User's Manual

# E8a Emulator

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

Supported Devices:
M16C Family / M16C/50 Series
M16C/57 and M16C/5M

Renesas Electronics
www.renesas.com

Rev.2.02 Apr 2012

# 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 | M16C Family M16C/50 Series<br>M16C/57 and M16C/5M Groups |
| Available operating modes | Single-chip mode |

## 2.2 Emulator specifications

Table 2.2 shows the specifications of the emulator supported by the M16C 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 | M16C E8a Emulator Debugger V.1.04.00 or later | | |
| Operating Environment | Temperatures | Active<br>Inactive | : 10°C to 35°C<br>: –10°C to 50°C |
| | Humidity | Active<br>Inactive | : 35% RH to 80% RH, no condensation<br>: 35% RH to 80% RH, no condensation |
| | Vibrations | Active<br>Inactive<br>Transportation | : maximum 2.45 $m/s^2$<br>: maximum 4.9 $m/s^2$<br>: 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 | 3.0 - 5.5 V<br>For details, refer to the hardware manual of the MCU. | |
| Break functions | - Address match break, 8 points<br>- Data access break, 2 points<br>- Event A: Comparison with the address/data mask, and access condition (R, W, R/W) can be set.<br>- Event B: Comparison with the address mask, and access condition (R, W, R/W) can be set.<br>- PC break points (maximum 255 points)<br>- Forced break | |
| Trace functions | 32 branch instructions (branch source/destination PC)<br>or<br>Up to 64 data cycles can be specified | |
| Flash memory programming function | Available (when selecting the 'Program Flash' mode) | |
| User interface<br>(see Section 4.1.1 on page 11) | When communicating via CNVss pin:<br>When communicating via P6_4/P6_5/P6_6/P6_7: | 2-line clock-asynchronous serial<br>Clock-synchronous serial |
| MCU resources to be used | When communicating via CNVss pin<br>(see Section 4.1.1 on page 11) | - ROM size: 4 KB<br>- RAM size: 128 bytes<br>- Stack 14 bytes<br>- Address match interrupt<br>- UART1 function (external clock) and P6_5 [*1] |
| | When communicating via P6_4/P6_5/P6_6/P6_7<br>(see Section 4.1.1 on page 11) | - UART1 function and P6_4/P6_5/P6_6/P6_7 |
| Interface with host machine | USB (USB 1.1, full speed)<br>* Also connectable to host computers that support USB 2.0<br>* 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) [*2] | |

Notes:

[*1] Communication mode requiring UART1 CLK1 pin is not supported.

[*2] 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 for programming 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]<br>IAR EWM16C V.2.12 or later |

Note:

[*1] Notes on debugging the load modules created in ELF/DWARF2 format

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 4 “Examples of Pin Handling for Connecting the E8a”.
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) |

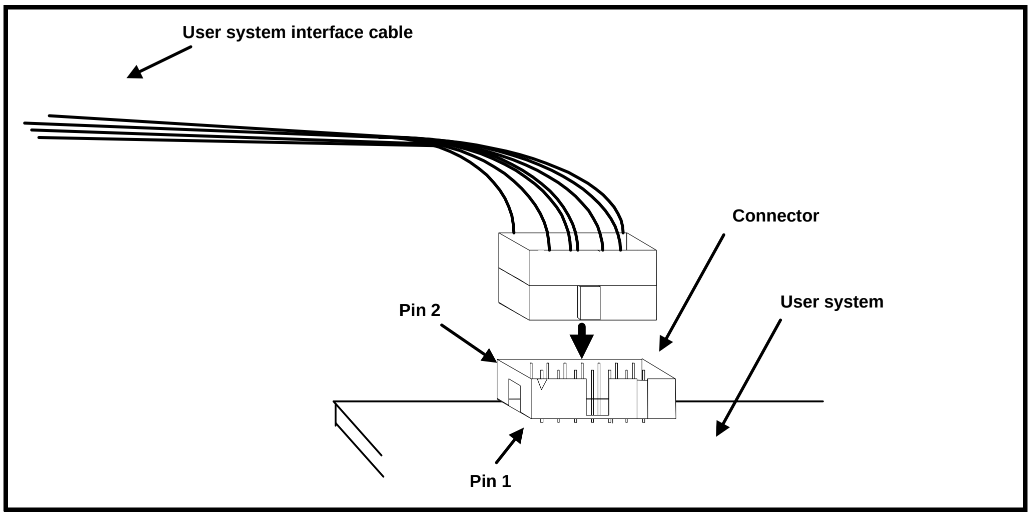
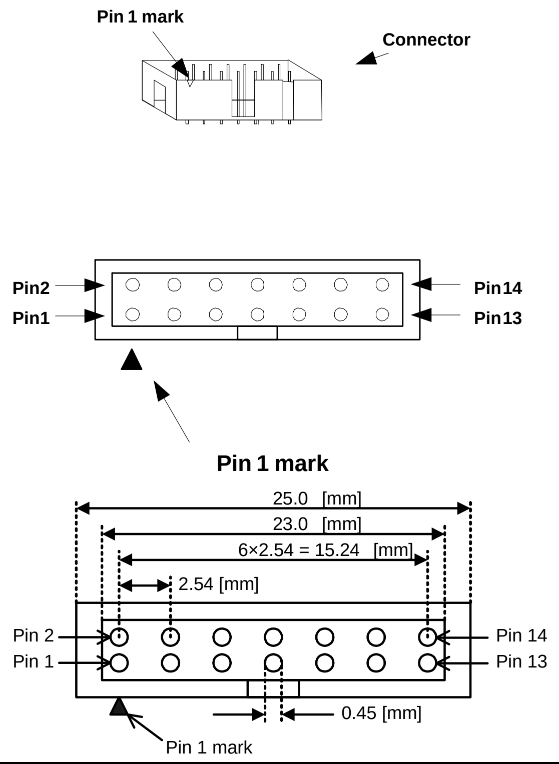


Figure 3.1 Connecting the User System Interface Cable with an E8a Connecting Connector

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, 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 | Communication via CNVss | Communication via P6_4/P6_5/P6_6/P6_7 |
|---|---|---|
| | MCU Signals | |
| 1 | P6_5(SCLK) | P6_5(SCLK) |
| 2 | Vss | Vss |
| 3 | CNVss | CNVss |
| 4 | N.C. | N.C. |
| 5 | N.C. | P6_7(TxD) |
| 6 | Vss | Vss |
| 7 | N.C. | N.C. |
| 8 | Vcc | Vcc |
| 9 | N.C. | P6_4(BUSY) |
| 10 | Vss | Vss |
| 11 | N.C. | P6_6(RxD) |
| 12 | Vss | Vss |
| 13 | RESET# | RESET# |
| 14 | Vss | 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, 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 Pin handling for connecting the E8a

### 4.1.1 Examples of pin handling for connecting the E8a (types of connection method)

Table 4.1 shows two types of connection method between the E8a and an MCU.

Table 4.1 Types of E8a connection method

| | Debugging by E8a Emulator Debugger | Flash memory programming by FDT [*1] |
|---|---|---|
| Communication via CNVss pin | Yes | Yes |
| Communication via P6_4/P6_5/P6_6/P6_7 | No | Yes |

### 4.1.2 Examples of pin handling for connecting the E8a (communication via CNVss pin)

The following shows examples of pin handling for connecting the E8a.

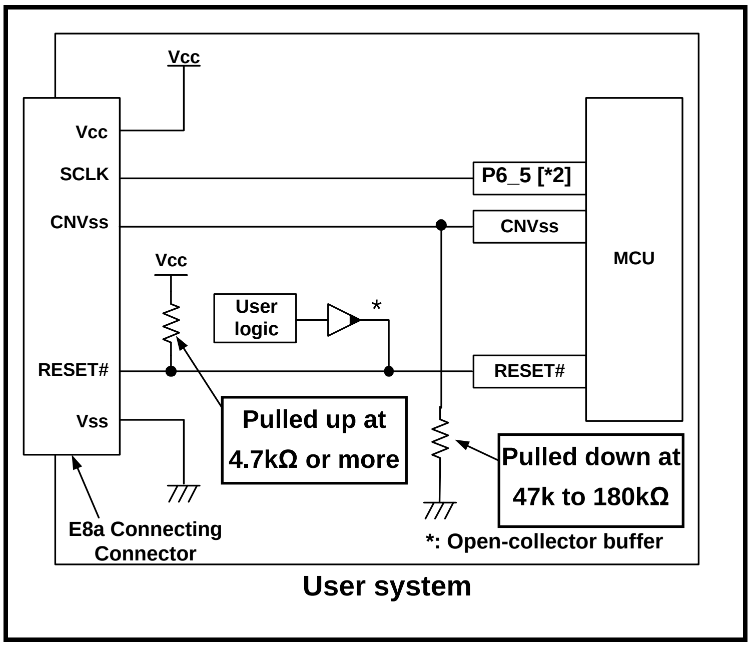


Figure 4.1 Example of an E8a Connection

Notes:

[*1] For the applicable version of the FDT, refer to the 'Flash Development Toolkit target device list' on the Flash Development Toolkit page of our website (http://www.renesas.com/tools).

[*2] Pin P6_5 (SCLK) is used exclusively by the E8a emulator.
Pull up the pins at the Vcc level or pull them down according to the MCU pin state after disconnecting the E8a emulator.

When adjacent resistors are used for pull-up, they may be affected by noise from other pins. In particular, separate the resistor for CNVss 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) CNVss pin

The E8a emulator uses the CNVss pin for MCU control.

Pull down the E8a emulator and MCU pins and connect the E8a emulator.

Do not connect a capacitor etc. to this pin.

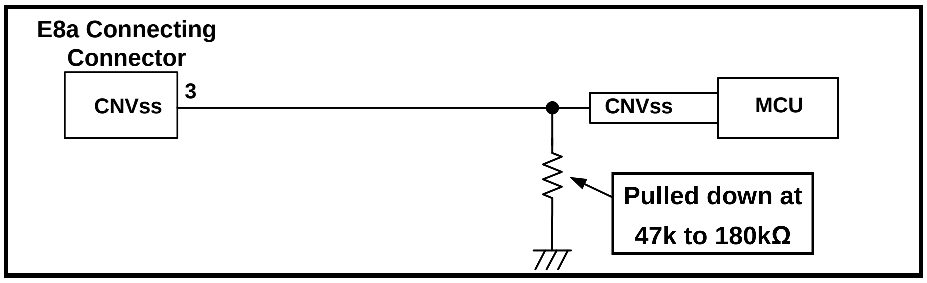


Figure 4.2 E8a Emulator and CNVss Pin Connection

(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.

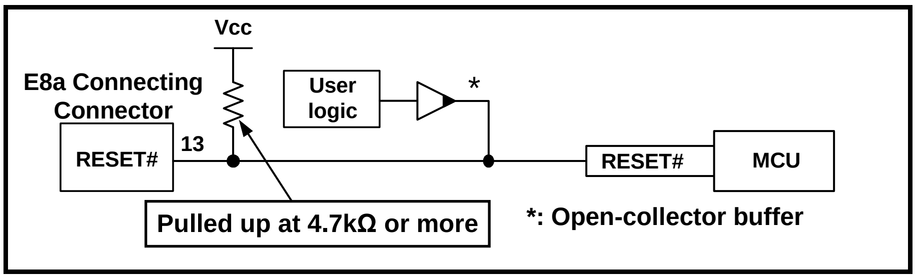


Figure 4.3 Example of a Reset Circuit

(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.
- If NMI# interrupts are used, make sure the NMI# pin is pulled up to the Vcc pin through a resistor.
- 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 that pins 2, 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.

### 4.1.3 Examples of pin handling for connecting the E8a (communication via P6_4/P6_5/P6_6/P6_7)

The following shows examples of pin handling for connecting the E8a.

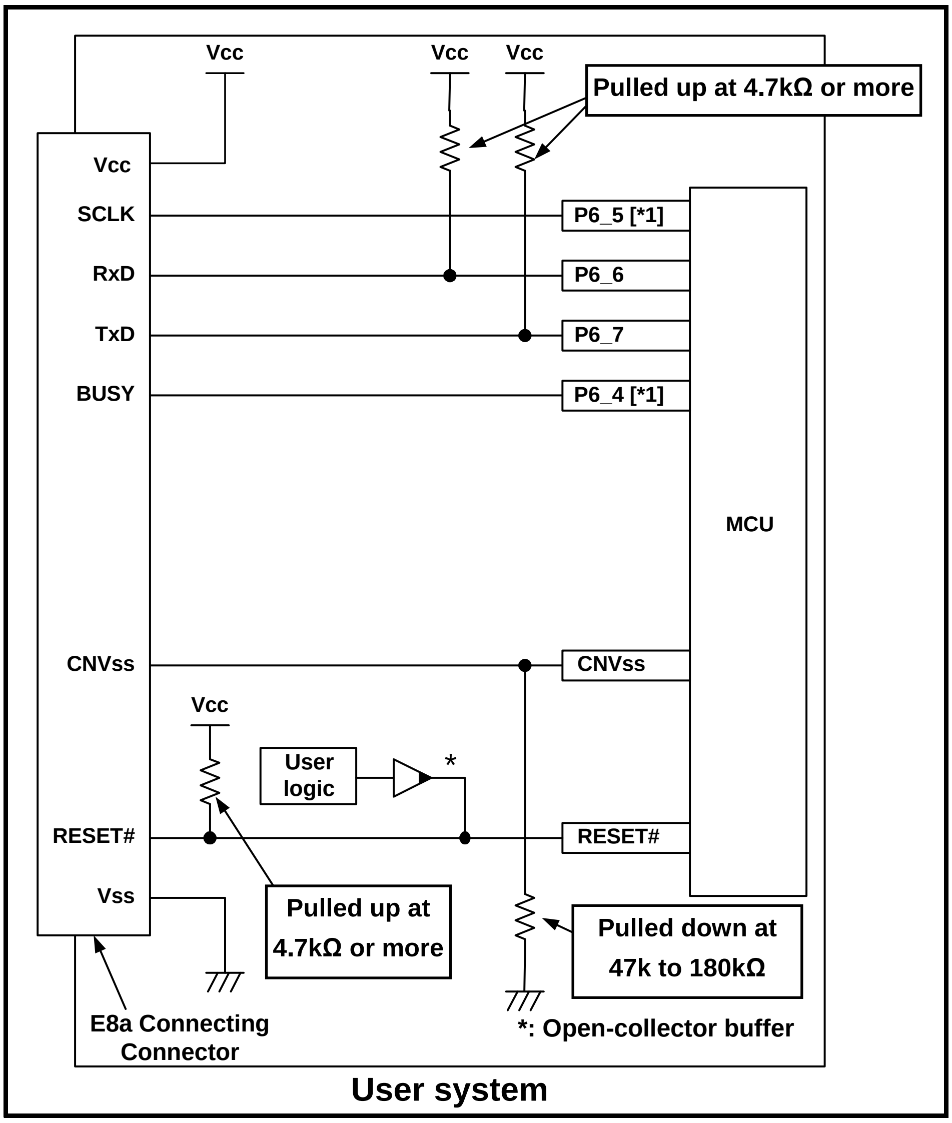


Figure 4.4 Example of an E8a Connection

Note:

[*1] For details on setting pins P6_4 and P6_5, refer to “(1) SCLK, RxD, TxD and BUSY pins” on page 14.

(1) SCLK, RxD, TxD and BUSY pins

Pins P6_4(BUSY), P6_5(SCLK), P6_6(RxD) and P6_7(TxD) are used exclusively by the E8a emulator.

Connect pins P6_6 and P6_7 to the E8a emulator after pulling up the MCU pins at the Vcc level.

For P6_4 and P6_5, pull up the pins at the Vcc level or pull down them according to the MCU pin state after disconnecting the E8a emulator.

P6_4 may be in a Hiz state while the E8a emulator is active. Therefore, set the pin resistance value so the voltage cannot be at the midpoint potential, depending on the voltage dividing of the resistance inside the E8a emulator (Figure 4.8 on page 16).

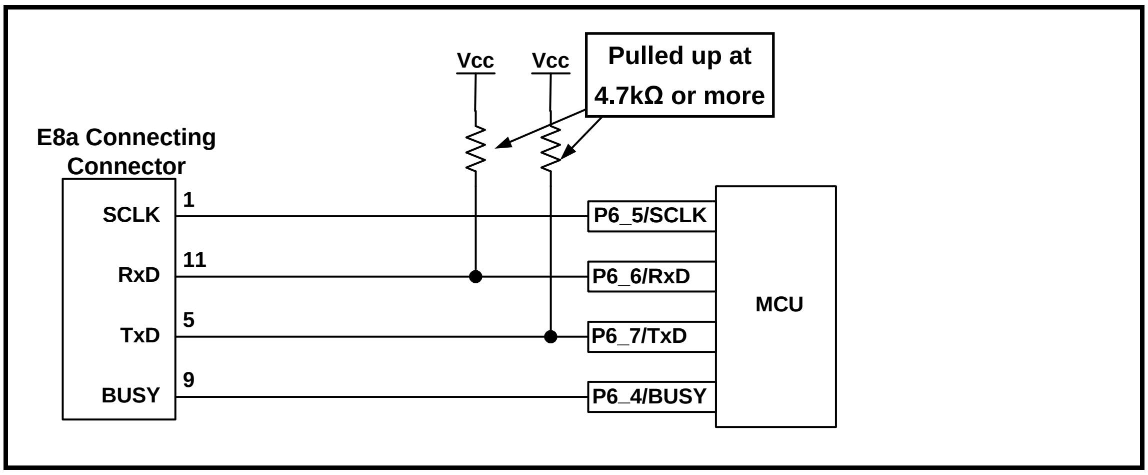


Figure 4.5 Connection of the SCLK, RxD, TxD and BUSY pins to the E8a Emulator

(2) CNVss pin

The E8a emulator uses the CNVss pin for MCU control.
Pull down the E8a emulator and MCU pins and connect the E8a emulator.
Do not connect a capacitor etc. to this pin.

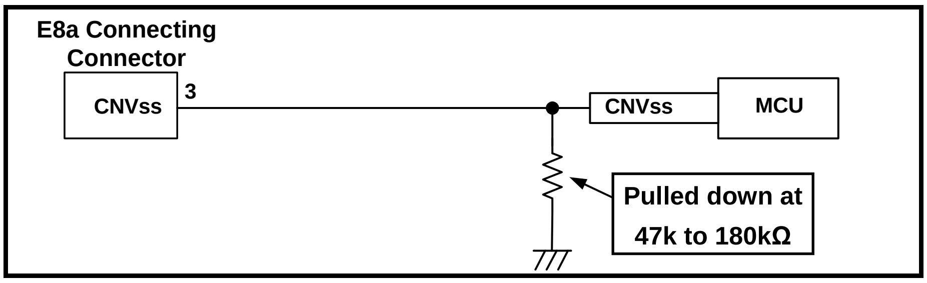


Figure 4.6 E8a Emulator and CNVss Pin Connection

(3) 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.

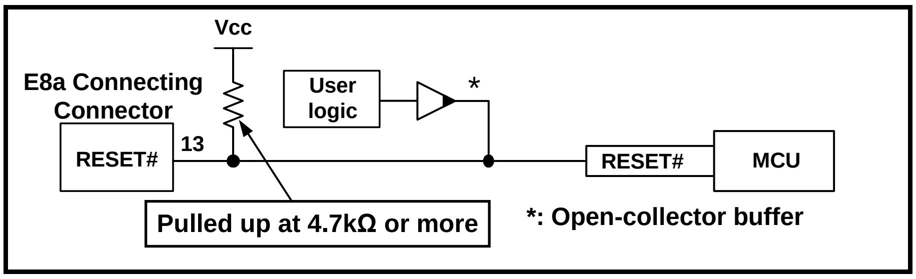


Figure 4.7 Example of a Reset Circuit

(4) 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.
- If NMI# interrupts are used, make sure the NMI# pin is pulled up to the Vcc pin through a resistor.
- 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 that pins 2, 6, 10, 12 and 14 are all connected to the Vss.
- Do not connect anything to the N.C. pin.

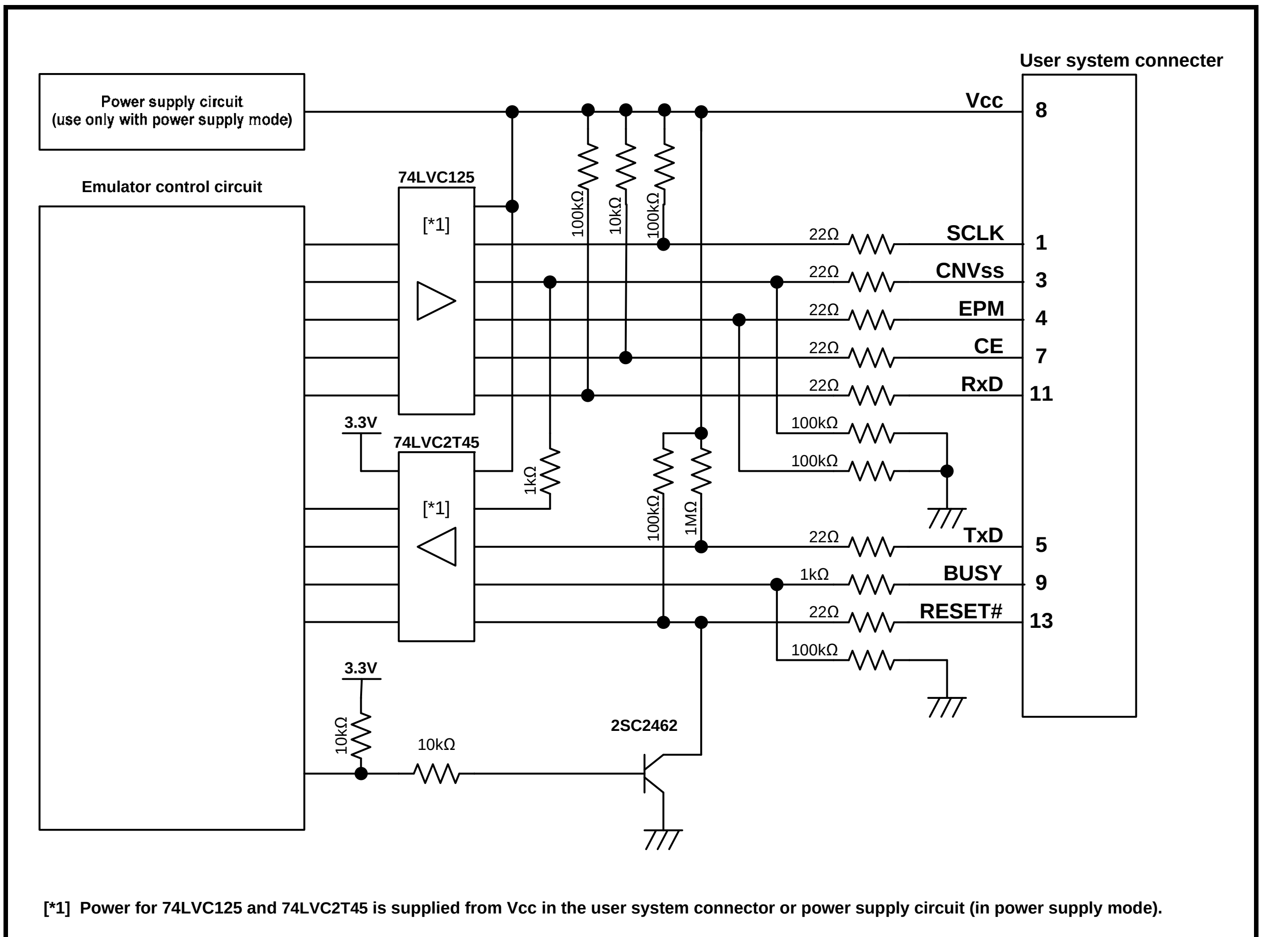
**⚠ WARNING**

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



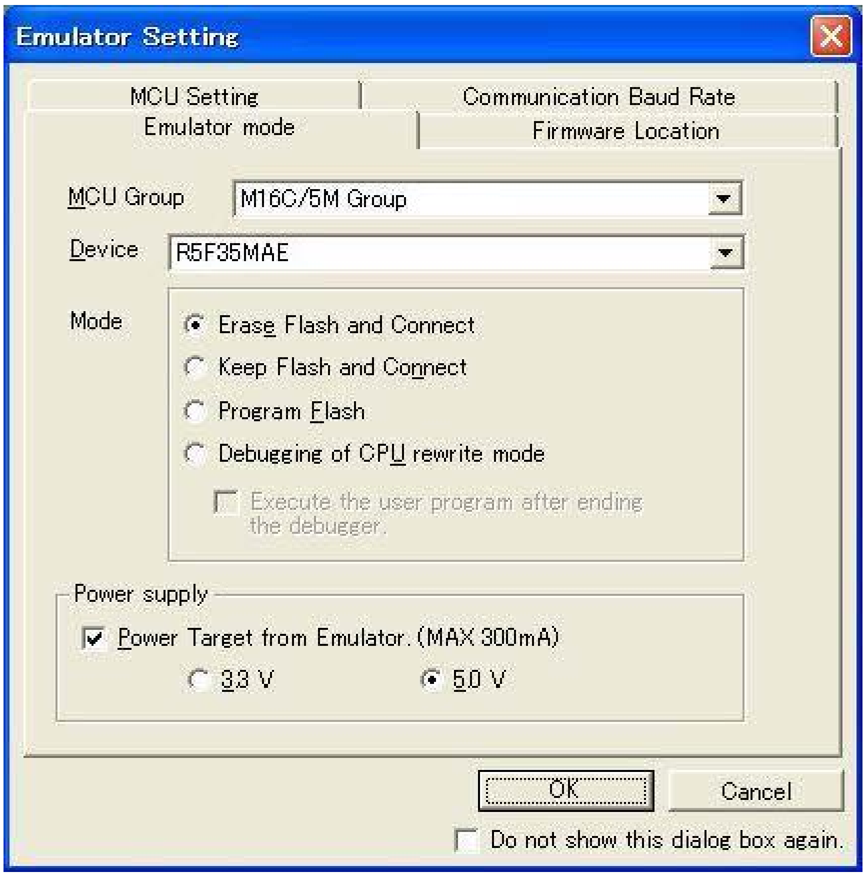
[*1] Power for 74LVC125 and 74LVC2T45 is supplied from Vcc in the user system connector or power supply circuit (in power supply mode).

Figure 4.8 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: M16C/5M Group

Device: R5F35MAE

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

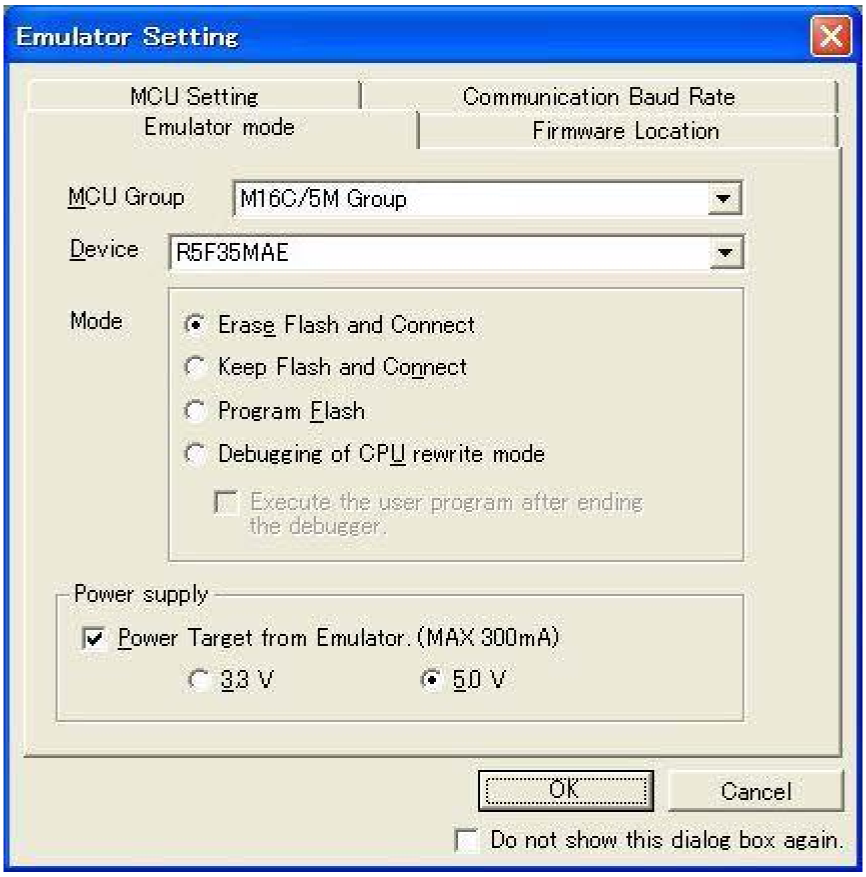
If you check “Do not show this dialog box again.” at the bottom of the [Emulator Setting] dialog box, the [Emulator Setting] dialog box will not be displayed the next time the debugger is launched.

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.

## 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

MCU Setting | Communication Baud Rate | Emulator mode | Firmware Location

MCU Group: M16C/5M Group

Device: R5F35MAE

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]

OK | Cancel

Do not show this dialog box again.

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

**[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 29 for the list of applicable MCUs.)

**[Mode]**
Select the mode to be used.
For details, see "5.2 (1) Selecting the Mode" (p.19).

**[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.

(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 OFS1 and ID code areas. |
| Keep Flash and Connect [*2] | | When launching the debugger, the E8a emulator retains the Flash memory data for the MCUs.<br><br>Note that the area for the E8a emulator program and the vector area used by the E8a emulator will change. The emulator rewrites the OFS1 and ID code areas. |
| Program Flash [*2] | Simple programmer [*3] | When downloaded, the E8a writes only the user program.<br>(E8a emulator program is not written.)<br>Therefore, the program cannot be debugged in this mode.<br><br>When [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 [*4] | Debugging only [*1] | Be sure to select this setting when debugging the program which rewrites the CPU.<br>In this mode, the following debug operation which rewrites the Flash memory cannot be executed.<br>- Setting the PC break points<br>- Changing the memory contents in the Flash memory area<br><br>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 OFS1 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.

The vector area, ID code area and the OFS1 area have their values rewritten by the emulator, so be aware that their checksums do not match.

[*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.

In this mode, the ID code settings made on the user program will be written to the internal flash memory of the MCU.

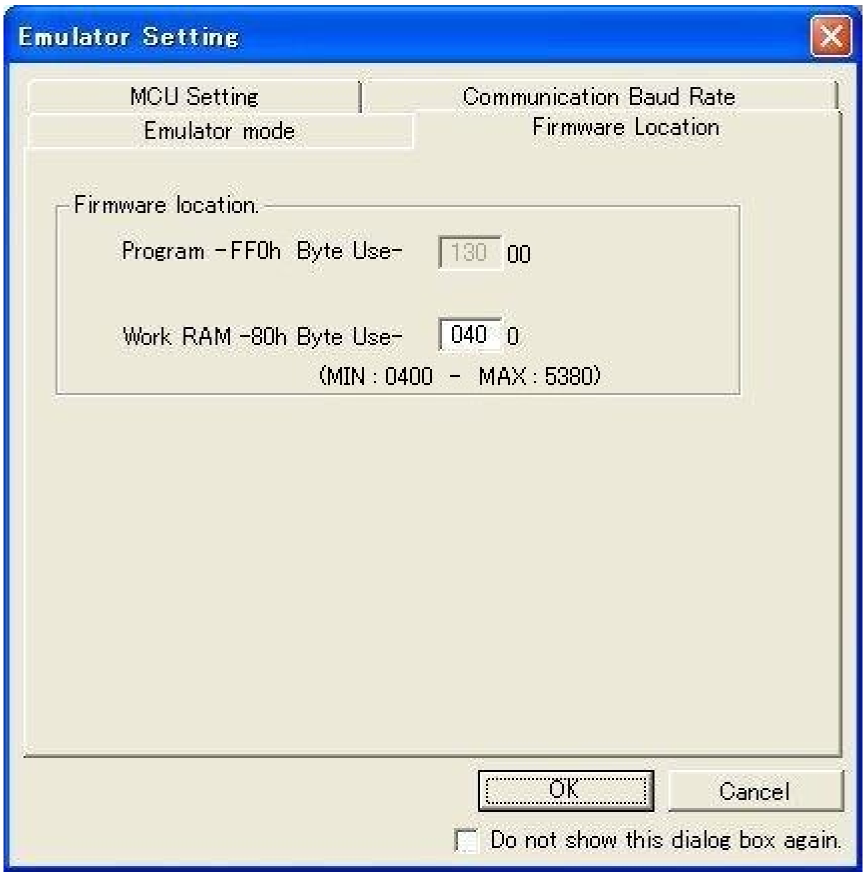
[*3] When downloading the user program in this mode, a checksum is displayed. The checksum covers the E$^2$dataFlash area.
However, when ECC is used, the ECC area is not covered by the checksum.
In addition, note that the E$^2$dataFlash area is erased when starting up in this mode.

[*4] 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. The MCU does not recognize the writing is continuously executed if the write instruction is interrupted by the memory reference or modification process.

- While write instruction is being executed to the register which requires continuous writing (ex. FMR01 bit)

## 5.3 [Firmware Location] tab

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


Emulator Setting

MCU Setting | Communication Baud Rate | Emulator mode | Firmware Location

Firmware location.

Program -FF0h Byte Use- 130 00

Work RAM -80h Byte Use- 040 0

(MIN : 0400 - MAX : 5380)

OK Cancel

Do not show this dialog box again.

**[Firmware Location]**

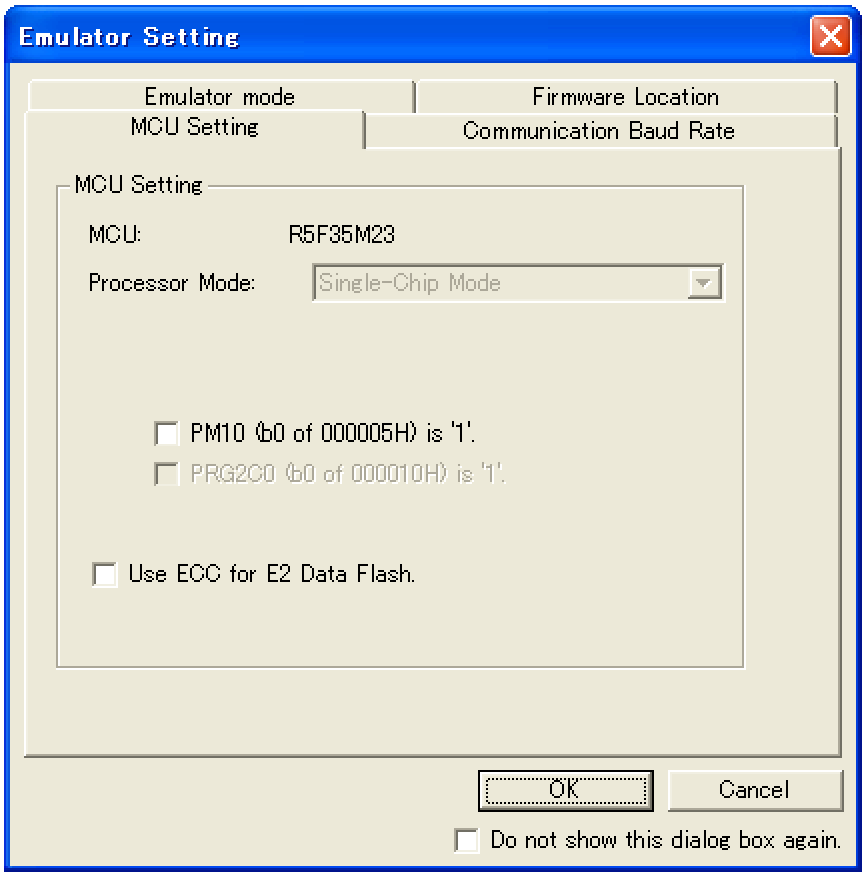
Select the area in which the firmware is located. Specify the address that will not be used by the user system in the ROM area or RAM area.

- Program
  Specify the ROM area in which the firmware is located. This setting is not required.
- Work RAM
  Specify the RAM area in which the firmware is located. Specify 128 bytes that will not be used by the user system.

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

## 5.4 [MCU Setting] tab

In the [MCU Setting] tab, set the operating condition of the MCU used in the user system.



MCU Setting

MCU: R5F35M23

Processor Mode: Single-Chip Mode

☐ PM10 (b0 of 000005H) is '1'.

☐ PRG2C0 (b0 of 000010H) is '1'.

☐ Use ECC for E2 Data Flash.

☐ Do not show this dialog box again.

Figure 5.4 [MCU Setting] Tab

**MCU**
The MCU selected in the [Device] drop-down list in the [Emulator mode] tab is displayed.

**Processor Mode**
Specify the processor mode according to the user system. Only the following mode can be specified for this product:
- Single-Chip Mode

**PM10 (b0 of 000005H) is '1'**
Specify whether PM10 (data flash enable bit) is set. When using the user program with PM10 set to "1", check this option. [*1]

**PRG2C0 (b0 of 000010H) is '1'**
Specify whether PRG2C0 (program 2 area control register) is set. PRG2C0 is fixed to "0".

**Use ECC for E2 Data Flash**
Select this checkbox if you want to use ECC for an MCU with E$^2$dataFlash. You cannot check this box if you selected an MCU without E$^2$dataFlash. Be sure to make the setting according to the setting of the E$^2$dataFlash of the program to debug. [*2]

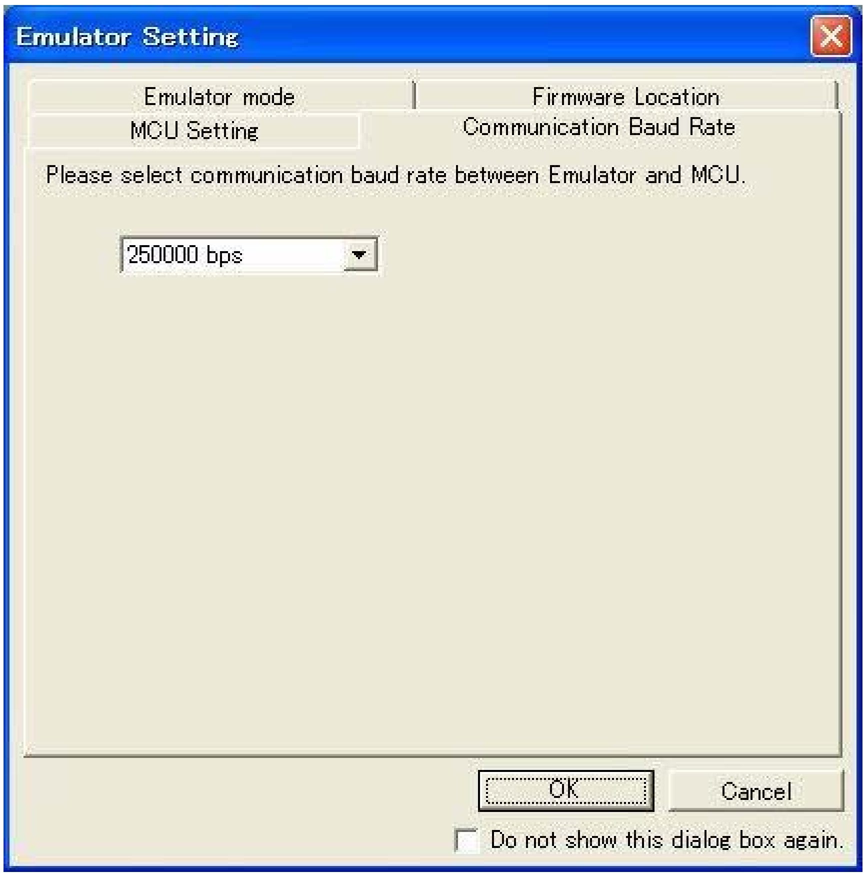
Notes:

[*1] For the case that the MCU in use does not have data flash, do not select this checkbox.

[*2] The block configuration of the E$^2$dataFlash is determined by the setting of this checkbox. If the setting is made erroneously, the E$^2$dataFlash area will be displayed incorrectly.

## 5.5 [Communication Baud Rate] tab

Select communication baud rate between the E8a and MCU in the [Communication Baud Rate] tab.
250000 bps (default setting) should be selected [*1] [*2].



Please select communication baud rate between Emulator and MCU.

250000 bps

Figure 5.5 [Communication Baud Rate] Tab

Notes:

[*1] Depending on the wired length of the CNVss 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, three break functions can be used: forced break, S/W break and on-chip break. The break functions can be set singly or multiply.
The list of break functions supported for the MCUs in this user's manual is shown in Table 6.1.

### 6.1.1 Forced break

The forced break function is used to forcibly cause a break in execution of the user program.

### 6.1.2 S/W break (software break)

This function breaks the program by rewriting the instruction of the specified address to an instruction (BRK instruction) dedicated to the debugger.
Since the op-code at the specified address is replaced by the instruction dedicated to the debugger, when a S/W breakpoint is set, a write to on-chip memory (flash memory and RAM) will occur. (Similarly, removing a S/W breakpoint involves a write to memory.)

### 6.1.3 On-chip break functions

With the MCUs in this user's manual, the following three on-chip break functions are available: Address match break, data access break and trace full break.

- 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.
  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.

Table 6.1 Break Functions

| Break type | | | Number of points that can be set (max.) | Break condition | Flash memory rewrite | Can a breakpoint be set while the program is running? |
|---|---|---|---|---|---|---|
| Forced break | | | | No | No | Yes |
| S/W break [*1] | | | 255 | Specified address | Yes | No |
| On-chip break | Address match break [*1] | | 8 | Specified address | No | Yes |
| | Data access break | EventA | 1 | Specified address & specified data | No | No |
| | | EventB | 1 | Specified address | No | No |
| | Trace full break | | | Trace buffer full | No | No |

Note:
[*1] When execution is restarted from the address where it stopped at a breakpoint (S/W breakpoints or address match breakpoints), the actual instruction at the address must be executed as a single step before further execution continues. Operation is thus not in real time.

### 6.1.4 On-chip trace functions

With the MCUs in this user's manual, either the branch trace or data trace function is available.

- Branch trace

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

- Data trace

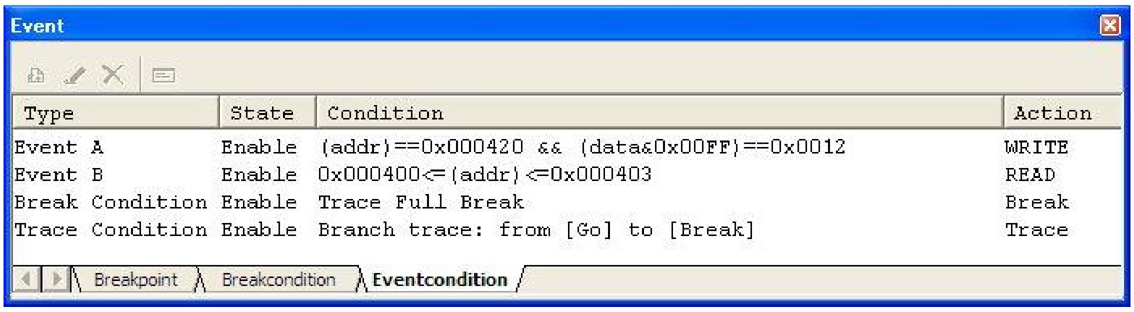
  This function displays data accesses when a data access event is encountered.

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 6.2.



```
Type            State   Condition                                       Action
Event A         Enable  (addr)==0x000420 && (data&0x00FF)==0x0012       WRITE
Event B         Enable  0x000400<=(addr)<=0x000403                      READ
Break Condition Enable  Trace Full Break                                Break
Trace Condition Enable  Branch trace: from [Go] to [Break]              Trace
```

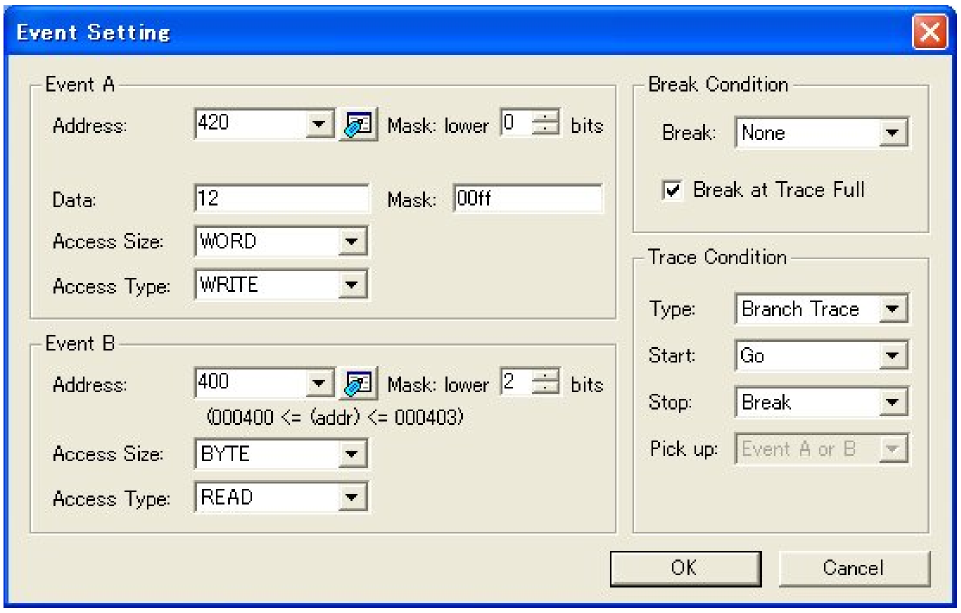
Figure 6.1 Eventpoints Window (Eventcondition tab)

Table 6.2 Display Contents of the Eventcondition Tab

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

Event A
Address: 420 Mask: lower 0 bits
Data: 12 Mask: 00ff
Access Size: WORD
Access Type: WRITE

Event B
Address: 400 Mask: lower 2 bits
(000400 <= (addr) <= 000403)
Access Size: BYTE
Access Type: READ

Break Condition
Break: None
[✓] Break at Trace Full

Trace Condition
Type: Branch Trace
Start: Go
Stop: Break
Pick up: Event A or B

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 6.3 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.<br>- R/W: READ or WRITE<br>- READ<br>- WRITE |

(2) Event B

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

Table 6.4 Contents of the Event B

| Option | Description |
|---|---|
| Address (with mask specification) | Same as the Event A. |
| Access Size | Same as the Event A. |
| Access Type | Same as the Event A. |

(3) Break Condition

Set the break condition.

Table 6.5 Break Condition

| Option | Description |
|---|---|
| Break | Select a break condition.<br>- None: None specified. (No break by event)<br>- Event A: Breaks the program when the Event A is encountered.<br>- Event A or B: Breaks the program when either the Event A or Event B is encountered.<br>- Event A and B: Breaks the program when both the Event A and Event B are encountered.<br>- Event B->A: Breaks the program when an event is encountered in the order of the Event B and Event A. |
| 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. |

(4) Trace Condition

Set the trace condition.

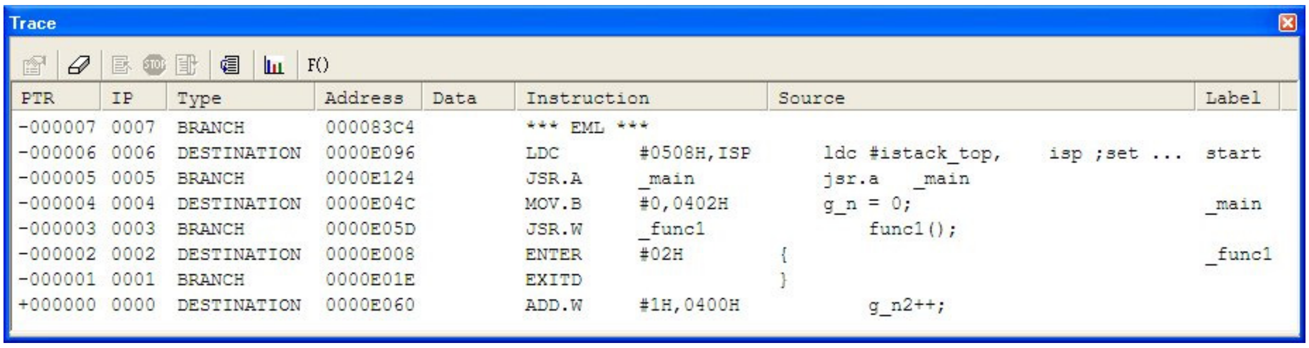
Table 6.6 Trace Condition

| Option | Description |
|---|---|
| Type | Select a trace type.<br>- Branch Trace<br>- Data Trace |
| Start | Select a start condition for the trace measurement.<br>- Go: Starts a measurement when starting executing the target program.<br>- Event A: Starts a measurement when the Event A is encountered.<br>- Event A or B: Starts a measurement when either the Event A or Event B is encountered.<br>- Event A and B: Starts a measurement when both the Event A and Event B are encountered.<br>- Event B->A: Starts a measurement when an event is encountered in the order of the Event B and Event A. |
| Stop | Select a stop condition for the trace measurement.<br>- Break: Stops a measurement when stopping executing the target program.<br>- Trace FULL: Stops a measurement when the trace data is filled.<br>- Event A: Stops a measurement when the Event A is encountered.<br>- Event A or B: Stops a measurement when either the Event A or Event B is encountered.<br>- Event A and B: Stops a measurement when both the Event A and Event B are encountered.<br>- Event B->A: Stops a measurement when an event is encountered in the order of the Event B and Event A. |
| Pick up | Select an event to record when tracing data.<br>- Event A: Records only data access which encounters the condition of the Event A.<br>- Event A or B: Records only data access which encounters the condition of either the Event A or Event B. |

## 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 6.7.



```
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 6.7 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.<br>“*** 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 6.8 below. [*1] [*2] [*3] [*4]

Table 6.8 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<br>Access size: BYTE<br>Access type: READ or R/W |
| Byte write to even address | Available | Address: 400h<br>Access size: BYTE<br>Access type: WRITE or R/W |
| Word read to even address | Available | Address: 400h<br>Access size: WORD<br>Access type: READ or R/W |
| Word write to even address | Available | Address: 400h<br>Access size: WORD<br>Access type: WRITE or R/W |
| Byte read to odd address | Available | Address: 401h<br>Access size: BYTE<br>Access type: READ or R/W |
| Byte write to odd address | Available | Address: 401h<br>Access size: BYTE<br>Access type: WRITE or R/W |
| Word read to odd address | Available | Address: 401h<br>Access size: BYTE [*5]<br>Access type: READ or R/W |
| Word write to odd address | Available | Address: 401h<br>Access size: BYTE [*5]<br>Access type: WRITE or R/W |

Notes:

[*1] 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.

[*2] 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.

[*3] 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.

[*4] 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.

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

# 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 and Table 7.2 list the program area 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 (1)

| Group | Part No. | ROM Size | RAM Size | Program Area for E8a Emulator | | |
|---|---|---|---|---|---|---|
| | | Program ROM | | Vector Area | ROM Area | RAM Area |
| M16C/5M | R5F35M23 | 96 KB | 8 KB | FFFE4h - FFFE7h, FFFE8h - FFFEBh, FFFECh - FFFEFh, FFFF4h - FFFF7h, FFFFCh - FFFFFh | 13000h - 13FEFh [*1] | 128 bytes [*2] |
| | R5F35M33 | 96 KB | 8 KB | | | |
| | R5F35M73 | 96 KB | 8 KB | | | |
| | R5F35M83 | 96 KB | 8 KB | | | |
| | R5F35MB3 | 96 KB | 8 KB | | | |
| | R5F35MC3 | 96 KB | 8 KB | | | |
| | R5F35ME3 | 96 KB | 8 KB | | | |
| | R5F35MF3 | 96 KB | 8 KB | | | |
| | R5F35M16 | 128 KB | 12 KB | | | |
| | R5F35M26 | 128 KB | 12 KB | | | |
| | R5F35M36 | 128 KB | 12 KB | | | |
| | R5F35M66 | 128 KB | 12 KB | | | |
| | R5F35M76 | 128 KB | 12 KB | | | |
| | R5F35M86 | 128 KB | 12 KB | | | |
| | R5F35MA6 | 128 KB | 12 KB | | | |
| | R5F35MB6 | 128 KB | 12 KB | | | |
| | R5F35MC6 | 128 KB | 12 KB | | | |
| | R5F35MD6 | 128 KB | 12 KB | | | |
| | R5F35ME6 | 128 KB | 12 KB | | | |
| | R5F35MF6 | 128 KB | 12 KB | | | |
| | R5F35M1E | 256 KB | 20 KB | | | |
| | R5F35M2E | 256 KB | 20 KB | | | |
| | R5F35M3E | 256 KB | 20 KB | | | |
| | R5F35M6E | 256 KB | 20 KB | | | |
| | R5F35M7E | 256 KB | 20 KB | | | |
| | R5F35M8E | 256 KB | 20 KB | | | |
| | R5F35MAE | 256 KB | 20 KB | | | |
| | R5F35MBE | 256 KB | 20 KB | | | |
| | R5F35MCE | 256 KB | 20 KB | | | |
| | R5F35MDE | 256 KB | 20 KB | | | |
| | R5F35MEE | 256 KB | 20 KB | | | |
| | R5F35MFE | 256 KB | 20 KB | | | |

Notes:

[*1] - The portion of the program ROM2 area (13000h - 13FEFh) is used by the E8a emulator program.
- This area overlaps with the user boot area. Therefore, when using any other mode than “Program Flash” mode (see 5.2 (1) Selecting the Mode on page 19) at the startup of the emulator debugger, note that this area (13000h - 13FEFh) is overwritten by the E8a emulator program.

[*2] - When starting the debugger, the [Emulator Setting] dialog box is displayed. Specify the area which will not be used by the user system. For details, see 5.3 [Firmware Location] tab on page 20.

Table 7.2 Program Area for the E8a Emulator (2)

| Group | Part No. | ROM Size | RAM Size | Program Area for E8a Emulator | | |
|---|---|---|---|---|---|---|
| | | Program ROM | | Vector Area | ROM Area | RAM Area |
| M16C/57 | R5F35723 | 96 KB | 8 KB | FFFE4h - FFFE7h, FFFE8h - FFFEBh, FFFECh - FFFEFh, FFFF4h - FFFF7h, FFFFCh - FFFFFh | 13000h - 13FEFh [*1] | 128 bytes [*2] |
| | R5F35733 | 96 KB | 8 KB | | | |
| | R5F35773 | 96 KB | 8 KB | | | |
| | R5F35783 | 96 KB | 8 KB | | | |
| | R5F35716 | 128 KB | 12 KB | | | |
| | R5F35726 | 128 KB | 12 KB | | | |
| | R5F35736 | 128 KB | 12 KB | | | |
| | R5F35766 | 128 KB | 12 KB | | | |
| | R5F35776 | 128 KB | 12 KB | | | |
| | R5F35786 | 128 KB | 12 KB | | | |
| | R5F3571E | 256 KB | 20 KB | | | |
| | R5F3572E | 256 KB | 20 KB | | | |
| | R5F3573E | 256 KB | 20 KB | | | |
| | R5F3576E | 256 KB | 20 KB | | | |
| | R5F3577E | 256 KB | 20 KB | | | |
| | R5F3578E | 256 KB | 20 KB | | | |

Notes:

[*1] - The portion of the program ROM2 area (13000h - 13FEFh) is used by the E8a emulator program.
- This area overlaps with the user boot area. Therefore, when using any other mode than “Program Flash” mode (see 5.2 (1) Selecting the Mode on page 19) at the startup of the emulator debugger, note that this area (13000h - 13FEFh) is overwritten by the E8a emulator program.

[*2] - When starting the debugger, the [Emulator Setting] dialog box is displayed. Specify the area which will not be used by the user system. For details, see 5.3 [Firmware Location] tab on page 20.

(2) Pins used by the E8a emulator

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

- When communicating via CNVss pin: P6_5 pin [*1], RESET# and CNVss pins
- When communicating via P6_4/P6_5/P6_6/P6_7: RESET#, CNVss, P6_4, P6_5, P6_6 and P6_7 pins

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

The BRK instruction interrupt, address match interrupt, single-step interrupt and DBC 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.

(4) Interrupts used by the E8a emulator program (NMI)

If NMI interrupts are used, be sure to take the necessary precautions before executing the user program like disabling the automatic update in the watch window or fix the display in the memory window before running the program so that memory accesses do not occur during an execution. If an NMI interrupt occurs while the user program halts or when memory contents are referenced or modified during user program execution, the E8a emulator cannot control the MCU.

(5) Stack area used by the E8a emulator

The E8a emulator uses up to 14 bytes of the stack pointer (ISP) during a user program break. Therefore, set aside 14 bytes for the stack area.

(6) Count source protection mode

Count source protection mode cannot be debugged with the E8a emulator.

(7) User boot function

When debugging with the E8a emulator, the user boot function cannot be used.
If the user boot function is set to be enabled in the user boot code area, the E8a emulator cannot connect with the MCU.
In this case, rewrite the area to disable the user boot function with a parallel programmer.

Note:

[*1] The communication mode that needs the CLK1 pin of UART1 cannot be used.
The external clock cannot be selected as a transfer clock for UART1. Be sure to set the internal clock for it.

(8) 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 7.3 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 |
|---|---|---|---|
| 0005h | Processor mode register 1 | PM1 | Bit 0 |
| 000Ah | Protect register | PRCR | Bit 0 |
| 0012h | Peripheral clock select register | PCLKR | Bit 2, 6, 7 |
| 018Ch | DMA0 control register | DM0CON | Bit 3 [*1] |
| 019Ch | DMA1 control register | DM1CON | Bit 3 [*1] |
| 01ACh | DMA2 control register | DM2CON | Bit 3 [*1] |
| 01BCh | DMA3 control register | DM3CON | Bit 3 [*1] |
| 004Bh | DMA0 interrupt control register | DM0IC | Bit 3 [*1] [*2] |
| 004Ch | DMA1 interrupt control register | DM1IC | Bit 3 [*1] [*2] |
| 0069h | DMA2 interrupt control register | DM2IC | Bit 3 [*1] [*2] |
| 006Ah | DMA3 interrupt control register | DM3IC | Bit 3 [*1] [*2] |

Notes:

[*1] DMAC during a user program halt
When the user program is halted or when the memory is referred to or modified during user program execution, DMA transfer is disabled. In such cases, the E8a emulator sets the registers below as following.
Therefore, if you refer to the registers below in the memory window, etc., it shows that DMA is disabled.

| | |
|---|---|
| - DMA0 control register (DM0CON) DMA enable bit (bit 3) | 0: DMA disabled |
| - DMA1 control register (DM1CON) DMA enable bit (bit 3) | 0: DMA disabled |
| - DMA2 control register (DM2CON) DMA enable bit (bit 3) | 0: DMA disabled |
| - DMA3 control register (DM3CON) DMA enable bit (bit 3) | 0: DMA disabled |
| - Interrupt control registers Interrupt request bit (bit 3) | 0: Interrupt not requested |

Do not enable DMA transfer from the memory window, etc., but enable it in the user program.

[*2] When restarting the user program, though the E8a emulator sets back the value of a DMA enable bit to the previous value that was set before the program stops, the interrupt request bit remains 0.

(9) SFRs exclusively used by the E8a emulator program

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

| Address | Register | Symbol | Bit | Notes on Using the E8a Emulator |
|---|---|---|---|---|
| 0010h | Program 2 area control register | PRG2C | Bit 0 | [*1] |
| 0022h | 40 MHz on-chip oscillator control register 0 | FRA0 | Bit 0 | [*2] |
| 020Eh | Address match interrupt enable register | AIER | All bits | [*3] |
| 020Fh | Address match interrupt enable register 2 | AIER2 | All bits | |
| 0210h - 0212h | Address match interrupt register 0 | RMAD0 | All bits | |
| 0214h - 0216h | Address match interrupt register 1 | RMAD1 | All bits | |
| 0218h - 021Ah | Address match interrupt register 2 | RMAD2 | All bits | |
| 021Ch - 021Eh | Address match interrupt register 3 | RMAD3 | All bits | |
| 0220h | Flash memory control register 0 | FMR0 | Bit 5 | [*1] |
| 0258h | UART1 transmit/receive mode register | U1MR | Bit 3 | |
| 03EEh | Port P6 direction register | PD6 | Bit 5 | |

(10) Pin Assignment Control Register (PACR)

When the system is launched, the E8a emulator initializes the Pin Assignment Control Register (PACR:370h) as shown below. Only set the bit values specified for the MCU to be debugged, otherwise the E8a cannot control the MCU.

| bit (Bit name) | The number of pins of MCUs to debug | Initial value set by E8a | Notes |
|---|---|---|---|
| b2, b1, b0 (pin enable bit) | 100 pins | 100 | [*1] |
| | 80 pins | 000 | [*4] |
| | 64 pins | 000 | [*4] |

Notes:

[*1] When operating this register, make changes using the bit operation instructions to avoid changing the bit values.

[*2] 40 MHz on-chip oscillator
When debugging with the E8a emulator, the 40 MHz on-chip oscillator does not stop although the options for 40 MHz on-chip oscillator start bit are available and FRA00 can be set to “40 MHz on-chip oscillator off”.
For this reason, functions to reduce power consumption, with the 40 MHz on-chip oscillator off, need to be checked using your final products or system for which only the user program is written to the MCU, with the E8a emulator disconnected.
The functions can also be checked by writing only the user program to the MCU in the ‘Program Flash’ mode, ending the debugger, then executing the user program. In the [Emulator Setting] dialog box displayed when starting the debugger, select [Program Flash], then check [Execute the user program after ending the debugger].

[*3] Do not change this register value.

[*4] Set the bit values specified for the MCU to be debugged in the user program as follows.

| | b2 | b1 | b0 |
|---|---|---|---|
| When debugging 64-pin version MCUs: | 0 | 1 | 0 |
| When debugging 80-pin version MCUs: | 0 | 1 | 1 |

(11) 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 7.5.

Table 7.5 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 | Work RAM Address for the E8a emulator + 80h [*1] |
| | SB | 0000h |
| | FLG | 0000h |

(12) 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.

Note:

[*1] The Work RAM address for the E8a emulator is specified in the [Firmware Location] tab of the [Emulator Setting] dialog box.

(13) Debugging during a watchdog timer operation [*1] [*2] [*3]

When running the E8a emulator program (when the user program is stopped), the E8a emulator program refreshes the watchdog timer. If memory access is executed through memory reference or modification, the watchdog timer will be refreshed by the E8a emulator program. Note that this timing will differ from the actual operational timing.

When starting the M16C E8a Emulator Debugger, set each of bit 0 and bit 7 of the optional function select address 1 (OFS1: FFFFFh) to 1b. Although these addresses can be rewritten and the changed values can be referred to in the memory window, etc., the changed values for these bits (bit 0 and bit 7) are invalid.

| | |
|---|---|
| - b0: Watchdog timer start select bit | 1: Watchdog timer is in a stopped state after reset. |
| - b7: After-reset count source protection mode select bit | 1: Count source protection mode disabled after reset |

For the optional function select address 2 (OFS2: FFFDBh), the values are always as follows in the E8a emulator and even if the values are changed, they are invalid.

| | |
|---|---|
| - b1, b0: Watchdog timer initial setting bit | 11: 3FFFh |
| - b3, b2: Watchdog timer refresh duty cycle setting bit | 11: 100% |

Notes:

[*1] If an underflow or other abnormal condition occurs immediately after the user program breaks, causing the watchdog timer to be reset immediately before it is refreshed, the emulator may become uncontrollable.

Note that if the user program uses the watchdog timer, the watchdog timer will be refreshed by the E8a emulator program during the user program halt, making the refresh timing differ from the actual operational timing.
Also, note that the watchdog timer is not refreshed during the execution of the user program.

[*2] Count source protection mode cannot be debugged with the E8a emulator.

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

## 7.2 Reset

(1) Reset function

Do not perform reset from other than the emulator debugger, otherwise the E8a emulator will run out of control.
Also note that if a reset occurs while the automatic update is enabled in the memory or watch window, the E8a emulator may run out of control.
Do not stop the user program while the reset pin remains in the “L” state. A timeout error will occur.

(2) Reset vector address

During a debug with the E8a emulator, the reset vector addresses are used by the E8a emulator program. If the MCU is reset while the user program is being executed, control is transferred to the E8a emulator program and the user program is forced to stop.

## 7.3 Notes on using the CAN module

(1) When using the CAN module [*1] and if BCLK (CPU clock) is used as the CAN clock source (fCAN)
The CPU clock should be used at 4MHz or more. If the CPU clock is used at less than 4MHz in this case, a communication error may occur.

(2) When using the CPU clock at more than 10MHz and if the program activates the CAN module
The following debugging operations, which rewrite the flash memory, cannot be performed. This is because of the limitation on the operation speed in flash rewrite mode (for details, refer to the hardware manual or data sheet of the MCU).
- Setting PC break points
- Changing memory contents in the flash memory area
If these debugging operations are performed, the rewrite operation on the flash memory may result in error.

(3) Do not activate the CAN module from the memory window, etc., except from the user program.

(4) When using the CAN module [*1], do not shift into stop mode.
Otherwise, a communication error may occur.

Note:
[*1] The CAN module is recognized as being used in the following status (other than in CAN sleep mode)
- Bit 2 (SLPST: CAN sleep status flag) of the CAN0 status register C0STR (D7C2h)
  0: Not in CAN sleep mode
- Bit 2 (SLPST: CAN sleep status flag) of the CAN1 status register C1STR (D4C2h)
  0: Not in CAN sleep mode

## 7.4 Flash memory

### 7.4.1 Notes on debugging in CPU rewrite mode

(1) Unrewritable area in CPU rewrite mode

When debugging in CPU rewrite mode, do not perform CPU rewrite operations to the following areas. If these areas are rewritten, the E8a emulator will not control the MCU.

- Block 0 area (addresses F0000h - FFFFFh) and block containing the E8a emulator program

(2) Operation in CPU rewrite mode

- When debugging in the CPU rewrite mode, select “Debugging of CPU rewrite mode” in the [Mode] section of the [Emulator mode] tab.
- 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, otherwise the E8a emulator may not control the MCU. And do not perform a step execution of the instruction which enables a CPU rewrite mode or enters an erase-suspend state.
  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.
- 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”.

### 7.4.2 Note 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

### 7.4.3 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. (In the same way, modification of the internal ROM area during a user program halt is made to the internal cache of the E8a emulator, and actual access to the flash memory is executed before the user program restarts.)

### 7.4.4 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.

### 7.4.5 Flash memory ID code

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

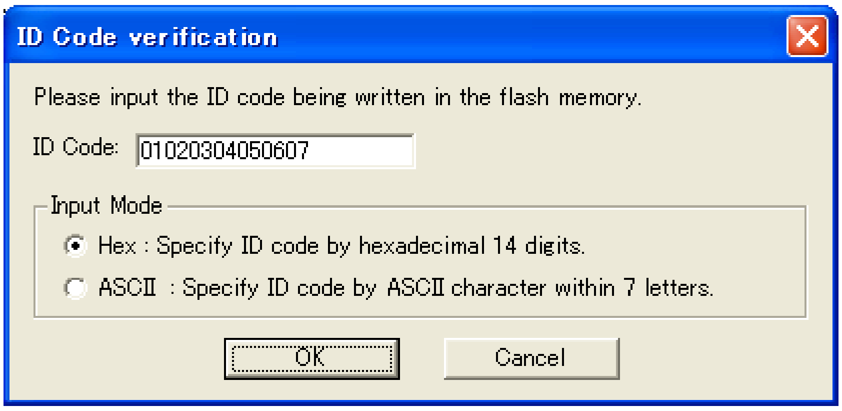
The ID code in Table 7.6 written to the flash memory of the MCU must match the ID code displayed in the Figure 7.1 [ID Code verification] Dialog Box at debugger startup, otherwise the debugger cannot be launched. Note that when the ID code is FFh, FFh, FFh, FFh, FFh, FFh, FFh, the ID code is regarded 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]: (regardless of the contents of the downloaded user program) | FFh, FFh, FFh, FFh, FFh, FFh, FFh |

Table 7.6 ID Code Storage Area

| Address | Description |
|---|---|
| FFFDFh | First byte of ID code |
| FFFE3h | Second byte of ID code |
| FFFEBh | Third byte of ID code |
| FFFEFh | Fourth byte of ID code |
| FFFF3h | Fifth byte of ID code |
| FFFF7h | Sixth byte of ID code |
| FFFFBh | Seventh byte of ID code |


ID Code verification

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.

## 7.5 Functions of the $E^2$dataFlash

(1) Program download

Program download is possible as well as user program download.

(2) Memory access to the $E^2$dataFlash area (during user program halt)

Memory access to the $E^2$dataFlash area is possible as well as the ROM and RAM.

When the memory is accessed from the memory window, read/write is performed according to the mode selected with the [Use ECC for E2 Data Flash] checkbox in the [MCU Setting] tab (see 5.4 [MCU Setting] tab) in the init dialog box which appears at debugger startup. If the setting of this checkbox differs from that of the ECC Control Bit (ECC enabled /ECC disabled) in the user program, memory reference/modification cannot be performed correctly.

(3) Memory access to the $E^2$dataFlash area (during user program execution)

Memory access to the $E^2$dataFlash is not possible. If the $E^2$dataFlash area is displayed in the memory or other windows, “1” will be displayed for all the bits of the displayed area. Do not stop the user program when it is processing the access to the $E^2$dataFlash area. If the user program stops, the operation to the $E^2$dataFlash is suspended, and the access to the $E^2$dataFlash may not be processed properly even after the user program is restarted. Disable the automatic update in the windows before running the user program so access to the $E^2$dataFlash does not occur during an execution.

(4) Other

PC break points cannot be set in the $E^2$dataFlash area.

## 7.6 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) Note on 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.

## 7.7 Operation during a user program halt

(1) Peripheral I/Os during a 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.

(2) Operation clock during a user program halt (for MCUs that have CAN modules)

When the user program halts, the operation clock differs depending on the status of the CAN module (status of the CAN sleep status flag).

a) In CAN sleep mode (SLPST = 1)

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 (the peripheral clock coming from the CPU clock originates from the internal high-speed on-chip oscillator clock).

The internal high-speed on-chip oscillator clock is changed as follows to meet the frequency specification for rewriting the flash memory of the MCU.

- When firmware is operating: Approx. 10 MHz
- When writing flash memory: Approx. 8 MHz or approx. 4 MHz

b) Not in CAN sleep mode (SLPST = 0)

The clock specified by the user program is used as the operation clock. [*1]

(3) Operation clock during a user program halt (for MCUs that do not have CAN modules)

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 (the peripheral clock coming from the CPU clock originates from the internal high-speed on-chip oscillator clock).

The internal high-speed on-chip oscillator clock is changed as described in a) in (2) to meet the frequency specification for rewriting the flash memory of the MCU.

(4) 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. [*2]

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)

Notes:

[*1] When using the CAN module for communication, the clock specified by the user program is used as the operation clock. Therefore, the following should be considered.

- PC break point

If a low-speed clock such as the sub clock is used as the operation clock of the MCU, setting or canceling PC breaks may take time. Use event breaks as the first choice.

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

## 7.8 Final evaluation of the program

Before entering the mass-production phase, be sure to perform a final evaluation of the program singly, without the E8a emulator connected.

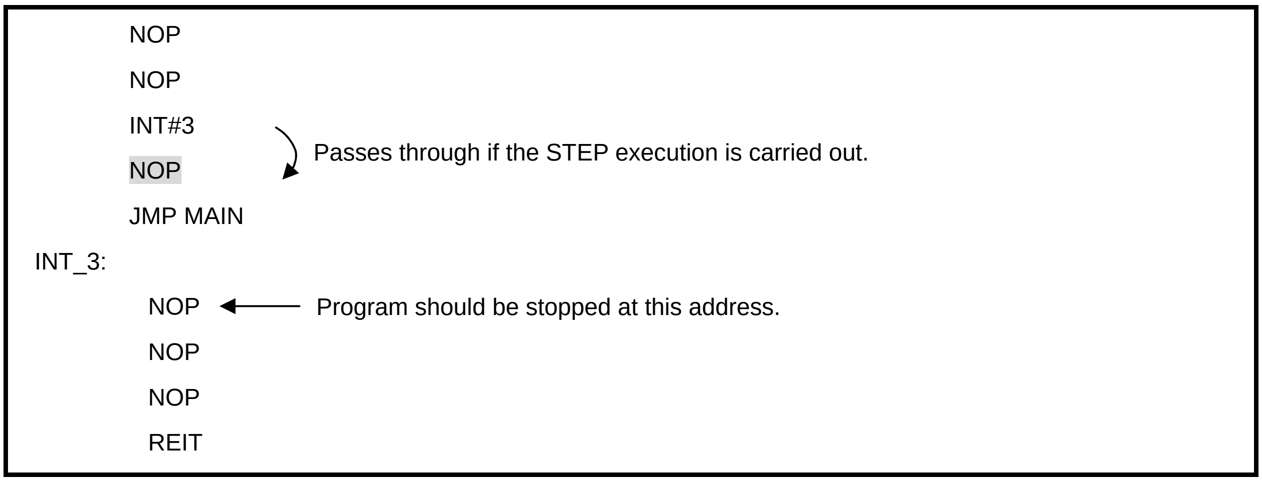
## 7.9 Debug functions

### 7.9.1 Step execution

(1) 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).



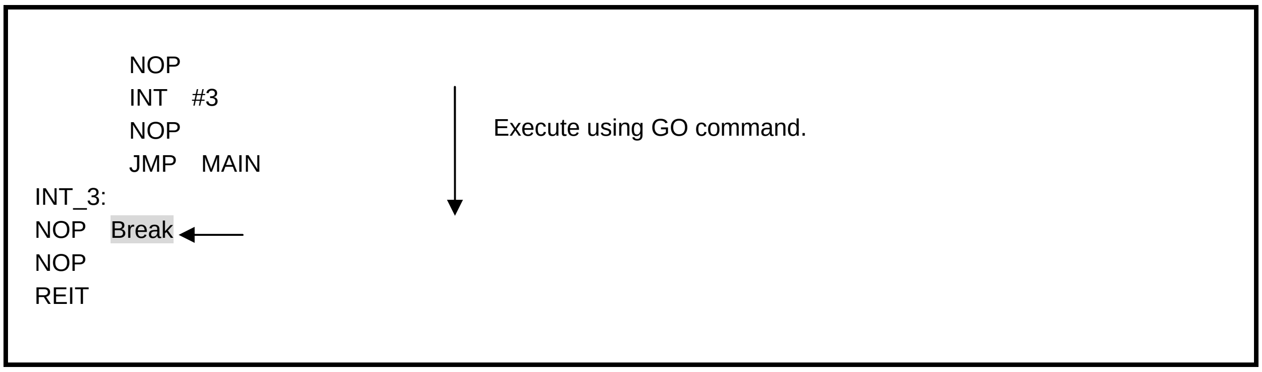
```
          NOP
          NOP
          INT#3
          NOP
          JMP MAIN
  INT_3:
            NOP
            NOP
            NOP
            REIT
```

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
```

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
```

### 7.9.2 Other debug functions

(1) “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 you set become invalid, while all the PC breaks remain valid.

(2) Debugging in stop mode or wait 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.

Also, if memory contents are referenced or changed during the stop mode or wait mode, the program—after exiting the stop or wait mode and accessing the MCU memory—restarts from the instruction next to the one by which it was placed into the stop or wait mode. If the program enters the stop or wait mode in the middle of a memory access, a reference or a change of memory contents may not be performed normally.

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.

(3) Note on 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.

(4) Note on the CPU clock

Do not use the CPU clock at less than 32.768 kHz (sub-clock).

(5) Low power consumption mode

When debugging in low-current consumption read mode, slow 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) 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.

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

Publication Date: Apr 26, 2012 Rev.2.02

Published by: Renesas Electronics Corporation

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

**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

R20UT2013EJ0202
(Previous Number:
REJ10J1968-0201)