

TChapter 1. Target Devices.....	2
Chapter 2. User's Manuals	3
Chapter 3. Key Points for Selecting Uninstallation Method.....	4
Chapter 4. Changes.....	5
Chapter 5. Cautions.....	6
Chapter 6. Restrictions	25
Chapter 7. Changes in User's Manual.....	26

Chapter 1. Target Devices

The target devices supported by the CubeSuite+ are listed on the Website.
Please see this URL.

CubeSuite+ Product Page:

<http://www.renesas.com/cubesuite+>

Chapter 2. User's Manuals

Please read the following user's manuals together with this document.

Manual Name	Document Number
CubeSuite+ V1.02.00 Start	R20UT0975EJ0100
CubeSuite+ V1.00.00 78K0 Design	R20UT0546JJ0100
CubeSuite+ V1.00.00 78K0R Design	R20UT0547JJ0100
CubeSuite+ V1.02.00 RL78 Design	R20UT0976JJ0100
CubeSuite+ V1.00.00 V850 Design	R20UT0549JJ0100
CubeSuite+ V1.01.00 78K0 Debug	R20UT0731EJ0100
CubeSuite+ V1.01.00 78K0R Debug	R20UT0732EJ0100
CubeSuite+ V1.02.00 RL78 Debug	R20UT0978JJ0100
CubeSuite+ V1.01.00 V850 Debug	R20UT0734EJ0100
CubeSuite+ V1.02.00 RX Debug	R20UT1143JJ0100
CubeSuite+ V1.02.00 Analysis	R20UT0979EJ0100
CubeSuite+ V1.02.00 Message	R20UT0980EJ0100

Chapter 3. Key Points for Selecting Uninstallation Method

There are two ways to uninstall this product.

- Use the integrated uninstaller (uninstalls CubeSuite+)
- Use separate uninstaller (uninstalls this product only)

To use the separate uninstaller, select the following from the Control Panel:

- Add/Remove Programs (Windows XP)
- Programs and Features (Windows Vista / Windows 7)

Then select "CubeSuite+".

Chapter 4. Changes

This chapter describes changes from V1.02.00 to V1.02.01

4.1 Addition of debug tool functions

4.1.1 New function for EQU defined bit registration and reference in watch pane

When using 78K0, the bit of the EQU definition can do registration and a reference in watch panel.

4.1.2 Extension of measurement time in AZ for 78K0R and RL78, when IECUBE trace function is used

When using IECUBE, the measurement time in AZ for 78K0R and RL78 is extended.

4.2 Removal of cautions

4.2.1 Note on Using CubeSuite+ Integrated Development Environments

Converting to CubeSuite+ project for V850, RX, and RL78 from a High-performance Embedded Workshop project for SH, R8C, M16C, H8SX, H8S, and H8 fails due to the following error

Error (E0202002)
Opening a project failed.

[Direct Error Cause]
A project using an unsupported toolchain was specified. : Toolchain name(E0292005)

4.2.2 Note on Using CubeSuite+ Integrated Development Environments for RX Family

(1) Problem with operating the library generator

In the Properties panel of CubeSuite+, if you make either of the following changes of options and then execute the build, the library generator does not operate

- In the Common options tab, you make a change to the value of the Address value of base register that sets the address value text box.
- In the Library generate option tab, you select the Enables string.h(C89/C99) property and click the drop-down arrow at the right end of the row (at the second column of the row, YES or NO has been selected) to open a drop-down list. Then you click YES or NO to make a change to the above selection.

(2) Problem with using Compile Options and Link Options (Build Options)

If you execute a build by using an option to optimize the accesses to external variables, incorrect code may be generated.

4.2.3 Note on Using Real time RAM monitor in on-chip debug for RL78

After on-chip debugging a system with RL78 by CubeSuite+ V1.01.01 or earlier, debug tool connection is failed with the following error when using Real time RAM monitor by CubeSuite+ V1.02.00.

Download failed.

[Direct Error Cause]
Extended monitor area is not blank [E1903128]

Chapter 5. Cautions

This section describes cautions for CubeSuite+.

5.1 Cautions for CubeSuite+ (general)

5.1.1 Cautions for file names

The following cautions apply to folder and file names.

- Folder and file names

Do not use folder or file names that cannot be created from Windows Explorer.

- Source file names, load module file names, and project file names

File names consist of the characters a-z, A-Z, 0-9, period (.), underscore (_), plus (+), and minus (-).

File names cannot end with a period (.).

File names cannot start with a period (.).

File names cannot start with a plus sign (+) or minus sign (-).

File names are case-insensitive.

File names may be up to 259 characters, including the path.

- File names other than the above.

File names comply with Windows conventions.

Note that the following characters cannot be used in file names.

\\ : * ? " < > | ;

File names cannot start with a period (.) or space.

The uppercase and lowercase characters of the file name are not distinguished.

File names may be up to 259 characters, including the path.

- Folder names

Comply with Windows file name conventions.

Note that the following characters cannot be used in file names. (Excluding RL78, 78K0, 78K0R, and V850 projects)

() , =

5.1.2 Caution for panel displays

If your hardware environment does not meet the recommended specifications for CubeSuite+, the [Property] panel may appear small, and the contents scrambled.

If this happens, move the [Property] panel outside the split panel area.

- Enable Dockable, and make it a Docking panel

- Enable Floating, and make it a Floating panel

5.1.3 Caution for User Account Control (UAC) function (Windows Vista)

If the UAC function is disabled on Windows Vista / Windows 7, then if a user without administrator privileges creates a project, and no Device Dependence Information is installed, then the installation of the Device Dependence Information will begin, but the installation will fail. If the UAC function is disabled, create projects after logging in with administrator privileges.

5.1.4 Caution for command accelerators included in split panels/categories

Although accelerators are displayed in the command menus of split panels and categories, pressing the keyboard shortcuts will have no effect. Use the mouse to select menu items.

5.1.5 Caution for Windows update program

Your computer may suffer "blue screen" errors if you apply the KB2393802 patch published by Microsoft Corporation. If this error occurs, please apply the patch provided by your computer's manufacturer or other source.

5.1.6 Caution for Renesas Electronics real-time OS

If you use the real-time operation system for the RX family provided by Renesas Electronics, install CubeSuite+ to a folder path that does not contain parentheses. If you install it to the 64-bit version of Windows, it will be installed in the "Program Files (x86)" folder by default, and if the folder path includes parenthesis characters, it will result in an error.

5.1.7 Cautions for Editor panel

- When you change the active file using the File tab, the "Forward to Next Cursor Position" and "Back to Last Cursor Position" features may not work.
- The "Show whitespace marks" feature does not display double-byte spaces.
- The Page Setup dialog box is not available.
- Although there is a Copy button on the Print Preview toolbar, it cannot be used.
- Line numbers are not printed/displayed by the Print and Print Preview features. Coverage lines, address lines, event lines, and main lines are also not printed or displayed. If the Outline feature is enabled, both collapsed and expanded views are printed/displayed.
- By default, shortcut keys are not assigned to the "Forward to Next Cursor Position" and "Back to Last Cursor Position" commands. Assign shortcut keys if needed.
- The Outline feature does not support conditional compilation (e.g., "#if" and "#else"). Outlining assumes that there are no conditional-compilation expressions.

Example:

```
#if AA
void main(void) {
    int test=0;
#else
void main(int argc, char *argv[]) {
    Int test=1;
#endif
    test++;
}
sub()
{
}
```

In source code like the above, outlining will recognize the final "sub()" in the "main" function as the endpoint.

- Dragging and dropping a file into the client area of the editor does not open the file. Select the file to open in the Project Tree, and double click the file to open it.
- The Jump to Function feature will not jump to a static function defined in another file.
- The following cautions apply to editor, when the source file of the same name from which a folder is different was registered with a main project and a sub project, and downloading a load module of a main project and a sub project both.
 - The address of the main project is displayed on the file.
 - At jumping to a source file from disassembly code, the file registered with a main project is opened.
 - If the file is opened from whichever project s, only 1 file is opened.

5.1.8 Caution for conversion from PM+ to CubeSuite+ project.

CubeSuite+ can't read the CA850 project, if the project is made by PM+ V6.00/V6.10/V6.11 and is added a new Build Mode. It'll be as follows.

- 1)When Debug Build or Release Build is specified, information on the added Build Mode can't be read.
- 2)When the added Build Mode is specified, it'll be an error.

[Workaround]

Please read the project by more than PM+ V6.20 and save it.

5.1.9 Cautions for Debugging Tool Settings during Project Composition

When you create a project by composite the settings of another project, only the settings for the default debugging tool will be imported. In the RX family, however, internal processing is common to the emulator and simulator, so the settings are imported regardless of which debugging tool is selected.

5.1.10 Cautions for Online Help

If you close the online help while the Search tab is displayed, and you then display the online help again and display the Contents tab, the Coding and Build editions may disappear.

If this happens, close the online help with the Contents tab displayed, and then display the online help again.

5.1.11 Cautions for project conversion

When changed conversion device of a project in [project convert setting] dialog when opened High-performance Embedded Workshop/PM+/CubeSuite, return a value chosen in [king of project] to the value that an initial value appears.

For example:

A kind of a project is replaced by the top (for example, [Application]) when chooses a device again in the back.

5.1.12 Note on Conversion of High-performance Embedded Workshop Projects

If you attempt to load a High-performance Embedded Workshop project into the CubeSuite+ under certain conditions, an error may occur during conversion or building of the project.

- (1) Converting a High-performance Embedded Workshop project to make it compatible with the CubeSuite+ fails when any of the following conditions is satisfied.

- No toolchain from Renesas Electronics Corp. is selected for the project.
- The project contains no tps file that is used to set up the High-performance Embedded Workshop environment. (the tps file is automatically created when the project is opened through the High-performance Embedded Workshop). To avoid this problem, you should open the project through the High-performance Embedded Workshop once before starting conversion.
- The project contains multiple CFG files, each of which is used to set up the realtime OS from Renesas Electronics Corp.

- (2) Converting a High-performance Embedded Workshop project to make it compatible with the CubeSuite+ succeeds but building of the project leads to an error when any of the following conditions is satisfied.

- Placeholder \$(TCINSTALL) is used in the project.
\$(TCINSTALL) remains in the project even after conversion but the CubeSuite+ does not recognize \$(TCINSTALL). Placeholder \$(TCINSTALL) that has been used as a parameter for [Options] in the High-performance Embedded Workshop is simply passed to the CubeSuite+ and may cause an unintended result (e.g. an error) upon building of the project. For this reason, you should manually change \$(TCINSTALL) after converting the project.

- Placeholder \$(WORKSPDIR) is used in the project.
If you select a HEW project file (with extension hwp) in the CubeSuite+, it is automatically converted to "%ProjectDir%\.." (the directory above the project directory). An error may occur during building of the project if the workspace does not exist in the directory indicated by "%ProjectDir%\..".
For this reason, you should manually change "%ProjectDir%\.." after converting the project.
- A custom build phase is used in the project.
Since all custom phases are deleted upon conversion, an error may occur during building of the project that involved a file output created for a custom build phase in the High-performance Embedded Workshop.
After converting the project, register the custom build-phase command with the CubeSuite+ as a command to be executed before or after each phase as required.
- A custom placeholder is used in the project.
Custom placeholders are not converted because the CubeSuite+ does not recognize them. Any custom placeholder that has been used as a parameter for [Options] in the High-performance Embedded Workshop is simply passed to the CubeSuite+ and may cause an unintended result (e.g. an error) upon building of the project. For this reason, you should manually change the custom placeholder after converting the project.

5.1.13 Caution for Pack Functions

When rapid start is valid and packed project with CubeSuite+ V1.02.00 or V1.02.01 is trying to be executed, both packed CubeSuite+ rapid started CubeSuite+ are execute

Work-around:

Before executing packed CubeSuite+, you should exit CubeSuite+ that is already rapid-started in the notification area (system tray).

5.2 Cautions for Design Tool

5.2.1 Caution for changing packages

If you change the package name in the pin layout properties, the data input in the device top view and device pin list will be cleared.

5.2.2 Caution for saving projects

If you save a project that has sub-projects while the Device Top View or Device Pin List panel is open, then the device top view and device pin list of the last sub-project in the Project Tree will always appear.

5.2.3 Caution for saving projects

Applies to: 78K0 / 78K0R / RL78

A value of "Use on-chip debug" or "Set user option byte" on [Link Option] Tab may be different between on saving a project file and on reading a project file.

[Condition]

- 1) The log-in user's own .mtud file doesn't exist when reading a project file
 Example 1
 Another log-in user B read the project file that a log-in user A saved
 Example 2
 A log-in user A read a project after the user saved the project file and deleted .mtud file by intent.
- 2) The log-in user's own .mtud file exists and the cord generation panel is displayed on the forefront.

[Procedure]

After "read project file" or before "build", verify that the values are correct.

5.3 Cautions for Build Tool

5.3.1 Cautions for startup node

In the case of a CX project, the following warning will be output if an object module file (.obj) is registered to the startup node. Please ignore this warning.

W0560111: The same file is specified multiple times as an input file.

In the case of a multicore project, the following error will be output if an object module file (.obj) is registered to the startup node. Please register an assembler file (.asm) to the startup node.

F0560208: Symbol "xxx" has been defined more than once.

< ----- within 67 characters ----- >

5.4 Cautions for debugging tool

The below abbreviated names are used in this section.

OCD(Serial) : MINICUBE2, E1 Emulator(Serial), E20 Emulator(Serial)
 OCD(JTAG) : MINICUBE, E1 Emulator (JTAG), E20 Emulator (JTAG)

5.4.1 Caution for adding sub-projects

Applies to: All debugging tools, Common to all devices

Disconnect the debugging tool before adding a sub-project that handles a different device than the main project.

5.4.2 Caution for executing a boot swap

Applies to: Simulator/ OCD(JTAG)/ OCD(Serial), V850 / 78K0 / 78K0R / RL78

If a software break is set in a boot-swap area, then a break instruction will be written to the Flash ROM. For this reason, a break instruction will remain after the boot swap.

- OCD(JTAG)/ OCD(Serial) : Use a hardware break if you wish to set a breakpoint.
- Simulator : Don't use break point in this area.

5.4.3 Caution for executing programs in internal RAM area

Applies to: Simulator, V850

The following cautions apply when executing programs within the internal RAM (addresses from 0x0fff0000 to 0x0ffffeff) of the V850E/MA3 and other V850E microcontrollers.

- When CPU stops, the displayed address on disassemble panel is "0x03ff0000 to 0x03ffff".
- In the case using "Step-In" execution for the function in internal RAM, the actual operation become "Step -Over".

5.4.4 Caution for standby mode

Applies to: All debugging tools, V850 / 78K0 / 78K0R / RL78

If a forced break is performed while in standby mode (e.g. STOP mode or HALT mode), or an instruction to move to standby mode is made while in step execution, then behavior will differ between the simulator and the emulator (IECUBE, OCD(JTAG), and OCD(Serial)).

- Emulator: The forced break will release standby mode. In step execution, it will not go into standby mode.
- Simulator: The forced break will not release standby mode. In step execution, it will go into standby mode.

In either case, the program counter (PC) row upon forced break will break at the next instruction after the standby mode instruction (e.g. HALT). Thus in the case of the simulator, it will appear that standby mode has been released. Check the status bar to see if standby mode has been released. If the simulator is in standby mode, "Halt" or "Standby" will appear in the status bar.

5.4.5 Caution on low-power consumption modes

Applies to: All debugging tools, RX

When a forced break occurs in a low-power consumption mode (e.g. sleep, stop, or standby) or an instruction that makes the CPU enter a low-power consumption mode is executed during stepped execution, the behavior of the simulator and the emulator will differ as follows.

- Emulator: A forced break releases the CPU from the low-power consumption mode. On the other hand, the CPU can enter a low-power consumption mode during stepped execution.
- Simulator: Transition to a low-power consumption mode (e.g. by a register setting) is not supported. Executing a WAIT instruction causes a break, with the PC placed at the address of the next instruction. During stepped execution, the CPU also does not enter a low-power consumption mode and the PC is placed at the address of the next instruction.

5.4.6 Caution for multipliers/dividers

Applies to: Simulator, 78K0

When simulating 78K0 instructions, the multiplier and divider are not supported. For this reason, to perform multiplication or division within the program, from the build tool, open the Property panel, and on the [Compiler Options] tab, from the [Use multiplier/divider] drop-down list, select [No].

5.4.7 Caution for Memory bank function

Applies to: Simulator, 78K0

When simulating 78K0 by instruction mode, the memory bank function is not supported.

5.4.8 Caution for CPU operation clock

Applies to: Simulator, 78K0R / RL78

- When simulating 78K0R by instruction mode, the frequency of on chip oscillator is 8MHz.
- When simulating RL78 by instruction mode, the CPU operation clock operates by the specification of RL78/G13

5.4.9 Caution for Multiplier and Divider/Multiply-Accumulator

Applies to: Simulator, 78K0R / RL78

When simulating 78K0R or RL78 by instruction mode, cautions of Multiplier and Divider/Multiply-Accumulator are following.

- (1) When using it by division mode, the division processing will be finished in by 1 clock.
- (2) When using it by division mode, the interrupt "INTMD" (the end of division operation) is not occurred. But DIVST bit of Multiplication/Division Control Register "MDUC" is changed. (DIVST bit displays division operation status.)

5.4.10 Caution for traces in arbitrary intervals

Applies to: Simulator, all devices

If you perform a trace from a trace start event until a trace end event, the simulator will not display the trace end event as the results of the trace. For this reason, if you are using a simulator, set the trace end event to one line below the range that you wish to display as the trace data.

5.4.11 Caution for runtime measurement over arbitrary intervals

Applies to: Simulator, V850 / 78K0 / 78K0R / RL78

If you measure run time from a timer start event until a timer end event, the simulator will not include the time for the timer end event in the measurement results. For this reason, if you are using a simulator, set the timer end event to one line below the range for which you wish to measure the run time.

5.4.12 Caution for CPU operation clock

Applies to: Simulator, V850

The clock generator is not simulated in V850 instruction simulation mode. For this reason, the CPU operation clock will always have the main clock frequency set in the Properties panel (even if a built-in peripheral I/O register with a clock generator is manipulated, the CPU's operation clock will not change)

5.4.13 Caution for displaying maximum address space in memory display panel

Applies to: OCD(Serial) / IECUBE, 78K0

To access the device maximum internal ROM, internal fast RAM, or internal extended RAM sizes from the memory panel or the like, set a hook process in the memory size switch register (IMS) and internal extended RAM size switch register (IXS).

5.4.14 Caution for retuning execution and displaying the call stack

Applies to: All debug tools, 78K0R / RL78

If step execution is performed from the editor panel (in source mode), the debugging tool determines whether an interrupt is being processed via the NP, EP, and ID flags in the POWER switch register. For this reason, if the above flags or register are changed (e.g. when using multiple interrupts), then the return execution and displaying the call stack may be incorrect.

5.4.15 Software breakpoints when ROMification has been performed

Applies to: All debug tool, RL78 / V850

If there is code (text sections) to be ROMified, any breakpoint instructions set for that code will be deleted during rcopy. For this reason, no break will occur. Use a hardware break if you are using OCD(JTAG) or OCD(Serial) or IECUBE. Note that if you are using a simulator, execution will not break even if a hardware breakpoint is used, but it will break if the tracer or timer is turned on.

5.4.16 Adding sub-projects

[Applies to] Common to all debug tools and devices

If you add a sub-project while a debugging tool is connected, downloading and the like may fail. Add sub-projects while the debugging tool is disconnected.

5.4.17 Configuring Flash options

[Applies to] OCD(JTAG), V850E2M

The bit shown below indicating the following Flash option has been locked to 1. Use a Flash programmer if you wish to write 0 to it.

- Bit 95 of on-chip debugging security ID (security lock signal release)
- Bit 31 of option byte 0 (debug interface connection disabled bit)

5.4.18 Caution for Stack-trace display

[Applies to] All debugging tools, 78K0

The stack-frame display function may fail to correctly display up to the **main** function if a function is used that does not push the frame pointer (HL) onto the stack (e.g. **noauto** or **norec** function), or if the memory bank is used.

Additionally, a free-run state may occur if a return is executed from a function that does not push the frame pointer (HL) onto the stack (e.g. **noauto** or **norec** function), or if a memory bank function is used.

5.4.19 Caution for stepping into main bank

[Applies to] All debugging tools, 78K0

If you step into a user-defined library function or function without debugging information in the memory bank at the source level, execution will break in the bank-switching library.

5.4.20 Caution for local-variable display

[Applies to] All debugging tools, 78K0

Local variables outside the scope of the current PC are not displayed correctly in the stack trace panel.

5.4.21 Caution for disassemble window

[Applies to] All debugging tools, 78K0

When displaying instructions in the common area in the disassemble window, if the displayed instruction uses a symbol in the memory bank area, a symbol from a different bank may be displayed.

5.4.22 Breakpoint and other settings become invalid

[Applies to] Common to all debugging tools and devices

If you differentiate function or variable names by leading underscores, then the debugger may misrecognize them, and convert symbols or make breakpoint settings invalid.

This applies for cases like when you have two functions, one named `_reset` and the other named `__reset`.

5.4.23 Caution for conflicting breakpoints

[Applies to] IECUBE/ OCD(JTAG)/ OCD(Serial), V850

If there is a conflict between a software breakpoint and one of the following hardware breakpoints, the PC value may be invalidly corrected. Use a hardware break instead of a software break.

- (1) Trace full break
- (2) Non-map break
- (3) Write-protect break
- (4) Illegal I/O access break
- (5) Forced break due to Stop button press
- (6) Event break (hardware break)
- (7) Timeout break

5.4.24 Simulating with V850E2

[Applies to] Simulator, V850E2

The instruction simulator for V850E2 supports as following functions. Other functions are not supported.

- CPU instruction
- Exceptions
- System register protection
- Memory protection
- Timing supervision function
- Floating-point operation function

Be sure to following notes.

- (1) The access to external memory area is not supported
- (2) A simulation result of the floating-point unit [FPU] has a margin of errors compared to real devices. The simulator uses the floating-point library of Visual C++, and store a result calculated by 80bit in a register.
- (3) Following exception is not supported.
System error exception, Memory error exception
- (4) The simulation of cache memory is not supported.
- (5) The instructions (SYNCE/SYNCM/SYNCP) are not supported. If these were executed, the operation is same as NOP execution.
- (6) The operation clock of CPU is always 4MHz. The setting of main clock on property panel is ignored.
- (7) It is impossible to use data flash area. If CPU access this area, CPU breaks and error is happen.
- (8) The value of Option byte storage register "OPBT0" is always "0".
- (9) EH_RESET register features are not supported. In the case of a CPU reset, the reset address will always be "0x0".
- (10) The number of execution clocks of each instruction will be the number of execution clocks when another instruction is executed immediately after that instruction is executed.

5.4.25 Caution on two or more variables with the same name

Applies to: All debug tools, RX

When two or more variables with the same name are defined in unnamed name spaces written in different source files, the Watch panel only shows the information on the variable that was found first.

5.4.26 Caution on member-variable pointers

Applies to: All debug tools, RX

After the member-variable pointer "mp1" defined in the program below is registered with the Watch and Local Variables panels, the type of the pointer is indicated as "int **", not "int Foo::*".

```
class Foo {
    int m1;
};
int Foo::*mp1 = &Foo::m1;
```

5.4.27 Caution on unions assigned to registers

Applies to: All debug tools, RX

When a union is assigned to a register, it is assumed that the members of the union are assigned to the lower-order bytes of the register. For this reason, the values of the members in big endian being displayed are incorrect.

5.4.28 Caution on functions with the same name and char-type parameters

Applies to: All debug tools, RX

When three functions with char-type parameters are defined as shown below, the address of "Func(signed char)" is not displayed (i.e. the address of "Func(char)" is displayed instead).

```
void Func(char);
void Func(signed char);
void Func(unsigned char);
```

5.4.29 Caution on char-type one-dimensional arrays

Applies to: All debug tools, RX

When a char-type one-dimensional array is assigned to multiple locations in registers or memory as shown below, no character string will be displayed in the value column of the Watch or Local Variables panel even after the array "array" has been registered with the panel. (" " is shown in the column.)

```
char array[5] = "ABCD";
```

5.4.30 Caution on changing the priority section of overlay sections

Applies to: All debug tools, RX

Changing the priority section of overlay sections does not immediately reflect the debugger operation. To update the display of addresses in the editor, for example, you need to close the file and open it again. To update the display of variables in the Watch panel, single step in the program.

5.4.31 Caution on variables assigned to registers

Applies to: All debug tools, RX

When the selection for [Scope] in the Local Variables panel is not "Current", the values of variables assigned to registers are not displayed correctly. Editing these values is also not possible.

5.4.32 Caution on the locations where variables are assigned

Applies to: All debug tools, RX

When a defined variable satisfies both of the two conditions given below, the location of its member variables indicated in the Watch and Local Variables panels is actually the location of the entire variable.

Conditions:

(1) The variable is assigned to two or more addresses or registers (i.e. two or more addresses or registers are displayed in the address column).

(2) A structure-, class-, array-, or union-type member is defined in the variable.

Example:

```
struct Mem {
    long m_base;
};
struct Sample {
    long m_a;
    struct Mem m_b;  <- Condition (2)
};

main () {
    struct Sample obj;
}
```

Display in the Watch and Local Variables panels:

"obj"		-	{ R1:REG, R2:REG }	(struct Sample)
L m_a	0x00000000		{ R1:REG }	(long)
L m_b	-		{ R1:REG, R2:REG }	(struct Base)
L m_base	0x00000000		{ R2:REG }	(long)

5.4.33 Caution on casting variables

Applies to: All debug tools, RX

In the Watch panel, class-type variables cannot be cast into base classes or derived classes. Also, structure- or union-type variables cannot be cast into any other type.

```
class AAA [
    int m_aaa;
} objA;
class BBB : public AAA { // BBB inherits AAA.
    int m_bbb;
} objB;
class CCC { // CCC does not inherit AAA.
    int m_ccc;
} objC
```

```
class AAA* pa = objA;
class BBB* pb = objB;
class CCC* pc = objC;
```

"(AAA*)pa"	Available
"(BBB*)pb"	Available
"(AAA*)pb"	Not available due to a restriction although class BBB inherits AAA pointed to by pb.
"(CCC*)pc"	Available
"(AAA*)pc"	Not available because class CCC does not inherit AAA pointed to by pc.

5.4.34 Caution on memory panel while using RRM function

Applies to: E20 Emulator(JTAG),RX

If the value of the variable that was set in the watch panel by using RRM function is displayed in a memory panel, the memory panel displays not "" but "00".

Use Watch panel instead of Memory panel.

5.4.35 Caution on hardwarebreak

Applies to: OCD(JTAG),OCD(Serial),RX

If it uses [Go to Here] or [Execute to the specified symbol after CPU Reset],and a hardware break is set while running a program, the execution does not stop by break.

Please do not set hardware break during execution.

5.4.36 Caution on sleep state of PC

Applies to: OCD(JTAG), OCD(Serial),RX

When PC shifts to a sleep state using Windows Vista or Windows 7, Cubeuite+ cannot be continued to debug after a return from sleep state.

Please set up PC, in order not to shift to a sleep state.

5.4.37 Caution on the trace stop and restart during program execution

Applies to: All debug tool,RX

When the trace start event or the end event set up, the trace can not stop and restart during the program execution.

5.4.38 Caution on the timestamp of trace

Applies to: OCD(JTAG), OCD(Serial),RX

When the time between frames exceeded a trace counter (20 bits), and when trace type was lost,the time stamp of trace does not display the right time.

5.4.39 Flash self emulation

Applies to: IECUBE, V850

If you perform Flash self programming on the IECUBE, check whether the Flash functions shown below can be emulated, and check the related cautions.

Flash self programming Type 01

Flash Function	Functional Outline and Restriction	Availability of Emulation
FlashEnv	Flash environment initialization/end function	Emulated
FlashBlockErase	Block erasure function	Emulated
FlashWordWrite	One word writing function Restriction: If an address in the guard area is specified as the third argument, a fail-safe break occurs at an unexpected address.	Restricted
FlashBlockVerify	Block internal verify processing function	Emulated
FlashBlockBlankCheck	Block blank check function	Emulated
FlashGetInfo	Flash information acquisition function	
	Option = 2: CPU number and total number of blocks held by CPU Restriction: The device name (four-digit number) set in the Configuration dialog box is returned as the CPU number.	Restricted
	Option = 3: Security information	Emulated
	Option = 4: Acquisition of boot area swapping information Restriction: Boot area swapping information is not reflected.	Restricted
	Option = 5 + Block number: Acquisition of last address of block	Emulated
FlashSetInfo	Flash information setting function Restriction: The boot area swapping setting is ignored.	Restricted
FlashStatusCheck	Function for checking operation status of flash function that was executed most recently Restriction: For FlashBlockErase and FlashBlockBlankCheck, the timing at which the return value changes from FE_BUSY to FE_OK differs from that in the actual device.	Restricted
FlashBootSwap	Boot area block swapping function	Not emulated
FlashSetUserHandler	User interrupt handler registration function	Emulated
FlashFLMDCheck	FLMD0 pin status check function	Emulated
FlashSetInfoEx	Flash information setting function Restriction: The boot area swapping setting is ignored.	Restricted
FlashNWordRead	Function for reading N words Restriction: If an address in the guard area is specified as the third argument, a fail-safe break occurs at an unexpected address.	Restricted

Flash self programming Type 02c

Flash Function	Functional Outline and Restriction	Availability of Emulation
FlashEnv	Flash environment initialization/end function	Emulated
FlashBlockErase	One block erasure function	Emulated
FlashWordWrite	One word writing function Restriction : If an address in the guard area is specified as the third argument, a failsafe break occurs at an unexpected address.	Restricted
FlashBlockIVerify	One block internal verify processing function	Emulated
FlashBlock-BlankCheck	One block blank check function	Emulated
FlashGetInfo	Flash information acquisition function	
	Option = 2 : CPU number and total number of blocks held by CPU Restriction : The device file name (four-digit number) is returned as the CPU number.	Restricted
	Option = 3 : Security information	Emulated
	Option = 4 : Acquisition of boot area swapping information Restriction : Boot area swapping information is not reflected.	Restricted
	Option = 5 + Block number : Acquisition of last address of block	Emulated
FlashSetInfo	Flash information setting function Restriction : The boot area swapping setting is ignored.	Restricted
FlashBootSwap	Boot area block swapping function	Not emulated
FlashFLMDCheck	FLMD0 pin status check function	Emulated
FlashWordRead	Data reading function Restriction : If an address in the guard area is specified as the third argument, a failsafe break occurs at an unexpected address.	Restricted

Flash self programming Type 03

Flash Function	Functional Outline and Restriction	Availability of Emulation
FlashEnv	Flash environment initialization/end function	Emulated
FlashBlockErase	Block erasure function	Emulated
FlashWordWrite	One word writing function Restriction: If an address in the guard area is specified as the third argument, a fail-safe break occurs at an unexpected address.	Restricted
FlashBlockIVerify	Block internal verify processing function	Emulated
FlashBlockBlankCheck	Block blank check function	Emulated
FlashGetInfo	Flash information acquisition function	
	Option = 2: CPU number and total number of blocks held by CPU Restriction: The device file name (four-digit number) is returned as the CPU number.	Restricted
	Option = 3: Security information	Emulated
	Option = 4: Acquisition of boot area swapping information Restriction: Boot area swapping information is not reflected.	Restricted
	Option = 5 + Block number: Acquisition of last address of block	Emulated
FlashSetInfo	Flash information setting function Restriction: The boot area swapping setting is ignored.	Restricted
FlashBootSwap	Boot area block swapping function	Not emulated
FlashFLMDCheck	FLMD0 pin status check function	Emulated
FlashWordRead	Data reading function Restriction: If an address in the guard area is specified as the third argument, a fail-safe break occurs at an unexpected address.	Restricted
FlashIVerify	Internal verify function (for EEPROM)	Not emulated
FlashBlankCheck	Blank check function (for EEPROM)	
EEPROM_Init	EEPROM area initialization function (for EEPROM)	
EEPROM_Write	EEPROM write function (for EEPROM)	
EEPROM_Read	EEPROM read function (for EEPROM)	
EEPROM_Copy	EEPROM copy function (for EEPROM)	
EEPROM_VChK	EEPROM valid area check function (for EEPROM)	
EEPROM_Erase	EEPROM erase function (for EEPROM)	

Flash self programming Type 04

Flash Function	Functional Outline and Restriction		Availability of Emulation
FlashInit	Self library initialization function		Emulated
FlashEnv	Flash environment initialization/end function		Emulated
FlashBlockErase	One block erasure function		Emulated
FlashWordWrite	One word writing function		Emulated
FlashBlockVerify	One block internal verify processing function		Emulated
FlashBlockBlankCheck	One block blank check function		Emulated
FlashGetInfo	Flash information acquisition function		
	Option = 2	Device information (total number of blocks and device number)	Emulated
	Option = 3	Security flag, last block number of boot block	Emulated
	Option = 4	Device information	Emulated
	Option = 5	Reset vector address	Emulated
	Option = 6 +block number n	Last address of block number <i>n</i>	Emulated
FlashSetInfo	Flash information setting function Restriction: Nothing but information setting is performed. The boot area swapping setting is ignored.		Restricted
FlashStatusCheck	Checking of flash function operation that was performed last Restriction: SELFLIB_BUSY is not returned.		Restricted
FlashBootSwap	Boot area block swapping function Restriction: Functions can be called but boot swapping is not executed.		Not emulated
FlashFLMDCheck	FLMD0 pin status check function		Emulated

The cautions on performing flash self programming are described below.

No.	Description
1	Flash memory self programming emulation is not enabled in the following cases. (A) When the internal ROM size is not the default size [Workaround] Set the internal ROM size to the default value in the Configuration dialog box. (B) When using two "break before execution" [Workaround] Disable or delete one "break before execution".
2	When flash memory self programming emulation is enabled, the following restrictions are applied to the debug function. (A) The internal ROM and internal RAM sizes cannot be changed. (B) The DMM and pseudo RRM functions are disabled. (C) An illegal break occurs in the program if the SP register value is 0 (not pointing to the internal RAM). If a break such as an event occurs before the SP register value is initialized to point to a relevant location (such as internal RAM), then it causes an illegal break for the stack area. If there is a possibility that such a break will occur during this period, set a relevant value to SP before executing the program. (D) An illegal break may occur if the restriction shown below applies to the IECUBE used. Clear the Non Map check box for the Internal RAM in the Fail-safe break setting dialog box. - An illegal break occurs during program execution in internal RAM
3	When flash memory self programming emulation is enabled, the 4-byte area starting from address 0 is reserved, a 4-byte instruction <i>jr 0xfffd6</i> is written to address 0. Therefore, when using this function at a reset vector address 0, allocate a startup routine to the area starting from address 4. If flash memory self programming emulation is disabled, 0 is written to the four bytes area starting from address 0. Do not describe codes in which execution branches to address 0, even if this function is used as a reset vector address 0. It is recommended to perform description as shown below in order to operate the same program as the one generated by emulation, in the actual device. <pre> # RESET handler (in the case of address 0) .section "RESET", text jr __start -- Overwritten by jr 0xfffd6 jr __start </pre>
4	If address 0 is specified as the reset vector handling specification address, the reset vector is set to address 4. If an address other than address 0 is specified, then the specified address is set as the reset vector without incrementing the value by four.
5	Regarding the operation of FlashStatusCheck() after FlashBlockErase() and FlashBlockBlankCheck() during emulation, the timing at which the return value of FlashStatusCheck() changes from FE_BUSY to FE_OK differs from that in the actual device.

No.	Description
6	If the address specified as the third argument of FlashWordWrite, FlashWordRead, or FlashNWordRead is located in the guard area, then an illegal memory address is accessed, and a fail-safe break occurs at an unexpected address. Correct the address to a relevant one for FlashWordWrite, FlashWordRead, or FlashNWordRead.
7	To enable the settings made in the Flash Option dialog box, be sure to reset the CPU and reexecute the program; otherwise, the setting may not take effect.
8	Secure a stack area of at least 84 (54H) bytes for the debugger's workspace. The debugger consumes a stack area of at least 84 (54H) bytes when a break occurs or during emulation processing of flash memory writing. When interrupts are enabled, a stack area of another 84 (54H) bytes is required as the debugger's workspace. If multiple interrupts are enabled, a stack area of 84 (54H) bytes must be secured per stage.
9	The data in the internal RAM is corrupted after a CPU reset. Normally, the internal RAM data after reset is not guaranteed in the actual device, but note that the operation may vary.
10	If a flash function is not used in accordance with the specifications or an unsupported flash function is called, "1" is returned.
11	The following restrictions apply to emulation of Type4. (1) An area of 48 bytes from the internal RAM end address is reserved for use by the debugger. (2) When using a device with a 1 MB internal flash memory, the internal flash area starting from address 0xFF300 or higher will be used by the debugger. (3) If a flash function is executed stepwise in assemble mode, the debugger code for emulation will be executed, which is different from the code actually executed by the device. During debugging, therefore, perform stepwise execution in source mode.

5.5 Cautions for analysis tool

5.5.1 Cautions for Function List panel (CC-RX (C++ language))

- The following cautions apply to template functions and member functions defined in template classes.
 "(No Definition)" will appear in the "File Name" column.
 Only the types will be displayed in the Arguments column. The argument names will not be displayed.
 A hyphen ("-") will appear in the "Start Address" and "End Address" columns of member functions defined in template classes.
 If a hyphen ("-") appears in the "Start Address" column, you will not be able to jump to the Editor panel, Disassemble panel, or Memory panel.
 The Find All References menu command does not display the locations of definitions. Information about the referencing functions and variables is also not displayed.
 This feature does not count the number of function references in template functions and member functions defined in template classes. Similarly, reference information does not appear in the "Find All References" menu command.
 It is not possible to set breakpoints at the start of member functions defined in template classes from the "Set Break to Function" menu command.
- If a member function defined in a class declaration is only declared and not used, the filename will not be displayed. It will be treated as a function with no defined location.
- If you specify a function parameter with a class type, a hyphen ("-") will be displayed in the "Start Address," "End Address," and "Code Size" columns.

- If you define a function with an argument of type signed char, and an overloaded function with an argument of type char, a hyphen ("-") will be displayed in the "Start Address," "End Address," and "Code Size" columns.

5.5.2 Cautions for Variable List panel (CC-RX (C++ language))

- This feature does not display static variables defined in template functions or member functions defined in template classes.
- This feature does not count the number of variable references in template functions and member functions defined in template classes.
- The compiler changes the types of const variables without an extern/volatile declaration to constants. As a result, they will not appear in the Variable List as variables.
- Global variables with the same name defined in anonymous namespaces in different files will be treated as having the same type.

5.5.3 Cautions for Call Graph panel (CC-RX (C++ language))

- By default, template functions and member functions defined in template classes do not appear in the Call Graph panel. To display them, set the "Display the function without definition at Call Graph panel" property to "Yes."
- Functions called from/variables referenced from template functions and member functions defined in template classes do not appear in the Call Graph panel.

5.5.4 Cautions for Class/Member panel (CC-RX (C++ language))

- The following cautions apply to template functions and member functions defined in template classes.
You cannot jump to the defined location using the Jump to Source menu command.
You cannot jump to the location of declaration in the source using the Jump to Declaration of Source menu command.
- Namespace aliases are not displayed.

5.5.5 Cautions for Variables panel

- The addresses and sizes of anonymous structs and unions cannot be displayed.
- Size information of variables that are defined only and not used will be eliminated by compiler optimization, and 0 will appear in the Size column. [CC-RX]

5.5.6 Cautions for Class/Member panel

- The "Define Macros and Constants" node is not displayed. [CA850]
- It is not possible to jump to the type-definition location from the struct, union, or enumeration nodes. It is not possible to jump from a member node to the source code where the member is defined in the case of structs and unions. Members are not displayed for enumerations. [CX]
- If you select the member of an enumerated type and jump to the source, it will jump to the location where the enumeration is defined. [CC-RX]

5.6 Cautions for Python Console

5.6.1 Caution for Japanese input

The Japanese input feature cannot be activated from the Python Console. To enter Japanese text, write it in an external editor or the like, and copy and paste it into the console.

5.6.2 Caution for prompt displays

The Python Console prompt of ">>>" may be displayed multiply, as ">>>>>>", or results may be displayed after the ">>>", and there may be no ">>>" prompt before the caret. If this happens, it is still possible to continue to enter functions.

5.6.3 Cautions for paths to folders and files

IronPython recognizes the backslash character (\) as a control character. For example, if a folder or file name starts with a "t", then the sequence "\t" will be recognized as a tab character. Please use [r + "path name"] to avoid this.

Example: r"c:\test\test.py"

A forward slash (/) can be used instead of a backslash (\). But some functions occur the error by using a backslash. Therefore, please don't use a backslash.

5.6.4 Caution for executing scripts for projects without load modules

If a script is specified in the startup options that uses a project without a load module file, or if project_file.py is placed in the same folder as the project file, then although the script will be executed automatically after normal project loading, it will not be executed if there is no load module file.

5.6.5 Cautions for forced termination

If the following operations are performed while a script like an infinite loop is running, then the results of function execution may be an error, because the function execution will be terminated forcibly.

1. Forcible termination by selecting "Forcibly terminate" from the context menu or pressing Ctrl+D in the Python Console
2. Changing the active project in a project with multiple projects

5.6.6 Caution for forced stops

Executing "Abort" from the context menu will forcibly stop an executing script or function, but hook and callback functions that had not started at the time the "Abort" was executed will execute in order afterward.

5.6.7 Cautions on using the RX (simulator and E1/E20 emulator)

debugger.Assemble.LineAssemble() function does not support Big Endian, so this function does not operate correctly when using it for Big Endian area.

5.6.8 Cautions on using "CubeSuiteExit()" function and Rapid start function

When closing CubeSuite+ by using "CubeSuiteExit()" function, CubeSuite+ does not become a resident program even if Rapid start function was used.

Chapter 6. Restrictions

This section describes restrictions for CubeSuite+.

6.1 Restrictions for debugging tool

The below abbreviated names are used in this section.

OCD(Serial) : MINICUBE2, E1 Emulator(Serial), E20 Emulator(Serial)

OCD(JTAG) : MINICUBE, E1 Emulator (JTAG), E20 Emulator (JTAG)

6.1.1 List of restrictions for debugging tool

No.	Target tool	Target device	Description
1	OCD(JTAG)	V850E2M	Restriction for Flash options
2	IECUBE	78K0R/Kx3	Restriction for Flash self emulation settings

6.1.2 Details of restrictions for debugging tool

No.1 Restriction for flash options

Applies to: OCD(JTAG), OCD(Serial), V850E2M

Description: All security settings and boot-block cluster settings of the Flash Option Settings property are invalid.

Workaround: There is no workaround

No.2 Restriction for Flash self emulation settings

Applies to: IECUBE, 78K0R/Kx3

Description: If opening project file which made by CubeSuite+ V1.00.02 (or older) or CubeSuite, The connection to IECUBE is failed and following error message is displayed.
"Error : Illegal the flash self emulation settings.[E062206]"

Workaround: Change "setting of Flash self end block value" from "FF" to "FFFF" (in Flash self emulation setting tab on property of debug tool) before connecting IECUBE.

And overwrite project file after connecting IECUBE. (No need to change this setting in next time to connect IECUBE by this overwriting.)

Chapter 7. Changes in User's Manual

This section describes Changes in User's Manual in CubeSuite+.

7.1 Modifications of Editor







This section describes additional functions of the editor. The editor is described in the Coding and Debugging editions.

7.1.1 Additional description of toolbar

[Before addition]

None

[After addition]

	<p>This toggles between normal and mixed display modes when the debugging tool is connected.</p> <p>In normal display mode, the source program is displayed.</p> <p>In mixed display mode, the source program and assembly code are displayed.</p>
	<p>This toggles between step execution at the source and assembly level when the debugging tool is connected and in mixed display mode.</p> <p>At the source level, the program counter indicates the line in the source program.</p> <p>At the assembly level, the program counter indicates the line in the assembly code.</p>
	<p>This displays the current position of the program counter (PC) when the debugging tool is connected and in mixed display mode.</p>
	<p>This jumps to the position before executing the Back to Last Cursor Position command.</p>
	<p>This returns to the position before executing the Jump to Function command.</p>
	<p>This toggles between showing and hiding the column. Clicking this displays the areas that can be toggled.</p>

7.1.2 Additional description of dragging and dropping

[Before addition]

None

[After addition]

You can add symbols to Watch panel or Analysis Chart panel by dragging and dropping them.

7.1.3 Additional description of mixed display

[Before addition]

None

[After addition]

The mixed display feature displays the source program and assembly code together; in normal display mode, only the source program is displayed.

You can set mixed display from the toolbar.

Note the following when using mixed display mode.

- Editing is not available.
- Files cannot be saved.
- Features that change a files content, such as cut and paste, delete, redo, replace, outline, and indent, are not available.
- The select all feature is not available.

7.1.4 Additional description of toolbar

[Before addition]

None

[After addition]

Normally, if the program counter moves between multiple files when debugging (e.g., when performing step-by-step execution), each of the source files would be opened in a separate editor panel. Recycle mode lets you display multiple source files in a single editor panel.

You can configure recycle mode from [Tools] – [Options] – [Text Editor].

7.1.5 Change to description of split bar

[Before addition]

You can split the Editor panel display by using horizontal and vertical split bars within the view. This panel can be split up to **four times**.

[After addition]

You can split the Editor panel display by using horizontal and vertical split bars within the view. This panel can be split up to **two times vertically, and two types horizontally**.

7.2 Modifications in Start

This section describes modifications in CubeSuite+ V1.02.00 Start User's Manual (document # R20UT0975EJ0100).

7.2.1 Additional description of utility to accelerate startup times

[Before addition]

None

[After addition]

The utility to accelerate startup times speeds up the launch of CubeSuite+ when the Rapid Start feature is not in use.

Run "AccelerationUtility.exe" (located in the same folder as the CubeSuite+ executable, and click "Accelerate Startup". The default installation folder is as follows.

C:\Program Files\Renesas Electronics\CubeSuite+

Note: The effectiveness of this utility will vary depending on your computer.

7.2.2 Addition of the list of Python class

[Location] Page 411

[Before addition]

Class Name	Function Description
BreakCondition	This class creates a break condition.
BuildCompletedEventArgs	This class holds the parameters when a build completes.

[After addition]

Class Name	Function Description
BreakCondition	This class creates a break condition.
BuildCompletedEventArgs	This class holds the parameters when a build completes.
BreakpointInfo	Break Point Information
DisassembleInfo	Disassembly Information
DownloadInfo	Download Information
MapInfo	Map Information
StackInfo	Stack Information
TraceInfo	Trace Information
XRunBreakInfo	XRunBreak Information
XTimeInfo	Timer Information

7.2.3 Additional description of Python class

[Location] Page 415

[Before addition]

None

[After addition]

BreakpointInfo

Information about break point (return value of debugger.Breakpoint.Information)

[Type]

```
class BreakpointInfo:
    Number = 0
    Name = None
    Enable = True
    BreakType = BreakType.Hardware
    Address1 = None
    Address2 = None
    Address3 = None
    Address4 = None
```

[Variable]

Variable	Description
<i>Number</i>	The event number.
<i>Name</i>	The name of the break point.
<i>Enable</i>	True if enabled. False if disabled.
<i>BreakType</i>	The BreakType. See "BreakCondition" for details about BreakType.
<i>Address1</i>	Address information 1 as a string.
<i>Address2</i>	Address information 2 as a string. (Only for combined breaks)
<i>Address3</i>	Address information 3 as a string. (Only for combined breaks)
<i>Address4</i>	Address information 4 as a string. (Only for combined breaks)

[Detailed description]

BreakpointInfo has a class format. It is passed as the return value from the debugger.Breakpoint.Information function.

[Example of use]

```
>>>info = debugger.Breakpoint.Information()
    1 Break0001 Enable test1.c#_main+2
    2 Break0002 Disable test2.c#_sub4+10
>>>print info[0].Number
1
>>>print info[0].Name
Break0001
>>>print info[0].BreakType
Hardware
>>>print info[0].Enable
True
>>>print info[0].Address1
test1.c#_main+2
>>>print info[0].Address2
None
>>>print info[1].Number
2
>>>print info[1].Name
Break0002
>>>print info[1].BreakType
Hardware
>>>print info[1].Enable
False
>>>print info[1].Address1
test2.c#_sub4+10
>>>print info[1].Address2
None
>>>
```

DisassembleInfo

Disassembly Information (return value of debugger.Assemble.Disassemble)

[Type]

```
class DisassembleInfo:  
    Address = 0  
    Code = None  
    Mnemonic = None
```

[Variable]

Variable	Description
<i>Address</i>	The address.
<i>Code</i>	Code information as a collection of bytes.
<i>Mnemonic</i>	Mnemonic information.

[Detailed description]

DisassembleInfo has a class format. It is the structure of the return value from the debugger.Assemble.Disassemble function.

[Example of use]

```
>>>info = debugger.Assemble.Disassemble("main", 4)      # disassemble command
0x000002DC    B51D    br _main+0x36
0x000002DE    0132    mov0x1, r6
0x000002E0    60FF3800  jarl _func_static1, lp
0x000002E4    63570100  st.w r10, 0x0[sp]
>>>print info[0].Address
732
>>>print info[0].Code[0]
181
>>>print info[0].Code[1]
29
>>>print Mnemonic
br _main+0x36
>>>print info[3].Address
740
>>>print info[3].Code[0]
99
>>>print info[3].Code[1]
87
>>>print info[3].Code[2]
1
>>>print info[3].Code[3]
0
>>>print info[3].Mnemonic
st.w r10, 0x0[sp]
>>>
```

DownloadInfo

Download information (return value of debugger.Download.Information)

[Type]

```
class DownloadInfo:
    Number = None
    Name = None
    ObjectDownload = True
    SymbolDownload = False
```

[Variable]

Variable	Description
<i>Number</i>	The download number.
<i>Name</i>	The filename.
<i>ObjectDownload</i>	True if the object information has been downloaded. False if it has not been downloaded.
<i>SymbolDownload</i>	True if the symbol information has been downloaded. False if it has not been downloaded.

[Detailed description]

DownloadInfo has a class format. It is the structure of the return value from the debugger.Download.Information function.

[Example of use]

```
>>>info = debugger.Download.Information()
1: DefaultBuild\sample.out
>>>print info[0].Number
1
>>>print info[0].Name
DefaultBuild\sample.out
>>>print info[0].ObjectDownload
True
>>>print info[0].SymbolDownload
True
>>>
```

MapInfo

Map information (return value of debugger.Map.Information)

[Type]

```
class MapInfo:
    Number = 0
    StartAddress = 0
    EndAddress = 0
    AccessSize = 0
    MapTypeName = None
```

[Variable]

Variable	Description
<i>Number</i>	The number.
<i>StartAddress</i>	The start address of the map area.
<i>EndAddress</i>	The end address of the map area.
<i>AccessSize</i>	The access size of the map area.
<i>MapTypeName</i>	The type of the map area.

[Detailed description]

MapInfo has a class format. It is the structure of the return value from the debugger.Map.Information function.

[Example of use]

```
>>>info = debugger.Map.Information()           # execute Map.Information function
1: 0x00000000 0x0003FFFF 32 (Internal ROM area)
2: 0x00040000 0x00048FFF 8 (Non map area)
3: 0x00049000 0x001003FF 8 (Emulation ROM area)
4: 0x00100400 0x03FF8FFF 8 (Non map area)
5: 0x03FF9000 0x03FFFFFF 32 (Internal RAM area)
6: 0x03FFF000 0x03FFFFFF 8 (I/O Register area)
>>>print info[0].StartAddress
0
>>>print info[0].EndAddress
262143
>>>print info[0].AccessSize
32
>>>print info[0].MapTypeName
Internal ROM area
>>>print info[5].StartAddress
67104768
>>>print info[5].EndAddress
67108863
>>>print info[5].AccessSize
8
>>>print info[5].MapTypeName
I/O Register area
>>>
```

StackInfo

Stack information (return value of debugger.Where)

[Type]

```
class StackInfo:
    Number = None
    AddressInfoText = None
```

[Variable]

Variable	Description
<i>Number</i>	The stack number.
<i>AddressInfoText</i>	The stack address as a string.

[Detailed description]

StackInfo has a class format. It is the structure of the return value from the debugger.Where function.

[Example of use]

```
>>>info = debugger.Where()
1: test2.c#
2: test1.c#main#41
>>>print info[0].Number
1
>>>print info[0].AddressInfoText
test2.c#
>>>info = debugger.Where()
1: test2.c#
--- Information below might be inaccurate.
2: test1.c#main#41
>>>print info[1].Number
None
>>>print info[1].AddressInfoText
--- Information below might be inaccurate.
>>>
```

TraceInfo

Trace information (return value of debugger.XTrace.Dump)

[Type]

```
class TraceInfo:
    FrameNumber = None
    Timestamp = None
    FetchAddress = None
    Mnemonic = None
    ReadAddress = None
    ReadData = None
    WriteAddress = None
    WriteData = None
    VectorAddress = None
    VectorData = None
    IsDma = True
```

[Variable]

Variable	Description
<i>FrameNumber</i>	Frame number information.
<i>Timestamp</i>	Timestamp information.
<i>FetchAddress</i>	Fetch address information.
<i>Mnemonic</i>	Mnemonic information.
<i>ReadAddress</i>	Read address information.
<i>ReadData</i>	Read data information.
<i>WriteAddress</i>	Write address information.
<i>WriteData</i>	Write data information.
<i>VectorAddress</i>	Vector address information.
<i>VectorData</i>	Vector data information.
<i>IsDma</i>	True if data is DMA. False if other than DMA.

[Detailed description]

TraceInfo has a class format. It is the structure of the return value from the debugger.XTrace.Dump function.

[Example of use]

```
>>>info = debugger.XTrace.Dump(10)
853 00h00min00s001ms704us000ns 0x000002c2 movhi 0xffff, gp, r1
854 00h00min00s001ms706us000ns 0x000002c6 id.w 0x7ff4[r1], r6
855 00h00min00s001ms706us000ns          0x03ff9000 R 0x00000000
856 00h00min00s001ms706us000ns 0x000002ca movhi 0xffff, gp, r1
857 00h00min00s001ms710us000ns 0x000002ce movea 0x7ff8, r1, r7
858 00h00min00s001ms712us000ns 0x000002d2 jarl _main+0x36
859 00h00min00s001ms716us000ns 0x000002dc br _main+0x36
860 00h00min00s001ms720us000ns 0x00000312 prepare lp, 0x4
861 00h00min00s001ms720us000ns          0x03ff9308 W 0x000002d6
862 00h00min00s001ms724us000ns 0x00000316 br _main+0x2

>>>print info[0].FrameNumber
853
>>>print info[0].Timestamp
1704000
>>>print info[0].FetchAddress
706
>>>print info[0].Mnemonic
movhi 0xffff, gp, r1
>>>print info[0].ReadAddress
None
>>>print info[0].ReadData
None
>>>print info[0].IsDma
False
>>>
>>>print info[2].FrameNumber
855
>>> print info[2].Timestamp
1706000
>>>print info[2].FetchAddress
None
>>>print info[2].Mnemonic
None
>>>print info[2].ReadAddress
67080192
```

XRunBreakInfo

XRunBreak information (return value of debugger.XRunBreak.Refer)

[Type]

```
class XRunBreakInfo:
    Value = 0
    TimeType = Timetype.Min
    IsPeriodic = True
```

[Variable]

Variable	Description
<i>Value</i>	Event interval value.
<i>TimeType</i>	Unit of interval value. See "debugger.XRunBreak.Set" for details.
<i>IsPeriodic</i>	Information about whether the callback is used periodically.

[Detailed description]

XRunBreakInfo has a class format. It is the structure of the return value from the debugger.XRunBreak.Refer function.

[Example of use]

```
>>>debugger.XRunBreak.Set(10, TimeType.S, True)
>>>info = debugger.XRunBreak.Refer()
10Second Periodic
>>>print info.Value
10
>>>print info.TimeType
S
>>>print info.IsPeriodic
True
>>>
```

XTimeInfo

Timer information (return value of debugger.XTime)

[Type]

```
class XTimeInfo:
    Value = 0
    IsCpuClock = False
    IsOverFlow = False
```

[Variable]

Variable	Description
<i>Value</i>	The timer measurement.
<i>IsCpuClock</i>	True if this is a CPU clock measurement. False otherwise.
<i>IsOverFlow</i>	True if an overflow has occurred. False if it has not occurred.

[Detailed description]

XTimeInfo has a class format. It is the structure of the return value from the debugger.XTime function.

[Example of use]

```
>>>info = debugger.XTime()
9820214200nsec
>>>print info.Value
9820214200
>>>print info.IsCpuClock
False
>>>print info.IsOverFlow
False
>>>
```

7.2.4 Addition / Change of the list of CubeSuite+ Python property (common)

[Location] Page 416

[Before addition]

Property Name	Function Description
common.ConsoleClear	This property sets or refers whether to clear the display of the Python console when changing the active project.
common.Output	This property refers the return value or the contents of an error.
common.ThrowExcept	This property sets or refers whether to throw an exception during the Python function is executed.
common.UsedRemoting	This property sets and displays the setting for enabling or disabling the function for linking to an external tool at Python console startup.
common.Version	This property refers the version of CubeSuite+.
common.ViewLine	This property sets or refers the number of screen lines for the Python console.
common.ViewOutput	This property sets and displays the setting for whether or not to display results of Python functions for CubeSuite+ and error messages in the Python console.

[After addition]

Property Name	Function Description
common.ConsoleClear	This property sets or refers whether to clear the display of the Python console when changing the active project.
<u>common.EnableRemotingStartup</u>	<u>This property sets and displays the setting for enabling or disabling the function for linking to an external tool at Python console startup.</u>
common.Output	This property refers the return value or the contents of an error.
common.ThrowExcept	This property sets or refers whether to throw an exception during the Python function is executed.
<u>common.UseRemoting</u>	This property sets and displays the setting for enabling or disabling the function for linking to an external tool <u>while Python console is operating.</u>
common.Version	This property refers the version of CubeSuite+.
common.ViewLine	This property sets or refers the number of screen lines for the Python console.
common.ViewOutput	This property sets and displays the setting for whether or not to display results of Python functions for CubeSuite+ and error messages in the Python console.

7.2.5 Additional description of Python property (common)

[Location] Page 417

[Before addition]

None

[After addition]

common.EnableRemotingStartup

This property sets and displays the setting for enabling or disabling the function for linking to an external tool at Python console startup.

[Specification format]

`common.EnableRemotingStartup = bool`

[Setting(s)]

Setting	Description
<i>bool</i>	Set to True to enable linking to external tools on startup of CubeSuite+. Set to False to disable. (Default is True) Use <code>common.UseRemoting</code> to enable or disable linking to external tools while running.

[Reference]

Current setting.

[Example of use]

```
>>>print common.EnableRemotingStartup
False
>>>common. EnableRemotingStartup = True
```

7.2.6 Change the name of "common.UsedRemoting"

[Location] Page 420 All descriptions "common.UsedRemoting"

[Before addition]

common.UsedRemoting

[After addition]

common.UseRemoting

7.2.7 Change the name of "common.UsedRemoting"

[Location] Page 420 All descriptions "at Python console startup"

[Before addition]

at Python console startup

[After addition]

while Python console is operating

7.2.8 Additional description of "common.UseRemoting"

[Location] Page 420

[Before addition]

Setting	Description
<i>bool</i>	Set whether to enable or disable the function for linking to an external tool while Python console is operating. True: Enable the function for linking to an external tool (default). False: Disable the function for linking to an external tool.

[After addition]

Setting	Description
<i>bool</i>	Set whether to enable or disable the function for linking to an external tool while Python console is operating. True: Enable the function for linking to an external tool (default). False: Disable the function for linking to an external tool. This will be True if common.EnableRemotingStartup is set to True on startup, and False otherwise.

7.3 Modifications in Analysis

This section describes modifications in CubeSuite+ V1.01.00 Analysis User's Manual (document # R2OUT0979EJ0100).

7.3.1 Additional description of the maximum number of plots in chart

[Location] Page 50

[Addition]

Table 2-11. Differences Depending on the Method for Acquiring Graph Data

Differences	Trace Data Analysis [IECUBE] [Simulator]	Real-time Sampling Analysis
The maximum number of plots	Without limitations	To one target, 10000 plots

7.4 Modifications in Build

This section describes modifications in CubeSuite+ V1.01.00 Build User's Manual (document # R20UT0730EJ0100, R20UT0783EJ0100).

7.4.1 Additional description of the stack usage tracer

[Location] Page 315 → Functions analyzed

[After addition] Consequently, functions in assembly files written by the user and library files created by the user are not analyzed. For this reason, the information for these files must be set using the Adjust Stack Size dialog box.

And, since an interruption function also becomes the outside of analysis management, it is necessary to set up applicable information using a adjust stack size dialog box.

[Location] Page 324 → Functions analyzed

[After addition] Consequently, functions in assembly files written by the user and library files created by the user are not analyzed. For this reason, the information for these files must be set using the Adjust Stack Size dialog box.

And, since an interruption function also becomes the outside of analysis management, it is necessary to set up applicable information using a adjust stack size dialog box.

7.5 Modifications in RX Debug

This section describes modifications in CubeSuite+ V1.02.00 Build User's Manual (document # R20UT1143EJ0100).

7.5.1 Additional description of jump to functions

[Location] Page 63

[Before addition]

(a) When CC-RX used

- The target is a function, variable, or label in C language.
- The focus is in the Editor panel.

[After addition]

(a) When CC-RX used

- The target is a function, variable, or label in C language.
- The focus is in the Editor panel.
- * Not-connected to Debug tool.
 - The type of the project specified as the active project is "Application".
 - A file with the symbol information is specified as the first file in the [Download files] property.
 - The information of target function is included in above file.
 - The target function is global function.
- * Connected to Debug tool.
 - The symbol information of the function exists in the downloaded load module file.
 - It's possible to call the target function from the address of the program counter(PC)
 - (For example) You can't jump to the static function defined besides the file of the address of the program counter(PC).

Further when using jump to the function in C++ programming, there is the following caution to specify the function. When the function can't be specified by a character string of a chosen function name, there is a possibility to jump to the different same name function.

(1) Member function of class

It's necessary to include the class name to which the target function of the target belongs. When the target function which is homonymous and different in an argument exists, please also include the argument type.

(Example) "memfun" : NG
 "Class::memfunc(short)" : OK

(2) Function defined in Namespace

It's necessary to include all Namespace to which the target function of the target belongs. When the target function which is homonymous and different in an argument exists, please also include the argument type.

(Example) "func" : NG
 "Namespace1::Namespace2::func(int)" : OK

(3) Template functions

Please include the type of the function argument a compiler generated.

(Example) "template" : NG
 "template(int, short)" : OK

7.6 Modification in 78K0 debug

This section describes modification in CubeSuite+ V1.01.00 78K0 Debug User's Manual (document# R20UT0731EJ0100).

7.6.1 Addition of point trace description Special Function Registers (SFRs) /variables of 2 byte size.

[Location] page 118 Addition before 「2.11.5 Display the collected execution history」

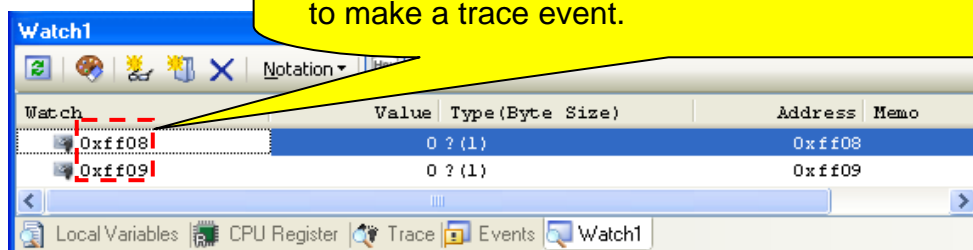
[Addition]

2.11.4(2) When an access to SFRs/variables of 2byte size occurs. [IECUBE]

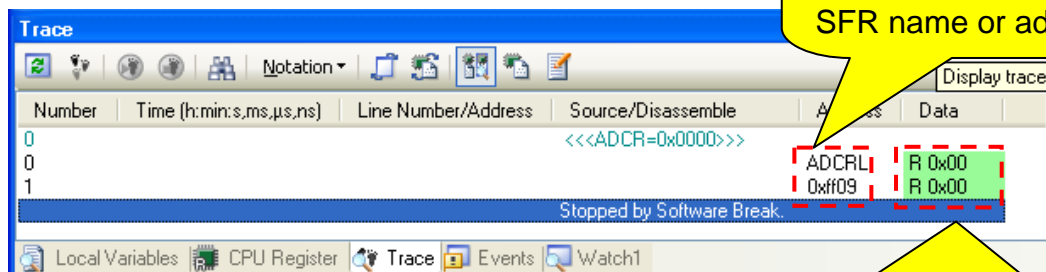
Register each address of the higher 8 bit and the lower 8 bit in watch panel directly and make a trace event to do point trace of SFRs/variables of 2 byte size. (See the figure below)

[Setting]

Example: The address higher 8 bit is 0xff08 and lower 8bit is 0xff09, right click and select [Trace output] and [Record Value] to make a trace event.



A trace result of this setting will be as follows.



[Trace result]

The data is displayed each 1 byte.

All trademarks and registered trademarks are the property of their respective owners

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



SALES OFFICES

Renesas Electronics Corporation

<http://www.renesas.com>

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.

2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited

1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH

Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.

7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.

Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited

Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852-2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.

13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886-2-8175-9670

Renesas Electronics Singapore Pte. Ltd.

1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.

Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jin Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.

11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141