User's Manual

RENESAS

E1/E20 Emulator

Additional Document for User's Manual
(Notes on Connection for 78K0R)

Supported Devices:
78K0R

Renesas Electronics
www.renesas.com

Rev.1.00 Sep 2011

CONTENTS

CHAPTER 1 OUTLINE

1.1. Features

E1/E20 Emulator (hereinafter referred to as E1/E20) is an on-chip debug emulator with flash programming function, which is used for debugging and programming a program to be embedded in on-chip flash memory microcontrollers. This product can debug with the target microcontroller connected to the user system, and can write programs to the on-chip flash memory of microcontrollers.

1.2 Cautions on Using E20

The functions used for debugging of the 78K0R device by using the E20 are the same as in the E1. Large trace function, characteristic functions of the E20, cannot be used. The power supply function from the E20 is not supported.

1.3 Configuration of Manuals

Documentation for the E1/E20 emulator manual is in two parts: the E1/E20 Emulator User’s Manual and the E1/E20 Emulator Additional Document for User’s Manual (this manual). Be sure to read both of the manuals before using the E1/E20 emulator.

(1) E1/E20 Emulator User’s Manual

The E1/E20 Emulator User’s Manual has the following contents:

- Components of the emulators
- Emulator hardware specification
- Connection to the emulator and the host computer and user system

(2) E1/E20 Emulator Additional Document for User’s Manual

The E1/E20 Emulator Additional Document for User’s Manual has the following contents:

- For use in hardware design, an example of connection and the interface circuit required to connect the emulator.
- Notes on using the emulator
- Software specifications and so on for using each microcomputers

CHAPTER 2 DESIGNING USER SYSTEM

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

2.1 Connecting the Emulator with the User System

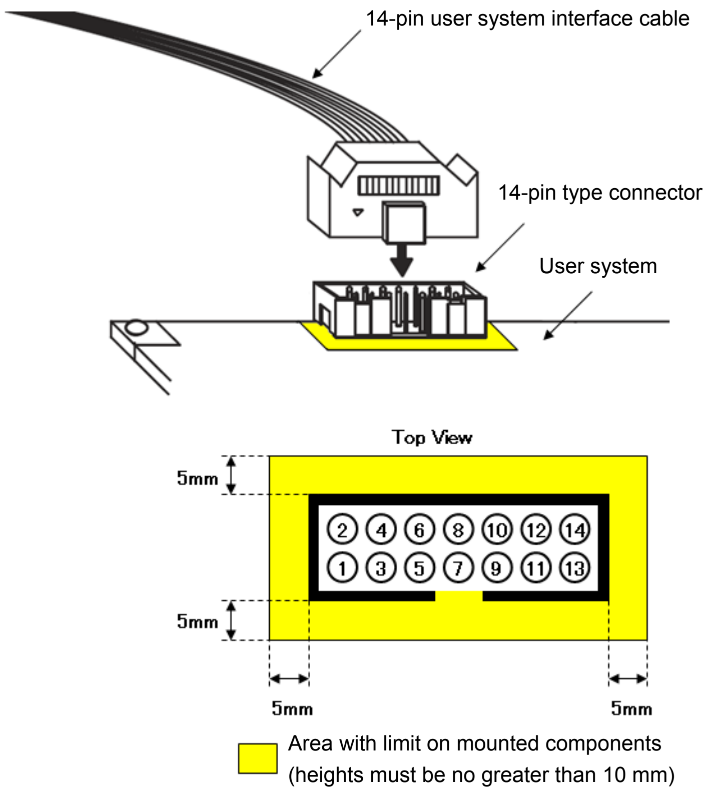
Table 2-1 shows the type numbers of the E1/E20 emulators

Table 2-1. Connector Type Numbers

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

Figure 2.1 shows examples of the connection between a user system interface cable of the 14-pin type. Do not mount other components with a height exceeding 10 mm within 5 mm of the connector on the user system. 38-pin of the E20 is not supported. To use the E20, use the 38-pin/14-pin conversion adapter [R0E000200CKA00] that comes with the E20 for connection.

Figure 2-1. Connecting the User System Interface Cable to the 14-pin Connector of the E1 Emulator


14-pin user system interface cable
14-pin type connector
User system
Top View
5mm
5mm
2 4 6 8 10 12 14
1 3 5 7 9 11 13
5mm
5mm
Area with limit on mounted components
(heights must be no greater than 10 mm)

2.2 Comumunication Mode

E1/E20 performs serial communication with the target device on the target system.
For serial communication, 1-wire mode (single-wire UART communication) using the TOOL0 pin, or 2-wire modeusing the TOOL0 and TOOL1 pins is used. Use 1-wire mode when performing flash programming. Use 1-wire modeor 2-wire mode when performing on-chip debugging. Differences between 1-wire mode and 2-wire mode are shown below.
There are no functional differences.

Table 2-2. Difference Between 1-Wire Mode and 2-Wire Mode

Communication Mode.	During Flash Programming	During Debuging
1-wire mode	No differences	User resources secured for debugging Internal ROM: 1036 bytes Internal RAM: 6 bytes (stack)
2-wire mode		User resources secured for debugging [Pseudo RRM/DMM function is used] Internal ROM: 1036 bytes Internal RAM: 6 bytes (stack) [Pseudo RRM/DMM function is not used] Internal ROM: 100 bytes Internal RAM: 6 bytes (stack)

2.3 Pin Assignments of the Connector on the User System

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

Table 2-3. Pin Assignments of the Connector on the User System (14-Pin)

Pin No.	Pin Name	Input/Output[Note 1]
1	TOOL1	Input
2	GND[Note 2]	-
3	R.F.U	-
4	FLMD0	Output
5	R.F.U	-
6	RESET_IN	Input
7	TOOL0	Output/Input
8	VDD	-
9	R.F.U	-
10	RESET_OUT[Note 3]	Output
11	R.F.U	-
12	GND[Note 2]	-
13	RESET_OUT[Note 3]	Output
14	GND[Note 2]	-

Notes **1.** As seen from E1/E20.
2. Securely connect pins 2, 12, and 14 of the connector to GND of the user system. These pins are used for electrical grounding as well as for monitoring of connection with the user system by the E1/E20.
3. Securely connect both pin 10 and pin 13. These pins are also used to monitor the user system.

Table 2-4. Pin Functions

Pin Name	Input/Output[Note]	Description
RESET_IN	Input	Pin used to input reset signal from the user system
RESET_OUT	Output	Pin used to output reset signal to the target device
FLMD0	Output	Pin used to set the target device to debug mode or programming mode.
TOOL0	Output/Input	Pin used to transmit command/data to the target device
TOOL1	Input	Pin used to input clock signal to the target device
R.F.U.	–	This pin is reserved. For the connection of the reserved pins, see each circuit related to the pins.

Note As seen from E1/E20.

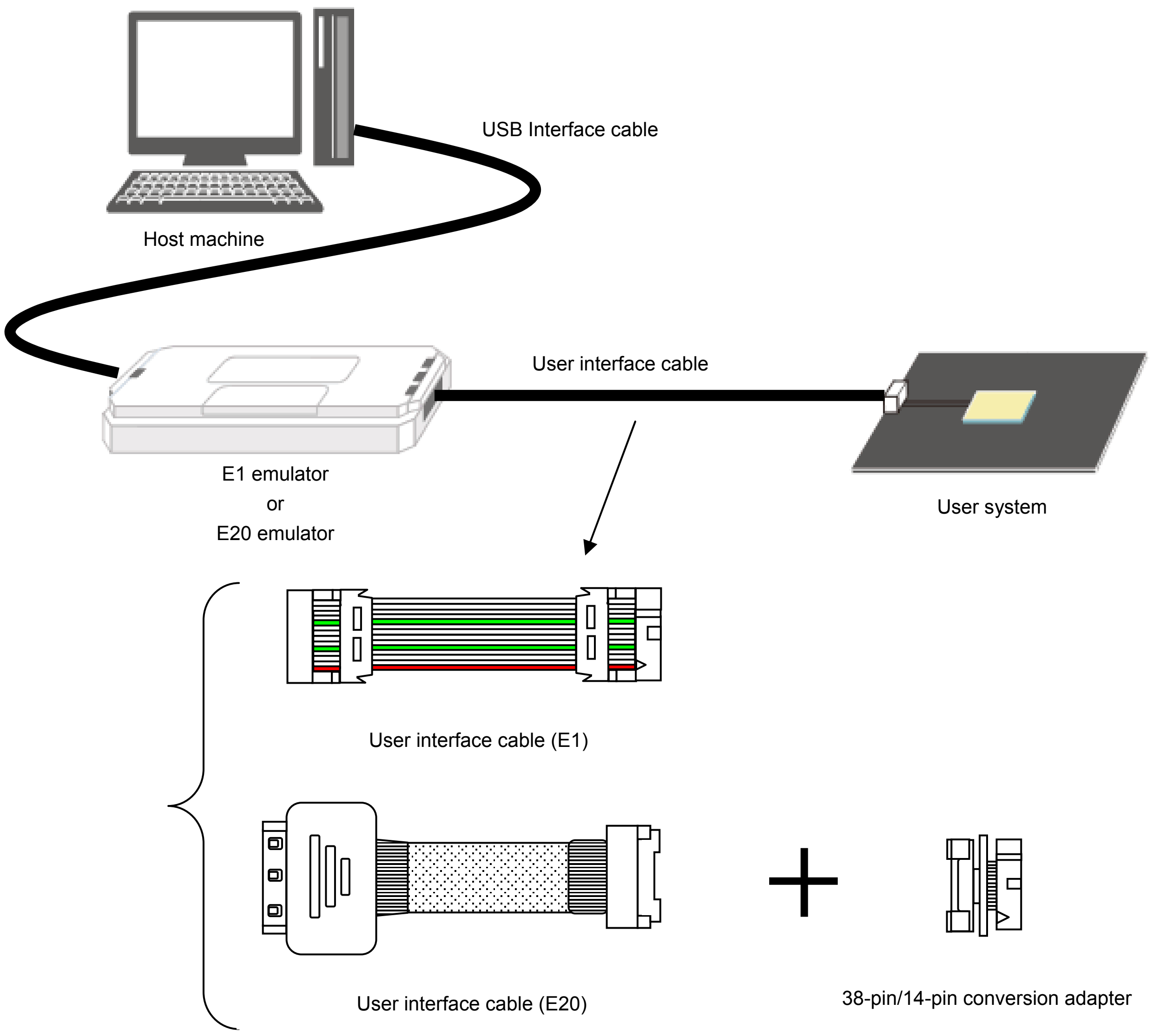
2.4 System Configuration

Figure 2-2 shows the system configuration used for the E1/E20. For cautions on connection, refer to the E1/E20 User's Manual. As software used on the host machine, use the "CubeSuite+" when on-chip debugging is used, or use the "Renesas Flash Programmer" for flash programming. For details, refer to the following URL's.

- Integrated development environment "CubeSuite+" website
  http://www.renesas.com/cubesuite+

- Flash writing tool "Renesas Flash Programmer" website
  http://www.renesas.com/rfp

Figure 2-2. Connection Diagram of E1/E20



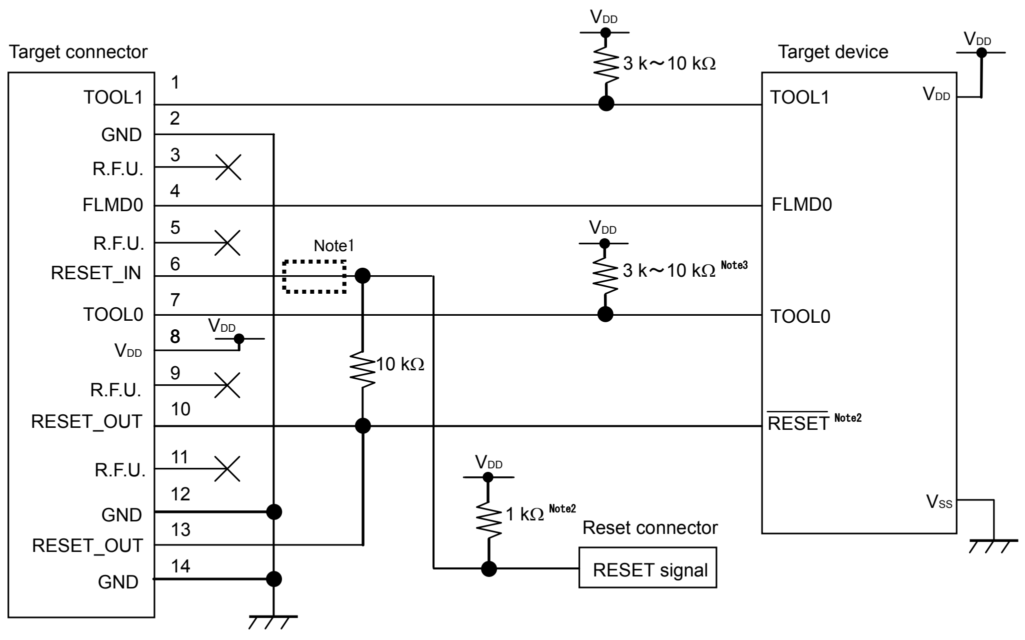
Remark To use it with the E20, connect the 38-pin/14-pin conversion adapter to the user interface cable (E20). 38-pin is not supported.

2.5 Recommend Circuit between Connector and MCU

2.5.1 Recommend Circuit Connection

Refer to 2-3 and design an appropriate circuit.
Be sure to take into consideration the specifications of the target device as well as measures to prevent noise when designing your circuit.

Figure 2-3. Recommend Circuit Communication



Notes 1. The circuit enclosed by a dashed line is not required when only flash programming is performed.

2. Refer to **2.5.2 connection of reset pin (1) Automatically switching the reset signal via resistor** about the pull-up resistor value of the reset circuit.

3. This is for pin processing when not used as a device.

Caution

- The circuits and resistance values listed are recommended but not guaranteed. Determine the circuit design and resistance values by taking into account the specifications of the target device and noise. For flash programming for mass production, perform sufficient evaluation about whether the specifications of the target device are satisfied.
- For processing of pins not used by the E1/E20, refer to the user's manual of the device.
- Securely connect pins 2, 12, and 14 of the connection to GND of the user system. These pins are used for electrical grounding as well as for monitoring of connection with the user system by the E1/E20.
- Securely connect both pin 10 and pin 13. These pins are also used to monitor the user system.

2.5.2 Connection of reset pin

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

During on-chip debugging, a reset signal from the target system is input to E1/E20, masked, and then output to the target device. Therefore, the reset signal connection varies depending on whether E1/E20 is connected.

For flash programming, the circuit must be designed so that the reset signals of the user system and E1/E20 do not conflict.

Select one of the following methods and connect the reset signal in the circuit. The details of each method are described on the following pages.

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

(2) Manually switching the reset signal with jumper

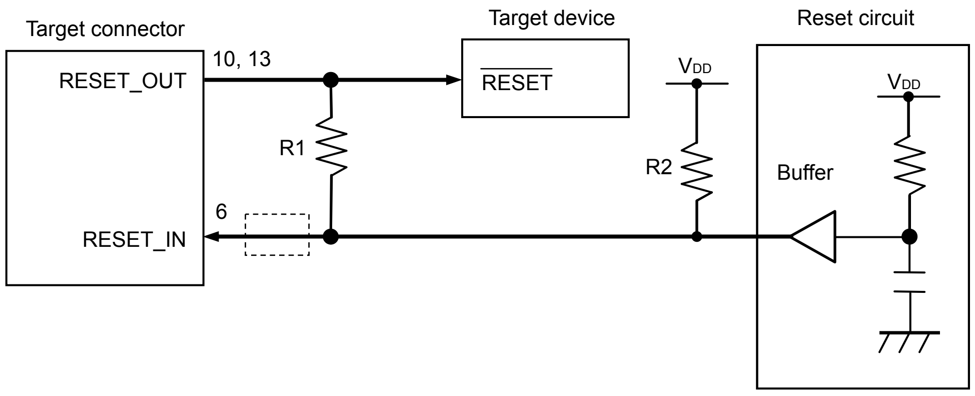
(3) Resetting the target device by power-on clear (POC) only

(1) Automatically switching the reset signal via resistor

Figure 2-4 illustrates the reset pin connection described in **2.5.1 Circuit connection examples**.

This connection is designed assuming that the reset circuit on the target system contains an N-ch open-drain buffer (output resistance: 100 Ω or less). The V_{DD} or GND level may be unstable when the logic of RESET_IN/OUT of E1/E20 is inverted, so observe the conditions described below in **Remark**.

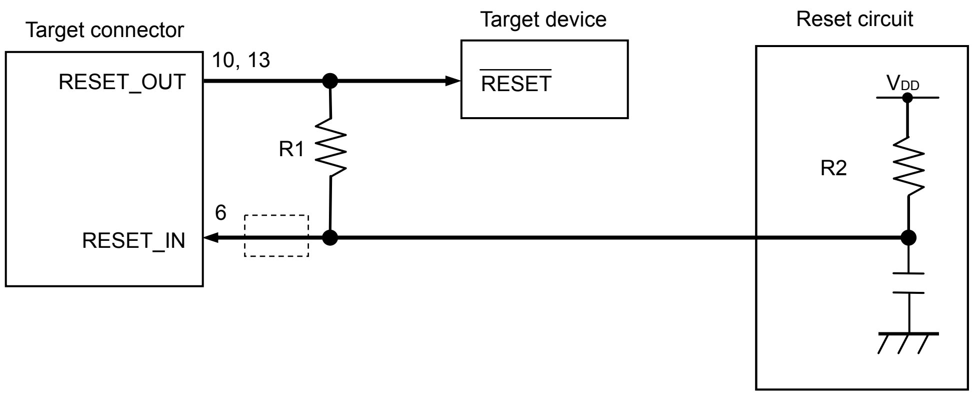
Figure 2-4. Circuit Connection with Reset Circuit That Contains Buffer



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

Figure 2-5 illustrates the circuit connection for the case where the reset circuit on the target system contains no buffers and the reset signal is only generated via resistors or capacitors. Design the circuit, observing the conditions described below in **Remark**.

Figure 2-5. Circuit Connection with Reset Circuit That Contains No Buffers

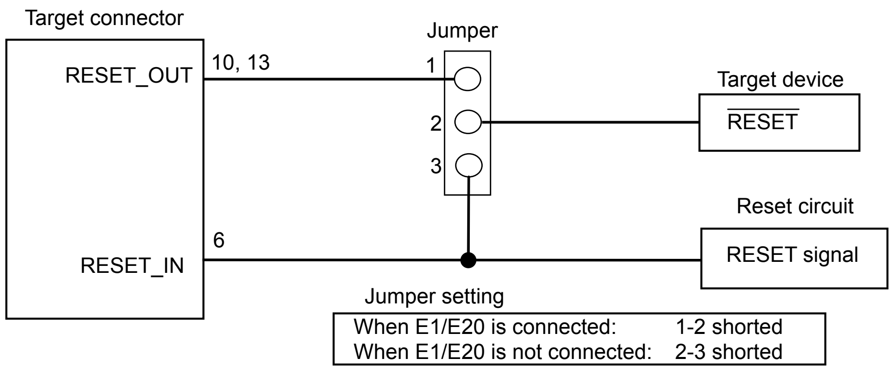


Remark Make the resistance of at least R1 ten times that of R2, R1 being 10 kΩ or more.
The circuit enclosed by a dashed line is not required when only flash programming is performed.

(2) Manually switching the reset signal with jumper

Figure 2-6 illustrates the circuit connection for the case where the reset signal is switched using the jumper, with or without E1/E20connected. This connection is simple, but the jumper must be set manually.

Figure 2-6. Circuit Connection for Switching Reset Signal with Jumper

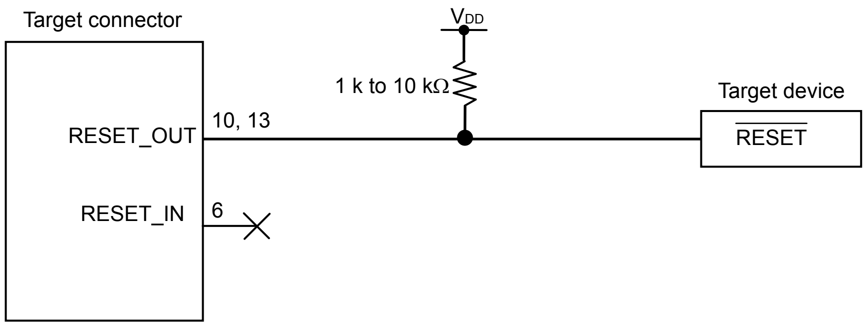


(3) Resetting the target device by power-on clear (POC) only

Figure 2-7 illustrates the circuit connection for the case where the target device is only reset via POC without using the reset pin. RESET_OUT becomes active when power is applied to E1/E20.

Even if power supply to the target system is turned off during debugging, pseudo POC function emulation is available because RESET_OUT becomes active.

Figure 2-7. Circuit Connection for the Case Where Target Device Is Only Reset via POC



CHAPTER 3 SETTING OF SECURITY ID AND SETTING OF DEBUGGING RESOURCES

The user must prepare the following to perform communication between E1/E20 emulator and the target device and implement each debug function. Refer to the descriptions on the following sections and set these items in the user program or using the build tool property.
When C-SPY manufactured by IAR Systems is used, read also the following material.
- IAR C-SPY Hardware Debugger Systems User Guide issued by IAR Systems

3.1 Setting of Security ID

This setting is required to prevent the memory from being read by an unauthorized person. Embed a security ID at addresses 0xC4 to 0xCD in the internal flash memory. The debugger starts only when the security ID that is set during debugger startup and the security ID set at addresses 0xC4 to 0xCD match. If the ID codes do not match, the debugger manipulates the target device in accordance with the value set to the on-chip debug option byte area (refer to **Table 3-2**).
If the user has forgotten the security ID to enable debugging, erase the flash memory and set the security ID again.

[How to set security ID]
A setting method of the security ID is following. When both (a) and (b) methods are done at a time, method (b) has a priority.
(a) Embed the security ID at addresses 0xC4 to 0xCD in the user program.
(b) Setting of the security ID by build tool common options. (In case of CubeSuite+)

(a) Embed a security ID at addresses 0xC4 to 0xCD in the user program.
For example If the security ID is embedded as follows, the security ID set by the debugger is ″0123456789ABCDEF1234″ (not case-sensitive).

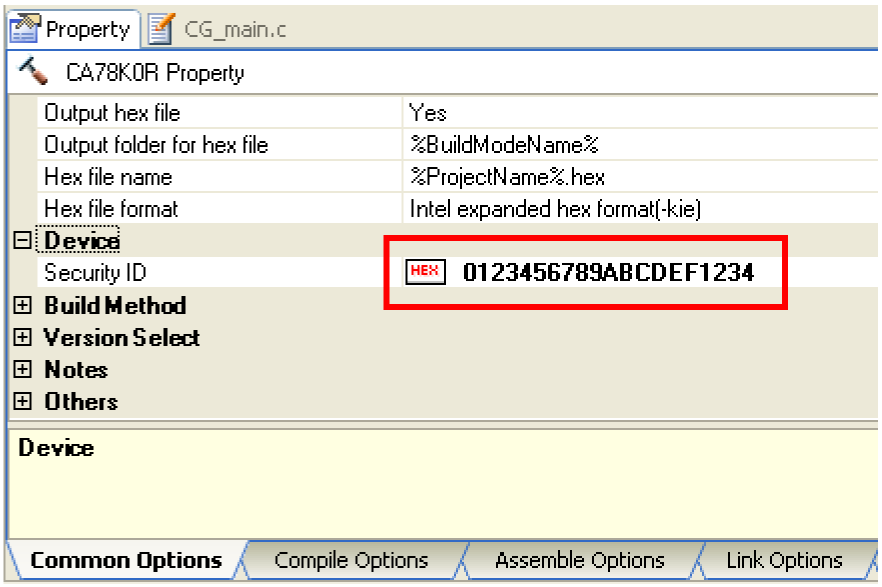
Table 3-1 Security ID

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

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

Set in “device” in the common options tab as figure 3-1.

Figure 3-1 Examples for Setting of the security ID



Output hex file	Yes
Output folder for hex file	%BuildModeName%
Hex file name	%ProjectName%.hex
Hex file format	Intel expanded hex format(-kie)
⊟ **Device**	
Security ID	HEX 0123456789ABCDEF1234
⊞ **Build Method**	
⊞ **Version Select**	
⊞ **Notes**	
⊞ **Others**	

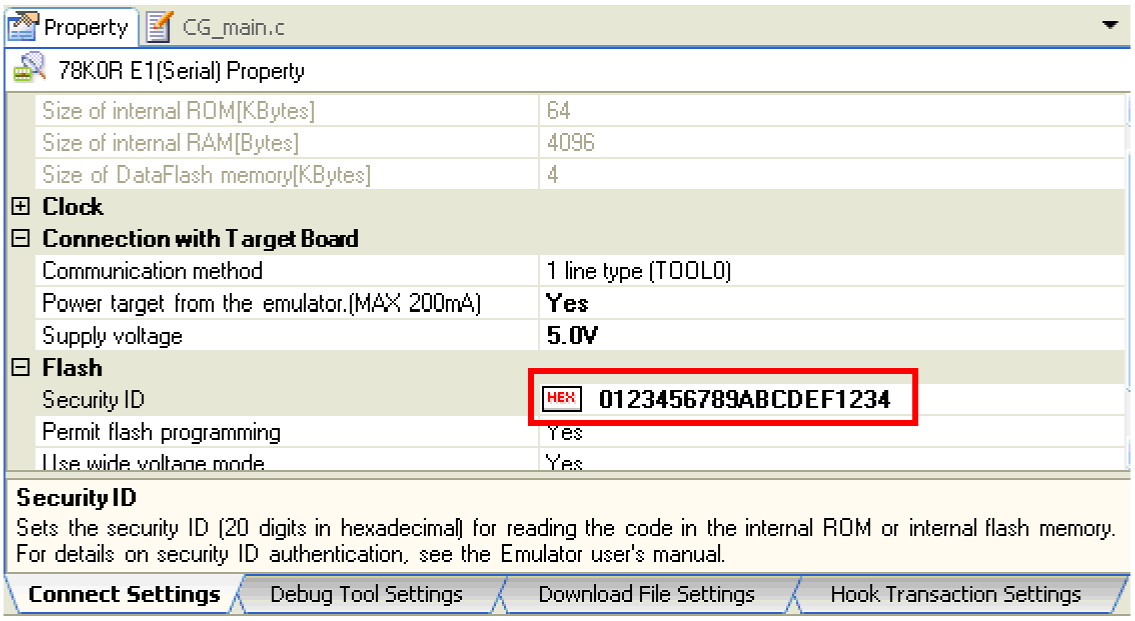
Device

[How to authenticate the security ID at debugger startup]

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

Set in “Flash” in the connect settings tab as figure 3-2.

Figure 3-2 Example for Setting of the security ID



Size of internal ROM[KBytes]	64
Size of internal RAM[Bytes]	4096
Size of DataFlash memory[KBytes]	4
⊞ **Clock**	
⊟ **Connection with Target Board**	
Communication method	1 line type (TOOL0)
Power target from the emulator.(MAX 200mA)	**Yes**
Supply voltage	**5.0V**
⊟ **Flash**	
Security ID	HEX 0123456789ABCDEF1234
Permit flash programming	Yes
Use wide voltage mode	Yes

Security ID
Sets the security ID (20 digits in hexadecimal) for reading the code in the internal ROM or internal flash memory.
For details on security ID authentication, see the Emulator user's manual.

3.2 Setting of On-chip debugging option byte

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

Table 3-2 On-Chip Debug Option Byte Setting and Operation

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

[How to secure areas]

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

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

(b) Set the On-chip debug option byte by build tool link options. (In case of CubeSuite+)

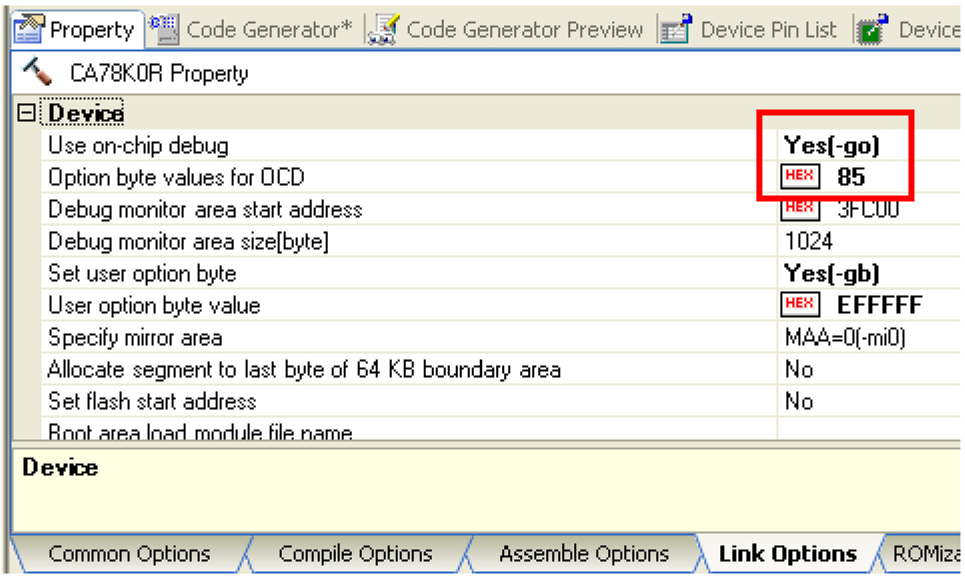
(a) Embed the On-chip debug option byte at addresses 0xC3 in the user program

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

(b) Set the On-chip debug option byte by build tool link options. (In case of CubeSuite+)

Set in "device" in the link options tab as figure 3-3.

Figure 3-3 Examples for Setting the On-chip debug option byte



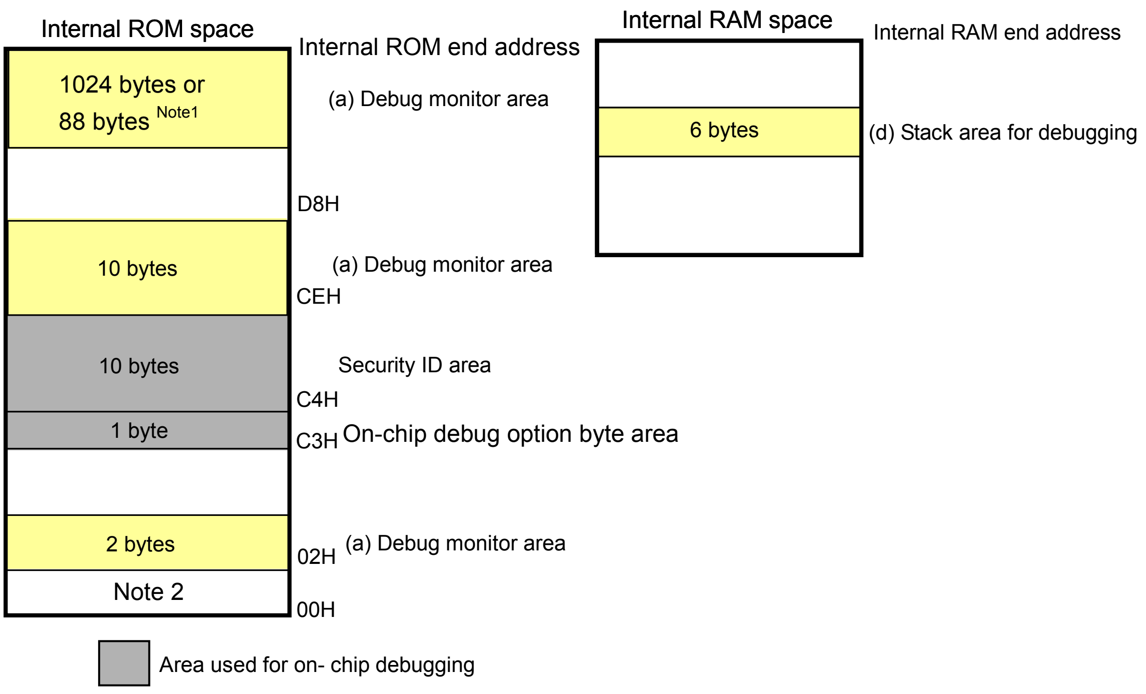
Device	
Use on-chip debug	Yes(-go)
Option byte values for OCD	85
Debug monitor area start address	3FC00
Debug monitor area size[byte]	1024
Set user option byte	Yes(-gb)
User option byte value	EFFFFF
Specify mirror area	MAA=0(-mi0)
Allocate segment to last byte of 64 KB boundary area	No
Set flash start address	No
Boot area load module file name	

Device

3.3 Securing of area for debugging

The yellow portions in Figure 3-4 are the areas reserved for placing the debug monitor program, so user programs or data cannot be allocated in these spaces. These spaces must be secured so as not to be used by the user program. Moreover, this area must not be rewritten by the user program.
Secure the resources for debugging with the contents explained by (a) and (b).

Figure 3-4 Memory Spaces Where Debug Monitor Programs Are Allocated



Note 1. When the pseudo RRM function is not used during 2-wire mode, it will be 88 bytes.

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

(a) Securing of debug monitor area

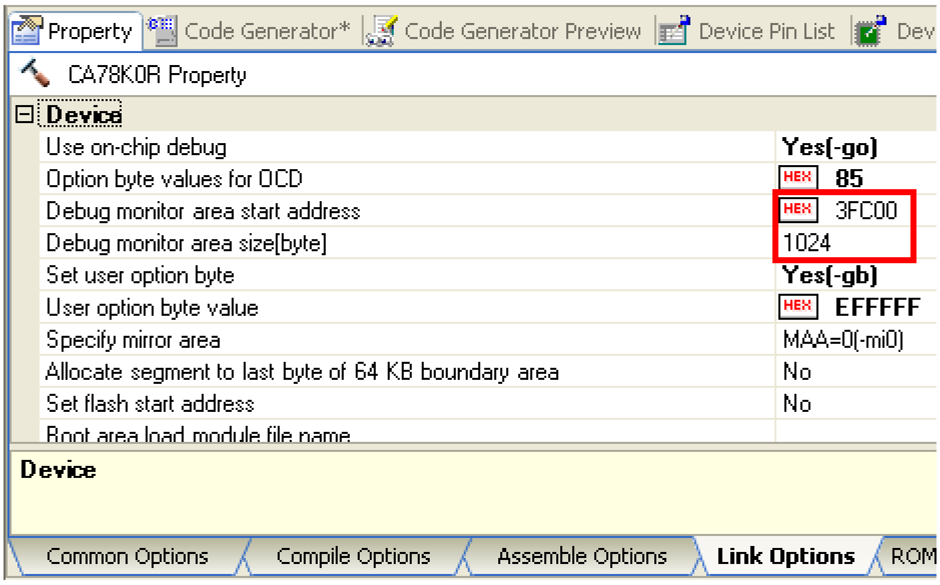
This is the area to which the debug monitor program is to be allocated. The monitor program performs initialization processing for debug communication interface and RUN or break processing for the CPU.
This user programs or data must not be placed in an area of 22 bytes near the on-chip debug option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

Note It is an area of 88 bytes when the pseudo RRM/DMM function is not used during debugging in 2-wire mode. If the internal ROM end address is 0x3FFFF, a monitor program of 88 bytes is allocated to the area from 0x3FFA8 to 0x3FFFF.

[How to secure areas]

It is not necessarily required to secure this area if the user program does not use this area.
However To avoid problems that may occur during the debugger startup, it is recommended to secure this area in advance, using the compiler. Figure 3-5 shows example for securing the area, using the CubeSuite+. Set in “device” in link options tab as figure 3-5.

Figure 3-5 Example for securing the debug monitor area



Device	
Use on-chip debug	Yes(-go)
Option byte values for OCD	HEX 85
Debug monitor area start address	HEX 3FC00
Debug monitor area size[byte]	1024
Set user option byte	Yes(-gb)
User option byte value	HEX EFFFFF
Specify mirror area	MAA=0(-mi0)
Allocate segment to last byte of 64 KB boundary area	No
Set flash start address	No
Boot area load module file name	

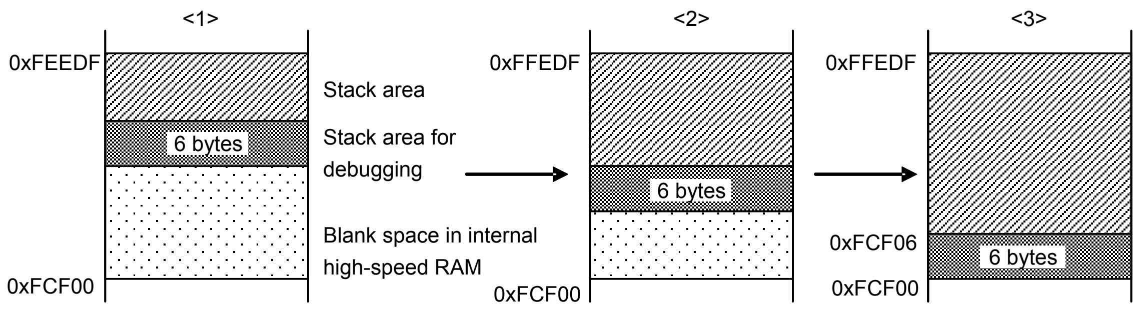
Device

(b) Securing of stack area for debugging

This area requires 6 bytes as the stack area for debugging Note. Since this area is allocated immediately before the stack area, the address of this area varies depending on the stack increase and decrease. That is, 6 extra bytes are consumed for the stack area used.
Figure 3-6 illustrates the case where the stack area is increased when the internal high-speed RAM starts from 0xFCF00.

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

Figure 3-6 Variation of Address of Stack Area for Debugging



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

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

CHAPTER 4 SPECIFICATIONS

Specifications are below table.

Table 4-1. E1/E20 Specification List

Large Item	Middle Item	Small Item	Specification	
			E1	E20
Hardware Common	Target host machine		Computer equipped with a USB port OS depends on the software.	←
	User system interface		14-pin connector	←
	Host machine interface		USB2.0 (Full speed/High speed)	←
	Connection to the user system		Connection by the provided user-system interface cable	←
	Power supply function		3.3 V or 5.0 V, set in software tool, can be supplied to the user system (with current up to 200 mA)	Cannot supply power.
	Power supply for the emulator		No need (the host computer supplies power through the USB)	←
Related debugging	Break	Software break	2000 points	←
		Hardware break	1 point (commonly used by execution and access)	←
		Forced break	Available	←
	Event	Number of events	1 point (commonly used by execution and access)	←
		Available function	Hardware break only	←
	Trace		Unavailable	←
	Performance measurement	Measurement item	From run to break	←
		Performance	Resolution 100 μs, Max. measurement time 100 hours	←
	Pseudo realtime RAM monitor (RRM)		Available (CPU is used when monitoring)	←
	Dynamic memory modification (DMM)		Available (CPU is used when changing)	←
	Hot plug-in		Unavailable	←
	Security		10-byte ID code authentication	←
Related programming	Clock supply		Clock mounted on the user system can be used	←
	Security flag setting		Available	←
	Standalone operation		Unavailable (must be connected to host machine)	←

CHAPTER 5 NOTES ON USAGE

This section describes cautions on use of the E1/E20 emulator. To use the E1/E20 properly, read the cautions thoroughly.

5.1 Lists

Table 5-1. List of Notes on Usage

No.	Item
1	Handling the device used for debugging
2	Flash self programming
3	Operation after a reset
4	Debugging with real machine running without using E1/E20
5	Operation when debugger starts
6	Debugging after program is written by flash programming
7	LVI default start function setting (address C1H)
8	On-chip debugging option byte setting (address C3H)
9	FLMD0 pin output status while debugger is running
10	Operation at voltage with which flash memory cannot be written
11	Debugging in 1-wire mode
12	Pseudo real-time RAM monitor function
13	Relation between Standby function and Break function
14	Cautions on using step-in (step execution)
15	Step-in (step execution) of Division operation

5.2 Details

No.1 Handling of device that was used for debugging

Do not mount a device that was used for debugging on a mass-produced product. (Because the flash memory was rewritten during debugging and the number of rewrites of the flash memory cannot be guaranteed.) Do not embed the monitor program for debugging in a mass-produced product.

No.2 Flash self programming

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

No.3 Operation after a reset

After an external pin reset or internal reset, the monitor program performs debug initialization processing. Consequently, the time from reset occurrence until user program execution differs from that in the actual device operation. If “No” is selected in Permit flash programming in property of the debug tool, the time until the user program is executed compared with the time when “Yes” is selected is delayed several 100 ms.

No.4 Debugging with real machine running without using E1/E20

If debugging is performed with a real machine running, without using E1/E20, write the user program using the Renesas Flash Programmer. Programs downloaded by the debugger include the monitor program, and such a program malfunctions if it includes processing to make the TOOL0 pin low level.

No.5 Operation when debugger starts

When the debugger is started, if “Communicatuin method” in the property of the debug tool is different from the setting for the previous debugging, the internal flash memory is erased.

No.6 Debugging after program is written by flash programming

If a program is written to the internal flash memory using the Renesas Flash Programmer or PG-FP5, debugger erase internal flash ROM memory automatically and download the program to the memory area.

No.7 LVI default start function setting (address C1H)

During debugging, the debug monitor program stops the LVI default start function at address C1H. Consequently, the LVI default start function is kept stopped even after debugging is completed, unless the setting to address C1H is changed through flash programming.

No.8 On-chip debugging option byte setting (address C3H)

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

No.9 FLMD0 pin output status while debugger is running

In accordance with the setting in Permit flash programming in property of the debugger, the FLMD0 pin output status while the debugger is running changes as follows. Rewriting by flash self-programming is not possible when the output status is low level.

- When “Yes” is selected: High level (low level for about 100 μs after reset release)
- When “No” is selected: Low level

No.10 Operation at voltage with which flash memory cannot be written

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

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

<1> Writing to internal flash memory
<2> Setting or canceling of software breakpoint
<3> Starting execution at the set software breakpoint position
<4> Step execution at the set software breakpoint position
<5> Step-over execution, Return Out execution
<6> Come Here
<7> If “Yes” is selected in Permit flash programming in property of debug tool, the following operations cannot be performed.
a) Setting, changing, or canceling of hardware breaks
b) Masking/unmasking of internal reset
c) Switching of peripheral breaks

No.11 Debugging in 1-wire mode

In the condition that debugging is performed in 1-wire mode, when the internal high-speed oscillator is used for the CPU operating clock, breaks may not occur normally if the frequency variation between debugger startup and break occurrence (except for when changing the register) is too large. This situation may occur when the variation of operating voltage or temperature is too large.

No.12 Pseudo real-time RAM monitor function

Note the following points when using the pseudo real-time RAM monitor function.

<1> Standby mode (HALT or STOP) may be cancelled during monitoring.
<2> The pseudo real-time RAM monitor function does not operate while the CPU operating clock is stopped.
<3> If the targets to be monitored are too numerous, the operability of the debugger may be affected because the monitoring speed is slow when using the pseudo RRM function in 1-wire mode. When using the CubeSuite+, therefore, monitoring by using the Watch panel, rather than the Memory panel, is recommended.

No.13 Relation between Standby function and Break function

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

- Stops execution of the user program.
- Step execution of the standby instruction (Stops user program after execution instruction)
- Pseudo real-time RAM monitor function (Break When Readout)
- Pseudo Dynamic Memory Modification (Break When Write)
- Breakpoint setting executing of the user program.

No.14 Cautions on using step-in (step execution)

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

Stepping into code:	Instructions in the user-created program are executed one by one.
Continuous execution:	The user-created program is executed from the current PC value.

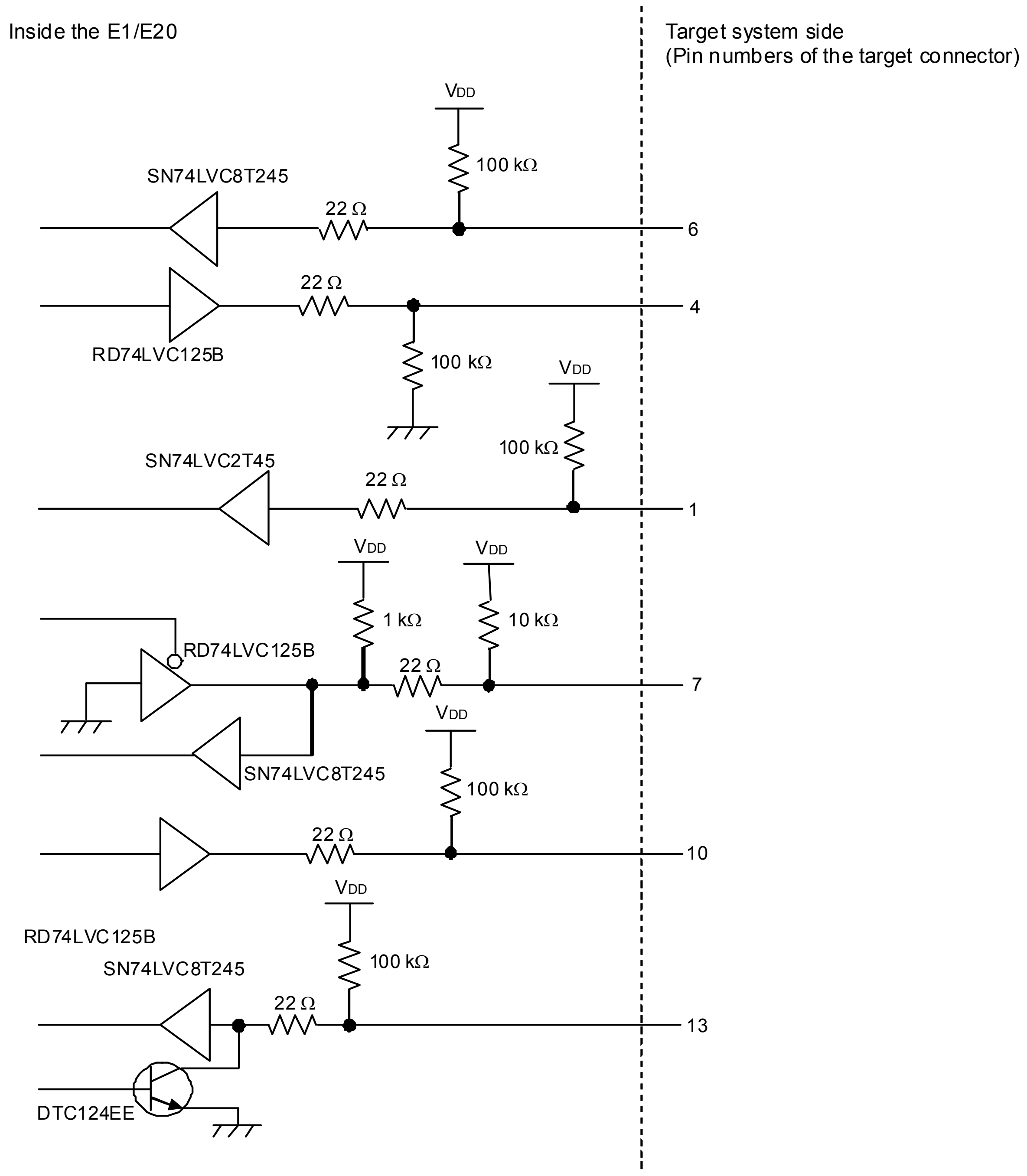
No.15 Step-in (step execution) of Division operation

When the instruction which sets (1) the bit 0 (DIVST) of Multipllcation/Division control register (MDUC) is stepped, the division operation is not finished.
The step execution of the division operation by a C source level is not relevant.

APPENDIX EQUIVALENT CIRCUIT FOR E1/E20-78K0R CONNECTION

The internal equivalent circuit related to the communication interface between the E1/E20 and user system is shown below. An example of circuit connection for the user system is shown in this document. Please use it as a reference when determining parameters in board design.

Figure A-1. E1/E20 Equivalent Circuit



E1/E20 Emulator
Additional Document for User's Manual
(Notes on Connection for 78K0R)

Publication Date: Sep 21, 2011 Rev.1.00

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

E1/E20 Emulator
Additional Document for User’s Manual
(Notes on Connection for 78K0R)

Renesas Electronics Corporation

R20UT0781EJ0100