RX62N Group, RX621 Group

USB Host Flash Boot Loader

R01AN0892EJ0100
Rev.1.00
Mar 28, 2012

Introduction

This application note describes a USB host flash boot loader that operates, in single-chip mode, the RX62N and RX621 Group microcontroller USB 2.0 host/function module in host mode and rewrites the microcontroller's internal flash memory over a USB connection.

Note that this application note uses the sample code and drivers described in the following application notes.

- Erasing and writing internal flash memory
  RX600 Series Simple Flash API for RX600 Rev.2.20 (R01AN0544EU0220)
- USB communication
  Renesas USB Device USB Basic Firmware Rev.1.10 (R01AN0512EJ0110)
  Renesas USB Device USB Host Mass Storage Class Driver Rev.1.10 (R01AN0513EJ0110)
- FAT file system
  M3S-TFAT-Tiny: Fat File System Software Rev.1.00 (R20AN0038EJ0100)

This application note has the following features.

- An S format program stored on a USB memory device can be written to flash memory.
  When the connection of a USB mass storage device (USB memory) that holds an S format program is recognized, the microcontroller’s internal flash memory is erased and that program is written to the erased flash memory.
- The written program can be run.
  The S format program written to the microcontroller’s internal flash memory can be executed on the microcontroller.
- USB specifications
  USB 2.0 standard full speed transfers are supported.
  USB mass storage class bulk-only transport (BOT) is supported.
  USB mass storage subclass SFF-8070i (ATAPI) and SCSI are supported.

Target Device

RX62N and RX621 Group microcontrollers

If the code provided with this application note is used on any other microcontroller, it must be modified according to the specifications of that microcontroller and thoroughly tested.

Contents

1. Specifications

This application note's sample code operates on the RX62N RSK board.

If a reset is cleared with switch SW3 on the RX62N RSK not held down, the S format program (filename: download.mot) in the connected USB memory will be written to the microcontroller's internal flash memory. After this write has completed, if a reset is cleared with switch SW3 in the pressed state, the program written to the microcontroller's internal flash memory (also referred to as the downloaded code) will be executed.

Note that the area that this sample code can overwrite is limited to part of the user MAT and the area used by the sample code itself is not overwritten. See section 5.1, Operation Overview, for details.

The result of writing the program to internal flash memory is displayed in the LCD and LEDs on the RX62N RSK. See section 5.5, Sample Code LCD and LED Display, for details on the content displayed.

Table 1.1 lists the peripheral function used and their uses, and figure 1.1 shows an example of using this application note.

Table 1.1 Peripheral Functions and their Uses

Peripheral Function	Use
ROM (Flash memory used for storing program code)	The internal flash memory is programmed using ROM P/E mode.
USB 2.0 host/function module	Communication with the USB memory device

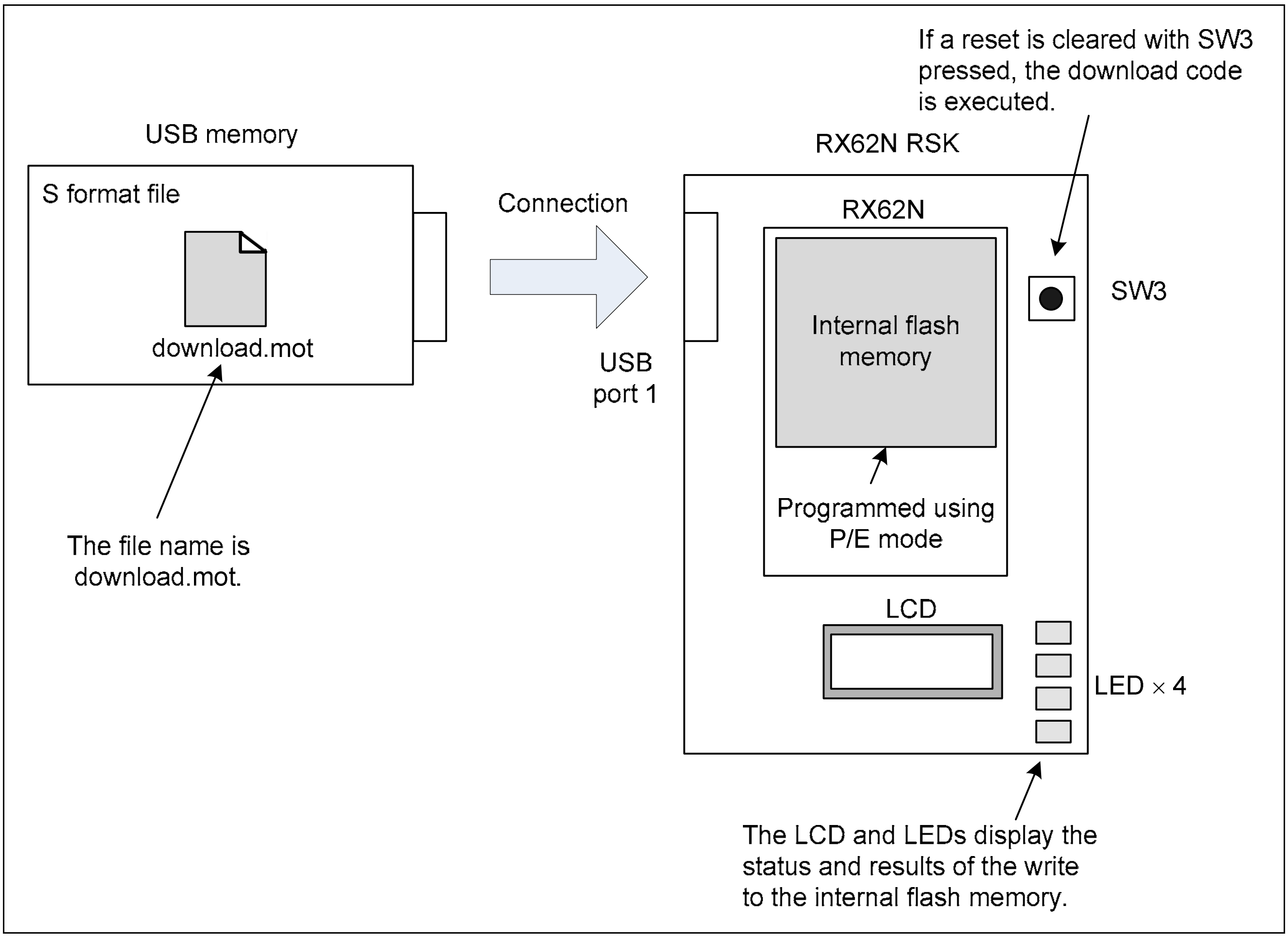


Figure 1.1 Usage Example

2. Confirmed Operating Conditions

The sample code provided with this application note has been confirmed to operate under the following conditions.

Table 2.1 Confirmed Operating Conditions

Item	Description
Microcontroller used	RX62N Group (R5F562N8BDBG)
Device used	R5F562N8BDBG
Operating frequencies	• EXTAL: 12 MHz • ICLK: 48 MHz • BCLK: 48 MHz • PCLK: 24 MHz • UCLK: 48 MHz
Operating voltage	3.3 V
Integrated development environment	Renesas Electronics High-performance embedded Workshop Version 4.09.00.007
C compiler	Renesas Electronics RX Standard Toolchain Version 1.1.0.0
	'-cpu=rx600 -include="$(WORKSPDIR)\WorkSpace\USBCSTDFW\include"," $(WORKSPDIR)\WorkSpace\USBSTDFW\include"," $(WORKSPDIR)\WorkSpace\USB2STDFW\include"," $(WORKSPDIR)\WorkSpace\SmplMain\APL"," $(WORKSPDIR)\WorkSpace\HwResourceForUSB"," $(WORKSPDIR)\WorkSpace\MSCCFW\include"," $(WORKSPDIR)\WorkSpace\MSC2FW\TFAT\lib_src"," $(WORKSPDIR)\WorkSpace\MSC2FW\include"," $(WORKSPDIR)\WorkSpace\FLASH" -define=USB2_FUNCSEL_PP=USBC_HOST_PP, USBC_FW_PP=USBC_FW_NONOS_PP,USBC_TFAT_USE_PP=1 -output=obj="$(CONFIGDIR)\$(FILELEAF).obj" -debug -nostuff -optimize=0 -nologo
Operating mode	Single-chip mode
Version of the sample code	Version 1.00
Board used	The RSK + RX62N packed with the Renesas Development Tools (catalog number: R0K5562N0S000BE) is used.

3. Related Application Notes

The following application notes are related to this document and should be referred to when using this application note.

- Renesas USB Device USB Basic Firmware Rev.1.10 (R01AN0512EJ)
- Renesas USB Device USB Host Mass Storage Class Driver Rev.1.10 (R01AN0513EJ)
- RX600 Series Simple Flash API for RX600 Rev.2.20 (R01AN0544EU)
- M3S-TFAT-Tiny: FAT File System Software Rev.1.00 (R20AN0038EJ)

4. Hardware Description

4.1 List of Used Pins

Table 4.1 lists the pins and functions used.

Table 4.1 Used Pins and their Functions

Pin Name	I/O	Description
USB1_DP	Input/Output	D+ I/O pin of the port 1 USB on-chip transceiver This pin should be connected to the D+ pin of the USB bus.
USB1_DM	Input/Output	D– I/O pin of the port 1 USB on-chip transceiver This pin should be connected to the D– pin of the USB bus.
USB1_VBUSEN-B	Output	VBUS (5 V) supply enable signal for port 1 external power supply chip
USB1_OVRCURA	Input	Port 1 external overcurrent detection signals should be connected to these pins. VBUS comparator signals should be connected to these pins when the OTG power supply chip is connected.
P07	Input	Sample code mode selection pin
P02	Output	LED connection pin
P03	Output	LED connection pin
P05	Output	LED connection pin
P34	Output	LED connection pin
P84	Output	LCD module control pin
P85	Output	LCD module control pin
P94	Output	LCD module control pin
P95	Output	LCD module control pin
P96	Output	LCD module control pin
P97	Output	LCD module control pin

5. Software Description

5.1 Operation Overview

5.1.1 Operation after a reset is cleared

After a reset is cleared, the sample code checks the state of switch SW3 (pin P07). If this switch is not being pressed (if pin P07 is high), it runs the USB host flash boot loader and programs the internal flash memory over the USB bus. If this switch is being pressed (if pin P07 is low), it runs the download code.

Figure 5.1 shows the operation after a reset.

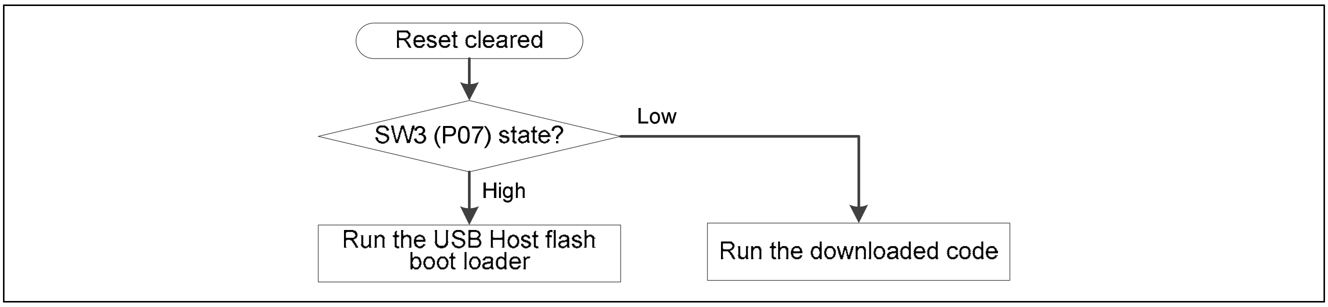
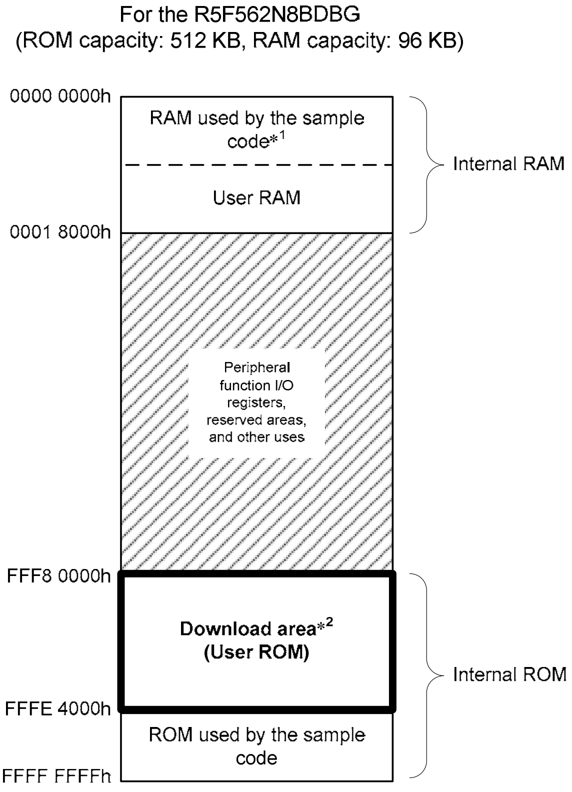


Figure 5.1 Operation After a Reset is Cleared

5.1.2 Object of Overwriting

The object area that the USB host flash boot loader overwrites is restricted to a certain part of the user MAT (referred to as the download area in this document). The area used for the sample code itself, FFFE 4000h to FFFF FFFFh, is not overwritten.

Figure 5.2 shows the memory allocation.



Notes: 1. User code may use this area when the download code runs.
2. Area to which the download code is written.

Figure 5.2 Memory Allocation

5.1.3 Programming the Download Area

The USB host flash boot loader uses the following procedure to program the download area. Figure 5.3 shows the download area programming procedure.

(1) Monitor the USB device connection.
(2) If a USB connection is detected, acquire the information for the connected device and determine whether or not access is possible.
(3) If access to the connected USB device (USB memory) is possible, search for an S format file with the filename download.mot.
(4) If an S format file with the filename download.mot is recognized, erase the download area.
(5) After erasing the download area, read 2048 bytes of data from the S format file in the USB memory and store it in internal RAM.
(6) After storing the data in RAM, analyze the data and write it to the download area in 256 byte units.
(7) Repeat the processing of steps (5) and (6) until all the data has been written.
Note that the end of the S format file is recognized by the occurrence of an end record (an S7, S8, or S9 record).
(8) If the erase and programming of the download area completes normally, report that normal completion in the LCD and LEDs connected to the I/O ports.

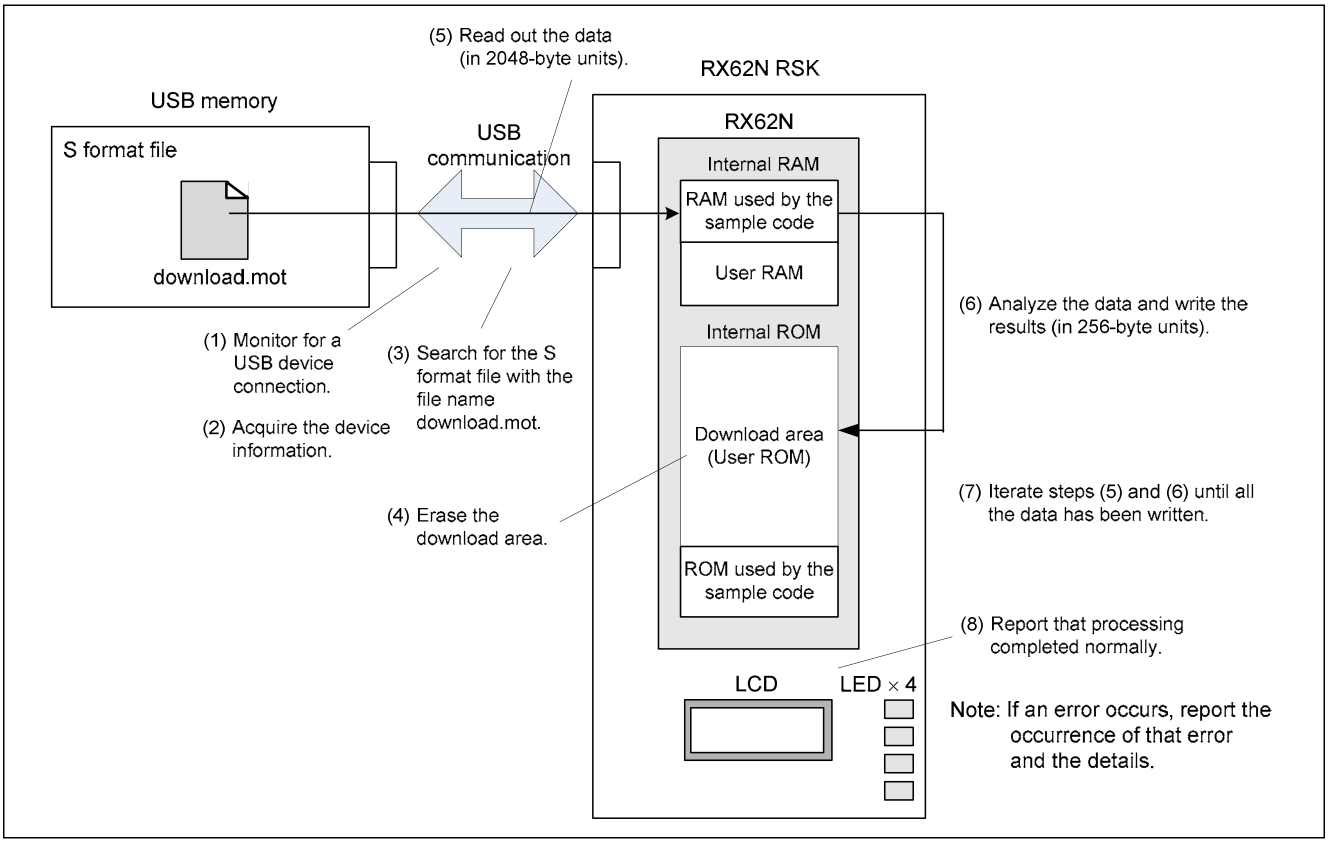


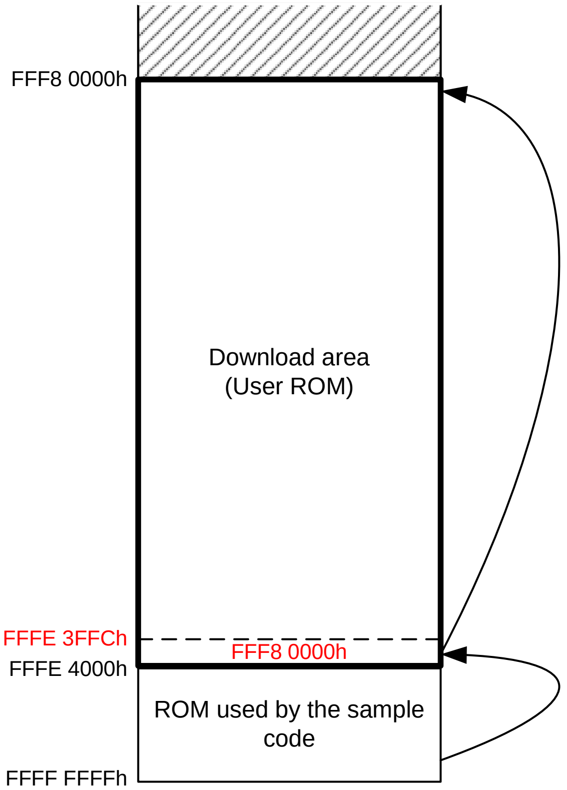
Figure 5.3 Programming the Download Area

Note: If an error occurs during sample code execution, the details of that error are reported in the LCD and LEDs. See section 5.5, Sample Code LCD and LED Display, for details on error occurrence conditions and the LCD and LED display.

5.2 Download Code Execution Start Position

If the switch SW3 state is the low level when a microcontroller reset is cleared, the sample code will run the download code. At this time, the starts executing the download code from the address stored at location FFFE 3FFCh. That is, the reset vector for the download code is FFFE 3FFCh. Therefore the download code must store its start address at location FFFE 3FFCh.

Figure 5.4 shows the reset vector for the download code.



If the switch SW3 state is the low level execution starts from the address written to location FFFE 3FFCh, the download code reset vector.

Note: Processing will stop if the value at the download code reset vector is FFFF FFFFh.

Figure 4 Download Code Reset Vector

Note: If nothing is written to the download code reset vector (that is, if the value at the download code reset vector is FFFF FFFFh), the sample code executes a while(1) infinite loop to stop processing.

5.3 Software Structure of the Sample Code

The sample code uses the Renesas USB Device USB Basic Firmware and the Renesas USB Device USB Host Mass Storage Class Driver for USB communication.

It uses the M3S-TFAT-Tiny: Fat File System Software as its FAT file system.

It also uses the RX Family RX600 Simple Flash API for erase and write processing for the internal flash memory.

Figure 5.5 shows software structure of the sample code and table 5.1 gives an overview of the software.

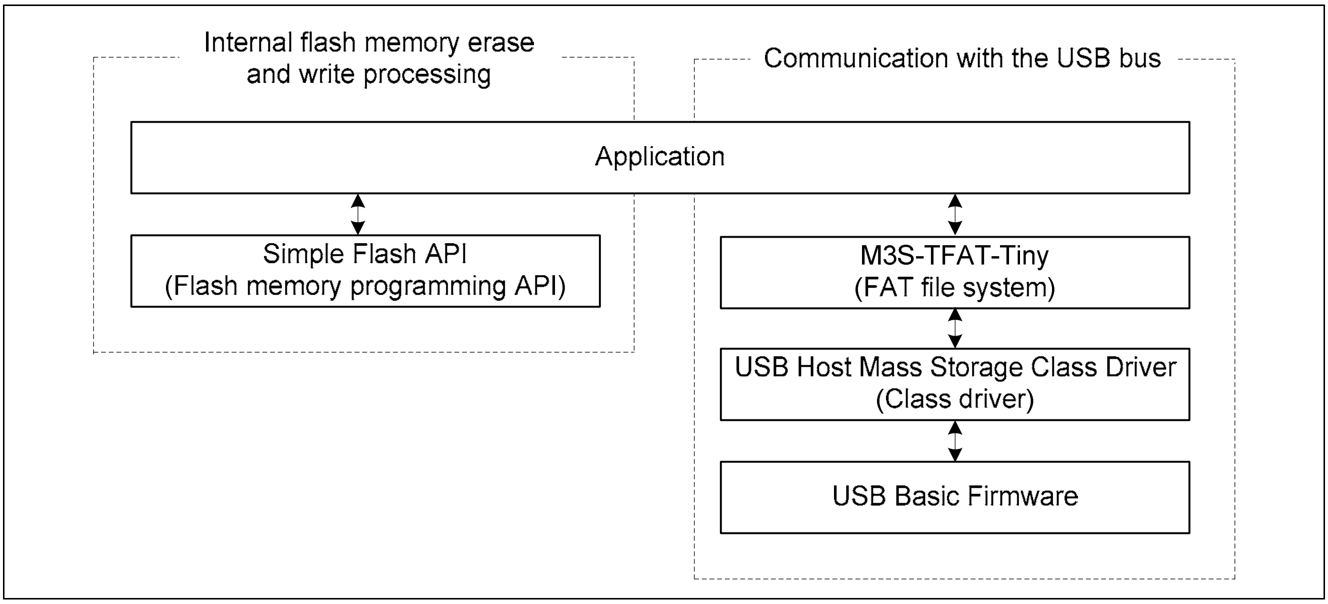


Figure 5.5 Software Structure of the Sample Code

Table 5.1 Software Overview

Module	Overview
Application	The application uses FAT library functions to read an S format program from the USB memory. It then uses the Simple Flash API functions to erase the internal flash memory and write that program to internal flash memory.
Simple Flash API for RX600	API used to erase and write the internal flash memory
M3S-TFAT-Tiny	A FAT file system that supports FAT12 and FAT16.
USB Host Mass Storage Class Driver	A class driver that supports the USB mass storage class bulk-only transport (BOT) protocol.
USB Basic Firmware	A sample program that controls the USB interface.

5.4 Data Flow During Write

Figure 5.6 shows the data flow internal to the microcontroller when the download code is written to flash memory.

(1) The data acquired from the USB driver is transferred to a receive ring buffer.
(2) One record of the S format data is copied to an S format buffer (this is ASCII data).
(3) At the same time as analyzing the S format data header section, the ASCII data is converted to binary and stored in an S format buffer (for binary data).
See section 7, S Format, for the S format data analysis specifications used in this application note.
(4) The data is stored in a write buffer.
In the RX62N and RX621 group microcontrollers, data is written to the user MAT in units of 256 bytes. Therefore, the sample code iterates steps (2) to (4) above until a total of 256 bytes of write data has been stored in the write buffer. Also, if the total amount of write data exceeds 256 bytes, the excess data is stored temporarily and used for the next write of 256 bytes of data.
(5) The assembled 256 bytes of data are written to flash memory using the Simple Flash API.

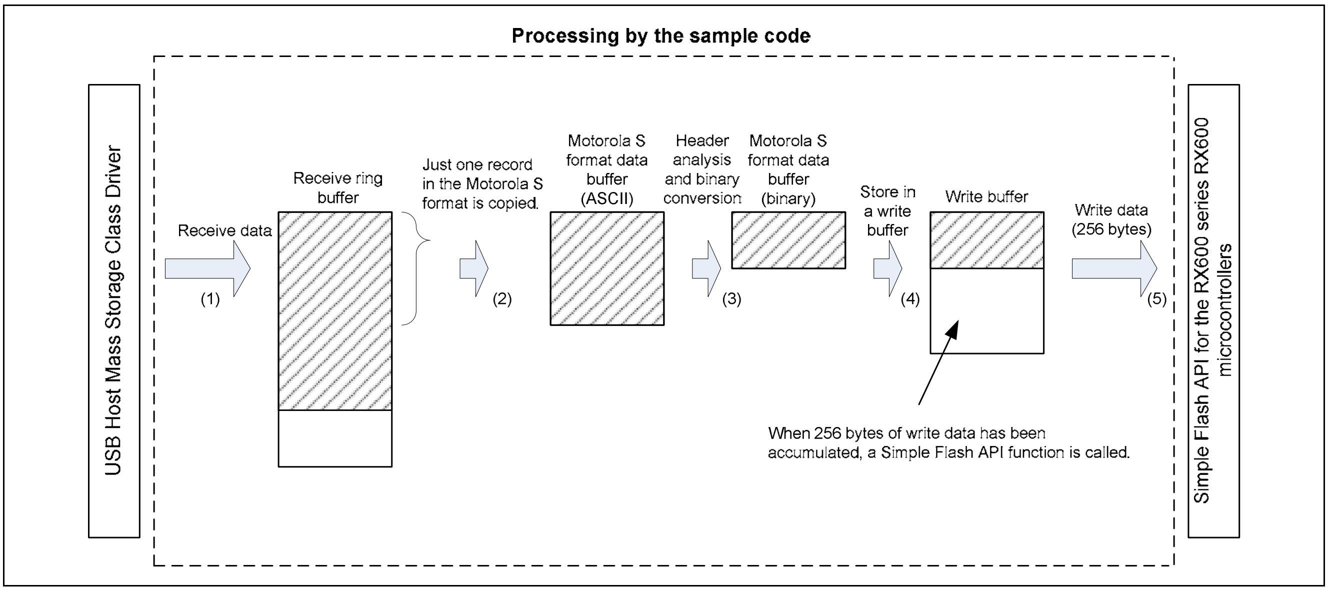


Figure 5.6 Data Flow During Write

Figure 5.7 shows the data structures used when writing data to flash memory.

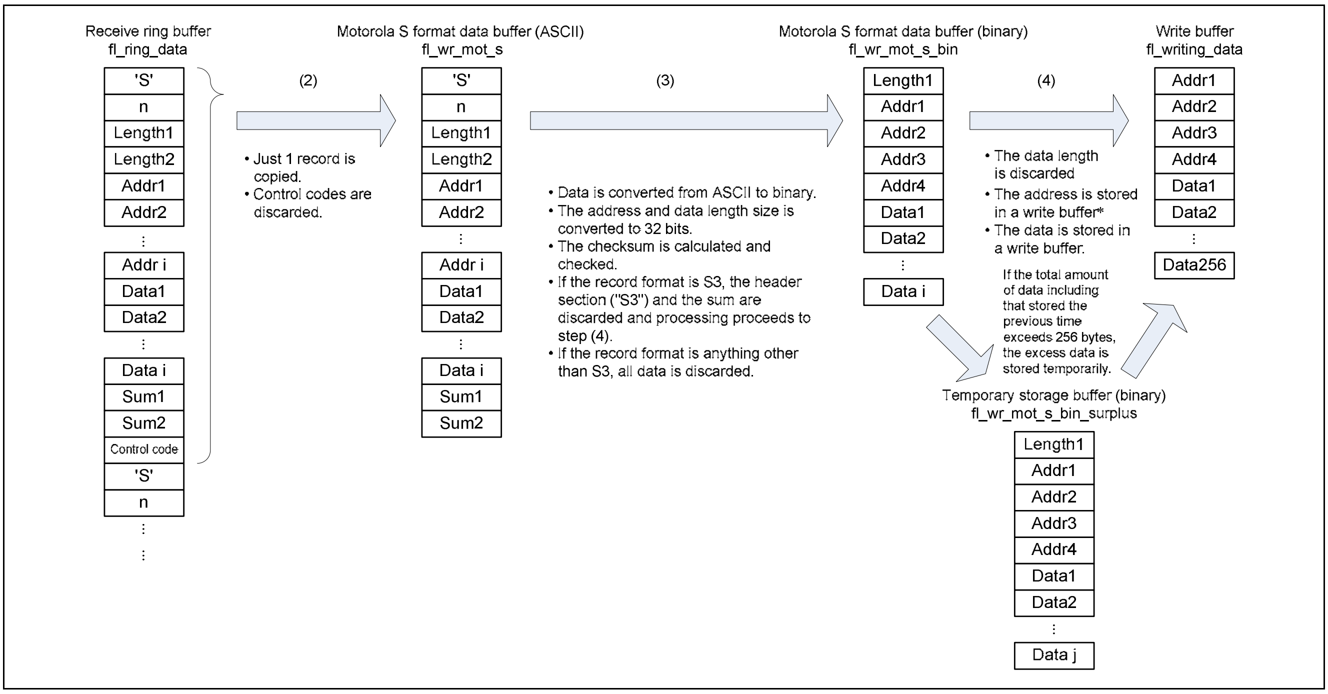


Figure 5.7 Data Structures Used for Writing

Note: In the RX62N and RX621 group microcontroller internal flash memory, a start address used for a write operation must be aligned on a 256-byte boundary. Accordingly, the sample code performs processing to assure that write start addresses are aligned on 256-byte boundaries when storing addresses to write buffers. See the flowchart in section 5.12.11, Download area write data creation, for details on this processing.

5.5 Sample Code LCD and LED Display

The sample code displays the state of progress of the program and the results in the LCD and LEDs mounted on the RX62N RSK board. Table 5.2 lists the LED patterns displayed by the sample code.

Table 5.2 Sample Code LED Display

O: On, ×: Off

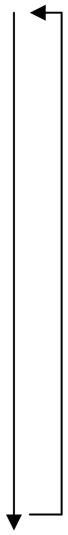
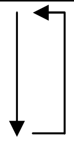
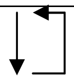
LED Display State					
LED3	**LED2**	**LED1**	**LED0**	**Order**	**Description**
×	×	×	×		The LEDs display a binary counter during the download processing. The LED display is updated each time the write of a single block (16 KB write) completes.
×	×	×	O		
×	×	O	×		
×	×	O	O		
×	O	×	×		
×	O	×	O		
×	O	O	O		
O	×	×	×		
O	×	×	O		
O	×	O	×		
O	×	O	O		
O	O	×	×		
O	O	×	O		
O	O	O	×		
O	O	O	O		
×	×	×	O		When the sample code completes normally, a shifting display pattern is displayed in the LEDs. The LED display is updated once every 500 ms.
×	×	O	×		
×	O	×	×		
O	×	×	×		
×	×	×	×		If the sample code terminates abnormally, the LEDs blink. The LED display is updated once every 500 ms.
O	O	O	O		

Table 5.3 lists the LCD patterns displayed by the sample code.

Table 5.3 Sample Code LCD Display

LCD Display State	Description
FLASH BOOT	Displayed when the USB host flash boot loader is run after a reset is cleared.
DETACH	Indicates that no USB was connected after a reset was cleared.*[1]
ATTACH	Indicates that there was a USB connected after a reset was cleared. *[1]
PROGRAM UPDATE..	Indicates that download processing is in progress.
ERROR!! D OPEN	Indicates that no accessible drive was detected. (Drive open error)
ERROR!! D MOUNT	Indicates that drive mount processing failed. (Drive mount error)
ERROR!! F OPEN	Indicates that file open processing failed. (File open error)
ERROR!! F READ	Indicates that file read processing failed. (File read error)
ERROR!! F CLOSE	Indicates that file close processing failed. (File close error)
ERROR!! SUM	See section 7, S Format. (Checksum error)
ERROR!! MOTS	See section 7, S Format. (Format error)
ERROR!! ERASE	Indicates that erase of the download area failed. (Erase error)
ERROR!! WRITE	Indicates that write of the download area failed. (Write error)
ERROR!! ADDRESS	See section 7, S Format. (Address error)
ERROR!! VERIFY	Indicates that the result of verifying the data written to the download area was that an abnormality was detected. (Verify error)
ERROR!! F END	Indicates that no S format end record had been received even though an end of file was detected by the FAT library. (File end error)
ERROR!! ILL DET	Indicates that a USB detach was detected during either drive open or download processing. (Illegal detach error)

Note: 1. The DETACH display used when no USB is connected, and the ATTACH display used when a USB is connected, follow the specifications of the USB Host Mass Storage Class Driver.

5.6 Memory Requirements

Table 5.4 lists the required memory sizes.

Table 5.4 Memory Requirements

Memory Used	Size	Notes
ROM	105,028 bytes	Since the sample code is allocated to locations FFFE 4000h to FFFF FFFFh, the amount of ROM that can be written is the total ROM capacity minus 114,688 bytes.
RAM	28,476 bytes	The user code can use this area when it runs.

Note: The sizes of required memory areas vary with the version and compiler options of the C compiler.

5.7 File Structure

Table 5.5 lists the files that make up the sample code.

Table 5.5 File Structure

File Name	Description	Notes
r_flash_api_rx600.c	The RX600 Series RX600 Simple Flash API program	For details, see the RX600 Series RX600 Simple Flash API application note.
r_flash_api_rx600.h	External reference include header for the RX600 Series RX600 Simple Flash API program.	For details, see the RX600 Series RX600 Simple Flash API application note.
r_flash_api_rx600_private.h	External reference include header for the RX600 Series RX600 Simple Flash API program.	For details, see the RX600 Series RX600 Simple Flash API application note.
r_flash_api_rx600_config.h	Parameter settings include header for the RX600 Series RX600 Simple Flash API program.	For details, see the RX600 Series RX600 Simple Flash API application note.
mcu_info.h	Parameter settings include header for the RX600 Series RX600 Simple Flash API program.	For details, see the RX600 Series RX600 Simple Flash API application note.
r_Flash_main.c	Flash programming data processing	
r_Flash_main.h	External reference include header for the flash programming data processing	
r_Flash_buff.c	USB receive ring buffer related processing	
r_Flash_buff.h	External reference include header for the USB receive ring buffer related processing	
TrgtPrgDmmy.c	Dummy program for allocating the download code area	
Other files	The USB Host Mass Storage Class Driver program	See the Renesas USB device USB Host Mass Storage Class Driver and USB Basic Firmware application notes for details.

5.8 Constants

Table 5.6 lists the constants used in the sample code.

Table 5.6 Constants Used in the Sample Code

Constant	Set Value	Description
FL_RINGBUFF_SIZE	4096	Size of the USB data reception ring buffer
FL_MOTS_ADDR_SIZE	4	S format data address buffer size
FL_MOTS_SUM_SIZE	1	S format data checksum buffer size
FL_START_BLOCK_NUM	10	First block in the download area
FL_END_BLOCK_NUM	37	Last block in the download area
FL_START_WRITE_ADDRESS	FFF80000h	First address in the download area
FL_END_WRITE_ADDRESS	FFFE3FFFh	Last address in the download area
FL_USB_RCV_BLANK_SIZE	2048	Ring buffer capacity
FL_MODE_ENTRY_WAIT_LCD_PERIOD	100000	Wait time at mode entry
FL_UPDATE_WAIT_LED_PERIOD	64	Time for the interval between LED display updates during download processing
FL_ERROR_WAIT_LED_PERIOD	500	Time for the interval between LED display updates during error handling
FL_DONE_WAIT_LED_PERIOD	500	Time for the interval between LED display updates during normal termination
FL_UPDATE_FILE_NAME	"download.mot"	Download code file name
FL_PORT_MDE	PORT0.PORT.BIT.B7	PORT register for the port connected to SW3
FL_DDR_MDE	PORT0.DDR.BIT.B7	DDR register for the port connected to SW3
FL_ICR_MDE	PORT0.ICR.BIT.B7	ICR register for the port connected to SW3

5.9 Structures and Unions

Figure 5.8 shows the structures and unions used in the sample code.

```
/* buffer for mot S format data */
typedef struct {
    uint8_t type[2];                  /* "S0", "S1" and so on */
    uint8_t len[2];                   /* "0-255" */
    uint8_t addr_data_sum[512];
} Fl_prg_mot_s_t;

/* buffer for write data
     (this data is the converted data from mot S format data) */
typedef struct {
    uint8_t len;
    uint32_t addr;
    uint8_t data[256];
} Fl_prg_mot_s_binary_t;

/* buffer for writing flash */
typedef struct {
    uint32_t addr;
    uint8_t data[256];
} Fl_prg_writing_data_t;
```

Figure 5.8 Structures and Unions Used in the Sample Code

5.10 Functions

Table 5.7 lists the functions. Note, however, that the USB Mass Storage Class Driver, Simple Flash API, and FAT file system software functions are not shown here.

Table 5.7 Functions

Function	Description
R_Fl_Mode_Entry	Mode selection
R_Fl_Flash_Update	Main function for flash memory write processing
R_Fl_EraseTrgtArea	Erase processing
R_Fl_Ers_EraseFlash	Erase download area
R_Fl_PrgramTrgtArea	Write download area
R_Fl_Prg_PrgramTrgtArea	Write processing
R_Fl_Prg_StoreMotS	Store S format data
R_Fl_Prg_ProcessForMotS_data	Header analysis, binary conversion, and write of an S format record
R_Fl_Prg_MotS_AsciiToBinary	Convert S format data from ASCII to binary
R_Fl_Prg_MakeWriteData	Create write data for the download area
R_Fl_Prg_WriteData	Write to download area
R_Fl_Prg_ClearMotSVariables	Clear the variables related to the S format data
R_Fl_Run_StopUSB	Stop USB
R_Fl_RcvDataString	Store USB receive data
R_Fl_Error	Error handling
R_Fl_RingCheckBlank	Verify amount of free space in receive ring buffer
R_Fl_RingEnQueue	Store data in receive ring buffer
R_Fl_RingDeQueue	Read data from receive ring buffer
R_Fl_RingCheck	Verify data count in receive ring buffer
R_Fl_AsciiToHexByte	Convert data from ASCII to binary

5.11 Function Specifications

This section shows the specifications of the functions in the sample code.

R_Fl_Mode_Entry

Overview	Mode entry
Header	r_Flash_main.h, iodefine.h
Declaration	void R_Fl_Mode_Entry(void)
Description	• Checks the state of SW3. • If SW3 is low, runs the USB host flash boot loader. • If SW3 is high, runs the download code.
Arguments	None
Return values	None
Notes	

R_Fl_Flash_Update

Overview	Flash memory write processing main routine
Header	r_Flash_main.h
Declaration	void R_Fl_Flash_Update (void)
Description	• Calls the function that performs the download area erase processing. • Calls FAT library functions to read data from the S format file in the USB memory. • Calls functions that analyze the S format data and write the data to the download area. • Calls an error handler if execution of a FAT library function fails.
Arguments	None
Return values	None
Notes	

R_Fl_EraseTrgtArea

Overview	Erase processing
Header	None
Declaration	static void R_Fl_EraseTrgtArea(void)
Description	• Calls the function that erases the download area. • Calls an error handler if erase of the download area fails.
Arguments	None
Return values	None
Notes	

R_Fl_Ers_EraseFlash

Overview	Erase download area
Header	None
Declaration	static Fl_API_SMPL_rtn_t R_Fl_Ers_EraseFlash(void)
Description	Erases the download area.
Arguments	None
Return values	• If the erase operation completes normally: FLASH_API_SAMPLE_OK • If the erase operation does not complete normally: FLASH_API_SAMPLE_NG
Notes	The processor status word (PSW) interrupt priority level (IPL) is modified to prevent ROM access by interrupts during the erase operation.

R_Fl_PrgramTrgtArea

Overview	Write download area
Header	None
Declaration	static void R_Fl_PrgramTrgtArea (void)
Description	• Calls the function that performs the required write processing. • Calls an error handler if the end of file is reached before receiving an S format end record.
Arguments	None
Return values	None
Notes	

R_Fl_ Prg_PrgramTrgtArea

Overview	Write processing
Header	None
Declaration	static Fl_API_SMPL_rtn_t R_Fl_Prg_PrgramFlash(void)
Description	• If there is data in the receive ring buffer, calls the function that analyzes a single S format record. • When a single S format record has been analyzed, calls the function that performs header analysis, conversion to binary, and writing to the download area.
Arguments	None
Return values	• If writing to the download area terminates normally: FLASH_API_SAMPLE_OK • If writing to the download area did not terminate: FLASH_API_SAMPLE_NG
Notes	

R_Fl_Prg_StoreMotS

Overview	Store S format data
Header	None
Declaration	static Fl_API_SMPL_rtn_t R_Fl_Prg_StoreMotS(uint8_t)
Description	• Stores the data passed in the argument as S format data one byte at a time. • Discards all data until the first 'S' (ASCII data) is acquired.
Arguments	First argument: mot_data : S format data
Return values	• If a single S format data item (from the 'S' to the checksum) was stored: FLASH_API_SAMPLE_OK • If a single S format data item was not stored: FLASH_API_SAMPLE_NG
Notes	• This function is used by passing S format data 1 byte at a time in the argument. • The checksum is not checked.

R_Fl_Prg_ProcessForMotS_data

Overview	Header analysis, binary conversion, and write of an S format record
Header	None
Declaration	static void R_Fl_Prg_ProcessForMotS_data(void)
Description	• Analyses the S format header and calls the function that converts to binary. • Calls the function that stores data in a write buffer. • Calls the function that writes data to the download area. • Calls an error handler if data that differs from the Motorola S format is received.
Arguments	None
Return values	None
Notes	

R_Fl_Prg_MotS_AsciiToBinary

Overview	Convert S format data from ASCII to binary
Header	None
Declaration	static Fl_API_SMPL_rtn_t R_Fl_Prg_MotS_AsciiToBinary (Fl_prg_mot_s_t *, Fl_prg_mot_s_binary_t *)
Description	• Converts S format data in ASCII code to binary data. • Verifies the checksum of the converted binary data. • Calls an error handler if data that differs from the Motorola S format is received. • Calls an error handler if a checksum error occurs.
Arguments	First argument: *tmp_mot_s : Pointer to S format data in ASCII Second argument: *tmp_mot_s_binary : Pointer to variable that holds the converted to binary data
Return values	• If conversion completed normally: FLASH_API_SAMPLE_OK • If conversion does not complete normally: FLASH_API_SAMPLE_NG
Notes	

R_Fl_Prg_MakeWriteData

Overview	Create write data for the download area
Header	None
Declaration	static Fl_API_SMPL_rtn_t R_Fl_Prg_MakeWriteData(void)
Description	Creates data divided at each 256-byte unit.
Arguments	None
Return values	• If creation of 256 bytes of write data completed: FLASH_API_SAMPLE_OK • If creation of 256 bytes of write data did not complete: FLASH_API_SAMPLE_NG
Notes	

R_Fl_Prg_WriteData

Overview	Write to download area
Header	None
Declaration	static Fl_API_SMPL_rtn_t R_Fl_Prg_WriteData(void)
Description	• Performs the write to the download area. • Verifies the data written. • Calls the error handler if the write failed. • Calls an error handler if a verify error occurs.
Arguments	None
Return values	• If the write completed normally: FLASH_API_SAMPLE_OK • If the write did not complete normally: FLASH_API_SAMPLE_NG
Notes	The processor status word (PSW) interrupt priority level (IPL) is modified to prevent ROM access by interrupts during the write operation.

R_Fl_Prg_ClearMotSVariables

Overview	Clear the variables related to the S format data
Header	None
Declaration	static void R_Fl_Prg_ClearMotSVariables(void)
Description	Clears the variables related to the S format data.
Arguments	None
Return values	None
Notes	

R_Fl_Run_StopUSB

Overview	Stop USB
Header	r_Flash_main.h
Declaration	void R_Fl_Run_StopUSB(void)
Description	Stops the USB.
Arguments	None
Return values	None
Notes	

R_Fl_RcvDataString

Overview	Store USB receive data
Header	None
Declaration	static Fl_API_SMPL_rtn_t R_Fl_RcvDataString(void *, uint16_t)
Description	Stores the data received over the USB in a receive ring buffer.
Arguments	First argument: *tranadr : Pointer to a buffer to hold data received over the USB Second argument: *length : Length of the data received over the USB
Return values	• If the store completes: FLASH_API_SAMPLE_OK • If the receive ring buffer was full: FLASH_API_SAMPLE_NG
Notes	

R_Fl_Error

Overview	Error handling
Header	r_Flash_main.h
Declaration	void R_Fl_Error(Fl_err_tbl_num_t err_num)
Description	• Calls the USB stop function. • Displays the error in the LCD and LEDs.
Arguments	First argument: err_num : Error number
Return values	None
Notes	

R_Fl_RingCheckBlank

Overview	Verify amount of free space in receive ring buffer
Header	r_Flash_buff.h
Declaration	Fl_API_SMPL_rtn_t R_Fl_RingCheckBlank(void)
Description	Verifies whether or not there is space in the receive ring buffer for one data unit (2048 bytes) read by the file read function.
Arguments	None
Return values	• If there is adequate free space: FLASH_API_SAMPLE_OK • If there is not adequate free space: FLASH_API_SAMPLE_NG
Notes	

R_Fl_RingEnQueue

Overview	Store data in receive ring buffer
Header	r_Flash_buff.h
Declaration	Fl_API_SMPL_rtn_t R_Fl_RingEnQueue(uint8_t)
Description	Stores data in the receive ring buffer.
Arguments	First argument: enq_data : Data to be stored
Return values	• If the store completed: FLASH_API_SAMPLE_OK • If the buffer was full: FLASH_API_SAMPLE_NG
Notes	

R_Fl_RingDeQueue

Overview	Read data from receive ring buffer
Header	r_Flash_buff.h
Declaration	Fl_API_SMPL_rtn_t R_Fl_RingDeQueue(uint8_t *)
Description	Reads data from the receive ring buffer.
Arguments	First argument: *deq_data : Pointer to buffer to store data read
Return values	• If the data was read out normally: FLASH_API_SAMPLE_OK • If there was no data to read: FLASH_API_SAMPLE_NG
Notes	

R_Fl_RingCheck

Overview	Verify data count in receive ring buffer
Header	r_Flash_buff.h
Declaration	uint32_t R_Fl_RingCheck(void)
Description	Verifies the number of data items in the receive ring buffer.
Arguments	None
Return values	Returns the number of receive data items.
Notes	

R_Fl_AsciiToHexByte

Overview	Convert data from ASCII to binary
Header	r_Flash_buff.h
Declaration	uint8_t R_Fl_AsciiToHexByte(uint8_t, uint8_t)
Description	Converts a 2-byte ASCII coded data item to 1 byte of binary data.
Arguments	First argument: in_upper : ASCII code data (high order) Second argument: in_lower : ASCII code data (low order)
Return values	Returns the converted binary data.
Notes	

5.12 Flowcharts

5.12.1 Main USB Processing

Figure 5.9 shows the flowchart for main USB processing.

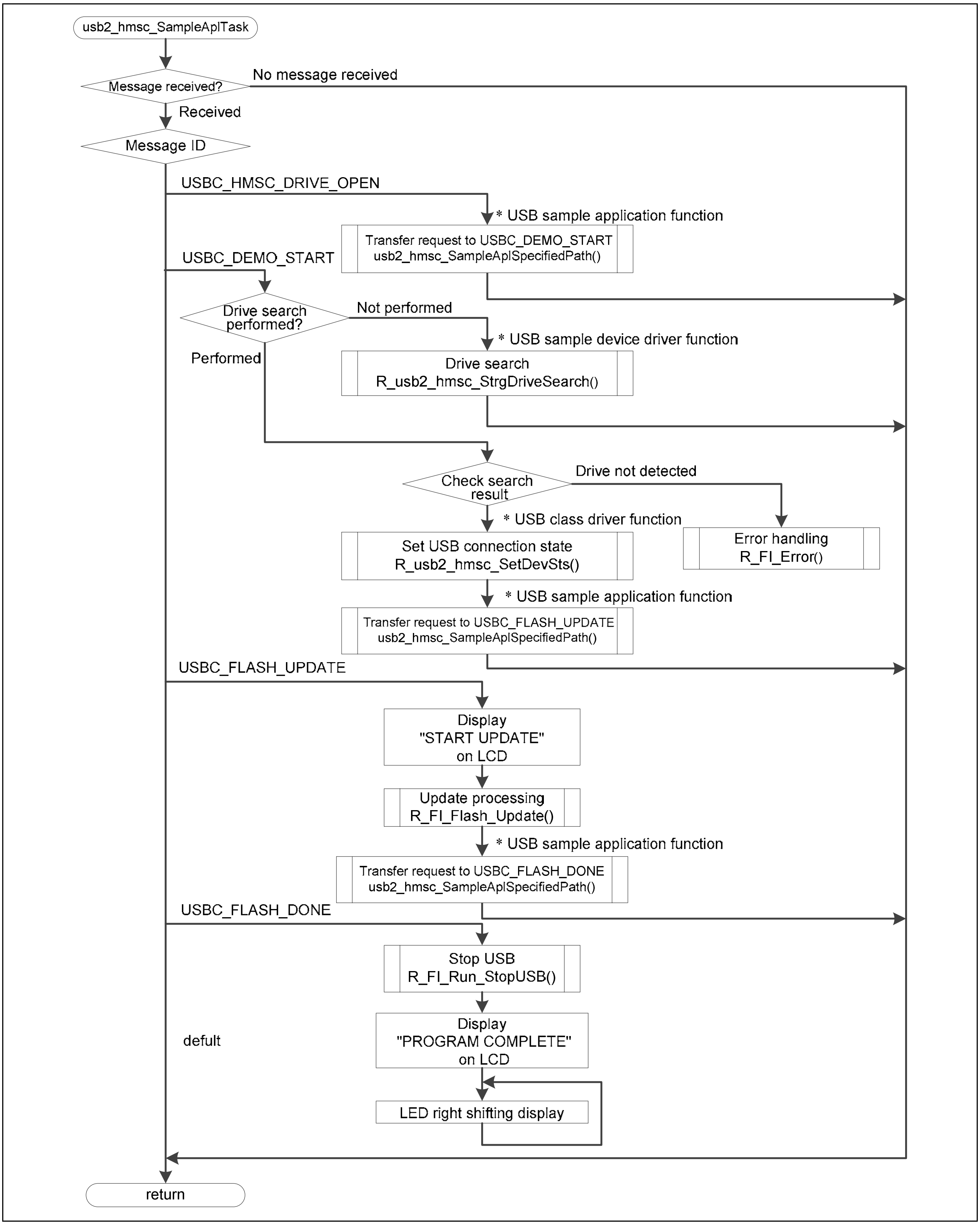


Figure 5.9 Main USB Processing

5.12.2 Mode Entry

Figure 5.10 shows the flowchart for mode entry.

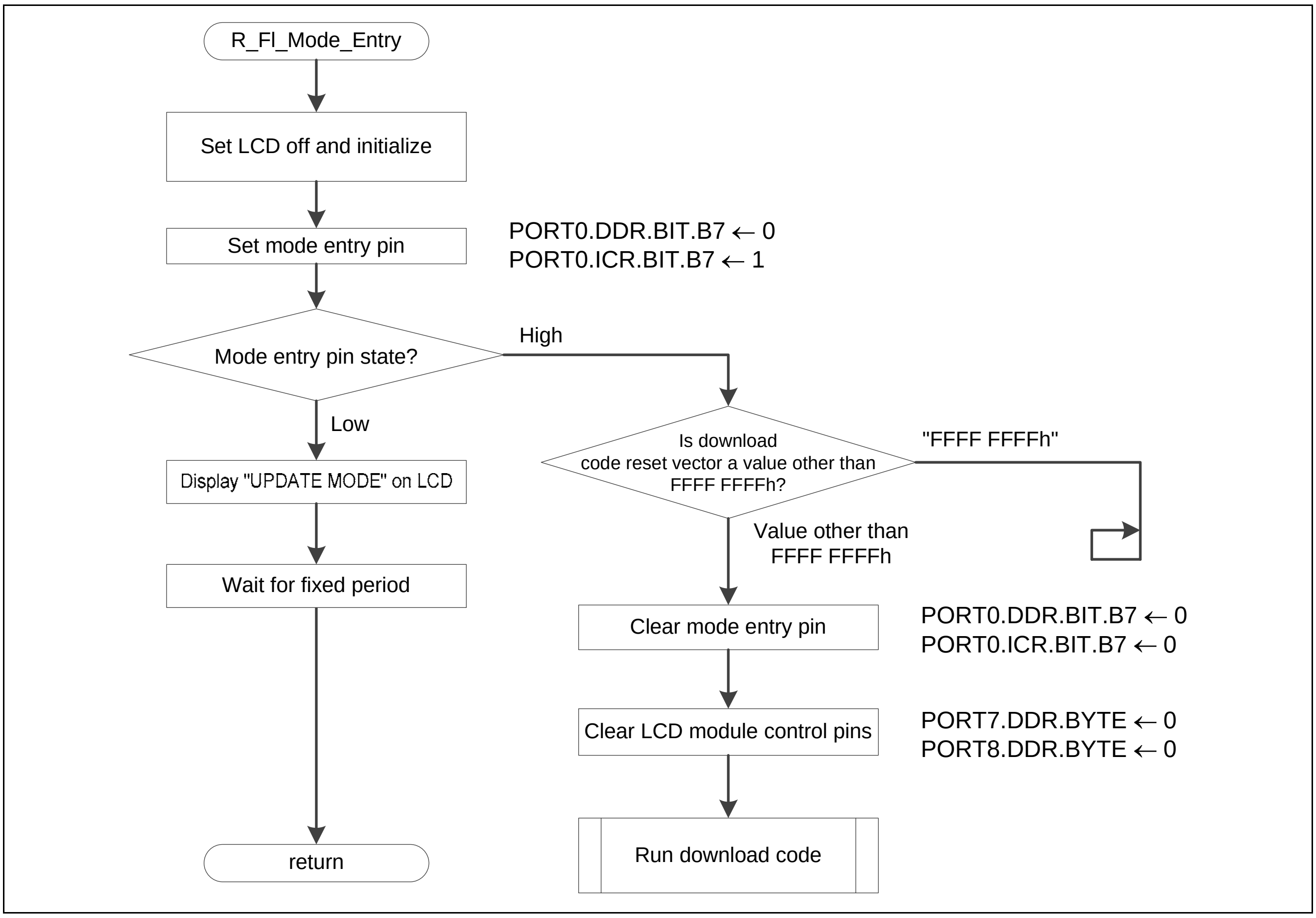


Figure 5.10 Mode Entry

5.12.3 Main Write Processing

Figure 5.11 shows the flowchart for main write processing.

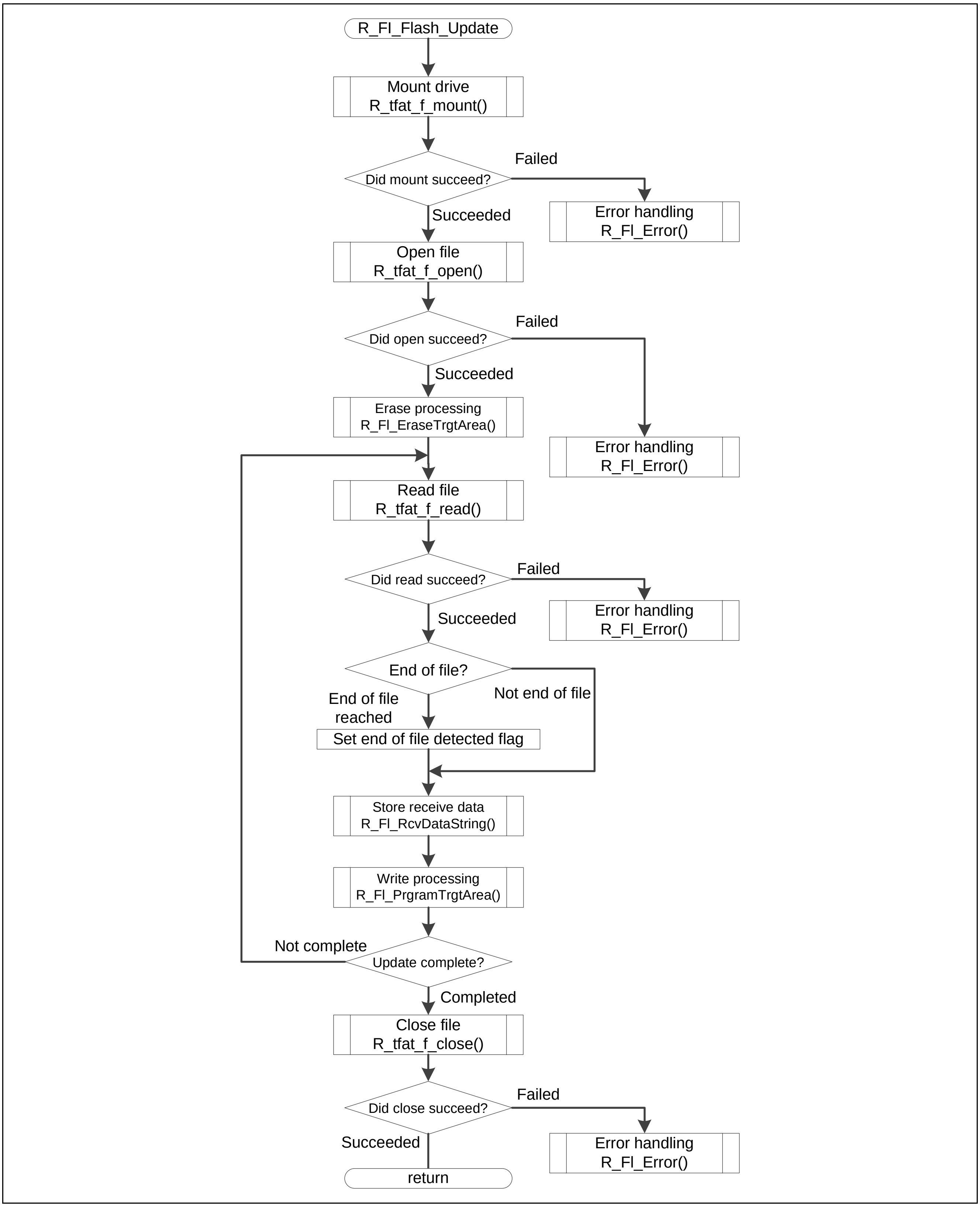


Figure 5.11 Main Write Processing

5.12.4 Erase Processing

Figure 5.12 shows the flowchart for erase processing.

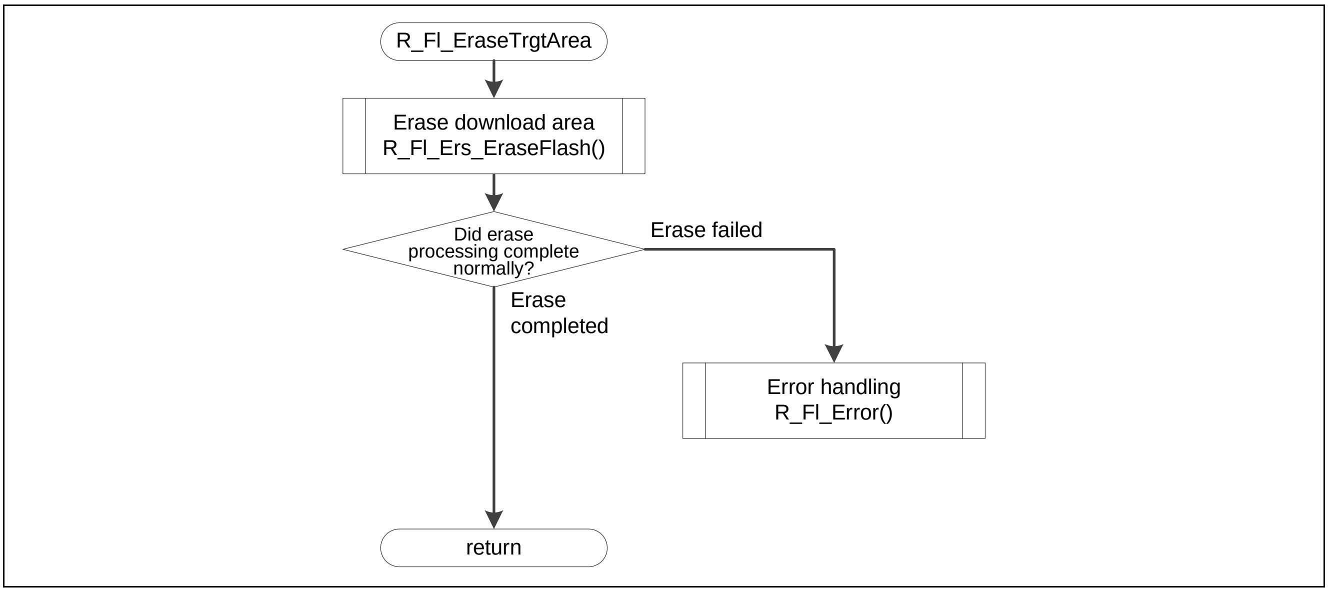


Figure 5.12 Erase Processing

5.12.5 Erase Download Area

Figure 5.13 shows the flowchart for erase download area.

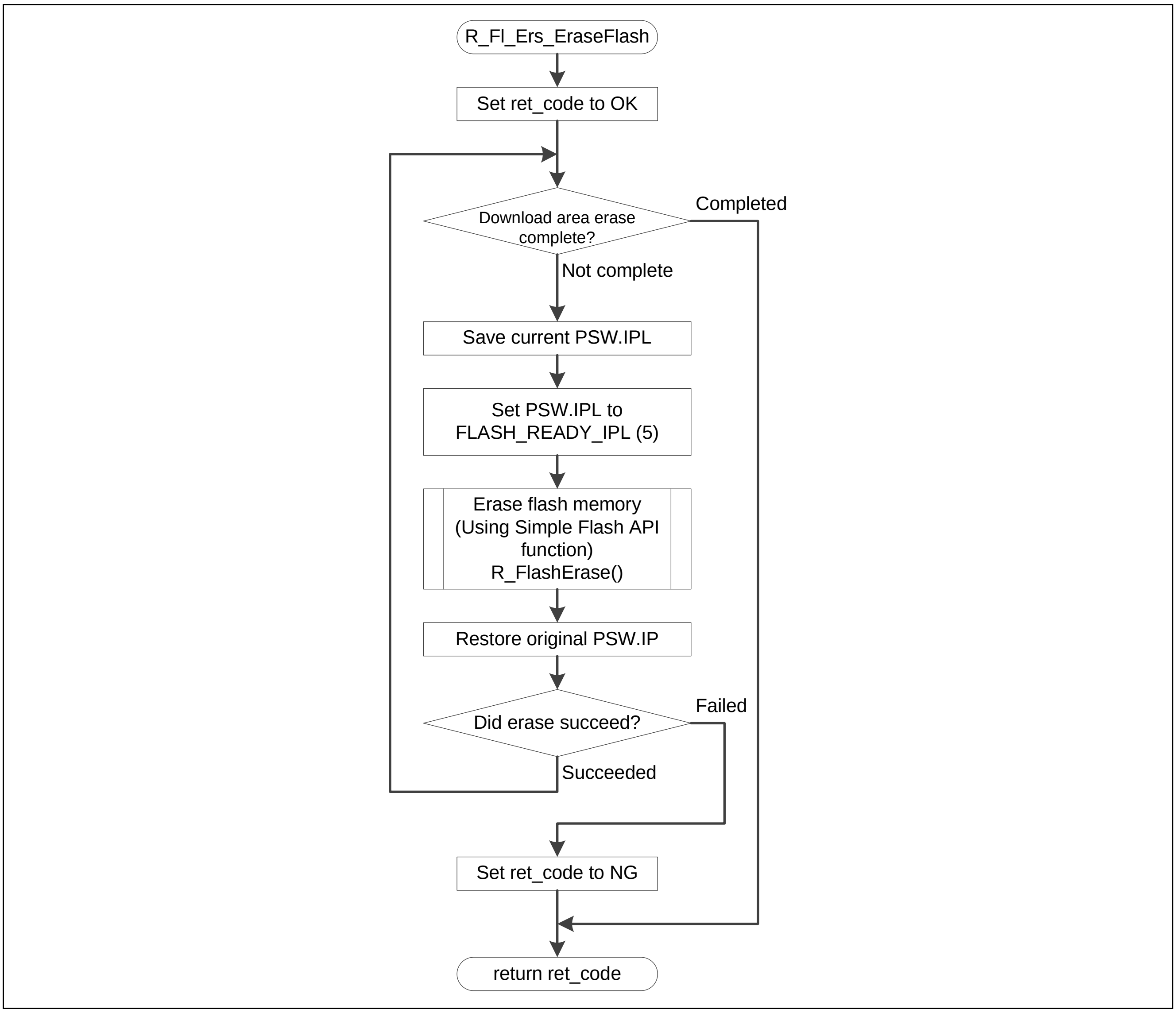


Figure 5.13 Erase Download Area

5.12.6 Write Processing

Figure 5.14 shows the flowchart for write processing.

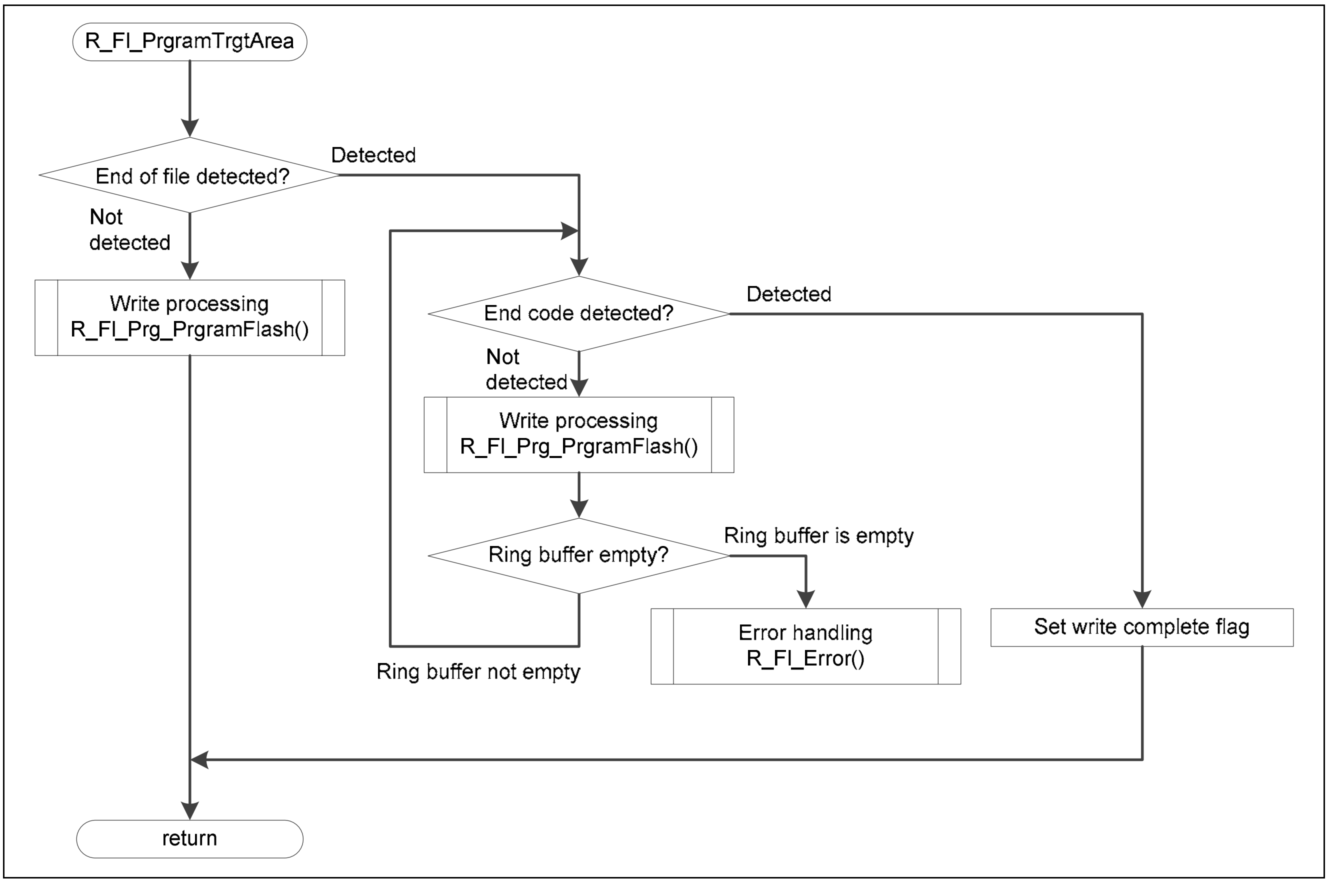


Figure 5.14 Write Processing

5.12.7 Download Area Write Operation

Figure 5.15 shows the flowchart for download area write operation.

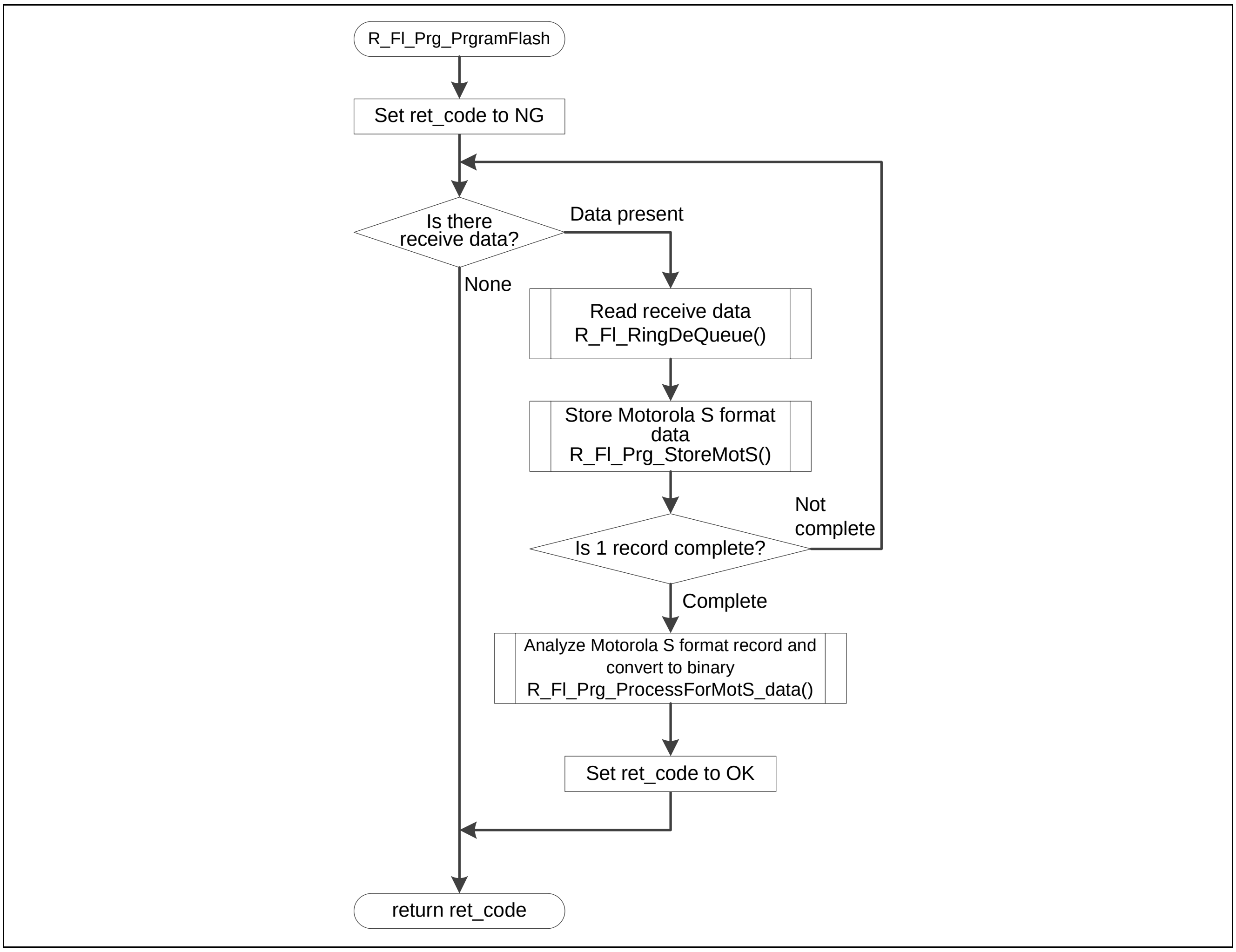


Figure 5.15 Download Area Write Operation

5.12.8 S Format Data Store Operation

Figure 5.16 shows the flowchart for S format data store operation.

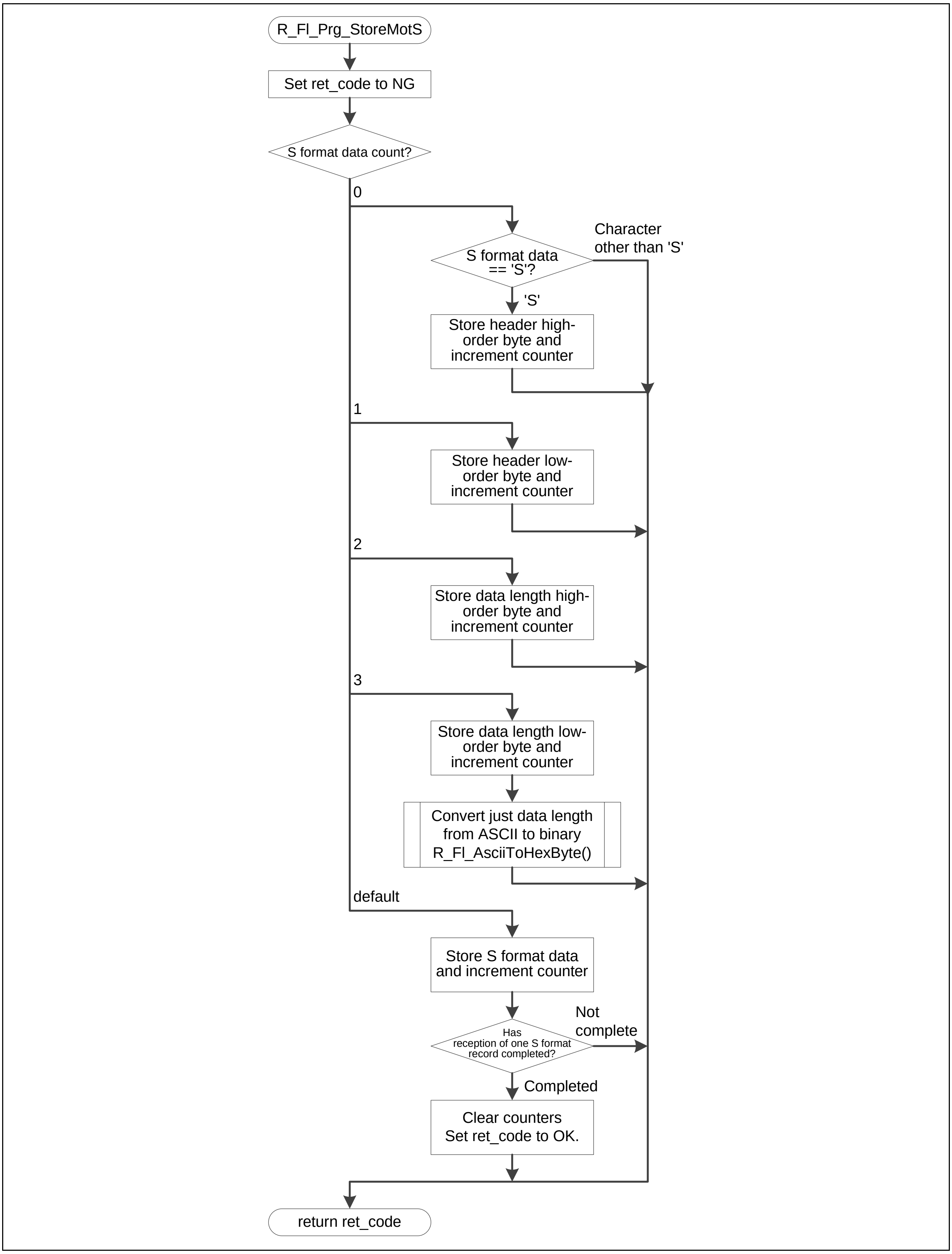


Figure 5.16 S Format Data Store Operation

5.12.9 S Format Header Analysis, Conversion to Binary, and Write Operations

Figure 5.17 shows the flowchart for S format header analysis, conversion to binary, and write operations.

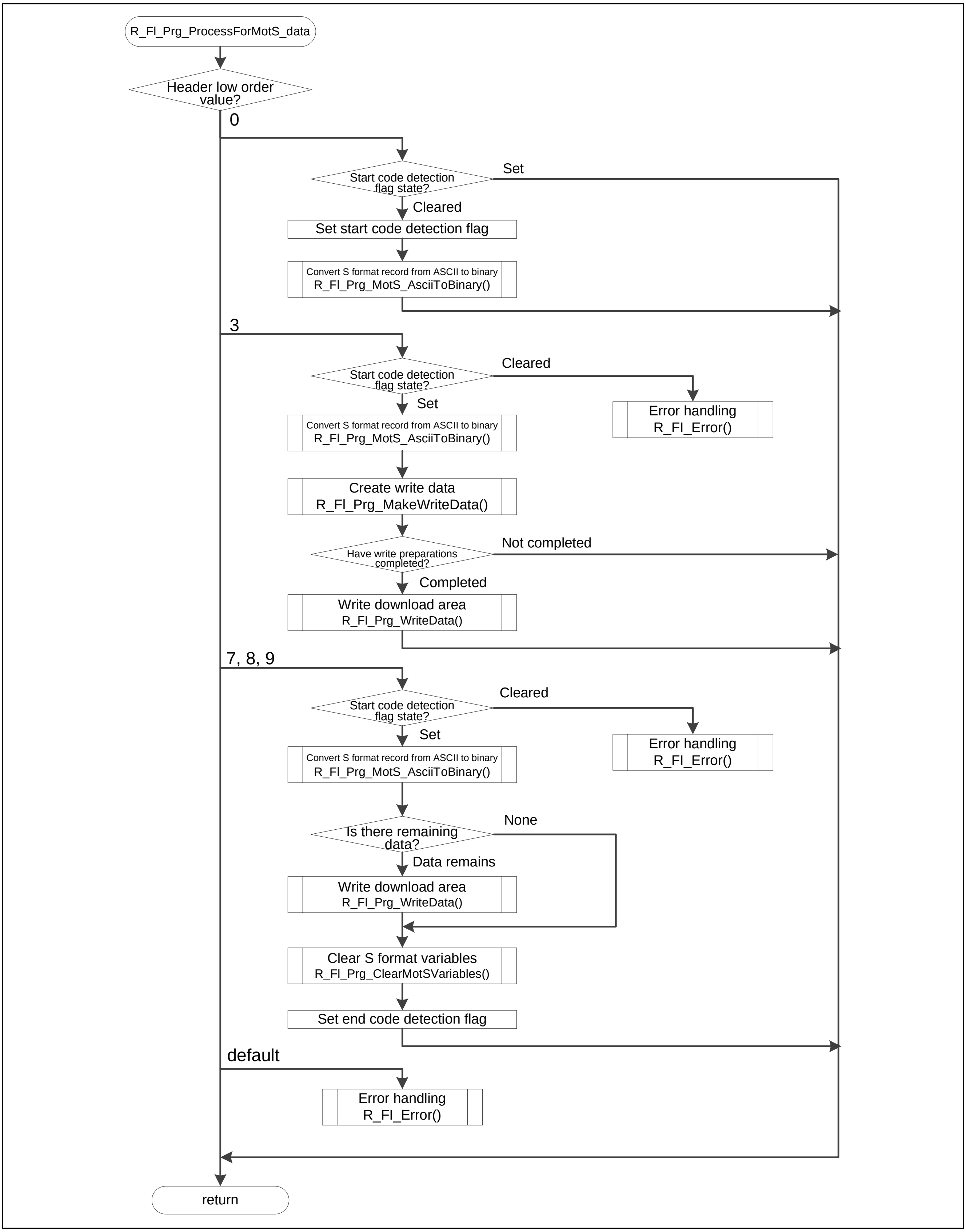


Figure 5.17 Format Header Analysis, Conversion to Binary, and Write Operations

5.12.10 S Format Data ASCII to Binary Conversion

Figure 5.18 shows the flowchart for S format data ASCII to binary conversion.

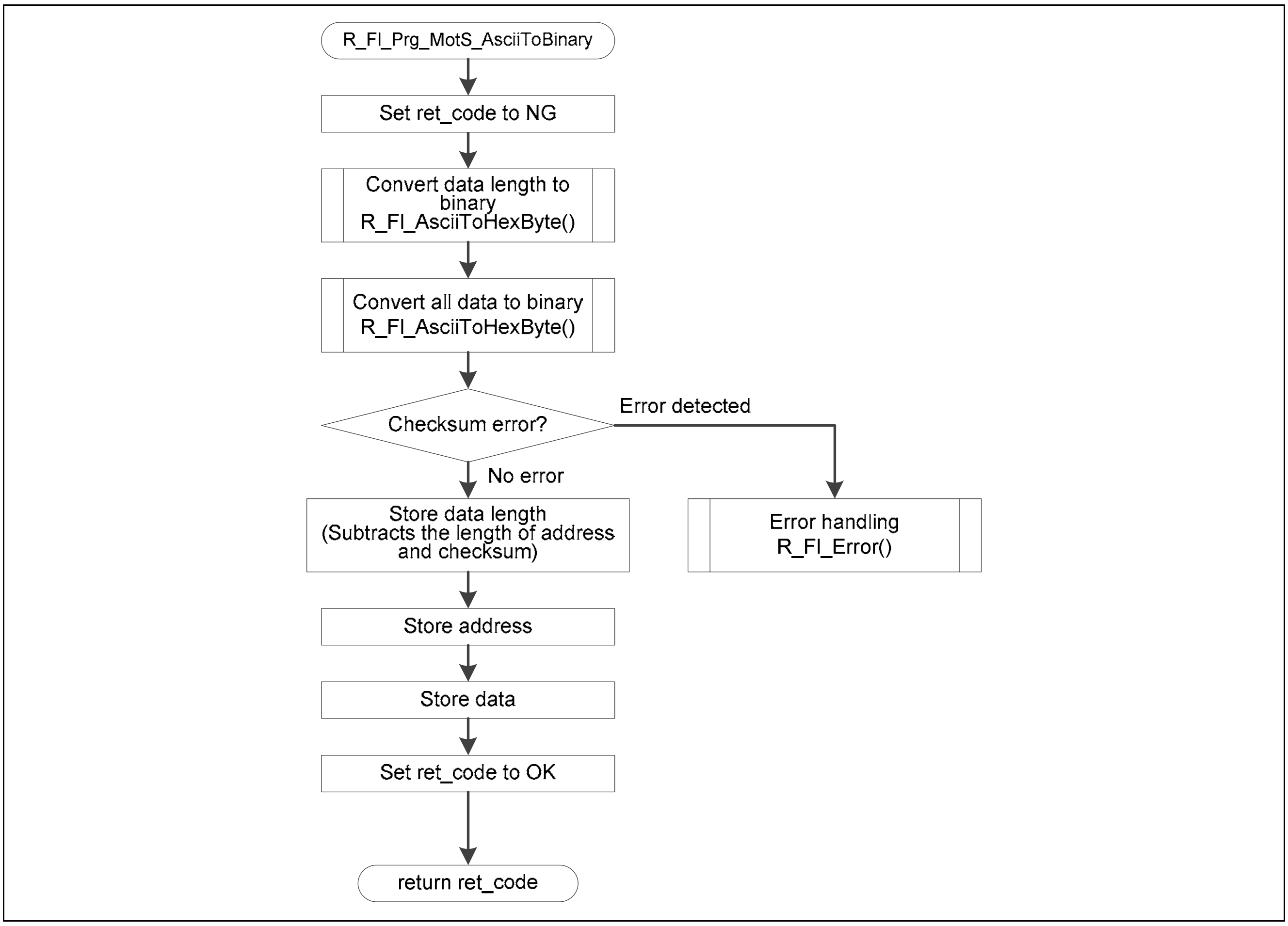


Figure 5.18 S Format Data ASCII to Binary Conversion

5.12.11 Download Area Write Data Creation

Figure 5.19 shows the flowchart for creating the data to be written to the download area.

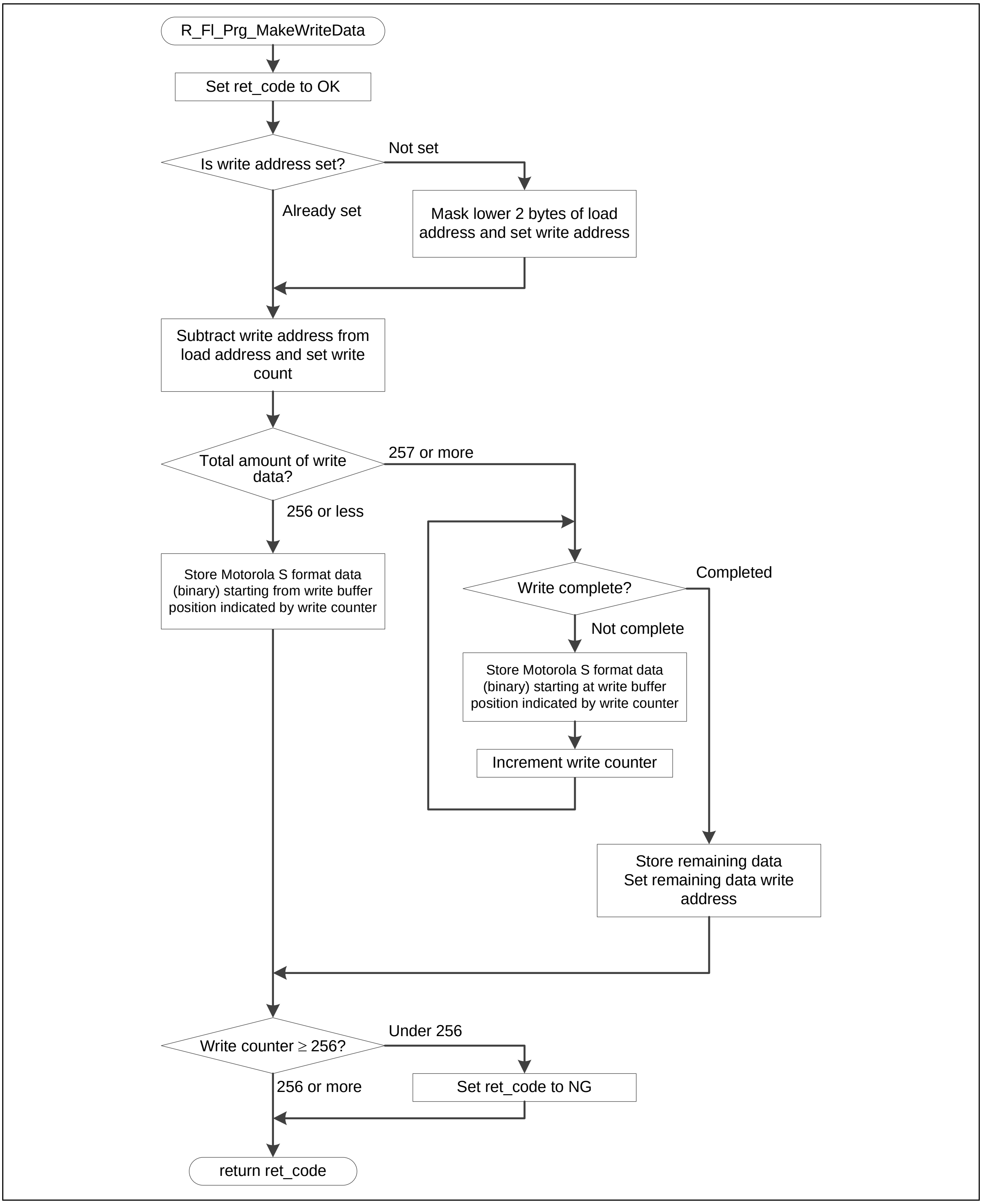


Figure 5.19 Download Area Write Data Creation

5.12.12 Download Area Write

Figure 5.20 shows the flowchart for download area write.

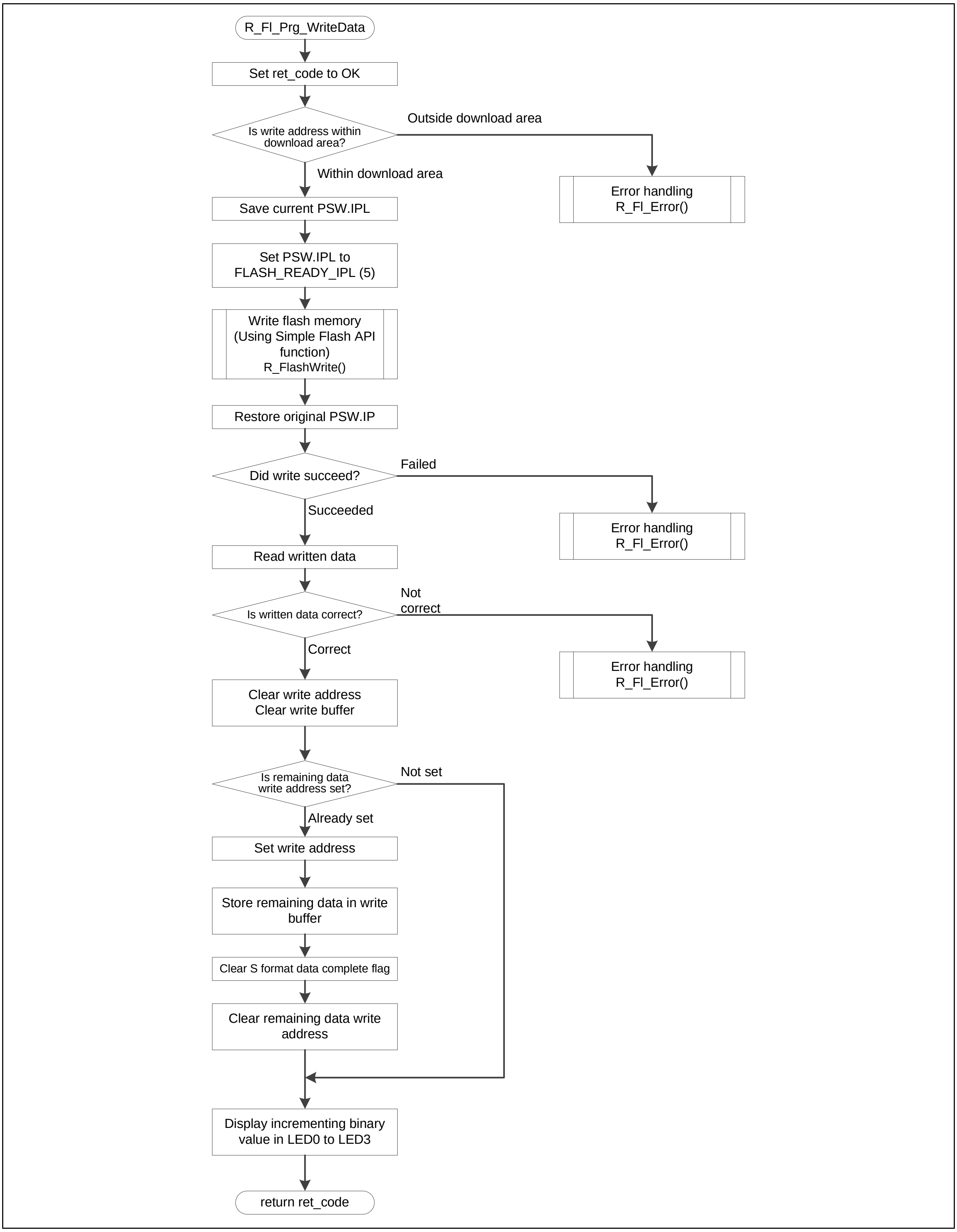


Figure 5.20 Download Area Write

5.12.13 Clear S Format Data Related Variables

Figure 5.21 shows the flowchart for clear S format data related variables.

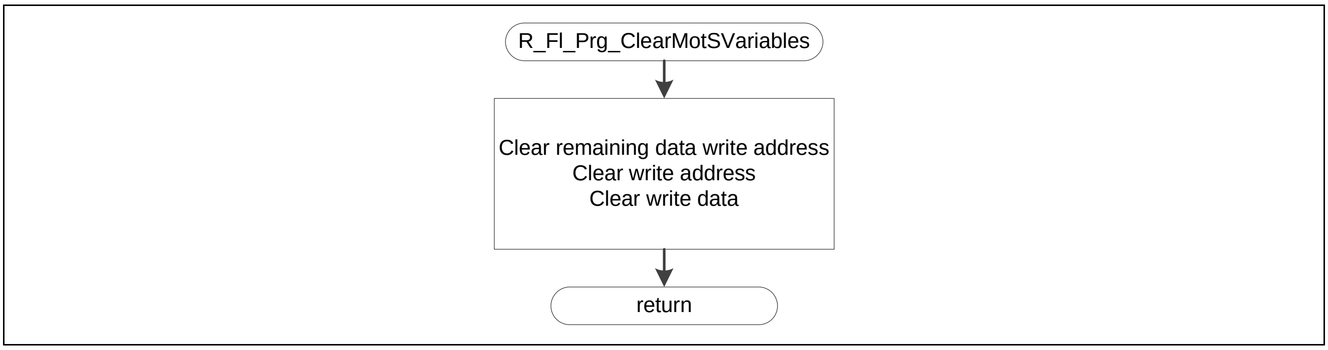


Figure 5.21 Clear S Format Data Related Variables

5.12.14 Store USB Receive Data

Figure 5.22 shows the flowchart for store USB receive data.

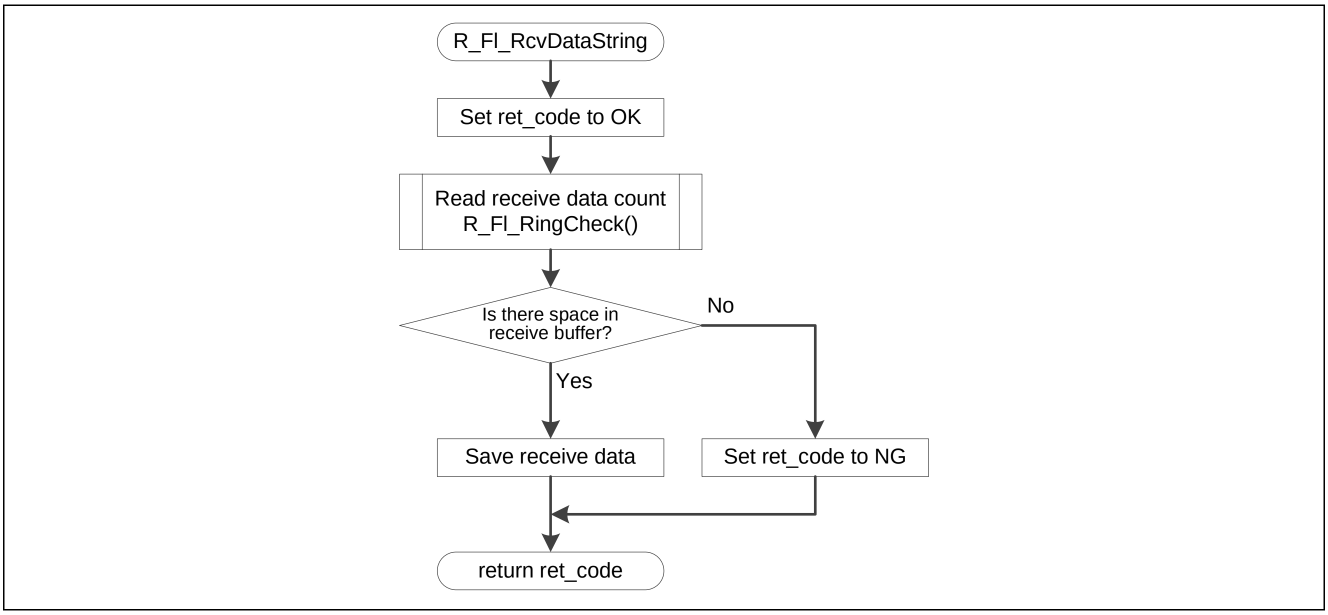


Figure 5.22 Store USB Receive Data

5.12.15 Check for Empty Space in Receive Data Storage Ring Buffer

Figure 5.23 shows the flowchart for check for empty space in receive data storage ring buffer.

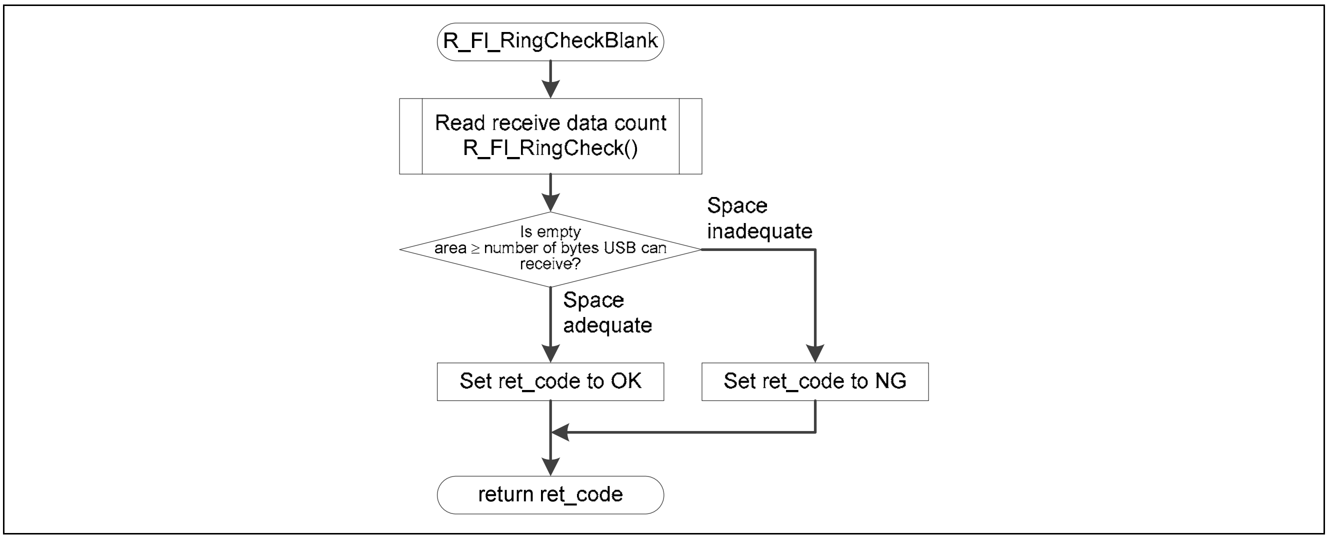


Figure 5.23 Check for Empty Space in Receive Data Storage Ring Buffer

5.12.16 Stop USB

Figure 5.24 shows the flowchart for stop USB.

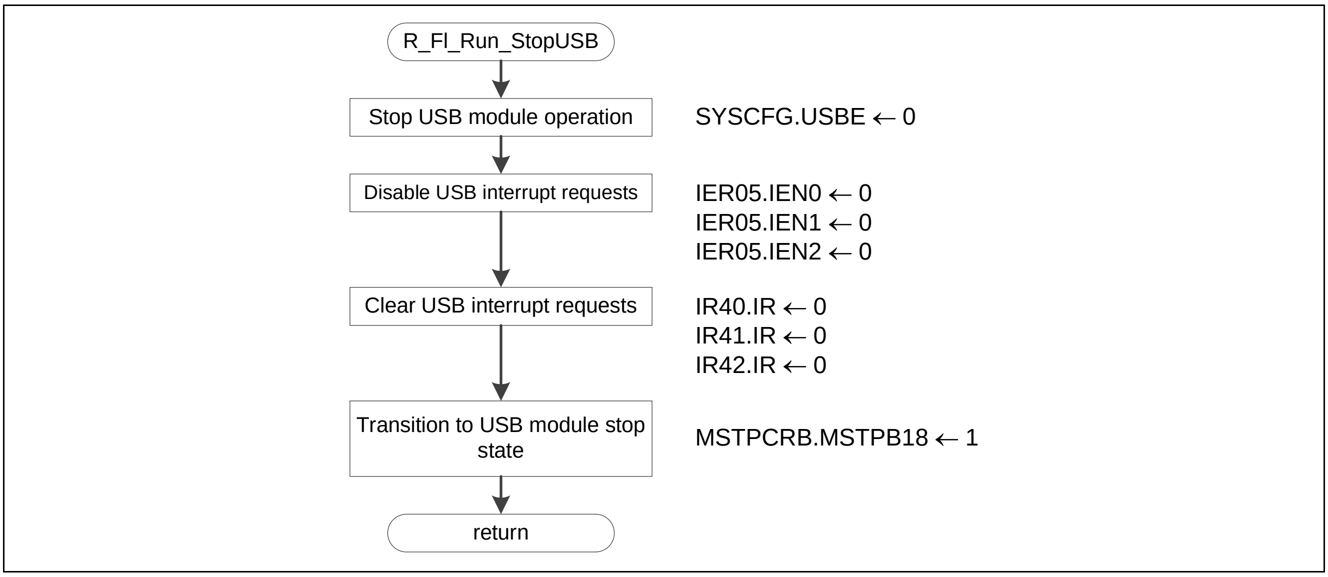


Figure 5.24 Stop USB

5.12.17 Error Handling

Figure 5.25 shows the flowchart for error handling.

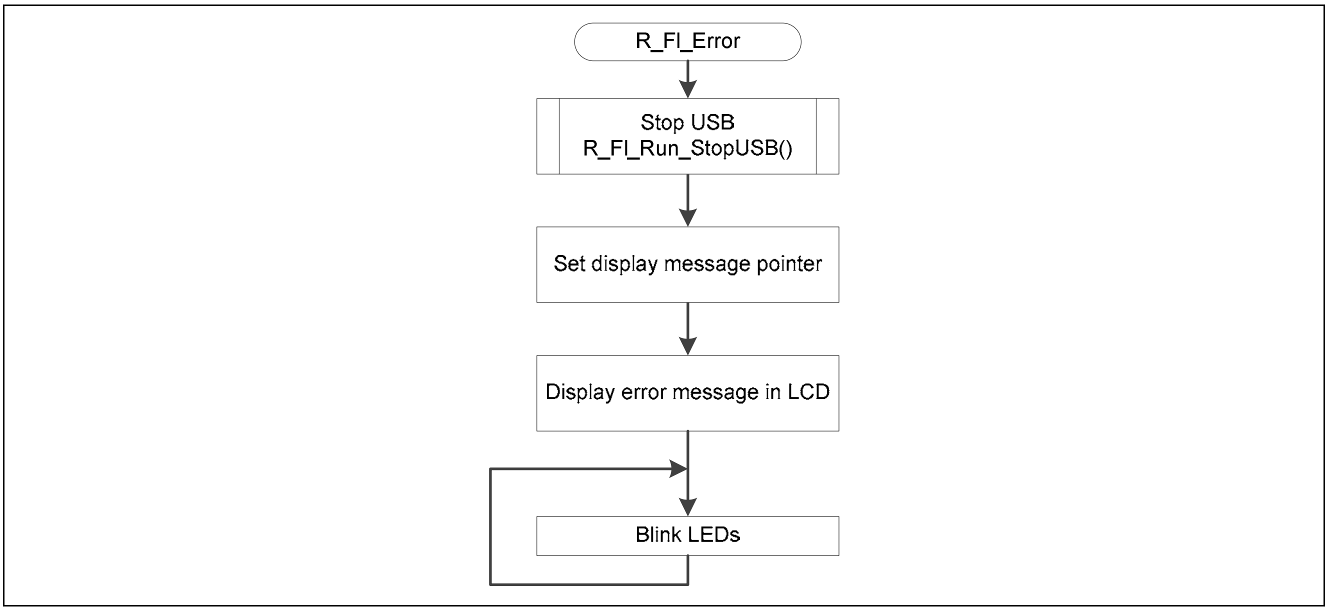


Figure 5.25 Error Handling

5.12.18 Store Data in Receive Data Ring Buffer

Figure 5.26 shows the flowchart for store data in receive data ring buffer.

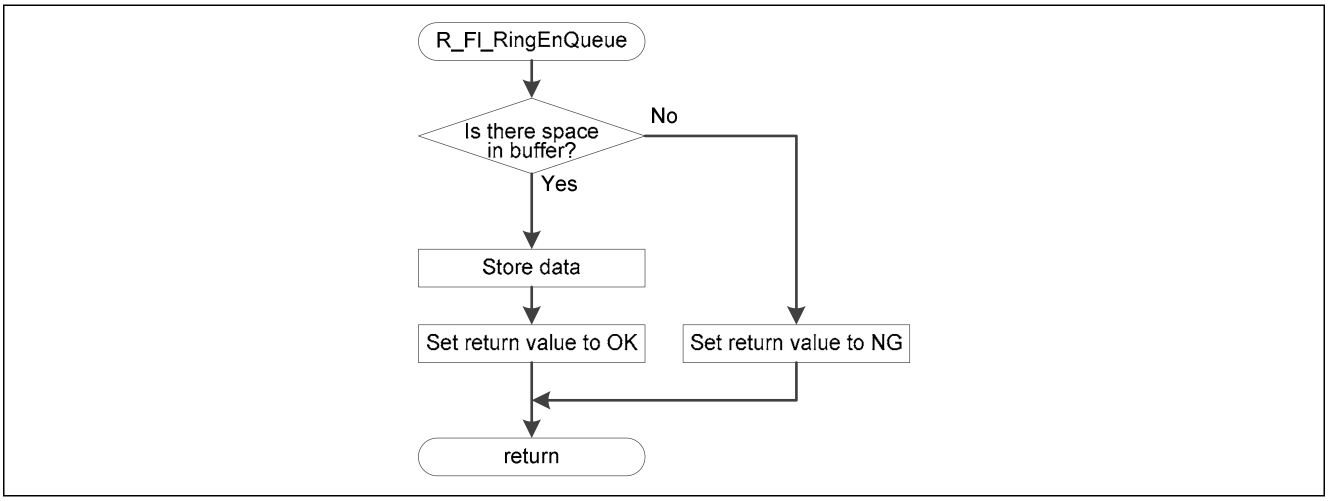


Figure 5.26 Store Data in Receive Data Ring Buffer

5.12.19 Read Data from Receive Data Ring Buffer

Figure 5.27 shows the flowchart for read data from receive data ring buffer.

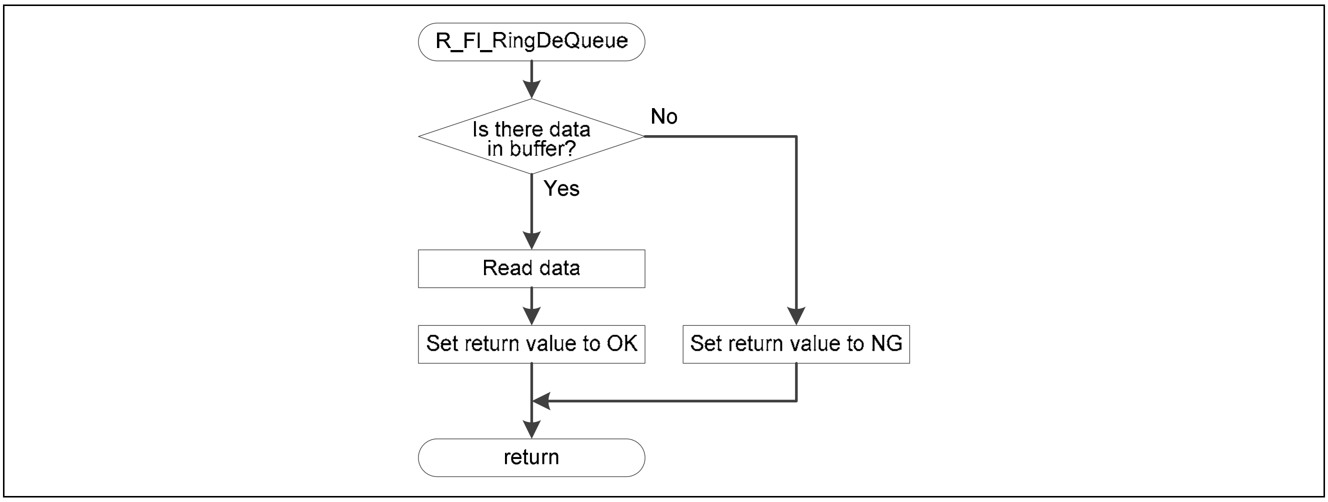


Figure 5.27 Read Data from Receive Data Ring Buffer

5.12.20 Check Data Count in Receive Data Ring Buffer

Figure 5.28 shows the flowchart for check data count in receive data ring buffer.

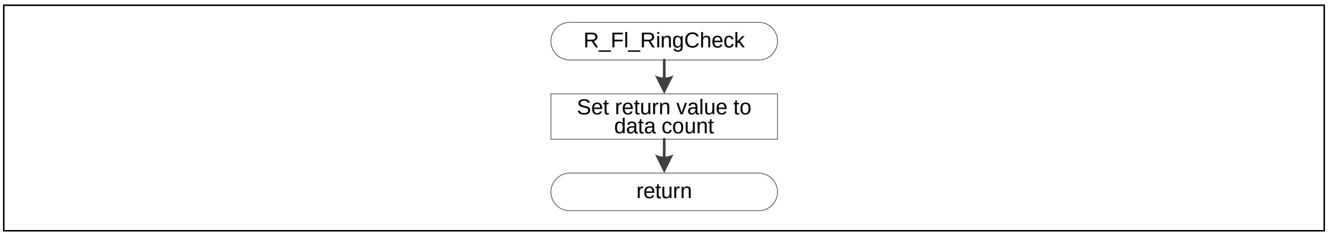


Figure 5.28 Check Data Count in Receive Data Ring Buffer

5.12.21 Convert Data from ASCII to Binary

Figure 5.29 shows the flowchart for convert data from ASCII to binary.

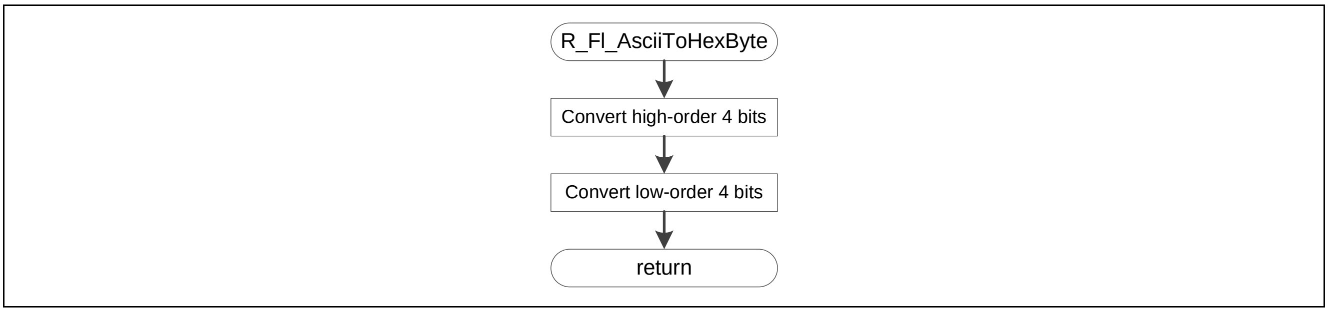


Figure 5.29 Convert Data from ASCII to Binary

6. Sample Download Code

This application note includes a sample download code file (download.zip). This program lights in sequence the LEDs on the board described in section 2, Confirmed Operating Conditions. Refer to this program for examples of download reset vector and section settings. Note that the download code is expected to use 512 KB of ROM.

7. S Format

This section describes the S format supported by the sample code.

7.1 Record Formats

Figure 7.1 shows the record formats supported by the sample code.

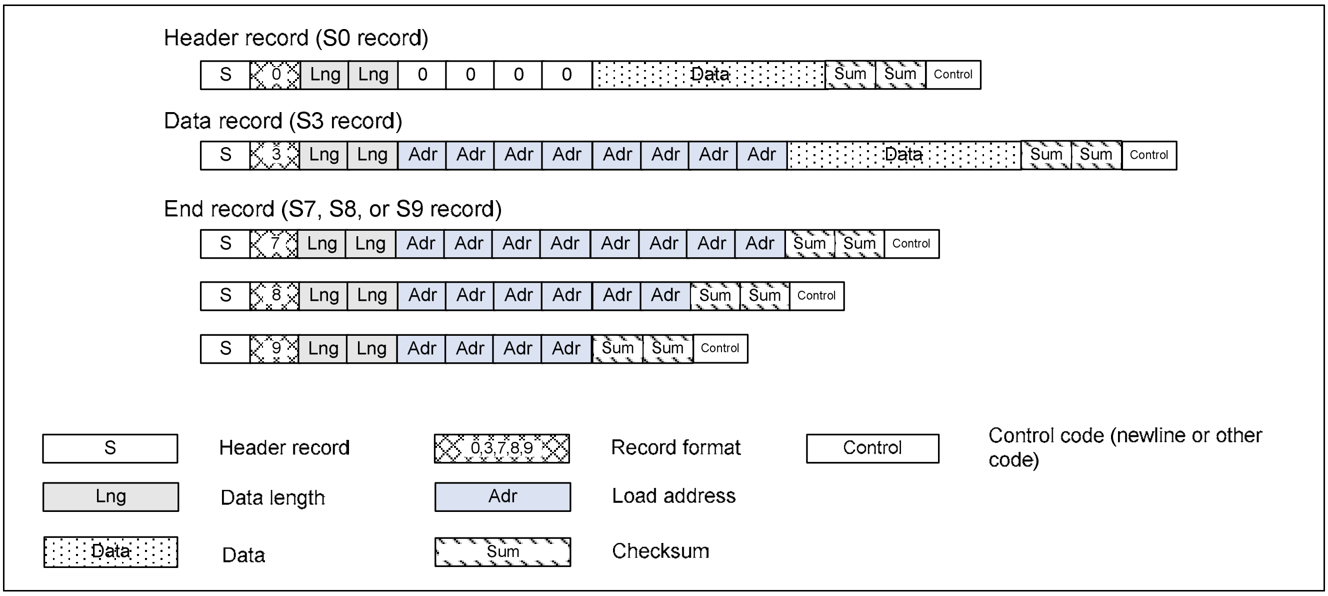


Figure 7.1 Record Formats Supported by Sample Code

7.2 Record Structure

Figure 7.2 shows the record structure supported by the sample code. S type format record sequences with orders other than those shown in figure 7.2 are not supported.

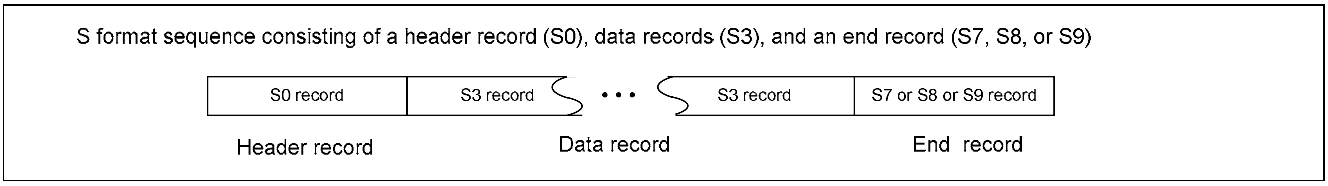


Figure 7.2 Record Structure Supported by the Sample Code

7.3 Load Address

The sample code only supports S format files with increasing load addresses. Do not use decreasing order or out of order load address S format files with the sample code.

7.4 Error Detection

The sample code detects errors if there are problems with the S format file received.

(a) Checksum error

The sample code verifies the checksum at each received S format record. A checksum error is detected if that verification finds an abnormality.

(b) Format error

A format error is detected if the sample code receives an S format file that meets any of the following conditions.

If an unsupported record (S1, S2, S4, S5, or S6) is detected

If a header record (S0) is detected twice

If a data record (S3) or an end record (S7, S8, or S9) is detected before a header record.

Figure 7.3 shows the format error detection conditions.

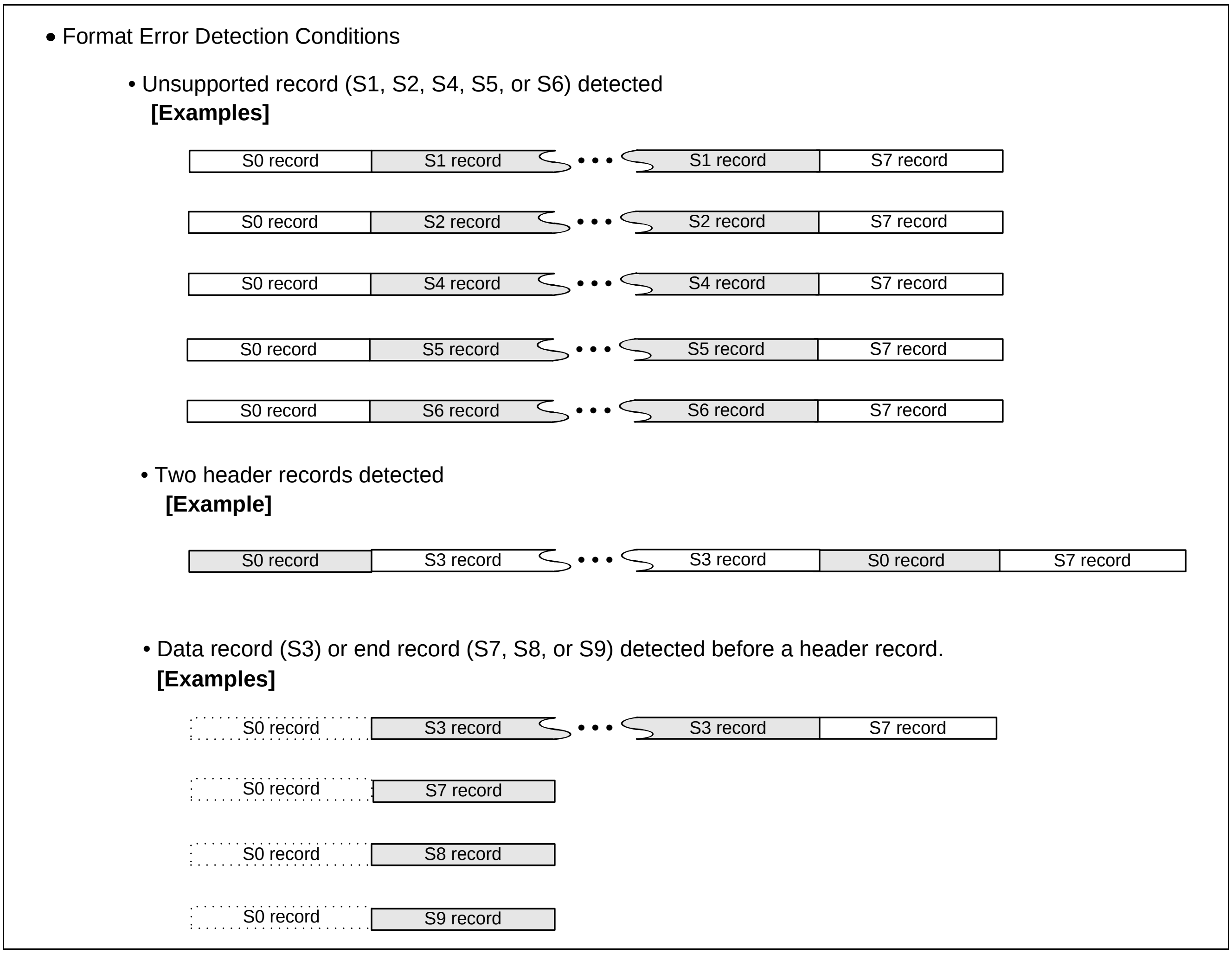


Figure 7.3 Format Error Detection Conditions

(c) Address error

An address error is detected if write data for any address outside the download area is received.

8. Notes

8.1 USB Disconnection During Write or Erase

Do not disconnect the USB while erasing or writing the download area.

8.2 HEW Settings

The sample code runs by copying the code in ROM to RAM during flash memory write operations. See the RX Family RX600 Simple Flash API application note for details on the settings.

8.3 Fixed Vector Table Interrupts

Of the fixed vector table interrupts, this sample code only handles the reset interrupt. If any other fixed vector table interrupts are used, the sample code must be modified to handle those interrupts.

8.4 Reset Vector for the Download Code

The execution start position for the download code written using the sample code is determined by the value written at the download reset vector (FFFE 3FFCh). Therefore the download code must be set up so that its reset vector is allocated at FFFE 3FFCh. See section 5.2, Download Code Execution Start Position, for details.

Also, see section 6, Sample Download Code, for details on the download code.

8.5 Changing the ROM Capacity

The ROM capacity of the microcontroller used by the sample code is 512 KB.

If a microcontroller with a ROM capacity of 384 KB, or 256 KB, is used, change FL_END_BLOCK_NUM #define directive in the file r_Flash_main.h to match the capacity used.

Table 8.1 lists the ROM capacities.

Table 8.1 ROM Capacities

Catalog Number	ROM Capacity	Download Area ROM Capacity	Download Area Start Address	Download Area Block Numbers
R5F562x8	512 K	400 K	FFF8 0000h	EB13 to EB37
R5F562x7	384 K	272 K	FFFA 0000h	EB13 to EB29
R5F562x6	256 K	144 K	FFFC 0000h	EB13 to EB21

8.6 while(1) Processing

Note that the sample code locks up by executing a while(1) loop if the USB ring buffer overflows.

8.7 Endian Order

The sample code in this application note supports both little endian and big endian orders. Note that both the flash boot loader and the download code must be set to the same endian order.

8.7.1 Using Little Endian

When operating using the little endian order, perform the following settings.

1. Mode pin (MDE) setting
   Pin 3 in SW4 on the RX62N RSK board must be set to ON (MDE = low).
2. Compiler option settings
   Specify "Little-endian data" as the compiler option endian setting.
3. Changes to the user system definitions file (r_usbc_cDefUsr.h)
   Set the value of the USBC_CPUBYTE_PP macro definition to USBC_BYTE_LITTLE_PP in the r_usbc_cDefUsr.h file.
4. Changing the TFAT library file (tfat_rx_little_v100.lib or tfat_rx_big_v100.lib)
   Specify WorkSpace\MSC2FW\TFAT\lib\tfat_rx_little_v100.lib as the linker input library file option.

8.7.2 Using Big Endian

When operating using the big endian order, perform the following settings.

1. Mode pin (MDE) setting
   Pin 3 in SW4 on the RX62N RSK board must be set to OFF (MDE = high).
2. Compiler option settings
   Specify "Big-endian data" as the compiler option endian setting.
3. Changes to the user system definitions file (r_usbc_cDefUsr.h)
   Set the value of the USBC_CPUBYTE_PP macro definition to USBC_BYTE_BIG_PP in the r_usbc_cDefUsr.h file.
4. Changing the TFAT library file (tfat_rx_little_v100.lib or tfat_rx_big_v100.lib)
   Specify WorkSpace\MSC2FW\TFAT\lib\tfat_rx_big_v100.lib as the linker input library file option.

8.8 Changes to the RX600 Simple Flash API

This application note uses sample code from the RX600 Simple Flash API. See the RX600 Simple Flash API application note for the specifications of the RX600 Simple Flash API.

8.8.1 Changes

The files in the RX600 Simple Flash API that are changed are r_flash_api_rx600_config.h and mcu_info.h.

- Changes to the file r_flash_api_rx600_config.h

  (1) To prevent ROM access by interrupts during flash write and erase operations, the processor status word (PSW) interrupt priority level (IPL) field is changed to the value specified in the following macro definition. In this application note, the value 5 is used.
  Macro definition: #define FLASH_READY_IPL 5

  (2) The following Simple Flash API settings are changed.

  Before change: #define IGNORE_LOCK_BITS
  #define COPY_CODE_BY_API
  #define FLASH_API_USE_R_BSP
  After change: //#define IGNORE_LOCK_BITS
  //#define COPY_CODE_BY_API
  //#define FLASH_API_USE_R_BSP

- Changes to the file mcu_info.h

  (1) The files stored in the Simple Flash API r_bsp/board/rskrx62n folder are used.
  (2) The following Simple Flash API settings are changed.

  Before change: #define ICLK_HZ (96000000)
  #define PCLK_HZ (48000000)
  #define BCLK_HZ (12000000)
  After change: #define ICLK_HZ (48000000)
  #define PCLK_HZ (24000000)
  #define BCLK_HZ (48000000)

8.9 Changes to the USB Host Mass Storage Class Driver

The sample code uses the USB Host Mass Storage Class Driver program code. For specifications of the USB Device USB Host Mass Storage Class Driver, see the USB Device USB Host Mass Storage Class Driver and USB Device USB Basic Firmware application notes.

8.9.1 Changes

The following four files in the USB Host Mass Storage Class Driver are modified for used with this sample code.

- r_usb2_HMSC_apl.c
- dbsct_hmsc.c
- RX62NRSK.c
- resetprg.c

- Places changed in the file r_usb2_HMSC_apl.c:
  Places indicated by #ifdef R_FLASH_USB.

- Places changed in the file dbsct_hmsc.c:
  Places indicated by the comment "// Flash table".

- Places changed in the file RX62NRSK.c:
  The IRQ8 and IRQ9 setting functions in the usb2_cstd_TargetInit() function has been deleted.
  Deleted: usbc_cpu_IRQ9_Enable();
  Deleted: usbc_cpu_IRQ8_Enable();

- Places changed in the file resetprg.c:
  1. An include file is added.
     Added: #include "r_Flash_main.h"
  2. The INTB setting in the function PowerON_Reset_PC() is changed.
     Before change: set_intb(0xFFF80000);
     After change: set_intb(__sectop("INTERRUPT_VECTOR"));
  3. The mode entry function is called in the function PowerON_Reset_PC().
     Added: R_Fl_Mode_Entry();

8.9.2 Added Files

See section 5.7, File Structure, for the files added to the USB Host Mass Storage Class Driver.

8.9.3 Added Sections

Table 8.2 lists the sections added to the USB Host Mass Storage Class Driver.

Table 8.2 Added Sections

Section	Description
R_flash_api_sec	Section for variables in the flash programming code that operates in RAM
RPFRAM	Section for the flash programming code that operates in RAM
TRGT_DMMY_FIXEDVECT	Section for the download code fixed vector

8.9.4 Include File Directories

WorkSpace\FLASH is added to the include directories.

8.9.5 Macro Definitions (Compiler options)

- R_FLASH_USB is added as a compiler option macro definition.
- USBC_SDRAM_USE_PP is removed from a compiler option macro definition.

8.9.6 Linker Settings

The following linker settings that map from ROM to RAM are added.

- ROM PFRAM is mapped to RPFRAM.
- ROM D_flash_api_sec is mapped to R_flash_api_sec.

9. Sample Programs

The sample program can be downloaded from the Renesas Electronics Web site.

10. Reference Documents

- RX62N Group, RX621 Group User’s Manual: Hardware, Rev.1.20
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- Technical Updates/Technical News
  (The latest information can be downloaded from the Renesas Electronics Web site.)
- C Compiler Manual
  RX Family C/C++ Compiler Package V.1.10 Release 00
  RX Family C/C++ Compiler Package User’s Manual Rev.1.10 R00
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- Application Notes
  RX600 Series Simple Flash API for RX600 Rev.2.20 (R01AN0544EU)
  (The latest version can be downloaded from the Renesas Electronics Web site.)
  Renesas USB Device USB Basic Firmware Rev.1.10 (R01AN0512EJ)
  (The latest version can be downloaded from the Renesas Electronics Web site.)
  Renesas USB Device USB Host Mass Storage Class Driver Rev.1.10 (R01AN0513EJ)
  (The latest version can be downloaded from the Renesas Electronics Web site.)
  M3S-TFAT-Tiny: Fat File System Software Rev.1.00 (R20AN0038EJ)
  (The latest version can be downloaded from the Renesas Electronics Web site.)

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Mar.28.12	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

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

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

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

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

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

4. Clock Signals

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

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

5. Differences between Products

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

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

"Standard":	Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
"High Quality":	Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
"Specific":	Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

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

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

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

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2012 Renesas Electronics Corporation. All rights reserved.

Colophon 1.1