User's Manual

E8a Emulator

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

Supported Devices:
R8C Family / R8C/Mx Series
R8C/M11A, R8C/M12A and R8C/M13B

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

Renesas Electronics
www.renesas.com

Rev.2.02 Dec 2011

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

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

Contents

1.Inside the E8a Emulator User’s Manual

The E8a manual consists of two documents: the E8a User’s Manual and the E8a Additional Document for User’s Manual (this document). Be sure to read BOTH documents before using the E8a emulator.
In this user’s manual, the symbol # is used to show active LOW. (e.g. RESET#)

(1) E8a Emulator User’s Manual
The E8a Emulator User’s Manual describes the hardware specifications and how to use the emulator debugger.

- E8a emulator hardware specifications
- Connecting the E8a emulator to the host computer or user system
- Operating the E8a emulator debugger
- Tutorial: From starting up the E8a emulator debugger to debugging

(2) E8a Additional Document for User’s Manual
The E8a Additional Document for User’s Manual describes content dependent on the MCUs and precautionary notes.

- MCU resources used by the E8a emulator
- Example of the E8a emulator connection or interface circuit necessary for designing the hardware
- Notes on using the E8a emulator
- Setting the E8a emulator debugger during startup

Note:
For the specifications and supported MCUs of the optional FDT, please check the Flash Development Tool Kit page of our website (http://www.renesas.com/tools).
FDT stands for the Flash Development Toolkit.

Trademarks
Microsoft, MS-DOS, Visual SourceSafe, Windows and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
All other company or product names are the trademarks or registered trademarks of their respective owners.

2. E8a Emulator Specifications

2.1 Target MCUs

Table 2.1 shows the target MCUs covered in this user's manual.

Table 2.1 Target MCUs

Item	Description
Target MCUs	R8C Family R8C/Mx Series R8C/M11A, R8C/M12A and R8C/M13B Groups
Available operating modes	Single-chip mode

2.2 Emulator specifications

Table 2.2 shows the specifications of the emulator supported by the R8C E8a Emulator Debugger. Table 2.3 shows the E8a emulator specifications when using the target MCU.

Table 2.2 E8a Emulator Specifications

Item	Description		
Emulator power supply	Unnecessary (USB bus powered, power supplied from the host machine)		
Applicable emulator debugger	R8C E8a Emulator Debugger V.1.05.00 or later		
Operating Environment	Temperatures	Active Inactive	: 10°C to 35°C : –10°C to 50°C
	Humidity	Active Inactive	: 35% RH to 80% RH, no condensation : 35% RH to 80% RH, no condensation
	Vibrations	Active Inactive Transportation	: maximum 2.45 m/s^2 : maximum 4.9 m/s^2 : maximum 14.7 m/s^2
	Ambient gases	No corrosive gases	

Table 2.3 E8a Emulator Specifications when Using the Target MCU

Item	Description
Power voltages	1.8 - 5.5 V For details, refer to the hardware manual of the MCU.
Break functions	- Address match break, 4 points - Data access break, 1 point - Event A: Comparison with the address/data mask, and access condition (R, W, R/W) can be set. - PC break points (maximum 255 points) - Forced break
Trace functions	3 branch instructions (branch source/destination PC) or 6 branch instructions (branch source PC) Up to 6 data cycles can be specified.
Flash memory programming function	Available (when selecting the ‘Program Flash’ mode)
User interface	1-line clock-asynchronous serial interface (communication via MODE pin)
MCU resources to be used	- Stack 8 bytes - Address match interrupt
Interface with host machine	USB (USB 1.1, full speed)* * Also connectable to host computers that support USB 2.0 * Operation with all combinations of host machine, USB device and USB hub is not guaranteed for the USB interface.
Power supply function	Can supply 3.3 V or 5.0 V to the user system (maximum 300 mA) [*1]

Note:

[*1] Do not use the power-supply function of the emulator when it is being used to program flash memory as part of a mass-production process. Separately supply power from the user system in accord with the specifications of the MCU.
Use FDT when you need to program flash memory during mass-production, etc.
Voltage supplied from the E8a emulator depends on the quality of the USB power supply of the host computer, and as such, precision is not guaranteed.

2.3 Applicable tool chain and third-party products

You can debug a module created by the inhouse tool chain and third-party products listed in Table 2.4 below.

Table 2.4 Applicable Tool Chain and Third-party Products

Tool chain	M3T-NC30WA V.5.20 Release 01 or later
Third-party products	TASKING M16C C/C++/EC++ Compiler V.2.3r1 or later [*1] IAR EWM16C V.2.12 or later

Note:

[*1] If the load module was created in ELF/DWARF2 format using TASKING M16C C/C++/EC++ compiler V3.0r1, the precautionary note described below must be observed when displaying member variables of the base class in the watch window.

Precautionary Note:
If any class object with a base class is defined, the following problems may occur:
Case 1: Member variables of the base class cannot be referenced directly from the class object (*1).
=>Use indirect references from the class object to refer to member variables of the base class (*2) (*3).
Case 2: If the PC value resides in any member function of a derived class, member variables of the base class cannot be referenced directly (*4).
=> Use indirect references from “this” pointer to refer to member variables of the base class (*5) (*6).

Figure 2.1 shows a code example, and Figure 2.2 shows a [Watch] window registration example.

```
///////////////////////////////////////////////////////
  *.h
        class BaseClass
        {
        public:
              int m_iBase;
        public:
              BaseClass() {
                    m_iBase = 0;
              }
              void BaseFunc(void);
        };

        class DerivedClass : public BaseClass
        {
        public:
              int m_iDerive;
        public:
              DerivedClass() {
                    m_iDerive     = 0;
              }
              void DerivedFunc(void);
        };


  *.cpp
        main()
        {
              class DerivedClass ClassObj;
              ClassObj.DerivedFunc();
              return;
        }

       void BaseClass::BaseFunc(void)
        {
              m_iBase = 0x1234;
        }

        void DerivedClass::DerivedFunc(void)
        {
              BaseFunc();
              m_iDerive     = 0x1234;
        }
///////////////////////////////////////////////////////
```

Figure 2.1 Example Code

```
///////////////////////////////////////////////////////
     Case 1: If the PC value resides in the main() function
     (1)"ClassObj.m_iBase"                     : Cannot be referenced (*1)
     (2)"ClassObj.__b_BaseClass.m_iBase"       : Can be referenced (*2)
     (3)"ClassObj"
              -"__b_BaseClass"
                    -"m_iBase"                 : Can be referenced (*3)
              -"m_iDerive"
                                   -: Expansion symbol

     Case 2: If the PC value resides in the DerivedClass::DerivedFunc() function
     (1)"m_iBase"                              : Cannot be referenced (*4)
     (2)"this->__b_BaseClass.m_iBase"          : Can be referenced (*5)
     (3)"__b_BaseClass.m_iBase"                : Can be referenced (*5)
     (4)"this"
              -"*"
                 -"__b_BaseClass"
                       -"m_iBase"              : Can be referenced (*6)
                 -"m_iDerive"
     (5)"__b_BaseClass"
              -"m_iBase"                       : Can be referenced (*6)
///////////////////////////////////////////////////////
```

Figure 2.2 Watch Window Registration Example

3. Connecting the E8a Emulator to the User System

3.1 Connector for connecting the E8a emulator and the user system

Before connecting the E8a emulator to the user system, a connector must be installed in the user system so a user system interface cable can be connected. Table 3.1 shows the recommended connector for the E8a emulator and Figure 3.2 shows E8a connecting connector pin assignments.

When designing the user system, refer to Figure 3.2 “E8a Connecting Connector Pin Assignments” and Section 3 “Connecting the E8a Emulator to the User System”.
Before designing the user system, be sure to read the E8a Emulator User’s Manual and related device hardware manuals.

Table 3.1 Recommended Connector

	Type Number	Manufacturer	Specification
14-pin connector	2514-6002	3M Limited	14-pin straight type (for use outside Japan)
	7614-6002	3M Limited	14-pin straight type (for use in Japan)

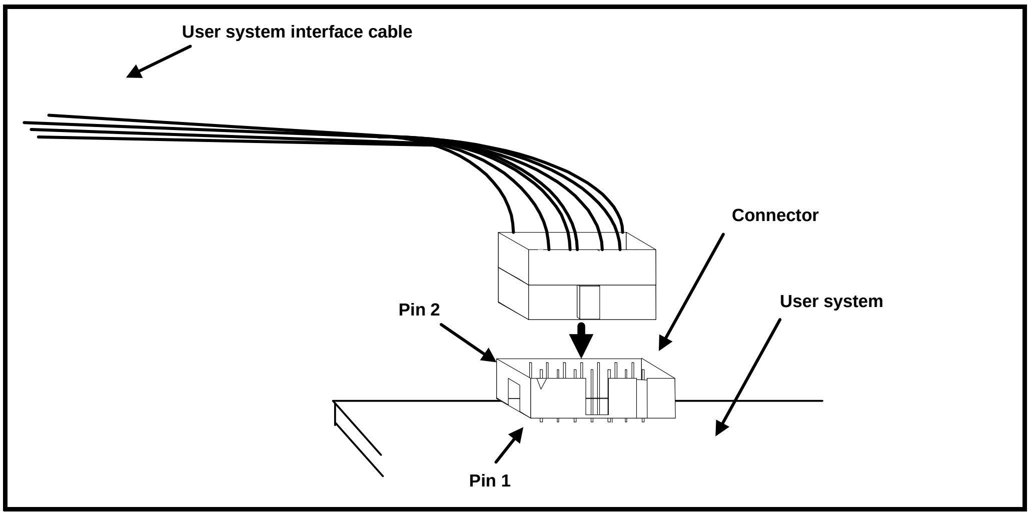
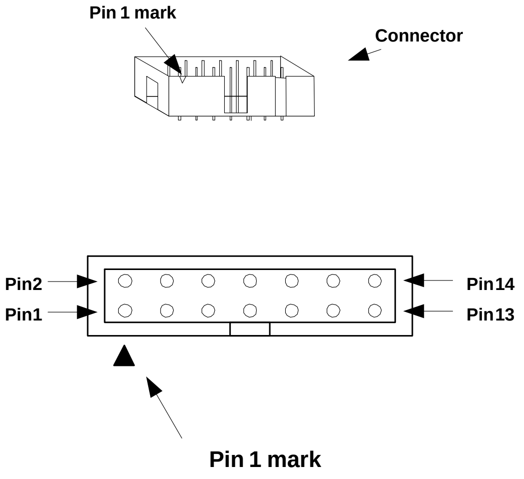


Figure 3.1 Connecting the User System Interface Cable with an E8a Connecting Connector

Notes:

- Do not place any components within 3 mm area of the connector.
- When using the E8a emulator as a programmer, connect it to the user system in the same way.
- Connect E8a connecting connector pins 2, 4, 6, 10, 12 and 14 firmly to the GND on the user system board. These pins are used as an electric GND and monitor the connection of the user system connector.
- When inserting or removing the user system interface cable from the connector section of the user system, be sure to hold the connector cover at the head of the cable. Removal by pulling the cable portion instead of grasping the cover causes breakage of the cable connection.



Pin NO	MCU Signals
1	N.C.
2	Vss
3	N.C.
4	Vss
5	N.C.
6	Vss
7	MODE
8	Vcc
9	N.C.
10	Vss
11	N.C.
12	Vss
13	RESET#
14	Vss

Figure 3.2 E8a Connecting Connector Pin Assignments

Notes:

- Pin 14 is used for checking the connection between the E8a and the user system, and is not directly connected to the Vss inside the E8a. Make sure pins 2, 4, 6, 10, 12 and 14 are all connected to the Vss.
- Note the pin assignments for the user system connector.
- Do not connect anything to the N.C. pin.

4. Examples of Pin Handling for Connecting the E8a

4.1 Examples of pin handling for connecting the E8a

Figure 4.1 shows an example of pin handling when connecting the E8a.
When using the E8a as a programmer, the connection specification between the E8a and the MCUs is the same as shown in Figure 4.1.

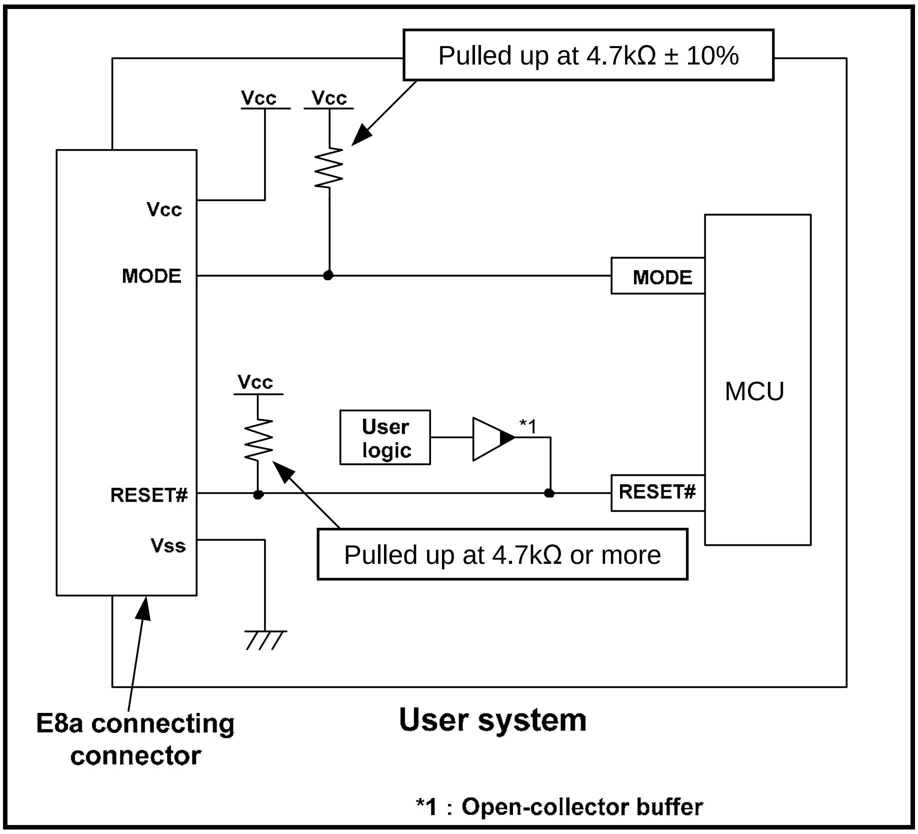


Figure 4.1 Example of an E8a Connection

Notes:

- When adjacent resistors are used for pull-up, they may be affected by noise from other pins. In particular, separate the resistor for MODE from the other resistors.
- Wiring patterns between the connector and the MCU must be as short as possible (within 50 mm is recommended). Do not connect the signal lines between the connector and MCU to other signal lines.
- For the handling of pins while the E8a emulator is not in use, refer to the hardware manual for the MCU.

(1) MODE pin

The E8a emulator uses the MODE pin for MCU control and forced break control. Pull up the E8a emulator and MCU pins and connect the E8a emulator.

Do not connect a capacitor etc. to this pin.

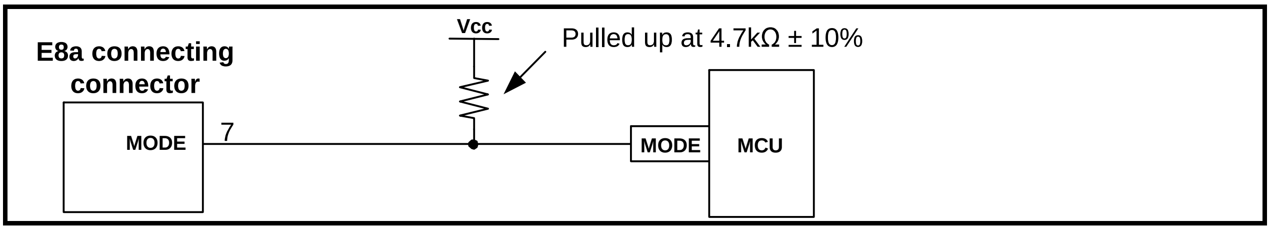


Figure 4.2 E8a Emulator and MODE Pin Connection

(2) RESET# pin

The RESET# pin is used by the E8a emulator. Therefore, use an open-collector output buffer or a CR reset circuit as the reset circuit for the user system. The recommended pull-up value is 4.7 kΩ or more.

The MCU can be reset by outputting “L” from the E8a emulator. However, if the reset IC output is “H”, the user system reset circuit cannot be set to “L”. As such, the E8a emulator will not operate normally.

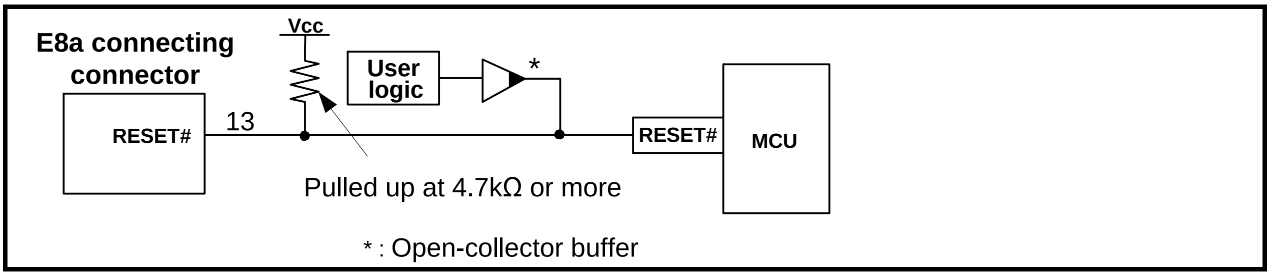


Figure 4.3 Example of a Reset Circuit

(3) Other pins

- Connect Vss and Vcc to the Vss and Vcc of the MCU, respectively.
- The amount of voltage input to Vcc must be within the specified range of the MCU.
- Pin 14 is used for checking the connection between the E8a and the user system, and pins 4, 6 and 10 are connected to the internal circuit. These pins are not directly connected to the Vss inside the E8a.
- Make sure pins 2, 4, 6, 10, 12 and 14 are all connected to the Vss.
- Do not connect anything to the N.C. pin.

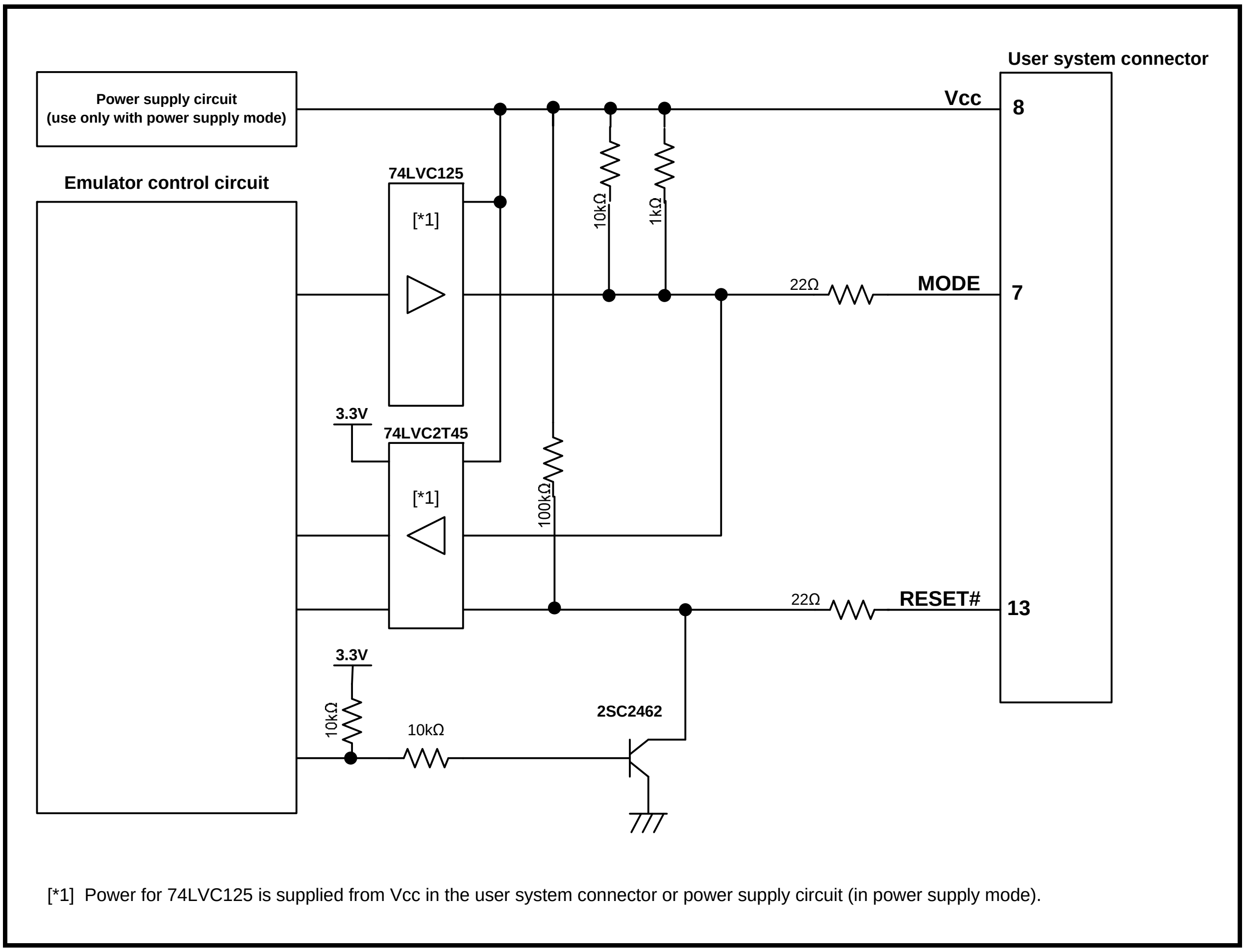
⚠ WARNING

About Power Supply Circuit of the User System:

When supplying power, ensure that there are no short circuits between Vcc and GND. Only connect the E8a emulator after confirming that there are no mismatches in pin assignments of the E8a connecting connector. Incorrect connection will result in the host computer, the emulator, and the user system emitting smoke or catching fire.

4.2 Interface circuit in the E8a emulator

Figure 4.4 shows the interface circuit in the E8a emulator. Use this figure as a reference when determining the pull-up resistance value.



[*1] Power for 74LVC125 is supplied from Vcc in the user system connector or power supply circuit (in power supply mode).

Figure 4.4 Interface Circuit inside the E8a Emulator (For Reference)

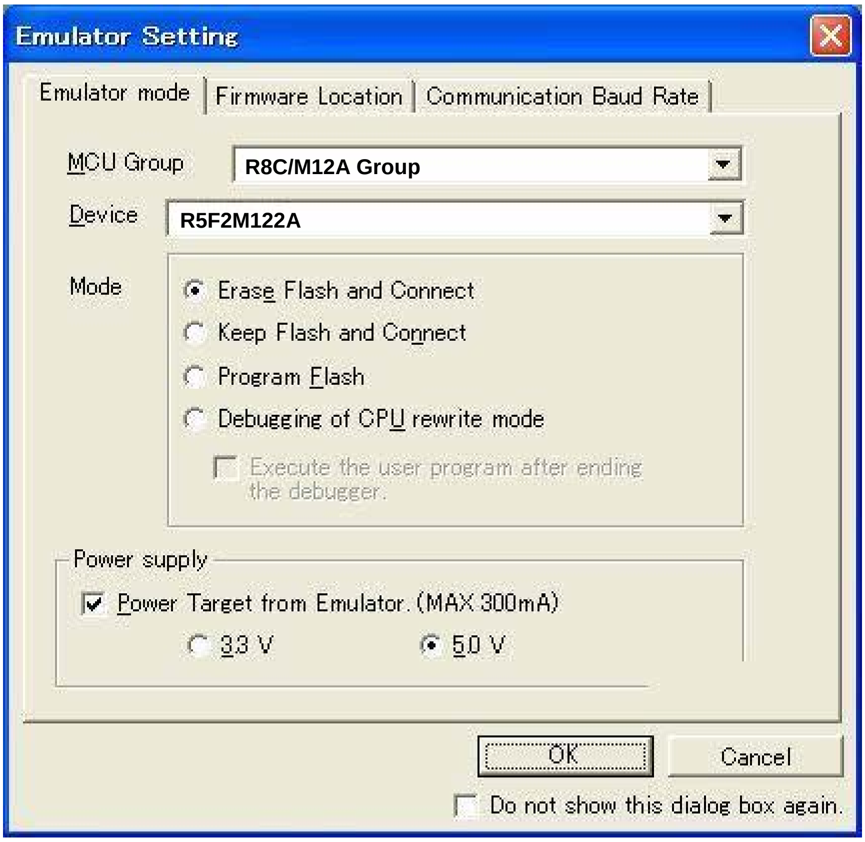
5. Emulator Debugger Setting

5.1 [Emulator Setting] dialog box

The [Emulator Setting] dialog box is provided for setting items that need to be set when the debugger is launched. The contents set from this dialog box (excluding [Power Supply] group box items) also become valid the next time the debugger is launched.

When launching the debugger for the first time after creating a new project work space, the [Emulator Setting] dialog box is displayed with the Wizard.

The settings you have made here cannot be changed after the emulator is booted up. To change the settings, you need to cancel the process of booting-up and then reboot the emulator.



MCU Group: R8C/M12A Group

Device: R5F2M122A

Mode

● Erase Flash and Connect
○ Keep Flash and Connect
○ Program Flash
○ Debugging of CPU rewrite mode
□ Execute the user program after ending the debugger.

Power supply

☑ Power Target from Emulator. (MAX 300mA)
○ 3.3 V ● 5.0 V

□ Do not show this dialog box again.

Figure 5.1 [Emulator Setting] Dialog Box

If you check “Do not show this dialog box again.” at the bottom of the [Emulator Setting] dialog box, the [Emulator Setting] dialog box will not be displayed the next time the debugger is launched.

You can open the [Emulator Setting] dialog box using one of the following methods:

- After the debugger is launched, select Menu -> [Setup] -> [Emulator] -> [Emulator Setting...].
- Hold down the Ctrl key while launching the debugger.

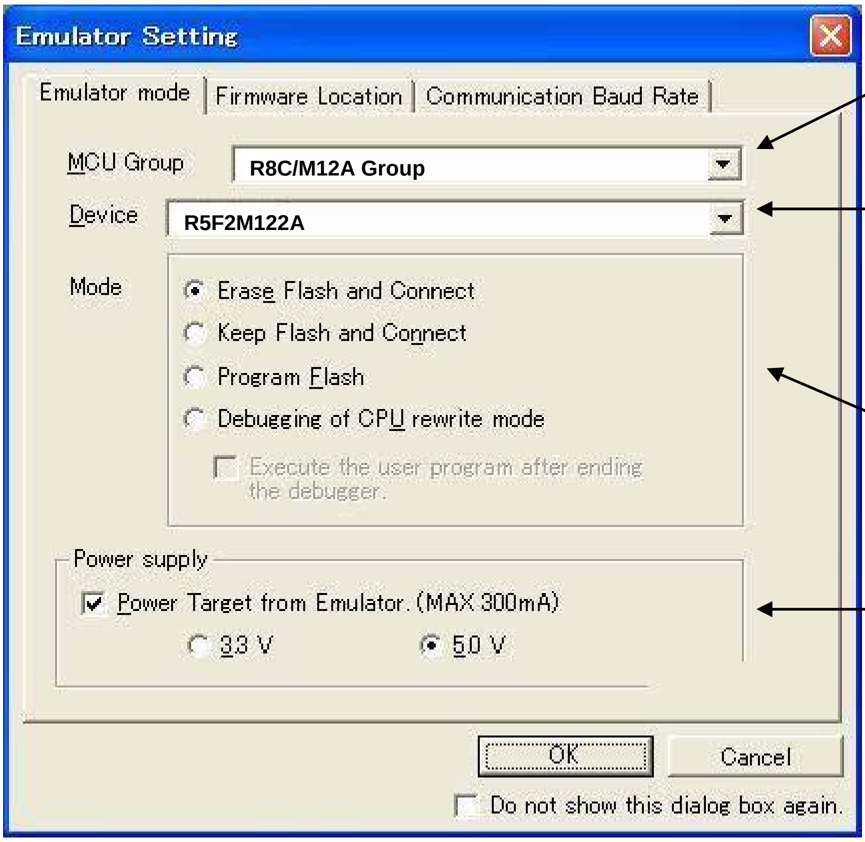
When “Do not show this dialog box again.” is checked, the E8a does not supply power to the user system.

Note:

Unsupported options are grayed out depending on the selected types of the MCU.

5.2 [Emulator mode] tab

Device selection, mode specification and power supply setting are made from the [Emulator mode] tab of the [Emulator Setting] dialog box.



Emulator Setting

Emulator mode | Firmware Location | Communication Baud Rate

MCU Group: R8C/M12A Group

Device: R5F2M122A

Mode:
- ● Erase Flash and Connect
- ○ Keep Flash and Connect
- ○ Program Flash
- ○ Debugging of CPU rewrite mode
- □ Execute the user program after ending the debugger.

Power supply
- ☑ Power Target from Emulator. (MAX 300mA)
- ○ 3.3 V ● 5.0 V

□ Do not show this dialog box again.

[MCU Group]
Select the name of the MCU group to be used from the [MCU Group] drop-down list.

[Device]
Select the type of MCU to be used from the [Device] drop-down list.
(See “7.1 MCU resources used by the E8a emulator” on page 25 for the list of applicable MCUs.)

[Mode]
Select the mode to be used.
For details, see “5.2 (1) Selecting the Mode” (p.15).

[Power supply]
Select the power supply to the user system.
- When supplying power to the user system from the E8a, click the [Power Target from Emulator. (MAX 300mA)] checkbox.
  Select either [3.3V] or [5.0V] by checking the checkbox.

Figure 5.2 [Emulator mode] Tab of [Emulator Setting] Dialog Box

(1) Selecting the Mode

Table 5.1 Selecting the Mode

Mode	Usage	Description
Erase Flash and Connect [*2]	Debugging only [*1]	When starting the debugger, the E8a emulator erases the Flash memory data for the MCUs and simultaneously writes the area for the E8a emulator program and the vector area used by the E8a emulator. The emulator rewrites the OFS, OFS2 and ID code areas.
Keep Flash and Connect [*2]		When launching the debugger, the E8a emulator retains the Flash memory data for the MCUs. Note that the area for the E8a emulator program and the vector area used by the E8a emulator will change. The emulator rewrites the OFS, OFS2 and ID code areas.
Program Flash [*2]	Simple programmer	The E8a emulator starts as a simple programmer. When downloaded, the E8a writes only the user program (E8a emulator program is not written). Therefore, in this mode, you can only download the program. You cannot change the memory contents by using the fill command, etc. When the Reset/PA0 pin is used as the PA0 pin and [Execute the user program after ending the debugger.] is selected, with the E8a emulator connected to the user system, the user program is executed at the same time the debugger is terminated. This check box setting is available only when the [Program Flash] mode is selected.
Debugging of CPU rewrite mode [*2] [*3]	Debugging only [*1]	Be sure to select this setting when debugging the program which rewrites the CPU. In this mode, the following debug operation which rewrites the Flash memory cannot be executed. - Setting the PC break points - Changing the memory contents in the Flash memory area In this mode, when starting the debugger, the E8a emulator erases the Flash memory data for the MCUs and simultaneously writes the area for the E8a emulator program and the vector area used by the E8a emulator. The emulator rewrites the OFS, OFS2 and ID code areas.

Notes:

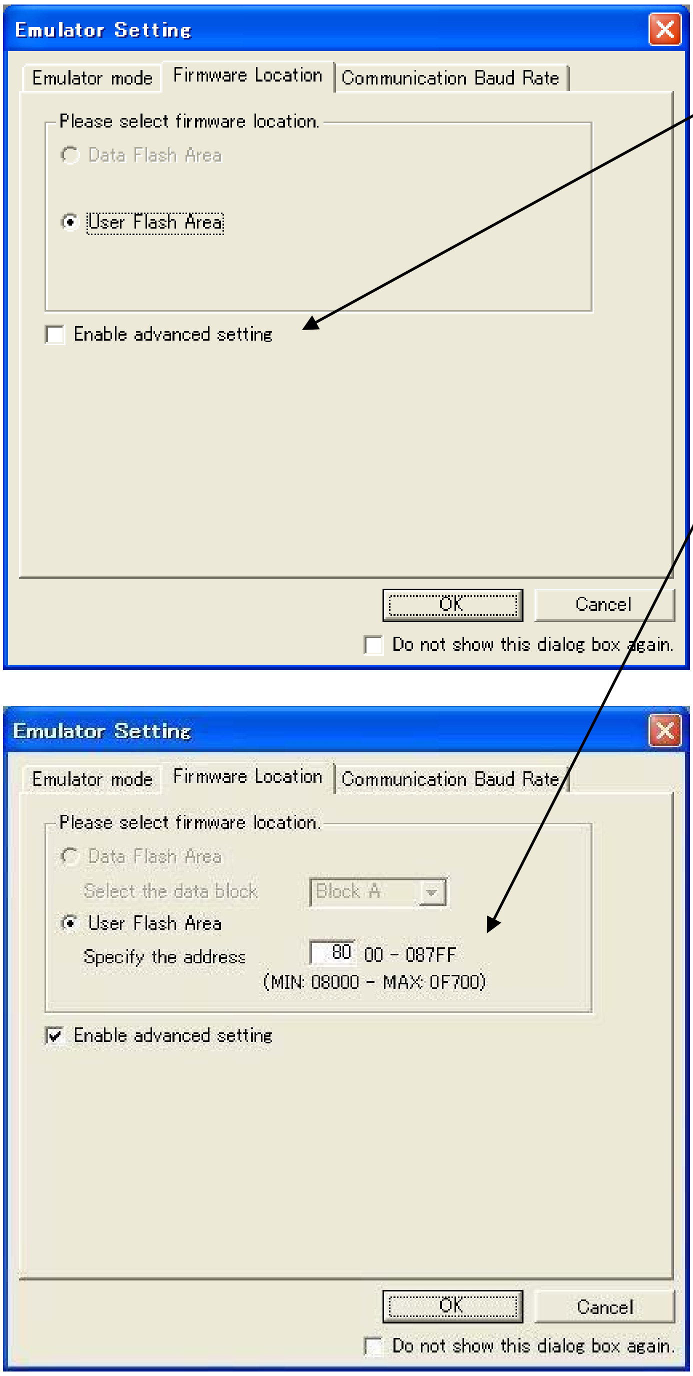
[*1] In this mode, vector addresses are used by the E8a emulator program. After a program has been downloaded, you cannot disconnect the emulator and operate the user system as a stand-alone unit. (Programs written in this mode cannot be executed from the MCU.) If you want to execute a program from the MCU, use [Program Flash] mode.

[*2] When starting up in these modes, lock bits in all the blocks of the flash memory will be unlocked. Note that the lock bits of the downloaded blocks will be unlocked after downloading the user program.

[*3] When debugging a program in CPU rewrite mode, memory reference or modification functions can be used. However, do not use these functions in the following condition.
- While write instruction is being executed to the register which requires continuous writing (ex. FMR13 bit)
The MCU does not recognize the writing is continuously executed if the write instruction is interrupted by the memory reference or modification process.

5.3 [Firmware Location] tab

You can specify the address of the firmware location in the [Firmware Location] tab.



[Firmware Location]
Select the area in which the firmware is located. Specify the address that will not be used by the user system in the ROM area or RAM area. This setting is not necessary.

Emulator Setting

Emulator mode | Firmware Location | Communication Baud Rate

Please select firmware location.

○ Data Flash Area

● User Flash Area

□ Enable advanced setting

OK Cancel

□ Do not show this dialog box again.

Emulator Setting

Emulator mode | Firmware Location | Communication Baud Rate

Please select firmware location.

○ Data Flash Area

Select the data block: Block A

● User Flash Area

Specify the address: 80 00 - 087FF

(MIN: 08000 - MAX: 0F700)

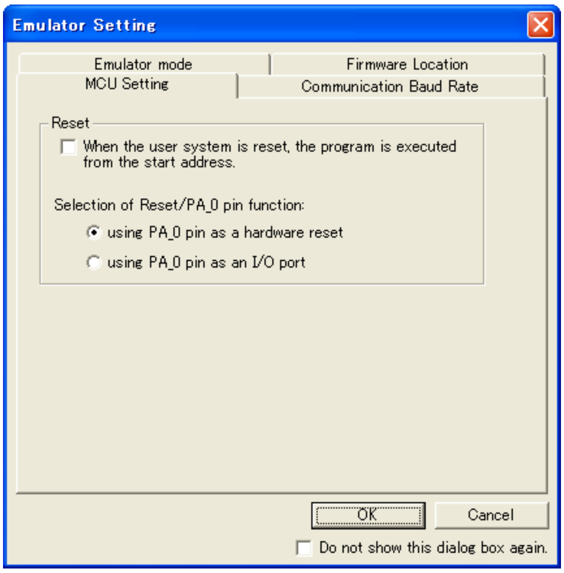
☑ Enable advanced setting

OK Cancel

□ Do not show this dialog box again.

Figure 5.3 [Firmware Location] tab of [Emulator Setting] Dialog Box

5.4 [MCU Setting] tab



Emulator Setting

Reset
□ When the user system is reset, the program is executed from the start address.

Selection of Reset/PA_0 pin function:
● using PA_0 pin as a hardware reset
○ using PA_0 pin as an I/O port

□ Do not show this dialog box again.

Figure 5.4 [MCU Setting] Tab

Selecting the program operation on reset

Select operation of the program when the user system is reset by the E8a emulator.

- When the user system is reset, the program is executed from the start address.
  When the MCU is reset (hardware set) while the user program is running, one of the following two modes of program operation can be selected.

a) When the check box is unchecked
The user program stops, with control transferred to the E8a emulator program.

b) When the check box is checked
The user program continues to run from the start address without being stopped. In this case, however, a finite time may be required, whether the user program be stopped or executed. This is because the reset vector address is rewritten with that of the user program when the user program is running, or with the one for the E8a emulator when the user program has stopped.

- Selection of Reset/PA_0 pin function:
  Since the Reset/PA0 pin of the R8C/M11A, R8C/M12A, and R8C/M13B has dual functions, it is necessary to select which function of this pin to be used in the user program [*1].

Note:

[*1] (1) In the following cases, always be sure to turn off the power to the user system temporarily before making a reconnection.
When the debugger starts up, the E8a emulator outputs "L" to the MCU via Reset/PA0 pin to reset, but it is possible that the MCU will not be correctly reset depending on its PA0 pin state, or that there will be a signal collision ("L" from the E8a while the MCU has its PA0 set to output "H").

a) When control from the E8a emulator became impossible inadvertently, due to limitations on the E8a emulator or for other reasons

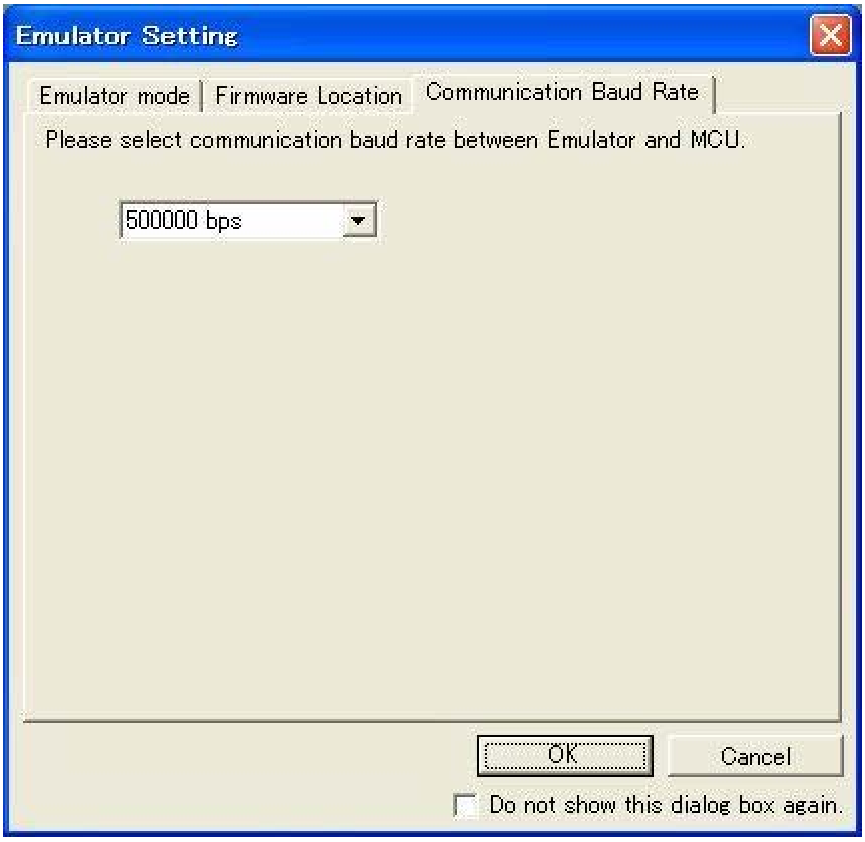
b) When used under the following condition
- Where the power for the MCU is supplied from the user system, and not from the E8a
- Where the MCU is started with its Reset/PA0 pin selected as port PA0 in the user program before it has the debugger and the Flash Development Toolkit connected.

(2) If the option "using PA_0 pin as a hardware reset" is selected on this tab, the E8a outputs "L" to the MCU via Reset/PA0 pin to reset it when the debugger closes. Also, if the option "using PA_0 pin as an I/O port" is selected, the E8a outputs no signal to the Reset/PA0 pin when the debugger closes.

(3) When the option "using PA_0 pin as an I/O port" is selected, do not use a reset from the user system. This is because control from the E8a emulator becomes impossible.

5.5 [Communication Baud Rate] tab

Select communication baud rate between the E8a and MCU in the [Communication Baud Rate] tab.
500000 bps (default setting) should be selected during normal use. [*1] [*2]



Please select communication baud rate between Emulator and MCU.

500000 bps

□ Do not show this dialog box again.

Figure 5.5 [Communication Baud Rate] Tab

Notes:

[*1] Depending on the wired length of the MODE signal and how it is wired on the user system, communication at the selected baud rate may not be performed. Reducing this baud rate may help to solve the problem.
Also, the communication information you set here cannot be changed after the emulator debugger has started. To change the communication baud rate, you need to disconnect the emulator and the MCU temporarily and then reconnect.

[*2] The baud rate of 57600 bps or below is designated for checking purpose in case there is a failure in the connection with the emulator. With such a low baud rate, it takes a long time to write into the flash memory of the target MCU, and the emulator debugger may appear to be giving no response.
Also note if the data of 1024 bytes or larger is handled when displaying the memory contents or in memory fill function, a time-out error may occur because the communication takes up much time.

6. E8a Emulator Functions (Supplement on the User’s Manual)

6.1 E8a emulator functions

With the MCUs in this user’s manual, the following functions in the device can be used.

(1) Break Function

- Address match break

This function breaks the program immediately before a specified address instruction is executed. It can be realized using the address match interrupt of the MCU. Up to 4 points of the address match break can be used.

Set the address match breakpoint in the Break condition sheet of the Eventpoints window. You can also set it by double-clicking the Event column in the Editor window.
For details, refer to the E8a User’s Manual.

- Data access break

This function breaks the program when a specified event is encountered. You can combine two points of the data access event.

- Trace full break

This function breaks the program when the trace buffer is filled

(2) Trace Function

- Branch trace

This function displays addresses, mnemonics and source lines of the branch source and destination.

- Data trace

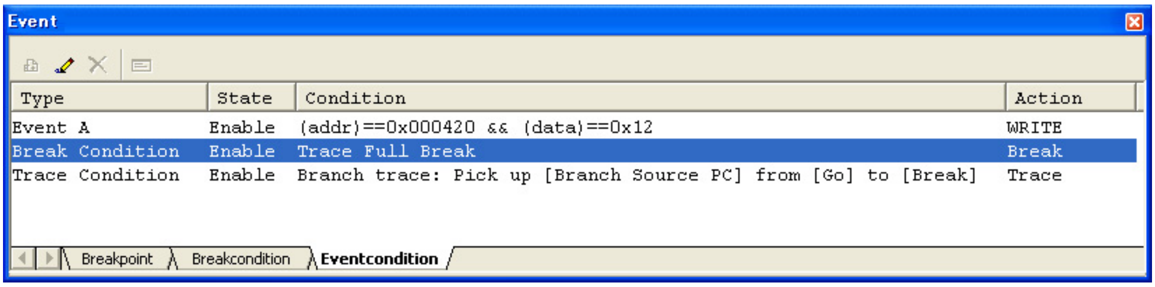
This function displays data accesses when a data access event is encountered.

For the data access event and trace condition, set them in the Event condition sheet of the Eventpoints window.

6.2 Eventcondition tab of the Eventpoints window

Set the contents of the data access event, break condition and trace condition.

Double-clicking each item in this window will open the Event Setting dialog box to change the conditions. The items displayed in the sheet are shown in Table 6.1.



Type	State	Condition	Action
Event A	Enable	(addr)==0x000420 && (data)==0x12	WRITE
Break Condition	Enable	Trace Full Break	Break
Trace Condition	Enable	Branch trace: Pick up [Branch Source PC] from [Go] to [Break]	Trace

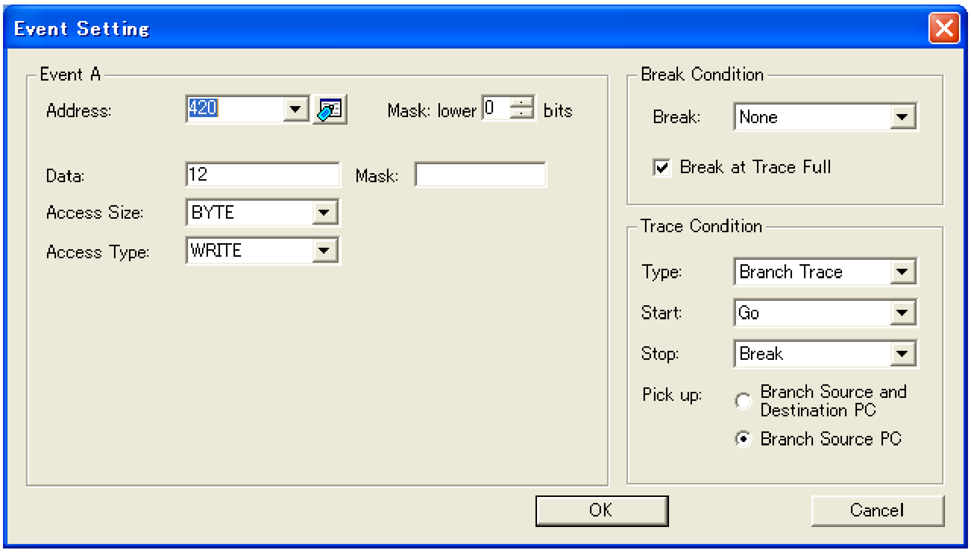
Figure 6.1 Eventpoints Window (Eventcondition tab)

Table 6.1 Display Contents of the Eventcondition Tab

Item	Description
Type	Displays the event types. - Event A - Break Condition - Trace Condition
State	Shows the event is enable or disable. - Enable - Disable
Condition	Displays the set condition.
Action	For the Event A, the access types are displayed. - R/W: READ or WRITE - READ - WRITE For the Break Condition and Trace Condition, Break/Trace is always displayed.

6.3 Event Setting dialog box

The conditions in the Event condition sheet can be set.



Event A

Address: 420

Mask: lower 0 bits

Data: 12

Mask:

Access Size: BYTE

Access Type: WRITE

Break Condition

Break: None

☑ Break at Trace Full

Trace Condition

Type: Branch Trace

Start: Go

Stop: Break

Pick up: ○ Branch Source and Destination PC

● Branch Source PC

Figure 6.2 Event Setting Dialog box

(1) Event A

Set the contents of the Event A. You can set the conditions of the address comparison with mask specification and data comparison with mask specification for the Event A.

Table 6.2 Contents of the Event A

Option	Description
Address (with mask specification)	Specify an address to detect the data access. Specify the bit number to set the address mask. The specified lower bits of the specified address are masked.
Data (with mask specification)	If you compare data, specify the data and data mask. When selecting BYTE for the Access Size, you can specify to FF. When selecting WORD for the Access Size, you can specify to FFFF. If you do not compare data, leave the Data item empty or enter 0 in the Mask. If you do not use the data mask, leave the Mask item empty.
Access Size	Select one from BYTE, WORD or Not specify for the Access Size. If a data access which does not match the specified access size occurs, the event is not encountered. When specifying WORD for the Access Size, specify the even address for the Address item.
Access Type	Select an access type. - R/W: READ or WRITE - READ - WRITE

(2) Break Condition

Set the break condition.

Table 6.3 Break Condition

Option	Description
Break	Select a break condition. - None: None specified. (No break by event) - Event A: Breaks the program when the Event A is encountered.
Break at Trace Full	Check it to break the program when the trace buffer is filled. It can be set with the break condition by event.

(3) Trace Condition

Set the trace condition.

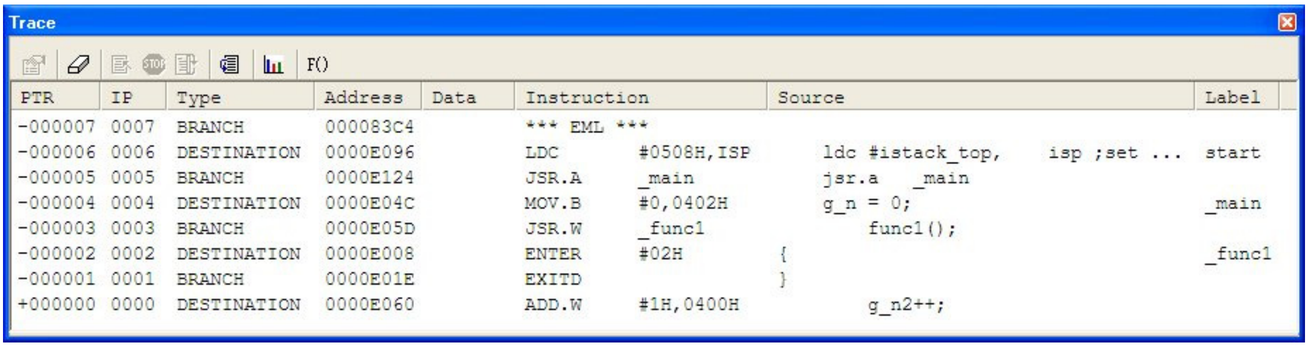
Table 6.4 Trace Condition

Option	Description
Type	Select a trace type. - Branch Trace - Data Trace
Start	Select a start condition for the trace measurement. - Go: Starts a measurement when starting executing the target program. - Event A: Starts a measurement when the Event A is encountered.
Stop	Select a stop condition for the trace measurement. - Break: Stops a measurement when stopping executing the target program. - Trace FULL: Stops a measurement when the trace data is filled. - Event A: Stops a measurement when the Event A is encountered.
Pick up	Select an event to record when tracing data. - Branch Source and Destination PC: Records only data access which encounters the condition of the Event A. (Records branch/destination information). - Branch Source PC: Records only data access which encounters the condition of the Event A. (Destination information is not recorded.)

6.4 Display contents of the Trace window

To display the trace results, open the Trace window.

For each function of the popup menu, refer to the E8a User's Manual. The items displayed in the sheet are shown in Table 6.5.



PTR	IP	Type	Address	Data	Instruction		Source	Label
-000007	0007	BRANCH	000083C4		*** EML ***			
-000006	0006	DESTINATION	0000E096		LDC	#0508H,ISP	ldc #istack_top, isp ;set ...	start
-000005	0005	BRANCH	0000E124		JSR.A	_main	jsr.a _main	
-000004	0004	DESTINATION	0000E04C		MOV.B	#0,0402H	g_n = 0;	_main
-000003	0003	BRANCH	0000E05D		JSR.W	_func1	func1();	
-000002	0002	DESTINATION	0000E008		ENTER	#02H	{	_func1
-000001	0001	BRANCH	0000E01E		EXITD		}	
+000000	0000	DESTINATION	0000E060		ADD.W	#1H,0400H	g_n2++;	

Figure 6.3 Trace Window

Table 6.5 Trace Display

Item	Description
PTR	Displays the pointer numbers in the trace buffer. Displays them in ascending order with the trace end position as 0.
IP	Displays the instruction pointer.
Type	Displays the type of trace information. When the branch trace is set, BRANCH/DESTINATION is displayed. When the data trace is set, READ/WRITE is displayed.
Address	When the branch trace is set, an address of the branch source and destination is displayed. When the data trace is set, an address or address range set for the encountered event is displayed.
Data	When the data trace is set, the accessed value is displayed. When the branch trace is set, nothing is displayed.
Instruction	When the branch trace is set, the mnemonic of the address is displayed. When the data trace is set, nothing is displayed. "*** EML ***" may be displayed in the Instruction column. This shows that the target program accessed the area of emulator use to control breaks, etc. It is not an error.
Source	If there is a source line information correspondent to the Instruction, the correspondent source line is displayed. When the data trace is set, nothing is displayed.
Label	If there is a label correspondent to an address in the Instruction, the correspondent label is displayed. When the data trace is set, nothing is displayed.

6.5 Notes on the event settings of the access break and trace function

When setting the Event A or Event B for the access break and trace function, set the address, access size and access type referring to Table 6.6 below.

Table 6.6 Availability of the Event Setting

Event setting condition	Availability of event setting	Example of Event Setting dialog box
Byte read to even address	Available	Address: 400h Access size: BYTE Access type: READ or R/W
Byte write to even address	Available	Address: 400h Access size: BYTE Access type: WRITE or R/W
Word read to even address	Available	Address: 400h Access size: WORD Access type: READ or R/W
Word write to even address	Available	Address: 400h Access size: WORD Access type: WRITE or R/W
Byte read to odd address	Available	Address: 401h Access size: BYTE Access type: READ or R/W
Byte write to odd address	Available	Address: 401h Access size: BYTE Access type: WRITE or R/W
Word read to odd address	Available	Address: 401h Access size: BYTE[*1] Access type: READ or R/W
Word write to odd address	Available	Address: 401h Access size: BYTE [*1] Access type: WRITE or R/W

Notes:

[*1] For the access size, specify "BYTE". In this condition, the lower one byte data can be compared.

[*2] Note on the trace start condition
When setting an event (other than "Go") for the trace start condition, a data when the event is encountered is not recorded to the trace data. The data of the event which is encountered the next time is recorded.

[*3] Notes on the trace stop condition
When the trace start and trace stop conditions occur simultaneously, the trace stop condition becomes invalid.
When setting other than "Break" for the trace stop condition, the display contents of the Trace window will not be updated until the user program stops even after a trace stop condition is encountered.

[*4] Note on setting the Event A
When setting an event for the Event A, you cannot specify a mask for an address and data simultaneously. If you mask them simultaneously, an event will not be encountered.

[*5] Note on setting an event
Do not specify the following addresses as the address of the event. Otherwise, an unauthorized break may occur.
- Address in the interrupt vector table
- Address set in the interrupt vector table (interrupt routine start address)
- Branch address of the branch instruction
Both fixed vector table and variable vector table are included with the interrupt vector table above.

7. Notes on Using the E8a Emulator

7.1 MCU resources used by the E8a emulator

(1) Program area for the E8a emulator

Table 7.1 lists the program areas for the E8a emulator. Do not change this area, otherwise the E8a emulator will not control the MCU. In this case, disconnect the debugger and then reconnect it.

Table 7.1 Program Area for the E8a Emulator

Group	Part No.	ROM Size		Program Area for E8a Emulator	
		Program ROM	Data Flash	Vector Area	ROM Area
R8C/M11A	R5F2M110A	2 KB	1 KB (2 blocks)	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFFCh - FFFEh	-
	R5F2M111A	4 KB			-
	R5F2M112A	8 KB			-
R8C/M12A	R5F2M120A	2 KB			-
	R5F2M121A	4 KB			-
	R5F2M122A	8 KB			-
R8C/M13B	R5F2M131B	4 KB			-
	R5F2M132B	8 KB			-
	R5F2M134B	16 KB			-

(2) Pins used by the E8a emulator

The E8a emulator controls the MCUs by using the following pins depending on the usage.

- For debugging/programming: RESET# pin and MODE pin

(3) Registers initialized by the E8a emulator

When the system is launched, the E8a emulator initializes the general registers and some of the flag registers as shown in Table 7.2.

Table 7.2 E8a Emulator Register Initial Values

Status	Register	Initial Value
E8a Emulator Activation	PC	Reset vector value in the vector address table
	R0 to R3 (bank 0, 1)	0000h
	A0, A1 (bank 0, 1)	0000h
	FB (bank 0, 1)	0000h
	INTB	00000h
	USP	0000h
	ISP	05FFh (differs from the specification of the MCU)
	SB	0000h
	FLG	0000h

(4) SFRs used by the E8a emulator program

The SFRs listed in Table 7.3 are used by the E8a emulator program as well as the user program.

- Do not change the value in the memory window, etc., by other than the user program.
- Note that although the SFRs can be changed during user program execution, the changed value cannot be read at the break.

The SFRs listed in Table 7.4 are used by the E8a emulator program, not the user program.

- Do not change the registers, otherwise the E8a cannot control the MCU.
- The SFRs listed in Table 7.3 and Table 7.4 are not initialized by selecting [Debug] -> [Reset CPU] or by using the RESET command. If register contents are referred to, a value that has been set in the E8a emulator program will be read out.

Table 7.3 SFRs Used by the E8a Emulator Program (1)

Address	Register	Symbol	Bit
0013h	Protect register	PRCR	Bit 0

Table 7.4 SFRs Used by the E8a Emulator Program (2)

Address	Register	Symbol	Bit	Notes on Using the E8a Emulator
01C0h - 01C2h	Address match interrupt register 0	AIADR0L	All bits	[*1]
		AIADR0M	All bits	[*1]
		AIADR0H	All bits	[*1]
01C3h	Address match interrupt enable register 0	AIEN0	All bits	[*1]
01C4h - 01C6h	Address match interrupt register 1	AIADR1L	All bits	[*1]
		AIADR1M	All bits	[*1]
		AIADR1H	All bits	[*1]
01C7h	Address match interrupt enable register 1	AIEN1	All bits	[*1]

Note [*1]: Do not change this register value.

(5) Stack area used by the E8a emulator

The E8a emulator uses up to 8 bytes of the stack pointer (ISP) during a user program break. Therefore, set aside 8 bytes for the stack area.

(6) Reset

The reset vector is used by the E8a emulator program. The program operation when the MCU is reset (hardware reset, software reset and watchdog timer reset) while executing the user program differs depending on the contents set on the [MCU Setting] tab at debugger startup. For details, refer to “5.4 [MCU Setting] tab” on page 17. Do not perform any reset other than the hardware reset, software reset or watchdog timer reset, otherwise the E8a emulator will run out of control. [*1]

If the automatic memory update is enabled in the memory or watch window, do not perform a hardware reset to the MCU. Otherwise the E8a emulator will run out of control.

Note:

[*1] If the reset circuit of the user system has a watchdog timer, disable it when using the emulator.

(7) Interrupts used by the E8a emulator program (unusable)

The BRK instruction interrupt, address match interrupt, single-step interrupt and address break interrupt are used by the E8a emulator program. Therefore, make sure the user program does not use any of these interrupts. The E8a emulator changes these interrupt vector values to the values to be used by the emulator. No problems occur if the interrupt vector values are written in the user program.

(8) Reserved area

The addresses not specified in the Hardware Manual of MCUs are reserved area. Do not change the contents. Otherwise, the E8a emulator cannot control the MCU.

- The value of this area is undefined when referenced in the memory window.
- In this area, the memory window’s search, compare and move functions do not work normally.

(9) Watchdog timer start select bit

If the MCU is reset during user program execution while debugging with the E8a emulator, the watchdog timer will not start automatically after the reset is released regardless of the value of the watchdog timer start select bit (WDTON).

(10) High-speed on-chip oscillator

When debugging with the E8a emulator, the high-speed on-chip oscillator does not stop although the options for high-speed on-chip oscillator oscillation enable bit are available. Also note that the timers for which fHOCO is selected as a count source do not stop even in the following cases because the high-speed on-chip oscillator is always oscillating.

- In wait mode
- In stop mode
- When the high-speed on-chip oscillator is off
  (i.e., when bit 0 (HOCOE) of OCOCR register is “0” (0: High-speed on-chip oscillator off))

(11) Option function select register (OFS)

The E8a emulator cannot debug the voltage monitor 0 reset function.

Therefore, the following bits do not change even if you set bit 6 (LVDAS: voltage detection 0 circuit start bit) of OFS to “0” (voltage monitor 0 reset enabled after reset). In other words, the values after the reset are the same as those when LVDAS is “1”.

- Bit 5 of voltage detect register 2 (VCA2):	VC0E (voltage detection 0 enable bit)
- Bit 0 of voltage monitor 0 circuit control register (VW0C):	VW0C0 (voltage monitor 0 reset enable bit)

7.2 Internal ROM area (flash memory)

7.2.1 Notes on debugging in CPU rewrite mode

(1) Unrewritable area in CPU rewrite mode

When debugging in CPU rewrite mode, do not rewrite CPU for the flash memory block containing the following area. If this area is rewritten, the E8a emulator will not control the MCU. CPU rewrite can be executed only for the data area.

- Fixed interrupt vector area

(2) Operation in CPU rewrite mode

- When debugging in the CPU rewrite mode, do not halt the user program while the CPU rewrite mode is enabled or while in the erase suspend state. And do not perform a step execution of the instruction which enables a CPU rewrite mode or enters an erase-suspend state.
  If the user program is halted, the E8a emulator may not control the MCU. In addition, disable the automatic update in the watch window or fix the display in the memory window before running the program so memory accesses do not occur during an execution.
- To check the data after executing the CPU rewrite mode, halt the program after releasing the CPU rewrite mode and refer to the memory window, etc.
  If CPU rewrite can be executed for the data area, and erase process can be suspended, do not use software breaks.
- When rewriting the Flash memory in the program area, select Menu -> [Setup] -> [Emulator] -> [System...] to open the [Configuration] dialog box in the High-performance Embedded Workshop. In this dialog box, change the [Flash memory synchronization] setting to [Flash memory to PC] and set the debugger cache to OFF. In this setting, the Flash memory is read whenever a break occurs, which takes some time. Use it with the [Disable] setting except when debugging in CPU rewrite mode.
  Setting the debugger cache to OFF is not necessary if the debugger is started in ‘Debugging of CPU rewrite mode’.

7.2.2 Notes on rewriting flash memory by the E8a emulator

Do not reset nor execute debugging operations to the MCU while the internal ROM (flash memory) is being written by the E8a emulator.

Flash memory rewrite ends when the “Flash memory write end” is displayed in the output window of the High-performance Embedded Workshop.

If the MCU is reset or debugged when rewriting the flash memory, the user program or the E8a emulator program may be disrupted.

Flash memory rewrite occurs:

- When downloading the user program
- After setting PC breaks in the flash memory and executing the user program
- After canceling PC breaks in the flash memory and executing the user program
- After rewriting the value of the flash memory in the memory window and executing the user program

7.2.3 Note on flash memory during user program execution

Modification of the internal ROM area (program ROM) attempted except from the user program (such as from the memory window) while the user program is being executed is always made to the internal cache of the E8a emulator. Actual access to the flash memory is executed before the user program restarts and immediately after it has stopped.

7.2.4 MCUs used for debugging

When debugging, the flash memory is frequently rewritten by the E8a emulator. Therefore, do not use an MCU that has been used for debugging in products.

Also, as the E8a emulator program is written to the MCU while debugging, do not save the contents of the MCU flash memory which were used for debugging nor use them as the ROM data for products.

7.2.5 Flash memory ID code

This MCU function prevents the Flash memory from being read out by anyone other than the user.

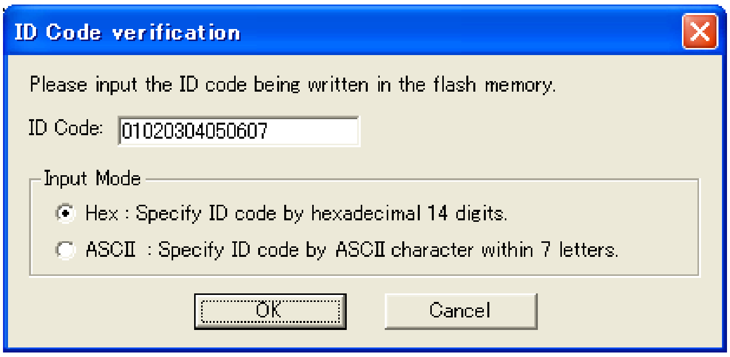
The ID code in Table 7.5 written to the flash memory of the MCU must match the ID code displayed in Figure 7.1 [ID Code verification] Dialog Box at debugger startup, otherwise the debugger cannot be launched. Note that when the ID code is FFh, FFh, FFh, FFh, FFh, FFh, FFh, the ID code is regarded as undefined. In this case, the ID code is automatically authenticated and the [ID Code verification] dialog box is not displayed.

The values written into the ID code area differs depending on the mode.

- 'Program Flash' mode [*1]:	Contents of the user program
- Modes other than 'Program Flash' mode [*2]:	FFh, FFh, FFh, FFh, FFh, FFh, FFh (regardless of the contents of the downloaded user program)

Table 7.5 ID Code Storage Area

Address	Description
FFDFh	First byte of ID code
FFE3h	Second byte of ID code
FFEBh	Third byte of ID code
FFEFh	Fourth byte of ID code
FFF3h	Fifth byte of ID code
FFF7h	Sixth byte of ID code
FFFBh	Seventh byte of ID code



Please input the ID code being written in the flash memory.

ID Code: 01020304050607

Input Mode

● Hex : Specify ID code by hexadecimal 14 digits.

○ ASCII : Specify ID code by ASCII character within 7 letters.

Figure 7.1 [ID Code verification] Dialog Box

Notes:

[*1] Notes on 'Program Flash' mode:
When the ID code is specified by the -ID option of the lmc30, download the MOT file or HEX file.

When the X30 file is downloaded, the ID code is not valid. When downloading the X30 file, specify the ID code using an assembler directive command such as ".BYTE".

The file to which the ID code specified by the assembler directive command ".ID" is output varies depending on the version of the assembler. For details, refer to the Assembler User's Manual.

[*2] If the ID code written to the ID code area of the MCU matches the one entered for the [ID Code] box in the [ID Code verification] dialog displayed at emulator debugger startup, the E8a emulator writes FFh, FFh, FFh, FFh, FFh, FFh, FFh to the ID code area. Therefore, the [ID Code verification] dialog will not be displayed the next time the debugger starts up.

7.3 Power supply

(1) Consumption current

When the E8a emulator does not supply power to the user system, it consumes the power voltage of the user system from several mA to more than 10 mA. This is because the user power supply drives 74LVC125, 74LVC1T45 and 74LVC2T45 to make the communication signal level match the user system power supply voltage.

(2) E8a emulator power supply

When writing a program with the E8a emulator for mass production processes, the program requires reliability, so do not use the E8a emulator power supply function. Supply power separately to the user system according to the allowable voltage for MCU writing.

Voltage supplied from the E8a emulator depends on the quality of the USB power supply of the PC, and as such, precision is not guaranteed.

7.4 Operation during a user program halt

(1) Operation clock during a user program halt

When the user program halts, the emulator changes the CPU clock to the internal high-speed on-chip oscillator clock to operate. However, the peripheral features operate with the clock specified by the user program.

(2) Peripheral I/Os during a user program halt

During a user program halt, the maskable interrupt request cannot be accepted, because the emulator disables interrupts. However, since peripheral I/Os continue to run, the interrupt request is accepted immediately after the user program execution is started.

For example, a timer interrupt is not accepted although the timer continues to count when a user program is stopped by a break after the timer started.

(3) Notes on accessing the SFR areas

You can reference or set the contents of the SFR areas in the memory window or the IO window. Note the followings.

a) When accessing the special registers

You may not be able to access some special registers successfully during the user program halt. [*1]

These registers include:

- Access prohibited addresses
- Registers for which access order is specified from high-order byte to low-order byte
- Registers that can be accessed only by a specific instruction
- Registers whose bus width specification does not match the bus width set in the memory window
- Registers that can be accessed on conditions (one of which is that fOCO-F must be faster than the CPU clock to access the register)

Note:

[*1] Follow the instructions in the hardware manual of the target MCU to access to the SFR areas.

7.5 Debug functions

(1) PC break point

When downloading a user program after modifying it, the set address of PC break may not be corrected normally depending on the modification. Therefore, break points other than the set PC breaks may shift.
After downloading a user program, check the setting of PC breaks in the event point window and reset it.

(2) “Go to cursor” function

The “Go to cursor” function is actualized using an address match break. Therefore, when you execute the “Go to cursor” command, all the address match breaks and hardware breaks you set become invalid, while all the PC breaks remain valid.

(3) Debugging in stop mode

When debugging in stop mode or wait mode, do not operate windows until the program stops at the breakpoint by setting the breakpoint at the line of the program which will be executed after the stop mode or wait mode is cancelled.
In addition, disable the automatic update in the memory window or watch window or fix the display in the memory window before running the program, and do not make refresh operations during an execution so memory accesses do not occur during user program execution.

When the program is forcibly stopped or when the memory is referred to or modified in stop mode or wait mode, these modes will be cancelled, and the memory reference or modification may not be performed properly.

When debugging the program which enters stop mode immediately after it changes the high-speed on-chip oscillator frequency, a communication error may occur. In order to avoid this problem, either of the following measures must be taken.

a) Breaking the program immediately before entering stop mode

After the high-speed on-chip oscillator frequency is changed on the user program and the oscillation is stabilized, yet before the program enters stop mode, break the program. If you wish to use a data access break, set the break point so no instruction which shifts the program into stop mode should be executed.

b) Changing the time until when stop mode is entered

Allow approximately two or more seconds from when the high-speed on-chip oscillator frequency is changed on the user program until when stop mode is entered.

(4) Note on the CPU clock

Do not use the CPU clock at less than 15.6 kHz (low-speed OCO divided by 8).

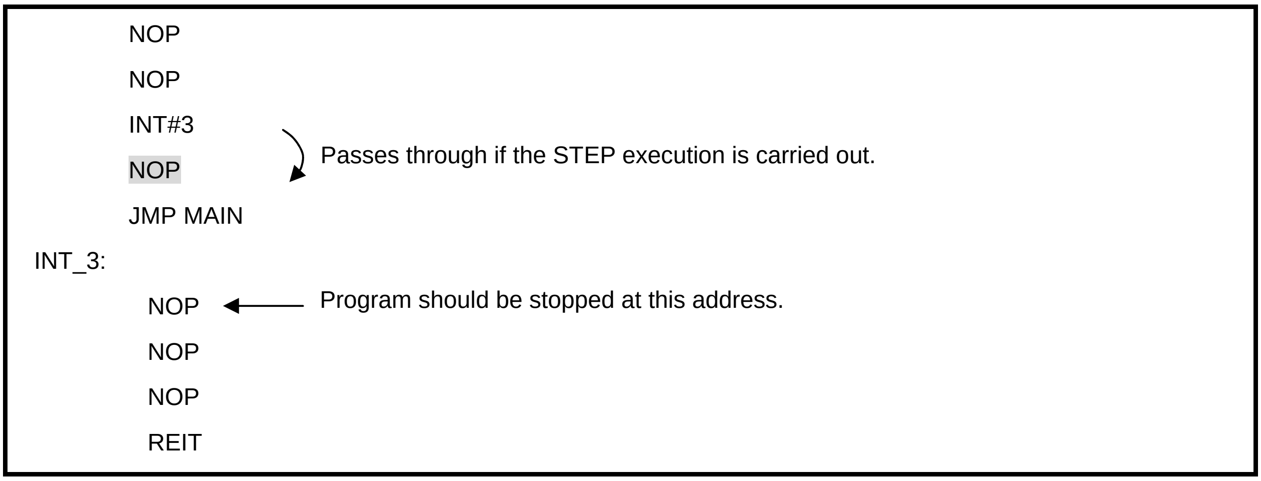
(5) Low-current-consumption read mode

When debugging in low-current-consumption read mode or the state that the flash memory is stopped, do not operate windows until the program stops at the breakpoint by setting the breakpoint at the line of the program which will be executed after each mode or state is cancelled.

(6) Exceptional step execution

a) Software interrupt instruction

Step execution cannot be performed in the internal processing of instructions (undefined, overflow, BRK and INT) which generate a software interrupt continuously in the program (see Figure 7.2).



```
          NOP
          NOP
          INT#3
          NOP
          JMP MAIN
INT_3:
           NOP
           NOP
           NOP
           REIT
```

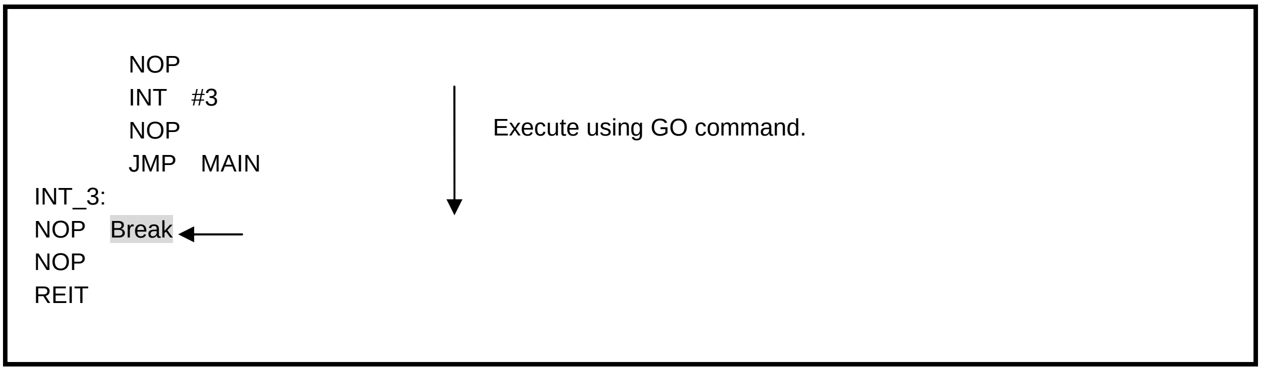
Passes through if the STEP execution is carried out.

Program should be stopped at this address.

Figure 7.2 Example of Software Interrupt Instruction

b) INT instruction

To debug the user program with the INT instruction, set a PC break for the internal processing of the INT instruction and execute the program with the GO command (see Figure 7.3).



```
          NOP
          INT   #3
          NOP
          JMP   MAIN
INT_3:
NOP   Break
NOP
REIT
```

Execute using GO command.

Figure 7.3 Example of INT Instruction

c) Other: Flag manipulating instructions

The following instructions, when single-stepped, only manipulate a flag in the E8a emulator, with no MCU operations involved. Therefore, when these instructions are executed, be aware that the Start/Stop function does not work.

LDC	src, FLG
STC	FLG, dest
LDINTB	src

(7) Note on using automatic memory update

When the automatic memory update is enabled in the memory or watch window, do not execute Step Out or Multiple-step. Otherwise, it will take longer to update memory data and the operation will be delayed.

(8) Note on internal power low consumption

Make sure that bit 0 of voltage detect register 2 (VCA2) for the E8a emulator is set to “0: Low consumption disabled”. If “1” is selected, the E8a emulator will not control the MCU.

E8a Emulator (R0E00008AKCE00)
Additional Document for User's Manual (Notes on Connection)

Publication Date: Dec 14, 2011 Rev.2.02

Published by: Renesas Electronics Corporation

Edited by: Microcomputer Tool Development Department 2
Renesas Solutions Corp.

RENESAS

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.
7F, 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

Colophon 1.0

E8a Emulator
Additional Document for User's Manual
(Notes on Connection)

Renesas Electronics Corporation