CubeSuite+ V1.01.01

Release Note

R20UT0989EJ0100
Rev. 1.00
18 January, 2012

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+ V1.01.00 Start	R20UT0727EJ0100
CubeSuite+ V1.00.00 78K0 Design	R20UT0546JJ0100
CubeSuite+ V1.00.00 78K0R Design	R20UT0547JJ0100
CubeSuite+ V1.01.00 RL78 Design	R20UT0728EJ0100
CubeSuite+ V1.00.00 V850 Design	R20UT0549JJ0100
CubeSuite+ V1.01.00 78K0 Debug	R20UT0731EJ0100
CubeSuite+ V1.01.00 78K0R Debug	R20UT0732EJ0100
CubeSuite+ V1.01.00 RL78 Debug	R20UT0733EJ0100
CubeSuite+ V1.01.00 V850 Debug	R20UT0734EJ0100
CubeSuite+ V1.01.00 RX Debug	R20UT0769EJ0100
CubeSuite+ V1.01.00 Analysis	R20UT0735EJ0100
CubeSuite+ V1.01.00 Message	R20UT0736EJ0100

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 V1.01.00 to V1.01.01.

4.1 Overall Improvements for CubeSuite+

4.1.1 Improvement of Memory Usage

The memory usage of a host machine has been improved.

4.1.2 Improvement of the processing time of building

The processing time of building has been improved.

4.1.3 Support for project file of RSK for RX

The project file of RSK for RX is supported.

4.2 Removal of restriction

4.2.1 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]"

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 (.).

  Filenames 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 Drag and Drop feature is not available from the Editor panel.
- 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.

- You can specify symbols in the Jump to Specified Line dialog box only when the debugging tool is connected.
- 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.

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.

5.2 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 0x0fffefff) of the V850E/MA3 and other V850E microcontrollers.

- When CPU stops, the displayed address on disassemble panel is "0x03ff0000 to 0x03ffefff".
- 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 variables with the same type

Applies to: All debug tools, RX

Consider a case where there are two variables of different types with the same name (including parameters of functions) and one of them can be expanded (e.g. an array or structure) and the other is a pointer variable. When the display of the Watch panel is updated as soon as the program stops at the address of either of the variables, their values being displayed may not be correct.

To view correct values, register the variables with the Watch panel again or give different names to the variables.

5.4.35 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
FlashBlockIVerify	Block internal verify processing function	Emulated
FlashBlockBlankCheck	Block blank check function	Emulated
FlashGetInfo	Flash information aquisition 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 aquisition 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
FlashBlockIVerify	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 *n*	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 *jr 0xfffd6* 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 no 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. # RESET handler (in the case of address 0) .section "RESET", text jr __start -- Overwritten by jr 0xfffd6 jr __start
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 fromaddress 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. Do the following to avoid this.

・In a quoted string (“”), prepend the letter “r” to make IronPython recognize the string as a path.

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

・ Use a forward slash (/) instead of a backslash (\).

Example: "c:/test/test.py"

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 Caution for functions specified as hooks and callbacks

If a function is specified as a hook or callback, it does not wait for the function to complete.

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

(1) While you are using the simulator or E1/E20 emulator for the RX, the following commands are not usable.

- debugger.Download.Binary64Kb
- debugger.Download.BinaryBank
- debugger.Download.Coverage
- debugger.Download.Hex64Kb
- debugger.Download.HexBank
- debugger.Download.HexIdTag
- debugger.GetCpuStatus
- debugger.Map.Clear
- debugger.Map.Set
- debugger.Upload.Coverage
- debugger.Upload.IntelIdTag
- debugger.Upload.MotorolaIdTag
- debugger.Upload.Tektronix
- debugger.Upload.TektronixIdTag
- debugger.Option.OpenBreak
- debugger.Option.ReuseCoverageData
- debugger.Option.Timer
- debugger.Option.UseTraceData

(2) debugger.Erase function

While you are using the E1/E20 emulator for the RX, it is not possible to erase the flash memory in external space (i.e. option EraseOption.External is not usable).

(3) debugger.Jump.Address function

Option JumpType.Memory is not specifiable.

(4) debugger.Assemble.LineAssemble function

debugger.Assemble.LineAssemble function doesn’t support Big Endian.

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 in Start

This section describes modifications in CubeSuite+ V1.01.00 Start User's Manual (document # R20UT0727EJ0100).

7.1.1 Additional description of Python Console panel

[Location] Page 250

[Addition]

Clear	Clears all output results.
Python Initialize	Initializes Python.

7.1.2 New list of Python functions (for projects)

[Location] Page 294

[Addition]

Function Name	Function Overview
project.Change	Changes active project.
project.File.Add	Adds a file to the active project.
project.File.Remove	Removes a file from the active project.
project.Information	Displays project information.

7.1.3 New descriptions of Python functions (for projects)

[Location] Page 296

[Addition]

project.Change Changes active project.

[Summary]

project.Change project.Change: Changes the active project

[Format]

project.Change(*projectName*)

[Parameters]

Parameter	Description
projectName	Specifies the full path to the project or sub-project file to change.

[Functionality]

The project specified in projectName becomes the active project.

The project file specified in projectName must be in a currently open project.

[Return values]

True if the project was switched successfully

False if the function failed to switch the project

[Example]

```
>>>project.Change("c:/project/sample/sub1/subproject.mtsp")
>>>
```

project.Close Closes a project.

[Summary]

project.Close project.Close: Closes the currently open project.

[Format]

project.Close(*save* = False)

[Parameters]

Parameter	Description
save	Specifies whether to save changes to the project and all files being edited when the project is closed. (Default is False – do not save.)

[Functionality]

If save is True, the project is closed after saving the project and all files being edited. If it is False, the project is closed without saving the project or files being edited.

[Return values]

True if the project was closed successfully

False if the function failed to close the project

[Example]

```
>>>project.Close()
>>>
```

project.File Manipulates files in the active project.

[Summary]

project.File.Add project.File.Add: Adds a file to the active project.

[Format]

project.File.Add(*filenName*, *category* = "")

[Parameters]

Parameter	Description
fileName	Specifies the full path to the file to add to the active project.
category	Specifies the category of the file to add. Multiple hierarchies are not supported. (Unspecified by default)

[Functionality]

The file specified in fileName is added to the active project. If category is specified, it is added below that category. If the specified category does not exist, it is created. You can only specify one hierarchy level.

[Return values]

None

[Example]

```
>>>project.File.Add("c:/project/sample/src/test.c", "test")
>>>
```

[Summary]

project.File.Remove project.File.Remove: Removes a file from the active project.

[Format]

project.File.Remove(*filenName*)

[Parameters]

Parameter	Description
fileName	Specifies the full path to the file to remove from the active project.

[Functionality]

The file specified in fileName is removed from the active project. The file is not deleted.

[Return values]

None

[Example]

```
>>>project.File.Remove("c:/project/sample/src/test.c")
>>>
```

project.Information Displays a list of project files.

[Summary]

project.Information project.Information: Displays a list of project files.

[Format]

project.Information()

[Parameters]

None

[Functionality]

Displays a list of the project files in the sub-projects of the loaded project.

[Return values]

List of project filenames

[Example]

>>>project.Information()

c:\project\sample\test.mtpj

c:\project\sample\sub1\sub1project.mtsp

c:\project\sample\sub2\sub2project.mtsp

>>>

7.1.4 New list of Python functions (for build tools)

[Location] Page 297

[Addition]

Function Name	Function Overview
build.ChangeBuildMode	Changes the build mode.

7.1.5 New descriptions of Python functions (for build tools)

[Location] Page 300

[Addition]

build.ChangeBuildMode	Changes the build mode..

[Summary]

build.ChangeBuildMode: Changes the build mode.

[Format]

build.ChangeBuildMode(*buildMode*)

[Parameters]

Parameter	Description
buildMode	Specifies the build mode to change to as a string.

[Functionality]

The build mode of the project and its sub-projects are changed to the build mode specified by buildmode. If the specified build mode does not exist in the project, then a new build mode is created based on DefaultBuild, and the build mode is changed to that.

[Return values]

None

[Example]

>>>build.ChangeBuildMode("test_release")

>>>

7.1.6 Modification to description of Debugger.GetBreakStatus function

[Location] Page 326

The returned strings describing the reason for the break have been changed as follows.

[After change]

Cautions for Table: "MINICUBE2 Note 1" applies to all of the following: MINICUBE2, E1Serial, E20Serial, and EZ_Emulator

"MINICUBE Note 2" applies to all of the following: MINICUBE, E1Jtag, E20Jtag, and MINICUBE2Jtag

		78K0			RL78, 78K0R			V850			
Description of Reason for Break	Break Reason String	IECUBE	MINICUBE2 注 1	Simulator	IECUBE	MINICUBE2 注 1	Simulator	EZ_Emulator	IECUBE	MINICUBE 注 2	MINICUBE2 注 1
No break	None	○	○	—	○	○	—	○	○	○	—
Forced break	Manual	○	○	○	○	○	○	○	○	○	○
Break due to event	Event	○	○	○	○	○	○	○	○	○	○
Software break	Software	○	○	—	○	○	—	○	○	○	—
Break due to trace full	TraceFull	○	—	○	○	—	○	○	—	—	○
Break due to trace delay	TraceDelay	○	—	—	○	—	—	—	—	—	—
Access to non-mapped area	NonMap	○	—	○	○	—	○	○	—	—	○
Write to write-protected area	WriteProtect	○	—	○	○	—	○	○	—	—	○
Read from read-protected area	ReadProtect	○	—	—	—	—	—	—	—	—	—
Illegal SFR access	SfrIllegal	○	—	—	—	—	—	—	—	—	—
Read from non-readable SFR	SfrReadProtect	○	—	—	○	—	—	—	—	—	—
Write to non-writable SFR	SfrWriteProtect	○	—	—	○	—	—	—	—	—	—
Illegal access to peripheral I/O register (with address)	IorIllegal	—	—	—	—	—	—	○	—	—	—
Break due to stack overflow	StackOverflow	○	—	—	○	—	—	—	—	—	—
Break due to stack underflow	StackUnderflow	○	—	—	○	—	—	—	—	—	—
Break due to uninitialized stack pointer	UninitializeStackPointer	○	—	—	○	—	—	—	—	—	—
Read uninitialized memory	UninitializeMemoryRead	○	—	—	○	—	—	—	—	—	—
Execution timeout detected	TimerOver	○	—	—	○	—	—	○	—	—	—
Illegal operation in user program relating to peripheral chip features	UnspecifiedIllegal	○	—	—	○	—	—	—	—	—	—
Break due to illegal write to IMS/IXS register	ImsIxsIllegal	○	—	—	—	—	—	—	—	—	—
Pre-execution break	BeforeExecution	○	—	—	○	—	—	—	—	—	—

Accessed security-protected region	SecurityProtect	—	—	—	—	—	—	—	—	—	—
Flash macro service active	FlashMacroService	—	—	—	—	—	—	—	○	○	—
Number of retries exceeded limit	RetryOver	○	—	—	—	—	—	—	—	—	—
Illegal Flash break	FlashIllegal	○	—	—	○	—	—	—	—	—	—
Illegal Flash break	Peripheral	○	—	—	○	—	—	—	—	—	—
Word access to odd address	WordMissAlign Access	—	—	—	○	—	○	—	—	—	—
Temporary break	Temporary	○	○	○	○	○	○	○	○	○	○
Escape break	Escape	—	—	—	—	—	—	○	○	○	
Fetched from guard area or area where fetches are prohibited	Fetch	○	—	—	○	—	—	—	—	—	—
Wrote to IRAM guard area (with address)[1]	IRamWriteProtect	—	—	—	—	—	—	○	—	—	—
Break due to illegal instruction exception	IllegalOpcodeTrap	—	—	—	—	—	—	○	△*4	—	—
Step execution break [2]	Step	○	○	○	○	○	○	—	—	—	○
Fetch guard break [2]	FetchGuard	○	—	—	○	—	—	—	—	—	—
Trace stop [2]	TraceStop	○	—	—	○	—	—	—	—	—	—
Execution failed [3]	ExecutionFails	○	○	—	○	○	—	○	○	○	—

1. Performed a verification check on the IRAM guard area during break,
   and the value was overwritten (if this affects multiple addresses, only the first address is shown).x
2. This is only a break cause during trace.
3. This is only a break cause during a break.
4. Not displayed with V850-MINICUBE on V850E/ME2, etc. (same core) when a post-execution event is used.

		RX		V850E2			
Description of Reason for Break	Break Reason String	E1Jtag,E1Serial E20Jtag,E20Serial	SIM	IECUBE2	MINICUBE 注 2	MINICUBE2 注 1	Simulator
No break	None	○	—	○	○	○	—
Forced break	Manual	○	○	○	○	○	○
Break due to event	Event	○	○	○	○	○	○
Software break	Software	○	—	○	○	○	—
Break due to trace full	TraceFull	○	○	○	—	—	○
Access to non-mapped area	NonMap	—	—	—	—	—	○

Write to write-protected area	WriteProtect	－	－	－	－	－	○
Execution timeout detected	TimerOver	－	－	○	○	－	－
Flash macro service active	FlashMacroService	－	－	○	○	○	－
Temporary break	Temporary	○	○	○	○	○	○
Break due to illegal instruction exception	IllegalOpcodeTrap	－	－	○	○	－	－
Step execution break [2]	Step	○	－	－	－	－	○
Execution failed [3]	ExecutionFails	○	－	○	○	○	－
Break due to WAIT instruction execution	WaitInstruction	－	○	－	－	－	－
Break due to undefined instruction exception	UndefinedInstructionException	－	○	－	－	－	－
Break due to privileged instruction exception	PrivilegeInstructionException	－	○	－	－	－	－
Break due to access exception	AccessException	－	○	－	－	－	－
Break due to floating-point exception	FloatingPointException	－	○	－	－	－	－
Break due to interrupt	InterruptException	－	○	－	－	－	－
Break due to INT instruction exception	IntInstructionException	－	○	－	－	－	－
Break due to BRK instruction exception	BrkInstructionException	－	○	－	－	－	－
Break due to peripheral function simulation	IOFunctionSimulationBreak	－	○	－	－	－	－
Break due to illegal memory access	IllegalMemoryAccessBreak	－	○	－	－	－	－
Break due to streaming I/O error	StreamIoError	－	○	－	－	－	－
Failed to allocate coverage memory	CoverageMemoryAllocationFailure	－	○	－	－	－	－
Failed to allocate trace memory	TraceMemoryAllocationFailure	－	○	－	－	－	－

[2] Only a break cause during trace. [3] Only a break cause during break.

7.2 Modifications in Analysis

This section describes modifications in CubeSuite+ V1.01.00 Analysis User's Manual (document # R20UT0735EJ0100).

7.2.1 Additional description of the maximum number of plots in chart

[Location] Page 47

[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

7.3 Modifications in Build

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

7.3.1 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.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. 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.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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.
5. When exporting the 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. You should not use Renesas Electronics products or the 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. 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.
6. 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.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. 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 categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   - "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.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. 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.
9. 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 system manufactured by you.
10. 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.
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.

RENESAS

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, Jln 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

Colophon 1.1