User's Manual

E1/E20 Emulator, E2 Emulator

Additional Document for User's Manual (Notes on Connection of RH850/F1KH and RH850/F1KM)

Supported Devices:

RH850 Family

RH850/F1x Series

Renesas Electronics
www.renesas.com

Rev.3.00 May 2025

Notice

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

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

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

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

(Rev.5.0-1 October 2020)

Corporate Headquarters

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

www.renesas.com

Trademarks

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

Contact information

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

Important

Before using the emulator, be sure to read the emulator user’s manual carefully.

Keep the user’s manual, and refer to it when you have questions about the emulator.

When using the emulator:

(1) 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.
(2) Renesas Electronics Corporation will not assume responsibility of direct or indirect damage caused by an accidental failure or malfunction in the emulator.

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.

Precautions for Safety

This chapter, by showing the relevant diagrammatic symbols and their meanings, describes the precautions which should be taken in order to use this product safely and properly. Be sure to read and understand this chapter before using this product.

Contact us if you have any questions about the precautions described here.

WARNING WARNING indicates a potentially dangerous situation that will cause death or heavy wound unless it is avoided.

CAUTION CAUTION indicates a potentially dangerous situation that will cause a slight injury or a medium-degree injury or property damage unless it is avoided.

To avoid a possible danger, the following diagrammatic symbols are used to call your attention.

△ means WARNING or CAUTION.

Example:  CAUTION AGAINST AN ELECTRIC SHOCK

⃠ means PROHIBITION.

Example:  DISASSEMBLY PROHIBITED

● means A FORCIBLE ACTION.

Example:  UNPLUG THE POWER CABLE FROM THE RECEPTACLE.

⚠ WARNING

Warnings for AC Power Supply:

Do not repair or remodel the emulator product by yourself in order to prevent danger such as an electric shock or fire and for the sake of quality assurance. For after-sale services in case of a mechanical or electrical fault, please contact your local distributor.

Always switch off the Host machine and user system before connecting or disconnecting any cables or parts. Neglect of this precaution will result in getting an electric shock or will result in the emulator product or user system emitting smoke or catching fire. Also, the user program under debug will be destroyed.

Make sure that the connectors on both ends of the user interface cable are facing the right way relative to the user-side connector on the emulator and the connector on the user system, respectively.

Neglect of this precaution will result in getting an electric shock or will result in the emulator product or user system emitting smoke or catching fire.

Warning for Modification:

Do not modify the emulator. Personal injury due to electric shock may occur if the emulator is modified. Modifying the product will void your warranty.

Warning for Installation:

Do not set the emulator in water or areas of high humidity. Make sure that the product does not get wet. Spilling water or some other liquid into the product may cause un-repairable damage.

Warning for Use Temperature:

The emulator is to be used in an environment with a maximum ambient temperature of 35°C.

Care should be taken that this temperature is not exceeded.

⚠ CAUTION

Caution to Be Taken for Handling the Emulator:

Exercise caution when handling the emulator. Be careful not to apply a mechanical shock.

Do not touch the connector pins of the emulator and the target MCU connector pins directly. Static electricity may damage the internal circuits.

When attaching and removing the cable, hold the plug of the cable and do not touch the cable. When installing the emulator, do not flex the cable excessively or pull the emulator or the board by the cable connected to it. The cable may cause a break.

Do not tape the flexible cable or apply adhesives to secure the cable. The shielding material on the surface of the cable may come off.

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 user system.

(2) After a lapse of 10 seconds, turn ON the power of the emulator and the user system again, then launch the emulator debugger.

Caution to Be Taken for Disposal:

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

European Union Regulatory Notices:

The WEEE (Waste Electrical and Electronic Equipment) regulations put responsibilities on producers for the collection and recycling or disposal of electrical and electronic waste. Return of WEEE under these regulations is applicable in the European Union only. This equipment (including all accessories) is not intended for household use. After use the equipment cannot be disposed of as household waste, and the WEEE must be treated, recycled and disposed of in an environmentally sound manner.

Renesas Electronics Europe GmbH can take back end of life equipment, register for this service at "http://www.renesas.eu/weee"

Table of Contents

1. Outline

1.1 Features of an E1, E20, or E2 emulator

An E1, E20, or E2 emulator is an on-chip debugging emulator that includes a flash programming function, which is used for debugging and programming programs to be embedded in microcontrollers that have on-chip flash memory. That is, either product can debug a program while the target microcontroller is connected to the user system, and can write programs to the on-chip flash memory of microcontrollers.

1.2 Caution on using the E20 emulator

The functions used for debugging of the RH850 family by using the E20 emulator are the same as in the E1 emulator. Large trace function, a characteristic function of the E20 emulator, cannot be used.

1.3 Configuration of manuals

When using the E1, E20, or E2 emulator in debugging with an RH850 family product, be sure to read the manuals (1) and (2) below. Also read the application note (3) if required.

(1) E1 or E20 emulator user's manual and E2 emulator user's manual

The E1/E20 Emulator User's Manual and E2 Emulator User's Manual describe hardware specifications including the following items:

- Components of the emulators
- Emulator hardware specifications
- Connecting the emulator to a host computer and user system

(2) E1 or E20 emulator, E2 emulator additional document for user's manual (this manual)

The E1/E20 Emulator, E2 Emulator Additional Document for User's Manual describes functions of a debugger, and its contents depend on the given set of MCUs. In general, an additional document has notes on items including the following:

- For use in hardware design, an example of connection and the interface circuits required to connect the emulator.
- Notes on using the emulator

(3) E2 emulator application note

The E2 Emulator Application Note includes an explanation, descriptions of usage, and notes on the extended functions of the E2 emulator.

2. Connecting the Emulator and User System

To connect the E1, E20, or E2 emulator, a connector for the user system interface cable must be mounted on the user system. When designing the user system, read this chapter of this manual and the hardware manual for the MCUs to be used.

2.1 Connector mounted on the user system

Table 2-1 shows the recommended 14-pin connector for connection of the E1, E20, or 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.

Table 2-1 Recommended Connector

	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)

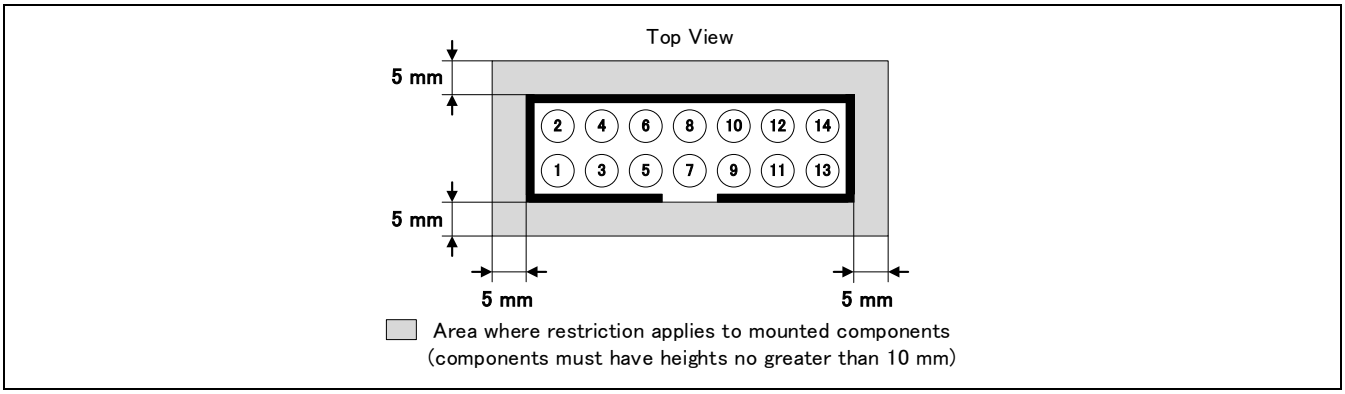

Figure 2-1 Area where Restriction Applies to Mounted Components

- For the connection of an E1 emulator
  Figure 2-2 shows an example of the connection of the user system interface cable of an E1 emulator to a 14-pin connector.

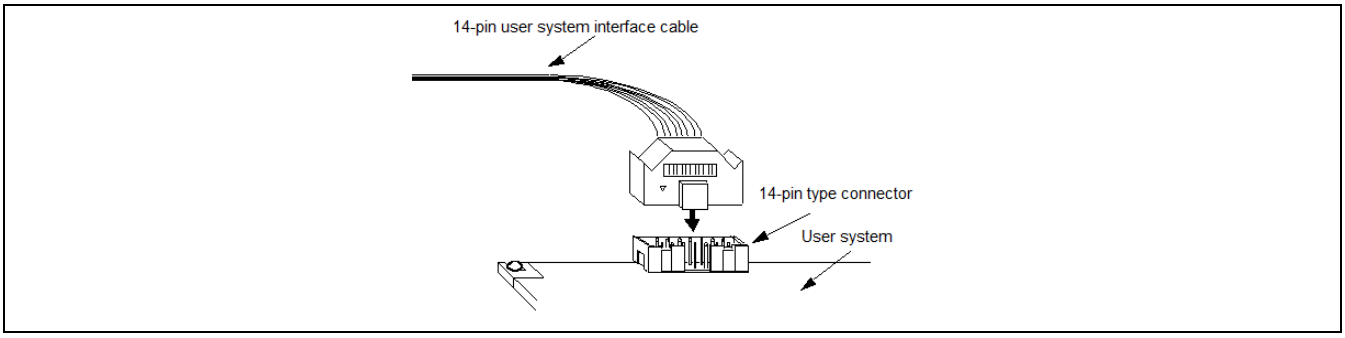

Figure 2-2 Connecting the User System Interface Cable to the 14-pin Connector in the E1 Emulator

- For the connection of an E20 emulator
  To use an E20 emulator with a 14-pin connector, use the 38-pin/14-pin connector conversion adapter [R0E000200CKA00] that comes with the E20.

- For the connection of an E2 emulator

  To use an E2 emulator with a 14-pin connector, use the connector conversion adapter that comes with the E2. Figure 2-3 shows an example of the connection.

  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 2-2.

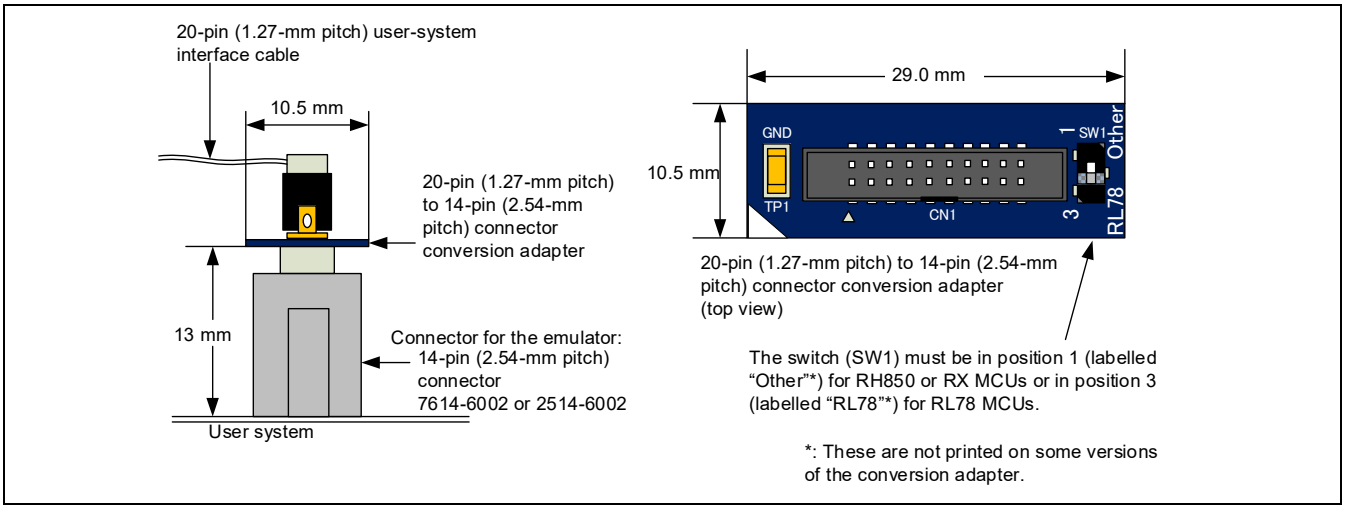

Figure 2-3 Connecting the User System Interface Cable to the 14-pin Connector in the E2 Emulator

Table 2-2 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 user system, grasp the connector cover at the end of the cable or both sides of the board of the connector conversion adapter. Pulling the cable itself will damage the wiring.

If a 20-pin (1.27-mm pitch) user-system interface cable of the E2 emulator becomes disconnected, purchase the following discontinued product: RTE0T00020KCAC0000J.

CAUTION

Note on connector insertion and removal (2):

Be aware that the user-system interface cable or the connector conversion adapter must be inserted with the correct orientation. Connecting the user-system interface cable or the connector conversion adapter with the wrong orientation may cause damage.

2.2 Pin assignments of the connector

Table 2-3 shows the pin assignments of the 14-pin connector.

Table 2-3 Pin Assignments of the 14-pin Connector

Pin No.	Signal name (#: active low, -: unused)						I/O (*4)
	Debugging			Programming (RFP)			
	4-pin LPD	1-pin LPD	JTAG	2-wire UART	1-wire UART	CSI	
1	LPDCLK	—	TCK	—	—	FPCK	Input
2 (*1)	GND	GND	GND	GND	GND	GND	—
3	LPDRST#	—	TRST#	—	—	—	Input
4	FPMD0	FPMD0	FPMD0	FPMD0	FPMD0	FPMD0	Input
5	LPDO	—	TDO	FPDT	—	FPDT	Output
6	—	—	—	FPMD1	FPMD1	FPMD1	Input
7	LPDI/LPDIO	LPDIO	TDI	FPDR	FPDR	FPDR	I/O
8	TVDD	TVDD	TVDD	TVDD	TVDD	TVDD	—
9	—	—	TMS	—	—	—	Input
10 (*3)	EVTO	—	EVTO	—	—	—	Output
11	LPDCLKO	—	RDY#	—	—	—	Output
12 (*1)	GND	GND	GND	GND	GND	GND	—
13 (*2)	RESET#	RESET#	RESET#	RESET#	RESET#	RESET#	Input
14 (*1)	GND	GND	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 E1, E20, or E2 emulator.

2. Be particularly sure to connect pin 13 before using the emulator.

3. 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.

4. 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 Connection interface and modes

The operating mode and the connection interface of an E1, E20, or E2 emulator is switched in the ways shown in Table 2-4 according to whether it is in use for debugging (when a debugger is in use) or programming (when the Flash Programmer is in use). The serial programming mode may still be used even if the debugger is in use. When flash memory is programmed by the downloading function of the debugger, the flash self-programming function is used.

Table 2-4 Modes and Connection Interfaces

Usage	Tool		Device Mode	Connection Interface
Programming	Renesas Flash Programmer (RFP)		Serial programming mode	1- or 2-wire UART or CSI
Debugging	CS+, MULTI*1 or e2 studio	When OPJTAG is automatically set (connected)*2	Serial programming mode	When 1-pin LPD is selected, 1-wire UART is used. When 4-pin LPD is selected, 2-wire UART is used.
		During debugging	Normal operation mode or user boot mode	1- or 4-pin LPD or JTAG

Notes: 1. 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.

2. OPJTAG automatic setting function: When a device is debugged, the OPJTAG bit in the option byte register determines the type of connection interface. Debugging will not start if the interface selected by the OPJTAG bit does not match that selected by the debugger. If the OPJTAG automatic setting function is enabled, the emulator makes a transition to the serial programming mode without fail and reads the OPJTAG bit. If the interface differs from that selected by the debugger, the OPJTAG bit is rewritten, the mode is switched to the normal operating mode, and debugging will start.

   When this function is enabled to start debugging, since the mode is switched to the serial programming mode, some emulation may be impossible since the initial values in memory and of ECC errors after a reset are undefined. Therefore, only use the OPJTAG automatic setting function when the OPJTAG bit in the option byte register is to be modified. For details on setting this function, refer to the user's manual for the debugger you are using.

   With CS+, select "Yes" as the [Set OPJTAG in LPD connection before connecting] property on the [Connect Settings] tabbed page to enable the OPJTAG automatic setting function.

2.4 Examples of recommended connections between the connector and MCU

This section describes examples of recommended connections between the connector for the E1, E20, or E2 emulator and the target device. Since there are various examples of recommended connections according to the purpose of the emulator, select the appropriate circuit with reference to Table 2-5. Be sure to take the specifications of the target device as well as measures to prevent noise into consideration when designing your circuit.

Table 2-5 Purpose of the E1, E20, or E2 Emulator and the Corresponding Example of Recommended Connections

Purpose	Figure
Debugging (1-pin LPD, 4-pin LPD or JTAG) and programming (1-wire UART, 2-wire UART or CSI)	Figure 2-4
Debugging (1-pin LPD) and programming (1-wire UART)	Figure 2-5
Only programming (1-wire UART or 2-wire UART)	Figure 2-6
Only programming (CSI)	Figure 2-7

2.4.1 Example of recommended connections for debugging (1-pin LPD, 4-pin LPD or JTAG) and programming (1-wire UART, 2-wire UART or CSI)

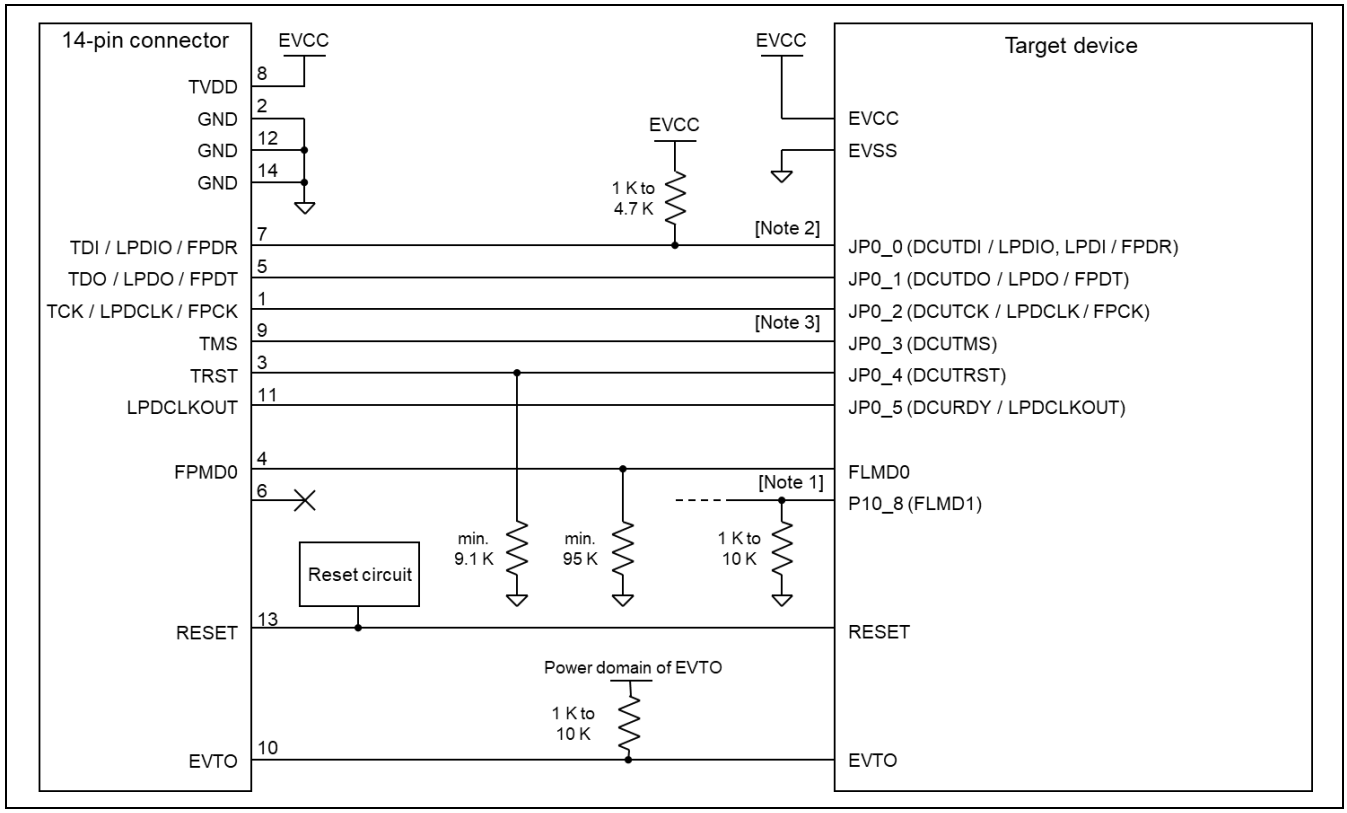

Figure 2-4 Example of Connection

- Note 1: Design the circuit so that the FLMD1 pin must be at the low level during programming.
- Note 2: Pulling up of JP0_0 is only necessary if you are using the 1-pin LPD interface.
- Note 3: When debugging proceeds with an LPD connection by using an E1 emulator, do not enable the pull-up function of the JP0-3 ports in the user program.
- Refer to section 2.4.5, Connecting the RESET pin, for more information on the reset circuit.
- For details on TVDD, refer to section 2.4.6, 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 target device 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.
- During programming (using the RFP or automatically setting OPJTAG), it outputs the high level on FPMD0 and the low level on FPMD1 to place the device in the serial programming mode. If necessary, connect FPMD1 and FLMD1.
- In usual, there is no output on FPMD0 and FPMD1 during debugging. That is, the pins are placed in the high-impedance (Hi-z) state. The high level may be output on FPMD0 during rewriting of the internal flash memory by the downloading function and so on.

- Handle the mode pins correctly in accord with the instructions in the user's manual for the target device when using the user boot mode.

CAUTION

Connection of emulators from other manufacturers:

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

2.4.2 Example of recommended connections for debugging (1-pin LPD) and programming (1-wire UART)

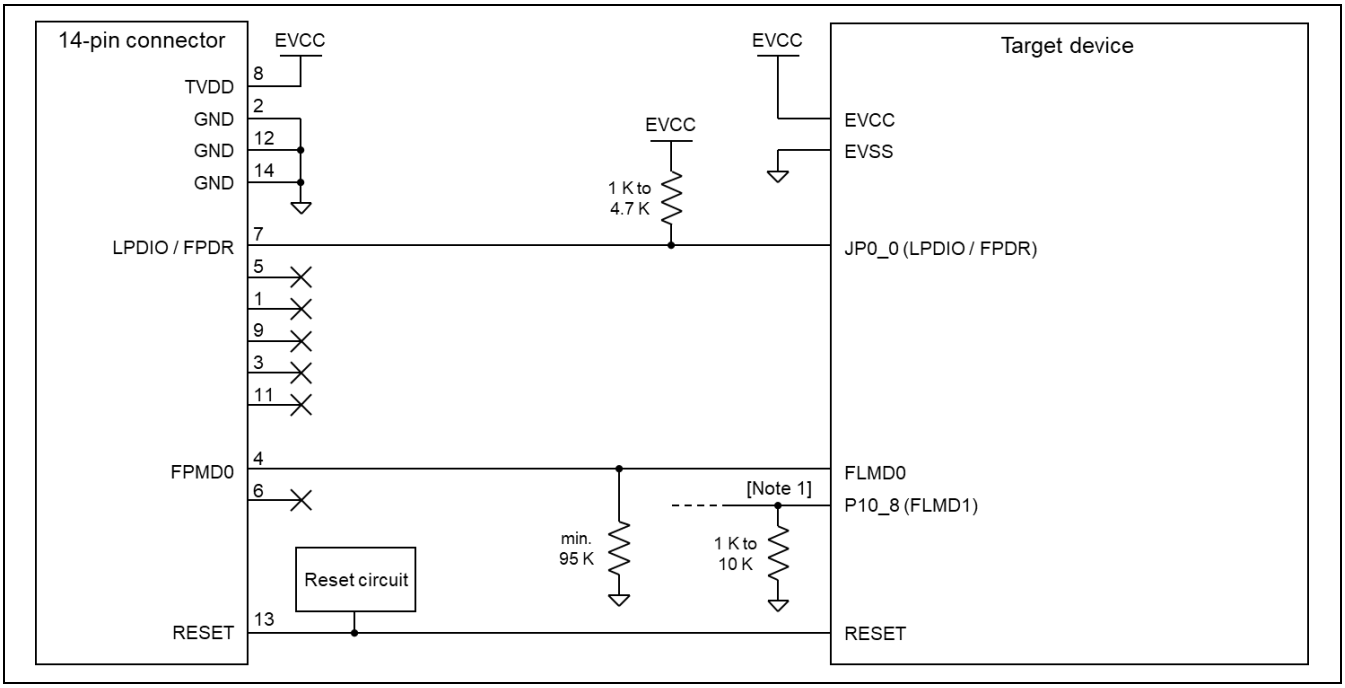

Figure 2-5 Example of Connection

- Note 1: Design the circuit so that the FLMD1 pin must be at the low level during programming.
- Refer to section 2.4.5, Connecting the RESET pin, for more information on the reset circuit.
- For details on TVDD, refer to section 2.4.6, 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 target device 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.
- During programming (using the RFP or automatically setting OPJTAG), it outputs the high level on FPMD0 and the low level on FPMD1 to place the device in the serial programming mode. If necessary, connect FPMD1 and FLMD1.
- In usual, there is no output on FPMD0 and FPMD1 during debugging. That is, the pins are placed in the high-impedance (Hi-z) state. The high level may be output on FPMD0 during rewriting of the internal flash memory by the downloading function and so on.
- Handle the mode pins correctly in accord with the instructions in the user's manual for the target device when using the user boot mode.

2.4.3 Example of recommended connections for only programming (1-wire UART or 2-wire UART)

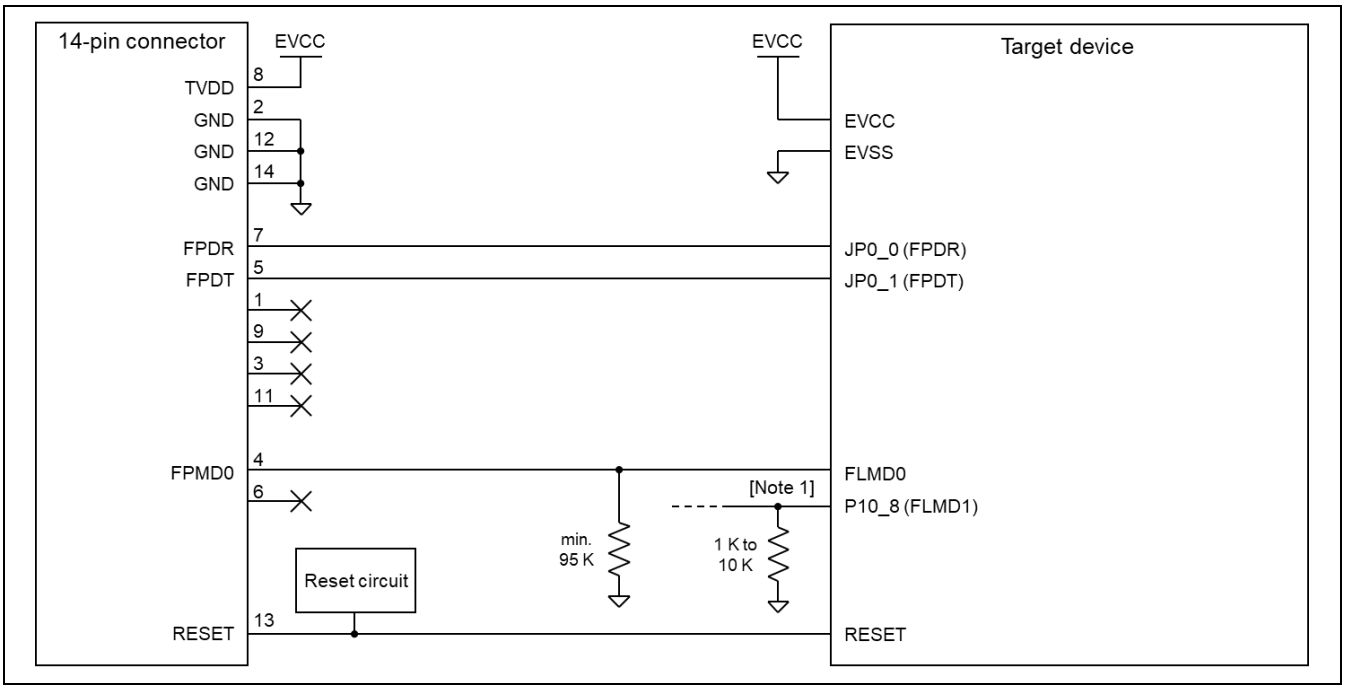

Figure 2-6 Example of Connection

- Note 1: Design the circuit so that the FLMD1 pin must be at the low level during programming.
- Refer to section 2.4.5, Connecting the RESET pin, for more information on the reset circuit.
- For details on TVDD, refer to section 2.4.6, 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 target device 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.
- During programming (using the RFP), it outputs the high level on FPMD0 and the low level on FPMD1 to place the device in the serial programming mode. If necessary, connect FPMD1 and FLMD1.

2.4.4 Example of recommended connections for only programming (CSI)

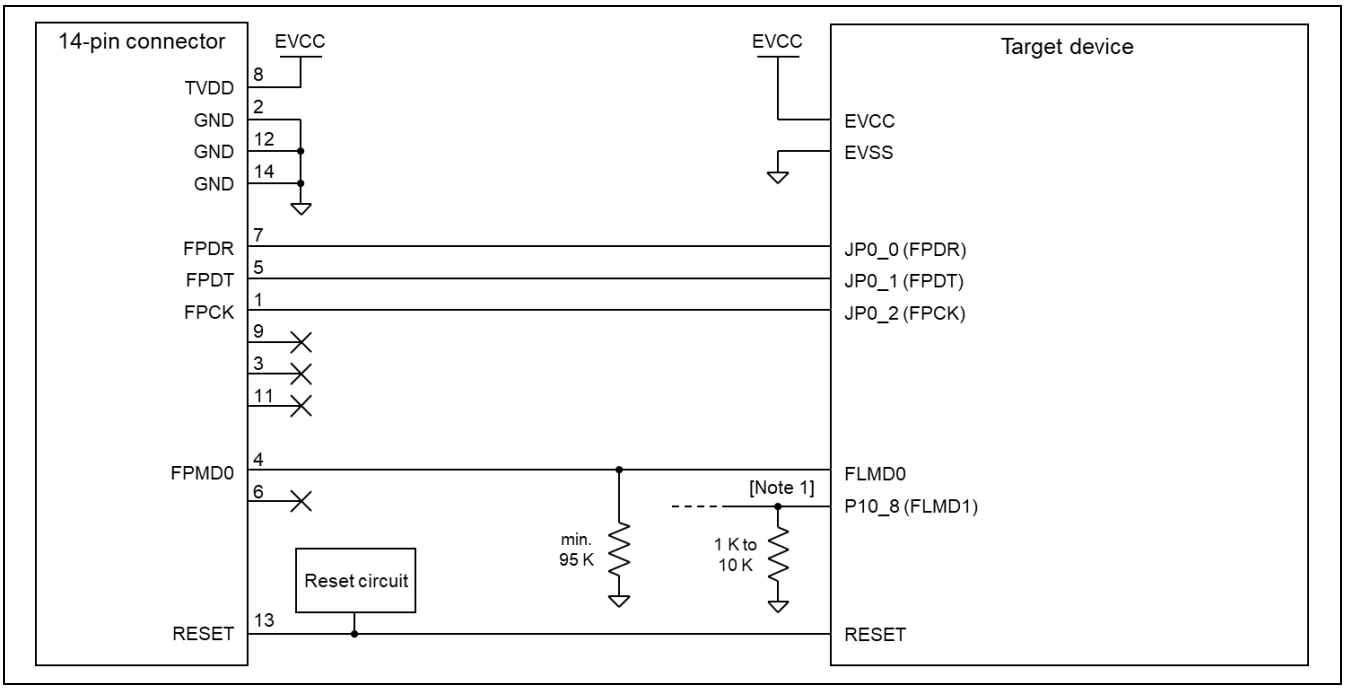

Figure 2-7 Example of Connection

- Note 1: Design the circuit so that the FLMD1 pin must be at the low level during programming.
- Refer to section 2.4.5, Connecting the RESET pin, for more information on the reset circuit.
- For details on TVDD, refer to section 2.4.6, 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 target device 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.
- During programming (using the RFP), it outputs the high level on FPMD0 and the low level on FPMD1 to place the device in the serial programming mode. If necessary, connect FPMD1 and FLMD1.

2.4.5 Connecting the RESET pin

While you are using the E1, E20, or E2 emulator, pin 13 (RESET pin) of the 14-pin connector must be connected to the reset pin of the target device. Figure 2-8 shows an example.

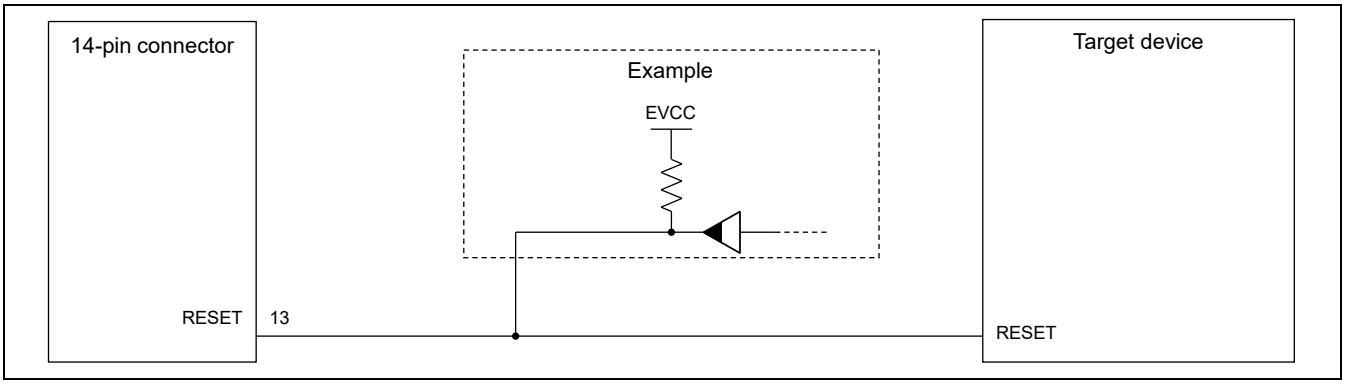

Figure 2-8 Example of Connecting a Reset Circuit

The E1, E20, or E2 emulator fixes the RESET pin to the low level before the debugger is activated. After the debugger 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.

- 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).
- For the target device in this user's manual, pull the RESET signal up to the EVCC voltage.
- Pin 13 (RESET) of the E1, E20, or E2 emulator is pulled up (by a 100-kΩ resistor) within the emulator (refer to chapter 5, Internal Circuits of the 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 E1, E20, or 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.

2.4.6 Connecting the TVDD pin

(1) Power source monitoring function

Connect the power source on the user system to pin 8 (TVDD pin) of the 14-pin connector. For the target device in this user's manual, this will be the source of the EVCC voltage.

The power source connected to the TVDD pin provides power to the final stage output buffer and first stage input buffer on the E1/E20/E2 emulator circuit. When the E1, E20, or E2 emulator is connected, it will draw current as described below in addition to the current drawn by the user system.

- E1/E2 emulator: Approx. 20 mA when TVDD is 3.3 V, and approx. 40 mA when TVDD is 5.0 V
- E20 emulator: Approx. 40 mA when TVDD is 3.3 V, and approx. 100 mA when TVDD is 5.0 V

(2) Power supply function (applies only to the E1 or E2 emulator)

The E1 or 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 E1 or E2 emulator.
- Do not use this function for a user system which draws a current of 200 mA or more. The E1 or 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.
- E1 emulator: The 5.0-V supply depending on the environment of the host machine in use, the voltage might be lower than 5.0 V by 0.5 V or more.
- E2 emulator: The 5.0-V supply depending on the environment of the host machine in use, the voltage might be lower than 5.0 V by 0.3 V or more.

Power supply from the E1 or E2 emulator depends on the quality of the USB power supply of the host machine, and as such, precision is not guaranteed. When writing a program that requires reliability, do not use the power supply function of the E1 or 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 E1, E20, or 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.

2.4.7 Hot plug-in connection

If there is a possibility you will be using hot plug-in connection, you will need to configure the circuit as shown below. Pin 8 of the E1 emulator is connected to a 4.7-uF capacitor as shown in (1) in Figure 2-9, so hot plug-in connection of the emulator may lead to a momentary drop in the power-supply voltage on the user system. This might cause the MCU to be reset.

As shown in (2) in Figure 2-9, 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. Note that hot plug-in connection is only for use during debugging, and a separately sold hot plug-in adapter is necessary to use this function otherwise.

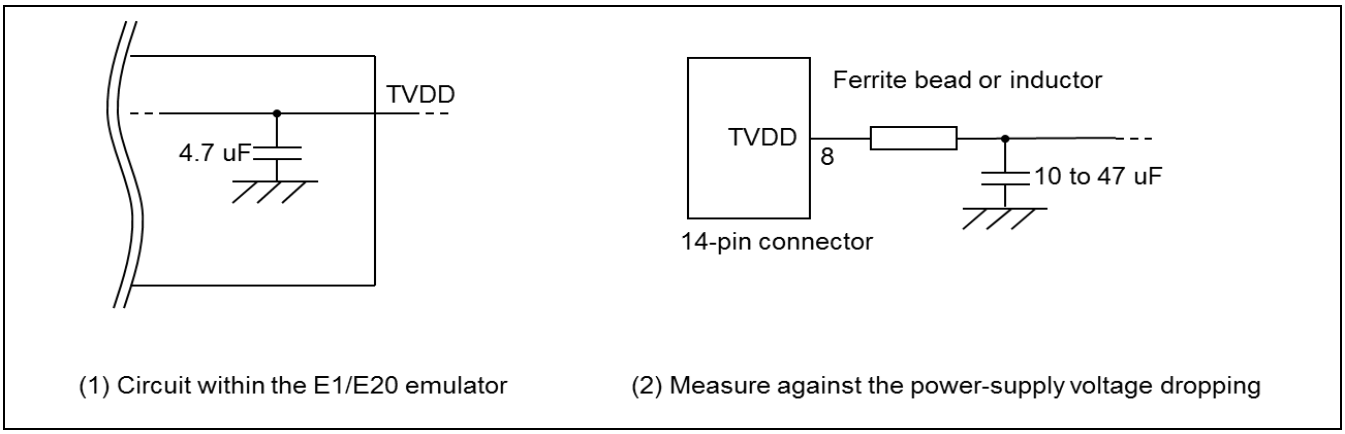

Figure 2-9 Circuit Configuration for Hot Plug-in

- Hot plug-in adapter for the E1 emulator
  For hot plug-in connection to the E1 emulator, use the hot plug-in adapter for the E1 emulator (R0E000010ACB00) that is separately available from Renesas.

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.

The E1, E20, or E2 emulator does not support the hot plug-out function. Do not disconnect the user-system interface cable during debugging.

CAUTION

Note on the hot plug-out function:

While the power of the user system is on, do not unplug the user-system interface cable.
The emulator and the user system may be damaged.

2.4.8 Isolator for the E1 emulator

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 for the E1 emulator (R0E000010ACB20) which is separately available from Renesas.

2.4.9 Small connector conversion adapter for the E1 emulator

A small connector conversion adapter for the E1 emulator (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 E1 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 for the E1 emulator, be aware that the pin assignments of the connector differ from those of the standard interface connector for the E1 emulator.

3. Specifications

Table 3-1 shows specifications common to the E1, E20, and E2 emulators.

Table 3-2 shows specifications specific to the E2 emulator.

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 3-1 Specifications Common to the E1, E20, and E2 Emulators

Broad Category	Medium Category	Narrow Category	Specification
Hardware in general	Corresponding host machine		Computer equipped with a USB port, OS depends on the debugger
	User system interface		14-pin connector
	Host machine interface		USB 2.0 (full speed or high speed)
	Connection to the user system		Connection by the provided user system interface cable
	Power supply function (only when the emulator is an E1 or E2)		3.3 V or 5.0 V (with current up to 200 mA) can be supplied from TVDD to the user system (make settings with the debugger)
	Power supply for the emulator		No need (the host computer supplies power through the USB)
Debugging-related items	Break	Software break	In ROM and RAM areas combined: 2000 points
		Hardware break	12 points including those used for both execution and access conditions (8 points only for execution conditions, and 4 points for either execution or CPU access conditions)
		Event break	Available
		Forced break	Available
		Trace-full break	Available (internal trace memory and E2 storage)
		External trigger input break	Available (E2 emulator only)
	Event	Number of events that can be set	8 points for execution, 8 points for CPU access, 4 points for DMA access, and 4 points for GRAM access (RH850/F1KH only)
		Available function	Break, trace, performance measurement
		Combination of events	OR, sequential

Broad Category	Medium Category		Narrow Category	Specification
Debugging-related items (cont.)	Tracing (only for devices including an internal trace RAM)		Destination for storage	Internal trace memory
			Size	Branch only: 1,000 branches Data trace only: 1,000 cycles of access Software trace only: 1,000 to 2,000 instructions
			Traced data	Branches, cycles of data access, cycles of DMA access, cycles of GRAM access (RH850/F1KH only), and software trace
			Conditions to start and stop recording of data	Stopping of program execution, event condition settings
			Data-trace conditions	Event conditions
			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 Halting tracing due to the input of an external trigger (E2 emulator only)
	Perfor-mance mea-sure-ment	Time (1)	Measurement section	From run to break
			Item measured	Execution time*4
			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*4
			Performance	32-bit counters (for three sections)
		Other than time	Items measured	Number of instructions executed (all or branches only), number of interrupts accepted (EI level or FE level), number of exceptions accepted (instruction asynchronous or instruction synchronous), clock cycles (all, while interrupts are inhibited, or other than for the processing of interrupts)
			Measurement section	From run to break, or between two event points
			Items measured	Latest value, total value, pass count, maximum value, minimum value
			Performance	32-bit counters (for four sections)
	Pseudo real-time RAM monitoring			Available (occupies a bus (steals cycles))*1
	Direct memory modification			
	Debugging console			Not available

Broad Category	Medium Category	Narrow Category	Specification
Debugging-related items (cont.)	Downloading of the external flash memory		Not possible
	Hot plug-in		Possible (To use with the E1 emulator, requires a separately sold hot plug-in adapter)
	Peripheral breaks		Available*2
	Emulator detection by user programs		Available*3 Debugging startup register Initial value: 0000 0000H Address: FA00 2078H (CPU1) Address: FA00 2078H (CPU2: RH850/F1KH only)
	Security		16-byte ID code authentication
	Security ID settings		Not available
	Security flag settings		Not available
	Activating the settings of the Intelligent Cryptographic Unit (Master/Slave type) (ICUM/ICUS)		Not possible
	Main core debug		Possible
	Connection interface		1-pin LPD 500 kbps/1 Mbps/2 Mbps (E1, E20 or E2) 4-pin LPD 5.5 MHz/11 MHz (E1, E20 or E2) JTAG 6.25 MHz/11 MHz (E2 only)
Programming-related items	Security ID settings		Available
	Security flag settings		Available
	Activating the settings of the Intelligent Cryptographic Unit (Master/Slave type) (ICUM/ICUS)		Possible
	Connection interface		2-wire UART, 1-wire UART, CSI

Notes: 1. Only available for the local RAM, global RAM, and retention RAM areas.

2. The function to stop peripheral I/O operation in a break is called the peripheral break function.
   Whether peripheral emulation functions are set or not is determined by the debugger.
   Refer to the manual for the debugger you are using for how to set them.
   Refer to the manual for the MCU you are using to check whether peripheral emulation functions are set.
3. For this function, 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 (refer to section 4.2.33).
4. 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 3-2 Specifications Specific to the E2 Emulator

Broad Category	Medium Category	Narrow Category	Specification
Debugging-related items	Software tracing (LPD output)*1	Target CPU	Selection of a single CPU. For multiple-core devices: 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 (only available in the synchronous debugging mode).
		Destination for storage	"E2 storage": memory for storage in the E2 emulator
		Internal buffer	Eight stages*4
		Traced data	Software trace data + timestamps (given by the E2 emulator)*2 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*1	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 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*3
		Operation when a trigger is output	Output of a low or high pulse (for from 1 μsec to 65535 μsec) can be specified.

Notes: 1. When the software tracing (LPD output), 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. In addition, using the [Debug initial stop state] property that executes a program in the initially stopped state (for CS+) and the FETCHSTOP command (for the MULTI integrated development environment from Green Hills Software) are also not available.

2. 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.

3. In general, 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, but the period becomes 100-μsec when the low-speed internal oscillator (LS IntOSC) is the operating clock for the emulator.

4. 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.

3.1 Overview of specifications specific to the E2 emulator

3.1.1 Software tracing (LPD output)

Devices of the RH850 family support debugging instructions for the output of software trace data. Software trace data are stored in the internal trace memory of the device and output to the emulator via the LPD pins, which is the debugging-connection interface. 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 to the emulator side as trace data. Make use of this function as a new way of debugging. The debugger of CS+ provides useful functionality for applying this software tracing function (via the LPD interface). For details, refer to the user's manual and application note for CS+.

For details of the debugging instructions, refer to the RH850 G3M/G3MH/G3K/G3KH User's Manual: Debugging Instructions. Table 3-3 gives an overview of these instructions.

When the emulator is not connected and the debugging instructions embedded in a program are executed, Software trace data are not output from the LPD interface.

Table 3-3 Debugging Instructions for Software Tracing

Debugging Instruction	Function	Interval between Execution of the Embedded Instruction and the LPD Output*	
		4-pin LPD (11 MHz)	1-pin LPD (2 Mbps)
DBCP	Outputs the current PC value as software trace data.	5.182 usec	28.500 usec
DBTAG imm10	Outputs a 10-bit immediate (imm10) value as software trace data. Output of the PC value is also selectable.	1.727 usec (without the PC value)	9.500 usec (without the PC value)
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.	5.182 usec (Output one register without the PC value)	28.500 usec (Output one register without the PC value)

Note: This item indicates the time required for the LPD output of software trace data generated by executing a debugging instruction. When this interval follows the execution of a debugging instruction, overflows (losses) of software trace data can be avoided. Even if the debugging instruction is executed with a short interval, 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 DBPUSH instruction, set the total number of the registers to less than 5 to avoid an overflow.

3.1.2 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. For details on the function, refer to Table 3-2. For details on the expansion interface, refer to the E2 Emulator User's Manual.

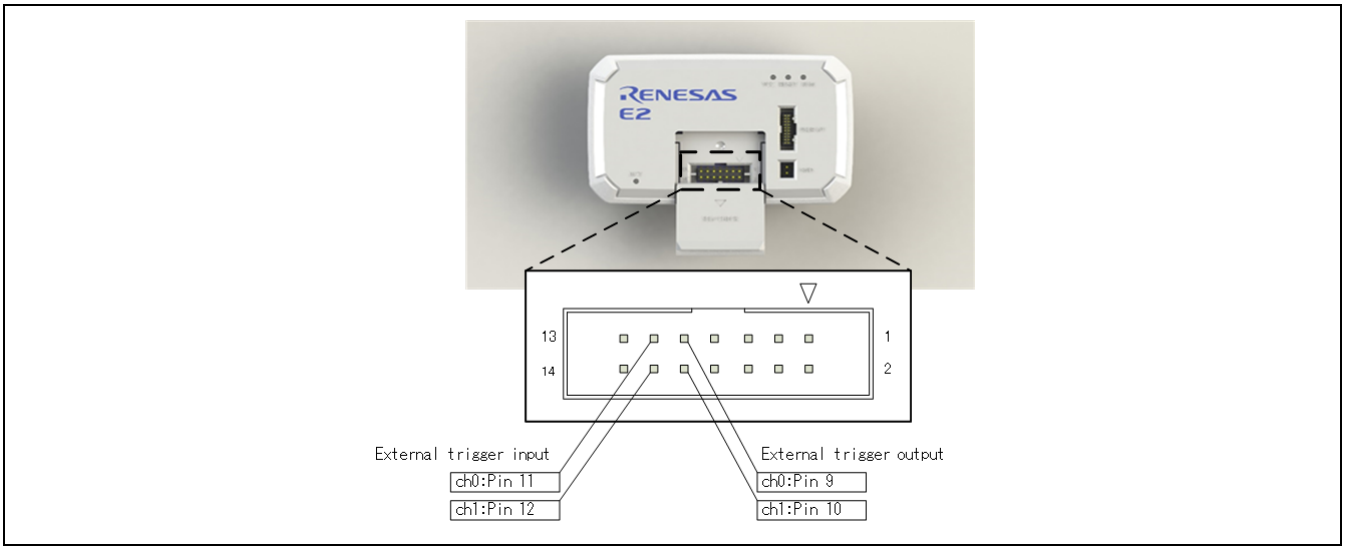

Figure 3-1 Expansion Interface of the E2 Emulator

4. Notes on Usage

Cautionary notes on using the E1, E20, or E2 emulator are given below.

4.1 Notes on differences in operation between the actual device and the E1, E20, or E2 emulator

4.1.1 DBTRAP instruction

The DBTRAP instruction is used for software break functions and thus cannot be used in programs with the emulator.

4.1.2 Serial programming function

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

4.1.3 Current drawn

More current is drawn when an emulator is connected than when it is not connected. That is, the target device consumes more power during debugging than in normal operation since the debugging functions are operating.

4.1.4 Initialization of RAM areas

When an emulator is connected, local RAM, global RAM, and retention RAM areas are initialized to 0000 0000_H. This leads to the following differences from the actual device.

- The initial values in the RAM area after starting an emulator are different from the initial values (undefined values).
- ECC errors due to non-initialization of RAM are not detected with the emulator. If the emulator is not connected and the operation is incorrect, check that the RAM areas have been initialized.

To emulate ECC errors, set the following options.

- The RAM area is not initialized when the emulator is started.
- OPJTAG is not set for an LPD connection before the emulator is connected.

However, if a RAM area is not initialized, the following functions are not available.

- 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
- Rewriting of the option byte

4.1.5 OTP flag

Do not set the one-time programming (OTP) flag in self-programming with the emulator. Note that setting of the flag makes downloading from the debugger to flash memory impossible.

4.1.6 Option byte register

The debugger cannot write new values to the bits of the option byte register indicated below since they are used by the emulator. Also, do not attempt self-programming to write new values to these bits.

- OPJTAG1 and OPJTAG0 bits (bits 30 and 29 of the OPBT0 register)
  The values of the OPJTAG1 and OPJTAG0 bits are as follows.
  When the 4-pin LPD interface is selected: 01B
  When the 1-pin LPD interface is selected: 10B
  When the JTAG interface is selected: 11B

4.1.7 Initially stopped state (when a debugger is connected or a reset is applied)

When a debugger is connected or applies a reset, it forcibly releases all CPUs from their initially stopped states and places the target device in the break state, regardless of whether the debugger is in the synchronous or asynchronous debugging mode. If a program is to be executed in this situation, note that all CPUs will not have entered the initially stopped state.

Using the [Debug initial stop state] property (for CS+) and the FETCHSTOP command (for the MULTI integrated development environment from Green Hills Software) 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 (R20AN0577EJ0100).

When software tracing (LPD output) and E2 expansion interfaces (external trigger input and output) are in use, the function for executing a program in the initially stopped state after a reset cannot be used.

4.2 Cautionary notes on debugging

4.2.1 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.

4.2.2 Power to the target system while debugging

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

4.2.3 Hardware break (access) function (the timing of a break occurring)

When the hardware break (access) function is in use, a break in response to the reading or writing of specified data will occur after the instruction. Other hardware breaks (access) occur before the instruction.

4.2.4 Multiplexed functions of pins used for OCD signals

Multiplexed functions of pins used for on-chip debugging (OCD) cannot be used during debugging.

4.2.5 Debugging interface

The E1, E20, and E2 emulators support 1- and 4-pin LPD interfaces. The E2 emulator also supports the JTAG interface.

Operation is as follows if the setting of the OPJTAG1 and OPJTAG0 bits of the option byte 0 register is "11B" (JTAG: the JTAG interface is selected in the case of a blank chip) and the emulator is connected via the 1- and 4-pin LPD interfaces.

- When starting (connecting) the E1, E20, or E2 emulator
  Settings of the option byte 0 register are changed from the setting for JTAG to that for 1- or 4-pin LPD by the debugger on connection to an emulator.
  Therefore, the OPJTAG1 and OPJTAG0 bits of the option byte 0 register are either "10B" (1-pin LPD) or "01B" (4-pin LPD) during emulator operation.
- When exiting from a session with (disconnecting) the E1, E20, or E2 emulator
  Settings of the option byte 0 register can be changed by the debugger.
  - The value of the OPJTAG1 and OPJTAG0 bits of the option byte 0 register can be changed to "11B" (for JTAG), which requires rewriting of the flash memory.
  - The setting of the OPJTAG1 and OPJTAG0 bits of the option byte 0 register can be left as "01B" (for 4-pin LPD) or "10B" (for 1-pin LPD).

  When the LPD interface is also used the next time the emulator is connected, we recommend exit from the program without changing the settings from that for the LPD interface.

If power to the target system is turned off because of an abnormal end to the emulator session, the OPJTAG1 and OPJTAG0 bits of the option byte 0 register are not rewritten and so retain the value "01B" (for 4-pin LPD) or "10B" (for 1-pin LPD). If you wish to change the OPJTAG1 and OPJTAG0 bits of the option byte 0 register to "11B" (for JTAG), please do so at the end of the E1, E20, or E2 emulator session.

4.2.6 Debugging interface (in the case where the high-speed internal oscillator is input to the PLL) (skipped number)

The information that was previously under this number has been integrated into section 4.2.5.

4.2.7 Initialization of RAM areas

All RAM areas for use by a program must be initialized when an emulator is in use. Before the emulator is used, if any setting is made to initialize the RAM area when the emulator is started, ECC errors are not generated since the debugger initializes the RAM area. However, when the actual device is operated with a program which does not initialize the RAM area, ECC errors will be generated, preventing normal program operation.

ROMization is also required because any data downloaded from the emulator to the RAM area before program execution will also be initialized. For details, refer to the user's manual for the compiler you are using.

4.2.8 Trace function (when a device with a trace function is in use)

The following restrictions apply to the trace function.

- In the case of section trace, for example, the instruction immediately before the fetched instruction that actually caused tracing to start might be included in trace data.
- In some cases, acquired trace information will be lost. This depends on the program being executed. The lost information cannot be restored, but the fact of the loss is indicated (displayed). Information might be lost when access to data by the CPU is continuous and frequent.
- When priority in tracing is given to non-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.
- When data-qualified tracing (point tracing), i.e. tracing only of data in access to a specific address, is specified, tracing proceeds with any data conditions ignored, even if read access conditions are set. Tracing is still governed by conditions other than data conditions.

4.2.9 Power-saving modes

When a power-saving mode is in use, the restrictions below apply.

- For debugging of a program, ensure that the program sets WUFMSK0[31] to 0.
- Release from the stop, deep stop, or cyclic stop mode follows any of the following operations or conditions while the user program is being executed.
  - Break
  - Memory access
  - Forcibly stopping tracing
  - Setting an event
  - Enabling of the trace-full stop or trace-delay stop functions
- The power supply to the Iso area (CPU, RAM, peripheral module, etc.) is not stopped in the deep stop mode during debugging. For this reason, RAM or registers which have undefined initial values retain these values. Be sure to initialize them after returning to the run mode.

4.2.10 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 E1, E20, or E2 emulator.
- The MCU, E1, E20, or E2 emulator, and software are used as described in respective user's manuals.
- The supply of power to the user system is stable.

4.2.11 Turning the power on/off

Turn the power of the E1, E20, or E2 emulator and the user system following the procedure below.

- 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. When using the E20 emulator, check its power switch is 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 E1 or E2 emulator is turned on by connecting the USB interface cable. When using the E20 emulator, turn on its power switch.
(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.
When using the E20 emulator, turn off its power switch. Disconnect the USB interface cable from the E1, E20, or E2 emulator. The E1 or E2 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

Note on the user system power supply:

While the power of the user system is on, do not turn off the host machine, unplug the USB interface cable, or turn off the power switch of the E20 emulator.
The user system may be damaged due to leakage current.

- When power is supplied to the user system from the E1 or E2 emulator

<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.

4.2.12 Resets while the emulator is in use

Table 4-1 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, the emulator masks the user-system resets so that it can continue to emulate each line of 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 a temporary breakpoint and execute the user program. Accordingly, this document cannot define whether a reset is masked by the emulator or not; refer to the manual for the debugger you are using.

Table 4-1 State of a Device and Masking of User-system Resets by the Emulator

		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.
- Resets generated in the states marked (*) in Table 4-1 are held pending. For example, when a setting for software-reset processing is made during single-stepped execution or a software reset by setting a register is applied by the debugger during a break, the reset is held pending and performed after the reset mask is removed.
- Do not allow the generation of a reset in the form of a pin reset from the target system other than while a program is in execution regardless of the presence of masking as described above. A reset generated while the program is running may cause the debugger to hang.

4.2.13 Interrupts while the emulator is in use

Table 4-2 shows the states of a device while the emulator is in use and the operation of interrupts. During single stepping, the emulator masks interrupts so that it can continue 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 whether interrupts are masked by the emulator or not; refer to the manual for the debugger you are using.

Table 4-2 State of a Device and Masking of Interrupts by the Emulator

State of a device			
In breaks	**In single stepping**	**In user program execution**	**In C-source-level stepping**
Interrupts masked*		Interrupts not masked (operation according to the specification of the user system)	Depends on the debugger

- Interrupts (EIINT, FEINT, and FPI) generated in the state marked (*) in Table 4-2 are held pending and interrupt processing proceeds after interrupt masking is canceled.

4.2.14 HALT mode and stepped execution of the HALT instruction

A break leads to release from HALT mode.

When a HALT instruction is encountered during single-stepped 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 state. When a HALT instruction is encountered during C-source-level stepped execution, whether or not the transition to the HALT state proceeds depends on the facilities of the debugger.

4.2.15 Stepped execution of an instruction which would lead to a transition to the deep stop mode on the actual device

Stepped execution has two variants: Single step execution (execution in units of assembly instructions) and C-source-level stepped execution (execution in units of statements or functions in C language source code). When an instruction which would lead to a transition to the deep stop mode on the actual device is executed during single step execution, the program breaks at the address at the time of the reset and does not switch to the deep stop mode. When an instruction which would lead to a transition to the deep stop mode on the actual device is executed during C-source-level stepped execution, whether or not the transition to the deep stop mode proceeds depends on the facilities of the debugger.

4.2.16 Cautionary point regarding connecting an emulator (pin reset)

The reset signal continuing to be asserted while communications between the emulator and MCU are being prepared when the emulator is connected causes the possibility of incorrect communications. Thus, ensure that the reset signal does not remain asserted when the emulator is connected.

4.2.17 Cautionary point regarding connecting an emulator (time required for preparing to communicate)

When an emulator is connected, a program which was written to the MCU is executed from the reset vector before the emulator and MCU become able to communicate. Take care on this point.

When debugging of a program written to the MCU creates a problem, eliminate the problem by inserting a waiting time* indicated below before executing the program following release from the reset state.

- For the 1-pin LPD interface, waiting for at least 140 ms is required.
- For the 4-pin LPD interface, waiting for at least 12 ms is required.

Note: Time required for preparing communications depends on the host PC environment of the E1, E20, or E2 emulator and the operating frequency of the MCU.

4.2.18 Cautionary point regarding connecting an emulator (internal reset)

When the stored program generates an internal reset (software reset or reset caused by the watchdog timer overflowing) immediately after a release from the initial reset state, the internal reset may be generated before communications between the emulator and MCU have been established after the emulator is connected, causing the possibility of incorrect communications.

Accordingly, insert a waiting time* indicated below before applying an internal reset after a release from the initial reset state when debugging a program which includes an internal reset.

- For the 1-pin LPD interface, waiting for at least 140 ms is required.
- For the 4-pin LPD interface, waiting for at least 12 ms is required.

Note: Time required for preparing communications depends on the host PC environment of the E1, E20, or E2 emulator and the operating frequency of the MCU.

4.2.19 Cautionary point regarding connecting an emulator (deep stop mode)

When the stored program makes a mode transition to the deep stop mode immediately after release from the initial reset state, the MCU switches to the deep stop mode before communications between the emulator and MCU are established when the emulator is connected. This causes incorrect communications. Accordingly, insert a waiting time* indicated below before executing an instruction which changes the mode to the deep stop mode after a release from the initial reset state when debugging a program that changes the mode to the deep stop mode.

- For the 1-pin LPD interface, waiting for at least 140 ms is required.
- For the 4-pin LPD interface, waiting for at least 12 ms is required.

Note: Time required for preparing communications depends on the host PC environment of the E1, E20, or E2 emulator and the operating frequency of the MCU.

4.2.20 Cautionary points regarding hot plug-in connection

- Before proceeding with hot plug-in connection, set the OPJTAG [1:0] bits in the corresponding option byte register according to the debugging interface to be selected. When the OPJTAG [1:0] bits of the option byte register do not match the debugging interface to be used, a connection error occurs.
- Allowing hot plug-in connection prevents usage of the optional isolator for the E1 emulator (the isolator is only for use with the RH850 and RL78 groups).
- Allowing hot plug-in connection prevents the supply of power to the user system by the E1 or E2 emulator.
- If hot plug-in connection is not to be used, the RAM area will be initialized* when the emulator is started. Allowing hot plug-in connection prevents this initialization and makes the masking of pins impossible. Thus, when the emulator is started without initializing the RAM area to be used by a program, ECC errors occur. Therefore, make sure to initialize the RAM area to be used by a program before setting up the system for hot plug-in connection.
- After completing hot plug-in connection, the user program will be running. At this time, only the emulator functions listed below are available.
  - Read or write access to the internal RAM area
  - Forced break
  - CPU reset

  Apply a forced break if you wish to return to using all functions supported by the emulator. After the forced break, functions equivalent to those that can be used after normal starting of a program become available.

Note: The RAM area is only initialized if the setting is made to initialize the RAM area when the emulator is started.

4.2.21 Cases where hot plug-in connection is not possible

Hot plug-in connection cannot be used when the microcontroller is in any of states listed below.

- Reset input state
- Cyclic run mode
- Cyclic stop mode

4.2.22 Standby mode released by hot plug-in connection

Hot plug-in connection releases the microcontroller from standby in the form of either of the states below.

- Stop mode
- Deep stop mode (A reset will occur due to the specifications of the device.)

4.2.23 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.

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

4.2.24 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 (for the E2 emulator) or the 9-KB area (for the E1 or E20 emulator) from the last address of the local RAM area of CPU1 is initially set as the working RAM area. If a device has no local RAM area, the retention RAM area is used.

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)

4.2.25 Rewriting of on-chip flash memory (clock monitor)

The debugger changes the PLL multipliers when the flash memory is being rewritten*. Thus, rewriting the flash memory raises a possibility of the frequency becoming higher than that currently in use. If the frequency surpasses the upper limit which was set by the clock monitor (CLMA), this prevents rewriting of the flash memory. If the change in the clock frequency due to the debugger is a problem, set [Change the clock to flash writing] in the [Property] panel to [No].

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)

4.2.26 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].

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

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

4.2.27 Event functions (64-bit access)

Do not set any access events with the condition in 64-bit units. The emulator may detect access in a unit other than 64 bits as satisfying such conditions or other events may not operate normally.

4.2.28 Event functions (in 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.

- An event is set for consecutive instructions but the two instructions are executed at the same time.
- An access event detects adjacent read and write instructions, since the timing of event detection differs in write and read access and 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.

4.2.29 Event functions (bit-manipulation instructions)

When a read or write access condition is set for an event, the writing cycle of read-modify-write generated by a bit-manipulation instruction is not detected as an event. This condition cannot be used as a trigger for a break, trace acquisition, or performance measurement in the case of such instructions.

4.2.30 Satisfaction of two break conditions before a single break

If another read-access event is detected immediately before a transition to the break state due to a forced break or an event break, a further break will occur immediately after execution of the program is resumed because the break request was accepted as a read-access event at the time of resumption.

4.2.31 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.

4.2.32 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 4.1.7, Initially stopped state (when a debugger is connected or a reset is applied).

4.2.33 Emulator detection by user programs

Even if debugging information is specified, note that the value will be initialized to 0000 0000_H if a reset is generated. Debugging information is specified again when a break occurs in all CPUs and when the user program is re-executed after a reset.

4.2.34 Software tracing (LPD output) function when the 1-pin LPD interface is selected (only for the E2 emulator)

When the 1-pin LPD is selected and 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, software tracing (LPD output) cannot be used when execution of the program is subsequently resumed. To proceed with software tracing (LPD output) again, re-connect the emulator to the debugger.

4.2.35 Cautionary point regarding trace data acquired by software tracing (LPD output) (only for the E2 emulator)

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.

4.2.36 Cyclic run mode and cyclic stop mode

This item only applies to the F1KH group.

When the microcontroller enters the cyclic run mode or cyclic stop mode, the core other than CPU1 is stopped. Debugging must only proceed as asynchronous debugging. That is, synchronous debugging is not usable in this situation. Otherwise, refer to section 4.1.7, Initially stopped state (when a debugger is connected or a reset is applied). Also, since the core other than CPU1 is stopped during asynchronous debugging, it cannot be debugged.

4.2.37 Cautionary point regarding asynchronous debugging mode (peripheral break function)

This item only applies to the F1KH group.

In the asynchronous debugging mode, peripheral break functions cannot be used. Even if peripheral break functions are enabled, peripheral functions are not stopped.

4.2.38 Cautionary point regarding asynchronous debugging mode (reset)

This item only applies to the F1KH group.

In the asynchronous debugging mode, when any of CPUs is in the break state, no resets are acceptable.

4.2.39 Cautionary point regarding asynchronous debugging mode (watchdog timer)

This item only applies to the F1KH group.

In the asynchronous debugging mode, when any of CPUs is in the break state, counters are stopped in WDTA0, WDTA1, and WDTA2.

4.2.40 Cautionary point regarding asynchronous debugging mode (ECC error)

This item only applies to the F1KH group.

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.

4.2.41 Cautionary point regarding asynchronous debugging mode (specific sequence)

This item only applies to the F1KH group.

During execution of a user program, there may be a case that the specific sequence is not satisfied.

Example: When any CPU accesses the specific I/O register during execution of a user program and another CPU which is in the break state accesses the same peripheral function in the I/O register window at the same timing, the specific sequence from any CPU is not satisfied and normal accessing is disabled.

4.2.42 Initially stopped state of CPU2 (forced break)

This item only applies to the F1KH group.

You cannot select CPU2 and cause a forced break when CPU2 is in its initially stopped state immediately after release from the reset state. A forced break is not possible until CPU1 has released CPU2 from its initially stopped state.

4.2.43 Initially stopped state of CPU2 (synchronous break)

This item only applies to the F1KH group.

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 4.1.7, Initially stopped state (when a debugger is connected or a reset is applied).

4.2.44 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.

4.2.45 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. Internal Circuits of the Emulator

Figure 5-1 and Figure 5-2 show the internal circuits of the E1 or E20 emulator. Figure 5-3 and Figure 5-4 respectively show those of the E2 emulator (product revision C) and the E2 emulator (product revision D).

The alphabet at the end of the serial No. written on the E2 emulator main unit indicates the product revision. Refer to these figures when determining parameters in board design.

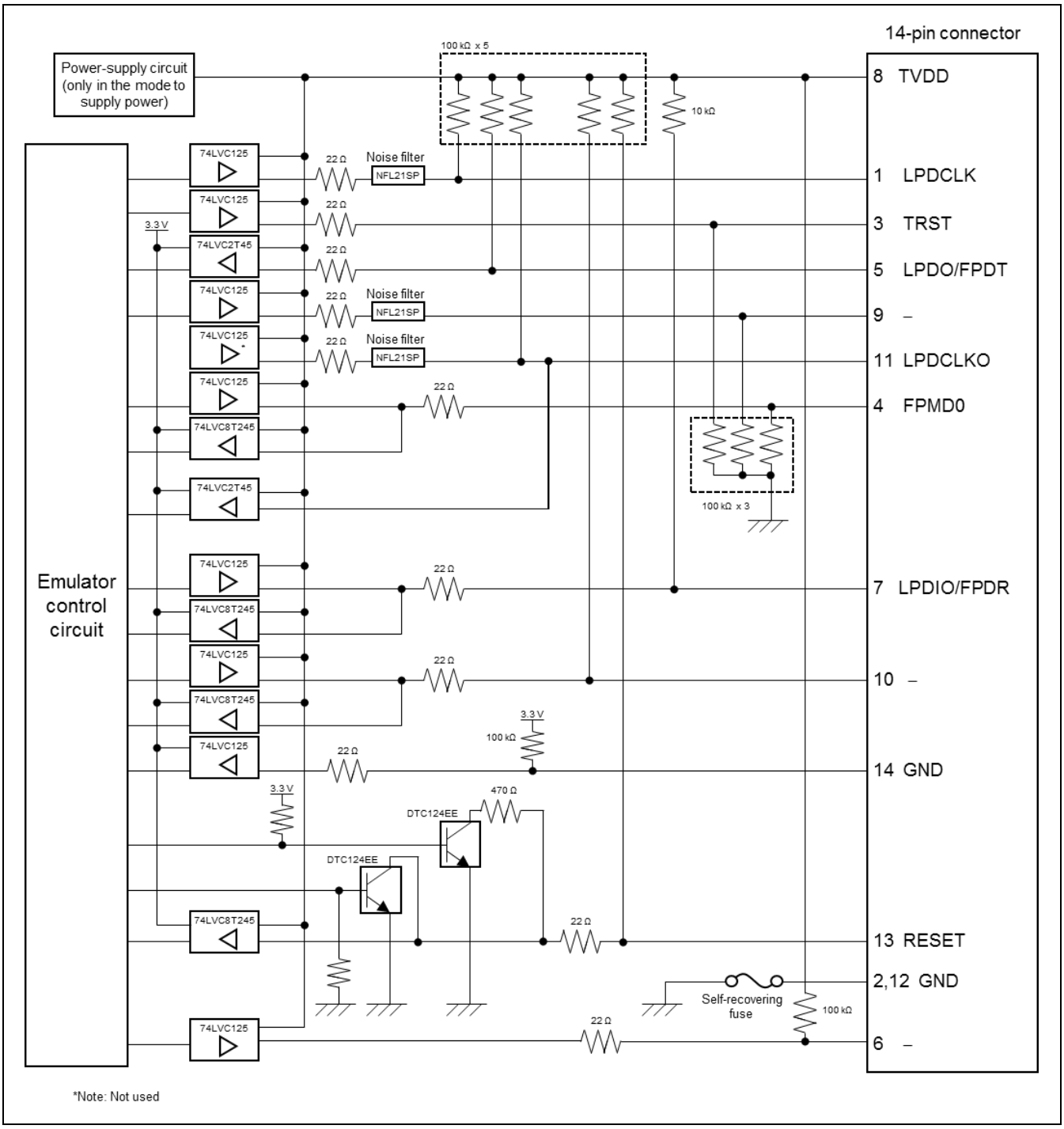

Figure 5-1 Interface Circuits in the E1 or E20 Emulator (1-Pin LPD, 1-Wire UART)

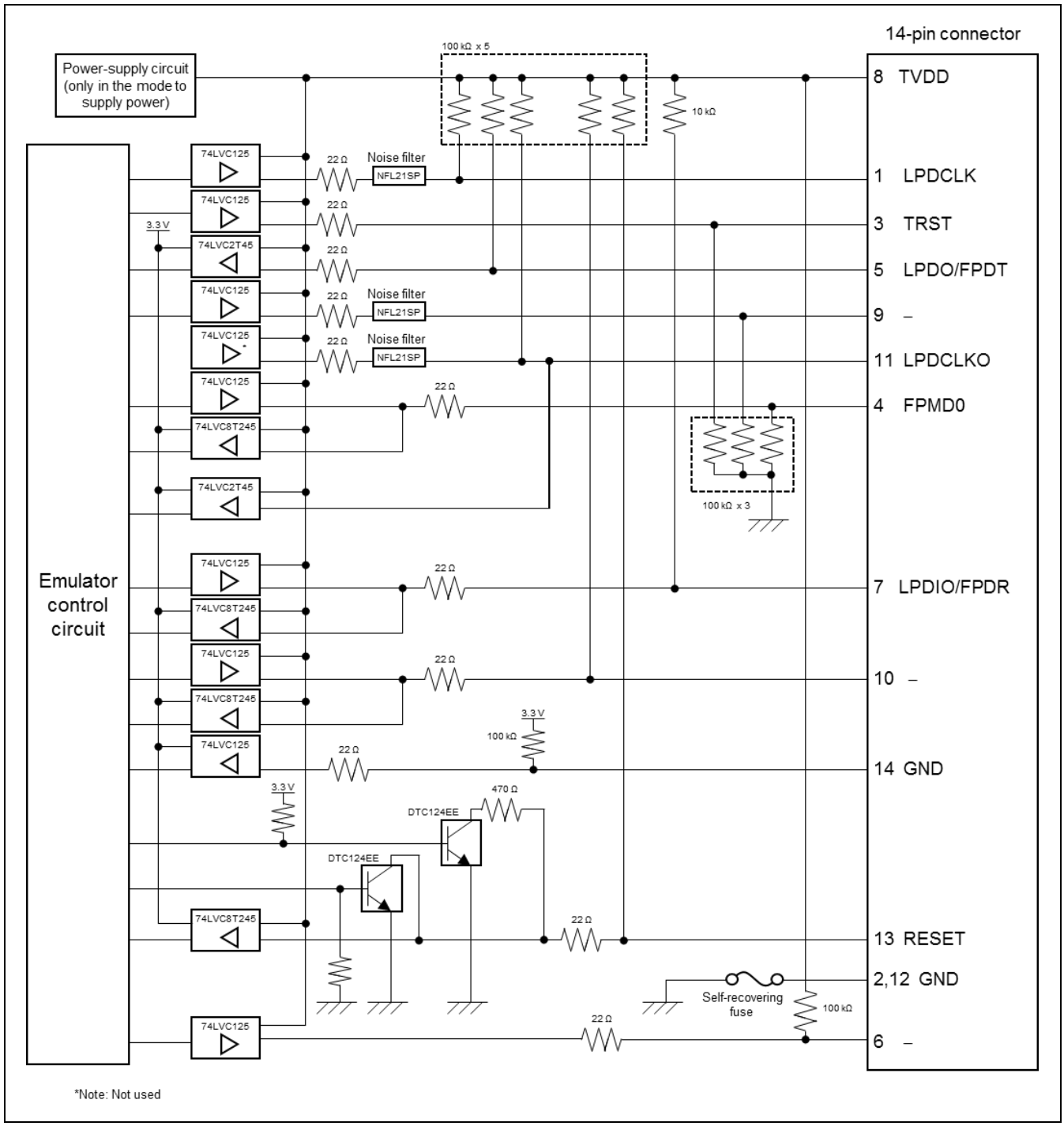

Figure 5-2 Interface Circuits in the E1 or E20 Emulator (4-Pin LPD, 2-Wire UART)

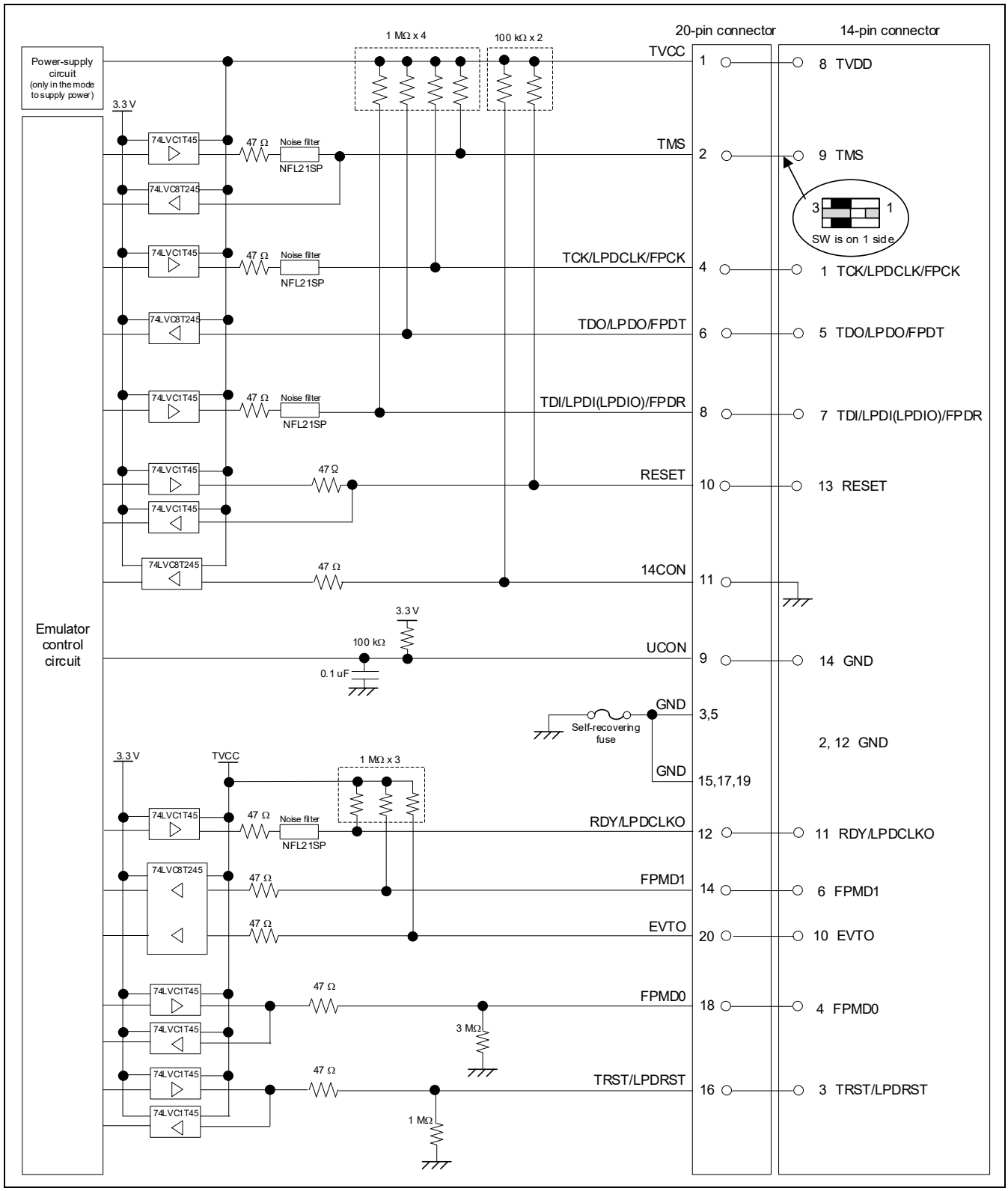

Figure 5-3 Interface Circuits in the E2 Emulator
(1-Pin LPD, 4-Pin LPD, JTAG, 1-Wire UART, 2-Wire UART, CSI) (Rev. C)

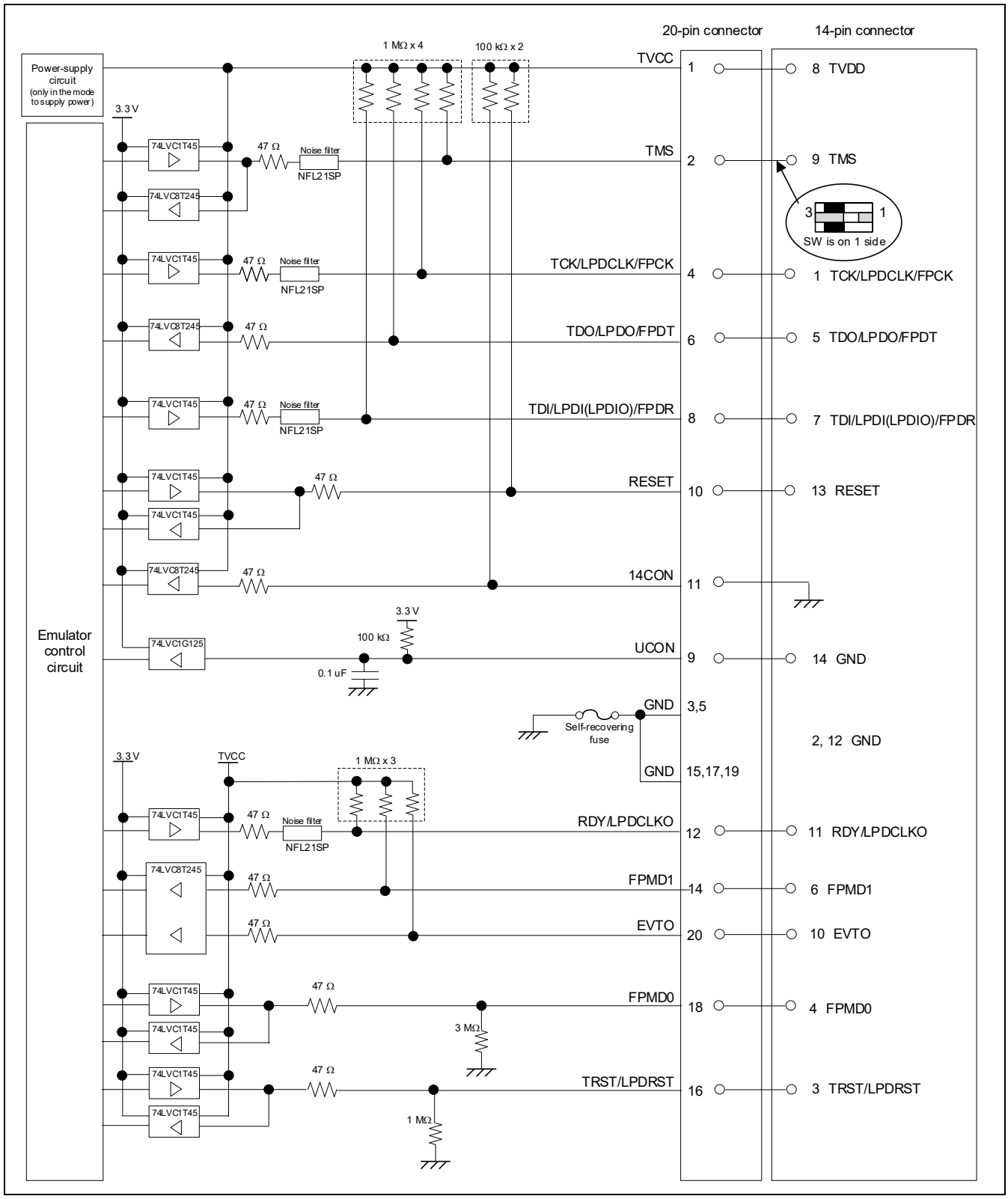

Figure 5-4 Interface Circuits in the E2 Emulator
(1-Pin LPD, 4-Pin LPD, JTAG, 1-Wire UART, 2-Wire UART, CSI) (Rev. D)

6. Troubleshooting

This chapter gives examples of problems that may arise while the E1, E20, or E2 emulator is being used in combination with a debugger and of remedies for these problems. Also read the sections of the E1 or E20 emulator user's manual, E2 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.

6.1 Problems when the emulator is connected

Table 6-1 Problems when the Emulator is Connected (1/3)

Problem	Remedy	Error Code in CS+
Inability to connect with the debugging tool (emulator) This error may occur in the flash programming mode.	When OPJTAG automatic setting is enabled for the setting of the debugger, switch the device to the flash programming mode when it is connected and check and change the value of the OPJTAG bit in the option byte (see section 2.3). If this fails, the error message shown at right will appear. Check the following items. • Control of pin resets for the transition to the flash programming mode may be wrong. When an emulator is connected, do not input a reset signal to the pin on the circuit other than from the emulator. Check the notes (e.g. the time the signal takes to reach the high level from the low level) given in section 2.4.5 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 2.4, Examples of recommended connections between the connector and MCU, and check the circuit between the emulator and the target device. • Check that mode pins such as FLMD1, which are not controlled by the emulator, are being handled in ways that allow transitions to the flash programming mode.	E1203237
	• The value set for MainOSC may be wrong. Check whether the frequency of MainOSC on the board matches the value set for connecting the debugger.	E1203275
	• The connection between the emulator and the target device (particularly that of the FLMD0 pin) may be wrong. Refer to section 2.4, Examples of recommended connections between the connector and MCU, and check the circuit between the emulator and the target device.	E1203276

Table 6-2 Problems when the Emulator is Connected (2/3)

Problem	Remedy	Error Code in CS+
Inability to connect with the debugging tool (emulator) Error in LPD connection	• The OPJTAG bit in the option byte may not be specifying the correct connection interface (LPD). Enable OPJTAG automatic setting as the setting for the debugger to allow rewriting of the option byte when the emulator is started or use a flash programmer (e.g. the RFP) to change the value of the OPJTAG bit before connecting the debugger. • The condition in cautionary note given in section 4.2.17 on the time required for preparing communications before the emulator is connected to the target device may not be being satisfied. Use a flash programmer (e.g. the RFP) to erase the code flash memory and check whether this makes the emulator connectable to the target device. • 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 2.4.5 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 2.4, Examples of recommended connections between the connector and MCU, 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. • The value of the option byte may not be correct. Check that the value of the option byte has been specified with suitable settings according to the hardware manual for the MCU in use by using a Flash Programmer (RFP, etc.).	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. • The condition in cautionary note given in section 4.2.17 on the time required for preparing communications before the emulator is connected to the target device may not be being satisfied. Use a flash programmer (e.g. the RFP) to erase the code flash memory and check whether this makes the emulator connectable to the target device.	C0602202

Table 6-3 Problems when the Emulator is Connected (3/3)

Problem	Remedy	Error Code in CS+
Inability to connect with the debugging tool (emulator) Errors in JTAG connection	• The OPJTAG bit in the option byte may not be specifying the correct connection interface (JTAG). Use a flash programmer (e.g. the RFP) to change the value of the OPJTAG bit before connecting the debugger. • The condition in cautionary note given in section 4.2.17 on the time required for preparing communications before the emulator is connected to the target device may not be being satisfied. Use a flash programmer (e.g. the RFP) to erase the code flash memory and check whether this makes the emulator connectable to the target device. • 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 2.4.5 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 2.4, Examples of recommended connections between the connector and MCU, 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. • The value of the option byte may not be correct. Check that the value of the option byte has been specified with suitable settings according to the hardware manual for the MCU in use by using a Flash Programmer (RFP, etc.).	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

6.2 Problems after the emulator is connected

Table 6-4 Problems 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. • Supply of a clock signal to the CPU may have stopped. Do not stop supply of a clock signal to the CPU. If supply of a clock signal to the CPU is stopped, power to the target must be turned off and then on again. Turn off the power to the target and re-connect the debugger. • The condition in cautionary note given in section 4.2.9 on the power-saving modes may not be being satisfied. Wait for the target device to be switched to the normal mode or cancel masking of resets and input a reset to the pin on the target board. During debugging, be sure to set a wake-up source in the WUFMSK0 register.	E1200674

Revision History	E1/E20 Emulator, E2 Emulator Additional Document for User's Manual (Notes on Connection of RH850/F1KH and RH850/F1KM)

Rev.	Date	Description	
		Page	Summary
1.00	Jul.01.17	—	First Edition issued
1.10	Jan.15.18	Whole document	Description related to the F1KH group were added.
		5-6	Descriptions were added in section 2.1.
		19	Table 3-1 was updated.
		24	“Top cover” was changed to “the connector for the interface can be found by removing the cover”.
		36	Descriptions were added to section 4.2.24.
1.20	Oct.01.18	6	Figure 2-3 was updated.
		Whole document	Remove the descriptions about “high-speed internal oscillator is the input to the PLL”.
		10-14	The descriptions were added about the connection of FLMD1 and FPMD1.
		25	Descriptions were added to section 4.1.7.
		27	Section 4.2.6 was skipped.
		35	Section 4.2.25 was modified.
2.00	Oct.09.20	20	Note 4 was added to table 3-1.
		22	Note 4 was added to table 3-2.
		34	Statements were added to section 4.2.20, Cautionary point regarding hot plug-in connection.
		37	Statements were added to section 4.2.32, Cases where no break occurs.
		38	A statement was added to section 4.2.36, Cyclic run mode and cyclic stop mode.
		40	Sections 4.2.44 and 4.2.45 were newly added.
		45	Remedies for two types of error (E1203275 and E1203276) were added to table 6-1.
		46	A remedy for one type of error (E1203274) was added to table 6-2.

3.00	May.30.25	3	A description on the user-system interface cable (discontinued product) of the E2 emulator was added in section 2.1.
		4	Pin assignments of JTAG and CSI were added to table 2-3 in section 2.2.
		5	JTAG and CSI were added to table 2-4 in section 2.3. A description was added as a note.
		6	JTAG and CSI were added to table 2-5 in section 2.4.
		7	An example of recommended connections for JTAG was added to section 2.4.1. A description was added as a note.
		9	A description was added as a note in section 2.4.2.
		11	An example of recommended connections for CSI was added to section 2.4.4.
		13	Section 2.4.7, Hot plug-in connection, was added.
		17	JTAG was added to connection interface in table 3-1 in chapter 3.
		23	The notation of JTAG was added to section 4.1.6.
		24	A description of the debugging interface was added to section 4.2.5.
		30	A description of the debugging interface was added to section 4.2.20.
		38, 39	Internal circuits were divided into Rev. C and Rev. D and descriptions related to JTAG were added in chapter 5.
		42	Statements on troubleshooting for errors in JTAG connection were added to table 6-3 in section 6.1.

E1/E20 Emulator, E2 Emulator
Additional Document for User's Manual
(Notes on Connection of RH850/F1KH and RH850/F1KM)

Publication Date: Rev.1.00 Jul.01.17
Rev.1.10 Jan.15.18
Rev.1.20 Oct.01.18
Rev.2.00 Oct.09.20
Rev.3.00 May.30.25

Published by: Renesas Electronics Corporation

E1/E20 Emulator, E2 Emulator

Additional Document for User's Manual

(Notes on Connection of RH850/F1KH and RH850/F1KM)

Renesas Electronics Corporation

R20UT3985EJ0300