

# **R8C,M16C Integrated Development Environment for RL78 Family**

**Migration to New Integrated Development Environment “CubeSuite+”: Onchip Debug**

**Renesas Electronics Corporation**  
MCU Business Unit  
MCU Software Division  
MCU Tool Product Marketing Department  
2012/10/18 Rev. 1.02

**R20UT2150EJ0100**

# Introduction

This document describes how to migrate from the High-performance Embedded Workshop for R8C,M16C Family to CubeSuite+ for RL78 and how to operate E1 and E20 emulators in the CubeSuite+ environment, this explanation is based on **CubeSuite+ V1.02.00**.

For toolchains, refer to the following three materials.

- *Integrated Development Environment for RL78 Family Migration to Integrated Development Environment “CubeSuite+”: Build,.*
- *Integrated Development Environment for RL78 Family Migration to Integrated Development Environment “CubeSuite+”: Coding,.*
- *Integrated Development Environment for RL78 Family Migration to Integrated Development Environment “CubeSuite+”: Starting,.*

Also refer to the tutorial guide provided by CubeSuite+ for how to use tools.

The tutorial guide is available by selecting [Help] -> [Tutorial] from the CubeSuite+ menu.






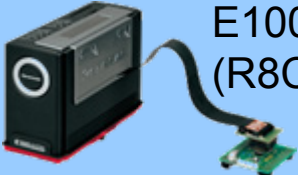



Tutorial Guide

# Contents

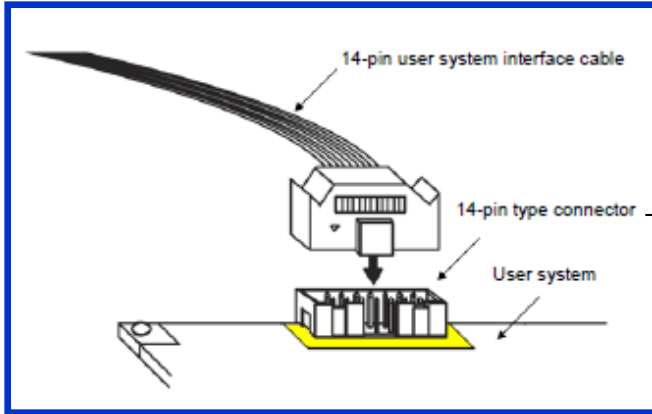
- 1. Integrated Development Environment and Emulators
- 2. Differences between the Target Interfaces (for OCD)
- 3. Changing the Debugger
- 4. Entering an ID Code
- 5. Securing Resources
- 6. Setting the On-Chip Debugging Option Byte
- 7. Where Do We Make Settings when Connecting an Emulator?
- 8. Connecting an Emulator
- 9. Disconnecting the Emulator
- 10. Downloading a Program
- 11. Registering Additional Download Files
- 12. Starting/Stopping a Program
- 13. Difference in MCU Operation during a Break (Peripheral Break Function)
- 14. Viewing/Changing Memory Data and Variables While the Program Is Running
- 15. Automatically Updating Memory Data and Variables While the Program Is Running
- 16. Setting Breakpoints
- 17. Causing a Break on Access to a Variable
- 18. Filling Memory
- 19. Saving Memory Data
- 20. Flash Self-Programming
- 21. How to program to check Operation on the Stand-Alone MCU
- 22. Action Event (Printf Event)
- 23. Viewing Lists of Variables and Functions
- 24. Analytical Graphs
- 25. Debugging Functions of Emulators (OCD)

# 1. Integrated Development Environment and Emulators

	R8C,M16C	RL78
Integrated Development Environment (IDE)	 <p>High-performance Embedded Workshop</p>	 <p>A new integrated development environment that supports 8- to 32-bit MCUs from Renesas</p>
On-chip Debuggers (OCD)	 <p>E8a(R8C,M16C) E1(R8C)</p>	<p>Provides basic debugging functions at a low price</p> 
Full-spec. Emulators*	<p>CPE (R8C,M16C)</p>  <p>E100 (R8C,M16C)</p> 	<p>High-performance full-spec. emulators with more advanced debugging functions</p> 

\*This document describes about On-chip Debuggers.

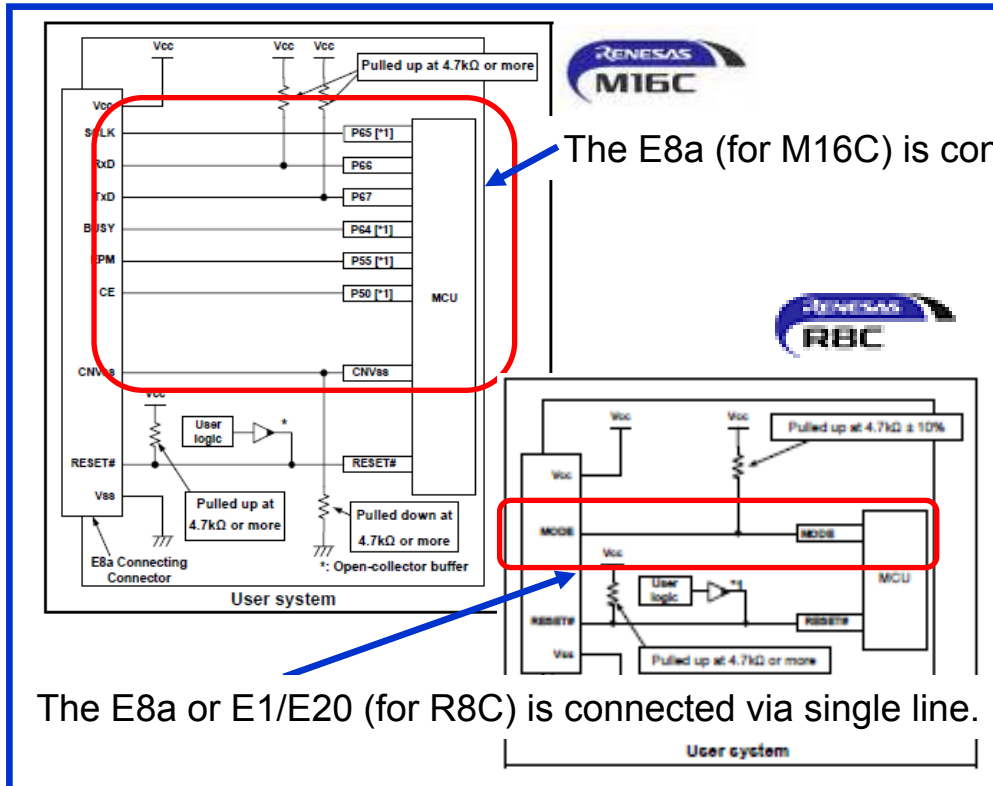
## 2. Differences between the Target Interfaces (for OCD)



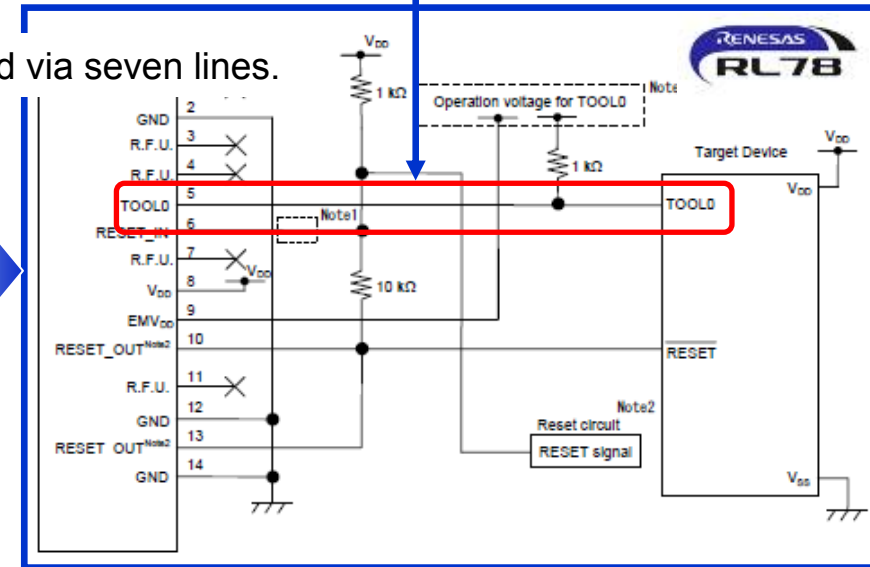
The RL78 and E1 emulator are connectable via the 14-pin connectors listed below.

	Type Number	Manufacturer	Specification
14-pin connector	7614-8002	Sumitomo 3M Limited	14-pin straight type (Japan)
	2514-8002	3M Limited	14-pin straight type (other countries)

Although these are the same connectors as for the E8a and E1/E20 for R8C and M16C products, **the communications interface is different.**

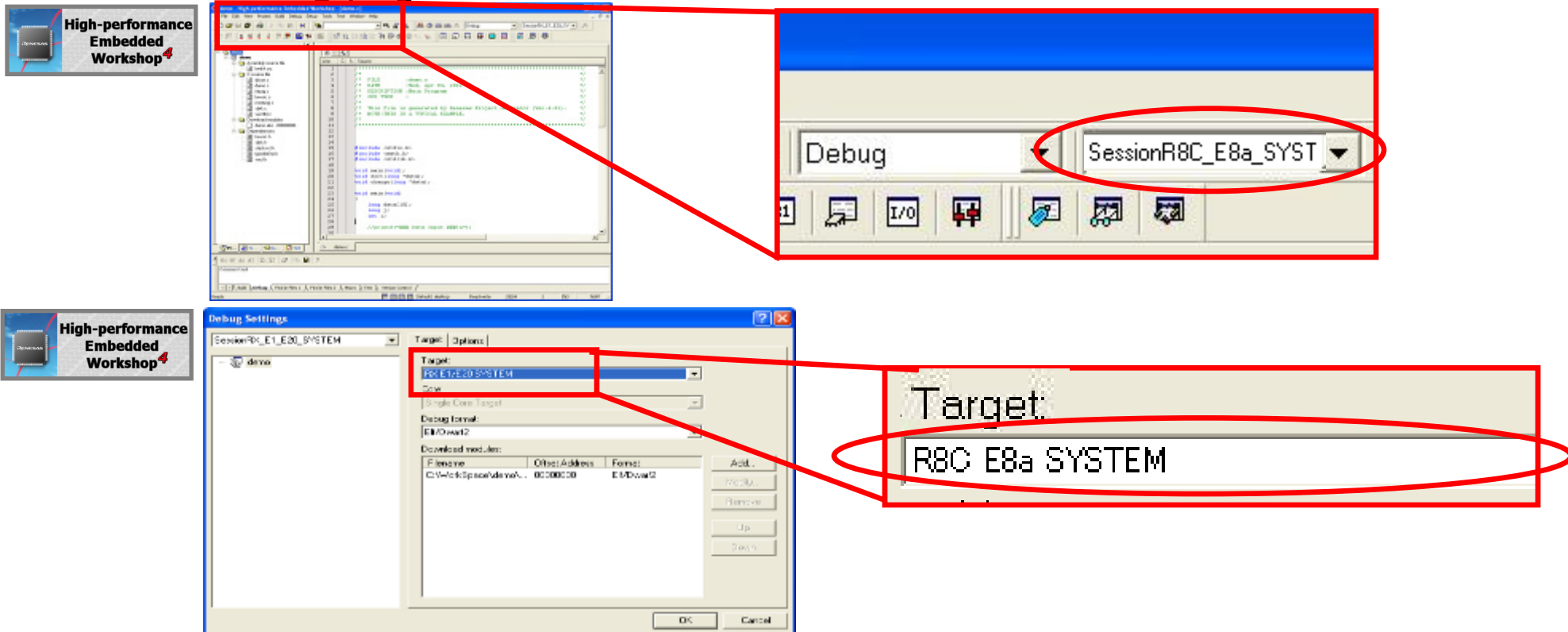


The E1 (for RL78) communicates via a **single line.**



### 3.Changing the Debugger

The HEW allowed users to select a debugger (E8a,E1/E20 emulator or simulator) in the process of changing the debug session or the target shown in the [Debug Settings] dialog box. The CubeSuite+, on the other hand, allows users to select a debugger on the project tree. The procedure to change the debugger is described on the following pages.



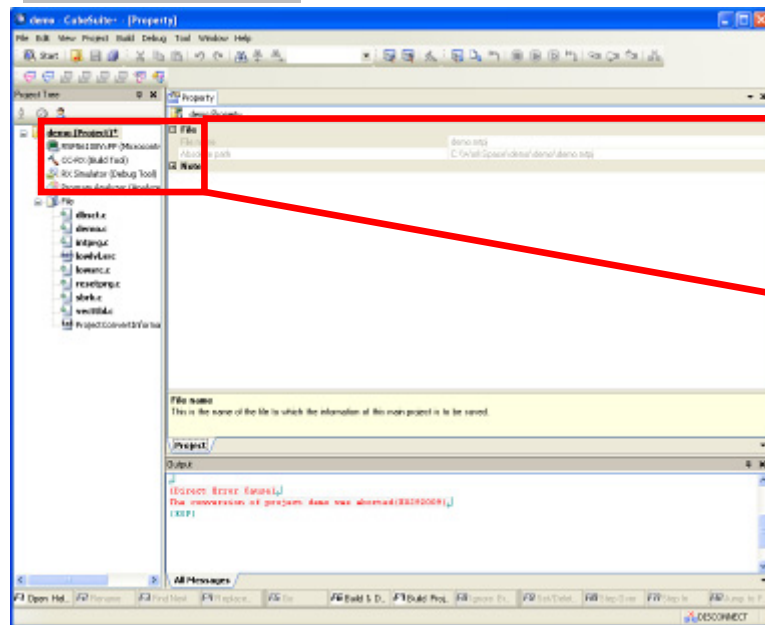
Selecting a Debugger (E8a,E1/E20 Emulator or Simulator) in the HEW

### 3. Changing the Debugger

(1) The debug tool name (debug tool) on the project tree panel indicates the currently selected debugger.

The following example shows that the E1 Emulator is selected:

CubeSuite+



Selected debugger

- CA78K0M (Build Tool)
- RL78 E1 (Serial) (Debug Tool)
- Program Analyzer (Analyze Tool)

### 3. Changing the Debugger

(2) To change the debugger, right-click the debug tool name (debug tool) to open a pop-up menu. Select [Using Debug Tool] from the pop-up menu to select the debug tool you want to use.

The screenshot shows the CubeSuite+ interface. A red box highlights the 'Using Debug Tool' option in the context menu for 'RL78 E1(Serial) (Debug Tool)'. A yellow callout points to the 'Using Debug Tool' option, and another yellow callout points to the 'RL78 IECUBF' option in the sub-menu.

Right-click

Select the debug tool you want to use.

Using Debug Tool

RL78 IECUBF

RL78 E1(Serial)

RL78 E20(Serial)



## 4. Entering an ID Code

Both the R8C/M16C and RL78 require entering of an ID code.

However, there are some differences regarding the setting and authentication of the ID code and the action that is taken if the ID code does not match.

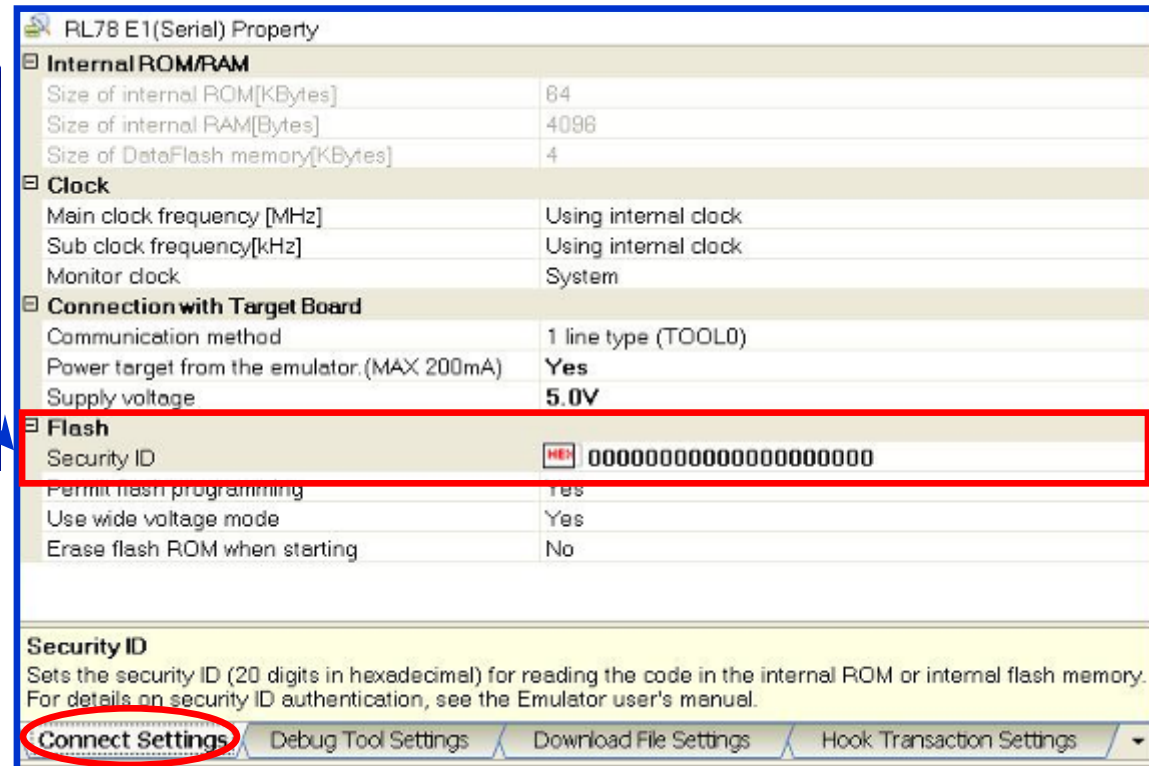
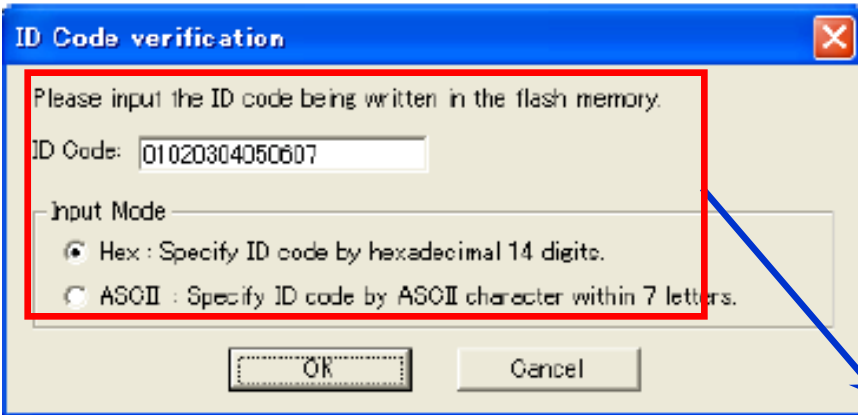
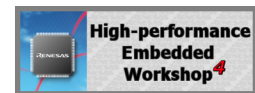
	ID code size	Address of the ID Code	Setting the ID Code	Authenticating the ID Code	Action Taken When the ID Code Does Not Match	Valid for the On-board Programmer?
R8C M16C	7 bytes	0xFFDF, 0xFFE3, 0xFFEB, 0xFFEF, 0xFFF3, 0xFFF7, 0xFFFB	Embed the code in the user program when building.	When all ID is FFh, It is authenticated automatically at a debugger start-up, and when other, enter ID into a dialog.	A debugger is not started ( the contents of the flash memory are held ).	Yes
RL78	10 bytes	0xC4 to 0xCD	Embed the code in the user program when building.	Enter an ID code for the debugger in advance.	Depends on the setting of the on-chip debugging option byte*	No (only valid during debugging)

For details, see *E1/E20 Emulator Additional Document for User's Manual (Notes on Connecting RL78)*.

## 4. Entering an ID Code

In the HEW, the [ID Code verification] dialog box opens at startup if an ID code has been written in the MCU. In CubeSuite+, on the other hand, an ID code must be set on the [Property] panel **before the emulator is started up**.

Set an ID code by referring to the following figures:

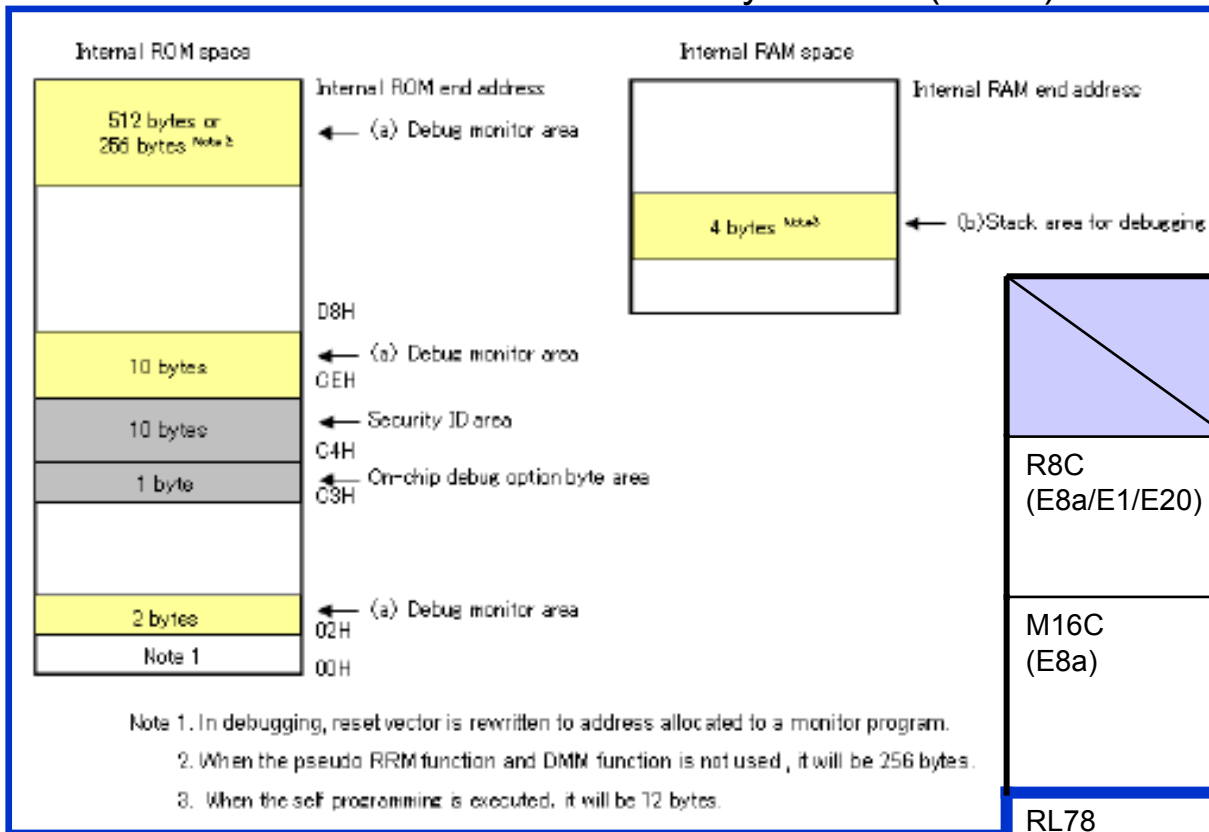


Example:  
[ID Code] dialog box of the E8a for the R8C

# 5. Securing Resources

When in use with the RL78, OCD takes up some user resources. These areas should not be used by the user program so keep them reserved (e.g. by using the build tool).

Reserved areas to be used by E1/E20 (RL78)\*



For how to allocate resources through CubeSuite+, see the next page.

Comparison with the R8C and M16C

	Does OCD Take up Resources?	Size	What Should be Used to Allocate the User Resources?
R8C (E8a/E1/E20)	Yes (depending on the MCU)	Up to 2 Kbytes of ROM, 8 bytes of stack, and some vectors	Build tool (setting up the linkage editor to avoid the areas to be taken up by OCD)
M16C (E8a)	Yes (depending on the MCU)	Up to 2 Kbytes of ROM, 128 bytes of RAM, 14bytes of stack, and some vectors	
RL78 (E1/E20)	Yes	See the figure at left.	Build tool (through the GUI of CubeSuite+)

\*For details, see *E1/E20 Emulator Additional Document for User's Manual (Notes on Connecting RL78)*.

## 5. Securing Resources

The address of the area for monitoring by the debugger can be specified on the [Link Options] sheet of the [Property] panel of the build tool.



The screenshot displays the 'CA78K0R Property' panel in CubeSuite+. The left sidebar shows a project tree with 'CA78K0R (Build Tool)' selected. The main panel shows the 'Device' section with the following settings:

Property	Value
Use on-chip debug	Yes(-go)
Option byte values for OCD	HEX 84
Debug monitor area start address	HEX FE00
Debug monitor area size[byte]	512
Set user option byte	Yes(-gb)
User option byte value	HEX EFFFEB
Specify mirror area	MAA=0(-mi0)
Set flash start address	No
Boot area load module file name	
Control allocation to self RAM area	No

The 'Link Opti...' tab is highlighted with a red circle at the bottom of the panel.

## 6. Setting the On-Chip Debugging Option Byte

The on-chip debugging option byte can be set on the [Link Options] sheet of the [Property] panel of the build tool.

CubeSuite+

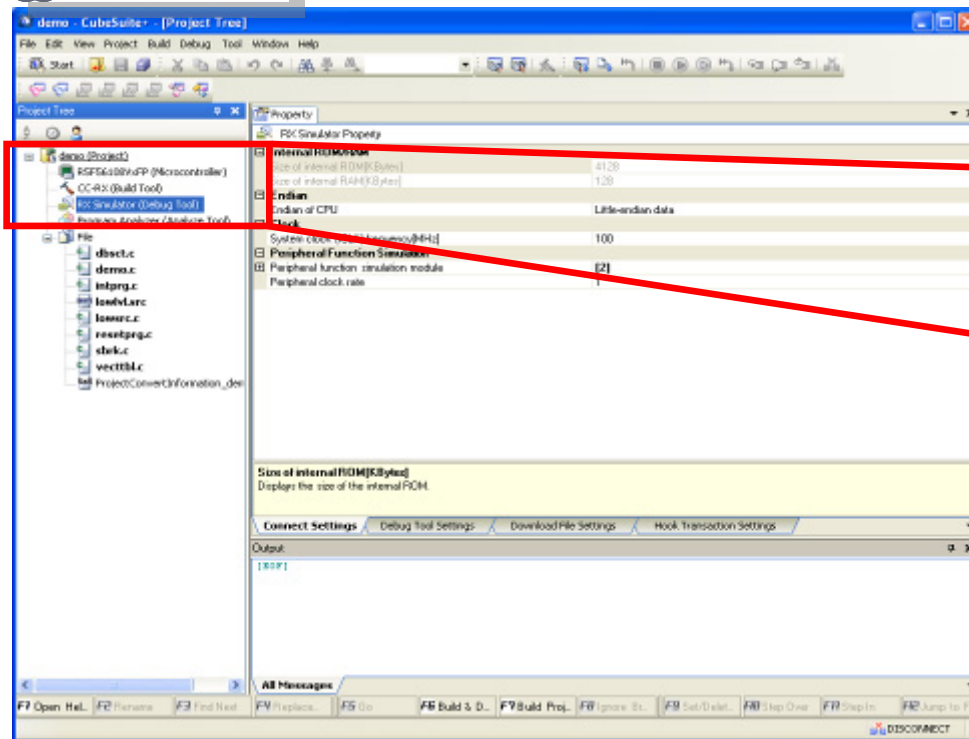
The screenshot shows the CubeSuite+ interface. On the left is a project tree with 'CA78K0R (Build Tool)' selected. On the right is the 'CA78K0R Property' panel. The 'Device' section is expanded, and the 'Option byte values for OCD' field is highlighted with a red box, showing a value of 'HEX 84'. The 'Link Options' tab at the bottom is also highlighted with a red circle.

Property	Value
<b>Debug Information</b>	
Add debug information	Yes
<b>Input File</b>	
Using link directive file	r_lk.dr
<b>Output File</b>	
Output folder	%BuildModeName%
Output file name	%ProjectName%.lmf
Force linking against error	No
<b>Library</b>	
<b>Device</b>	
Use on-chip debug	Yes(-oo)
Option byte values for OCD	HEX 84
Debug monitor area start address	HEX FE00
Debug monitor area size[byte]	512
Set user option byte	Yes(-gb)
User option byte value	HEX EFFFEB
Specify mirror area	MAA=0(-mi0)
Set flash start address	No
Boot area load module file name	
Control allocation to self RAM area	No
<b>Message</b>	
<b>Stack</b>	
<b>Library</b>	

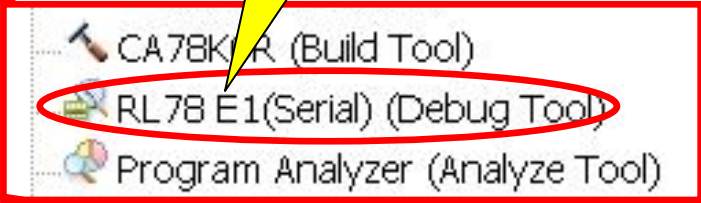
# 7. Where Do We Make Settings when Connecting an Emulator?

In the HEW, the [Emulator Setting] dialog boxes open to make settings when connecting an emulator. In the CubeSuite+, on the other hand, you need to make settings on the [Property] panel **before connecting an emulator** by taking the following procedure.

Double-click the debug tool name (debug tool) on the [Project Tree] panel to open the Properties window of the debug tool.



Double-click



# 7. Where Do We Make Settings when Connecting an Emulator?

In the case of HEW, settings required for connection are made in the [Emulator Setting] dialog box during the process of connecting the emulator. In the case of CubeSuite+, on the other hand, these settings must be made in the [Property] panel of the debugger before connecting the emulator.

(1) The [Emulator Setting] dialog box of HEW corresponds to the [Connect Settings] tab of CubeSuite+.

The image shows two screenshots side-by-side. The left screenshot is the 'Emulator Setting' dialog box from High-performance Embedded Workshop (HEW). It has three tabs: 'Emulator mode' (circled in red), 'Firmware Location', and 'Communication Baud Rate'. The 'Emulator mode' tab is active, showing 'MCU Group' set to 'R8C/35A Group' and 'Device' set to 'R5F21356A'. Under 'Mode', 'Erase Flash and Connect' is selected. A 'Power supply' section at the bottom has 'Power Target from Emulator. (MAX 300mA)' checked, with '3.3 V' and '5.0 V' radio buttons. The right screenshot is the 'CubeSuite+' interface showing the 'RL78 E1 (Serial) Property' panel. The 'Connection with Target Board' section has 'Power target from the emulator. (MAX 200mA)' set to 'Yes' and 'Supply voltage' set to '5.0V', both circled in red. A text box to the right of these settings says 'Some functions including power supply are correspondent.' The 'Connect Settings' tab at the bottom is also circled in red. Blue arrows point from the HEW dialog to the CubeSuite+ settings, indicating their correspondence.

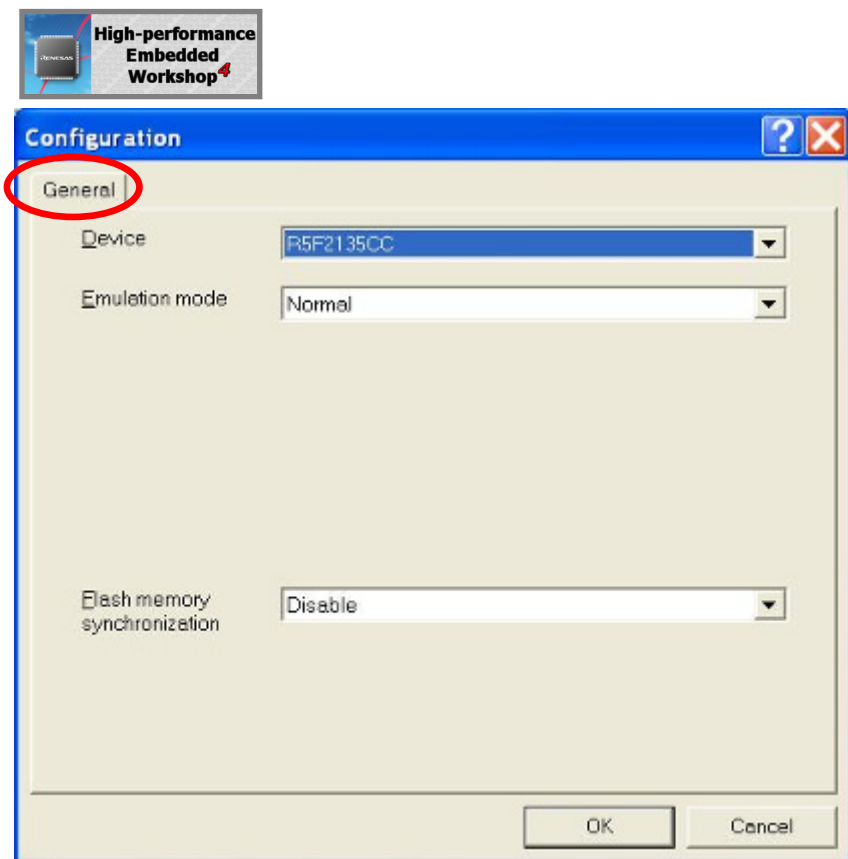
HEW Emulator Setting	CubeSuite+ Connect Settings
MCU Group: R8C/35A Group	Internal ROM/RAM: Size of internal ROM [KBytes] 64, Size of internal RAM [Bytes] 4096, Size of DataFlash memory [KBytes] 4
Device: R5F21356A	Clock: Main clock frequency [MHz] Using internal clock, Sub clock frequency [kHz] Using internal clock, Monitor clock System
Mode: Erase Flash and Connect	Connection with Target Board: Communication method 1 line type (TDO/L0)
Power supply: Power Target from Emulator. (MAX 300mA) checked, 5.0 V selected	Power target from the emulator. (MAX 200mA) Yes, Supply voltage 5.0V
	Flash: Security ID 0000000000000000, Permit flash programming Yes, Use wide voltage mode Yes, Erase flash ROM when starting No
	Security ID: Sets the security ID (20 digits in hexadecimal) for reading the code in the internal ROM or internal flash memory.

Example:  
[Emulator Setting] dialog box of the E8a for the R8C

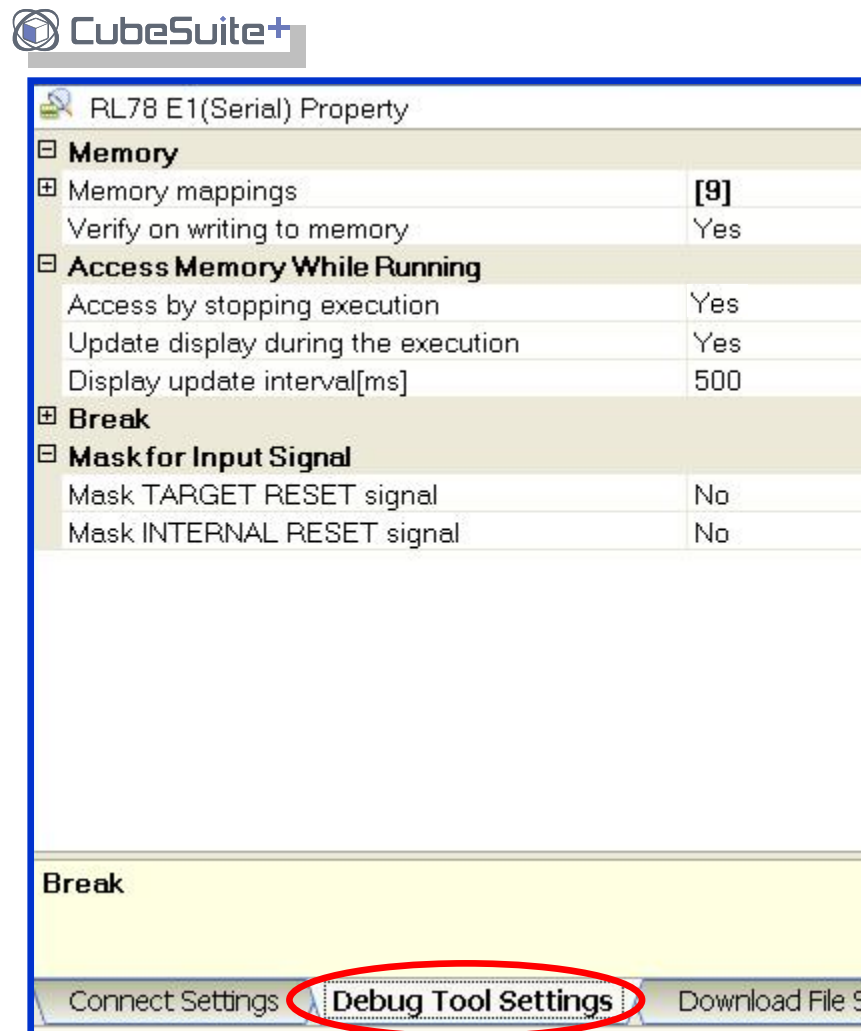
On CubeSuite+, the device needs to be selected during the process of creating a project.

# 7. Where Do We Make Settings when Connecting an Emulator?

(2) [Configuration] dialog box of the HEW corresponds to the [Debug Tool Settings] tab of the CubeSuite+.



Example:  
[Configuration] dialog box of the E8a for the R8C





## 8. Connecting an Emulator

Select [Debug] -> [Connect to Debug Tool] from the CubeSuite+ menu to establish connection to the selected emulator (debug tool).

Upon completion of the connection, the debug tool name appears on the status bar at the bottom right of the window.

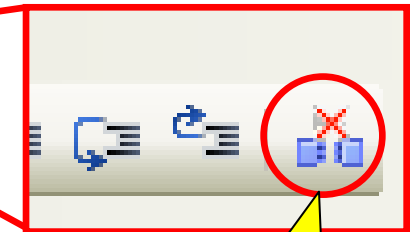
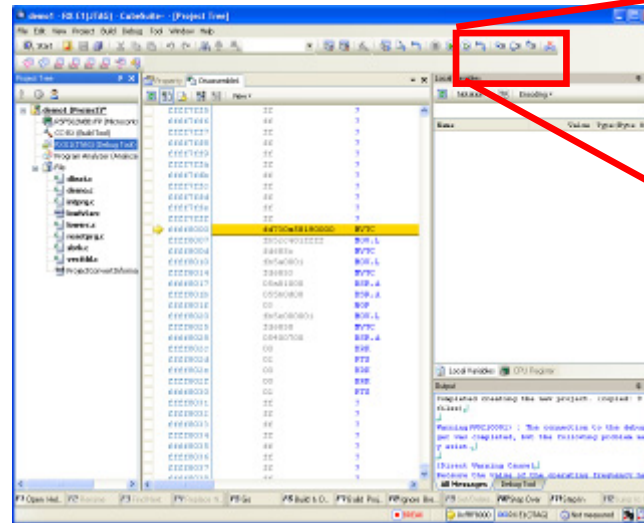
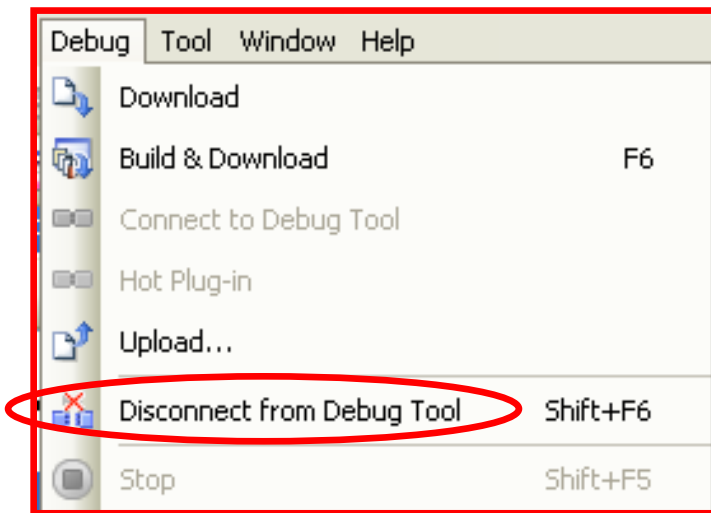


The screenshot shows the CubeSuite+ interface. The 'Debug' menu is open, and 'Connect to Debug Tool' is highlighted. A yellow callout bubble points to this option with the text 'Select [Connect to Debug Tool]'. The status bar at the bottom right shows 'DISCONNECT'. A red box highlights the 'Connect to Debug Tool' menu item and the status bar. A second red box highlights the 'Connect to Debug Tool' menu item and the status bar, with a yellow callout bubble pointing to the status bar and the text 'Before connection'. A third red box highlights the status bar after connection, showing 'RL78 E1 (Serial)' and a yellow callout bubble pointing to it with the text 'After connection'.

Note: If an ID code has been written in the MCU, set an ID code in advance according to “2. Entering an ID Code.”


## 9. Disconnecting the Emulator


To disconnect the emulator, select [Disconnect from Debug Tool] from the menu or click the  button on the debug toolbar.



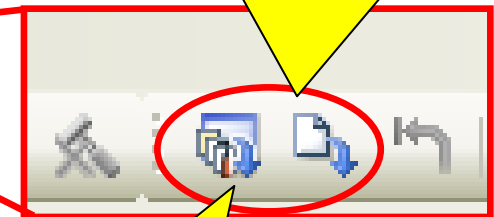
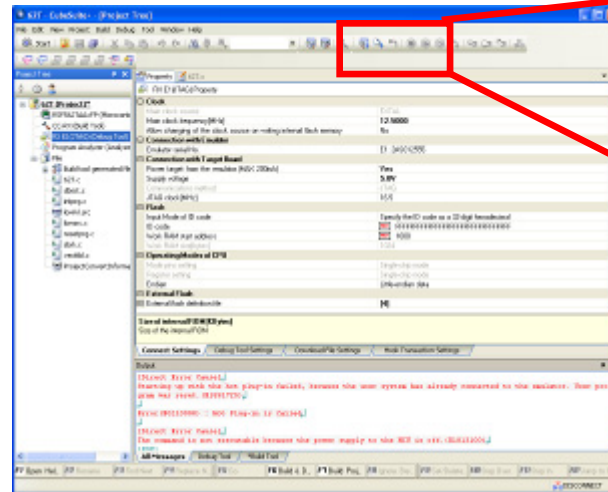
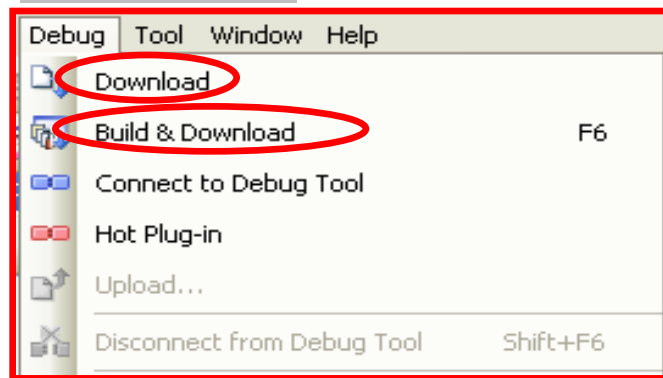
[Disconnect from Debug Tool] button

# 10. Downloading a Program

Selecting [Debug] -> [Download] from the menu or clicking the  button on the debug toolbar starts downloading specified files.

Selecting [Debug] -> [Build & Download] from the menu or clicking the  button on the debug toolbar builds a project and then starts downloading the specified files.

If no debug tool is connected, CubeSuite+ connect debug tool automatically before downloading.



# 11. Registering Additional Download Files

Add download files in the [Download File Settings] sheet on the [Property] panel.

(1) Select [Download files] and click [...] button on the right.

(2) The [Download Files] dialog box opens. Click the [Add] button.

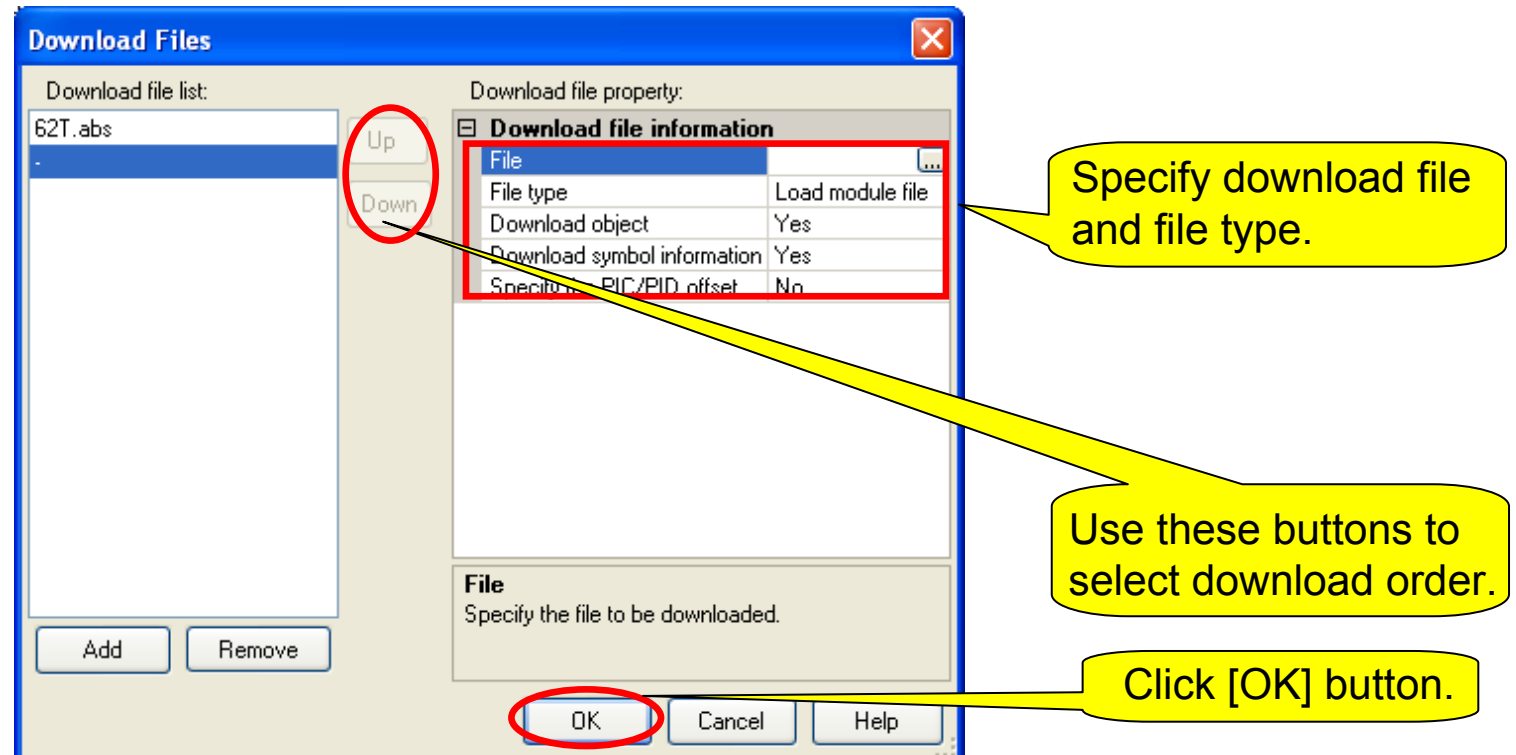
Select [Download files] and click [...].

Click [Add] button.

The screenshot shows the CubeSuite+ interface. On the left, the 'Property' panel is open to the '62T.c' project, showing the 'Download File Settings' tab. The 'Download files' section is expanded, showing a list with one entry: 'Debug\62T.abs'. A red circle highlights the [...] button to the right of this entry. A yellow callout bubble points to this button with the text 'Select [Download files] and click [...]'. Below the settings panel, the 'Download File Settings' tab is circled in red. On the right, the 'Download Files' dialog box is open. It has a 'Download file list' containing '62T.abs'. Below the list are 'Up' and 'Down' buttons. At the bottom of the dialog, the 'Add' button is circled in red, with a yellow callout bubble pointing to it containing the text 'Click [Add] button.'. The dialog also has 'Remove', 'OK', 'Cancel', and 'Help' buttons. The 'Download file property' section on the right shows details for the selected file: 'File: Debug\62T.abs', 'File type: Load module file', 'Download object: Yes', 'Download symbol info: Yes', and 'Specify the PIC/PID: No'.

# 11. Registering Additional Download Files

(3) Specify the file name and file type in the [Download file information] field and then click the [OK] button.



Note: When downloading is performed, all of the registered files are downloaded. To download only desired files, set [Download object] and [Download symbol information] to “Yes” in this window only for files you want to download.

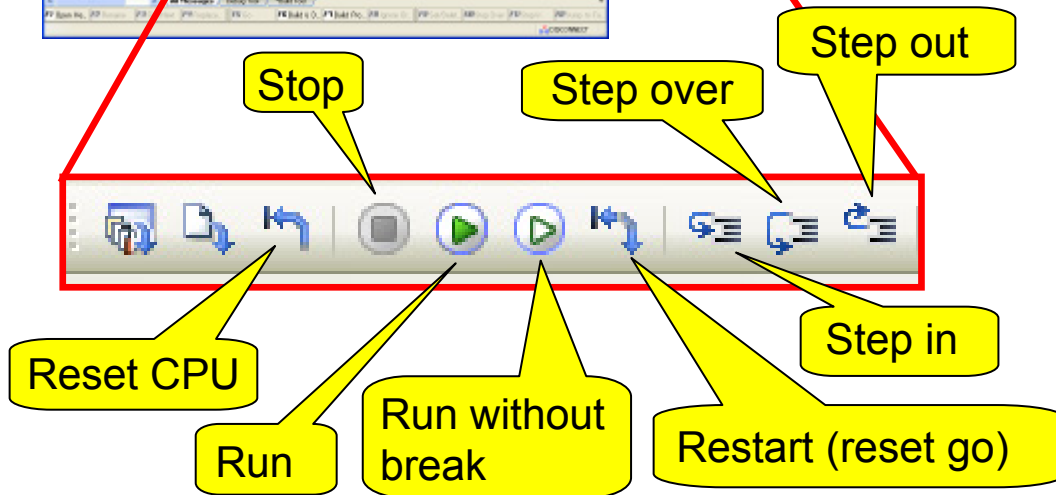
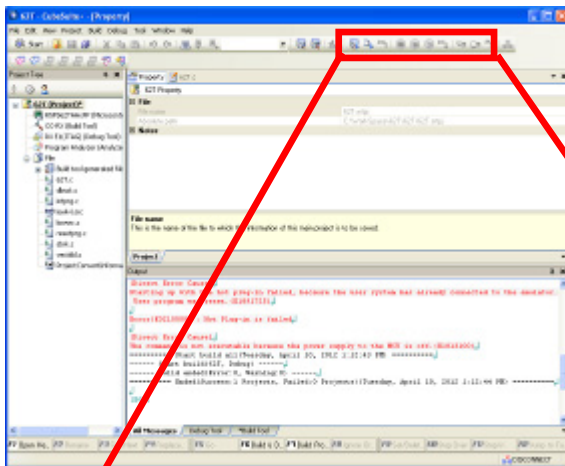
# 11. Registering Additional Download Files

(4) The available file formats (extensions) for the R8C/M16C and for the RL78 are not the same. For details, see the table below.

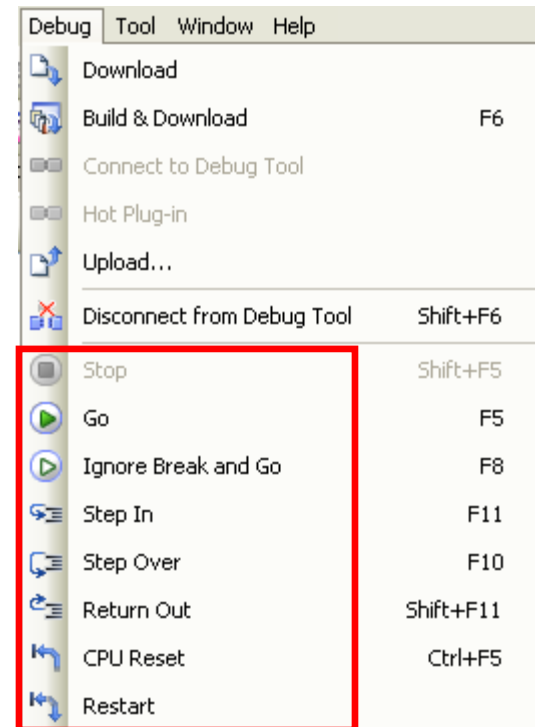
	Load Module Format	Hexadecimal File	Binary File
<b>R8C/M16C</b>	<b>*.x30 *.abs</b>	<b>*.mot *.hex</b>	<b>*.bin</b>
<b>RL78</b>	<b>*.lmf</b>	<b>*.hex</b>	<b>*.bin</b>
Purpose	To be downloaded for source-level debugging	To be used for writing by a ROM programmer, etc.	Data file

## 12. Starting/Stopping a Program

You can start or stop a program and reset the CPU from the menu or toolbar in the same way as the HEW (see below).



### CubeSuite+ menu



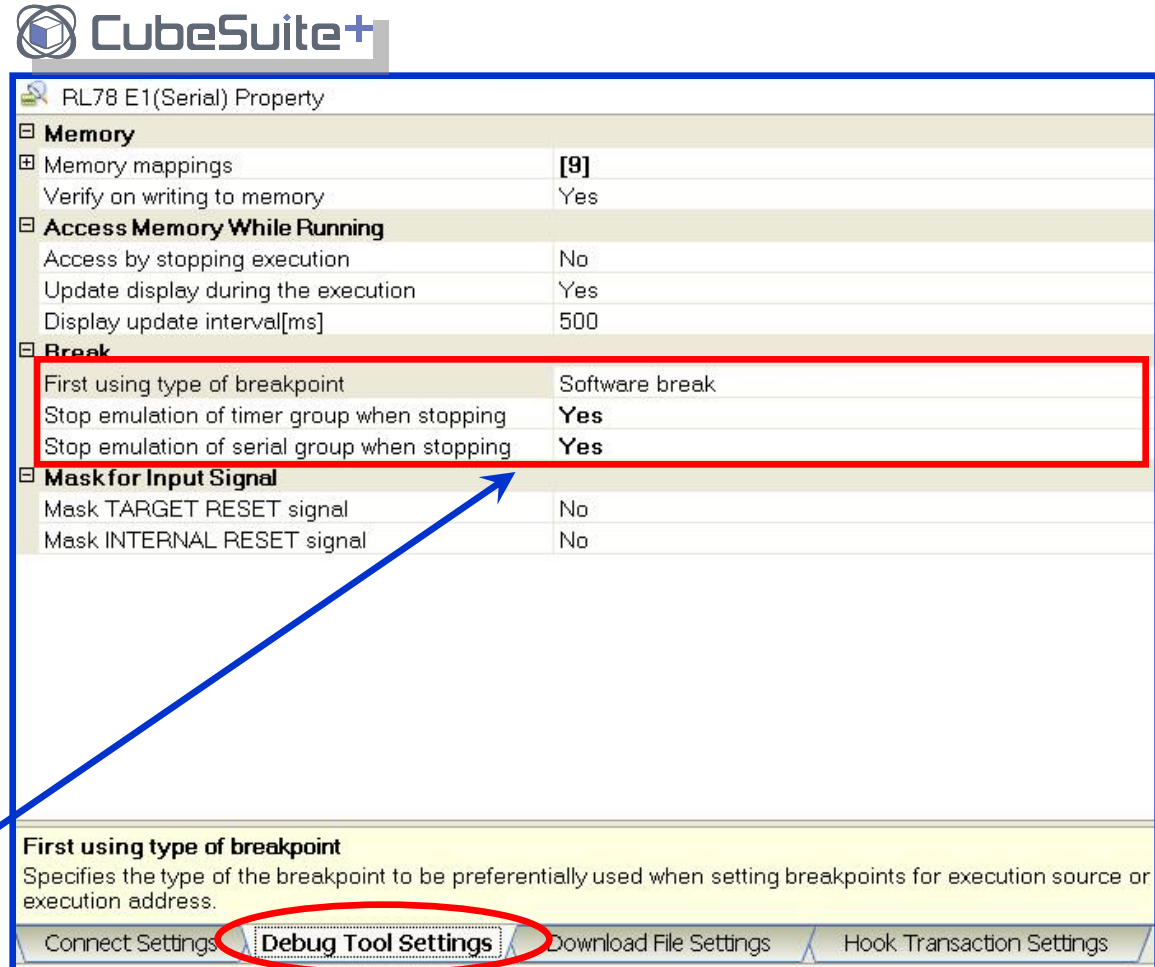
# 13. Difference in MCU Operation during a Break (Peripheral Break Function)

While timers and serial communication interfaces in the R8C/M16C continue to operate while CPU breaks in execution by the emulator, CubeSuite+ for the RL78 allows you to use the [Debug Tool Settings] sheet to select whether or not those modules are to be stopped while CPU breaks.



In the case of HEW for the R8C/M16C, you need to use the Start/Stop functions to create and embed code that will stop the peripheral modules if this is required. (R8C/5x series have the Peripheral Break Function)

Selectable on CubeSuite+ for the RL78





# 14. Viewing/Changing Memory Data and Variables While the Program Is Running

To view or change memory data and variables while the program is running in CubeSuite+, make settings on the [Property] panel by using the following procedure:

- (1) Open the [Debug Tool Settings] sheet on the [Property] panel of the debug tool.
- (2) Set [Access by stopping execution] in the [Access Memory While Running] field to [Yes]. Memory data and variables can be viewed while the program is running.



RL78 E1(Serial) Property		
<b>Memory</b>		
Memory mappings	[9]	
Verify on writing to memory	Yes	
<b>Access Memory While Running</b>		
Access by stopping execution	Yes	
Update display during the execution	Yes	
Display update interval[ms]	500	
<b>Break</b>		
First using type of breakpoint	Software break	
Stop emulation of timer group when stopping	Yes	
Stop emulation of serial group when stopping	Yes	
<b>Mask for Input Signal</b>		
Mask TARGET RESET signal	No	
Mask INTERNAL RESET signal	No	
<b>Access by stopping execution</b>		
Specifies whether to access a memory area by momentarily stopping area that cannot be accessed during execution....		
Connect Settings	<b>Debug Tool Settings</b>	Download File Settings

Change this to [Yes].

If [No] is selected, "\*\*\*" is displayed on the memory panel while the program is running.

# 15. Automatically Updating Memory Data and Variables While the Program Is Running

To automatically update memory data and variables via CubeSuite+, make settings on the [Property] panel by using the following procedure:

- (1) Open the [Debug Tool Settings] sheet on the [Property] panel of the debug tool.
- (2) Set [Access by stopping execution] and [Update display during execution] in the [Access Memory While Running] field to [Yes].

Information displayed on the memory and watch panels is automatically updated while the program is running.

To change the update interval, modify the [Display update interval] value.

The image shows a screenshot of the CubeSuite+ interface. On the left, the 'RL78 E1(Serial) Property' window is open to the 'Debug Tool Settings' tab. The 'Access Memory While Running' section is highlighted with a red box, and its settings are as follows:

Setting	Value
Access by stopping execution	Yes
Update display during the execution	Yes
Display update interval[ms]	500

A yellow callout bubble points to the 'Access by stopping execution' and 'Update display during the execution' settings, stating: "Change two items to [Yes].". Another yellow callout bubble points to the 'Display update interval[ms]' setting, stating: "Set update interval.". The 'Debug Tool Settings' tab is circled in red. On the right, the 'Watch1' window is open, showing a table with the following data:

Watch	Value	Type (Byt)
z	24	int (4)

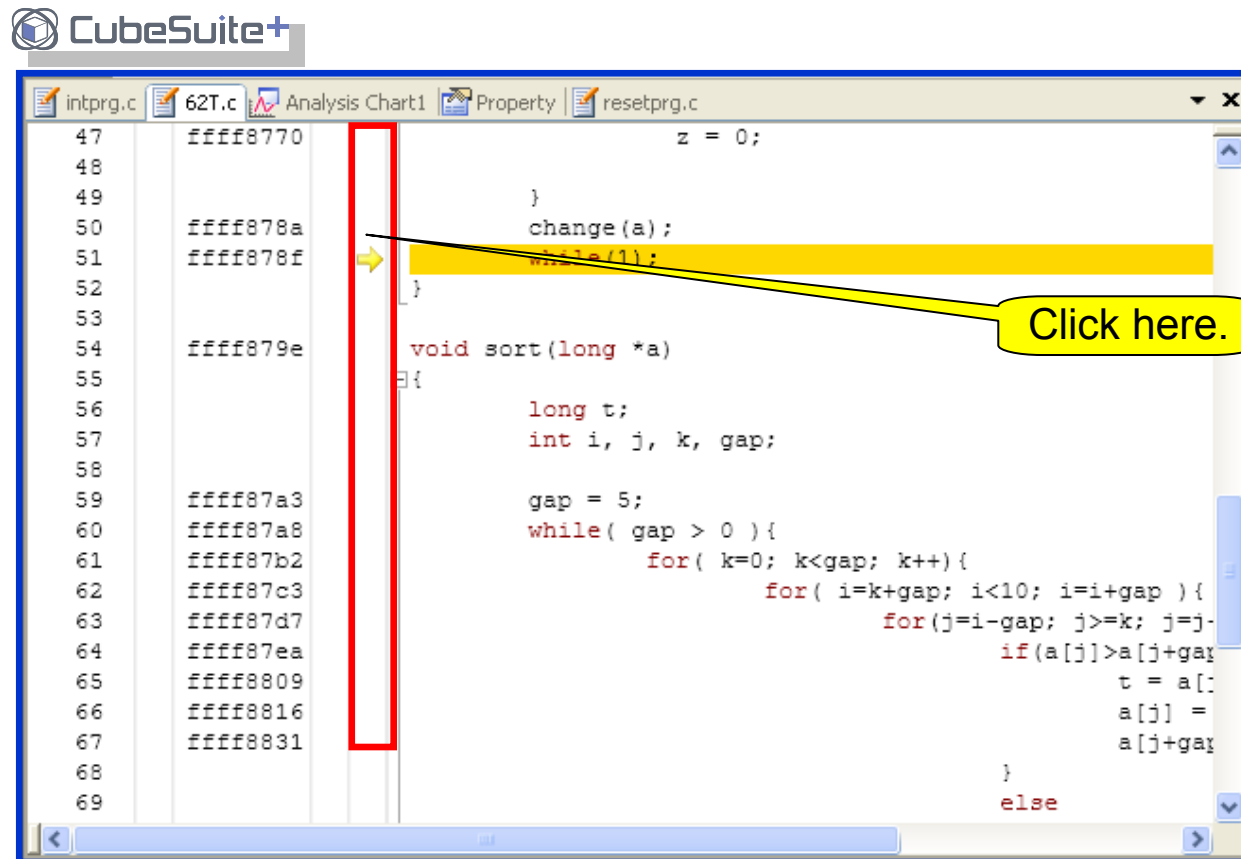
A yellow callout bubble points to the 'Watch1' window, stating: "Variables registered in the watch panel and memory data in the memory panel are automatically updated while the program is running.".

## 16. Setting Breakpoints

(1) You can set breakpoints in the main area (enclosed by a red line in the figure below) on the editor panel of CubeSuite+.

Set break points: Single-clicking a line with an address.

Delete break points: Single-clicking a line for which a breakpoint has been set.



## 16. Setting Breakpoints

(2) Select a breakpoint type (software break or hardware break) for [First using type of breakpoint] in the [Debug Tool Settings] sheet on the [Property] panel. (Software break is selected in the example below.)



The screenshot shows the 'RL78 E1 (Serial) Property' dialog box. The 'Break' section is highlighted with a red box, and the 'First using type of breakpoint' is set to 'Software break'. A yellow callout bubble points to this setting with the text 'Set a breakpoint type to be preferentially used.' The 'Debug Tool Settings' tab is selected at the bottom.

Memory	
Memory mappings	[9]
Verify on writing to memory	Yes

Access Memory While Running	
Access by stopping execution	No
Update display during the execution	Yes
Display update interval[ms]	500

Break	
First using type of breakpoint	Software break
Stop emulation of timer group when stopping	Yes
Stop emulation of serial group when stopping	Yes

Mask for Input Signal	
Mask TARGET RESET signal	No
Mask INTERNAL RESET signal	No

First using type of breakpoint  
Specifies the type of the breakpoint to be preferentially used when setting br execution address.

Connect Settings **Debug Tool Settings** Download File Settings

Set a breakpoint type to be preferentially used.

(3) If the number of breakpoints of the selected type exceeds the limit, the other type of breakpoints are used.

Event marks indicate the types of breakpoints.



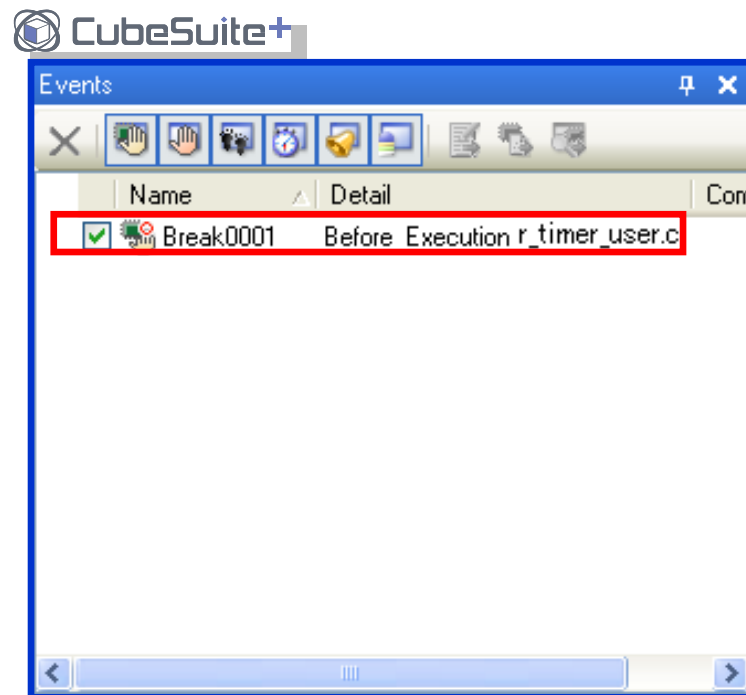
: Software break



: Hardware break

## 16. Setting Breakpoints

(4) You can check the breakpoint setting on the [Events] panel. Select [View] -> [Event] from the CubeSuite+ menu to open the [Events] panel. Unnecessary breakpoints can be deleted or disabled on the [Events] panel.



## 17. Causing a Break on Access to a Variable

You can use the watch or editor panel to make a setting to cause a break on access to a specific variable.

(1) On the watch or editor panel, right-click the variable that you want to set a break when it is accessed.

(2) Select [Access Break] (or [Break Settings] on the editor panel) and select [Set Read Combination Break to], [Set Write Combination Break to], or [Set R/W Combination Break to].

The image shows a screenshot of the CubeSuite+ IDE. On the left, the 'Watch' window displays a variable 'z' with a value of 2146513710 and type 'int (4)'. The variable 'z' is circled in red. In the center, a code editor shows the following code:

```
15 #include <math.h>
16 #include <stdlib.h>
17
18 int z = 0;
19
```

The line 'int z = 0;' is circled in red. On the right, a yellow callout bubble points to the code with the text: "Right-click the variable to open a pop-up menu." Below the code editor, a context menu is open, showing the following options:




- Set Read Combination Break to
- Set Write Combination Break to
- Set R/W Combination Break to
- Access Break
- Timer Settings

The 'Access Break' option is circled in red. A yellow callout bubble points to it with the text: "Select [Access Break]." Another yellow callout bubble points to the 'Set Read Combination Break to' option with the text: "Select a break condition (read, write, or read/write)." A red arrow points from the 'z' in the Watch window to the 'Access Break' option in the context menu.

## 17. Causing a Break on Access to a Variable

(3) Enter a value to set a data condition (or leave the box blank if no data condition is needed).

CubeSuite+

	Set Read Combination Break to	10	Access Break ▶
	Set Write Combination Break to		Trace Output ▶
	Set R/W Combination Break to		Timer Settings ▶

Enter a data condition if necessary.

Note: Enter a decimal number here. When entering a hexadecimal number, add “0x” to the head (e.g. 0xAA).

## 18. Filling Memory

Memory can be filled (batch change) by using the [Memory Initialize] dialog box.

(1) Right-click on the [Memory] panel to open a pop-up menu, and select [Fill] from the pop-up menu.

(2) The [Memory Initialize] dialog box opens. Enter addresses (start address and end address) and initialization data, and then click the [OK] button.

The image shows the CubeSuite+ interface. On the left, the 'Memory1' panel displays a memory dump with columns for address and data. A right-click context menu is open over the memory dump, with the 'Fill...' option highlighted by a red box. A yellow callout bubble points to this menu with the text: 'Right-click here to open a pop-up menu and select [Fill]'. On the right, the 'Memory Initialize' dialog box is shown. It has three input fields: 'Start address/symbol:' containing '0x0000', 'End address/symbol:' containing '0xffff', and 'Initialize data:' containing 'HEX 0xaa'. Each of these fields is enclosed in a red box. A yellow callout bubble points to these fields with the text: 'Enter a start address, end address, and initialization data. This example shows filling 0x0000 to 0xffff with 0xaa'. At the bottom of the dialog, the 'OK' button is circled in red.

Address	Hex Data
00000000	0E AA FF ED 2E 3E 73 BF 7A 95 2F 61
00000010	94 2F 1C B3 60
00000020	AA 4A DF AB 46
00000030	48 15 A1 E7 6C
00000040	70 04 E8 BA 4A
00000050	FA 26 B1 5E 09
00000060	47 66 95 6B CD
00000070	C8 AC 54 EB 6B
00000080	03 61 F7 E5 A9
00000090	C9 E5 FD 73 9D

Note: Enter decimal numbers here. When entering hexadecimal numbers, add “0x” to the head of each number.



## 19. Saving Memory Data

[Data Save] dialog box is used to save memory data.

Select [Debug] -> [Upload...] from the menu.

The [Data Save] dialog box opens. Specify the file name, type, and range of memory data you want to save, and then click the [Save] button.



The screenshot shows the 'Data Save - Upload' dialog box. It has a blue title bar with a close button. The dialog contains the following fields and controls:

- File Name:** A text field containing 'C:\Documents and Settings\toolgi.RENESAS-L8ELZKPM\My Docum' with a browse button to its right.
- File Type:** A dropdown menu currently set to 'Motorola S-format (\*.mot)'.
- Save Range Address/Symbol:** Two dropdown menus. The first contains '0xffff8000' and the second contains '0xffffffff'.
- Buttons:** 'Save', 'Cancel', and 'Help' buttons at the bottom. The 'Save' button is circled in red.

Enter the name of a file to be saved.

Specify a file type (Intel Hex, Motorola S, or binary).

Specify a memory range.

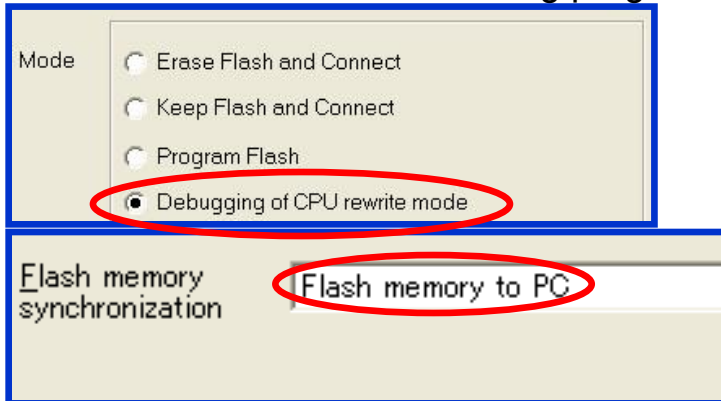
Note: Enter decimal numbers here. When entering hexadecimal numbers, add "0x" to the head of each number.

## 20. Flash Self-Programming

The RL78 supports a self-programming feature for the rewriting of data in flash memory by user programs. This is accomplished for user applications by using the self-programming library for the RL78.

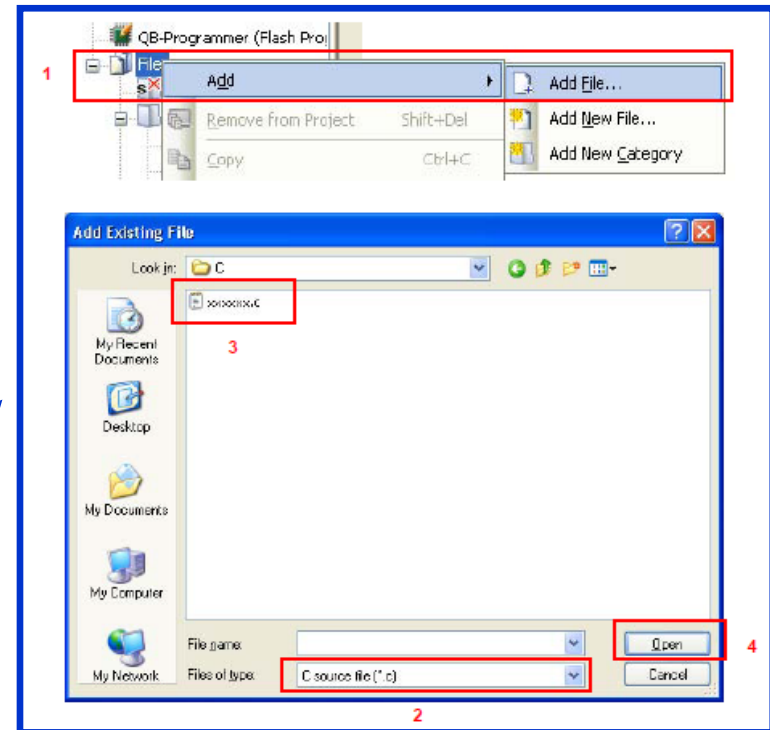


In the case of HEW for the R8C/M16C, you essentially need to place the debugger in a special mode and create a flash-rewriting program.



In the case of CubeSuite+ for the RL78, on the other hand, you need to install the self-programming library (provided for free) in the project.

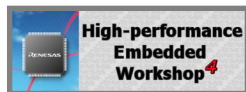
For details on how to install the library, visit the following URL: [http://www.renesas.com/products/tools/flash\\_prom\\_programming/flash\\_libraries/index.jsp](http://www.renesas.com/products/tools/flash_prom_programming/flash_libraries/index.jsp)



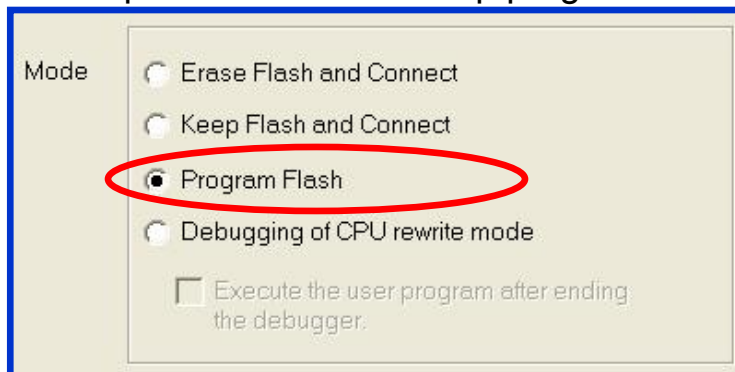
To add the flash self-programming file, right-click on **[File]** in **[Project Tree]** on CubeSuite+ and select **[Add -> Add File]** from the list, then click on the **[Files of type]** pull-down menu and select the file type.

# 21. How to program to check Operation on the Stand-Alone MCU

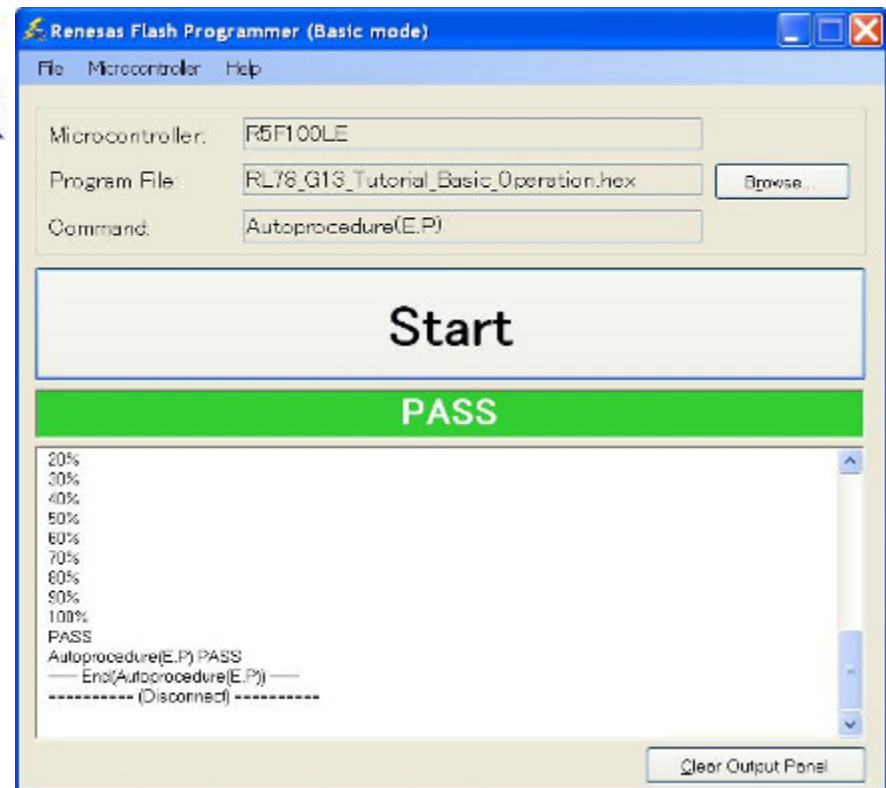
If you wish to check operation on the RL78 MCU as a stand-alone device after debugging, use the Renesas Flash Programmer (flash programming software) to program the data to the flash memory instead of using CubeSuite+.



In the case of HEW for the R8C/M16C, you need to select the mode that is suitable for debugging and also equivalent to the on-chip programmer.



In the case of the RL78, on the other hand, you need to use the Renesas Flash Programmer instead of CubeSuite+.



The Renesas Flash Programmer is software that is used to program to the flash memory of Renesas MCUs and is specialized for easy operation and functionality for programming.

## 22. Action Event (Printf Event)

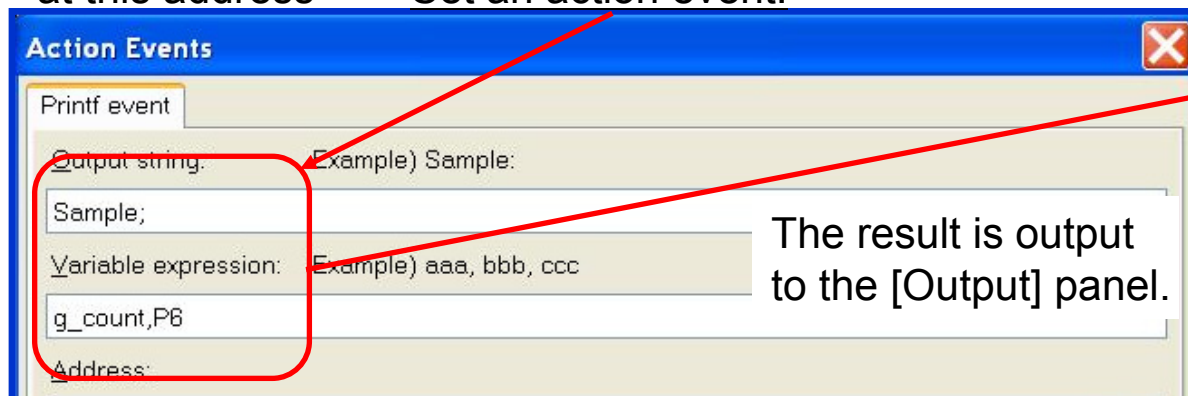
CubeSuite+ allows the setting of a Printf event as an action event.

A Printf event is used to stop the program momentarily at a specified address and make software execute the printf command. When a Printf event is set in the [Action Event] dialog box, the program stops before execution of the instruction at the address where the event is set, and CubeSuite+ outputs the value of the variables to the [Output] panel.



69	00196	outreg = inreg 1;
70	0019e	P6.2 = outreg;
71	001a4	P6.3 = inreg;
72	001ab	g_count++;
73		/* End user code. Do not edit comment generated here */
74	001ae	
75		

To insert code that is equivalent to `printf("Sample: g_count = %d, P6 = %d\n",g_count,P6)` at this address -> Set an action event.



Output

```
----- Start build(RL78_G13_Tutorial_Basic_Operation, DefaultBuild) -----
----- Build ended(Error:0, Warning:0) -----
===== Ended(Success:1 Projects, Failed:0 Projects) (Thursday, July 26, 2012 3:47:41 PM) =====
```

Sample;g\_count=1(0x0001), P6=0x8(0x08)

Sample;g\_count=2(0x0002), P6=0x4(0x04)

Sample;g\_count=3(0x0003), P6=0x8(0x08)

Sample;g\_count=4(0x0004), P6=0x4(0x04)

Sample;g\_count=5(0x0005), P6=0x8(0x08)

Sample;g\_count=6(0x0006), P6=0x4(0x04)

Sample;g\_count=7(0x0007), P6=0x8(0x08)

Sample;g\_count=8(0x0008), P6=0x4(0x04)

Sample;g\_count=9(0x0009), P6=0x8(0x08)

Stopped by Software Break.

[EOF]

All Messages \*Build Tool \*Debug Tool

## 23. Viewing Lists of Variables and Functions

CubeSuite+ can automatically display lists of variables and functions used in the project.

Select [View -> Program Analyzer] from the menu.



Variable Name	File Name	Attribute	Type
CGC	(No Definition)	far_const	-
g_count	r_timer_user.c	near	unsigned...
dval	r_timer_user.c	near	double
*Total*	(No Definition)	-	-
*Total*	r_timer_user.c	-	-

Function Name	File Name	Attribute	Return Type	Arguments
R_CGC_Create	r_cgc.c	far	void	void
R_CGC_Get_R...	r_cgc_user.c	far	void	void
R_PORT_Create	r_port.c	far	void	void
R_TAUD_Create	r_timer.c	far	void	void
R_TAUD_Chann...	r_timer.c	far	void	void
R_TAUD_Chann...	r_timer.c	far	void	void
R_TAUD_Chann...	r_timer_user.c	interrupt	void	void
main	r_main.c	far	void	void
R_Systeminit	r_systeminit.c	far	void	void
hdwinit	_systeminit.c	far	void	void
DI	(No Definition)	-	-	-
FI	(No Definition)	-	-	-

Clicking on a variable or function name opens the corresponding source file.

## 24. Analytical Graphs

CubeSuite+ has an analytical graphing feature, which shows line graphs indicating the relationships between the values of variables, registers, and addresses and time.

The graphs shown by CubeSuite+ during on-chip debugging of the RL78 are based on data acquired through the pseudo-RRM function.

The screenshot shows the 'Analysis Chart' window in CubeSuite+. The interface includes a top control bar with 'Analysis method' (Sampling, Reflect), 'Zoom' (Zoom1), and channel selection buttons (1-4). Below this is a 'Trigger information' box showing 'Trigger: Auto, ch1: 00[Rising] Position: 0'. The main area is a graph with two cursors, 'Cursor A' and 'Cursor B', and a 'Cursor information area' on the right. The bottom section is the 'Channel information area' showing channel settings for ch1 through ch8.

**Graph control area**

**Trigger information**

**Cursor information area**

**Channel information area**

**Cursor A**

**Cursor B**

Analysis method: Sampling Reflect Zoom: Zoom1

Trigger: Auto, ch1: 00[Rising] Position: 0

Cursor: X axis (Time) Y axis (Value)

Target Cursor

Time:	-	-	-	-
ch1:	-	-	-	-
ch2:	-	-	-	-
ch3:	-	-	-	-
ch4:	-	-	-	-
ch5:	-	-	-	-
ch6:	-	-	-	-
ch7:	-	-	-	-
ch8:	-	-	-	-

ch1:  g\_count Val/Div: 3.6

ch2:  dval Val/Div: 2.0

ch3:  (none) Val/Div: 25.5

ch4:  (none) Val/Div: 25.5

ch5: Val/Div: 3.6

ch6: Val/Div: 2.0

ch7: Val/Div: 25.5

ch8: Val/Div: 25.5

Variable Value Changing Chart Execution Time(Percentage) Chart

## 25. Debugging Functions of Emulators (OCD)

Debugging Function		RL78 (E1/E20)	R8C (E8a/E1/E20)	M16C (E8a)
Breaks	Software breaks	2000 points	255 points	255 points
	Hardware breaks	1 to 2 points shared between instruction-execution and access events*	2 to 10 points shared between instruction-execution and access events*	6 to 10 points shared between instruction-execution and access events*
	Forced breaks	Supported	Supported	Supported
Events	Number of event points	1 to 2 points shared between instruction-execution and access events*	1 to 2 points*	0 to 2 points*
	Usage of events	For hardware breaks only	For hardware breaks only	For hardware breaks only
Tracing		Branch trace*	Branch trace/Data trace*	Branch trace/Data trace*
Performance measurement	Measurement item	From the start to the end of execution	From the start to the end of execution	From the start to the end of execution
	Performance	Resolution: 100 $\mu$ s Measurement time: Up to 100 hours	E8a Resolution: 1 ms Note: A timer in the host machine is required as a resource. E1/E20 Resolution: 1 $\mu$ s	Resolution: 1 ms Note: A timer in the host machine is required as a resource.
Pseudo realtime RAM monitor (RRM)		Supported: the CPU is occupied during monitoring.	Supported: the CPU is occupied during monitoring. / Debug DMA	Supported: the CPU is occupied during monitoring. / Debug DMA
Dynamic memory modification (DMM)		Supported: the CPU is occupied during modification.	Supported: the CPU is occupied during modification./ Debug DMA	Supported: the CPU is occupied during modification. / Debug DMA
Hot plug-in		Not supported	Not supported	Not supported
Security		Authentication of 10-byte ID**	Authentication of 2- or 4-byte ID**	Authentication of 4-byte ID**
Number of pins taken up		1 (TOOL0)	1 (MODE)	1 or 2 or 7
Peripheral breaks		Supported	Not supported (R8C/5x: Supported)	Not supported

\* Varies with the MCU.

\*\* For details on differences in specifications of the ID code, refer to 4. *Entering an ID Code* in this document.



**Renesas Electronics Corporation**

©2012. Renesas Electronics Corporation, All rights reserved.