E8a Emulator

Additional Document for User's Manual
R0E00008AKCE00EP34

Renesas Microcomputer Development Environment System
M16C Family / M16C/60 Series
Notes on Connecting the M16C/64

Renesas Electronics
www.renesas.com

Rev.3.00   Apr 2010

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

RENESAS

2. E8a Emulator Specifications

2.1 Emulator specifications

Table 2.1 shows the E8a emulator specifications for the M16C/64 Group.

Table 2.1   E8a Emulator Specifications for the M16C/64 Group

Target MCUs		M16C Family M16C/60 Series M16C/64 Group
Available operating modes		Single-chip mode, Memory expansion mode * Microprocessor mode is not supported.
Power voltages	M16C/64	2.7 - 5.5 V (f(BCLK)=25MHz)
Debug functions		
Break functions		- Address match break, 4 points - PC break points (maximum 255 points) - Forced break
Trace functions		None
Flash memory programming function		Available
User interface		Clock-synchronous serial (communication via P64/P65/P66/P67)
MCU resources to be used		- ROM size: 3 KB - RAM size: 128 bytes - Stack 14 bytes - Address match interrupt - UART1 function and P64/P65/P66/P67 - Pins P50 and P55
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		M16C E8a emulator debugger V.1.02 Release 00 or later

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

<div style="border:2px solid black">

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).

</div>

```
////////////////////////////////////////////////////
 *.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
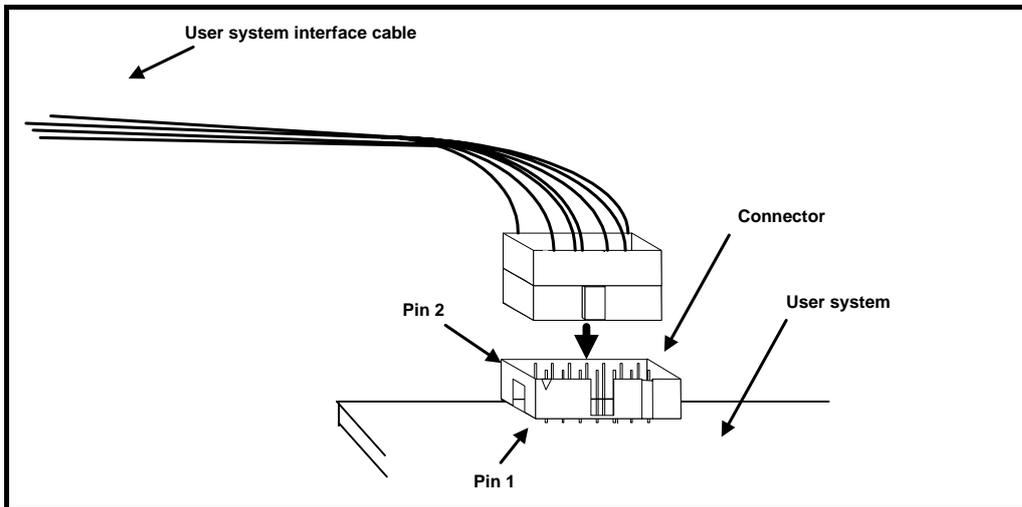
Figure 3.1   Connecting the User System Interface Cable with an E8a Connecting Connector

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	M16C/64 MCU Signals
1	P65(SCLK)
2	Vss
3	CNVss
4	P55(EPM)
5	P67(TxD)
6	Vss
7	P50(CE)
8	Vcc
9	P64(BUSY)
10	Vss
11	P66(RxD)
12	Vss
13	RESET#
14	Vss
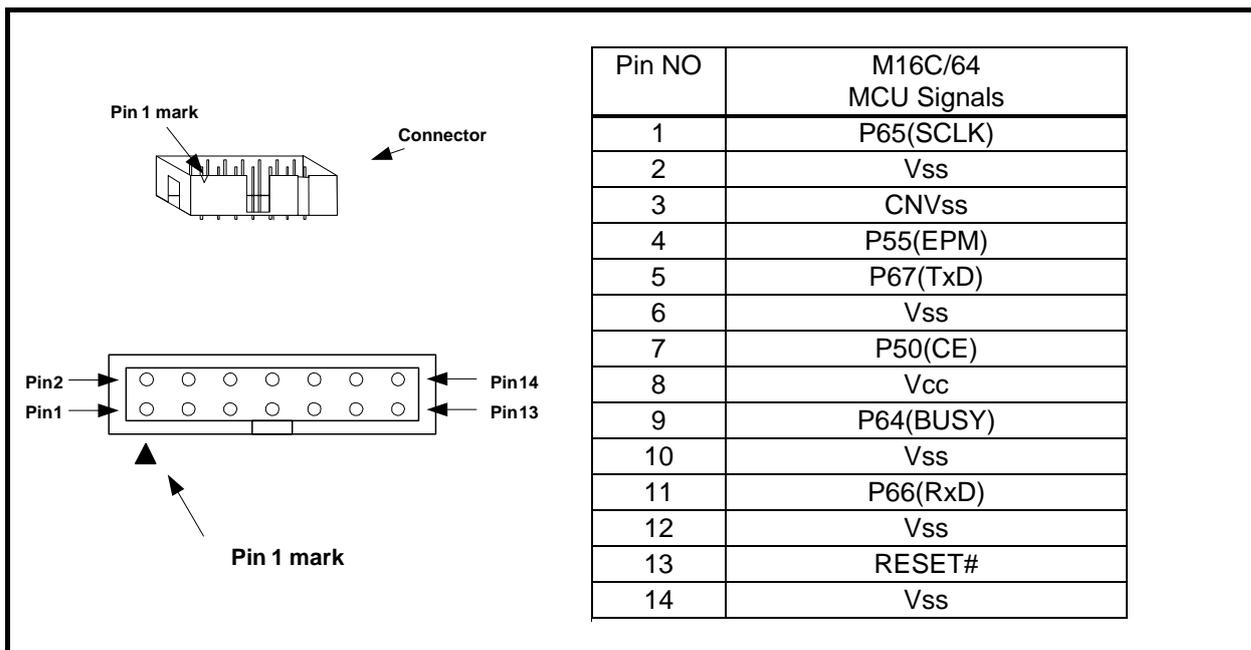
Figure 3.2   E8a Connecting Connector Pin Assignments

Notes

● Pin 14 is used for checking the connection between the E8a and the user system, and 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.

● 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

The following show examples of pin handling for connecting the E8a. When using the E8a as a programmer, the connection specification between the E8a and the MCUs is the same as shown below.

- Single power supply and single-chip mode: See Figure 4.1.
- Single power supply and memory expansion mode: See Figure 4.2.
- Dual power supply and single-chip mode: See Figure 4.3.
- Dual power supply and memory expansion mode: See Figure 4.4.
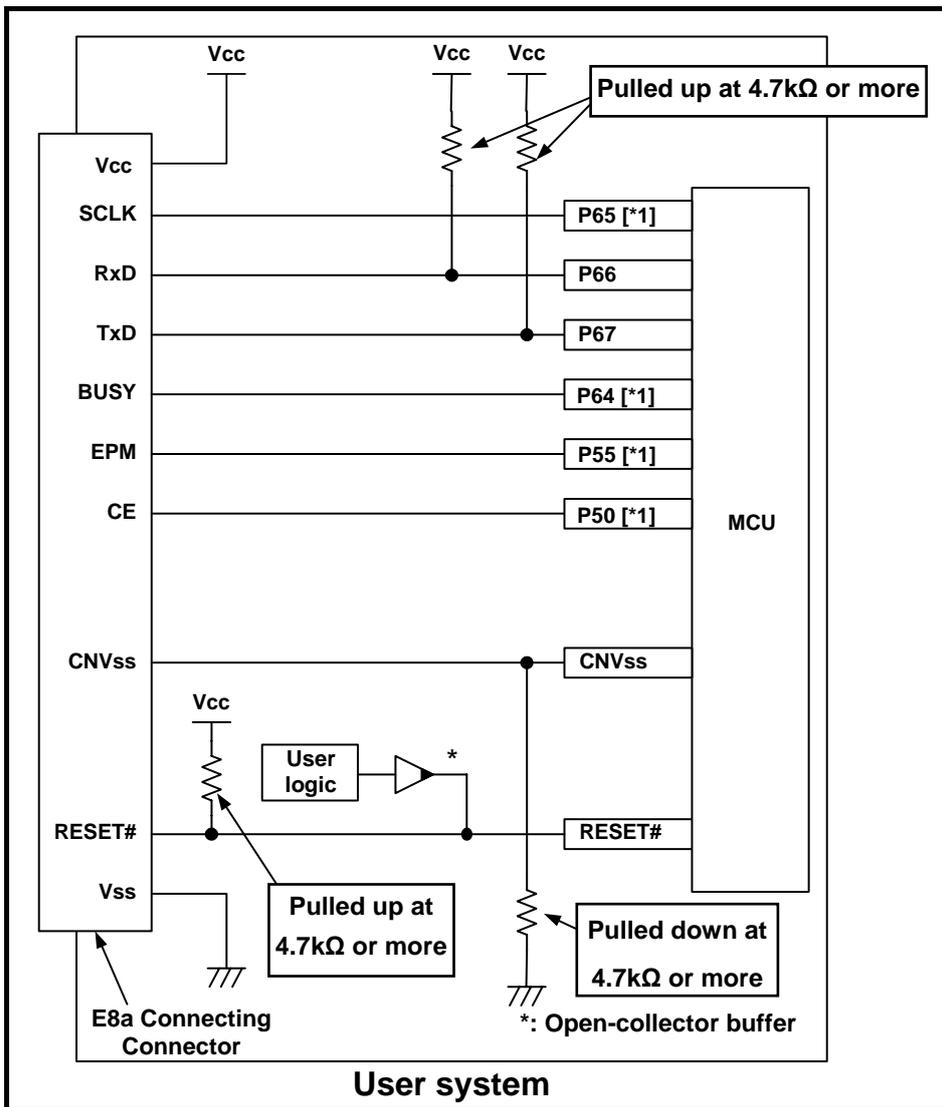
Figure 4.1   Example of an E8a Connection (Single Power Supply and Single-chip Mode)

Note

[*1]   For details on setting pins P64 and P65, refer to "SCLK, RxD, TxD and BUSY pins" on page 14.
        For details on setting pins P50 and P55, refer to "EPM# and CE# pins" on page 15.
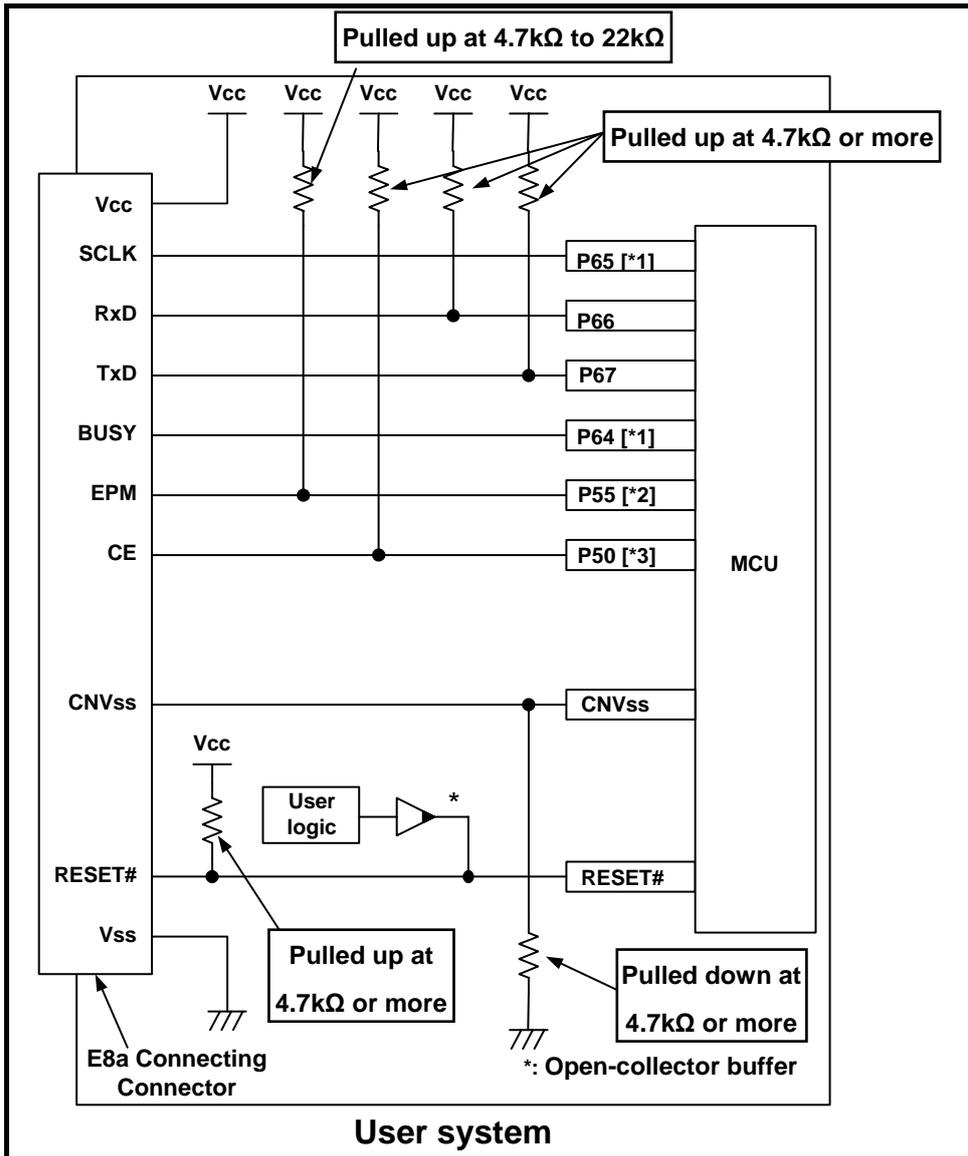
Figure 4.2   Example of an E8a Connection (Single Power Supply and Memory Expansion Mode)

Notes
[*1]   For details on setting pins P64 and P65, refer to "SCLK, RxD, TxD and BUSY pins" on page 14.
[*2]   The HOLD# signal cannot be used. Pull up P55 on the user system.
[*3]   P50 is used as the WRL#/WR# pin. The E8a emulator outputs "H" to the CE pin when going to boot mode (resetting the MCU). In other cases, the CE pin is in a Hiz state. This prevents signal collision between the E8a emulator and the MCU. The WRL#/WR# pin does not affect the memory because the pin has a low active signal.
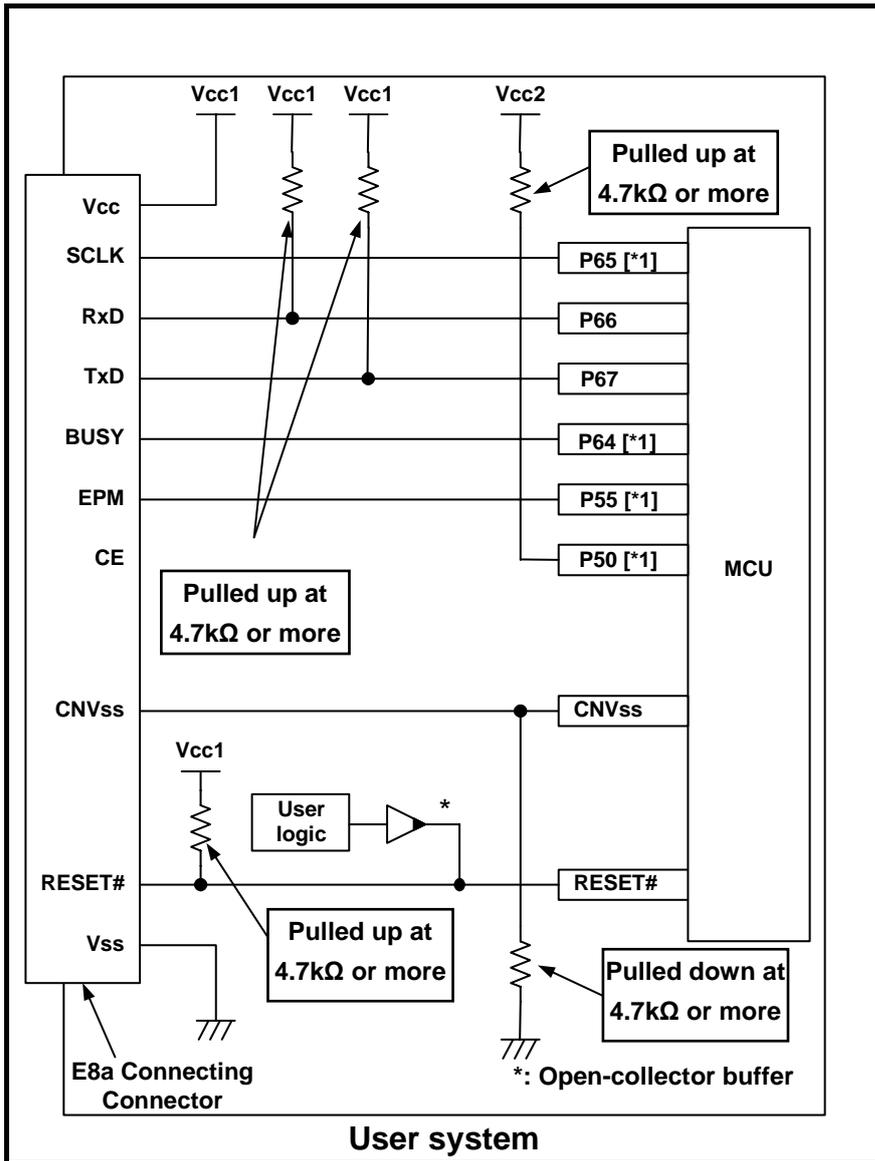
Figure 4.3   Example of an E8a Connection (Dual Power Supply and Single-chip Mode)

Note

[*1]   For details on setting pins P64 and P65, refer to "SCLK, RxD, TxD and BUSY pins" on page 14.
        For details on setting pins P50 and P55, refer to "EPM# and CE# pins" on page 15.
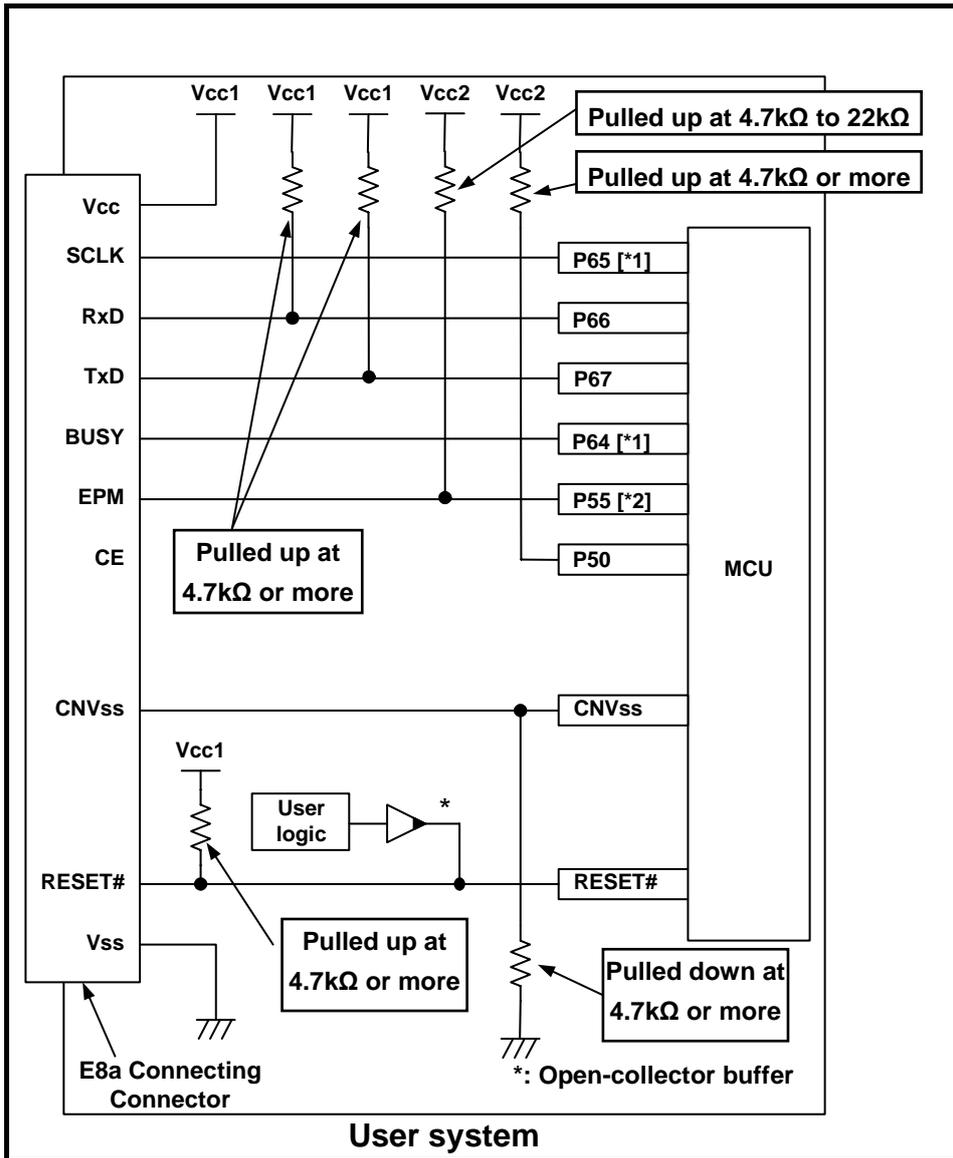
Figure 4.4   Example of an E8a Connection (Dual Power Supply and Memory Expansion Mode)

Notes

[*1]   For details on setting pins P64 and P65, refer to "SCLK, RxD, TxD and BUSY pins" on page 14.

[*2]   The HOLD# signal cannot be used. Pull up P55 on the user system.

(1) SCLK, RxD, TxD and BUSY pins

Pins P64(BUSY), P65(SCLK), P66(RxD) and P67(TxD) are used exclusively by the E8a emulator.

Connect pins P66 and P67 to the E8a emulator after pulling up the MCU pins at the Vcc (Vcc1) level.

For P64 and P65, pull up the pins at the Vcc (Vcc1) level or pull down them according to the MCU pin state after disconnecting the E8a emulator.

P64 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 4.12 on page 18).
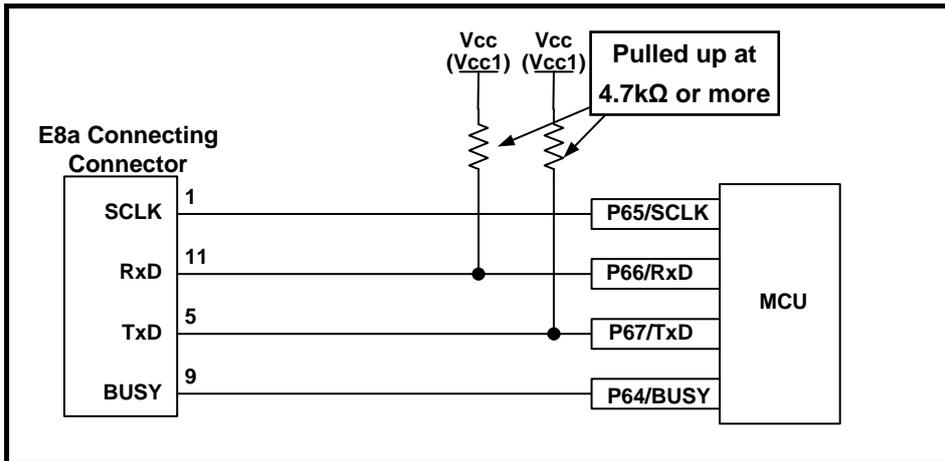
Figure 4.5   E8a Emulator and MCU Connection

(2) EPM# and CE# pins

The E8a emulator uses pins P50(CE#) and P55(EPM#) for MCU control.

Connect the E8a emulator to the MCU pins.

1.   Single power supply and single-chip mode

For P50 and P55, pull up the pins at the Vcc level or pull down them according to the MCU pin state after disconnecting the E8a emulator. P50 and P55 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 4.12 on page 18).
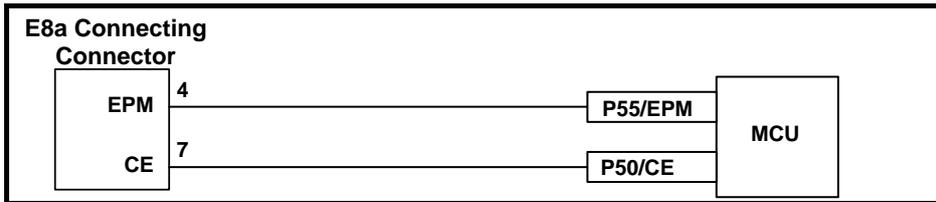
Figure 4.6   Connection of E8a Emulator and Pins P50 and P55 (Single Power Supply and Single-chip Mode)

2.   Single power supply and memory expansion mode
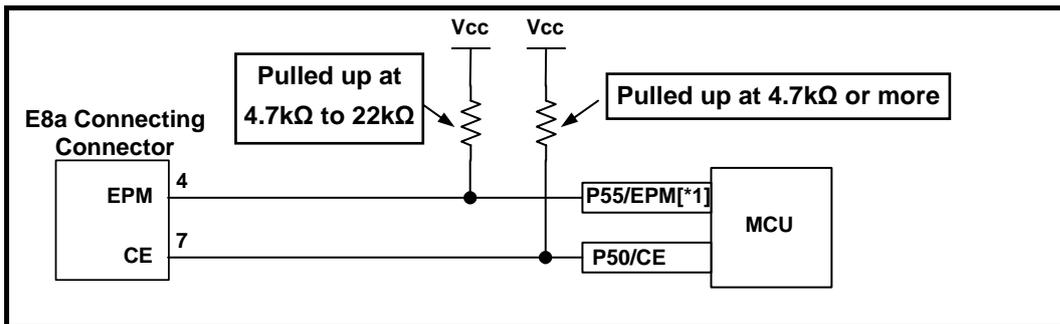
Figure 4.7   Connection of E8a Emulator and Pins P50 and P55 (Single Power Supply and Memory Expansion Mode)

Note
[*1]   The HOLD# signal cannot be used. Pull up P55 at the Vcc level on the user system.

3. Dual power supply and single-chip mode

Pull up P55 at the Vcc2 level or pull down it according to the MCU pin state after disconnecting the E8a emulator. P55 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 4.12 on page 18).
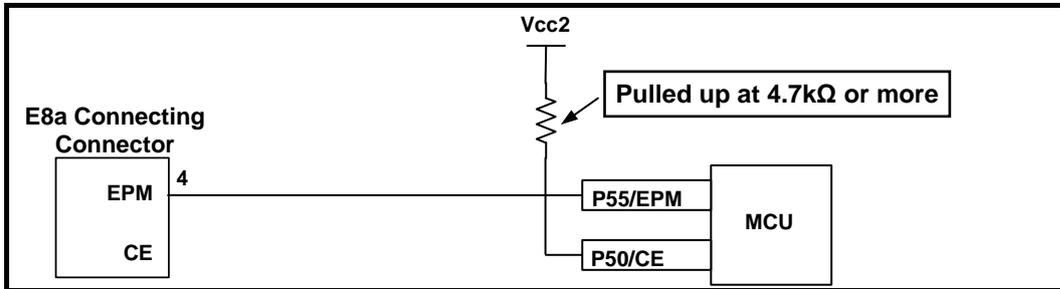
Figure 4.8   Connection of E8a Emulator and Pins P50 and P55 (Dual Power Supply and Single-chip Mode)

4. Dual power supply and memory expansion mode
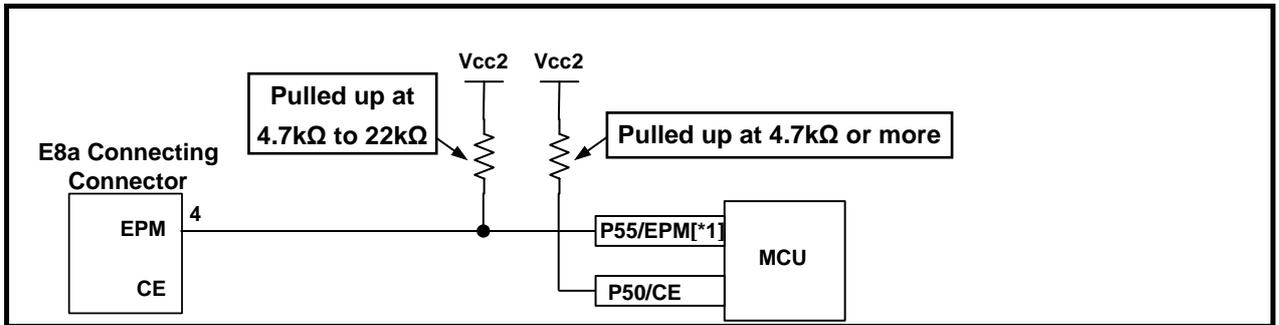
Figure 4.9   Connection of E8a Emulator and Pins P50 and P55 (Dual Power Supply and Memory Expansion Mode)

Note

[*1]   The HOLD# signal cannot be used. Pull up P55 at the Vcc2 level on the user system.

(3) 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.
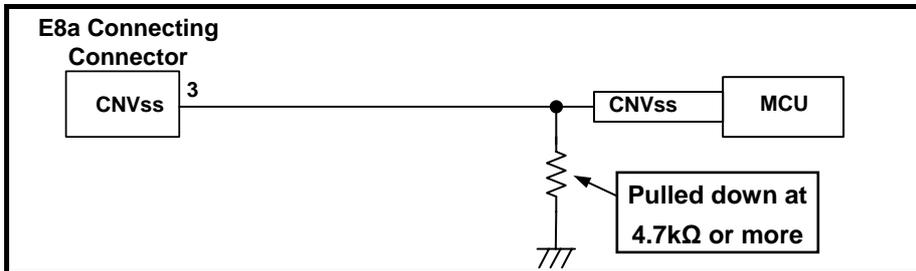
Figure 4.10   E8a Emulator and CNVss Pin Connection

(4) 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.
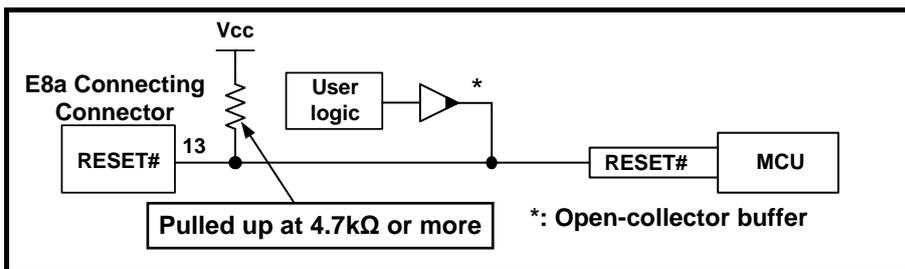
Figure 4.11   Example of a Reset Circuit

(5) Other pins

- Connect Vss and Vcc to the Vss and Vcc (Vcc1) of the MCU, respectively.
- The amount of voltage input to Vcc (Vcc1, Vcc2) must be within the specified range of the MCU.
- If NMI# interrupts are not used, make sure the NMI# pin is pulled up to the Vcc (Vcc1) 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.

4.2 Interface circuit in the E8a emulator

Figure 4.12 shows the interface circuit in the E8a emulator. Use this figure as a reference when determining the pull-up resistance value.
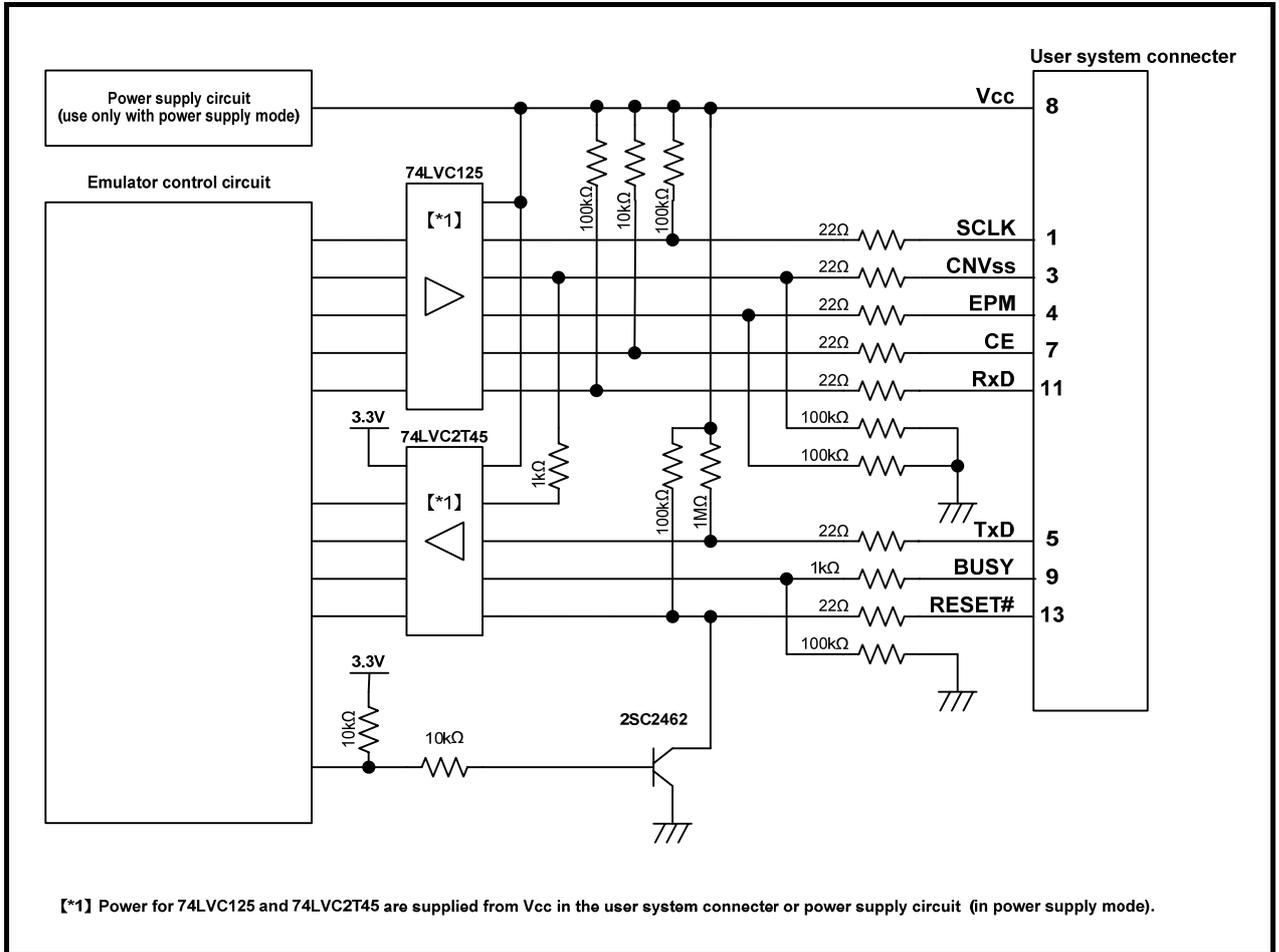
Figure 4.12   Interface Circuit inside the E8a Emulator (For Reference)

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.
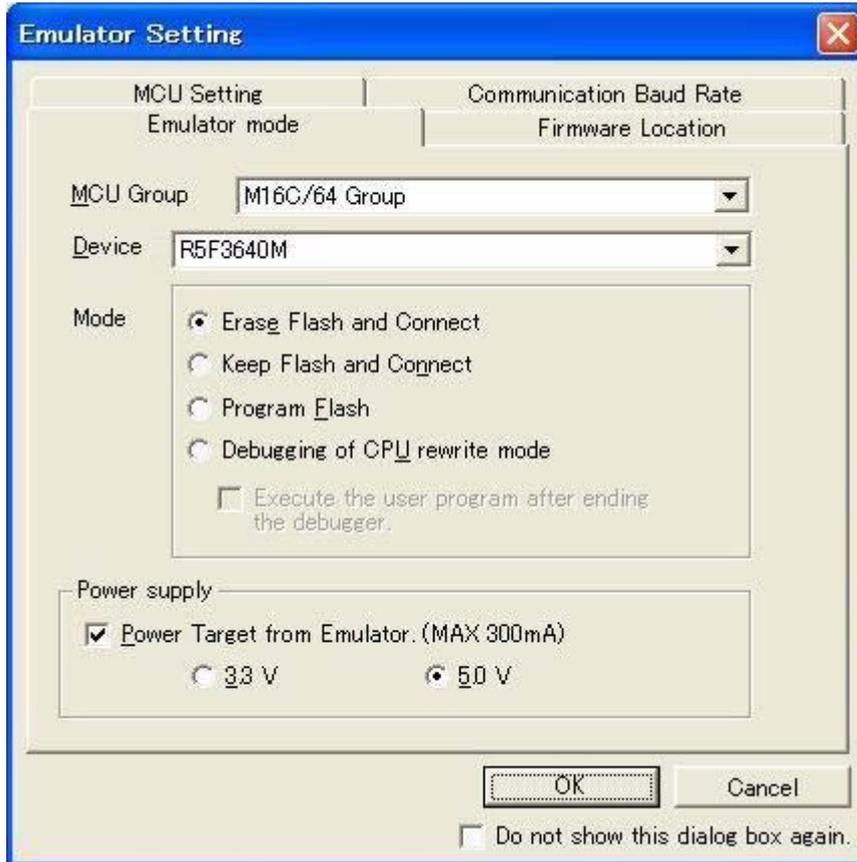
Figure 5.1   [Emulator Setting] Dialog Box

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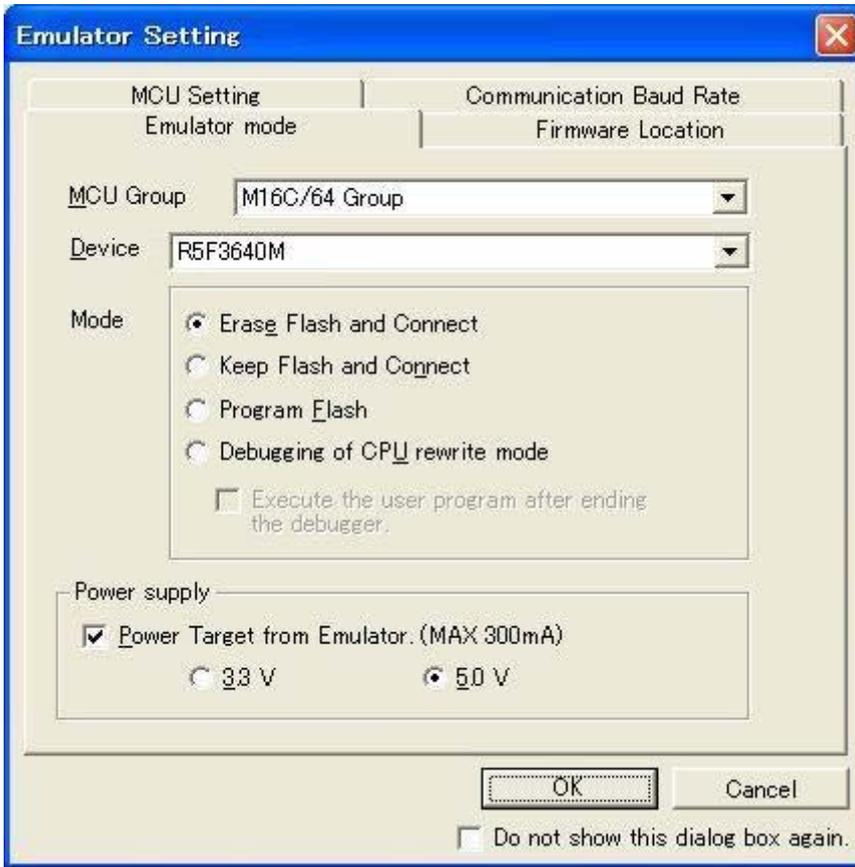
[MCU Group]
Select the name of the MCU group to be used from the [MCU Group] drop-down list.

[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.21).

[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)] check box.

Note that when debugging the system which operates the MCU with a dual power supply, power cannot be supplied from the E8a.

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		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.

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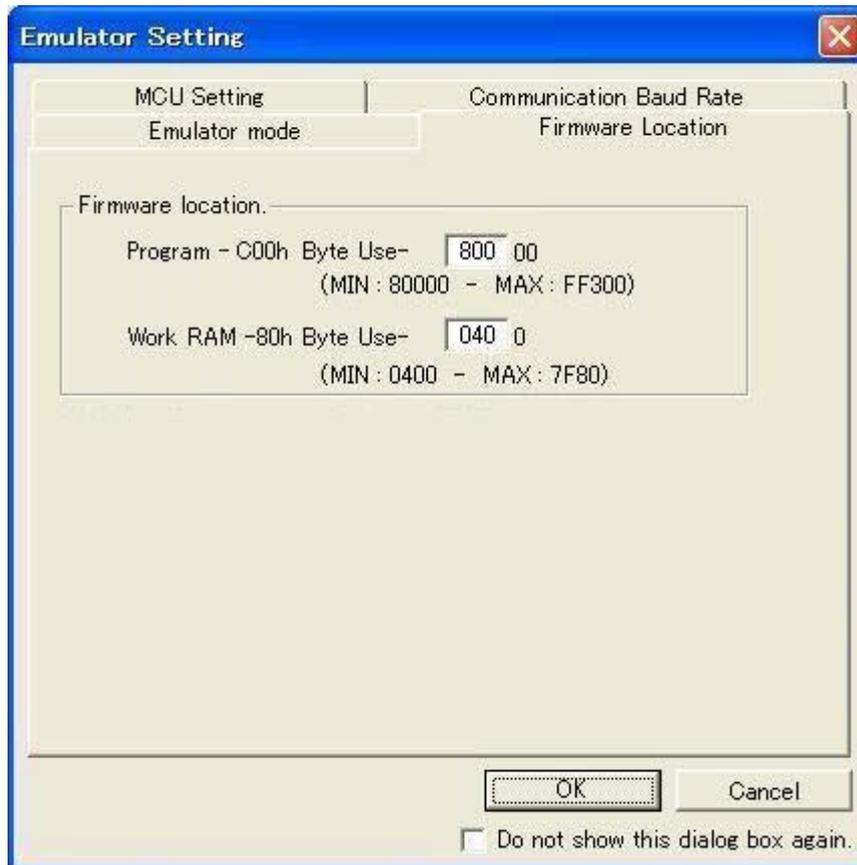
[**Firmware Location**]

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 3k bytes that will not be used by the user system. The data area cannot be specified.
- Work RAM
  Specify the RAM area in which the firmware is located. Specify 128 bytes that will not be used by the user system.

5.4 [MCU Setting] tab

In the [MCU Setting] tab, set the operating condition of the MCU used in the user system.
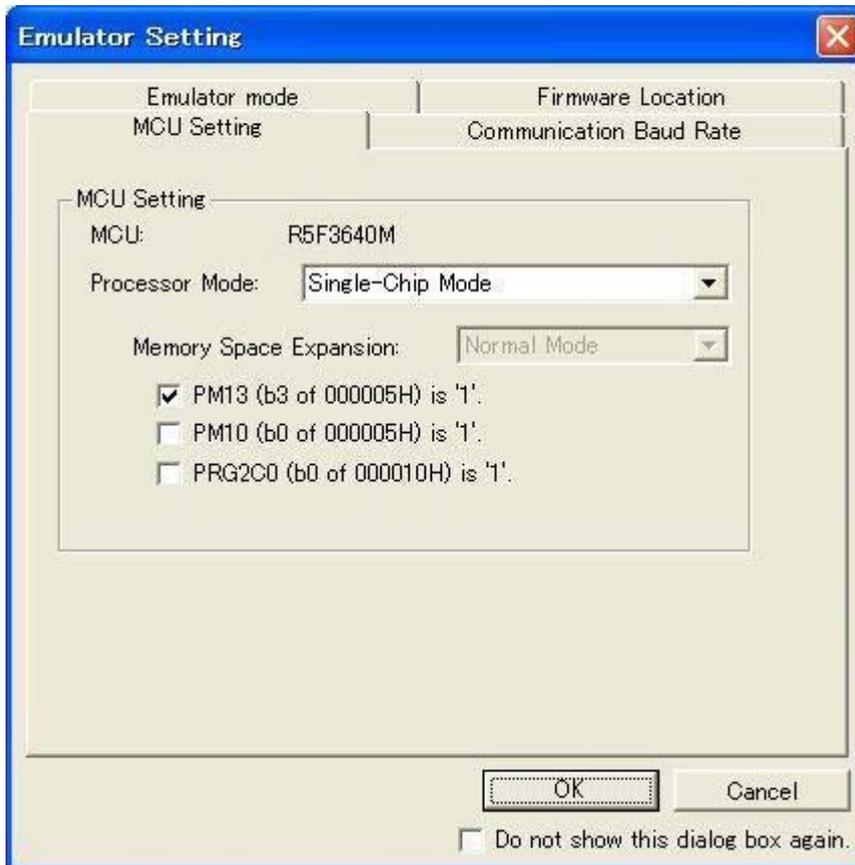
Figure 5.4 [MCU Setting] Tab

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. One of the following can be specified:
- Single-Chip Mode
- Memory Expansion Mode [*1]

Memory Expansion Mode
When Memory Expansion Mode is selected, specify whether the memory space expansion function will be used.

"Normal Mode": When not using the memory space expansion function
"4 MB Mode": When using the memory space expansion function

PM13 (b3 of 000005H) is '1'
Specify whether PM13 (third bit of processor mode register 1) is set. When using the user program with PM13 set to "1", check this option.

PM10 (b0 of 000005H) is '1'
Specify whether PM10 (zero bit of processor mode register 1) is set. When using the user program with PM10 set to "1", check this option.

PRG2C0 (b0 of 000010H) is '1'
Specify whether PRG2C0 (zero bit of program 2 area control register) is set. When using the user program with PRG2C0 set to "1", check this option.

Note

[*1]   - When the external area cannot be rewritten via normal memory access, software breaks cannot be used in that area.
- No address match break can be used in the external memory space.
- The "Go to Cursor" function cannot be used in the external memory space. If this function is used in the external memory space, the program will be in a state of execution.
- To access the memory space expansion area in the download, the editor window (MIX display or disassembled display mode), the memory window or the watch window while operating in the memory space expansion 4 MB mode, be aware that only bank 7 can be accessed. In this case, the data bank offset depends on the offset bits of the data bank register.
- When using the memory space expansion function 4 MB mode, execute the command for the memory space expansion function 4 MB mode to access each bank.

5.5 [Communication Baud Rate] tab

Select communication baud rate between the E8a and MCU in the [Communication Baud Rate] tab. 2000000 bps (default setting) should be selected.
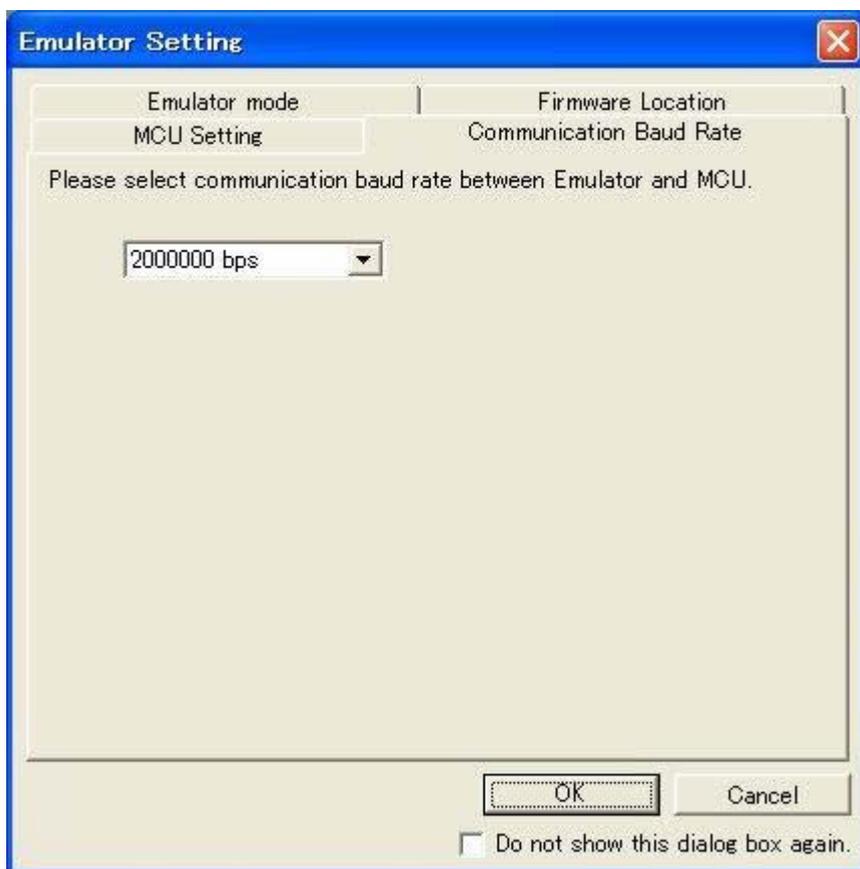
Figure 5.5 [Communication Baud Rate] Tab

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

6.1 Command for memory space expansion function 4 MB mode

6.1.1 Command for memory space expansion function 4 MB mode

The following show the command for memory space expansion function 4 MB mode. These commands can be executed in the command line window.

Command	Description
Memory_Compare_Ext	Compares the memory area (between the start address and the end address) with the memory starting at destination address.
Memory_Display_Ext	Displays memory contents.
Memory_Fill_Ext	Fills an area of memory.
Memory_Find_Ext	Finds a string in a memory range.
Memory_Move_Ext	Moves memory.

6.1.2 Details of command for memory space expansion function 4 MB mode

The following show the details of the command for memory space expansion function 4 MB mode.

(1) Memory_Compare_Ext
- Abbreviation: MCE
- Description

Compares the memory area (between the start address and the end address) with the memory starting at destination address. This cannot be used during program execution.
- Syntax

MCE <bank> <offsetbit> <start> <end> <destination> [<mode>]

Parameter	Type	Description
<bank>	Numeric	Bank (0 – 7)
<offsetbit> *	Numeric	Offset bit (0: no offset, 1: offset)
<start>	Numeric	Start address
<end>	Numeric	End address (including this address)
<destination>	Numeric	Destination address
<mode>	Keyword	Format (optional, default = BYTE)
	BYTE	1 byte
	WORD	2 bytes
	LONG	4 bytes
	DOUBLE	8 bytes

*: When "PM13 is 1" is not selected in the MCU setting dialog box, set the offset bit to "0".

(2) Memory_Display_Ext

 - Abbreviation: MDE

 - Description

   Displays memory contents. This cannot be used during program execution.

 - Syntax

   MDE <bank> <offsetbit> <address> [<length>] [<mode>]

Parameter	Type	Description
<bank>	Numeric	Bank (0 – 7)
<offsetbit> *	Numeric	Offset bit (0: no offset, 1: offset)
<address>	Numeric	Start address
<length>	Numeric	Length (optional, default = 0x100 bytes)
<mode>	Keyword	Display format (optional, default = BYTE)
	BYTE	Bytes
	WORD	Words (2 bytes)
	LONG	Long words (4 bytes)
	ASCII	ASCII
	SINGLE	Single-precision floating-point (4 bytes)
	DOUBLE	Double-precision floating-point (8 bytes)

   *: When "PM13 is 1" is not selected in the MCU setting dialog box, set the offset bit to "0".

(3) Memory_Fill_Ext

 - Abbreviation: MFE

 - Description

   Fills an area of memory. This cannot be used during program execution.

 - Syntax

   MFE <bank> <offsetbit> <start> <end> <data> [<mode>] [<verify>]

Parameter	Type	Description
<bank>	Numeric	Bank (0 – 7)
<offsetbit> *	Numeric	Offset bit (0: no offset, 1: offset)
<start>	Numeric	Start address
<end>	Numeric	End address
<data>	Numeric	Data value
<mode>	Keyword	Data size (optional, default = BYTE)
	BYTE	Byte
	WORD	Word (2 bytes)
	LONG	Long word (4 bytes)
	ASCII	ASCII
	SINGLE	Single-precision floating-point (4 bytes)
	DOUBLE	Double-precision floating-point (8 bytes)
<verify>	Keyword	Verify flag (optional, default = V)
	V	Verification
	N	No verification

   *: When "PM13 is 1" is not selected in the MCU setting dialog box, set the offset bit to "0".

(4) Memory_Find_Ext

    - Abbreviation: MIE

    - Description

      Finds a string in a memory range. This cannot be used during program execution.

    - Syntax

      MIE \<bank\> \<offsetbit\> \<start\> \<end\> \<string\> [\<mode\>]

Parameter	Type	Description
\<bank\>	Numeric	Bank (0 – 7)
\<offsetbit\> *	Numeric	Offset bit (0: no offset, 1: offset)
\<start\>	Numeric	Start address
\<end\>	Numeric	End address (including this address)
\<string\>	Numeric	String to search for
\<mode\>	Keyword	Format (optional, default = BYTE)
	BYTE	Bytes
	WORD	Words (2 bytes)
	LONG	Long words (4 bytes)
	ASCII	ASCII
	SINGLE	Single-precision floating-point (4 bytes)
	DOUBLE	Double-precision floating-point (8 bytes)

    *: When "PM13 is 1" is not selected in the MCU setting dialog box, set the offset bit to "0".

(5) Memory_Move_Ext

    - Abbreviation: MVE

    - Description

      Moves memory. This cannot be used during program execution.

    - Syntax

      MVE \<bank\> \<offsetbit\> \<start\> \<end\> \<destination\> [\<verify\>] [\<mode\>]

Parameter	Type	Description
\<bank\>	Numeric	Bank (0 – 7)
\<offsetbit\> *	Numeric	Offset bit (0: no offset, 1: offset)
\<start\>	Numeric	Source start address
\<end\>	Numeric	Source end address (including this address)
\<destination\>	Numeric	Destination start address
\<verify\>	Keyword	Verify flag (optional, default = V)
	V	Verification
	N	No verification
\<mode\>	Keyword	Format (optional, default = BYTE)
	BYTE	1 byte
	WORD	2 bytes
	LONG	4 bytes
	DOUBLE	8 bytes

    *: When "PM13 is 1" is not selected in the MCU setting dialog box, set the offset bit to "0".

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	RAM Area
M16C/64	R5F36406	128 KB	4 KB (2 blocks)	12 KB	FFFE4h - FFFE7h, FFFE8h - FFFEBh, FFFECh - FFFEFh, FFFF4h - FFFF7h, FFFFCh - FFFFFh	3 KB of the ROM area [*1]	128 B [*1]
	R5F3640D	256 KB		16 KB			
	R5F3640M	512 KB		31 KB			

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#, CNVss, P50, P55, P64, P65, P66, and P67 pins

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

Note

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

(4) SFRs used by the E8a emulator program

The SFRs listed in Table 7.3 are used by the E8a emulator program, not the user program.

- Do not change the registers, otherwise the E8a cannot control the MCU.

- The SFRs listed in Table 7.3 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

Address	Register	Symbol	Bit	Notes on Using the E8a Emulator
020Eh	Address match interrupt enable register	AIER	All bits	[*1]
020Fh	Address match interrupt enable register 2	AIER2	All bits	[*1]
0210h - 0212h	Address match interrupt register 0	RMAD0	All bits	[*1]
0214h - 0216h	Address match interrupt register 1	RMAD1	All bits	[*1]
0218h - 021Ah	Address match interrupt register 2	RMAD2	All bits	[*1]
021Ch - 021Eh	Address match interrupt register 3	RMAD3	All bits	[*1]
0250h	UART transmit/receive control register 2	UCON	Bits 1, 3, 4, 5 and 6	[*2]
0258h	UART1 transmit/receive mode register	U1MR	All bits	[*1]
025Ah, 025Bh	UART1 transmit buffer register	U1TB	All bits	[*1]
025Ch	UART1 transmit/receive control register 0	U1C0	All bits	[*1]
025Dh	UART1 transmit/receive control register 1	U1C1	All bits	[*1]
025Eh, 025Fh	UART1 receive buffer register	U1RB	All bits	[*1]
03ECh	Port P6 register	P6	Bits 4, 5, 6 and 7	[*2]
03EEh	Port P6 direction register	PD6	Bits 4, 5, 6 and 7	[*2]

Notes

[*1]   Do not change this register value.

[*2]   Do not change the value of the bits listed in the column to the left. When operating this register, make changes using the bit operation instructions to avoid changing the bit values.

[*3]   UART1 transmit interrupt control register S1TIC and UART1 receive interrupt control register S1RIC always read out values used by the emulator.

(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) Reset

The reset vector is used by the E8a emulator program. If the MCU is reset (hardware reset 1) 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, watchdog timer reset and oscillation stop detection reset, otherwise the E8a emulator will not operate normally.

(7) 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.

(8) 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.

(9) 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.

7.2 Flash memory

7.2.1 Note on debugging in CPU rewrite mode

(1) Unrewritable area in CPU rewrite mode

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

- Block 0 area (addresses FF0000h - FFFFFFh) and block containing the E8a emulator program

(2) Operation in CPU rewrite mode

- 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.
- 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.

7.2.2 Note on rewriting flash memory

(1) Do not reset the MCU when rewriting the flash memory.

The flash memory is rewritten when the "Flash memory write end" is displayed in the output window of the High-performance Embedded Workshop. If the MCU is reset 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 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.

RENESAS

7.2.4 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 7.4 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 7.4   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
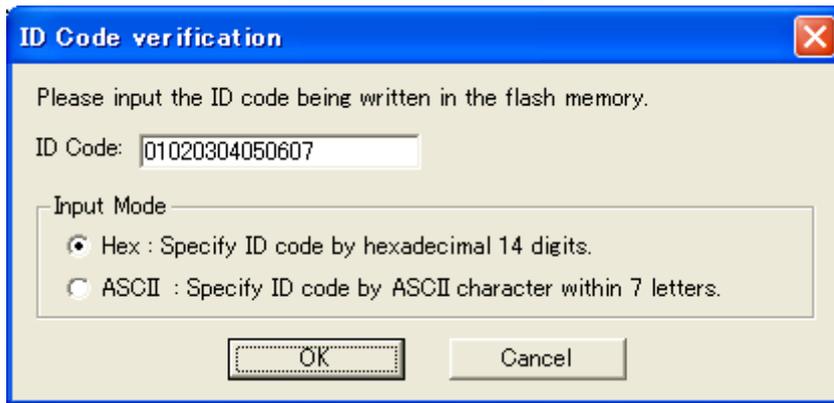
Figure 7.1 [ID Code verification] Dialog Box

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 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.

Note that when debugging the system which operates the MCU with a dual power supply, power cannot be supplied from the E8a.

7.4 Operation during a user program halt

(1) Peripheral I/Os during a 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.

7.5 Debug functions

(1) Memory access during user program execution

When referring to or modifying the memory contents, the user program is temporarily halted. For this reason, a real-time emulation cannot be performed.

When a real-time emulation is necessary during a program execution, disable 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.

(2) Setting of address match break during user program execution

When adding or cancelling the address match break, the user program is temporarily halted. For this reason, a real-time emulation cannot be performed.

(3) 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.

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.

RENESAS

(4) "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.

(5) 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.

(6) Low power consumption mode

When debugging in low power consumption 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 low power consumption mode is cancelled.

(7) Notes on downloading the user program

When the user program is downloaded, the operation clock is switched to Xin. After the download completed, the CPU is reset. Therefore, the CPU is always reset whether the "Reset CPU after download module" checkbox in the [Options] tab of the [Debug Settings] dialog box is selected or not. To open the dialog box, select the [Debug] menu -> [Debug Settings].

RENESAS

(8) 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 Register
    Interrupt request bit (bit 3)   0: Interrupt not requested [*1]

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

Note

[*1]   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) 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
                            Passes through if the STEP execution is carried out.
        NOP

        JMP MAIN
INT_3:
        NOP    ◄——— Program should be stopped at this address.

        NOP

        NOP

        REIT
```
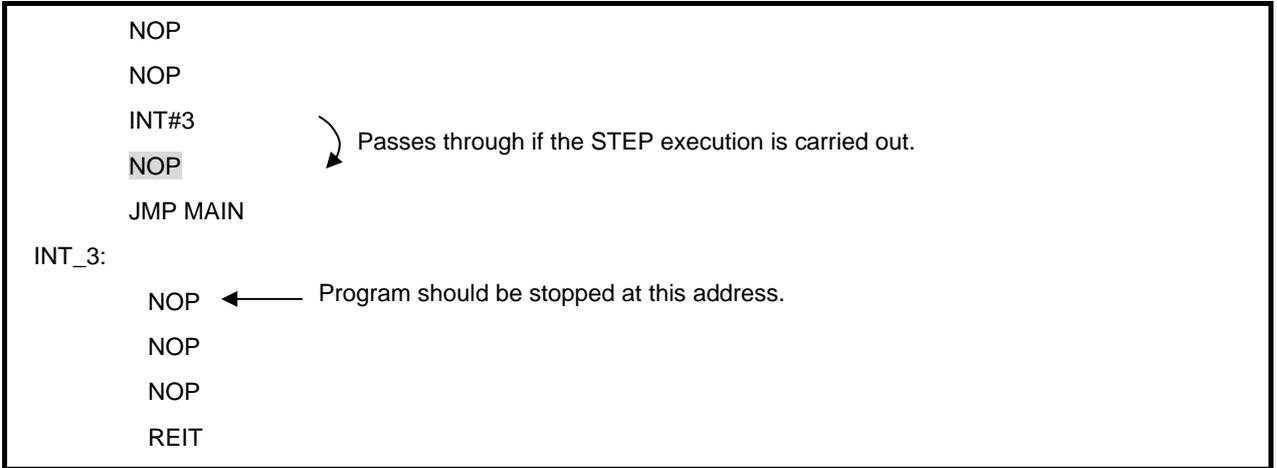
Figure 7.2   Example of Software Interrupt Instruction

  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
        NOP                       Execute using GO command.
        JMP   MAIN
INT_3:
NOP   Break ◄———
NOP
REIT
```
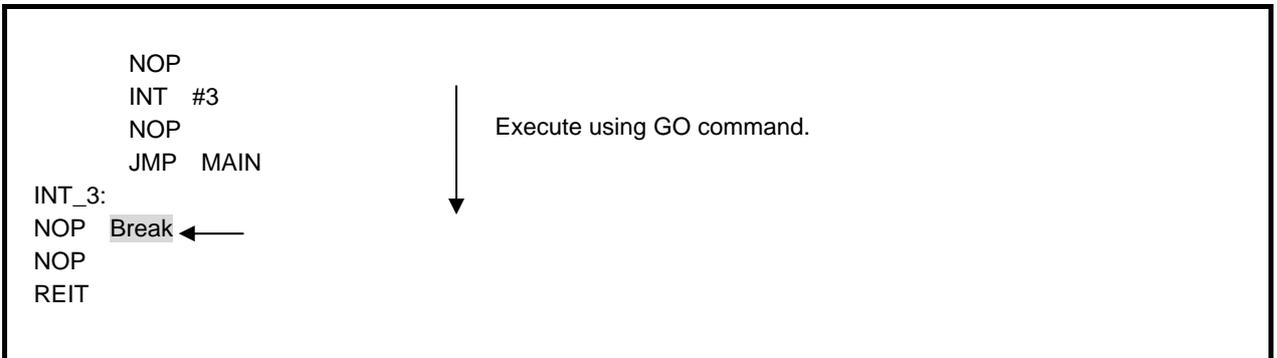
Figure 7.3   Example of INT Instruction

RENESAS

Colophon 1.0

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