



# EZ-CUBE

## On-Chip Debug Emulator with Programming Function

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (<http://www.cn.renesas.com>)

#### 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; anticrime 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 majorityowned subsidiaries.

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

## Preface

The EZ-CUBE emulator (YRCNEZCUBE01) is designed for use with the MCU's made by Renesas Electronics. You can download the latest manuals from the Renesas Tools homepage (<http://www.cn.renesas.com/EZ-CUBE>).

## Important

Before using the emulator, be sure to read this user's manual carefully. Keep this user's manual, and refer to it when you have questions about the emulator.

### Emulator:

"Emulator" in this user's manual collectively refers to the EZ-CUBE emulator manufactured by Renesas Electronics Corporation.

"Emulator" herein encompasses neither the customer's target system nor the host machine.

### Purpose of use of the emulator:

This emulator is a device to support the development of systems that uses the Renesas microcomputers. It provides support for system development in both software and hardware. The emulator is not guaranteed for use in the production line. Be sure to use the emulator correctly according to said purpose of use. Please avoid using the emulator other than for its intended purpose of use.

### For those who use the emulator:

The emulator can only be used by those who have carefully read the user's manual and know how to use it. Use of the emulator requires basic knowledge of electric circuits, logical circuits, and MCUs.

### When using the emulator:

- (1)The emulator is a development-support unit for use in your program development and evaluation stages. When a program you have finished developing is to be incorporated in a mass-produced product, the judgment as to whether it can be put to practical use is entirely your own responsibility, and should be based on evaluation of the device on which it is installed and other experiments.
- (2)In no event shall Renesas Electronics Corporation be liable for any consequence arising from the use of the emulator.
- (3)Renesas Electronics Corporation strives to provide workarounds for and correct trouble with products malfunctions. However, this does not necessarily mean that Renesas Electronics Corporation guarantees the provision of a workaround or correction under any circumstances.
- (4)The emulator covered by this document has been developed on the assumption that it will be used for program development and evaluation in laboratories.
- (5)Renesas Electronics Corporation cannot predict all possible situations and possible cases of misuse that carry a potential for danger. Therefore, the warnings in this user's manual and the warning labels attached to the emulator do not necessarily cover all such possible situations and cases. The customer is responsible for correctly and safely using the emulator.
- (6)The emulator covered by this document has not been through the process of checking conformance with UL or other safety standards and IEC or other industry standards. This fact must be taken into account when the emulator is taken from Japan to some other country.
- (7)Renesas Electronics Corporation will not assume responsibility of direct or indirect damage caused by an accidental failure or malfunction in the emulator.

### When disposing of the emulator:

Penalties may be applicable for incorrect disposal of this waste, in accordance with your national legislation.

**Usage restrictions:**

The emulator has been developed as a means of supporting system development by users. Therefore, do not use it as an embedded device in other equipment. Also, do not use it to develop systems or equipment for use in the following fields.

- (1) Transportation and vehicular
- (2) Medical (equipment that has an involvement in human life)
- (3) Aerospace
- (4) Nuclear power control
- (5) Undersea repeaters
- (6) Military related business or development of Weapon of Mass Destruction

If you are considering the use of the emulator for one of the above purposes, please be sure to consult your local distributor.

**About product changes:**

We are constantly making efforts to improve the design and performance of our product. Therefore, the specification or design of the emulator, or this user's manual, may be changed without prior notice.

**About rights:**

- (1) We assume no responsibility for any damage or infringement on patent rights or any other rights arising from the use of any information, products or circuits presented in this user's manual.
- (2) The information or data in this user's manual does not implicitly or otherwise grant a license to patent rights or any other rights belonging to Renesas or to a third party.
- (3) This user's manual and the emulator are copyrighted, with all rights reserved by Renesas. This user's manual may not be copied, duplicated or reproduced, in whole or part, without prior written consent from Renesas.

**About diagrams:**

Some diagrams in this user's manual may differ from the objects they represent.

**Terminology**

The meanings of the terms used in this manual are described in the table below.

Term	Meaning
EZ-CUBE	Generic name of EZ-CUBE
Target device	This is the device to be emulated.
Target system	This is the system to be debugged (user-created system). It includes software and hardware created by the user.
CS+	It is an integrated development environment.
Firmware	Program embedded in the device for controlling EZ-CUBE
RFP	Renesas Flash Programmer, GUI software used to perform flash programming.

# CAUTION

## **Caution to Be Taken for System Malfunctions:**

If the emulator malfunctions because of interference like external noise, do the following to remedy the trouble.

- (1) Exit the emulator debugger, and shut OFF the emulator and the target system.
- (2) After a lapse of 10 seconds, turn ON the power of the emulator and the target system again, then launch the emulator debugger.

## **Note**

Renesas Electronics (China) does not assume any liability for the user does not follow the user manual for the use of non-normal and non-practice due to loss of product failure and other related.

# CONTENTS

<b>CHAPTER 1</b>	<b>OVERVIEW</b> .....	<b>7</b>
1.1	Features .....	7
1.2	Notes Before Using EZ-CUBE .....	8
1.3	Hardware Specifications .....	9
1.4	Firmware Update .....	9
1.5	Standard configuration .....	10
<b>CHAPTER 2</b>	<b>NAMES AND FUNCTIONS OF HARDWARE</b> .....	<b>11</b>
2.1	Part Names and Functions of EZ-CUBE .....	11
2.2	System configuration .....	13
2.3	Setup .....	13
2.3.1	<b>Installing Emulator Software</b> .....	13
2.3.2	<b>System startup procedure</b> .....	13
2.3.3	<b>System shutdown procedure</b> .....	14
<b>CHAPTER 3</b>	<b>HOW TO USE EZ-CUBE WITH RL78 MICROCONTROLLER</b> .....	<b>15</b>
3.1	Target System Design .....	16
3.1.1	<b>Pin assignment</b> .....	16
3.1.2	<b>Circuit connection example</b> .....	17
3.1.3	<b>Connection of reset pin</b> .....	18
3.2	On-Chip Debugging .....	20
3.2.1	<b>Debug functions</b> .....	20
3.2.2	<b>Securing of user resources and setting of security ID and on-chip debug option byte debugging resources</b> .....	20
3.2.3	<b>Cautions on debugging</b> .....	25
3.3	Flash Programming .....	27
3.3.1	<b>Specifications of programming function</b> .....	27
3.3.2	<b>Cautions on flash programming</b> .....	27
<b>CHAPTER 4</b>	<b>HOW TO USE EZ-CUBE WITH 78K0R MICROCONTROLLER</b> .....	<b>28</b>
4.1	Target System Design .....	29
4.1.1	<b>Pin assignment</b> .....	30
4.1.2	<b>Circuit connection example</b> .....	31
4.1.3	<b>Connection of reset pin</b> .....	32
4.2	On-Chip Debugging .....	33
4.2.1	<b>Debug functions</b> .....	33
4.2.2	<b>Securing of user resources and setting of security ID and on-chip debug option byte</b> .....	33
4.2.3	<b>Cautions on debugging</b> .....	37
4.3	Flash Programming .....	40
4.3.1	<b>Specifications of programming function</b> .....	40
4.3.2	<b>Cautions on flash programming</b> .....	40
<b>CHAPTER 5</b>	<b>HOW TO USE EZ-CUBE WITH 78K0 MICROCONTROLLER</b> .....	<b>41</b>
5.1	Target System Design .....	42

5.1.1	Pin assignment .....	42
5.1.2	Circuit connection examples.....	43
5.1.3	Connection of reset pin.....	45
5.1.4	Cautions on Target system Design .....	47
5.1.5	Clock Setting .....	47
5.2	On-Chip Debugging.....	49
5.2.1	Debug functions .....	49
5.2.2	Securing of user resources and setting of security ID .....	49
5.2.3	Cautions on debugging.....	54
5.3	Flash Programming.....	56
5.3.1	Specifications of programming function .....	56
5.3.2	Cautions on flash programming.....	56
<b>CHAPTER 6 HOW TO USE EZ-CUBE WITH RX MICROCONTROLLER.....</b>		<b>57</b>
6.1	Target System Design.....	58
6.1.1	Pin assignment .....	58
6.1.2	Recommended Circuit Connection.....	59
6.1.3	Notes on Connection .....	62
6.2	On-Chip Debugging.....	64
6.2.1	Debug functions .....	64
6.2.2	Notes on debugging.....	65
6.3	Flash Programming .....	66
6.3.1	Specifications of programming function .....	66
6.3.2	Cautions on flash programming .....	66
<b>CHAPTER 7 HOW TO USE EZ-CUBE WITH V850 MICROCONTROLLER .....</b>		<b>67</b>
7.1	Target System Design.....	68
7.1.1	Pin assignment .....	68
7.1.2	Circuit connection examples.....	69
7.1.3	Connection of reset pin.....	70
7.2	On-Chip Debugging.....	72
7.2.1	Debug functions .....	72
7.2.2	Securing of user resources and setting of security ID .....	72
7.2.3	Cautions on debugging.....	75
7.3	Flash Programming.....	78
7.3.1	Specifications of programming function .....	78
7.3.2	Cautions on flash programming .....	78

## CHAPTER 1 OVERVIEW

EZ-CUBE Emulator (hereinafter referred to as EZ-CUBE) is an on-chip debug emulator with flash programming function, which is used for debugging and programming a program to be embedded in on-chip flash memory microcontrollers. This product can debug with the target microcontroller connected to the target system, and can write programs to the on-chip flash memory of microcontrollers.

### 1.1 Features

- On-chip debugging
  - Can debug with the target microcontroller connected to the target system.
- Flash memory programming
  - Can write programs to the on-chip flash memory of microcontrollers.
- USB connection
  - Can be connected to the host machine via USB interface 2.0.
  - Since EZ-CUBE operates on power supplied via USB, an external power supply is unnecessary.
- Variety of supported devices and expandability
  - EZ-CUBE supports a wide variety of Renesas Electronics 8-bit to 32-bit on-chip flash memory microcontrollers.
  - RL78 Microcontrollers
  - 78K0 Microcontrollers
  - 78K0R Microcontrollers
  - RX Microcontrollers
  - V850 Microcontrollers



## 1.2 Notes Before Using EZ-CUBE

Chapters 1 and 2 present an overview and the basic specifications of EZ-CUBE, and the following chapters provide separate descriptions for the target devices and the purpose of use. To utilize this manual effectively, refer to the following table and see the relevant chapter for your target device and purpose of use.

**Table 1-1. Chapters Corresponding to Usage**

Target Device	Purpose of Use	Relevant Chapter
RL78		CHAPTER 3 HOW TO USE EZ-CUBE WITH RL78 MICROCONTROLLER
	Target system design	3.1 Target System Design
	On-chip debugging	3.2 On-Chip Debugging
	Flash memory programming	3.3 Flash Programming
78K0R		CHAPTER 4 HOW TO USE EZ-CUBE WITH 78K0R MICROCONTROLLER
	Target system design	4.1 Target System Design
	On-chip debugging	4.2 On-Chip Debugging
	Flash memory programming	4.3 Flash Programming
78K0		CHAPTER 5 HOW TO USE EZ-CUBE WITH 78K0 MICROCONTROLLER
	Target system design	5.1 Target System Design
	On-chip debugging	5.2 On-Chip Debugging
	Flash memory programming	5.3 Flash Programming
RX		CHAPTER 6 HOW TO USE EZ-CUBE WITH RX MICROCONTROLLER
	Target system design	6.1 Target System Design
	On-chip debugging	6.2 On-Chip Debugging
	Flash memory programming	6.3 Flash Programming
V850		CHAPTER 7 HOW TO USE EZ-CUBE WITH V850 MICROCONTROLLER
	Target system design	7.1 Target System Design
	On-chip debugging	7.2 On-Chip Debugging
	Flash memory programming	7.3 Flash Programming

### 1.3 Hardware Specifications

This section describes the EZ-CUBE hardware specifications.

The specifications related to the on-chip debug and flash memory programming functions are described in the following chapters.

**Table 1-2. Hardware Specifications**

Classification	Item	Specifications
EZ-CUBE	Operating power supply	Supplied via USB interface (5 V)
	Operating environment conditions	Temperature: $\pm 0$ to $+40^{\circ}\text{C}$ Humidity: 40 to 80% RH (no condensation)
	Storage environment conditions	Temperature: $-15$ to $+60^{\circ}\text{C}$ Humidity: 40 to 80% RH (no condensation)
	External dimensions	60 × 36 × 13 mm
	Weight	Approximately 40 g
Host machine interface	Target host machine	IBM PC/AT <sup>TM</sup> compatibles
	Target OS	Windows XP(32bit), Vista, Windows 7, Windows 8
	USB	2.0
	USB cable	1 m
	Current consumption	500 mA max.
Target interface	Target cable length	8-pin cable
	Supply voltage	5.0 V $\pm$ 0.3V (typ.)
	Supply current	100 mA max.
	Voltage range	2.7 to 5.5 V

### 1.4 Firmware Update

**(1) Installation of the USB driver**

This USB driver is required to connect the host machine and EZ-CUBE. Download it from the CD. Please install this USB driver at first.

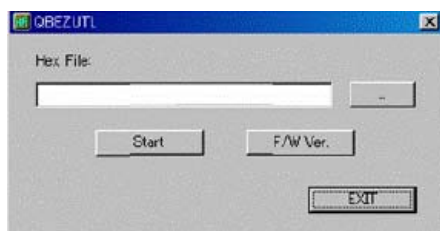
**(2) Connecting the USB cable**

Connect EZ-CUBE to the host machine. **Do not connect EZ-CUBE to the target system.** The mode LED glows red after connection.

**(3) Startup of EZ-CUBE firmware update tool**

Start the EZ-CUBE firmware update tool (QBEZUTL.exe).

QBEZUTL.exe V1.14 or later must be necessary.

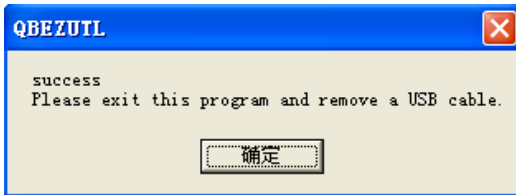


**(4) Select firmware**

Click the [...] button. Select firmware of EZ-CUBE (\*.hex) and click the [OK] button.

**(5) Update firmware**

Click the [Start] button. Start to update the EZ-CUBE firmware. If firmware update is finished, the following dialog box appears.

**(6) Quit EZ-CUBE firmware update tool**

Click the [Exit] button. Quit EZ-CUBE firmware update tool.

**(7) Unplugging the USB cable**

Unplug the USB cable from EZ-CUBE or the host machine.

**1.5 Standard configuration**

Main EZ-CUBE emulator unit (YRCNEZCUBE01)

USB interface cable (A pulg-mini B plug)

Target system interface cable (8-pin)

CD-ROM

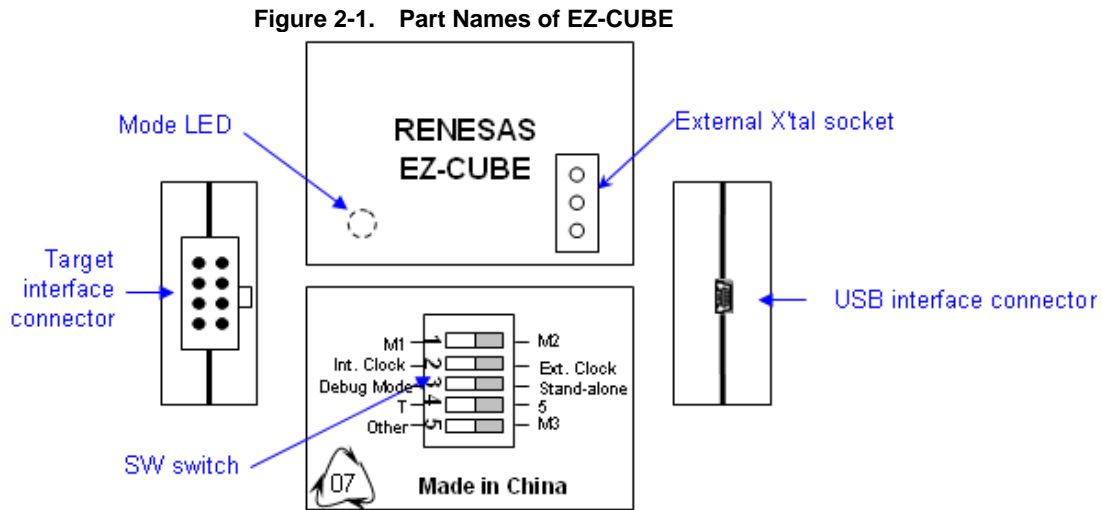
## CHAPTER 2 NAMES AND FUNCTIONS OF HARDWARE

This chapter describes the part names and functions of EZ-CUBE and its accessories.

The part names described in this chapter are used throughout this document. This chapter provides an overview of the various functions. Reading it through, the reader will gain a basic grasp of EZ-CUBE. While reading this chapter, also check if the hardware has a defect.

### 2.1 Part Names and Functions of EZ-CUBE

Figure 2-1 shows the part names of the EZ-CUBE main unit. For their functions, refer to (1) to (9) below.



#### (1) SW-1 switch

The position of this switch depends on the target devices. The detail describes, please see the follow chapter. This switch is set to "M2" at shipment.

#### (2) SW-2 switch

This switch is used to set clock. Table 2-1 describes the setting details. This switch is set to "Int. Clock" at shipment.

**Table 2-1. Setting of Clock Select Switch**

Setting	Description
Int. Clock	When this SW-2 is turned to Int. Clock position, 8MHz fixed freq. is to be supplied to the target board.
Ext. Clock	If the other freq. is required, it should be turned to Ext. Clock position (Ext. X'tal should be connected in this case).

#### (3) SW-3 switch

This switch is used to set run mode of program. Table 2-2 describes the setting details. This switch is set to "Debug Mode" at shipment.

Table 2-2. Setting of Program Run Select Switch

Setting	Description
Debug Mode	On " <b>Debug</b> " position, the user's program will run when RUN command is issued from the debugger GUI.
Stand Alone	On " <b>Stand Alone</b> " position, the user's program is automatically run when the reset is released even the cables from EZ-CUBE is connected to the target board.

**(4) SW-4 switch**

This switch is used to set the power supplied to the target system. Table 2-3 describes the setting details. This switch is set to "5" at shipment.

**Caution** Do not change the switch setting while the USB cable is connected.

Table 2-3. Setting of Power Select Switch

Setting	Description
5	5 V $\pm$ 0.3V is supplied from EZ-CUBE to the target system <sup>Note</sup> . The supplied power is fed back to EZ-CUBE and used only for power detection.
T	Power supply of the target system is used. EZ-CUBE only detects the power for the target system.

**Note** The maximum rating of the current is 100mA, so do not use EZ-CUBE with the target system with the higher current rating. The power is always supplied after EZ-CUBE is connected to the host machine.

**(5) SW-5 switch**

The position of this switch depends on the target devices. The detail describes, please see the follow chapter. This switch is set to "Other" at shipment.

**(6) USB interface connector**

This is a connector used to connect EZ-CUBE with the host machine, via a USB cable.  
A USB 2.0 compliant mini-B connector is employed.

**(7) Target interface connector**

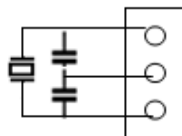
This is a connector used to connect EZ-CUBE with the target system, via a 8-pin (2\*4pin) target cable.

**(8) External X'tal socket(Only for 78K0 and RX)**

SW-2 set for **Ext. Clock**.

Connect an oscillator or oscillation circuit on the External X'tal socket. (Select for "Clock board" in the Configuration dialog box of the debugger.) For the operation this step, refer to the user's manual for CS+.

Figure 2-2. Mounting Diagram



**(9) Mode LED**

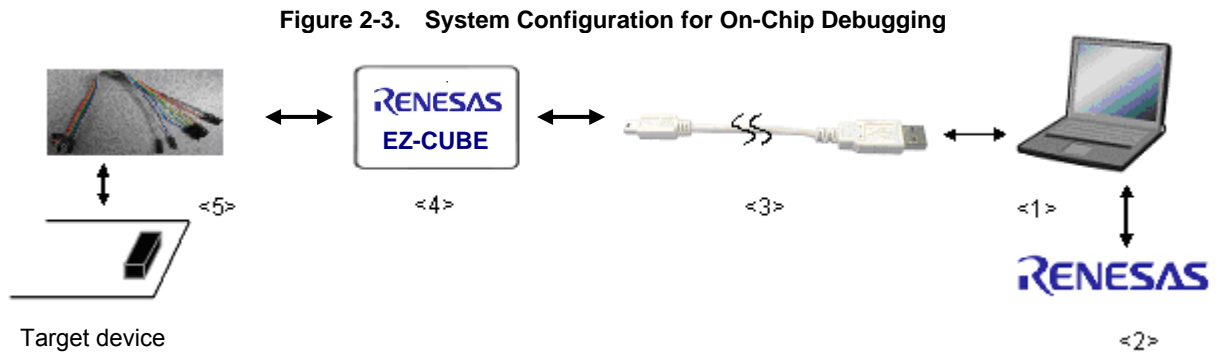
The appearance of the mode LED changes according to the status of hardware and software, as shown in Table 2-4.

**Table 2-4. Mode LED Status**

Mode LED Color	Appearance	Description	
		USB Connection	Software Operation
-	Extinguished	Not connected	Not yet started
<b>Red</b>	Glowing	Connected	Power or the CPU is in the break mode.
<b>Green</b>	Glowing		A debugger has been started

**2.2 System configuration**

Figure 2-3 illustrates the system configuration for on-chip debugging.



- <1> Host machine  
Products with USB ports
- <2> Software  
Includes CS+, firmware update tool, and so on. <3>
- USB cable (accessory)
- <4> EZ-CUBE (this product)
- <5> 8-pin user-system interface cable (accessory)

**2.3 Setup****2.3.1 Installing Emulator Software**

Install the development software (CS+) into the host machine.

CS+ V3.00 or later must be necessary.

**2.3.2 System startup procedure**

Turn the power of the EZ-CUBE emulator and the target system following the procedure below.

(1) Check the power is off

Check that the target system is turned off.

(2) Connect the target system

Connect the emulator and the target system with a user-system interface cable.

(3) Connect the host machine and turn on the emulator

Connect the emulator and the host machine with a USB interface cable. The EZ-CUBE emulator is turned on by connecting the USB interface cable.

(4) Turn on the target system

Turn on the target system. This step is not necessary when power is supplied to the target system from the EZ-CUBE emulator.

(5) Launch the emulator debugger

Launch the emulator debugger. For the operation after this step, refer to the user's manual for CuebSuite+.

If the debugger does not start normally or the operation is unstable, the possible causes may be the following.

Communication error between EZ-CUBE and target system

- Whether firmware is update.
- Whether switch is selected normally.
- Whether communication is performed normally.
- The user resource has not been secured or the security ID has not been set  
To perform debugging with EZ-CUBE, the user resource must be secured and the security ID must be set.
- Unsupported software (debugger, device file, or firmware) is used  
The software used may not support debugging of the target device.
- Defect of EZ-CUBE  
EZ-CUBE may have a defect.

### **2.3.3 System shutdown procedure**

Terminate debugging and shutdown the system in the following order.

If the following order is not observed, the target system or EZ-CUBE may be damaged.

(1) Close the emulator debugger

Close the emulator debugger.

(2) Turn off the target system

Turn off the target system. This step is not necessary when power is supplied to the target system from the EZ-CUBE emulator.

(3) Turn off the emulator and disconnect the emulator.

Disconnect the USB interface cable from the emulator. The EZ-CUBE emulator is turned off by disconnecting from the USB interface cable.

(4) Disconnecting the target system

Disconnect the user-system interface cable from the target system.

## CHAPTER 3 HOW TO USE EZ-CUBE WITH RL78 MICROCONTROLLER

This chapter describes how to use EZ-CUBE when performing on-chip debugging and flash programming for a RL78 microcontroller.

On-chip debugging is a method to debug a microcontroller mounted on the target system, using a debug function implemented in the device. Since debugging is performed with the target device operating on the board, this method is suitable for field debugging.

Flash programming is a method to write a program to the flash memory embedded in a device. Erasing, writing and verifying the program can be performed on-board with the device.

Please update firmware for RL78 at first. Refer to description (1) to (3) on the following order. For detail, refer to **1.4 Firmware update**.

- (1) Connect EZ-CUBE to the host machine. **Do not connect EZ-CUBE to the target system.**
- (2) Start the EZ-CUBE firmware update tool"QBEZUTL.exe". Select firmware of RL78 (RL78G10\_OCD\_FW.hex or RL78\_OCD\_FW (except G10).hex).
- (3) Click the [**Start**] button. Start to update the EZ-CUBE firmware.

Read the following chapters if you are using EZ-CUBE for the first time with a RL78 microcontroller as the target device.

### 3.1 Target System Design

For communication between EZ-CUBE and the target system, communication circuits must be mounted on the target system. This section describes the circuit design and mounting of connectors.

### 3.2 On-Chip Debugging

This section describes the system configuration and startup method to perform on-chip debugging with EZ-CUBE.

### 3.3 Flash Programming

This section describes the system configuration and startup method to perform flash programming with EZ-CUBE.

## Supporting MCU

Table 3-1 shows the supporting MCUs of RL78 EZ-CUBE firmware.

**Table 3-1 Supporting MCUs of RL78 EZ-CUBE firmware**

Items	Contents	Firmware
Supporting MCUs	RL78/G10	RL78G10_OCD_FW.hex
	RL78/G12,G13,G14,I1A	RL78_OCD_FW (except G10).hex
Supporting Operation mode	UART mode	

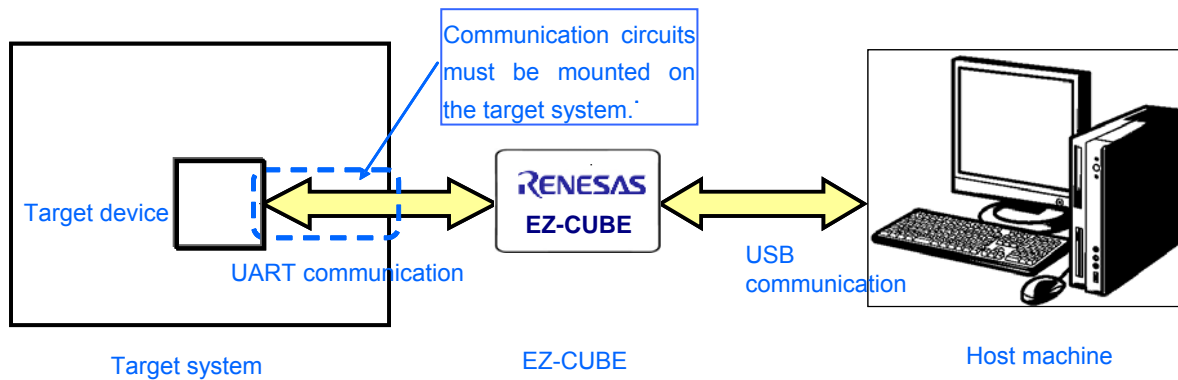


### 3.1 Target System Design

This section describes the target system circuit design required for on-chip debugging and flash programming.

Figure 3-1 presents an overview of the EZ-CUBE communication interface. As shown on the left side of the figure, EZ-CUBE performs serial communication with the target device on the target system. For this communication, communication circuits must be mounted on the target system. Refer to this section to design circuits appropriately.

Figure 3-1. Outline of Communication Interface



#### 3.1.1 Pin assignment

This section describes the interface signals used between EZ-CUBE and the target system. Table 3-2 lists the pin assignment. Table 3-3 describes the functions of each pin.

Table 3-2. Pin Assignment

Pin No.	Pin name <sup>Note</sup>
1	GND
2	RESET_IN
3	Vdd
4	FLMD0
5	CLK
6	RxD.
7	RESET_OUT
8	TOOL0

**Note** Signal names in EZ-CUBE

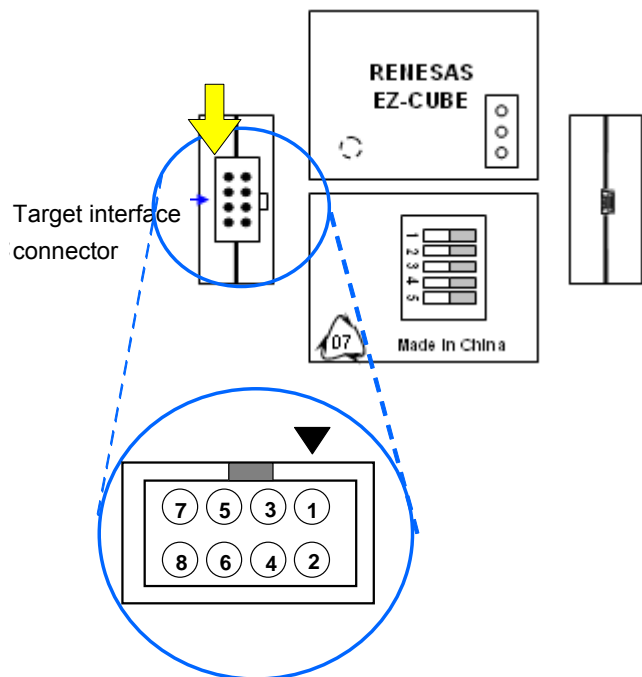


Table 3-3. Pin Functions

Pin Name	IN/OUT <sup>Note</sup>	Description
RESET_IN	IN	Pin used to input reset signal from the target system
RESET_OUT	OUT	Pin used to output reset signal to the target device
FLMD0	OUT	Pin used to set the target device to debug mode or programming mode
RXD	IN/OUT	Pin used to transmit/receive command/data between the target device
TOOL0	IN/OUT	Pin used to transmit/receive command/data between the target device
CLK	IN	Pin used to input handshake signal from the target device

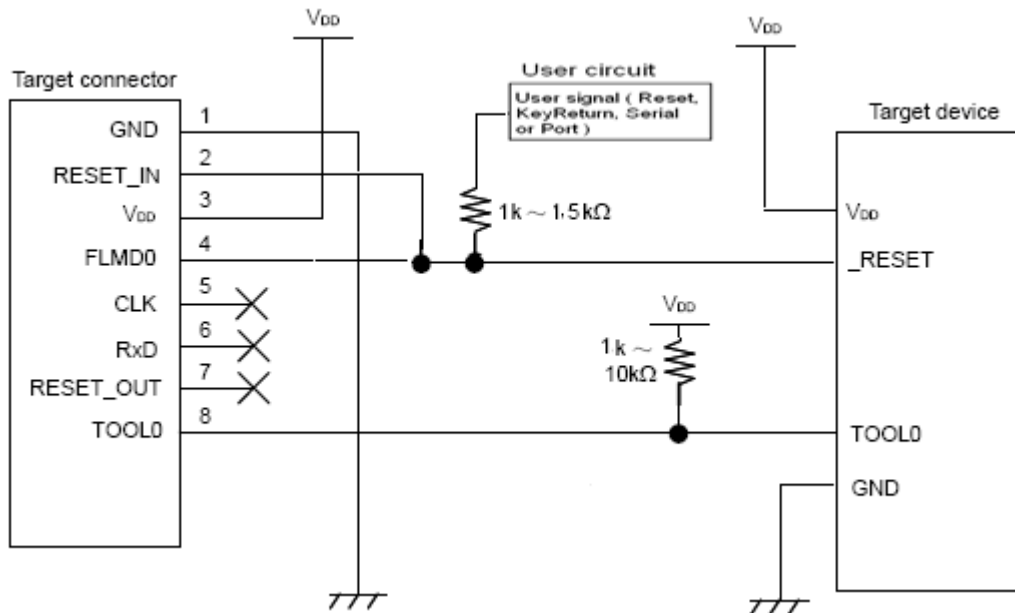
**Note** As seen from EZ-CUBE

3.1.2 Circuit connection example

Refer to Figure 3-2 and Figure 3-3 design an appropriate circuit.

**Caution** The constants described in the circuit connection example are reference values. If you perform flash programming aiming at mass production, thoroughly evaluate whether the specifications of the target device are satisfied.

Figure 3-2. RL78/ G10, G12(20pin,24pin), Recommended Circuit Connection



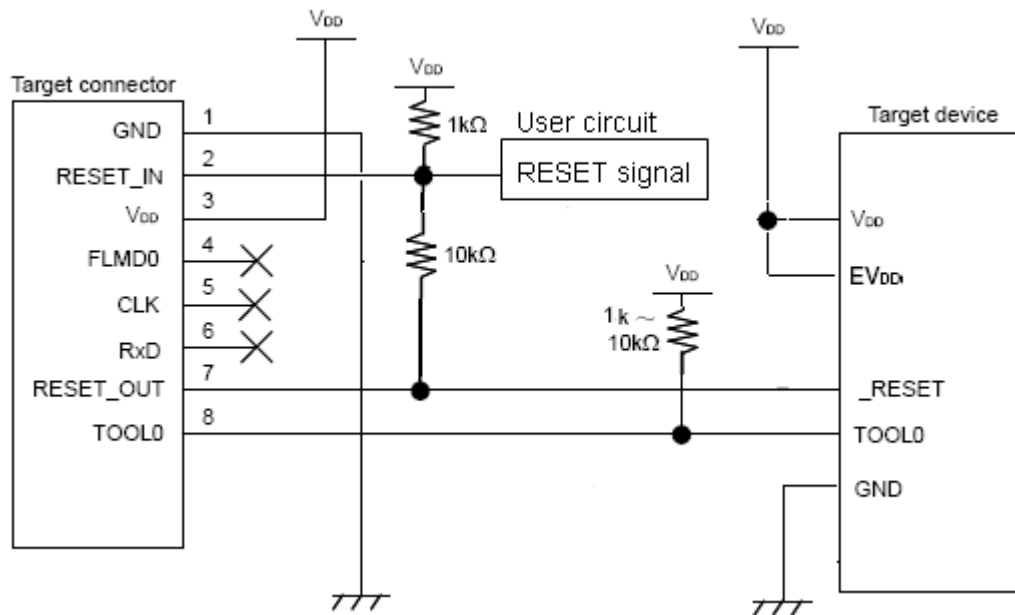
EZ-CUBE Switch Setting

For RL78/ G10, G12(20pin, 24pin)

- SW-1: Select switch to "M2".
- SW-2: Select switch to "Int. Clock".
- SW-3: Select switch to "Debug Mode".
- SW-4: Depend on the target system environment.
- SW-5: Select switch to "M3".

<b>CAUTION</b>	
Notes on the Target System Power Supply:	
	<ol style="list-style-type: none"> <li>1. Do not change the switch setting after connecting USB cable.</li> <li>2. The EZ-CUBE can supply power up to 100 mA current. Do not exceed the maximum supply current. The power is supplied from EZ-CUBE after connecting EZ-CUBE to the host machine.</li> </ol>

Figure 3-3. RL78/G12(30pin), RL78/G13, RL78/G14, RL78/I1A Recommended Circuit Connection



### EZ-CUBE Switch Setting

For RL78/G12(30pin), RL78/G13, RL78/G14, RL78/I1A

- SW-1: Select switch to "M2".
- SW-2: Select switch to "Int. Clock".
- SW-3: Select switch to "Debug Mode".
- SW-4: Depend on the target system environment.
- SW-5: Select switch to "Other".

## ! CAUTION

Notes on the Target System Power Supply:



1. Do not change the switch setting after connecting USB cable.
2. The EZ-CUBE can supply power up to 100 mA current. Do not exceed the maximum supply current. The power is supplied from EZ-CUBE after connecting EZ-CUBE to the host machine.

### 3.1.3 Connection of reset pin

This section describes the connection of the reset pin, for which special attention must be paid, in the circuit connection example shown in the previous section.

During on-chip debugging, a reset signal from the target system is input to EZ-CUBE, masked, and then output to the target device. Therefore, the reset signal connection varies depending on whether EZ-CUBE is connected.

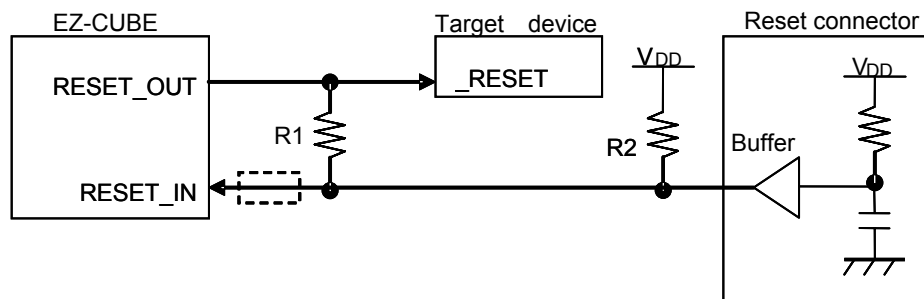
For flash programming, the circuit must be designed so that the reset signals of the target system and EZ-CUBE do not conflict.

**Recommend automatically switching the reset signal via series resistor.**

Figure 3-4 illustrates the reset pin connection described in 3.1.2 **Circuit connection example**.

This connection is designed assuming that the reset circuit on the target system contains an N-ch open-drain buffer (output resistance: 100Ω or less). The VDD or GND level may be unstable when the logic of RESET\_IN/OUT of EZ-CUBE is inverted, so observe the conditions described below in **Remark**.

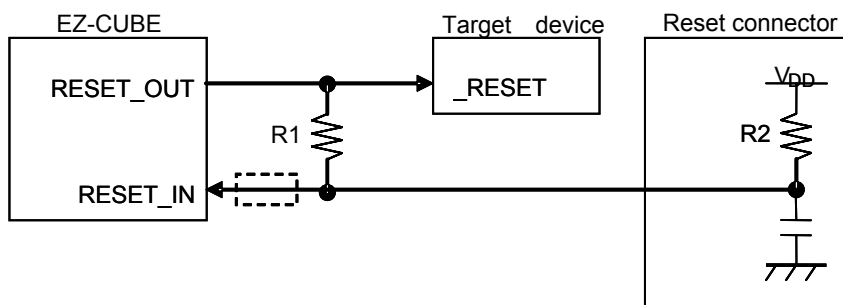
Figure 3-4. Circuit Connection with Reset Circuit That Contains Buffer



**Remark** Make the resistance of at least R1 ten times that of R2, R1 being 10 k $\Omega$  or more.  
 Pull-up resistor R2 is not required if the buffer of the reset circuit consists of CMOS output.  
 The circuit enclosed by a dashed line is not required when only flash programming is performed.

Figure 3-5 illustrates the circuit connection for the case where the reset circuit on the target system contains no buffers and the reset signal is only generated via resistors or capacitors. Design the circuit, observing the conditions described below in **Remark**.

Figure 3-5. Circuit Connection with Reset Circuit That Contains No Buffers



**Remark** Make the resistance of at least R1 ten times that of R2, R1 being 10 k $\Omega$  or more.  
 The circuit enclosed by a dashed line is not required when only flash programming is performed.

## 3.2 On-Chip Debugging

This section describes the system configuration, startup/shutdown procedure and cautions for debugging when on-chip debugging is performed with EZ-CUBE.

### 3.2.1 Debug functions

Table 3-4 lists the debug functions when a RL78 microcontroller is the target device.

**Table 3-4. Debug Functions**

Functions	Specifications
Target device	RL78 RL78/G10,12,G13,G14,11A
Security	10 byte ID code authentication
Download	Available
Execution	Go, Step In, Step Over, CPU Reset, Restart
Hardware break	1 point
Software break	Multiple points
User spaces used for debugging	1-wire mode: Internal ROM: 1024 bytes + 22 bytes, Internal RAM: 6 bytes <sup>Note</sup>
Function pins used for debugging	TOOL0

**Note** For details, refer to **3.2.2 Securing of user resources and setting of security ID and on-chip debug option byte**.

### 3.2.2 Securing of user resources and setting of security ID and on-chip debug option byte debugging resources

The user must prepare the following to perform communication between EZ-CUBE and the target device and implement each debug function. Refer to the descriptions on the following sections and set these items in the user program or using the build tool property.

#### (a) Setting of security ID

This setting is required to prevent the memory from being read by an unauthorized person. Embed a security ID at addresses 0xC4 to 0xCD in the internal flash memory. The debugger starts only when the security ID that is set during debugger startup and the security ID set at addresses 0xC4 to 0xCD match. If the ID codes do not match, the debugger manipulates the target device in accordance with the value set to the on-chip debug option byte area (refer to Table 3-6).

If the user has forgotten the security ID to enable debugging, erase the flash memory and set the security ID again.

[How to set security ID]

A setting method of the security ID is following. When both (1) and (2) methods are done at a time, method (2) has a priority.

- (1) Embed the security ID at addresses 0xC4 to 0xCD in the user program.
- (2) Setting of the security ID by build tool common options. (In case of CS+)

- (1) Embed a security ID at addresses 0xC4 to 0xCD in the user program.

For example If the security ID is embedded as follows, the security ID set by the debugger is

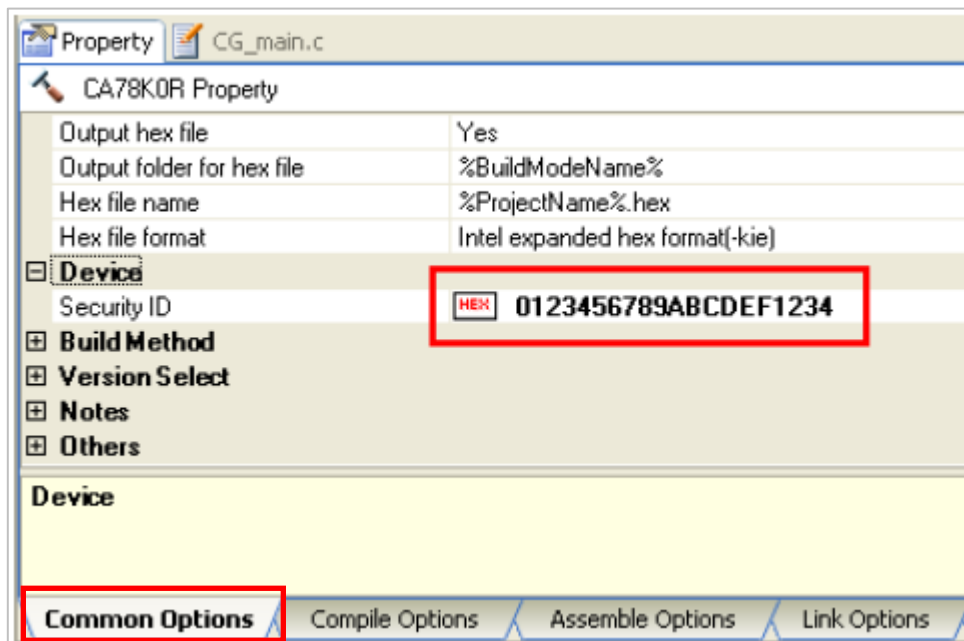
" 0123456789ABCDEF1234 " (not case-sensitive).

Table 3-5 Security ID

Address	Value
0xC4	0x01
0xC5	0x23
0xC6	0x45
0xC7	0x67
0xC8	0x89
0xC9	0xAB
0xCA	0xCD
0xCB	0xEF
0xCC	0x12
0xCD	0x34

(2) Setting of the security ID by build tool common options. (In case of CS+) Set in “device” in the common options tab as figure 3-6.

Figure 3-6. Security ID Setting Example



[How to authenticate the security ID at debugger startup]

When connecting a debugger to the device set the security ID, it is necessary to specify the security ID by connection settings in debug tool property. (Default security ID is set in build tool property.)

**(b) Setting of On-chip debugging option byte**

This is the area for the security setting to prevent the flash memory from being read by an unauthorized person. The debugger manipulates the target device in accordance with the set value, as shown below.

**Table 3-6. On-Chip Debug Option Byte Setting and Operation**

Set Value	Description	Remark
0x04	Debugging is disabled	This setting is available only for flash programming and self programming.
0x85	The on-chip flash memory is not erased no matter how many times the security ID code authentication fails.	-
0x84	All on-chip flash memory areas are erased if the security ID code authentication fails.	-
Other than above	Setting prohibited	-

[How to secure areas]

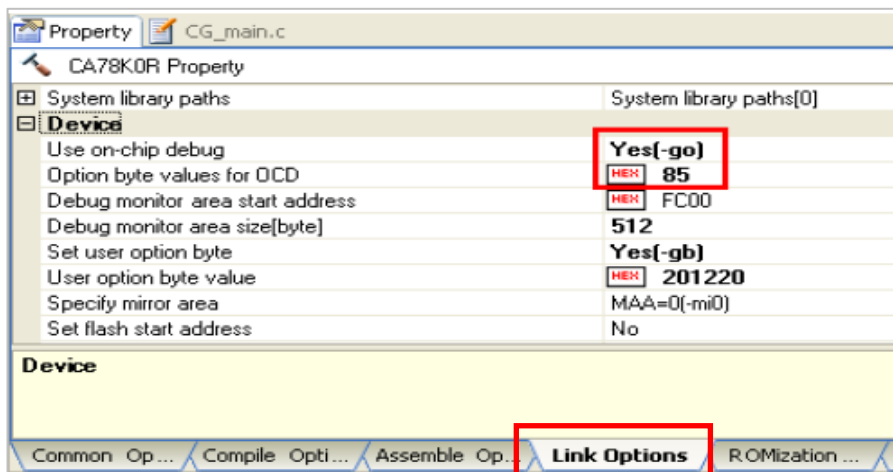
A setting method of On-chip debug option byte is following. When setting each other, priority is (2).

- (1) Embed the On-chip debug option byte at addresses 0xC3 in the user program.
- (2) Set the On-chip debug option byte by build tool link options. (In case of CS+)

(1) Embed the On-chip debug option byte at addresses 0xC3 in the user program  
Embed the On-chip debug option byte at addresses 0xC3 in the user program

(2) Set the On-chip debug option byte by build tool link options. (In case of CS+) Set in “device” in the link options tab as figure 3-7.

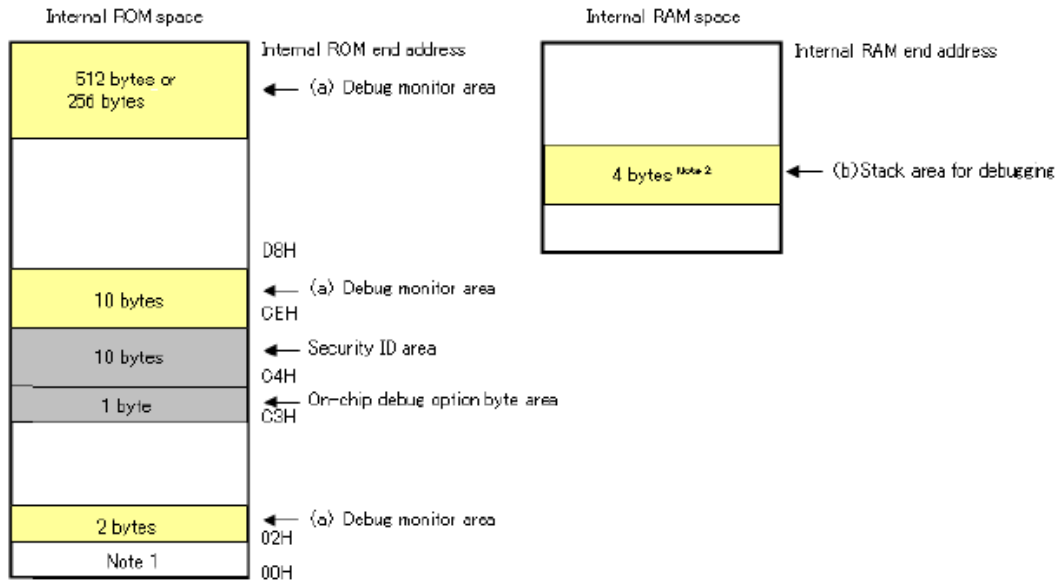
**Example** Setting 0x85 for control value

**Figure 3-7. On-Chip Debug Option Byte Setting Example**

**(C) Securing of area for debugging**

The yellow portions in Figure 3-8 are the areas reserved for placing the debug monitor program, so user programs or data cannot be allocated in these spaces. These spaces must be secured so as not to be used by the user program. Moreover, this area must not be rewritten by the user program. Secure the resources for debugging with the contents explained by (1) and (2).

**Figure 3-8. Memory Spaces Where Debug Monitor Programs Are Allocated**



- Note** 1. In debugging, reset vector is rewritten to address allocated to a monitor program.
- 2. When the self programming is executed, it will be 12 bytes.

**(1) Securing of debug monitor area**

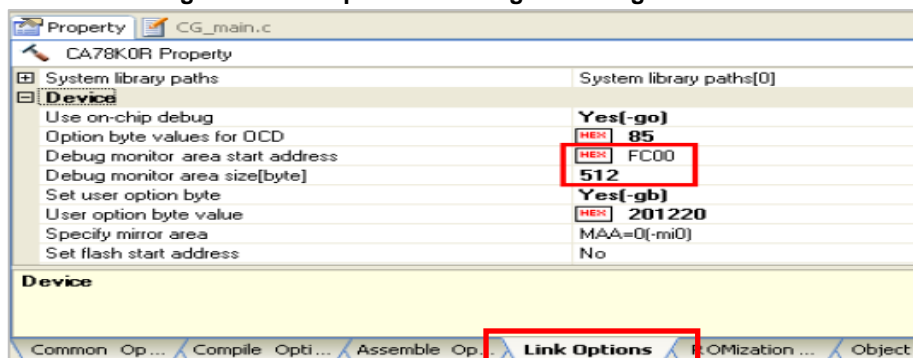
This is the area to which the debug monitor program is to be allocated. The monitor program performs initialization processing for debug communication interface and RUN or break processing for the CPU. This user programs or data must not be placed in an area of 22 bytes near the on-chip debug option byte, and an area of 1024 bytes before the internal ROM end address. In addition, reset vector is rewritten to address allocated to a monitor program.

[How to secure areas]

It is not necessarily required to secure this area if the user program does not use this area.

However To avoid problems that may occur during the debugger startup, it is recommended to secure this area in advance, using the compiler. Figure 3-9 shows example for securing the area, using the CS+. Set in “device” in link options tab as figure 3-9.

**Figure 3-9 Example for securing the debug monitor area**



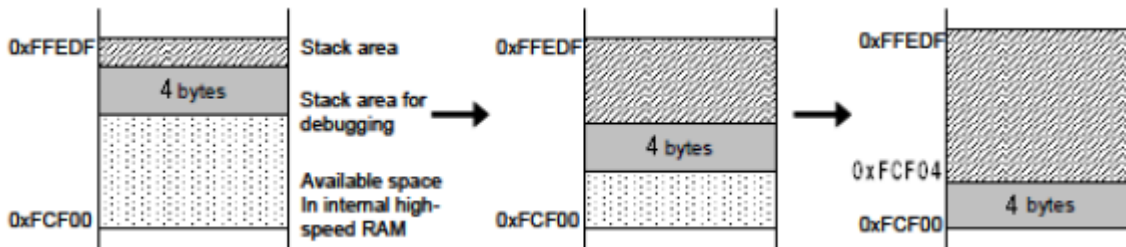


**(2) Securing of stack area for debugging**

This area requires 4 bytes as the stack area for debugging. Since this area is allocated immediately before the stack area, the address of this area varies depending on the stack increase and decrease. That is, 4 extra bytes are consumed for the stack area used.

Figure 3-10 illustrates the case where the stack area is increased when the internal high-speed RAM starts from 0xFCF00.

**Figure 3-10. Variation of Address of Stack Area for Debugging**



[How to secure areas]

Set the stack pointer by estimating the stack area consumed by the user program + 4 bytes. Make sure that the stack pointer does not extend beyond the internal high-speed RAM start address.

**Remark** Refer to the self programming manual for how to secure the stack area for self programming.

### 3.2.3 Cautions on debugging

This section describes cautions on performing on-chip debugging for a RL78 microcontroller.

Be sure to read the following to use EZ-CUBE properly.

#### (1) Handling of device that was used for debugging

Do not mount a device that was used for debugging on a mass-produced product, because the flash memory was rewritten during debugging and the number of rewrites of the flash memory cannot be guaranteed. Moreover, do not embed the debug monitor program into mass-produced products.

#### (2) Flash self programming

If a space where the debug monitor program is allocated is rewritten by flash self programming, the debugger can no longer operate normally. This caution also applies to boot swapping for such an area.

#### (3) Operation after reset

After an external pin reset or internal reset, the monitor program performs debug initialization processing. Consequently, the time from reset occurrence until user program execution differs from that in the actual device operation.

#### (4) Checking operation of a device after debugging

After downloading a load module file to the device to for on-chip debugging, do not check the operation of this device without EZ-CUBE.

A device after debugging contains the specific program for on-chip debugging, so it is different from actual operation.

#### (5) Current consumption when On-chip debugging

On-chip debugging circuit in the device operates during on-chip debugging. Therefore current consumption of the device increases.

When evaluations current consumption of device, please do not connect a debugger.

#### (6) On-chip debugging option byte setting (address C3H)

The on-chip debugging option byte setting is rewritten arbitrarily by the debugger.

#### (7) Operation at voltage with which flash memory cannot be written

If the following debugger operations are executed at voltage with which flash memory cannot be written, the debugger outputs an error and the operation is ignored. Because these operations are included flash memory rewriting.

- <1> Writing to internal flash memory
- <2> Setting or canceling of software breakpoint
- <3> Starting execution at the set software breakpoint position
- <4> Step execution at the set software breakpoint position
- <5> Step-over execution, Return Out execution
- <6> Come Here
- <7> Setting, changing, or canceling of hardware breaks
- <8> Masking/unmasking of internal reset
- <9> Switching of peripheral breaks

**(8) Relation between Standby function and Break function**

The break is interrupt function of CPU. The standby mode is released by the break for using the following debug function.

<1> Stops execution of the user program.

<2> Step execution of the standby instruction (Stops user program after execution instruction)

<3> Pseudo real-time RAM monitor function (Break When Readout)

<4> Pseudo Dynamic Memory Modification (Break When Write)

<5> Breakpoint setting executing of the user program.

**(9) Cautions on using step-in (step execution)**

The value of some SFRs (special function registers) might remain unchanged while stepping into code. If the value of the SFRs does not change while stepping into code, operate the microcontroller by continuously executing the instructions instead of executing them in steps.

Stepping into code: Instructions in the user-created program are executed one by one.

Continuous execution: The user-created program is executed from the current PC value.

**(10) Emulation of flash memory CRC accumulator function**

Please check the operation of high-speed CRC by using IECUBE or using device without EZ-CUBE.

### 3.3 Flash Programming

This section describes the system configuration and startup/shutdown procedure when flash programming is performed for a RL78 microcontroller, using EZ-CUBE.

#### 3.3.1 Specifications of programming function

**Table 3-7. Specifications of Programming Function**

Functions	Specifications
Host interface	USB 2.0
Target interface	UART (1-wire mode)
Target system voltage	2.7 to 5.5 V (depends on the target device)
Clock supply	Internal high-speed oscillation clock is used
Power supply	5 V $\pm$ 0.3 V (maximum current rating: 100 mA)
Acquisition of device-specific information	Parameter file for Renesas Electronics is used
Programming software	RFP <sup>Note</sup> (Renesas Flash Programmer)
Security flag setting	Available
Standalone operation	Unavailable (must be connected to host machine)

Note For detailed usage of the RFP, refer to the **RFP User's Manual**.

#### 3.3.2 Cautions on flash programming

This section describes the cautions for flash programming. Be sure to read the following for the proper use of EZ-CUBE.

- To improve the writing quality, fully understand, verify, and evaluate the following items before using EZ-CUBE.
  - Circuits are designed as described in the user's manuals for the device and EZ-CUBE.
  - The device, RFP and EZ-CUBE are used as described in each user's manual.
  - The power supplied to the target system is stable.

## CHAPTER 4 HOW TO USE EZ-CUBE WITH 78K0R MICROCONTROLLER

This chapter describes how to use EZ-CUBE when performing on-chip debugging and flash programming for a 78K0R microcontroller.

On-chip debugging is a method to debug a microcontroller mounted on the target system, using a debug function implemented in the device. Since debugging is performed with the target device operating on the board, this method is suitable for field debugging.

Flash programming is a method to write a program to the flash memory embedded in a device. Erasing, writing and verifying the program can be performed on-board with the device.

Please update firmware for 78K0R at first. Refer to description (1) to (3) on the following order. For detail, refer to **1.4 Firmware update**.

- (4) Connect EZ-CUBE to the host machine. **Do not connect EZ-CUBE to the target system.**
- (5) Start the EZ-CUBE firmware update tool"QBEZUTL.exe". Select firmware of 78K0R (78K0R\_OCD\_FW.hex).
- (6) Click the **[Start]** button. Start to update the EZ-CUBE firmware.

Read the following chapters if you are using EZ-CUBE for the first time with a 78K0R microcontroller as the target device.

### 4.1 Target System Design

For communication between EZ-CUBE and the target system, communication circuits must be mounted on the target system. This section describes the circuit design and mounting of connectors.

### 4.2 On-Chip Debugging

This section describes the system configuration and startup method to perform on-chip debugging with EZ-CUBE.

### 4.3 Flash Programming

This section describes the system configuration and startup method to perform flash programming with EZ-CUBE.

## Supporting MCU

Table 4-1 shows the supporting MCUs of 78K0R EZ-CUBE firmware.

**Table 4-1 Supporting MCUs of 78K0R EZ-CUBE firmware**

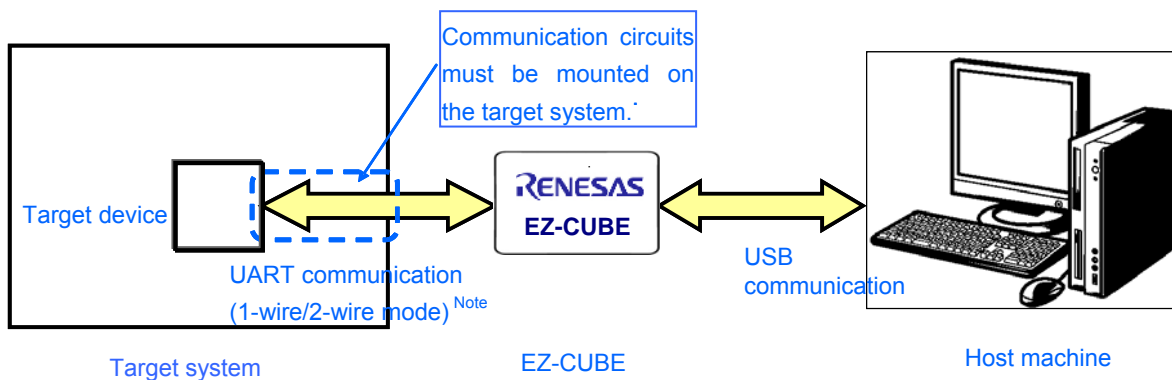
Items	Contents	Firmware
Supporting MCUs	78K0R/Kx3,Lx3	78K0R_OCD_FW.hex
Supporting Operation mode	UART mode	

### 4.1 Target System Design

This section describes the target system circuit design required for on-chip debugging and flash programming.

Figure 4-1 presents an overview of the EZ-CUBE communication interface. As shown on the left side of the figure, EZ-CUBE performs serial communication with the target device on the target system. For this communication, communication circuits must be mounted on the target system. Refer to this section to design circuits appropriately.

Figure 4-1. Outline of Communication Interface



**Note**      1-wire mode: Single-wire UART communication using TOOL0 pin  
 2-wire mode: Single-wire UART communication using TOOL0 and TOOL1 pins

Table 4-2. Differences Between 1-Wire Mode and 2-Wire Mode

Communication Mode	Flash Programming Function	Debugging Function
1-wire mode	Available	<ul style="list-style-type: none"> <li>User resources secured for debugging</li> <li>Internal ROM: 1036 bytes</li> <li>Internal RAM: 6 bytes (stack)</li> </ul>
2-wire mode	Available	<ul style="list-style-type: none"> <li>User resources secured for debugging</li> <li>Internal ROM: 100 bytes</li> <li>Internal RAM: 6 bytes (stack)</li> </ul>

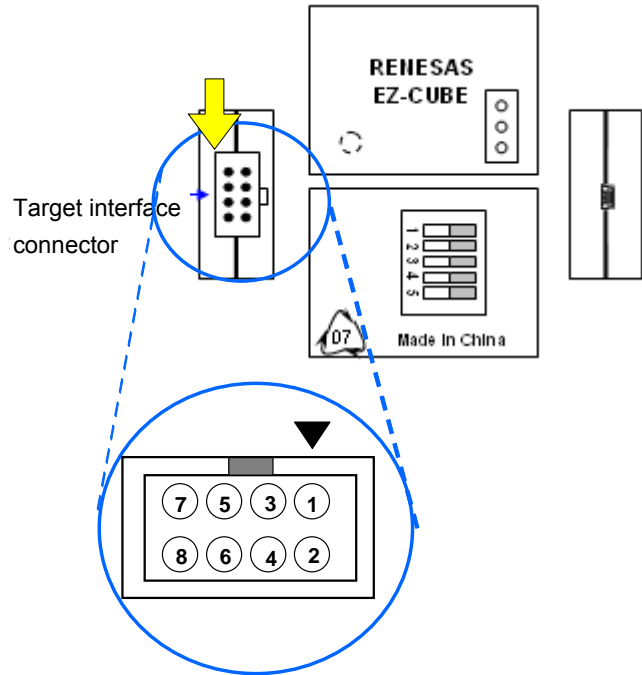
### 4.1.1 Pin assignment

This section describes the interface signals used between EZ-CUBE and the target system. Table 4-3 lists the pin assignment. Table 4-4 describes the functions of each pin. The pin assignment varies between 1-wire and 2-wire modes, so design the circuit appropriately according to the circuit connection examples described on the following sections.

**Table 4-3. Pin Assignment**

Pin No.	Pin name <sup>Note</sup>
1	GND
2	RESET_IN
3	Vdd
4	FLMD0
5	CLK
6	RxD.
7	RESET_OUT
8	TxD

**Note** Signal names in EZ-CUBE



**Table 4-4. Pin Functions**

Pin Name	IN/OUT <sup>Note 1</sup>	Description
RESET_IN	IN	Pin used to input reset signal from the target system
RESET_OUT	OUT	Pin used to output reset signal to the target device
FLMD0	OUT	Pin used to set the target device to debug mode or programming mode
RXD	IN/OUT	Pin used to transmit/receive command/data between the target device
TXD	IN/OUT	Pin used to transmit/receive command/data between the target device
CLK	IN	Pin used to input handshake signal from the target device

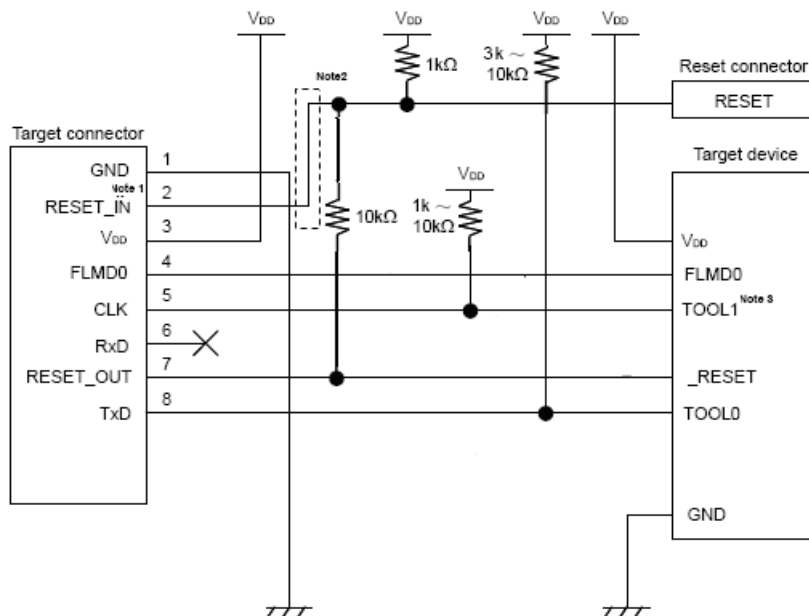
**Note** As seen from EZ-CUBE

### 4.1.2 Circuit connection example

Refer to Figure 4-2 and design an appropriate circuit.

**Caution** The constants described in the circuit connection example are reference values. If you perform flash programming aiming at mass production, thoroughly evaluate whether the specifications of the target device are satisfied.

Figure 4-2. Recommended Circuit Connection



- Notes**
1. This connection is designed assuming that the RESET signal is output from the N-ch open-drain buffer (output resistance: 100Ω or less). For details, refer to **4.1.3 Connection of reset pin**
  2. The circuit enclosed by a dashed line is not required when only flash programming is performed.
  3. This connection is required for 2-wire communication, but not for 1-wire communication. This pin is left open when EZ-CUBE is not connected, so connect a pull-up or pull-down resistor to this pin before using.

### EZ-CUBE Switch Setting

- SW-1: Select switch to "M2".
- SW-2: Select switch to "Int. Clock".
- SW-3: Select switch to "Debug Mode".
- SW-4: Depend on the target system environment.
- SW-5: Select switch to "Other".

### ! CAUTION

Notes on the Target System Power Supply:

1. Do not change the switch setting after connecting USB cable.
2. The EZ-CUBE can supply power up to 100 mA current. Do not exceed the maximum supply current.

The power is supplied from EZ-CUBE after connecting EZ-CUBE to the host machine.



### 4.1.3 Connection of reset pin

This section describes the connection of the reset pin, for which special attention must be paid, in the circuit connection example shown in the previous section.

During on-chip debugging, a reset signal from the target system is input to EZ-CUBE, masked, and then output to the target device. Therefore, the reset signal connection varies depending on whether EZ-CUBE is connected.

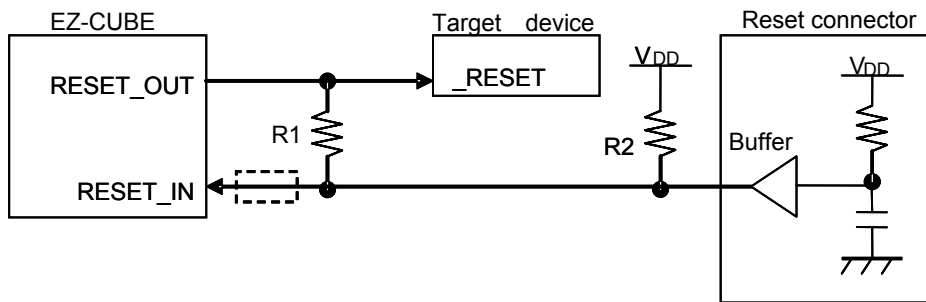
For flash programming, the circuit must be designed so that the reset signals of the target system and EZ-CUBE do not conflict.

**Recommend automatically switching the reset signal via series resistor.**

Figure 4-3 illustrates the reset pin connection described in 4.1.2 **Circuit connection example**.

This connection is designed assuming that the reset circuit on the target system contains an N-ch open-drain buffer (output resistance: 100Ω or less). The VDD or GND level may be unstable when the logic of RESET\_IN/OUT of EZ-CUBE is inverted, so observe the conditions described below in **Remark**.

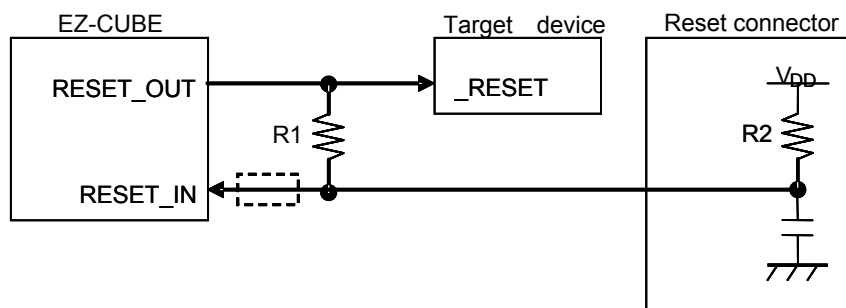
**Figure 4-3. Circuit Connection with Reset Circuit That Contains Buffer**



**Remark** Make the resistance of at least R1 ten times that of R2, R1 being 10 kΩ or more.  
Pull-up resistor R2 is not required if the buffer of the reset circuit consists of CMOS output.  
The circuit enclosed by a dashed line is not required when only flash programming is performed.

Figure 4-4 illustrates the circuit connection for the case where the reset circuit on the target system contains no buffers and the reset signal is only generated via resistors or capacitors. Design the circuit, observing the conditions described below in **Remark**.

**Figure 4-4. Circuit Connection with Reset Circuit That Contains No Buffers**



**Remark** Make the resistance of at least R1 ten times that of R2, R1 being 10 kΩ or more.  
The circuit enclosed by a dashed line is not required when only flash programming is performed.

## 4.2 On-Chip Debugging

This section describes the system configuration, startup/shutdown procedure and cautions for debugging when on-chip debugging is performed with EZ-CUBE.

### 4.2.1 Debug functions

Table 4-5 lists the debug functions when a 78K0R microcontroller is the target device.

**Table 4-5. Debug Functions**

Functions	Specifications
Target device	78K0R 78K0R/Kx3,Lx3
Security	10 byte ID code authentication
Download	Available
Execution	Go, Step In, Step Over, CPU Reset, Restart
Hardware break	1 point (commonly used by execution and access)
Software break	Multiple points
User spaces used for debugging	1-wire mode: Internal ROM: 1036 bytes, Internal RAM: 6 bytes <sup>Note</sup> 2-wire mode: Internal ROM: 100 bytes, Internal RAM: 6 bytes <sup>Note</sup>
Function pins used for debugging	1-wire mode: TOOL0 2-wire mode: TOOL0, TOOL1

**Note** For details, refer to 4.2.2 Securing of user resources and setting of security ID and on-chip debug option byte.

### 4.2.2 Securing of user resources and setting of security ID and on-chip debug option byte

The user must prepare the following to perform communication between EZ-CUBE and the target device and implement each debug function. Refer to the descriptions on the following sections and set these items in the user program or using the build tool property.

#### (a) Setting of Security ID

This setting is to prevent the flash memory from being read by an unauthorized person. Embed a security ID at addresses 0xC4 to 0xCD in the internal flash memory. The debugger starts only when the security ID that is set during debugger startup and the security ID set at addresses 0xC4 to 0xCD match. If the ID codes do not match, the debugger manipulates the target device in accordance with the value set to the on-chip debug option byte area (refer to Table 4-7).

If the user has forgotten the security ID to enable debugging, erase the flash memory and set the security ID again.

[How to set]

A setting method of the security ID is following. When both (1) and (2) methods are done at a time, method (2) has a priority.

(1) Embed the security ID at addresses 0xC4 to 0xCD in the user program.

(2) Setting of the security ID by build tool common options. (In case of CS+)

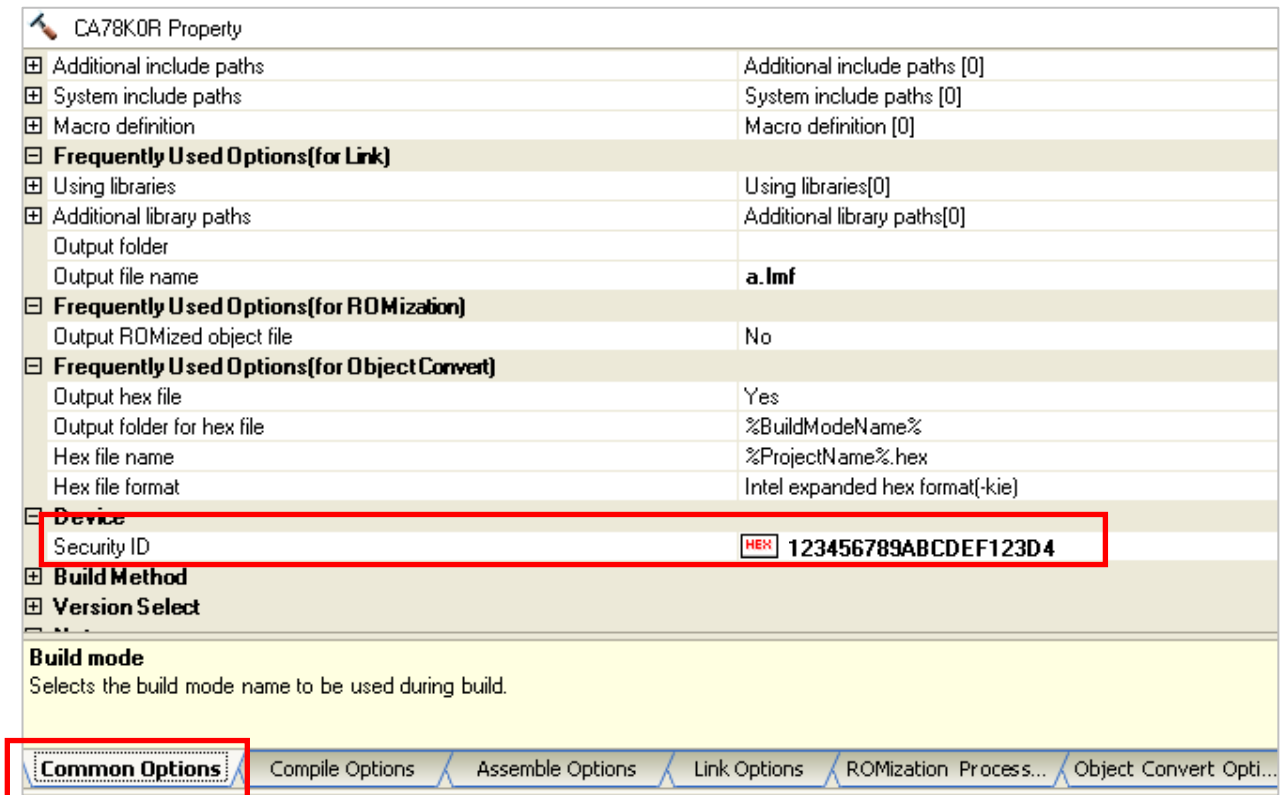
**(1) Embed a security ID at addresses 0xC4 to 0xCD in the user program.**

For example If the security ID is embedded as follows, the security ID set by the debugger is "0123456789ABCDEF1234" (not case-sensitive).

**Table 4-6 Security ID**

Address	Value
0xC4	0x01
0xC5	0x23
0xC6	0x45
0xC7	0x67
0xC8	0x89
0xC9	0xAB
0xCA	0xCD
0xCB	0xEF
0xCC	0x12
0xCD	0x34

**(2) Setting of the security ID by build tool common options. (In case of CS+)** Set in "device" in the common options tab as figure 4-5.

**Figure 4-5. Security ID Setting Example**

**(b)Setting of On-chip debug option byte area**

This is the area for the security setting to prevent the flash memory from being read by an unauthorized person. The debugger manipulates the target device in accordance with the set value, as shown below.

**Table 4-7. On-Chip Debug Option Byte Setting and Operation**

Set Value	Description	Remark
0x04	Debugging is disabled	This setting is available only for flash programming and self programming.
0x85	The on-chip flash memory is not erased no matter how many times the security ID code authentication fails.	-
0x84	All on-chip flash memory areas are erased if the security ID code authentication fails.	-
Other than above	Setting prohibited	-

[How to set]

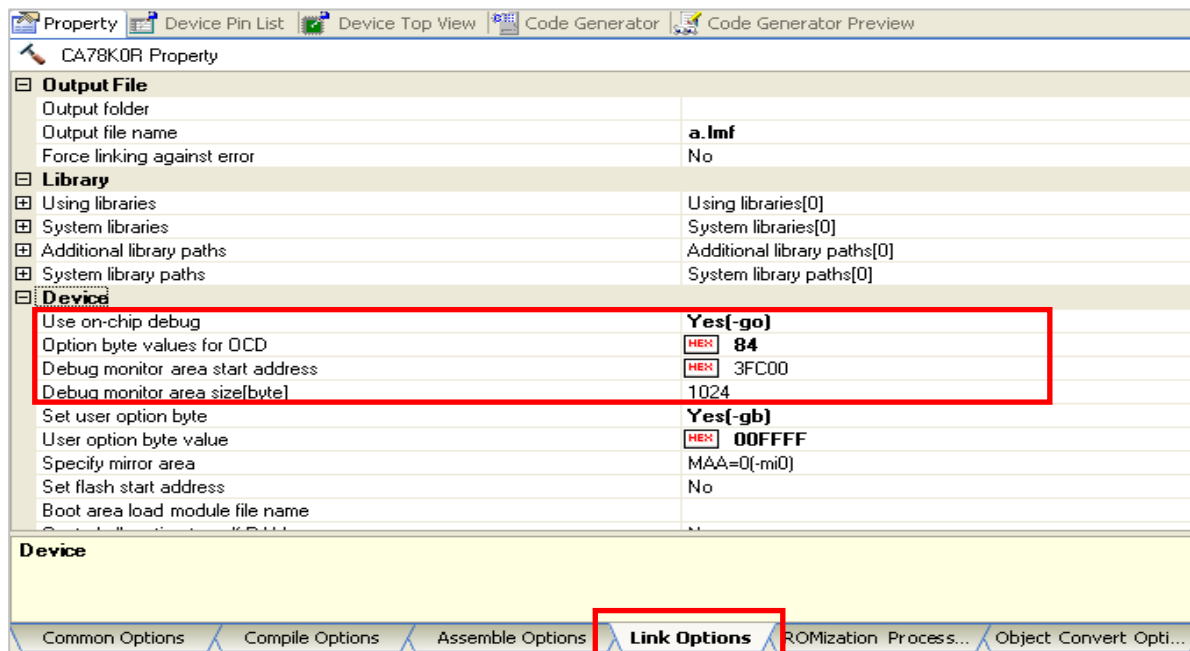
A setting method of On-chip debug option byte is following. When setting each other, priority is (2).

- (1) Embed the On-chip debug option byte at addresses 0xC3 in the user program.
- (2) Set the On-chip debug option byte by build tool link options. (In case of CS+)

(1) Embed the On-chip debug option byte at addresses 0xC3 in the user program  
Embed the On-chip debug option byte at addresses 0xC3 in the user program

(2) Set the On-chip debug option byte by build tool link options. (In case of CS+) Set in “device” in the link options tab as figure 4-6. **Example** Setting 0x84 for control value

**Figure 4-6. On-Chip Debug Option Byte and Monitor Area Setting Example**



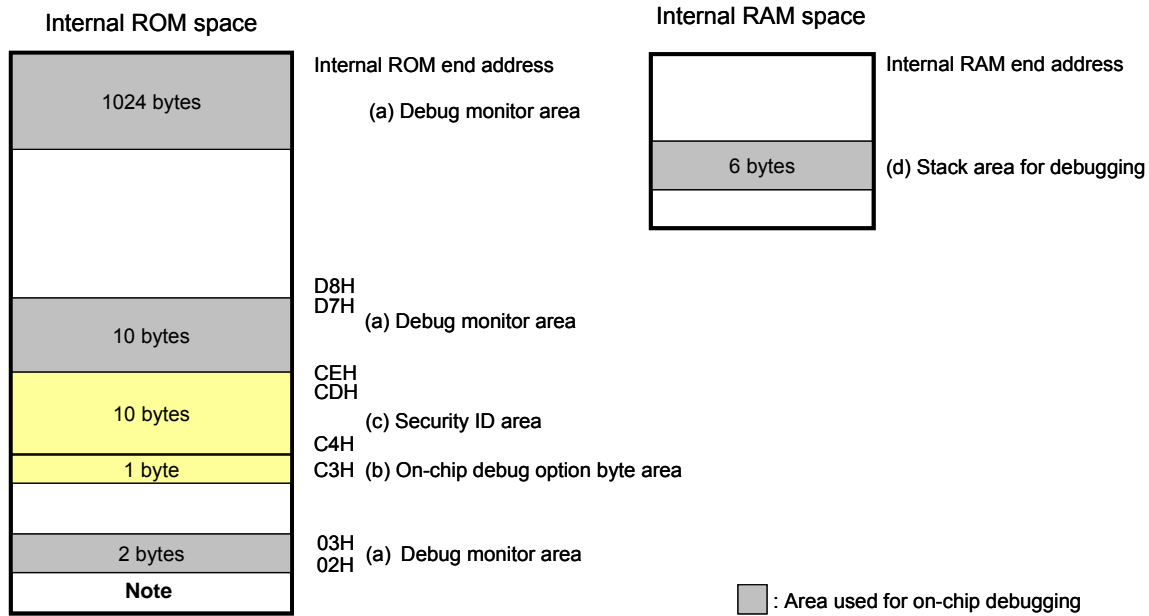
**(C) Securing of stack area for debugging**

The yellow portions in Figure 4-7 are the areas reserved for placing the debug monitor program, so user programs or data cannot be allocated in these spaces. These spaces must be secured so as not to be used by the user program.

Moreover, this area must not be rewritten by the user program.

Secure the resources for debugging with the contents explained by (1) and (2).

**Figure 4-7. Memory Spaces Where Debug Monitor Programs Are Allocated**



**Note** In debugging, reset vector is rewritten to address allocated to a monitor program.

**(1) Securing of debug monitor area**

This is the area to which the debug monitor program is to be allocated. The monitor program performs initialization processing for debug communication interface and RUN or break processing for the CPU. This user programs or data must not be placed in an area of 22 bytes near the on-chip debug option byte, and an area of 1,024 bytes before the internal ROM end address. In addition, reset vector is rewritten to address allocated to a monitor program.

[How to secure areas]

It is not necessarily required to secure this area if the user program does not use this area. However To avoid problems that may occur during the debugger startup, it is recommended to secure this area in advance, using the compiler. Figure 4-6 shows example for securing the area, using the CS+. Set in "device" in link options tab as figure 4-6.

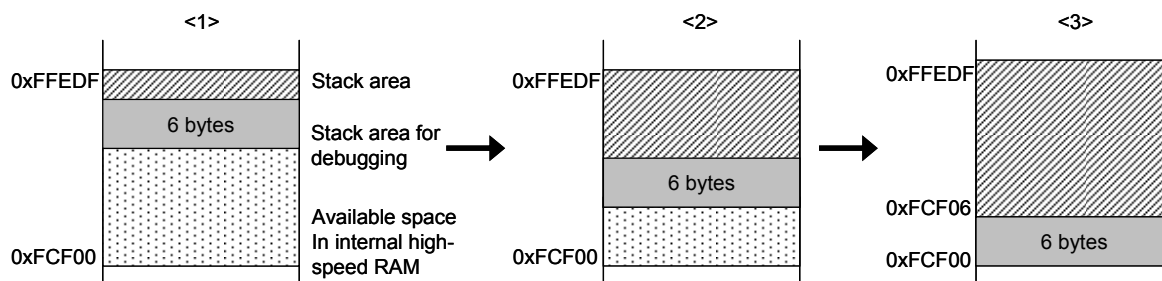
**(2) Securing of stack area for debugging**

This area requires 6 bytes as the stack area for debugging<sup>Note</sup>. Since this area is allocated immediately before the stack area, the address of this area varies depending on the stack increase and decrease. That is, 6 extra bytes are consumed for the stack area used.

Figure 4-8 illustrates the case where the stack area is increased when the internal high-speed RAM starts from 0xFCF00.

**Note** When the self programming is executed, it will be 12 bytes.

Figure 4-8. Variation of Address of Stack Area for Debugging



[How to secure areas]

Set the stack pointer by estimating the stack area consumed by the user program + 6 bytes. Make sure that the stack pointer does not extend beyond the internal high-speed RAM start address.

**Remark** Refer to the self programming manual for how to secure the stack area for self programming.

#### 4.2.3 Cautions on debugging

This section describes cautions on performing on-chip debugging for a 78K0R microcontroller.

Be sure to read the following to use EZ-CUBE properly.

##### (1) Handling of device that was used for debugging

Do not mount a device that was used for debugging on a mass-produced product, because the flash memory was rewritten during debugging and the number of rewrites of the flash memory cannot be guaranteed. Moreover, do not embed the debug monitor program into mass-produced products.

##### (2) Flash self programming

If a space where the debug monitor program is allocated is rewritten by flash self programming, the debugger can no longer operate normally. This caution also applies to boot swapping for such an area.

##### (3) Operation after reset

After an external pin reset or internal reset, the monitor program performs debug initialization processing. Consequently, the time from reset occurrence until user program execution differs from that in the actual device operation.

##### (4) Debugging with real machine running without using EZ-CUBE

If debugging is performed with a real machine running, without using EZ-CUBE, write the user program using the RFP. Programs downloaded by the debugger include the monitor program, and such a program malfunctions if it includes processing to make the TOOL0 pin low level.

##### (5) Operation when debugger starts

When the debugger is started, if the Target Device Connection setting in the Configuration dialog box of the debugger is different from the setting for the previous debugging, the internal flash memory is erased.

##### (6) Debugging after program is written by flash programming

If a program is written to the internal flash memory using the RFP, on-chip debugging is disabled even if it is enabled in the on-chip debugging option byte setting. To perform debugging of the target device after that, erase the internal flash memory using the RFP and then download the program using the debugger.

**(7) LVI default start function setting (address C1H)**

The LVI setting at address C1H in the internal flash memory during debugging is set as follows.

- When EZ-CUBE is connected: The LVI default start function is available.
- When EZ-CUBE is not connected: The LVI default start function is unavailable.

**(8) On-chip debugging option byte setting (address C3H)**

The on-chip debugging option byte setting is rewritten arbitrarily by the debugger.

**(9) Operation at voltage with which flash memory cannot be written**

If any of the following debugger operations <1> to <7>, which involve flash memory rewriting, is performed while flash memory cannot be rewritten, the debugger automatically changes the register setting so as to enable flash memory rewriting, and restores the register setting after the operation is completed. If any of the following operations <1> to <7> is performed while flash memory rewriting has been disabled or operation is performed at a voltage with which flash memory cannot be rewritten, however, the debugger outputs an error and the operation is ignored.

To prevent the flash memory from being rewritten, select "No" in permit flash programming in property of debug tool. To prevent the frequency from being switched automatically, select "User" in the Monitor clock in property of debug tool.

<1> Writing to internal flash memory

<2> Setting or canceling of software breakpoint

<3> Starting execution at the set software breakpoint position

<4> Step execution at the set software breakpoint position

<5> Step-over execution, Return Out execution

<6> Come Here

<7> If "Yes" is selected in Permit flash programming in property of debug tool, the following operations cannot be performed.

- a) Setting, changing, or canceling of hardware breaks
- b) Masking/unmasking of internal reset
- c) Switching of peripheral breaks

**(10) Debugging in 1-wire mode**

Note the following points when debugging is performed in 1-wire mode (selected by choosing TOOL0 in the Connection with Target Board area in the Communication method dialog box of the debugger).

When the internal high-speed oscillator is used for the CPU operating clock, breaks may not occur normally if the frequency variation between debugger startup and break occurrence (except for when changing the register) is too large. This situation may occur when the variation of operating voltage or temperature is too large.

**(11) Relation between Standby function and Break function**

The break is interrupt function of CPU. The standby mode is released by the break for using the following debug function.

- Stops execution of the user program.
- Step execution of the standby instruction (Stops user program after execution instruction)
- Pseudo real-time RAM monitor function (Break When Readout)

- Pseudo Dynamic Memory Modification (Break When Write)
- Breakpoint setting executing of the user program.

**(12)Cautions on using step-in (step execution)**

The value of some SFRs (special function registers) might remain unchanged while stepping into code. If the value of the SFRs does not change while stepping into code, operate the microcontroller by continuously executing the instructions instead of executing them in steps.

Stepping into code: Instructions in the user-created program are executed one by one.

Continuous execution: The user-created program is executed from the current PC value.



### 4.3 Flash Programming

This section describes the system configuration and startup/shutdown procedure when flash programming is performed for a 78K0R microcontroller, using EZ-CUBE.

#### 4.3.1 Specifications of programming function

**Table 4-8. Specifications of Programming Function**

Functions	Specifications
Host interface	USB 2.0
Target interface	UART (1-wire mode)
Target system voltage	2.7 to 5.5 V (depends on the target device)
Clock supply	Internal high-speed oscillation clock is used
Power supply	5 V $\pm$ 0.3 V (maximum current rating: 100 mA)
Acquisition of device-specific information	Parameter file for Renesas Electronics is used
Programming software	RFP <sup>Note</sup> (Renesas Flash Programmer)
Security flag setting	Available
Standalone operation	Unavailable (must be connected to host machine)

**Note** For detailed usage of the RFP, refer to the **RFP User's Manual**.

#### 4.3.2 Cautions on flash programming

This section describes the cautions for flash programming. Be sure to read the following for the proper use of EZ-CUBE.

- To improve the writing quality, fully understand, verify, and evaluate the following items before using EZ-CUBE.
  - Circuits are designed as described in the user's manuals for the device and EZ-CUBE.
  - The device, RFP and EZ-CUBE are used as described in each user's manual.
  - The power supplied to the target system is stable.

## CHAPTER 5 HOW TO USE EZ-CUBE WITH 78K0 MICROCONTROLLER

This chapter describes how to use EZ-CUBE when performing on-chip debugging and flash programming for a 78K0 microcontroller.

On-chip debugging is a method to debug a microcontroller mounted on the target system, using a debug function implemented in the device. Since debugging is performed with the target device operating on the board, this method is suitable for field debugging.

Flash programming is a method to write a program to the flash memory embedded in a device. Erasing, writing and verifying the program can be performed on-board with the device.

Please update firmware for 78K0 at first. Refer to description (1) to (3) on the following order. For detail, refer to **1.4 Firmware Update**.

- (1) Connect EZ-CUBE to the host machine. **Do not connect EZ-CUBE to the target system.**
- (2) Start the EZ-CUBE firmware update tool"QBEZUTL.exe". Select firmware of 78K0 (78K0\_OCD\_FW.hex).
- (3) Click the [**Start**] button. Start to update the EZ-CUBE firmware.

Read the following chapters if you are using EZ-CUBE for the first time with a 78K0 microcontroller as the target device.

### 5.1 Target System Design

For communication between EZ-CUBE and the target system, communication circuits must be mounted on the target system. This section describes the circuit design and mounting of connectors.

### 5.2 On-Chip Debugging

This section describes the system configuration and startup method to perform on-chip debugging with EZ-CUBE.

### 5.3 Flash Programming

This section describes the system configuration and startup method to perform flash programming with EZ-CUBE.

## Supporting MCU

Table 5-1 shows the supporting MCUs of 78K0 EZ-CUBE firmware.

**Table 5-1 Supporting MCUs of 78K0 EZ-CUBE firmware**

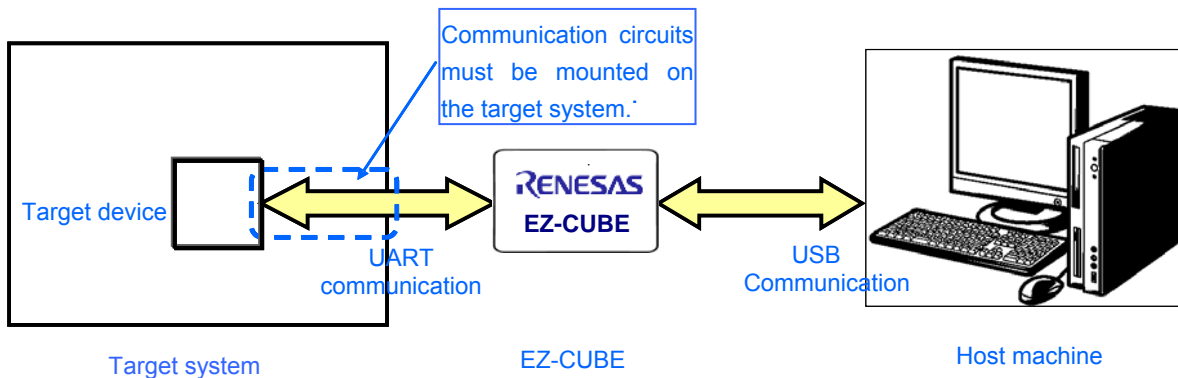
Items	Contents	Firmware
Supporting MCUs	78K0/Kx2, Lx3	78K0_OCD_FW.hex
Supporting Operation mode	UART mode	

### 5.1 Target System Design

This section describes the target system circuit design required for on-chip debugging and flash programming.

Figure 5-1 present overviews of the EZ-CUBE communication interface. For communication between EZ-CUBE and the target system, communication circuits must be mounted on the target system, as shown on the left side of the figure. Refer to this section to design circuits appropriately.

Figure 5-1. Outline of Communication Interface for On-Chip Debugging



#### 5.1.1 Pin assignment

This section describes the interface signals used between EZ-CUBE and the target system. Table 5-2 lists the pin assignment. Table 5-3 describes the functions of each pin.

Table 5-2. Pin Assignment

Pin No.	Pin name <sup>Note</sup>
1	GND
2	RESET_IN
3	Vdd
4	FLMD0
5	CLK
6	RxD.
7	RESET_OUT
8	TxD

**Note** Signal names in EZ-CUBE

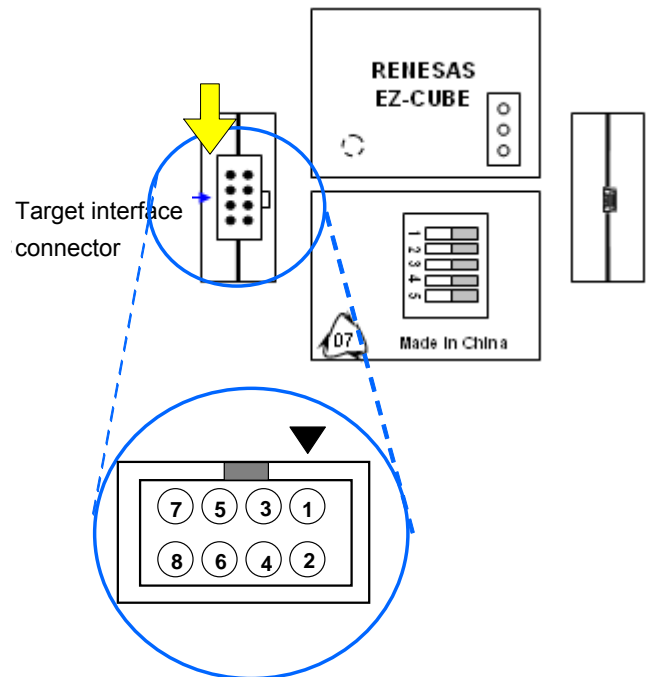


Table 5-3. Pin Functions

Pin Name	IN/OUT <sup>Note</sup>	Description
RESET_IN	IN	Pin used to input reset signal from the target system
RESET_OUT	OUT	Pin used to output reset signal to the target device
CLK	OUT	Pin used to output clock signal to the target device
FLMD0	OUT	Pin used to set the target device to debug mode or programming mode
RxD	IN	Pin used to receive command/data from the target device
TxD	OUT	Pin used to transmit command/data to the target device

**Note** As seen from EZ-CUBE.

### 5.1.2 Circuit connection examples

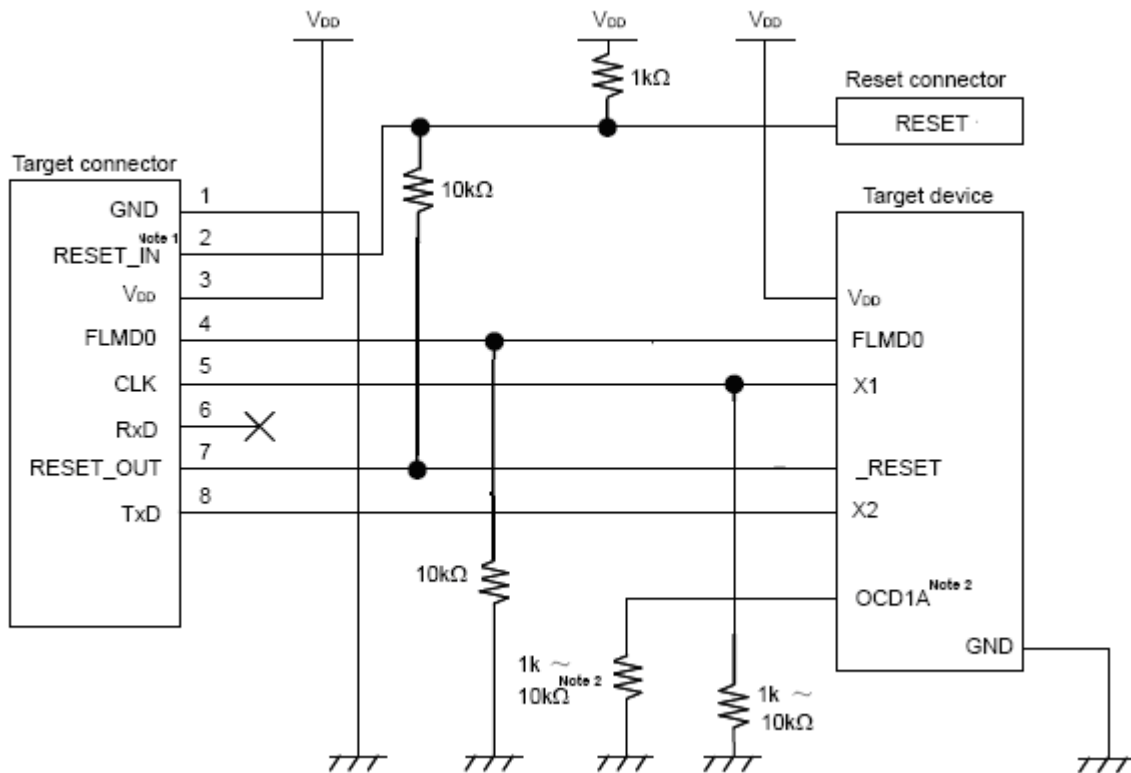
The circuit design on the target system varies depending on the used connector and interface signals.

The following (1) to (2) are the major purpose of use. Confirm the purpose, refer to Table 5-2 and see the relevant circuit connection example for specifications.

**Caution** The constants described in the circuit connection example are reference values. If you perform flash programming aiming at mass production, thoroughly evaluate whether the specifications of the target device are satisfied.

- (1) Used to perform on-chip debugging.
- (2) Used to perform flash programming.

Figure 5-2. When Debugging is Performed



- Notes**
1. This connection is designed assuming that the RESET signal is output from the N-ch open-drain buffer (output resistance: 100Ω or less). For details, refer to 5.1.3 **Connection of reset pin**.
  2. OCD1A may be a different name, depending on the device used. For details, refer to the user's manual for the target device.

#### EZ-CUBE Switch setting

- SW-1: Select switch to "M2".
- SW-2: Depend on the target system environment.
- SW-3: Select switch to "Debug Mode".
- SW-4: Depend on the target system environment.
- SW-5: Select switch to "Other".

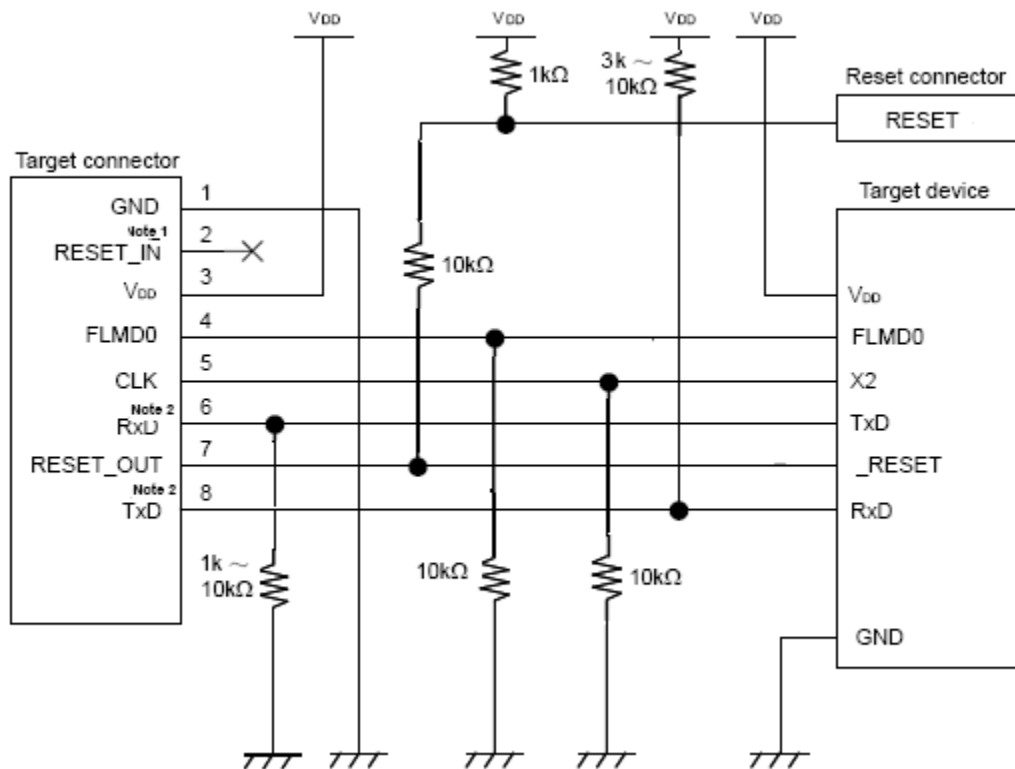
### ! CAUTION

Notes on the Target System Power Supply:



1. Do not change the switch setting after connecting USB cable.
2. The EZ-CUBE can supply power up to 100 mA current. Do not exceed the maximum supply current. The power is supplied from EZ-CUBE after connecting EZ-CUBE to the host machine.

Figure 5-3. When Programming is Performed



- Notes**
1. This connection is designed assuming that the RESET signal is output from the N-ch open-drain buffer (output resistance: 100 Ω or less). For details, refer to **5.1.3 Connection of reset pin**.
  2. Connect Tx/D (transmit side) of the target device to Rx/D (receive side) of the target connector, and Tx/D (transmit side) of the target connector to Rx/D (receive side) of the target device.

#### EZ-CUBE Switch setting

- SW-1: Select switch to "M1".
- SW-2: Depend on the target system environment.
- SW-3: Select switch to "Debug Mode".
- SW-4: Depend on the target system environment.
- SW-5: Select switch to "Other".

### ! CAUTION

Notes on the Target System Power Supply:



1. Do not change the switch setting after connecting USB cable.
2. The EZ-CUBE can supply power up to 100 mA current. Do not exceed the maximum supply current. The power is supplied from EZ-CUBE after connecting EZ-CUBE to the host machine.

#### 5.1.3 Connection of reset pin

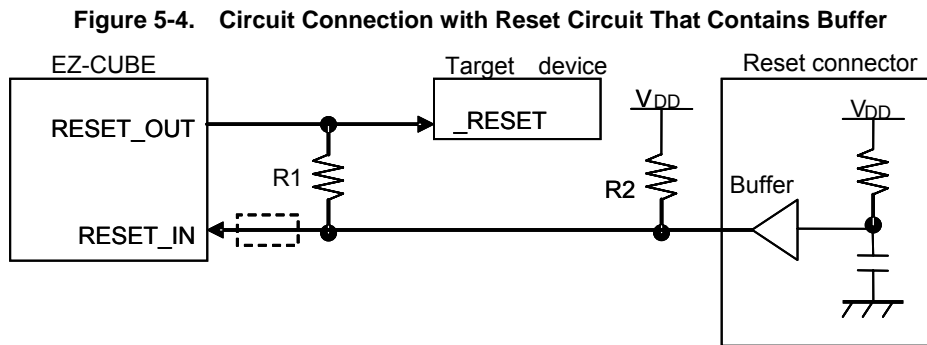
This section describes the connection of the reset pin, for which special attention must be paid, in circuit connection examples shown in the previous section.

During on-chip debugging, a reset signal from the target system is input to EZ-CUBE, masked, and then output to the target device. Therefore, the reset signal connection varies depending on whether EZ-CUBE is connected.

For flash programming, the circuit must be designed so that the reset signals of the target system and EZ-CUBE do not conflict.

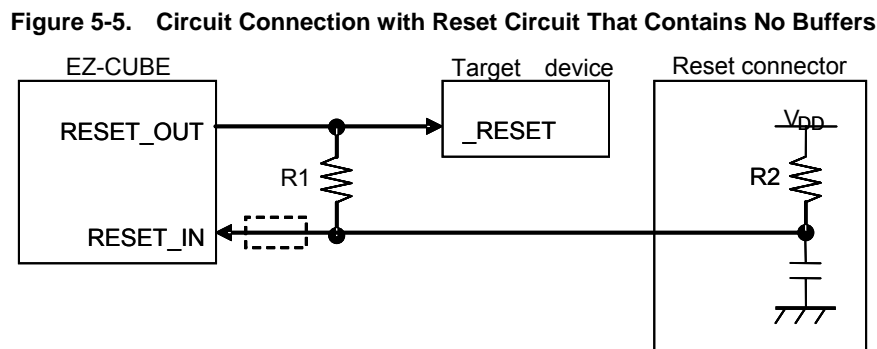
**Automatically switching the reset signal via resistor** (recommended; described in recommended circuit connection in the previous section).

Figure 5-4 illustrates the reset pin connection described in **5.1.2 Circuit connection examples**. This connection is designed assuming that the reset circuit on the target system contains an N-ch open-drain buffer (output resistance: 100Ω or less). The VDD or GND level may be unstable when the logic of RESET\_IN/OUT of EZ-CUBE is inverted, so observe the conditions described below in **Remark**.



**Remark** Make the resistance of at least R1 ten times that of R2, R1 being 10 kΩ or more.  
 Pull-up resistor R2 is not required if the buffer of the reset circuit consists of CMOS output.  
 The circuit enclosed by a dashed line is not required when only flash programming is performed.

Figure 5-5 illustrates the circuit connection for the case where the reset circuit on the target system contains no buffers and the reset signal is only generated via resistors or capacitors. Design the circuit, observing the conditions described below in **Remark**.



**Remark** Make the resistance of at least R1 ten times that of R2, R1 being 10 kΩ or more.  
 The circuit enclosed by a dashed line is not required when only flash programming is performed.

### 5.1.4 Cautions on Target system Design

Note the following cautions when designing the target system.

- If possible, do not create sections in which the communication lines for debugging run in parallel in the target system. If this cannot be prevented, shorten the sections as much as possible.
- Use a GND pattern to shield the communication lines for debugging to reduce their capacitive load, because the lines are used for high-speed communication.
- Make the distance between the target connector and the target device as short as possible.
- Before shipping the product, use jumpers or other means to physically separate the X1/OCD1A and X2/OCD1B pins from the target connector in order to ensure normal clock oscillation.
- To use X1/OCD1A and X2/OCD1B as communication pins for debugging, remove elements such as resonator capacitance and feedback resistors, so that the signals do not degrade due to capacitive load.

### 5.1.5 Clock Setting

The clock signal generated by the X1 oscillator, internal high-speed oscillator can be used for the clock signal of the target device during on-chip debugging. Setting up each is described below.

#### (a) Using the internal high-speed oscillator

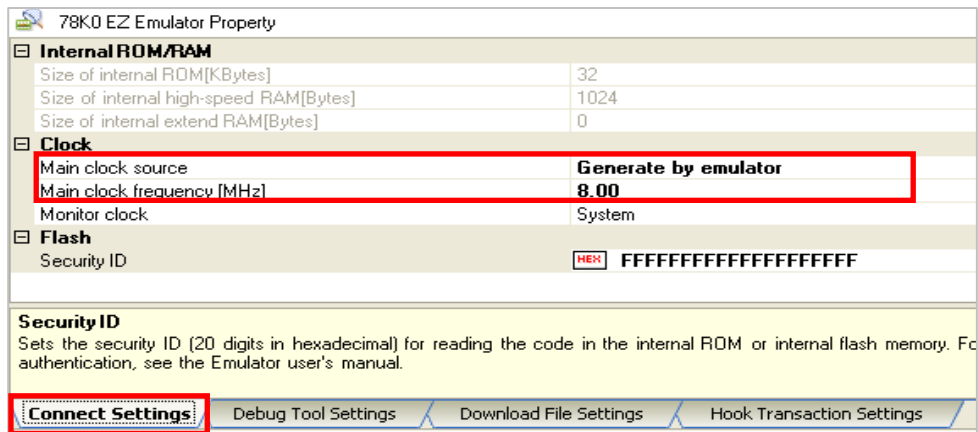
EZ-CUBE can supply a clock to be used as the high-speed system clock (4, 8, or 16 MHz).

#### EZ-CUBE Select SW-2 is “Int. Clock”

Remove the oscillator or oscillation circuit (Selected for “Generate by emulator” in the Configuration dialog box of the debugger).

For the settings, refer to the user’s manual for CS+.

Figure 5-6. Clock setting

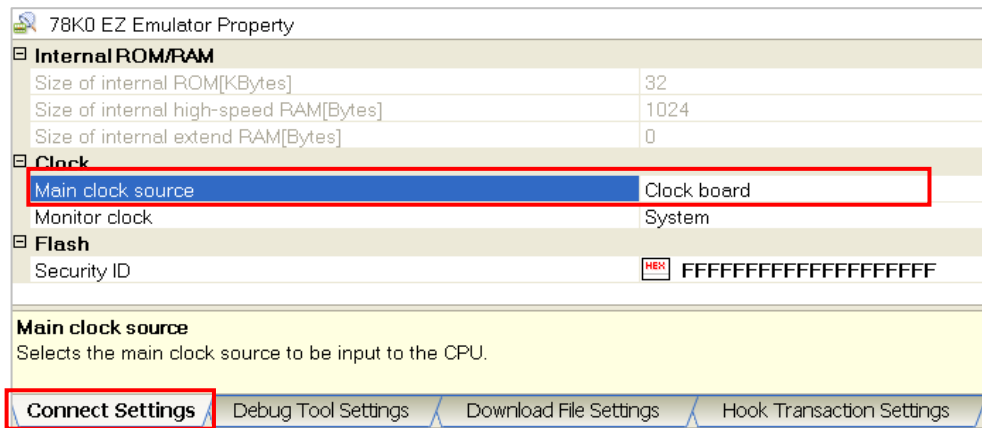




## (b) Using the External oscillator

**EZ-CUBE Select SW-2 is “Ext. Clock”**

Connect an oscillator or oscillation circuit on the External X'tal socket. (Select for “Clock board” in the Configuration dialog box of the debugger.) For the operation this step, refer to the user's manual for CS+.

**Figure 5-7. Clock setting**

## 5.2 On-Chip Debugging

This section describes the system configuration, startup/shutdown procedure and cautions for debugging when on-chip debugging is performed with EZ-CUBE.

### 5.2.1 Debug functions

Table 5-4 lists the debug functions when a 78K0 microcontroller is the target device.

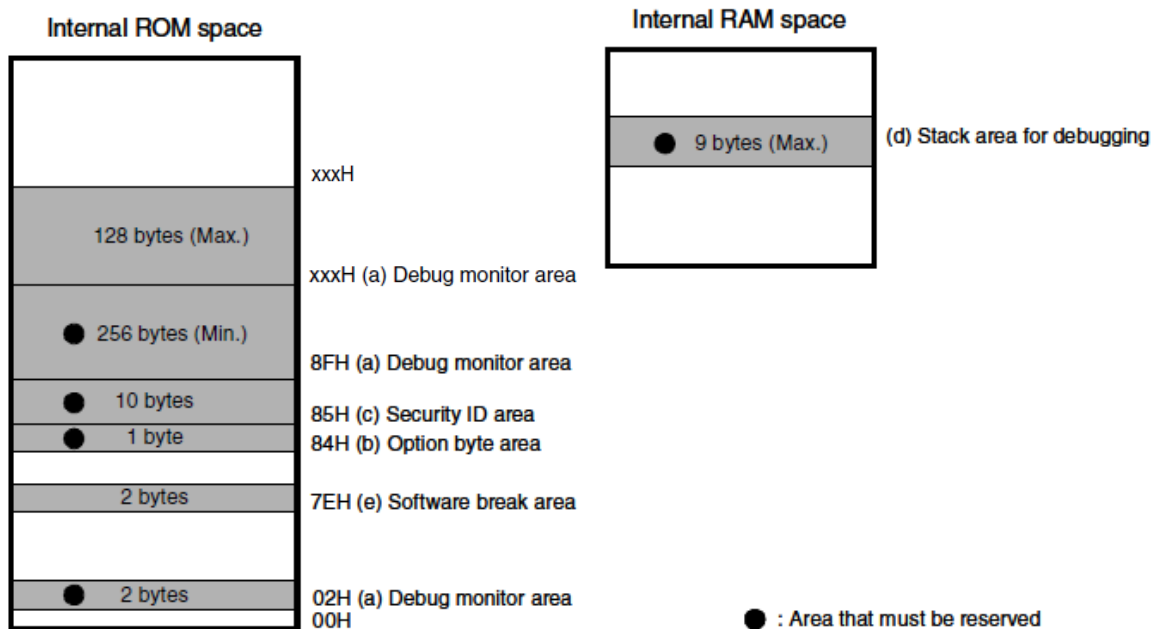
**Table 5-4. Debug Functions**

Functions	Specifications
Target device	78K0 78K0/Kx2, Lx3
Security	10-byte ID code authentication
Download	Available
Execution	Go, Step In, Step Over, CPU Reset, Restart
Hardware break	1 point
Software break	Multiple points
User spaces used for debugging	Internal ROM: 256 to 400 bytes Internal RAM: 7 to 9 bytes depending on the device used
Function pins used for debugging	X1, X2

### 5.2.2 Securing of user resources and setting of security ID

EZ-CUBE uses the user memory spaces (shaded portions in Figure 5-8) to implement communication with the target device, or each debug functions. The areas marked with a dot (●) are always used for debugging, and other areas are used for each debug function used. Refer to the descriptions of (a) to (e) on the following pages and secure these spaces in the user program or using the compiler options.

**Figure 5-8. Reserved Area Used by EZ-CUBE**



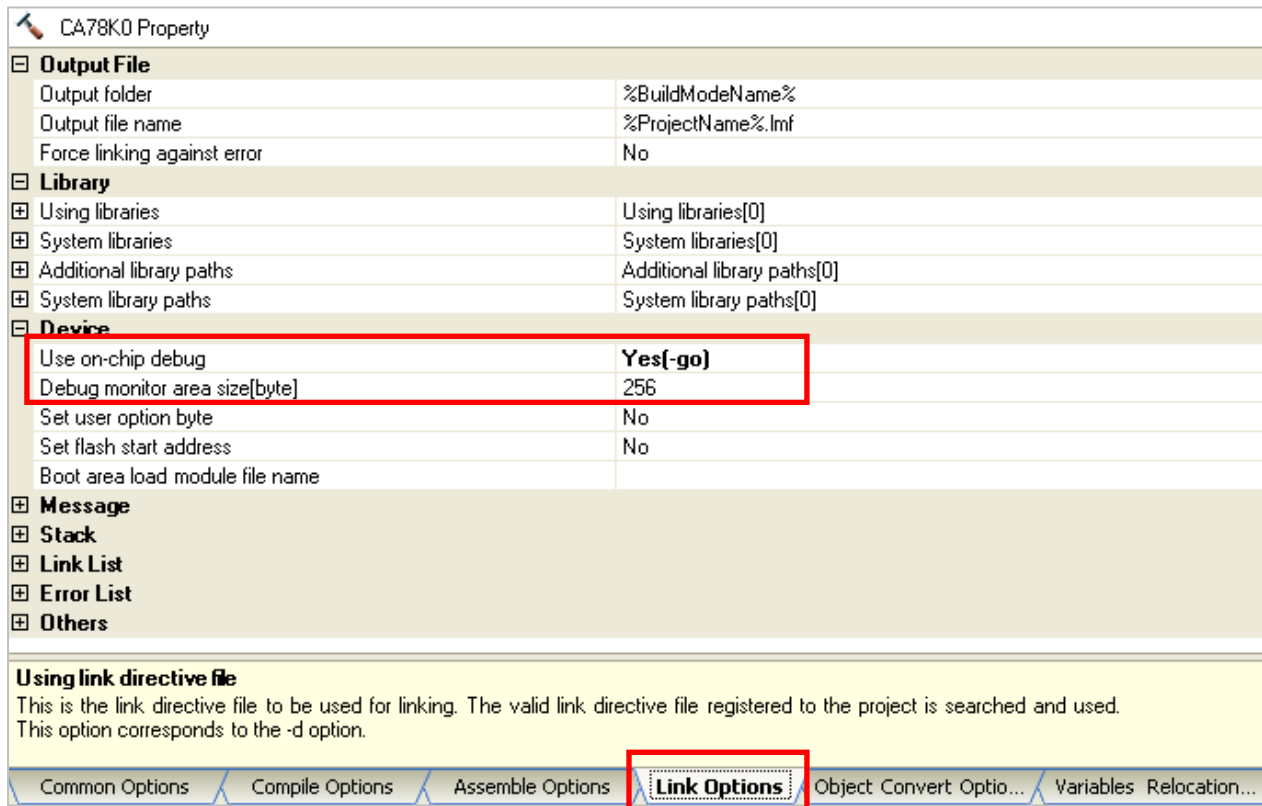
**(a) Debug monitor areas (areas must be secured)**

Addresses 0x02, 0x03 and area starting from address 0x8F must be secured to embed the debug monitor program. Be sure to reserve this area. The monitor program performs initialization process for the communication interface for debugging as well as run/break processing of the CPU. If this area is rewritten by user program or flash self-programming, on-chip debugging can no longer be performed.

[Area reservation method]

Figure 5-9 shows an example of reserving an area with the CS+. Figure 5-9 is the setting dialog for link option of the CS+. In [Use on-chip debug] in the red rectangle in Figure 5-9, set [Yes (-go)] and specify [Debug monitor area size] (256 byte). This setting reserves the area of 0x02, 0x03, and 0x8F and on for the debug monitor.

**Figure 5-9. Link Options Setting (Debug Monitor Area)**

**(b) Option byte area (required)**

This is the area for the security setting to prevent the flash memory from being read by an unauthorized person. The target device operates in accordance with the set value, as shown below. For the detailed settings of the option byte area, refer to the user's manual of the device.

Table 5-5. Option Byte Setting and Operation

Set Value	Description
0x00	The debugger cannot be started even if EZ-CUBE is connected.
0x02	The internal flash memory is not erased regardless of how many times authenticating the security ID code fails.
0x03	The entire internal flash memory area is erased if authenticating the security ID code fails.
Other than above	Setting prohibited

[How to set]

There are the following 2 methods to set option bytes to the internal flash memory. If both (1) and (2) are set, the setting of (2) has precedence.

#### (1) Setting method with the program

Embed option bytes to the user program. Add the code to the assembler source by referring to the following example.

**Example** Setting 80H:03, 81H:00 82H:00 83H:00H 84H:02H

```

SSS    CSEG    AT      080H;    "SSS" is any symbol name (eight characters or less)
                DB      03H;
                DB      00H;
                DB      00H;
                DB      00H;
                DB      02H;

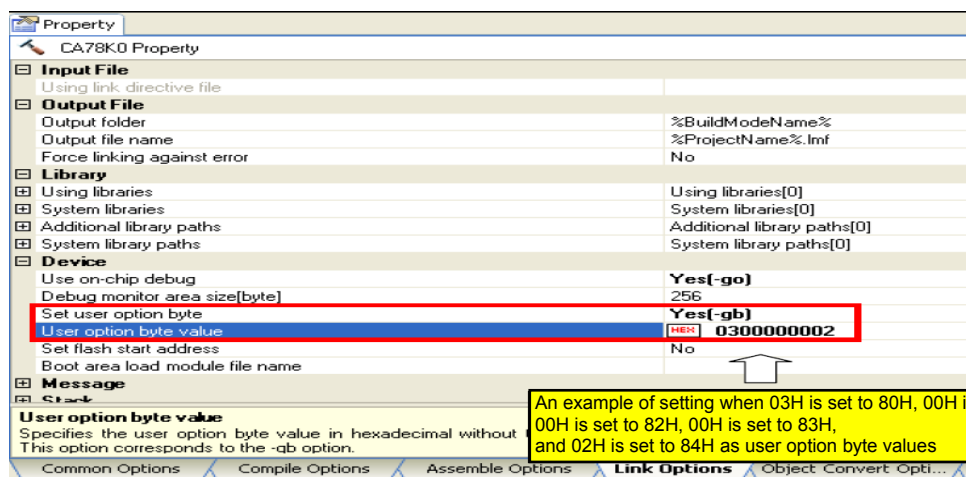
```

**Caution** If address 0x84 is overwritten by 0x00 by self programming, communication is disabled, and debugging and connection can no longer be performed even if the debugger is restarted. In such as case, erase the memory via flash programming.

#### (2) Setting method with the CS+

You can also set the option byte area with the CS+ setting. In [Set user option byte] in the red rectangle in Figure 5-10, select [Yes (-gb)], and specify the option byte values for the addresses from 80H to 84H.

Figure 5-10. User Option Byte Setting



**(c) Security ID area (essential)**

This is the area for the security setting to prevent the flash memory from being read by an unauthorized person. The security ID functions as a password for starting the debugger. The debugger starts only when the security ID that is input during debugger startup and the security ID embedded in this area match.

If you forget the security ID, erase the flash memory, and set a new security ID.

[How to set]

There are the following 2 methods to set a security ID to the internal flash memory. If both (1) and (2) are set, the setting of (2) has precedence.

**(1) Embedding the security ID in 0x85-0x8E in the user program**

Embed the security ID in 0x85-0x8E in the user program. For example, if the security ID is embedded as described below, the security ID set in the debugger is "0123456789ABCDEF1234" (not case-sensitive).

Example) Setting the security ID "0123456789ABCDEF1234" to the addresses 0x85 to 0x8E

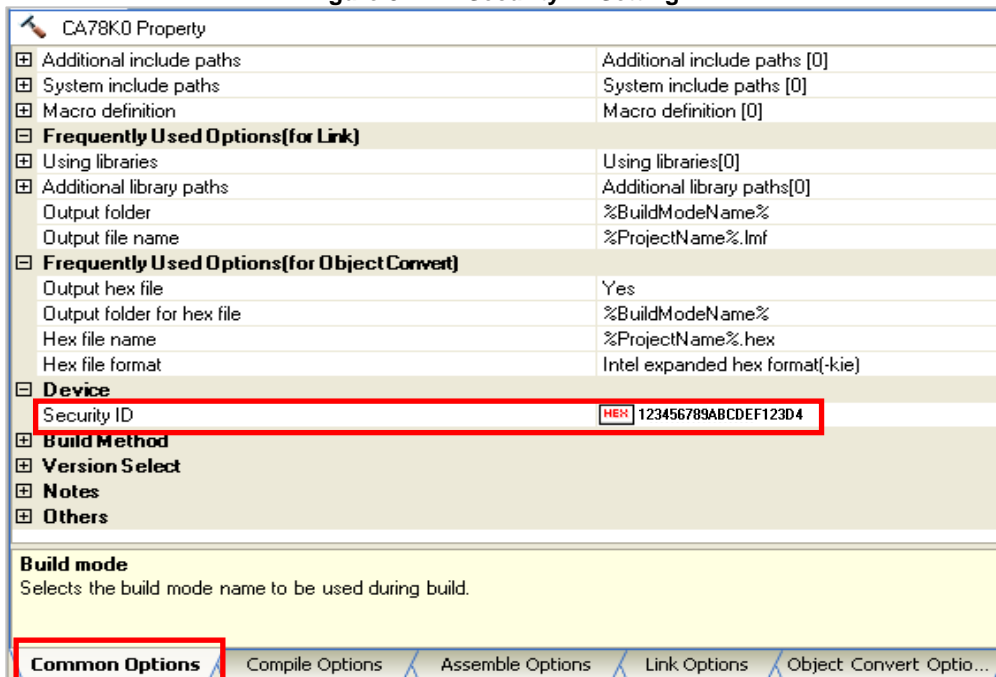
SSS CSEG AT 85H; "SSS" is any symbol name (up to 8 characters).

```
DB 01H;
DB 23H;
DB 45H;
DB 67H;
DB 89H;
DB ABH;
DB CDH;
DB EFH;
DB 12H;
DB 34H;
```

**(2) Setting method with the CS+**

You can also set the security ID with the CS+ setting. In the right field of [Security ID] in the red rectangle in Figure 5-11, specify the values in hexadecimal successively for the addresses from 85H to 8FH.

**Figure 5-11. Security ID Setting**

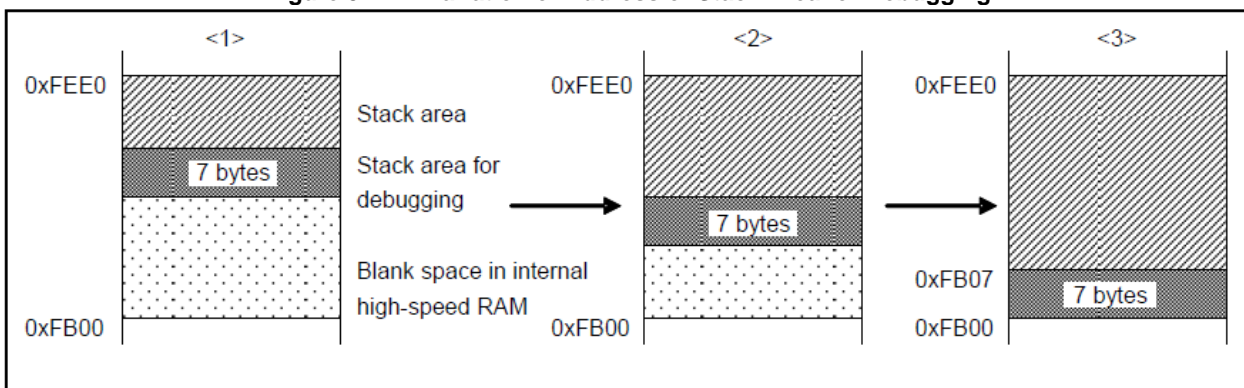


**(d) Stack area for debugging (this area must be secured)**

This area requires 7 to 9 bytes as the stack area for debugging. Since this area is allocated immediately before the stack area, the address of this area varies depending on the stack increase and decrease.

Figure 5-12 illustrates the case where the stack area is increased when the internal high-speed RAM starts from 0xFB00.

**Figure 5-12. Variation of Address of Stack Area for Debugging**



The size of this area also varies depending on whether software breaks or pseudo real-time RAM monitor is used.

**Table 5-6. Size of Stack Area for Debugging**

Item	Size of Stack Area for Debugging
Standard	7 bytes
When software breaks are used	9 bytes

[How to secure areas]

Refer to the address range shown below and set the stack pointer.

**Example** When internal high-speed RAM starts from 0xFB00

- Standard

Within the range 0xFB07 to 0xFEE0

- When software breaks are used (also refer to (e))

Within the range 0xFB09 to 0xFEE0

**(e) Area for software break**

This area is used for software breaks.

[How to secure areas]

Refer to the following and secure the area.

SSS CSEG AT 07EH; "SSS" is any symbol name (eight characters or less)  
 DB 0FFH, 0FFH

### 5.2.3 Cautions on debugging

This section describes cautions on performing on-chip debugging for a 78K0 microcontrollers. Be sure to read the following to use EZ-CUBE properly.

#### (1) Handling of device that was used for debugging

Do not mount a device that was used for debugging on a mass-produced product, because the flash memory was rewritten during debugging and the number of rewrites of the flash memory cannot be guaranteed.

#### (2) Overwriting flash memory during on-chip debugging

If the following operations are performed during on-chip debugging, the flash memory in the device is overwritten.

<1> Writing to internal flash memory

<2> Program execution after specifying or canceling software breakpoints

<3> Step-over execution, Return Out execution

<4> Come Here

<5> If Permit is selected in the Target Power off area in the Configuration dialog box, the following operations cannot be performed:

- a) Specifying, changing, or canceling hardware breakpoints
- b) Masking/unmasking internal resets
- c) Switching peripheral breakpoints
- d) Program execution
- e) Software reset (a reset performed by the debugger)

<6> Adding, changing, or deleting the monitor address when using the pseudo real-time RAM monitor function

<7> Performing operations without using breakpoints when software breakpoints are specified

<8> When the debugger is started or terminated

It takes time from completion of flash memory programming until the control is passed to GUI.

#### (3) Software break

During program running, do not rewrite the data at the address where a software break is set. This includes self programming and rewriting to RAM. If performed, the instruction placed at the address may be invalid.

#### (4) Self programming

If the space where the monitor program for debugging is rewritten by flash self programming, the debugger does not operate correctly. This also holds true when boot swapping is executed.

#### (5) Boot swapping during self programming

The boot swapping function cannot be emulated. This is because boot swapping moves the memory spaces used for debugging, and thus the debug communication can no longer be performed.

#### (6) Break function for stack pointer initialization failure

This function executes a break when an interrupt occurs or a PUSH instruction is executed while the initial setting has not been made for the stack pointer.

If the manipulation or instruction shown below is executed immediately after a reset operation, the break function for stack pointer initialization failure becomes invalid.

- Setting a software break
- Write to the stack pointer from the Register window
- Write to the flash memory from the Memory window, etc

If a software break occurs while the initial setting has not been made for the stack pointer, the message "Uninitialized Stack Pointer" is displayed on the status bar.

The subsequent operations are not performed normally, so make sure to set the SP value in the user program.

**(7) Caution on downloading a HEX file**

When downloading a HEX file, do not set specify a filling value other than 0xFF for the object converter option (-U).

**(8) Cautions when stepping into code**

The value of some SFRs (special function registers) might remain unchanged while stepping into code. If the value of the SFRs does not change while stepping into code, operate the microcontroller by continuously executing the instructions instead of executing them in steps.

Stepping into code: Instructions in the user-created program are executed one by one.

Continuous execution: The user-created program is executed from the current PC value.

**(9) Emulation of POC function**

The POC function of the target device cannot be emulated. Make sure that the power to the target system is not shut down during debugging



### 5.3 Flash Programming

This section describes the system configuration and startup/shutdown procedure when flash programming is performed for a 78K0 microcontroller, using EZ-CUBE.

#### 5.3.1 Specifications of programming function

**Table 5-7. Specifications of Programming Function**

Functions	Specifications
Host interface	USB 2.0
Target interface	UART
Target system voltage	2.7 to 5.5 V (depends on the target device)
Clock supply	8MHz clock can be supplied Clock mounted on the target system can be used
Power supply	5 ±0.3 V (maximum current rating: 100 mA)
Acquisition of device-specific information	Parameter file for Renesas Electronics is used
Programming software	RFP <sup>Note</sup> (Renesas Flash Programmer)
Security flag setting	Available
Standalone operation	Unavailable (must be connected to host machine)

**Note** For detailed usage of the RFP, refer to the **RFP User's Manual**.

#### 5.3.2 Cautions on flash programming

This section describes the cautions for flash programming. Be sure to read the following for the proper use of EZ-CUBE.

- To improve the writing quality, fully understand, verify, and evaluate the following items before using EZ-CUBE.
  - Circuits are designed as described in the user's manuals for the device and EZ-CUBE.
  - The device, RFP and EZ-CUBE are used as described in each user's manual.
  - The power supplied to the target system is stable.

## CHAPTER 6 HOW TO USE EZ-CUBE WITH RX MICROCONTROLLER

This chapter describes how to use EZ-CUBE when performing on-chip debugging and flash programming for a RX microcontroller.

On-chip debugging is a method to debug a microcontroller mounted on the target system, using a debug function implemented in the device. Since debugging is performed with the target device operating on the board, this method is suitable for field debugging.

Flash programming is a method to write a program to the flash memory embedded in a device. Erasing, writing and verifying the program can be performed on-board with the device.

Please update firmware for RX at first. Refer to description (1) to (3) on the following order. For detail, refer to **1.4 Firmware update**.

- (1) Connect EZ-CUBE to the host machine. **Do not connect EZ-CUBE to the target system.**
- (2) Start the EZ-CUBE firmware update tool"QBEZUTL.exe". Select firmware of RX (RX\_OCD\_FW.hex).
- (3) Click the [**Start**] button. Start to update the EZ-CUBE firmware.

Read the following chapters if you are using EZ-CUBE for the first time with a RX microcontroller as the target device.

### 6.1 Target System Design

For communication between EZ-CUBE and the target system, communication circuits must be mounted on the target system. This section describes the circuit design and mounting of connectors.

### 6.2 On-Chip Debugging

This section describes the system configuration and startup method to perform on-chip debugging with EZ-CUBE.

### 6.3 Flash Programming

This section describes the system configuration and startup method to perform flash programming with EZ-CUBE.

## Supporting MCU

Table 6-1 shows the supporting MCUs of RX EZ-CUBE firmware.

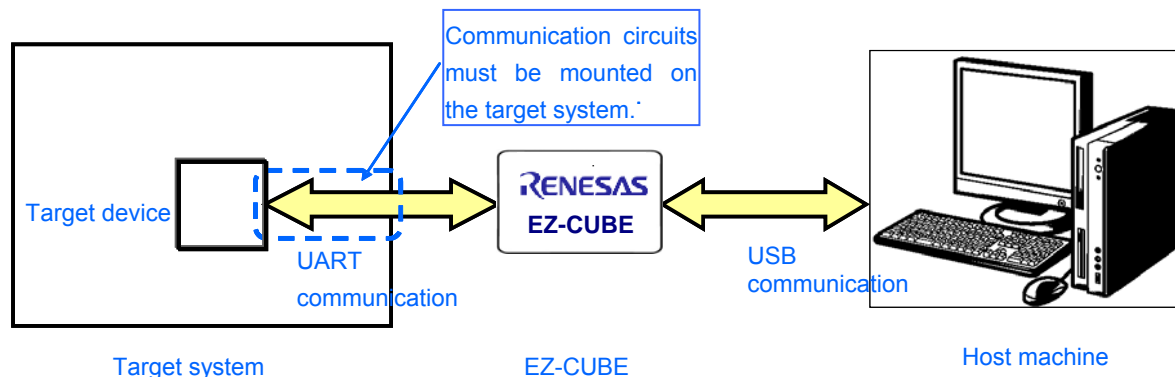
**Table 6-1 Supporting MCUs of RX EZ-CUBE firmware**

Items	Contents	Firmware
Supporting MCUs	RX600 series RX63T group [R5F563T4/5/6]	RX_OCD_FW.hex
	RX200 series RX210, RX220 groups	
	RX100 series RX111 group	
Supporting Operation mode	Single chip mode	

## 6.1 Target System Design

Figure 6-1 shows the outline of EZ-CUBE communication interface.

**Figure 6-1. Outline of Communication Interface**



**Note:** Single chip mode

### 6.1.1 Pin assignment

Table 6-2 shows the pin assignments of EZ-CUBE. Table 6-3 shows the pin functions.

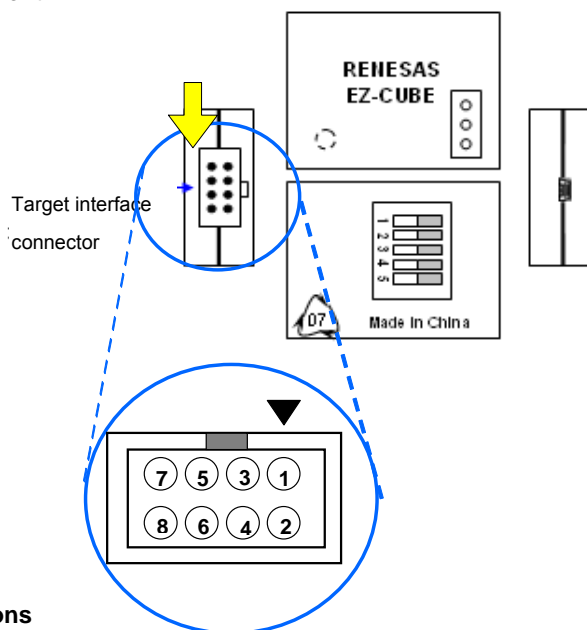
**Table 6-2. Pin Assignment**

Pin No.	Pin name *1	
	RX63T [R5F563T4/5/6]	RX210, RX220, RX111
1	GND	GND
2	RES#	RES#
3	VCC	VCC
4	N.C.	N.C.
5	FINEC *2	N.C.
6	RxD *3	RxD *3
7	N.C.	N.C.
8	MD/FINED, TxD *3	MD/FINED, TxD *3

\*1 Signal name on EZ-CUBE

\*2 Mount the 16MHz three-terminal oscillator on EZ-CUBE, and set SW-2 to "Ext. Clock".

\*3 TxD and RxD pins are necessary for flash programming by flash programmer.



**Table 6-3. Pin Functions**

Pin name	IN/OUT *	Explanation
FINEC	OUT	Communication clock output (Only RX63T group [R5F563T4/5/6])
MD/FINED, TxD	IN/OUT	Debugger communication data input-output (MD/FINED), or Flash programmer communication data output (TxD)
RxD	IN	Flash programmer communication data input
RES#	IN/OUT	Reset signal input-output
VCC	Power	Power input (2.7V - 5V) or Power output (5V)
GND	Power	Ground

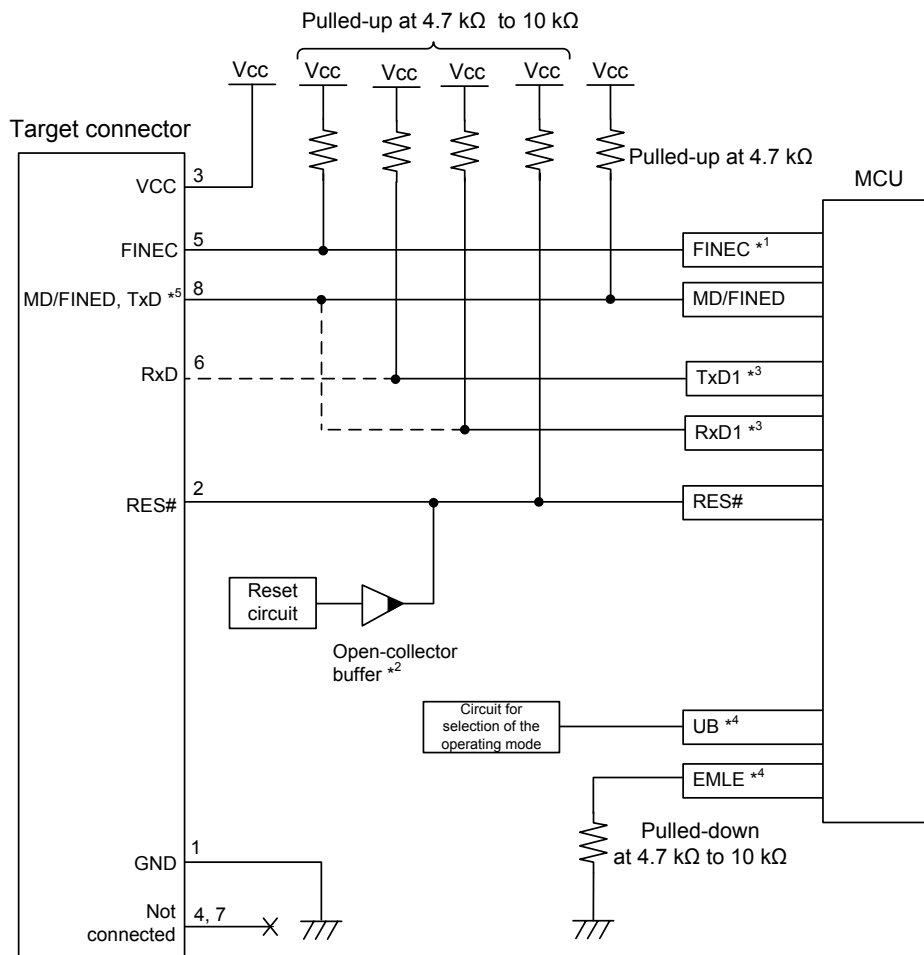
\* Seen from EZ-CUBE side

6.1.2 Recommended Circuit Connection

(1) Recommended Circuit Connection for Debugging

Table 36-2 shows the recommended circuit connection for debugging.

Figure 6-2. Recommended Circuit Connection

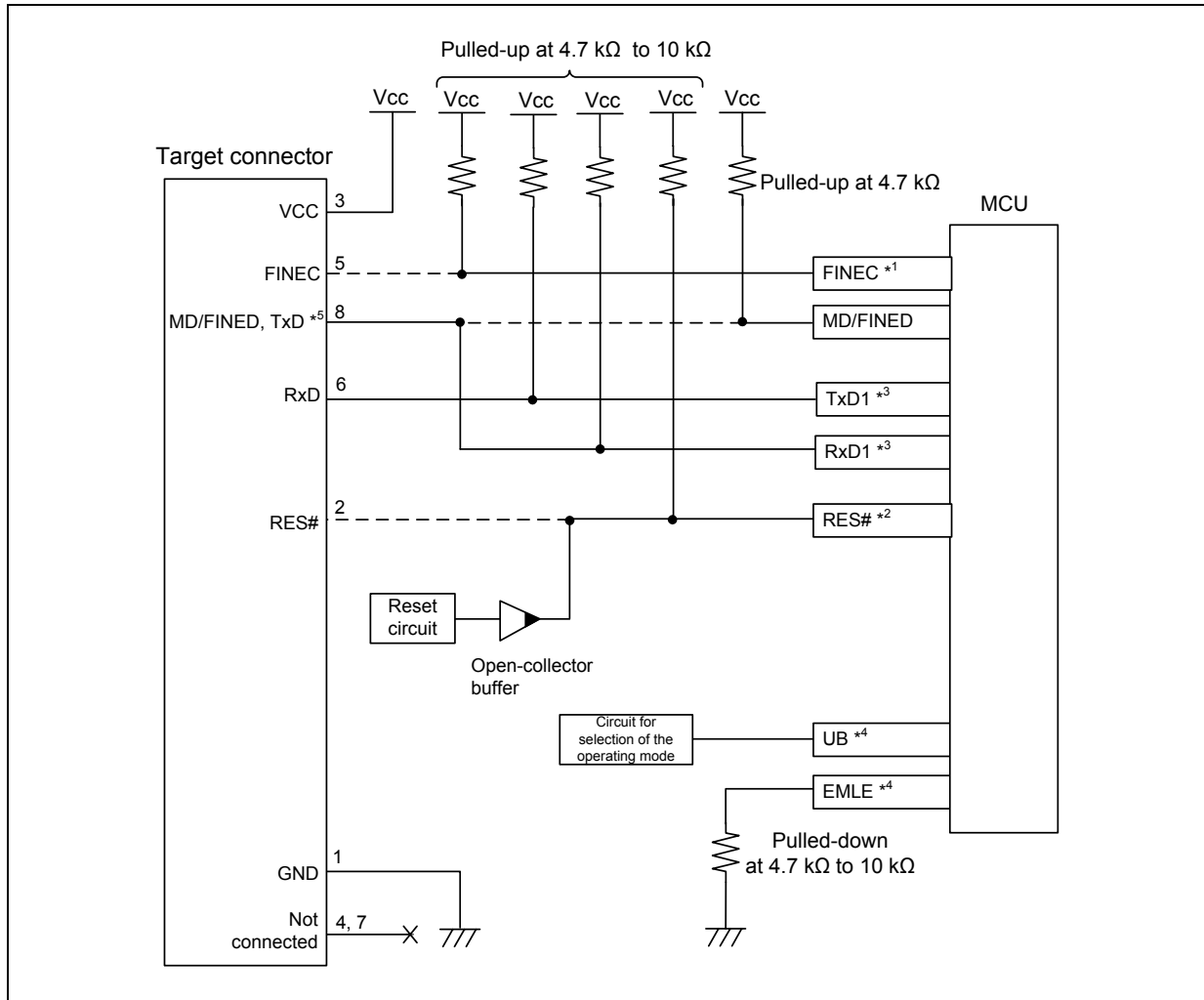


- \*1 Connect FINEC pin for debugging RX63T group [R5F563T4/5/6]. No need connection for debugging RX210, RX220 and RX111 groups. This pin can be used as the normal port.
- \*2 The output of the reset circuit of the target system must be open collector.
- \*3 TxD1 and RxD1 pins are not used for debugging. No need connection.
- \*4 Refer to "Notes on Connection" for detailed process of UB and EMLE pins.
- \*5 Connected pin of the MCU for debugging is different from for flash programming. Connect to MD/FINED pin of the MCU and pull-up RxD1 pin of the MCU.

**(2) Recommended Connection Circuit for Flash Programming**

Figure 6-3 shows the recommended circuit connection for flash programming.

**Figure 6-3 Recommended Circuit Connection for Flash Programming**



\*1 FINEC pin is not used for flash programming. No need connection.

\*2 RES# pin is not used for flash programming. No need connection.

\*3 Connect TxD1 and RxD1 pins for flash programming.

\*4 Refer to "Notes on Connection" for detailed process of UB and EMLE pins.

\*5 Connected pin of the MCU for flash programming is different from for debugging. Connect to RxD1 pin of the MCU and pull-up MD/FINED pin of the MCU for flash programming.

**EZ-CUBE Switch setting****RX63T group [R5F563T4/5/6]**

SW-1: Select "M2"

SW-2: Select "Ext. Clock", Need mounting 16MHz oscillator.

SW-3: Select "Debug Mode"

SW-4: Select "T"

SW-5: Select "M3"

**RX210, RX220 group and RX111 group**

SW-1: Select "M2"

SW-2: Select "Int. Clock"

SW-3: Select "Debug Mode"

SW-4: Depend on the target system environment

SW-5: Select "M3"

**CAUTION**

Notes on the Target System Power Supply:



1. Do not change the switch setting after connecting USB cable.
2. The EZ-CUBE can supply power up to 100 mA current. Do not exceed the maximum supply current. The power is supplied from EZ-CUBE after connecting EZ-CUBE to the host machine.
3. RX111 operate voltage up to 3.6V.

### 6.1.3 Notes on Connection

#### (1) About the FINEC and MD/FINED pins

For the RX63T Groups, FINE interface only supports a 2-wire system using FINEC and MD/FINED pins. The FINEC and MD/FINED pins are exclusively used by the emulator. Any functions that are multiplexed on the FINEC pin are not available.

For the RX210, RX220 and RX111 Groups, FINE interface supports a 1-wire system using the MD/FINED pin. Only the MD/FINED pin is exclusively used by the emulator. It is not necessary to connect the FINEC pin since this pin is not used. The FINEC pin can be used as a port.

Pull up the FINEC signal at 4.7 k $\Omega$  to 10 k $\Omega$ . Pull up the MD/FINED signal at 4.7 k $\Omega$ . Do not arrange these signal lines in parallel with or across other high-speed signal lines.

MD/FINED signal and TxD signal are assigned to the same pin. Connect this pin to MD/FINED pin of the MCU when debugging.

#### (2) About the TxD and RxD pins

TxD and RxD signals are NOT required for debugging. These are only used for internal flash programming with Renesas Flash Programmer.

If the MCU has multiple TxD1 or RxD1 pins, confirm which one of the respective pins is used in boot mode in the hardware manual of the MCU.

MD/FINED signal and TxD signal are assigned to the same pin. Connect this pin to RxD1 pin of the MCU for internal flash programming with Renesas Flash Programmer.

#### (3) About the RES# pin

The emulator uses the RES# pin. If the target system includes a user logic reset circuit, the output signal from the reset circuit must be connected to the RES# pin of the connector via an open-collector buffer. If there is no reset circuit, on the other hand, the RES# pin of the connector must be directly connected to the RES# pin of the MCU.

#### (4) About the EMLE pin (Only RX63T group [R5F563T4/5/6])

Pull the levels on the EMLE down at 4.7 k $\Omega$  to 10 k $\Omega$  on the target system.

#### (5) About the UB pin

UB pin is the port for entering the user boot mode and the USB I/F mode. Which port is the UB pin depends on the MCU. Refer to the section on operation modes in the hardware manual of the MCU to be used.

The handling of pins is not necessary for debugging RX63T group [R5F563T4/5/6] because of not having the user boot mode.

Pull the levels on the UB down at 4.7 k $\Omega$  to 10 k $\Omega$  on the target system not to transit to the user boot mode for debugging RX210 and RX220 groups.

Pull the levels on the UB up at 4.7 k $\Omega$  to 10 k $\Omega$  on the target system not to transit to the USB I/F mode for debugging RX111 group.

**(6) About VCC**

Connect the VCC of the connector to the VCC (power supply) of the target system.

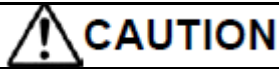
Use the emulator within the power supply voltage of 2.7V to 5.5V and within the operating voltage range of the MCU.

EZ-CUBE can supply 5V power to a simple evaluation system. Up to 100 mA current can be supplied. In case of supplying 5V power from the emulator, set SW-4 to "5".

When using the power supply function of EZ-CUBE, check the voltage supplied to the target system. The voltage may drop 0.5V or more since it depends on the USB VBUS power supply voltage.

**(7) About GND**

The pins of the connector marked "GND" must be at the same ground level as the VSS pin of the MCU.



Warning for Turning the Power On/Off:

When supplying power, ensure that there are no shorts between Vcc and GND. Only connect the EZ-CUBE after confirming that there are no mismatches of alignment on the target system port connector. Incorrect connection will result in the host machine, the emulator, and the target system emitting smoke or catching fire.



## 6.2 On-Chip Debugging

This section describes the system configuration, startup/shutdown procedure and cautions for debugging when on-chip debugging is performed with EZ-CUBE.

### 6.2.1 Debug functions

Table 6-4 shows the debugging functions list.

**Table 6-4. Debugging functions list**

Functions		Specifications		
		RX600 series	RX200 series	RX100 series
Target device		RX63T group [R5F563T4/5/6]	RX210, RX220 group	RX111 group
Supporting voltage		2.7 - 3.6 V	2.7 - 5 V	2.7 - 3.6 V
Maximum operating frequency		100 MHz	50 MHz	32 MHz
Operation mode		Single chip mode		
Download to internal ROM		Available		
Execution control		Go, Stop, Step in, Step over, Return out, CPU Reset, Restart		
Software break		Maximum 256 points		
Event	Execution address	Maximum 8 points	Maximum 4 points	
	Data access	Maximum 4 points	Maximum 2 points	
	Number of passes	Maximum 256 times	None	
On-chip break	Pre-PC break	Maximum 8 points	Maximum 4 points	
	Combination of events	OR/AND(cumulative)/Sequential		
	Other	Trace full break		
Trace (Internal trace)		Maximum 256 branches	Maximum 64 branches (RX210) Maximum 32 branches (RX220)	Maximum 32 branches
Performance measurement		Execution cycle (Maximum 2 points) or Number of executions (32-bit counter x 2)	Execution cycle (1 point) (24-bit counter x 1)	None
Debug communication pin		FINEC, MD/FINED	MD/FINED	

## 6.2.2 Notes on debugging

### (1) Reset during the User Program Execution

If an internal reset occurs during user program execution, it becomes impossible to control from the emulator. Do not generate an internal reset such as those generated by the watchdog timer.

When a pin reset has occurred during the execution of the target system, user program may hang up. If a pin reset input is detected, input a reset from the emulator and re-execute a user program, but it may become a break state to be unable to re-execute.

An error message "A timeout error. The MCU is in the reset state. Is system reset issued?" is displayed in cases of contention between a reset (through a pin, from the watchdog timer, etc.) and operations by the emulator system (memory reference in the [Memory] window, etc.). The emulator is initialized and the user program stops. After a system reset is issued, the trace record is initialized too. Debugging can be continued.

### (2) Reset cancellation time of the target system

Exceed the VIH voltage within 100ms after cancelling the reset input.

### (3) MCUs that are used in debugging

MCUs that are connected to the emulator and used in debugging are placed under stress by repeated programming of flash memory during emulation. Do not use MCUs that were used in debugging in mass-production for end users.

### (4) High-Speed Clock Oscillator (HOCO)

The emulator uses a device's internal high-speed clock oscillator (hereafter the HOCO) to achieve communications with RX200 series and RX100 series MCUs via FINE interface. Therefore, the HOCO is always in an oscillating state no matter how the HOCO-related registers are set.

If there is a contention between switching of the HOCO frequency and memory access, the memory access operation is not guaranteed.

### (5) Final Evaluation of the User Program

Before entering the mass-production phase, be sure to perform a final evaluation of the program which is written to a flash ROM using a flash programmer. Be sure to perform the evaluation singly, without the emulator connected.

### 6.3 Flash Programming

This section describes the system configuration and startup/shutdown procedure when flash programming is performed for a RX microcontroller, using EZ-CUBE.

#### 6.3.1 Specifications of programming function

**Table 6-5. Specifications of Programming Function**

Functions	Specifications
Host interface	USB 2.0
Target interface	Single chip mode
Target system voltage	2.7 to 5.5 V (depends on the target device)
Clock supply	Internal high-speed oscillation clock is used
Power supply	5 V $\pm$ 0.3 V (maximum current rating: 100 mA)
Acquisition of device-specific information	Parameter file for Renesas Electronics is used
Programming software	RFP <sup>Note</sup> (Renesas Flash Programmer)
Security flag setting	Available
Standalone operation	Unavailable (must be connected to host machine)

Note For detailed usage of the RFP, refer to the **RFP User's Manual**.

#### 6.3.2 Cautions on flash programming

This section describes the cautions for flash programming. Be sure to read the following for the proper use of EZ-CUBE.

- To improve the writing quality, fully understand, verify, and evaluate the following items before using EZ-CUBE.
  - Circuits are designed as described in the user's manuals for the device and EZ-CUBE.
  - The device, WriteEZ5 and EZ-CUBE are used as described in each user's manual.
  - The power supplied to the target system is stable.

## CHAPTER 7 HOW TO USE EZ-CUBE WITH V850 MICROCONTROLLER

This chapter describes how to use EZ-CUBE when performing on-chip debugging and flash programming for a V850 microcontroller.

On-chip debugging is a method to debug a microcontroller mounted on the target system, using a debug function implemented in the device. Since debugging is performed with the target device operating on the board, this method is suitable for field debugging.

Flash programming is a method to write a program to the flash memory embedded in a device. Erasing, writing and verifying the program can be performed on-board with the device.

Please update firmware at first. Follow step (1)-(3). For detail on update firmware, refer to **1.4 Firmware Update**.

- (1) Connect EZ-CUBE to the host machine. **Do not connect EZ-CUBE to the target system.**
- (2) Start the EZ-CUBE firmware update tool"QBEZUTL.exe". Select firmware of V850" V850\_OCD\_FW.hex".
- (3) Click the [**Start**] button. Start to update the EZ-CUBE firmware.

Read the following chapters if you are using EZ-CUBE for the first time with a V850 microcontroller as the target device.

### 7.1 Target System Design

For communication between EZ-CUBE and the target system, communication circuits must be mounted on the target system. This section describes the circuit design and mounting of connectors.

### 7.2 On-Chip Debugging

This section describes the system configuration and startup method to perform on-chip debugging with EZ-CUBE.

### 7.3 Flash Programming

This section describes the system configuration and startup method to perform flash programming with EZ-CUBE.

## Supporting MCU

Table 7-1 shows the supporting MCUs of V850 EZ-CUBE firmware.

**Table 7-1 Supporting MCUs of V850 EZ-CUBE firmware**

Items	Contents	Firmware
Supporting MCUs	V850ES/Jx3	V850_OCD_FW.hex
	V850ES/Jx3-L	
Supporting Operation mode	UART mode	

## 7.1 Target System Design

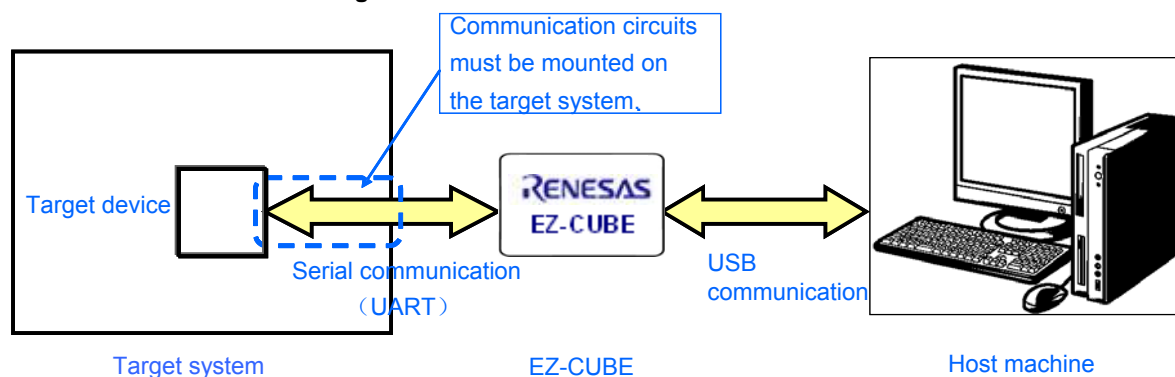
This section describes the target system circuit design required for on-chip debugging and flash programming.

Figure 7-1 presents an overview of the EZ-CUBE communication interface. As shown on the left side of the figure, EZ-CUBE performs serial communication with the target device on the target system. For this communication, communication circuits must be mounted on the target system. Refer to this section to design circuits appropriately.

UART are supported as communication modes.

The pins used for serial communication are basically the same as those of the flash memory programmer (such as EZ-CUBE), but some devices do not support some of them.

Figure 7-1. Outline of Communication Interface



### 7.1.1 Pin assignment

This section describes the interface signals used between EZ-CUBE and the target system. Table 7-2 lists the pin assignment. Table 7-3 describes the functions of each pin.

Table 7-2. Pin Assignment

Pin No.	Pin Name <sup>Note</sup>
1	GND
2	RESET_IN
3	VDD
4	FLMD0
5	CLK
6	RxD
7	RESET_OUT
8	TxD

**Note** Signal names in EZ-CUBE

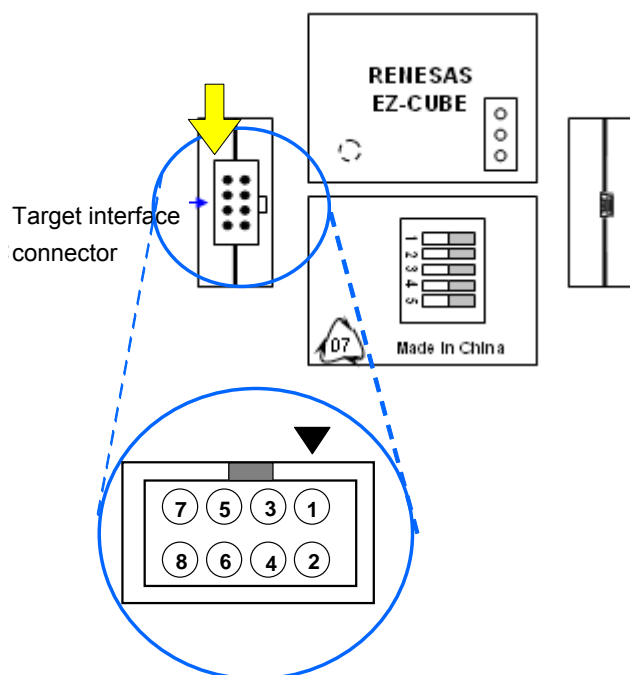


Table 7-3. Pin Functions

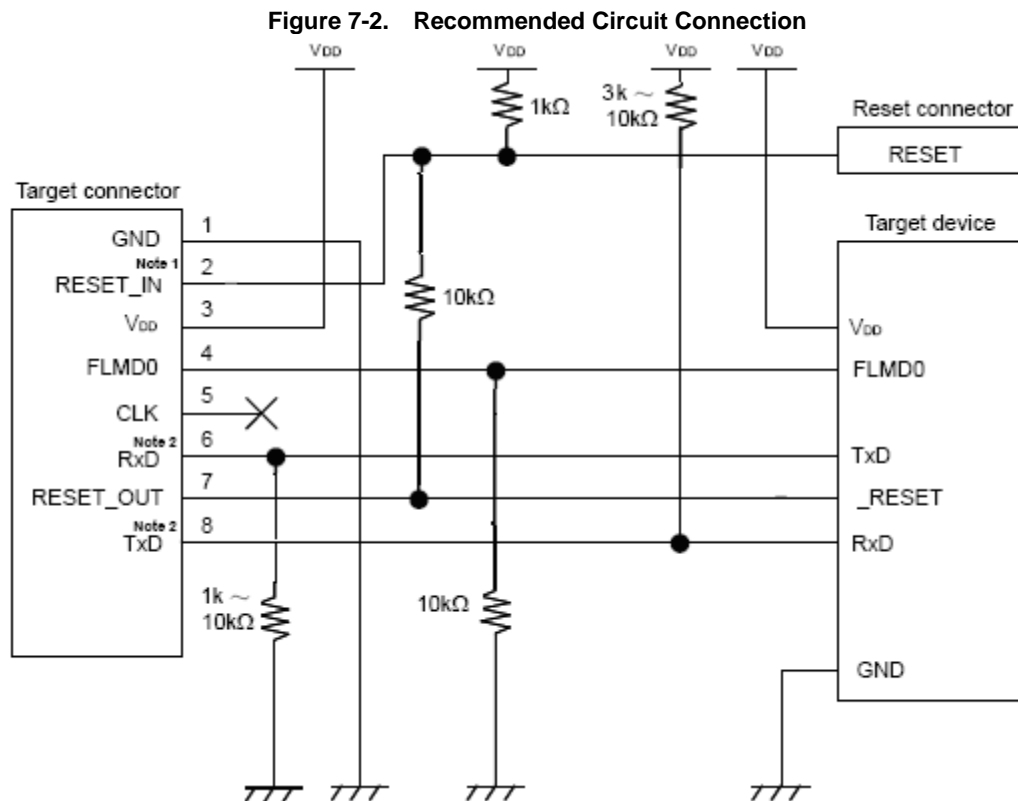
Pin Name	IN/OUT <sup>Note</sup>	Description
RESET_IN	IN	Pin used to input reset signal from the target system
RESET_OUT	OUT	Pin used to output reset signal to the target device
CLK	OUT	Pin used to output clock signal to the target device
FLMD0	OUT	Pin used to set the target device to debug mode or programming mode
RxD	IN	Pin used to receive command/data from the target device
TxD	OUT	Pin used to transmit command/data to the target device

**Note** As seen from EZ-CUBE

### 7.1.2 Circuit connection examples

The circuit design on the target system varies depending on the communication interface mode. Refer to the following table and see the relevant circuit connection example.

**Caution** The constants described in the circuit connection example are reference values. If you perform flash programming aiming at mass production, thoroughly evaluate whether the specifications of the target device are satisfied.




- Notes**
1. This connection is designed assuming that the RESET signal is output from the N-ch open-drain buffer (output resistance: 100Ω or less). For details, refer to **7.1.3 Connection of reset pin**.
  2. Connect TxD (transmit side) of the target device to RxD (receive side) of the target connector, and TxD (transmit side) of the target connector to RxD (receive side) of the target device. Read the serial interface

pin names on the target device side as those for flash programming supported by the target device.

### EZ-CUBE Switch setting

- SW-1: Select switch to "M1".
- SW-2: Select switch to "Int. Clock".
- SW-3: Select switch to "Debug Mode".
- SW-4: Depend on the target system environment.
- SW-5: Select switch to "Other".

 <b>CAUTION</b>	
Notes on the Target System Power Supply:	
	<ol style="list-style-type: none"> <li>1. Do not change the switch setting after connecting USB cable.</li> <li>2. The EZ-CUBE can supply power up to 100 mA current. Do not exceed the maximum supply current. The power is supplied from EZ-CUBE after connecting EZ-CUBE to the host machine.</li> </ol>

### 7.1.3 Connection of reset pin

This section describes the connection of the reset pin, for which special attention must be paid, in circuit connection examples shown in the previous section.

During on-chip debugging, a reset signal from the target system is input to EZ-CUBE, masked, and then output to the target device. Therefore, the reset signal connection varies depending on whether EZ-CUBE is connected.

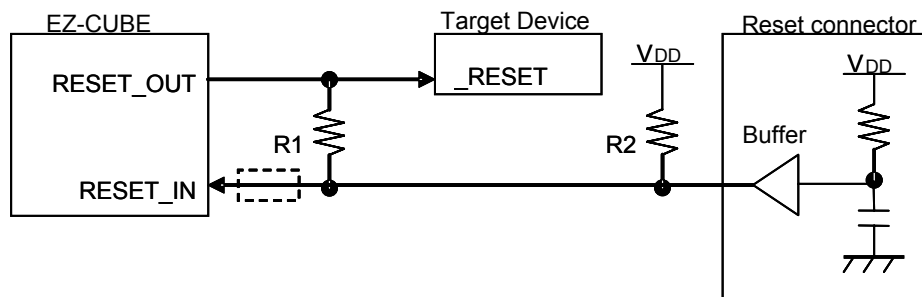
For flash programming, the circuit must be designed so that the reset signals of the target system and EZ-CUBE do not conflict.

**Automatically switching the reset signal via series resistor** (recommended; described in recommended circuit connection in the previous section)

Figure 7-3 illustrates the reset pin connection described in 7.1.2 **Circuit connection examples**.

This connection is designed assuming that the reset circuit on the target system contains an N-ch open-drain buffer (output resistance: 100Ω or less). The VDD or GND level may be unstable when the logic of RESET\_IN/OUT of EZ-CUBE is inverted, so observe the conditions described below in **Remark**.

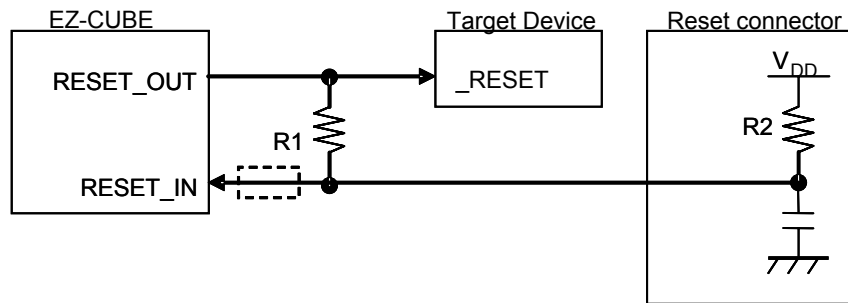
**Figure 7-3. Circuit Connection with Reset Circuit That Contains Buffer**



- Remark** Make the resistance of at least R1 ten times that of R2, R1 being 10 kΩ or more.  
 Pull-up resistor R2 is not required if the buffer of the reset circuit consists of CMOS output.  
 The circuit enclosed by a dashed line is not required when only flash programming is performed.

Figure 7-4 illustrates the circuit connection for the case where the reset circuit on the target system contains no buffers and the reset signal is only generated via resistors or capacitors. Design the circuit, observing the conditions described below in **Remark**.

Figure 7-4. Circuit Connection with Reset Circuit That Contains No Buffers



**Remark** Make the resistance of at least R1 ten times that of R2, R1 being 10 k $\Omega$  or more.  
The circuit enclosed by a dashed line is not required when only flash programming is performed.



## 7.2 On-Chip Debugging

This section describes the system configuration, startup/shutdown procedure and cautions for debugging when on-chip debugging is performed with EZ-CUBE.

### 7.2.1 Debug functions

Table 7-4 lists the debug functions when a V850 microcontroller is the target device and the debugger is used.

**Table 7-4. Debug Functions**

Functions	Specifications
Target Device	V850 V850ES/Jx3, V850ES/Jx3-L
Security	10-byte ID code authentication
Download	Available
Execution	Go, Step In, Step Over, CPU Reset, Restart
Hardware break	1 point
Software break	Multiple points
RAM monitoring	Available
Function pins used for debugging	RXD, TXD

### 7.2.2 Securing of user resources and setting of security ID

The user must prepare the following to perform communication between EZ-CUBE and the target device and implement each debug function. Refer to the descriptions on the following pages and set these items in the user program or using the compiler options.

#### (a) Security ID setting

This setting is required to prevent the memory from being read by an unauthorized person. Embed a security ID at addresses 0x70 to 0x79 in the internal flash memory. The debugger starts only when the security ID that is set during debugger startup and the security ID set at addresses 0x70 to 0x79 match.

If bit 7 of address 0x79 is "0", however, debugging is disabled. In such a case, there are no methods to start the debugger. Debugging is mainly disabled for mass-produced devices.

If the user has forgotten the security ID or to enable debugging, erase the flash memory and set the security ID again.

[How to set security ID]

Embed a security ID at addresses 0x70 to 0x79 in the user program.

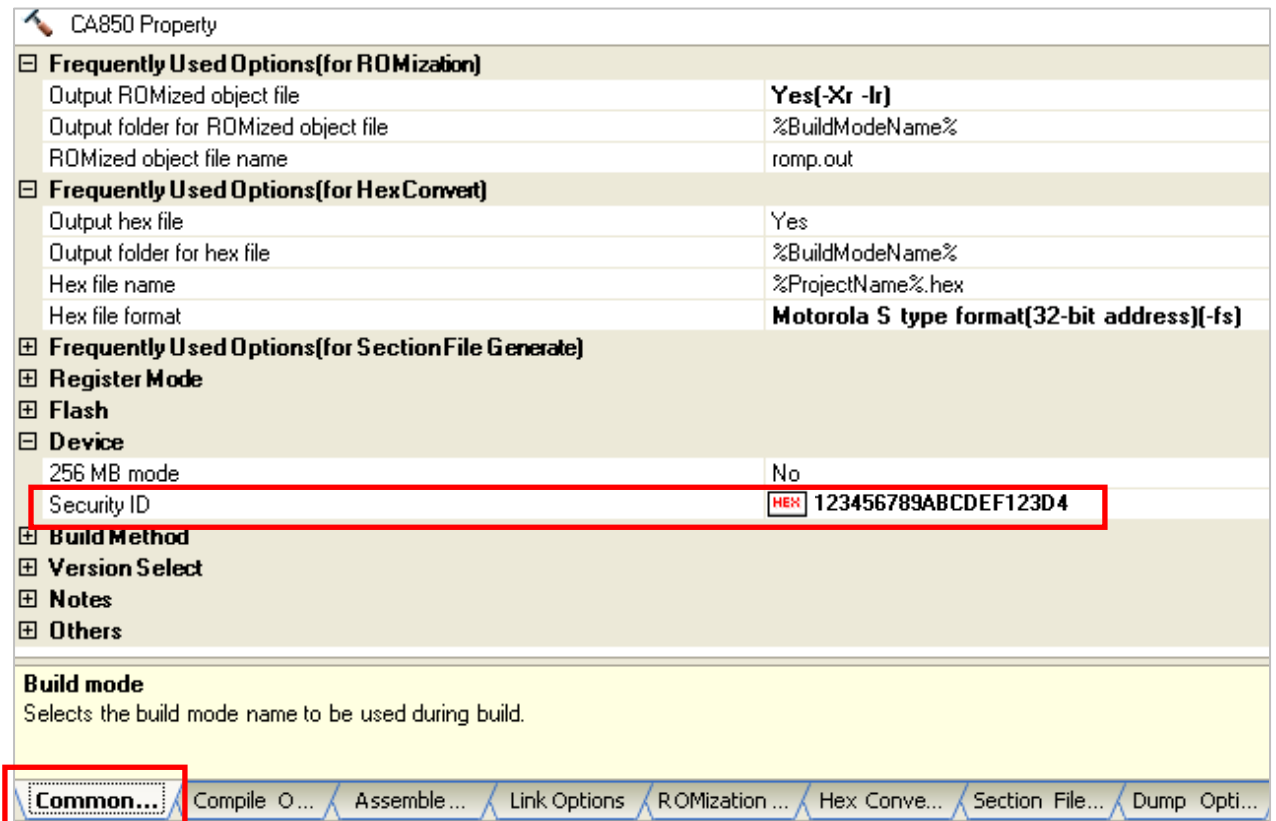
If the security ID is embedded as follows, for example, the security ID set by the debugger is "123456789ABCDEF123D4" (not case-sensitive).

Address	Value
0x70	0x12
0x71	0x34
0x72	0x56
0x76	0x78
0x74	0x9A
0x75	0xBC

0x76	0xDE
0x77	0xF1
0x78	0x23
0x79	0xD4

If Renesas Electronics compiler is used, the security ID can be set using the Compiler Common Options menu.

Figure 7-5. Setting Security ID



#### (b) Reset handler

A reset handler includes the jump instruction for the debug monitor program.

[How to secure areas]

It is not necessary to secure this area intentionally. When downloading a program, however, the debugger rewrites the reset vector in accordance with the following cases. If the rewritten pattern does not match the following cases, the debugger generates an error.

- When two nop instructions are placed in succession from address 0
 

Before writing	→	After writing
0x0 nop		Jumps to debug monitor program at 0x0
0x2 nop		0x4 xxxx
0x4 xxxx		

- When two 0xFFFF are successively placed from address 0 (already erased device)

Before writing		After writing
0x0 0xFFFF	→	Jumps to debug monitor program at 0x0
0x2 0xFFFF		0x4 xxxx
0x4 xxxx		

- The *jr* instruction is placed at address 0 (when using Renesas Electronics compiler CA850)

Before writing		After writing
0x0 jr disp22	→	Jumps to debug monitor program at 0x0
		0x4 jr disp22 - 4

- The jump instruction for the debug monitor program is placed at address 0

Before writing		After writing
Jumps to debug monitor program at 0x0	→	No change

### (c) Securing of area for debugging

The area for debugging is for performing initialization processing for debug communication interface and RUN or break processing for the CPU. The internal ROM area must be filled with 0xFF. This area must not be rewritten by the user program.

[How to secure areas]

It is not necessarily required to secure this area if the user program does not use this area.

To avoid problems that may occur during the debugger startup, however, it is recommended to secure this area in advance, using the compiler. The following shows examples for securing the area, using the Renesas Electronics compiler CA850. Add the assemble source file and link directive code, as shown below.

- Assemble source (Add the following code as an assemble source file.)

```
-- Secures 2 KB space for monitor ROM section
.section "MonitorROM", const
.space 0x800, 0xffNote

-- Secures interrupt vector for debugging
.section "DBG0"
.space 4, 0xff

-- Secures interrupt vector for serial communication
-- Change the section name according to serial communication mode used
.section "INTCSI00"
.space 4, 0xff

-- Secures 16 byte space for monitor RAM section
.section "MonitorRAM", bss
.lcomm monitorramsym, 16, 4 /* defines monitorramsym symbol */
```

**Note** The downloading speed can be increased by replacing this line with the statement "monitorramsym:" to perform a symbol definition only. This effect is not applicable if values are filled into a hole (area without a code). When performing filling, the filling value must be 0xFF for securing the area.

- Link directive (Add the following code to the link directive file.)

The following shows an example when the internal ROM end address is 0x3ffff and internal RAM end

address is 0x3ffeff.

```

MROMSEG : !LOAD ?R V0x03f800{
    MonitorROM      = $PROGBITS      ?A MonitorROM;
};

MRAMSEG : !LOAD ?RW V0x03ffeff0{
    MonitorRAM      = $NOBITS        ?AW MonitorRAM;
};

```

#### (d) Securing of communication serial interface

UART is used for communication between EZ-CUBE and the target system. The settings related to the serial interface modes are performed by the debug monitor program, but if the setting is changed by the user program, a communication error may occur.

To prevent such a problem from occurring, communication serial interface must be secured in the user program.

[How to secure communication serial interface]

Create the user program observing the following points.

- Serial interface registers  
Do not set the registers related to UART in the user program.
- Interrupt mask register  
When UART is used, do not mask receive end interrupts <sup>Note</sup>.  
**Note** Do not mask receive error interrupts.
- Port registers  
When UART is used, do not set port registers to make the TxD and RxD pins invalid.

### 7.2.3 Cautions on debugging

This section describes cautions on performing on-chip debugging for a V850 microcontroller.

Be sure to read the following to use EZ-CUBE properly.

#### (1) Handling of device that was used for debugging

Do not mount a device that was used for debugging on a mass-produced product, because the flash memory was rewritten during debugging and the number of rewrites of the flash memory cannot be guaranteed. Moreover, do not embed the debug monitor program into mass-produced products.

#### (2) Notes on downloading

When debugging, reset CPU before downloading. If DMA transfer to the internal RAM is performed while a program is being downloaded to the flash memory, downloading of the program may not be performed normally. When breaks cannot be executed

#### (3) Regarding ROM correction function

Do not use the ROM correction function or else unexpected breaks will occur.

#### (4) Regarding current consumption

The current consumption in the target device increases during debugging compared with that in normal operation mode, because the OCD unit of the target device operates during debugging.

**(5) Regarding standby release with debugging functions**

In case using the RRM function and DMM function, the standby mode is released when the memory is read or written.

**(6) Notes on flash self programming**

Do not break in ROM area during flash environment. In case of monitoring memory with RRM function, a temporary break is executed. So do not use RRM function when using flash self programming.

Do not modify the debug monitor area when using debugging interface.

**(7) Regarding POC function and emulation of turning OFF**

Make sure that the power to the target system is not shut down during debugging. Regarding to check the operation of POC function and tuning OFF, perform without the emulator. In case the target system is turning OFF instantaneously, the debugger may hang up.

**(8) Regarding I/O buffer when using reset mask**

The I/O buffer (port pin) may enter the reset status depending on the target device when a reset is input from the pin, even if reset is masked by the mask function.

**(9) When forced break, RRM function and DMM function do not operate**

Forced breaks, RRM function and DMM function cannot be executed if one of the following conditions is satisfied.

- Interrupts are disabled (DI)
- Interrupts issued for UART interface are masked
- Standby mode is entered while standby release by a maskable interrupt is prohibited
- When using UART interface, the main clock has been stopped
- When using UART interface, a clock different from the one specified in the debugger is used for communication

**(10) Writing quality of flash programming**

To improve the writing quality, fully understand, verify, and evaluate the following items before using EZ-CUBE emulator.

- Circuits are designed as described in the user's manuals for the device.
- The device and the software are used as described in each user's manual.
- The power supplied to the target system is stable.

**(11) Debugging with real machine running**

If debugging is performed with a real machine running, without using emulator, write the user program using the programming software. Programs downloaded by the debugger include the monitor program, and such a program malfunctions if it is not controlled via EZ-CUBE emulator.

**(12) Regarding watchdog timer**

The watchdog timer is forcibly stopped by the debug monitor program. Therefore, do not use the option byte to specify that the watchdog timer cannot be stopped. For details about the option byte settings, see the user's manual of the target device.

**(13)Regarding external reset**

A break occurs when an external reset occurs (except when resets are masked) or an internal reset occurs.

**(14)Regarding reset vector handling**

Reset vector handling is not supported.

### 7.3 Flash Programming

This section describes the system configuration and startup/shutdown procedure when flash programming is performed for a V850 microcontroller, using EZ-CUBE.

#### 7.3.1 Specifications of programming function

**Table 7-5. Specifications of Programming Function**

Functions	Specifications
Host interface	USB 2.0
Target interface	UART
Target system voltage	2.7 to 5.5 V (depends on the target device)
Clock supply	8MHz clock can be supplied Clock mounted on the target system can be used
Power supply	5 V±0.3V (maximum current rating: 100 mA)
Acquisition of device-specific information	Parameter file for Renesas Electronics is used
Programming software	RFP <sup>Note</sup> (Renesas Flash Programmer)
Security flag setting	Available
Standalone operation	Unavailable (must be connected to host machine)

**Note** For detailed usage of the RFP, refer to the **RFP User's Manual**.

#### 7.3.2 Cautions on flash programming

This section describes the cautions for flash programming. Be sure to read the following for the proper use of EZ-CUBE.

- To improve the writing quality, fully understand, verify, and evaluate the following items before using EZ-CUBE.
  - Circuits are designed as described in the user's manuals for the device and EZ-CUBE.
  - The device, RFP and EZ-CUBE are used as described in each user's manual.
  - The power supplied to the target system is stable.