEXT:
   MOVW    AX,HL
   CMPW    AX,#LOWW(STARTOF(section-name) + SIZEOF(section-name))
   BNZ     $.Lm1_TEXT
```

注1。对于section-name，指定要部署的节的名称。

注2。为要部署的每个部分添加前一组条目。

注3。对于m，指定您选择的任何数值。确保为每个部分指定不同的值。

4.3 Setting the ROM Size Specification Constant

Conditional compilation allows this sample program to support several ROM sizes for the RL78/G23 microcontroller.

The following table lists the constants that correspond to the supported ROM sizes. In the r_cg_userdefine.h file, these constants are commented out. Enable the constant for the installed ROM by uncommenting it.

Table4-3 Constants for the Supported ROM Sizes

Constant Name	Supported ROM Size
ROM_SIZE_96KB	96-KB product
ROM_SIZE_128KB	128-KB product
ROM_SIZE_192KB	192-KB product
ROM_SIZE_256KB	256-KB product
ROM_SIZE_384KB	384-KB product
ROM_SIZE_512KB	512-KB product
ROM_SIZE_768KB	768-KB product

4.4 On-chip Debug Security ID

The RL78/G23 microcontroller provides the on-chip debug security ID area at addresses 000C4H to 000CDH in the flash memory so that the memory content is not read by third parties.

If boot swapping is performed during self-programming, the area at addresses 000C4H to 000CDH and the area at addresses 010C4H to 010CDH are swapped. Therefore, the same value that is set in the area at 000C4H to 000CDH must also be set in the area at 040C4H to 040CDH.

4.3设置ROM尺寸规格常数

条件编译允许此示例程序支持rl78G23微控制器的多种ROM尺寸。

下表列出了与支持的ROM大小相对应的常量。在r_cg_userdefine。h文件，这些常量被注释掉。通过取消注释来启用已安装ROM的常量。

表4-3支持ROM大小的常量

常量名称	支持的ROM大小
ROM_SIZE_96KB	96-KB product
ROM_SIZE_128KB	128-KB product
ROM_SIZE_192KB	192-KB product
ROM_SIZE_256KB	256-KB product
ROM_SIZE_384KB	384-KB product
ROM_SIZE_512KB	512-KB product
ROM_SIZE_768KB	768-KB product

4.4片上调试安全ID

RL78G23微控制器在闪存中的地址000c4h至000cdh处提供片上调试安全ID区域，以便第三方不会读取存储器内容。

如果在自编程期间执行引导交换，则交换地址000C4H至000cdh处的区域和地址010c4h至010cdh处的区域。因此，在000c4h至000CDH的区域中设置的相同值也必须在040C4H至040cdh的区域中设置。

4.5 Resources Used by the Sample Program

4.5.1 List of Sections in the ROM Area

Table4-4 lists the sections that the sample program uses in the ROM area.

Table4-4 List of the Sections in the ROM Area

Section Name	Description
RFD_DATA_n	Data section for RFD RL78 Type01
RFD_CMN_f	Program section for the API functions that control the common flash memory
RFD_CF_f	Program section for the API functions that control the code flash memory
RFD_EX_f	Program section for the API functions that control the extra area
RFD_DF_f	Program section for the API functions that control the data flash memory
SMP_CMN_f	Program section for the sample functions that control the common flash memory
SMP_CF_f	Program section for the sample functions that control the code flash memory
BOOT_AREA1	Program section for boot cluster 1
USER_APPLICATION	Program section for the user application
COPY_FLAG_f	Program section for storing the copy completion flag
TEMPORARY_AREA	Program section for storing the receive data

4.5.2 List of the Sections in the RAM Area

Table4-5 lists the sections that the sample program uses in the RAM area.

Table4-5 List of the Sections in the RAM Area

Section Name	Description
RFD_DATA_nR	Data section for RFD RL78 Type01
RFD_CMN_fR	Program section for the API functions that control the common flash memory
RFD_CF_fR	Program section for the API functions that control the code flash memory
RFD_EX_fR	Program section for the API functions that control the extra area
SMP_CMN_fR	Program section for the sample functions that control the common flash memory
SMP_CF_fR	Program section for the sample functions that control the code flash memory

4.5示例程序使用的资源

4.5.1ROM区段列表Table4-4列出了示例程序在ROM区中使用的段。

表4-4ROM区的部分列表

部分名称	Description
RFD_DATA_n	RFDRL78Type01数据部分
RFD_CMN_f	控制通用闪存的API函数的程序部分
RFD_CF_f	控制代码闪存的API函数的程序部分
RFD_EX_f	控制额外区域的API函数的程序部分
RFD_DF_f	控制数据闪存的API函数的程序部分
SMP_CMN_f	用于控制公共闪存的示例函数的程序部分
SMP_CF_f	控制代码闪存的示例函数的程序部分
BOOT_AREA1	引导群集1的程序部分
用户应用程序的USER_APPLICATION程序部分	
COPY_FLAG_f	用于存储复制完成标志的程序部分
TEMPORARY_AREA	用于存储接收数据的程序部分

4.5.2RAM区域中的区段列表Table4-5列出了示例程序在RAM区域中使用的区段。

表4-5RAM区域的部分列表

部分名称	Description
RFD_DATA_nR	RFDRL78Type01数据部分
RFD_CMN_fR	控制通用闪存的API函数的程序部分
RFD_CF_fR	控制代码闪存的API函数的程序部分
RFD_EX_fR	控制额外区域的API函数的程序部分
SMP_CMN_fR	用于控制公共闪存的示例函数的程序部分
SMP_CF_fR	控制代码闪存的示例函数的程序部分

4.6 List of Constants

Table4-6 and Table4-7 list the constants that are used in the sample program.

Table4-6 List of Constants (1/2)

Constant Name	Value Set By This Constant	Description
ROM_SIZE_96KB	01H	Value that sets the ROM size to 96 KB
ROM_SIZE_128KB	01H	Value that sets the ROM size to 128 KB
ROM_SIZE_192KB	01H	Value that sets the ROM size to 192 KB
ROM_SIZE_256KB	01H	Value that sets the ROM size to 256 KB
ROM_SIZE_384KB	01H	Value that sets the ROM size to 384 KB
ROM_SIZE_512KB	01H	Value that sets the ROM size to 512 KB
ROM_SIZE_768KB	01H	Value that sets the ROM size to 768 KB
LED_ON	01H	LED ON
LED_OFF	00H	LED OFF
WRITE_DATA_SIZE	0100H	Size of data written to the code flash memory (256 bytes)
CF_BLOCK_SIZE	0800H	Block size of the code flash memory (2,048 bytes)
BT1_START_ADDRESS	00004000H	Start address of boot cluster 1
BT1_END_ADDRESS	00007FFFH	End address of boot cluster 1
EXECUTE_START_ADDRESS	00008000H	Start address of the Execute area
EXECUTE_END_ADDRESS[Note]	00013FFFH	End address of the Execute area
TEMPORARY_START_ADDRESS[Note]	00014000H	Start address of the Temporary area
TEMPORARY_END_ADDRESS[Note]	0001FFFFH	End address of the Temporary area
CPU_FREQUENCY	32	CPU operating frequency
COMMAND_START	02H	Command code for the START command
COMMAND_WRITE_BOOT1	03H	Command code for the WRITE_BOOT1 command
COMMAND_WRITE_TEMP	04H	Command code for the WRITE_TEMP command
COMMAND_END	05H	Command code for the END command
VALUE_U08_MASK1_FSQ_STATUS_ERR_ERASE	01H	Error status mask value for the execution results of the flash memory sequencer Bit 0: Erase command error
VALUE_U08_MASK1_FSQ_STATUS_ERR_WRITE	02H	Error status mask value for the execution results of the flash memory sequencer Bit 1: Write command error
VALUE_U08_MASK1_FSQ_STATUS_ERR_BLANKCHECK	08H	Error status mask value for the execution results of the flash memory sequencer Bit 3: Blank check command error
VALUE_U08_MASK1_FSQ_STATUS_ERR_CFDF_SEQUENCER	10H	Error status mask value for the execution results of the flash memory sequencer Bit 4: Code/data flash area sequencer error
VALUE_U08_MASK1_FSQ_STATUS_ERR_EXTRA_SEQUENCER	20H	Error status mask value for the execution results of the flash memory sequencer Bit 5: Extra area sequencer error

Note: The address differs depending on the product used.

Table4-7 List of Constants (2/2)

VALUE_U08_SHIFT_ADDR_TO_BLOCK_CF	11	Constant for bit shifting performed for calculating the block number of the code flash memory
VALUE_U01_MASK0_1BIT	0	Constant for arithmetic operation (0)

4.6常量列表

Table4-6和Table4-7列出了示例程序中使用的常量。

Table4-6常量列表(12)

常量名称	此常量设置的值	Description
ROM_SIZE_96KB	01H	将ROM大小设置为96KB的值
ROM_SIZE_128KB	01H	将ROM大小设置为128KB的值
ROM_SIZE_192KB	01H	将ROM大小设置为192KB的值
ROM_SIZE_256KB	01H	将ROM大小设置为256KB的值
ROM_SIZE_384KB	01H	将ROM大小设置为384KB的值
ROM_SIZE_512KB	01H	将ROM大小设置为512KB的值
ROM_SIZE_768KB	01H	将ROM大小设置为768KB的值
LED_ON	01H	领导着
LED_OFF	00H	引开
WRITE_DATA_SIZE	0100H	写入代码闪存的数据大小（256字节）
CF_BLOCK_SIZE	0800H	代码闪存的块大小（2 048字节）
BT1_START_ADDRESS	00004000H	启动集群1的起始地址
BT1_END_ADDRESS	00007FFFH	引导集群1的结束地址
EXECUTE_START_ADDRESS	00008000H	执行区域的起始地址
EXECUTE_END_ADDRESS[Note]	00013FFFH	执行区域的结束地址
TEMPORARY_START_ADDRESS[Note]	00014000H	临时区域的起始地址
TEMPORARY_END_ADDRESS[Note]	0001FFFFH	临时区域的结束地址
CPU_FREQUENCY	32	CPU工作频率
COMMAND_START	02H	启动命令的命令代码
COMMAND_WRITE_BOOT1	03H	WRITE_BOOT1命令的命令代码
COMMAND_WRITE_TEMP	04H	WRITE_TEMP命令的命令代码
COMMAND_END	05H	结束命令的命令代码
VALUE_U08_MASK1_FSQ_STATUS_ERR_ERASE	01H	闪存定序器位0的执行结果的错误状态掩码值：擦除命令错误
VALUE_U08_MASK1_FSQ_STATUS_ERR_WRITE	02H	闪存定序器第1位执行结果的错误状态掩码值：写命令错误
VALUE_U08_MASK1_FSQ_STATUS_ERR_BLANKCHECK	08H	闪存定序器第3位执行结果的错误状态掩码值：空白检查命令错误
VALUE_U08_MASK1_FSQ_STATUS_ERR_CFDF_SEQUENCER	10H	闪存定序器执行结果的错误状态掩码值第4位：代码数据闪存区定序器错误
VALUE_U08_MASK1_FSQ_STATUS_ERR_EXTRA_SEQUENCER	20H	闪存定序器执行结果的错误状态掩码值第5位：额外区域定序器错误

注意：地址因使用的产品而异。

Table4-7常量列表(22)

VALUE_U08_SHIFT_ADDR_TO_BLOCK_CF	11	用于计算代码闪存的块号而执行的位移位的常数
VALUE_U01_MASK0_1BIT	0	算术运算常数(0)

VALUE_U01_MASK1_1BIT	1	Constant for arithmetic operation (1)
VALUE_U08_MASK0_8BIT	00H	Constant for arithmetic operation (00H)
VALUE_U08_MASK1_8BIT	FFH	Constant for arithmetic operation (FFH)
COPY_FLAG_USUAL	AAAA5555H	Value set in the copy flag section

4.7 Enumeration Type

Table4-8 defines the enumeration-type variable used by the sample program.

Table4-8 enum e_ret (Enumeration Variable Name: e_ret_t)

Symbol Name	Value	Description
ENUM_RET_STS_OK	00H	Normal status
ENUM_RET_STS_RECEIVING	01H	Waiting for a command to be sent, or receiving a command
ENUM_RET_ERR_CFDF_SEQUENCER	02H	Code/data flash area sequencer error
ENUM_RET_ERR_EXTRA_SEQUENCER	03H	Extra area sequencer error
ENUM_RET_ERR_ERASE	04H	Erase error
ENUM_RET_ERR_WRITE	05H	Write error
ENUM_RET_ERR_BLANKCHECK	06H	Blank error
ENUM_RET_ERR_CHECK_WRITE_DATA	07H	Error in comparison between the written data against the read value
ENUM_RET_ERR_MODE_MISMATCHED	08H	Mode mismatch error
ENUM_RET_ERR_PARAMETER	09H	Parameter error
ENUM_RET_ERR_CONFIGURATION	0AH	Device configuration error
ENUM_RET_ERR_PACKET	0BH	Packet reception error

VALUE_U01_MASK1_1BIT	1	算术运算常数(1)
VALUE_U08_MASK0_8BIT	00H	算术运算常数(00H)
VALUE_U08_MASK1_8BIT	FFH	算术运算常数(FFH)
COPY_FLAG_USUAL	AAAA5555H	复制标志部分中设置的值

4.7枚举类型

Table4-8定义了示例程序使用的枚举类型变量。

Table4-8 enum e_ret (Enumeration Variable Name: e_ret_t)

符号名称	值	描述
ENUM_RET_STS_OK	00H	正常状态
ENUM_RET_STS_RECEIVING	01H	等待发送命令，或接收命令
ENUM_RET_ERR_CFDF_SEQUENCER	02H	代码数据闪存区序列发生器错误
ENUM_RET_ERR_EXTRA_SEQUENCER	03H	额外区域测序器错误
ENUM_RET_ERR_ERASE	04H	擦除错误
ENUM_RET_ERR_WRITE	05H	写入错误
ENUM_RET_ERR_BLANKCHECK	06H	空白错误
ENUM_RET_ERR_CHECK_WRITE_DATA	07H	写入数据与读取值之间的比较错误
ENUM_RET_ERR_MODE_MISMATCHED	08H	模式不匹配错误
ENUM_RET_ERR_PARAMETER	09H	参数错误
ENUM_RET_ERR_CONFIGURATION	0AH	设备配置错误
ENUM_RET_ERR_PACKET	0BH	数据包接收错误

4.8 List of Variables

Table4-9 lists the global variables that are used in the sample program.

Table4-9 List of Global Variables

Type	Variable Name	Description	Function Supporting the Variable
uint8_t	f_UART0_sendend	Flag indicating that data sending by the UART0 was completed	r_Send_nByte r_Config_UART0_callback_sendend
uint32_t	g_copy_end	Flag indicating that data copy was ended normally	main
uint8_t	g_recv_data [261]	Receive data buffer	R_Config_UART0_Receive r_AsyncRecvPacketData
uint8_t	g_soft_recv_overrun	Flag indicating that data larger than the receive data buffer was received	r_Config_UART0_callback_softwareoverrun r_ClearUARTRecvBuff r_AsyncRecvPacketData

4.9 List of Functions

Table4-10 lists the functions that are used in the sample program.

Table4-10 List of Functions

Function Name	Summary
r_rfd_initialize	Initialization processing for RFD RL78 Type01
r_cmd_start	START command processing
r_cmd_end	END command processing
r_CF_RangeErase	Range erase processing for the code flash memory
r_CF_EraseBlock	Block erase processing for the code flash memory
r_CF_WriteVerifySequence	Write-and-verify processing for the code flash memory
r_CF_WriteData	Write processing for the code flash memory
r_CF_VerifyData	Verify processing for the code flash memory
r_CheckCFDFSequencerEnd	Sequence end processing for the code flash memory
r_CheckExtraSequencerEnd	Sequence end processing for the extra area
r_RequestBootSwap	Boot swapping execution processing
r_Config_UART0_callback_sendend	Callback processing at a sending completion interrupt for UART0
r_Send_nByte	Data sending processing by UART0
r_SendACK	Normal response sending processing by UART0
r_CF_TempCopy	Processing to copy data from the Temporary area
r_CF_MemoryWrite	Processing to reprogram the code flash memory
r_AsyncRecvPacketData	Processing to receive asynchronous command packets
r_GetUARTRecvSize	Processing to obtain the size of the receive data
r_ClearUARTRecvBuff	Processing to clear the receive buffer
userApplicationLoop	Function to implement user application
updateLoop	Processing to receive and run the firmware update command
errorLedOn	Processing to turn on the error LED

4.8变量列表

表4-9列出了示例程序中使用的全局变量。

Table4-9全局变量列表

Type	变量名称	Description	支持变量的函数
uint8_t	f_UART0_sendend	指示UART0发送数据已完成的标志	r_Send_nByte r_Config_UART0_callback_sendend
uint32_t	g_copy_end	指示数据复制正常结束的标志	main
uint8_t	g_recv_data [261]	接收数据缓冲区	R_Config_UART0_Receive r_AsyncRecvPacketData
uint8_t	g_soft_recv_overrun	指示接收到大于接收数据缓冲区的数据的标志	r_Config_UART0_callback_softwareoverrun r_ClearUARTRecvBuff r_AsyncRecvPacketData

4.9函数列表

表4-10列出了示例程序中使用的函数。

表4-10函数列表

函数名称	Summary
r_rfd_initialize	RFDRL78Type01的初始化处理
r_cmd_start	启动命令处理
r_cmd_end	结束命令处理
r_CF_RangeErase	代码闪存的范围擦除处理
r_CF_EraseBlock	代码闪存的块擦除处理
r_CF_WriteVerifySequence	代码闪存的写和验证处理
r_CF_WriteData	代码闪存的写入处理
r_CF_VerifyData	验证代码闪存的处理
r_CheckCFDFSequencerEnd	码闪存的序列结束处理
r_CheckExtraSequencerEnd	额外区域的序列结束处理
r_RequestBootSwap	引导交换执行处理
r_Config_UART0_callback_sendend	在发送完成中断的回调处理
r_Send_nByte	Uart0的数据发送处理
r_SendACK	Uart0的正常响应发送处理
r_CF_TempCopy	从临时区域复制数据的处理
r_CF_MemoryWrite	处理以重新编程代码闪存
r_AsyncRecvPacketData	处理以接收异步命令分组
r_GetUARTRecvSize	处理以获得接收数据的大小
r_ClearUARTRecvBuff	处理以清除接收缓冲区
userApplicationLoop	实现用户应用程序的功能
updateLoop	处理以接收和运行固件更新命令
errorLedOn	处理以打开错误LED

4.10 Specifications of Functions

This section describes the specifications of the functions used in the sample code.

r_rfd_initialize	
Summary	Initialization processing for RFD RL78 Type01
Header	r_rfd_common_api.h, r_rfd_code_flash_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_rfd_initialize(void);
Explanation	This function performs the processing to initialize RFD RL78 Type01.
Arguments	None
Return values	ENUM_RET_STS_OK: Normal end ENUM_RET_ERR_CONFIGURATION: Clock configuration error ENUM_RET_ERR_PARAMETER: Frequency setting error

r_cmd_start	
Summary	START command processing
Header	r_rfd_common_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_cmd_start(void);
Explanation	This function performs processing in response to reception of the START command.
Arguments	None
Return values	ENUM_RET_STS_OK: Normal end ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_ERASE: Erase error

r_cmd_end	
Summary	END command processing
Header	r_rfd_common_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_cmd_end(void);
Explanation	This function performs processing in response to reception of the END command.
Arguments	None
Return values	ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error

r_CF_RangeErase	
Summary	Range erase processing for the code flash memory
Header	r_rfd_common_api.h, r_rfd_code_flash_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_RangeErase(uint32_t start_addr, uint32_t end_addr);
Explanation	This function erases data in the code flash memory. Data is erased in blocks. The blocks in the range of addresses specified for arguments will be erased.
Arguments	uint32_t start_addr: Erase start address uint32_t end_addr: Erase end address
Return values	ENUM_RET_STS_OK: Normal end ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_ERASE: Erase error

4.10功能规格

本节介绍示例代码中使用的函数的规范。

r_rfd_initialize	
Summary	RFDRL78Type01的初始化处理
Header	r_rfd_common_api.h, r_rfd_code_flash_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_rfd_initialize(void);
Explanation	该函数执行初始化RFDRL78001的处理。
Arguments	None
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_CONFIGURATION：时钟配置错误 ENUM_RET_ERR_PARAMETER：频率设置错误

r_cmd_start	
Summary	启动命令处理
Header	r_rfd_common_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_cmd_start(void);
Explanation	该功能响应于启动命令的接收而执行处理。
Arguments	None
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_MODE_MISMATCHED：模式不匹配错误 ENUM_RET_ERR_ERASE：擦除错误

r_cmd_end	
Summary	结束命令处理
Header	r_rfd_common_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_cmd_end(void);
Explanation	该功能响应于结束命令的接收而执行处理。
Arguments	None
返回值	ENUM_RET_ERR_MODE_MISMATCHED：模式不匹配错误

r_CF_RangeErase	
Summary	代码闪存的范围擦除处理
Header	r_rfd_common_api.h, r_rfd_code_flash_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_RangeErase(uint32_t start_addr, uint32_t end_addr);
Explanation	此功能擦除代码闪存中的数据。 数据以块为单位被擦除。为参数指定的地址范围内的块将被擦除。
Arguments	uint32_tstart_addr:擦除起始地址uint32 _tend_addr:擦除结束地址
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_MODE_MISMATCHED：模式不匹配错误 ENUM_RET_ERR_ERASE：擦除错误

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

r_CF_WriteVerifySequence	
Summary	Write-and-verify processing for the code flash memory
Header	r_rfd_common_api.h, r_rfd_code_flash_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_WriteVerifySequence(uint32_t write_start_addr, uint16_t write_data_length, uint8_t __near *write_data);
Explanation	This function writes data to the code flash memory and verifies the written data.
Arguments	uint32_t start_addr,: Write start address uint16_t write_data_length: Size of the data to be written uint8_t __near *write_data: Data to be written
Return values	ENUM_RET_STS_OK: Normal end ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_ERASE: Erase error

r_CF_WriteData	
Summary	Write processing for the code flash memory
Header	r_rfd_common_api.h, r_rfd_code_flash_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_WriteData(uint32_t start_addr, uint16_t write_data_length, uint8_t __near *write_data);
Explanation	This function writes data to the code flash memory.
Arguments	uint32_t start_addr,: Write start address uint16_t write_data_length: Size of the data to be written uint8_t __near *write_data: Data to be written
Return values	ENUM_RET_STS_OK: Normal end ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_WRITE: Write error

r_CF_EraseBlock	
Summary	代码闪存的块擦除处理
Header	r_rfd_common_api.h, r_rfd_code_flash_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_EraseBlock(uint32_t start_addr);
Explanation	此功能擦除代码闪存中的数据。 一个数据块被擦除。包含为参数指定的地址的块将被擦除。
Arguments	uint32_tstart_addr：擦除起始地址
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_MODE_MISMATCHED：模式不匹配错误 ENUM_RET_ERR_ERASE：擦除错误

r_CF_WriteVerifySequence	
Summary	代码闪存的写和验证处理
Header	r_rfd_common_api.h, r_rfd_code_flash_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_WriteVerifySequence(uint32_t write_start_addr, uint16_t write_data_length, uint8_t __near *write_data);
Explanation	该函数将数据写入代码闪存并验证写入的数据。
Arguments	uint32_tstart_addr :写入开始地址uint16_twrite_data_length:要写入的数据的大小uint8_t__near*write_data:要写入的数据
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_MODE_MISMATCHED：模式不匹配错误 ENUM_RET_ERR_ERASE：擦除错误

r_CF_WriteData	
Summary	代码闪存的写入处理
Header	r_rfd_common_api.h, r_rfd_code_flash_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_WriteData(uint32_t start_addr, uint16_t write_data_length, uint8_t __near *write_data);
Explanation	此函数将数据写入代码闪存。
Arguments	uint32_tstart_addr :写入开始地址uint16_twrite_data_length:要写入的数据的大小uint8_t__near*write_data:要写入的数据
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_MODE_MISMATCHED：模式不匹配错误 ENUM_RET_ERR_WRITE：写入错误

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

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

r_CheckExtraSequencerEnd	
Summary	Sequence end processing for the extra area
Header	r_rfd_common_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CheckExtraSequencerEnd (void);
Explanation	This function confirms that the extra area sequence has terminated.
Arguments	None
Return values	ENUM_RET_STS_OK: Normal end ENUM_RET_ERR_EXTRA_SEQUENCER: Extra area sequencer error ENUM_RET_ERR_ERASE: Erase error ENUM_RET_ERR_WRITE: Write error ENUM_RET_ERR_BLANKCHECK: Blank error

r_CF_VerifyData	
Summary	验证代码闪存的处理
Header	r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_VerifyData(uint32_t start_addr, uint16_t data_length, uint8_t __near * write_data);
Explanation	此函数验证写入代码闪存的数据。
Arguments	Uint32_tstart_addr：验证开始地址uint16_tdata_lengt h：数据大小uint8_t__near*write_data：要与之比较的 数据
返回值	ENUM_RET_STS_OK：正常结束（匹配）ENUM_RET_ERR_CHECK_WRITE_DAT A：写入数据和读取值之间的比较错误（不匹配）

r_CheckCFDFSequencerEnd	
Summary	码闪存的序列结束处理
Header	r_rfd_common_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CheckCFDFSequencerEnd(void);
Explanation	此功能确认代码闪存序列已终止。
Arguments	None
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_CFDF_SEQUENCER：代码数据闪存序列器错误ENUM_RET_ERR _ERASE：擦除错误ENUM_RET_ERR_WRITE：写入错误 ENUM_RET_ERR_BLANKCHECK：空白错误

r_CheckExtraSequencerEnd	
Summary	额外区域的序列结束处理
Header	r_rfd_common_api.h, r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CheckExtraSequencerEnd (void);
Explanation	此功能确认额外区域序列已终止。
Arguments	None
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_EXTRA_SEQUENCER：额外区域序列器错误 ENUM_RET_ERR_ERASE：擦除错误ENU M_RET_ERR_WRITE：写入错误 ENUM_RET_ERR_BLANKCHECK：空白错误

r_RequestBootSwap	
Summary	Boot swapping execution processing
Header	r_rfd_common_api.h, r_rfd_extra_area_api.h , r_cg_userdefine.h
Declaration	e_ret_t r_RequestBootSwap(void);
Explanation	After a reset is performed, this function enables the boot swapping settings, and then generates an internal reset to restart the CPU.
Arguments	None
Return values	ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error

r_Config_UART0_callback_sendend()	
Summary	Callback processing at a sending completion interrupt for UART0
Header	r_cg_macrodriver.h
Declaration	static void r_Config_UART0_callback_sendend(void);
Explanation	This is a callback function that is called at a sending completion interrupt for UART0.
Arguments	None
Return values	None

r_Send_nByte	
Summary	Data sending processing by UART0
Header	Config_UART0.h, Config_WDT.h
Declaration	MD_STATUS r_Send_nByte(uint8_t *tx_buff, const uint16_t tx_num);
Explanation	This function performs sending processing by UART0. This function waits until sending of the number of characters specified for an argument is completed.
Arguments	uint8_t *rx_buff: Pointer to the send data storage buffer const uint16_t rx_num: Number of characters to be sent
Return values	MD_OK: Normal end (sending completed) MD_ARGERROR: Parameter error

r_SendACK	
Summary	Normal response sending processing by UART0
Header	Config_UART0.h, Config_WDT.h
Declaration	MD_STATUS r_SendACK (void);
Explanation	This function uses UART0 to perform sending processing for normal response (01H).
Arguments	None
Return values	MD_OK: Normal end (sending completed) MD_ARGERROR: Parameter error

r_CF_TempCopy	
Summary	Processing to copy data from the Temporary area
Header	r_cg_userdefine.h, string.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_TempCopy(void);
Explanation	This function copies data from the Temporary area.
Arguments	None
Return values	ENUM_RET_STS_OK: Normal end

r_RequestBootSwap	
Summary	引导交换执行处理
Header	r_rfd_common_api.h, r_rfd_extra_area_api.h , r_cg_userdefine.h
Declaration	e_ret_t r_RequestBootSwap(void);
Explanation	执行重置后，此功能启用引导交换设置，然后生成内部重置以重新启动CPU。
Arguments	None
返回值	ENUM_RET_ERR_MODE_MISMATCHED：模式不匹配错误

r_Config_UART0_callback_sendend()	
Summary	Uart0的发送完成中断处的回调处理
Header	r_cg_macrodriver.h
Declaration	静态voidr_Config_UART0_callback_sendend（void）；
Explanation	这是一个回调函数，在发送完成中断时调用
Arguments	None
返回值	None

r_Send_nByte	
Summary	Uart0的数据发送处理
Header	Config_UART0.h, Config_WDT.h
Declaration	MD_STATUS r_Send_nByte(uint8_t *tx_buff, const uint16_t tx_num);
Explanation	该功能由UART0执行发送处理。 此函数等待，直到完成为参数指定的字符数的发送。
Arguments	uint8_t*rx_buff：指向发送数据存储缓冲区的指针constui nt16_trx_num：要发送的字符数
返回值	MD_OK：正常结束（发送完成） MD_ARGERROR：参数错误

r_SendACK	
Summary	Uart0的正常响应发送处理
Header	Config_UART0.h, Config_WDT.h
Declaration	MD_STATUS r_SendACK (void);
Explanation	该功能使用UART0执行正常响应(01h)的发送处理。
Arguments	None
返回值	MD_OK：正常结束（发送完成） MD_ARGERROR：参数错误

r_CF_TempCopy	
Summary	从临时区域复制数据的处理
Header	r_cg_userdefine.h, string.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_TempCopy(void);
Explanation	此功能从临时区域复制数据。
Arguments	None
返回值	ENUM_RET_STS_OK：正常结束

r_CF_MemoryWrite	
Summary	Processing to reprogram the code flash memory
Header	r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_MemoryWrite(uint32_t* write_start_addr, uint32_t write_end_addr, uint8_t __near * write_data);
Explanation	This function writes data to memory.
Arguments	uint32_t* write_start_addr: Write start address uint32_t write_end_addr: Write end address uint8_t __near * write_data: Data to be written
Return values	ENUM_RET_STS_OK: Normal end ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_ERASE: Erase error

r_AsyncRecvPacketData	
Summary	Processing to receive asynchronous command packets
Header	r_cg_userdefine.h
Declaration	__far uint8_t r_AsyncRecvPacketData(uint8_t *p_cmd_type, uint8_t rdata[]);
Explanation	This function analyzes asynchronously received data and returns the status.
Arguments	uint8_t *p_cmd_type: Command information uint8_t rdata[]: Receive data buffer
Return values	ENUM_PACKET_STATUS_OK: Normal end ENUM_PACKET_STATUS_ERROR: Packet reception error ENUM_PACKET_STATUS_RECEIVING: Now receiving packets

r_GetUARTRecvSize	
Summary	Processing to obtain the size of the receive data
Header	r_cg_userdefine.h Config_UART0.h
Declaration	uint16_t r_GetUARTRecvSize(void);
Explanation	This function returns the length of the received data.
Arguments	None
Return values	Size: Length of the received data

r_ClearUARTRecvBuff	
Summary	Processing to clear the receive buffer
Header	r_cg_userdefine.h Config_UART0.h
Declaration	void r_ClearUARTRecvBuff(void);
Explanation	This function clears the buffer that stores received data.
Arguments	None
Return values	None

r_CF_MemoryWrite	
Summary	处理以重新编程代码闪存
Header	r_cg_userdefine.h
Declaration	R_RFD_FAR_FUNC e_ret_t r_CF_MemoryWrite(uint32_t* write_start_addr, uint32_t write_end_addr, uint8_t __near * write_data);
Explanation	此函数将数据写入内存。
Arguments	uint32_t*write_start_addr：写入开始地址uint32_twrite_end_addr：写入结束地址uint8_t__near*write_data：要写入的数据
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_MODE_MISMATCHED：模式不匹配错误 ENUM_RET_ERR_ERASE：擦除错误

r_AsyncRecvPacketData	
Summary	处理以接收异步命令分组
Header	r_cg_userdefine.h
Declaration	__far uint8_t r_AsyncRecvPacketData(uint8_t *p_cmd_type, uint8_t rdata[]);
Explanation	此函数分析异步接收的数据并返回状态。
Arguments	uint8_t*p_cmd_type：命令信息uint8_trdata[]：接收数据缓冲区
返回值	ENUM_PACKET_STATUS_OK：正常结束 ENUM_PACKET_STATUS_ERROR：数据包接收错误 ENUM_PACKET_STATUS_RECEIVING：现在接收数据包

r_GetUARTRecvSize	
Summary	处理以获得接收数据的大小
Header	r_cg_userdefine.h Config_UART0.h
Declaration	uint16_t r_GetUARTRecvSize(void);
Explanation	此函数返回接收数据的长度。
Arguments	None
返回值	Size：接收数据的长度

r_ClearUARTRecvBuff	
Summary	处理以清除接收缓冲区
Header	r_cg_userdefine.h Config_UART0.h
Declaration	void r_ClearUARTRecvBuff(void);
Explanation	此函数清除存储接收数据的缓冲区。
Arguments	None
返回值	None

userApplicationLoop	
Summary	Function to implement user application
Header	r_cg_userdefine.h
Declaration	void userApplicationLoop(void);
Explanation	Sample application implemented that blinks LED1/LED8.
Arguments	None
Return values	None

updateLoop	
Summary	Processing to receive and run the firmware update command
Header	r_cg_userdefine.h
Declaration	e_ret_t updateLoop(void);
Explanation	This function receives and runs the firmware update command.
Arguments	None
Return values	ENUM_RET_STS_OK: Normal end ENUM_RET_ERR_CONFIGURATION: Clock configuration error ENUM_RET_ERR_PARAMETER: Frequency setting error ENUM_PACKET_STATUS_ERROR: Packet reception error ENUM_PACKET_STATUS_RECEIVING: Receiving packets ENUM_RET_ERR_MODE_MISMATCHED: Mode mismatch error ENUM_RET_ERR_ERASE: Erase error

errorLedOn	
Summary	Processing to turn on the error LED
Header	r_cg_userdefine.h
Declaration	void errorLedOn(void);
Explanation	This function turns on LED7 and turns off the other LEDs if an error occurs.
Arguments	None
Return values	None

userApplicationLoop	
Summary	实现用户应用程序的功能
Header	r_cg_userdefine.h
Declaration	void userApplicationLoop(void);
Explanation	实现闪烁LED1LED8的示例应用程序。
Arguments	None
返回值	None

updateLoop	
Summary	处理以接收和运行固件更新命令
Header	r_cg_userdefine.h
Declaration	e_ret_t updateLoop(void);
Explanation	此功能接收并运行固件更新命令。
Arguments	None
返回值	ENUM_RET_STS_OK：正常结束 ENUM_RET_ERR_CONFIGURATION：时钟配置错误 ENUM_RET_ERR_PARAMETER：频率设置错误 ENUM_PACKET_STATUS_ERROR：数据包接收错误ENUM_PACKET_STATUS_RECEIVING：接收数据包 ENUM_RET_ERR_MODE_MISMATCHED：模式不匹配错误 ENUM_RET_ERR_ERASE：擦除错误

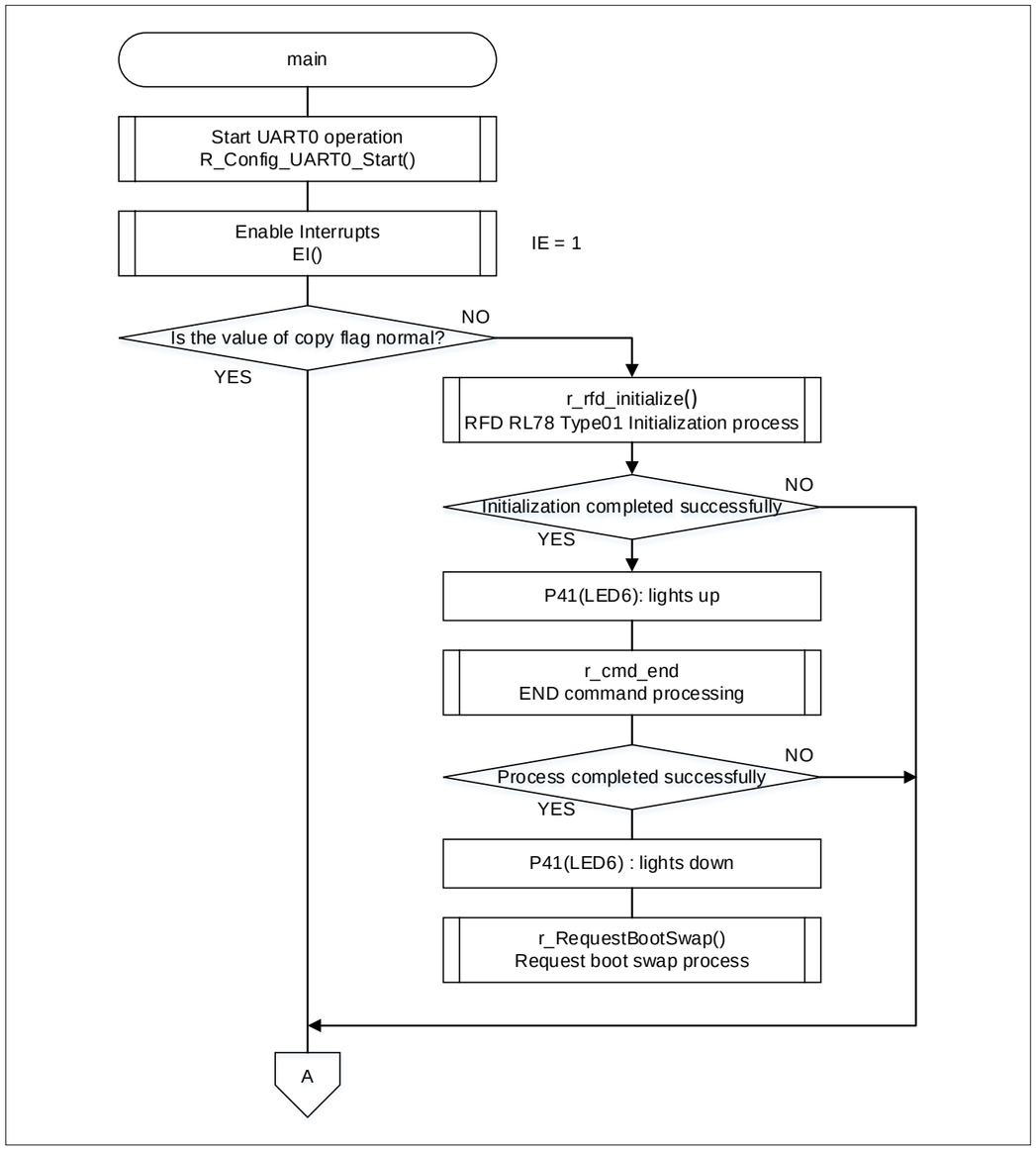
errorLedOn	
Summary	处理以打开错误LED
Header	r_cg_userdefine.h
Declaration	void errorLedOn(void);
Explanation	如果发生错误，此功能将打开LED7并关闭其他Led。
Arguments	None
返回值	None

4.11 Flowcharts

4.11.1 Main Processing

Figure 4-1, Figure 4-2 shows the flowchart of the main processing.

Figure 4-1 Main Processing (1/2)



4.11 Flowcharts

4.11.1主处理图4-1、图4-2示出了主处理的流程图。

图4-1主要处理(12)

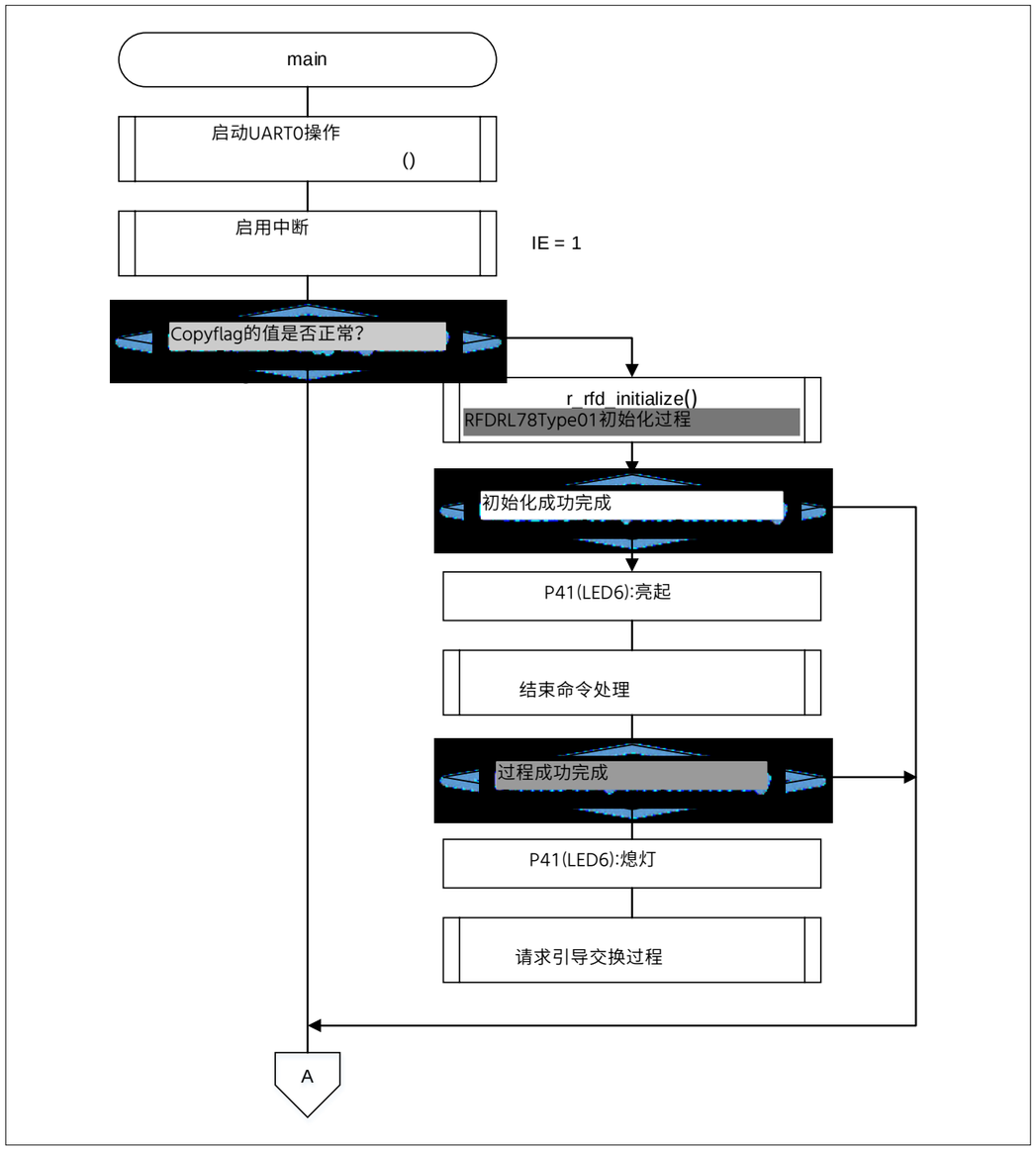


Figure 4-2 Main Processing (2/2)

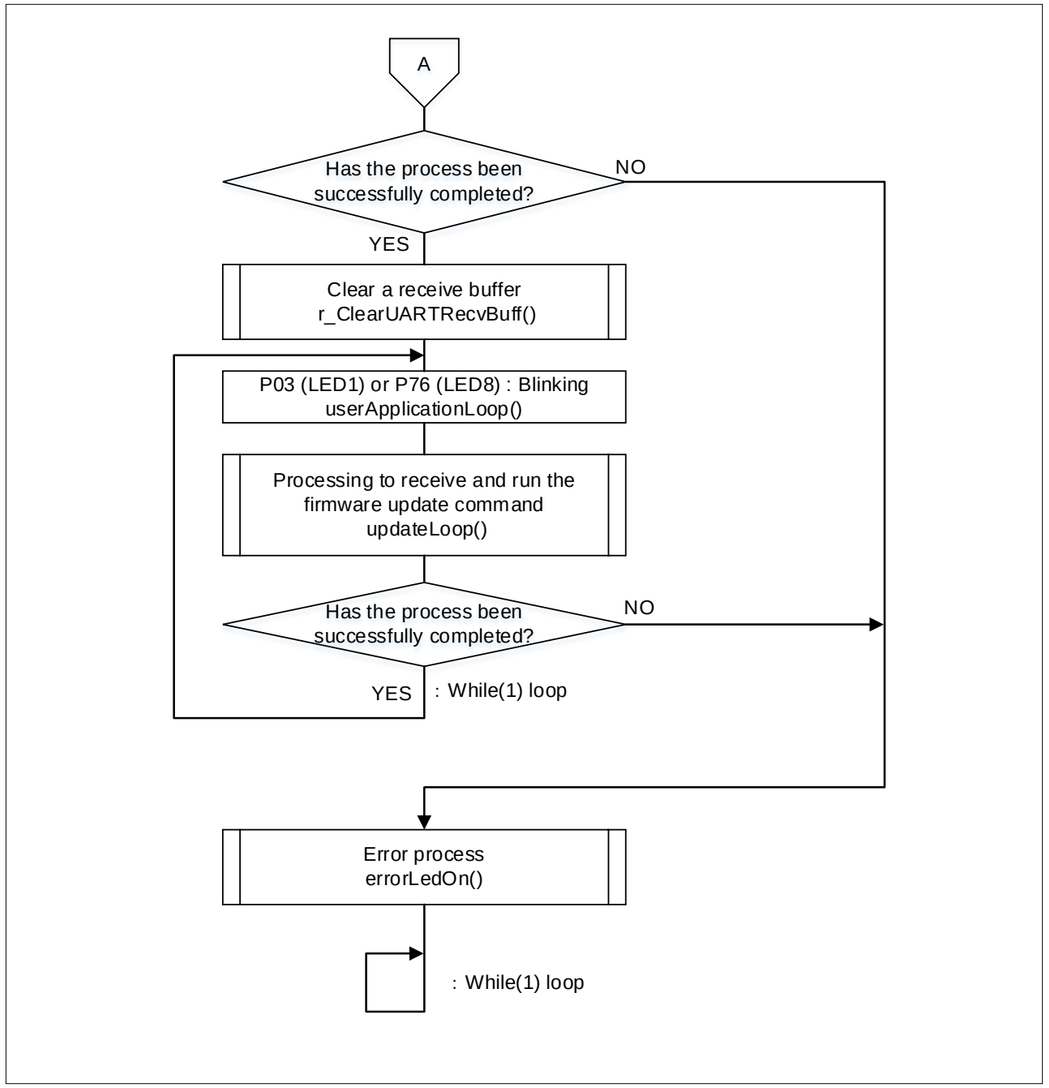
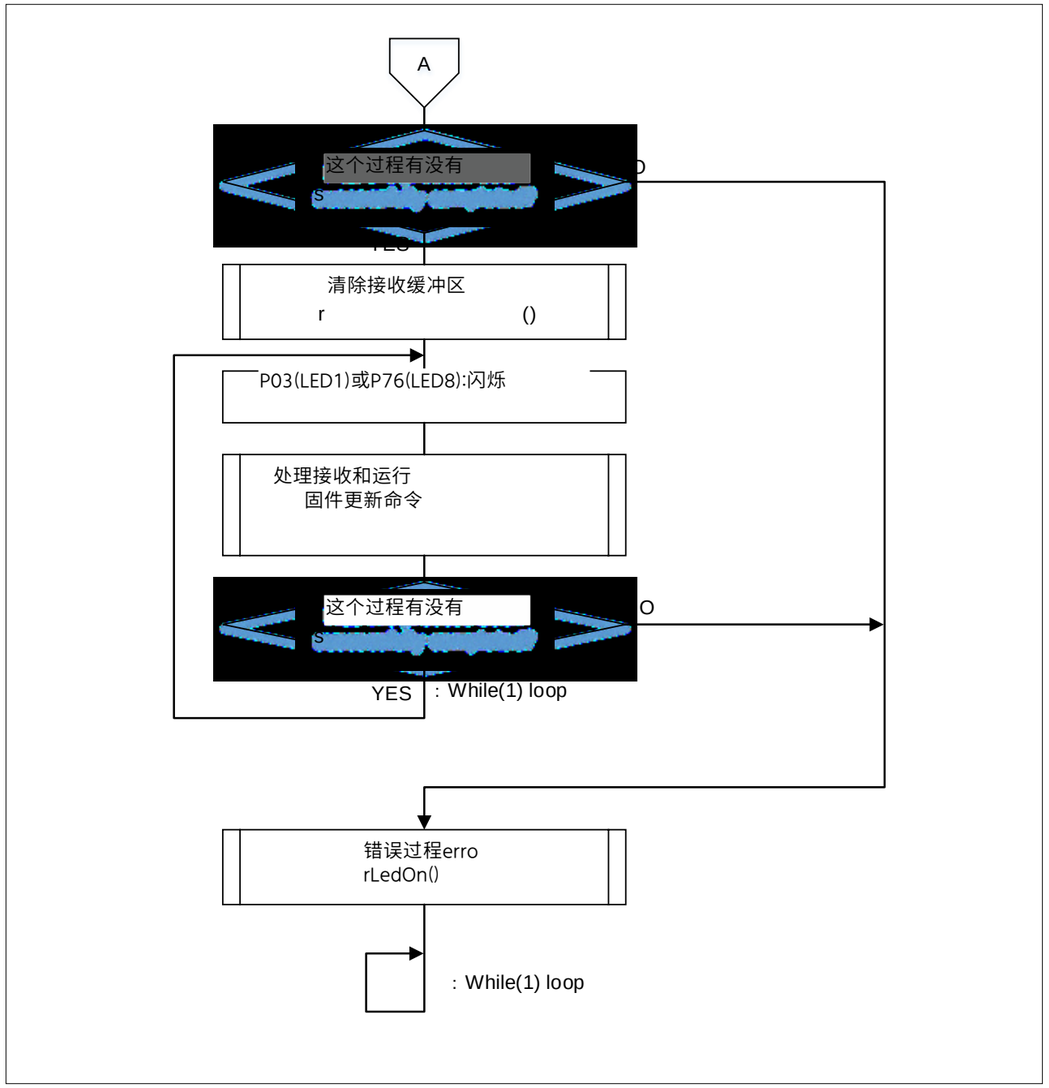


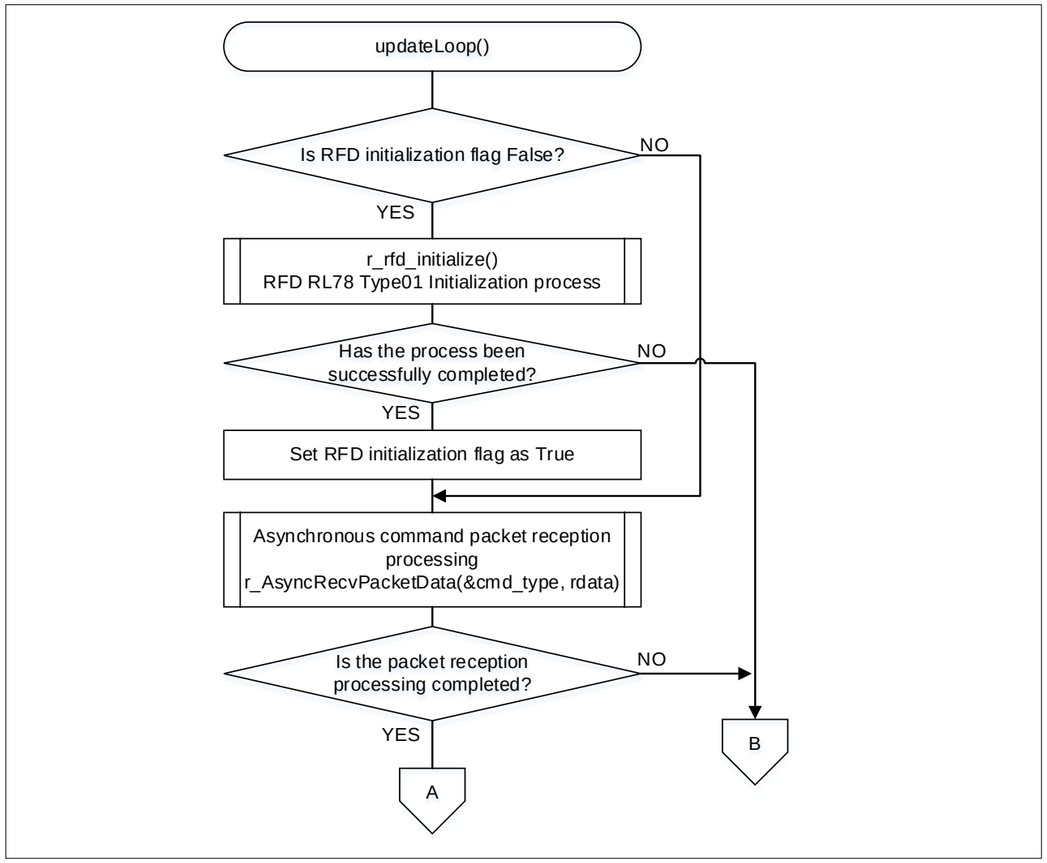
图4-2主要处理(22)



4.11.2 Processing to receive and run the firmware update command

Figure 4-3, Figure 4-4, and Figure 4-5 shows the flowchart of processing to receive and run the firmware update command

Figure 4-3 Processing to receive and run the firmware update command (1/3)



4.11.2接收和运行固件更新命令的处理图4-3、图4-4和图4-5示出了接收和运行固件更新命令的处理流程图

图4-3接收和运行固件更新命令的处理(13)

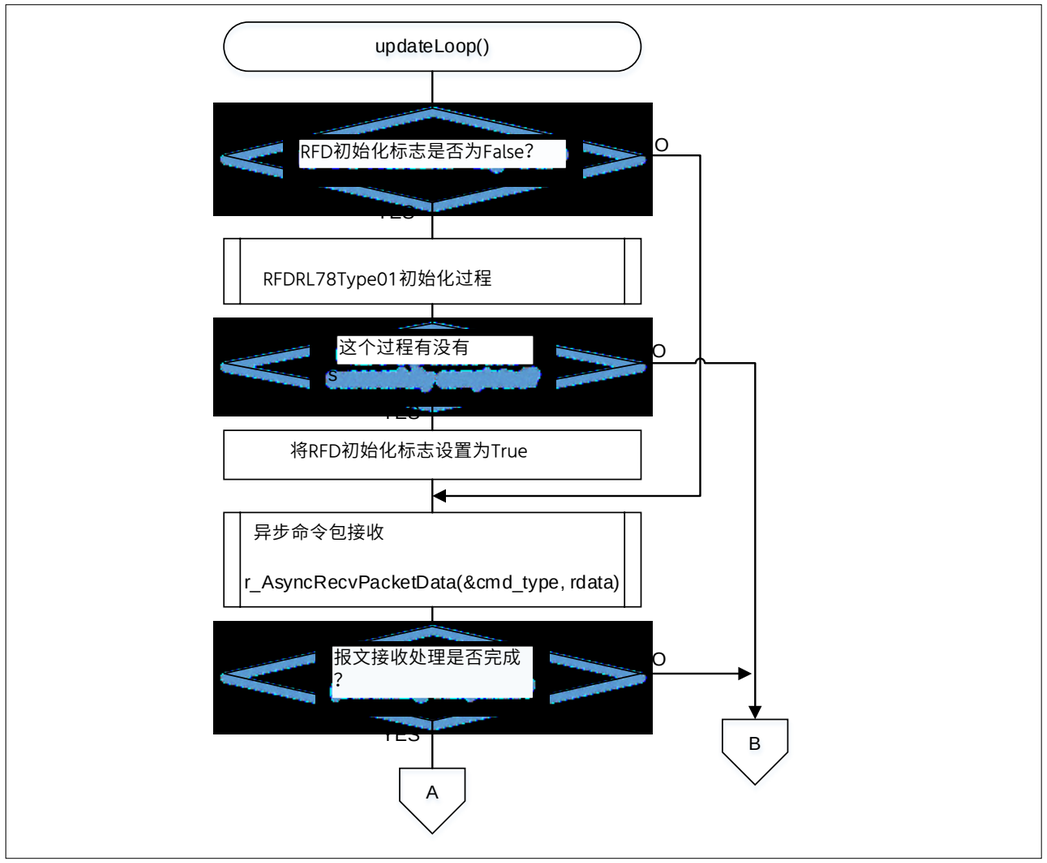


Figure 4-4 Processing to receive and run the firmware update command (2/3)

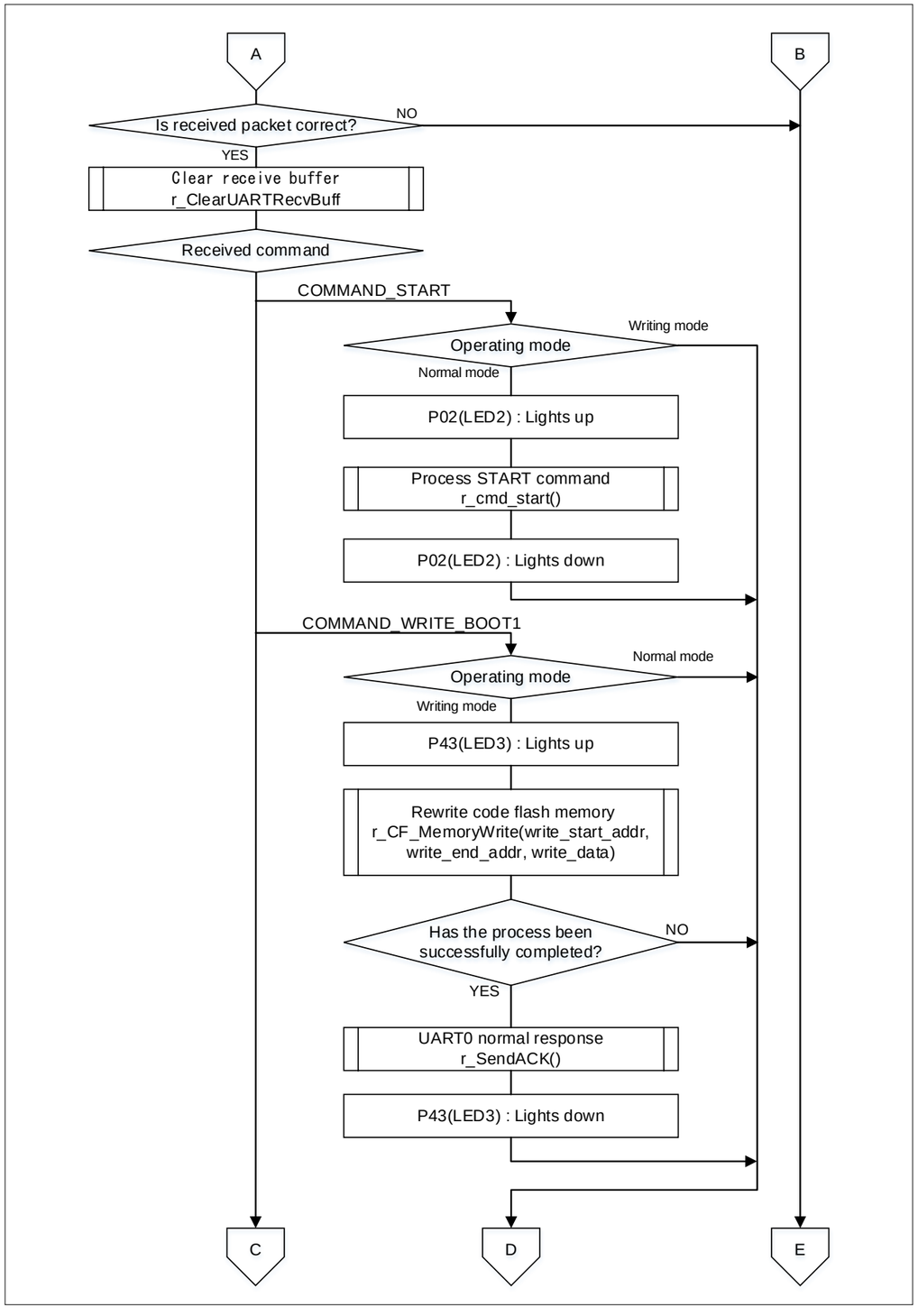


图4-4接收和运行固件更新命令的处理(23)

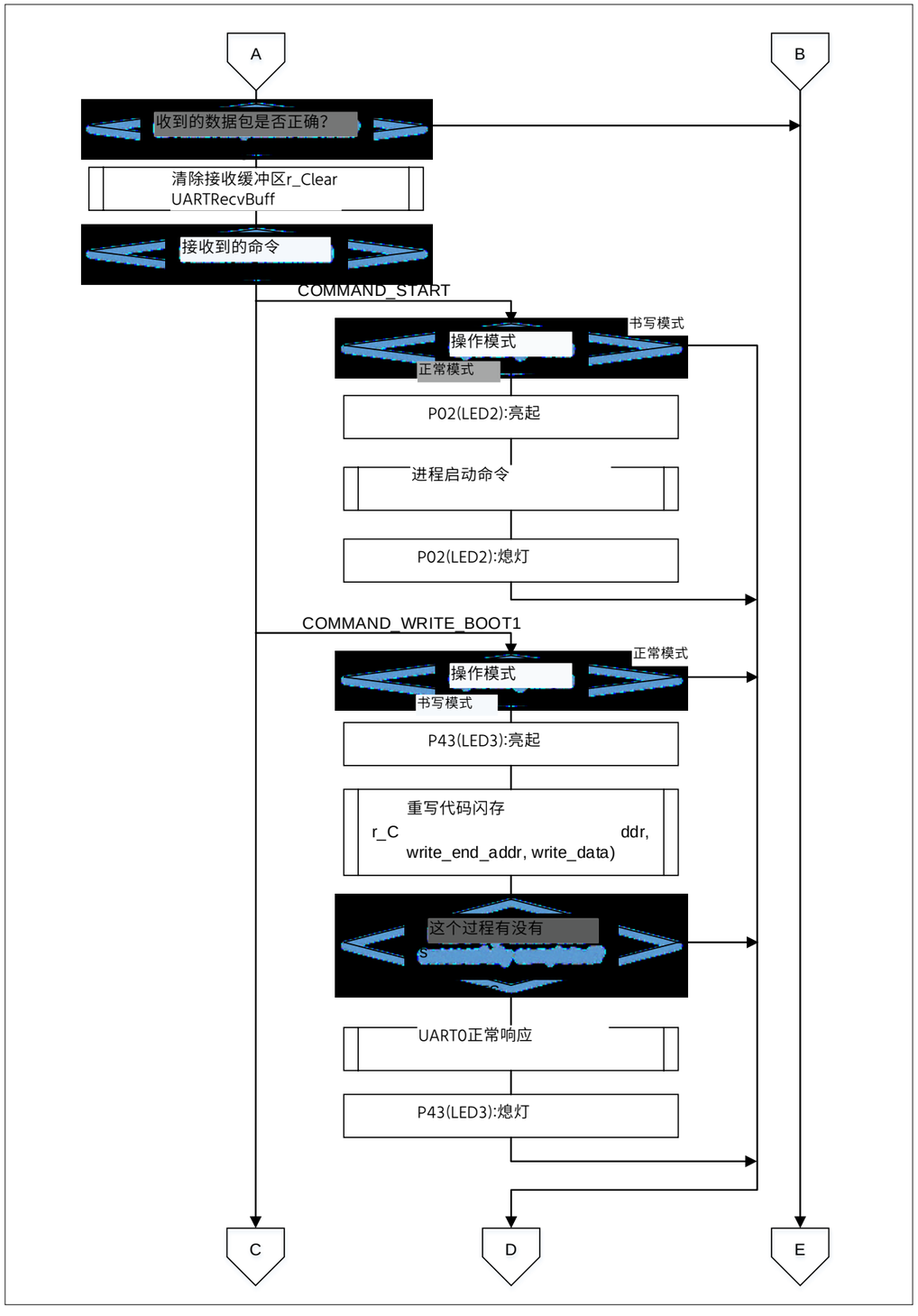


Figure 4-5 Processing to receive and run the firmware update command (3/3)

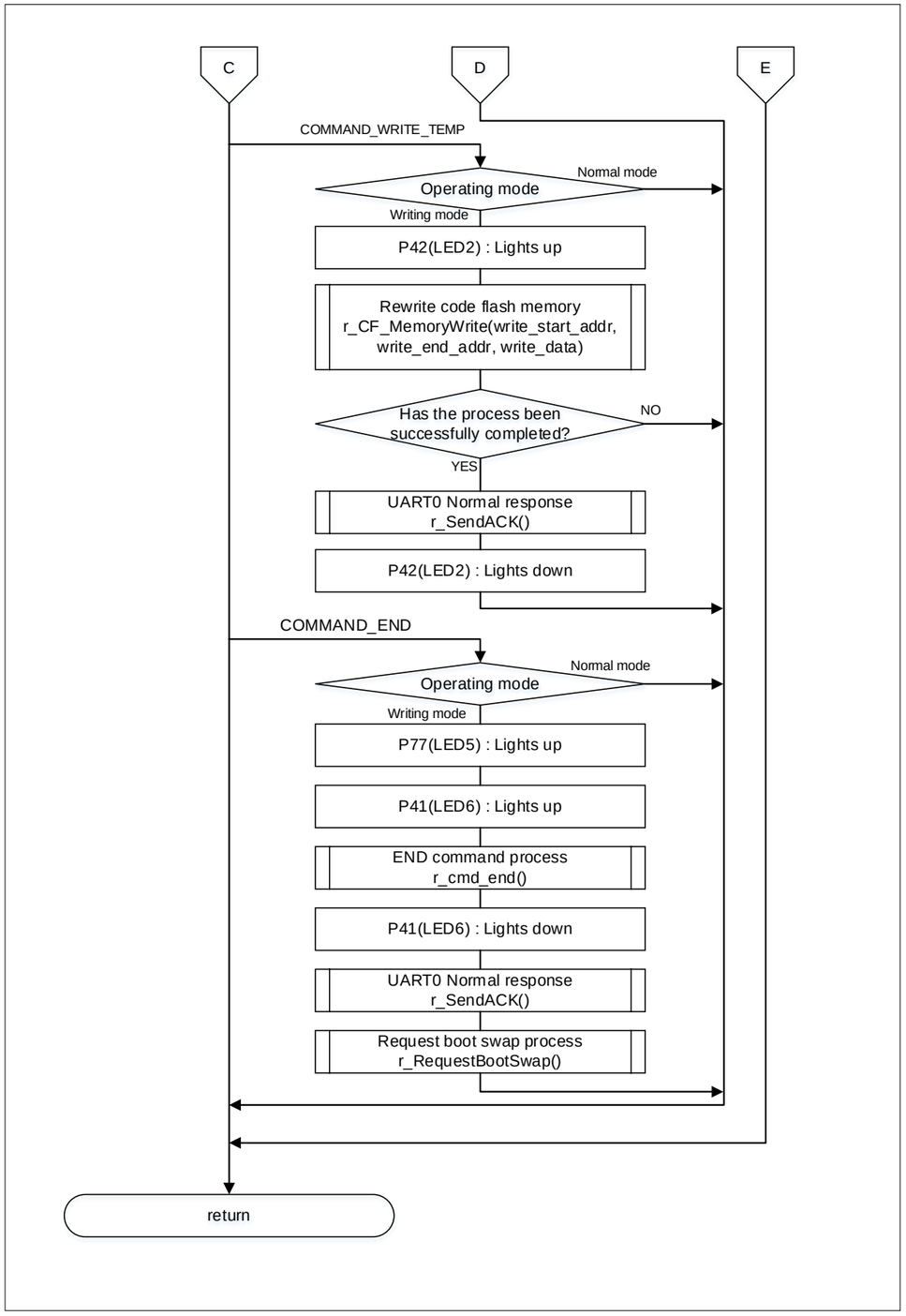
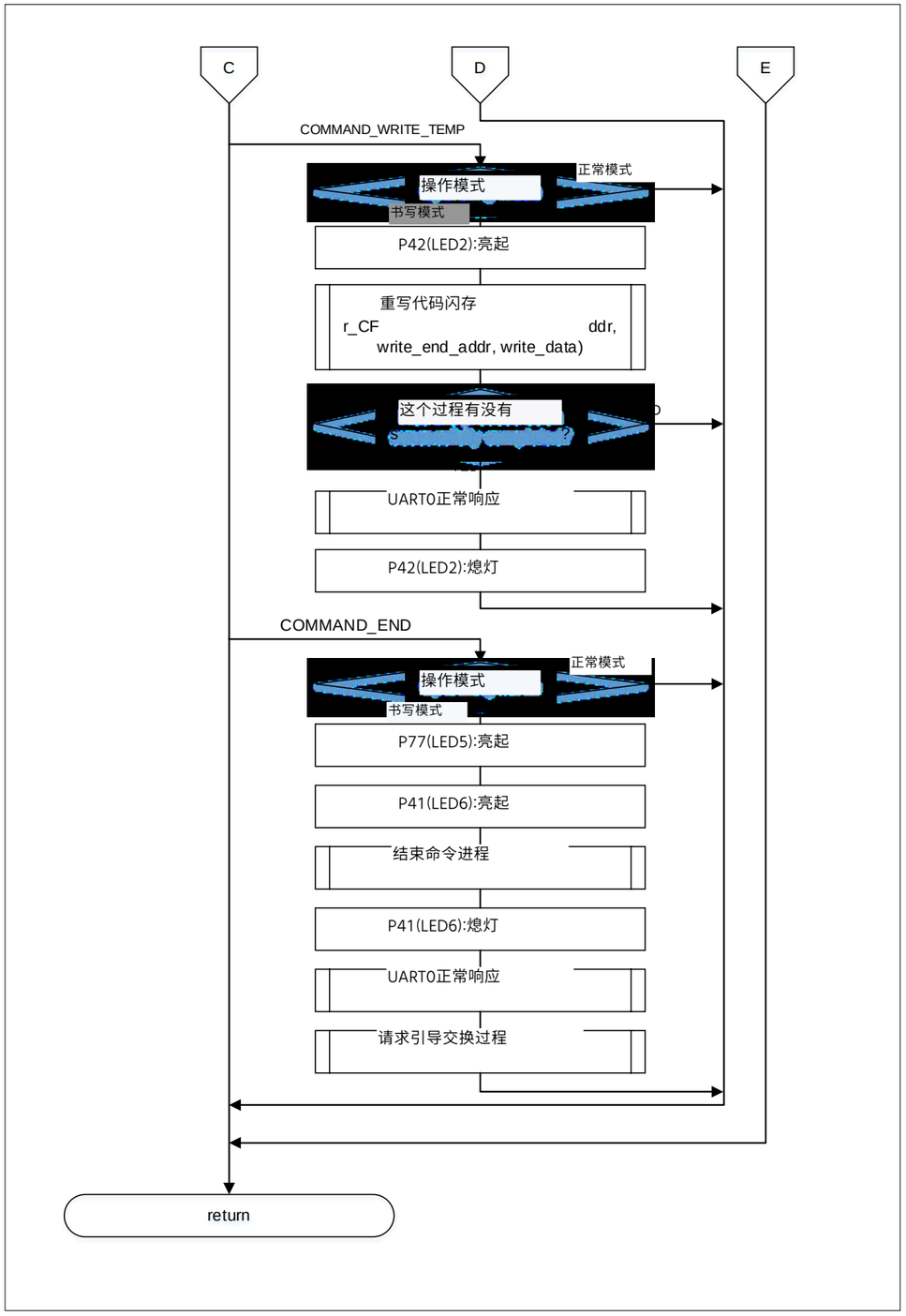


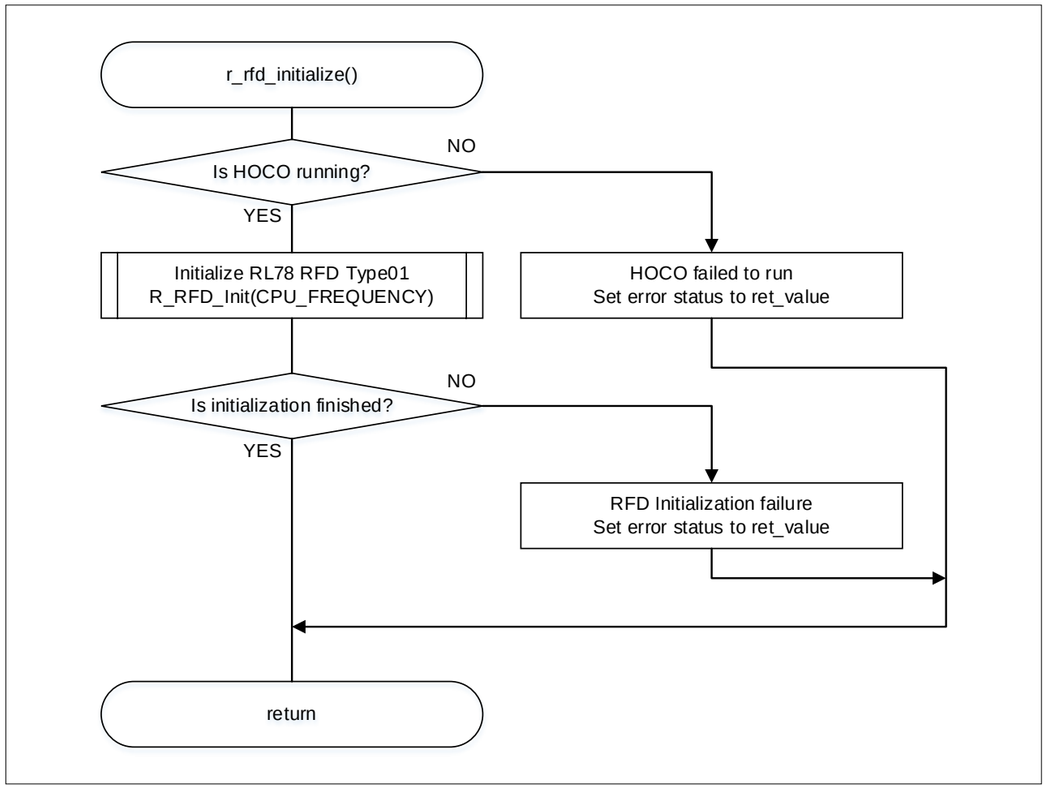
图4-5接收和运行固件更新命令的处理(3/3)



4.11.3 Initialization processing for RFD RL78 Type01

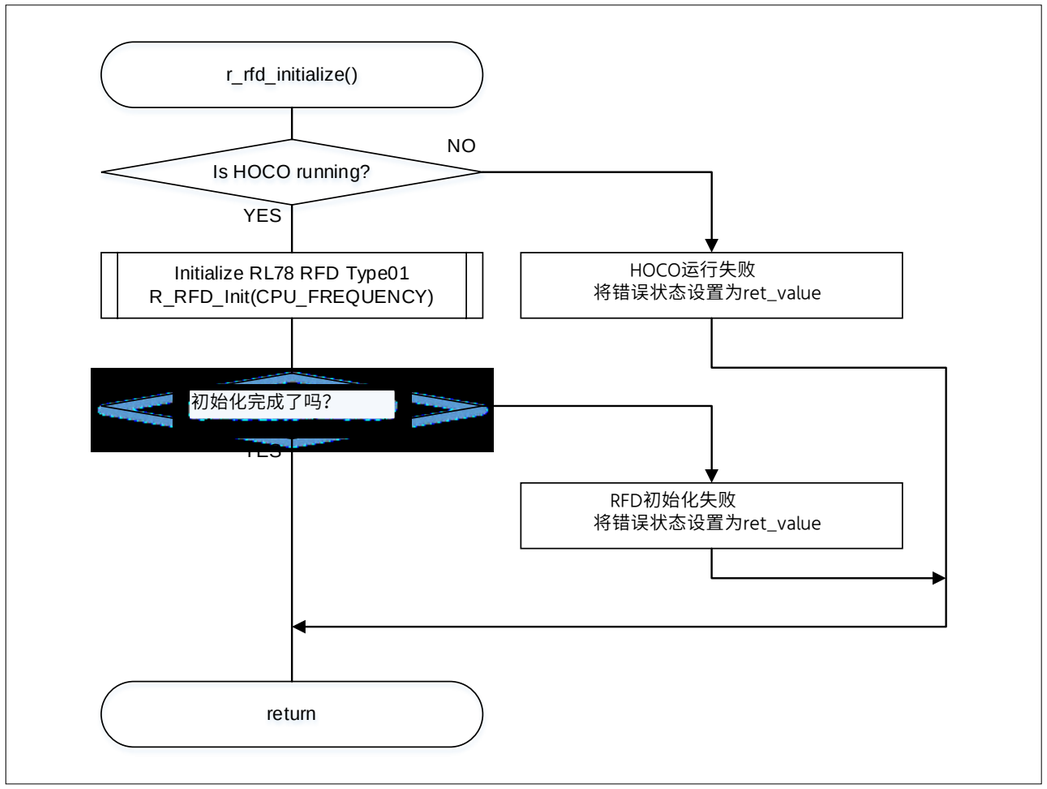
Figure 4-6 shows the flowchart of Initialization processing for RFD RL78 Type01.

Figure 4-6 Initialization processing for RFD RL78 Type01



4.11.3用于RFDRL78Type01的初始化处理图4-6表示用于RFDRL78Type01的初始化处理的流程图。

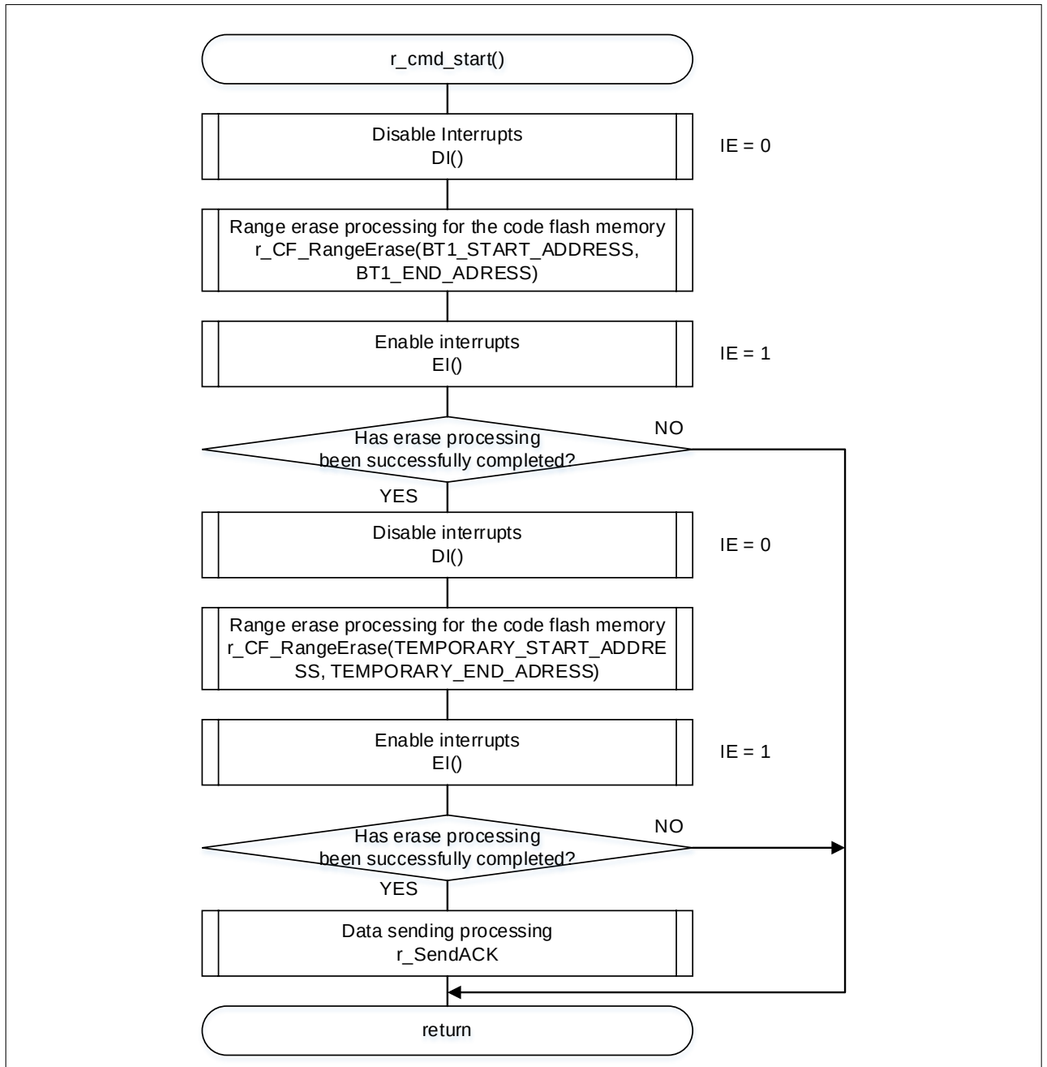
图4-6RFDRl78Type01的初始化处理



4.11.4 START command processing

Figure 4-7 shows the flowchart of START command processing

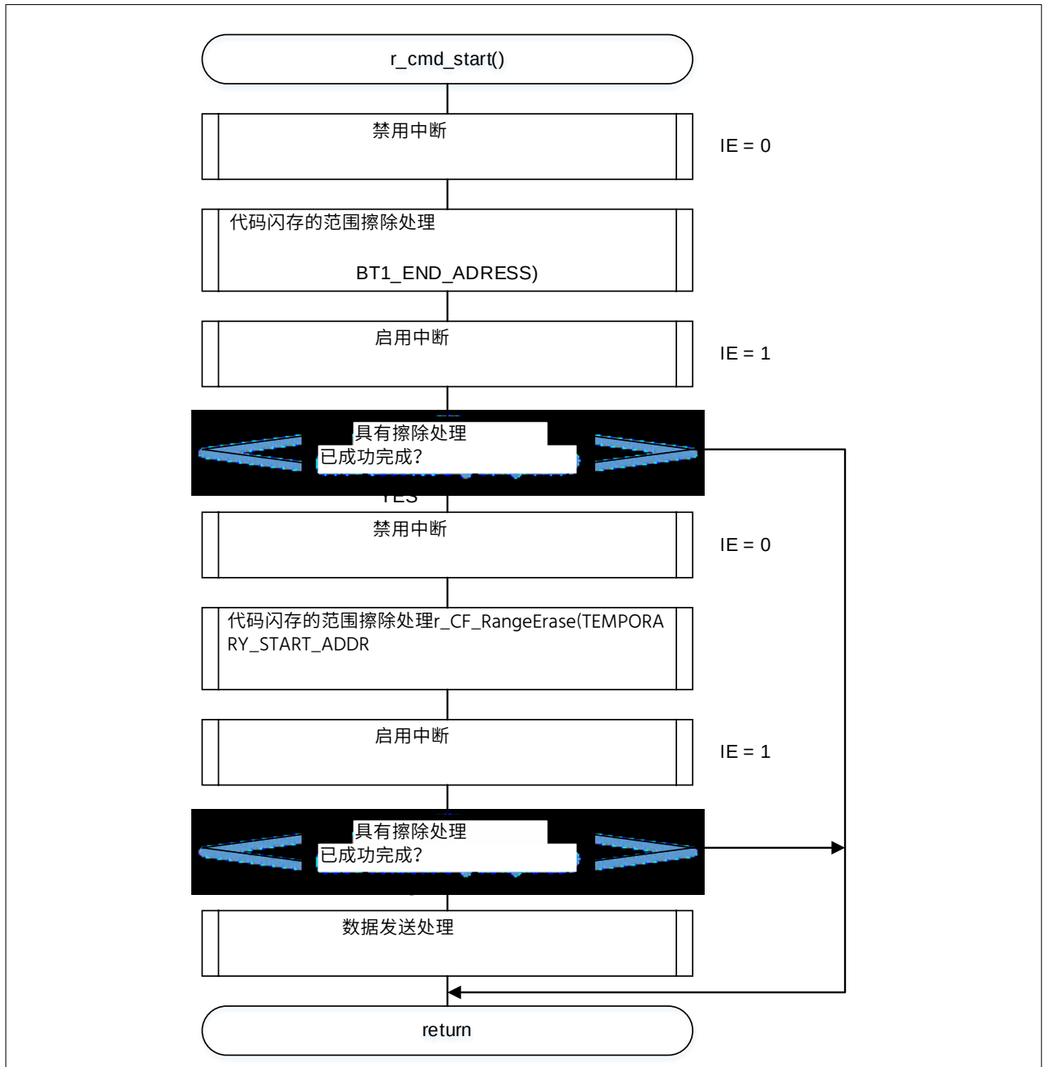
Figure 4-7 START command processing



4.11.4启动命令处理

图4-7示出了启动命令处理的流程图

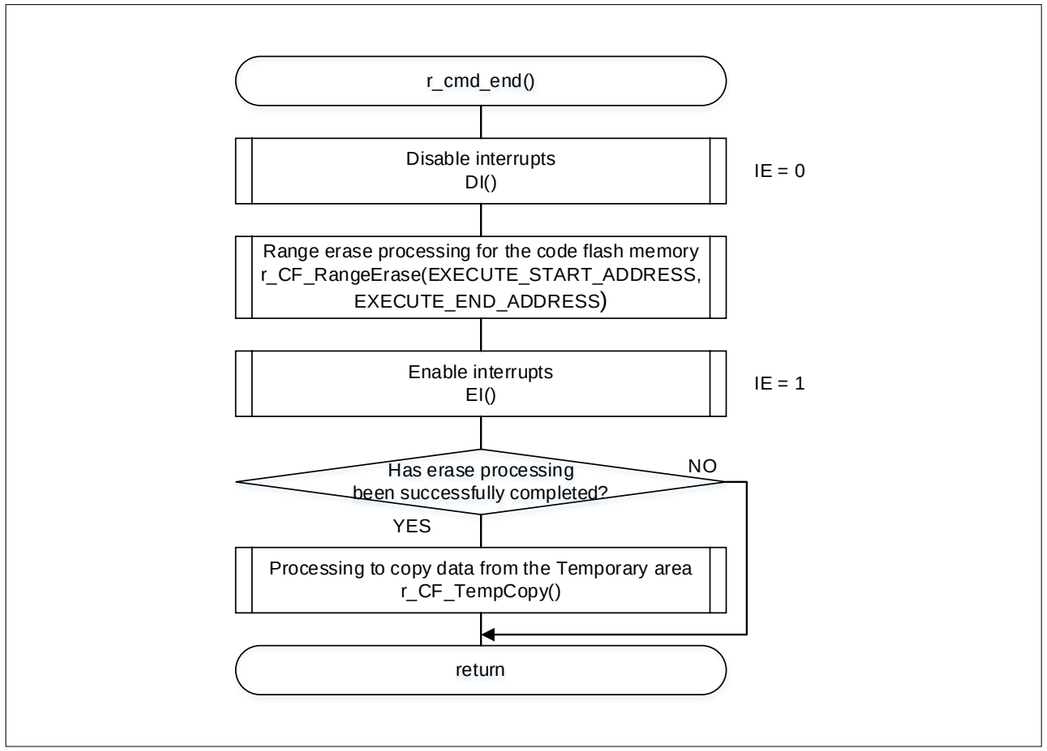
图4-7启动命令处理



4.11.5 END command processing

Figure 4-8 shows the flowchart of END command processing

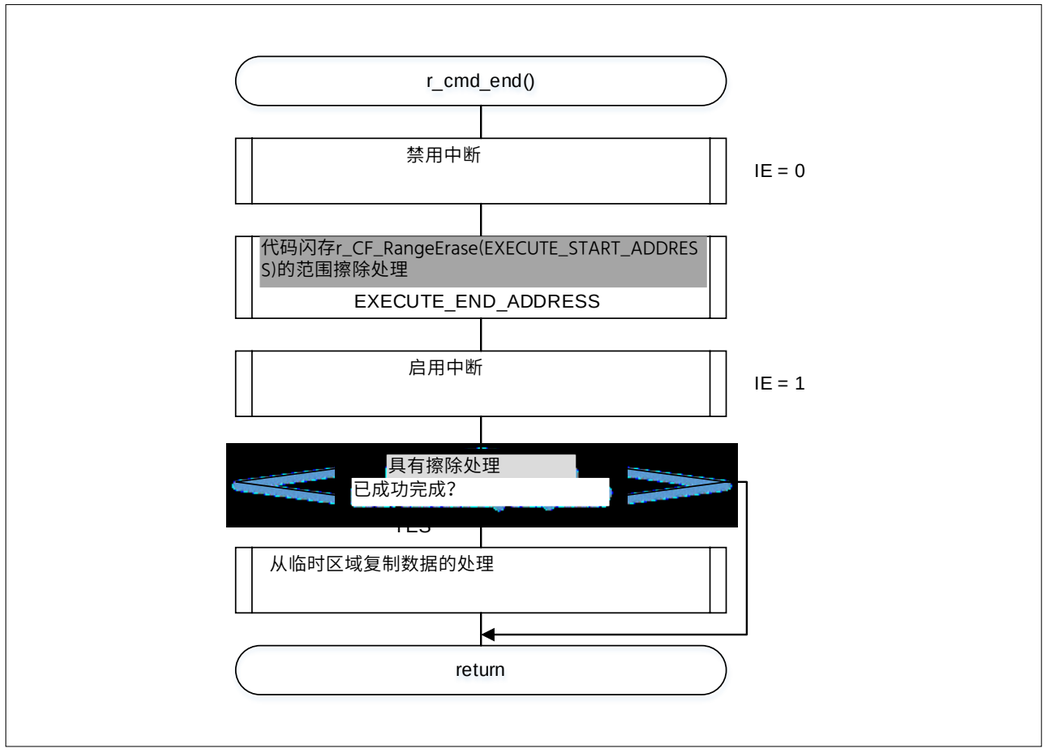
Figure 4-8 END command processing



4.11.5结束命令处理

图4-8示出了结束命令处理的流程图

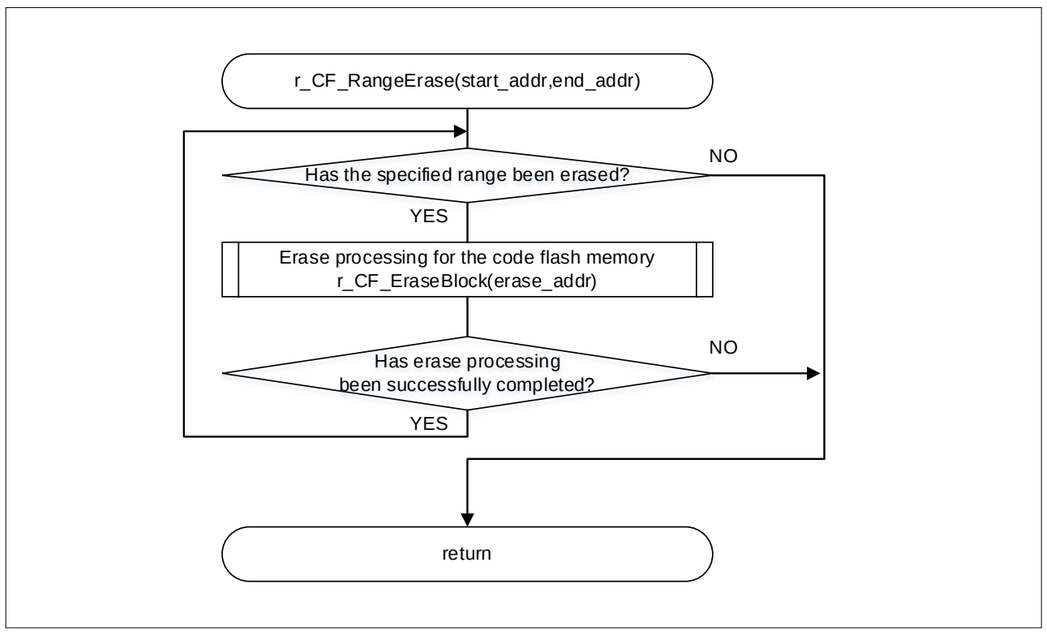
图4-8结束命令处理



4.11.6 Range erase processing for the code flash memory

Figure 4-9 shows the flowchart of range erase processing for the code flash memory

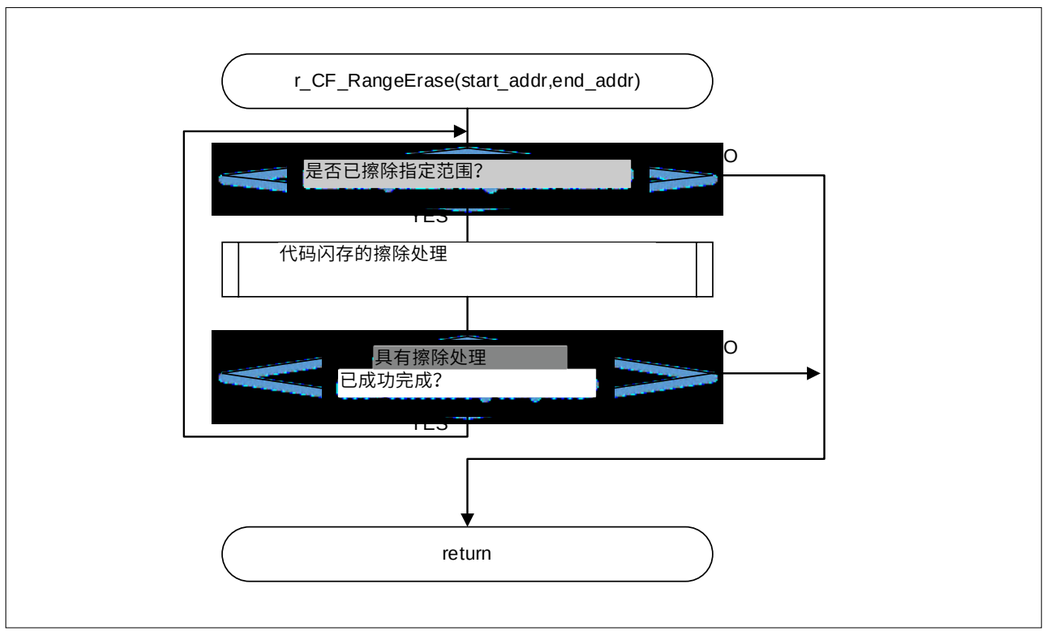
Figure 4-9 Range erase processing for the code flash memory



4.11.6代码闪存的范围擦除处理

图4-9显示了代码闪存的范围擦除处理的流程图

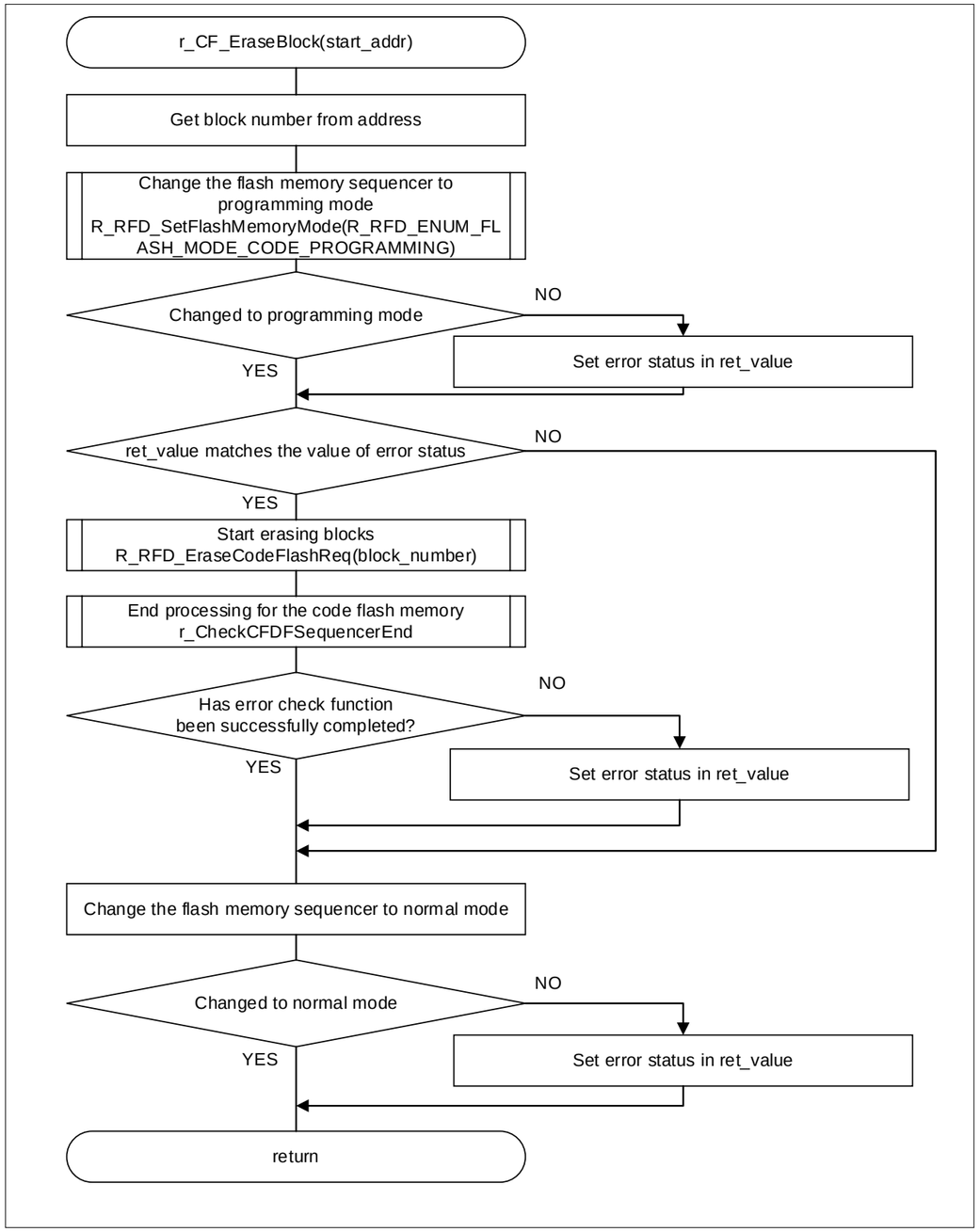
图4-9代码闪存的范围擦除处理



4.11.7 Block erase processing for the code flash memory

Figure 4-10 shows the flowchart of block erase processing for the code flash memory

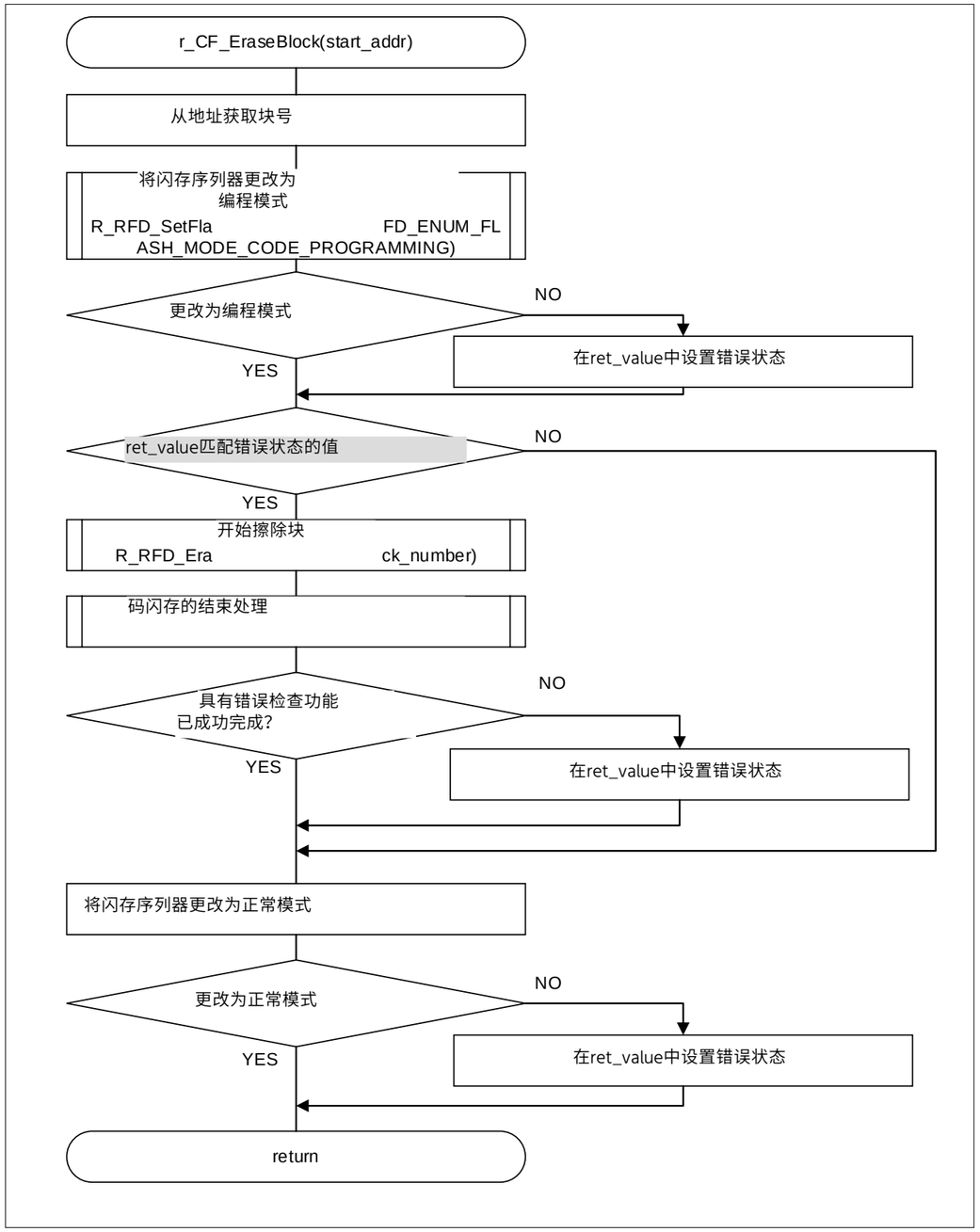
Figure 4-10 Block erase processing for the code flash memory



4.11.7代码闪存的块擦除处理

图4-10示出了用于代码闪存的块擦除处理的流程图

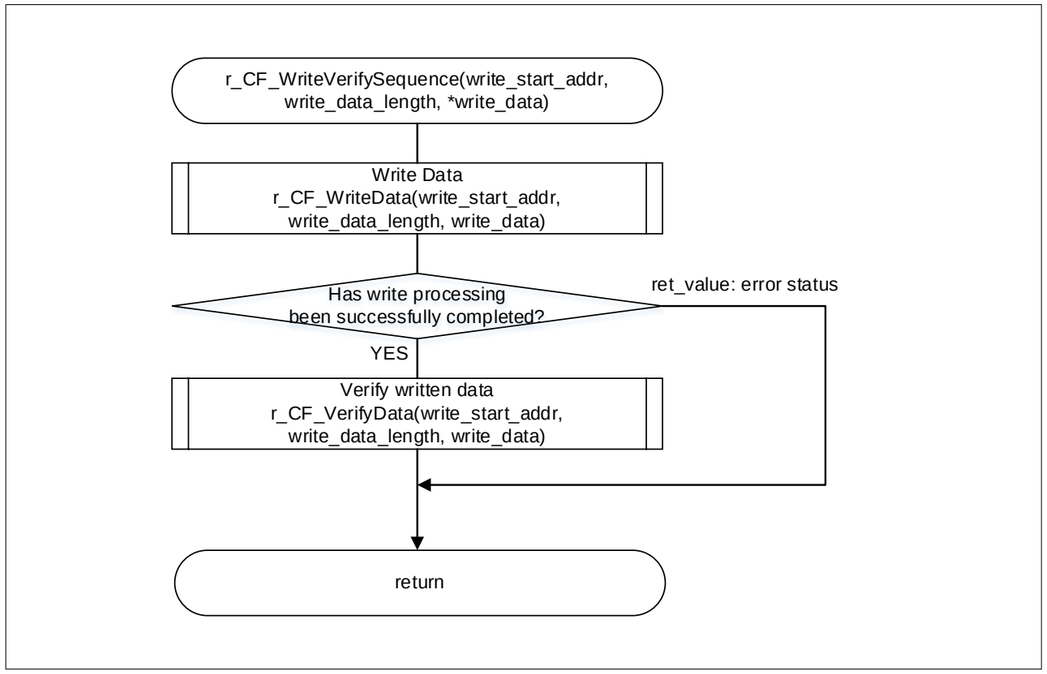
图4-10代码闪存的块擦除处理



4.11.8 Write-and-verify processing for the code flash memory

Figure 4-11 shows the flowchart of write-and-verify processing for the code flash memory

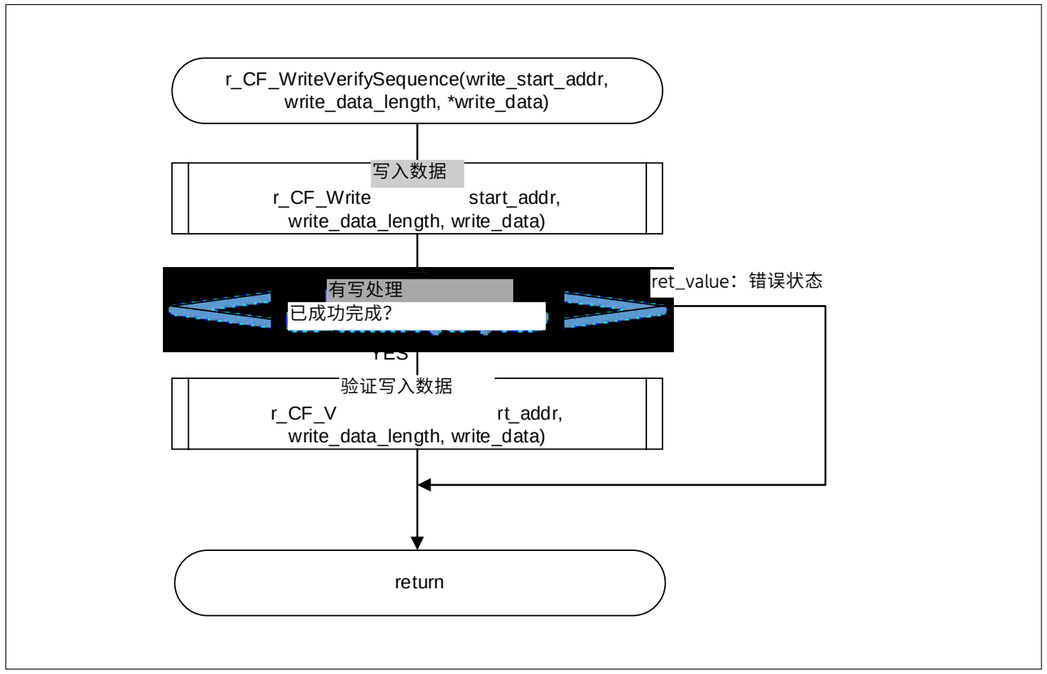
Figure 4-11 Write-and-verify processing for the code flash memory



4.11.8代码闪存的写验证处理

图4-11显示了代码闪存的写和验证处理流程图

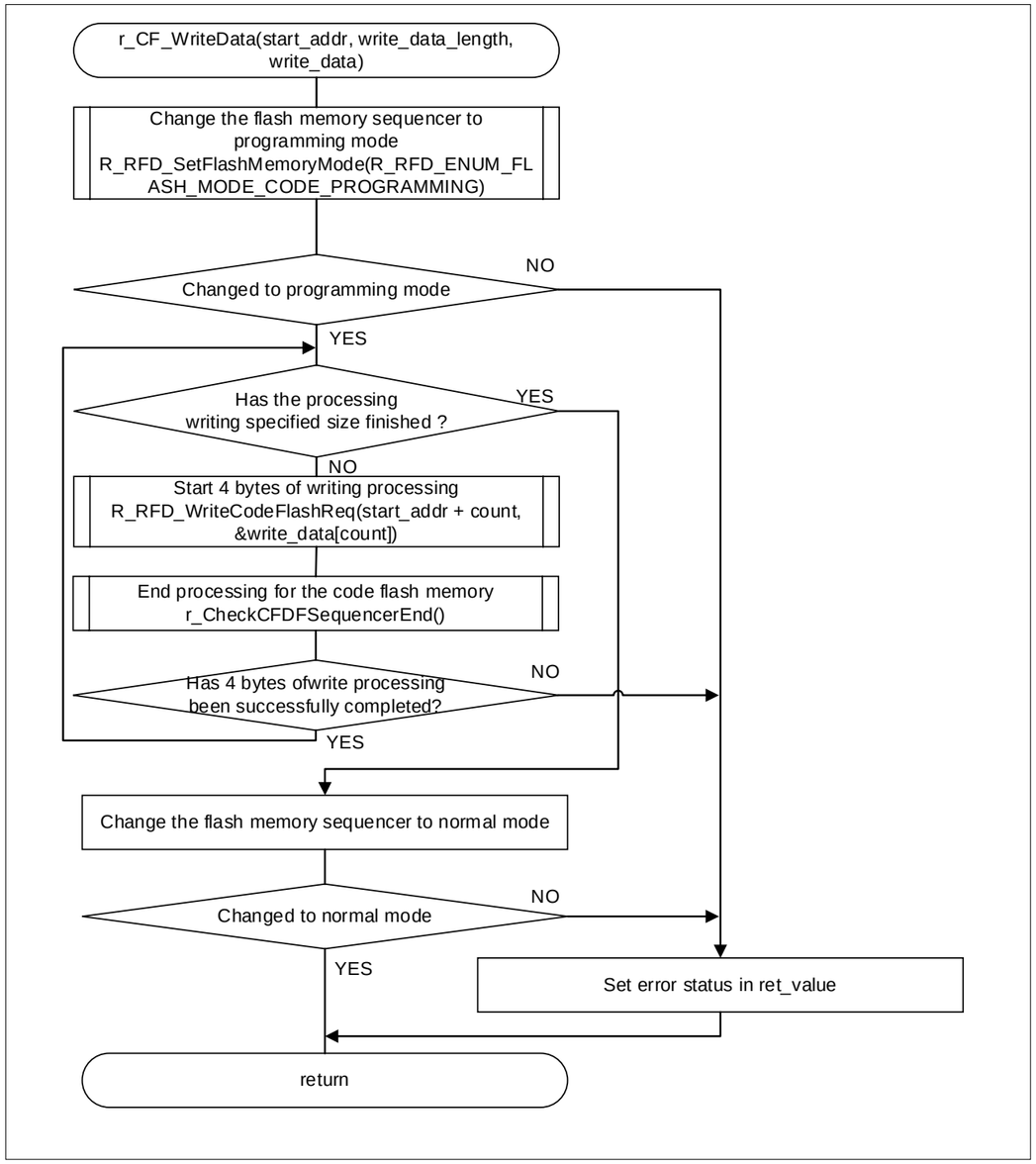
图4-11代码闪存的写入和验证处理



4.11.9 Write processing for the code flash memory

Figure 4-12 shows the flowchart of write processing for the code flash memory

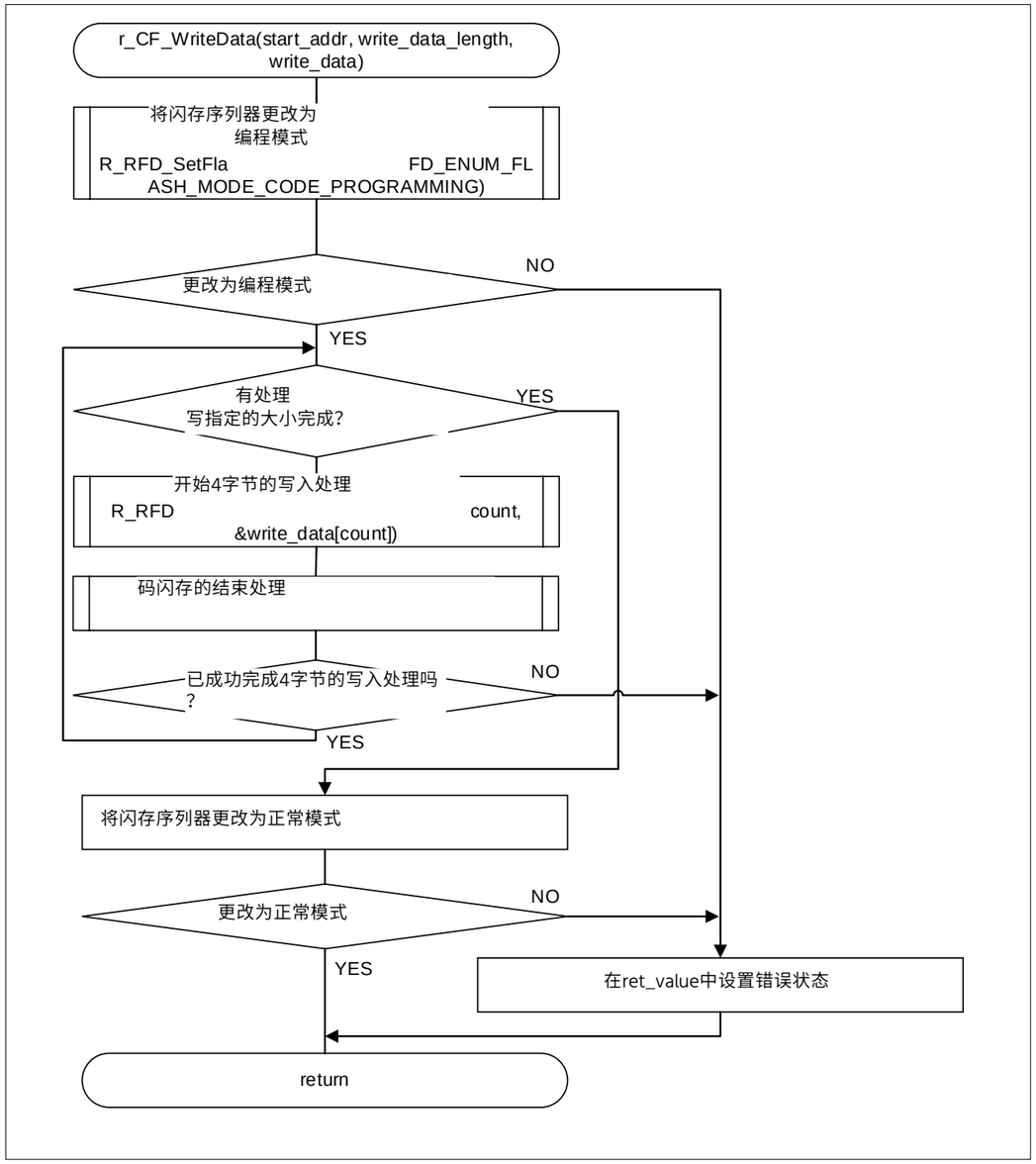
Figure 4-12 Write processing for the code flash memory



4.11.9代码闪存的写入处理

图4-12表示代码闪存的写入处理的流程图

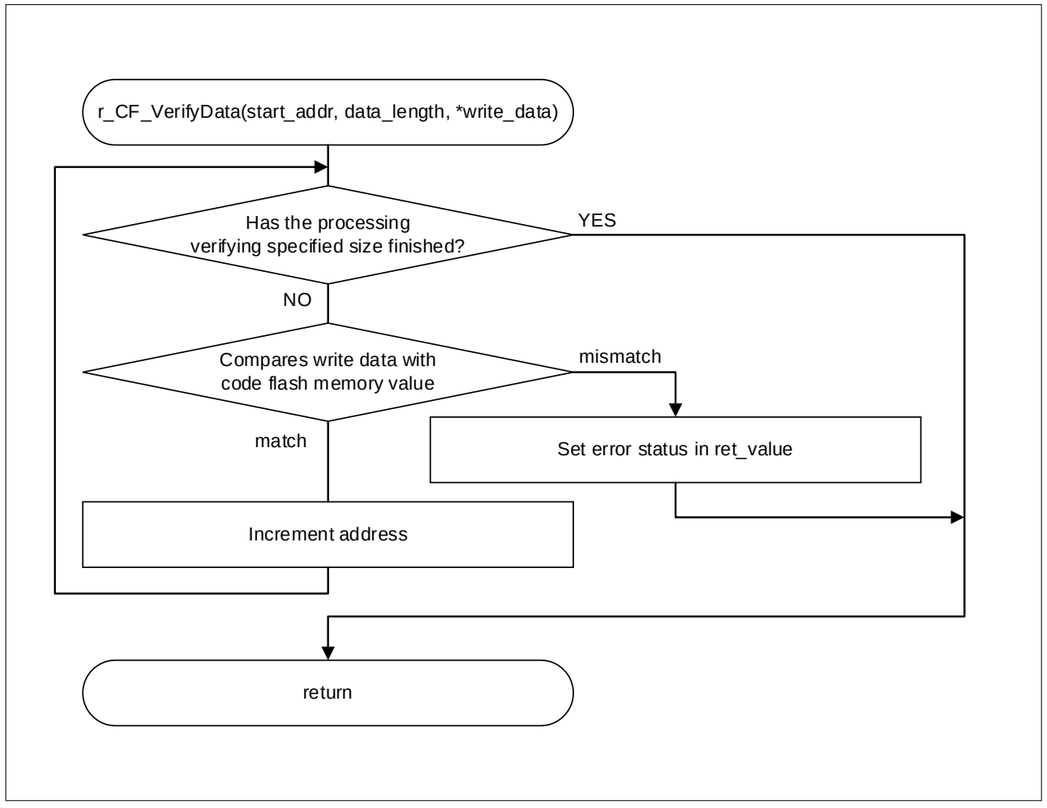
图4-12代码闪存的写入处理



4.11.10 Verify processing for the code flash memory

Figure 4-13 shows the flowchart of verify processing for the code flash memory

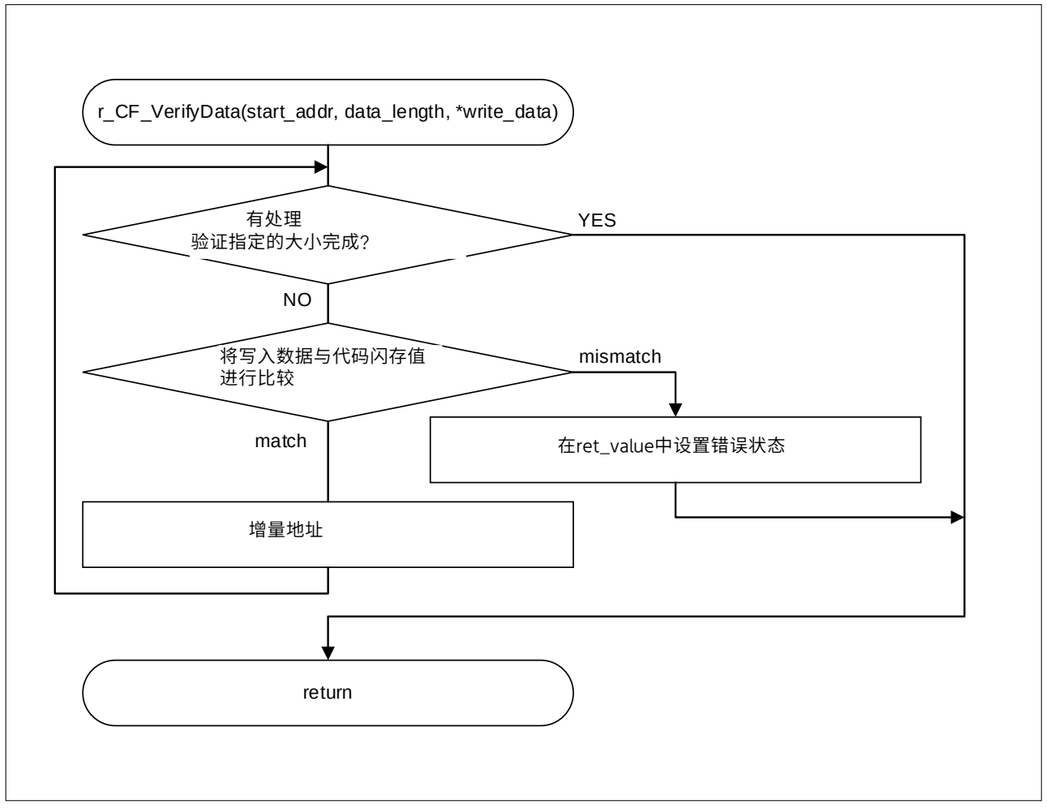
Figure 4-13 Verify processing for the code flash memory



4.11.10 验证代码闪存的处理

图4-13显示了代码闪存的验证处理流程图

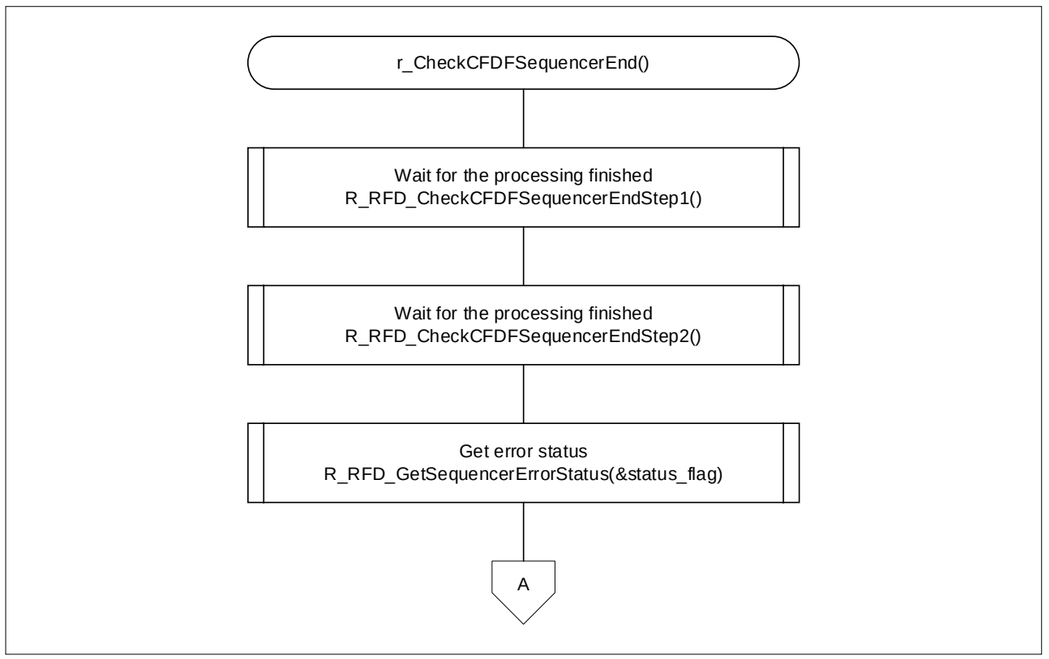
图4-13代码闪存的验证处理



4.11.11 Sequence end processing for the code flash memory

Figure 4-14 and Figure 4-15 shows the flowchart of sequence end processing for the code flash memory

Figure 4-14 Sequence end processing for the code flash memory (1/2)



4.11.11 码闪存的序列结束处理

图4-14和图4-15示出了用于代码闪存的序列结束处理的流程图

图4-14代码闪存的序列结束处理(12)

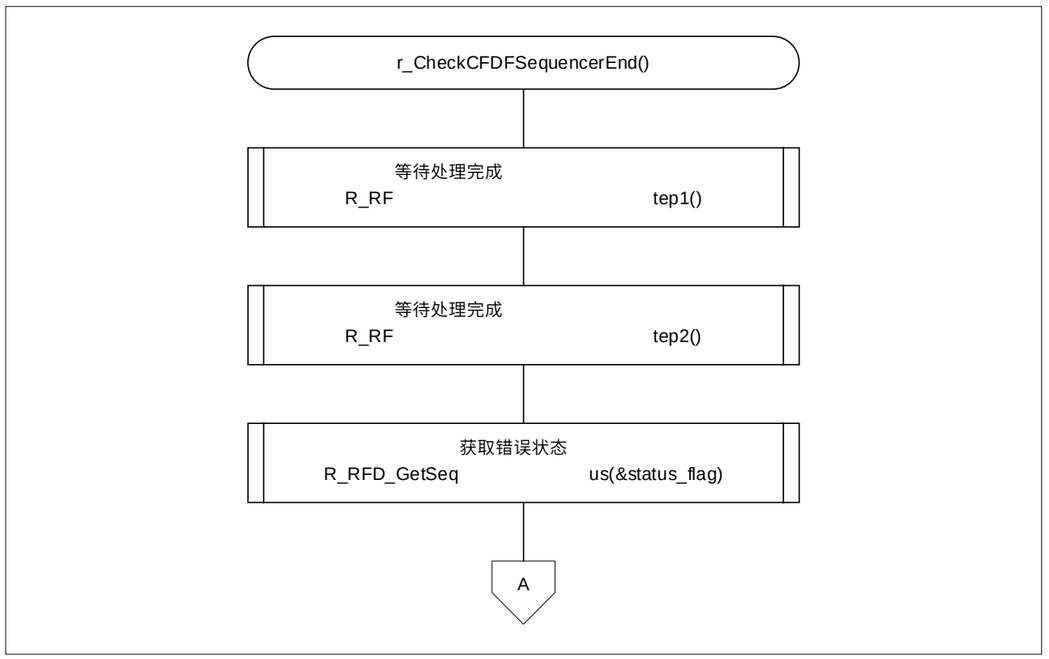


Figure 4-15 Sequence end processing for the code flash memory (2/2)

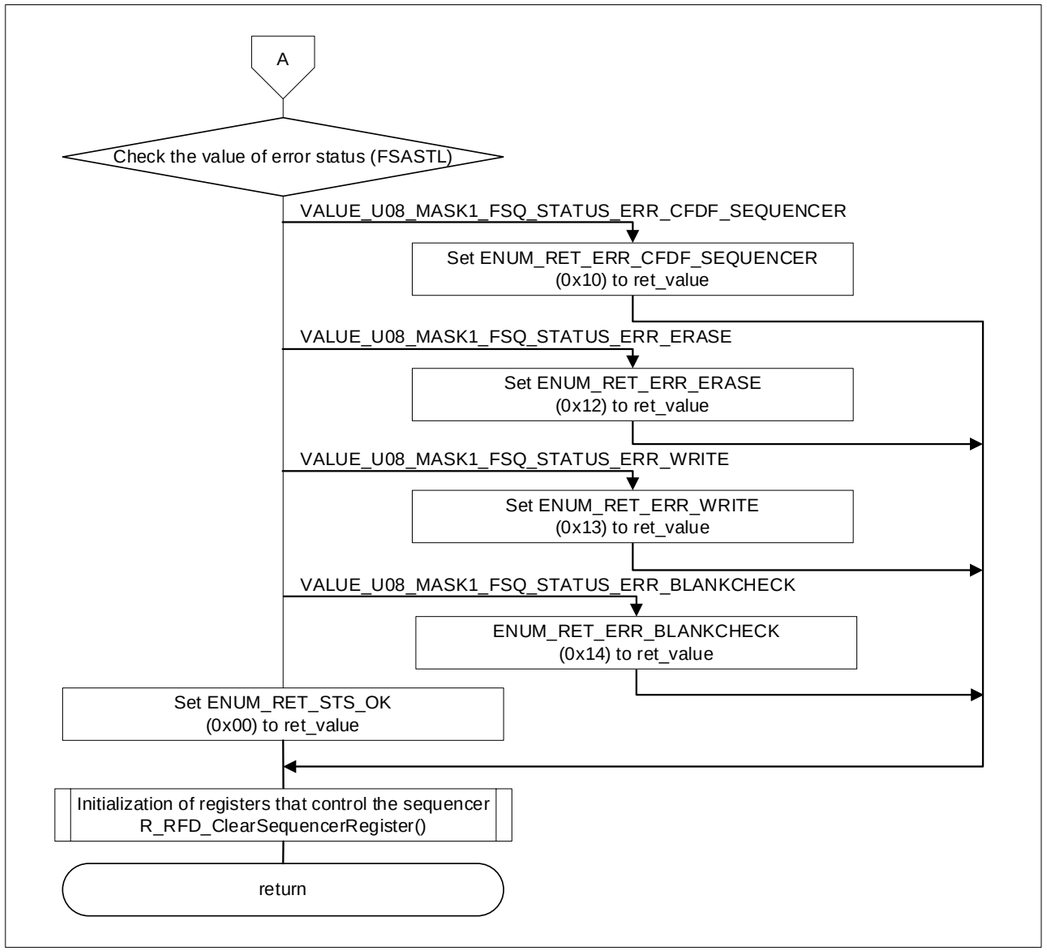
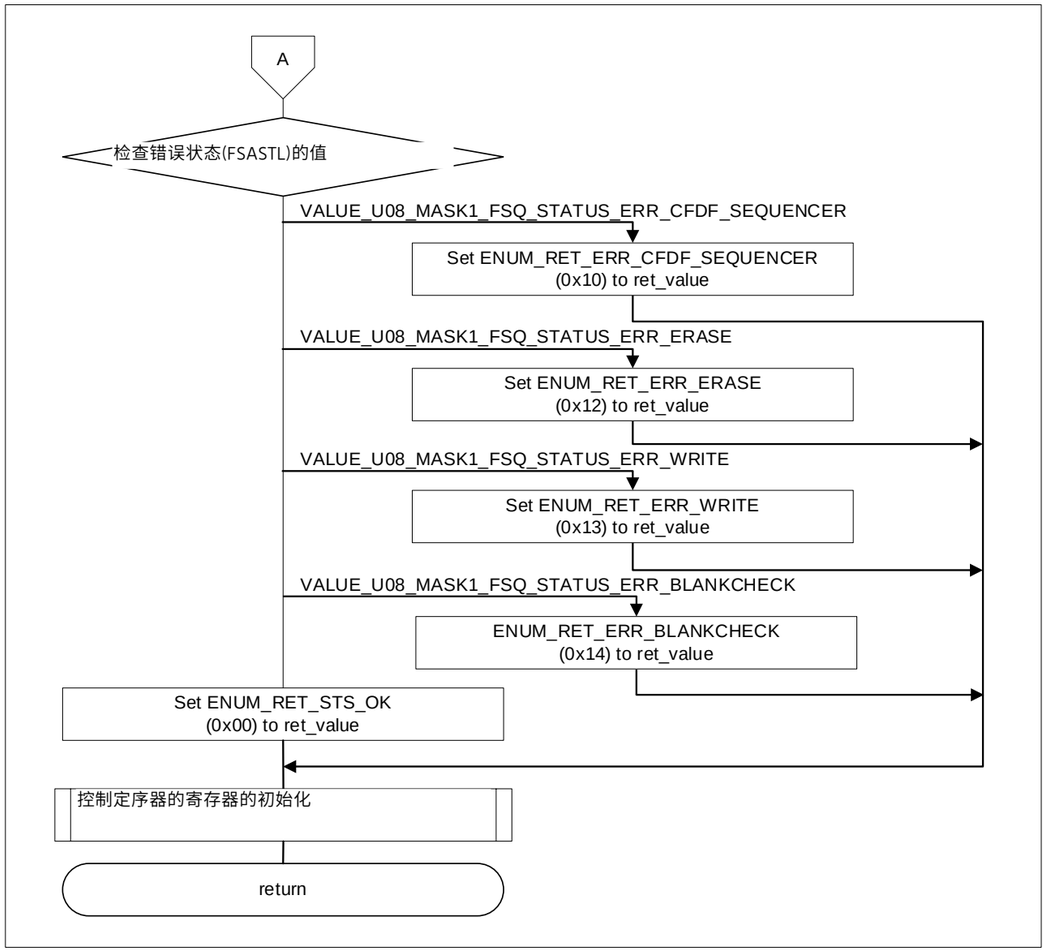


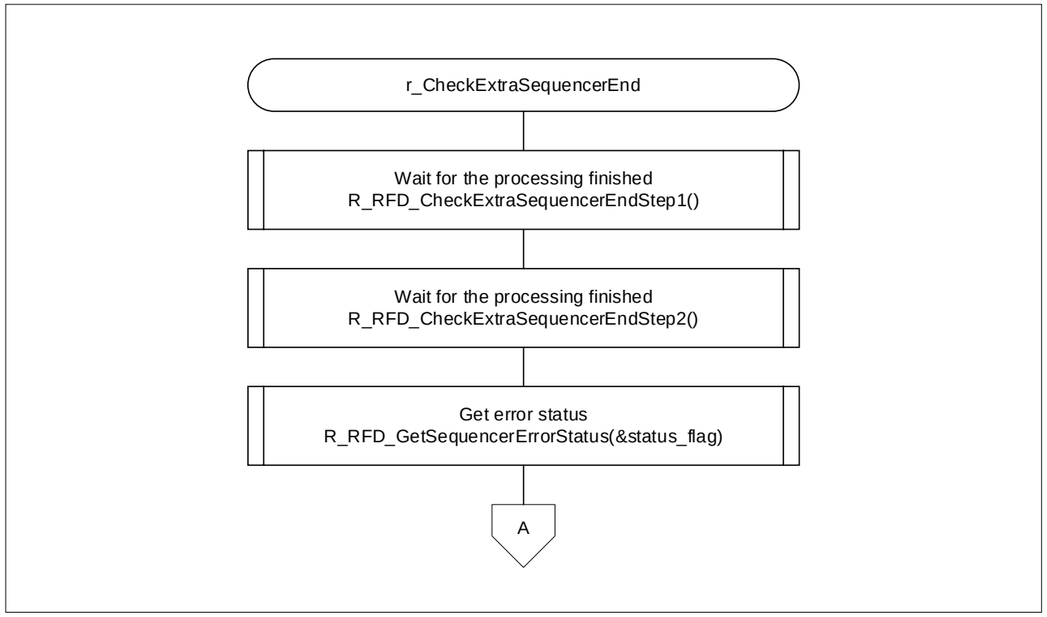
图4-15代码闪存的序列结束处理(2/2)



4.11.12 Sequence end processing for the extra area

Figure 4-16 and Figure 4-17 shows the flowchart of sequence end processing for the extra area

Figure 4-16 Sequence end processing for the extra area (1/2)



4.11.12 额外区域的序列结束处理

图4-16和图4-17示出了用于额外区域的序列结束处理的流程图

图4-16额外区域的序列结束处理(12)

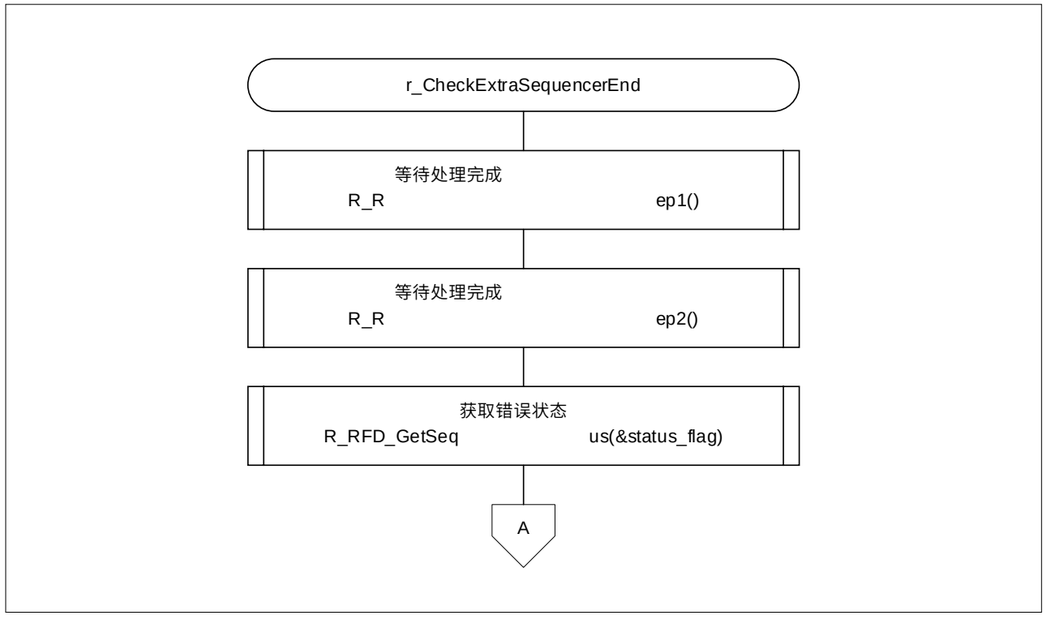


Figure 4-17 Sequence end processing for the extra area (2/2)

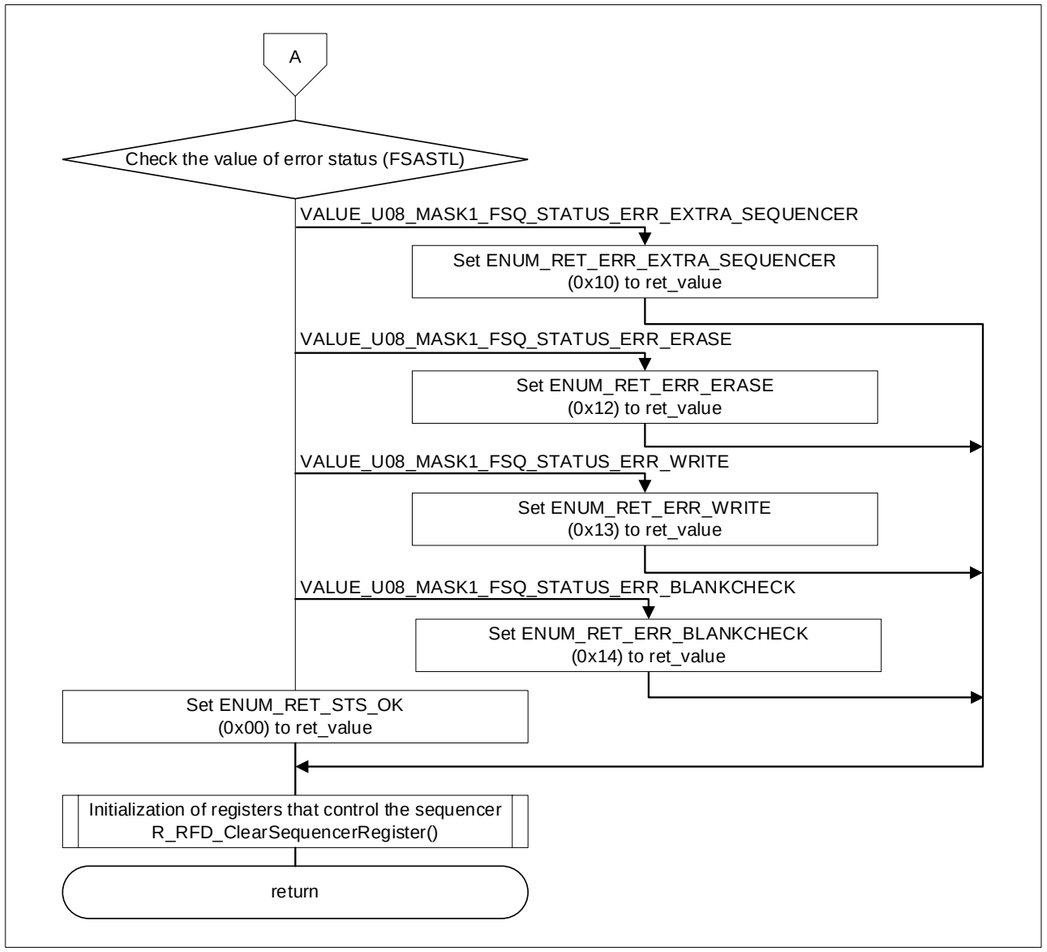
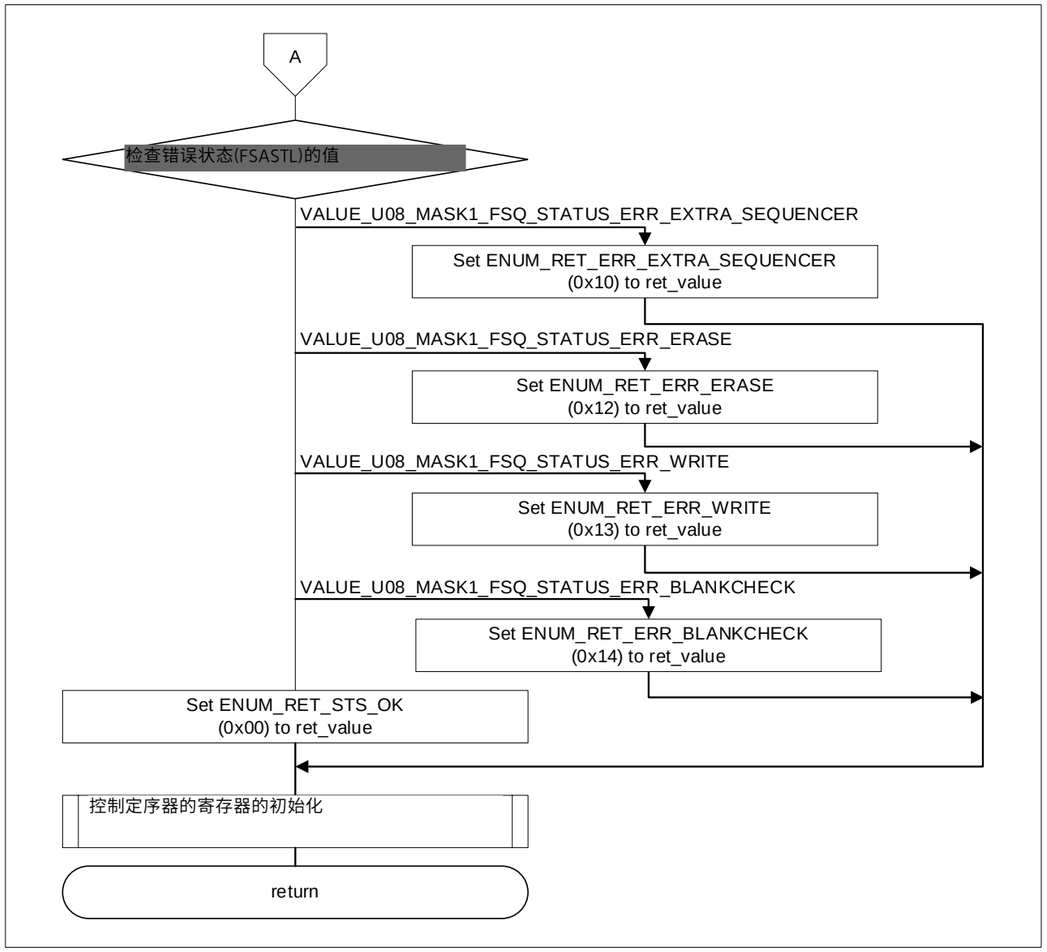


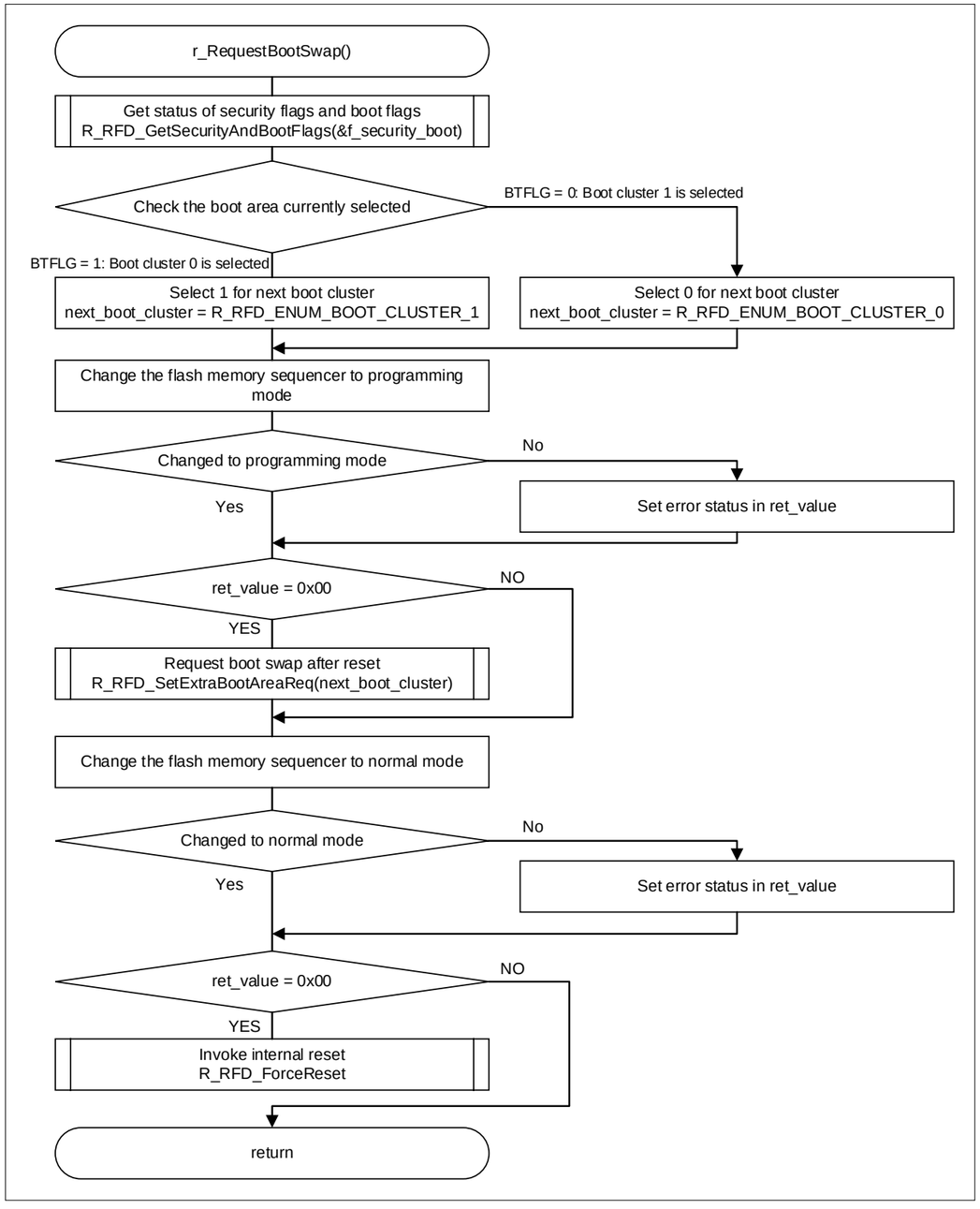
图4–17额外区域的序列结束处理(22)



4.11.13 Boot swapping execution processing

Figure 4-18 shows the flowchart of boot swapping execution processing

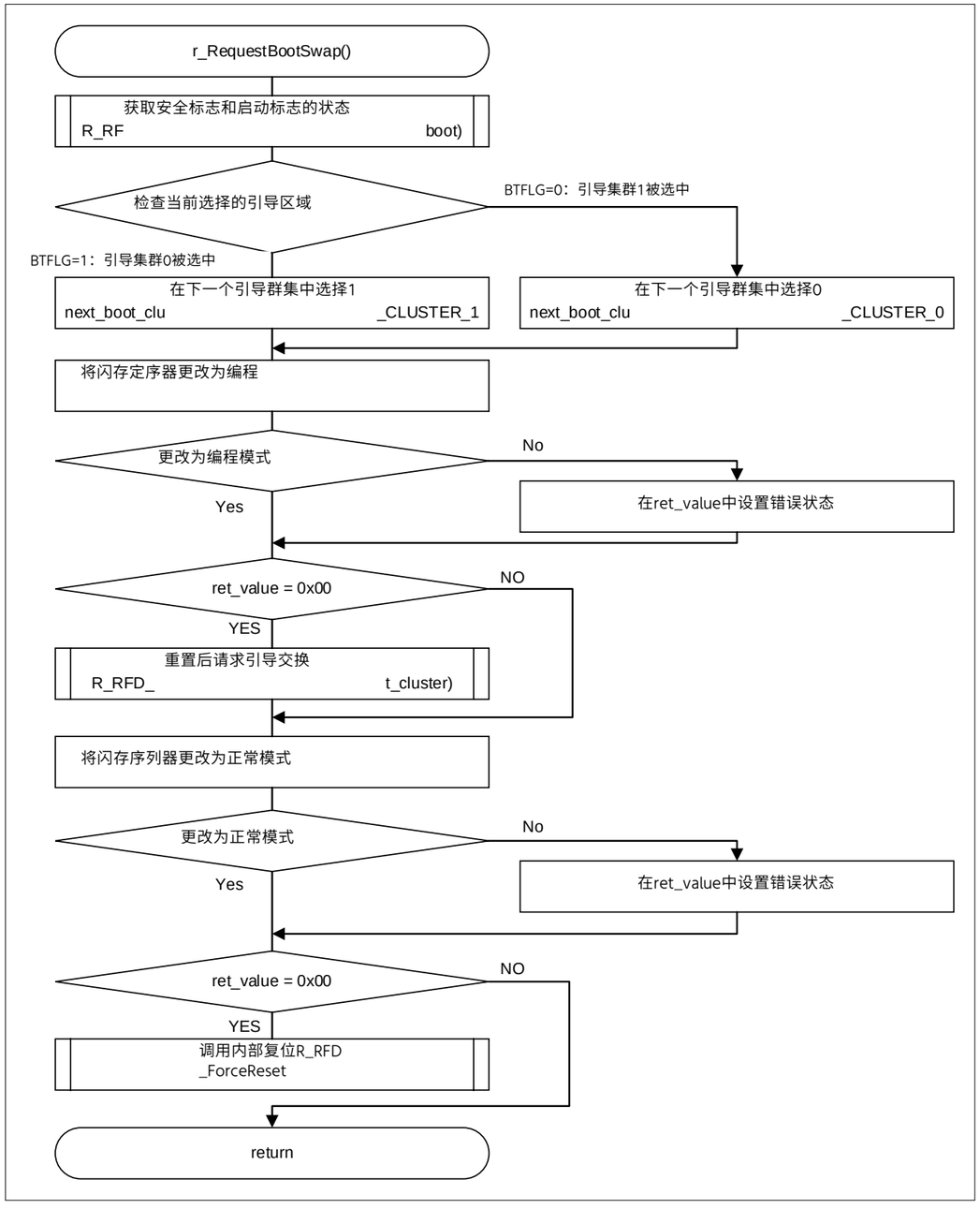
Figure 4-18 Boot swapping execution processing



4.11.13 引导交换执行处理

图4-18表示引导交换执行处理的流程图

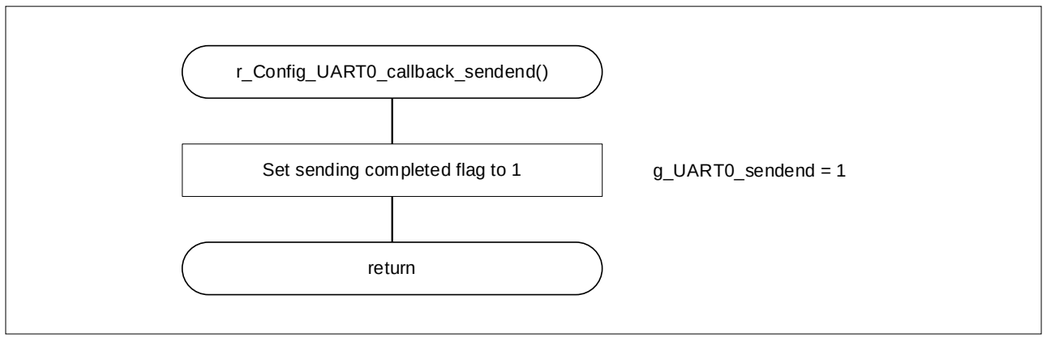
图4-18引导交换执行处理



4.11.14 Callback processing at a sending completion interrupt for UART0

Figure 4-19 shows the flowchart of callback processing at a sending completion interrupt for UART0

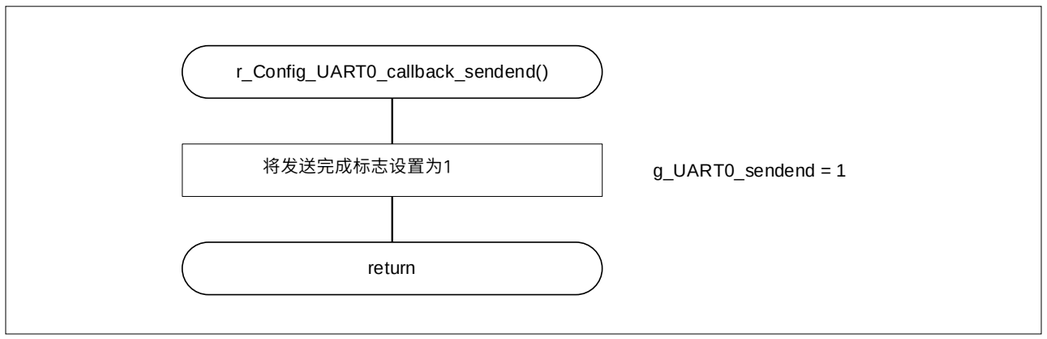
Figure 4-19 Callback processing at a sending completion interrupt for UART0



4.11.14 Uart0的发送完成中断处的回调处理

图4-19显示了uart0发送完成中断处的回调处理流程图

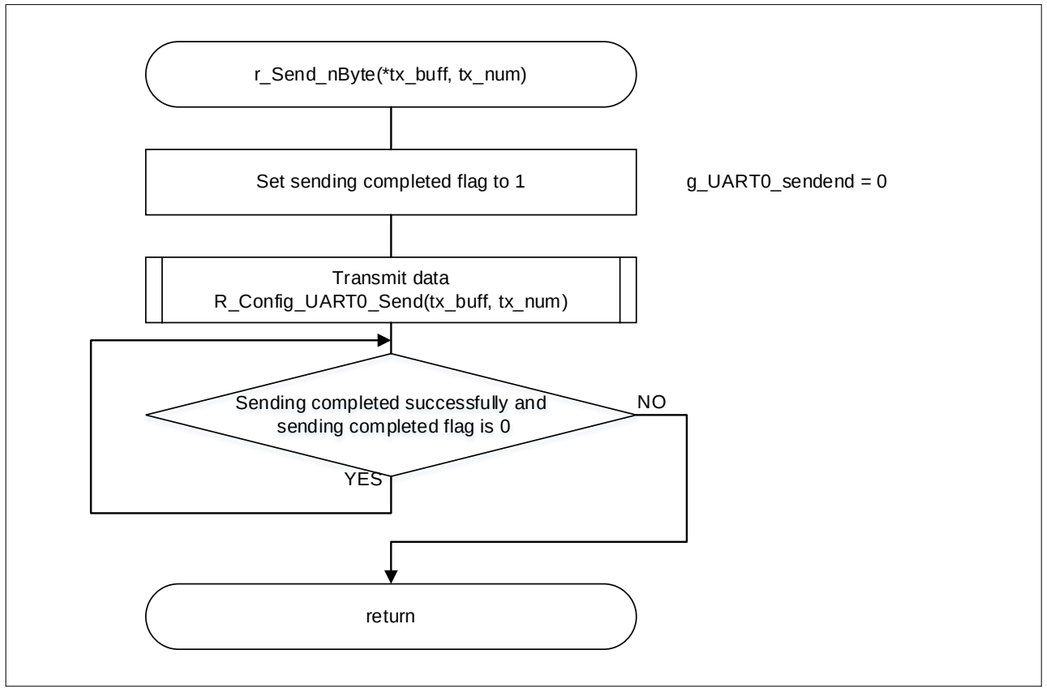
图4-19Uart0发送完成中断处的回调处理



4.11.15 Data sending processing by UART0

Figure 4-20 shows the flowchart of Data sending processing by UART0

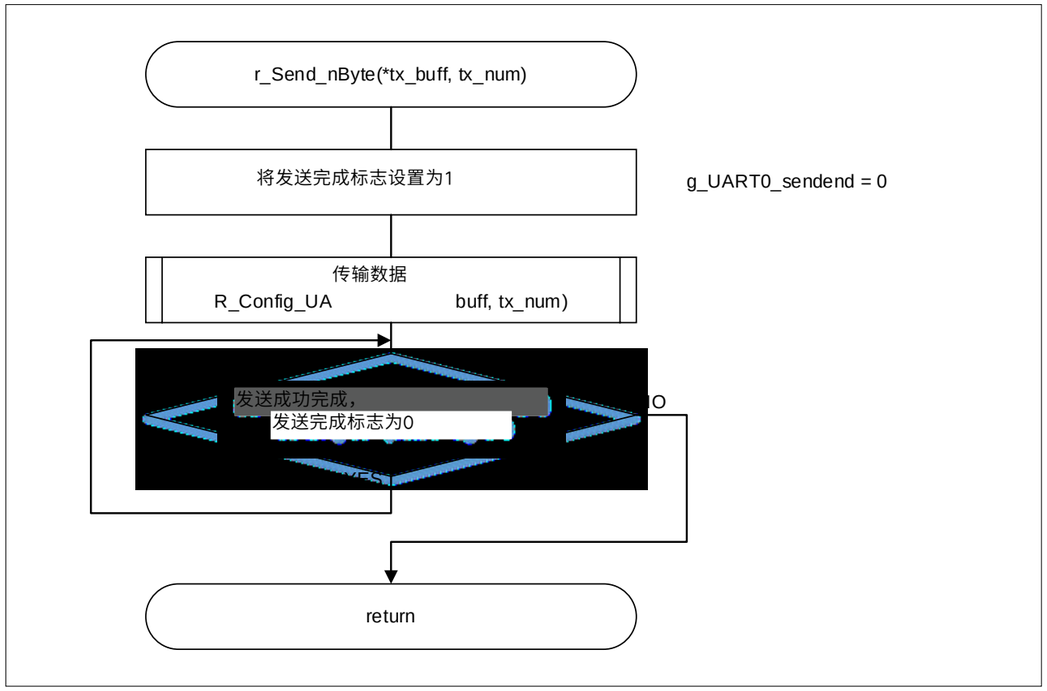
Figure 4-20 Data sending processing by UART0



4.11.15 Uart0的数据发送处理

图4-20示出了UART0进行数据发送处理的流程图

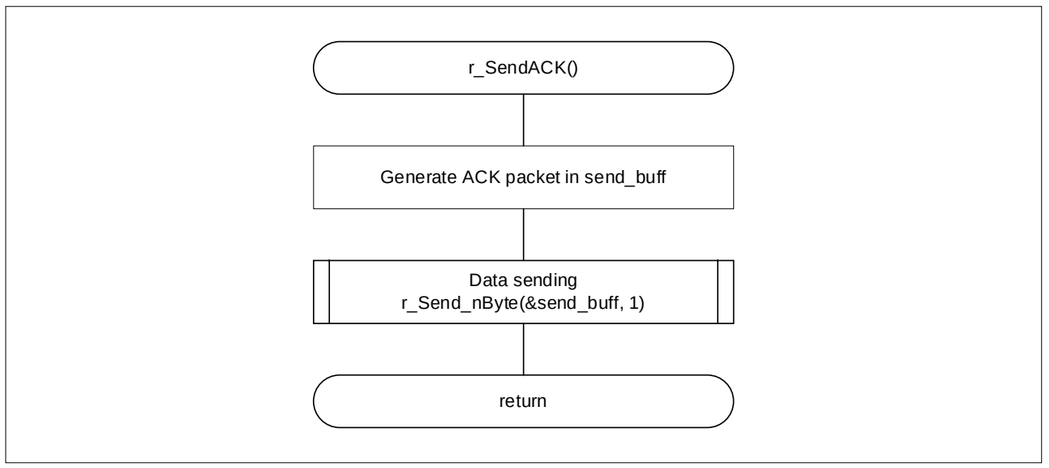
图4-20UART0的数据发送处理



4.11.16 Normal response sending processing by UART0

Figure 4-21 shows the flowchart of normal response sending processing by UART0

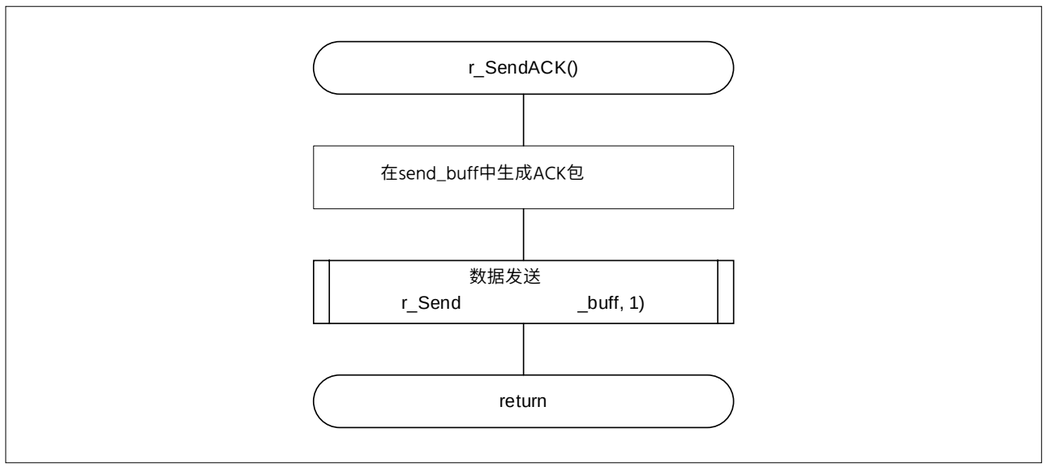
Figure 4-21 Normal response sending processing by UART0



4.11.16 Uart0的正常响应发送处理

图4-21所示为uart0正常响应发送处理的流程图

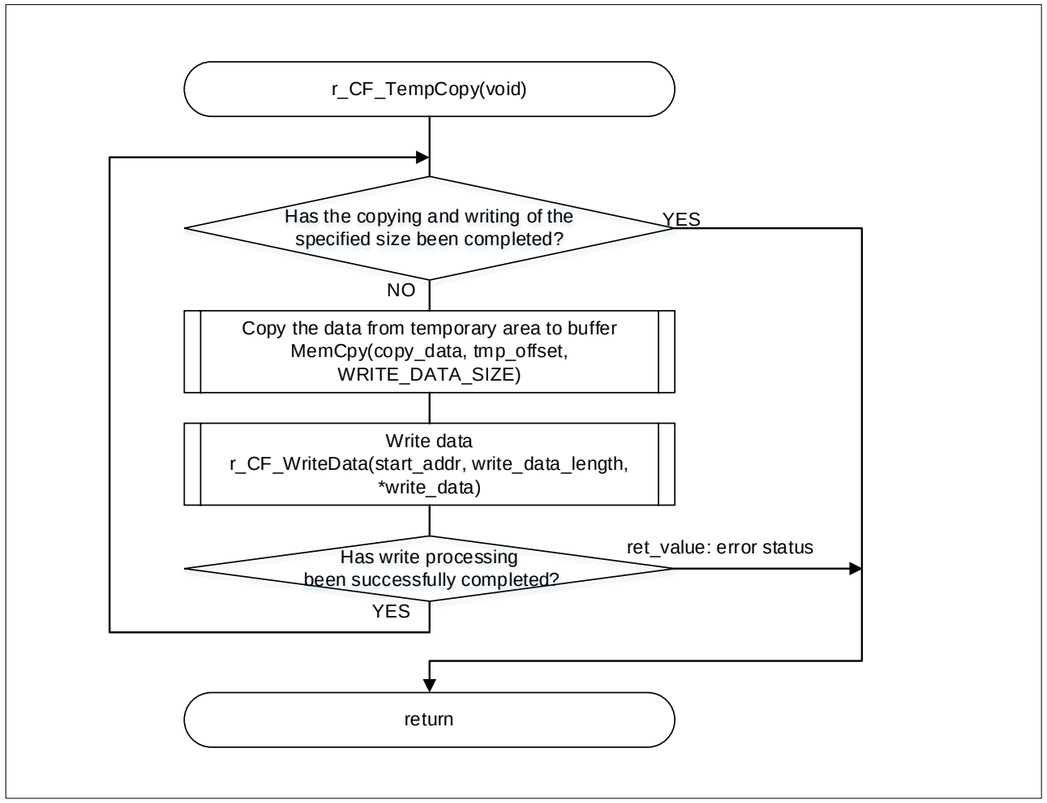
图4-21Uart0的正常响应发送处理



4.11.17 Processing to copy data from the Temporary area

Figure 4-22 shows the flowchart of processing to copy data from the Temporary area

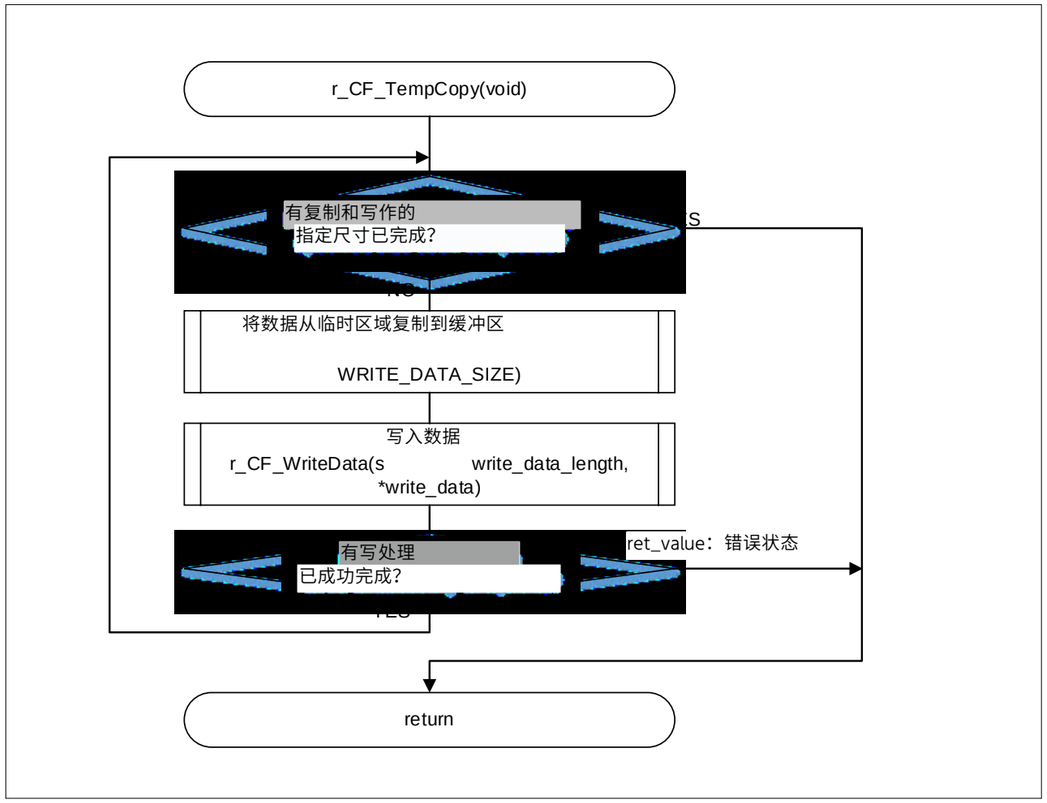
Figure 4-22 Processing to copy data from the Temporary area



4.11.17 从临时区域复制数据的处理

图4-22示出了从临时区域复制数据的处理的流程图

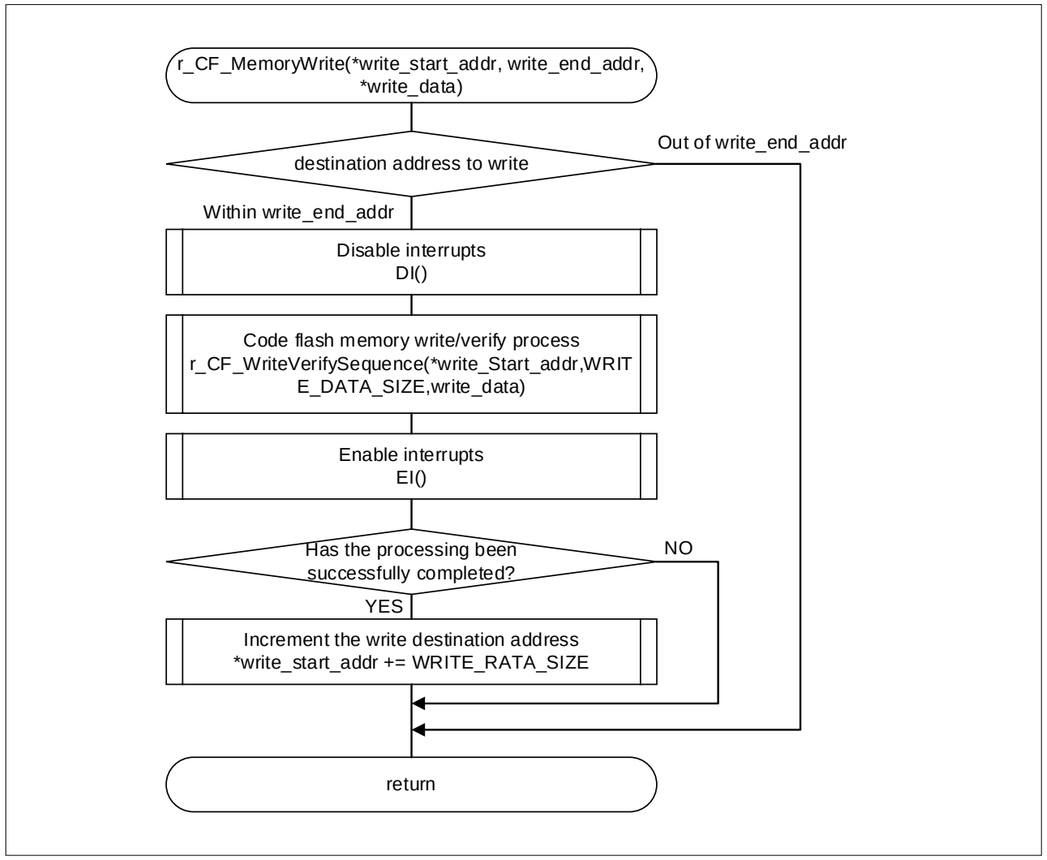
图4-22从临时区域复制数据的处理



4.11.18 Processing to reprogram the code flash memory

Figure 4-23 shows the flowchart of processing to reprogram the code flash memory

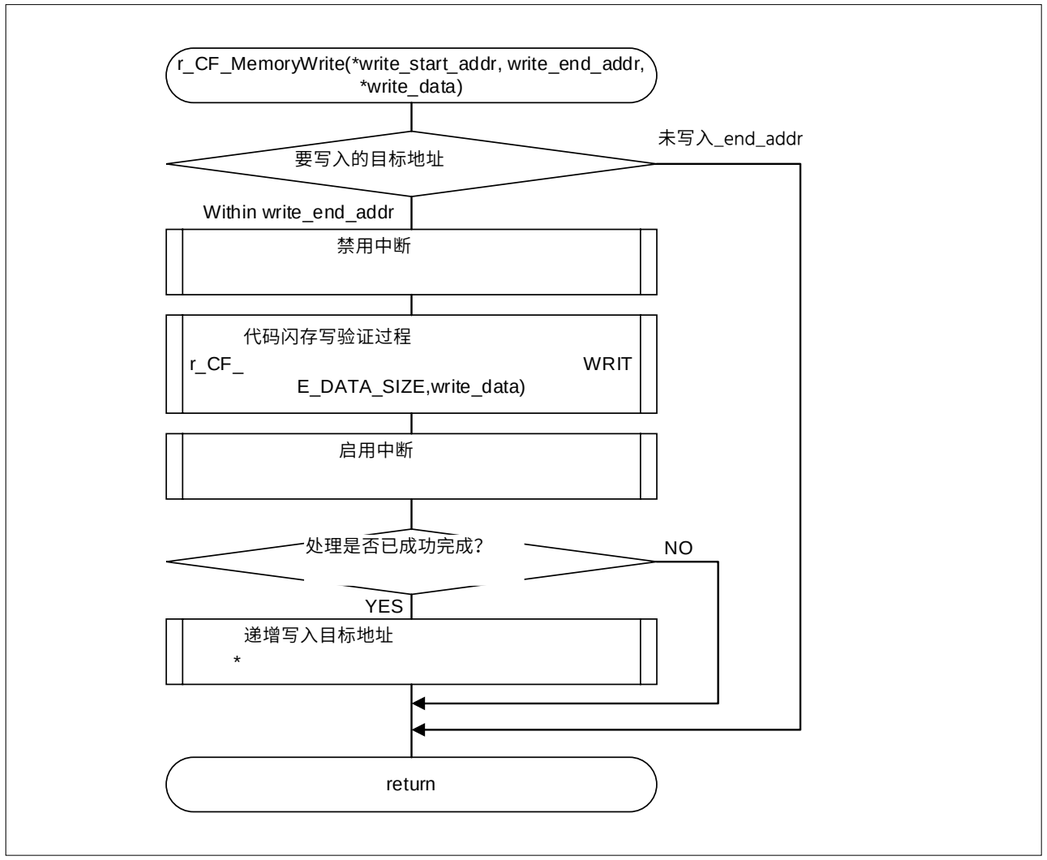
Figure 4-23 Processing to reprogram the code flash memory



4.11.18 处理以重新编程代码闪存

图4-23示出了对代码闪存进行重新编程的处理的流程图

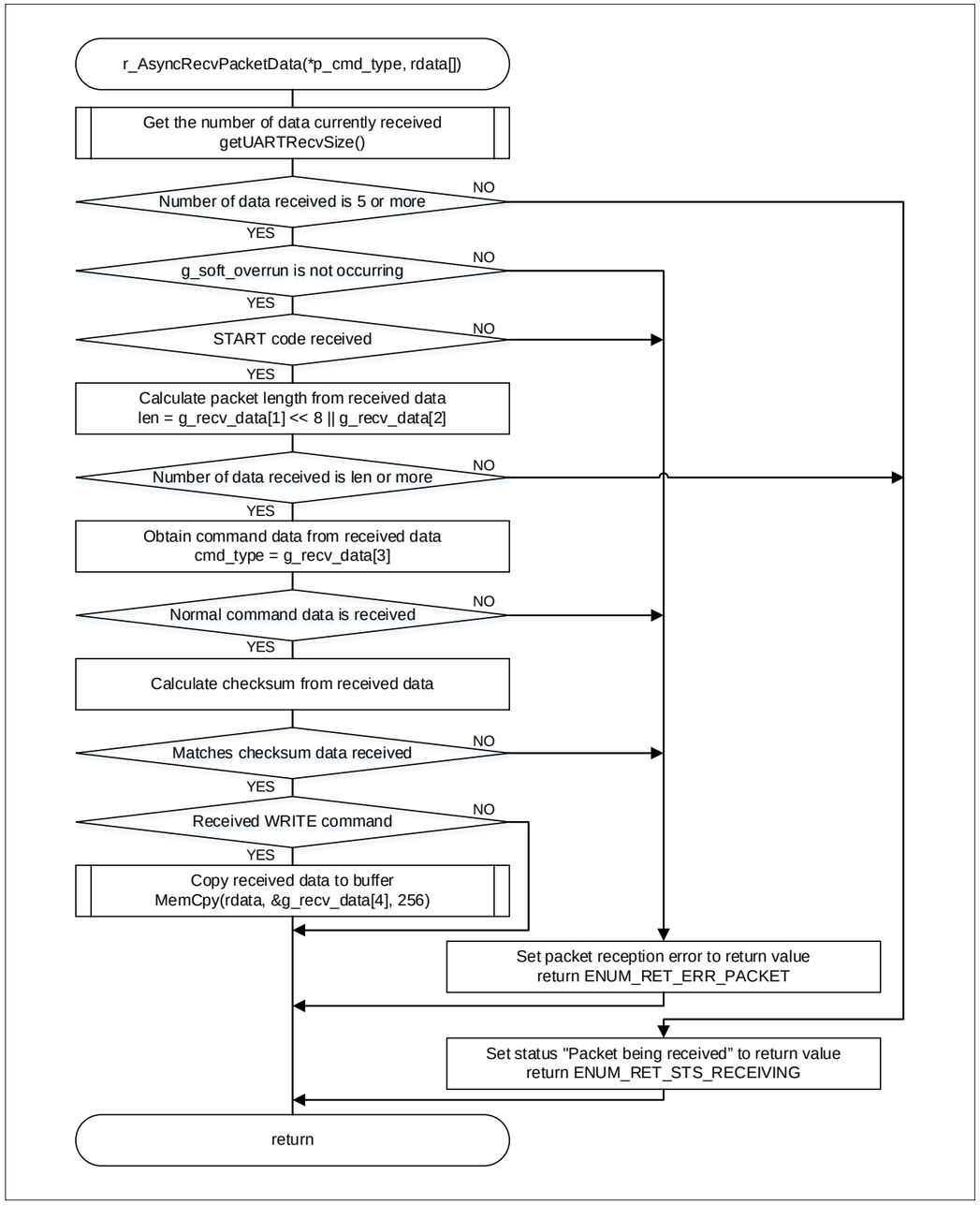
图4-23代码闪存重新编程的处理



4.11.19 Processing to receive asynchronous command packets

Figure 4-24 shows the flowchart of processing to receive asynchronous command packets

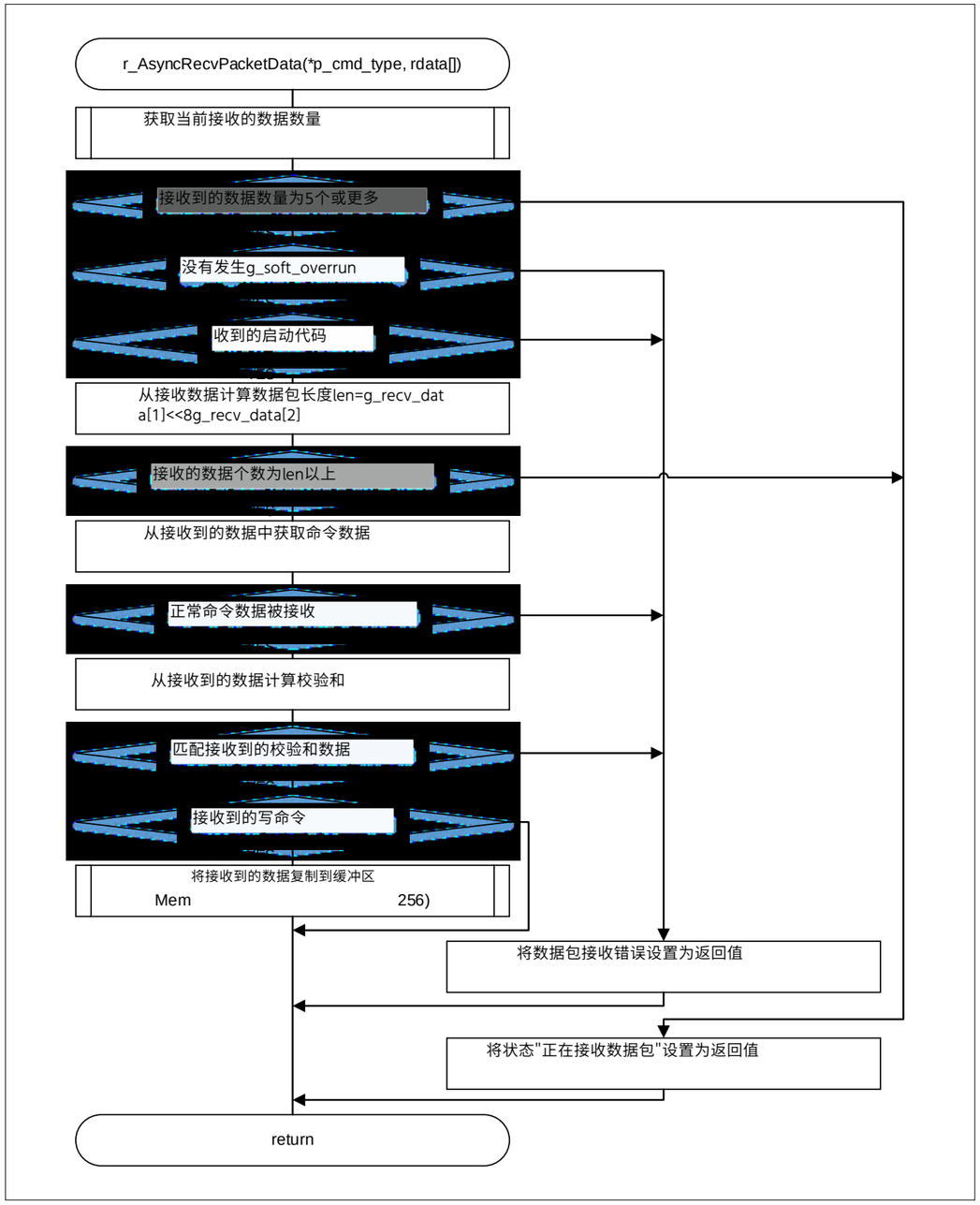
Figure 4-24 Processing to receive asynchronous command packets



4.11.19 处理以接收异步命令分组

图4-24示出了处理以接收异步命令分组的流程图

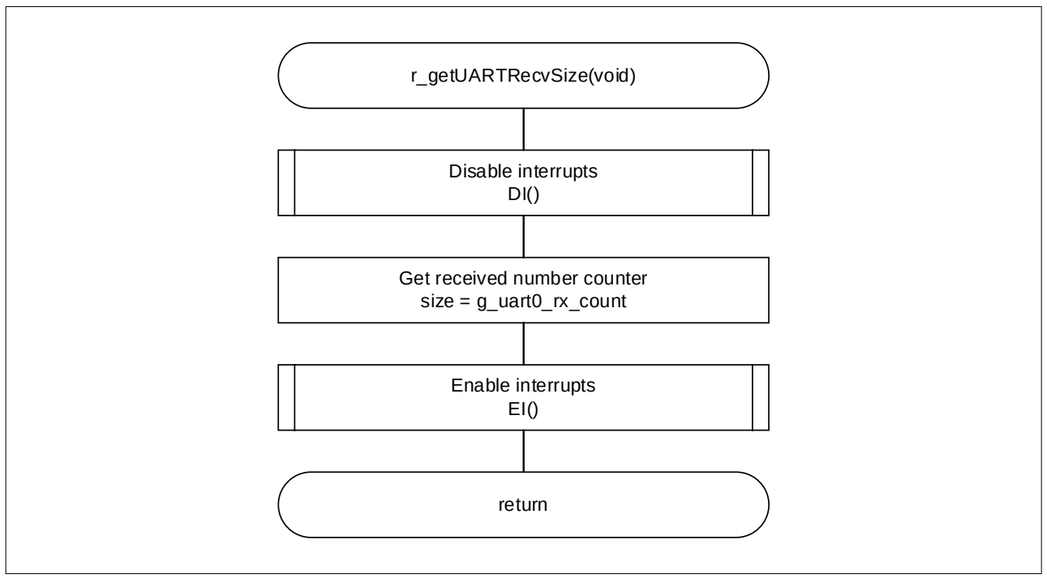
图4-24接收异步命令报文的处理



4.11.20 Processing to obtain the size of the receive data

Figure 4-25 shows the flowchart of processing to obtain the size of the receive data

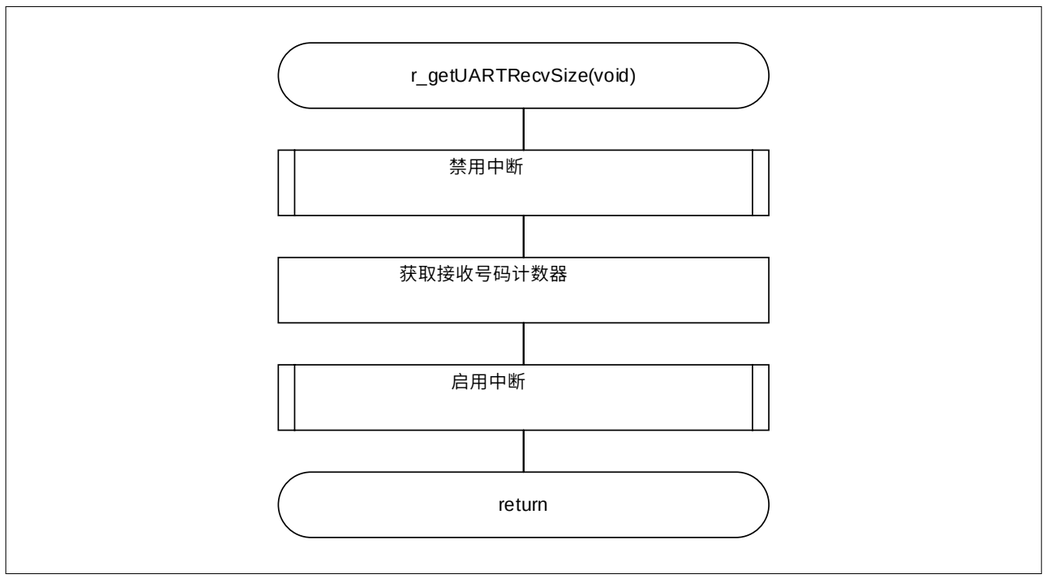
Figure 4-25 Processing to obtain the size of the receive data



4.11.20 处理以获得接收数据的大小

图4-25示出了获取接收数据大小的处理的流程图

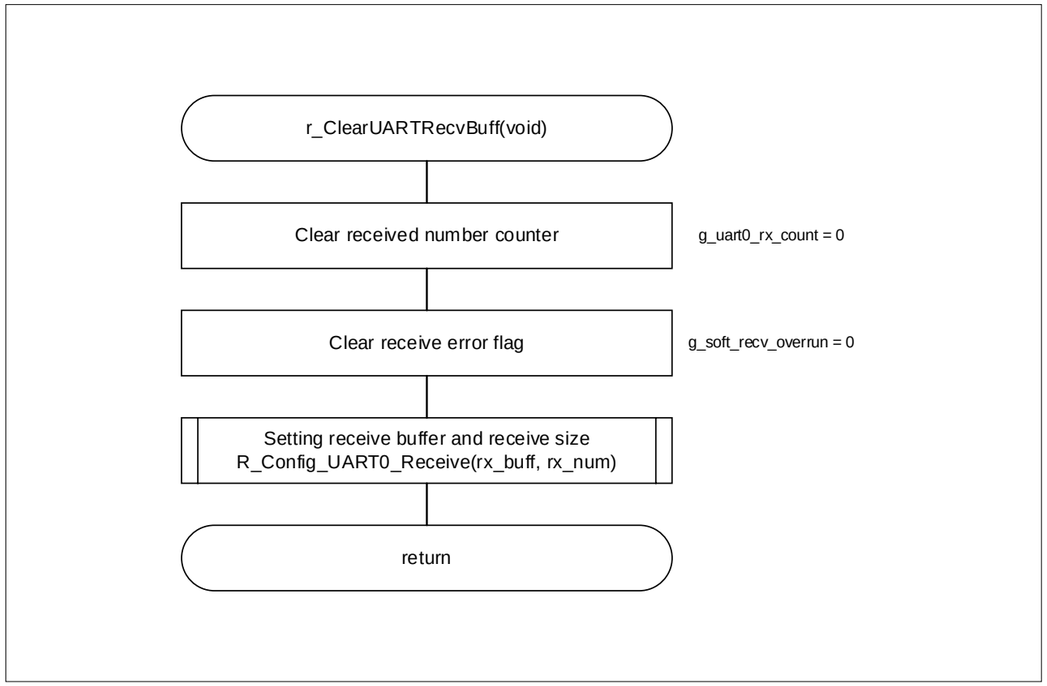
图4-25处理获取接收数据的大小



4.11.21 Processing to clear the receive buffer

Figure 4-26 shows the flowchart of processing to clear the receive buffer

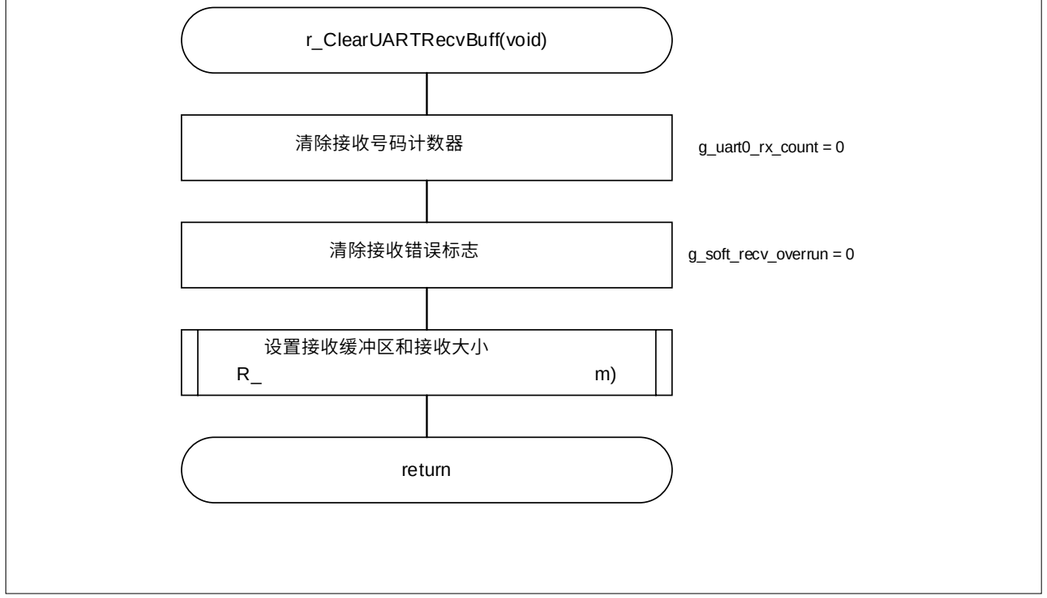
Figure 4-26 Processing to clear the receive buffer



4.11.21 处理以清除接收缓冲区

图4-26显示了清除接收缓冲区的处理流程图

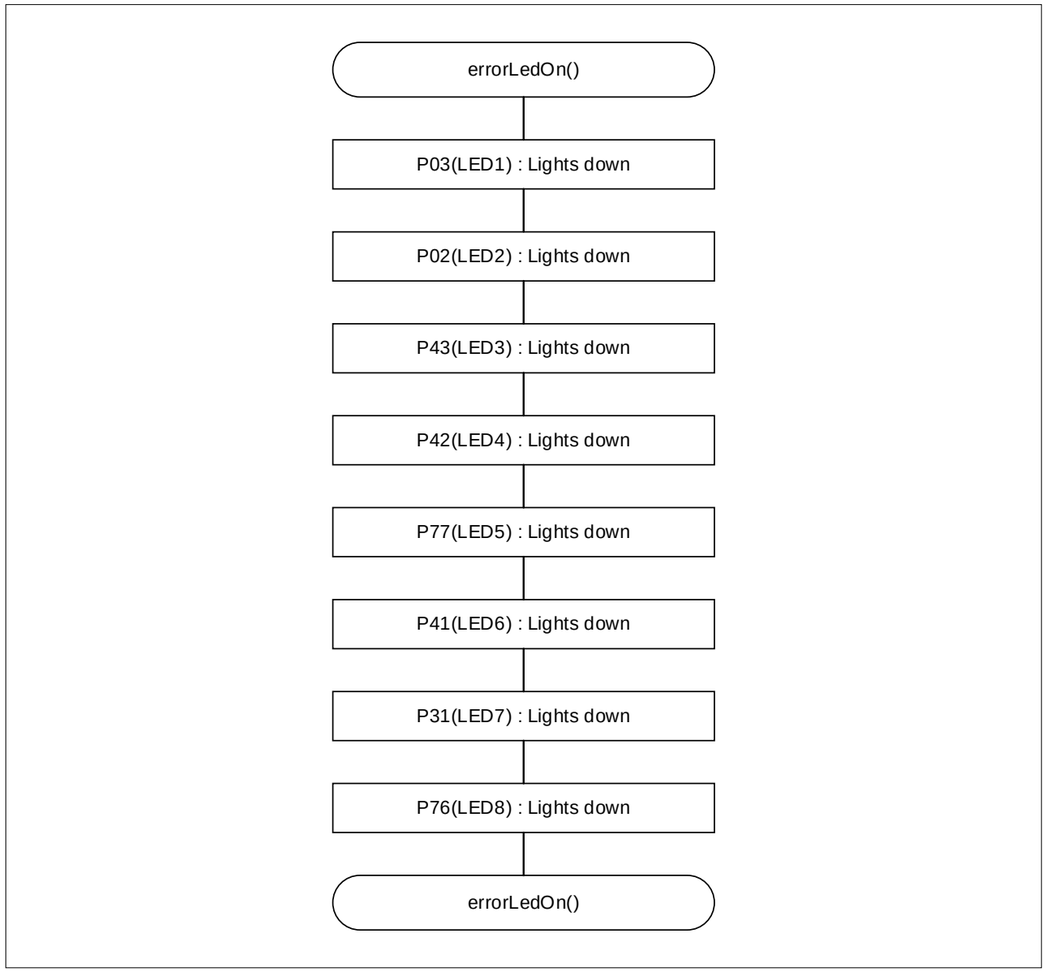
图4-26清除接收缓冲区的处理



4.11.22 Processing to turn on the error LED

Figure 4-27 shows the flowchart of processing to turn on the error LED

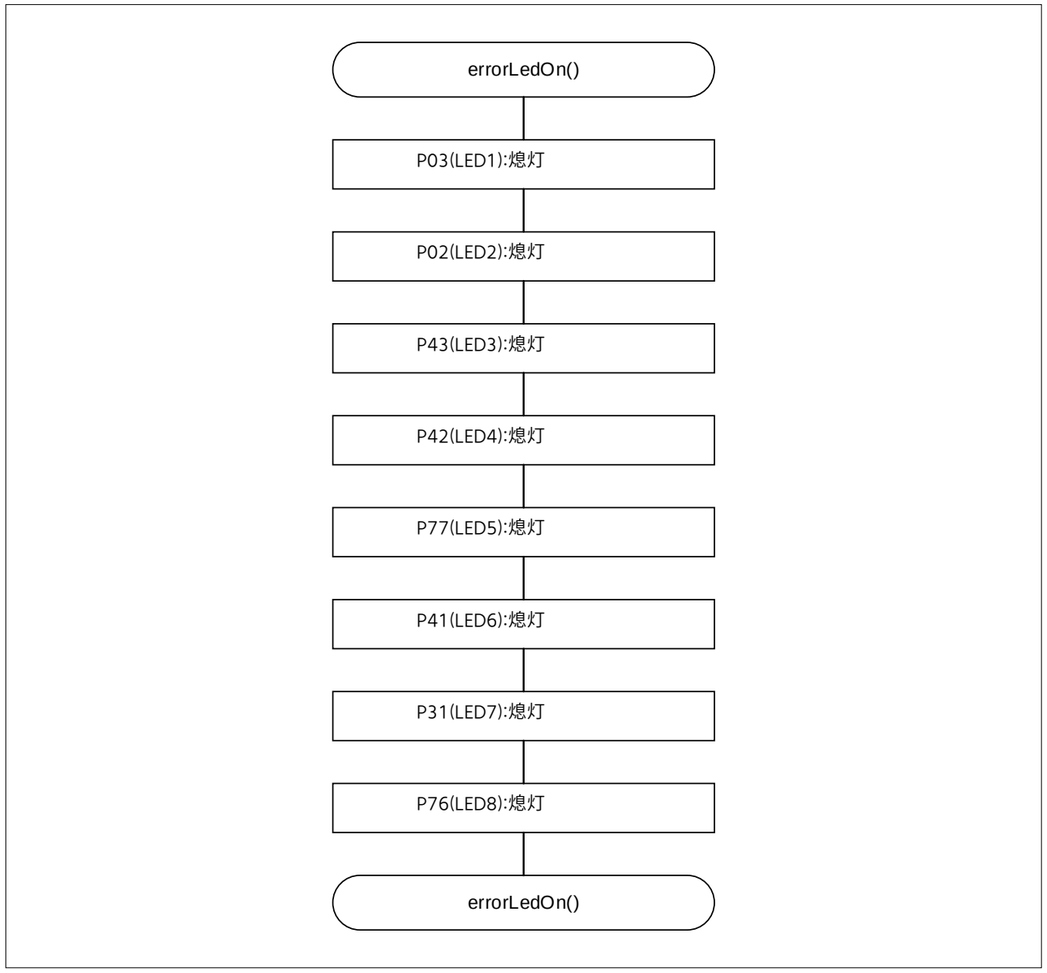
Figure 4-27 Processing to turn on the error LED



4.11.22 处理以打开错误LED

图4-27显示了打开错误LED的处理流程图

图4-27打开错误LED的处理



5. GUI-Based Tool for Writing Data

This chapter describes the GUI-based tool for writing data to the target device simply by running an executable file (.exe). Select the binary file (.bin) that contains the data to be written. To perform a write again, restart the tool.

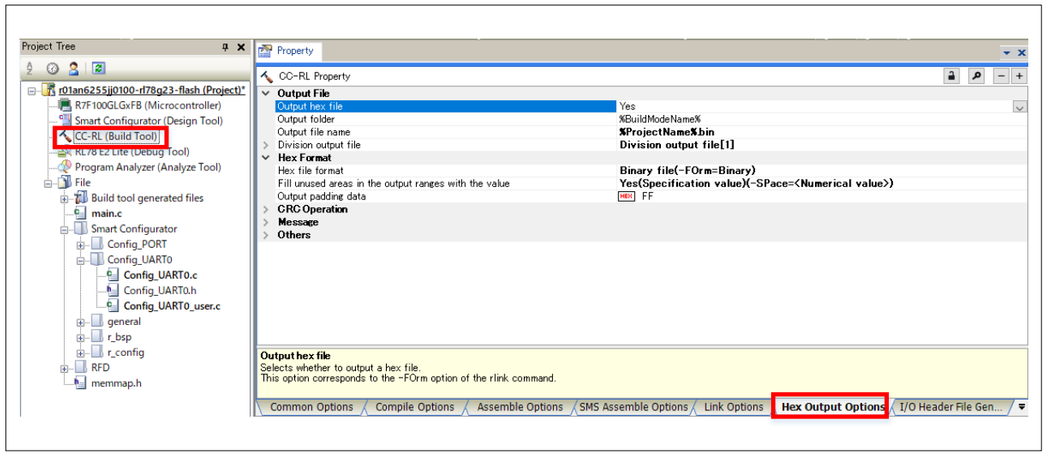
5.1 Generating a File Required to Write Data

Before you can use the GUI-based tool, generate a binary file (.bin) that will be written. For details about how to generate a binary file, see the following sections.

5.1.1 Using CS+ to Generate a Binary File

In the [Project Tree], select [CC-RL (Build Tool)], and then open the [Hex Output Options] tab.

Figure 5-1 Generate a binary file in CS+ (1/6)



5.基于GUI的数据写入工具

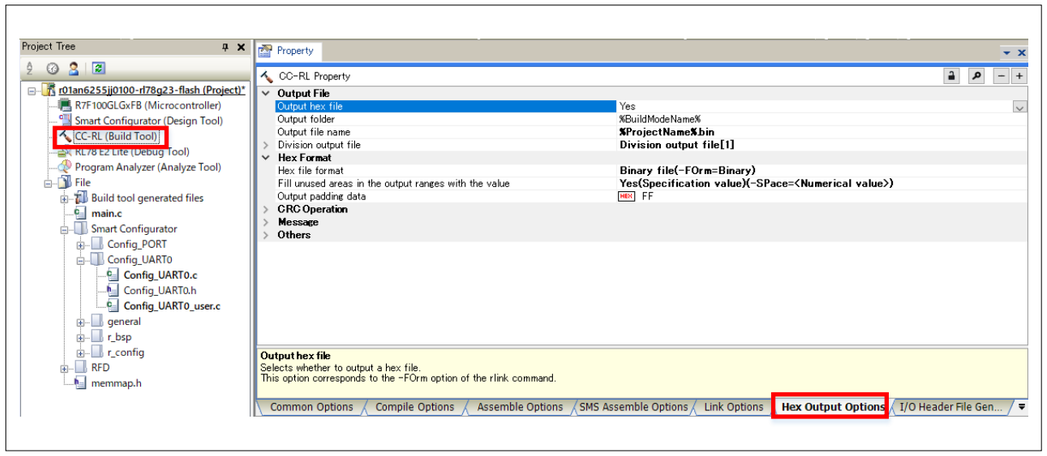
本章描述了基于GUI的工具，只需通过运行可执行文件（.exe)。选择二进制文件(.bin)包含要写入的数据。要再次执行写操作，请重新启动该工具。

5.1生成写入数据所需的文件

在您可以使用基于GUI的工具之前，请生成二进制文件（。bin），将被写入。有关如何生成二进制文件的详细信息，请参阅以下部分。

5.1.1使用CS+在[项目树]中生成二进制文件，选择[CC-RL（构建工具）]，然后打开[十六进制输出选项]选项卡。

图5-1在CS+中生成二进制文件(16)



In the [Hex Output Options] tab, under [Output File], set [Yes] for [Output hex file].

Select [Division output file], and then, in the dialog box that appears, enter a character string in the following pattern:

XXXXXXXXXX`.bin=0-`*YYYYYYYYY*

For *XXXXXXXXXX*, specify the project name. For *YYYYYYYYY*, specify the last address of the code flash memory of the device to be used.

Figure 5-2 Generate a binary file in CS+ (2/6)

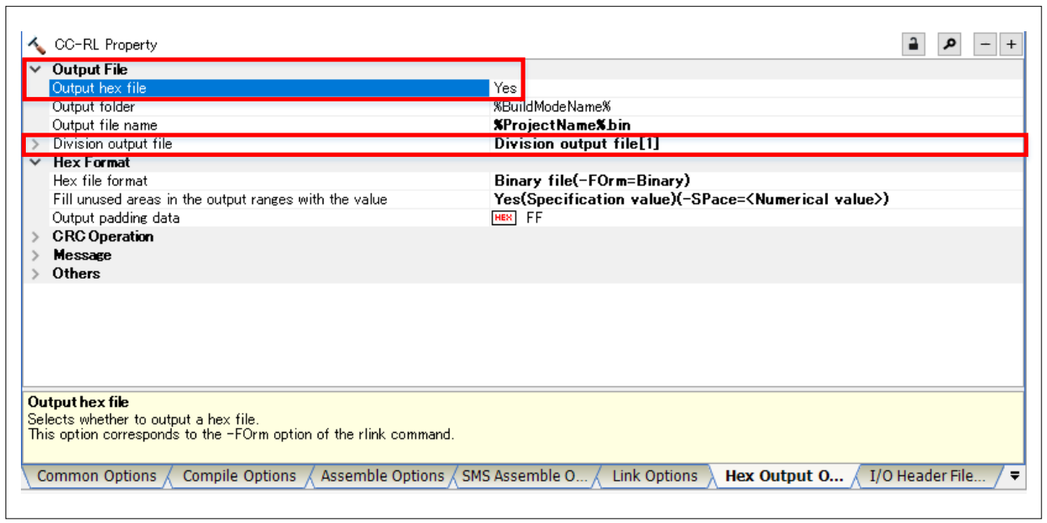
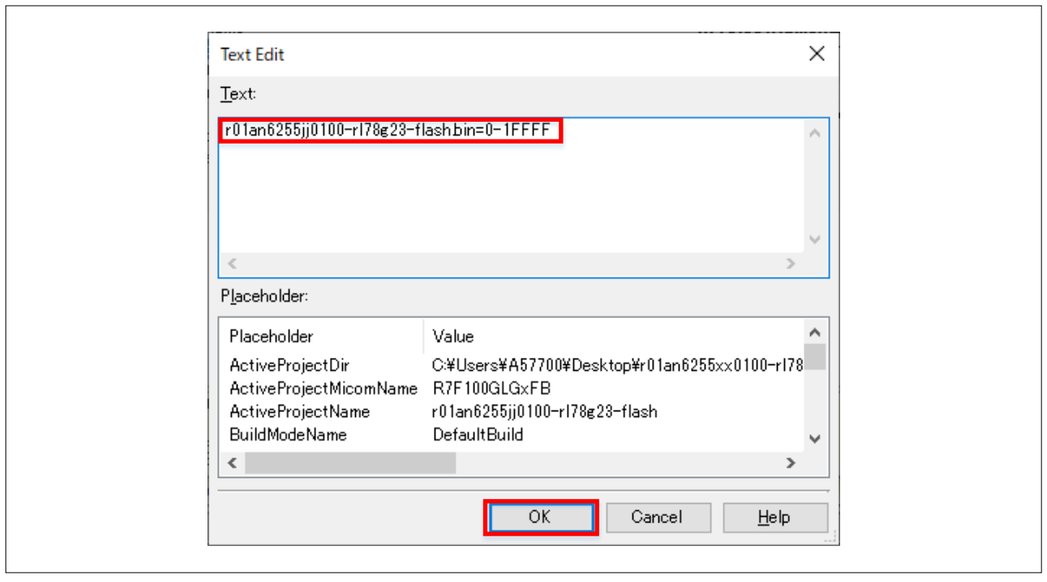


Figure 5-3 Generate a binary file in CS+ (3/6)



In the [Hex Output Options: tab, under [Hex Format], set [Hex file format] to [Binary file (-FOrm=Binary)].

在[十六进制输出选项]选项卡中，在[输出文件]下，为[输出十六进制文件]设置[是]。

选择[分割输出文件]，然后在出现的对话框中，按以下模式输入字符串:

XXXXXXXXXX`.bin=0-`*YYYYYYYYY*

对于XXXXXX，指定项目名称。对于YYYYYYYYYYYY，指定要使用的设备的代码闪存的最后地址。

图5-2在CS+中生成二进制文件(26)

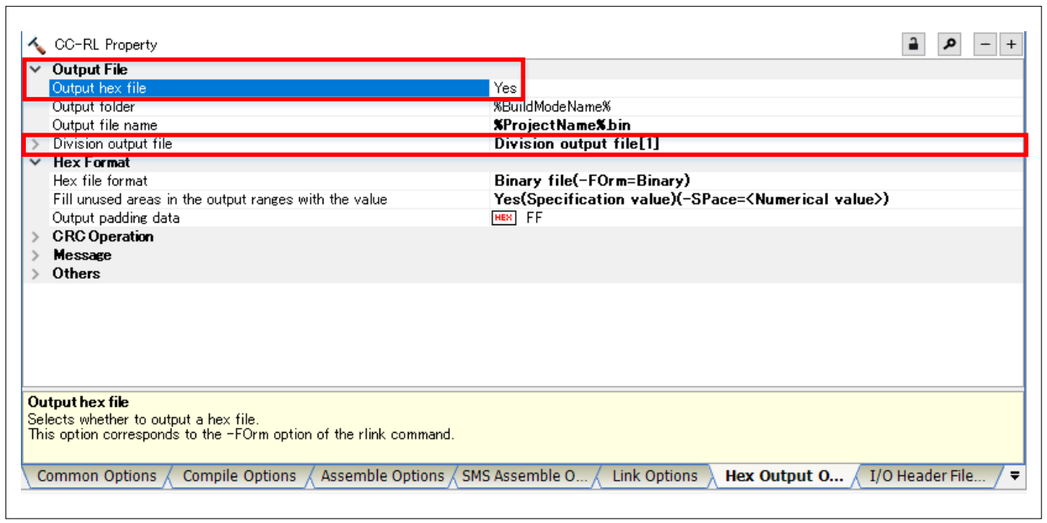
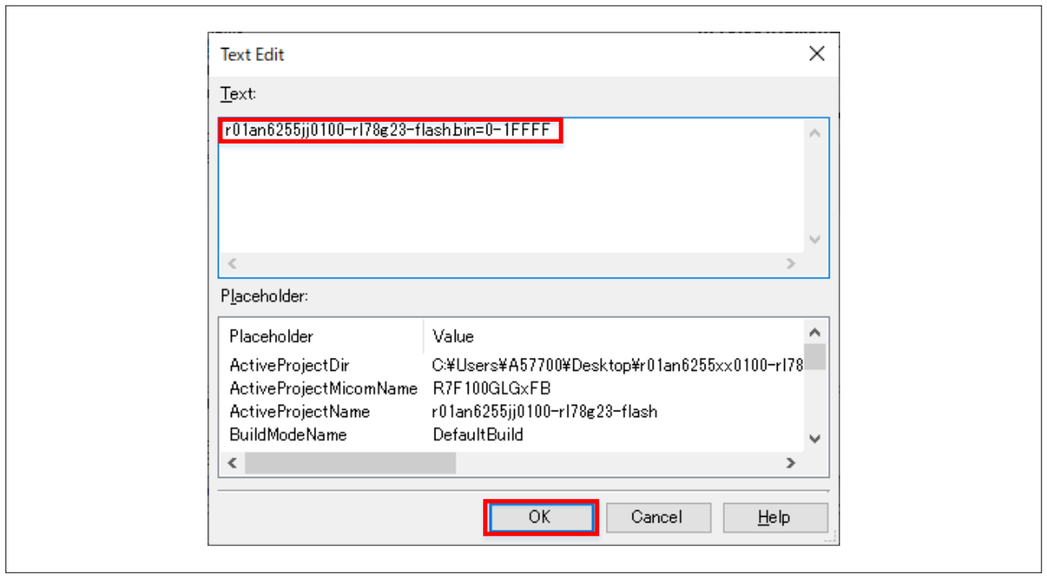


图5-3在CS+(36)中生成二进制文件



在[十六进制输出选项：选项卡中，在[十六进制格式]下，将[十六进制文件格式]设置为[二进制文件（-FOrm=二进制）]。

Figure 5-4 Generate a binary file in CS+ (4/6)

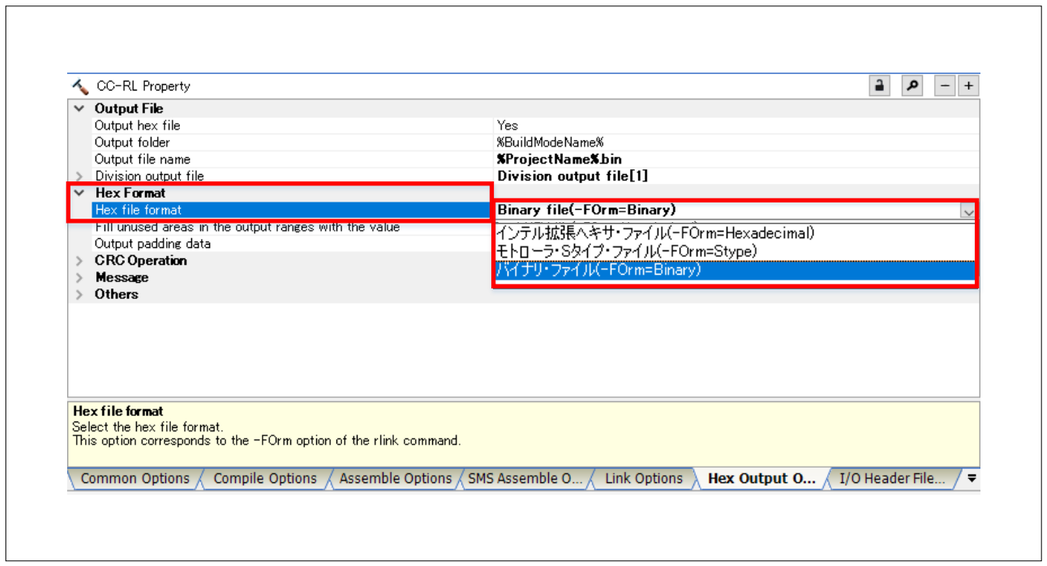
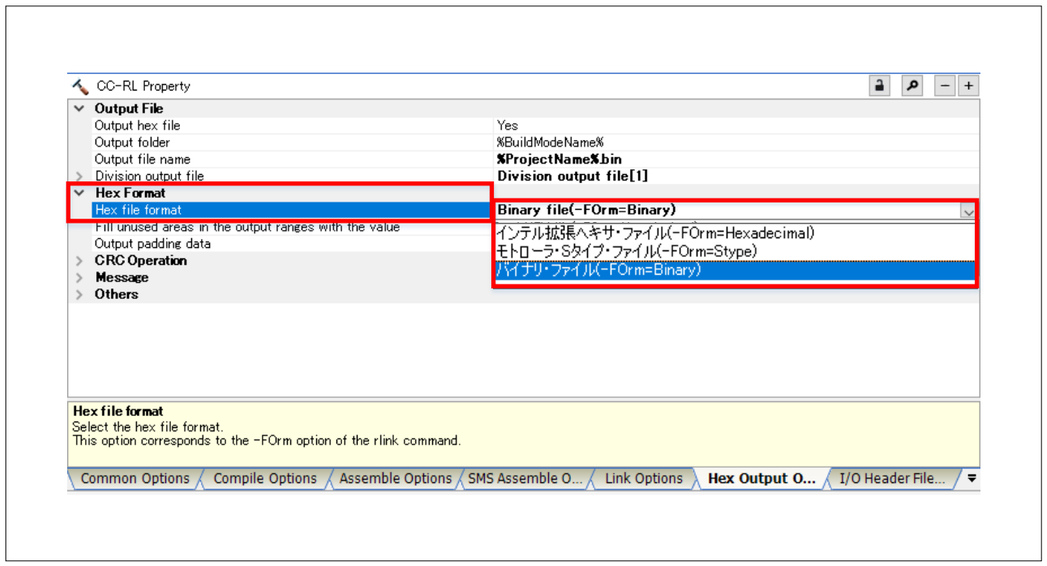
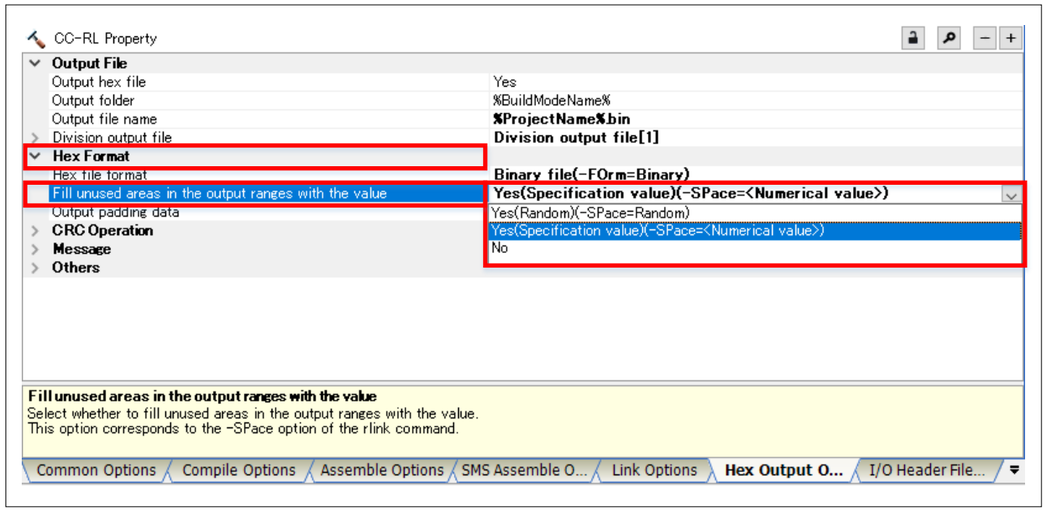


图5-4在CS+中生成二进制文件(46)



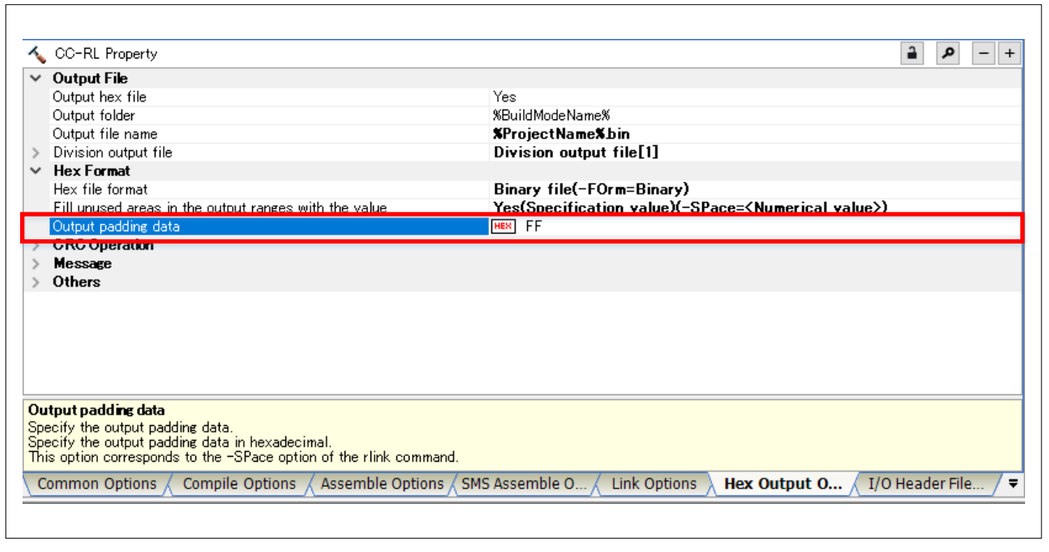
In the [Hex Output Options] tab, under [Hex Format], set [Fill unused areas in the output ranges with the value] to [Yes (Specification value) (-SPace=<Numerical value>)].

Figure 5-5 Generate a binary file in CS+ (5/6)



In the [Hex Output Options] tab, under [Hex Format], set [Output padding data] to [FF].

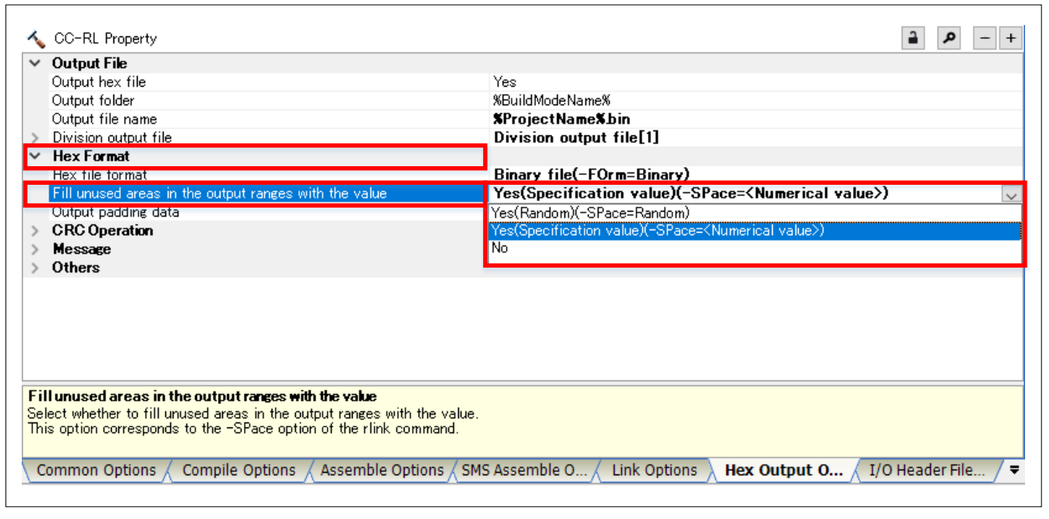
Figure 5-6 Generate a binary file in CS+ (6/6)



A binary file is generated when you build a project.

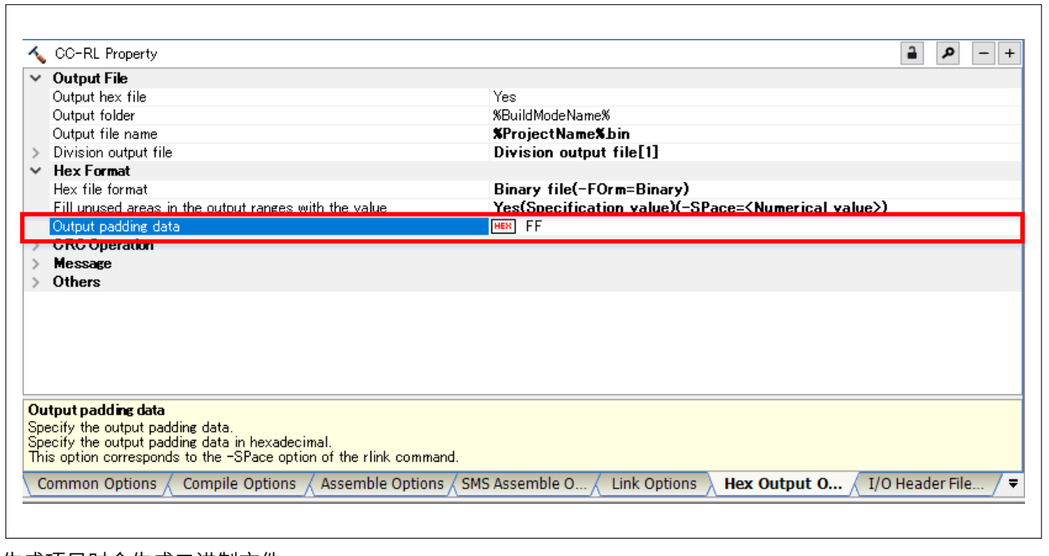
在[十六进制输出选项]选项卡中，在[十六进制格式]下，将[用值填充输出范围中未使用的区域]设置为[是（规格值）（-SPace=<数值>）]。

图5-5在CS+中生成二进制文件(56)



在[十六进制输出选项]选项卡中，在[十六进制格式]下，将[输出填充数据]设置为[FF]。

图5-6在CS+中生成二进制文件(66)



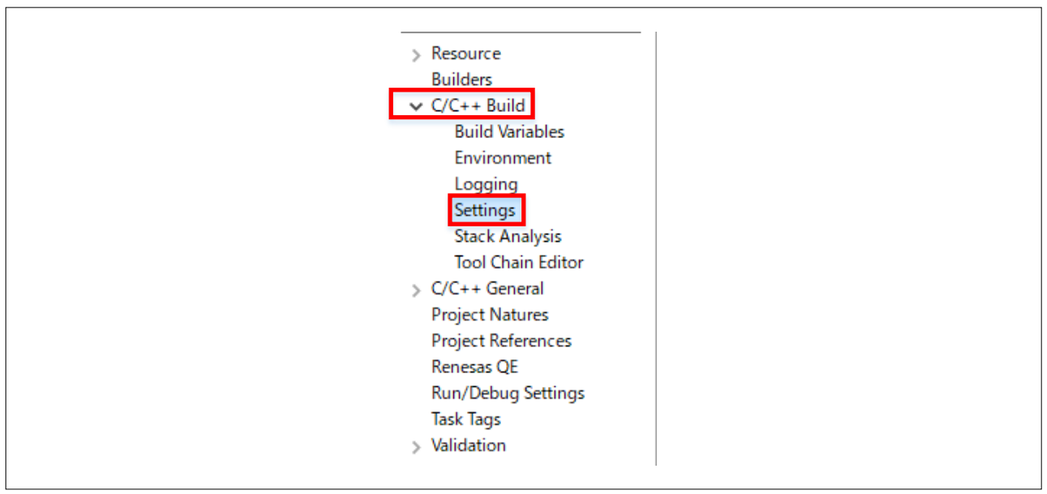
生成项目时会生成二进制文件。

5.1.2 Using e2studio to Generate a Binary File

In the [Project] tab, select [Properties].

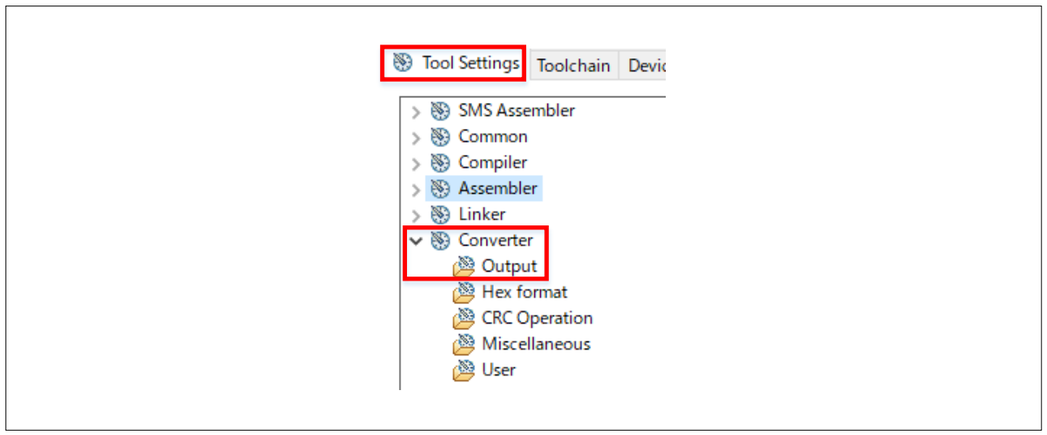
Under [C/C++ Build], select [Settings].

Figure 5-7 Generate a binary file in e2 studio (1/3)



Select [Converter] and [Output] in the [Tool Settings] tab.

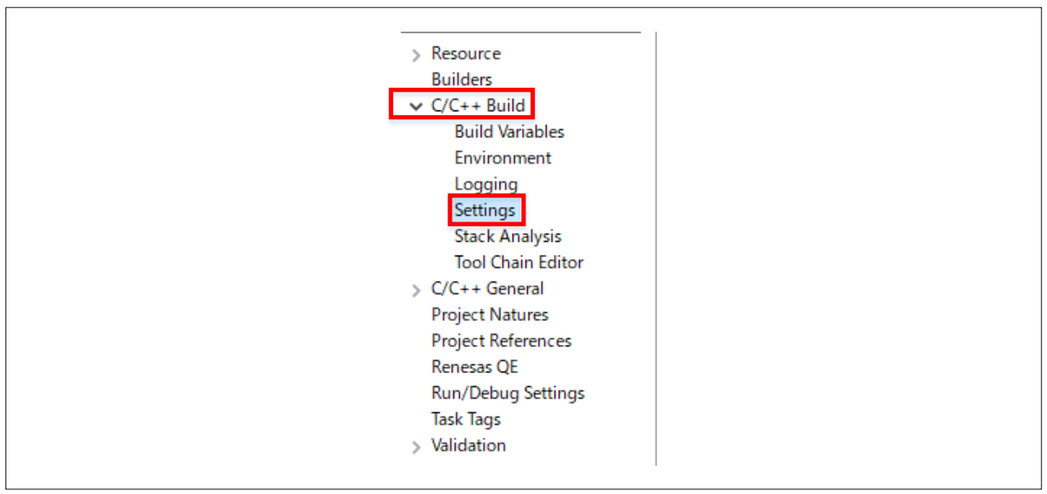
Figure 5-8 Generate a binary file in e2 studio (2/3)



5.1.2使用e2studio生成二进制文件在[项目]选项卡中选择[属性]。

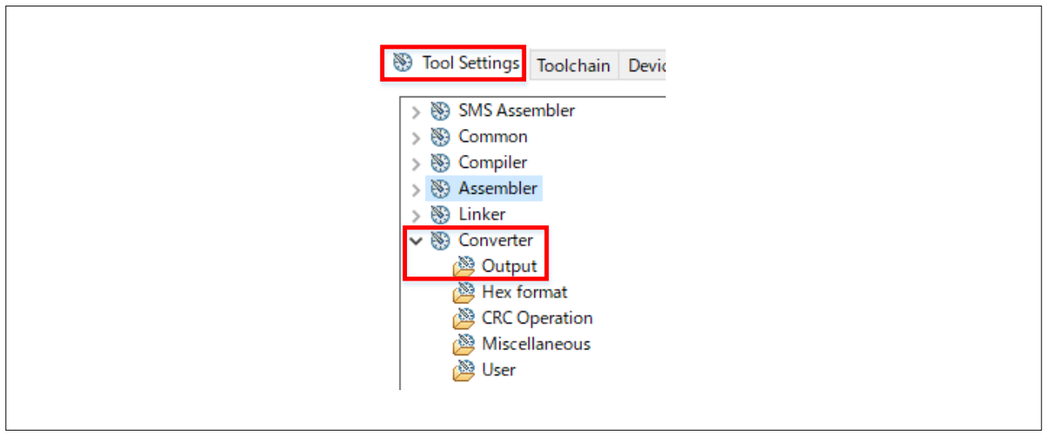
在[CC++Build]下，选择[Settings]。

图5-7在e2studio中生成二进制文件(13)



在[工具设置]选项卡中选择[转换器]和[输出]。

图5-8在e2studio中生成二进制文件(23)



Select the [Run the load module converter] check box.

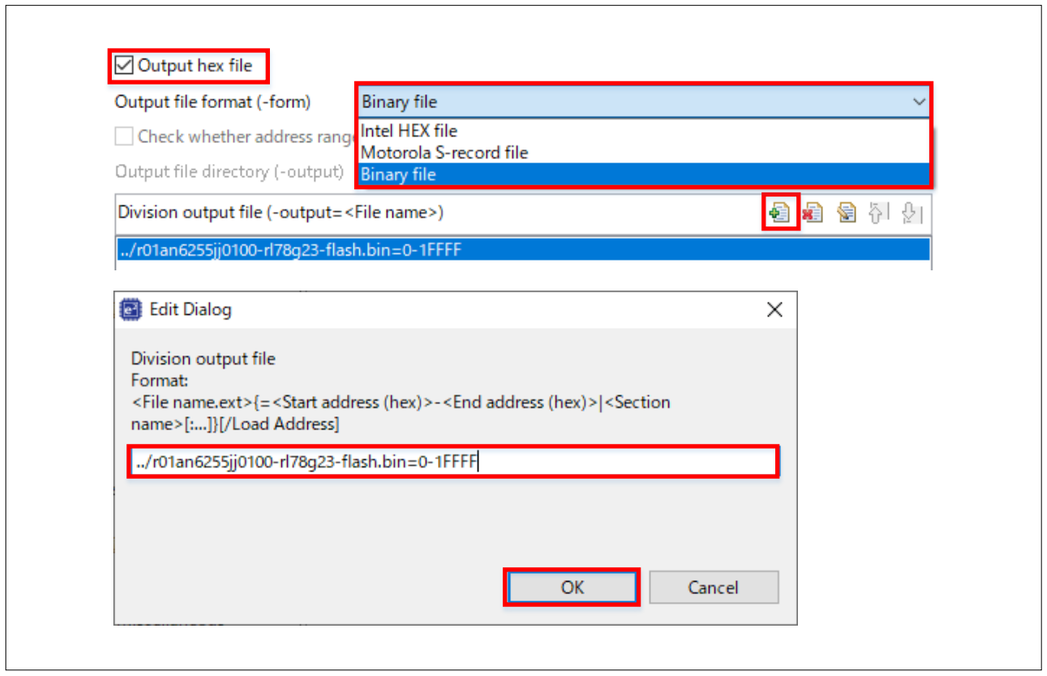
From the [Output file format] drop-down list, select [Output a binary file].

Click the [Add] button, and then enter a character string in the following pattern:

`../XXXXXX.bin=0-YYYYYY`

For *XXXXXX*, specify the project name. For *YYYYYY*, specify the last address of the code flash memory of the device to be used.

Figure 5-9 Generate a binary file in e2 studio (3/3)



A binary file is generated when you build a project.

选中[运行加载模块转换器]复选框。

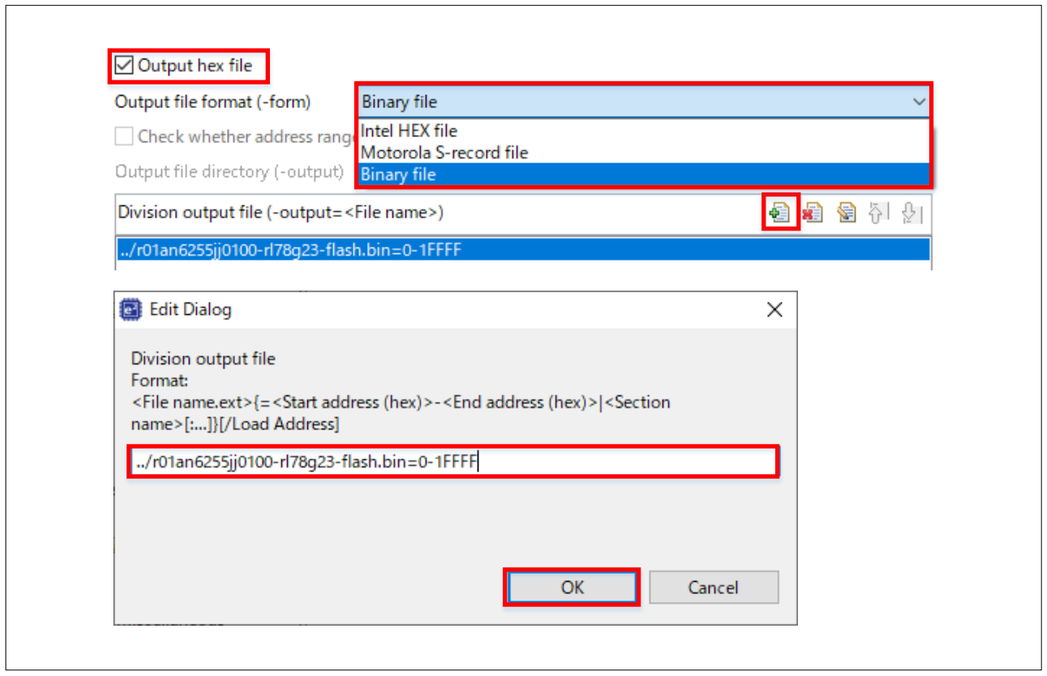
从[输出文件格式]下拉列表中，选择[输出二进制文件]。

单击[添加]按钮，然后按以下模式输入字符串:

`../XXXXXX.bin=0-YYYYYY`

对于XXXXXX，指定项目名称。对于YYYYYY，指定要使用的设备的代码闪存的最后地址。

图5-9在e2studio中生成二进制文件(33)



生成项目时会生成二进制文件。

5.1.3 Using IAR EW to Generate a Binary File

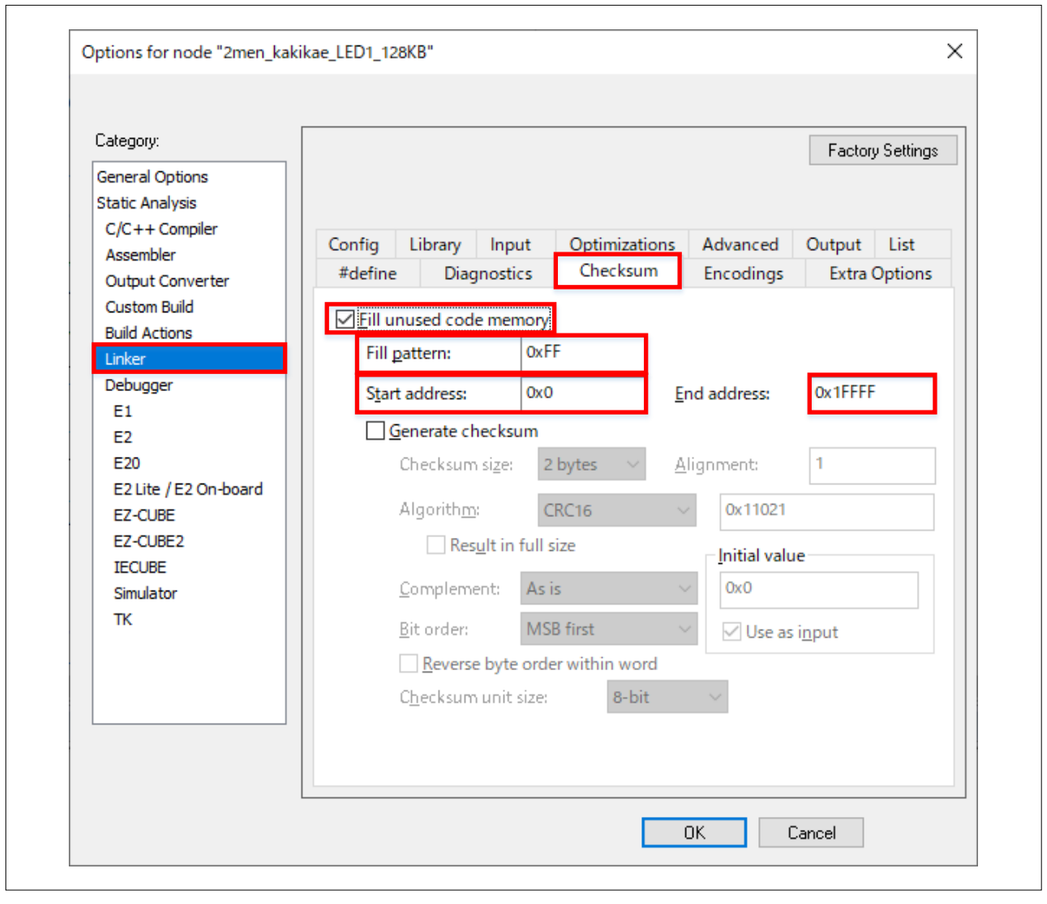
In the [Project] tab, select [Options].

In the [Category] list box, select [Linker], and then select the [Checksum] tab.

Select the [Fill unused code memory] check box.

For [Fill pattern], specify 0xFF. For [Start address], specify 0x0. For [End address], specify the last address of the code flash memory of the device to be used with a hexadecimal number prefixed by "0x".

Figure 5-10 Generate a binary file in IAR EW (1/2)



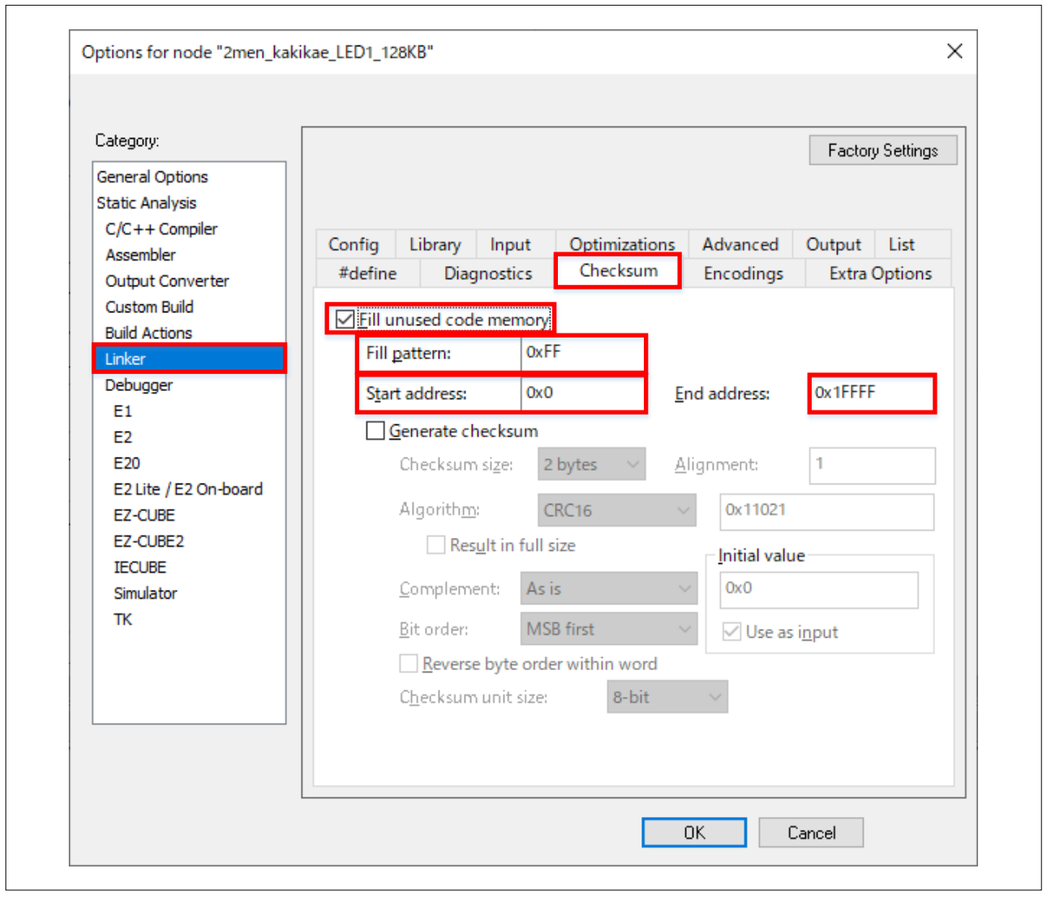
5.1.3使用IAREW生成二进制文件在[项目]选项卡中选择[选项]。

在[类别]列表框中，选择[链接器]，然后选择[校验和]选项卡。

选中[填充未使用的代码内存]复选框。

对于[填充模式]，指定0xFF。对于[起始地址]，指定0x0。对于[结束地址]，指定要与以"0x"为前缀的十六进制数字一起使用的设备的代码闪存的最后地址。

图5-10在IAREW中生成二进制文件(12)

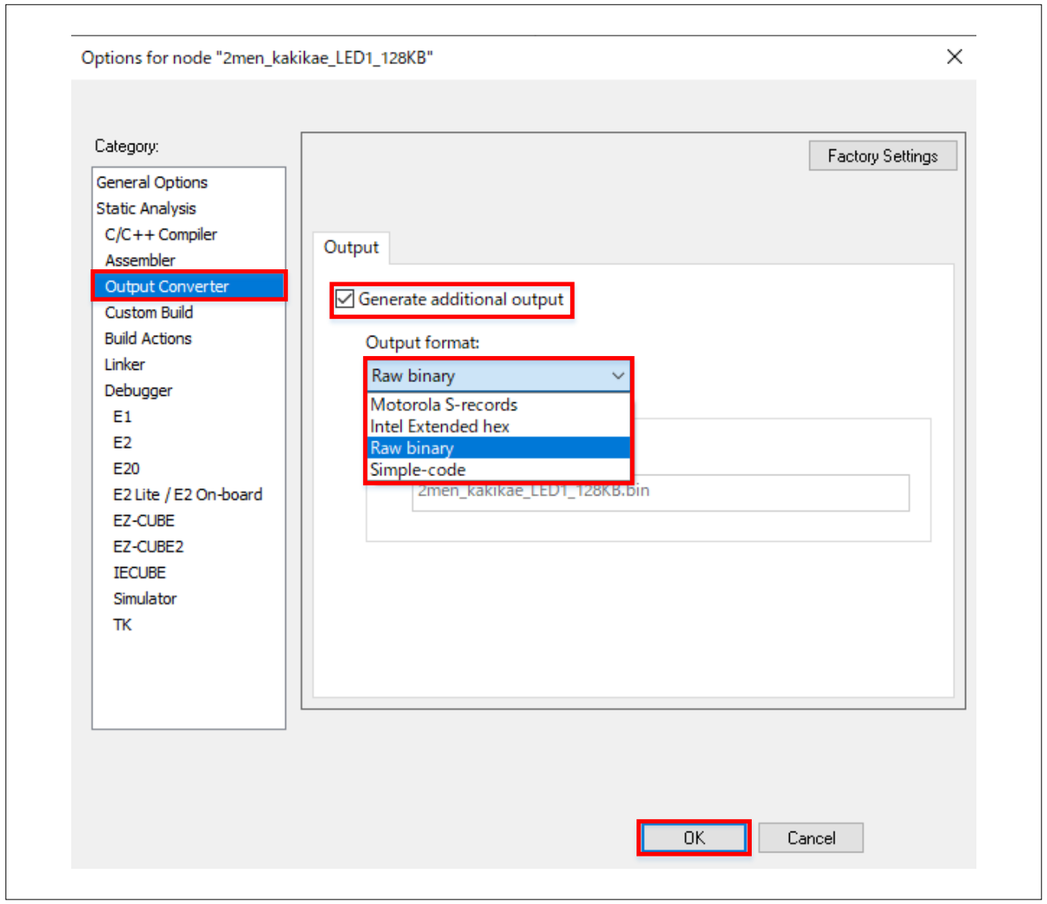


In the [Output Converter] tab, select the [Generate additional output] check box.

From the [Output format] drop-down list, select [Raw binary].

Then, click [OK]. A binary file is generated when you build a project.

Figure 5-11 Generate a binary file in IAR EW (2/2)

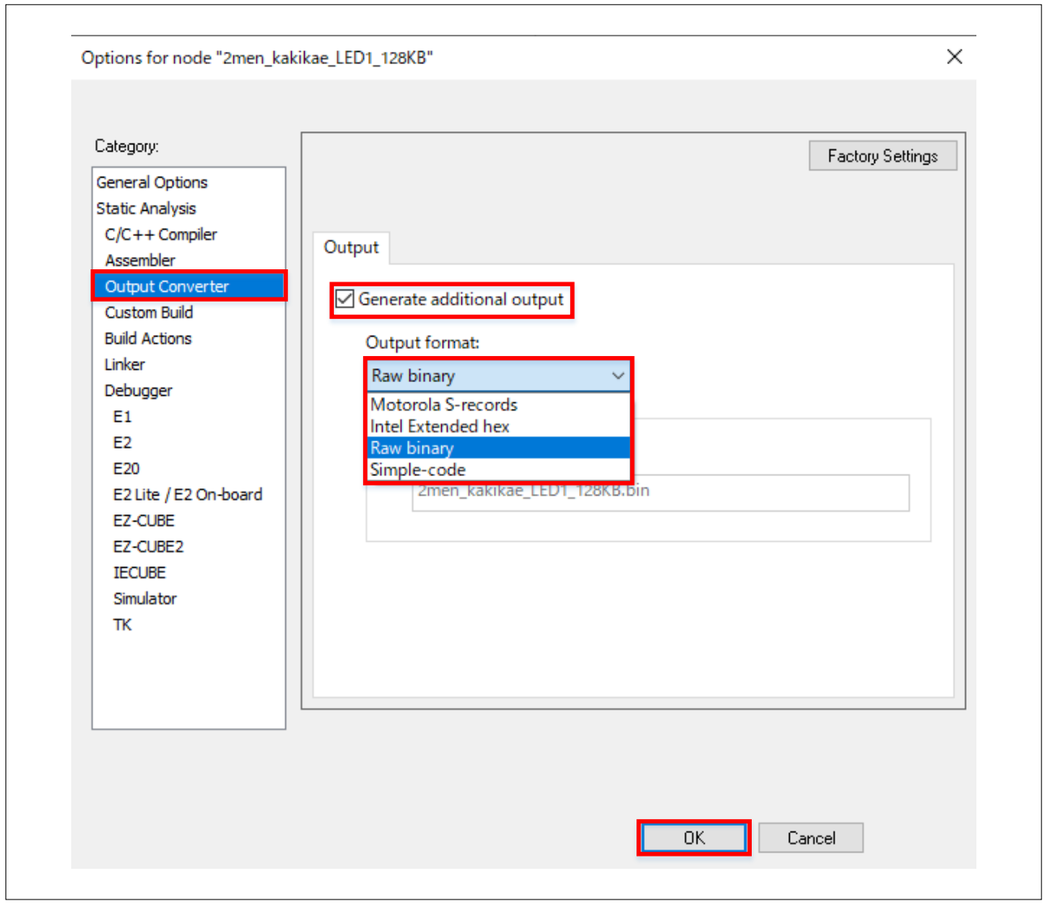


在[输出转换器]选项卡中，选中[生成其他输出]复选框。

从[输出格式]下拉列表中，选择[原始二进制]。

然后，单击[确定]。生成项目时会生成二进制文件。

图5-11在IAREW中生成二进制文件(2/2)

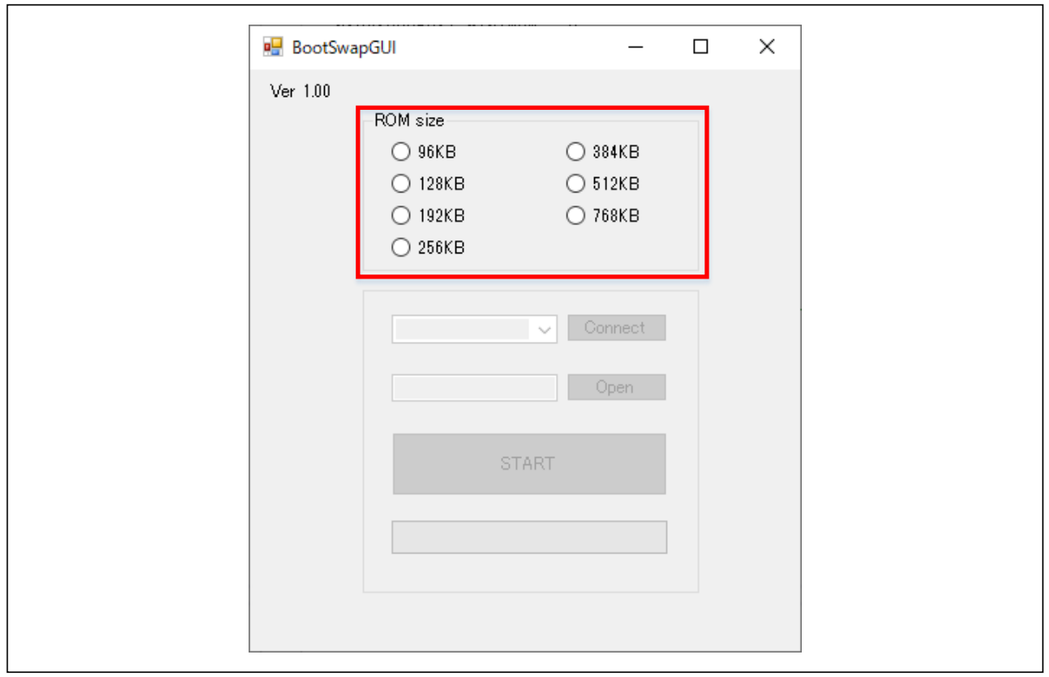


5.2 Using GUI-Based Tool

Run BootSwapGUI.exe.

Select the radio button for the ROM size of the device to be used.

Figure 5-12 Description of GUI (1/2)

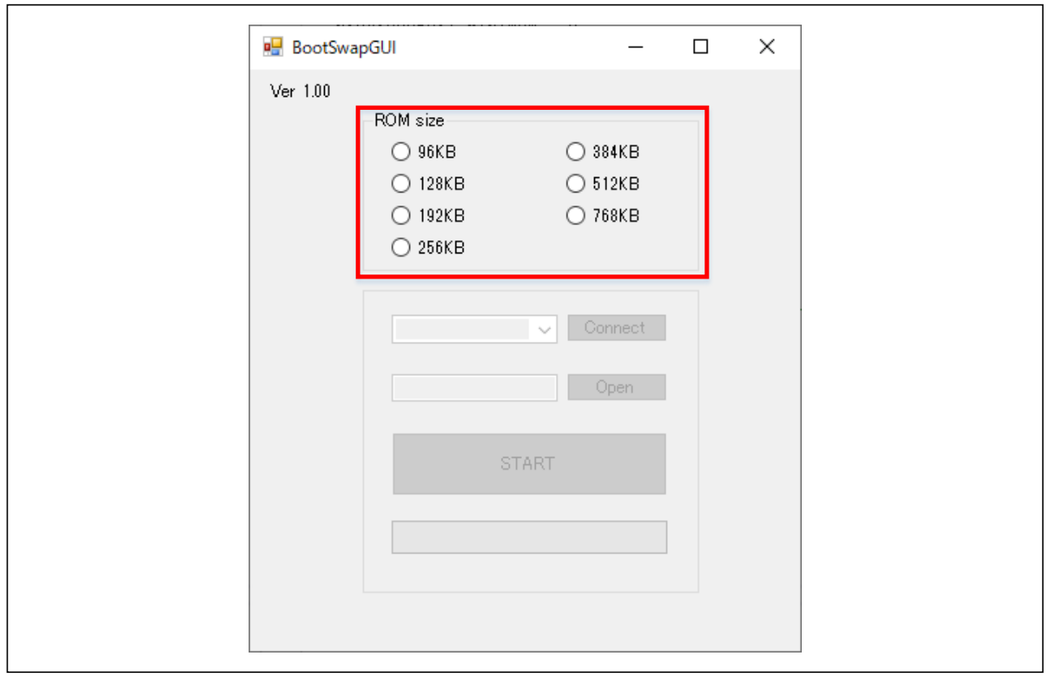


5.2使用基于GUI的工具

Run BootSwapGUI.exe.

选择要使用的设备的ROM大小的单选按钮。

图5-12GUI的说明(12)



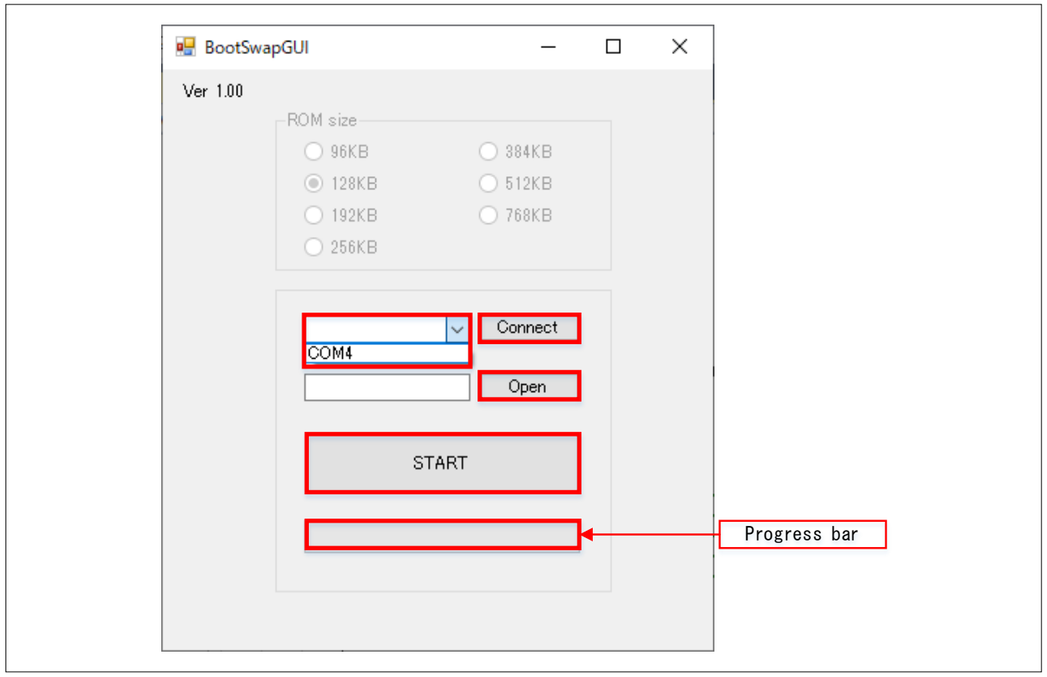
From the drop-down list, select an available COM port, and then click [Connect] to connect to the target device.

Click [Open], and then select the program (.bin) to be written.

Click [START] to start writing the program.

The progress bar shows the progress of write processing.

Figure 5-13 Description of GUI (2/2)



After writing is complete, exit BootSwapGUI.exe.

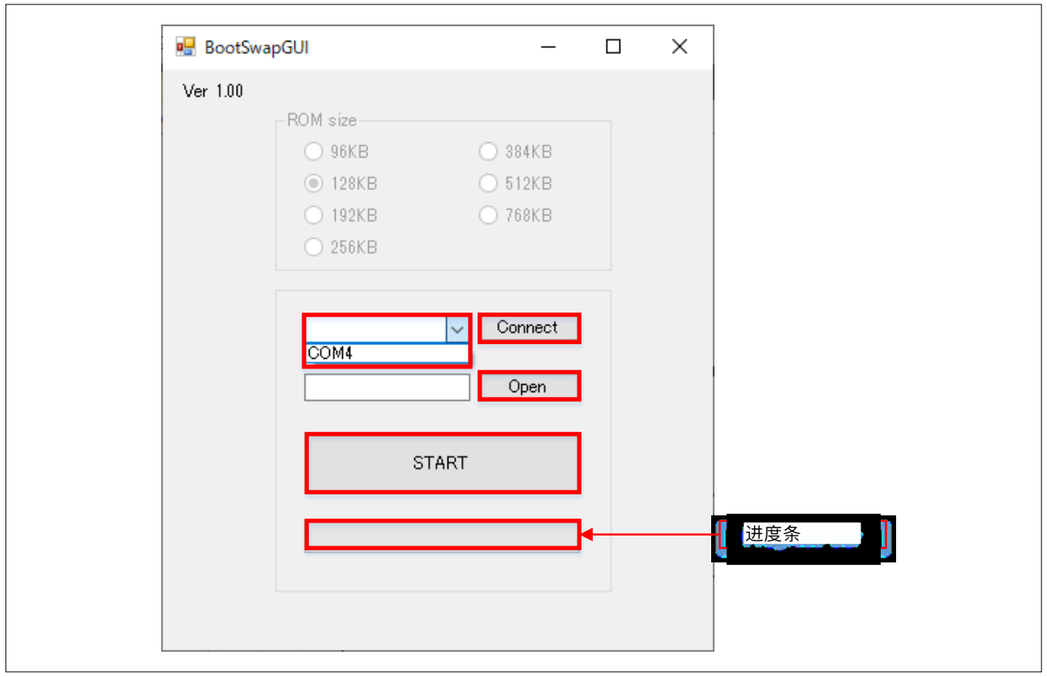
从下拉列表中，选择一个可用的COM端口，然后单击[连接]以连接到目标设备。

单击[打开]，然后选择程序（。bin)来写入。

单击[开始]开始编写程序。

进度条显示写入处理的进度。

图5-13GUI的说明(22)



写入完成后，退出BootSwapGUI。exe的。

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

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

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

Website and Support

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

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

6.示例代码

示例代码可从瑞萨电子网站下载。

7.参考文件

RL78G23用户手册:硬件(R01UH0896)RL78家庭用户
手册软件(R01US0015)
最新版本可从瑞萨电子网站下载。
技术更新

最新版本可从瑞萨电子网站下载。

网站及支援

瑞萨电子网站
http://www.renesas.com/

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Aug.04.22	—	First Edition

修订历史

Rev.	Date	Description	
		Page	Summary
1.00	Aug.04.22	—	第一版

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.

处理微处理单元和微控制器的一般注意事项
单位产品

以下使用说明适用于瑞萨电子的所有微处理单元和微控制器单元产品。有关本文档所涵盖产品的详细使用说明，请参阅本文档的相关部分以及为产品发布的任何技术更新。

1. 防止静电放电(ESD)的预防措施
   当暴露于CMOS器件时，强电场会导致栅极氧化物的破坏并最终降低器件的工作性能。步骤必须采取尽可能停止静电的产生，并在发生时迅速消散。环境控制必须是足够了。干燥时，应使用加湿器。建议避免使用容易产生静电的绝缘体。半导体器件必须在防静电容器、静电屏蔽袋或导电材料中储存和运输。所有测试及包括工作台和地板在内的测量工具必须接地。操作员还必须使用腕带接地。半导体 半导体设备不得徒手触摸。对于安装有半导体器件的印刷电路板必须采取类似的预防措施。
2. 开机处理
   产品的状态在供电时未定义。LSI内部电路的状态是不确定的，并且在供电时，寄存器设置和引脚未定义。在复位信号施加到外部复位的成品中引脚，引脚的状态不保证从电源供应到复位过程完成。以类似的方式，引脚的状态在一个由片上电源复位功能复位的产品不能保证从供电的时间，直到功率达到指定重置的级别。
3. 在断电状态下输入信号
   器件断电时请勿输入信号或IO上拉电源。输入这样的信号或IO产生的电流注入上拉电源可能导致故障，此时在设备中通过的异常电流可能导致内部退化元素。按照产品文档中所述的关断状态下输入信号指南。
4. 处理未使用的引脚
   按照手册中处理未使用的引脚给出的指示处理未使用的引脚。CMOS产品的输入引脚是一般处于高阻抗状态。在操作与未使用的引脚在开路状态下，额外的电磁噪声在附近感应在LSI中，一个相关的直通电流在内部流动，由于错误地将引脚状态识别为输入信号而发生故障成为可能。
5. 时钟信号
   在施加复位之后，只有在工作时钟信号变得稳定之后才释放复位线。程中切换时钟信号时执行，等到目标时钟信号稳定。当用外部谐振器或从外部振荡器产生时钟信号时在复位期间，确保复位线只有在时钟信号完全稳定后才被释放。此外，当切换到时钟信号用外部谐振器或由外部振荡器在程序执行进行时产生，等待直到目标时钟信号稳定。
6. 输入引脚电压施加波形
   由于输入噪声或反射波而产生的波形失真可能导致故障。如果CMOS器件的输入停留在VIL之间的区域（马克斯。）和VIH（Min。例如，由于噪声，设备可能发生故障。注意防止颤动噪音进入设备时，输入电平是固定的，并且还在输入电平通过VIL(Max.）和VIH（Min。）.
7. 禁止访问保留地址
   禁止访问保留地址。保留的地址是为将来可能的功能扩展而提供的。不要访问这些不能保证地址作为LSI的正确操作。
8. 产品之间的差异
   在从一个产品更改为另一个产品之前，例如更改为具有不同部件号的产品，请确认更改不会导致问题。同一组中但具有不同部件号的微处理单元或微控制器单元产品的特性可能在条款上有所不同内部存储器容量、布局模式和其他影响电气特性范围的因素，如特性值工作裕度、抗噪声能力和辐射噪声量。更改为具有不同部件号的产品时，实施系统-定产品的评价试验。

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/.

Notice

1. 本文档中提供的电路、软件和其它相关信息的描述仅用于说明半导体产品的操作和应用示例。在您的产品或系统的设计中，您对电路、软件和信息的合并或任何其他使用负全部责任。对于您或第三方因使用这些电路、软件或信息而造成的任何损失和损害，瑞萨电子不承担任何及所有责任。
2. 瑞萨电子特此明确声明，对于因使用本文件所述的瑞萨电子产品或技术信息（包括但不限于产品数据、图纸、图表、程序、算法和应用示例）而引起的涉及第三方专利、版权或其他知识产权的侵权或任何其他索赔，瑞萨电子不承担任何保证和责任。
3. 在此，瑞萨电子或其他公司的任何专利、版权或其他知识产权均未授予任何明示、暗示或其他许可。
4. 您应负责确定需要从任何第三方获得哪些许可证，并根据需要获得此类许可证，以便合法进口，出口，制造，销售，利用，分销或其他处置包含瑞萨电子产品的
5. 您不得更改、修改、复制或逆向工程任何瑞萨电子产品，无论是全部还是部分。瑞萨电子不对您或第三方因此类更改、修改、复制或逆向工程而造成的任何损失或损害承担任何及所有责任。
6. 瑞萨电子产品根据以下两个质量等级进行分类："标准"和"高质量"。每个瑞萨电子产品的预期应用取决于产品的质量等级，如下所示。

   "标准"：计算机；办公设备；通讯设备；测试测量设备；视听设备；家用电器；机床；个人电子设备；工业机器人等。

   "高品质"：运输设备（汽车、火车、轮船等）。）；交通控制（交通灯）；大型通信设备；关键金融终端系统；安全控制设备等。除非在瑞萨电子数据表或其他瑞萨电子文件中明确指定为高可靠性产品或恶劣环境的产品，否则瑞萨电子产品不打算或授权用于可能对人类生命或身体伤害构成直接威胁的产品或系统（人工生命支持设备或系统；手术植入等）。），或可能造成严重财产损失（空间系统；海底中继器；核电控制系统；飞机控制系统；关键设备系统；军事设备等）。).对于因使用与瑞萨电子数据表、用户手册或其他瑞萨电子文档不一致的任何瑞萨电子产品而对您或任何第三方造成的任何损害或损失，瑞萨电子不承担任何及所有责任。

7. 没有任何半导体产品是绝对安全的。尽管瑞萨电子硬件或软件产品可能实施任何安全措施或功能，但瑞萨电子对任何漏洞或安全漏洞（包括但不限于任何未经授权访问或使用瑞萨电子产品或使用瑞萨电子产品的系统）不承担任何责任。瑞萨电子不保证或保证瑞萨电子产品或使用

   瑞萨电子产品将无懈可击或不受腐败、攻击、病毒、干扰
   黑客攻击，数据丢失或被盗，或其他安全入侵（"漏洞问题"）。瑞萨电子否认任何及
   任何漏洞问题引起或与之相关的所有责任或义务。此外，在某种程度上
   在适用法律允许的情况下，瑞萨电子放弃任何及所有明示或暗示的保证，
   关于本文档以及任何相关或随附的软件或硬件，包括但不限于
   适销性或特定用途适用性的默示保证。
8. 使用瑞萨电子产品时，请参阅最新的产品信息（数据表、用户手册、应用说明、"
   处理和使用半导体器件"中的可靠性手册等。），并确保使用条件在由
   瑞萨电子在最大额定值、工作电源电压范围、散热特性、安装等方面.瑞萨电子对因使用超出上述规定范围的瑞萨电子产品而引起的任何故障、故障或事故不承担任何及所有责任。

9.虽然瑞萨电子努力提高瑞萨电子产品的质量和可靠性，但半导体产品具有特定的特性，例如以一定速率发生故障和在某些使用条件下发生故障。除非瑞萨电子数据表或其他瑞萨电子文档中指定为高可靠性产品或用于恶劣环境的产品，否则瑞萨电子产品不受辐射电阻设计的限制。您有责任实施安全措施，以防止因火灾造成的人身伤害，伤害或损害的可能性，以及在瑞萨电子产品发生故障或故障时对公众造成的危险，例如硬件和软件的安全设计，包括但不限于冗余，火灾控制和故障预防，老化退化的适当处理或任何其他适当措施。由于单独评估微型计算机软件非常困难且不切实际，因此您有责任评估您制造的最终产品或系统的安全性。10.请联系瑞萨电子销售处，了解有关环境事宜的详细信息，例如每个瑞萨电子产品的环境兼容性。您有责任仔细和充分地调查规范包含或使用受控物质的适用法律和法规，包括但不限于欧盟RoHS指令，并根据所有这些适用法律和法规使用瑞萨电子产品。对于因您不遵守适用法律和法规而造成的损害或损失，瑞萨电子不承担任何及所有责任。11.瑞萨电子产品和技术不得用于或纳入任何适用的国内或国外法律或法规禁止制造、使用或销售的任何产品或系统。您应遵守由对双方或交易具有管辖权的任何国家的政府颁布和管理的任何适用的出口管制法律和法规。12.瑞萨电子产品的买方或分销商，或分销、处置或以其他方式向第三方销售或转让产品的任何其他方，有责任提前通知该第三方本文件规定的内容和条件。13.未经瑞萨电子事先书面同意，不得以任何形式全部或部分转载、复制或复制本文件。14.如果您对本文件或瑞萨电子产品中包含的信息有任何疑问，请联系瑞萨电子销售处。

(Note1) 本文件中使用的"瑞萨电子"是指瑞萨电子公司，也包括其直接或间接控制的子公司。
(Note2) "瑞萨电子产品"是指由瑞萨电子开发或制造或为瑞萨电子生产的任何产品。

(Rev.5.0-1 October 2020)

公司总部

TOYOSU FORESIA, 3-2-24 Toyosu,
江东区 东京135-0061 日本
www.renesas.com

Trademarks

瑞萨电子和瑞萨电子标志是瑞萨电子公司的商标.所有商标和注册商标均为其各自所有者的财产。

联络资料

有关产品、技术、文档的最新版本或离您最近的销售办事处的更多信息，请访问：www.renesas.com联系。