User's Manual

E2 Emulator, IE850A

Additional Document for User's Manual

(Notes on Connection of RH850/U2B Series)

Supported Devices:

RH850 Family

RH850/U2B Series

Renesas Electronics
www.renesas.com

Rev.2.00 Mar 2023

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

www.renesas.com

Contact information

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

Trademarks

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

Contents

1. Outline

1.1 Features of the E2 and IE850A emulators from Renesas

The E2 emulator (RTE0T00020KCE00000R) and the IE850A (RTE0T0850AKCT00000J) from Renesas are advanced emulators based on the concept of improving efficiency in development.

The E2 emulator is also usable as a flash programming tool (Renesas Flash Programmer). Furthermore, for MCUs of the RH850 family, extended facilities such as a CAN communication time measurement solution are also available in addition to the basic debugging facilities.

The IE850A supports the external tracing (Aurora tracing) interface of MCUs of the RH850 family and has a large-capacity trace memory.

1.2 Configuration of manuals

This manual mainly describes details that depend on the target device and gives notes on usage during the debugging of software on the target device by using an emulator (E2 or IE850A) from Renesas.

Be sure to read this manual along with the user's manuals for the emulator you are using.

Name of Document	Document No.
E2 Emulator RTE0T00020KCE00000R User's Manual	R20UT3538E
IE850A Emulator RTE0T0850AKCT00000J User's Manual	R20UT4461E

Be sure to read the following user's manual when you debug the DFP IP installed in the RH850/U2B.

Name of Document	Document No.
E2 Emulator Additional Document for User's Manual (Setting up Interlocked Debugging of the DFP in an RH850/U2B-Series Device)	R20UT5150E

1.3 Use cases for debugging

Since some devices of the RH850/U2B series include a DFP IP module, there are two use cases for debugging. The supported emulator differ in each case.

Use Case for Debugging	Target Device		Emulator		Remarks
	G4MH	DFP	E2	IE850A	
Use case (1)	Debugged	Not debugged	Supported	Supported	The DFP is handled as a peripheral macro of a G4MH.
Use case (2)	Debugged	Debugged	Supported	Not supported	Debugging the DFP requires a dedicated DFP debugger.

2. Connecting the Emulator and User System

2.1 Connector

To connect the emulator to a user system, a connector must be mounted on the system. The 14-pin connector and 34-pin connector can be used for external tracing. Mount the 34-pin connector if you intend to use external tracing. Table 2-1 shows the three types of connector and supported emulators.

Table 2-1 Connectors and Supported Emulators

		Supported Emulator	
		E2	IE850A
Connector	14-pin connector	Connectable.	Not connectable.
	34-pin connector for external tracing	Not connectable.	Connectable.

2.1.1 14-pin connector

Table 2-2 shows the recommended connectors for connection of the E2 emulator. When other components are mounted around the connector, do not mount components with heights exceeding 10 mm within 5 mm of the connector on the user system as shown in Figure 2-1.

When designing the user system that mounts the 14-pin connector, refer to this section in this manual and the user's manual for the target device.

Table 2-2 Recommended 14-pin Connectors

	Type Number	Manufacturer	Specification
14-pin connector	7614-6002	3M Japan Limited	14-pin straight type (Japan)
	2514-6002	3M Limited	14-pin straight type (other countries)

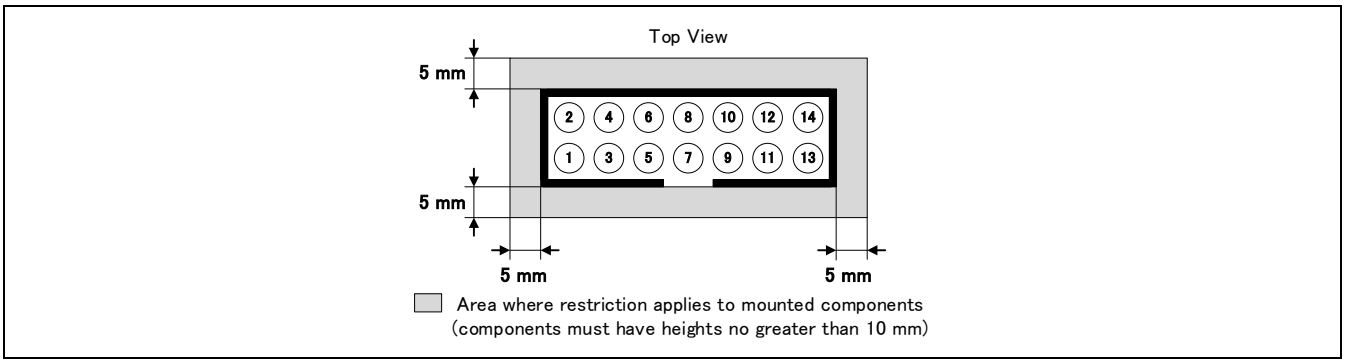

Figure 2-1 Area where Restriction Applies to Mounted Components

To connect an E2 emulator to the connector on the user system, use the connector conversion adapter that comes with the E2 and the user system interface cable. Figure 2-2 shows an example of the connection.

After connecting the user system interface cable to the connector conversion adapter, connect the connector conversion adapter to the connector on the user system.

The connector conversion adapter is provided with a switch. Setting for the switch must be on the "1" side for the RH850. Operation is not guaranteed if the switch is on the "3" side. For setting the switch, refer to Table 2-3.

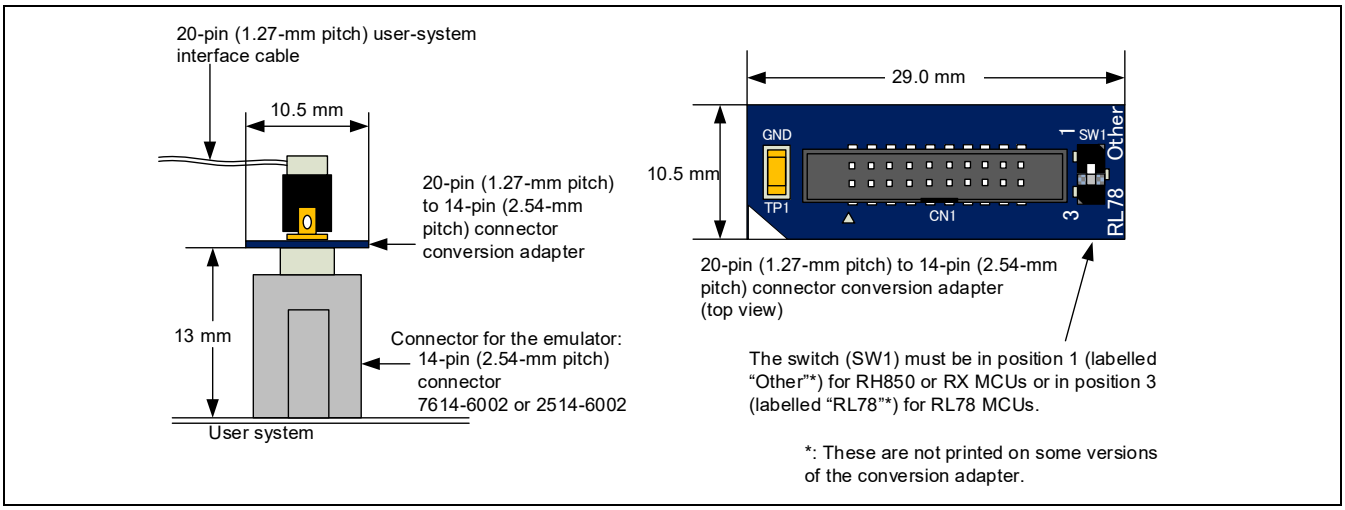

Figure 2-2 Connecting the User System Interface Cable to the 14-pin Connector in the E2 Emulator

Table 2-3 Setting of Switches (SW1)

Setting	Description
1	The target device is an RH850 microcontroller (default setting).
3	The target device is an RL78 microcontroller.

CAUTION

Note on connector insertion and removal (1):

When connecting or disconnecting the user-system interface cable and the connector conversion adapter, grasp both sides of the board of the connector conversion adapter. Pulling the user-system interface cable itself will damage the wiring.

If a 20-pin (1.27-mm pitch) user-system interface cable becomes disconnected, purchase the following discontinued product: RTE0T00020KCAC0000J.

CAUTION

Note on connector insertion and removal (2):

Be aware that the connector conversion adapter must be inserted with the correct orientation. Connecting the connector conversion adapter with the wrong orientation may cause damage.

2.1.2 34-pin connector for external tracing

Table 2-4 shows the 34-pin connector for connection of the emulator for external tracing.

Table 2-4 Recommended 34-pin Connector for External Tracing

	Manufacturer	Type Number
34-pin connector	SAMTEC	ASP-137973-01

When designing a user system on which a 34-pin connector is to be mounted, refer to the IE850A Emulator User's Manual and the user's manual for the target device. Note that a connector mounted on the emulation adapter is for use with the 46-pin connector. Connect the 34-pin to 46-pin conversion adapter that comes with the emulation adapter. For details, refer to the user's manual for the emulation adapter.

2.2 Pin assignments of the connector

2.2.1 14-pin connector

Table 2-5 shows the pin assignments of the 14-pin connector.

Table 2-5 Pin Assignments of the 14-pin Connector

Pin No.	Signal name (#: Low active, -: unused)				I/O (*3)
	Debugging		Programming (RFP)		
	4-pin LPD	JTAG	2-wire UART	CSI	
1	LPDCLK	TCK	-	FPCK	Input
2 (*1)	GND	GND	GND	GND	-
3	LPDRST#	TRST#	-	-	Input
4	FPMD0	FPMD0	FPMD0	FPMD0	Input
5	LPDO	TDO	FPDT	FPDT	Output
6	-	-	FPMD1	FPMD1	Input
7	LPDI(LPDIO)	TDI	FPDR	FPDR	I/O
8	TVDD	TVDD	TVDD	TVDD	-
9	-	TMS	-	-	Input
10 (*2)	EVTO	EVTO	-	-	-
11	LPDCLKO	RDY#	-	-	Output
12 (*1)	GND	GND	GND	GND	-
13	RESET#	RESET#	RESET#	RESET#	Input
14 (*1)	GND	GND	GND	GND	-

Notes: 1. Securely connect pins 2, 12, and 14 of the connector to GND of the user system. These pins are used for electrical GND and to monitor connection with the user system by the E2 emulator.

2. The EVTO pin provides for the output of event signals from the device to the E2 emulator. Although connecting the EVTO pin is not essential, we recommend connecting this pin in advance.
   In some devices, the EVTO pin is not present or is only available as a pin function multiplexed with other functions. When the EVTO pin is a multiplexed pin function and the event output function is to be used, set the EVTO pin so that it will function as the EVTO pin by making the required register settings described in the user's manual for the device.

3. Input and output are defined from the perspective of the target device.

CAUTION

Unused pins:

Do not apply signals from the user system to unused pins. Doing so may damage the pins.

2.3 Examples of recommended connections between the connector and the target device

Different tools are used with the E2 emulator according to whether the purpose is debugging or flash programming. Table 2-6 shows the relationships between the operating modes and connection interfaces.

Table 2-6 Relationships between the Operating Modes and Connection Interfaces

Usage	Tool	Device Mode	Connection Interface
Programming	Renesas Flash Programmer (RFP)	Serial programming mode	2-wire UART or CSI
Debugging	CS+, MULTI (*), or e^2 studio	Normal operation mode or user boot mode	4-pin LPD or JTAG

Note: This refers to the MULTI integrated development environment from Green Hills Software. It is simply referred to as MULTI in the remainder of this document.

This section describes examples of recommended connections between the E2 emulator and the target device. Since there are various examples of recommended connections according to the purpose of the E2 emulator, select the appropriate circuit with reference to Table 2-7. Be sure to take the specifications of the target device as well as measures to prevent noise into consideration when designing your circuit.

Table 2-7 Purpose of the E2 Emulator and the Corresponding Example of Recommended Connections

Purpose	Figure
Both debugging (4-pin LPD or JTAG) and programming (2-wire UART or CSI)	Figure 2-3
Only programming (2-wire UART)	Figure 2-4
Only programming (CSI)	Figure 2-5

2.3.1 Example of recommended connections for both debugging (4-pin LPD or JTAG) and programming (2-wire UART or CSI)

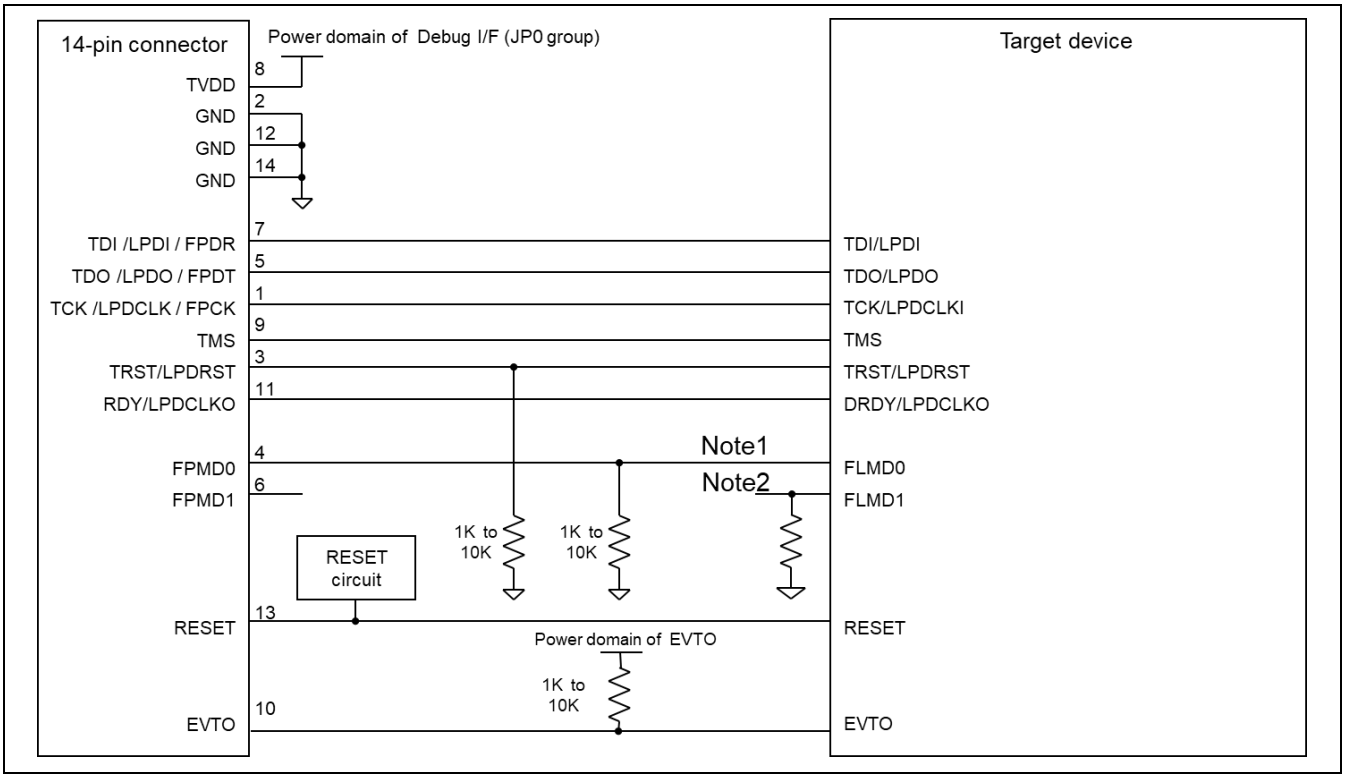

Figure 2-3 Example of Connection

- Note 1: The FLMD0 pin and the debugging interface are in different power domains to those in the target device. However, the interface voltage of the E2 emulator is based on the power supply that is connected to TVDD of the 14-pin connector. Refer to chapter 6, Internal Circuit of the E2 Emulator. Therefore, when the RFP is in use, the E2 emulator drives the FLMD0 pin by TVDD (power domain of the debugging interface). After confirming the specification of the FLMD0 pin of the target device, insert level shifters as required.
- Note 2: The target device must be transferred to serial programming mode when the RFP is in use. Therefore, the E2 emulator outputs the high level on FPMD0 and the low level on FPMD1. Connect FLMD1 and FPMD1 as required after having confirmed the specifications for operation modes of the target device.
- Refer to section 2.3.4, Connecting the RESET pin, for more information on the reset circuit.
- For details on TVDD, refer to section 2.3.5, Connecting the TVDD pin.
- Make wiring runs between the 14-pin connector and target device as short as possible (within 50 mm is recommended). Do not connect the signal lines between the connector and MCU to other signal lines.
- Use GND to apply a guard ring for the wiring which runs between the 14-pin connector and target device. Do not route high-speed signal lines parallel to each other or allow them to cross each other.
- Pin names may vary among target devices. Refer to the user's manual for the target device you are using for the actual pin names.
- Proceed with appropriate processing for pins of target devices which do not require connection to the emulator in accord with the descriptions in "Handling of Unused Pins" in the user's manual for the target device.

CAUTION

Connection of emulators from other manufacturers:

If you use an emulator from other manufactures for debugging, be sure to read its manual beforehand.

2.3.2 Example of recommended connections for only programming (2-wire UART)

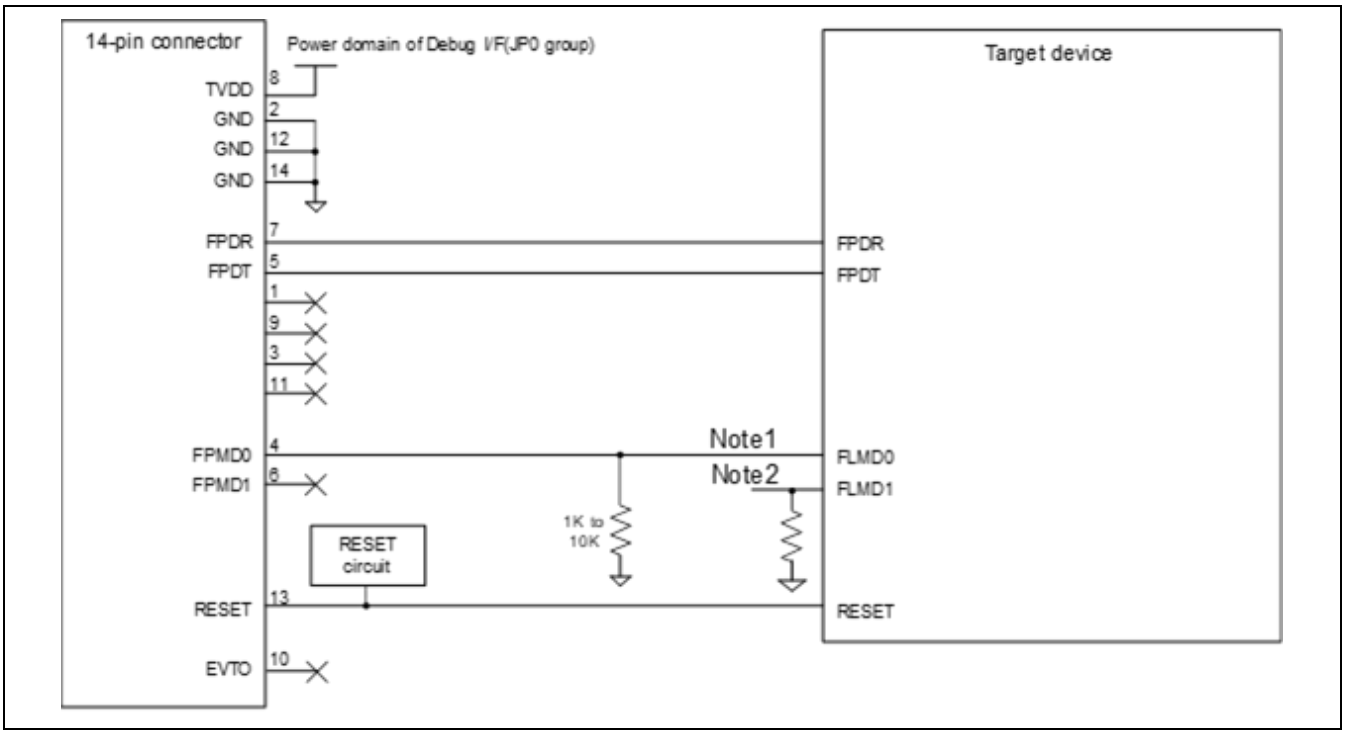

Figure 2-4 Example of Connection

- Note 1: The FLMD0 pin and the debugging interface are in different power domains to those in the target device. However, the interface voltage of the E2 emulator is based on the power supply that is connected to TVDD of the 14-pin connector. Refer to chapter 6, Internal Circuit of the E2 Emulator. Therefore, when the RFP is in use, the E2 emulator drives the FLMD0 pin by TVDD (power domain of the debugging interface). After confirming the specification of the FLMD0 pin of the target device, insert level shifters as required.
- Note 2: The target device must be transferred to serial programming mode when the RFP is in use. Therefore, the E2 emulator outputs the high level on FPMD0 and the low level on FPMD1. Connect FLMD1 and FPMD1 as required after having confirmed the specifications for operation modes of the target device.
- Refer to section 2.3.4, Connecting the RESET pin, for more information on the reset circuit.
- For details on TVDD, refer to section 2.3.5, Connecting the TVDD pin.
- Make wiring runs between the 14-pin connector and target device as short as possible (within 50 mm is recommended). Do not connect the signal lines between the connector and MCU to other signal lines.
- Use GND to apply a guard ring for the wiring which runs between the 14-pin connector and target device. Do not route high-speed signal lines parallel to each other or allow them to cross each other.
- Pin names may vary among target devices. Refer to the user's manual for the target device you are using for the actual pin names.
- Proceed with appropriate processing for pins of target devices which do not require connection to the emulator in accord with the descriptions in "Handling of Unused Pins" in the user's manual for the target device.

2.3.3 Example of recommended connections for only programming (CSI)

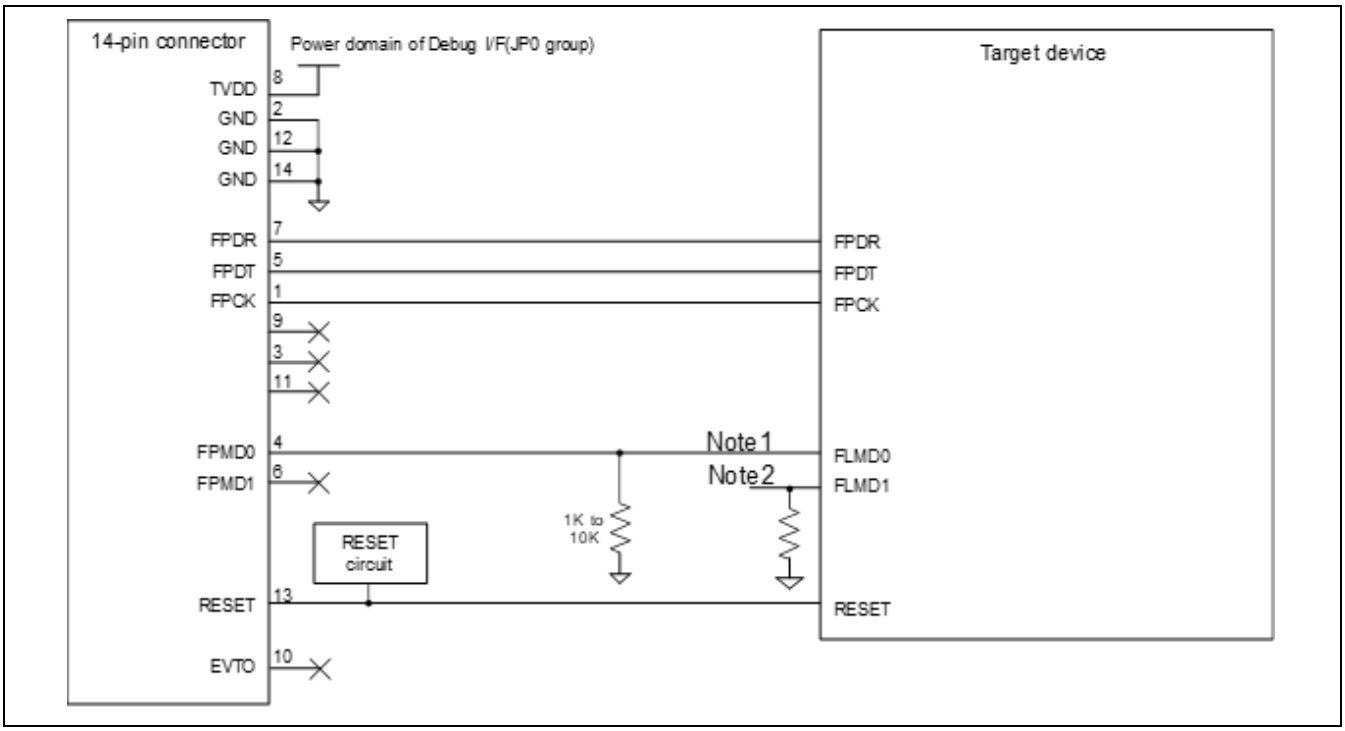

Figure 2-5 Example of Connection

- Note 1: The FLMD0 pin and the debugging interface are in different power domains to those in the target device. However, the interface voltage of the E2 emulator is based on the power supply that is connected to TVDD of the 14-pin connector. Refer to chapter 6, Internal Circuit of the E2 Emulator. Therefore, when the RFP is in use, the E2 emulator drives the FLMD0 pin by TVDD (power domain of the debugging interface). After confirming the specification of the FLMD0 pin of the target device, insert level shifters as required.
- Note 2: The target device must be transferred to serial programming mode when the RFP is in use. Therefore, the E2 emulator outputs the high level on FPMD0 and the low level on FPMD1. Connect FLMD1 and FPMD1 as required after having confirmed the specifications for operation modes of the target device.
- Refer to section 2.3.4, Connecting the RESET pin, for more information on the reset circuit.
- For details on TVDD, refer to section 2.3.5, Connecting the TVDD pin.
- Make wiring runs between the 14-pin connector and target device as short as possible (within 50 mm is recommended). Do not connect the signal lines between the connector and MCU to other signal lines.
- Use GND to apply a guard ring for the wiring which runs between the 14-pin connector and target device. Do not route high-speed signal lines parallel to each other or allow them to cross each other.
- Pin names may vary among target devices. Refer to the user's manual for the target device you are using for the actual pin names.
- Proceed with appropriate processing for pins of target devices which do not require connection to the emulator in accord with the descriptions in "Handling of Unused Pins" in the user's manual for the target device.

2.3.4 Connecting the RESET pin

While you are using the E2 emulator, pin 13 (RESET pin) of the 14-pin connector must be connected to the reset pin of the target device. An example is shown in the figure below.

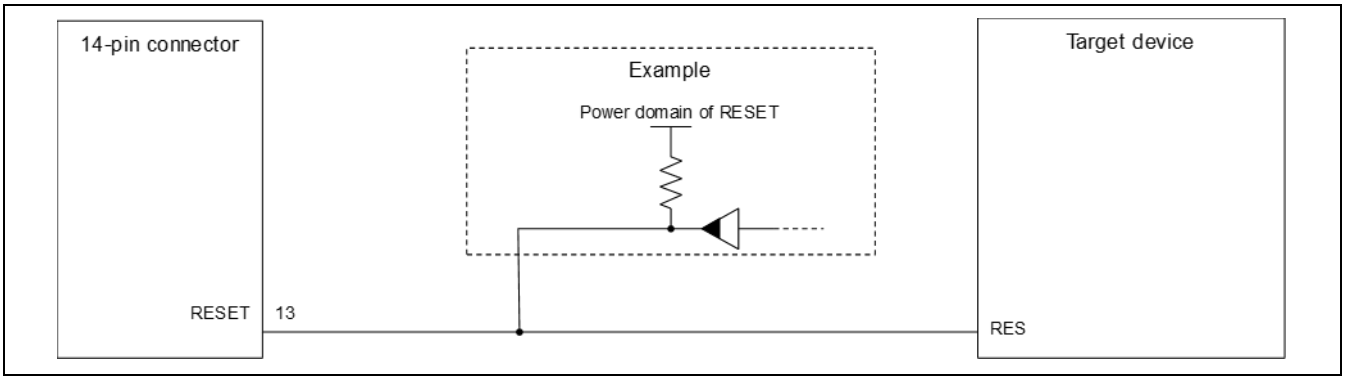

Figure 2-6 Example of Connecting a Reset Circuit

The E2 emulator fixes the RESET pin to the low level before the debugger or RFP is activated. After the debugger or RFP is activated, the emulator either keeps the pin at the low level or places it in the high-impedance state in accord with the operation of the debugger or RFP.

- Output of the reset circuit should be either n-channel open drain or be a signal generated solely by a resistor and capacitor (and possible other components).
- Use the power source for the power domain of the RESET pin of the target device as the destination voltage for pulling up.
- Pin 13 (RESET) of the E2 emulator is pulled up (by a 100-kΩ resistor) within the emulator (refer to chapter 6, Internal Circuit of the E2 Emulator).
- The RESET pin of the target device may be pulled up or down within the device. On this point, refer to the user's manual for the target device.
- The maximum sink current accepted by the RESET pin of the E2 emulator is 2 mA. Select an appropriate pull-up resistance which does not surpass this value.
- Adjust the time constant of the reset circuit so that the time elapsing before the signal reaches 80% of the high level from the low level is within 900 μs.
- When you use hot plug-in, consider installation of a capacitor between the reset signal and GND in order to suppress a noise. In this case, however, the specifications of the time described above must be satisfied.

2.3.5 Connecting the TVDD pin

(1) Power source monitoring function

Be sure to connect the power source for the power domain of the debug interface (JP0 port group) to pin 8 (TVDD pin) of the 14-pin connector.

The power source connected to the TVDD pin provides power to the final stage output buffer and first stage input buffer on the E2 emulator circuit. When the E2 emulator is connected, it will draw current as described below in addition to the current drawn by the user system.

- Approx. 20 mA when TVDD is 3.3 V, and approx. 40 mA when TVDD is 5.0 V

If there is a possibility you will be using hot plug-in connection, be sure to refer to section 2.3.6, Hot plug-in connection.

(2) Power supply function

The E2 emulator can also supply power at 3.3 V or 5.0 V from the TVDD pin to the user system (at a current of up to 200 mA). When using this function, take care of the following points.

— Do not use this function if power is being separately supplied to the user system. Attempting to do so might break the E2 emulator.

— **Do not use this function for a user system which draws a current of 200 mA or more. The E2 emulator or USB interface of the host machine might be broken.**

— Make sure that the supplied voltage is within the voltage range required by the user system.

— If 5.0 V is supplied, the voltage might be lower than 5.0 V by 0.3 V or more depending on the environment of the host machine that is in use.

For power supply from the E2 emulator, precision is not guaranteed. When writing a program that requires reliability, do not use the power supply function of the E2 emulator. Use a stable, separate power supply for the user system. When writing a program for mass production processes, use the Renesas Flash Programmer.

⚠ CAUTION

Turning the Power On/Off:

When supplying power, ensure that there are no shorts between the user system and power circuit. Only connect the E2 after confirming that there are no mismatches of alignment on the user system port connector. Incorrect connection will result in the host machine, the emulator, and the user system emitting smoke or catching fire.

2.3.6 Hot plug-in connection

Hot plug-in connection of the emulator may lead to a momentary drop in the power-supply voltage on the user system. As shown in Figure 2-7, this effect can be reduced by placing a ferrite bead (or inductor) and relatively large capacitor with low equivalent series resistance near the TVDD line of the connector for connection of the emulator. However, this measure will not completely eliminate the voltage drop.

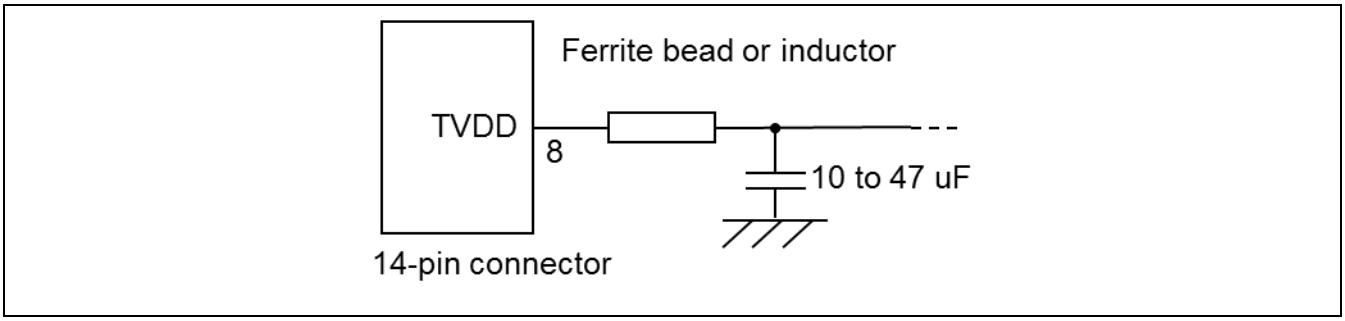

Figure 2-7 Circuit Configuration for Hot Plug-in

Hot plug-in connection can be used with the E2 emulator without the need for a hot plug-in adapter. For details, refer to the E2 Emulator User's Manual.

2.3.7 Isolator

For a debugging environment where there is a difference in potential between the GND of the user system and that of the host PC, use the isolator (R0E000010ACB20) which is separately available from Renesas.

2.3.8 Small connector conversion adapter

A small connector conversion adapter (R0E000010CKZ11) is separately available from Renesas for user system boards which are too small to mount the 14-pin connector that is the standard connector for the E2 emulator. By using the adapter, you can reduce the area taken up by the connector mounted on your system.

However, when you use the small connector conversion adapter, be aware that the pin assignments of the connector differ from those of the standard interface connector for the E2 emulator.

2.4 E2 expansion interface (external trigger input and output)

Using the expansion interface of the E2 emulator (the connector for the interface can be found by removing the cover on which SELF CHECK is printed) enables the input and output of external triggers.

If you will be using the input or output of external triggers, connect the expansion interface (GND: pin 13) on the E2 emulator to the GND on the user system via one of the test leads which are provided with the E2 emulator. Then, according to the pin assignments shown in Figure 2-8, connect the expansion interface of the E2 emulator to the pins on the user system via the test leads.

For details on the expansion interface, refer to the E2 Emulator User's Manual.

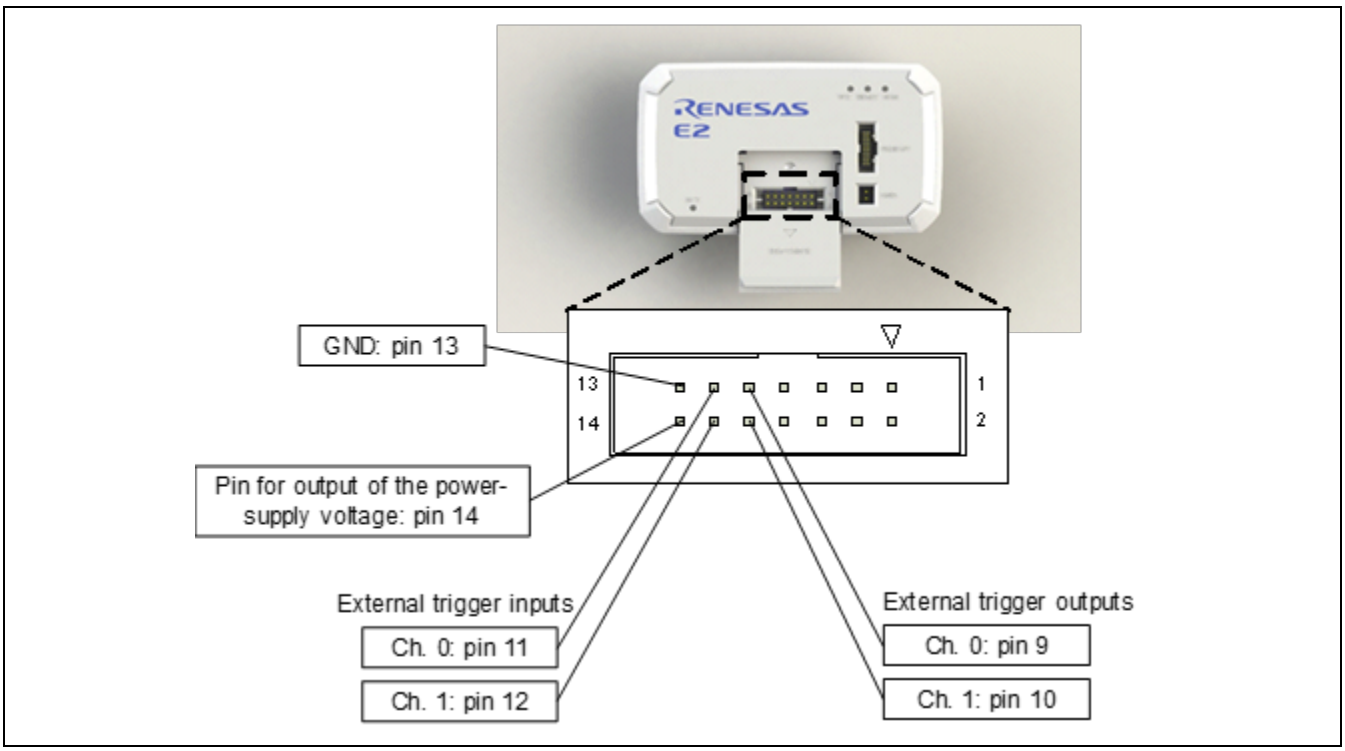

Figure 2-8 Expansion Interface of the E2 Emulator

CAUTION

Expansion interface:

Connect the expansion interface with attention to the range of voltages avoiding conflicts between signals. Incorrect connection may damage the E2 emulator and the user system or result in them emitting smoke or catching fire.

3. Procedure for Connecting the Emulator to the User System

Connect the E2 emulator to the user system and turn the power on and off by following the procedures below. For the procedure for connecting the IE850A to the user system, refer to the IE850A Emulator User's Manual.

3.1 When a separate power supply is used for the user system

<When using the emulator>

(1) Check the power is off.
Check that the user system is turned off.
(2) Connect the user system.
Connect the emulator and the user 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 emulator is turned on by connecting the USB interface cable.
(4) Turn on the user system.
Turn on the user system.
(5) Launch the debugger.
Launch the debugger.

<When finished using the emulator>

(1) Close the debugger.
Close the debugger.
(2) Turn off the user system.
Turn off the user system.
(3) Turn off the emulator and disconnect the emulator.
Disconnect the USB interface cable from the emulator. The emulator is turned off by disconnecting from the USB interface cable.
(4) Disconnect the user system.
Disconnect the user-system interface cable from the user system.

CAUTION

Notes on the User System Power Supply:

While the power of the user system is on, do not turn off the host machine or unplug the USB interface cable.
The user system may be damaged due to leakage current.

3.2 When power is supplied to the user system from the E2 emulator

Do not use the function that supplies the power to the user system from the E2 emulator for a user system which draws a current of 200 mA or more. The E2 emulator or USB interface of the host machine might be broken.

<When using the emulator>

(1) Check the power is off.
Check that the user system is turned off.
(2) Connect the user system.
Connect the emulator and user system with a user-system interface cable.
(3) Connect the host machine and turn on the emulator.
Connect the emulator and host machine with a USB interface cable, then turn on the emulator.
(4) Launch the debugger.
Launch the debugger and select the setting of power supply to the user system.

<When finished using the emulator>

(1) Close the debugger.
Close the debugger.
(2) Turn off the emulator and disconnect the emulator.
Disconnect the USB interface cable from the emulator, then turn off the emulator.
(3) Disconnect the user system.
Disconnect the user-system interface cable from the user system.

4. Functional Overview

4.1 List of functions

Specifications that the E2 emulator and IE850A support are listed in the table below. **Support for some debugging-related functions also depends on the debugger. Refer to the user's manual, etc. for the debugger you are using.**

Table 4-1 Common Functions of the E2 Emulator and IE850A

Parameter		E2 Emulator	IE850A
Connection interface for the flash programming		2-wire UART or CSI	—
Connection interface for the debugging		4-pin LPD 5.5 MHz/11 MHz/16.5 MHz/20 MHz/25 MHz JTAG 6.25 MHz/11 MHz/16.5 MHz/20 MHz/25 MHz/33 MHz	
Interface for external tracing		—	One-lane, two-lane, or four-lane Aurora tracing Two-lane or four-lane Aurora tracing is only available when the emulation adapter is in use.
Hot plug-in connection		Available	
Multi-core debugging	Mode selection	Asynchronous debugging or synchronous debugging	
	Initially stopped debugging	Programs can be executed in the initially stopped state after a reset.	
Break	Software break	In ROM and RAM areas combined: 2000 points	
	Hardware break	12 points including those used for both execution and CPU access conditions (8 points only for execution conditions, and 4 points for either execution or access conditions) Comparison with data in access cannot be used as a condition for a hardware break.	
	Event break	Available (depends on the debugger)	
	Forced break	Available	
	Trace-full break	Available (during tracing and software tracing (with LPD output))	Available (during tracing)
	Trace delay break	—	Available (during external tracing)
	External trigger input break	Available	—
Event	Number of events that can be set	Devices that do not include trace functions: 8 points for execution and 8 points for CPU access Devices that include trace functions or the emulation adapter: 8 points for execution, 8 points for CPU access, 4 points for DMA access, and 4 points for CRAM access Note: The availability to specify a range of events for execution depends on the target device and its version number.	
	Available function	Break, performance measurement, trace	
	Combination of events	OR, sequential	

Parameter			E2 Emulator	IE850A
Tracing (when you are using a device that includes trace functions, or the emulation adapter)		Output destination for traced data (storage capacity)	• Internal trace memory (size of the trace RAM of the target device) • Global Emulation RAM	• Trace memory of the IE850A (up to 9 GB) • Internal trace memory (size of the trace RAM of the target device) • Global Emulation RAM
		Traced data	Branches, cycles of data access, cycles of DMA access, cycles of CRAM access, software trace, and GTM trace	
		Conditions to start and stop recording of data	Stopping of program execution, the setting of event condition and tracing methods	
		Priority	Realtime trace mode (speed is given priority) Non-realtime trace mode (comprehensive data collection is given priority)	
		Tracing methods	Full stop, full break, overwriting (ring buffer), delay stop and halting tracing due to the input of an external trigger of E2 emulator	Full stop, full break, overwriting (ring buffer), delay stop and delay break
Perfor-mance mea-sure-ment	Time (1)	Measurement section	From run to break	
		Item measured	Execution time*	
		Performance	32-bit counters	
	Time (2)	Measurement section	From run to break, or between two event points	
		Items measured	Execution time, total execution time, pass count, maximum execution time, minimum execution time*	
		Performance	32-bit counters (for seven sections per device)	
	Time (3) (when you are using a device that includes trace functions, or the emulation adapter)	Items measured	Number of instructions executed (all or branches only), number of interrupts accepted, etc.	
		Measurement section	From run to break, or between two event points	
		Items measured	Maximum value, minimum value, latest value, total value, pass count	
		Performance	32-bit counters (for eight sections per device)	
Pseudo real-time RAM monitoring Direct memory modification			Available (occupies a bus (steals cycles)) Note: Only available for the general local RAM, cluster RAM, H-Bus, P-Bus, I-Bus, and CPU peripheral areas.	
Reset masking			Available (for selecting whether resets are masked or not during the execution of a program)	
Peripheral breaks			Available	

Parameter	E2 Emulator	IE850A
Emulator detection by user programs	Available Any 32-bit value which is debugging information from the debugger is specified and held in the debugging startup register while the emulator is connected. This function can be used to determine the state of the emulator being connected or not from within user programs. Debugging startup register (DBGIFR0) Initial value: 0000 0000H Address: FF0B 0160H	
Download function	Available	
Security ID authentication	Available	
Debugging when ICU-M is enabled	Available	
Main-core debugging	Available	
Debugging of the virtualization facility	Available	
Software tracing (LPD output)	Available (Refer to Table 4-2.)	—
External trigger input/output (E2 expansion interface)	Available (Refer to Table 4-2.)	—
GTM debugging	Available (Refer to Table 4-5.)	Available (Refer to Table 4-5.)
DFP debugging	Available	—

Note: The resolution of the measured times depends on the interface used for the connection (e.g., 90.9-nsec resolution for a 4-pin LPD connection running at 11 MHz).

Table 4-2 Functions Specific to the E2 Emulator

Parameter		E2 Emulator
Software tracing (LPD output) (Refer to section 0.)	Condition of the debugging mode	Only available in the synchronous debugging mode. Not available in GTM debugging.
	Target CPU	Selection of a single CPU. (When the debugger is connected to the emulator, a single target CPU is selected. If the target CPU is changed, the debugger must be re-connected to the emulator.)
	Destination for storage	"E2 storage": Memory for storage in the E2 emulator
	Internal buffer	Eight stages*
	Traced data	Software trace data + timestamps (given by the E2 emulator) Resolution: 8.333 ns, maximum 27 days
	Conditions to start and stop recording of data	Starting and stopping of program execution (breaks)
	Priority of trace acquisition	Real-time trace mode (priority given to speed) Non-real-time trace mode (priority given to data)
	Recording of trace memory	Ring mode (overwriting mode) Trace-full stop mode Trace-full break mode
External trigger input/output	Input signal channels	E2 expansion interface: 2 (ch. 0: pin 11, ch. 1: pin 12)
	Output signal channels	E2 expansion interface: 2 (ch. 0: pin 9, ch. 1: pin 10)
	Interface voltage	When the emulator is not supplying power to the user system: TVDD voltage or any voltage between 1.8 V to 5.0 V When the emulator is supplying power to the user system: Voltage being supplied to the user system
	Conditions for detection of trigger inputs	Edge detection (rising, falling, or both edges) Level detection (low or high)
	Operation when a trigger is input	When software tracing (LPD output) is in use: Break When software tracing (LPD output) is not in use: Break or stopping of recording in internal trace memory
	Condition for detection of trigger outputs	Break detection
	Operation when a trigger is output	Output of a low or high pulse (for from 1 μsec to 65535 μsec) can be specified.

Note: The output of the combination of a PC value and the corresponding immediate or register value uses one stage of the internal buffer. When software tracing data have been stored up to the seventh stage of the internal buffer, an overflow message is stored in the eighth stage.

4.2 Software tracing function

Devices of the RH850 family support debugging instructions for the output of software trace data. Table 4-3 shows the output destinations of software trace data and supported emulators.

Table 4-3 Software Tracing and Supported Emulators

Output Destination of Software Trace Data	Supported Emulator
Internal trace memory (trace RAM of the target device)	E2 emulator and IE850A
External tracing (Aurora tracing) interface	IE850A
LPD output	E2 emulator

Unlike conventional tracing, the software tracing function does not cater for the setting of events or conditions so that trace data are output when the settings match the results of program execution; instead, this function helps the user to embed debugging instructions in the program to be executed as checkpoints or for the purpose of the output of specific information or register values and output of the history execution as trace data. Make use of this function as a new way of debugging.

The E2 emulator provides trace data for display and supports LPD output. We also provide several types of helpful solution functions for the E2. For details, refer to the relevant application notes on the following Web site.

https://www.renesas.com/us/en/products/software-tools/tools/emulator/e2.html

For details of the debugging instructions and the numbers of clock cycles required to execute them, refer to the RH850 User's Manual: Debugging Instructions. Table 4-4 gives an overview of these instructions.

Note that the debugging instructions embedded in a program do not affect the CPU internally and there is no output of software trace data unless an emulator is connected.

Table 4-4 Debugging Instructions for Software Tracing

Debugging Instruction	Function
DBCP	Outputs the current PC value as software trace data.
DBTAG imm10	Outputs a 10-bit immediate (imm10) value as software trace data. Output of the PC value is also selectable.
DBPUSH rh-rt (General-purpose registers are specified as rh ≤ rt (in ascending order).)	Outputs the register numbers and values of general-purpose registers from rh to rt as software trace data. Output of the PC value is also selectable.

4.3 GTM debugging function

Specifications for debugging the GTM are listed in the table below. The table only shows functions for the GTM. The functions for the CPU shown in Table 4-1, except for downloading, can be used at the same time as those for the GTM. **Support for some debugging-related functions also depends on the debugger. Refer to the user's manual, etc. for the debugger you are using.**

Table 4-5 GTM Debugging Functions

Parameter		E2 Emulator	IE850A
Break	Software break	—	
	Hardware break	—	
	Event break	Available (depends on the debugger)	
	Forced break	Available	
	Trace-full break	Available	Available
	Trace delay break	—	Available (during external tracing)
	External trigger input break	Available	—
Event	Number of events that can be set	MCS: 2 points for execution and 2 points for data access ARU, ATOM, TIM, and TBU: 2 points for data access	
	Available function	Break, trace	
	Combination of events	—	
Tracing (when you are using a device that includes trace functions, or the emulation adapter)	Output destination for traced data (storage capacity)	• Internal trace memory (size of the trace RAM of the target device) • Global Emulation RAM	• Trace memory of the IE850A (up to 9 GB) • Internal trace memory (size of the trace RAM of the target device) • Global Emulation RAM
	Traced data	MCS: Branches and cycles of data access ARU, ATOM, TOM, TIM, and DPLL: Cycles of data access	
	Conditions to start and stop recording of data	Executing or stopping of program execution, the setting of event condition and tracing methods	
	Priority	Realtime trace mode (speed is given priority) Non-realtime trace mode (comprehensive data collection is given priority)	
	Tracing methods	Full stop, overwriting (ring buffer), and delay stop	Full stop, full break, overwriting (ring buffer), delay stop and delay break
Download function		Not available (downloading from within the user program)	

5. Notes on Usage

Cautionary notes on using the E2 emulator and the IE850A are given below.

Conventions:

[E2]: Only applicable to the E2 emulator.

[IE850A]: Only applicable to the IE850A.

Otherwise, the notes apply in common to the E2 emulator and the IE850A.

5.1 General cautionary notes

5.1.1 Handling of devices which were used for debugging

Do not use devices that were used for debugging in mass-production. This is because writing to the flash memory of such devices has already proceeded during debugging, so we cannot guarantee the number of times rewriting of the flash memory can proceed. Debugger errors occur when programming of the flash memory is no longer possible. Replace the device in such situations.

5.1.2 Quality of flash programming

To improve the quality, follow the guidelines below.

- Circuits are designed as described in the user's manuals for the MCU and emulator.
- The MCU, emulator, and software are used as described in respective user's manuals.
- The supply of power to the user system is stable.

5.2 Notes on connecting the emulator

5.2.1 Entering the ID code

To start debugging after the emulator has been connected, the OCD ID, customer ID A, and data flash ID must be authenticated. Authentication of other ID codes may also be required (refer to section 5.4.3).

Enter the ID codes that have been set in the Security Setting area in the following order from the debugger when the emulator is connected.

ID7 (ID[255:224]), ID6 (ID[223:192]), ID5 (ID[191:160]), ID4 (ID[159:128]),

ID3 (ID[127:96]), ID2 (ID[95:64]), ID1 (ID[63:32]), ID0 (ID[31:0])

The ID codes that have been set in the Security Setting area matching those entered from the debugger means success in authentication, so debugging can be started.

IDs of the target device are written in debuggers as shown below.

ID of the Target Device	Notation for CS+	Notation for e^{2}studio	Notation for MULTI
OCD ID	OCD ID	OCD ID (HEX)	-id
Customer ID A	Customer ID	Customer ID (HEX)	-csid
Data Flash ID	Data Flash ID	Data Flash (HEX)	-dfid
Serial Programmer ID	Serial Programmer ID	Serial Programmer ID	-spid
C-TEST ID	C-TEST ID	C-TEST ID	-ctid
RHSIF ID	ID0	ID0	-optid0
Customer ID B	ID1	ID1	-optid1
Customer ID C	ID2	ID2	-optid2

5.2.2 Setting of the OIDDIS bit of S_OPBT0

It is not possible to start debugging if the setting of the OIDDIS bit of S_OPBT0 is 0 (disabling ID authentication).

5.2.3 Setting of the MOSC_FREQ [2:0] bit of OPBT10

Set the MOSC_FREQ [2:0] bit of OPBT10 according to the input frequency of the main OSC. When changing the MOSC_FREQ [2:0] bit, download it only to the Configuration Setting area.

5.2.4 Cases where connecting the debugger is not possible

It is not possible to start debugging if the target device is in any of the following states.

- Reset input state (except for a reset input from the emulator)
- Cyclic run and cyclic stop modes (only applicable to hot plug-in connection)
- Power-off standby mode

5.2.5 Setting the SVR parameters

When the debugger is started for the first time after the power of the target system that is connected to the emulator is turned from off to on, desired SVR parameters can be transferred to the SVR controller from the debugger by using the SVR parameter setting function. In such cases, the SVR controller works with the SVR parameters (SVRCFG0 to SVRCFG9) that have been set by the debugger instead of the values of OPBT16 to OPBT23, OPBT25, and OPBT28. Note that the SVR parameters set by the debugger will not have been written to OPBT16 to OPBT23, OPBT25, and OPBT28. If writing of the parameters to these option bytes is required, this must be done by using the download function to write to OPBT16 to OPBT23, OPBT25, and OPBT28. To enable the values written to OPBT16 to OPBT23, OPBT25, and OPBT28 instead of using the SVR parameters, turn the power of the target system on and start the debugger without using its SVR parameter setting function.

5.2.6 Setting for connection through the external tracing (Aurora tracing) interface

When the external tracing (Aurora tracing) interface supported by MCUs of the RH850 family is to be used, set the number of lanes to suit the number of Aurora lanes of the target device to be connected.

- Connecting the actual device: one lane
- Connecting the emulation adapter: one, two, or four lanes

Use a transfer rate within the range of the specifications after having confirmed the specifications of the target device.

5.3 Notes on differences in operation between the actual device and the emulator

5.3.1 Serial programming function

The serial programming function cannot be used with the emulator during debugging.

5.3.2 Current drawn

The target device draws more current when an emulator is connected than when it is not. That is, the target device consumes more power while it is connected an emulator than while it is not, since the debugging circuit is operating.

5.3.3 Multiplexed pins for the debugging interface

The multiplexed pin functions for a debugging interface cannot be used during debugging.

5.3.4 Initialization of RAM areas

Be sure to initialize RAM from within the user program. If RAM is not initialized by the user program, a program may operate incorrectly when the device is not connected to the emulator but operate normally when the emulator is connected.

If any setting is made to initialize the RAM area when the emulator is connected, the debugger initializes local RAM and cluster RAM areas to 0000 0000H. This leads to the following differences between the cases where the emulator is and is not connected.

- The initial values in the RAM area immediately after starting the emulator are different from the initial values (which are undefined) of the device.
- ECC errors due to non-initialization of the RAM are not detected with the emulator connected.

To emulate ECC errors, make the setting not to initialize the RAM area when the emulator is connected. However, if the setting is for not initializing the RAM area when the emulator is connected, the following functions are not available in the [Memory] window.

- Downloading to on-chip flash memory
- Changes to on-chip flash memory by using the [Memory] panel or the [Disassemble] panel
- Setting of software breaks

ECC errors occur when the RAM area is displayed in the [Memory] window before the RAM area is initialized by the user program.

5.3.5 Executing a program in the initially stopped state after a reset

When a reset is applied while a debugger is connected, it forcibly releases all CPUs from their initially stopped states and places them in the break state, regardless of whether the debugger is in the synchronous debugging mode or asynchronous debugging mode. If a program is executed in this state, note that the target device may not operate normally.

Using the [Debug initial stop state] property (for CS+) and the FETCHSTOP command (for MULTI) enables execution of a program with the CPUs in the initially stopped state. For details of the procedure, refer to the user's manual for the debugger.

In addition, when an initially stopped core or a core on standby is to be debugged and operation that is close to that of the actual device is required, set the [Debug the initial stop state and the standby mode] property (for CS+) or the initstop boot-up option (for MULTI) to enable this. In this case, an initially stopped core will stay in the initially stopped state following release from a reset and applications that include use of the standby mode

of devices can be synchronously debugged. For details of the method, refer to the relevant application note (R20AN0558EJ0100).

5.3.6 Setting the STMSEL1 bit of OPBT3

When STMSEL1 is set to 1 and the emulator is not connected (TRST/LPDRST = low), the device enters the serial programming mode. When the emulator is connected (TRST/LPDRST = high), the device enters the normal operating mode or user boot mode (for details, refer to the user's manual for the target device). When STMSEL1 is set to 1 and the emulator is not connected, the device enters the serial programming mode and does not start execution of the user program.

5.3.7 Power off standby mode

The emulator does not support emulation of power off standby mode. Check the operation of power off standby mode when the emulator is not connected.

5.3.8 BIST

BIST is skipped when the emulator is connected (TRST/LPDRST = high).

5.3.9 Reading registers with undefined initial values and an ECM error

Reading a register with undefined initial values, such as the EIPC and general-purpose registers when they have not been initialized, leads to different values being read by the master CPU and checker CPU, which is detected as an error by the error control module (ECM).

After the debugger is started (exclusive of the case of a hot plug-in connection), the debugger internally initializes the EIPC and general-purpose registers. Thus, if you have forgotten to initialize these registers in the user program, there is a difference between the connected and non-connected states in that an ECM error does not occur when the emulator is connected but does occur when the emulator is not connected.

Be sure to initialize and read registers with undefined initial values in the user program.

The debugger saves the EIPC and general-purpose registers whenever a break occurs and restores the saved values before execution of the user program is restarted. If a break occurs when these registers have not been initialized, an ECM error will occur due to processing for saving by the debugger.

5.3.10 VMON reset

When an emulator is connected (TRST/LPDRST = high), the VMON reset cannot be emulated.

5.4 Cautionary notes on debugging

5.4.1 Power to the user system while debugging

Do not turn the power to the user system off during debugging. Doing so will require reconnection of the debugger.

5.4.2 OTP flag

Do not set the one-time programming (OTP) flag in self-programming with the emulator. Setting the flag makes further debugging impossible.

The OTP flag cannot be set in downloading by the debugger. Note that downloading is suspended if the file to be downloaded includes the OTP setting. We recommend that files to be downloaded are divided into those for the area for setting OTP and for other areas.

5.4.3 Authenticating and downloading ID code

When the debugger is started, it is possible to authenticate IDs other than the OCD ID, customer ID A, and data flash ID, however, authentication of such IDs is invalidated by a user-system reset and a debugger reset. Be sure to access areas protected by these IDs (in terms of downloading to flash memory and reading memory) before issuing a reset after the debugger is started.

In case of downloading to areas including those protected by such IDs after a reset has been issued so that authentication of the IDs has been invalidated, note that programming of the protected areas will fail and subsequent downloading to the protected areas is suspended. We recommend that files to be downloaded are divided into those for the areas protected by IDs and for other areas.

5.4.4 Downloading to Configuration Setting, Security Setting, Block Protection, and Switch areas

Downloading to Configuration Setting, Security Setting, Block Protection, and Switch areas from the debugger is possible. However, after downloading to these areas, continued debugging is not possible. Disconnect and re-connect the emulator on the debugger.

We recommend that files to be downloaded are divided into those for these areas and for the user, user boot, data flash, and extended data areas.

When downloading to Configuration Setting, Security Setting, Block Protection, and Switch areas from the debugger, the data are downloaded to the back side. The debugger changes the back side and front side automatically after the download.

5.4.5 Modifying the PEx_DISABLE bit of OPBT3

Do not modify the PEx_DISABLE bit of OPBT3 by self-programming during execution of a user program. The debugger cannot keep track of the number of cores in operation if this is dynamically altered. If PEx_DISABLE is modified, the debugger cannot continue debugging and must be re-connected.

5.4.6 Debugging for code flash (mirror) area

Instruction execution on code flash (mirror) area is not supported.

5.4.7 Event detection

Write access that has been suppressed in response to an STC instruction failing and read/write access which has been suppressed in response to the inhibition of MDP exceptions may be detected when an access event is set without data comparison but not detected when an access event is set with data comparison.

5.4.8 The order of event detection

In the following cases, since the orders of instructions and event detection may not operate as set, to measure the time or performance in sequential events, section tracing, and desired sections may not be possible.

- Adjacent read and write instructions may be detected as a single access event since multiple instructions are executed at the same time and the timing of event detection differs in write and read access events which are set for consecutive instructions. The timing may be detected in the order of reading then writing, even though the instructions are executed in the order of writing then reading.
- Access events may be detected at the same time since local RAM and cluster RAM can be accessed simultaneously and up to four access events are detectable.

5.4.9 Access trace function

The following restrictions apply to access to trace data.

- When tracing is performed with an access-type point or a range event, data comparison conditions are always ignored regardless of whether or not such conditions have been set and access in which any condition except for a data comparison condition is detected is traced.
- Write access which has been suppressed in response to an STC instruction failing and read/write access which has been suppressed in response to the inhibition of MDP exceptions may not be traced.

5.4.10 Losing trace information

In some cases, acquired trace information will be lost (trace overflow). This depends on the program being executed. The lost information cannot be restored, but the fact of the loss is indicated (displayed).

5.4.11 Recording trace memory during non-realtime tracing

When trace data are stored in internal trace memory and priority in tracing is not given to realtime operation, the functions to stop tracing when the trace memory becomes full (trace-full stop function) and when a specified number of trace messages have been acquired following an event (trace delay-stop function) are not available. To use these functions, give priority to realtime operation.

5.4.12 Performance measurement

In the case of measuring a specific section, if the intervals between the start and the end of one measurement, and between the end of that measurement and the start of the next is short, the measurement is not possible. To obtain correct measurements, the interval* should be long enough.

*: The required detection interval depends on the operating frequency and the LPD communications frequency of the MCU.

5.4.13 Stepped execution of the HALT mode and the HALT instruction

A break leads to release from HALT mode.

When a HALT instruction is encountered during single step execution (execution in units of assembly instruction), a break is set at the next instruction following the HALT instruction, and the mode does not change to the HALT mode. When a HALT instruction is encountered during C-source-level stepped execution, whether or not the transition to the HALT mode proceeds depends on the facilities of the debugger.

5.4.14 Masking of resets while the emulator is in use

Table 5-1 shows the states of a device while the emulator is in use and the operation of resets issued by the user system or the user program (i.e. user system reset). During single stepping, resets are masked to emulate each step in the source code of the program in non-realtime. In C-source-level stepping, resets are masked in different ways depending on the debugger; one method is to use single stepping and another is to set an internal breakpoint and execute the user program. Accordingly, this document cannot define enabling and disabling of reset masking per debugger; refer to the manual for the debugger you are using.

Table 5-1 State of a Device and Masking of Resets

		State of a device			
		In breaks	**In single stepping**	**In user program execution**	**In C-source-level stepping**
Reset mask specification on the debugger	**Resets not masked**	Resets masked		Resets not masked	Depends on the debugger
	Resets masked	Resets masked			

- When a reset is issued by a debugger (by pressing the reset button of the debugger or in some other way), the reset is always enabled regardless of enabling or disabling of reset masking. After a reset is issued by the debugger, breaks are generated for all CPUs.
- Do not issue a pin reset from the user system, regardless of the presence or absence of masking, other than in cases where a user program is running.
- The software reset might not be generated when execution resumes following a break that occurs before a software reset instruction.

5.4.15 Masking of interrupts while the emulator is in use

Table 5-2 shows the states of a device while the emulator is in use and the operation of interrupts (EIINT, FEINT, or FENMI). During single stepping, interrupts are masked to emulate each line of source code of the program in non-realtime. When interrupt processing is to be stepped through, set a breakpoint at the beginning of the interrupt processing and generate an interrupt during execution of the user program. A break will then be generated at the beginning of the interrupt processing. In C-source-level stepping, interrupts are masked in different ways depending on the debugger; one method is to use single stepping and another is to set a temporary breakpoint and execute the user program. Accordingly, this document cannot define enabling and disabling of reset masking per debugger; refer to the manual for the debugger you are using.

Table 5-2 State of a Device and Masking of Interrupts

State of a device			
In breaks	**In single stepping**	**In user program execution**	**In C-source-level stepping**
Interrupt masked*		Interrupt masking disabled (operation according to the specification of the user system)	Depends on the debugger

- An interrupt generated in the state marked (*) in Table 5-2 is kept pending and interrupt processing proceeds after interrupt masking is canceled.

5.4.16 Rewriting of on-chip flash memory (working RAM)

When the debugger performs any operation that involves programming of the flash memory* during a break, part of the internal RAM area is used as a working RAM area. The 4-KB area from the last address of the local RAM (self) area of CPU0 are initially set as the working RAM area.

The debugger can change the working RAM area. After the debugger has saved the values from the working RAM area and rewrites the flash memory, it restores the saved values to the working RAM area. To guarantee the values, it is required to set an area to which there will be no access by the DMAC or any external master to the working RAM area so that operation may continue even if the device enters the break state.

Note: Rewriting of flash memory proceeds in response to any of the operations below.

- Downloading to on-chip flash memory
- Changes to on-chip flash memory by using the [Memory] panel or the [Disassemble] panel
- Setting or cancellation of software breaks
- Re-execution after a software break is encountered (including stepped execution)

5.4.17 Rewriting of on-chip flash memory (modifying the clock settings)

The debugger temporarily changes the clock settings to improve the speed of processing while the flash memory is being rewritten* and restores the previous clock settings after this processing.

If the change in the clock frequency due to the debugger causes a problem, e.g. the frequency surpasses the upper limit which was set by the clock monitor (CLMA), stop the modification of clock settings by using the setting of the [Change the clock to flash writing] property (for CS+) or the FLASHCLOCK command or the -noflashclock option (for MULTI).

Note: Rewriting of flash memory proceeds in response to any of the operations below.

- Downloading to on-chip flash memory
- Changes to on-chip flash memory by using the [Memory] panel or the [Disassemble] panel
- Setting or cancellation of software breaks
- Re-execution after a software break is encountered (including stepped execution)

5.4.18 Breaks during execution of code for making clock settings

The flash memory cannot be programmed if a break occurs while the MCU is running code written to memory for making clock settings (setting of the main oscillator or PLL frequency divider and so on).

If you wish either of the following types of operation to proceed when a break has occurred during clock settings, set [Change the clock to flash writing] in the [Property] panel to [No].

a. Any operation that involves programming of the flash memory (e.g. re-downloading)
b. Setting or deleting software breakpoints

Also, do not set software breakpoints within code for making clock settings.

5.4.19 Software break functions (RAM areas)

The software break function is implemented by replacing instructions. Thus, note that no break will occur if the value at an address where a software break has been set is rewritten by a user program which is running. A break will also not occur if a reset is generated during the execution of a user program, since RAM areas may be initialized by hardware with certain settings.

5.4.20 Cases where no break occurs

No breaks occur when the CPU is in any of states listed below.

- Initially stopped state
- Reset state
- Deep stop mode (in case of synchronous debugging mode)
- Cyclic run mode (in case of synchronous debugging mode)
- Cyclic stop mode (in case of synchronous debugging mode)

If you want to generate a break for a CPU in run mode with a device that includes an initially stopped CPU or for a CPU in cyclic run mode during synchronous debugging, refer to section 5.3.5, Executing a program in the initially stopped state after a reset.

5.4.21 Cautionary point regarding synchronous debugging mode

In the synchronous debugging mode, a break may occur for all CPUs. However, if any of CPUs is in the initially stopped state, note that no break will occur. A breakpoint must be set in the program at a point after all CPUs have been released from the initially stopped state. If you want to generate a break earlier than this, use the asynchronous debugging mode or refer to section 5.3.5, Executing a program in the initially stopped state after a reset.

5.4.22 Cautionary points regarding asynchronous debugging mode

- In the asynchronous debugging mode, peripheral break functions cannot be used. Even if peripheral break functions are enabled, peripheral functions are not stopped.
- In the asynchronous debugging mode, when any of CPUs is in the break state, no user system resets are acceptable.
- During execution of a user program, there may be a case that the ECC error function does not normally operate for flash memory.

  Example: When any CPU accesses flash memory during execution of a user program causing an ECC error and another CPU which is in the break state accesses the same resources in the memory window at the same timing, the debugger temporarily controls the ECC error and no ECC error occurs in any CPU.

5.4.23 Standby mode

To proceed with debugging in standby mode, set WUFMSK0/1_A2 [2] to 0 in the program as a wake-up source signal to be issued in response to a break request. During debugging, power continues to be supplied to the Iso area (CPU, RAM, peripheral modules, etc.) and the PWRCTL pin is kept at the high level in deep stop mode. Thus, after the device is returned to run mode, be sure to initialize the values of RAM or registers for which the initial values are undefined.

5.4.24 Cyclic run mode and cyclic stop mode

When the target device enters the cyclic run mode or cyclic stop mode, the core other than CPU0 is stopped. That is, synchronous debugging is not usable in this situation. Debugging must only proceed as asynchronous debugging or refer to section 5.3.5, Executing a program in the initially stopped state after a reset. Also, since the core other than CPU0 is stopped during asynchronous debugging, it cannot be debugged.

5.4.25 Map mode

The map mode (single map mode or double map mode) specified by OPBT12 must match the map mode that is to be set when the debugger is started.

5.4.26 Cautionary points regarding trace data acquired by software tracing (LPD output) [E2]

Table 5-3 shows the times required for the LPD output of software trace data generated by executing debugging instructions. When this interval follows the execution of a debugging instruction, overflows (losses) of software trace data can be avoided. Even if debugging instructions are executed in shorter intervals than those listed, the device has an internal buffer for tracing and an overflow (a loss of data) will not occur immediately; however, note that an overflow occurs if the internal buffer becomes full. For a DBPUSH instruction, set the total number of registers to less than 5 to avoid an overflow.

Table 5-3 Interval between Execution of the Embedded Instruction and the LPD Output

Debugging Instruction	Interval between Execution of the Embedded Instruction (4-pin LPD (33 MHz))
DBCP	1.727 usec
DBTAG imm10	0.576 usec (without the PC value)
DBPUSH rh-rt	1.727 usec (One register is output without the PC value.)

- When the software tracing (LPD output) is in use, access to memory during the execution of a program, changes to event conditions, the reading of internal trace memory, and the display of state indicators such as STOP are disabled.
- A timestamp indicates the time that the E2 emulator acquires the software tracing data, not the time the instruction in the software being debugged was executed. The E2 emulator requires execution of the program by the MCU to start only after it has started counting its timestamp values. Since the start of counting of timestamp values cannot be precisely synchronized with the start of program execution, the timestamps which have been added to the software tracing data stored from the head of the E2 storage may include some errors.
- When a break is generated as a forced break, a trace-full break from the E2 storage, or a break due to the input of an external trigger, information from a debugging instruction that was executed immediately before the break will not be stored in the E2 storage.
- When a debugging instruction is executed during single-stepped execution and a software break or hardware break is specified and executed by the debugging instruction, software trace data are not output through the LPD interface.
- When trace acquisition is stopped due to a break generated by a software break, hardware break, event break, or trace-full break from internal trace memory, the history of execution from a DBCP instruction executed in the debugging area is stored as the final trace data in the E2 storage and internal trace memory after the break in execution.

5.4.27 Cautionary points regarding external trigger input/output [E2]

- When external trigger input or external trigger output functions are in use, access to memory during the execution of a program, changes to event conditions, the reading of internal trace memory, and the display of state indicators such as STOP are disabled.
- When the software tracing (LPD output) function is not in use, breaks are not detectable during the 10-μsec period after a program has started to run.

5.4.28 Trace-full break [IE850A]

In external tracing, trace data are stored in the trace memory in the IE850A. The emulator requests a break in execution by the target device before the trace memory becomes full (trace-full break). If the request is not accepted immediately, the output of trace data may continue even after the trace memory is full and data will thus be lost.

5.4.29 Debugging the virtual facility

- If a trace-full break occurs while the ICU-M is enabled even if the setting has been made so that no break should occur in contexts that are non-targets for debugging, a break will occur in those contexts.
- When the software tracing (LPD output) function is to be used, do not set the output of tracing of the host mode of the CPU as a non-target for debugging. If this setting is made, break states will not be detectable.

5.4.30 Cautionary points regarding GTM debugging

- For GTM debugging, the GTM must be enabled by the setting of an option byte.
- If a debugger resets the device during GTM debugging, multiple resets will be internally applied and a clock signal will temporarily be supplied to the GTM.
- The debugger cannot access the GTM area before the clock signal is supplied to the GTM.
- The debugging target is one instance of the MCS. To debug other instances implemented in the MCS, re-connect the debugger.
- The program counter of the MCS cannot be modified.
- If a break occurs during GTM debugging, the value of the program counter of the MCS may be incorrect. In such cases, re-connect the debugger since continued debugging is not possible.

5.4.31 Cautionary point regarding hot plug-in connection

Allowing hot plug-in connection prevents usage of the following debugging functions.

- Initializing RAM areas
- Masking of pins

Accordingly, reading of the RAM areas before the user program has initialized them will lead to an ECC error. Refer to section 5.3.4, Initialization of RAM areas.

During hot plug-in connection, the EIPC and general-purpose registers are not initialized. Accordingly, reading of these registers before the user program has initialized them will lead to an ECM error. Refer to section 5.3.9, Reading registers with undefined initial values and an ECM error.

5.4.32 Cautionary point regarding execution of the user program immediately after hot plug-in connection

During execution of the user program until a break occurs due to a forced break or a CPU reset after hot plug-in connection, only the debugging functions listed below are available.

- Read or write access to the internal RAM area or peripheral I/O registers
- Forced break
- CPU reset

After the break, all debugging functions become available in the same way as for normal starting of the program.

5.4.33 Breakpoints in the code flash P/E mode or data flash P/E mode

During debugging of a user program which makes the target device enter the code flash P/E mode or data flash P/E mode, we recommend using hardware breakpoints rather than software breakpoints.

Since flash memory cannot be programmed while the target device is in the code flash P/E mode or data flash P/E mode, software breakpoints can neither be added to nor deleted from the code flash memory. Accordingly, actually adding or deleting them on the target device is not possible. Only add or delete software breakpoints in the code flash memory after the target device is in a mode other than the code flash P/E mode or data flash P/E mode.

If a break is generated at a software breakpoint in the code flash memory while the target device is in the code flash P/E mode or data flash P/E mode, the break will be generated at the current address (that of the software breakpoint), so attempting to execute the user program will again lead to a break and the program will not run beyond the current address. In such cases, apply a reset.

5.4.34 Software breakpoints in the code flash memory in the cyclic run mode

Since flash memory cannot be programmed while the target device is in the cyclic run mode, software breakpoints can neither be added to nor deleted from the code flash memory. Accordingly, actually adding or deleting them on the target device is not possible. Only add or delete software breakpoints in the code flash memory after the target device is in a mode other than the cyclic run mode.

5.5 Notes on debugging the DFP (for use case for debugging (2))

5.5.1 Settings when using the interlocked DFP resumption facility

When a DFP is to be debugged with the use of the interlocked DFP resumption and halting facility, also stop the peripheral macros. If the peripheral macros are not stopped, the G4MH cores and DFP will be individually debugged without the use of the interlocked resumption and halting facility.

5.5.2 Procedures and operation when using the interlocked DFP resumption facility

When a DFP is to be debugged with the use of the interlocked DFP resumption and halting facility, issue a request for executing programs to the G4MH cores then the DFP to proceed interlocked resumption. If a request for executing programs is not issued to the DFP even if the timeout time has elapsed after issuing the request for executing programs to the G4MH cores, a timeout error will occur and the G4MH cores will not start executing programs.

If a request for executing programs is issued to the DFP before the G4MH cores, interlocked resumption will not proceed with the G4MH cores and DFP and only the DFP will start the execution of a program. However, if hart0 of the DFP has not been activated by the G4MH cores, the DFP will remain disabled.

5.5.3 Execution of a program by the G4MH cores during execution by the DFP

When a DFP is to be debugged with the use of the interlocked DFP resumption and halting facility, if a request for executing programs is issued to the G4MH cores while the DFP is executing a program, an error will occur and the G4MH cores will not execute the programs. Use the interlocked DFP resumption and halting facility by issuing a request for executing programs to the G4MH cores and then, through interlocked resumption, the DFP, while the DFP is halted.

When the interlocked DFP resumption and halting facility is not to be used, debugging individually proceeds for the G4MH cores and DFP and the programs can be executed or stopped regardless of the current states in terms of execution and halting.

5.5.4 Stepped execution when using the interlocked DFP resumption facility

When a DFP is to be debugged with the use of the interlocked DFP resumption and halting facility, stepped execution is not interlocked. When a G4MH core is step-executed, stepped execution only proceeds for the G4MH core. When the DFP is step-executed, stepped execution only proceeds for the DFP.

6. Internal Circuit of the E2 Emulator

The internal interface circuits related to the communications interface between the E2 emulator and user system are shown in Figure 6-1 and Figure 6-2. Please refer to these figures when determining parameters in board design.
The alphabet at the end of the serial No. written on the E2 emulator main unit indicates the product revision.

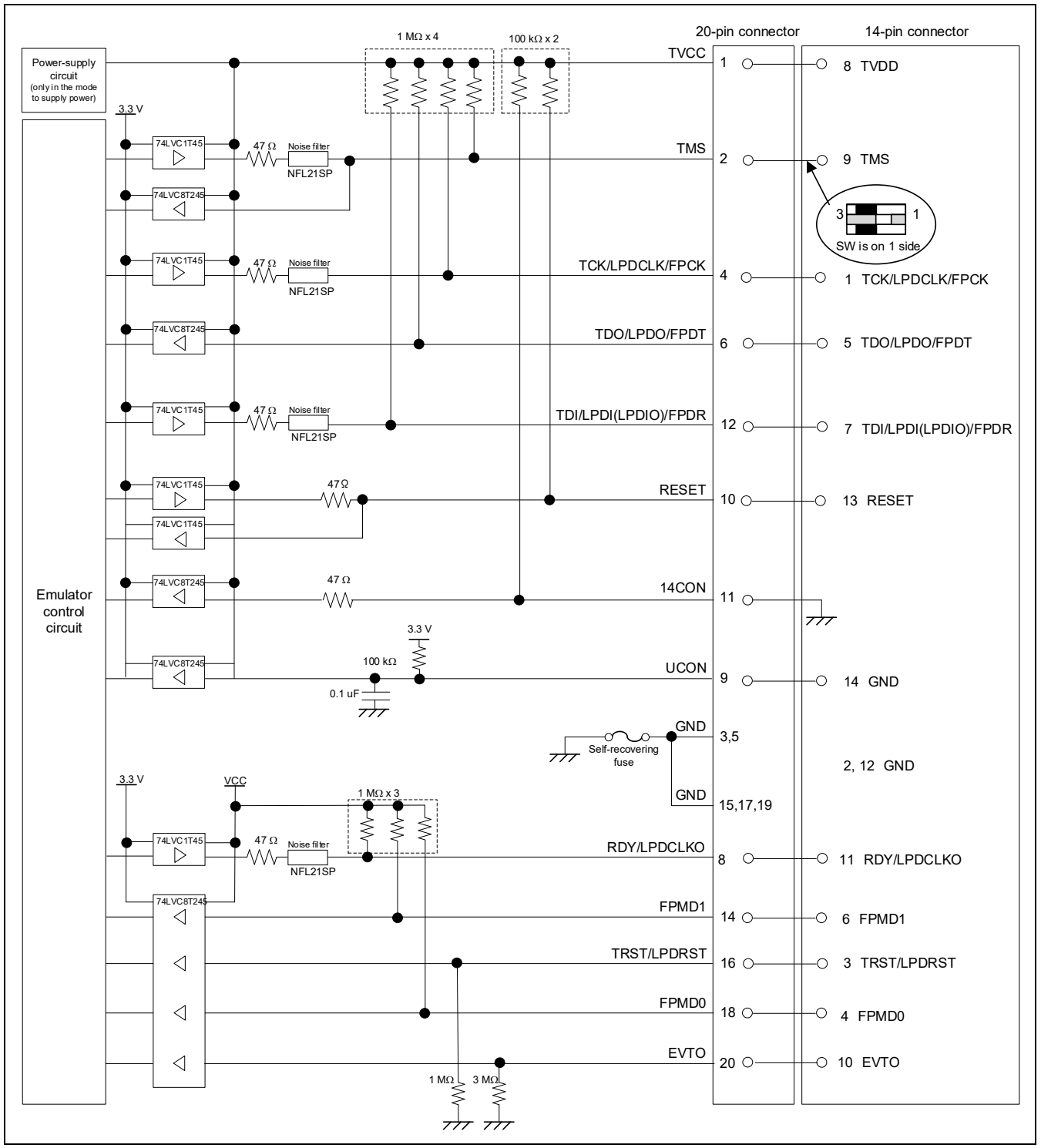

Figure 6-1 Interface Circuits in the E2 Emulator (4-Pin LPD, JTAG, 2-Wire UART, CSI) (Rev. C)

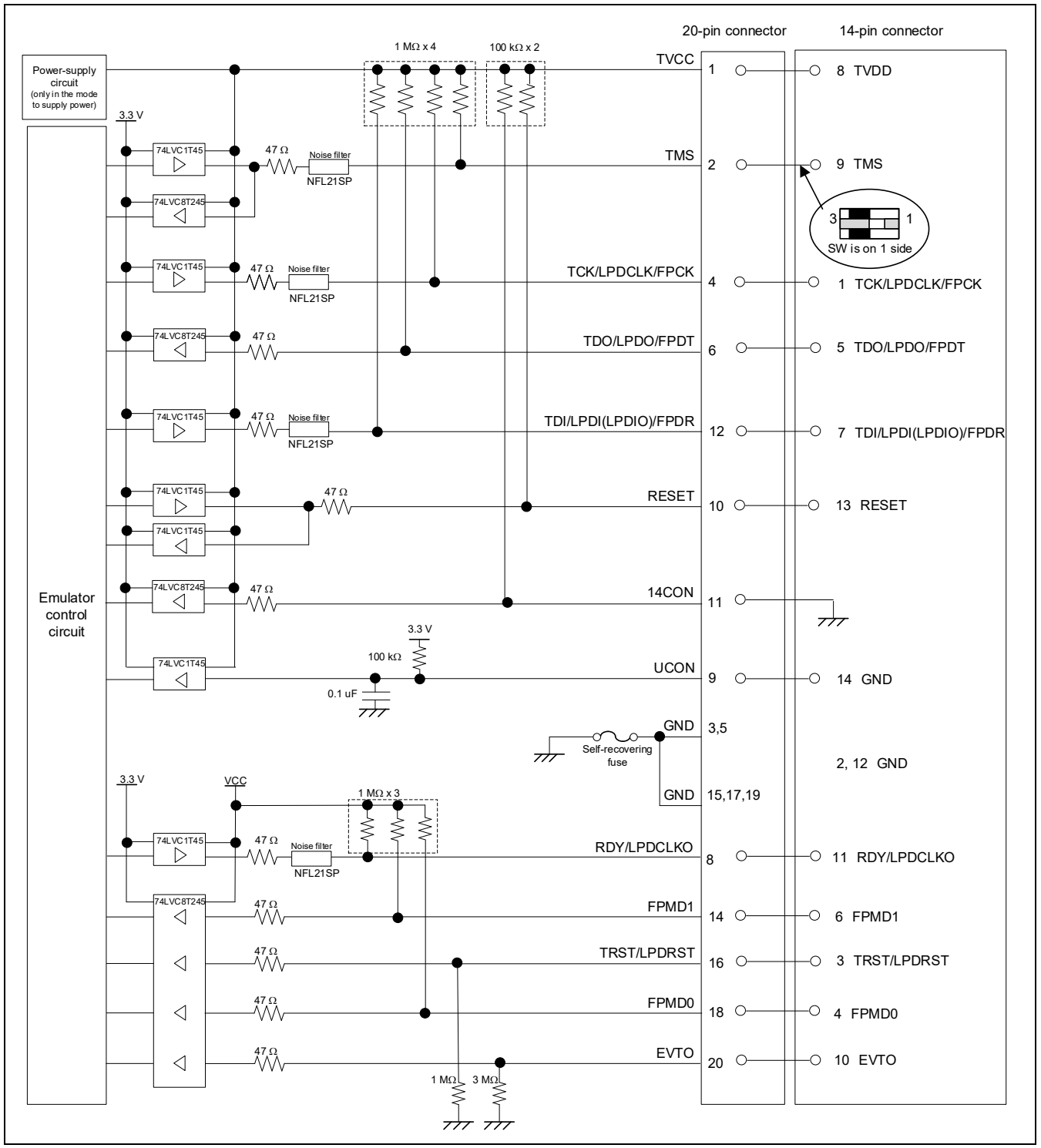

Figure 6-2 Interface Circuits in the E2 Emulator (4-Pin LPD, JTAG, 2-Wire UART, CSI) (Rev. D)

7. Troubleshooting

This chapter gives examples of problems that may arise while the E2 emulator or IE850A is being used in combination with a debugger and of remedies for these problems. Also read the sections of the E2 emulator user's manual or the IE850A emulator user's manual, on the Renesas homepage, and in user's manuals for debuggers which include FAQs or information on troubleshooting. The error codes for CS+ are also listed below. If you are using a debugger other than that of CS+, refer to the user's manual for the given debugger.

7.1 Problems when the emulator is connected

Table 7-1 Problems when the Emulator is Connected (1/2)

Problem	Remedy	Error Code in CS+
Inability to connect with the debugging tool (emulator) Errors in LPD connection	• When the emulator is connected other than with a hot plug-in connection, although the emulator controls the pin reset, this may fail. Check the notes (e.g. the time the signal takes to reach the high level from the low level) given in section 2.3.4 or whether the electrical characteristics requirements of the reset pin of the device are satisfied. • The connection between the emulator and the target device may be wrong. Refer to section 2.3, Examples of recommended connections between the connector and the target device, and check the circuit between the emulator and the target device. • The specifications for communications may not be being satisfied due to the state of the target board. Set the LPD transfer rate to a low rate and check whether the emulator can then be re-connected.	E1203240
	• The RESET pin of the target device may be active. Make sure that the RESET pin is at the inactive level during connection of the emulator.	E1203274
Inability to connect with the debugging tool (emulator) Non-matching of security IDs	• ID authentication may fail when the debugger is connected. Check that the entered ID code is correct.	C0602202

Table 7-2 Problems when the Emulator is Connected (2/2)

Problem	Remedy	Error Code in CS+
Inability to connect with the debugging tool (emulator) Errors in JTAG connection	• When the emulator is connected other than with a hot plug-in connection, although the emulator controls the pin reset, this may fail. Check the notes (e.g. the time the signal takes to reach the high level from the low level) given in section 2.3.4 or whether the electrical characteristics requirements of the reset pin of the device are satisfied. • The connection between the emulator and the target device may be wrong. Refer to section 2.3, Examples of recommended connections between the connector and the target device, and check the circuit between the emulator and the target device. • The specifications for communications may not be being satisfied due to the state of the target board. Set the JTAG transfer rate to a low rate and check whether the emulator can then be re-connected.	E1203331
	• The RESET pin of the target device may be active. Make sure that the RESET pin is at the inactive level during connection of the emulator.	E1203332
Inability to connect with the debugging tool (emulator) Errors in Aurora connection	• Power supplies for Aurora tracing (EMUVCC and EMUVDD) of the target device may not have been turned on. Turn on those power supplies when Aurora tracing is to be used.	E1200265
Traced data is not recorded in trace memory of the IE850A	• The _AURORES pin of the target device may be active. Make sure that the _AURORES pin is at the inactive level during connection of the emulator through Aurora tracing.	W1201361
	• The specifications for communications may not be being satisfied due to the state of the target board. Set the AURORA transfer rate to a low rate and check whether the emulator can then be re-connected. • The connection between the emulator and the target device may be wrong. Check the circuit between the emulator and the target device.	–

7.2 Problem after the emulator is connected

Table 7-3 Problem after the Emulator is Connected

Problem	Remedy	Error Code in CS+
Inability to generate breaks	• The reset signal may have been at the active level for a long time. If a reset is input for more than 8 seconds, forced breaks will be disabled. Wait for the end of the reset input or change the setting for masking resets.	E1200674

Revision History	E2 Emulator, IE850A Additional Document for User's Manual (Notes on Connection of RH850/U2B Series)

Rev.	Date	Description	
		Page	Summary
0.51	Nov.26.21	—	First Edition issued
0.52	Dec.28.21	40	Chapter 7 was deleted.
0.60	Apr.20.22	1	Section 1.3, Use cases for debugging, was added.
		3	Section 2.1.2 was deleted.
		5	Section 2.2.2 was deleted.
		11	Section 2.3.4 was deleted.
		12	Section 2.3.5 was deleted.
		13	Section 2.3.6 was deleted.
		20	"DFP debugging" and "Interlocked DFP resumption and halting facility" were added to section 4.1, List of functions.
		33	Section 5.5, Notes on debugging the DFP (for use case for debugging (2)), was added.
1.00	Jul.20.22	1	In section 1.2, "E2 Emulator Additional Document for User's Manual (Setting up Interlocked Debugging of the DFP in an RH850/U2B-Series Device)" was added.
			In section 1.3, the IE850A is supported during debugging in the G4MH.
		17 to 19	In section 4.1, the column and descriptions for the IE850A were added to table 4-1, Common Functions of the E2 Emulator and IE850A.
		19	In section 4.1, the interlocked DFP resumption and halting facility was unified to DFP debugging in table 4-1.
		20	In section 4.1, software tracing and external trigger input/output were supported and those descriptions were added to table 4-2.
		21	Section 4.2, Software tracing function, was added.
		22	Section 4.3, GTM debugging function, was added.
		27	Section 5.3.11, Peripheral breaks, was added.
		39	Chapter 7, Troubleshooting, was added.
2.00	Mar.17.23	18	Section 3, descriptions on the IE850A were added.
		20	In section 4.1, Hot plug-in connection, changed from unavailable to available in table 4-1.
		21	In section 4.1, Global emulation RAM was added trace output destinations in table 4-1.
		25	In section 4.3, Global emulation RAM was added trace output destinations in table 4-5.
		26	Section 5, conventions were added.
		28	Section 5.2.6, Setting for connection through the external tracing (Aurora tracing) interface, was added.
		30	Section 5.3.5, Corrected referenced application note numbers.
		30	Section 5.3.11 was deleted.
		40	Section 6, descriptions on revision of emulator were added.
		43	Remedies for errors in an Aurora connection were added to table 7-2, Problems when the Emulator is Connected (2/2).

E2 Emulator, IE850A
Additional Document for User's Manual
(Notes on Connection of RH850/U2B Series)

Publication Date: Rev.1.00 Jul.20.22
Rev.2.00 Mar.17.23

Published by: Renesas Electronics Corporation

E2 Emulator, IE850A

Additional Document for User's Manual

(Notes on Connection of RH850/U2B Series)

Renesas Electronics Corporation

R20UT5052EJ0200