S1C31 Family Application Note

S1C31 Series Boot Loader Manual

SEIKO EPSON CORPORATION

Rev.1.0

Evaluation board/kit and Development tool important notice

1. This evaluation board/kit or development tool is designed for use for engineering evaluation, demonstration, or development purposes only. Do not use it for other purposes. It is not intended to meet the requirements of design for finished products.
2. This evaluation board/kit or development tool is intended for use by an electronics engineer and is not a consumer product. The user should use it properly and in a safe manner. Seiko Epson dose not assume any responsibility or liability of any kind of damage and/or fire coursed by the use of it. The user should cease to use it when any abnormal issue occurs even during proper and safe use.
3. The part used for this evaluation board/kit or development tool may be changed without any notice.

NOTICE

Table of Contents

1. Overview 2
  1.1. Operating Environment 2
  1.2. Precautions for use 3
2. Software Configuration 4
  2.1. Folder Configuration 4
  2.2 User program format 5
3. Boot loader Function 6
  3.1 Boot Loader 6
    3.1.1 Peripheral circuit to use 7
    3.1.2 Upgrade indicator 7
    3.1.3 Boot loader behavior 8
  3.2 User Program 10
    3.2.1 How to rewrite the upgrade indicator 10
  3.3 Terminal software settings 11
4. Memory Usage 12
  4.1 Reserved RAM for S1C31 self-Modifying Library 12
5. About debugging 13
Appendix. Function specifications 14
Revision History 21

1. Overview

The S1C31 boot loader is sample software that uses the terminal software on the PC to rewrite the built-in flash memory of the target model with a new user program.

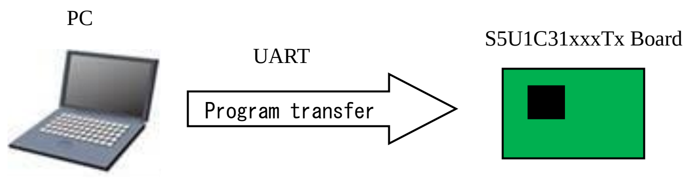

Figure 1.1 Diagram

This document describes the following two types of sample software.

- bootloader
  The user program is received by UART communication and the built-in flash memory is rewritten.
  The S1C31 self-modifying library (seFlashLibrary) is used to rewrite the built-in flash memory.

- loadsample
  This is a program that writes to the built-in flash memory of the target model via the boot loader.

This software is included in the S1C31xxx peripheral circuit sample software package. The S1C31xxx peripheral circuit sample software package is available on our website.
In addition to this manual, please also refer to the "S1C31xxx Technical Manual".

1.1. Operating Environment

Before running sample software, prepare the following components:

- Evaluation Board
  - S5U1C31xxxT1 evaluation board equipped with S1C31xxx
- Debug Probes *1, *2
  - IAR Systems I-jet or SEGGER J-Link
- Integrated Development Environment
  - IAR Embedded Workbench for ARM® (IAR EWARM) or MDK-ARM® (uVision)
- S1C31SetupTool package
  - Flash loader and configuration files (.svd, etc.)
- S1C31xxx Peripheral circuit sample software package
- PC terminal software
- Other Devices (option)
  - An USB Adapter for UART

*1: The debug probe is not required when erasing or writing the flash memory by calling a function from the sample software.

*2: I-jet is available only with IAR EWARM. J-Link is available for both IAR EWARM and MDK-ARM.

For details on the above, refer to the manuals that come with each.

1.2. Precautions for use

This sample software is for reference only. We are not responsible for any problems caused by these. When using it on a product, perform sufficient operation verification.

This manual is common to all boot loaders provided for each model of the S1C31 series. For the specifications (terminals used, etc.) that differ depending on the model, refer to the readme included in the S1C31xxx peripheral circuit sample software package. For the S131 self-modifying library, refer to the "S1C31 Family Self-Modifying Library Manual".

2. Software Configuration

2.1. Folder Configuration

The configuration of the S1C31 boot loader and related programs included in the S1C31xxx peripheral circuit sample software package is as follows.

```
S1C31xxxSamplePKG_very_yy.zip
[S1C31xxxSamplePKG_very_yy]
    |- [Licenses]
    |- [Drivers] : Driver group
    |    |- [board] : Drivers related to the evaluation board
    |    |- [CMSIS] : CMSIS driver
    |    |    |- [Device]
    |    |    |    |- [S1C31xxx]
    |    |    |        |- [Include]
    |    |    |        |    |-S1C31xxx.h : CMSIS peripheral circuit access layer header file
    |    |    |        |    |- …
    |    |    |        |- [Source]
    |    |    |            |- [ARM]
    |    |    |            |- [IAR]
    |    |    |            |    |- startup_S1C31xxx.s : CMSIS startup program
    |    |    |            |- system_S1C31xxx.c : CMSIS peripheral circuit access layer program
    |    |    |- [Driver]
    |    |    |    |- [Include]
    |    |    |    |    |- Driver_Common.h : Common driver definition
    |    |    |    |    |- Driver_Flash.h : CMSIS Self-modifying library driver definition
    |    |    |    |    |- …
    |    |- [sePeripheralLibrary] : Peripheral circuit library
    |- [Middlewares] : Middleware group
    |    |- [seFlashLibrary] : Self-modifying library
    |    |    |- [Device]
    |    |    |    |- [S1C31xxx]
    |    |    |         |- seFlashLibraryS1C31xxx.a : IAR EAWRM library
    |    |    |         |- seFlashLibraryS1C31xxx.lib : MDK-ARM library
    |    |- …
    |- [Projects] : Sample software group
    |    |- [Applications] : Application software group
    |    |    |- [BOOTLOADER] : Sample software for boot loader
    |    |    |    |- [bootloader] : boot loader
    |    |    |    |    |- [ARM] : Project for MDK-ARM
    |    |    |    |    |- [IAR] : Project for IAR EWARM
    |    |    |    |    |- main.c
    |    |    |    |    |- …
    |    |    |    |- [loadsample] : User program
    |    |    |    |    |- [ARM] : Project for MDK-ARM
    |    |    |    |    |- [IAR] : Project for IAR EWARM
    |    |    |    |    |- main.c
    |    |    |    |- bootLoaderForS1c31xxx_readme_e.txt : read me
    |    |    |    |- bootLoaderForS1c31xxx_readme_j.txt
    |
    README_e.txt
    README_j.txt
```

Figure 2.1.1 S1C31xxx Sample software package configuration

2.2 User program format

The data format that can be written as a new user program is Motorola S-Record. Also, the end of the data must be one of the S7 / S8 / S9 records. The boot loader now determines that the internal flash memory has been rewritten.

3. Boot loader Function

The boot loader realizes the function by using the terminal software that supports the serial port and the S1C31 self-modifying library (seFlashLibrary) that rewrites the built-in flash memory (Flash ROM).

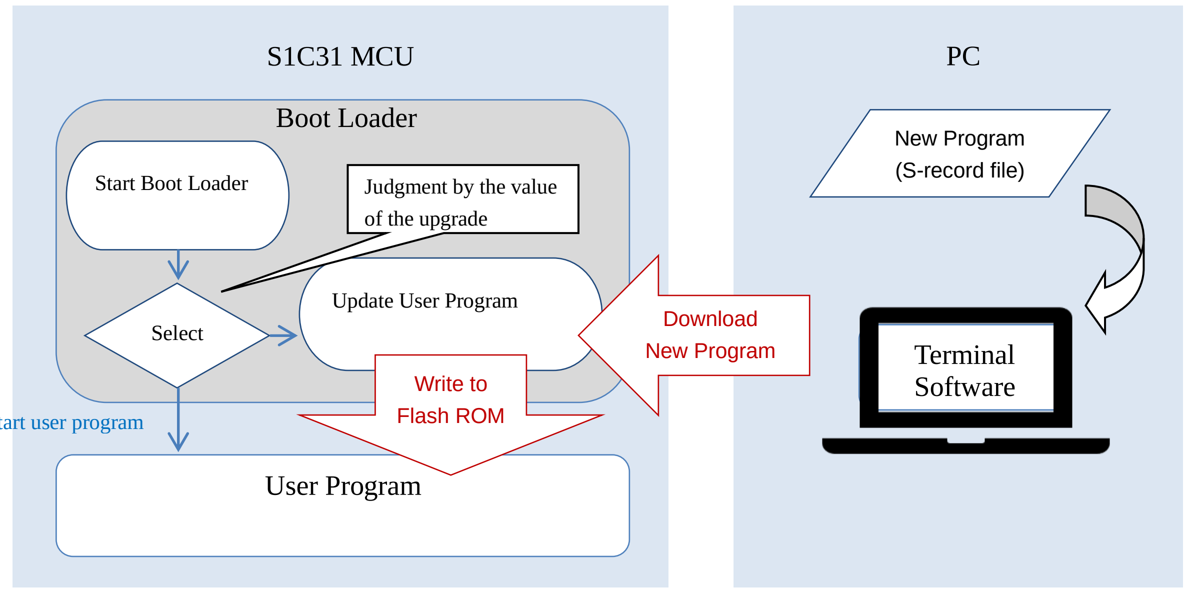

Figure 3.1 Boot loader function overview

3.1 Boot Loader

The project configuration of the bootloader, the library to be used, and the operating area are as follows.

- bootloader project configuration:

- board	
- CMSIS	
- sePeripheralLibrary	
- main.c	Boot loader
- indicator.c	Upgrade indicator definition
- srec.c	Motorola S record decoding process
- srec.h	Definition for decoding Motorola S records

- Library to use:

  S1C31 self-modifying library seFlashLibrary

  ※For the S1C31 self-modifying library, refer to "S1C31 Family Self-Modifying Library Manual".

- Operating area :

  Flash ROM 0x0000-0x3FFF(16KB）

The bootloader will start working after the MCU is reset and will perform one of the following functions depending on the value of the upgrade indicator:

➢ Start user program
➢ Update user program

3.1.1 Peripheral circuit to use

The boot loader uses the following peripheral circuits.

- UART CH0: Used for communication with PC

Table 3.1.1.1 Serial communication format

Communication speed	230400bps
Data length	8bit
Parity	None
Stop bit	1bit
Flow control	Software (XON/XOFF)

For the terminals used by UART channel 0, refer to the readme included in the S1C31xxx peripheral circuit sample software package.

- Watchdog timer: Uses the system for restarting when the end record is received
- 16-bit timer CH0: Used in S1C31 self-modifying library
- 16-bit timer CH1: Used as a timer for communication timeout

3.1.2 Upgrade indicator

The upgrade indicator is located at address 0x3000 and manages the update status of the user program. The size of the upgrade indicator is 4KB. However, only 4 bytes are actually used, and the 4-byte value indicates the following states.

Table 3.1.2.1 Upgrade indicator status

status	値
“upgrade completed” The update is complete and you can run the user program.	0xAA, 0xAA, 0xAA, 0xAA
“do upgrade” The user program has decided to update, but the update has not started.	0x00, 0x00, 0x00, 0x00
“now upgrading” The bootloader is updating the user program. The update is not complete.	0xFF, 0xFF, 0xFF, 0xFF

■ **About indicator.c**

The initial value of the upgrade indicator is defined in indicator.c.
The definition depends on the workbench used.
Normally, specify "0xaa, 0xaa, 0xaa, 0xaa". (Default)
If you want to debug the boot loader itself, set "0x00, 0x00, 0x00, 0x00".

For IAR EWARM

For µVision

```
#ifdef __IAR_SYSTEMS_ICC__
#pragma location = ".ConstSection1"
const unsigned char upgrade_indicator[]    = {
                        //0xFF, 0xFF, 0xFF, 0xFF,      /// now upgrading
                        0xaa, 0xaa, 0xaa, 0xaa,        /// upgrade_completed
                        //0x00, 0x00, 0x00, 0x00,      /// do_upgrade
};
#else
const unsigned char upgrade_indicator[] __attribute__((section(".ConstSection1"))) =    {
                        //0xFF, 0xFF, 0xFF, 0xFF,      /// now upgrading
                        0xaa, 0xaa, 0xaa, 0xaa,        /// upgrade_completed
                        //0x00, 0x00, 0x00, 0x00,      /// do_upgrade
};
#endif
```

3.1.3 Boot loader behavior

The boot loader works as follows.

■ **Start user program**
If the upgrade indicator is “upgrade completed” (0xAA, 0xAA, 0xAA, 0xAA), the bootloader jumps to the user program that exists after address 0x4000. At that time, change the vector table offset address (VTOR) to 0x4000.

■ **User program update**
If the upgrade indicator is “do upgrade” (0x00,0x00,0x00,0x00) or “now upgrading” (0xFF, 0xFF, 0xFF, 0xFF), the bootloader uses the UART to receive the new user program at address 0x4000. Write to the following area. Also, in the case of “do upgrade” (0x00,0x00,0x00,0x00), the existing user program will be deleted.
The bootloader continues to update the user program until it receives the end of the data. When it receives the end of the data, the bootloader stops and the WDT resets the MCU. After the reset, the new user program will start.

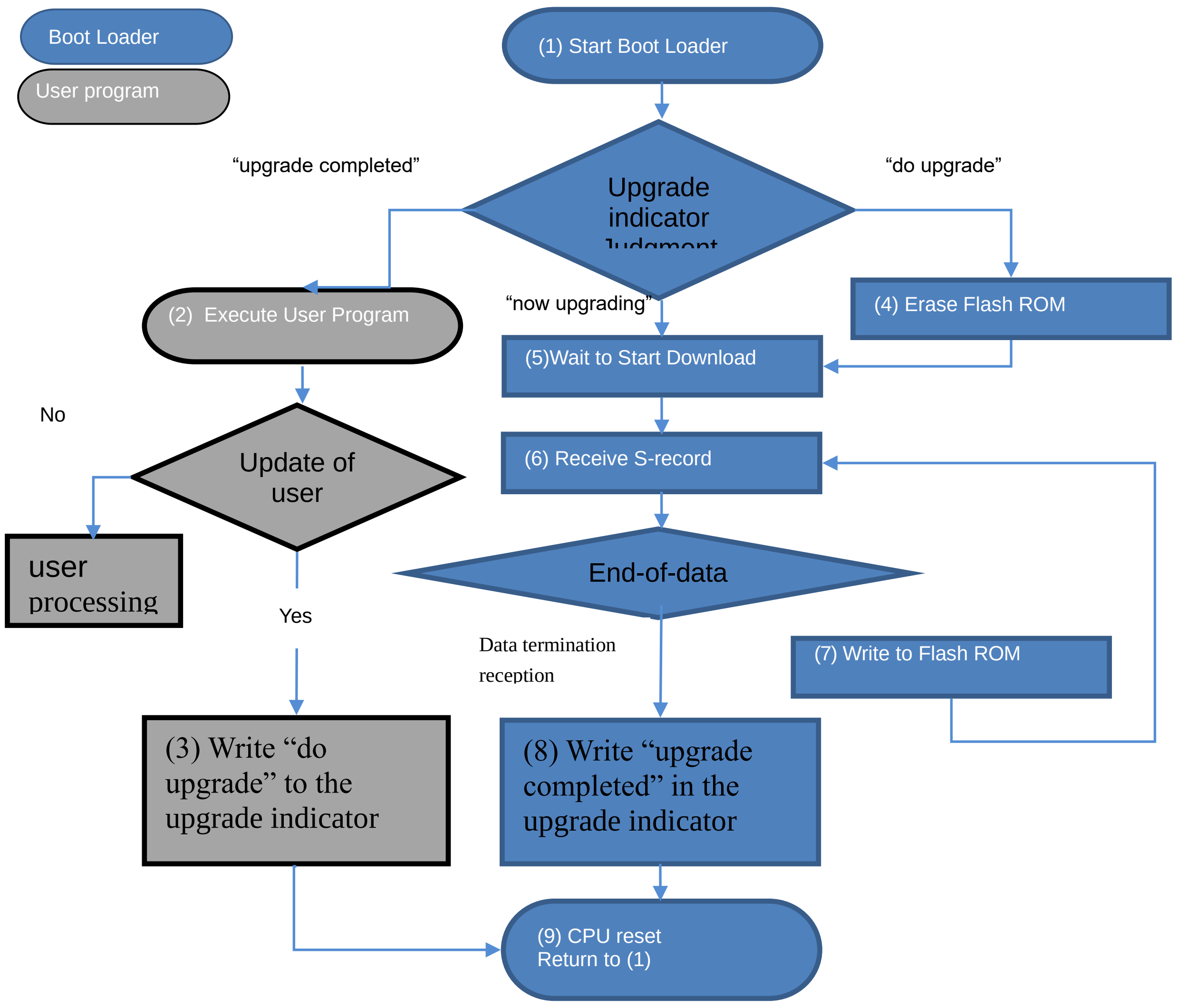

Figure 3.1.3.1 Boot loader operation flow

Please examine the following points when you implement the boot loader into your device:

A) Change the terminal for UART communication with the PC according to the hardware configuration.

B) Optimal baud rate setting
Change to a baud rate that does not cause a communication error according to the hardware configuration.
Consider changing the UART source vibration to a crystal oscillator instead of the built-in oscillator.

C) How should the boot loader operate when the problem has been detected ?
   i. Communication timeout.
   This boot loader (5) (6) monitors the UART reception or the time-out. But it retry (5) (6) again, even if the time-out occurs. The boot loader waits for time-out period (3ms) after transmitting XOFF. There is a possibility to shorten this time-out.
   ii. Communication error.
   If the error is caused in UART communication, it is necessary to demand to send the record again. This boot loader (5) (6) does not monitor the UART error, because the terminal software working on PC does not accept such request.
   iii. Wrong data.
   This boot loader confirms the checksum of the S-record (6), and stops without writing in case of the error (9). It is better to demand to send the record again.
   iv. Writing failure in flash ROM.
   There is possibility to rewrite data at the already written address. The same value can be written again. The value can be rewritten from 1 to 0. But the rewriting from 0 to 1 causes an error.
   The downloaded data causes this failure. When this error is caused, it is necessary to erase flash ROM, and download from the beginning again.

D) Does the boot loader work correctly even if it was reset ?
This boot loader does not erase flash ROM again form (1) to (5) when reset while updating. It will work correctly by erasing again, but it is necessary to rewrite everything.
In casse of the boot loader does not erase again, the following confirmation is necessary. These conditions cannot be judged only from the reception of the S-record. If these conditions are false, it is better to erase again.
   i. Is the data that has already been written correct ?
   ii. Is the data newly received the same program as the data that has already been received ?

E) When should the boot loader erase flash ROM ?
When the upgrade indicator is “do upgrade”, this boot loader (4) erase the user program and the upgrade indicator. The value of the upgrade indicator becomes 0xFF, and means “now upgrading” by the erase. The boot loader never erase until the upgrade is completed after this.
As a result, the boot loader never erases flash ROM at each reset. But the user program becomes impossible to upgrade when itself does not work because of wrong download and so on.
It is better to provide additional condition that the boot loader erases flash ROM.

F) How should the boot loader operate after the loading (8) ?
This boot loader activates the watchdog timer after flash programing (9) and resets the MCU. However, it is also possible to execute the new program at once.

G) Loader application on PC
It is possible to download new user program smoothly by preparing your loade application on PC. Such application should have the following functions:
   i. Inspect whether the program file is correct.
   ii. Pause an upgrade process.
   iii. Restart an upgrade process.
   iv. Resend a record.
   v. Confirm the upgrade is completed.

3.2 User Program

The project structure and operating area of the user program (loadsample) are as follows.

- loadsample project configuration:
  - board
  - CMSIS
  - sePeripheralLibrary
  - main.c user program

- Operating area:
  Flash ROM Place it on 0x4000 or later.

In order to update the user program, it is necessary to rewrite the upgrade indicator to “do upgrade” (0x00,0x00,0x00,0x00) in the user program.

3.2.1 How to rewrite the upgrade indicator

The user program can rewrite the upgrade indicator by calling the upgrade indicator update processing routine (boot loader vector table 47 (0x00BC)) prepared by the boot loader from the user program. Therefore, prepare the start_upgrade function in the user program and rewrite the upgrade indicator by executing the start_upgrade function.
See Appendix for the specifications of the start_upgrade function.

■ **How to call the start_upgrade function from the user program**

```
/// call "start_upgrade" function of the boot loader
start_upgrade (0x00BC);
```

To switch from the user program to the boot loader, reset the MCU after executing the start_upgrade function.

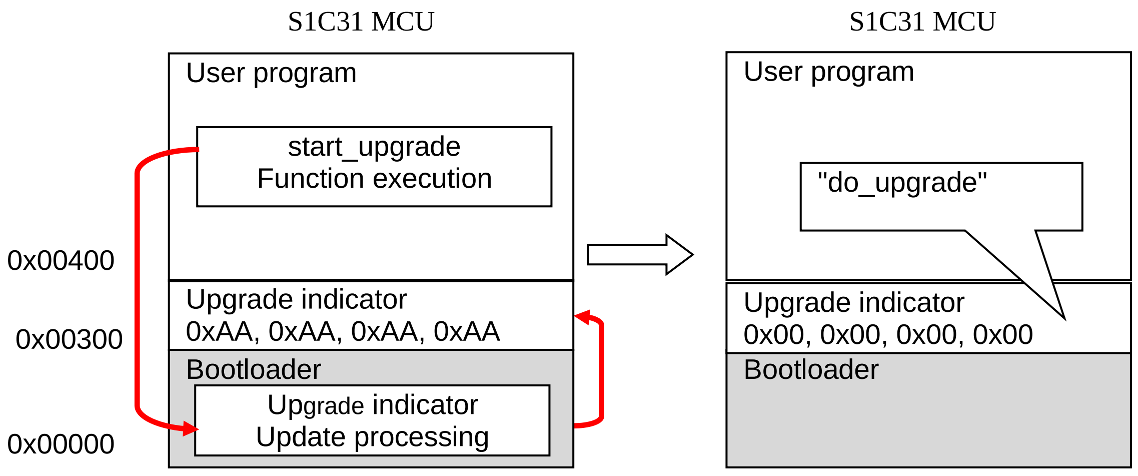
S1C31 MCU
User program
start_upgrade Function execution
0x00400
Upgrade indicator 0xAA, 0xAA, 0xAA, 0xAA
0x00300
Bootloader
Upgrade indicator Update processing
0x00000
S1C31 MCU
User program
"do_upgrade"
Upgrade indicator 0x00, 0x00, 0x00, 0x00
Bootloader

Figure 3.2.1.1 Rewriting upgrade indicator

3.3 Terminal software settings

Please download the new program by using terminal software that supports the serial port. For example, “TeraTerm” has been used to test the boot loader.

Tera Term : http://ttssh2.osdn.jp/index.html.en

Connect the PC and the UART of the S5U1C31xxxTx evaluation board with a USB-Serial cable, etc.

(1) Select [New connection] form [File] menu, and specify [Serial] and your [Port], then click the [OK] button.
(2) Select [Serial Port...] from [Setup] menu. The [Serial port setup] dialog box is displayed.
(3) Enter the following parameters:

Port:	COMx (x: your port number)
Baud Rate:	230400
Data:	8bit
Parity:	none
Stop:	1bit
Flow control:	Xon/Xoff
Transmit delay:	0msec/char, 0msec/line

(4) Click the [OK] button.
(5) Select [Send File] from [File] menu, and specify your PSA file. “TeraTerm” starts to download new program.

4. Memory Usage

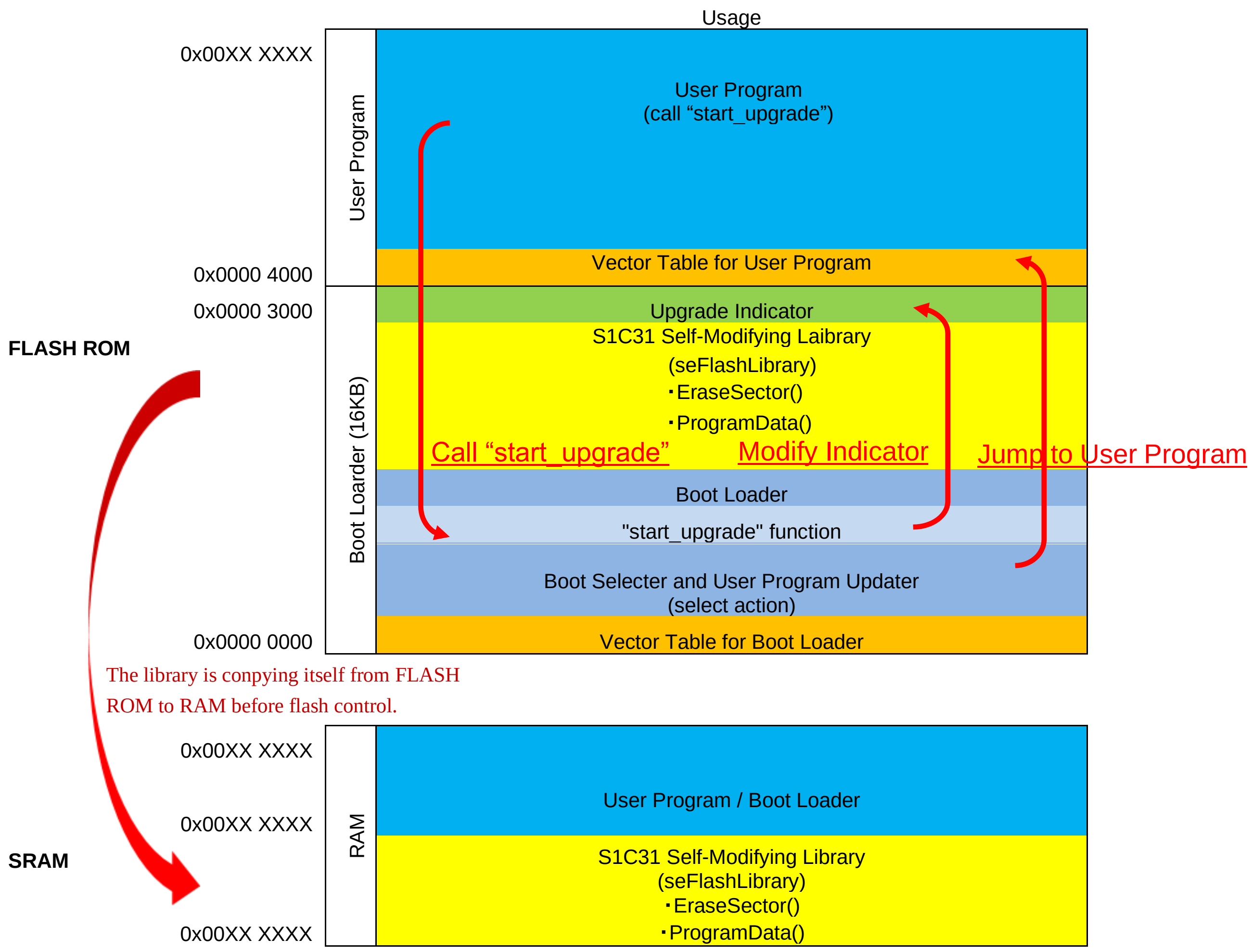

Figure 4.1 Memory Usage

The above 0x00XX XXXX depends on the memory size of the target model. For the memory size, refer to "S1C31xxx Technical Manual".
Also, please refer to flashLibraryForS1c31xxx_readme_j.txt for the memory size used by the S1C31 self-modifying library.

4.1 Reserved RAM for S1C31 self-Modifying Library

The S1C31 self-modifying library is included in the boot loader side, and the RAM used by the library is defined as a part of the boot loader.
When the user program (loaded program) uses "start_upgrade" function of the boot loader, the self-modifying library included in the boot loader works. At this time, the S1C31 self-modifying library uses part of RAM as working space. Please check the address range to be used in the map file of the boot loader project. Please arrange RAM used in the loadable program so as not to overlap with this area.

5. About debugging

Describes how to debug the bootloader and user programs.
The initial value of the upgrade indicator is defined in indicator.c. The boot loader behaves differently depending on the value of the upgrade indicator.

■ **When debugging the boot loader**
Change the initial value of the upgrade indicator to “do upgrade” (0x00,0x00,0x00,0x00) and write the bootloader (bootloader project) with the debugger.
The boot loader goes into update mode without calling the user program, allowing you to debug the boot loader at the source level.

■ **When debugging a user program**
(1) Change the initial value of the upgrade indicator to “upgrade_completed” (0xaa, 0xaa, 0xaa, 0xaa) and write the bootloader project with the debugger.
(2) Write the user program (loadsample project) with the debugger. In this state, the user program can be debugged at the source level.

The bootloader project only needs to be written once unless the Flash ROM is completely erased.
After that, modify the user program and write the debugger as necessary.

Appendix. Function specifications

The details of the functions provided by the boot loader and the user program are described below.

■ User program

Syntax		
void start_upgrade(uint32_t app_link_location)		
Arguments		
uint32_t	app_link_location	Normally set to 0x00bc
Return Value		
none		
Explanation		
Calls the upgrade indicator update processing routine stored in the boot loader vector table 47 (0x00BC) and rewrites the upgrade indicator to (0x00,0x00,0x00,0x00). (1) Disable interrupts (2) Call the function at the address defined in vector table 47 (0x00BC).		

■ Bootloader

Syntax
void Bootloader_IRQHandler(void)
Explanation
Upgrade indicator update processing routine Rewrite the upgrade indicator to “do upgrade” (0x00,0x00,0x00,0x00). (1) Disable interrupts (2) Write 0x00,0x00,0x00,0x00 to address 0x3000 with the self-modifying library.
Remarks
The address of this function is defined in vector table 47 (0x00BC). It is called by the user program.

Syntax
int main(void)
Explanation
This is the main routine of the boot loader. (1) WDT stop (2) Perform one of the following processes based on the judgment of the upgrade indicator. Jump to the user program (0x4000) Erase Flash user area UART reception processing / Flash data writing processing

Syntax		
void start_application(uint32_t app_link_location)		
Arguments		
uint32_t	app_link_location	User program address (0x4000)
Explanation		
Jump to the user program (0x4000) (1) Offset address (VTOR) is 0x4000 (2) Jump to the user program (0x4000)		

Syntax
int update(void)
Explanation
Waits for UART reception, analyzes the received data, and writes to Flash. (1) UART setting (2) Waiting for UART reception (3) Analysis of received data (4) Write data to Flash

Syntax		
void ConfigureDebugUART3_0(seUART_InitTypeDef* InitStruct)		
Arguments		
seUART_InitTypeDef*	InitStruct	UART set value structure
Return Value		
none		
Explanation		
Initialize the UART and set the port to be used. (1) Port setting used by UART (2) UART initial setting		
Remarks		
Edit if you want to change the UART port.		

Syntax	
int get_record(void)	
Return Value	
EXIT_SUCCESS	Line feed code received (when one line of S record is acquired)
EXIT_FAILURE	No line feed code received (when one line of S record is not acquired)
Explanation	

Monitors UART reception and acquires data for one line of S-record.

(1) Send the XON code and allow it to be sent to the PC terminal.

(2) Save the UART received data in the buffer, and when the line feed code is received, send the XOFF code to prohibit transmission to the PC terminal.

(3) If a line feed code is received, EXIT_SUCCESS is returned.

Syntax
void data_copy(void)
Explanation

Manages the UART receive buffer.

(1) Acquire odd data size less than one line of S record

(2) Copy odd data to the beginning of the buffer

Syntax		
int memory_write(struct srec_decode_t * srec)		
Arguments		
struct srec_decode_t *	srec	S record data
Return Value		
ARM_DRIVER_OK	Successful writing to Flash / SRAM	
ARM_DRIVER_ERROR	Failed to write to Flash / SRAM	
Explanation		

Writes the received S record data to Flash / SRAM.

(1) Judge whether the writing destination is Flash / SRAM from the address of the S record.

(2) For Flash, write data to Flash using the self-modifying library.

(3) In the case of SRAM, write the data to SRAM.

Syntax
void start_wdt(void)
Explanation
Start WDT for reset.

Syntax
void stop_wdt(void)
Explanation
Stop WDT for reset.

Syntax		
enum srec_type_t srec_decode_record(struct srec_decode_t * srec, char * record)		
Arguments		
struct srec_decode_t *	srec	Buffer address for storing decoded data
char *	record	S record data
Return Value		
srec_type_t	S-record type SREC_TYPE_S0~S9、SREC_TYPE_UNKNOWN, SREC_TYPE_ERROR_CHECKSUM, SREC_TYPE_ERROR_LENGTH	
Explanation		
Decodes S-records and converts them into types, addresses, data and checksums. (1) Get the type of S-record (2) Get of S-record address (3) Get of S-record data (4) Get and judgment of S-record checksum		
Remarks		
It checks the validity of the data received from the PC and converts it to a data format that can be written to Flash.		

Syntax		
void sbytes_decode_schars(unsigned char * sbytes, char * schars, int length)		
Arguments		
unsigned char *	sbyte	Buffer address for storing decoded data
char *	schars	ASCII data
int	length	Data length
Explanation		
Converts the ASCII data string to a numerical value and saves it in the decoded data storage buffer.		

Syntax		
unsigned char sbytes_checksum(unsigned char * sbytes, int length, unsigned char sum)		
Arguments		
unsigned char *	sbyte	Buffer address for storing decoded data
int	length	Data length
unsigned char	sum	checksum
Return Value		
unsigned char	checksum	
Explanation		
Calculates and returns the checksum of the decoded data.		

Syntax		
unsigned long sbytes_get_XXbit(unsigned char * sbytes)		
Arguments		
unsigned long	sbyte	Address Data storage buffer address
Return Value		
unsigned long	S-record address	
Explanation		
Converts an ASCII address (32Bit, 24Bit, 16Bit) into a numerical value and returns it. XX：32,24,16		

Syntax		
unsigned char schars_to_sbyte(char * schars)		
Arguments		
char *	schars	ASCIIcode
Return Value		
unsigned char	value	
Explanation		
Converts ASCII format data back to a number.		

Revision History

Attachment-1

Rev. No.	Date	Page	Category	Contents
Rev.1.0	Apl.12,2021	all	New	New establishment

EPSON

International Sales Operations

America

Epson America, Inc.

Headquarter:
3131 Katella Ave., Los Alamitos, CA 90720, USA
Phone: +1-562-290-4677

San Jose Office:
214 Devcon Drive
San Jose, CA 95112 USA
Phone: +1-800-228-3964 or +1-408-922-0200

Europe

Epson Europe Electronics GmbH
Riesstrasse 15, 80992 Munich,
Germany
Phone: +49-89-14005-0 FAX: +49-89-14005-110

Asia

Epson (China) Co., Ltd.
4th Floor, Tower 1 of China Central Place, 81 Jianguo Road, Chaoyang District, Beijing 100025 China
Phone: +86-10-8522-1199 FAX: +86-10-8522-1120

Shanghai Branch
Room 1701 & 1704, 17 Floor, Greenland Center II,
562 Dong An Road, Xu Hui District, Shanghai, China
Phone: +86-21-5330-4888 FAX: +86-21-5423-4677

Shenzhen Branch
Room 804-805, 8 Floor, Tower 2, Ali Center,No.3331
Keyuan South RD(Shenzhen bay), Nanshan District, Shenzhen
518054, China
Phone: +86-10-3299-0588 FAX: +86-10-3299-0560

Epson Taiwan Technology & Trading Ltd.
15F, No.100, Songren Rd, Sinyi Dist, Taipei City 110. Taiwan
Phone: +886-2-8786-6688

Epson Singapore Pte., Ltd.
438B Alexandra Road,
Block B Alexandra TechnoPark, #04-01/04, Singapore 119968
Phone: +65-6586-5500 FAX: +65-6271-7066

Epson Korea Co.,Ltd
10F Posco Tower Yeoksam, Teheranro 134 Gangnam-gu,
Seoul, 06235, Korea
Phone: +82-2-3420-6695

Seiko Epson Corp.
Sales & Marketing Division

Device Sales & Marketing Department
29th Floor, JR Shinjuku Miraina Tower, 4-1-6 Shinjuku,
Shinjuku-ku, Tokyo 160-8801, Japan

Document Code: 414177800
First Issue April 2021 in JAPAN