

RL78/G23

Self-programming using Boot Swap with BLE Communication

This application note describes the overview of self-programming using Bluetooth Low Energy (BLE) communication.

The smartphone sends write data via BLE communication and rewrites RL78/G23 flash memory through DA14531.

It uses the Flash Self Programming Library (Renesas Flash Driver RL78 Type01) to rewrite the flash memory and perform boot swap.

Target Device

RL78/G23

DA14531

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.

Reference

Documents related to this application note are listed below, refer to the following documents as well.

- RL78 Family Renesas Flash Driver RL78 Type01 User's Manual (R20UT4830)
- RL78 Family Renesas Flash Driver RL78 Type01 SC version (Code Flash) (R20AN0653)
- RL78 Family Renesas Flash Driver RL78 Type01 SC version (Data Flash) (R20AN0654)
- RL78/G23-128p Fast Prototyping Board User's Manual (R20UT4870)
- DA14531 SMARTBOND TINY™ MODULE Development Kit Hardware User Manual (UM-B-141)
- DA14531 Development Kit Pro Hardware User Manual (UM-B-114)
- DA14585-531 Serial Port Service Reference Application User Manual (UM-B-088)

Contents

1. Overview	4
1.1 Configuration	4
1.2 Project Configuration	5
2. Development Environments	6
2.1 Renesas Flash Driver RL78 Type01	7
2.2 SmartBond Flash Programmer	7
2.3 Serial Port Service (SPS)	7
2.4 Smart Console	7
3. External Specification	8
3.1 Sample Program Execution Procedure	8
3.1.1 Operation Environment Setup	8
3.1.1.1 Writing Firmware to DA14531	8
3.1.2 Sample Program Execution and Flash Memory Rewriting	10
3.1.2.1 Usage Procedure of Smart Console	11
3.2 LED Display Specifications	16
3.3 Message Specifications	16
3.4 Flowchart	17
3.4.1 Main Process	17
4. Hardware Descriptions	19
4.1 Example of Hardware Configuration	19
4.2 How to Connect	20
4.3 List of Used Pins	21
4.4 Setting of RL78/G23-128p Fast Prototyping Board	21
5. Internal Specification	22
5.1 Mode Selection	22
5.2 Option Byte Settings	22
5.3 Sharing Interrupt Vector	23
5.3.1 Registering Interrupt Handler of User Program	24
5.3.2 Boot Program Interrupt Handler	25
5.4 Smart Configurator Settings	27
5.5 Section Settings	28
5.5.1 Creating Program Files for RL78/G23	31
5.6 Defined Value	32
5.7 Specification of Functions (Boot Program)	33
5.7.1 List of Functions	33
5.7.2 Function Specifications	33
5.7.3 Note (Boot Program)	35
5.8 Specification of Functions (User Program)	36
5.8.1 List of Functions	36
5.8.2 Function Specifications	36
5.8.3 Note (User Program)	36

6. Reference Documents37

Revision History38

1. Overview

This application note describes a sample code to rewrite the flash memory of RL78/G23 and perform boot swap by self-programming.

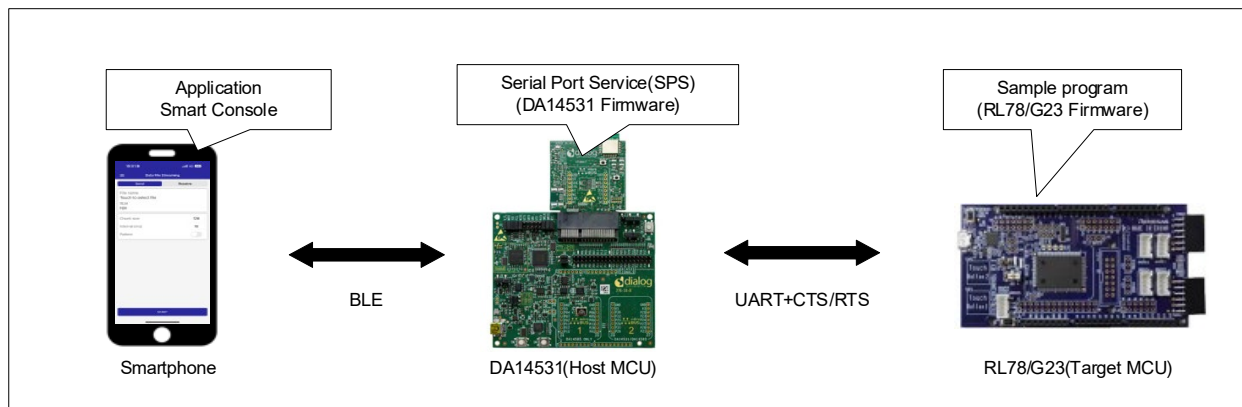
By sending data for writing from the smartphone using BLE communication of DA14531, RL78/G23 flash memory can be rewritten through DA14531.

1.1 Configuration

Connect RL78/G23 and DA14531 via UART (CTS/RTS flow-control) using RL78/G23-128p Fast Prototyping Board and DA14531MOD-00DEVKT-P.

- Smartphone
Use the app (Smart Console) to send RL78/G23 write data (Motorola S-format). The rewrite result is received by Smart Console.
Android and iOS are supported.
- Host MCU (DA14531)
It receives the write data sent from the smartphone via BLE and sends the data to the target MCU (RL78/G23) with UART (CTS/RTS flow-control).
In addition, the rewrite result is sent to the smartphone by BLE.
Firmware for DA14531 uses Serial Port Service (SPS).
- Target MCU (RL78/G23)
The write data is received from the host MCU (DA14531) by UART (CTS/RTS flow-control) and the data is used for rewriting the flash memory by self-programming.
Boot swap is executed after rewriting is completed.
This application note provides sample programming for RL78/G23.

Figure 1-1 System Configuration



1.2 Project Configuration

In this application note, the sample code of RL78/G23 is provided.

The sample code consists of two projects: a boot program and user program.

The boot program is a program that rewrites the flash memory by self-programming and is allocated to boot cluster 0 (0x00000 - 0x03FFF).

The user program is a program that implements user process and is allocated starting from address 0x08000.

Table 1-1 shows the project configuration.

Table 1-1 Project Configuration

Folder name	Description
rl78g23_RFD_with_UART	Sample program of boot program
rl78g23_FlashWriter_UserProgramSample	Sample program of user program

2. Development Environments

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

Table 2-1 Operation Confirmation Condition

Development tools		Descriptions
MCU (RL78/G23)		Part No.: R7F100GSN2DFB (Internal memory: Code Flash 128KB, RAM 48KB, ROM 144KB)
Evaluation board (RL78/G23)		RL78/G23-128p Fast Prototyping Board Part No.: RTK7RLG230CSN000BJ Note: Set VDD to 1.8 V or higher because this sample program runs in HS mode (high-speed on-chip oscillator frequency: 32 MHz).
Evaluation board (DA14531)		SmartBond TINY DA14531 Bluetooth Low Energy 5.1 System-on-Chip Development Kit Pro Part No.: DA14531MOD-00DEVKT-P
Debug tool (RL78/G23)		RL78 COM port debug tool of RL78/G23-128p Fast Prototyping Board is required to debug the sample program. When using E2 or E2 Lite, please refer to the RL78/G23-128p Fast Prototyping Board User's Manual and set jumpers, etc.
CS+	Integrated development environment	Renesas Electronics Corp. CS+ for CC V8.14.00
	C compiler	Renesas Electronics Corp. CC-RL V1.15.01
e2studio	Integrated development environment	Renesas Electronics Corp. e2studio V2025-07 (25.07.0)
	C compiler	Renesas Electronics Corp. CC-RL V1.15.01
IAR	Integrated development environment	IAR Systems Corp. IAR Embedded Workbench for Renesas RL78 V5.20
	C compiler	IAR Systems Corp. IAR C/C++ Compiler for Renesas RL78 V 5.20.1
Self-programming library		Renesas Flash Driver RL78 Type01
DA14531 Flash programmer tool		SmartBond Flash Programmer
DA14531 Firmware		Serial Port Service (SPS)
Application for smartphone		Smart Console

2.1 Renesas Flash Driver RL78 Type01

Renesas Flash Driver RL78 Type 01 (hereafter called RFD RL78 Type 01) is software for reprogramming the flash memory in the RL78/G2x.

For details, refer to Renesas Flash Driver RL78 Type01 Use's manual (R20UT4830).

Renesas Flash Driver RL78 Type01 (RFD) can be downloaded from the following URL.

<https://www.renesas.com/ja/software-tool/code-flash-libraries-flash-self-programming-libraries>

<https://www.renesas.com/ja/software-tool/data-flash-libraries>

2.2 SmartBond Flash Programmer

SmartBond Flash Programmer is a flash memory programming tool that supports DA14531, etc.

It can be obtained from the following URL.

<https://www.renesas.com/ja/software-tool/smartbond-flash-programmer>

2.3 Serial Port Service (SPS)

Serial Port Service (SPS) is a firmware that converts BLE communication and serial communication data compatible with DA14531, etc.

This application note uses SmartBond Flash Programmer to obtain firmware and write.

For more information on Serial Port Service (SPS), refer to the following URL.

<https://www.renesas.com/ja/software-tool/serial-port-service-sps>

2.4 Smart Console

Smart Console is an application for smartphones and supports Android and iOS.

The device to which the firmware of Serial Port Service (SPS) is written can be detected, and data transmission/reception by BLE communication is supported.

For more information about Smart Console, refer to the following URL.

<https://ipccs-docs.renesas.com/UM-140-DA145x-CodeLess/smartconsole.html>

3. External Specification

3.1 Sample Program Execution Procedure

3.1.1 Operation Environment Setup

The procedure for constructing the operating environment for this sample program is shown below.

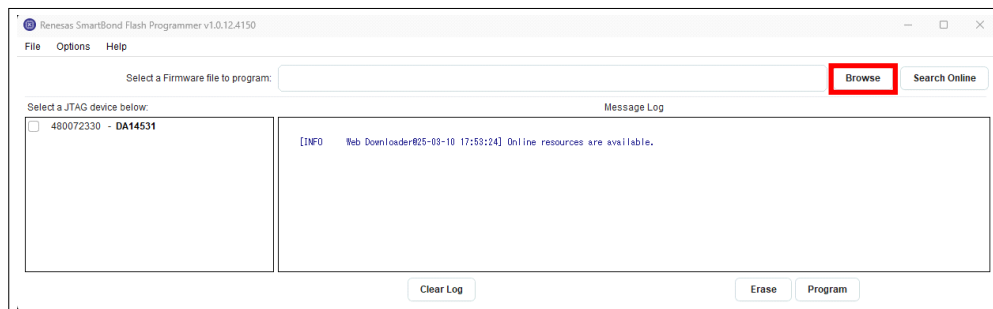
- ① Connect RL78/G23 and DA14531 referring to “4.2 How to Connect”.
- ② Build the example programs in “1.2 Project Configuration” and use Renesas Flash Programmer to write the firmware to RL78/G23.
For details about the file to be written, refer to “5.5.1 Creating Program Files for RL78/G23”.
- ③ Write the firmware “Serial Port Service (SPS)” to DA14531 referring to “3.1.1.1 Writing Firmware to DA14531”.

3.1.1.1 Writing Firmware to DA14531

To write the firmware “Serial Port Service (SPS)” to DA14531 (DA14531MOD-00DEVKT-P), follow the steps below.

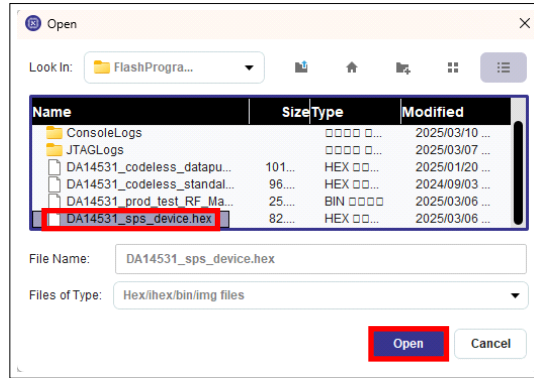
- ① Set DA14531MOD-00DEVKT-P jumper (J1) to the following settings:
1-2, 3-4, 5-6, 7-8, 9-10: Open (Use DA14531 SMARTBOND TINY MODULE SPI-Flash)
11-12, 13-14, 15-16, 17-18: Shorted (Use USB-UART conversion)
- ② Connect the PC and DA14531MOD-00DEVKT-P by USB cable.
- ③ Start SmartBond Flash Programmer.
- ④ Click “Browse”.

Figure 3-1 Click “Browse”



- ⑤ Select “DA14531_sps_device.hex” and click “Open”.

Figure 3-2 Select File

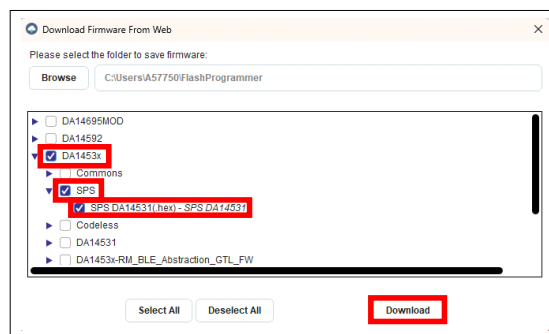


If “DA14531_sps_device.hex” does not exit, click “Search Online”, select “DA1453x”→“SPS”→“SPS DA14531(.hex) - SPS DA14531” and click “Download” to download “DA14531_sps_device.hex”.

Figure 3-3 Click “Search Online”



Figure 3-4 Select “SPS DA14531(.hex) - SPS DA14531” and Download



- ⑥ Select DA14531 from “Select a JTAG device below:”.

Figure 3-5 Select Device



- ⑦ Click “Erase” to erase DA14531 flash memory and then click “Program” to write the firmware.

Figure 3-6 Update DA14531 Firmware



3.1.2 Sample Program Execution and Flash Memory Rewriting

The following steps describe how to rewrite the flash memory of MCU(RL78/G23).

- ① Turn on the power to DA14531 (DA14531MOD-00DEVKT-P).
- ② Turn on the power to RL78/G23 while holding down the user switch (SW) on RL78/G23 (RL78/G23-128p Fast Prototyping Board). By starting RL78/G23 while holding down SW, the MCU enters the rewrite mode and the process on the boot program side is enabled.

If SW is not pressed, the user program is executed. For details on the rewrite mode, refer to “5.1 Mode Selection”.

- ③ Refer to “3.1.2.1 Usage Procedure of Smart Console” to send RL78/G23 data for writing from the smartphone and write the data.

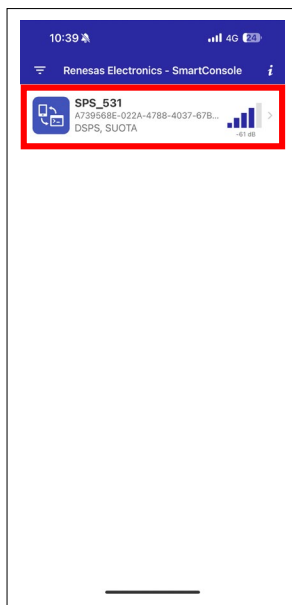
3.1.2.1 Usage Procedure of Smart Console

The following shows the usage procedure of Smart Console.

- ① Launch Smart Console app on the smartphone.

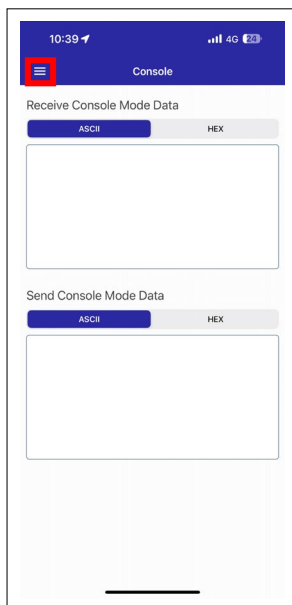
The selection screen of the connected device is displayed and select “SPS_531”.

Figure 3-7 Select Connected Device



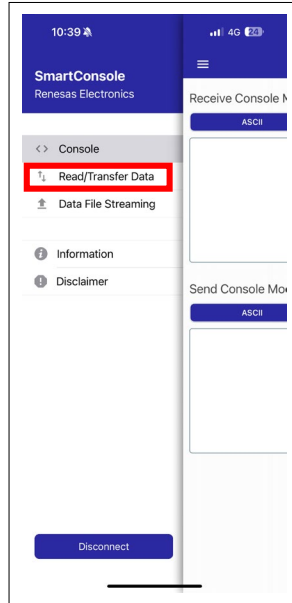
- ② Touch the upper left menu button.

Figure 3-8 Menu Button



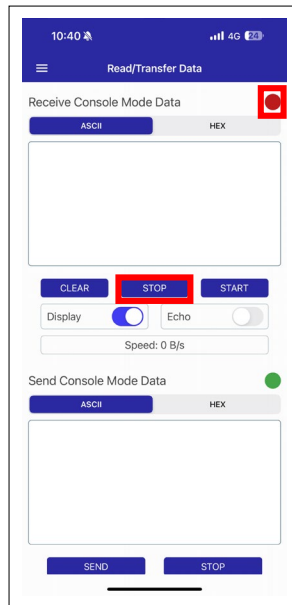
- ③ Select "Read/Transfer Data" menu.

Figure 3-9 Read/Transfer Data Menu



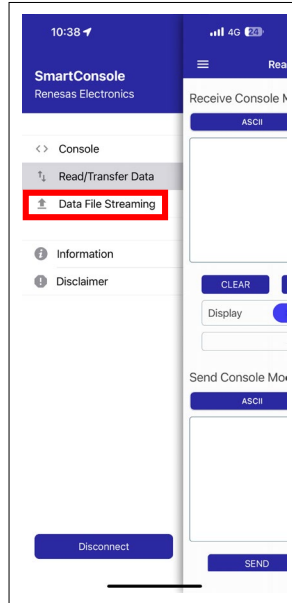
- ④ Touch "STOP" on "Receive Console Mode Data".
The circle in the upper right turns red when tapping "STOP" button.

Figure 3-10 STOP Button



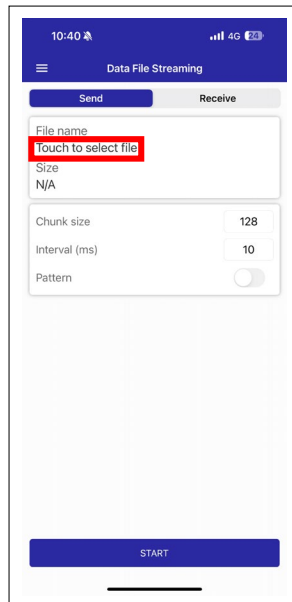
- ⑤ Select the menu button in the upper left and select “Data File Streaming” menu.

Figure 3-11 “Data File Streaming” Menu



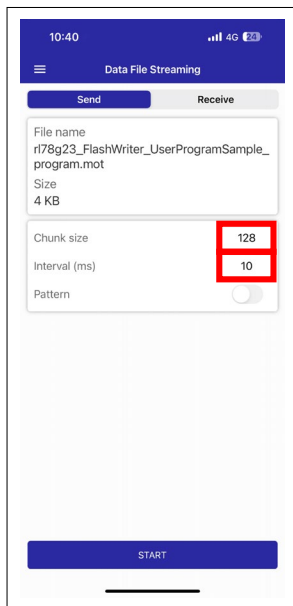
- ⑥ Touch File name area to select RL78/G23 writing data (Motorola S-format).
Specify the writing data for the user program or boot program.
For more details, refer to “5.5.1 Creating Program Files for RL78/G23”.

Figure 3-12 Select File



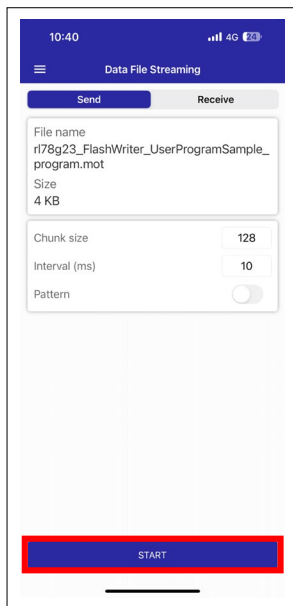
- ⑦ Set Chunk size to 128 and Interval (ms) to 10. (Initial settings)

Figure 3-13 Settings of Chunk size and Interval



- ⑧ Touch "START" to send the selected file (RL78/G23 writing data) via BLE communication.

Figure 3-14 Send File

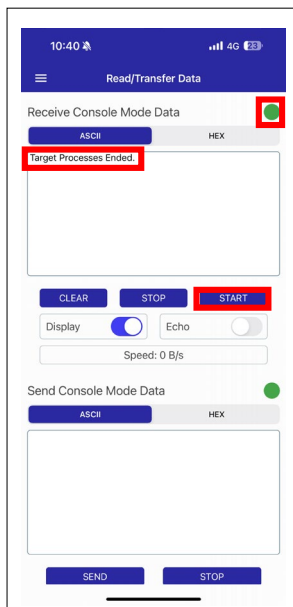


- ⑨ To check the result of writing, select the "Read/Transfer Data" menu from the menu button on the upper left, and touch the "START" button on the "Receive Console Mode Data".

The circle in the upper right turns green by touching "START" button. If writing is successful, "Target Processes Ended." is displayed.

For other message specifications, refer to "3.3 Message Specifications".

Figure 3-15 Confirm the Write Result



3.2 LED Display Specifications

Table 3-1 shows the relationship between RL78/G23 operating status and LED display of RL78/G23-128p Fast Prototyping Board.

Table 3-1 LED Display

Operating status	LED1	LED2
In operation	Lighting-on	Lighting-on
Abnormal end	Lighting-off	Lighting-off
Normal end	Lighting-off	Lighting-on

3.3 Message Specifications

RL78/G23 sends the write result to the smartphone as a string message via DA14531.

Table 3-2 shows the list of messages.

Table 3-2 Descriptions of Message

Message	Description
Target Processes Ended.	Write success Message when writing is successful.
S Record Error.	Invalid Motorola S-format data It occurs when the Motorola S format to be sent is invalid. This error also occurs when the data record in Motorola S format is not ascending address order.
S Record Address Error.	Invalid Motorola S-format address It occurs when the Motorola S-format address to be sent is invalid.
UART1 Occurred an Error.	UART communication error It occurs when there is an error in UART communication.
Target Occurred an Error.	Target write error It occurs when there is an error in the target writing process.

3.4 Flowchart

3.4.1 Main Process

Figure 3-16 shows the main processing operation of the RL78/G23 boot program.

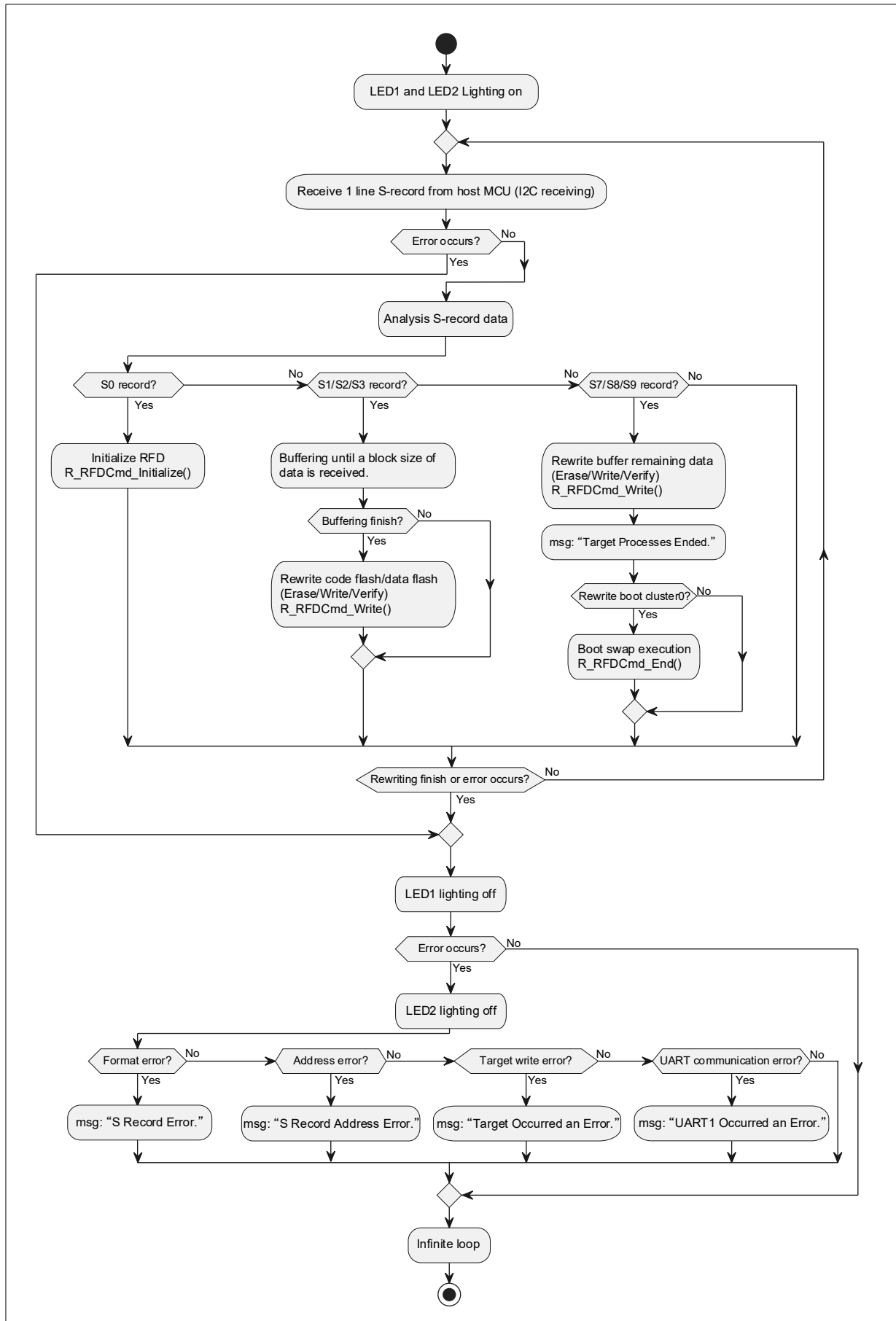
S-record data is received by UART (CTS/RTS flow control) from DA14531 and initialization, rewriting and boot swap (when rewriting boot cluster 1) are executed according to the record type.

The write target areas are code flash and data flash except boot cluster 1(0x004000 - 0x0007FFF).

The address range of the S-record file to be written to Boot Cluster 1 should be from 0x00000 to 0x03FFF. (Set the option bytes to 0x000C0 to 0x000C3 and the on-chip debug security IDs to 0x000C4 to 0x000CD) The boot program writes to Boot Cluster 1 by adding an offset of 0x04000 to the address.

When boot cluster 1 is rewritten, boot cluster 1 and boot cluster 0 are swapped because boot swap is executed last.

Figure 3-16 Main Process

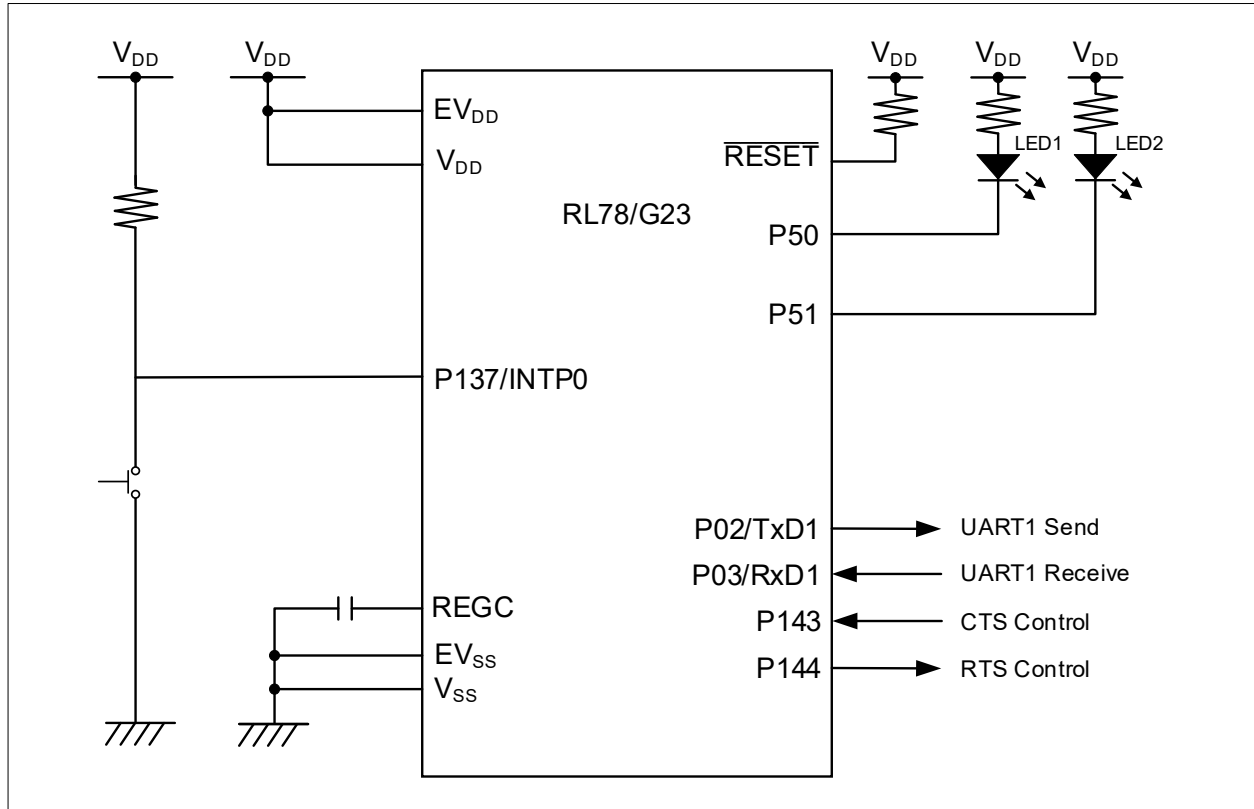


4. Hardware Descriptions

4.1 Example of Hardware Configuration

Figure 4-1 shows an example of the hardware configuration of RL78/G23 used in this application note.

Figure 4-1 RL78/G23 Hardware Configuration



Caution 1. This simplified circuit diagram was created to show an overview of connections only. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements. (Connect each input-only port to V_{DD} or V_{SS} through a resistor.)

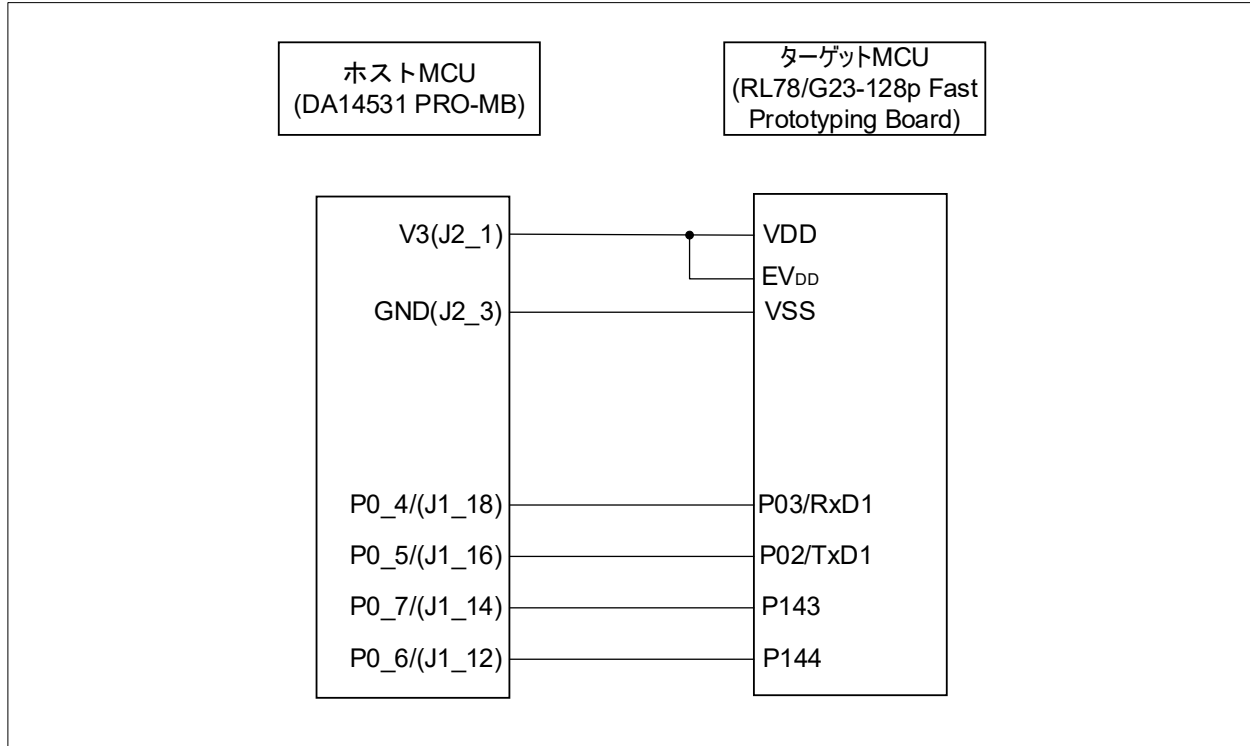
Caution 2. Connect the EV_{SS} pin to V_{SS} and the EV_{DD} pin to V_{DD} .

Caution 3. V_{DD} must be held at not lower than the reset release voltage (V_{LVD0}) that is specified as $LVD0$.

4.2 How to Connect

The following figure shows how to connect DA14531(DA14531MOD-00DEVKT-P) and RL78/G23.

Figure 4-2 Connection Diagram



Caution 1. The following settings are required for DA14531MOD-00DEVKT-P motherboard (DA14531 PRO-MB).

- Open 1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14, 15-16 and 17-18 of “J1”.
- Short-circuit (3.3V) 1-3 and 2-4 of “J5”.

For more details, refer to “DA14531 SMARTBOND TINY™ MODULE Development Kit Hardware User Manual (UM-B-141)” and “DA14531 Development Kit Pro Hardware User Manual (UM-B-114)”.

Caution 2. The following setting is required for RL78/G23-128p Fast Prototyping Board.

- Open “J20”.

For more details, refer to “RL78/G23-128p Fast Prototyping Board User’s Manual (R20UT4870)”.

4.3 List of Used Pins

Table 4-1 and Table 4-2 show the pins to be used and functions in the sample code of DA14531MOD-00DEVKT-P (Mother board: DA14531 PRO-MB) and RL78/G23.

Table 4-1 List of Used Pins (DA14531 PRO-MB)

Pin name	I/O	Function
P0_4(J1_18)	Output	UART transmission pin
P0_5(J1_16)	Input	UART receiving pin
P0_7(J1_14)	Output	RTS
P0_6(J1_12)	Input	CTS

Table 4-2 List of Used Pins (RL78/G23)

Pin name	I/O	Function
P02/TxD1	Output	UART transmission pin (UART1)
P03/RxD1	Input	UART receiving pin (UART1)
P143	Input	CTS
P144	Output	RTS
P50, P51	Output	Output to LED1 and LED2 pins
P137	Input	Mode selection (Rewriting mode/User mode)

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.

4.4 Setting of RL78/G23-128p Fast Prototyping Board

When using the E2 emulator Lite or the E2 emulator for debugging, make the following settings for the RL78/G23-128p Fast Prototyping Board.

- Cut the cut-patterns "TOOL0_USB", "RESET", and "T_RESET".
- Short-circuit (between 2 and 3 pins) pin headers "J15", "J16", and "J19".

Set the operating voltages of VDD and EVDD with "J20".

For details, refer to RL78/G23-128p Fast Prototyping Board User's Manual (R20UT4870).

5. Internal Specification

5.1 Mode Selection

The target MCU checks the status of RL78/G23-128p Fast Prototyping Board user switch (SW) in the startup routine of the boot program, and if SW is pressed (P137=Low) moves to the rewriting mode (boot program processing), and if SW is not pressed (P137=High) moves to the user mode (user program processing).

5.2 Option Byte Settings

Table 5-1 shows the option byte settings used in the sample program.

Table 5-1 Option Byte Settings

Address	Setting Value	Contents
000C0H/040C0H	01101111B	Disables the watchdog timer
000C1H/040C1H	11111110B	LVD0 detection voltage: Reset mode At rising edge TYP. 1.90 V (1.84 V ~ 1.95 V) At falling edge TYP. 1.86 V (1.80 V ~ 1.91 V)
000C2H/040C2H	11101000B	HS mode, High-speed on-chip oscillator clock: 32MHz
000C3H/040C3H	10000100B	Enables on-chip debugging

RL78/G23's option bytes consist of user option bytes (000C0H to 000C2H) and on-chip debug option byte (000C3H).

5.3 Sharing Interrupt Vector

Since the boot program and the user program are separate projects, and the interrupt vectors cannot be shared as is, a mechanism is provided for executing user program interrupt handler functions from the boot program.

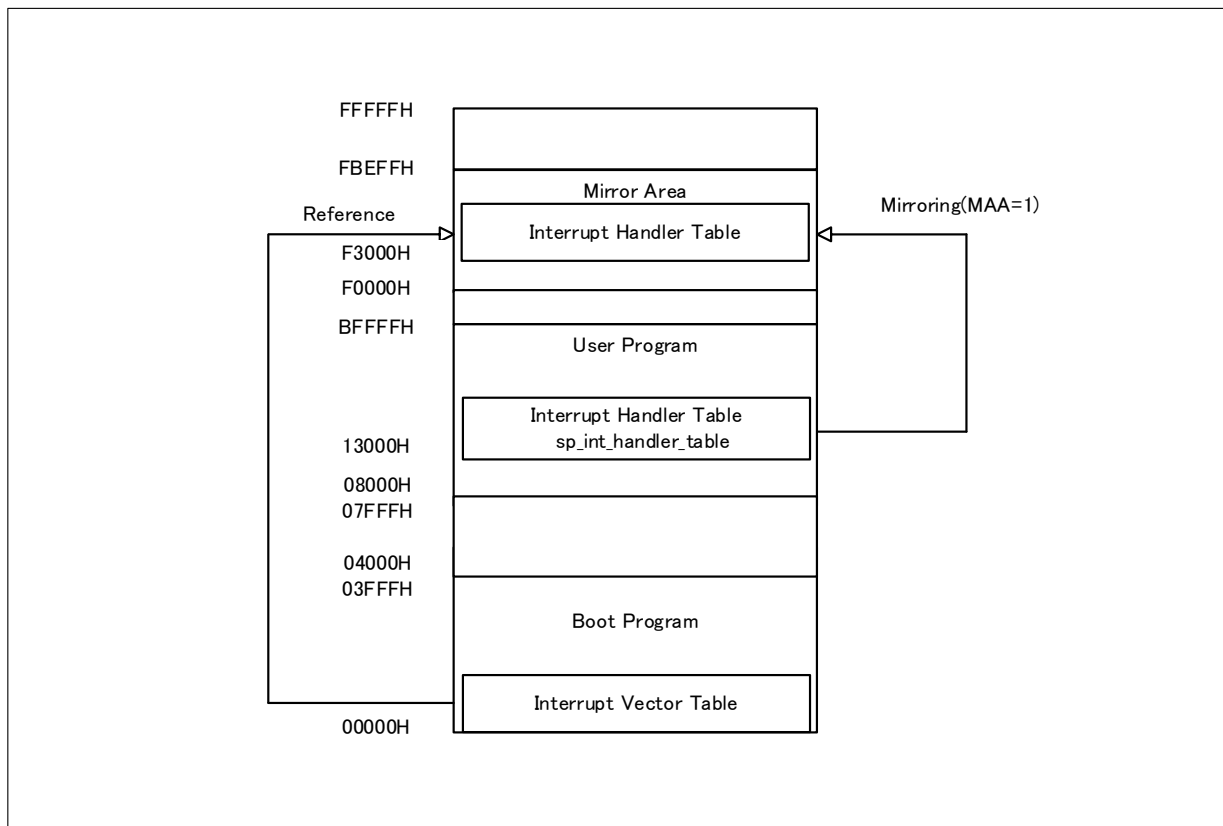
The user program allocates the interrupt handler table for registering the interrupt handler used by the user program at a fixed address (0x13000).

In addition, the user-program startup routines set MAA in PMC register to 1 so that the interrupt handler table is mirrored at 0xF3000 address.

The boot program implements the process of jumping to the corresponding vector address by referencing the interrupt handler table of the user program at 0xF3000 address in the interrupt handler of the boot program.

The interrupt (INTST1, INTSR1, INTSRE1) used in UART1 cannot be used in the user program because it is used in the boot program.

Figure 5-1 Sharing the Interrupt Vector



5.3.1 Registering Interrupt Handler of User Program

The interrupt handler table for the user program is defined in `int_handlers.c` in the project of the user program.

Add the interrupt handler used by the user program to `sp_int_handler_table`.

For `sp_int_handler_table[0]`, specify the user program start address.

In the sample program, `INTTM01` is used in LED blinking process.

Figure 5-2 Interrupt Handler Table

```

/*****
Macro definitions
*****/
#define USER_PROGRAM_START          (0x8000)

/*****
Static variables and functions
*****/
#pragma section const USER_INT_HANDLERS
const int_handler_t sp_int_handler_table[64] =
{
    (int_handler_t) (USER_PROGRAM_START), /* RST (Note: This must be a start point of your program.) */
    NULL, /* INTDBG(do't set a handler here!) */
    NULL, /* INTWDTI */
    NULL, /* INTLVI */
    NULL, /* INTPO */
    NULL, /* INTP1 */
    NULL, /* INTP2 */
    NULL, /* INTP3 */
    (Omitted.)
    R_Config_TAU0_1_interrupt, /* INTTM01 */
    NULL, /* INTTM14 */
    NULL, /* INTTM15 */
    NULL, /* INTTM16 */
    NULL, /* INTTM17 */
    NULL, /* No Use */
    NULL, /* No Use */
    NULL, /* BRK_I */
};
#pragma section

```

5.3.2 Boot Program Interrupt Handler

Each interrupt handler in the boot program refers to the interrupt handler table of the user program and implements the processing to jump to the corresponding vector address.

It is defined by `int_handlers.c` and `int_handlers_tmpl.h` in the project of the boot program.

Figure 5-3 Boot Program Interrupt Handler (`int_handlers.c`)

```
#define INT_VECT    INTWDTI
#define R_HANDLER  r_handler_INTWDTI_vect
#include "int_handlers_tmpl.h"

#define INT_VECT    INTLVI
#define R_HANDLER  r_handler_INTLVI_vect
#include "int_handlers_tmpl.h"

#define INT_VECT    INTPO
#define R_HANDLER  r_handler_INTPO_vect
#include "int_handlers_tmpl.h"

#define INT_VECT    INTP1
#define R_HANDLER  r_handler_INTP1_vect
#include "int_handlers_tmpl.h"

#define INT_VECT    INTP2
#define R_HANDLER  r_handler_INTP2_vect
#include "int_handlers_tmpl.h"

(Omitted.)

#define INT_VECT    INTTM16
#define R_HANDLER  r_handler_INTTM16_vect
#include "int_handlers_tmpl.h"

#define INT_VECT    INTTM17
#define R_HANDLER  r_handler_INTTM17_vect
#include "int_handlers_tmpl.h"

/* software break handler */
#define INT_VECT    0x7E
#define R_HANDLER  r_handler_BRK_vect
#include "int_handlers_tmpl.h"
```

Figure 5-4 Boot Program Interrupt Handler (int_handlers_tmpl.h)

```
#ifndef R_HANDLER
#if INT_VECT == 0x7E
#pragma interrupt_brk R_HANDLER
#else
#pragma interrupt R_HANDLER(vect=INT_VECT)
#endif

static void __near R_HANDLER(void)
{
    int_handler_t __far * int_handler_table= INT_HANDLER_ADDRESS;

    if (int_handler_table[INT_VECT>>1])
    {
        int_handler_table[INT_VECT>>1]();
    }
}

#undef R_HANDLER
#undef INT_VECT

#endif
```

5.4 Smart Configurator Settings

The settings of the boot program by Smart Configurator are shown below.

(1) Initialize the I/O port.

- Set P50, P51 and P144 to the output port.
- Set P143 to the input port.

(2) Initialize the I2C (Slave mode).

- Use UART1. (P02 is used as TXD1, P03 is used as RXD1.)
- Set the Operation clock to CK00, set the Clock source to fCLK.
- Set the Transfer mode to Single transfer mode.
- Set the Data length to 8 bits.
- Set the Transfer direction to LSB.
- Set the Parity to None.
- Set the Stop bit length to 1 bit.
- Set the Transfer data level to Non-reverse.
- Set the Transfer rate to 921600bps.
- Set the Transmit end interrupt priority (INTST1) to Level 3.
- Set the Reception end interrupt priority (INTSR1) to Level 3.
- Set the Reception error interrupt priority (INTSRE1) to Level 3.
- Select the Transmission end, Reception end and Reception error of the Callback function setting.

(3) Initialize the Voltage Detector.

- Select the Operation mode to Reset mode.
- Set the Reset generation level (VLVD0) to 1.86V.

5.5 Section Settings

Table 5-2 and Table 5-3 show the CC-RL section settings for the boot program and the user program.

- Sections beginning with "." are reserved by the compiler.
- Sections beginning with "RFD_" or "SMP_" are defined by the specifications of the Renesas Flash Driver RL78 Type01(RFD) and its sample program.
- Though reserved section names of IAR vary, IAR has a similar section configuration.

Table 5-2 List of Boot Program Section

Section Name	Address	Description
.text	0x000D8	Section for code (allocated to the near area)
.RLIB	(Automatic allocation)	Section for code of runtime libraries
.SLIB	(Automatic allocation)	Section for code of standard libraries
.textf	(Automatic allocation)	Section for code (allocated to the far area)
.constf	(Automatic allocation)	ROM data (allocated to the far area)
RFD_DATA_n	(Automatic allocation)	Section for RFD: Data section for initialized global variables
RFD_CMN_f	(Automatic allocation)	Section for RFD: Program section of API functions used in common for flash memory control
RFD_CF_f	(Automatic allocation)	Section for RFD: Program section of API functions for code flash memory control
RFD_DF_f	(Automatic allocation)	Section for RFD: Program section of API functions for data flash memory control
RFD_EX_f	(Automatic allocation)	Section for RFD: Program section of API functions for data flash memory control
SMP_CMN_f	(Automatic allocation)	Section for RFD sample: Program section of sample functions used in common for flash memory control
SMP_CF_f	(Automatic allocation)	Section for RFD sample: Program section of sample functions for code flash memory control
SMP_CMN_n	(Automatic allocation)	Section for SMP: Program section of sample functions used in common for flash memory control
SMP_DF_f	(Automatic allocation)	Section for RFD sample: Program section of sample functions for data flash memory control
.data	(Automatic allocation)	Section for near initialized data (with initial value)
.sdata	(Automatic allocation)	Section for initialized data (with initial value, variable allocated to saddr)
.dataR	0xF3F00	Section for near initialized data (with initial value) (Data copied from ROM area)
.bss	(Automatic allocation)	Section for data area (without initial value, allocated to the near area)
RFD_DATA_nR	(Automatic allocation)	Section for RFD: Data section for initialized global variables (Data copied from ROM area)
RFD_CMN_fR	(Automatic allocation)	Section for RFD: Program section of API functions used in common for flash memory control (Program copied from ROM area)
RFD_CF_fR	(Automatic allocation)	Section for RFD: Program section of API functions for code flash memory control (Program copied from ROM area)
RFD_EX_fR	(Automatic allocation)	Program section of API functions for extra area control (Program copied from ROM area)
SMP_CMN_fR	(Automatic allocation)	Section for RFD sample: Program section of sample functions used in common for flash memory control (Program copied from ROM area)
SMP_CF_fR	(Automatic allocation)	Section for RFD sample: Program section of sample functions for code flash memory control (Program copied from ROM area)
.sdataR	0xFFE20	Section for initialized data (with initial value, variable allocated to saddr) (Data copied from ROM area)
.sbss	(Automatic allocation)	Section for data area (without initial value, variable allocated to saddr)

Table 5-3 List of User Program Section

Section Name	Address	Description
.text	0x8000	Section for code (allocated to the near area)
.RLIB	(Automatic allocation)	Section for code of runtime libraries
.SLIB	(Automatic allocation)	Section for code of standard libraries
.textf	(Automatic allocation)	Section for code (allocated to the far area)
.constf	(Automatic allocation)	ROM data (allocated to the far area)
.data	(Automatic allocation)	Section for near initialized data (with initial value)
.sdata	(Automatic allocation)	Section for initialized data (with initial value, variable allocated to saddr)
USER_INT_HANDLERS_n	0x13000	Section for user interrupt handler table
.const	(Automatic allocation)	ROM data (allocated to the near area) (mirror area)
.dataR	0xF3F00	Section for near initialized data (with initial value) (Data copied from RAM area)
.bss	(Automatic allocation)	Section for data area (without initial value, allocated to the near area)
.sdataR	0xFFE20	Section for initialized data (with initial value, variable allocated to saddr) (Data copied from RAM area)
.sbss	(Automatic allocation)	Section for data area (without initial value, variable allocated to saddr)

5.5.1 Creating Program Files for RL78/G23

In this sample program, the boot program is assigned to the address 0x00000~0x03FFF and the user program is assigned to the address 0x08000~0xBFFFF to support the boot swap.

For the boot program, only 0x00000~0x03FFF data should be used because the build settings add data for OCD monitoring to the final area of the code flash memory.

For the user program, the data (interrupt vector table, optional bytes, security ID and OCD monitoring) is automatically generated below 0x08000 regardless of the section setting, so they are deleted.

In CS+ and e2studio, the split output file function of CC-RL enables the setting to output only the required data.

Please use r178g23_RFD_with_UART_boot.mot for the boot program and r178g23_FlashWriter_UserProgramSample_program.mot for the user program.

Figure 5-5 Split Output File Settings of Boot Program (CS+)

▼ Frequently Used Options(for Hex Output)	
Output hex file	Yes
Hex file format	Motorola S-record file(-F0m=Stype)
Output folder	%BuildModeName%
Output file name	%ProjectName%.mot
▼ Division output file [0]	Division output file[1] %BuildModeName%¥%ActiveProjectName%_boot.mot=0-7FFF

Figure 5-6 Split Output File Settings of User Program (CS+)

▼ Frequently Used Options(for Hex Output)	
Output hex file	Yes
Hex file format	Motorola S-record file(-F0m=Stype)
Output folder	%BuildModeName%
Output file name	%ProjectName%.mot
▼ Division output file [0]	Division output file[2] %BuildModeName%¥%ActiveProjectName%_boot.mot=0-7FFF
[1]	%BuildModeName%¥%ActiveProjectName%_program.mot=8000-BFFFF

For IAR, SRecordSplitter.bat is used because there is no split output file function.

(Located in r178g23_FlashWriter_UserProgramSample¥Workspace¥IAREW¥)

At the command prompt, run the following to extract only the records after 0x08000.

Table 5-4 How to use SRecordSplitter.bat

```
> ./SRecordSplitter.bat <original S-Record file> <new S-Record file>
```

5.6 Defined Value

SRECORD_TYPE is an enumeration and defines the S record type of the Motorola S-format.

Table 5-5 shows the specifications of SRECORD_TYPE.

Table 5-5 SRECORD_TYPE

Defined Value	Description
SRECORD_0	S0 record
SRECORD_1	S1 record
SRECORD_2	S2 record
SRECORD_3	S3 record
SRECORD_4	S4 record
SRECORD_5	S5 record
SRECORD_6	S6 record
SRECORD_7	S7 record
SRECORD_8	S8 record
SRECORD_9	S9 record
SRECORD_ERR	Invalid S record data

SRECORD_DATA structure is used to temporarily store Motorola S format S record analysis data.

Table 5-6 shows the specifications of SRECORD_DATA structure.

Table 5-6 SRECORD_DATA Structure

Member	Description
SRECORD_TYPE type	Motorola S record type
uint32_t address	Motorola S record address
uint16_t size	Motorola S record size
uint8_t data[SRECORD_DATA_SIZE]	Motorola S record data (SRECORD_DATA_SIZE=128)

5.7 Specification of Functions (Boot Program)

5.7.1 List of Functions

Table 5-7 shows the major functions used in the sample code (boot program).

Table 5-7 List of Functions (Boot Program)

Function Name	Outline
R_Decode_SRecord	Analyzes S record.
R_RFDCmd_Initialize	Initializes RFD.
R_RFDCmd_Write	Rewrite the flash memory.
R_RFDCmd_End	Executes boot swap.
R_Config_UART1_SendMessage	Sends message by using UART1.

5.7.2 Function Specifications

The function specifications of the sample program (boot program) are shown below.

R_Decode_SRecord	
Outline	Analyzes S record.
Declaration	void R_Decode_SRecord(uint8_t * srecord_str, uint16_t len, SRECORD_DATA * srecord_data)
Argument	srecord_str: S record string (1 line) len: Length of srecord_str srecord_data: Analysis data of S record
Return Value	None
Description	Analyzes S record and sets the analysis results in the SRECORD_DATA structure.

R_RFDCmd_Initialize	
Outline	Initializes RFD.
Declaration	e_rfd_ret_t R_RFDCmd_Initialize(void)
Argument	None
Return Value	R_RFD_ENUM_RET_STS_OK: Success Other: Failure
Description	Executes R_RFD_Init() and initializes RFD.

R_RFDCmd_Write	
Outline	Rewrite the flash memory.
Declaration	<code>e_rfd_ret_t R_RFDCmd_Write(uint32_t start_addr, uint16_t length, uint8_t __near * p_data)</code>
Argument	start_addr: Write start address length: Data size p_data: Data for writing
Return Value	R_RFD_ENUM_RET_STS_OK: Success Other: Failure
Description	The following process is used to rewrite the flash memory data. <ol style="list-style-type: none"> 1. Set interrupt vectors in RAM. 2. Set flash memory control mode to code flash programming mode or data flash programming mode. 3. Perform a blank check. 4. Perform a block erase (if non-blank). 5. Performs a writing operation. 6. Set flash memory control mode to non-write mode. 7. Perform Verify (compare write data with read data). 8. Return interrupt vector to ROM.

R_RFDCmd_End	
Outline	Executes boot swap.
Declaration	<code>e_rfd_ret_t R_RFDCmd_End(void)</code>
Argument	None
Return Value	R_RFD_ENUM_RET_STS_OK: Success Other: Failure
Description	After executing boot swap, executes software reset.

R_Config_UART1_SendMessage	
Outline	Sends message by using UART1.
Declaration	<code>MD_STATUS R_Config_UART1_SendMessage(const char * p_msg, uint8_t len)</code>
Argument	p_msg: Pointer for displayed message string len: Length of displayed message
Return Value	MD_OK: Success MD_ERROR : Failure
Description	Sends message to DA14531 using UART1.

5.7.3 Note (Boot Program)

If this sample program is re-generated after the Board support package (bsp) version has been changed, an error may occur. Please add the following description to cstart.asm in the newly generated file.

Figure 5-7 cstart.asm of the Boot Program

```

(Omitted.)
;-----
;  startup
;-----
.section .text, TEXT
_start:
;=====
; Jump to a user-program if SW is off (P137 == 1)
PORT13      .EQU 0xFFFD
USER_INT_HANDLERS .EQU 0x13000
    BF PORT13.7, $_jump_user_program_end
    MOV ES, #HIGHW(USER_INT_HANDLERS)
    MOVW HL, #LOWW(USER_INT_HANDLERS)
    MOVW AX, ES:[HL+0]
    BR AX
_jump_user_program_end:
;=====

;-----
;  setting register bank
;-----
$IFDEF BSP_CFG_ASM_RAM_GUARD_START_ADDRESS
    MOV !RAMSAR, #BSP_CFG_ASM_RAM_GUARD_START_ADDRESS
$ENDIF
(Omitted.)

```

5.8 Specification of Functions (User Program)

5.8.1 List of Functions

Table 5-8 shows the major function used in the sample code (user program).

Table 5-8 List of Functions (User Program)

Function Name	Outline
main	Main processing

5.8.2 Function Specifications

The function specifications of the sample code (user program) are shown below.

main	
Outline	Main processing
Declaration	void main(void)
Argument	None
Return Value	None
Description	Perform LED blinking using interval timer (INTTM01).

5.8.3 Note (User Program)

If this sample program is re-generated after the Board support package (bsp) version has been changed, an error may occur. Please change the mirror area setting in cstart.asm in the newly generated file as follows.

Figure 5-8 cstart.asm of the User Program

```
(Omitted.)

;-----
; setting mirror area
;-----
ONEB !PMC : mirror area = 10000-1FFFFH

;-----
; setting the stack pointer
;-----

(Omitted.)
```

6. Reference Documents

RL78/G23 User's Manual: Hardware (R01UH0896)

RL78/G23-128p Fast Prototyping Board User's Manual (R20UT4870)

RL78 Family Renesas Flash Driver RL78 Type 01 User's Manual (R20UT4830)

RL78 Family Renesas Flash Driver RL78 Type 01 SC version (Code Flash) (R20AN0653)

RL78 Family Renesas Flash Driver RL78 Type 01 SC version (Data Flash) (R20AN0654)

RL78 Family Renesas Flash Driver RL78 Type 01 SC version (Extra Area) (R20AN0655)

RL78 Family Renesas Flash Driver RL78 Type 01 SC version (Common) (R20AN0656)

DA14531 SMARTBOND TINY™ MODULE Development Kit Hardware User Manual (UM-B-141)

DA14531 Development Kit Pro Hardware User Manual (UM-B-114)

DA14585-531 Serial Port Service Reference Application User Manual (UM-B-088)

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

All trademarks and registered trademarks are the property of their respective owners.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Aug.31.25	—	First Edition

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

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

1. Precaution against Electrostatic Discharge (ESD)

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

2. Processing at power-on

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

3. Input of signal during power-off state

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

4. Handling of unused pins

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

5. Clock signals

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

6. Voltage application waveform at input pin

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

7. Prohibition of access to reserved addresses

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

8. Differences between products

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

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Trademarks

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

Contact information

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

www.renesas.com/contact/.