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

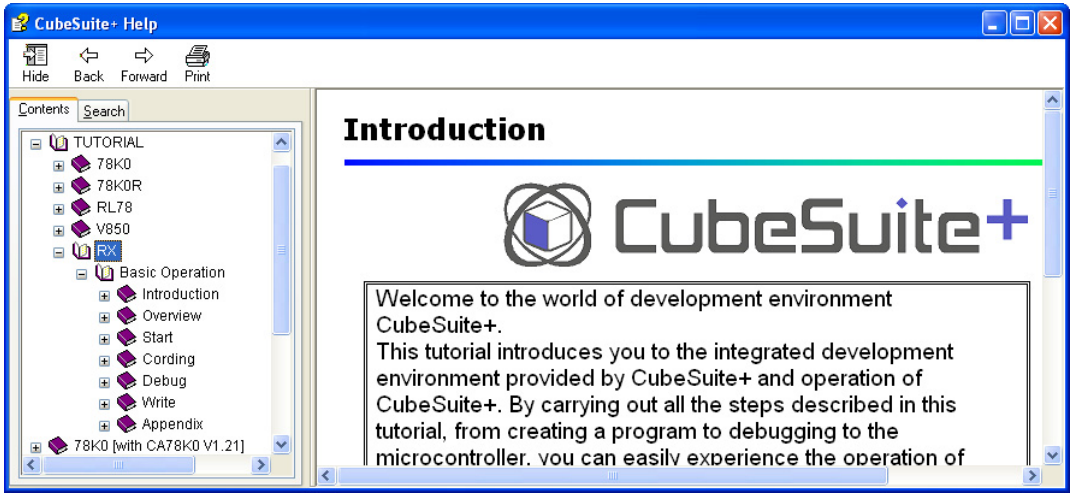
00000-A

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,.*

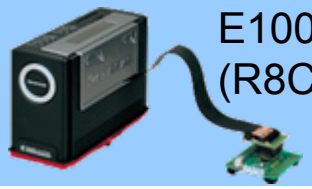
Also refer to the tutorial guide provided by CubeSuite+ for how to use tools.
The tutorial guide is available by selecting [Help] -> [Tutorial] from the CubeSuite+ menu.



Tutorial Guide

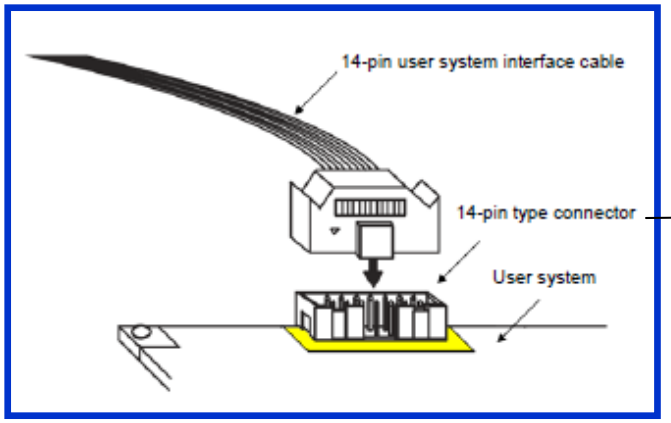
Contents

1. Integrated Development Environment and Emulators

	R8C,M16C	RL78
Integrated Development Environment (IDE)	High-performance Embedded Workshop 4 High-performance Embedded Workshop	CubeSuite+ A new integrated development environment that supports 8- to 32-bit MCUs from Renesas
On-chip Debuggers (OCD)	 E8a(R8C,M16C) E1(R8C)	Provides basic debugging functions at a low price 
Full-spec. Emulators*	CPE (R8C,M16C)   E100 (R8C,M16C)	High-performance full-spec. emulators with more advanced debugging functions 

*This document describes about On-chip Debuggers.

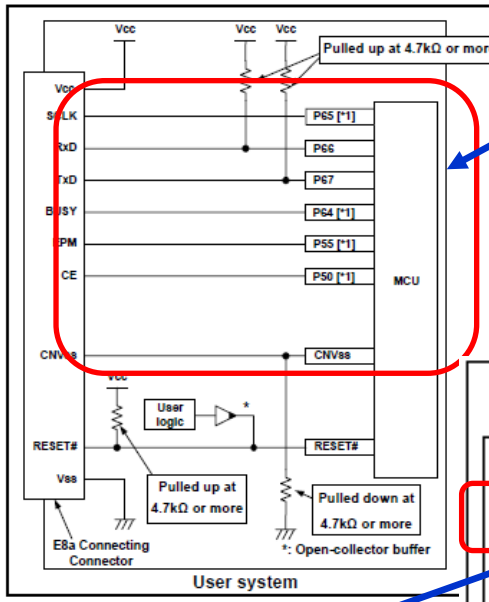
2. Differences between the Target Interfaces (for OCD)



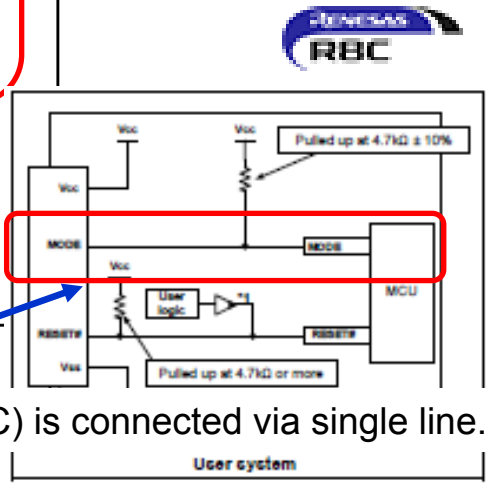
The RL78 and E1 emulator are connectable via the 14-pin connectors listed below.

	Type Number	Manufacturer	Specification
14-pin connector	7614-6002	Sumitomo 3M Limited	14-pin straight type (Japan)
	2514-6002	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.

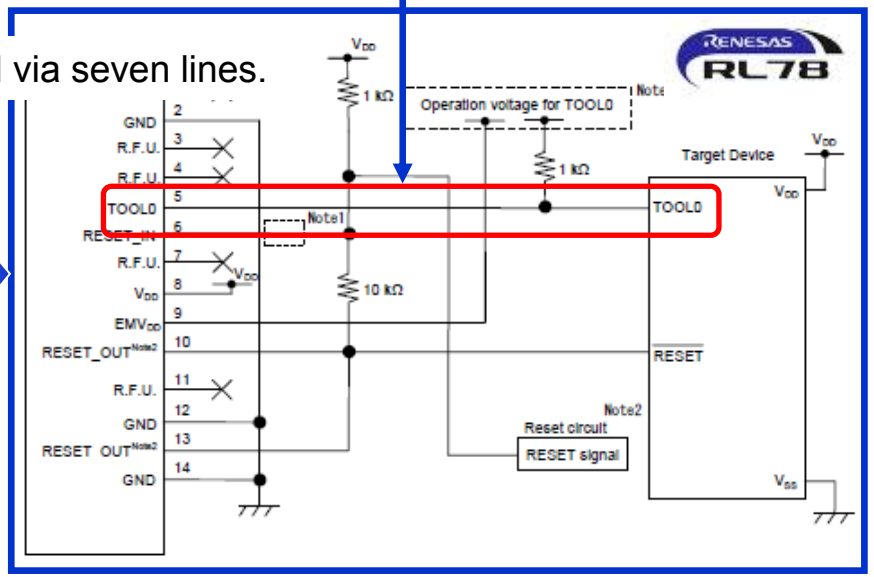


The E8a (for M16C) is connected via seven lines.



The E8a or E1/E20 (for R8C) is connected via single line.

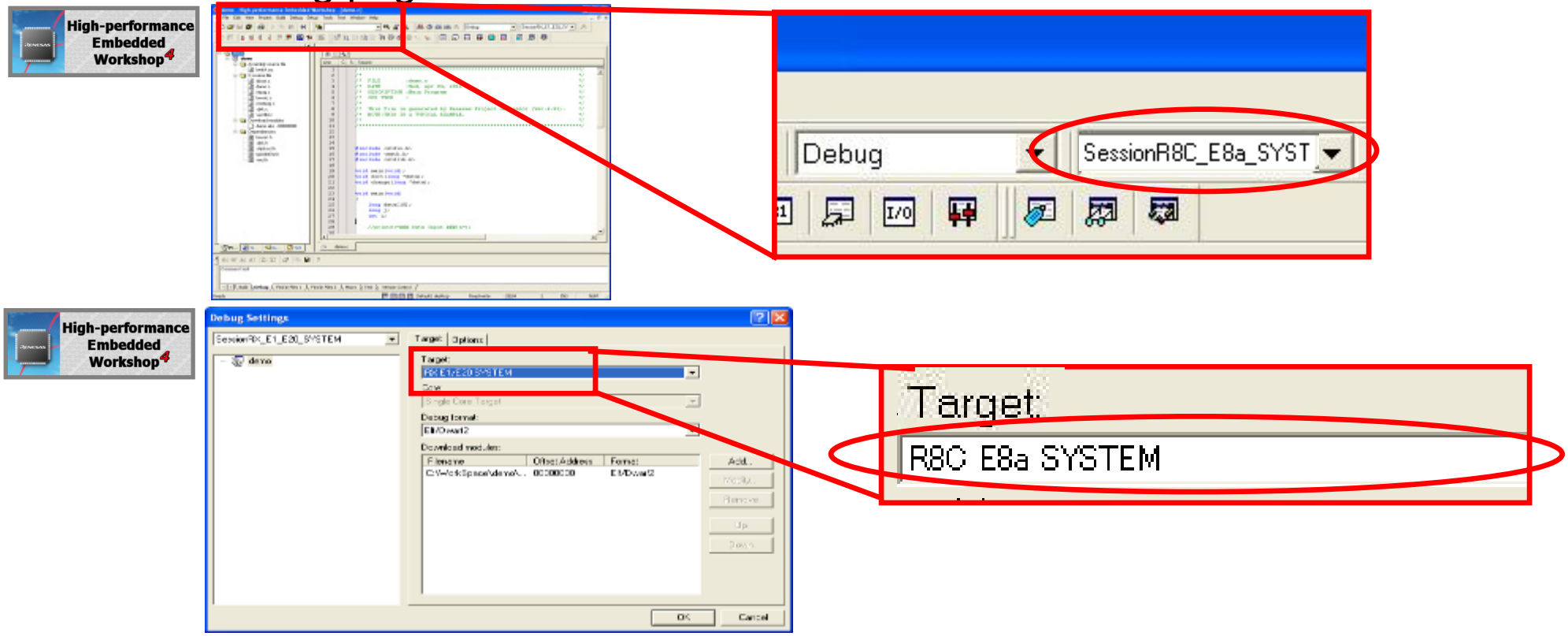
The E1 (for RL78) communicates via a single line.



The circuits for connection also vary with the MCU. For details, refer to *E1/E20 Emulator Additional Document for User's Manual (Notes on Connecting RL78)*.

3.Changing the Debugger

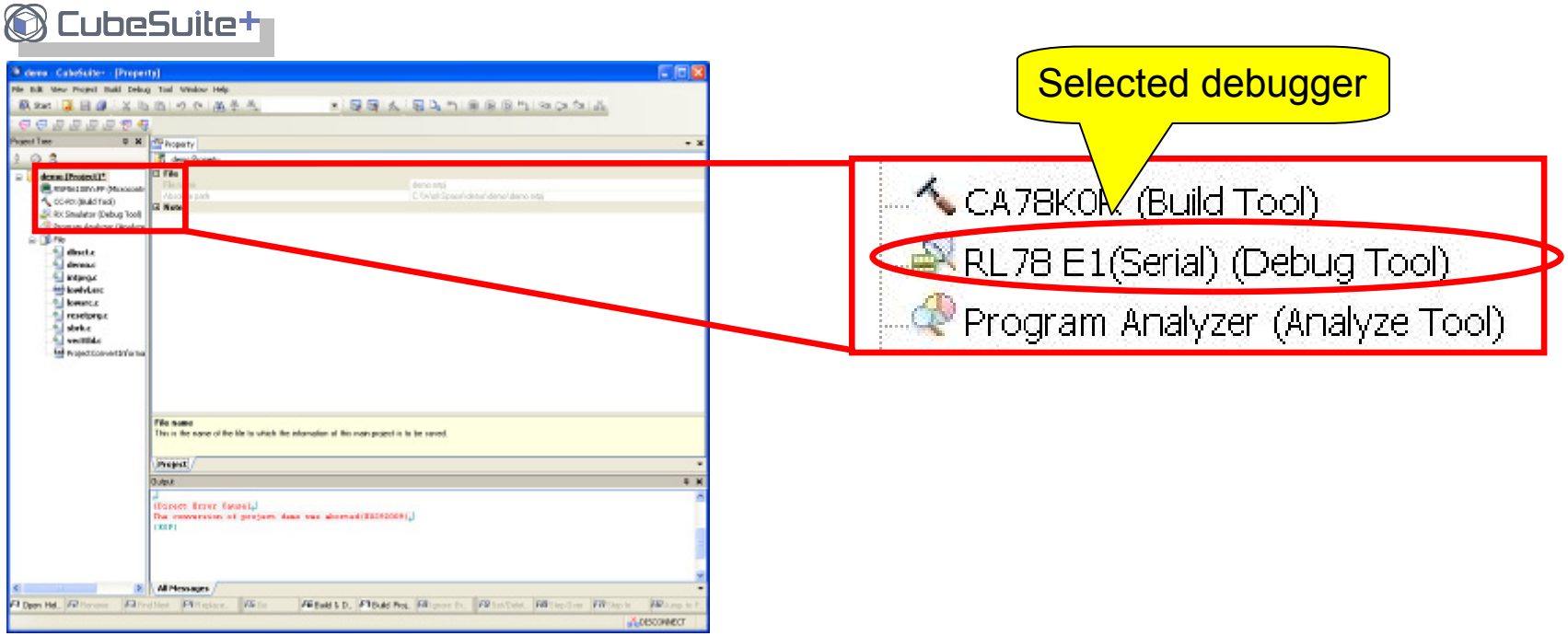
The HEW allowed users to select a debugger (E8a,E1/E20 emulator or simulator) in the process of changing the debug session or the target shown in the [Debug Settings] dialog box. The CubeSuite+, on the other hand, allows users to select a debugger on the project tree. The procedure to change the debugger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



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

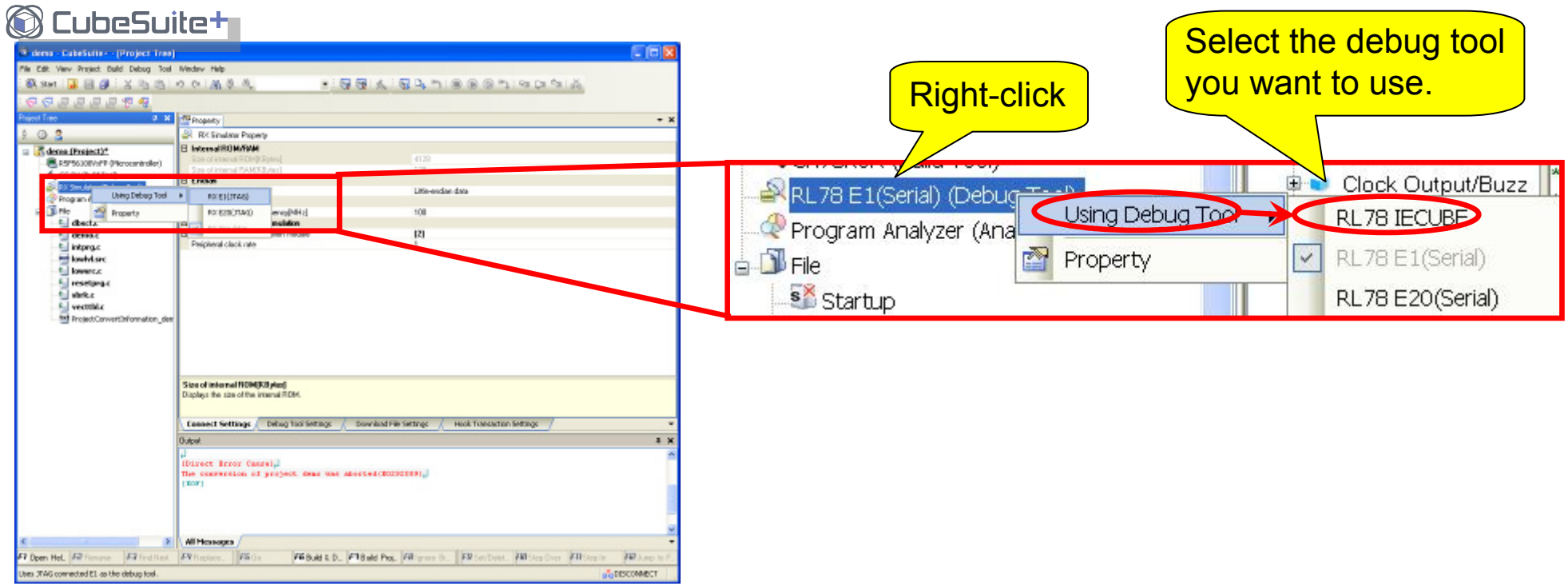
3. Changing the Debugger

(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:



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.



4. Entering an ID Code

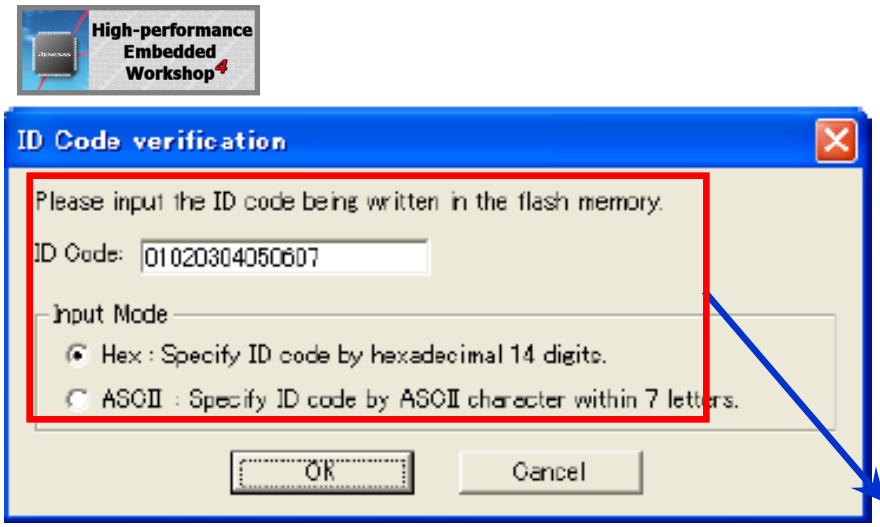
Both the R8C/M16C and RL78 require entering of an ID code.
However, there are some differences regarding the setting and authentication of the ID code and the action that is taken if the ID code does not match.

	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)

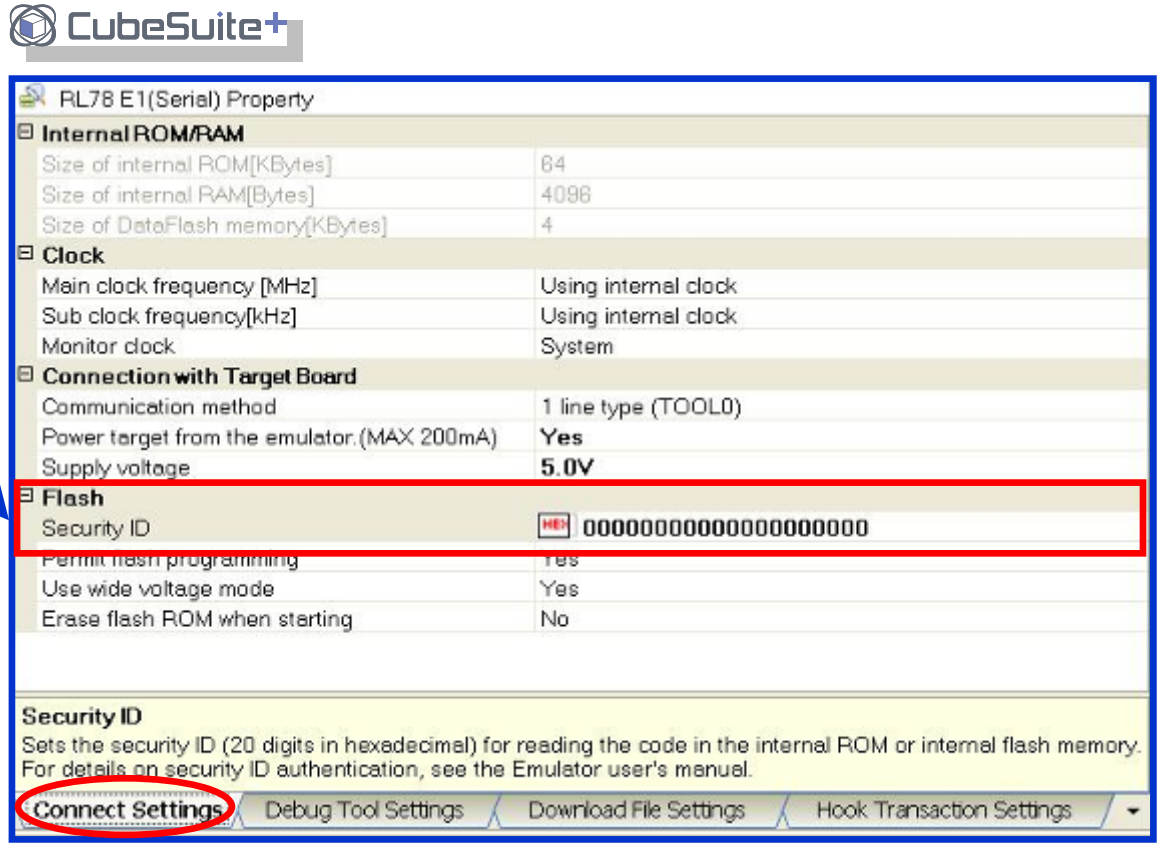
For details, see *E1/E20 Emulator Additional Document for User's Manual (Notes on Connecting RL78).*

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:



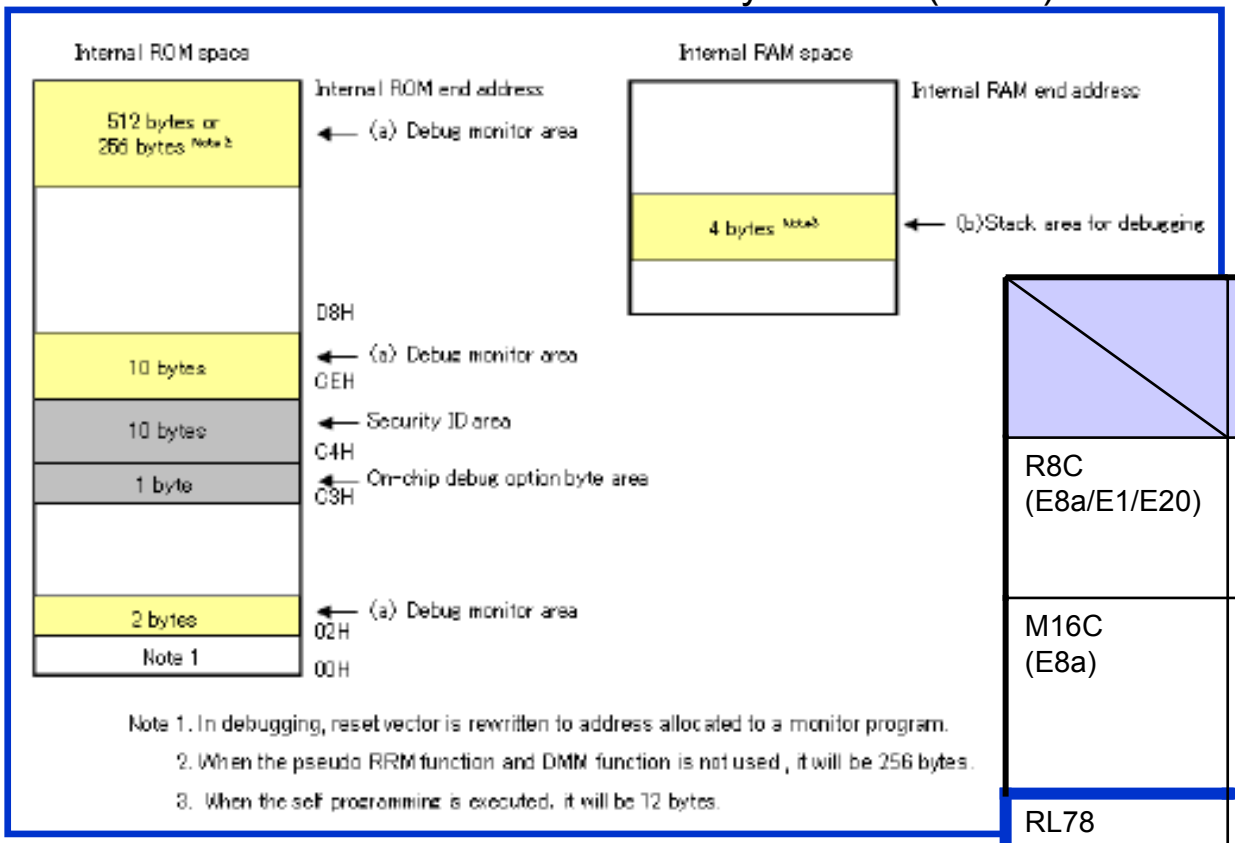
Example:
[ID Code] dialog box of the E8a for the R8C



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

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

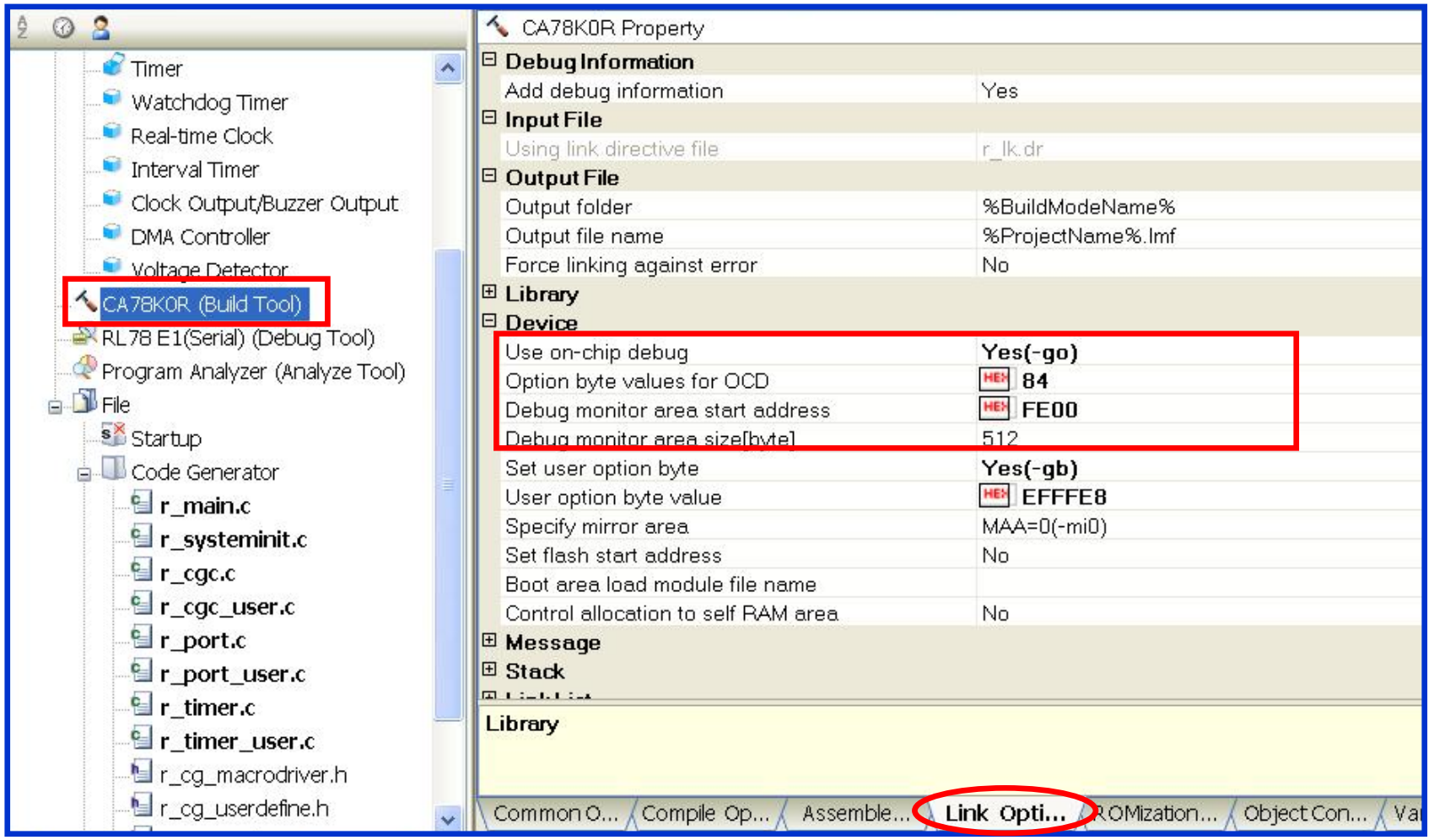
Comparison with the R8C and M16C

	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+)

5. Securing Resources

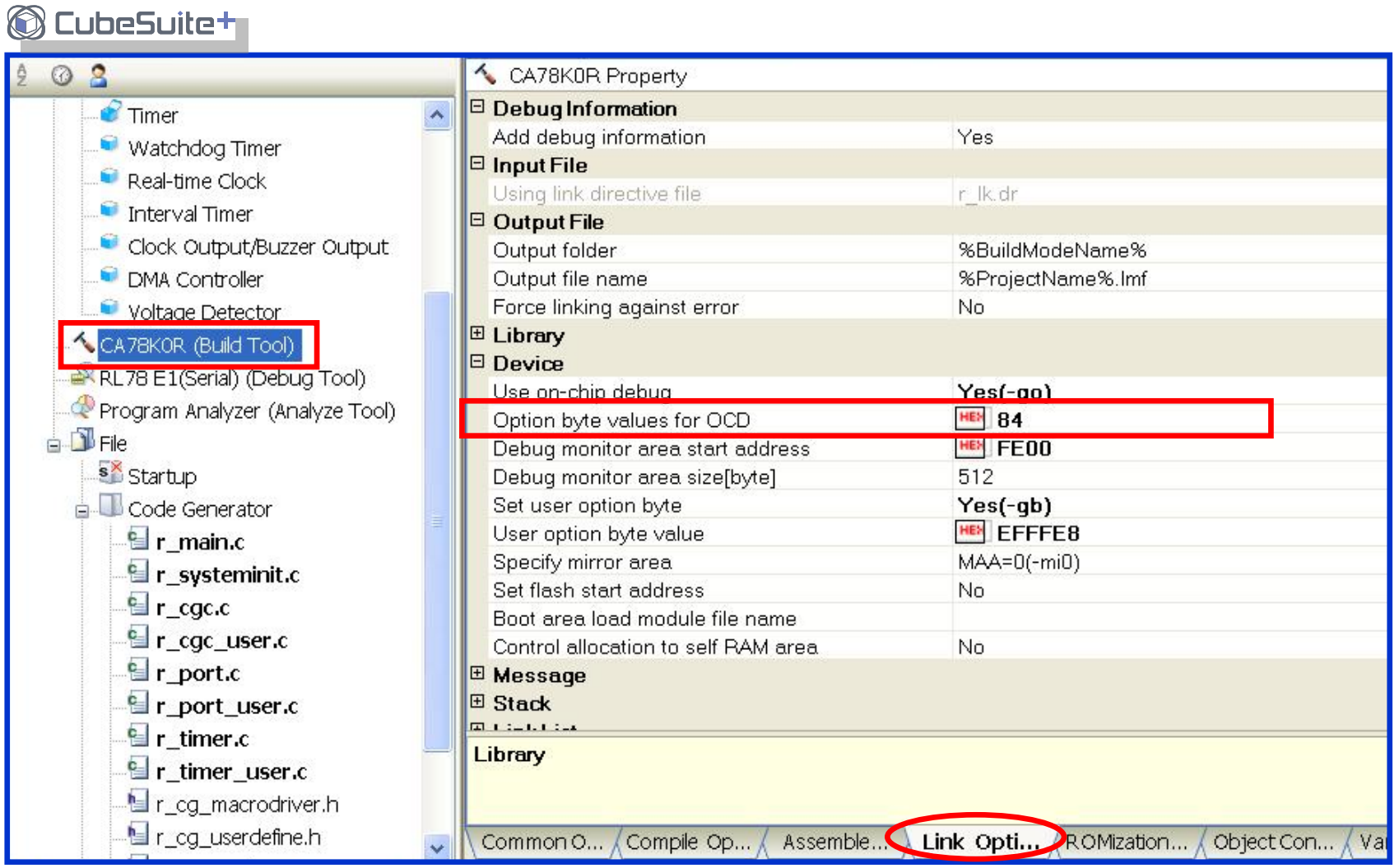
The address of the area for monitoring by the debugger can be specified on the [Link Options] sheet of the [Property] panel of the build tool.

CubeSuite+



6. Setting the On-Chip Debugging Option Byte

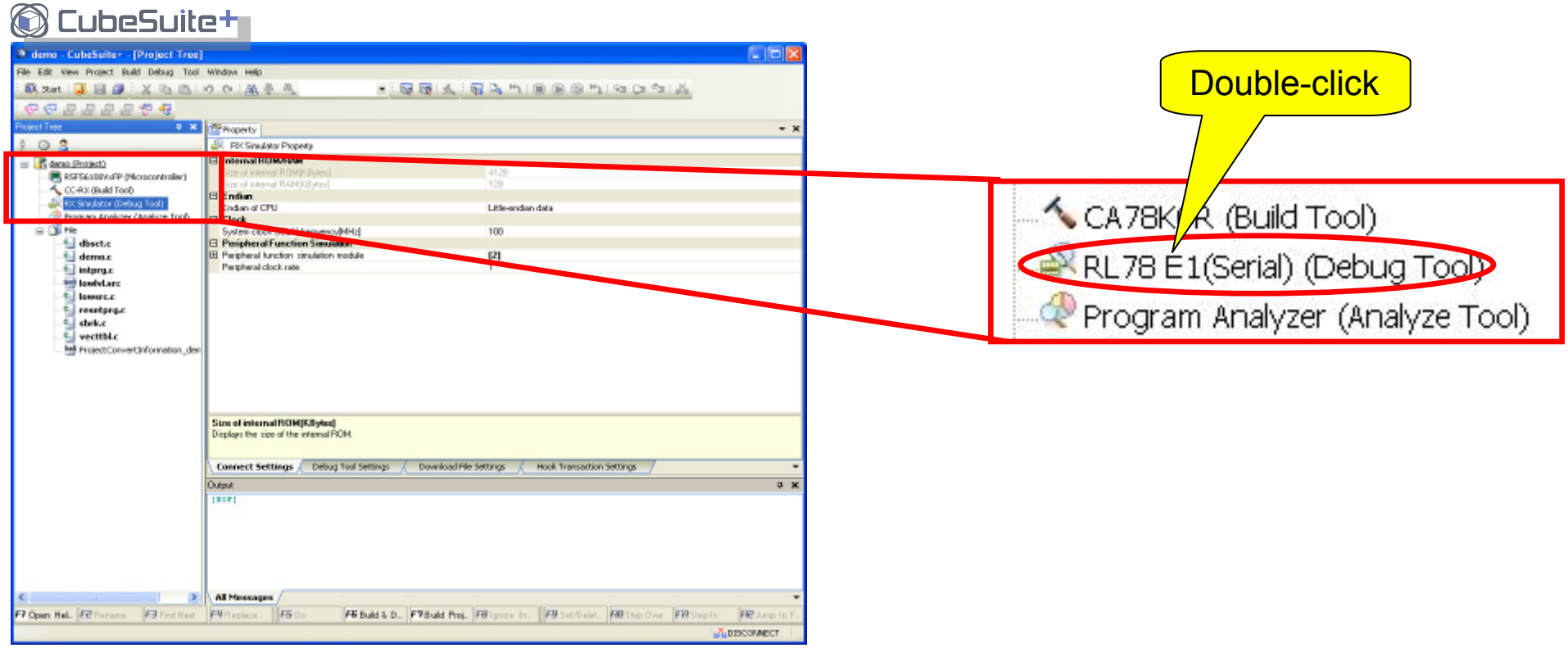
The on-chip debugging option byte can be set on the [Link Options] sheet of the [Property] panel of the build tool.



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.

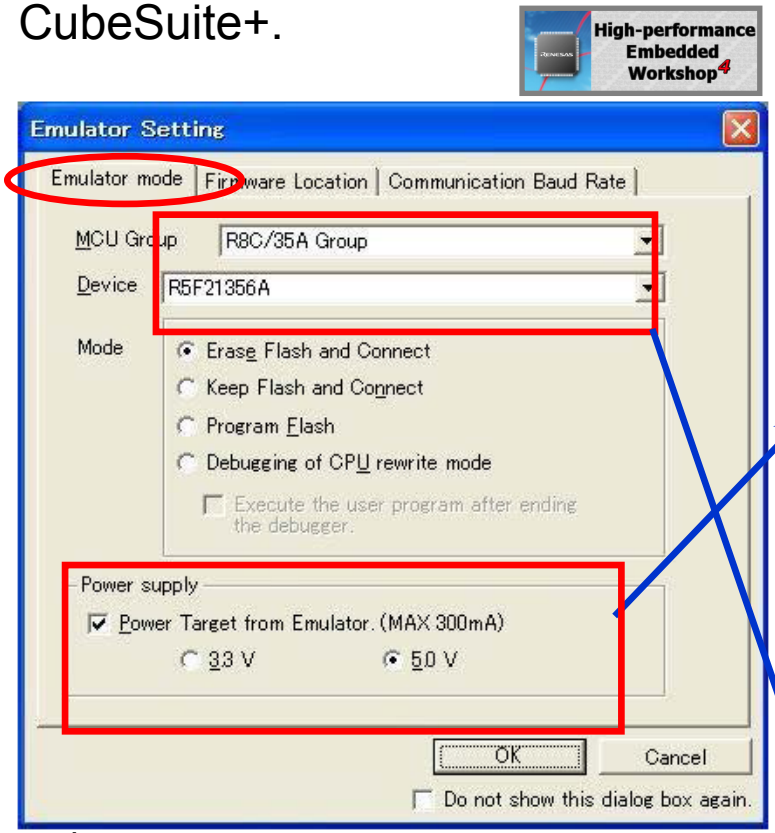
Double-click the debug tool name (debug tool) on the [Project Tree] panel to open the Properties window of the debug tool.



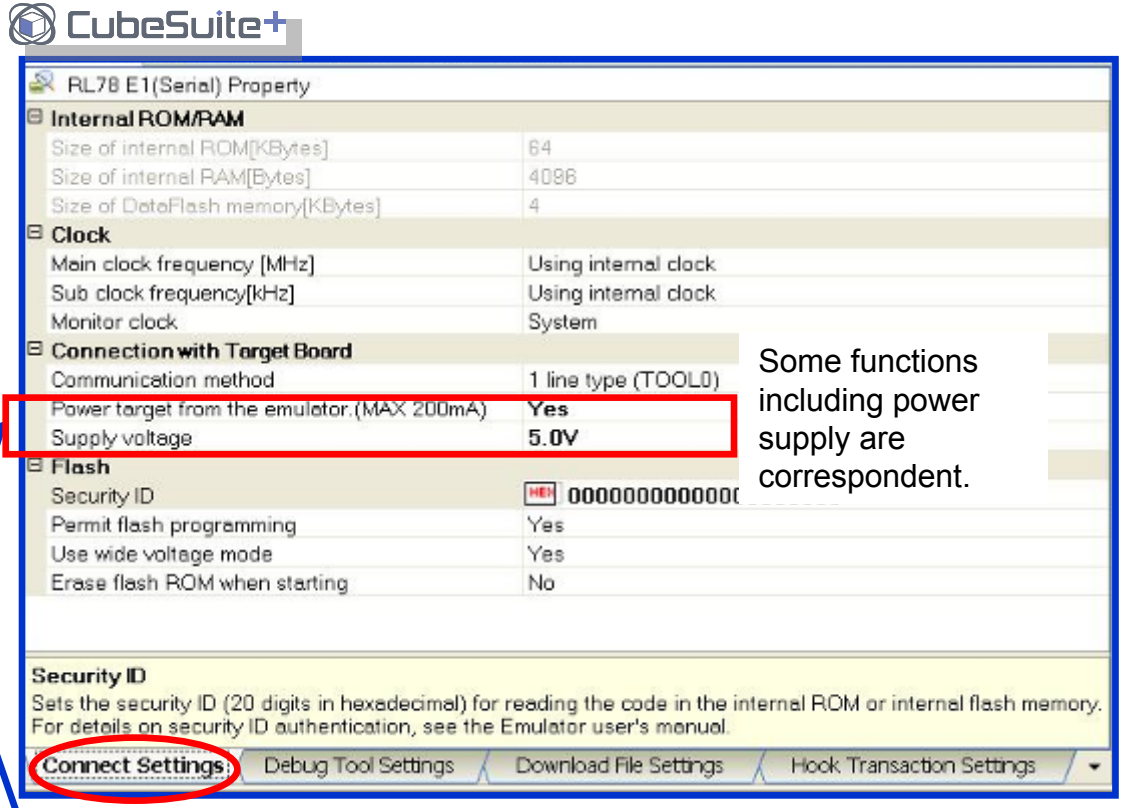
7.Where Do We Make Settings when Connecting an Emulator?

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+.



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

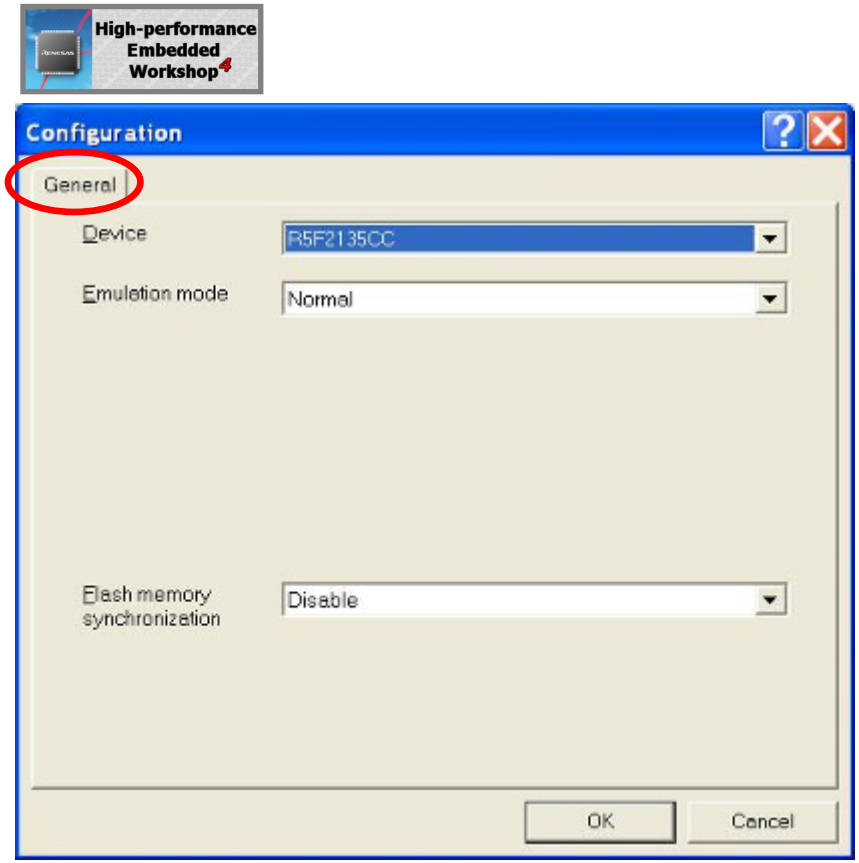


Some functions including power supply are correspondent.

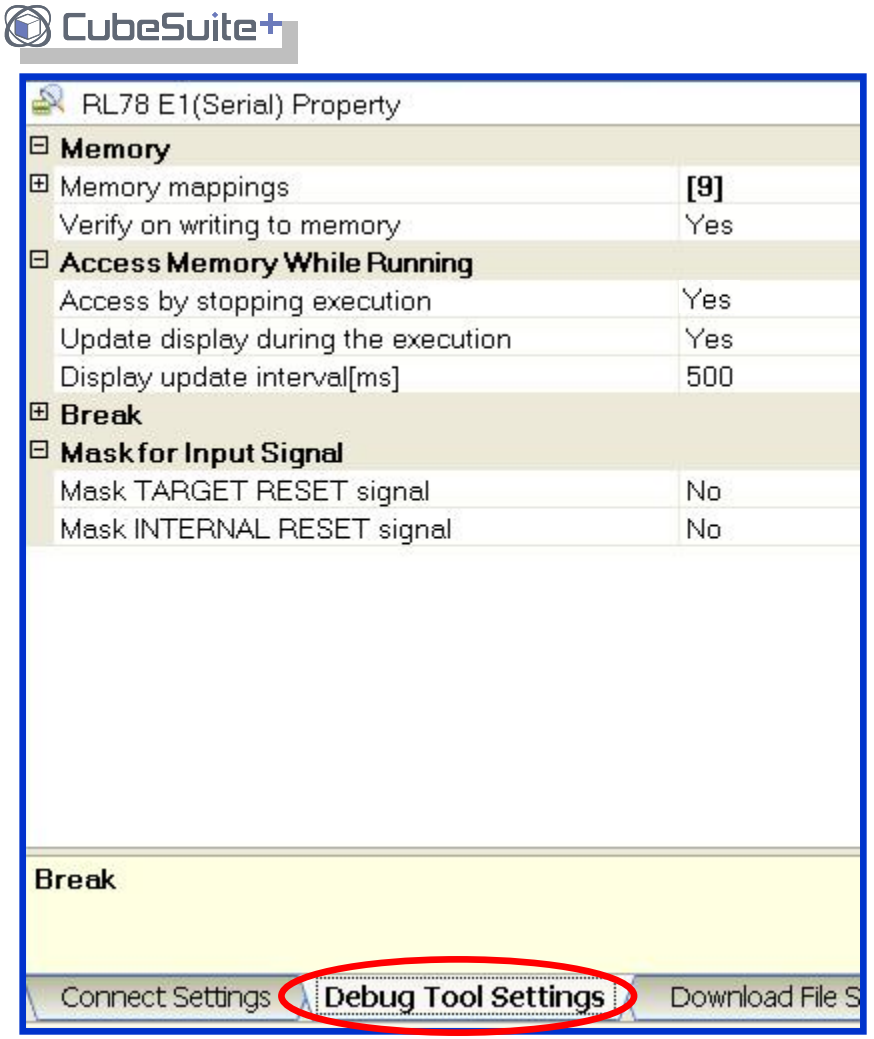
On CubeSuite+, the device needs to be selected during the process of creating a project.

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+.

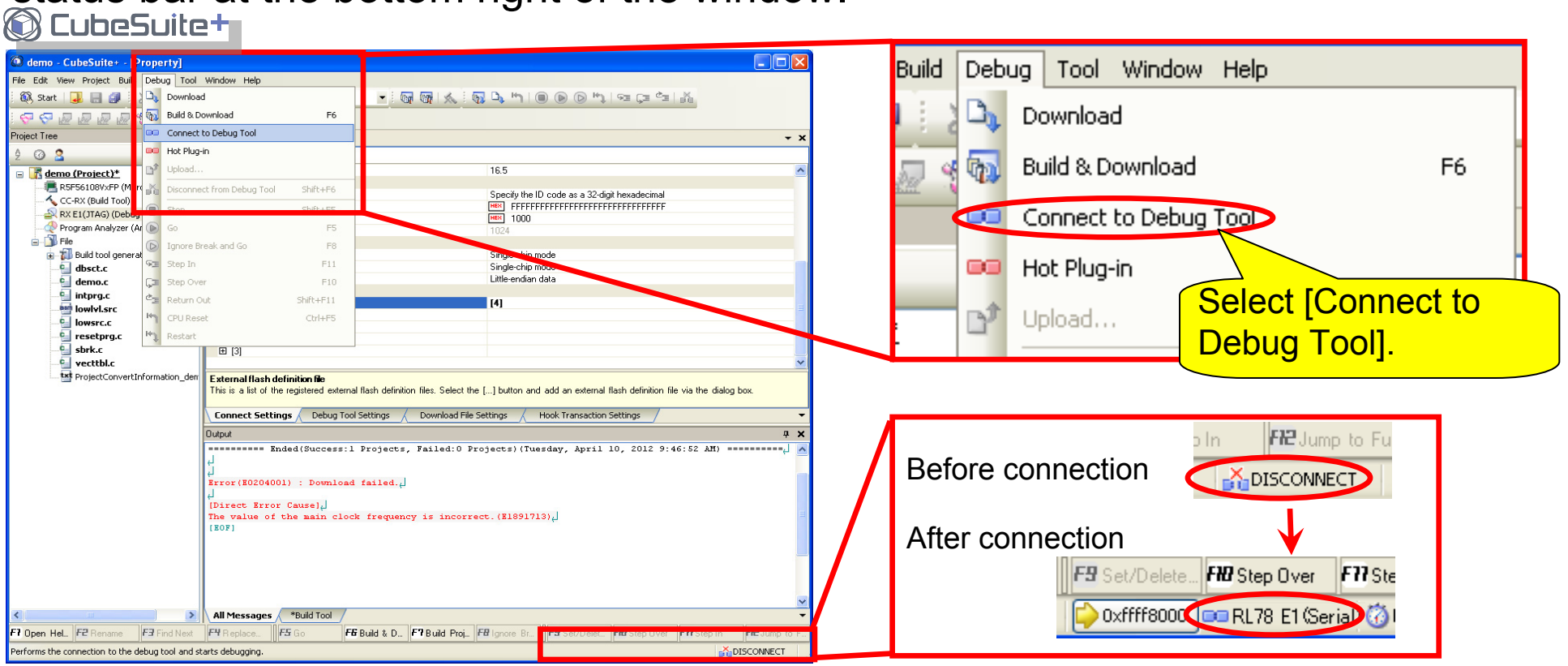


Example:
[Configuration] dialog box of the E8a for the R8C



8. Connecting an Emulator

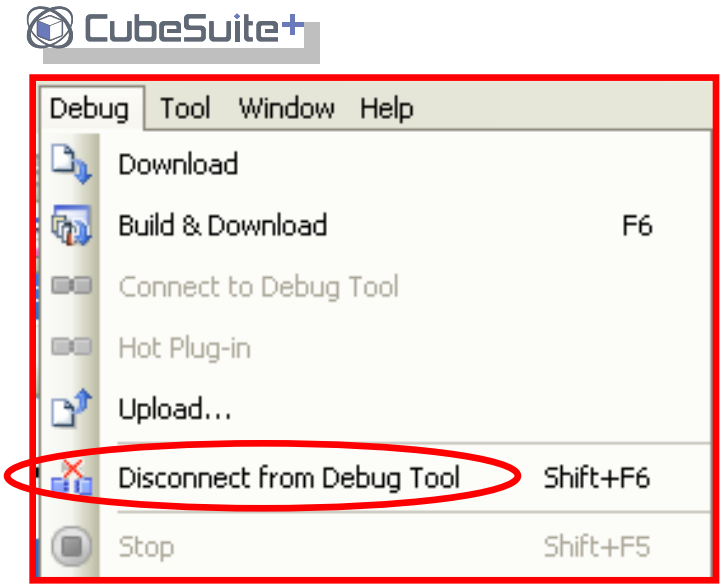
Select [Debug] -> [Connect to Debug Tool] from the CubeSuite+ menu to establish connection to the selected emulator (debug tool).
Upon completion of the connection, the debug tool name appears on the status bar at the bottom right of the window.

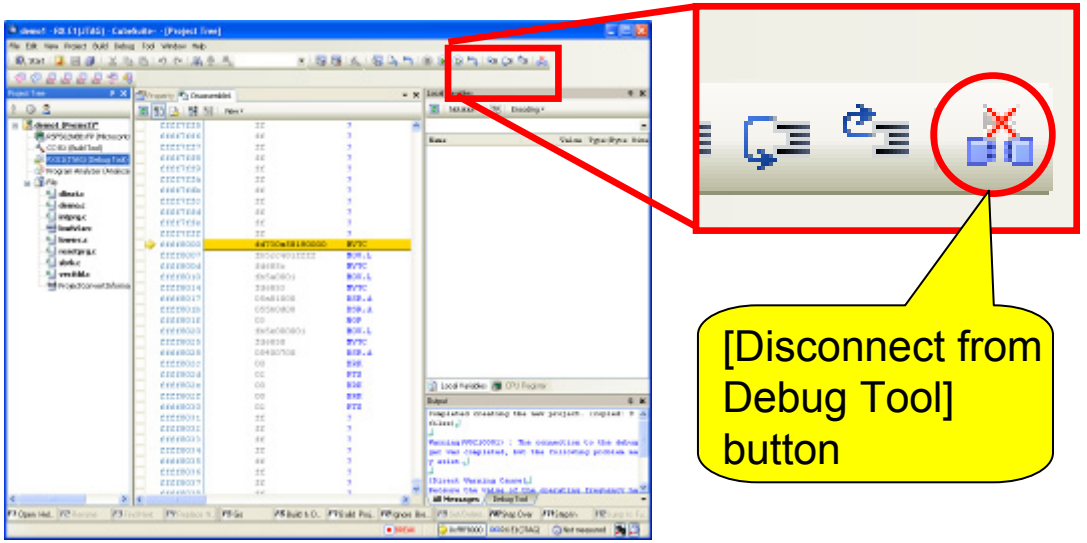


Note: If an ID code has been written in the MCU, set an ID code in advance according to “2. Entering an ID Code.”

9. Disconnecting the Emulator

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



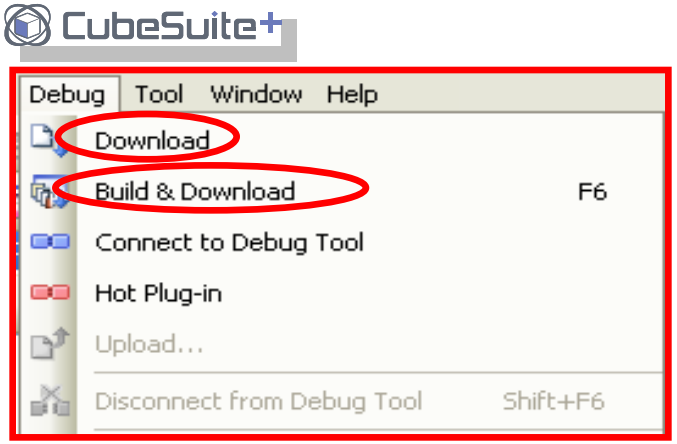


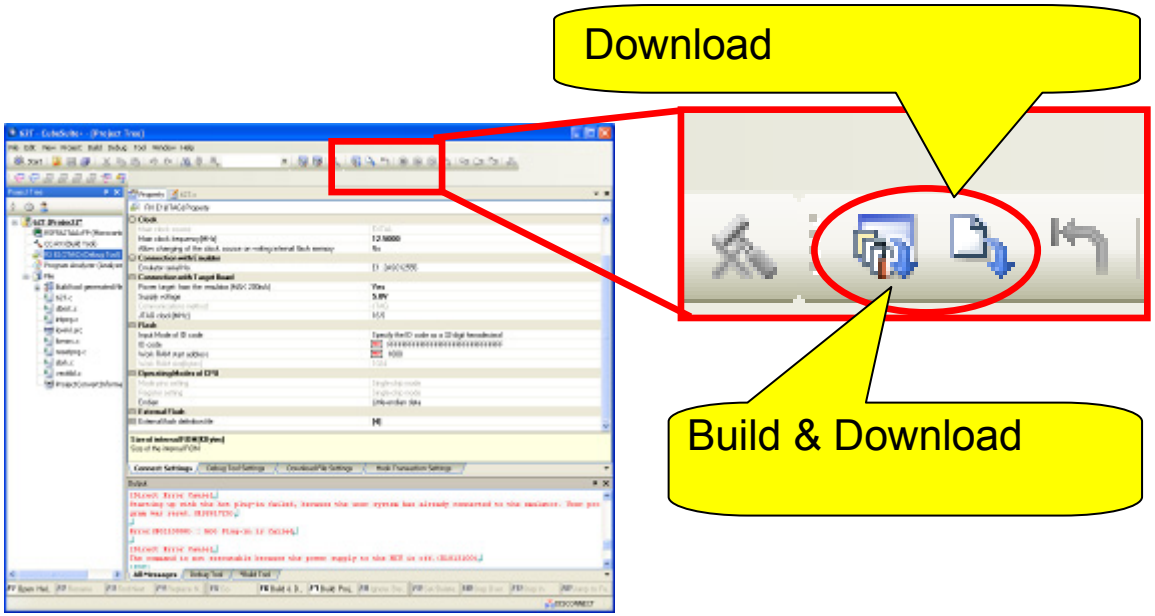
10. Downloading a Program

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.

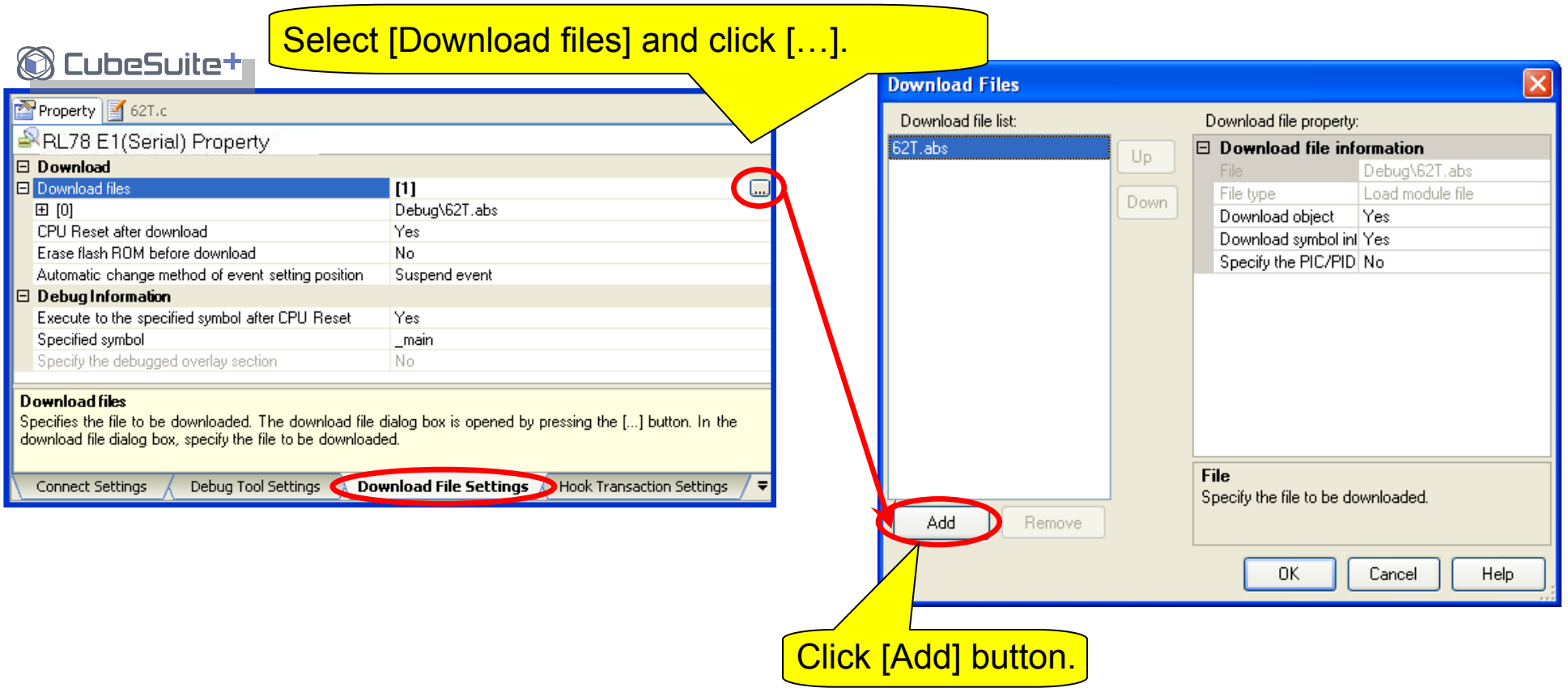
If no debug tool is connected, CubeSuite+ connect debug tool automatically before downloading.





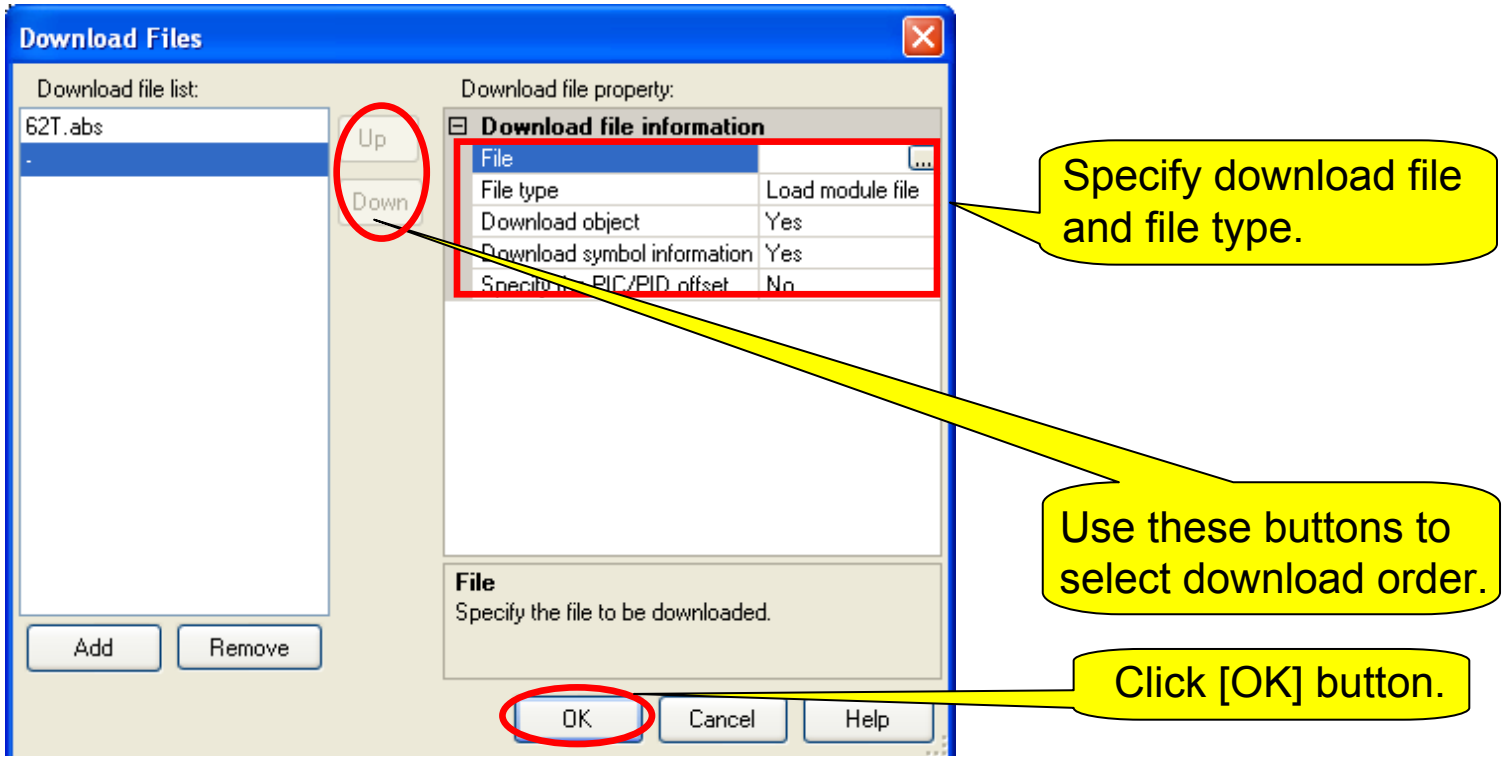
11. Registering Additional Download Files

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.



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

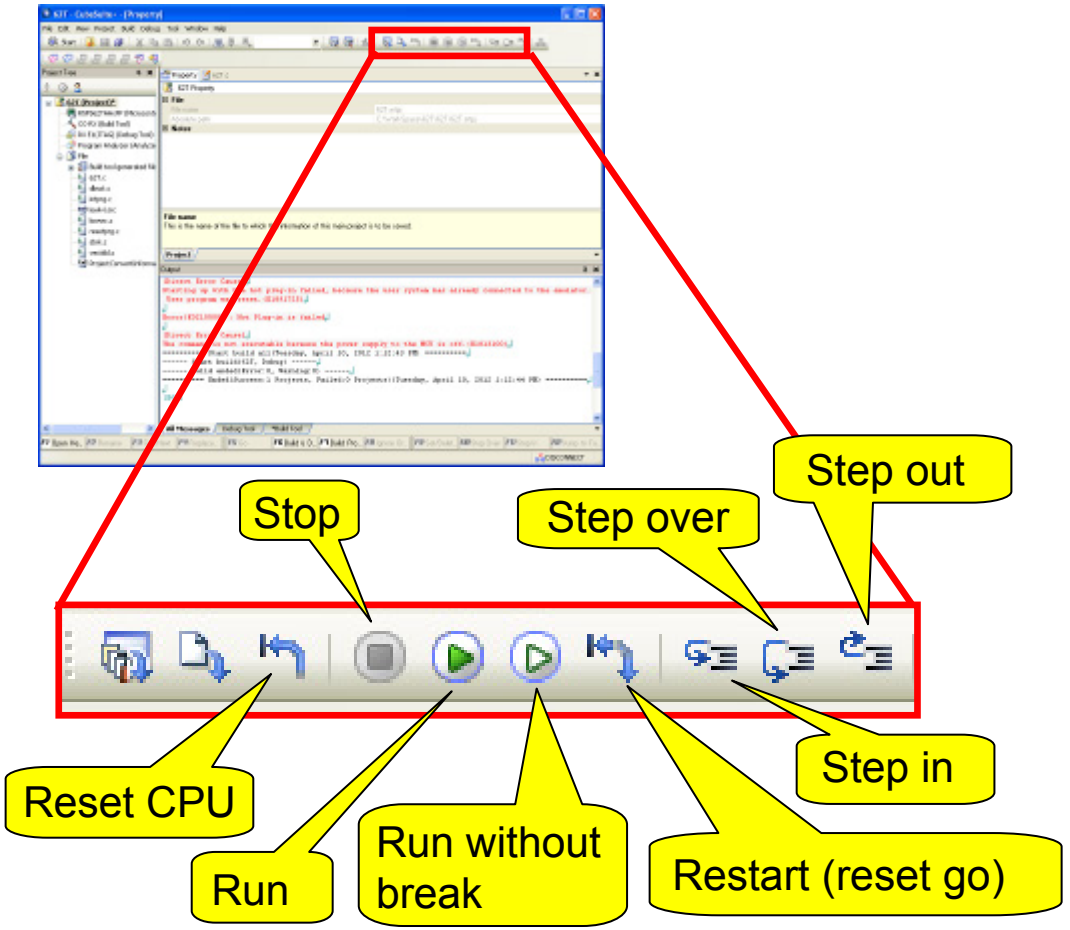
(4) The available file formats (extensions) for the R8C/M16C and for the RL78 are not the same. For details, see the table below.

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

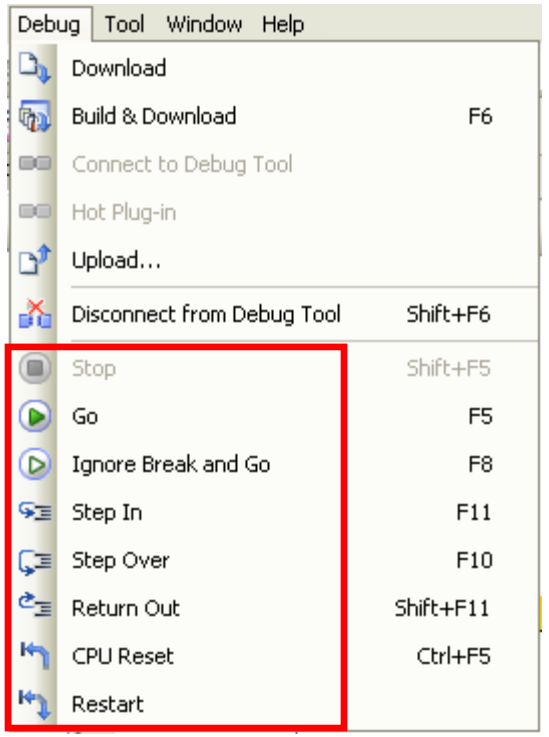
12. Starting/Stopping a Program

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+



CubeSuite+ menu

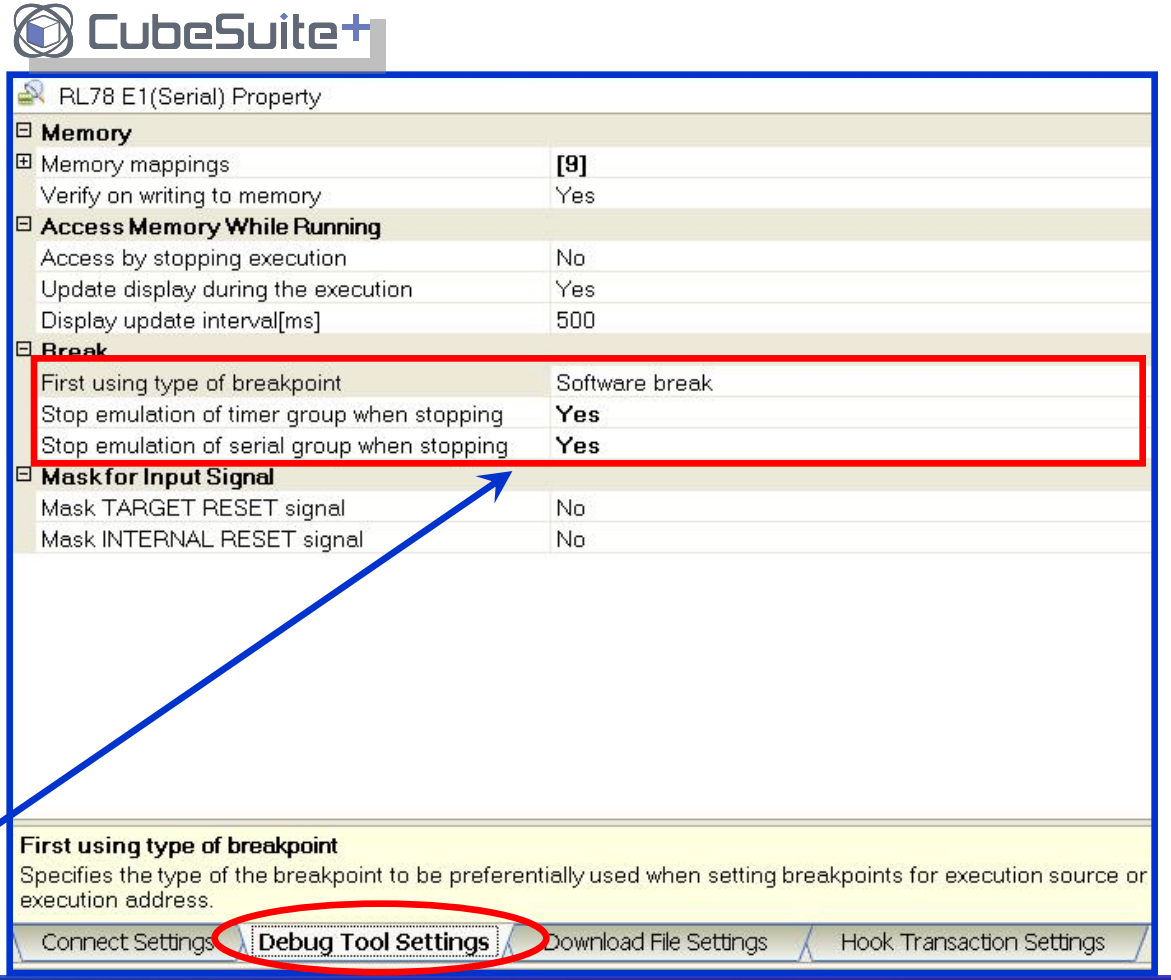


13. Difference in MCU Operation during a Break (Peripheral Break Function)

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)

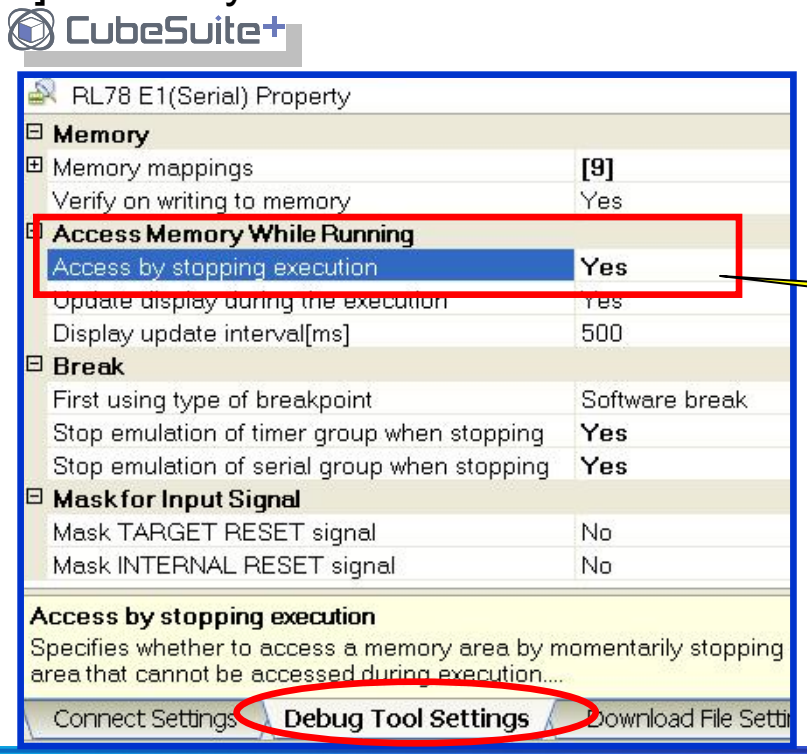
Selectable on CubeSuite+ for the RL78



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.

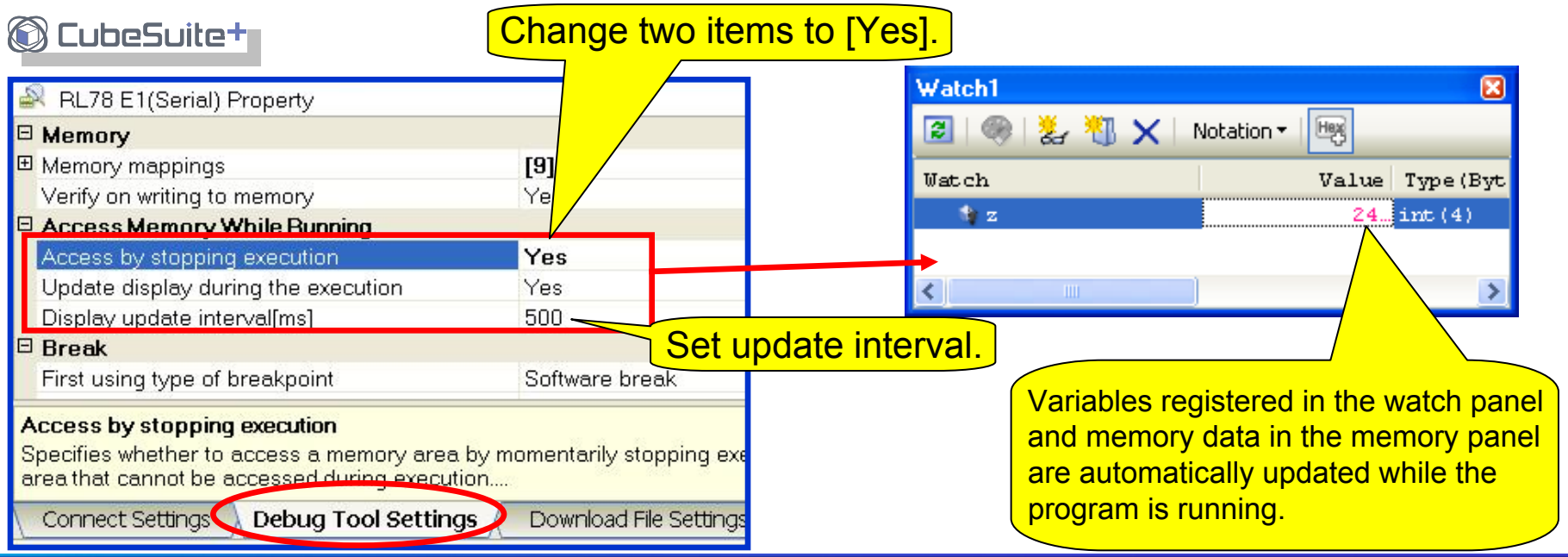


Change this to [Yes].

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

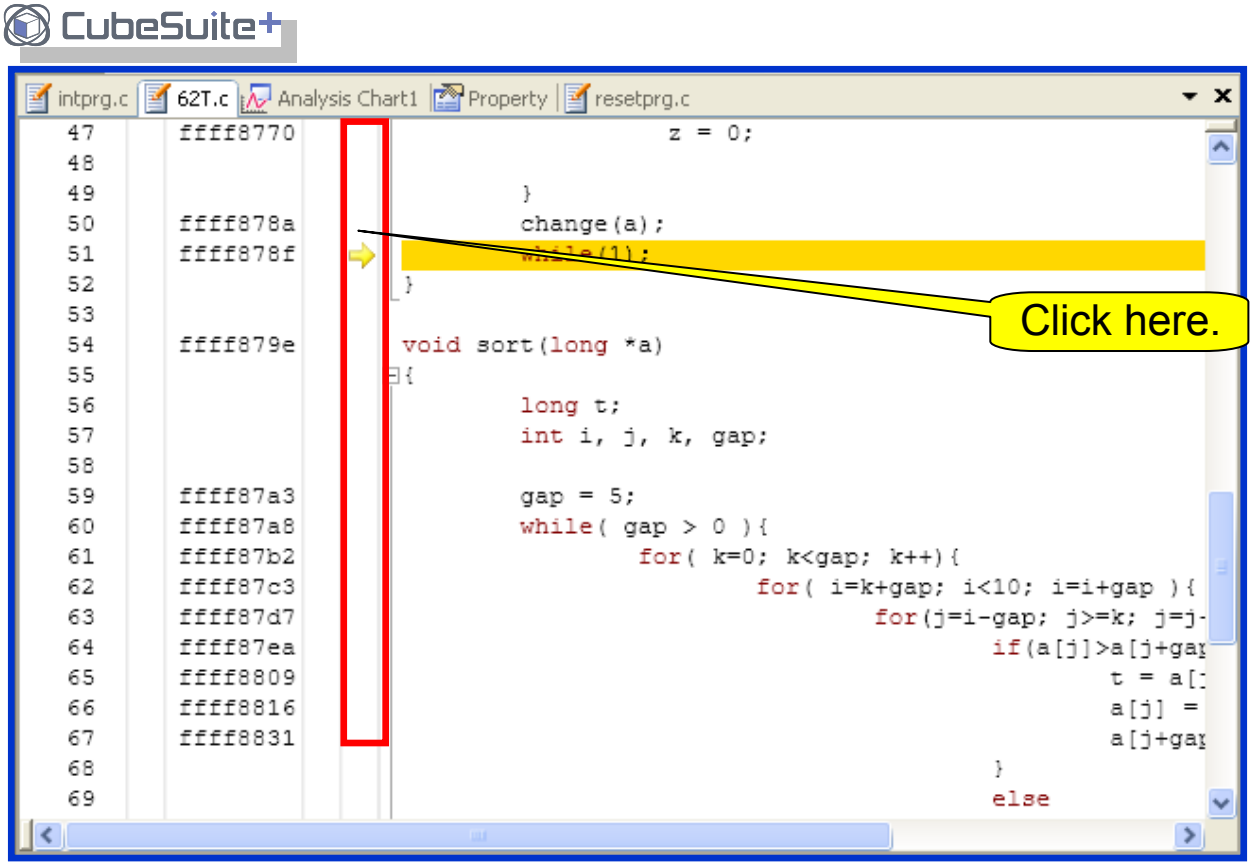
15. Automatically Updating Memory Data and Variables While the Program Is Running

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 [Propertiy] 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.



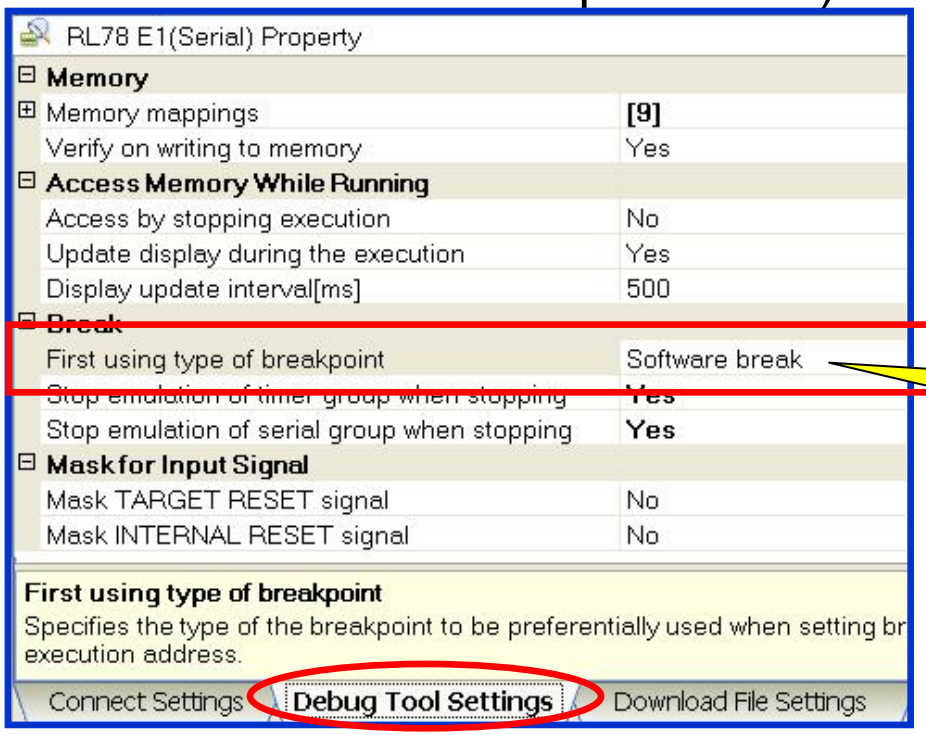
16. Setting Breakpoints

(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

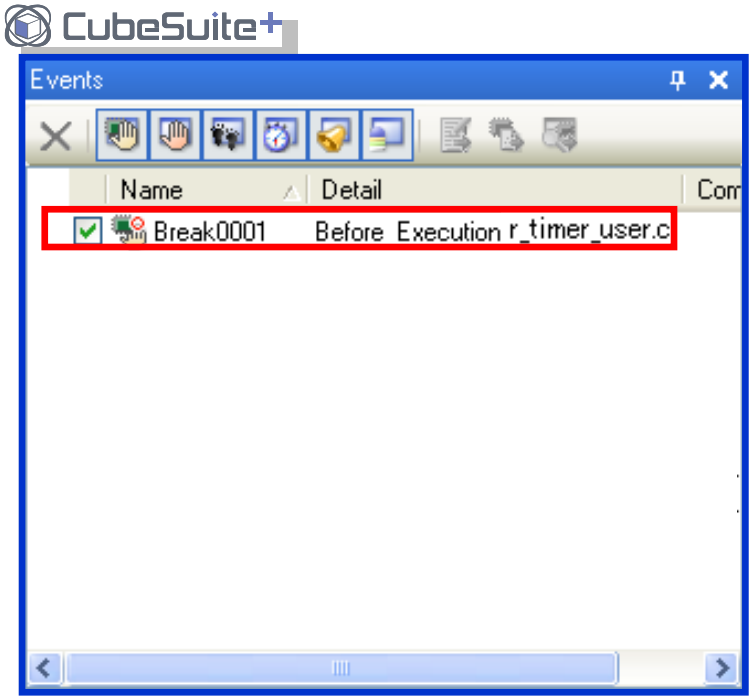
(2) Select a breakpoint type (software break or hardware break) for [First using type of breakpoint] in the [Debug Tool Settings] sheet on the [Property] panel. (Software break is selected in the example below.)



(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

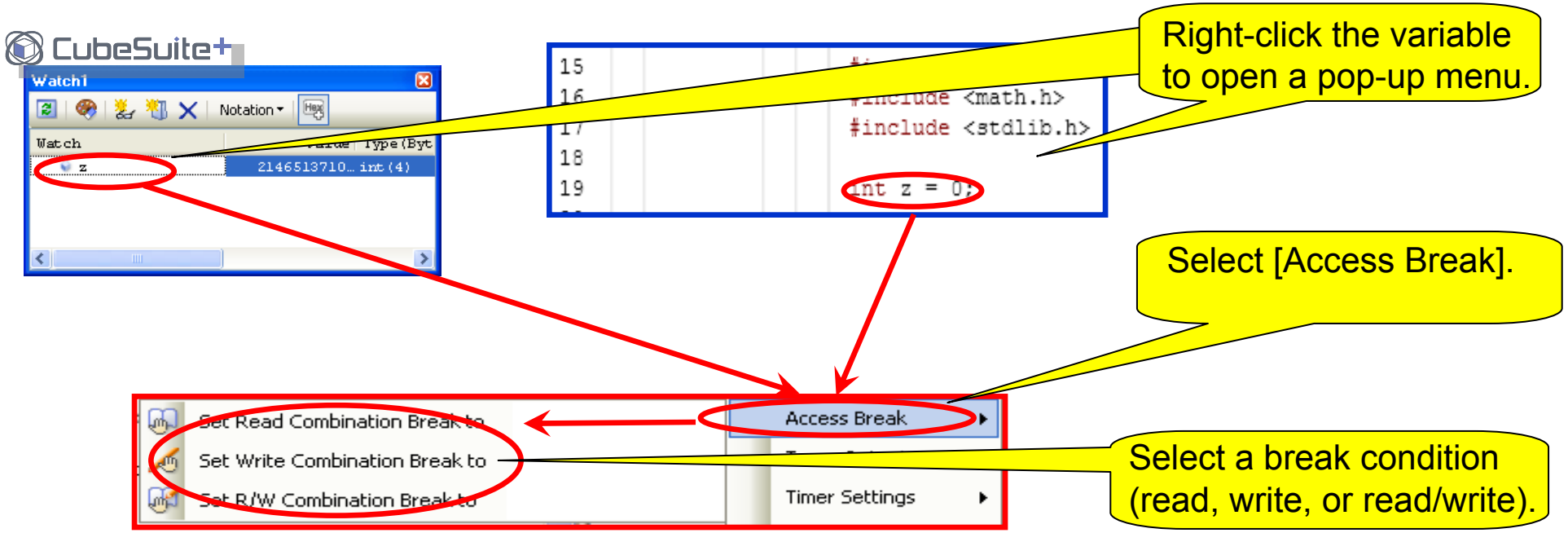
16. Setting Breakpoints

(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.



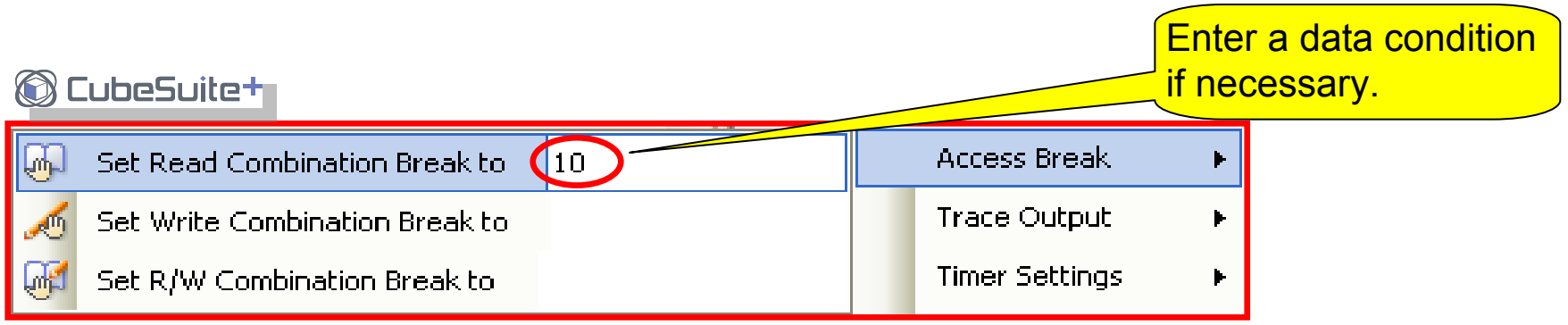
17. Causing a Break on Access to a Variable

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].



17. Causing a Break on Access to a Variable

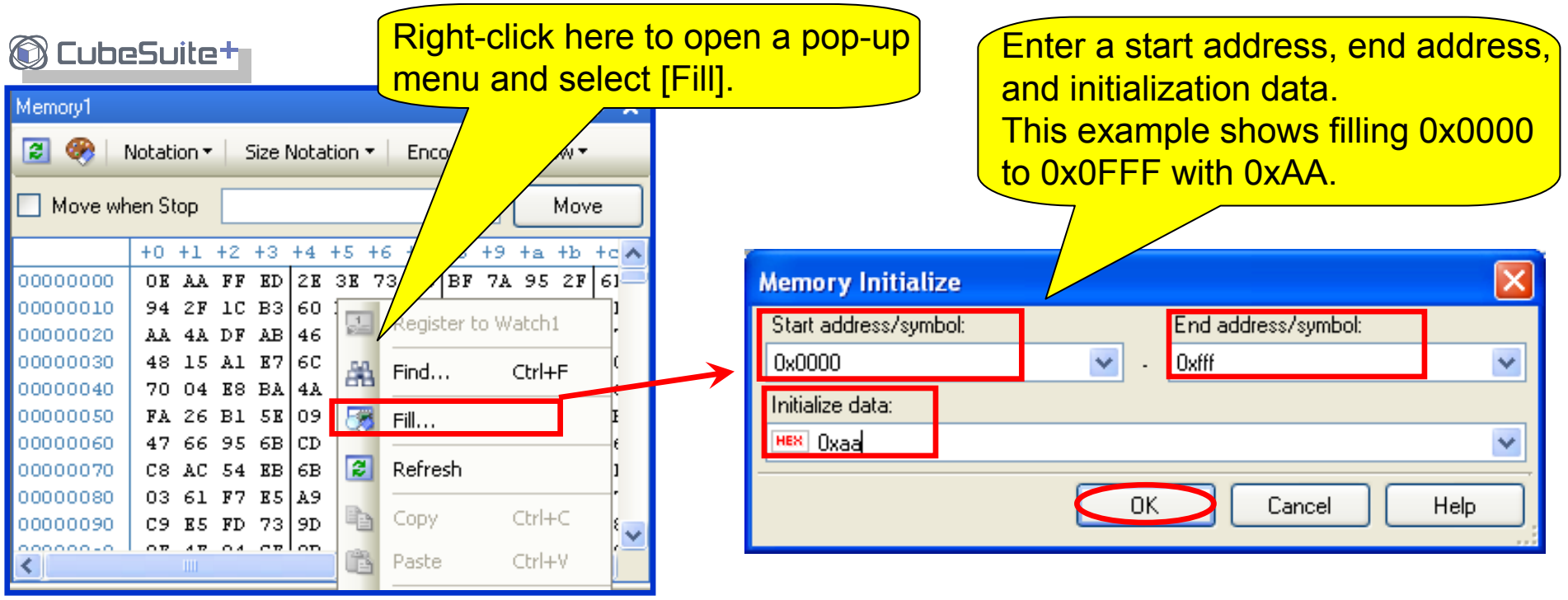
(3) Enter a value to set a data condition (or leave the box blank if no data condition is needed).



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

18. Filling Memory

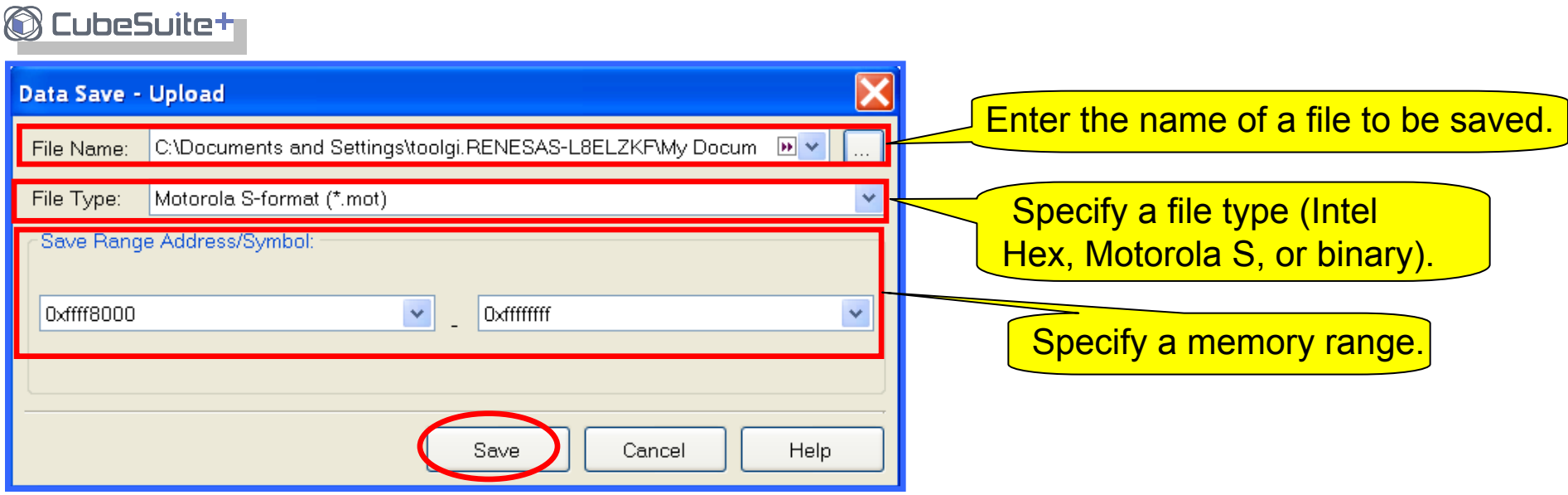
Memory can be filled (batch change) by using the [Memory Initialize] dialog box.
(1) Right-click on the [Memory] panel to open a pop-up menu, and select [Fill] from the pop-up menu.
(2) The [Memory Initialize] dialog box opens. Enter addresses (start address and end address) and initialization data, and then click the [OK] button.



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.



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

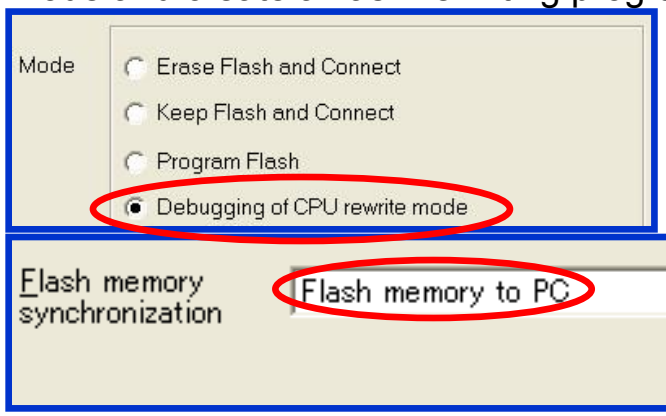
20. Flash Self-Programming

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.

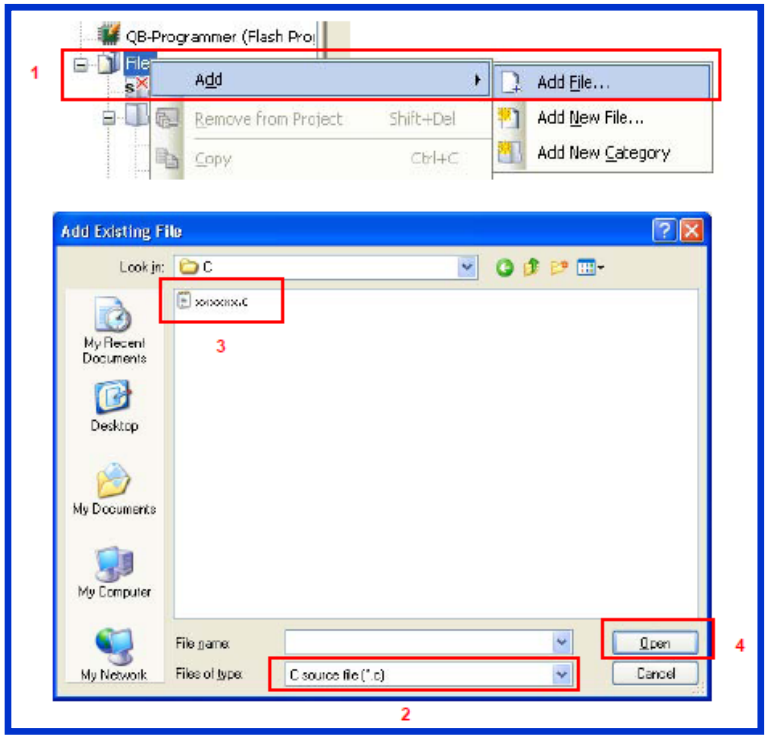
CubeSuite+

High-performance Embedded Workshop4

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

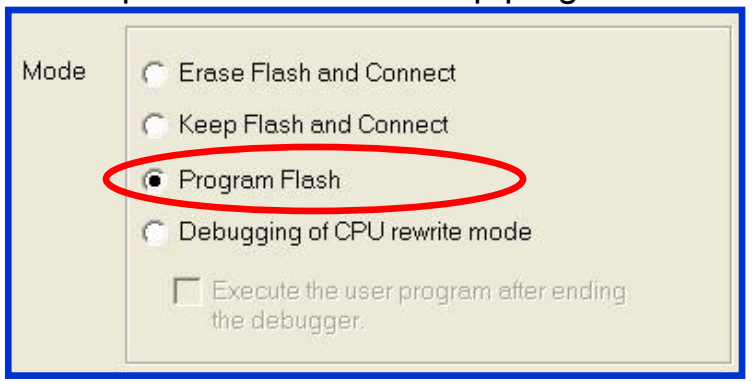


To add the flash self-programming file, right-click on [**File**] in [**Project Tree**] on CubeSuite+ and select [**Add** -> **Add File**] from the list, then click on the [**Files of type**] pull-down menu and select the file type.

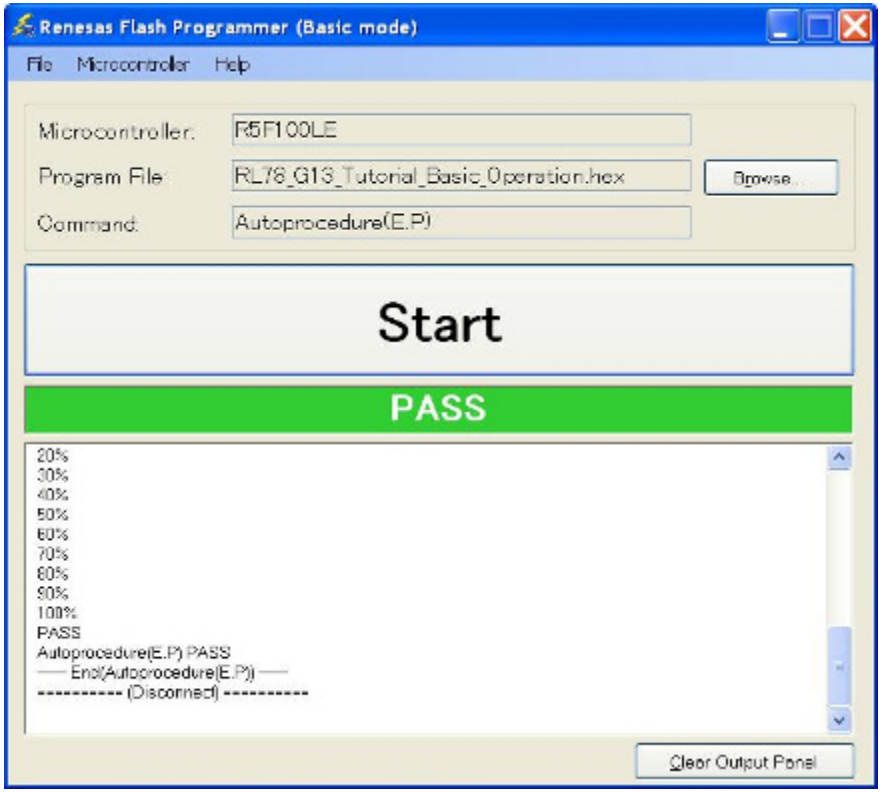
21. How to program to check Operation on the Stand-Alone MCU

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.

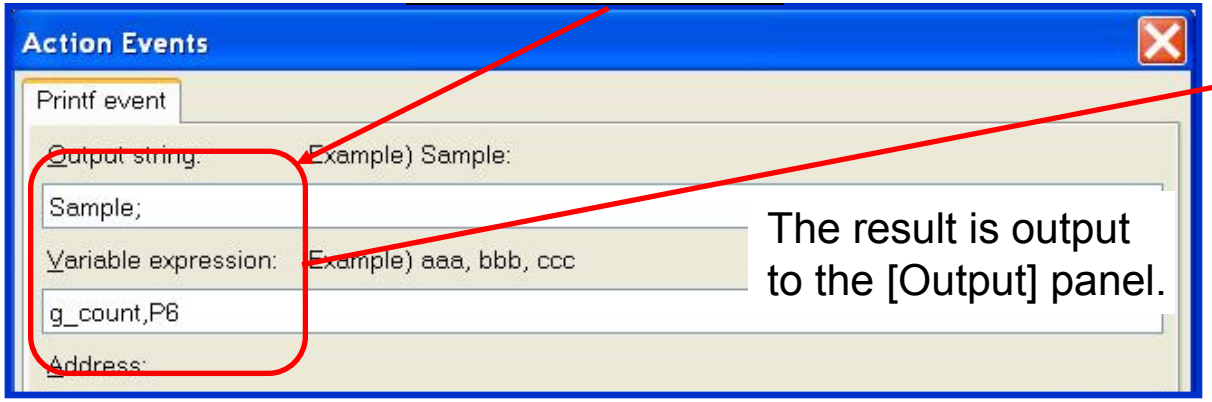
22. Action Event (Printf Event)

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.

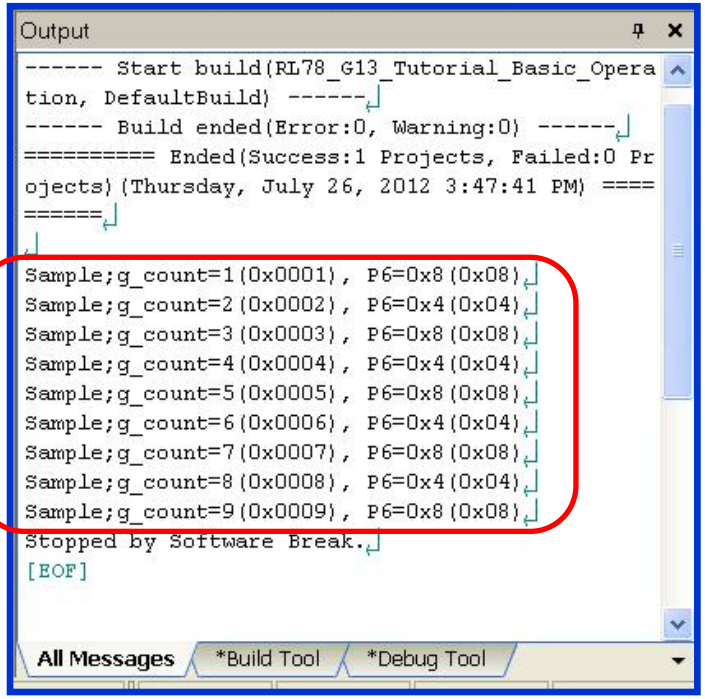
CubeSuite+

```
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.

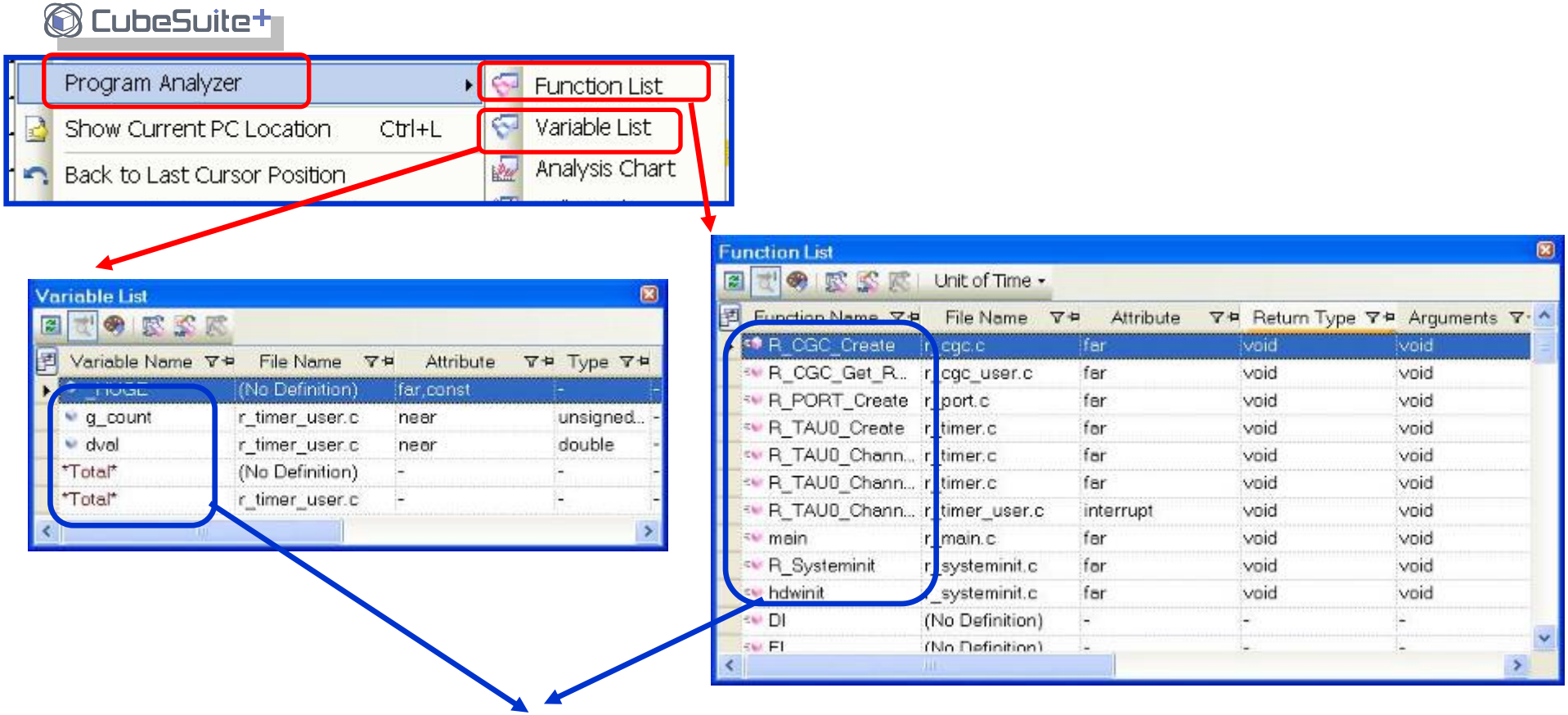


The result is output to the [Output] panel.



23. Viewing Lists of Variables and Functions

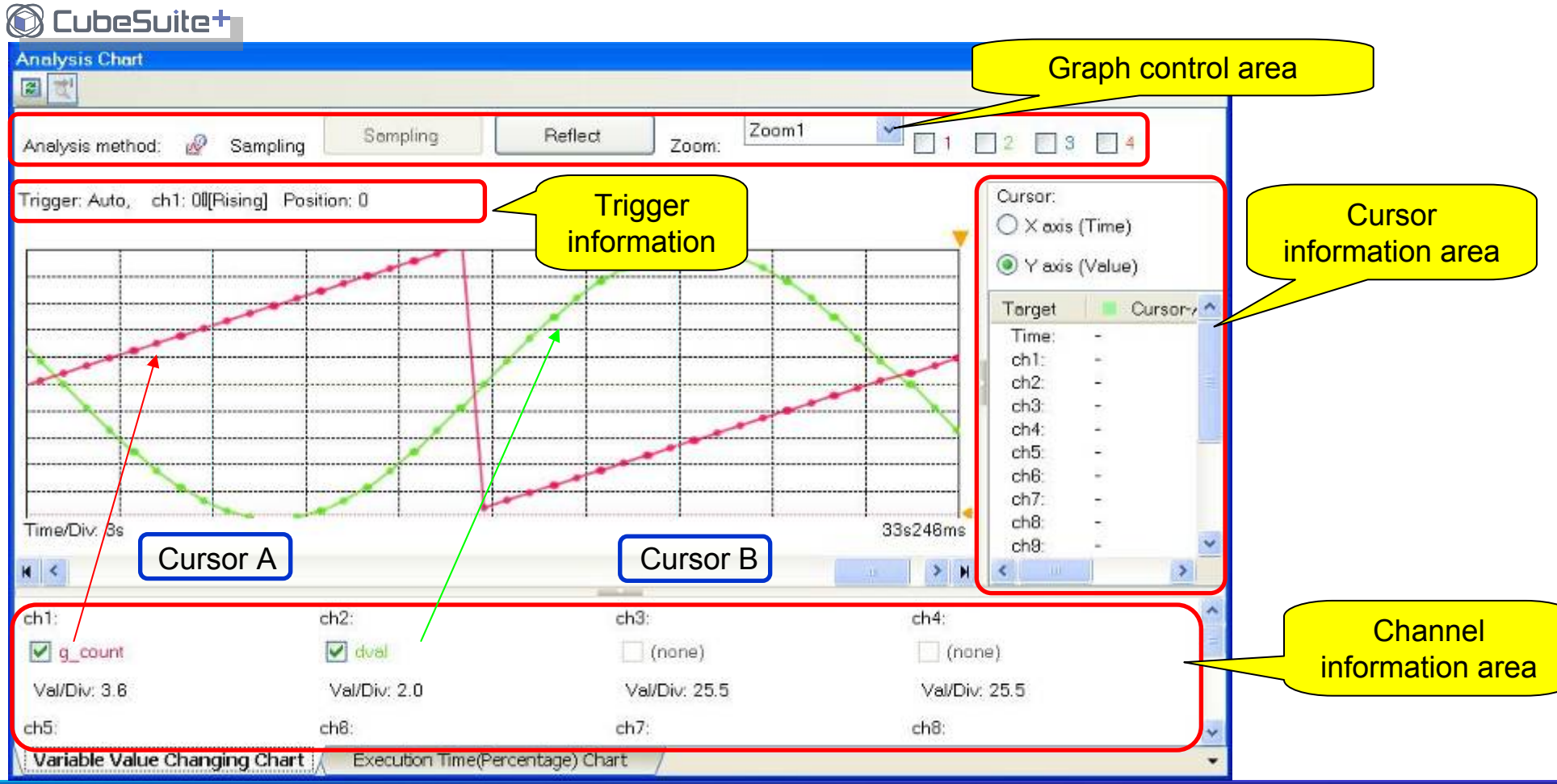
CubeSuite+ can automatically display lists of variables and functions used in the project.
Select [View -> Program Analyzer] from the menu.



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.



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 μs 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 μ 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