REJ10J1930-0100

Everywhere you imagine. RENESAS

E8a Emulator

Additional Document for User's Manual

R0E00008AKCE00EP57

Renesas Microcomputer Development Environment System
R8C Family / R8C/3x Series
Notes on Connecting the R8C/35A, R8C/36A and R8C/38A

User's Manual

Rev.1.00
Jun. 01, 2009

Renesas Technology
www.renesas.com

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

2. E8a Emulator Specifications

2.1 Emulator specifications

Table 2.1 shows the E8a emulator specifications for the R8C/35A, R8C/36A and R8C/38A Groups. Table 2.2 shows the operating environment of the E8a emulator.

Table 2.1 E8a Emulator Specifications for the R8C/35A, R8C/36A and R8C/38A Groups

Target MCUs	R8C Family R8C/3x Series R8C/35A, R8C/36A and R8C/38A Groups
Available operating modes	Single-chip mode
Power voltages	3.0 - 5.5 V (f(XIN)=20MHz) 2.7 - 5.5 V (f(XIN)=10MHz) 1.8 - 5.5 V (f(XIN)=5MHz) [*1]
Debug functions	
Break functions	- Address match break, 8 points - Data access break, 2 points - Event A: Comparison with the address/data mask, and access condition (R, W, R/W) can be set. - Event B: Comparison with the address mask, and access condition (R, W, R/W) can be set. - PC break points (maximum 255 points) - Forced break
Trace functions	4 branch instructions (branch source/destination PC) or Up to 8 data cycles can be specified.
Flash memory programming function	Available
User interface	1-line clock asynchronous serial interface (communication via MODE pin)
MCU resources to be used	- ROM size: 2 KB - Stack 8 bytes - Address match interrupt
Emulator power supply	Unnecessary (USB bus powered, power supplied from the PC)
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)
Applicable emulator debugger	R8C E8a Emulator Debugger V.1.03.00 or later

Note

[*1] Set the power voltage to 2.7 V or above for rewriting the flash memory.
For details, refer to “7.6 (4) Note on debugging at less than 2.7V” on page 32.

Table 2.2 Operating Environment

Temperatures	Active	: 10°C to 35°C
	Inactive	: –10°C to 50°C
Humidity	Active	: 35% RH to 80% RH, no condensation
	Inactive	: 35% RH to 80% RH, no condensation
Vibrations	Active	: maximum 2.45 m/s^2
	Inactive	: maximum 4.9 m/s^2
	Transportation	: maximum 14.7 m/s^2
Ambient gases	No corrosive gases	

2.2 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 2.3 below.

Table 2.3 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 IAR EWM16C V.2.12 or later

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).

```
///////////////////////////////////////////////////////////
  *.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

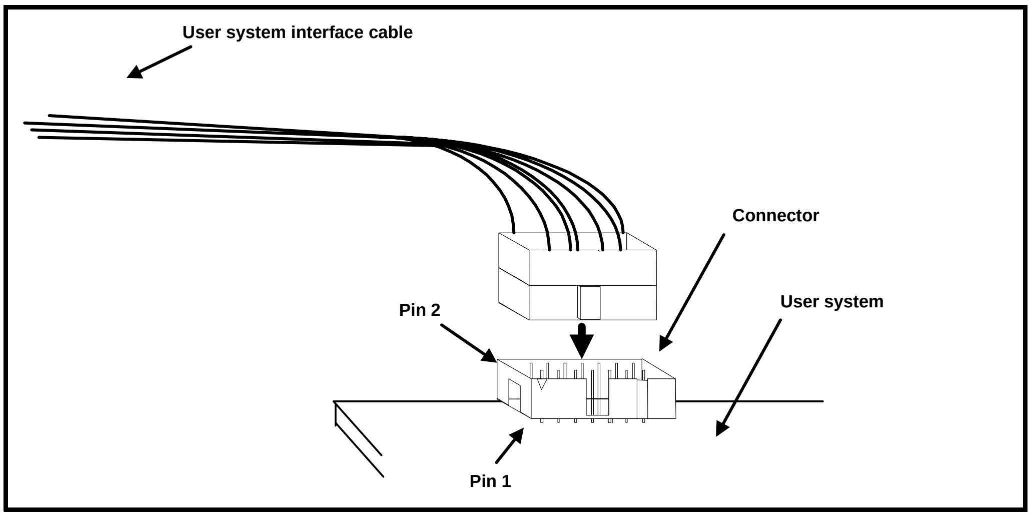
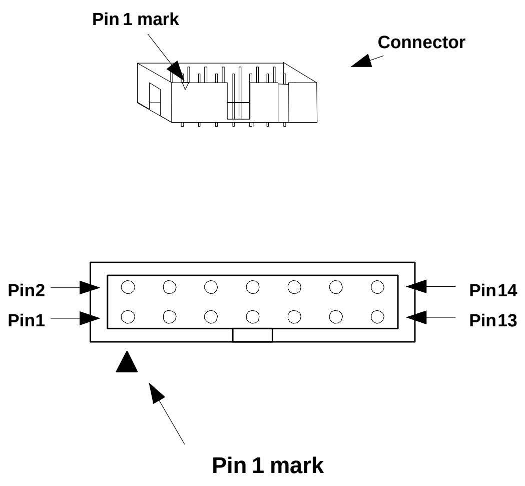


Figure 3.1 Connecting the User System Interface Cable with an E8a Connecting Connector

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.



Pin NO	R8C/35A, R8C/36A and R8C/38A 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.

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.

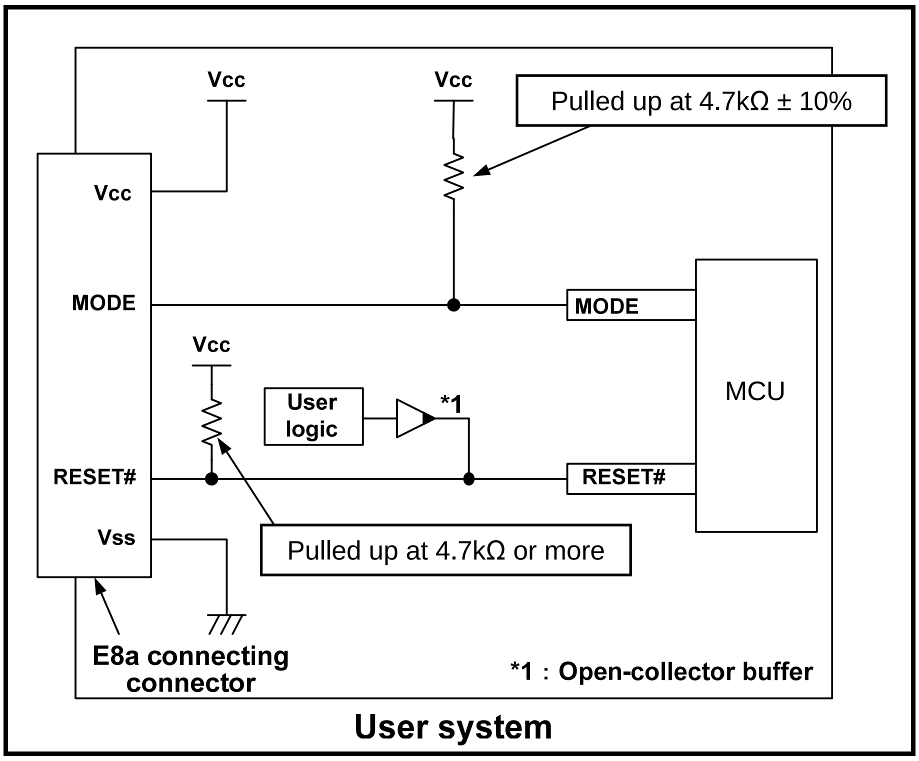


Figure 4.1 Example of an E8a Connection

(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.

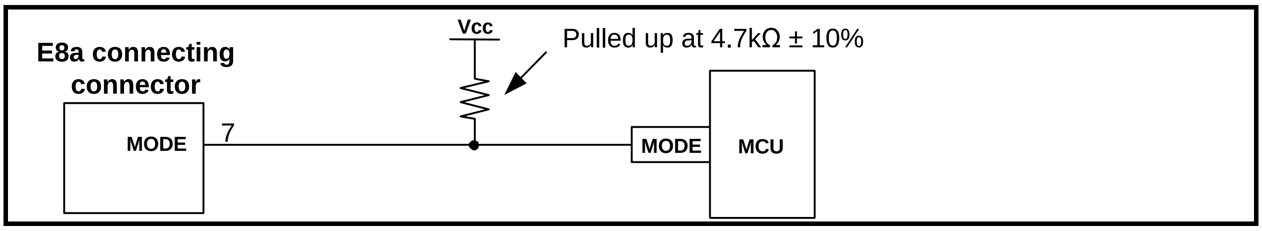


Figure 4.2 E8a Emulator and MODE Pin Connection

(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.

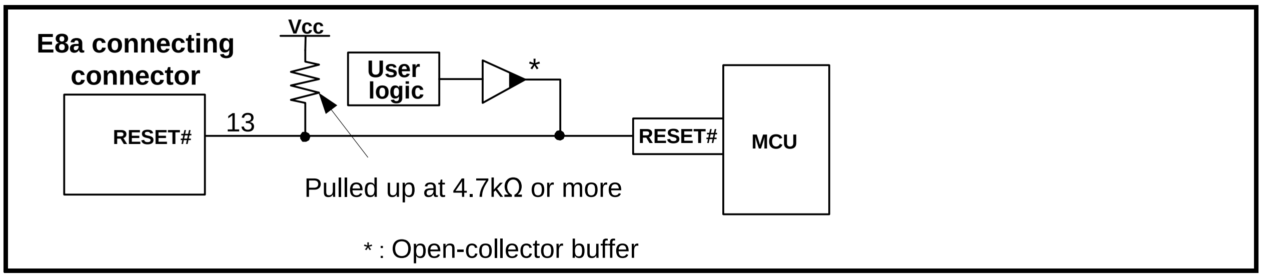


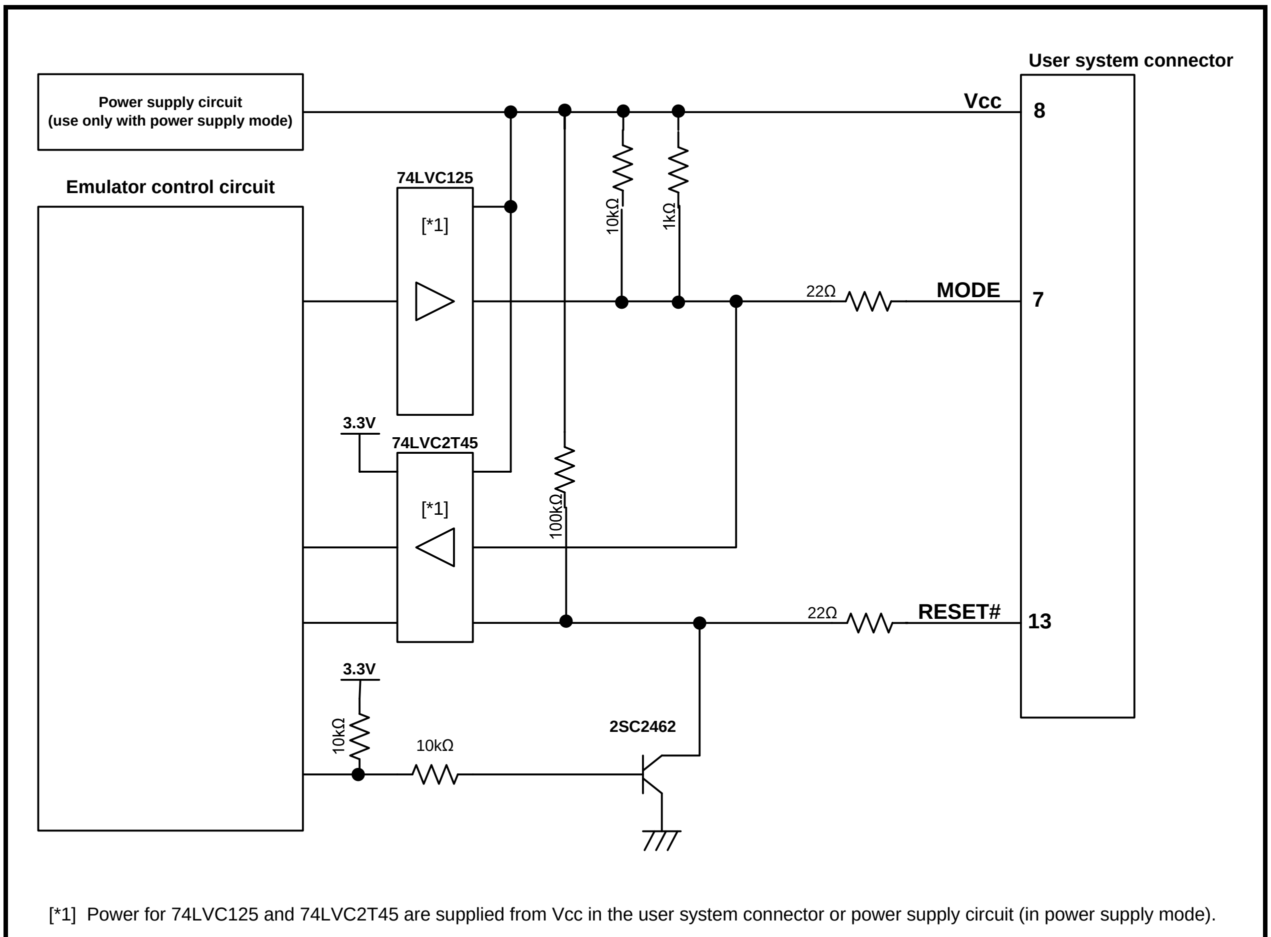
Figure 4.3 Example of a Reset Circuit

(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.

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.



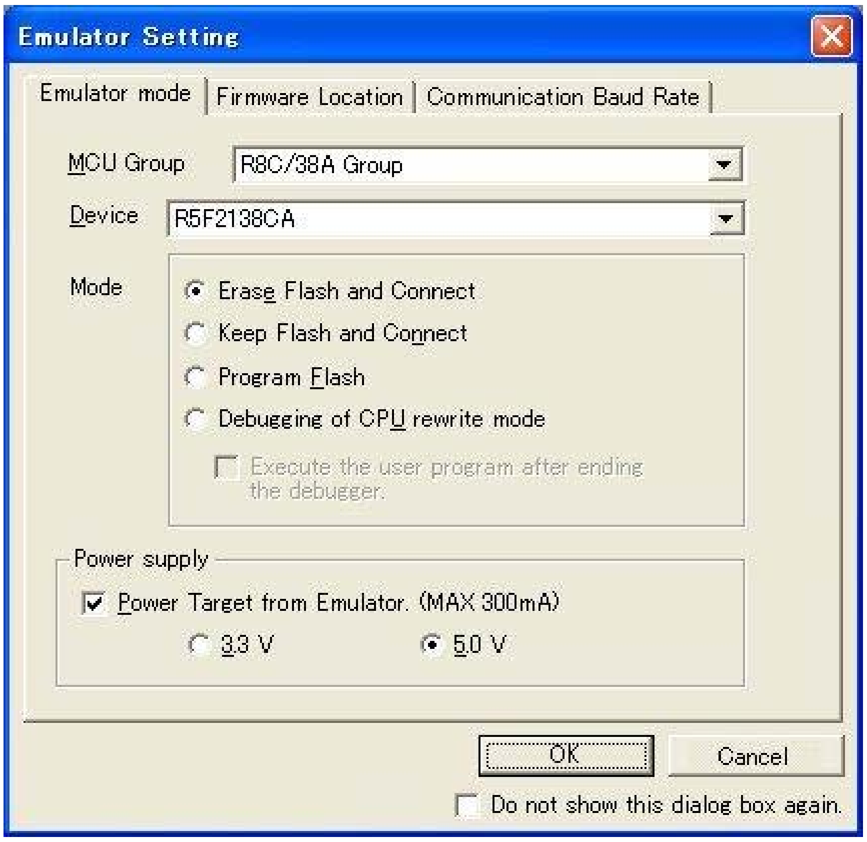
[*1] Power for 74LVC125 and 74LVC2T45 are supplied from Vcc in the user system connector or power supply circuit (in power supply mode).

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.


Emulator Setting

Emulator mode | Firmware Location | Communication Baud Rate

MCU Group: R8C/38A Group

Device: R5F2138CA

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 5.1 [Emulator Setting] Dialog Box

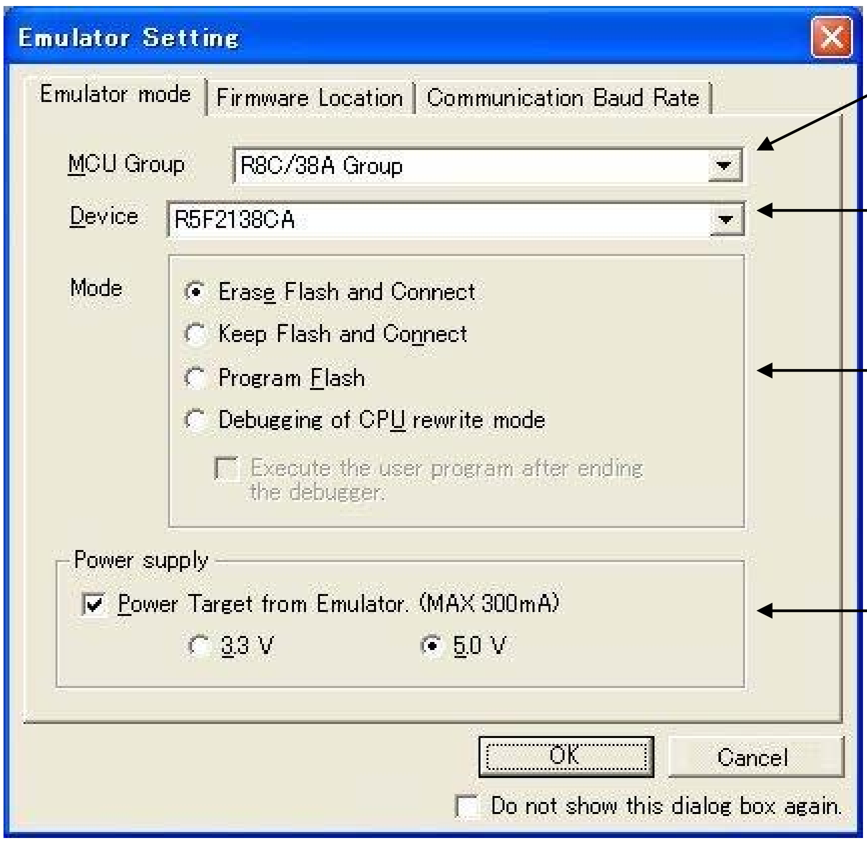
If you check “Do not show this dialog box again.” at the bottom of the [Emulator Setting] dialog box, the [Emulator Setting] dialog box will not be displayed the next time the debugger is launched.

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

Emulator mode | Firmware Location | Communication Baud Rate

MCU Group: R8C/38A Group

Device: R5F2138CA

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.

[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.

[Mode]
Select the mode to be used.
For details, see "5.2 (1) Selecting the Mode" (p.16).

[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.

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 E8a emulator program.
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.
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, the program cannot be debugged in this mode. 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	Debugging only [*1]	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 E8a emulator program.

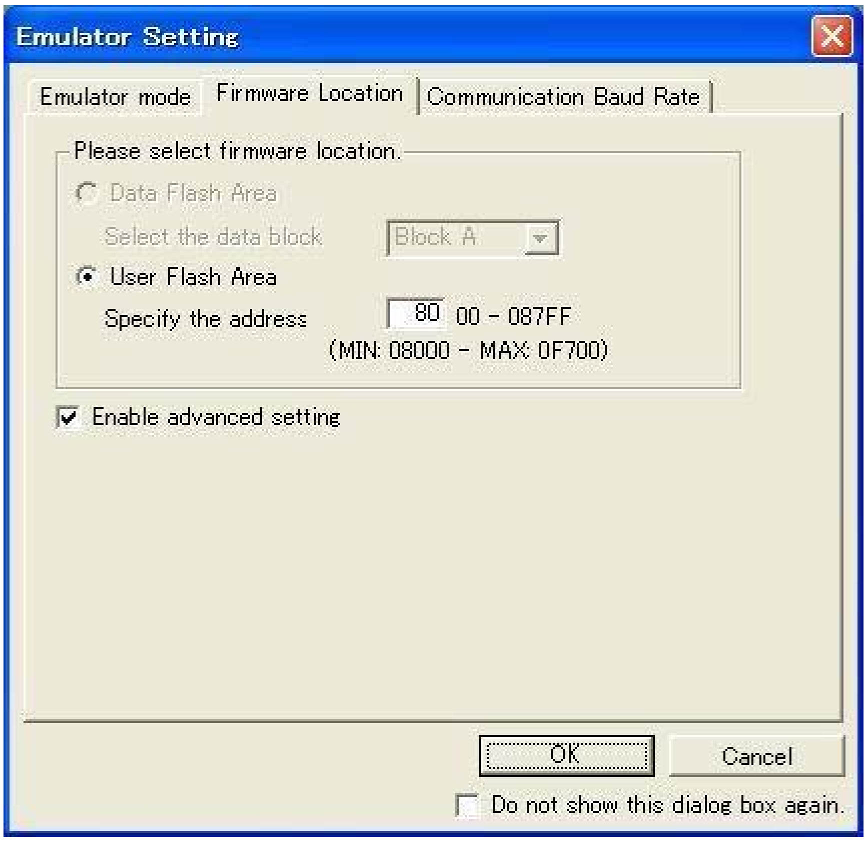
Notes

[*1] These modes are available only for debugging. Programs written in these modes cannot be executed from the CPU. If you want to execute a program from the CPU, 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.

5.3 [Firmware Location] tab

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



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

[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

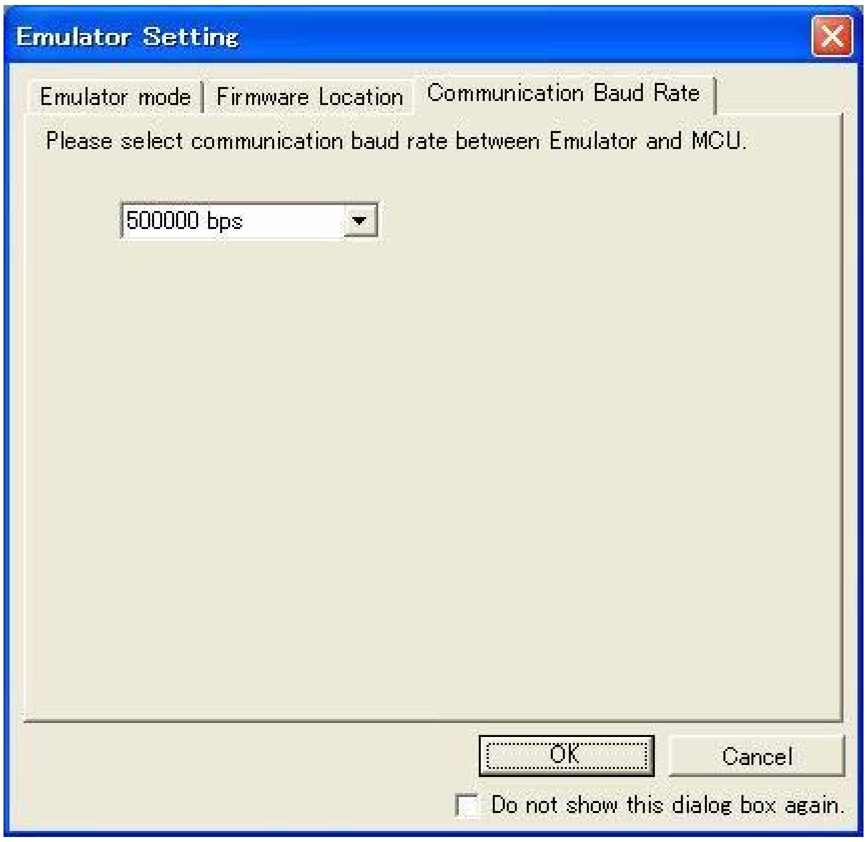
  Specify the ROM area in which the firmware is located. Specify 2k bytes that will not be used by the user system. [*1]

Note

[*1] When using the MCU whose ROM size is other than 128 KB, the options in this [Firmware Location] tab are displayed in gray because this setting is unnecessary.

5.4 [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.



Emulator Setting

Emulator mode | Firmware Location | Communication Baud Rate

Please select communication baud rate between Emulator and MCU.

500000 bps

OK Cancel

Do not show this dialog box again.

Figure 5.4 [Communication Baud Rate] Tab

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

6.1 E8a emulator functions

With the R8C/35A, R8C/36A and R8C/38A Groups, 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 8 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

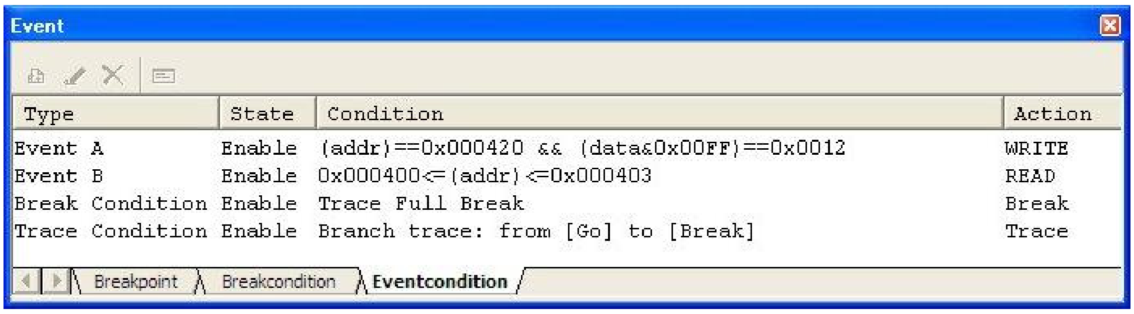
  This function displays data accesses when a data access event is encountered.

  For the data access event and trace condition, set them in the Eventcondition tab 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&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

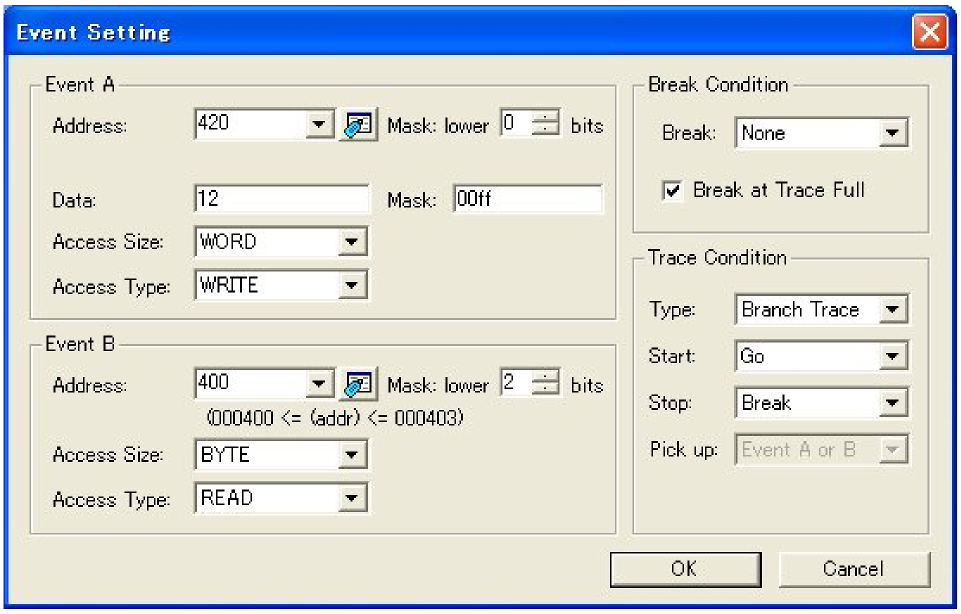
Figure 6.1 Eventpoints Window (Eventcondition tab)

Table 6.1 Display Contents of the Eventcondition Tab

Item	Description
Type	Displays the event types. - Event A - Event B - Break Condition - Trace Condition
State	Shows the event is enable or disable. - Enable - Disable
Condition	Displays the set condition.
Action	For the Event A and Event B, 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: 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 6.2 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) 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 6.3 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 6.4 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. - Event A or B: Breaks the program when either the Event A or Event B is encountered. - Event A and B: Breaks the program when both the Event A and Event B are encountered. - 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.

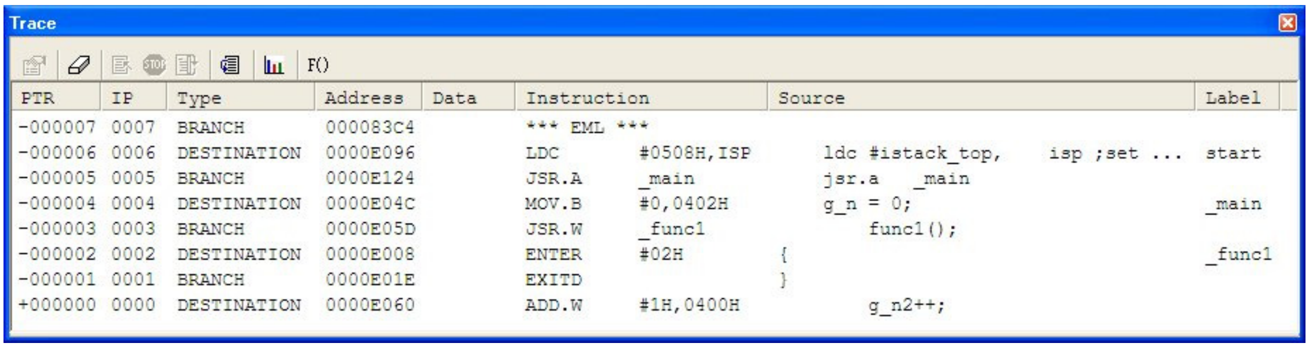
Table 6.5 Trace Condition

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. - Event A or B: Starts a measurement when either the Event A or Event B is encountered. - Event A and B: Starts a measurement when both the Event A and Event B are encountered. - 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. - 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. - Event A or B: Stops a measurement when either the Event A or Event B is encountered. - Event A and B: Stops a measurement when both the Event A and Event B are encountered. - 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. - Event A: Records only data access which encounters the condition of the Event A. - 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 6.6.



```
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 6.6 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 6.7 below.

Table 6.7 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

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

Notes

- 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.
- 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.
- 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.
- 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 allotted for the E8a emulator. Do not change this area allocation, otherwise the E8a emulator will not control the MCU. If settings were changed, disconnect the debugger and then reconnect it.

Table 7.1 Program Area for the E8a Emulator

Group	Part No.	ROM Size		RAM Size	Program Area for E8a Emulator	
		Program ROM	Data Flash		Vector Area	ROM Area
R8C/35A	R5F21357A	48 KB	1 KB (4 blocks)	4.0 KB	FFE4h - FFE7h, FFE8h - FFEBh, FFECh - FFEFh, FFF4h - FFF7h, FFF8h - FFFBh, FFFCh - FFFEh	-
	R5F21358A	64 KB		6.0 KB		-
	R5F2135AA	96 KB		8.0 KB		-
	R5F2135CA	128 KB		10 KB		2 KB of the ROM area [*1]
R8C/36A	R5F21364A	16 KB		1.5 KB		-
	R5F21365A	24 KB		2.0 KB		-
	R5F21366A	32 KB		2.5 KB		-
	R5F21367A	48 KB		4.0 KB		-
	R5F21368A	64 KB		6.0 KB		-
	R5F2136AA	96 KB		8.0 KB		-
	R5F2136CA	128 KB		10 KB		2 KB of the ROM area [*1]
R8C/38A	R5F21386A	32 KB		2.5 KB		-
	R5F21387A	48 KB		4.0 KB		-
	R5F21388A	64 KB		6.0 KB		-
	R5F2138AA	96 KB		8.0 KB		-
	R5F2138CA	128 KB		10 KB		2 KB of the ROM area [*1]

Note

[*1] 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.

(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	0000h
	USP	0000h
	ISP	05FFh
	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 Tables 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
000Ah	Protect register	PRCR	Bit 0
0023h	High-speed on-chip oscillator control register 0	FRA0	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	RMAD0	All bits	[*1]
01C3h	Address match interrupt enable register 0	AIER0	All bits	[*1]
01C4h - 01C6h	Address match interrupt register 1	RMAD1	All bits	[*1]
01C7h	Address match interrupt enable register 1	AIER1	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. If the MCU is reset (hardware reset) while executing the user program, control is transferred to the E8a emulator program and the user program is forced to stop. Do not use the software reset, power-on reset, voltage monitor 0 reset, and watchdog timer reset, otherwise the E8a emulator will run out of control.

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.

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

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

(9) Count source protection mode

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

7.2 Flash memory

7.2.1 Notes on debugging in CPU rewrite mode

(1) CPU rewrite can be executed only for the data area. If it is executed for the program area, the E8a emulator cannot control the MCU.

(2) When rewriting the data area, do not halt the user program, while setting up the CPU rewrite mode and releasing it. If 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.

(3) 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.

(4) 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.

7.2.2 Note on rewriting flash memory

(1) Do not reset nor execute debugging operations to the MCU when rewriting the flash memory.
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

Do not rewrite the flash area from the memory window, etc., except from the user program during user program execution.

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.

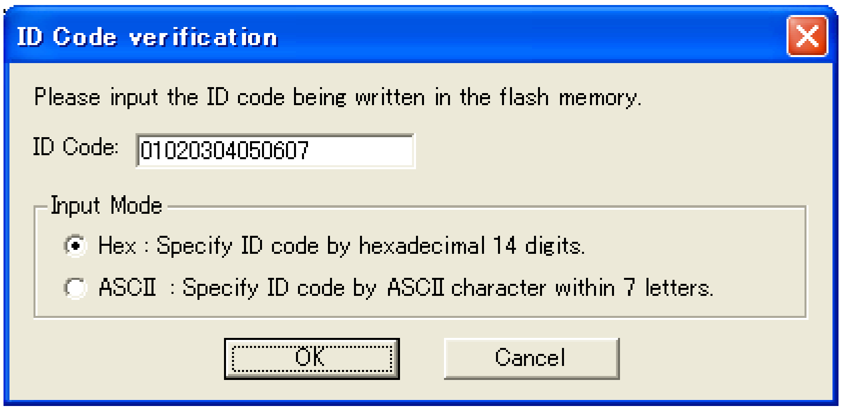
The ID code in Table 7.5 written to the flash memory of the MCU must match the ID code displayed in the Figure 7.1 [ID Code verification] Dialog Box at debugger startup, otherwise the debugger cannot be launched. Note that when the ID code is FFh, FFh, FFh, FFh, FFh, FFh, FFh, the ID code is regarded 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:	Contents of the user program
- Modes other than 'Program Flash' mode:	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


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

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.

7.3 Debugging during a watchdog timer operation

When running the E8a emulator program, the 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.

The E8a emulator sets each of bit 0 and bit 7 of the option function select register (OFS: 0FFFFh) 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 stopped after reset.
- b7: Count source protection mode after reset select bit	1: Count source protect mode disabled after reset

Also, the E8a emulator sets the lower 4 bits of the option function select register 2 (OFS2: 0FFDBh) to 1111b.

- b1, b0: Watchdog timer underflow period set bit	11: 3FFFh
- b3, b2: Watchdog timer refresh acknowledgement period set bit	11: 100%

7.4 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.

7.5 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 divided by 2 (approx. 20MHz, etc.) to operate. However, the peripheral features operate with the clock specified by the user program.

Also, when the user program halts, the following registers may not be overwritten in some cases.

DTMF control register:	016Ch
DTMF count register 0:	016Eh
DTMF count register 1:	016Fh
Timer RC counter:	0126h, 0127h
Timer RD counter:	0156h, 0157h
Timer RG counter:	0176h, 0177h

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

During a user program halt, interrupts are not accepted although peripheral I/Os continue to run. 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.

7.6 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 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. 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.

When the program is forcibly stopped or when the memory is referred to or modified in stop mode or wait mode, these mode will be cancelled.

(4) Note on debugging at less than 2.7V

As flash rewrite occurs when the operations below are executed, if the operating voltage of the MCU is less than 2.7V, do not perform these operations:

- Downloading the user program
- Setting and canceling PC breaks (Setting/canceling event breaks are available)
- Rewriting the value of the Flash memory in the memory window

(5) Note on the CPU clock

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

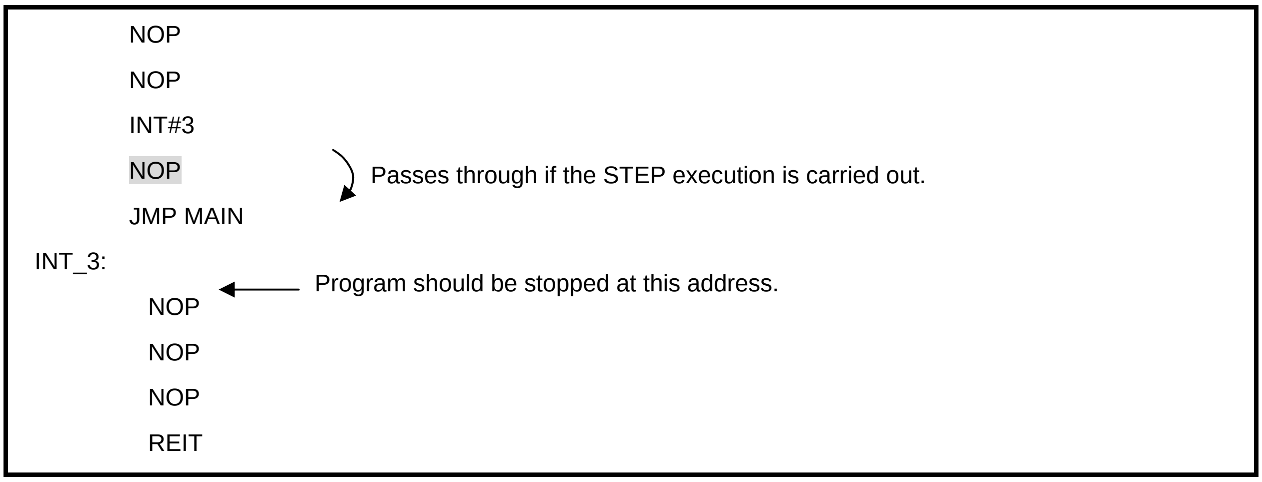
(6) Low-current-consumption read mode

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.

(7) 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).

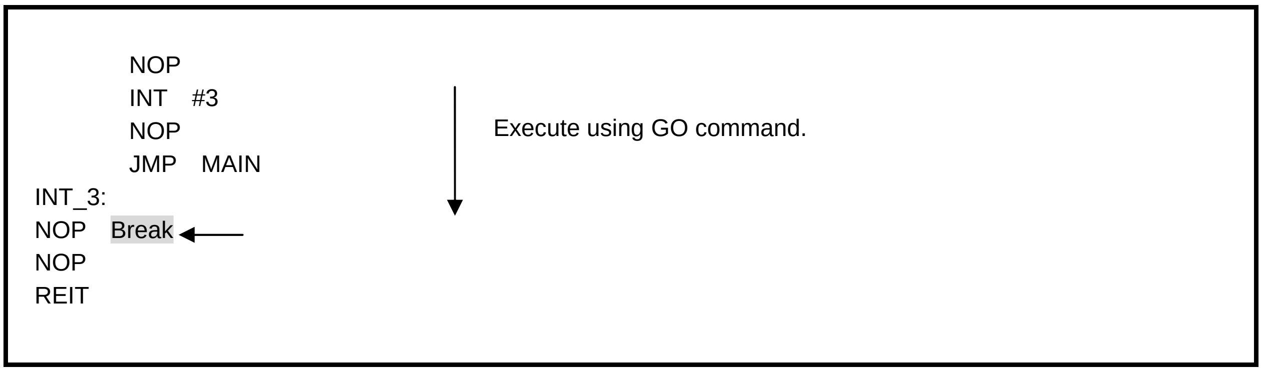


```
          NOP
          NOP
          INT#3
          NOP          Passes through if the STEP execution is carried out.
          JMP MAIN
INT_3:
            NOP  <---  Program should be stopped at this address.
            NOP
            NOP
            REIT
```

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                 Execute using GO command.
          NOP
          JMP   MAIN
INT_3:
NOP   Break <---
NOP
REIT
```

Figure 7.3 Example of INT Instruction

(8) DTC during a user program halt

When the user program halts, or when the memory is referred to or modified while the user program is runnnig, data transfer using DTC is prohibited.

(9) 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 Connecting the R8C/35A, R8C/36A and R8C/38A

Publication Date:	Jun. 01, 2009	Rev.1.00
Published by:	Sales Strategic Planning Div. Renesas Technology Corp.	
Edited by:	Microcomputer Tool Development Department Renesas Solutions Corp.	

E8a Emulator
Additional Document for User's Manual

Renesas Technology Corp.
2-6-2, Ote-machi, Chiyoda-ku, Tokyo, 100-0004, Japan