CubeSuite Ver.1.40

Integrated Development Environment

User's Manual: V850 Debug

Target Device
V850 Microcontroller

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.1.00   Oct 2010

How to Use This Manual

This manual describes the role of the CubeSuite integrated development environment for developing applications and systems for V850 microcontrollers, and provides an outline of its features.

CubeSuite is an integrated development environment (IDE) for V850 microcontrollers, integrating the necessary tools for the development phase of software (e.g. design, implementation, and debugging) into a single platform.

By providing an integrated environment, it is possible to perform all development using just this product, without the need to use many different tools separately.

Readers This manual is intended for users who wish to understand the functions of the CubeSuite and design software and hardware application systems.

Purpose This manual is intended to give users an understanding of the functions of the Cubesuite to use for reference in developing the hardware or software of systems using these devices.

Organization This manual can be broadly divided into the following units.

CHAPTER 1 GENERAL
CHAPTER 2 FUNCTIONS
APPENDIX A WINDOW REFERENCE
APPENDIX B USER OPEN INTERFACE
APPENDIX C INDEX

How to Read This Manual It is assumed that the readers of this manual have general knowledge of electricity, logic circuits, and microcontrollers.

Conventions

Data significance:	Higher digits on the left and lower digits on the right
Active low representation:	$\overline{XXX}$ (overscore over pin or signal name)
Note:	Footnote for item marked with Note in the text
Caution:	Information requiring particular attention
Remark:	Supplementary information
Numeric representation:	Decimal … XXXX
	Hexadecimal … 0xXXXX

Related Documents   The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

Document Name		Document No.
CubeSuite Integrated Development Environment User's Manual	Start	R20UT0256E
	Analysis	R20UT0265E
	Programming	R20UT0266E
	Message	R20UT0267E
	Coding for CX compiler	R20UT0259E
	Build for CX compiler	R20UT0261E
	78K0 Coding	R20UT0004E
	78K0 Build	R20UT0005E
	78K0 Debug	R20UT0262E
	78K0 Design	R20UT0006E
	78K0R Coding	U19382E
	78K0R Build	U19385E
	78K0R Debug	R20UT0263E
	78K0R Design	R20UT0007E
	V850 Coding	U19383E
	V850 Build	U19386E
	V850 Debug	This manual
	V850 Design	R20UT0257E

Caution   The related documents listed above are subject to change without notice. Be sure to use the latest edition of each document when designing.

All trademarks or registered trademarks in this document are the property of their respective owners.

[MEMO]

[MEMO]

[MEMO]

TABLE OF CONTENTS

CubeSuite is a platform of an integrated developing environment for the 78K0 microcontroller, 78K0R microcontroller and V850 microcontroller.

CubeSuite can run all the operations needed for developing the programs such as designing, cording, building, debugging, and flash programing.

In this manual, the debugging is explained out of those operations needed for the program development.

In this chapter, an overview of debugging products that CubeSuite provides is explained.

1.1   Summary

You can effectively debug/simulate the program developed for the V850 microcontroller, using the debugger which Cube Suite provides.

1.2   Features

The following are the features of the debugger provided by CubeSuite.

- Connecting to the various debug tools
  A pleasant debugging environment for target systems is provided by connecting to Full-spec Emulators (IECUBE/IECUBE2), On-chip Debugging Emulators (MINICUBE/MINICUBE2/E1) and Simulator.

- C source text and disassembled text are shown mixed
  The C source text and the disassembled text are shown mixed on the same panel.

- Source level debugging and instruction level debugging
  The source level debugging and the instruction level debugging for a C source program can be done.

- Support of flash self programming emulation
  [V850E1][V850ES]
  Flash self programming emulation can be performed with IECUBE.
  [V850E2M]
  Flash self programming emulation (including settings of option bytes) can be performed with all of the connectable debug tools.

- Data flash memory writing function
  When the selected microcontroller incorporates the data flash memory, the contents of data flash memory (including ID tag) can be displayed and modified by the same access method as an ordinary memory operation.

- Real-time display update function
  The contents of memory, registers and variables are automatically updated not only when the program execution is stopped, but also in execution.

- Save/restore the debugging environment
  The debugging environment such as breakpoints, event configuration information, file download information, display condition/position of the panel, etc. can be saved.

CHAPTER 2   FUNCTIONS

This chapter describes a debugging process of CubeSuite and main functions for debugging.

2.1   Overview

The basic debugging sequence for programs using CubeSuite is as follows:

(1) Start CubeSuite

Launch CubeSuite from the [Start] menu of Windows®.

Remark   For details on "Start CubeSuite", see "CubeSuite Start".

(2) Set a project

Create a new project, or load an existing one.

Remark   For details on "Set a project", see "CubeSuite Start".

(3) Create a load module

Create a load module by running a build after setting of the active project and the build tool to be used.

Remark   For details on "Create a load module" with CA850/CX, see "CubeSuite Build".

(4) Confirm the connection to a host machine

Connect the debug tool (IECUBE, IECUBE2, MINICUBE, MINICUBE2, E1, or Simulator) to be used to a host machine.

(5) Select the debug tool to use

Select the debug tool to use in a project.

Remark   The selectable debug tool differs depending on the microcontroller type to be used in a project.

(6) Configure operating environment of the debug tool

Configure the operating environment of the debug tool selected in steps (5).
- [IECUBE]/[IECUBE2]
- [MINICUBE]/[MINICUBE2]
- [E1]
- [Simulator]

(7) Connect the debug tool to CubeSuite

Connect the debug tool to CubeSuite to start communication.

(8) Execute downloading

Download the load module created in steps (3) to the debug tool.

(9) Display source files

Display the contents of the downloaded load module (source files) on the Editor panel or Disassemble panel.

(10) Execute programs

Execute the program by using the operation method corresponding to a purpose.

If you wish to stop the program at the arbitrary position, set a breakpoint/break event[Note] before executing the program (see "2.8.2   Stop the program at the arbitrary position (breakpoint)"/"2.8.3   Stop the program with the access to variables/I/O registers").

Note   These functions are implemented by setting events to the debug tool used.
      See "2.15.6   Notes for setting events", when you use events.

Remark   [V850E2M]
         When the selected microcontroller version supports multi-core, select a core (PE*n*) to be debugged on the Debug Manager panel [V850E2M] before executing the program.

(11) Stop the program manually

Stop the program currently being executed.

Note that if a breakpoint/break event has been set in steps (10), the program execution will be stopped automatically when the condition of the breakpoint/break event is met.

(12) Check the result of the program execution

Check the following information that the debug tool acquired by the program execution.

- Display/Change the Memory, Register and Variable
- Display Information on Function Call from Stack
- Collect Execution History of Programs [IECUBE][IECUBE2][Simulator][Note]
- Measure Execution Time of Programs[Note]
- Measure Coverage [IECUBE][IECUBE2][Simulator]

Note   These functions are implemented by setting events to the debug tool used.
      See "2.15.6   Notes for setting events", when you use events.

Debug the program, repeating steps (9) to (12) as required.

Note that if the program is modified during debugging, steps (3) and (8) also should be repeated.

Remarks 1.   Other than the above, you can also check the result of the program execution by using the following functions.
              - Set an Action into Programs
              - Use Hook Function
              - Use the Simulator GUI [Simulator]
2.   The acquired information can be saved to a file.
              - Save the disassembled text contents
              - Save the memory contents
              - Save the CPU register contents
              - Save the I/O register contents
              - Save the contents of local variables
              - Save the contents of watch-expressions
              - Save the contents of call stack information
              - Save the contents of execution history

(13) Execute uploading

Save the program (the memory contents) to a file in the arbitrary format (e.g. Intel hex format, binary data format, and etc.), as required.

(14) Disconnect the debug tool from CubeSuite

Disconnect the debug tool from CubeSuite to terminate communication.

(15) Save the project file

Save the setting information of the project to the project file.

> **Remark**   For details on "Save the project file", see "CubeSuite Start".

2.2   Preparation before Debugging

This section describes the preparation to start debugging the created program.

2.2.1   Confirm the connection to a host machine

Connection examples for each debug tool are shown.

Note that the relationship between types of the microcontroller selected in the project and the connectable debug tools is as follows:

Table 2-1.   Relationship between Types of Microcontroller and Connectable Debug Tools

Type of Microcontroller	Debug Tool				
	IECUBE	IECUBE2	MINICUBE E1 (JTAG)[Note]	MINICUBE2 E1 (Serial)[Note]	Simulator
V850E1	✓	—	✓	✓	✓
V850ES	✓	—	✓	✓	✓
V850E2M (Single-core)	—	✓	✓	—	✓
V850E2M (Multi-core)	—	—	✓	—	—

Note   E1 (JTAG):   When E1 is used with JTAG communications
        E1 (Serial):   When E1 is used with serial communications

(1)   [IECUBE]
(2)   [IECUBE2]
(3)   [MINICUBE]
(4)   [MINICUBE2]
(5)   [E1]
(6)   [Simulator]

(1) [IECUBE]

Connect a host machine and IECUBE. If required, connect a target board, too.

See IECUBE User's Manual for details on the connection method.

Figure 2-1.   Connection Example [IECUBE]
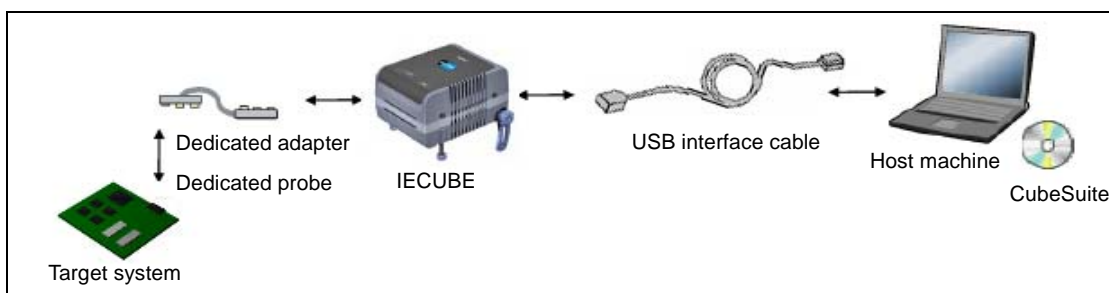
(2) [IECUBE2]

Connect a host machine and IECUBE2. If required, connect a target board, too.

See IECUBE2 User's Manual for details on the connection method.

Figure 2-2. Connection Example [IECUBE2]
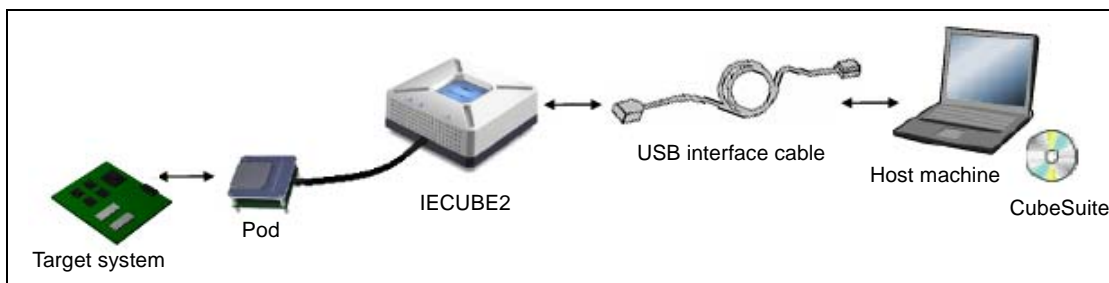
(3) [MINICUBE]

Connect a host machine and MINICUBE. If required, connect a target board, too.

See MINICUBE User's Manual for details on the connection method.

Figure 2-3. Connection Example [MINICUBE]
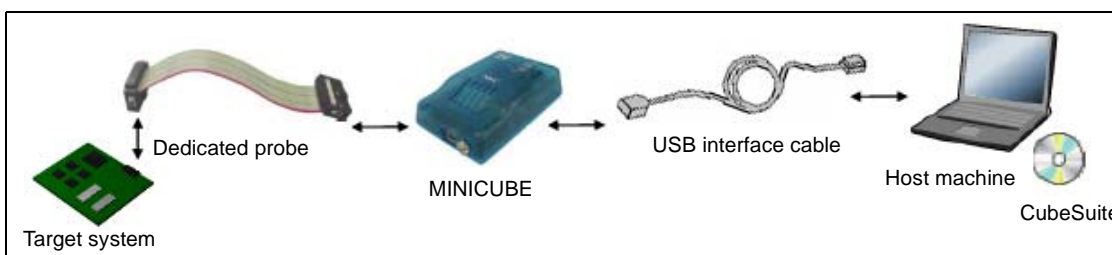
(4) [MINICUBE2]

Connect a host machine and MINICUBE2. If required, connect a target board, too.

See MINICUBE2 User's Manual for details on the connection method.

Figure 2-4. Connection Example [MINICUBE2]
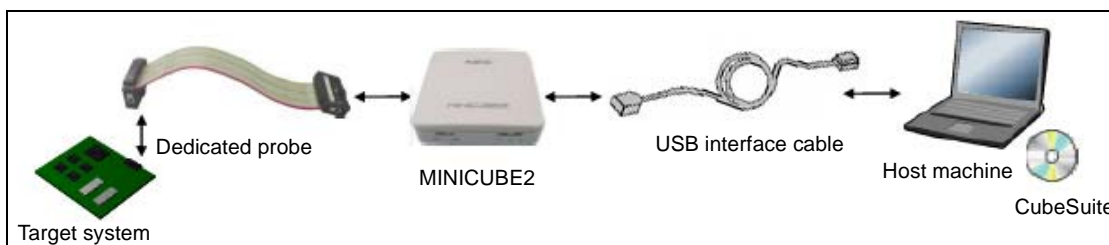
(5) [E1]

Connect a host machine and E1. If required, connect a target board, too.

See E1 User's Manual for details on the connection method.

Figure 2-5.   Connection Example [E1]
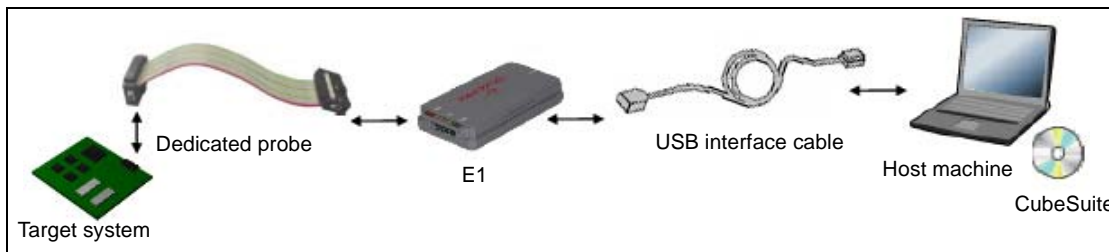
Remark You can make a choice between serial communications **[E1 (Serial)]** and JTAG communications **[E1 (JTAG)]** as the communication method to the target system (see "2.3.1   Select the debug tool to use").

(6) [Simulator]

A host machine is only needed for debugging (emulators are not needed).

Figure 2-6.   Connection Example [Simulator]
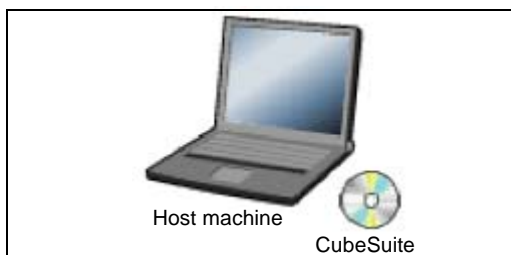
2.3 Configuration of Operating Environment of the Debug Tool

This section describes the configuration of the operating environment for each debug tool.

2.3.1 Select the debug tool to use

You can configure the operating environment in the Property panel corresponding to the debug tool to use.

Therefore, first, select the debug tool to be used in a project (the debug tool to be used can be specified in the individual projects).

To select or switch the debug tool, use the context menu shown by right clicking on the [*Microcontroller type Debug tool name* (Debug Tool)] node on the Project Tree panel.

Figure 2-7. Select/Switch Debug Tool to Use
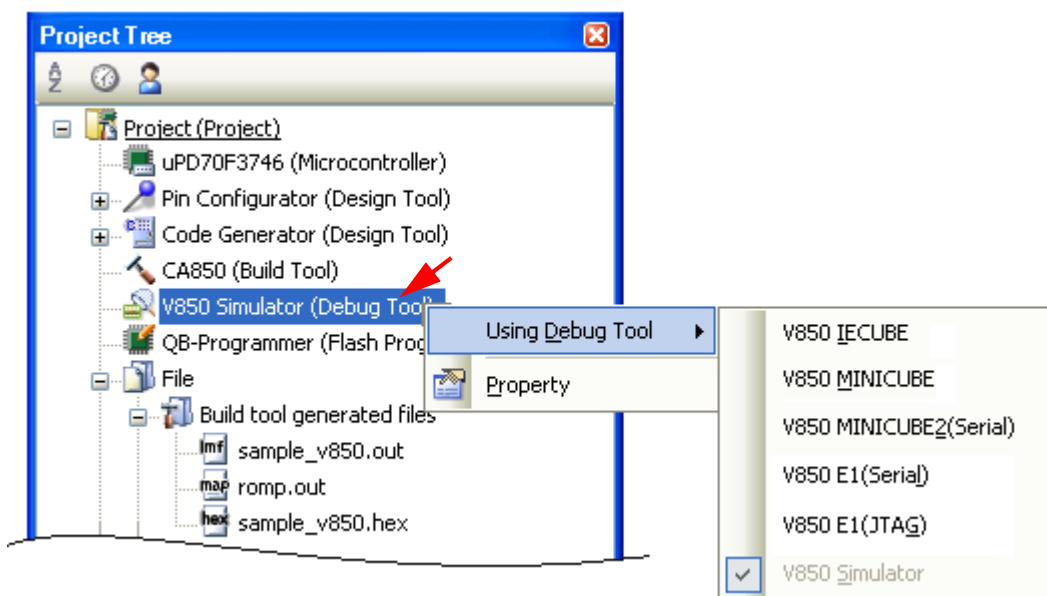
Caution   **The context menu items displayed differ depending on the microcontroller selected in the project (see "Table 2-1. Relationship between Types of Microcontroller and Connectable Debug Tools").**

Remark   **[E1]**

When serial communications will be performed as the communication method between E1 and the device in the target system, select [*Microcontroller type* E1 (Serial)], when JTAG communications will be performed, then select [*Microcontroller type* E1 (JTAG)].

If the Property panel is already open, click the [*Microcontroller type Debug tool name* (Debug Tool)] node again. The view switches to the Property panel of the selected debug tool.

If the Property panel is not open, double-click the above mentioned node to open the corresponding Property panel.

2.3.2 [IECUBE]/[IECUBE2]

Configure the operating environment on the Property panel below when using IECUBE/IECUBE2.

Figure 2-8. Property Panel [IECUBE]
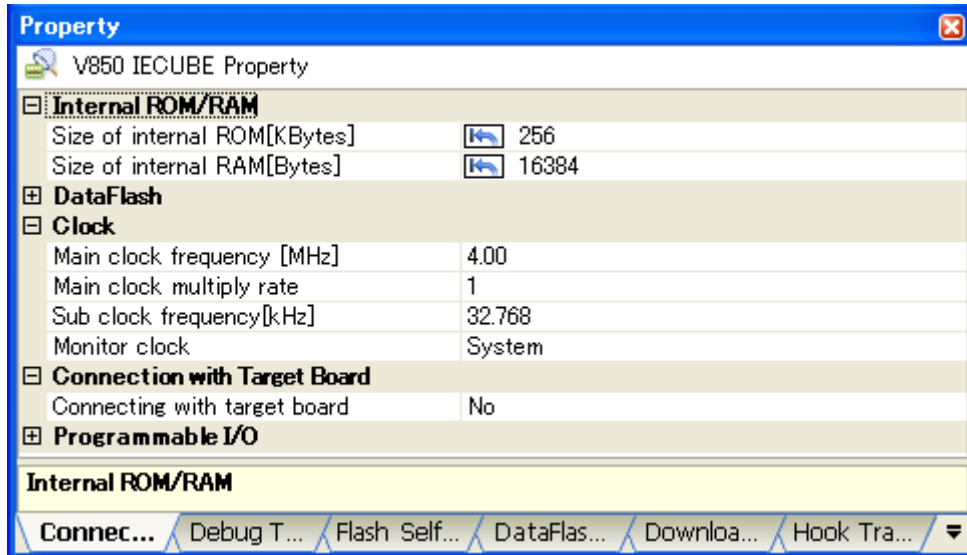
Figure 2-9. Property Panel [IECUBE2]
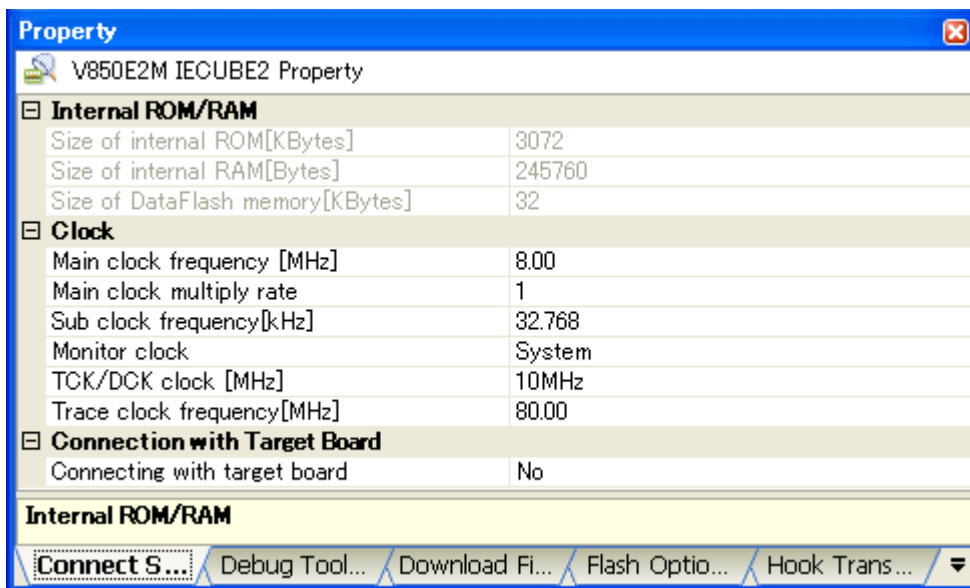
Follow the steps below by selecting the corresponding tab on the Property panel.

(1)   [Connect Settings] tab
(2)   [Debug Tool Settings] tab
(3)   [Flash Self Emulation Settings] tab [IECUBE]
(4)   [DataFlash Emulation Settings] tab [IECUBE]
(5)   [Download File Settings] tab
(6)   [Flash Options Settings] tab [IECUBE2]
(7)   [Hook Transaction Settings] tab

(1) [Connect Settings] tab

In the [Connect Settings] tab, configuration with regard to the connection to the debug tool can be done.

(a) [Internal ROM/RAM]
(b) [DataFlash] [IECUBE]
(c) [Clock]
(d) [Connection with Target Board]
(e) [Programmable I/O] [IECUBE]

(a) [Internal ROM/RAM]

You can configure internal ROM/RAM in this category.

The size of internal ROM/RAM of the selected microcontroller is specified by default (when the selected microcontroller is the V850E1 core and incorporates VSB Flash ROM/VSB RAM, the value including its size is specified).

Remark   There is no need to change the settings in this category if you wish to debug with the same memory mapping of the selected microcontroller.

Figure 2-10.   [Internal ROM/RAM] Category [IECUBE]

Internal ROM/RAM		
Size of internal ROM[KBytes]	↩	256
Size of internal RAM[Bytes]	↩	16384

Figure 2-11.   [Internal ROM/RAM] Category [IECUBE2]

Internal ROM/RAM	
Size of internal ROM[KBytes]	3072
Size of internal RAM[Bytes]	245760
Size of DataFlash memory[KBytes]	32

<1> [Size of internal ROM[KBytes]

- **[IECUBE]**

Specify the internal ROM size to emulate (unit: Kbytes).

To perform debugging using IECUBE memory resources after changing the memory mapping, make a selection form the drop-down list.

Note that when the selected microcontroller is a ROMless product, the property value cannot be changed.

- **[IECUBE2]**

The internal ROM size to emulate is displayed (unit: Kbytes).

You cannot change the value of this property.

<2> [Size of internal RAM[Bytes]]

- **[IECUBE]**

Specify the internal RAM size to emulate (unit: bytes).

To perform debugging using IECUBE memory resources after changing the memory mapping, make a selection form the drop-down list.

- **[IECUBE2]**

The internal RAM size to emulate is displayed (unit: bytes).

You cannot change the value of this property.

<3> [Size of DataFlash memory[KBytes]] [IECUBE2]

The size of the data flash memory area of the selected microcontroller is displayed (unit: Kbytes).

You cannot change the value of this property.

Caution   [IECUBE]

You should be careful not to overlap the area with other memory mapping area.

mark in the setting column indicates that if you change the setting of this property value, CPU reset is generated automatically.

(b) [DataFlash] [IECUBE]

You can configure the data flash function in this category.

Note that this category appears only when the selected microcontroller incorporates the data flash memory.

Figure 2-12.   [DataFlash] Category

<1> [Use DataFlash memory area]

Specify whether to use the data flash memory from the drop-down list.

Select [Yes] to use the data flash memory ([No] is selected by default).

<2> [Size of DataFlash memory[KBytes]]

This property appears only when the [Use DataFlash memory area] property is set to [Yes].

The size of the data flash memory area of the selected microcontroller is displayed (unit: Kbytes).

You cannot change the value of this property.

<3> [Chip select]

This property appears only when the [Use DataFlash memory area] property is set to [Yes].

Select the chip select used for mapping of the data flash memory from the drop-dawn list.

The selectable chip select values differ depending on the selected microcontroller.

Note that when the selected microcontroller is the V850ES core, you cannot change the value of the property.

(c) [Clock]

You can configure the clock in this category.

Figure 2-13.   [Clock] Category [IECUBE]

Figure 2-14.   [Clock] Category [IECUBE2]

```
⊟ Clock
   Main clock frequency [MHz]        8.00
   Main clock multiply rate          1
   Sub clock frequency[kHz]          32.768
   Monitor clock                     System
   TCK/DCK clock [MHz]               10MHz
   Trace clock frequency[MHz]        80.00
```

<1> [Main clock frequency [MHz]]

Specify the main clock frequency (before multiplication).

You can specify the frequency from the drop-down list or by directly entering a frequency value between 0.001 and 999.999 (unit: MHz).

The drop-down list displays the following frequencies (unit: MHz).

-   **[V850E1][V850ES]**

    2.50, 4.00 (default), 5.00, 6.00, 8.00, 13.50, 18.00

-   **[V850E2M]**

    4.00, 5.00, 7.20, 8.00 (default), 9.60, 10.00, 13.50, 16.00

<2> [Main clock multiply rate]

Specify the main clock frequency multiplier.

Select a value from the drop-down list or directly enter an integer value from 1 to 99.

The drop-down list displays the following multiplier values.

-   **[V850E1][V850ES]**

    1 (default), 2, 3, 4, 5, 6, 7, 8, 9, 10

-   **[V850E2M]**

    1 (default), 2, 3, 4, 5, 6, 7, 8, 9, 10, 20

<3> [Sub clock frequency[kHz]]

This property appears only when the selected microcontroller supports a sub clock.

Specify the sub clock frequency.

You can specify the frequency from the drop-down list or by directly entering a frequency value between 0.001 and 999.999 (unit: kHz).

The drop-down list displays sub clock frequency values which can be used by the currently selected microcontroller (if the frequency value which can be use is fixed, the drop-down list does not appear). The minimum usable sub clock frequency is specified by default.

<4> [Monitor clock]

This property appears only when the selected microcontroller supports a sub clock.

Specify from the following drop-down list a clock on which the monitor program operates while the program is stopped.

System	Operates with main clock (default).
User	Operates with the clock that the program specified.

<5> [TCK/DCK clock [MHz]] [IECUBE2]

Specify a clock supplied from IECUBE2 to the on-chip debug unit (DCU) from the drop-down list.

The drop-down list displays the following frequency values.

10MHz (default), 20MHz

> Cautions 1.   Normally, do not change the setting from [10MHz].
>   When [20MHz] is specified, it may not be possible to connect to the debug tool.
>   2.   This property cannot be changed while connecting to IECUBE2.

<6> **[Trace clock frequency[MHz]] [IECUBE2]**

Specify the clock frequency for using trace function.

Directly enter the value between 1 and 999.999 (unit: MHz)([80.00] is specified by default).

(d) **[Connection with Target Board]**

You can configure the connection to the target board in this category.

Figure 2-15.   [Connection with Target Board] Category [IECUBE][IECUBE2]

Connection with Target Board	
Connecting with target board	No

<1> **[Connecting with target board]**

Specify if the target board is connected to IECUBE/IECUBE2 or not from the drop-down list.

Select [Yes] when the target board is connected to IECUBE ([No] is selected by default).

> Caution   This property cannot be changed while connecting to IECUBE.

(e) **[Programmable I/O] [IECUBE]**

You can configure the programmable I/O area in this category.

Note that this category appears only when the selected microcontroller supports the programmable I/O area.

> Remark   Change this setting to temporarily change variables while debugging.
>   To set a common value for the project, in the Property panel of the build tool, on the [Common
>   Options] tab, in the [Device] category, set the value in [Programmable I/O area start address]
>   property.

Figure 2-16.   [Programmable I/O] Category

Programmable I/O	
Use Programmable I/O area	Yes
Programmable I/O area start address	HEX 0

<1> **[Use Programmable I/O area]**

Specify whether to use the programmable I/O area.

Select [Yes] to use the programmable I/O area ([No] is selected by default).

<2> **[Programmable I/O area start address]**

This property appears only when the [Use Programmable I/O area] property is set to [Yes].

Specify the start address of the programmable I/O area.

Specify the address expression within the programmable I/O area ([0] is specified by default).

The address is aligned to 16 Kbytes.

(2) [Debug Tool Settings] tab

In the [Debug Tool Settings] tab, general configurations on the debug tool can be done.

(a) [Memory]
(b) [Access Memory While Running]
(c) [Set Event While Running]
(d) [Break]
(e) [Fail-safe Break] [IECUBE]
(f) [Trace]
(g) [Timer]
(h) [Coverage]
(i) [Mask for Input Signal]
(j) [External Flash Memory Download] [IECUBE2]

(a) [Memory]

You can configure the memory in this category.

Figure 2-17.   [Memory] Category

<1> [Memory mappings]

Current memory mapping status is displayed for each type of memory area.

The memory mapping status cannot be changed on this panel. If it is necessary to add a memory mapping, click on the [Memory Mapping] property, and click on the [...] button that appears on the right end of the setting field. The Memory Mapping dialog box opens; perform the setting from there.

See the section for the Memory Mapping dialog box for details on how to configure the parameters.

Figure 2-18.   Opening the Memory Mapping Dialog Box

Caution   **If you are not connected to a debug tool, then only memory mapping areas added by user is displayed.**

Connecting to a debug tool (see "2.4.1   Connect the debug tool to CubeSuite") will display details for each memory type.

<2> [Verify on writing to memory]

Specify whether to perform a verify check when data has been written to memory from the drop-down list.

Select [Yes] to perform verification (default).

Remark   The verification is performed at when the download is performed, or when the values that have been modified on the Watch panel/Memory panel are written to the memory.

(b) [Access Memory While Running]

You can configure the memory access while executing a program in this category.

The settings of this category are required when using the real-time display update function. See "(4)   Display/ modify the memory contents during program execution" for details on the real-time display update function.

Figure 2-19.   [Access Memory While Running] Category [IECUBE]

Figure 2-20.   [Access Memory While Running] Category [IECUBE2]

<1> [Access by stopping execution]

Specify whether to allow access to the memory area where cannot be accessed while executing a program (target memory area/I/O register area) from the drop-down list.

Select [Yes] to allow access ([No] is selected by default).

<2> [Update display during the execution]

Specify whether to update the display in the Watch panel/Memory panel during a program execution.

Select [Yes] to update the display (default).

Caution   [IECUBE]

This property becomes invalid when the [Use for trace data] property in the [Trace] category is set to [Trace] (see "2.11.1   Configure the trace operation").

<3> [Display update interval[ms]]

This property appears only when the [Update display during the execution] property is set to [Yes].

Specify the interval in 100 ms unit to update the contents in the Watch panel/Memory panel display while executing a program.

Directly specify the Integer number between 100 and 65500 (rounding up the fractions less than 100ms) ([500] is selected by default).

<4> [Set update display during the execution automatically] [IECUBE]

This property appears only when the [Update display during the execution] property is set to [Yes].

Select [Yes] when letting the IECUBE set the area displayed in the Watch panel/Memory panel as automatically as possible as an area to be updated by the real-time display update function and updating the displayed content during execution of the program (default).

<5> [Access by release HALT mode]

This property appears only when the [Access by stopping execution] property is set to [Yes].

Specify whether to allow access to the memory area/CPU registers/I/O registers by releasing the HALT mode from the drop-down list.

Select [Yes] to allow access to the memory area/CPU registers/I/O registers by releasing the HALT mode (default).

(c) [Set Event While Running]

You can configure the setting of events while executing a program in this category.

Figure 2-21.   [Set Event While Running] Category

<1> [Set event by stopping execution momentarily]

Specify whether to forcibly pause the execution for events that cannot be set while executing a program or operating the tracer/timer.

For details on the event types that are affected by this property, see "Table 2-25.   Event Types That Can be Set and Deleted during Execution [V850E1][V850ES]".

Select [Yes] to set events above while execution ([No] is selected by default).

(d) [Break]

You can configure the break function in this category.

Figure 2-22.   [Break] Category [IECUBE][IECUBE2]

<1> [First using type of breakpoint]

Specify from the following drop-down list a breakpoint type to use with priority when setting it with a one click operation of the mouse in the Editor panel/Disassemble panel.

See "2.8.2   Stop the program at the arbitrary position (breakpoint)" for details on breakpoints.

Software break	Sets software breakpoint with priority (default).
Hardware break	Sets hardware breakpoint with priority.

<2> [Stop emulation of peripherals when stopping]

Specify from the drop-down list whether the peripheral emulation function of emulator is stopped while stopping the program execution.

Select [Yes] to terminate ([No] is selected by default).

<3> [Use open break function]

Specify from the following drop-down list whether to use the open break function.

Yes(Hi-Z)	The open break target pin becomes the Hi-Z state after the CPU is stopped.
No(Output signal)	The open break target pin outputs the signal even after the CPU is stopped (default).

(e) [Fail-safe Break] [IECUBE]

You can configure the fail-safe break function in this category.

See "2.8.4   Stop the program when an invalid execution is detected [IECUBE]" for details on the fail-safe break function and this category configuration.

(f) [Trace]

You can configure the trace function in this category.

See "2.11   Collect Execution History of Programs [IECUBE][IECUBE2][Simulator]" for details on the trace function and this category configuration.

(g) [Timer]

You can configure the timer function in this category.

See "2.12   Measure Execution Time of Programs" for details on the timer function.

Caution   [IECUBE2]

This category become invalid when the [Use for trace data] property in the [Trace] category is set to other than [Timer].

Figure 2-23.   [Timer] Category [IECUBE]

□ Timer
Rate of frequency division of timer   1/1(20ns/1.4min)

Figure 2-24.   [Timer] Category [IECUBE2]

□ Timer
Rate of frequency division of timer   1/1(5ns/21.4s)
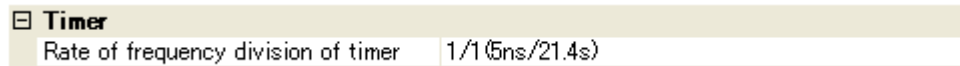
<1> [Rate of frequency division of timer]

Specify the frequency division ratio of the timer counter (32 bits, 50 MHz [IECUBE]/32 bits, 200 MHz [IECUBE2]) used for timer measurement from the drop-down list.

In the drop-down list, following frequency division ratios are shown (resolution/maximum measurement time are shown in "( )").

- **[IECUBE]**

1/1(20ns/1.4min) (default),   1/2(40ns/2.8min),   1/4(80ns/5.7min),   1/8(160ns/11.4min),
1/16(320ns/22.8min),   1/32(640ns/45.6min),   1/64(1280ns/1.5h),
1/128(2560ns/3.0h),   1/256(5120ns/6.0h),   1/512(10240ns/12.2h),
1/1024(20480ns/24.4h),   1/2048(40960ns/48.8h) ,   1/4096(81920ns/97.6h)

- **[IECUBE2]**

1/1(5ns/21.4s) (default),   1/2(10ns/42.9s),   1/4(20ns/1.4min),   1/8(40ns/2.8min),
1/16(80ns/5.71min),   1/32(160ns/11.42min),   1/64(320ns/22.8min),
1/128(640ns/45.6min),   1/256(1280ns/1.5h),   1/512(2560ns/3.0.h),
1/1024(5120ns/6.0h),   1/2048(10240ns/ 12.2h),   1/4096(20480ns/24.4h)

(h) [Coverage]

You can configure the coverage function in this category.

See "2.13 Measure Coverage [IECUBE][IECUBE2][Simulator]" for details on the coverage function and this category configuration.

(i) [Mask for Input Signal]

You can configure the input signal masking in this category.

Figure 2-25. [Mask for Input Signal] Category [IECUBE]

Mask for Input Signal	
Mask NMI0 signal	No
Mask NMI1 signal	No
Mask NMI2 signal	No
Mask HLDRQ signal	No
Mask RESET signal	No
Mask STOP signal	No
Mask WAIT signal	No

Figure 2-26. [Mask for Input Signal] Category [IECUBE2]

Mask for Input Signal	
Mask HLDRQ signal	No
Mask RESET signal	No
Mask STOP signal	No
Mask WAIT signal	No

With the properties shown below, select [Yes] to mask the signal from the drop-down list (all properties below are set to [No] by default).

- [Mask NMIn signal] **[IECUBE]**
- [Mask HLDRQ signal]Note
- [Mask RESET signal]
- [Mask STOP signal]
- [Mask WAIT signal]Note

Note When [No] is specified with the [Connection with Target Board] property in the [Connect Setting] tab, these properties are fixed to [Yes] after connecting to the debug tool (changes not allowed).

(j) [External Flash Memory Download] [IECUBE2]

You can configure downloading to an external flash memory in this category.

See "(5) Download to external flash memory [IECUBE2][MINICUBE][E1 (JTAG)]" for details on how to download to an external flash memory and this category configuration.

(3) [Flash Self Emulation Settings] tab [IECUBE]

In the [Flash Self Emulation Settings] tab [IECUBE] tab, general configurations on flash self programing emulation (Code flash) can be done.

- (a) [Flash Self Emulation]
- (b) [Macro Service Error]
- (c) [Security Flag Emulation]
- (d) [Reset Vector/Boot Swap]

(a) [Flash Self Emulation]

You can configure the flash self programing emulation function in this category.

Figure 2-27.   [Flash Self Emulation] Category

Flash Self Emulation	
Flash self-programming	No
Flash Macro service specification	Flash self-programming Ver. 2.00 or earlier

<1> [Flash self-programming]

Specify whether to use the flash self programming emulation function from the drop-down list.

Select [Yes] to use the flash self programming emulation function ([No] is selected by default).

<2> [Flash Macro service specification]

This property appears only when the flash memory process is Type1[Note]

Select a specification of the flash macro service from the following drop-down list.

Flash self programming Ver.2.00 or earlier (default)
Flash self programming Ver.3.00 or later

Note The determination of the flash memory process is made when connected to the debug tool.
Consequently, this will not be hidden when disconnected from the debug tool.

(b) [Macro Service Error]

You can configure the operation of flash functions of the self programing library for the flash macro service in this category.

For details on the flash functions, see "V850 Microcontrollers Flash Memory Self Programming Library Type0*x*".

Figure 2-28.   [Macro Service Error] Category

Macro Service Error	
Generate FlashBlockErase error	No
Generate FlashBlockIVerify error	No
Generate FlashWordWrite error	No
Generate FlashBlockBlankCheck error	No
Generate FlashSetInfo error	No
Generate FlashFLMDCheck error	No

With the following properties, specify whether to return the error value when the flash memory is damaged (the error returned when the flash memory is damaged will not be generated during normal emulation).

Select [Yes] to return the error value (all the properties are set to [Yes] by default).

- [Generate FlashBlockErase error]
- [Generate FlashBlockVerify error]
- [Generate FlashWordWrite error]
- [Generate FlashBlockBlankCheck error]
- [Generate FlashSetInfo error]
- [Generate FlashFLMDCheck error]

(c) [Security Flag Emulation]

You can configure the function on the security flag emulation in this category.

The initial value of the security flag is emulated when the security has been set to the flash memory.

Figure 2-29. [Security Flag Emulation] Category

<1> [Disable chip erase]

Specify whether to emulate to disable chip erase from the drop-down list.

Select [Yes] to emulate to disable chip erase ([No] is selected by default).

<2> [Disable block erase]

Specify whether to emulate to disable block erase from the drop-down list.

Select [Yes] to emulate to disable block erase ([No] is selected by default).

<3> [Disable read]

This property does not appear when the flash memory process is Type1 and the specification type of the flash self programming is Ver.2.00 or earlier[Note].

Specify whether to emulate to disable reading from the drop-down list.

Select [Yes] to emulate to disable reading ([No] is selected by default).

Note The determination of the flash memory process is made when connected to the debug tool.
Consequently, this will not be hidden when disconnected from the debug tool.

<4> [Disable program]

Specify whether to emulate to disable writing from the drop-down list.

Select [Yes] to emulate to disable writing ([No] is selected by default).

<5> [Disable boot block cluster reprogram]

This property appears only when the flash memory process is Type2 or Type3[Note].

Specify whether to emulate to disable rewrite boot area from the drop-down list.

Select [Yes] to emulate to disable rewrite boot area ([No] is selected by default).

Note The determination of the flash memory process is made when connected to the debug tool.
Consequently, this will not be hidden when disconnected from the debug tool.

(d) [Reset Vector/Boot Swap]

You can configure the reset vector and boot swap function in this category.

Figure 2-30. [Reset Vector/Boot Swap] Category

<1> [Reset vector address]

This property appears only when the flash memory process is Type1 and the specification type of the flash self programming is Ver.3.00 or later[Note].

Specify the reset vector address.

Directly enter the value in hexadecimal number from 0x0 to 0xFFFFFFFF ([0] is specified by default).

Note   The determination of the flash memory process is made when connected to the debug tool. Consequently, this will not be hidden when disconnected from the debug tool.

<2> [Boot swap cluster]

This property appears only when the flash memory process is Type4[Note].

Specify the target area of boot swap clusters.

Directly enter the value in decimal number from 0 to 127 ([0] is specified by default). The value set in this property is reflected to the target area in a boot block cluster. Prohibition of boot area rewriting can be set to the target area of boot swap clusters by using the microcontroller's security function.

Table 2-2. below shows the relationship between the value set in this property and the target area of boot block clusters.

Note   The determination of the flash memory process is made when connected to the debug tool. Consequently, this will not be hidden when disconnected from the debug tool.

Table 2-2.   Relationship between Boot Swap Cluster and Boot Block Cluster

Value to Be Set	When The Product's Flash Memory Size Is 256 KBytes or Smaller		When The Product's Flash Memory Size Is 384 KBytets or Larger	
	Boot Block Clusters	Boot Swap Clusters	Boot Block Clusters	Boot Swap Clusters
0	00000H - 007FFH	00000H - 01FFFH	00000H - 007FFH	RESV - 03FFFH
1	RESV - 00FFFH		RESV - 01FFFH	
2	RESV - 017FFH		RESV - 02FFFH	
3	RESV - 01FFFH		RESV - 03FFFH	
4	RESV - 027FFH	00000H - 03FFFH	RESV - 04FFFH	RESV - 07FFFH
:	:		:	
7	RESV - 03FFFH		RESV - 07FFFH	
8	RESV - 047FFFH	00000H - 07FFFH	RESV - 08FFFH	RESV - 0FFFFH
:	:		:	
15	RESV - 07FFFH		RESV - 0FFFFH	
16	RESV - 087FFH	00000H - 0FFFFH	RESV - 10FFFH	RESV - 1FFFFH
:	:		:	
31	RESV - 0FFFFH		RESV - 1FFFFH	
32	RESV - 107FFH		RESV - 20FFFH	
:	:		:	
127	RESV - 3FFFFH		RESV - 7FFFFH	

(4) [DataFlash Emulation Settings] tab [IECUBE]

In the [DataFlash Emulation Settings] tab [IECUBE], general configurations on the data flash emulation function can be done.

Note that this tab appears only when the selected microcontroller incorporates the data flash memory.

(a) [DataFlash Emulation]

You can configure the data flash emulation function in this category.

Figure 2-31. [DataFlash Emulation] Category

<1> [DataFlash emulation]

Specify whether to use the data flash emulation function from the drop-down list.

Select [Yes] to use the data flash emulation function ([No] is selected by default).

(b) [EEPROM Library Error]

You can configure the EEPROM emulation library in this category.

Specify the operation for the EEPROM emulation library function.

For details on the self programing library functions, see "V850 Microcontrollers Flash Memory Self Programming Library Type0*x*".

Figure 2-32. [EEPROM Library Error] Category

With the properties shown below, specify whether to return error values in the corresponding functions to emulate from the drop-down list (the error values in the functions will not be generated during normal emulation). All properties below are set to [No] by default.

Select [Yes] to return the error values forcibly, then specify an address within the data flash memory, at which the corresponding function error is to be generated, in the [Address generated *XXX* error] property which is listed newly in the lower area. Directly enter the address in hexadecimal number from 0x0 to 0xFFFFFFFF ([0] is specified by default).

- [Generate WordWrite error]
- [Generate BlockErase error]
- [Generate BlockIVerify error]
- [Generate BlockBlankCheck error]

(c) [Writing Time / Erasing Time]

You can configure the delay time for writing to and erasing the data flash memory in this category.

Figure 2-33.   [Writing Time / Erasing Time] Category

<1> [Writing time]

Specify the delay time when writing to the data flash memory from the following drop-down list.

No retry	Specifies "0" as the number of times of retry.
	The delay time is 0 (the writing speed is fastest).
Typical number of times that is assumed by flash macro specifications	Specifies the typical number of times that is assumed by flash macro specifications (default).
Maximum number of times that is assumed by flash macro specifications	Specifies the maximum number of times that is assumed by flash macro specifications.
Retries for the maximum number of times specified	Specifies the maximum number of times of retry.
	The delay time is maximum (the writing speed is longest).

<2> [Erasing time]

Specify the delay time when erasing the data flash memory from the following drop-down list.

No retry	Specifies "0" as the number of times of retry.
	The delay time is 0 (the erasing speed is fastest).
Typical number of times that is assumed by flash macro specifications	Specifies the typical number of times that is assumed by flash macro specifications (default).
Maximum number of times that is assumed by flash macro specifications	Specifies the maximum number of times that is assumed by flash macro specifications.
Retries for the maximum number of times specified	Specifies the maximum number of times of retry.
	The delay time is maximum (the erasing speed is longest).

(5) [Download File Settings] tab

In the [Download File Settings] tab, configuration on downloading file to the debug tool can be done.

See "2.5.1   Execute downloading" for details on each category configuration.

(6) [Flash Options Settings] tab [IECUBE2]

In the [Flash Options Settings] tab [V850E2M], configuration on options for the flash memory incorporated in the V850E2M microcontroller can be done. To configure options, specify the corresponding items on the Flash Options Setting dialog box [V850E2M], that is opened by clicking the [...] button appears at the right of the field by selecting the [Flash options] property in the [Flash Options] category on this tab (the [...] button appears only while connecting to the debug tool).

Click the [Write] button on this dialog box after specifying each item.

See Flash Options Setting dialog box [V850E2M] for details on the configuration.

Figure 2-34.   Opening the Flash Options Setting Dialog Box

Figure 2-35.   Flash Options Settings (Flash Options Setting Dialog Box [IECUBE2])
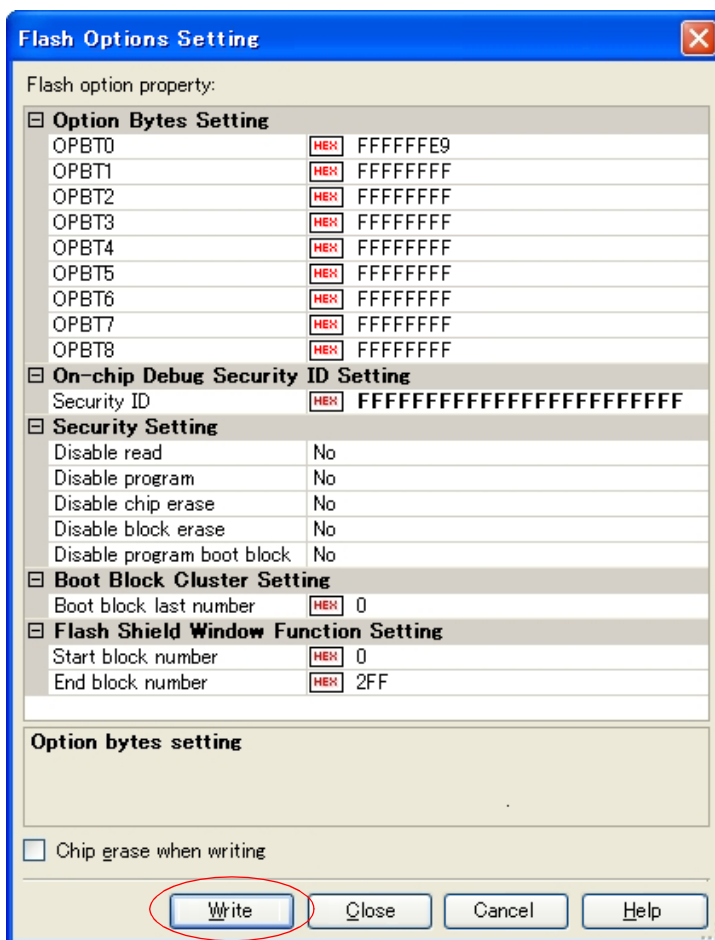
(7) [Hook Transaction Settings] tab

In the [Hook Transaction Settings] tab, configuration on the hook transaction can be done.

See "2.16   Use Hook Function" for details on each category configuration and the function of the hook transaction.

2.3.3 [MINICUBE]/[MINICUBE2]

Configure the operating environment on the Property panel below when using MINICUBE/MINICUBE2.

Figure 2-36.   Property Panel [MINICUBE[V850E1][V850ES]]
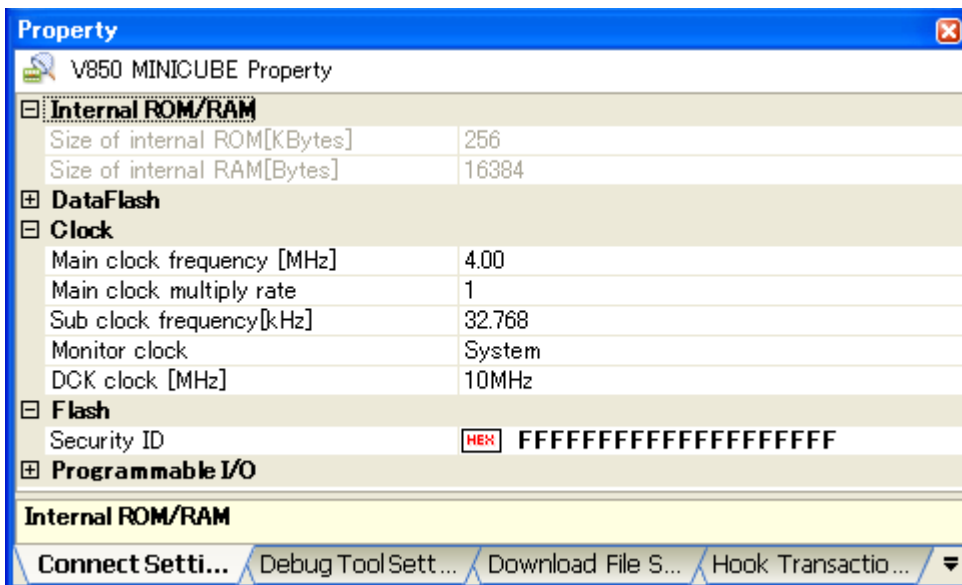
Figure 2-37.   Property Panel [MINICUBE[V850E2M]]
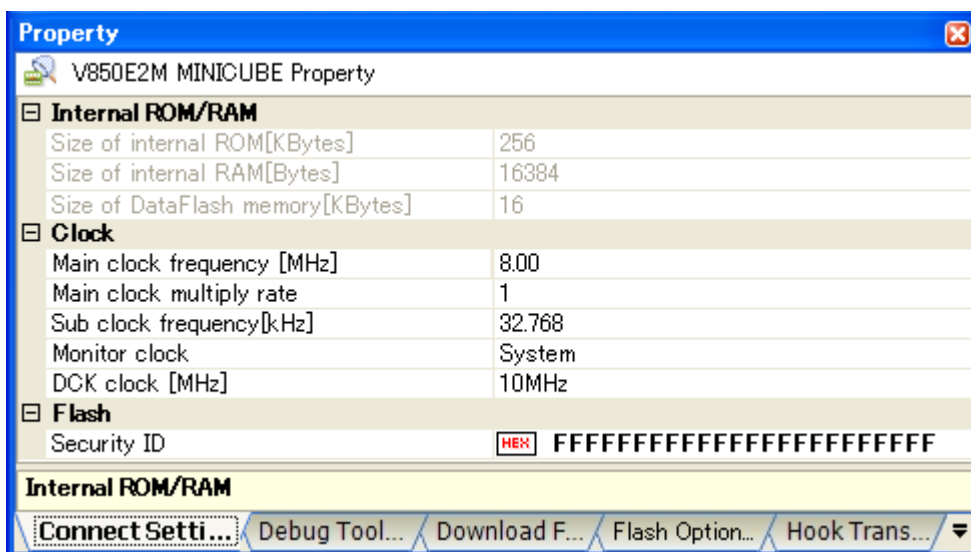
Figure 2-38.   Property Panel [MINICUBE2]

Follow the steps below by selecting the corresponding tab on the Property panel.

    (1) [Connect Settings] tab

    (2) [Debug Tool Settings] tab

    (3) [Download File Settings] tab

    (4) [Flash Options Settings] tab [MINICUBE[V850E2M]]

    (5) [Hook Transaction Settings] tab

(1) [Connect Settings] tab

    In the [Connect Settings] tab, configuration with regard to the connection to the debug tool can be done.

    (a) [Internal ROM/RAM]

    (b) [DataFlash] [V850E1][V850ES]

    (c) [Clock]

    (d) [Connection with Target Board]

    (e) [Flash]

    (f) [Programmable I/O] [V850E1][V850ES]

    (a) [Internal ROM/RAM]

    The configuration of internal ROM/RAM is displayed in this category.

      Figure 2-39.   [Internal ROM/RAM] Category [MINICUBE[V850E1][V850ES]][MINICUBE2]

Figure 2-40. [Internal ROM/RAM] Category [MINICUBE[V850E2M]]

Internal ROM/RAM	
Size of internal ROM[KBytes]	256
Size of internal RAM[Bytes]	16384
Size of DataFlash memory[KBytes]	16

<1> **[Size of internal ROM[KBytes]**

The internal ROM size to emulate is displayed (unit: Kbytes).

If the selected microcontroller is the V850E1 core and incorporates VSB Flash ROM, it displays the value including the size of VSB Flash ROM.

You cannot change the value of this property.

<2> **[Size of internal RAM[Bytes]]**

The internal RAM size to emulate is displayed (unit: bytes).

If the selected microcontroller is the V850E1 core and incorporates VSB RAM, it displays the value including the size of VSB RAM. If the selected microcontroller version supports multi-core, it displays the total size of the internal RAMs of all PE.

You cannot change the value of this property.

<3> **[Size of DataFlash memory[KBytes]] [MINICUBE[V850E2M]]**

The data flash memory size is displayed (unit: Kbytes).

If the currently selected microcontroller does not incorporate the data flash, [0] is displayed.

You cannot change the value of this property.

(b) **[DataFlash] [V850E1][V850ES]**

You can configure the data flash function in this category.

Note that this category appears only when the selected microcontroller incorporates the data flash memory.

Figure 2-41. [DataFlash] Category

DataFlash	
Use DataFlash memory area	**Yes**
Size of DataFlash memory[KBytes]	32
Chip select	CS0

<1> **[Use DataFlash memory area]**

Specify whether to use the data flash memory from the drop-down list.

Select [Yes] to use the data flash memory ([No] is selected by default).

<2> **[Size of DataFlash memory[KBytes]]**

This property appears only when the [Use DataFlash memory area] property is set to [Yes].

The size of the data flash memory area of the selected microcontroller is displayed (unit: Kbytes).

You cannot change the value of this property.

<3> **[Chip select]**

This property appears only when the [Use DataFlash memory area] property is set to [Yes].

Select the chip select used for mapping of the data flash memory from the drop-dawn list.

The selectable chip select values differ depending on the selected microcontroller.

Note that when the selected microcontroller is other than the V850ES core, you cannot change the value of the property.

(c) [Clock]

You can configure the clock in this category.

Figure 2-42. [Clock] Category [MINICUBE]

Clock	
Main clock frequency [MHz]	4.00
Main clock multiply rate	1
Sub clock frequency[kHz]	32.768
Monitor clock	System
TCK/DCK clock [MHz]	10MHz

Figure 2-43. [Clock] Category [MINICUBE2]

Clock	
Main clock frequency [MHz]	4.00
Main clock multiply rate	1
Sub clock frequency[kHz]	32.768
Monitor clock	System

<1> [Main clock frequency [MHz]]

Specify the main clock frequency (before multiplier).

You can specify the frequency from the drop-down list or by directly entering a frequency value between 0.001 and 999.999 (unit: MHz).

The drop-down list displays the following frequencies (unit: MHz).

- **[V850E1][V850ES]**

2.50, 4.00 (default), 5.00, 6.00, 8.00, 13.50, 18.00

- **[V850E2M]**

4.00, 5.00, 7.20, 8.00 (default), 9.60, 10.00, 13.50, 16.00

<2> [Main clock multiply rate]

Specify the main clock frequency multiplier.

Select a value from the drop-down list or directly enter a value from 1 to 99.

The drop-down list displays the following multiplier values.

- **[V850E1][V850ES]**

1 (default), 2, 3, 4, 5, 6, 7, 8, 9, 10

- **[V850E2M]**

1 (default), 2, 3, 4, 5, 6, 7, 8, 9, 10, 20

<3> [Sub clock frequency[kHz]]

This property appears only when the selected microcontroller supports a sub clock.

Specify the sub clock frequency.

You can specify the frequency from the drop-down list or by directly entering a frequency value number between 0.001 and 999.999 (unit: kHz).

The usable sub clock frequency values differ depending on the selected microcontroller.

The drop-down list displays sub clock frequency values which can be used by the currently selected microcontroller.

The minimum usable sub clock frequency is specified by default.

<4> [Monitor clock]

This property appears only when the selected microcontroller supports a sub clock.

Specify a clock for monitor programs to operate while the program is stopped.

Specify from the following drop-down list.

System	Operates with main clock (default).
User	Operates with the clock that the program specified.

<5> [TCK/DCK clock[MHz]] [MINICUBE]

Specify a clock supplied from MINICUBE to the on-chip debug unit (DCU) from the drop-down list.

The drop-down list displays the following frequency values.

10MHz (default), 20MHz

Cautions 1.   Normally, do not change the setting from [10MHz].

When [20MHz] is specified, it may not be possible to connect to the debug tool.

2.   This property cannot be changed while connecting to MINICUBE.

(d) [Connection with Target Board]

You can configure the connection to the target board in this category.

Figure 2-44.   [Connection with Target Board] Category [MINICUBE2]

<1> [Communication method]

Specify the port from the drop-down list to which to connect MINICUBE2 when performing serial communication between MINICUBE2 and the device in the target system.

The type of the port in the drop-down list differs depending on the selected microcontroller.

Note, however, that if the port that can be used for serial communication is limited by the currently selected microcontroller, it is not possible to change the communication method displayed by default (the default communication method depends on the currently selected microcontroller).

Caution   This property cannot be changed while connecting to MINICUBE2.

(e) [Flash]

You can configure the flash memory rewriting in this category.

Figure 2-45.   [Flash] Category [MINICUBE[V850E1][V850ES]]

Figure 2-46.   [Flash] Category [MINICUBE[V850E2M]]

   <1> [Security ID]

This property appears only when the selected microcontroller supports the ROM security function (on-chip debug security ID) for flash memory.

Specify a security ID for reading codes in the internal ROM or internal flash memory.

For details on the on-chip debug security ID, see MINICUBE/MINICUBE2 User's Manual.

    - **[V850E1][V850ES]**

Directly enter 20 digits hexadecimal number (10 bytes) ([FFFFFFFFFFFFFFFFFFFF] is specified by default).

    - **[V850E2M]**

Directly enter 24 digits hexadecimal number (12 bytes) ([FFFFFFFFFFFFFFFFFFFFFFFF] is specified by default).

Caution   This property cannot be changed while connecting to MINICUBE/MINICUBE2.

Remark   [V850E1][V850ES]

When you change the setting of the property, a message appears prompting you to select whether or not to change the security ID of the build tool.

   <2> [Chip erase when starting] [MINICUBE2]

Specify whether to erase the chip when connecting to the debug tool from the drop-down list.

Select [Yes] to erase the chip ([No] is selected by default).

Caution   This property cannot be changed while connecting to MINICUBE2.

(f) [Programmable I/O] [V850E1][V850ES]

You can configure the programmable I/O area in this category.

Note that this category is only valid when the selected microcontroller supports the programmable I/O area.

Remark   Change this setting to temporarily change variables while debugging.

To set a common value for the project, in the Property panel of the build tool, on the [Common Options] tab, in the [Device] category, set the value in [Programmable I/O area start address] property.

Figure 2-48.   [Programmable I/O] Category

   <1> [Use Programmable I/O area]

Specify whether to use the programmable I/O area.

Select [Yes] to use the programmable I/O are ([No] is selected by default).

<2> **[Programmable I/O area start address]**

This property appears only when the [Use Programmable I/O area] property is set to [Yes].

Specify the start address of the programmable I/O area.

Specify the address expression within the programmable I/O area ([0] is specified by default).

The address is aligned to 16 Kbytes.

(2) [Debug Tool Settings] tab

In the [Debug Tool Settings] tab, general configurations on the debug tool can be done.

(a) [Memory]
(b) [Access Memory While Running]
(c) [Set Event While Running]
(d) [Break]
(e) [Mask for Input Signal]
(f) [External Flash Memory Download] [MINICUBE]

(a) [Memory]

You can configure the memory in this category.

Figure 2-49.   [Memory] Category

<1> **[Memory mappings]**

Current memory mapping status is displayed for each type of memory area.

The memory mapping status cannot be changed on this panel. If it is necessary to add a memory mapping, click on the [Memory Mapping] property, and click on the [...] button that appears on the right end of the setting field. The Memory Mapping dialog box opens; perform the setting from there.

See the section for the Memory Mapping dialog box for details on how to configure the parameters.

Figure 2-50.   Opening the Memory Mapping Dialog Box

Caution   **If you are not connected to a debug tool, then only memory mapping areas added by user is displayed.**

Connecting to a debug tool (see "2.4.1   Connect the debug tool to CubeSuite") will display details for each memory type.

<2> **[Verify on writing to memory]**

Specify whether to perform a verify check when data has been written to memory from the drop-down list.

Select [Yes] to perform verification (default).

> **Remark**   The verification is performed at when the download is performed, or when the values that have been modified on the Watch panel/Memory panel are written to the memory.

(b) [Access Memory While Running]

You can configure the memory access while executing a program in this category.

The settings of this category are required when using the real-time display update function. See "(4)   Display/ modify the memory contents during program execution" for details on the real-time display update function.

Figure 2-51.   [Access Memory While Running] Category [MINICUBE]

Access Memory While Running	
Access by stopping execution	Yes
Update display during the execution	Yes
Display update interval[ms]	500
Access by release HALT mode	Yes

Figure 2-52.   [Access Memory While Running] Category [MINICUBE2]

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

<1> [Access by stopping execution]

Specify whether to allow access to the memory area while executing a program.

Select [Yes] to allow access ([No] is selected by default).

> **Caution   [V850E2M]**
>
> **This property is not displayed when the selected microcontroller version supports multi-core. Also note that the target range for the real-time display update function is limited to the internal RAM area and the HBUS shared memory area.**

<2> [Update display during the execution]

Specify whether to update the display in the Watch panel/Memory panel while executing a program.

Select [Yes] to update the display (default).

<3> [Display update interval[ms]]

This property is valid only when the [Update display during the execution] property is set to [Yes].

Specify the interval in 100ms unit to update the contents in the Watch panel/Memory panel display while executing a program.

Directly enter the Integer number between 100 and 65500 (rounding up the fractions less than 100ms) ([500] is specified by default).

<4> [Access by release HALT mode] [MINICUBE]

This property appears only when the [Access by stopping execution] property is set to [Yes].

Specify whether to allow access to the memory area/CPU registers/I/O registers by releasing the HALT mode from the drop-down list.

Select [Yes] to allow access to the memory area/CPU registers/I/O registers by releasing the HALT mode (default).

(c) [Set Event While Running]

You can configure the setting of events while executing a program in this category.

Figure 2-53.   [Set Event While Running] Category
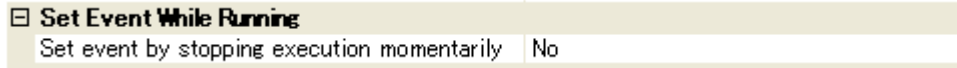
<1> [Set event by stopping execution momentarily]

Specify whether to forcibly pause the execution for events that cannot be set while executing a program or operating the tracer/timer.

For details on the event types that are affected by this property, see "Table 2-25.   Event Types That Can be Set and Deleted during Execution [V850E1][V850ES]".

Select [Yes] to set events above while execution ([No] is selected by default).

(d) [Break]

You can configure the break function in this category.

Figure 2-54.   [Break] Category [MINICUBE]
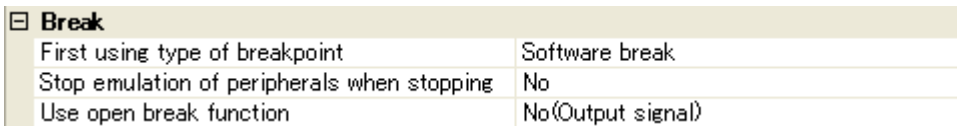
Figure 2-55.   [Break] Category [MINICUBE2]

<1> [First using type of breakpoint]

Specify from the following drop-down list a breakpoint type to use with priority when setting it with a one click operation of the mouse in the Editor panel/Disassemble panel.

See "2.8.2   Stop the program at the arbitrary position (breakpoint)" for details on breakpoints.

Software break	Sets software breakpoint with priority (default).
Hardware break	Sets hardware breakpoint with priority.

<2> [Stop emulation of peripherals when stopping] [MINICUBE]

Specify from the drop-down list whether to terminate the peripheral emulation while stopping the program execution.

Select [Yes] to terminate ([No] is selected by default).

<3> [Use open break function] [MINICUBE]

Specify from the following drop-down list whether to use the open break function.

Yes(Hi-Z)	The open break target pin becomes the Hi-Z state after the CPU is stopped.
No(Output signal)	The open break target pin outputs the signal even after the CPU is stopped.

(e) [Mask for Input Signal]

You can configure the input signal masking in this category.

Figure 2-56.   [Mask for Input Signal] Category [MINICUBE[V850E1][V850ES]]

Mask for Input Signal	
Mask NMI0 signal	No
Mask NMI1 signal	No
Mask NMI2 signal	No
Mask HLDRQ signal	No
Mask RESET signal	No
Mask STOP signal	No
Mask WAIT signal	No
Mask DBINT signal	No

Figure 2-57.   [Mask for Input Signal] Category [MINICUBE[V850E2M]]

Mask for Input Signal	
Mask HLDRQ signal	No
Mask RESET signal	No
Mask STOP signal	No
Mask WAIT signal	No

Figure 2-58.   [Mask for Input Signal] Category [MINICUBE2]

Mask for Input Signal	
Mask RESET signal	No

With the properties shown below, select [Yes] to mask the signal from the drop-down list (all properties below are set to [No] by default).

- [Mask NMI*n* signal] **[MINICUBE[V850E1][V850ES]]**
- [Mask HLDRQ signal] **[MINICUBE]**
- [Mask RESET signal]
- [Mask STOP signal] **[MINICUBE]**
- [Mask WAIT signal] **[MINICUBE]**
- [[Mask DBINT signal] **[MINICUBE[V850E1][V850ES]]**

(f) [External Flash Memory Download] [MINICUBE]

You can configure downloading to an external flash memory in this category.

See "(5)   Download to external flash memory [IECUBE2][MINICUBE][E1 (JTAG)]" for details on how to download to an external flash memory and this category configuration.

(3) [Download File Settings] tab

In the [Download File Settings] tab, configuration on downloading file to the debug tool can be done.

See "2.5.1   Execute downloading" for details on each category configuration.

(4) [Flash Options Settings] tab [MINICUBE[V850E2M]]

In the [Flash Options Settings] tab [V850E2M], configuration on options for the flash memory incorporated in the V850E2M microcontroller can be done. To configure options, specify the corresponding items on the Flash Options Setting dialog box [V850E2M], that is opened by clicking the [...] button appears at the right of the field by

selecting the [Flash options] property in the [Flash Options] category on this tab (the [...] button appears only while connecting to the debug tool).

Click the [Write] button on this dialog box after specifying each item.

See Flash Options Setting dialog box [V850E2M] for details on the configuration.

Figure 2-59.   Opening the Flash Options Setting Dialog Box

Figure 2-60.   Flash Options Settings (Flash Options Setting Dialog Box [MINICUBE[V850E2M]])
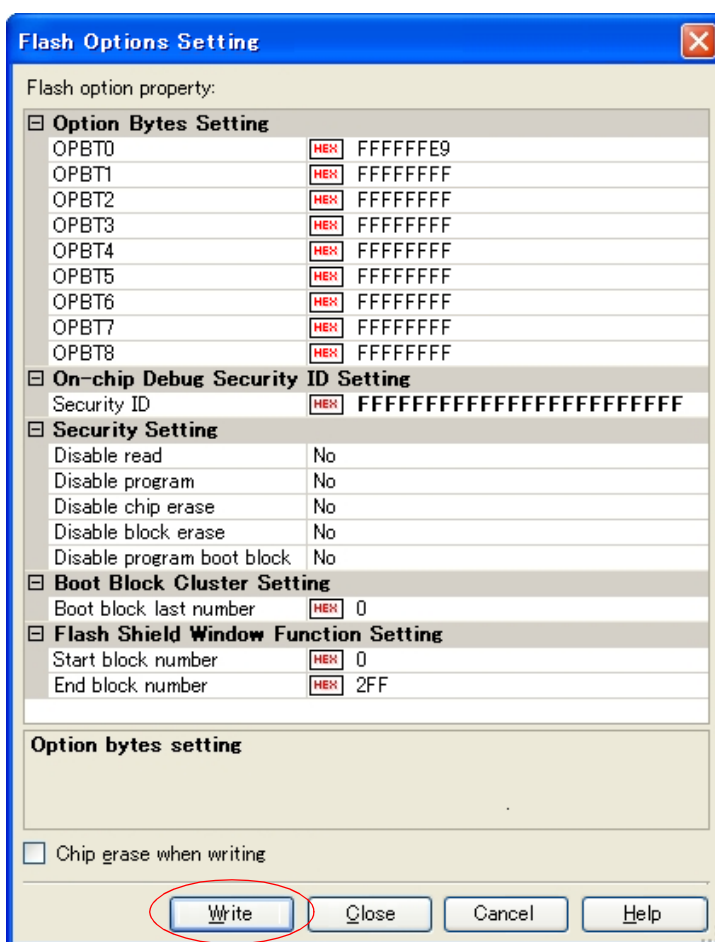
(5) [Hook Transaction Settings] tab

In the [Hook Transaction Settings] tab, configuration on the hook transaction can be done.

See "2.16   Use Hook Function" for details on each category configuration and the function of the hook transaction.

2.3.4   [E1]

Configure the operating environment on the Property panel below when using E1.

Note that the contents of the Property panel differ depending on the communication method (serial communications **[E1 (Serial)]** or JTAG communications **[E1 (JTAG)]**) between E1 and the target system.

Figure 2-61.   Property Panel [E1 (Serial)]

Property	
V850 E1(Serial) Property	
⊟ Internal ROM/RAM	
Size of internal ROM[KBytes]	256
Size of internal RAM[Bytes]	16384
⊞ DataFlash	
⊟ Clock	
Main clock frequency [MHz]	4.00
Main clock multiply rate	1
Sub clock frequency[kHz]	32.768
Monitor clock	System
⊟ Connection with Target Board	
Communication method	UARTA0
Power target from the emulator (MAX 200mA)	No
⊟ Flash	
Security ID	[HEX] FFFFFFFFFFFFFFFFFFFF
Chip erase when starting	No
⊞ Programmable I/O	
Internal ROM/RAM	

Connect Setti... / Debug Tool Setti... / Download File S... / Hook Transactio... / ▾

Figure 2-62.   Property Panel [E1 (JTAG)][V850E1][V850ES]]

Property	
V850 E1(JTAG) Property	
⊟ Internal ROM/RAM	
Size of internal ROM[KBytes]	256
Size of internal RAM[Bytes]	16384
⊞ DataFlash	
⊟ Clock	
Main clock frequency [MHz]	4.00
Main clock multiply rate	1
Sub clock frequency[kHz]	32.768
Monitor clock	System
TCK/DCK clock [MHz]	10MHz
⊟ Connection with Target Board	
Power target from the emulator (MAX 200mA)	No
⊟ Flash	
Security ID	[HEX] FFFFFFFFFFFFFFFFFFFF
⊞ Programmable I/O	
Internal ROM/RAM	

Connect Setti... / Debug Tool Setti... / Download File S... / Hook Transactio... / ▾

Figure 2-63.   Property Panel [E1 (JTAG)[V850E2M]]

Follow the steps below by selecting the corresponding tab on the Property panel.

    (1)   [Connect Settings] tab

    (2)   [Debug Tool Settings] tab

    (3)   [Download File Settings] tab

    (4)   [Flash Options Settings] tab [E1 (JTAG)[V850E2M]]

    (5)   [Hook Transaction Settings] tab

(1) [Connect Settings] tab

    In the [Connect Settings] tab, configuration with regard to the connection to the debug tool can be done.

        (a)   [Internal ROM/RAM]

        (b)   [DataFlash] [V850E1][V850ES]

        (c)   [Clock]

        (d)   [Connection with Target Board]

        (e)   [Flash]

        (f)   [Programmable I/O] [V850E1][V850ES]

    (a) [Internal ROM/RAM]

        The configuration of internal ROM/RAM is displayed in this category.

        Figure 2-64.   [Internal ROM/RAM] Category [E1 (Serial)][E1 (JTAG)[V850E1][V850ES]]

Figure 2-65. [Internal ROM/RAM] Category [E1 (JTAG)[V850E2M]]

Internal ROM/RAM	
Size of internal ROM[KBytes]	256
Size of internal RAM[Bytes]	16384
Size of DataFlash memory[KBytes]	16

<1> **[Size of internal ROM[KBytes]**

The internal ROM size to emulate is displayed (unit: Kbytes).

If the selected microcontroller is the V850E1 core and incorporates VSB Flash ROM, it displays the value including the size of VSB Flash ROM.

You cannot change the value of this property.

<2> **[Size of internal RAM[Bytes]]**

The internal RAM size to emulate is displayed (unit: bytes).

If the selected microcontroller is the V850E1 core and incorporates VSB RAM, it displays the value including the size of VSB RAM. If the selected microcontroller version supports multi-core, it displays the total size of the internal RAMs of all PE.

You cannot change the value of this property.

<3> **[Size of DataFlash memory[KBytes]] [E1 (JTAG)[V850E2M]]**

The data flash memory size is displayed (unit: Kbytes).

If the currently selected microcontroller does not incorporate the data flash, [0] is displayed.

You cannot change the value of this property.

(b) [DataFlash] [V850E1][V850ES]

You can configure the data flash function in this category.

Note that this category appears only when the selected microcontroller incorporates the data flash memory.

Figure 2-66. [DataFlash] Category

DataFlash	
Use DataFlash memory area	**Yes**
Size of DataFlash memory[KBytes]	32
Chip select	CS0

<1> **[Use DataFlash memory area]**

Specify whether to use the data flash memory from the drop-down list.

Select [Yes] to use the data flash memory ([No] is selected by default).

<2> **[Size of DataFlash memory[KBytes]]**

This property appears only when the [Use DataFlash memory area] property is set to [Yes].

The size of the data flash memory area of the selected microcontroller is displayed (unit: Kbytes).

You cannot change the value of this property.

<3> **[Chip select]**

This property appears only when the [Use DataFlash memory area] property is set to [Yes].

Select the chip select used for mapping of the data flash memory from the drop-dawn list.

The selectable chip select values differ depending on the selected microcontroller.

Note that when the selected microcontroller is other than the V850ES core, you cannot change the value of the property.

(c) [Clock]

You can configure the clock in this category.

Figure 2-67. [Clock] Category [E1 (Serial)]

Clock	
Main clock frequency [MHz]	4.00
Main clock multiply rate	1
Sub clock frequency[kHz]	32.768
Monitor clock	System

Figure 2-68. [Clock] Category [E1 (JTAG)]

Clock	
Main clock frequency [MHz]	4.00
Main clock multiply rate	1
Sub clock frequency[kHz]	32.768
Monitor clock	System
TCK/DCK clock [MHz]	10MHz

<1> [Main clock frequency [MHz]]

Specify the main clock frequency (before multiplier).

You can specify the frequency from the drop-down list or by directly entering a frequency value between 0.001 and 999.999 (unit: MHz).

The drop-down list displays the following frequencies (unit: MHz).

- **[V850E1][V850ES]**

2.50, 4.00 (default), 5.00, 6.00, 8.00, 13.50, 18.00

- **[V850E2M]**

4.00, 5.00, 7.20, 8.00 (default), 9.60, 10.00, 13.50, 16.00

<2> [Main clock multiply rate]

Specify the main clock frequency multiplier.

Select a value from the drop-down list or directly enter a value from 1 to 99.

The drop-down list displays the following multiplier values.

- **[V850E1][V850ES]**

1 (default), 2, 3, 4, 5, 6, 7, 8, 9, 10

- **[V850E2M]**

1 (default), 2, 3, 4, 5, 6, 7, 8, 9, 10, 20

<3> [Sub clock frequency[kHz]]

This property appears only when the selected microcontroller supports a sub clock.

Specify the sub clock frequency.

You can specify the frequency from the drop-down list or by directly entering a frequency value number between 0.001 and 999.999 (unit: kHz).

The usable sub clock frequency values differ depending on the selected microcontroller.

The drop-down list displays sub clock frequency values which can be used by the currently selected microcontroller.

The minimum usable sub clock frequency is specified by default.

<4> [Monitor clock]

This property appears only when the selected microcontroller supports a sub clock.

Specify a clock for monitor programs to operate while the program is stopped.

Specify from the following drop-down list.

System	Operates with main clock (default).
User	Operates with the clock that the program specified.

<5> [TCK/DCK clock[MHz]] [E1 (JTAG)]

Specify a clock supplied from E1 (JTAG) to the on-chip debug unit (DCU) from the drop-down list.

The drop-down list displays the following frequency values.

10MHz (default), 20MHz

Cautions 1.   Normally, do not change the setting from [10MHz].

When [20MHz] is specified, it may not be possible to connect to the debug tool.

2.   This property cannot be changed while connecting to E1 (JTAG).

(d) [Connection with Target Board]

You can configure the connection between E1 and the target board in this category.

Caution   Properties in this category cannot be changed while connecting to E1.

Figure 2-69.   [Connection with Target Board] Category [E1 (Serial)]

⊟ Connection with Target Board	
Communication method	UARTA0
Power target from the emulator (MAX 200mA)	**Yes**
Supply voltage	3.3V

Figure 2-70.   [Connection with Target Board] Category [E1 (JTAG)]

⊟ Connection with Target Board	
Power target from the emulator (MAX 200mA)	**Yes**
Supply voltage	3.3V

<1> [Communication method] [E1 (Serial)]

Specify the port from the drop-down list to which to connect E1 (Serial) when performing serial communication between E1 (Serial) and the device in the target system.

The type of the port in the drop-down list differs depending on the selected microcontroller.

Note, however, that if the port that can be used for serial communication is limited by the currently selected microcontroller, it is not possible to change the communication method displayed by default (the default communication method depends on the currently selected microcontroller).

<2> [Power target from the emulator (MAX 200mA)]

Specify whether to supply power to the target system from E1.

Select [Yes] to supply power to the target system ([No] is selected by default).

<3> [Supply voltage]

This property appears only when the [Power target from the emulator (MAX 200mA)] property is set to [Yes].

Specify the power voltage supplied to the target system from the following drop-down list.
3.3V (default), 5.0V

(e) [Flash]
You can configure the flash memory rewriting in this category.

Figure 2-71.   [Flash] Category [E1 (Serial) [V850E1][V850ES]]

Figure 2-72.   [Flash] Category [E1 (JTAG)[V850E1][V850ES]]

Figure 2-73.   [Flash] Category [E1 (JTAG)[V850E2M]]

<1> [Security ID]
This property appears only when the selected microcontroller supports the ROM security function (on-chip debug security ID) for flash memory.
Specify a security ID for reading codes in the internal ROM or internal flash memory.
For details on the on-chip debug security ID, see E1 User's Manual.
- **[V850E1][V850ES]**
Directly enter 20 digits hexadecimal number (10 bytes) ([FFFFFFFFFFFFFFFFFFFF] is specified by default).
- **[V850E2M]**
Directly enter 24 digits hexadecimal number (12 bytes) ([FFFFFFFFFFFFFFFFFFFFFFFF] is specified by default).

Caution   **This property cannot be changed while connecting to E1.**

Remark   **[V850E1][V850ES]**
When you change the setting of the property, a message appears prompting you to select whether or not to change the security ID of the build tool.

<2> [Chip erase when starting] [E1 (Serial)]
Specify whether to erase the chip when connecting to the debug tool from the drop-down list.
Select [Yes] to erase the chip ([No] is selected by default).

Caution   **This property cannot be changed while connecting to E1 (Serial).**

(f) [Programmable I/O] [V850E1][V850ES]
You can configure the programmable I/O area in this category.
Note that this category is only valid when the selected microcontroller supports the programmable I/O area.

Remark   Change this setting to temporarily change variables while debugging.

To set a common value for the project, in the Property panel of the build tool, on the [Common Options] tab, in the [Device] category, set the value in [Programmable I/O area start address] property.

Figure 2-74.   [Programmable I/O] Category

⊟ Programmable I/O	
Use Programmable I/O area	**Yes**
Programmable I/O area start address	HEX 0

<1> [Use Programmable I/O area]

Specify whether to use the programmable I/O area.

Select [Yes] to use the programmable I/O are ([No] is selected by default).

<2> [Programmable I/O area start address]

This property appears only when the [Use Programmable I/O area] property is set to [Yes].

Specify the start address of the programmable I/O area.

Specify the address expression within the programmable I/O area ([0] is specified by default).

The address is aligned to 16 Kbytes.

(2) [Debug Tool Settings] tab

In the [Debug Tool Settings] tab, general configurations on the debug tool can be done.

 (a) [Memory]
 (b) [Access Memory While Running]
 (c) [Set Event While Running]
 (d) [Break]
 (e) [Mask for Input Signal]
 (f) [External Flash Memory Download] [E1 (JTAG)]

(a) [Memory]

You can configure the memory in this category.

Figure 2-75.   [Memory] Category

⊟ **Memory**	
⊟ Memory mappings	**[4]**
⊞ [0]	Internal ROM area
⊞ [1]	Non-map area
⊞ [2]	Internal RAM area
⊞ [3]	I/O Register area
Verify on writing to memory	Yes

<1> [Memory mappings]

Current memory mapping status is displayed for each type of memory area.

The memory mapping status cannot be changed on this panel. If it is necessary to add a memory mapping, click on the [Memory Mapping] property, and click on the [...] button that appears on the right end of the setting field. The Memory Mapping dialog box opens; perform the setting from there.

See the section for the Memory Mapping dialog box for details on how to configure the parameters.

Figure 2-76.   Opening the Memory Mapping Dialog Box

Caution   **If you are not connected to a debug tool, then only memory mapping areas added by user is displayed.**
Connecting to a debug tool (see "2.4.1   Connect the debug tool to CubeSuite") will display details for each memory type.

<2> [Verify on writing to memory]
Specify whether to perform a verify check when data has been written to memory from the drop-down list.
Select [Yes] to perform verification (default).

Remark   The verification is performed at when the download is performed, or when the values that have been modified on the Watch panel/Memory panel are written to the memory.

(b) [Access Memory While Running]
You can configure the memory access while executing a program in this category.
The settings of this category are required when using the real-time display update function. See "(4)   Display/ modify the memory contents during program execution" for details on the real-time display update function.

Figure 2-77.   [Access Memory While Running] Category [E1 (Serial)]

Figure 2-78.   [Access Memory While Running] Category [E1 (JTAG)]

<1> [Access by stopping execution]
Specify whether to allow access to the memory area while executing a program.
Select [Yes] to allow access ([No] is selected by default).

Caution   **[V850E2M]**
This property is not displayed when the selected microcontroller version supports multi-core. Also note that the target range for the real-time display update function is limited to the internal RAM area and the HBUS shared memory area.

<2> [Update display during the execution]

Specify whether to update the display in the Watch panel/Memory panel while executing a program.

Select [Yes] to update the display (default).

<3> [Display update interval[ms]]

This property is valid only when the [Update display during the execution] property is set to [Yes].

Specify the interval in 100ms unit to update the contents in the Watch panel/Memory panel display while executing a program.

Directly enter the Integer number between 100 and 65500 (rounding up the fractions less than 100ms) ([500] is specified by default).

<4> [Access by release HALT mode] [E1 (JTAG)]

This property appears only when the [Access by stopping execution] property is set to [Yes].

Specify whether to allow access to the memory area/CPU registers/I/O registers by releasing the HALT mode from the drop-down list.

Select [Yes] to allow access to the memory area/CPU registers/I/O registers by releasing the HALT mode (default).

(c) [Set Event While Running]

You can configure the setting of events while executing a program in this category.

Figure 2-79.   [Set Event While Running] Category
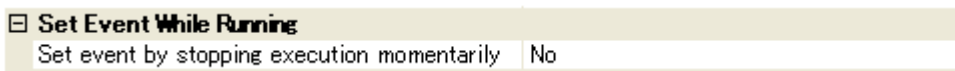
<1> [Set event by stopping execution momentarily]

Specify whether to forcibly pause the execution for events that cannot be set while executing a program or operating the tracer/timer.

For details on the event types that are affected by this property, see "Table 2-25.   Event Types That Can be Set and Deleted during Execution [V850E1][V850ES]".

Select [Yes] to set events above while execution ([No] is selected by default).

(d) [Break]

You can configure the break function in this category.

Figure 2-80.   [Break] Category [E1 (Serial)]

Figure 2-81.   [Break] Category [E1 (JTAG)]
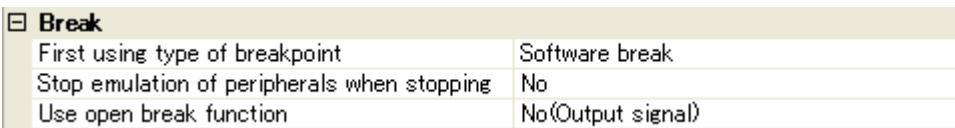
<1> [First using type of breakpoint]

Specify from the following drop-down list a breakpoint type to use with priority when setting it with a one click operation of the mouse in the Editor panel/Disassemble panel.

See "2.8.2   Stop the program at the arbitrary position (breakpoint)" for details on breakpoints.

Software break	Sets software breakpoint with priority (default).
Hardware break	Sets hardware breakpoint with priority.

<2> [Stop emulation of peripherals when stopping] [E1 (JTAG)]

Specify from the drop-down list whether to terminate the peripheral emulation while stopping the program execution.

Select [Yes] to terminate ([No] is selected by default).

<3> [Use open break function] [E1 (JTAG)]

Specify from the following drop-down list whether to use the open break function.

Yes(Hi-Z)	The open break target pin becomes the Hi-Z state after the CPU is stopped.
No(Output signal)	The open break target pin outputs the signal even after the CPU is stopped.

(e) [Mask for Input Signal]

You can configure the input signal masking in this category.

Figure 2-82.   [Mask for Input Signal] Category [E1 (Serial)]

Figure 2-83.   [Mask for Input Signal] Category [E1 (JTAG)[V850E1][V850ES]]

Figure 2-84.   [Mask for Input Signal] Category [E1 (JTAG)[V850E2M]]

With the properties shown below, select [Yes] to mask the signal from the drop-down list (all properties below are set to [No] by default).

- [Mask NMI*n* signal] **[E1 (JTAG)[V850E1][V850ES]]**

- [Mask HLDRQ signal] **[E1 (JTAG)]**

- [Mask RESET signal]

- [Mask STOP signal] **[E1 (JTAG)]**

- [Mask WAIT signal] **[E1 (JTAG)]**

- [[Mask DBINT signal] **[E1 (JTAG)][V850E1][V850ES]]**

(f)   [External Flash Memory Download] [E1 (JTAG)]

You can configure downloading to an external flash memory in this category.

See "(5)   Download to external flash memory [IECUBE2][MINICUBE][E1 (JTAG)]" for details on how to download to an external flash memory and this category configuration.

(3)   [Download File Settings] tab

In the [Download File Settings] tab, configuration on downloading file to the debug tool can be done.

See "2.5.1   Execute downloading" for details on each category configuration.

(4)   [Flash Options Settings] tab [E1 (JTAG)[V850E2M]]

In the [Flash Options Settings] tab [V850E2M], configuration on options for the flash memory incorporated in the V850E2M microcontroller can be done. To configure options, specify the corresponding items via the Flash Options Setting dialog box [V850E2M], that is opened by clicking the [...] button appears at the right of the field by selecting the [Flash options] property in the [Flash Options] category on this tab (the [...] button appears only while connecting to the debug tool).

Click the [Write] button on this dialog box after specifying each item.

See Flash Options Setting dialog box [V850E2M] for details on the configuration.

Figure 2-85.   Opening the Flash Options Setting Dialog Box

Figure 2-86. Flash Options Settings (Flash Options Setting Dialog Box [E1 (JTAG)[V850E2M]])
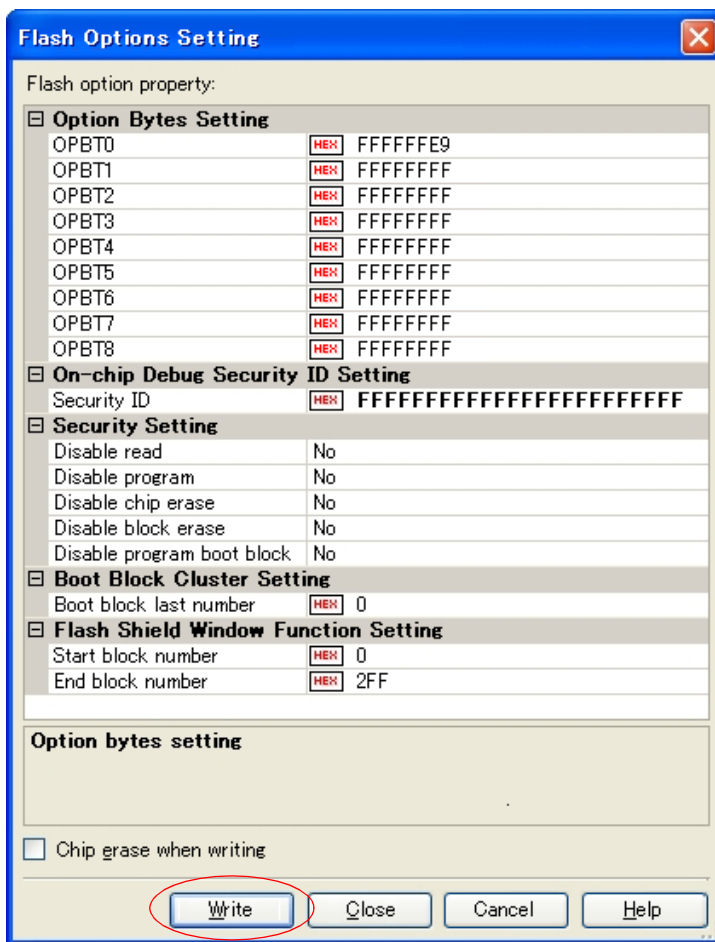
(5) [Hook Transaction Settings] tab

In the [Hook Transaction Settings] tab, configuration on the hook transaction can be done.

See "2.16 Use Hook Function" for details on each category configuration and the function of the hook transaction.

2.3.5 [Simulator]

Configure the operating environment on the Property panel below when using Simulator.

Figure 2-87. Property Panel [Simulator[V850E1][V850ES]]
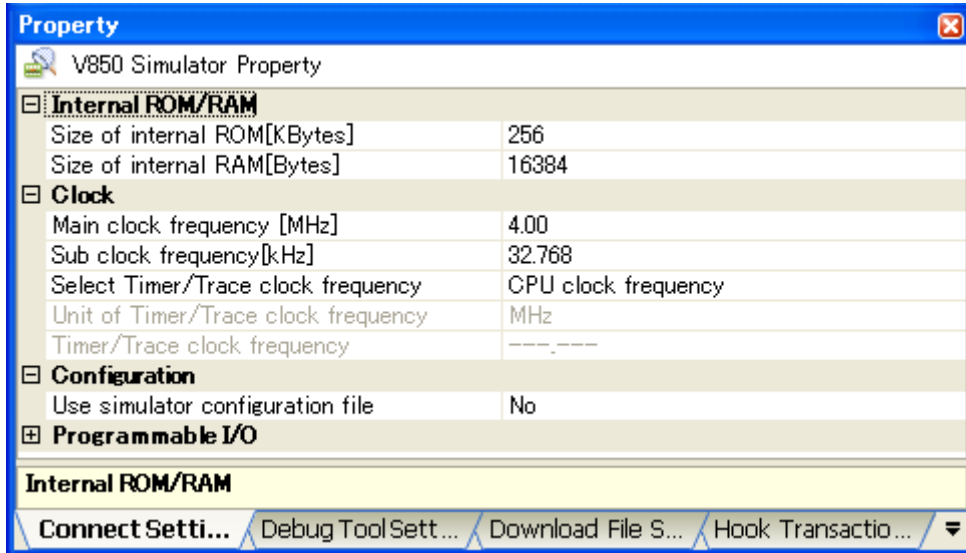
Figure 2-88. Property Panel [Simulator[V850E2M]]
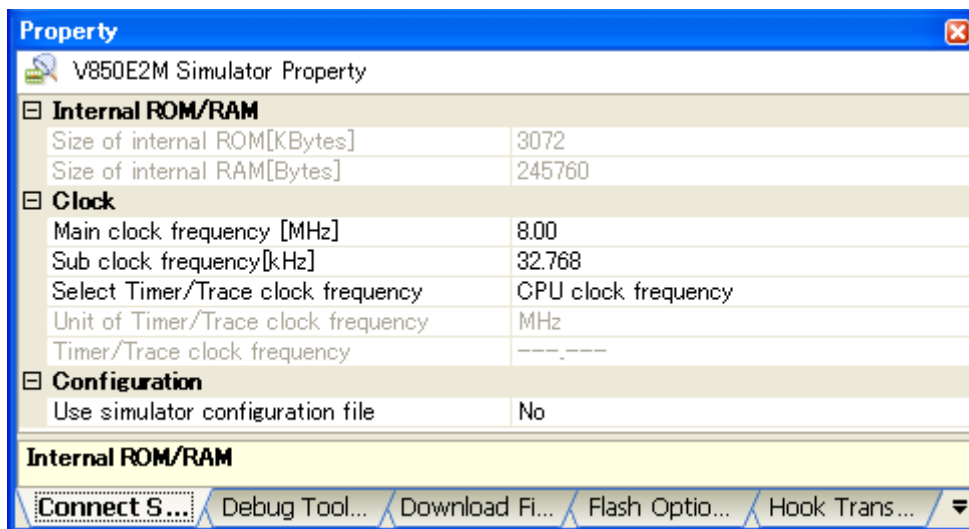
Follow the steps below by selecting the corresponding tab on the Property panel.

(1) [Connect Settings] tab
(2) [Debug Tool Settings] tab
(3) [Download File Settings] tab
(4) [Flash Options Settings] tab [V850E2M]
(5) [Hook Transaction Settings] tab

Remark When Simulator to be used corresponds to peripheral function simulations, you can use the Simulator GUI.
See "2.17 Use the Simulator GUI [Simulator]" for details on the Simulator GUI.

(1) [Connect Settings] tab

In the [Connect Settings] tab, configuration with regard to the connection to the debug tool can be done.

(a) [Internal ROM/RAM]
(b) [Clock]
(c) [Configuration]
(d) [Programmable I/O] [V850E1][V850ES]

(a) [Internal ROM/RAM]

You can configure internal ROM/RAM in this category.

The size of internal ROM/RAM of the selected microcontroller is specified by default (when the selected microcontroller is the V850E1 core and incorporates VSB Flash ROM/VSB RAM, the value including its size is specified).

There is no need to change the settings in this category if you wish to debug with the same memory mapping of the selected microcontroller.

Figure 2-89.   [Internal ROM/RAM] Category [Simulator[V850E1][V850ES]]

Internal ROM/RAM	
Size of internal ROM[KBytes]	256
Size of internal RAM[Bytes]	16384

Figure 2-90.   [Internal ROM/RAM] Category [Simulator[V850E2M]]

Internal ROM/RAM	
Size of internal ROM[KBytes]	3072
Size of internal RAM[Bytes]	245760

<1> [Size of internal ROM[KBytes]

- **[V850E1][V850ES]**

Specify the internal ROM size to simulate (unit: Kbytes).

To perform debugging after changing the memory mapping, make a selection form the drop-down list.

Note that when the selected microcontroller is a ROMless product, the property value cannot be changed.

- **[V850E2M]**

The internal ROM size to simulate is displayed (unit: Kbytes).

You cannot change the value of this property.

<2> [Size of internal RAM[Bytes]]

- **[V850E1][V850ES]**

Specify the internal RAM size to simulate (unit: bytes).

To perform debugging after changing the memory mapping, make a selection form the drop-down list.

- **[V850E2M]**

The internal RAM size to simulate is displayed (unit: bytes).

You cannot change the value of this property.

(b) [Clock]

You can configure the clock in this category.

Figure 2-91.   [Clock] Category [Simulator]

```
☐ Clock
   Main clock frequency [MHz]            4.00
   Sub clock frequency[kHz]             32.768
   Select Timer/Trace clock frequency   CPU clock frequency
   Unit of Timer/Trace clock frequency  MHz
   Timer/Trace clock frequency          ---.---
```

<1> [Main clock frequency [MHz]]

Specify the main clock frequency.

You can specify the frequency from the drop-down list or by directly entering a frequency value between 0.001 and 999.999 (unit: MHz).

The drop-down list displays the following frequencies (unit: MHz).

- **[V850E1][V850ES]**

  1.00, 2.00, 3.00, 3.57, 4.00 (default), 4.19, 4.91, 5.00, 6.00, 8.00, 8.38, 10.00, 12.00, 16.00, 20.00, 25.00, 30.00, 32.00, 33.33, 34.00, 40.00, 48.00, 50.00, 64.00, 80.00

- **[V850E2M]**

  1.00, 2.00, 3.00, 3.57, 4.00, 4.19, 4.91, 5.00, 6.00, 7.20, 8.00 (default), 8.38, 9.60, 10.00, 12.00, 16.00, 20.00, 25.00, 30.00, 32.00, 33.33, 34.00, 40.00, 48.00, 50.00, 64.00, 80.00

<2> [Sub clock frequency[kHz]]

This property appears only when the selected microcontroller supports a sub clock.

Specify the sub clock frequency.

You can specify the frequency from the drop-down list or by directly entering a frequency value between 0.001 and 999.999 (unit: kHz).

The drop-down list displays the following frequencies (unit: kHz).

   32.768 (default)

<3> [Select Timer/Trace clock frequency]

Specify the clock frequency for using timer/trace function.

Specify from the following drop-down list.

CPU clock frequency	Uses the CPU clock frequency (default).
Specify clock frequency	Specifies an arbitrary frequency (property items to specify become valid in the lower area).

<4> [Unit of Timer/Trace clock frequency]

This property appears only when the [Select Timer/Trace clock frequency] property is set to [Specify clock frequency].

Specify from the following drop-down list the unit of the clock frequency for timer/trace.

MHz	The unit of the frequency is in MHz (default).
KHz	The unit of the frequency is in kHz.

<5> [Timer/Trace clock frequency]

The operation of this property differs depending on the specification with the [Select Timer/Trace clock frequency] property.

- When [Specify clock frequency] is specified

Specify the clock frequency for timer/trace.

Directly enter the value between 1 [kHz] and 999.999 [MHz] ([4.00] is specified by default).

Unit is depending on the specification with the [Unit of Timer/Trace clock frequency] property.

- When [CPU clock frequency] is specified

While disconnecting from the debug tool, [---_---] is displayed. While connecting to the debug tool, [*CPU clock frequency*] is displayed.

(c) [Configuration]

You can configure the customization of the simulator in this category.

Figure 2-92.   [Configuration] Category

<1> [Use simulator configuration file]

Specify from the drop-down list whether to use the simulator configuration file to perform user customization (adding of user models) of the simulator.

Select [Yes] to use the simulator configuration file ([No] is selected by default).

<2> [Simulator configuration file]

This property appears only when the [Use simulator configuration file] property is set to [Yes]. Specify the simulator configuration file to use.

Directly enter the file name, or select the file with the Select Simulator Configuration File dialog box [Simulator] opened by clicking on the [...] button that appears on the right end of the setting field.

Caution   This property cannot be changed while connecting to Simulator.

(d) [Programmable I/O] [V850E1][V850ES]

You can configure the programmable I/O area in this category.

Note that this category is only valid when the selected microcontroller supports the programmable I/O area.

Remark   Change this setting to temporarily change variables while debugging.

To set a common value for the project, in the Property panel of the build tool, on the [Common Options] tab, in the [Device] category, set the value in [Programmable I/O area start address] property.

Figure 2-93.   [Programmable I/O] Category

<1> [Use Programmable I/O area]

Specify whether to use the programmable I/O area.

Select [Yes] to use the programmable I/O area ([No] is selected by default).

<2> [Programmable I/O area start address]

This property appears only when the [Use Programmable I/O area] property is set to [Yes].

Specify the start address of the programmable I/O area. Specify the address expression within the

programmable I/O area ([0] is specified by default).

The address is aligned to 16 Kbytes.

(2) [Debug Tool Settings] tab

In the [Debug Tool Settings] tab, general configurations on the debug tool can be done.

(a) [Memory]

(b) [Access Memory While Running]

(c) [Break]

(d) [Trace]

(e) [Timer]

(f) [Coverage]

(g) [Simulator GUI]

(a) [Memory]

You can configure the memory in this category.

Figure 2-94.   [Memory] Category

<1> [Memory mappings]

Current memory mapping status is displayed for each type of memory area.

The memory mapping status cannot be changed on this panel. If it is necessary to add a memory

mapping, click on the [Memory Mapping] property, and click on the [...] button that appears on the right

end of the setting field. The Memory Mapping dialog box opens; perform the setting from there.

See the section for the Memory Mapping dialog box for details on how to configure the parameters.

Figure 2-95.   Opening the Memory Mapping Dialog Box

Caution   **If you are not connected to a debug tool, then only memory mapping areas added by**

user is displayed.

Connecting to a debug tool (see "2.4.1   Connect the debug tool to CubeSuite") will

display details for each memory type.

(b) [Access Memory While Running]

You can configure the memory access while executing a program in this category.

The settings of this category are required when using the real-time display update function. See "(4) Display/ modify the memory contents during program execution" for details on the real-time display update function.

Figure 2-96.   [Access Memory While Running] Category [Simulator]

Access Memory While Running	
Update display during the execution	Yes
Display update interval[ms]	500

<1> [Update display during the execution]

Specify whether to update the display in the Watch panel/Memory panel during a program execution.

Select [Yes] to update the display (default).

<2> [Display update interval[ms]]

This property is valid only when the [Update display during the execution] property is set to [Yes].

Specify the interval in 100ms unit to update the contents in the Watch panel/Memory panel display while executing a program.

Directly enter the Integer number between 100 and 65500 (rounding up the fractions less than 100ms) ([500] is selected by default).

(c) [Break]

You can configure the break function in this category.

Figure 2-97.   [Break] Category [Simulator]

Break	
Execute instruction at breakpoint when break	No

<1> [Execute instruction at breakpoint when break]

Specify the timing to stop the program execution by breakpoints either after or before the execution of the instruction at the breakpoint.

Select [Yes] to stop after the execution of the instruction ([No] is selected by default).

See "2.8.2 Stop the program at the arbitrary position (breakpoint)" for details on breakpoints.

(d) [Trace]

You can configure the trace function in this category.

See "2.11 Collect Execution History of Programs [IECUBE][IECUBE2][Simulator]" for details on the trace function and this category configuration.

(e) [Timer]

You can configure the timer function in this category.

See "2.12 Measure Execution Time of Programs" for details on the timer function.

Figure 2-98.   [Timer] Category [Simulator]

Timer	
Use timer function	No

<1> **[Use timer function]**

Specify whether to use the timer function from the drop-down list.

Select [Yes] to use the timer function ([No] is selected by default).

(f) [Coverage]

You can configure the coverage function in this category.

See "2.13   Measure Coverage [IECUBE][IECUBE2][Simulator]" for details on the coverage function and this category configuration.

(g) [Simulator GUI]

You can configure the Simulator GUI function in this category.

See "2.17   Use the Simulator GUI [Simulator]" for details on the Simulator GUI function and this category configuration.

(3) [Download File Settings] tab

In the [Download File Settings] tab, configuration on downloading file to the debug tool can be done.

See "2.5.1   Execute downloading" for details on each category configuration.

(4) [Flash Options Settings] tab [V850E2M]

In the [Flash Options Settings] tab [V850E2M], configuration on options for the flash memory incorporated in the V850E2M microcontroller can be done. To configure options, specify the corresponding items on the Flash Options Setting dialog box [V850E2M], that is opened by clicking the [...] button appears at the right of the field by selecting the [Flash options] property in the [Flash Options] category on this tab (the [...] button appears only while connecting to the debug tool).

Click the [Write] button on this dialog box after specifying each item.

See Flash Options Setting dialog box [V850E2M] for details on the configuration.

Figure 2-99.   Opening the Flash Options Setting Dialog Box

Figure 2-100.   Flash Options Settings (Flash Options Setting Dialog Box [Simulator]])
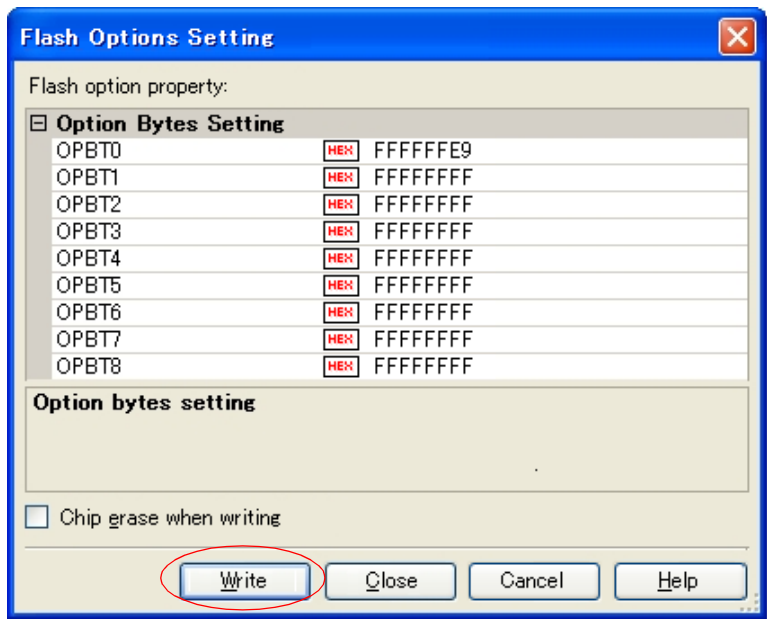
(5)   [Hook Transaction Settings] tab

In the [Hook Transaction Settings] tab, configuration on the hook transaction can be done.

See "2.16   Use Hook Function" for details on each category configuration and the function of the hook transaction.

2.4   Connect to/Disconnect from the Debug Tool

This section describes how to connect to/disconnect from the debug tool.

2.4.1   Connect the debug tool to CubeSuite

Select the [Debug] menu >> [Connect to Debug Tool] to connect to the debug tool selected in the currently active project.

After succeeding in the connection to the debug tool, the Statusbar of the Main window changes as follows:

For details on each item displayed on the Statusbar, see the section of the "Main window".

Figure 2-101.   The Statusbar during Connecting to Debug Tool (Example for Using Simulator)

The information of the debug tool appears at this area.
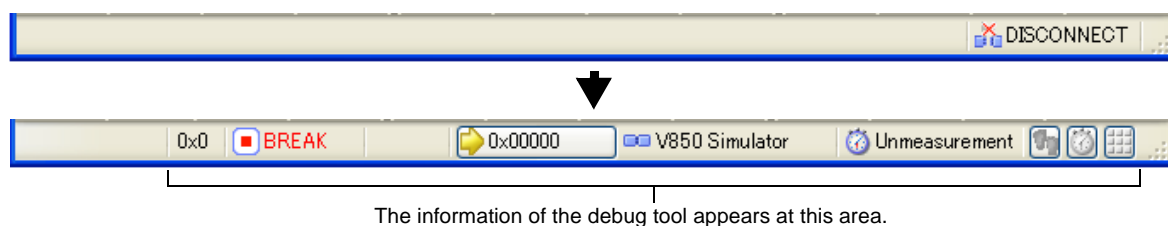
Remarks 1.   When the ⬚ button on the Debug toolbar is clicked, the specified file is downloaded automatically after connecting to the debug tool (see "2.5.1   Execute downloading").

When the ⬚ button on this toolbar is clicked, the project is built automatically, and then the built file is downloaded after connecting to the debug tool.

2.   [Simulator]

When a microcontroller whose Simulator supports peripheral function simulations is selected, the Simulator GUI window is automatically opened after connecting to the debug tool (default).

2.4.2   Disconnect the debug tool from CubeSuite

Click the ⬚ button on the Debug toolbar to disconnect from the currently connected debug tool.

After disconnecting from the debug tool, the Statusbar of the Main window changes as follows:

Figure 2-102.   The Statusbar during Disconnecting From Debug Tool

The information of the debug tool disappears at this area.   "DISCONNECT" is displayed.
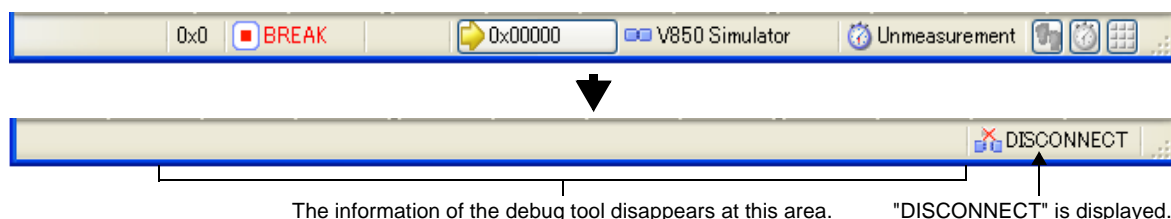
Remark   When the debug tool is disconnected, all panels and dialog boxes that are displayed only when the debug tool is connected are closed.

2.5   Download/Upload Programs

This section describes how to download programs (such as load module files) to debug to CubeSuite and how to upload the debugging memory contents from CubeSuite to files.

2.5.1   Execute downloading

Download the lode module file to debug to the debug tool that is currently connected.

Follow the steps below on the [Download File Settings] tab in the Property panel for the downloading, and then execute the downloading.

(1) [Download] category setting

Figure 2-103.   [Download] Category [IECUBE][IECUBE2][MINICUBE][E1 (JTAG)][Simulator]

Figure 2-104.   [Download] Category [MINICUBE2][E1 (Serial)]

(a) [Download files]

The names of files to be downloaded and the download conditions are displayed (the number enclosed with "[ ]" indicates the number of files to be download).

A file to be downloaded is automatically determined to a file that have been specified as the build target file in main project or subproject[Note].

However, you can manually change the download files and the condition. In this case, see "2.5.2   Advanced downloading".

Note   To download a load module file created by an external build tool (i.e. a compiler or assembler other than the build tool provided by CubeSuite), you must create a debug-dedicated project.

In the case of debugging for a debug-dedicated project, you must add a file to be downloaded to Download files node in the project tree in order to be reflected to this property.

See "CubeSuite Start" for details on the using an external build tool and a debug-dedicated project.

(b) [CPU Reset after download]

Specify whether to reset the CPU after downloading.

Select [Yes] to reset the CPU (default).

(c) [Erase flash ROM before download]

Specify whether to erase the flash ROM before downloading.

Select [Yes] to erase the flash ROM ([No] is selected by default).

(d) [Automatic change method of event setting position]

If the file is downloaded again during debugging then the location (address) set for the currently configured event may change to midway in the instruction.

Specify with this property how to handle the target event in this circumstance.

Select one of the options from the following drop-down list.

Move to the head of instruction	Sets the event to the top address of the instruction.
Suspend event	Disables the event (suspended state) (default).

Note, however, that this property setting only applies to the location setting of events without debugging information. The location setting of events with debug information is always moved to the beginning of the source text line.

(e) [Check reserved area overwriting when downloading] [MINICUBE2][E1 (Serial)]

Specify whether to output a message when overwriting to an area reserved for use by MINICUBE2/E1 (Serial) is attempted at the time of downloading.

Select [Yes] to output a message (default).

(2) [Debug Information] category setting

Figure 2-105.   [Debug information] Category

(a) [Execute to the specified symbol after CPU Reset]

Specify from the drop-down list whether to execute the program to the specified symbol position after CPU reset or downloading (for only when the [CPU Reset after download] property is set to [Yes]).

Select [Yes] to execute the program to the specified symbol position after CPU reset (default).

Remark   When the [CPU Reset after download] property is set to [Yes], the operation after downloading is as follows:

If [Yes] is selected for this property, the Editor panel will open automatically with displaying source text of the position specified with the [Specified symbol] property after downloading.

If [No] is selected for this property, the Editor panel will open with displaying source text of the reset address (when if the source text has not been allocated to the reset address, the contents of the reset address is displayed in the Disassemble panel).

(b) [Specified symbol]

This property appears only when the [Execute to the specified symbol after CPU Reset] property is set to [Yes].

Specify the position at which the program is stop after CPU reset.

Directly enter an address expression between 0 and "*last address in address space*" ([_main] is specified by default).

Note, however, that the program will not be executed if the specified address expression cannot be converted into an address.

> **Remark** Normally, specify the following.
>
> For assembler source: Start label corresponding to main function
> For C source: Symbol assigned to the start of the main function name

(c) [Startup start symbol]

Specify the start symbol of the text area (code area) of the startup routine.

Directly enter an address expression between 0 and "*last address in address space*" ([__start] is specified by default).

This setting is not needed if the source is assembly language.

(d) [Startup end symbol]

Specify the end symbol of the text area (code area) of the startup routine.

Directly enter an address expression between 0 and "*last address in address space*" ([__startend] is specified by default).

This setting is not needed if the source is assembly language.

> **Cautions 1.**   **To automatically display the source text right after downloading the file, the startup symbol must be correctly specified.**
>
> **2.**   **By default, CPU reset automatically occurs after downloading the file, and then the program is executed to the specified symbol position. If this operation above is not needed, specify [No] with both of the [CPU Reset after download] and [Execute to the specified symbol after CPU Reset] property.**

(3) Execute download

Click the ⬇ button on the Debug toolbar.

If this operation is performed while disconnecting from the debug tool, the application automatically connects to the debug tool, and then performs the download.

> **Remark** When a program that has been modified during debugging is re-downloaded, you can easily build and download it by selecting [Build & Download] from the [Debug] menu on the Main window.

If the load module file is successfully downloaded, the Editor panel opens automatically, and the contents of the downloaded file's source text are displayed.

> **Remarks 1.**   **[IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]**
>
> You can specify whether to verify memory writing when downloading the file by configuring the [Verify on writing to memory] property in the [Memory] category in the [Debug Tool Settings] tab of the Property panel.
>
> **2.**   You can automatically overwrite the value of I/O register/CPU register with the specified values before and after performing the download (see "2.16   Use Hook Function" for details).

2.5.2   Advanced downloading

You can change the download files and the condition to download.

With CubeSuite, the following file types can be downloaded.

Table 2-3.   Type of Files That Can be Downloaded

File Format	Extension	Note
Load module format	lmf **[CX]**	
	out **[CA850]**	
	extension **[GHS]**Note	Only for a debug-dedicated project
Intel Hex format (Standard)	hex	Except for the format with ID tag for the data flash memory
Intel Hex format (Extension)	hex	Can be up to 1 Mbyte
Intel Hex format (Extension linear)	hex	
Motorola S type Hex format (S0, S3, S7 - 32 bit-address)	hex	
Extended Tektronix Hex format	hex	
Binary data format	bin	

Note   Notes on using GHS compiler (Green Hills® Software, Inc., USA)
- Supported version
- MULTI (Ver. 5.*x* or later)
- Supported options
- Debug option: -G -dual_debug (recommended)
- Optimization option   -Ospace, -O, -Ospeed, -Onone
- Other than above: -prepare_dispose, -call
- C++ language and GNU C expanded specifications are not supported.
- Notes on debugging
- Debugging of assembler sources is not possible.
- No breakpoints can be set to functions defined in "#include".
- The value of a watch-expression of variable may be displayed incorrectly, because debug information does not include valid period information of variables. This item applies to cases where a variable is optimized and deleted, or a value is temporarily assigned to a register.
- Floating point rounding accuracy differs between the compiler and debugger.
- Only a current function is displayed in the Call Stack panel because the call stack information cannot be acquired.
- Step execution in "#include" statement and "setjmp"/"longjmp" function is not possible.
- A return out execution from a variable-length argument function may fail.
- **[IECUBE]**
When the flash library is ROMized, the flash self programing emulation function cannot be used.

You can change the download files or the download condition in the following Download Files dialog box.

The Download Files dialog box is opened when clicking the [...] button appears at right by selecting the [Download files] property in the [Download] category in the [Download File Settings] tab of the Property panel.

Figure 2-106. Opening the Download Files Dialog Box

Figure 2-107. Advanced Downloading (Download Files Dialog Box)

[Download file list] area   [Download file property] area
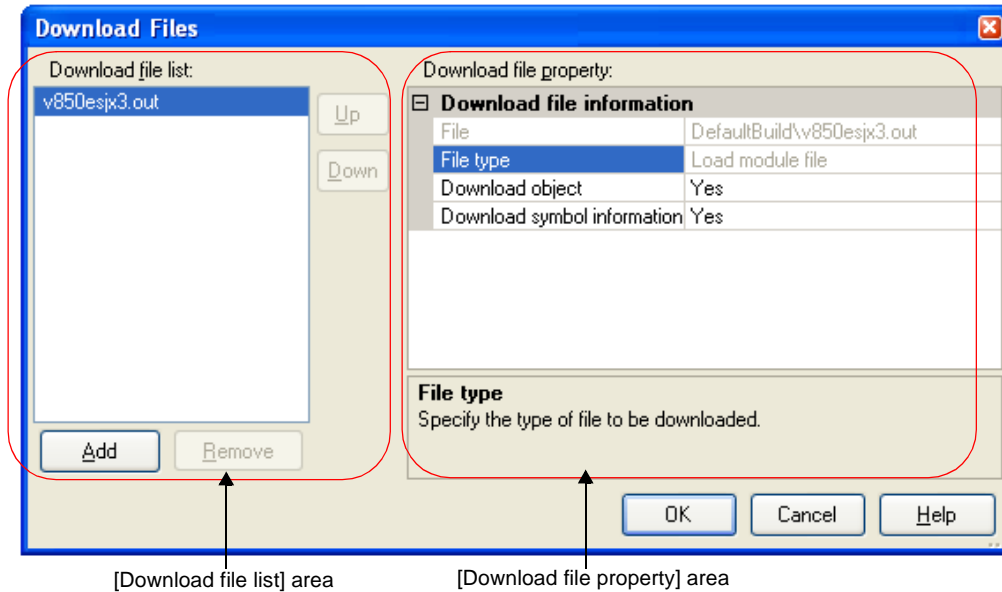
This section describes how to configure on the Download Files dialog box above when the following cases.

    (1)   Change download condition of a load module file
    (2)   Add download files (*.hex/*.bin)
    (3)   Download multiple load module files
    (4)   Perform source level debugging with a hex or binary data format file
    (5)   Download to external flash memory [IECUBE2][MINICUBE][E1 (JTAG)]

(1) Change download condition of a load module file

    Follow the steps below in the Download Files dialog box to change download condition (object information and symbol information) of load module files to download.

    (a) Select a load module file

        Select a load module file to download in the [Download file list] area.

    (b) Change download condition

        Current download condition of the selected load module file is displayed in the [Download file property] area.
        Change each items displayed in the property.

Download object	Specify whether to download the object information from the specified file.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Downloads object information.
		No	Does not download object information.
Download symbol information	Specify whether to download the symbol information from the specified file[Note].		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Downloads symbol information.
		No	Does not download symbol information.

Note   If the symbol information have not been downloaded, the source level debugging cannot be performed.

(c) Click the [OK] button
Enable all the configuration in this dialog box and change download condition.

(2) Add download files (*.hex/*.bin)
Follow the steps below to add download files (in hex format (*.hex)/in binary data format (*.bin)) in the Download Files dialog box.

(a) Click the [Add] button
When the [Add] button is clicked, a blank list item "-" is displayed in the last line of the [Download file list] area.

(b) Property configuration of the download files to add
Configure the download condition of the download file to add in the [Download file property] area.
Configure each item displayed with the following condition.
When the configuration is completed, the file name specified in this property is displayed in the blank list of the [Download file list] area.

File	Specify the file (*.hex/*.bin) to download (up to 259 characters).	
	Default	Blank
	Modifying	Directly enter from the keyboard, or specify with the Select Download File dialog box opened by clicking the [...] button.
	Available values	See "Table 2-3. Type of Files That Can be Downloaded".
File type	Specify the type of the file to download. Select [Hex file] or [Binary data file].	
	Default	Load module file
	Modifying	Select from the drop-down list.
	Available values	Either one of the following - Load module file - Hex file - Binary data file

Offset	This item appears only when the file to download is in hex format (*.hex). Specify the offset from the address at which the file's download is to start.	
	Default	0
	Modifying	Directly enter from the keyboard.
	Available values	0x0 to 0xFFFFFFFFFF in hexadecimal number
Start address	This item appears only when the file to download is in binary format (*.bin). Specify the address at which to start the file's download.	
	Default	0
	Modifying	Directly enter from the keyboard.
	Available values	0x0 to 0xFFFFFFFFFF in hexadecimal number

Remark   The settings of whether to download the object information or symbol information can be made only when the type of the file to download is load module files.

(c) Check the order of download

The order of the download is the display order of the files displayed in the [Download file list] area.
If you want to change the order, use the [Up]/[Down] button.

(d) Click the [OK] button

Enable all the configuration in this dialog box and add a download file (the file name is displayed in the [Download] category in the [Download File Settings] tab of the Property panel).

(3) Download multiple load module files

Follow the steps below on the Download Files dialog box to download multiple load module files.

Caution   **When debugging a program consisting of multiple load module files, care should be taken to avoid overlapping of location addresses.**

(a) Click the [Add] button

When the [Add] button is clicked, a blank list item "-" is displayed in the last line of the [Download file list] area.

(b) Property configuration of the download files to add

Configure the download condition of the download file to add in the [Download file property] area.
Configure each item displayed with the following condition.
When the configuration is completed, the file name specified in this property is displayed in the blank list of the [Download file list] area.

File	Specify the name of the file in the load module format to be added (up to 259 characters).	
	Default	Blank
	Modifying	Directly enter from the keyboard, or specify with the Select Download File dialog box opened by clicking the [...] button.
	Available values	See "Table 2-3. Type of Files That Can be Downloaded"

File type	Specify the type of the file to download. Here, select [Load module file].		
	Default	Load module file	
	Modifying	Select from the drop-down list.	
	Available values	Load module file	
		Hex file	
		Binary data file	
Download object	Specify whether to download the object information from the specified file.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Downloads object information.
		No	Does not download object information.
Download symbol information	Specify whether to download the symbol information from the specified file[Note].		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Downloads symbol information.
		No	Does not download symbol information.

Note If the symbol information have not been downloaded, the source level debugging cannot be performed.

Remark You can decrease the memory usage by selecting [No] for the [Download symbol information] item if the symbol information is not required for the module (in this case, however, the source level debugging can not be performed for the file).

(c) Check the order of download
The order of the download is the display order of the files displayed in the [Download file list] area.
If you want to change the order, use the [Up]/[Down] button.

(d) Click the [OK] button
Enable all the configuration in this dialog and add the specified load module file (the specified file name is displayed in the [Download] category in the [Download File Settings] tab of the Property panel).

(4) Perform source level debugging with a hex or binary data format file
You can also perform source level debugging when a hex format (*.hex) or a binary data format (*.bin) file is specified for download, by also downloading the symbol information for the load module file to be used for creation. To do so, follow the steps below on the Download Files dialog box.

(a) Click the [Add] button
When the [Add] button is clicked, a blank list item "-" is displayed in the last line of the [Download file list] area.

(b) Property configuration of the load module file to add
Configure each item displayed with the following condition in the [Download file property] area.

File	Specify the load module file that was the source for creating the hex format (*.hex) or binary data format (*.bin) file to be downloaded.
	Directly enter from the keyboard, or specify with the Select Download File dialog box opened by clicking the [...] button that appears at right by selecting this property.
File type	Select [Load module file] (default).
Download object	Specify [No].
Download symbol information	Select [Yes] (default).

(c) Click the [OK] button

Enable all the configuration in this dialog box and add the specified load module file (Only the symbol information included in the load module file will be downloaded).

(5) Download to external flash memory [IECUBE2][MINICUBE][E1 (JTAG)]

CubeSuite can download the file specified with the Download Files dialog box to a flash memory connected to the external bus of the target microcontroller.
To do so, follow the steps below.

(a) Configuration of the Property panel

Configure the following settings with the [External Flash Memory Download] category on the [Debug Tool Settings] tab of the Property panel.

Figure 2-108.   [External Flash Memory Download] Category

<1>   [External flash memory download]

Specify from the drop-down list whether to download load module files or hex files to an external flash memory.
Select [Yes] ([No] is selected by default).

<2>   [Information file]

Specify the external flash memory information file (FDB file).
Directly enter the file name (the reference point of the path is the project folder) or select the file with the dialog box opened by clicking on the [...] button that appears on the right end of the setting field.
The external flash memory information file can be downloaded from our website (Development Tools Download webpage).

<3>   [Start address]

Specify the start address of the external flash memory.
Directly enter the address from 0x0 to 0xFFFFFFFF ([0] is selected by default).

<4> [Data width]

Specify the data width (access size) of the external flash memory from the drop-down list (see "Table 2-4. Parameter Set Values (For External Flash Memory)").

The drop-down list displays the following data widths (unit: Kbytes).

    8 (default), 16, 32

<5> [Number of parallel]

Specify the number of external flash memories in parallel (see "Table 2-4. Parameter Set Values (For External Flash Memory)").

You can specify the number from the drop-down list or by directly entering an integer number between 1and 99.

The drop-down list displays the following numbers.

    1 (default), 2, 4
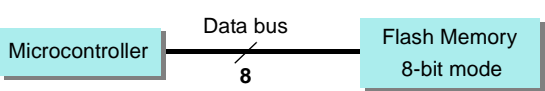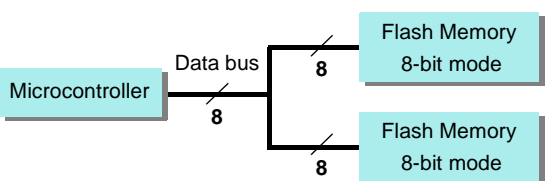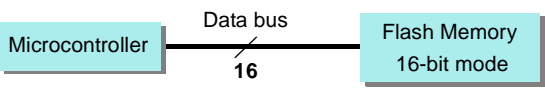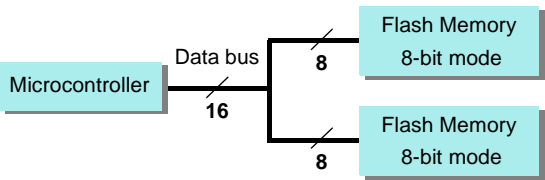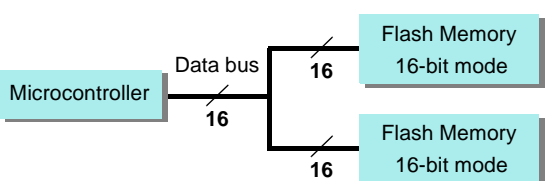
<6> [Number of serial]

Specify the number of external flash memories in serial (see "Table 2-4. Parameter Set Values (For External Flash Memory)").

You can specify the number from the drop-down list or by directly entering an integer number between 1and 99.

The drop-down list displays the following numbers.

    1 (default), 2

Table 2-4. Parameter Set Values (For External Flash Memory)

Connection of Flash Memory	Data Width	Number of Parallel	Number of Serial
Microcontroller — Data bus 8 — Flash Memory 8-bit mode	8	1	1
Microcontroller — Data bus 8 — (8) Flash Memory 8-bit mode / (8) Flash Memory 8-bit mode	8	1	2
Microcontroller — Data bus 16 — Flash Memory 16-bit mode	16	1	1
Microcontroller — Data bus 16 — (8) Flash Memory 8-bit mode / (8) Flash Memory 8-bit mode	16	2	1
Microcontroller — Data bus 16 — (16) Flash Memory 16-bit mode / (16) Flash Memory 16-bit mode	16	1	2

Connection of Flash Memory		Data Width	Number of Parallel	Number of Serial
		32	4	1
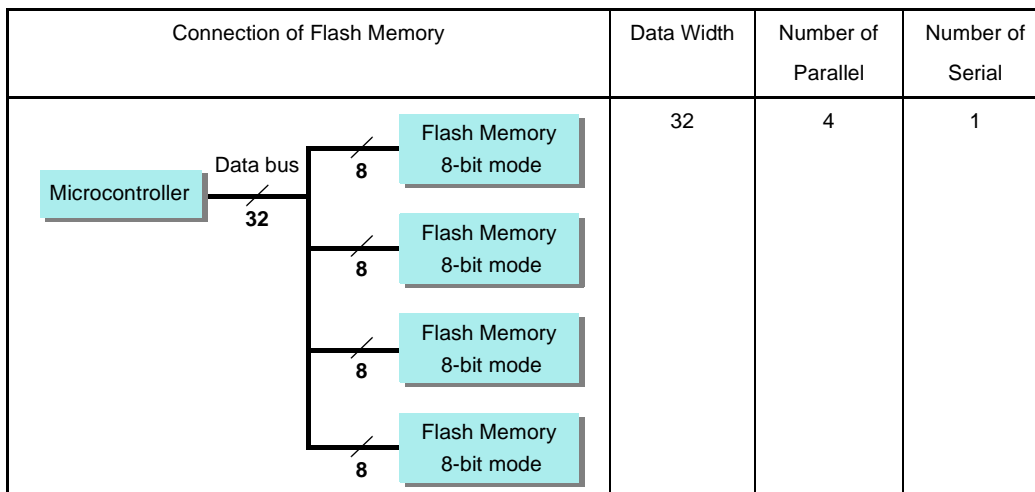
(b) Setting of external bus

Change the peripheral I/O register setting so as to use external buses, which enables accessing from the microcontroller to flash memory.

This setting can be changed via the IOR panel, but the specification of the batch processing to proceed right before downloading by using the hook function is recommended, because the setting must be changed for individual registers (see "2.16   Use Hook Function").

(c) Execute download

Execute download (see "(3)   Execute download").

2.5.3   Execute uploading

The contents of the memory of the debug tool currently connected can be saved (uploaded) in an arbitrary file.

You can upload the data in the Data Save dialog box opened by selecting the [Debug] menu >> [Upload...].

In this dialog box, follow the steps below.

Figure 2-109.   Execute Uploading (Data Save Dialog Box)
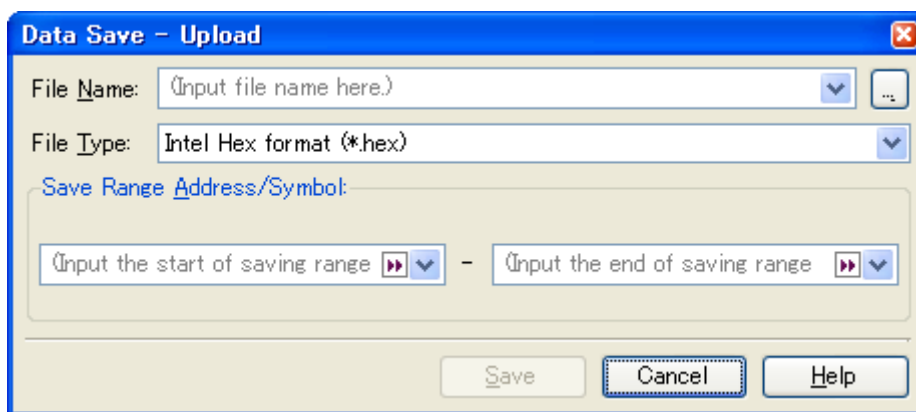
(1) Specify [File Name]

Specify the name of the file to save.

You can either type a filename directly into the text box (up to 259 characters), or select one from the input history via the drop-down list (up to 10 items). You can also specify the file by clicking the [...] button, and selecting a file via the Select Data Save File dialog box.

(2) Specify [File Type]

Select the format in which to save the file from the following drop-down list.

The following file formats can be selected.

Note, however, that the format with ID tag below appears only when the selected microcontroller incorporates the data flash memory and supports the ID tag function. Select these items to save the ID tag information.

Table 2-5.   Type of Files That Can be Uploaded

List Item	Format
Intel Hex format (*.hex)	Hex format[Note]
Motorola Hex format (S0, S3, S7- 32bit-address) (*.hex)	Hex format
Extended Tektronix Hex format (*.hex)	Hex format
Intel Hex format with ID Tag (*.hex) **[IECUBE][IECUBE2][INICUBE][MINICUBE2][E1]**	Hex format
Motorola Hex format with ID Tag (S0, S3, S7- 32bit-address) (*.hex) **[IECUBE][IECUBE2][INICUBE][MINICUBE2][E1]**	Hex format
Extended Tektronix Hex with ID Tag format (*.hex) **[IECUBE][IECUBE2][INICUBE][MINICUBE2][E1]**	Hex format
Binary data (*.bin)	Binary data format

Note   If the space specified via the "Specify [Save Range Address/Symbol]" is greater than 1 MB, then the Intel hex-format file will be saved as "Extended Linear (32-bit address)"; if it is within 1 MB, then it will be saved as "Extended (20-bit address)".

(3) Specify [Save Range Address/Symbol]

Specify the range of addresses to save via "start address" and "end addresses".

Directly enter hexadecimal number/address expression in each text box or select from the input history displayed in the drop-down list (up to 10 items).

(4) Click the [Save] button

Save the contents of the memory in the specified file in specified format as upload data.

2.6   Display/Change Programs

This section describes how to display and change programs when a load module file with the debug information is downloaded to a debug tool.

Downloaded programs can be displayed in the following panels.

- Editor panel

  The source file is displayed and can be edited.

  Furthermore, the source level debugging (see "2.7.3   Execute programs in steps") and the display of the code coverage measurement result (see "2.13.2   Display the coverage measurement result") can be performed in this panel.

- Disassemble panel

  The result of disassembling the downloaded program (the memory contents) is displayed and can be edited (line assemble).

  Furthermore, the instruction level debugging (see "2.7.3   Execute programs in steps") and the display of the code coverage measurement result (see "2.13.2   Display the coverage measurement result") can be performed in this panel. In this panel, the disassemble results can be displayed with the corresponding source text (default).

Remark   It is normally necessary to download a load module file with debugging information in order to perform the source level debugging, but it is also possible to do so by downloading a hex format (*.hex) or binary data format (*.bin) file (see "(4)   Perform source level debugging with a hex or binary data format file").

2.6.1   Display source files

The source file is displayed in the Editor panel below. The Editor panel automatically opens with displaying source text of the specified position (see "2.5.1   Execute downloading") when a load module file is successfully downloaded.

If you want to open the Editor panel manually, double-click on the source file in the Project Tree panel.

For details on the contents and function in each area, see the section for the Editor panel.

Figure 2-110.   Display Source File (Editor Panel)
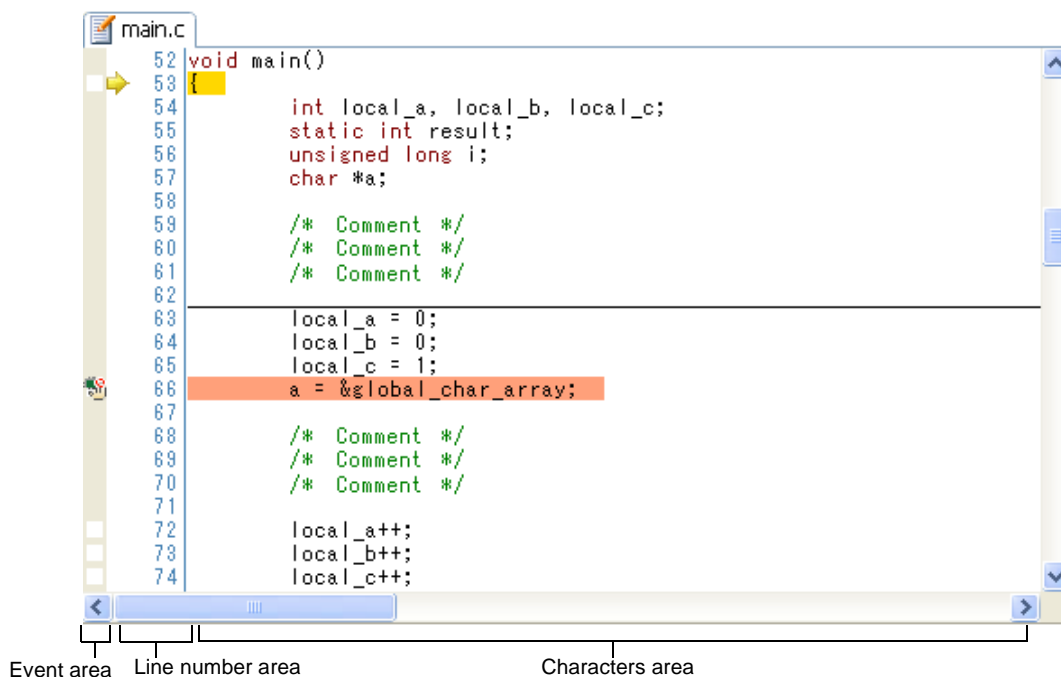
This section describes the following.

(1) Display variables

By superimposing the mouse cursor on variables on the source text, variable values are pop-up displayed.
The display format of the variable value is same as "Table A-12.   Display Format of Watch-Expressions (Default)"
depending on the type of the variable.

Figure 2-111.   Pop-up Display of Variables (Editor Panel)
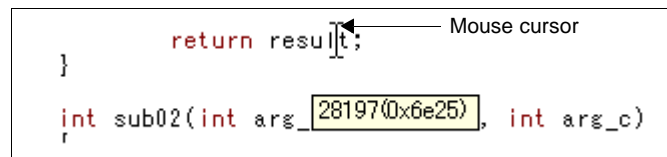
(2) Search characters

Character searching in the source text is taken place in the Search and Replace dialog box opens with selecting
the ▣ button on the toolbar. If the search is taken place with the range selection in the Editor panel, search is
taken place in the selected range.
In this dialog box, follow the steps below.

Figure 2-112.   Character Search in Source Text (Search and Replace Dialog Box)
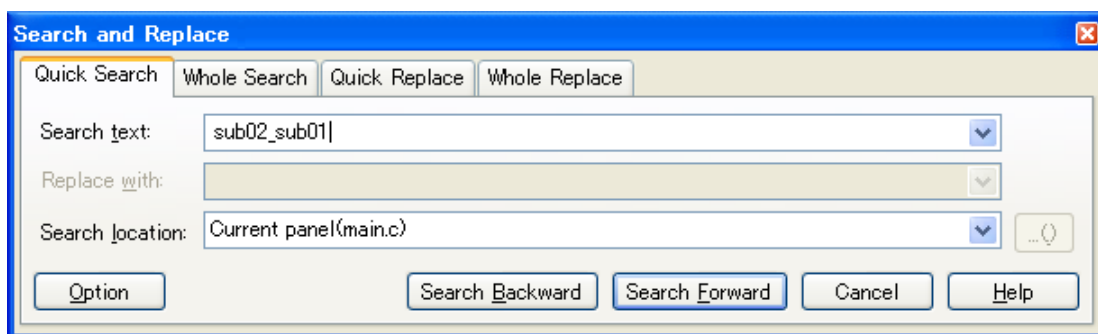
(a) Specify [Search text]

Enter characters to search.
A word (variable/function) at the caret position in the Editor panel is specified by default.
If you want to change it, directly enter the characters into the text box (up to 1024 characters) or select from
the input history in the drop-down list (up to 10 items).

(b) Specify [Search location]

Select [Current panel (*file name*)] from the drop-down list.
If you want to search in the selected range in the Editor panel, select [Selection area].

(c) Click the [Search Backward]/[Search Forward] button

When the [Search Backward] button is clicked, search is taken place in the order from the large address number to small and the search results are displayed selected in the Editor panel.

When the [Search Forward] button is clicked, search is taken place in the order from the small address number to small and the search results are displayed selected in the Editor panel.

Remarks 1. Click the [Option] button to specify to use wild card, case sensitivity, word by word search, and so on.

2. In the Search and Replace dialog box, various search/replace operation can be done with selecting either [Whole Search]/[Quick Replace]/[Whole Replace] tab.

(3) Move to the specified line

You can move to the specified line in the source text in the Go to the Location dialog box opens when selecting [Move to...] from the context menu.

In this dialog box, follow the steps below.

Figure 2-113.   Move to the Specified Line in Source Text (Go to the Location Dialog Box)
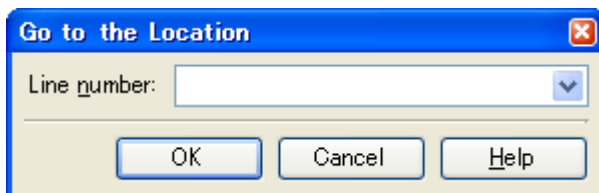
(a) Specify [Line number]

Specify the number of the line you want to move the caret to.

You can either type a location directly into the text box (only in decimal number), or select one from the input history via the drop-down list (up to 10 items).

(b) Click the [OK] button

Caret is moved to the specified line.

(4) Jump to functions

You can jump to the function (variable **[CX]**) determined from the selected characters or the word at the caret position (jumps to the first executable line in the target function).

Select [Jump to Function] from the context menu after moving the caret to the function on the source text.

Figure 2-114.   Jump to Functions
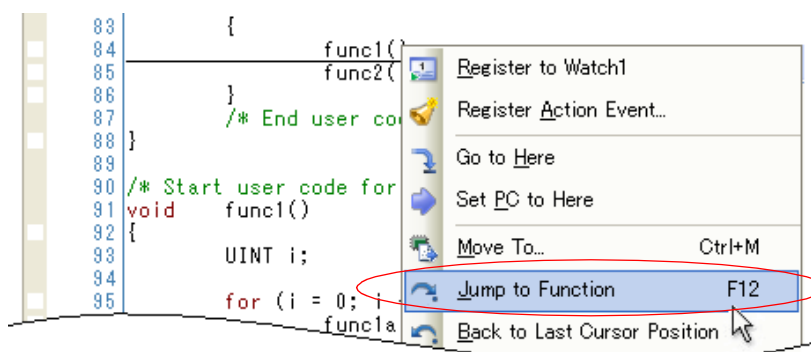
Note that this function is available only when the following conditions are satisfied depending on the build tool used:

(a) [CA850]
- The target function[Note 1] has been defined in an active project.
- The type of the project specified as the active project is "Application".
- A file[Note 2] with the symbol information has been specified with [Download files]. For while disconnecting from the debug tool, the file has been specified at the top in [Download files].

(b) [CX]
- The target is a function, variable, or label in C language.
- The focus is in the Editor panel.

(c) An External build tool
- The target function[Note 1] has been defined in an active project.
- The focus is in the Editor panel.
- A file[Note 2] with the symbol information has been specified with [Download files]. For while disconnecting from the debug tool, the file has been specified at the top in [Download files].

Notes 1. A jump to a static function cannot be made while disconnecting from the debug tool.
2. When the file is the hex format, the specification of downloading the symbol information is required (see "(4)   Perform source level debugging with a hex or binary data format file").

Caution   **When multiple statements are described in a line, a jump to an illegal location may be made.**

Remark   The word determination depends on the current build tool.

(5) Jump to a desired line (tag jump)
If the information of a file name, a line number and a column number exists in the line at the caret position, you can open the file in the another Editor panel and jump to the corresponding line and the corresponding column (if the Editor panel is already opened, jump to the panel).
Select [Tag Jump] from the context menu after moving the caret to the line on the source text.
The tag jump is operated as follows:

Table 2-6.   Operation of Tag Jump

Example of Character String	Operation
C:\work\src.c	Jumps to the top line of the file "C:\work\src.c".
Tmp\src.c	Jumps to the top line of the file ".Tmp\src.c". (The reference point of the path is the project folder.)
C:\work\src.c(10)	Jumps to the tenth line from the top of the file "C:\work\src.c".
C:\"work sub\src.c"(10)	Jumps to the tenth line from the top of the file "C:\"work sub\src.c"". (Path specification (path/file name) including space characters must be enclosed in "".)
C:\work\src.c(10,5)	Jumps to the fifth column of the tenth line from the top of the file "C:\work\src.c".

Figure 2-115.   Tag Jump
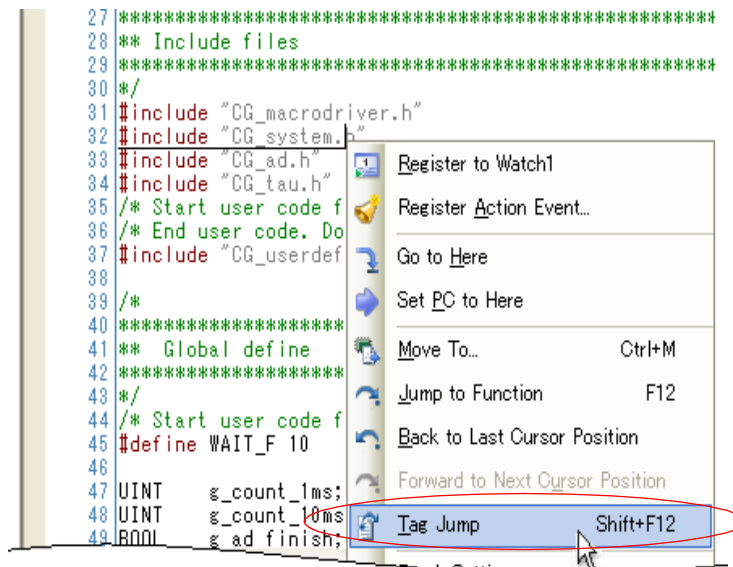
Remarks 1.   Jumps are case-insensitive.

2.   The reference point of the path is the project folder in which the file is registered. If the file is not
registered in any project, the reference point of the path will be the active folder.

3.   Path specifications (path/file names) including space characters must be enclosed in "".

2.6.2   Display the result of disassembling

The result of disassembling the downloaded program (disassembled text) is displayed in the Disassemble panel below.

Select [View] menu >> [Disassemble] >> [Disassemble*1 - 4*].

The maximum of 4 Disassemble panels can be opened. Each panel is identified by the names "Disassemble1",
"Disassemble2", "Disassemble3" and "Disassemble4" on the title bar.

For details on the contents and function in each area, see the section for the Disassemble panel.

Figure 2-116.   Display the Result of Disassembling (Disassemble Panel)
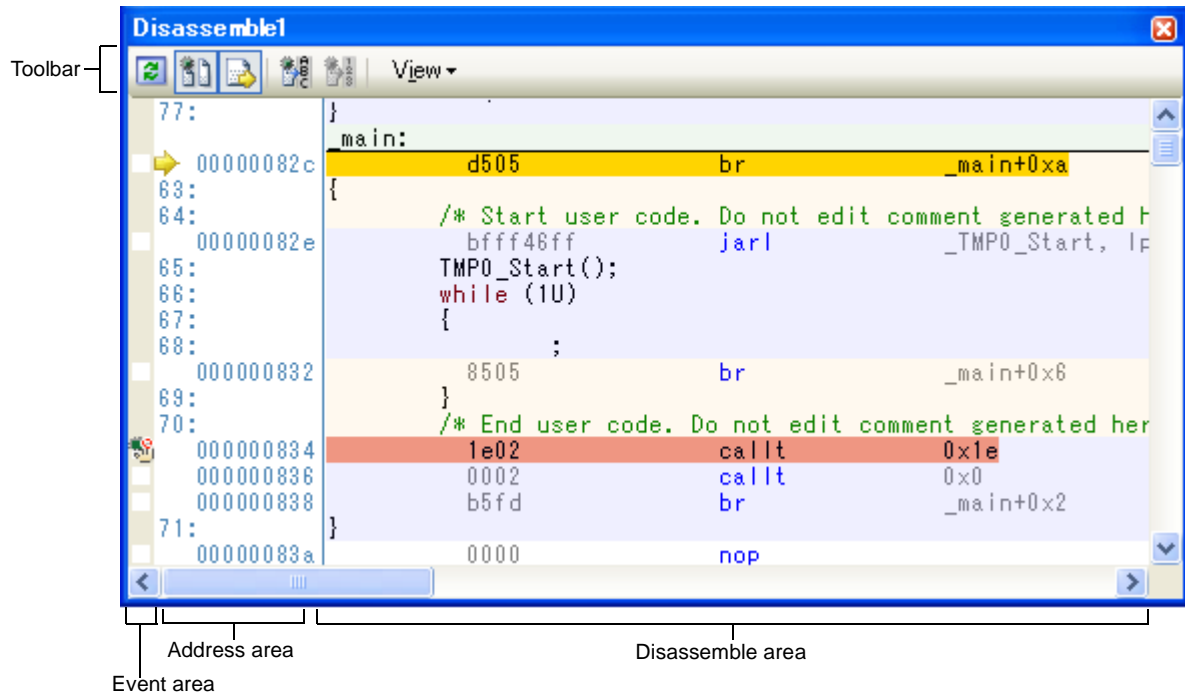
This section describes the following.

- (1) Change display mode
- (2) Change display format
- (3) Move to the specified address
- (4) Move to the symbol defined location
- (5) Save the disassembled text contents

(1) Change display mode

The result of disassembling is displayed in the mixed display mode (both the disassembled text and the source text) by default.

When you click the [button] button (toggle function) on the toolbar, the source text is displayed or hidden.

Figure 2-117.   Display Example of Source Text Hidden
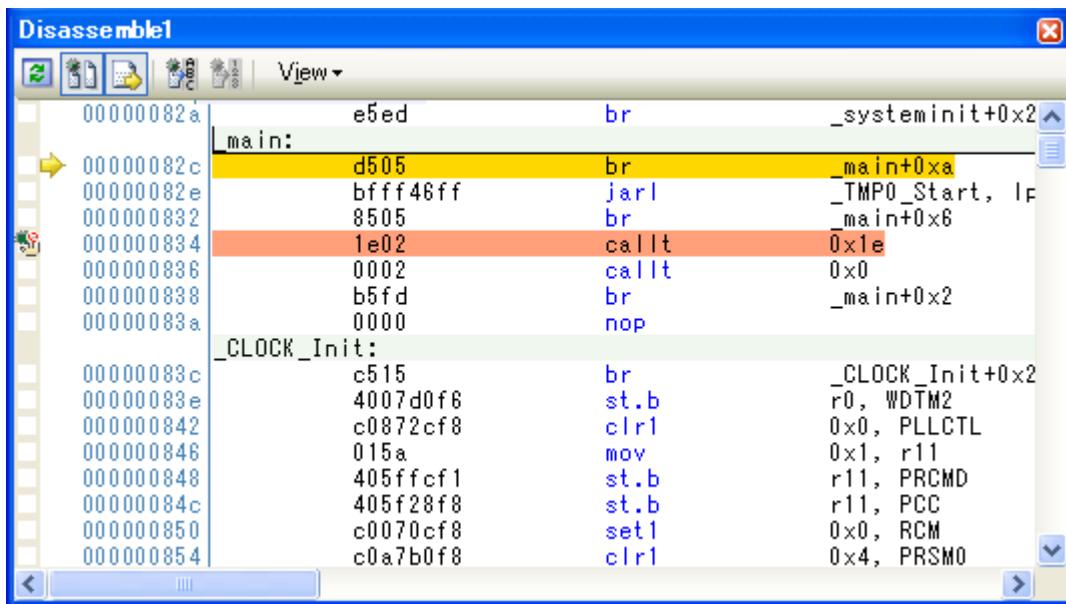
(2) Change display format

The display format of the disassemble area can be changed using buttons below on the toolbar.

View	The following buttons to change the display format are displayed.
	Displays the offset value of the label. The offset value from the nearest label is displayed when a label is defined for the address.
	Displays the address value as the result of disassembling in the format "symbol + offset value" (default). Note that when a symbol has been defined as the address value, only the symbol is displayed.
	Displays the name of the register by its function name (default).
	Displays the name of the register by its absolute name.

(3) Move to the specified address

You can move to the specified address in the disassembled text in the Go to the Location dialog box opens when selecting [Move to...] from the context menu.

In this dialog box, follow the steps below.

Figure 2-118.   Move to the Specified Address in Disassembled Text (Go to the Location Dialog Box)
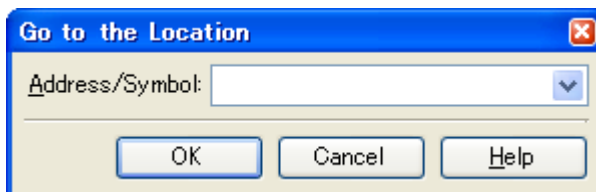
(a) Specify [Address/Symbol]

Specify the address you want to move the caret to.

You can either type address expressions directly into the text boxes, or select them from the input history via the drop-down list (up to 10 items).

(b) Click the [OK] button

Caret is moved to the specified address.

(4) Move to the symbol defined location

You can move the caret to the address where the symbol is defined.

Click the [icon] button on the toolbar after moving the caret to the instruction which refers to the symbol.

Furthermore, click the [icon] button on the toolbar following the previous operation returns the caret to the instruction which refers to the symbol at previous caret is defined.

(5) Save the disassembled text contents

Contents of the disassembled text can be saved in text files (*.txt)/CSV files (*.csv).

When saving to the file, the latest information is acquired from the debug tool, and it is saved according with the display format on this panel.

The Data Save dialog box can be opened by selecting the [File] menu >> [Save Disassemble Data As...] (when this operation is taken place with range selection on the panel, the disassemble data only in the selected range is saved).

In this dialog box, follow the steps below.

Figure 2-119.   Save the Disassembled Text Contents (Data Save Dialog Box)

(a) Specify [File Name]

Specify the name of the file to save.

You can either type a filename directly into the text box (up to 259 characters), or select one from the input history via the drop-down list (up to 10 items).

You can also specify the file by clicking the [...] button, and selecting a file via the Select Data Save File dialog box.

(b) Specify [File Type]

Select the format in which to save the file from the following drop-down list.

The following file formats can be selected.

List Item	Format
Text files (*.txt)	Text format (default)
CSV (Comma-Separated Variables)(*.csv)	CSV format[Note]

Note The data is saved with entries separated by commas (,).

If the data contains commas, each entry is surrounded by double quotes "" in order to avoid illegal formatting.

(c) Specify [Save Range Address/Symbol]

Specify the range of addresses to save via "start address" and "end addresses".

Directly enter hexadecimal number/address expression in each text box or select from the input history displayed in the drop-down list (up to 10 items).

If a range is selected in the panel, that range is specified as the default. If there is no selection, then the range currently visible in the panel is specified.

(d) Click the [Save] button

Disassembling data is saved in the specified file with the specified format.

Figure 2-120.   Output Example of Disassembling Data
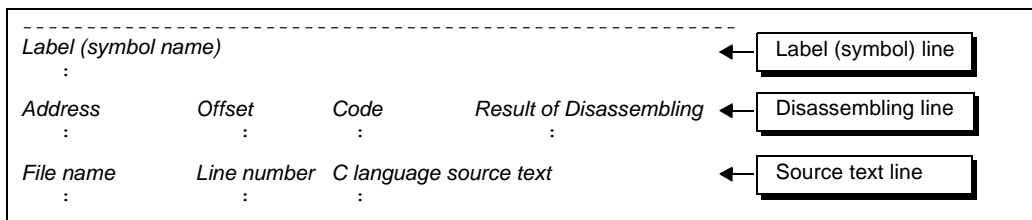
Remark When the contents of the panel are overwritten and saved by selecting [File] menu >> [Save Disassemble Data], the Disassemble panels (Disassemble1-4) are handled individually for these respectively. In addition, saving range is same as the previously specified address range.

2.6.3   Run a build in parallel with other operations

CubeSuite has a function that a build is started automatically when one of the following events occurs (rapid build function).

(1) When the target project type is other than a debug-dedicated project

- When a C source file, assembler source file, header file, link directive file, symbol information file, object module file or library file that is being added to the project has been updated
- When a build target file has been added to or removed from the project
- When the link order of object module files and library files has changed
- When any one of properties in the Property panel of the build tool or build target files has been changed

(2) When the target project type is a debug-dedicated project

- When a C source file, assembler source file, or header file that is being added to the debug-dedicated project has been saved after editing
- When a C source file, assembler source file, or header file has been added to or removed from the debug-dedicated project
- When any one of properties in the Property panel of the debug-dedicated project has been changed

If a rapid build is enabled, it is possible to perform a build in parallel with the above operations.

To enable/disable a rapid build, select [Rapid Build] from the [Build] menu. A rapid build is enabled by default.

Caution   **When an external text editor is used, check the [Observe registered files changing] check box on the [General - Build/Debug] category in the Option dialog box to enable this function.**

Remarks 1.   After editing source files, it is recommend to save frequently by pressing the [Ctrl] + [S] key.
   2.   Enabling/disabling a rapid build is set for the entire project (main project and subprojects).
   3.   If you disable a rapid build while it is running, it will be stopped at that time.

2.6.4   Perform line assembly

Instructions and code displayed in the Disassemble panel can be edited (line assembly).
This section describes the following.

   (1)   Edit instructions
   (2)   Edit code

(1)   Edit instructions

Follow the steps below to edit instructions.

   (a)   Switch to edit mode

   Double-click the instruction to edit or select [Edit Disassemble] from the context menu after moving the caret to the instruction to edit.

   (b)   Edit instructions

   Use keyboard to directly edit the instructions.

   (c)   Write to memory

   Press the [Enter] key to line assemble the edited instructions after editing. The code is automatically written to the memory.
   If the edited instruction is invalid, the instruction is shown in red and will not be written to the memory.

If there is a space because of overwriting the displayed result of disassembling by another instruction, its byte number is automatically compensated with nop instruction as follows:

Examples 1.   Overwriting the prepare instruction (8-byte instruction) with the jarl instruction (4-byte instruction)

Before editing	0432	mov	0x4, r6
	1d38	mov	r29, r7
	8f071b0effff0000	prepare r20, r21, r22, 0x1c, 0x0000ffff	
	0132	mov	0x1, r6
After editing	0432	mov	0x4, r6
	1d38	mov	r29, r7
	bfffe265	jarl	0x100, lp
	0000	nop	
	0000	nop	
	0132	mov	0x1, r6

2. Overwriting the mov instruction (2-byte instruction) with the jarl instruction (4-byte instruction)

Before editing	0432	mov	0x4, r6
	1d38	mov	r29, r7
	8f071b0effff0000	prepare	r20, r21, r22, 0x1c, 0x0000ffff
	0132	mov	0x1, r6
After editing	0432	mov	0x4, r6
	bfffe265	jarl	0x100, lp
	0000	nop	
	0000	nop	
	0000	nop	
	0132	mov	0x1, r6

Caution   Handling the prepare instruction and dispose instruction

The following table shows the instruction formats of the prepare instruction and dispose instruction. The operand "list12" comprises 12-bit value where a different register is assigned to each bit.

Instruction format of the prepare instruction	prepare	list12, imm5
	prepare	list12, imm5, sp/imm
Instruction format of the dispose instruction	dispose	imm5, list12
	dispose	imm5, list12, [reg1]

When displaying the results of disassembling the prepare instruction and dispose instruction in the Disassemble panel, the corresponding register names for the operand "list12" are displayed instead of its values as shown in the following examples.

Examples 1. When the code is "0xbf, 0x07, 0xe1, 0xff" (4-byte prepare instruction)

| View | prepare   r20, r21, r22, r23, r24, r25, r26, r27,r28, r29, r30, r31, 0x20 |
| Syntax | prepare   0xfff, 0x20 |

2. When the code is "0x90, 0x07, 0xbb, 0xaa 0xff, 0xff, 0xff, 0xff" (8-byte prepare instruction)

| View | prepare   r20, r22, r24, r26, r28, r31, 0x20, 0x7fffffff |
| Syntax | prepare   0x555, 0x20, 0x7fffffff |

3. When the code is "0x51, 0x06, 0xe0, 0xff" (4-byte dispose instruction)

| View | dispose   0x20, r20, r21, r22, r23, r24, r25, r26, r27, r28, r29, r30, r31 |
| Syntax | dispose   0x20, 0xfff |

4. When the code is "0x50, 0x06, 0xaa, 0xaa" (4-byte dispose instruction)

| View | dispose   0x20, r20, r22, r24, r26, r28, r31, [r10] |
| Syntax | dispose   0x20, 0x555, [r10] |

Note, however, that it is possible to specify both the value and the register name for the operand "list12" when line assembling the prepare instruction and dispose instruction.

Examples 1.   In both of the cases (1) and (2) below, the same set of values "0x91, 0x07, 0xe1, 0xff" will be generated as a result of line assembly.

(1)	`prepare   r20, r21, r22, r23, r24, r25, r26, r27,r28, r29, r30, r31, 0x20`
(2)	`prepare   0xfff, 0x20`

2.   In both of the cases (1) and (2) below, the same set of values "0xbe, 0x07, 0xbb, 0xaa 0xff, 0xff, 0xff, 0x7f" will be generated as a result of line assembly.

(1)	`prepare   r20, r22, r24, r26, r28, r31, 0x20, 0x7fffffff`
(2)	`prepare   0x555, 0x20, 0x7fffffff`

3.   In both of the cases (1) and (2) below, the same set of values "0x51, 0x06, 0xe0, 0xff" will be generated as a result of line assembly.

(1)	`dispose   0x20, r20, r21, r22, r23, r24, r25, r26, r27, r28, r29, r30, r31`
(2)	`dispose   0x20, 0xfff`

4.   In both of the cases (1) and (2) below, the same set of values "0x50, 0x06, 0xaa, 0xaa" will be generated as a result of line assembly.

(1)	`dispose   0x20, r20, r22, r24, r26, r28, [r10]`
(2)	`dispose   0x20, 0x555, [r10]`

(2) Edit code

Follow the steps below to edit code.

(a) Switch to edit mode

Double-click the code to edit or select [Edit Code] from the context menu after moving the caret to the code to edit.

(b) Edit code

Use keyboard to directly edit the code.

(c) Write to memory

Press the [Enter] key to write the code to the memory after editing.

If the edited instruction is invalid, the instruction is shown in red and will not be written to the memory.

When the code is written to the memory, the result of disassembling is also updated.

2.7   Execute Programs

This section describes how to execute programs.

Main operations in this section are taken place from the debug toolbar or the [Debug] menu in the Main window, where commands to control the execution of programs are included.

Caution   Items of the debug toolbar and the [Debug] menu are valid only while connecting to the debug tool.

Figure 2-121.   Debug Toolbar

Figure 2-122.   [Debug] Menu
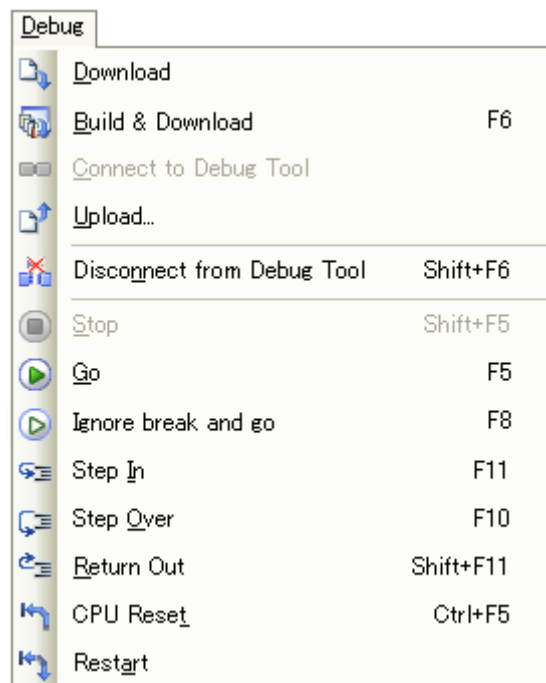
2.7.1   Reset microcontroller (CPU)

To reset CPU, click the [icon] button on the debug toolbar.

When CPU is reset, the current PC value is set to the reset address.

Remark   You can automatically overwrite the value of I/O register/CPU register with the specified values after CPU reset (see "2.16   Use Hook Function" for details).

2.7.2   Execute programs

The following types of CubeSuite execution functions are provided.

Select any of the following operations according to the purpose of debugging.

See "2.8   Stop Programs (Break)" for details on how to stop the program in execution.

(1)   Execute after resetting microcontroller (CPU)

(2)   Execute from the current address

(3)   Execute after changing PC value

Remark   You can automatically overwrite the value of I/O register/CPU register with the specified values before executing the program (see "2.16   Use Hook Function" for details).

(1) Execute after resetting microcontroller (CPU)

Click the [icon] button on the debug toolbar.

Reset CPU and start execution of the program from the reset address.

When this operation is performed, the program continues executing until either of the following occurs:

- The [icon] button has been clicked (see "2.8.1   Stop the program manually").
- The PC has reached a breakpoint (see "2.8.2   Stop the program at the arbitrary position (breakpoint)").
- A break event condition has been met (see "2.8.3   Stop the program with the access to variables/I/O registers").
- A fail-safe break has occurred (see "2.8.4   Stop the program when an invalid execution is detected [IECUBE]").
- The Other break causes have occurred.

Remark   This operation is the same as when the [icon] button is clicked after clicking the [icon] button.

(2) Execute from the current address

Perform any of the following operations to start executing the program from the address at the current PC value.

(a) Normal execution

Click the [icon] button on the debug toolbar.

When this operation is performed, the program continues executing until either of the following occurs:

- The [icon] button has been clicked (see "2.8.1   Stop the program manually").
- The PC has reached a breakpoint (see "2.8.2   Stop the program at the arbitrary position (breakpoint)").
- A break event condition has been met (see "2.8.3   Stop the program with the access to variables/I/O registers").
- A fail-safe break has occurred (see "2.8.4   Stop the program when an invalid execution is detected [IECUBE]").
- The Other break causes have occurred.

(b) Execution ignoring break-related events

Click the [icon] button on the debug toolbar.

When this operation is performed, the program continues executing until either of the following occurs:

- The [icon] button has been clicked (see "2.8.1   Stop the program manually").
- A fail-safe break has occurred (see "2.8.4   Stop the program when an invalid execution is detected [IECUBE]").
- The Other break causes have occurred.

Remark   The action event that is currently set does not occur while the program with this operation is in execution.

(c) Execution to the caret position

To start this operation, move the caret to the line/instruction to stop the program in the Editor panel/Disassemble panel, then select [Go to Here] from the context menu.

When this operation is performed, the program continues executing until either of the following occurs:

- The PC has reached the address of the caret position.
- The 🔴 button has been clicked (see "2.8.1   Stop the program manually").
- A fail-safe break has occurred (see "2.8.4   Stop the program when an invalid execution is detected [IECUBE]").
- The Other break causes have occurred.

Caution   **When the corresponding address of the line at the caret position does not exist, the program is executed to the corresponding address of the lower valid line (if the corresponding address does not exist, an error message will appear).**

Remark   The action event that is currently set does not occur while the program with this operation is in execution.

(3)   Execute after changing PC value

The program is executed after forcibly changing the current PC value to an arbitrary position.
To start this operation, move the caret to the line/instruction to start the program in the Editor panel/Disassemble panel, then select [Set PC to Here] from the context menu (the current PC value is set to the address of the line/instruction where the caret currently exists).
Then execute either one of the execution method described in "(2)   Execute from the current address".

2.7.3   Execute programs in steps

When either of the following operation is taken place, the program is automatically stopped after step execution in source level units (for one line source text)/in instruction level units (for one instruction).
As contents in each panel are automatically updated by the minute after the program is stopped, the step debugging is effective to debug the transition execution in source level units/instruction units.
An execution unit for step execution is determined automatically by the panel in which the current focus is as follows:

- When the focus is not in the Disassemble panel: Step execution[Note] in source level units
- When the focus is in the Disassemble panel: Step execution in instruction units

Note   If the line information does not exist in the address at the current PC value, the step execution is taken place in instruction units.

The following are types of step execution.

(1)   Step in function (Step in execution)
(2)   Step over function (Step over execution)
(3)   Execute until return is completed (Return out execution)

Cautions 1.   **Breakpoints, break events, and action events that are being set do not occur during step execution.**

2.   **[IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]**
Interrupts are not acknowledged and fail-safe breaks do not occur, during step execution.
[Simulator]
You may jump to a interrupt handler during step execution.

3.   **An error message will appear while processing a function prologue or epilogue if the return address cannot be acquired.**

(1) Step in function (Step in execution)

When the function is called, the program is stopped at the top of the called function.

Click the [icon] button on the debug toolbar to perform Step in execution.

Caution   **If Step in execution is performed for the longjmp function, execution processing may not complete and may wait for a time-out.**

(2) Step over function (Step over execution)

In the case of a function call by the jarl instruction, all the source lines/instructions in the function are treated as one step and executed until the position where execution returns from the function (step execution will continue until the same nest is formed as when the jarl instruction has been executed).

Click the [icon] button on the debug toolbar to perform Step over execution.

In the case of an instruction other than jarl, operation is the same as when the [icon] button is clicked.

Caution   **If Step over execution is performed for the longjmp function, execution processing may not complete and may wait for a time-out.**

(3) Execute until return is completed (Return out execution)

Execute the current function until the program returns to the original function.

The program returns to the original function without executing the rest of the instructions in the current function when the step execution with this instruction is taken place at the termination of the source lines/instructions execution that need to check in a function.

Click the [icon] button on the debug toolbar to perform Return out execution.

Cautions 1.   **Return out execution cannot be performed within main function (the state of the program becomes "RUN").**

2.   **Return out execution cannot be performed immediately after stepping in a function.**

3.   **Return out execution cannot be performed while processing a function prologue or epilogue.**

4.   **If Return out execution is performed in a function that called the longjmp function, breaks may not occur.**

5.   **If Return out execution is performed in a recursive function, the program will be executed in free-run mode.**

6.   **If Return out execution is performed in a memory bank function, the program may be executed in free-run mode.**

2.8　Stop Programs (Break)

This section describes how to stop the program in execution.

> **Remark**　When the program in execution is stopped, a statement of the cause of the break appears on the Statusbar in the Main window.

2.8.1　Stop the program manually

The program in execution is forcibly stopped by clicking the ■ button on the debug toolbar.

2.8.2　Stop the program at the arbitrary position (breakpoint)

The program in execution can be stopped at the arbitrary position by setting a breakpoint. A breakpoint can be set by one-clicking with the mouse.

You need to configure the type of breakpoints to use before setting a breakpoint.

This section describes the following.

- (1) Configure types/movements of breakpoints to use
- (2) Set a breakpoint
- (3) Delete a breakpoint

(1)　Configure types/movements of breakpoints to use

Breakpoints to use are set in the [Break] category in the [Debug Tool Settings] tab of the Property panel.

The setting method differs depending on the debug tool used.

- (a) [IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]
- (b) [Simulator]

(a)　[IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]

Figure 2-123.　[Break] Category [IECUBE][IECUBE2][MINICUBE][E1 (JTAG)]

Figure 2-124.　[Break] Category [MINICUBE2][E1 (Serial)]

Specify the types of the breakpoint to use with priority (breakpoint that can be set with one-click operation of the mouse) in the [First using type of breakpoint] property.

Note, however, that if the number of the set breakpoints of the specified type exceeds the limit settable (see "(1)　Maximum number of enabled events"), a breakpoint of another type will be used.

Select from the drop-down list below for each use of the breakpoint.

Software break	Temporary insert the instruction for the break right before/after the code at the specified address and stop the program (default). This breakpoint is handled as a Software Break event.

Hardware break	The debug tool checks the execution of the code at the address specified while the program in execution by the minute and stop the program when it is executed[Note]. This breakpoint is handled as a Hardware Break event (execution).

Note There are two types of Hardware Break event: "before execution break," which breaks before the instruction at the specified address is executed; and "after execution break," which breaks after the instruction at the specified address is executed. Both of these break types are implemented using the debug tool's resources.

CubeSuite starts by using "before execution break" resource to set hardware breakpoints, and as soon as that resource becomes unavailable, uses "after execution break" resource (see "(1)   Maximum number of enabled events"). For this reason, the user cannot select between "before" and "after" execution.

(b) [Simulator]

Figure 2-125.   [Break] Category [Simulator]
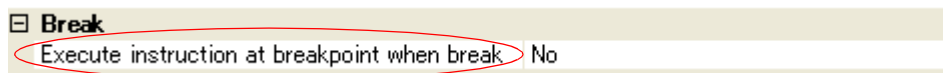
You can specify the timing to stop the program execution by breakpoints whether after or before the execution of the instruction at the breakpoint. Specify in the [Execution instruction at breakpoint when break] property whether to break after executing the instruction.

Select [Yes] to break after execution of the instruction ([No] is selected by default).

All set breakpoints are handled as Hardware Break events.

Caution   When [Yes] is selected, all of action events currently being set are handled as Hardware Break events (see "2.14   Set an Action into Programs").

(2) Set a breakpoint

Perform this operation in the Editor panel/Disassemble panel in which the source text/disassembly text is displayed.

In the panel's event area, click on the location where you want to set a breakpoint.

A breakpoint is set to the instruction at the start address corresponding to the clicked line.

When a breakpoint is set, the following event mark appears at the breakpoint location, and the source text line/ disassembled text line is highlighted.

It is interpreted as if a break event (Software or Hardware Break event) has been set at the target address, and it is managed in the Events panel (see "2.15   Manage Events" for details).

Table 2-7.   Event Marks of the Breakpoint

Event Type	Event Mark
Software Break **[IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]**	
Hardware Break	

Figure 2-126.   Breakpoint Setting Example
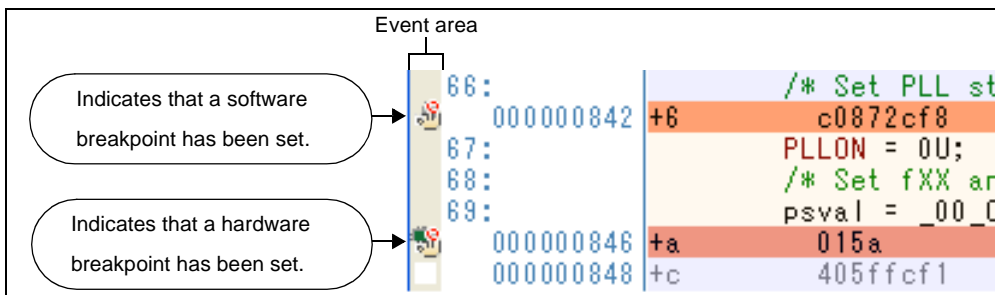
Cautions 1.   Since a breakpoint is set as a break event and managed as a event, restrictions apply to the number of breakpoints that can be simultaneously set. Also see "2.15.6   Notes for setting events" for details on breakpoints (e.g. limits on the number of enabled events).

2.   [IECUBE][IECUBE2][MINICUBE][E1 (JTAG)]
The real-time display update function and software break function are exclusive of each other. Therefore, software breaks cannot be used when the real-time display update function is enabled (see "(4)   Display/modify the memory contents during program execution").

3.   No software breaks can be set to the data flash memory area.

4.   Breakpoints cannot be set at lines in the event area with the background color in gray (showing to the corresponding address of the line that no change is allowed).
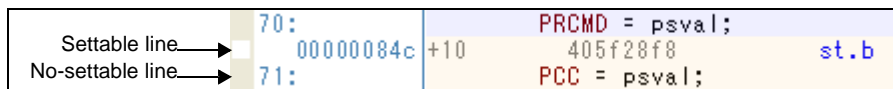
Remarks 1.   Event marks differ depending on the event state (see "2.15.1   Change the state of set events (valid/invalid)").
When an event is set at the point which other event is already set, the event mark ( ) is displayed meaning more than one event is set at the point.

2.   [Simulator]
The type of breakpoint that can be set is locked to hardware breakpoints.

3.   [IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]
You can set hardware breakpoints/software breakpoints without depending on the specification of "(1)   Configure types/movements of breakpoints to use" by following the step below.

Type	Operation1	Operation2
Hardware breakpoint	[Ctrl] + mouse click	Select [Break Settings] >> [Set Hardware Break] from the context menu.
Software breakpoint	[Shift] + mouse click	Select [Break Settings] >> [Set Software Break] from the context menu.

(3) Delete a breakpoint
Click event marks displayed in the Editor panel/Disassemble panel to delete set breakpoints (the event mark will be erased).

2.8.3   Stop the program with the access to variables/I/O registers

By setting a break event with the access, the program can be stopped when an arbitrary variable or I/O register is accessed with the specified type.

You can also limit the accessed value.
The following types can be specified with the access.

Table 2-8.   Types of Accesses to Variables

Access Type	Description
Read	The program is stopped with the read access to (after reading) the specified variable/I/O register.
Write	The program is stopped with the write access to (after writing) the specified variable/I/O register.
Read/Write	The program is stopped with the read access/write access to (after reading or writing) the specified variable/I/O register.

Caution   The program is not stopped with the access via DMA (Dynamic Memory Access.).

This section describes the following.

    (1) Set a break event to a variable/I/O register
    (2) Delete a break event

(1) Set a break event to a variable/I/O register
Follow the steps below to set a break event to stop programs with the access to a variable/I/O register.

Caution   Also see "2.15.6   Notes for setting events" for details on breakpoints (e.g. limits on the number of enabled events).

(a) Set a break event to a variable/I/O register in the source text/disassembled text
Perform this operation in the Editor panel/Disassemble panel in which the source text/disassembly text is displayed.
Follow the operation listed below from the context menu after selecting an arbitrary variable or I/O register in the source text/disassembled text. Note, however, that only global variables, static variables inside functions, and file-internal static variables can be used.
By performing the following operation, it is interpreted as if a Hardware Break event (access) has been set at the target variable/I/O register, and it is managed in the Events panel (see "2.15   Manage Events" for details).

Access Type	Operation
Read	Select [Break Settings] >> [Set Read Break to], and then press the [Enter] key. Break occurs when the variable/I/O register is read with the specified value while a value is specified in the text box in the menu. If values are not specified, break occurs when the selected variable/I/O register is read regardless of its value.
Write	Select [Break Settings] >> [Set Writ Break to], and then press the [Enter] key. Break occurs when the variable/I/O register is written with the specified value while a value is specified in the text box in the menu. If values are not specified, break occurs when the selected variable/I/O register is written regardless of its value.
Read/Write	Select [Break Settings] >> [Set R/W Break to], and then press the [Enter] key. Break occurs when the variable/I/O register is read/written with the specified value while a value is specified in the text box in the menu. If values are not specified, break occurs when the selected variable/I/O register is read/written regardless of its value.

Cautions 1.   Variables within the current scope can be specified.

2.   Break events cannot be set to variables/I/O register on lines in the event area with the
        background color in gray (showing to the corresponding address of the line that no
        change is allowed). Set the break event to a variable/I/O register on the line in the event
        area with background color in white.

Figure 2-127.   Example of Setting a Break Event on a Variable in the Source Text

(b) Set a break event to a registered watch-expression

You can set break events in the Watch panel.

Follow the operation listed below from the context menu after selecting the registered watch-expression.

Note, however, that only global variables, static variables inside functions, file-internal static variables, and I/O register can be used.
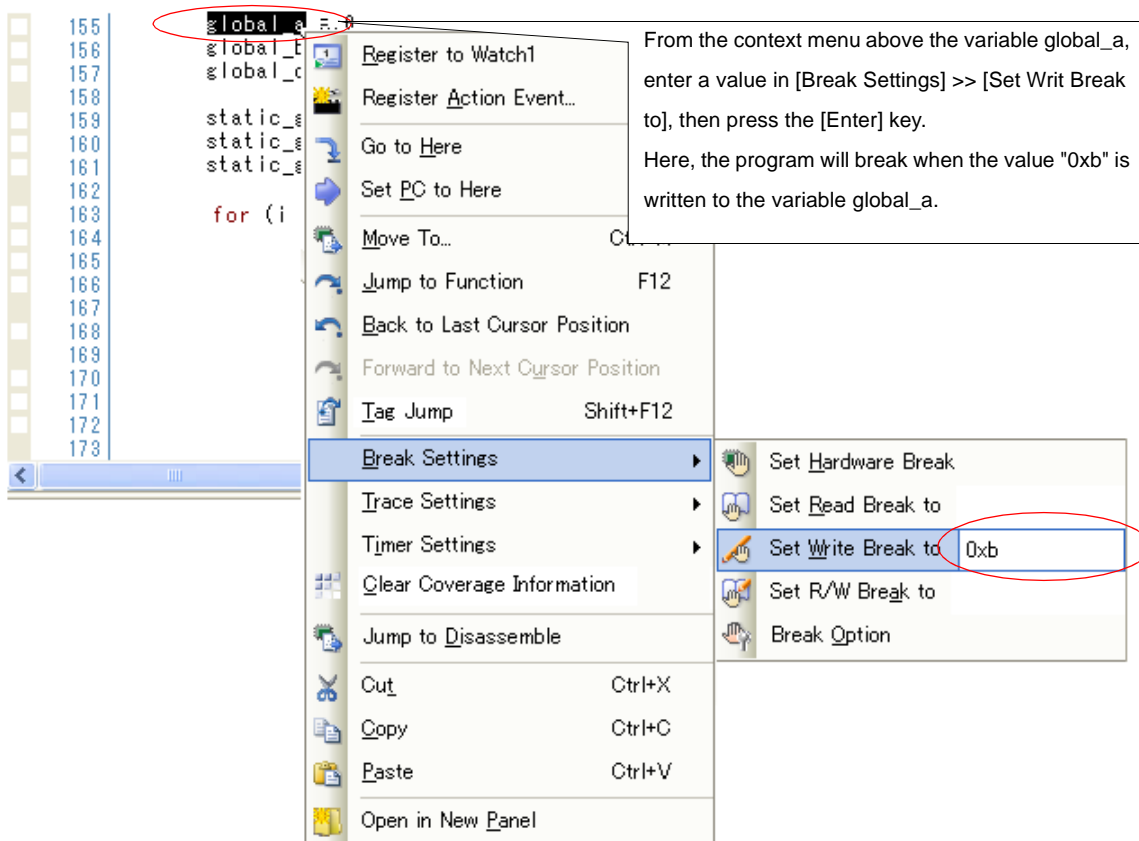
By performing the following operation, it is interpreted as if a Hardware Break event (access) has been set at the target watch-expression, and it is managed in the Events panel (see "2.15   Manage Events" for details).

Access Type	Operation
Read	Select [Access Break] >> [Set Read Break to], and then press the [Enter] key.
	Break occurs when the watch-expression is read with the specified value while a value is specified in the text box in the menu. If values are not specified, break occurs when the selected watch-expression is read regardless of its value.
Write	Select [Access Break] >> [Set Writ Break to], and then press the [Enter] key.
	Break occurs when the watch-expression is written with the specified value while a value is specified in the text box in the menu. If values are not specified, break occurs when the selected watch-expression is written regardless of its value.

Access Type	Operation
Read/Write	Select [Access Break] >> [Set R/W Break to], and then press the [Enter] key. Break occurs when the watch-expression is read/written with the specified value while a value is specified in the text box in the menu. If values are not specified, break occurs when the selected watch-expression is read/written regardless of its value.

Remark   A watch-expression within the current scope can be specified.
             To target a watch-expression outside the current scope, select a watch-expression with a specified scope.

Figure 2-128.   Example of Setting a Break Event on a Watch-Expression
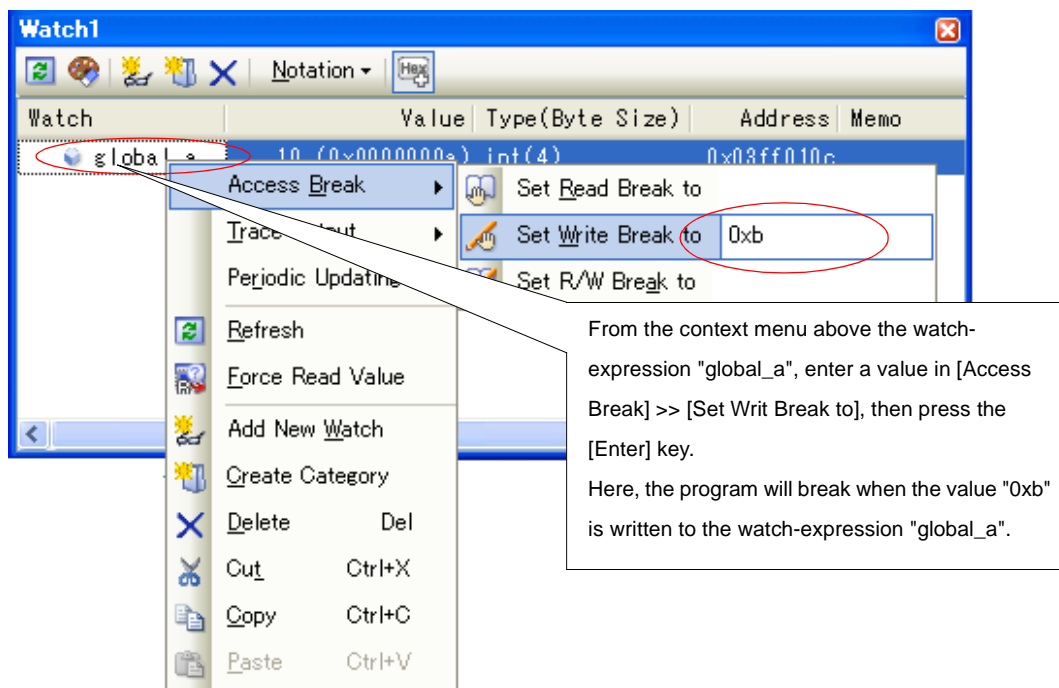
(2) Delete a break event

To delete break events, select the event name to delete in the Events panel opened when selecting the [View] menu >> [Event], then press the ✕ button in the toolbar in the same panel (see "2.15   Manage Events").

2.8.4   Stop the program when an invalid execution is detected [IECUBE]

The system forcibly breaks the program execution when unexpected program behavior such as invalid access to internal ROM/internal RAM/I/O register/external memory is detected (fail-safe break function).

This function has various break conditions. Enable/disable each break condition in the [Fail-safe Break] category in the [Debug Tool Settings] tab of the Property panel.

Figure 2-129.   [Fail-safe Break] Category

In the following property setting, select [Yes] to enable and [No] to disable the function from the drop-down list.

All the properties are set to [Yes] by default.

[icon] mark in the setting column indicates that if you change the setting of this property value, CPU reset is generated automatically.

- [Stop when wrote to Internal ROM][Note 1]
- [Stop when accessed to non-mapping Internal ROM][Note 1]
- [Stop when accessed to non-mapping Internal RAM]
- [Verify when wrote to Read-only Internal RAM][Note 2]
- [Stop when read from read protected I/O Register]
- [Stop when wrote to write protected I/O Register]
- [Stop when accessed to non-mapping I/O Register]
- [Stop when wrote to write protected External memory]
- [Stop when accessed to non-mapping External Memory]

Notes 1.   This property is invalid when the selected microcontroller is a ROMless product.

2.   Specify whether to perform a verify check that detects a write operation to the write protected area in the internal RAM when the program execution is stopped.

2.8.5   Other break causes

The cause of the break other than the described above is as follows:

Moreover, you can confirm the break cause on the Statusbar in the Main window.

Table 2-9.   Break Cause (Other Than the Above)

Break Cause	Debug Tool to Use				
	IECUBE	IECUBE2	MINICUBE E1 (JTAG)	MINICUBE2 E1 (Serial)	Simulator
Full of the trace memory[Note 1]	✓	✓	-	-	✓
An occurrence of Timer Over Break[Note 2]	✓	✓	✓ [Note 3]	-	-
An access to non-mapped area	✓	-	-	-	✓
A writing to write-protected area	✓	-	-	-	✓
Flash macro service in progress	-	✓	✓	✓	-
An occurrence of Temporary Break	✓	✓	✓	✓	✓
An occurrence of Escape Break	✓	✓	✓	✓	-
A writing to IRAM write-protected area	✓	-	-	-	-

Break Cause	Debug Tool to Use				
	IECUBE	IECUBE2	MINICUBE E1 (JTAG)	MINICUBE2 E1 (Serial)	Simulator
An occurrence of the illegal instruction exception	✓	✓	✓	-	-
Failure to execute/uncertain cause	✓	✓	✓	✓	-

Notes 1. The operation depends on the setting of the [Operation after trace memory is full] property in the [Trace] category on the [Debug Tool Settings] tab of the Property panel.

    2. See "2.12.3   Measurable time ranges".

    3. Only V850E2M core

2.9   Display/Change the Memory, Register and Variable

This section describes how to display/change the memory, register and variable.

2.9.1   Display/change the memory

The contents of the memory can be displayed and its values can be changed in the Memory panel below.

Select the [View] menu >> [Memory] >> [Memory *1 - 4*].

The maximum of 4 Memory panels can be opened. Each panel is identified by the names "Memory1", "Memory2", "Memory3" and "Memory4" on the title bar.

For details on the contents and function in each area, see the section for the Memory panel.

Figure 2-130.   Display the Memory Contents (When the Microcontroller without Data Flash Memory)
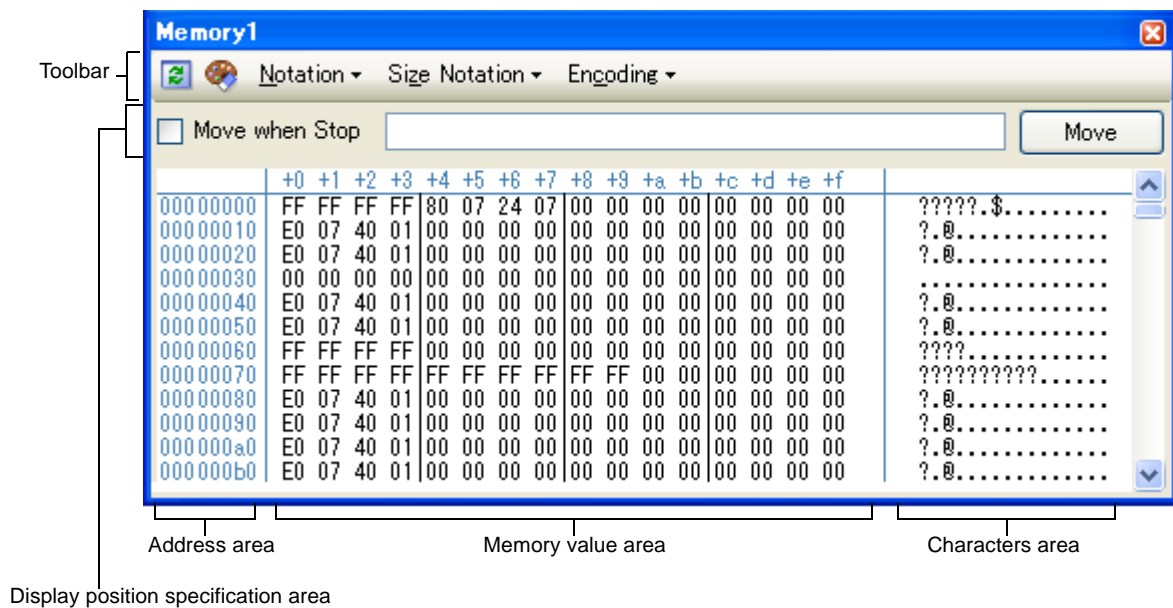
Figure 2-131.   Display the Memory Contents (When the Microcontroller with Data Flash Memory)
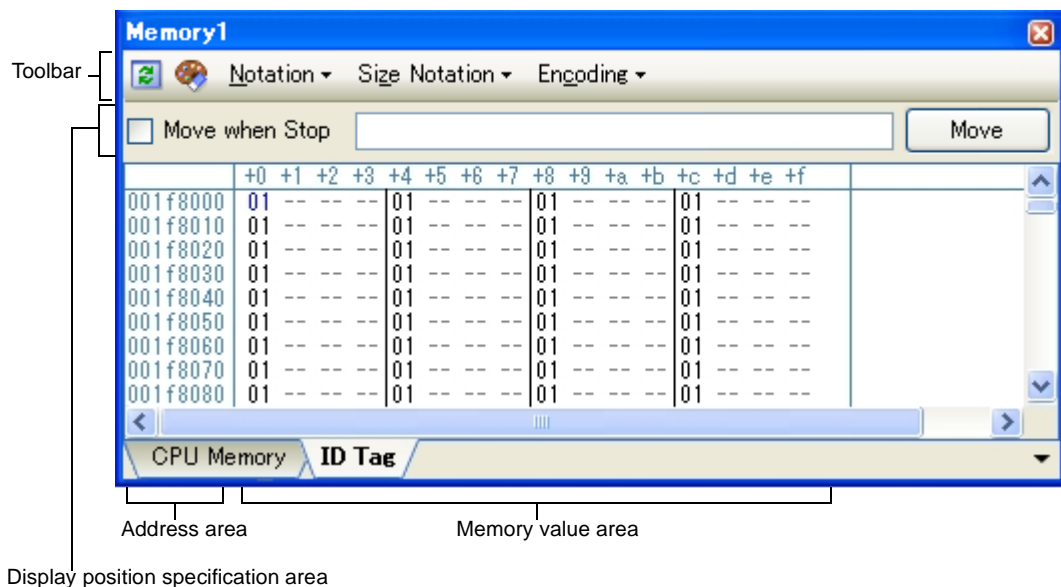
Remark   [IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]
         In the Memory panel, the value of the ID tag for the data flash memory can also be displayed/changed when
         the selected microcontroller incorporates the data flash memory.
         For details on the ID tag, see the section for the Memory panel.

This section describes the following.

(1) Specify the display position
It is possible to specify the display start position of the memory contents by specifying an address expression in the
display position specification area (starting with address "0" by default).

Figure 2-132.   Display Position Specification Area (Memory Panel)
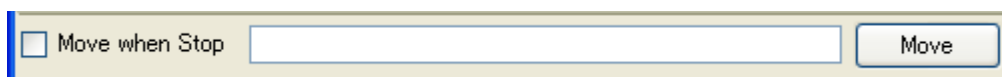
(a) Specify an address expression
Directly enter the address expression of the memory value address to display in the text box. You can specify
an input expression with up to 1024 characters. The result of the expression is treated as the display start
position address.
Note that address values greater than the microcontroller address space cannot be specified.

(b) Specify automatic/manual evaluation of the address expression
The timing to change the display start position can be determined by specifying in the [Move when Stop] check
box and the [Move] button.

[Move when Stop]	☑	The caret is moved to the address which is automatically calculated from the address expression after the program is stopped.
	☐	The address expression is not automatically evaluated after the program is stopped. Click the [Move] button to manually evaluate the address expression.
[Move]		When the [Move when Stop] check box is not checked, click this button to evaluate the address expression and move the caret to the result address of the evaluation.

Remark   If the specified address expression is the symbol and its size can be recognized, everything from the
         start address to the end address of that symbol is displayed selected.

(2) Change display format of values
The display format of the memory value area/character string area can be changed using buttons below on the
toolbar.
Note that the button is invalid while program is executed.

Notation	The following buttons to change the notation of memory values are displayed.
Hex	Displays memory values in hexadecimal number (default).
S Dec	Displays memory values in signed decimal number.
U Dec	Displays memory values in unsigned decimal number.
Oct	Displays memory values in octal number.
Bin	Displays memory values in binary number.
Size Notation	The following buttons to change the notation of sizes of memory values are displayed.
4	Displays memory values in 4-bit width.
8	Displays memory values in 8-bit width (default).
16	Displays memory values in 16-bit width.
32	Displays memory values in 32-bit width.
64	Displays memory values in 64-bit width.
Encoding	The following buttons to change the encoding of character strings are displayed.
SJIS	Displays character strings in ASCII code (default).
SJIS	Displays character strings in Shift_JIS code.
EUC	Displays character strings in EUC-JP code.
UTF	Displays character strings in UTF-8 code.

(3) Modify the memory contents

The memory values can be edited.

Directly edit from the keyboard after moving the caret to the line to modify in memory value area/characters area. The color of the memory value changes when it is in editing. Press the [Enter] key to write the edited value to the target memory (if the [Esc] key is pressed before the [Enter] key is pressed, the editing is cancelled).

However, the character string that can be inputted during the editing is limited to that character string that can be handled by the display notation that has been currently specified. In the characters string area, modification can only be made with "ASCII" character code.

Remark   This operation can be taken place while the program is in execution. See "(4)   Display/modify the memory contents during program execution" for details on how to operate it.

When you modify the values, be aware of the following examples.

Examples 1.   The value exceeds the upper limit of the display bit wide
If you edit the display value "105" as "1" to "3" in the decimal 8-bit display, the value will be changed to the upper limit of "127".

2.   The symbol, "-" is entered between numbers
If you edit the display value "32768" as "32-68" with signed decimal 16-bit display, "3" and "2" are changed to the blank and the value is changed to "-68".

3.   The blank symbol (space) is entered between numbers
If you edit the display value "32767" as "32 67", "3" and "2" are changed to the blank and the value is changed to "67".

4.   The same value is entered
Even if the same value as the current memory value is specified, the specified value is written to the memory.

(4) Display/modify the memory contents during program execution

The Memory panel/Watch panel has the real-time display update function that can update/modify the display contents of the memory/watch-expression in real-time while executing the program.

Using the real-time display update function enables to display/modify the value of memory/watch-expression not only while the program is stopped, but also in execution.

Caution **[IECUBE]**

Some of the trace function, the real-time display update function (RRM function) and the coverage function are used on a mutually exclusive basis. Therefore, you need to configure the following setting in the same category in order to specify a function used preferentially, before configuring the settings described in this section.

- [Trace] category >> [Use for trace data] property >> [RRM] (default)

Remark The real-time display update function is implemented using the debug tool's RRM (Real-time RAM Monitor) function, pseudo RRM function, DMM (Dynamic Memory Modification) function and pseudo DMM function. The memory area is read/modified by software simulation while the pseudo RRM function/pseudo DMM function is executed, so execution of the program momentarily breaks upon a read/modification.

The target range where the values can be read/modified using the real-time display update function differs depending on the debug tool and the settings on the Property panel.

Based on the following settings of properties in the [Access memory while running] category on the [Debug Tool Settings] tab of the Property panel, configure "Table 2-10."/"Table 2-11."/"Table 2-12."/"Table 2-13." in accordance with the debugging purpose.

Mark	Properties in [Access memory while running] Category	Setting Value	Note
A	[Update display during the execution]	[Yes] (default)	Enables the RRM/DMM function.
	[Display update interval[ms]]	[*Integer number between 100 and 65500*]	
B	[Access by stopping execution]	[Yes]	Enables the pseudo RRM/DMM function. (Except for Simulator and the microcontroller that supports multi-core)
C	[Set update display during the execution automatically]	[Yes] (default)	**[IECUBE]**

(a)　Reading

Table 2-10.　Target Range for Real-Time Display Update Function (Reading) [V850E1][V850ES]

Area	IECUBE		MINICUBE E1 (JTAG)		MINICUBE2 E1 (Serial)		Simulator	
	Setting	Target	Setting	Target	Setting	Target	Setting	Target
Internal ROM	A+C	Automatically determined range[Note]	A+B	Entire range	Impossible		A	Entire range
Internal RAM					A+B	Entire range		
Emulation memory			-		Impossible			
Target memory			A+B	Entire range				
CPU register	A+B	Entire range						
I/O register								
Data flash (except ID tag)					A+B	Entire range	Impossible	

Note　[IECUBE]

The target range for the RRM function is limited to up to 8 locations within 2048 bytes.

Therefore, in CubeSuite, it follows the determination rule (priority order) below and determines the target of the real-time display update within the above limitation (the panel displayed in front right before the program execution is the only target).

(1) Ascending order of watch-expressions listed in the Watch panel (if more than one panel is opened, the smallest in number is configured first).

(2) Ascending order of the addresses listed in the Memory panel (if more than one memory panel is opened, the smallest in number is configured first).

Note that when a read-protected area is included in this target range, the display contents of the area are not updated.

Table 2-11.　Target Range for Real-Time Display Update Function (Reading) [V850E2M]

Area	IECUBE		MINICUBE E1 (JTAG)		Simulator	
	Setting	Target	Setting	Target	Setting	Target
Internal ROM	A+B	Entire range	A+B	Entire range[Note]	A	Entire range
Internal RAM						
HBUS shared memory						
Back-up RAM						
Emulation memory						
Target memory						
CPU register						
I/O register						
Data flash (except ID tag)					Impossible	

Note   When the selected microcontroller version supports multi-core, the target range is limited to the internal RAM area and the HBUS shared memory area.

(b) Modifying

Table 2-12.   Target Range for Real-Time Display Update Function (Modifying) [V850E1][V850ES]

Area	IECUBE		MINICUBE E1 (JTAG)		MINICUBE2 E1 (Serial)		Simulator	
	Setting	Target	Setting	Target	Setting	Target	Setting	Target
Internal ROM	Impossible		Impossible		Impossible		A	Entire range
Internal RAM	A+B	Entire range	A+B	Entire range	A+B	Entire range		
Emulation memory			-		-			
Target memory			A+B	Entire range	Impossible			
CPU register					A+B	Entire range		
I/O register (Standard I/O register only)								
Data flash	Impossible		Impossible		Impossible		Impossible	

Table 2-13.   Target Range for Real-Time Display Update Function (Modifying) [V850E2M]

Area	IECUBE		MINICUBE E1 (JTAG)		Simulator	
	Setting	Target	Setting	Target	Setting	Target
Internal ROM	Impossible		Impossible		A	Entire range
Internal RAM	A+B	Entire range	A+B	Entire range		
HBUS shared memory						
Back-up RAM						
Emulation memory	-		-			
Target memory	A+B	Entire range	A+B	Entire range		
CPU register						
I/O register						
Data flash	Impossible		Impossible		Impossible	

Cautions 1.   When CPU status shifts to the standby mode during this operation:
               [IECUBE][IECUBE2][MINICUBE][E1 (JTAG)]
               When the [Access by release HALT mode] property in the [Access memory while running] category has been set to [Yes], if CPU status shifts to the HALT mode, the real-time display update function will continue to operate by automatically releasing the HALT mode. If CPU status shifts to the STOP/IDLE mode, a monitor time-out error will occur.
               When that property has been set to [No], if CPU status shifts to the HALT/STOP/IDLE mode, a monitor time-out error will occur.

[MINICUBE2][E1 (Serial)][Simulator]

If CPU status shifts to the HALT/STOP/IDLE mode, a monitor time-out error will occur.

2.　**Local variables are not subject to the real-time display update function.**

3.　**[IECUBE][IECUBE2][MINICUBE][E1 (JTAG)]**

The real-time display update function and software breaks are exclusive of each other.

Therefore, software breaks cannot be used when the settings above have been configured.

4.　**[IECUBE]**

If the values of a variable expression are aligned across RRM area banks (256-byte boundaries), the upper address will be an illegal value (the value will be displayed in the error color).

Remark　See "(3)　Modify the memory contents" or "(6)　Modify the contents of watch-expressions" for details on how to modify values in the Memory panel/Watch panel.

Values of memory/watch-expression for which the real-time display update function is being operated are highlighted as follows (character colors and background colors depend on the configuration in the [General - Font and Color] category of the Option dialog box):

Figure 2-133.　Example of Memory Values where the Real-time Display Update Function Is Being Operated
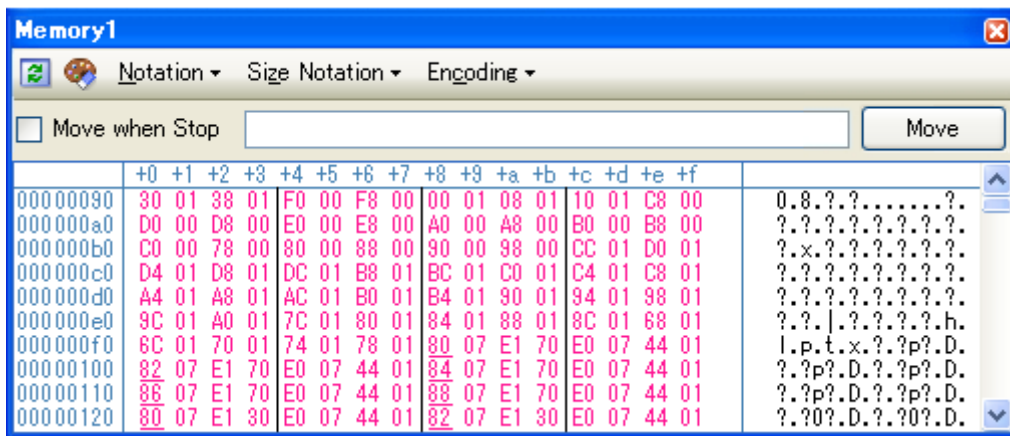
(5) Search the memory contents

Values of memory can be searched in the Memory Search dialog box opened by selecting [Find...] from the context menu. The search is operated either in the memory value area or characters string area, in which a caret is.

In this dialog box, follow the steps below.

Figure 2-134.　Search the Memory Contents (Memory Search Dialog Box)
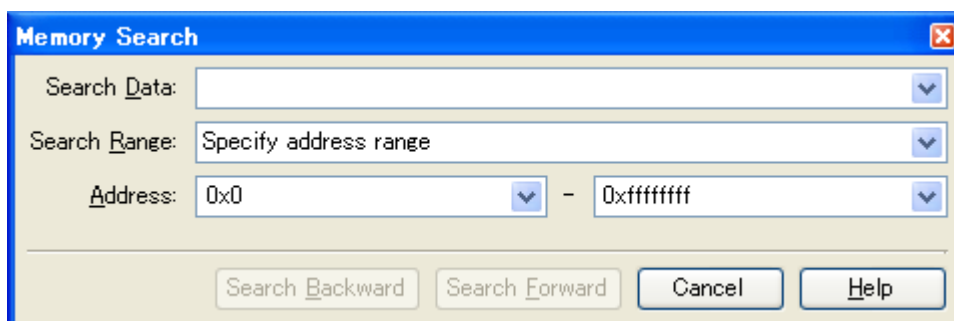
Cautions 1.   Search for the contents of the memory cannot be performed while executing the program.

2.   Search for values of the ID tag for the data flash memory cannot be performed.

(a) Specify [Search Data]

Specify data to search.

You can either type a value directly into the text box (up to 256 bytes), or select one from the input history via the drop-down list (up to 10 items).

If the search is performed in the memory value area, the value must be entered in the same display format (notation and size) as that area.

If the search is performed in the character string area, then the target of the search must be a string. The specified string is converted into the encoding format displayed in that area, and searched for.

If a memory value was selected immediately prior to opening this dialog box, then that value will appear as default.

(b) Specify [Search Range]

Select the range to search from the following drop-down list.

Specify address range	Searches in the address range specified in the [Address] area.
Memory mapping	Searches within the selected memory mapping range.
	This list item displays individual memory mapping configured in the Memory Mapping dialog box (except the non-mapped area).
	Display format: *<memory type> <address range> <size>*

(c) Specify [Address]

This item is only enabled if [Specify address range] is selected in the (b)   Specify [Search Range].

Specify the range of memory address to search via the start and end addresses. You can either type address expressions directly into the text boxes (up to 1024 characters), or select them from the input history via the drop-down list (up to 10 items).

The results of calculating the address expressions you have entered are treated as start and end addresses, respectively.

Note, however, that the largest address that can be searched is the maximum address of the program space (0x03FFFFFF) (the mirror area cannot be searched).

An address value greater than the value expressed within 32 bits cannot be specified.

Remarks 1.   If the start address field is left blank, it is treated as if "0x0" were specified.

2.   If the end address field is left blank, then it is treated as if the maximum value in the microcontroller's address space were specified.

(d) Click the [Search Backward]/[Search Forward] button

When the [Search Backward] button is clicked, search is taken place in the order from the large address number to small and the search results are displayed selected in the Memory panel.

When the [Search Forward] button is clicked, search is taken place in the order from the small address number to small and the search results are displayed selected in the Memory panel.

(6) Modify the memory contents in batch (initialize)

Contents of the memory can be modified in batch (initialize).

When [Fill...] from the context menu is selected, the Memory Initialize dialog box opens to modify the memory value of the specified address range in batch.

In this dialog box, follow the steps below.

Figure 2-135.   Modify the Memory Contents in Batch (Memory Initialize Dialog Box)
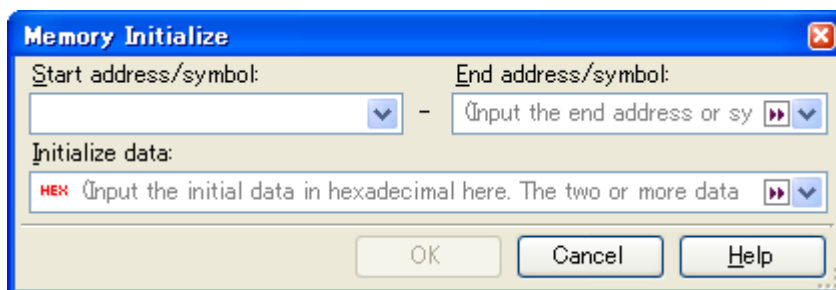
Caution   This operation cannot be performed for values of the ID tag for the data flash memory.

(a) Specify [Start address/symbol] and [End address/symbol]
Specify the range of memory address to initialize via the [Start address/symbol] and [End address/symbol].
You can either type address expressions directly into the text boxes (up to 1024 characters), or select them
from the input history via the drop-down list (up to 10 items).
The results of calculating the address expressions you have entered are treated as start and end addresses,
respectively.
Note that address values greater than the microcontroller address space cannot be specified.

(b) Specify [Initialize data]
Specify the initializing data to write to the memory.
You can either type the initial value into the text box directly in hexadecimal number, or select one from the
input history via the drop-down list (up to10 items).
You can specify more than one initial value. Specify up to 16 values of up to 4 bytes (8 characters) each,
separated by spaces.
Each initial data value is parsed from the end of the string, with each two characters interpreted as a byte.
If the string has an odd number of characters, then the first character is interpreted as one byte.

Example   Endianness specified by debug tool is little endian

Input Character String (Initialization Data)	How Data is Overwritten (in Bytes)
1	01
0 12	00 12
00 012 345	00 12 00 45 03
000 12 000345	00 00 12 45 03 00

(c) Click the [OK] button
Click the [OK] button.
The memory area in the specified address range is repeatedly overwritten with the specified initial data
pattern. If the end address is reached in the middle of the pattern, then writing ends at that point.
Note that if an illegal value is specified, a message will be appear, and the memory value will not be initialized.

(7) Save the memory contents
Contents of the memory can be saved with range selection in text files (*.txt)/CSV files (*.csv).
When saving to the file, the latest information is acquired from the debug tool, and it is saved according with the
display format on this panel.

The Data Save dialog box can be opened by selecting the [File] menu >> [Save Memory Data As...] (when this operation is taken place with range selection on the panel, the memory data only in the selected range is saved). In this dialog box, follow the steps below.

Figure 2-136.   Save the Memory Data (Data Save Dialog Box)

(a) Specify [File Name]

Specify the name of the file to save.

You can either type a filename directly into the text box (up to 259 characters), or select one from the input history via the drop-down list (up to 10 items).

You can also specify the file by clicking the [...] button, and selecting a file via the Select Data Save File dialog box.

(b) Specify [File Type]

Select the format in which to save the file from the following drop-down list.

The following file formats can be selected.

List Item	Format
Text files (*.txt)	Text format (default)
CSV (Comma-Separated Variables)(*.csv)	CSV format[Note]

Note   The data is saved with entries separated by commas (,).

If the data contains commas, each entry is surrounded by double quotes "" in order to avoid illegal formatting.

(c) Specify [Save Range Address/Symbol]

Specify the range of addresses to save via "start address" and "end addresses".

Directly enter hexadecimal number/address expression in each text box or select from the input history displayed in the drop-down list (up to 10 items).

If a range is selected in the panel, that range is specified as the default. If there is no selection, then the range currently visible in the panel is specified.

(d) Click the [Save] button

Saves the memory data to a file with the specified filename, in the specified format.

Figure 2-137.   Output Example of the Memory Data

[Text files (*.txt)]

(Hexadecimal notation/8-bit width/ASCII code)

```
             +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f
00000000 |   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |
00000010 |   11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  |
```

[CSV files (*.csv)]

(Hexadecimal notation/8-bit width/ASCII code)

```
00000000,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
00000010,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,
```

Remark   When the contents of the panel are overwritten with selecting the [File] menu>> [Save Memory Data],
each Memory panel (Memory1-4) is treated as a different panel.
In addition, saving range is same as the previously specified address range.

2.9.2   Display/change the CPU register

The contents of the CPU register (program register/system register) can be shown and the value can be changed in the
CPU Register panel below.

Select the [View] menu >> [CPU Register].

For details on the contents and function in each area, see the section for the CPU Register panel.

Figure 2-138.   Display the Contents of the CPU Register (CPU Register Panel)
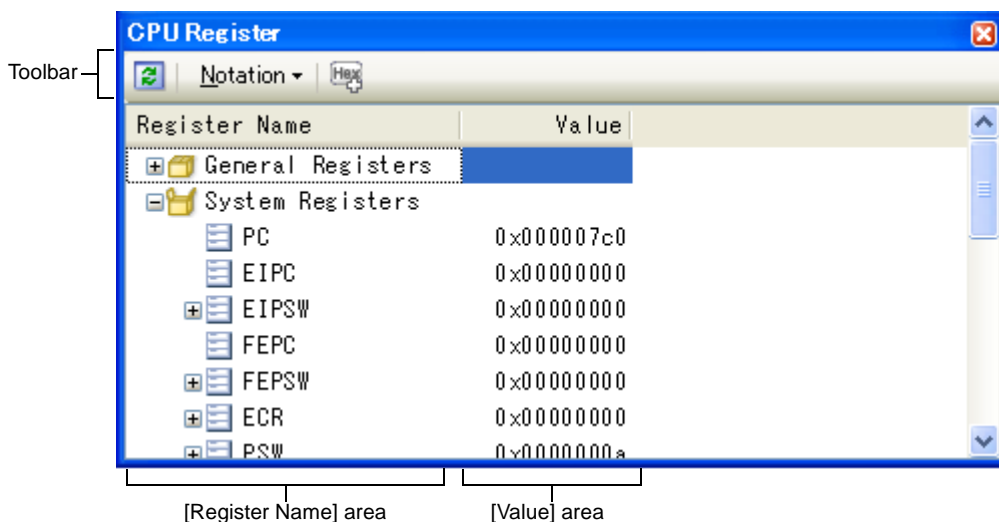
This section describes the following.

(1) Change display format of values
(2) Modify the CPU register contents
(3) Display/modify the CPU register contents during program execution
(4) Save the CPU register contents

(1) Change display format of values

The display format of the [value] area can be changed using buttons below on the toolbar.

Notation	The following buttons to change the notation of a data value are displayed.
auto	Displays the value of the selected item (including sub-items) in the default notation (default).
Hex	Displays the value of the selected item (including sub-items) in hexadecimal number.
S Dec	Displays the value of the selected item (including sub-items) in signed decimal number.
U Dec	Displays the value of the selected item (including sub-items) in unsigned decimal number.
Oct	Displays the value of the selected item (including sub-items) in octal number.
Bin	Displays the value of the selected item (including sub-items) in binary number.
Asc	Displays the character string of the selected item (including sub-items) in ASCII code. If the character size is 2 bytes and above, it is displayed with the characters for each 1 byte arranged side-by-side.
Flo	Displays the value of the selected item in Float. Note that when the value is not 4-byte data, displays it in the default notation.
Dbl	Displays the value of the selected item in Double. Note that when the value is not 8-byte data, displays it in the default notation.
Hex	Adds the value in hexadecimal number enclosing with "()" at the end of the value.

(2) Modify the CPU register contents

The CPU register values can be edited.

Double-click the value of the CPU register to edit in the [Value] area to switch the value to edit mode (press the [Esc] key to cancel the edit mode).

To write the edited value to the target memory, directly enter the value from the keyboard then press the [Enter] key.

Caution   This operation cannot be performed during program execution.

(3) Display/modify the CPU register contents during program execution

By registering a CPU register to the Watch panel as a watch-expression, the value of the CPU register can be displayed/modified not only while the program is stopped, but in execution.

See "2.9.6   Display/change watch-expressions" for details on the watch-expression.

(4) Save the CPU register contents

The Save As dialog box can be opened by selecting the [File] menu >> [Save CPU Register Data As...], and all the contents in the CPU register can be saved to a text file (*.txt) or CSV file (*.csv).

When saving to files, retrieve the latest information from the debug tool.

Figure 2-139.   Output Example of CPU Register Data

```
Register name           Value
----------------------------------------------------------------------
Category name
 -Register name          Value
      :                    :
```

2.9.3   Display/change the I/O register

Contents of the I/O register can be displayed and its values can be changed in the IOR panel below.

Select the [View] menu >> [IOR]

For details on the contents and function in each area, see the section for the IOR panel.

Figure 2-140.   Display the Contents of the I/O register (IOR Panel)

[IOR] area   [Value] area   [Type (Byte Size)] area   [Address] area
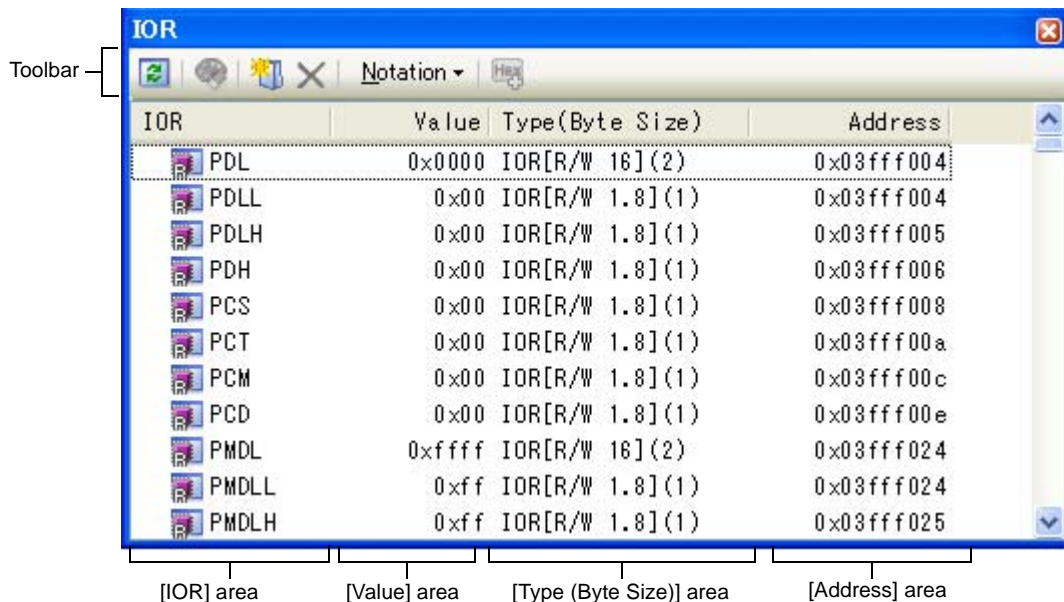
This section describes the following.

- (1) Organize I/O registers
- (2) Change display format of values
- (3) Modify the I/O register contents
- (4) Display/modify the I/O register contents during program execution
- (5) Save the I/O register contents

(1) Organize I/O registers

The each I/O register can be categorized (by folders) and displayed in the tree view.

Cautions 1.   Categories cannot be created within categories.

2.   I/O registers cannot be added or deleted.

(a) Create new category

Move the caret to the I/O register name to create a new category then click the [icon] button in the toolbar and directly enter the new category name.

(b) Edit category name

Click the category name to edit, and click it again, then directly modify the category name from the keyboard.

(c) Delete categories

Select categories to delete then click the [icon] button in the toolbar.

However, the categories that can be deleted are only the empty categories.

(d) Change the display order

I/O register name is categorized when I/O register is dragged and dropped in the category.

Also, the display order of the categories and the I/O register names (upper or lower position) can be changed easily by drag and drop operation.

(2) Change display format of values

The display format of the [value] area can be changed using buttons below on the toolbar.

Notation	The following buttons to change the notation of a data value are displayed.
Hex	Displays the value of the selected item in hexadecimal number (default).
S Dec	Displays the value of the selected item in signed decimal number.
U Dec	Displays the value of the selected item in unsigned decimal number.
Oct	Displays the value of the selected item in octal number.
Bin	Displays the value of the selected item in binary number.
Asc	Displays the value of the selected item in ASCII code.
Hex	Adds the value in hexadecimal number enclosing with "()" at the end of the value of the selected item.

(3) Modify the I/O register contents

The I/O register values can be edited.

Double-click the value of the I/O register to edit in the [Value] area to switch the value to edit mode (press the [Esc] key to cancel the edit mode).

To write the edited value to the target memory, directly enter the value from the keyboard then press the [Enter] key.

Cautions 1. **This operation cannot be performed during program execution.**
2. **The value of the read-only I/O register cannot be edited.**

ASCII characters can be entered to the I/O register value.

Examples 1. When the numeric "0x41" is written to the I/O register "P0"
>> "0x41" is written in the port "P0".
2. When the ASCII character "'A'" is written to the I/O register "P0"
>> "0x41" is written in the port "P0".

(4) Display/modify the I/O register contents during program execution

By registering an I/O register to the Watch panel as a watch-expression, the value of the I/O register can be displayed/modified not only while the program is stopped, but in execution.

See "2.9.6 Display/change watch-expressions" for details on the watch-expression.

(5) Save the I/O register contents

The Save As dialog box can be opened by selecting the [File] menu >> [Save IOR Data As...], and all the contents of the I/O register can be saved in a text file (*.txt) or CSV file (*.csv). At this time, the values of all I/O registers become targets irrespective of the setting of display/non-display on this panel.

When saving the contents to the file, the values of the I/O register are reacquired and save the latest values acquired.

Note that the values of read-protected I/O register are not re-read. If you want to save the latest values of those, select [Force Read Value] from the context menu then save the file.

Figure 2-141.   Output Example of I/O register

```
IOR name        Value       Type (Byte Size)    Address
------------------------------------------------------------
Category name
 -IOR name       Value       Type (Byte Size)    Address
     :             :             :                   :
```

2.9.4   Display/change global variables/static variables

Global variables or static variables are displayed and its values can be changed in the Watch panel.

Register the variables to display/modify their values to the Watch panel as the watch-expressions.

For details, see "2.9.6   Display/change watch-expressions".

2.9.5   Display/change local variables

Contents of local variables can be displayed and its values can be changed in the Local Variables panel below.

Select the [View] menu >> [Local Variable].

Specify the scope in the scope area to display the contents of the target local variable.

In the Local Variables panel, the name of local variables and functions are displayed. The argument of the function is also displayed as the local variable.

For details on the contents and function in each area, see the section for the Local Variables panel.

Caution   Nothing is displayed on this panel during execution of a program.
When the program is stopped, items in each area are displayed.

Figure 2-142.   Display the Contents of Local Variables (Local Variables Panel)

[Name] area   [Value] area   [Type (Byte Size)] area   [Address] area
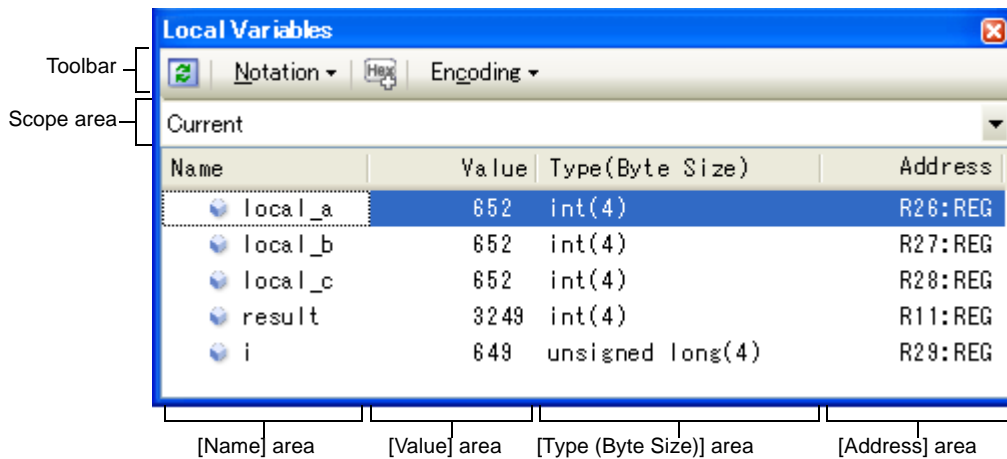
This section describes the following.

(1) Change display format of values

(2) Modify the contents of local variables

(3) Save the contents of local variables

(1) Change display format of values

The display format of the [value] area can be changed using buttons below on the toolbar.

Notation	The following buttons to change the notation of a data value are displayed.
auto	Displays values on this panel in the default notation according to the type of variable (default).
Hex	Displays values on this panel in hexadecimal number.
Dec	Displays values on this panel in decimal number.
Oct	Displays values on this panel in octal number.
Bin	Displays values on this panel in binary number.
Dec	Displays array indexes on this panel in decimal number (default).
Hex	Displays array indexes on this panel in hexadecimal number.
Flo	Displays values on this panel in Float. Note that when the value is not 4-byte data, or has the type information, displays it in the default notation.
Dbl	Displays values on this panel in Double. Note that when the value is not 4-byte data, or has the type information, displays it in the default notation.
Hex	Adds the value in hexadecimal number enclosing with "()" at the end of the value.
Encoding	The following buttons to change the encoding of character variables are displayed.
Asc	Displays character variables in ASCII code (default).
SJIS	Displays character variables in Shift_JIS code.
EUC	Displays character variables in EUC-JP code.
UTF	Displays character variables in UTF-8 code.

(2) Modify the contents of local variables

Values and arguments of local variables can be edited.

Select the value of the local variables/arguments to edit in the [Value] area, then click on it again to switch the value to edit mode (press the [Esc] key to cancel the edit mode).

To write the edited value to the target memory, directly enter the value from the keyboard then press the [Enter] key.At this time, the edited value is checked and if it is incompatible with the type, the editing is invalidated.

Caution   This operation cannot be performed during program execution.

ASCII characters can be entered to values of local variables.

Examples 1.   Entering via an ASCII character
In the [Value] area for the variable "ch", enter "'A'"
>> "0x41" will be written to the memory area allocated to "ch"

2.   Entering via a numeric value
In the [Value] area for the variable "ch", enter "0x41"
>> "0x41" will be written to the memory area allocated to "ch"

3.   Entering via an ASCII string
Set the display format of character array "str" to ASCII, and in the [Value] area, enter ""ABC""
>> "0x41, 0x42, 0x43, 0x00" will be written to the memory area allocated to "str"

Remarks 1.   If a number with fewer digits than the size of the variable is entered, the higher-order digits will be padded with zeroes.

2.   If a number with more digits than the size of the variable is entered, the higher-order digits will be masked.

3.   If the display format of a character array (type char or unsigned char) is set to ASCII, then the value can also be entered as a string (ASCII/Shift_JIS/EUC-JP/Unicode (UTF-8)).

(3) Save the contents of local variables

The Save As dialog box can be opened by selecting the [File] menu >> [Save Local Variables Data As...], and all the contents in the local variables can be saved in a text file (*.txt) or CSV file (*.csv).

When saving to files, retrieve the latest information from the debug tool. If arrays, pointer type variables, structures/unions, and CPU registers (only those with the part name) are displayed expanded, the value of each expanded element is also saved. When they are not expanded, "+" mark is added on the top of the item and the value becomes blank.

Figure 2-143.   Output Example of Local Variables

```
Scope : Current scope
[V]Variable  [P]Parameter  [F]Function
Name                       Value   Type (Byte Size)   Address
-----------------------------------------------------------
[V]Variable name[1]        Value   Type               Address
  - [V]Variable name[0]    Value   Type               Address
            :                :       :                    :
```

2.9.6   Display/change watch-expressions

By registering C language variables, CPU register, I/O register, and assembler symbols to the Watch panel as watch-expressions, you can always retrieve their values from debug tools and monitor the values in batch.

The values of the watch-expressions can be updated during the program is in execution (see "(7)   Display/modify the contents of watch-expressions during program execution").

Select the [View] menu >> [Watch] >> [Watch *1* - *4*] to open the Watch panel.

The Watch panel can be opened up to 4 panels. Each panel is identified by the names "Watch1", "Watch2", "Watch3" and "Watch4" on the title bar, and the watch-expressions can be registered/deleted/moved individually, and they are saved as the user information of the project.

For details on the contents and function in each area, see the section for the Watch panel.

Figure 2-144.   Display the Contents of Watch-Expression (Watch Panel)
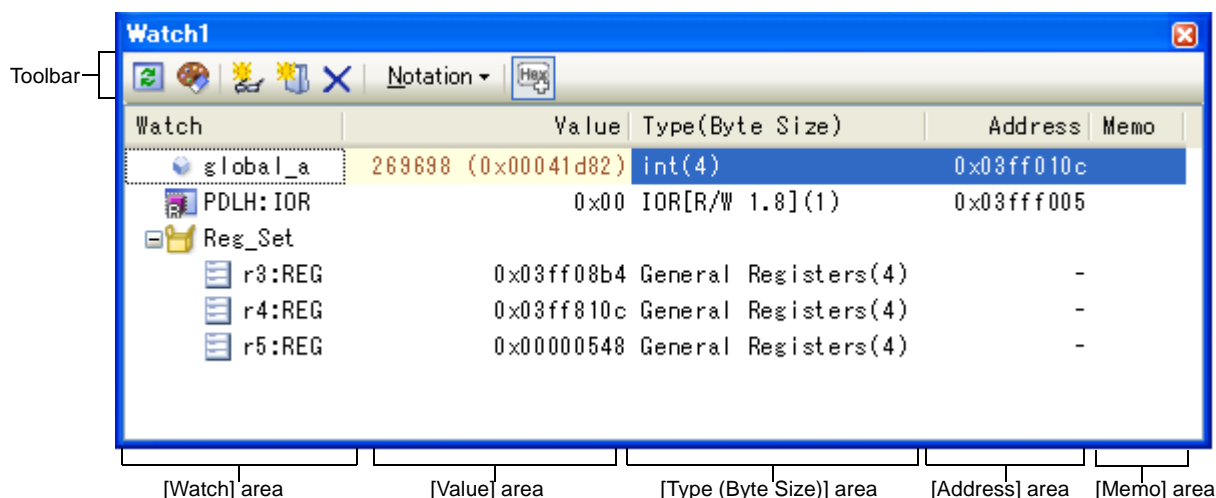
This section describes the following.

- (1) Register a watch-expression
- (2) Organize the registered watch-expressions
- (3) Edit the registered watch-expressions
- (4) Delete a watch-expression
- (5) Change display format of values
- (6) Modify the contents of watch-expressions
- (7) Display/modify the contents of watch-expressions during program execution
- (8) Save the contents of watch-expressions

(1) Register a watch-expression

There are three ways as follows to register watch-expressions (watch-expressions are not registered as default).

Cautions 1. **Watch-expressions can be registered up to 128 in one watch panel (if this restriction is violated, a message appears).**

2. **Due to compiler optimization, the data for the target variable may not be on the stack or in a register in blocks where that variable is not used. In such cases, if the variable is registered as a watch-expression, then the value will be displayed as a question mark "?".**

Remarks 1. Each watch-expression registered in each watch panel (Watch1 to Watch4) is managed in each panel and saved as the user information of the project.

2. More than one watch-expression with the same name can be registered.

(a) Register from other panels

Watch-expressions can be registered from other panel in CubeSuite.

In other panel, drag and drop the watch-expression to register in any watch panel (Watch1 to Watch4).

For the relationship between panels that can use this operation and targets that can be registered as watch-expressions, see "Table A-7. Relationship between Panels and Targets That Can be Registered as Watch-Expressions".

Figure 2-145. Registering the Watch-Expression from Other Panels
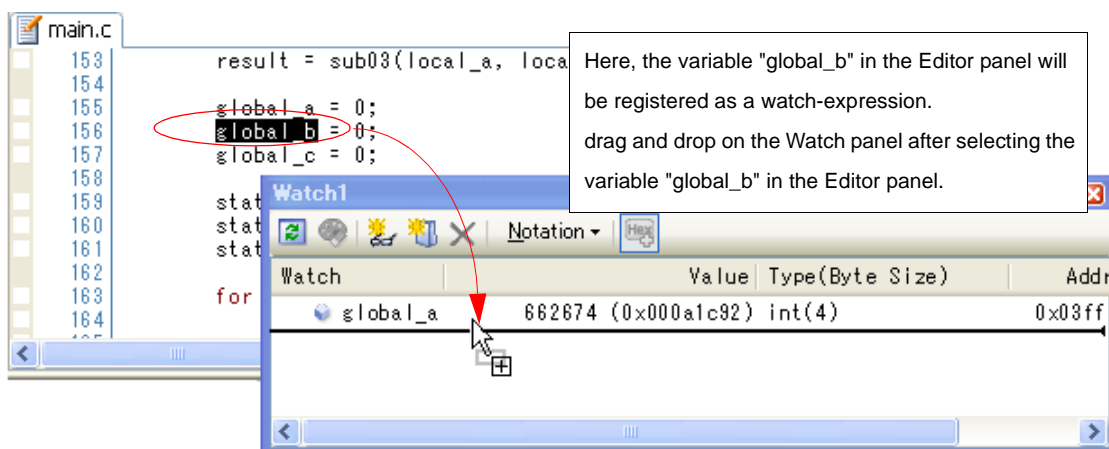
Remark You can also add a watch-expression by doing the following. First, select the target for which you wish to register a watch-expression, or move the caret to one of the target strings (the target is determined automatically). Next, from the context menu, select [Register Watch1] (but this is limited to the Watch panel (Watch 1)).

(b) Directly register in the Watch panel

Click the ![icon] button in the toolbar in any the Watch panel (Watch1 to Watch4) to display the following entry box in the [Watch] area.

Figure 2-146. Entry Box of the Watch-Expression

Directly enter a watch-expression in this area.
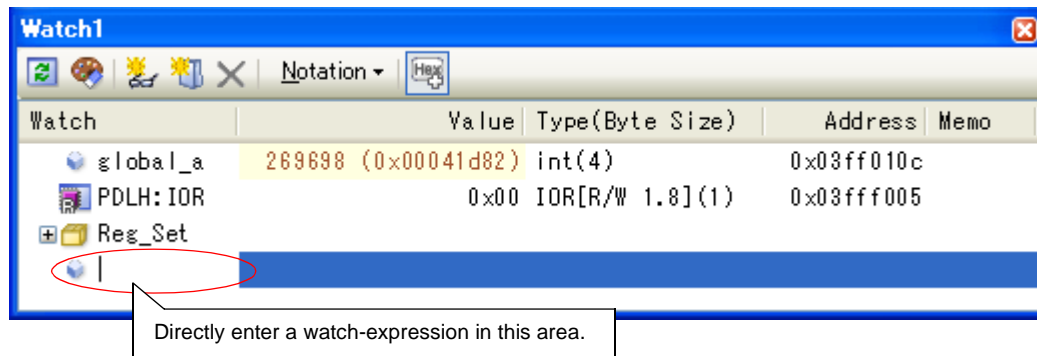
Directly input a watch-expression from the keyboard in the entry box then press the [Enter] key.
The input format of the watch-expression is the following tables.

- "Table A-8. Input Format of Watch-Expression"
- "Table A-9. Scope Specification of C language Used with Watch-Expression Registration"
- "Table A-10. Scope Specification of CPU Register with Watch-Expression Registration"
- "Table A-11. Scope Specification of I/O register with Watch-Expression Registration"

(c) Register from other application

Select the character strings of C language variables/CPU registers/I/O register/assembler symbols from an external editor and drag and drop it in the Watch panel (Watch 1 to Watch 4).
The dropped character strings are registered as a watch-expression.

(2) Organize the registered watch-expressions

Registered watch-expressions can be organized in categories (folders) and displayed in tree view (there is no category as default).

Cautions 1. Categories cannot be created within categories.

2. Up to 64 categories can be created in one watch panel (if this restriction is violated, a message appears).

(a) Create new category

Move the caret to the position to create a new category then click the ![icon] button in the toolbar and directly enter the new category name.

(b) Edit category name

Click the category name to edit, and click it again, then directly modify the category name from the keyboard.

(c) Delete categories

Select categories to delete then click the ![icon] button in the toolbar.

(d) Change the display order

Registered watch-expressions are categorized when they are dragged and dropped in the category.

Also, the display order of the categories and the watch-expressions (upper or lower position) can be changed easily by drag and drop operation.

Remark   Drag and drop the watch-expressions/categories in other watch panel (Watch1 to Watch4) to copy them.

(3) Edit the registered watch-expressions

Registered watch-expressions can be edited.

Double-click the watch-expression to edit to switch the watch-expression to edit mode (press the [Esc] key to cancel the edit mode).

Directly edit from the keyboard and then press the [Enter] key.

(4) Delete a watch-expression

To delete watch-expressions, select the one you want to delete in the Watch panel then click the ⊠ button in the toolbar.

(5) Change display format of values

The display format of the [value] area can be changed using buttons below on the toolbar.

Notation	The following buttons to change the notation of a data value are displayed.
auto	Displays the value of the selected watch-expression in the default notation (see "Table A-12. Display Format of Watch-Expressions (Default)") according the type of variable (default).
Hex	Displays the value of the selected item in hexadecimal number.
S Dec	Displays the value of the selected item in signed decimal number.
U Dec	Displays the value of the selected item in unsigned decimal number.
Oct	Displays the value of the selected item in octal number.
Bin	Displays the value of the selected item in binary number.
Asc	Displays the value of the selected item in ASCII code.
Flo	Displays the value of the selected item in Float. Note that this item becomes valid only when the selected watch-expression value is 4-byte data.
Dbl	Displays the value of the selected item in Double. Note that this item becomes valid only when the selected watch-expression value is 8-byte data.
Hex	Adds the value in hexadecimal number enclosing with "()" at the end of the value of the selected item (except the item displayed in hexadecimal number).

(6) Modify the contents of watch-expressions

The watch-expressions values can be edited.

Double-click the value of the watch-expression to edit in the [Value] area to switch the value to edit mode (press the [Esc] key to cancel the edit mode).

To write the edited value to the target memory, directly enter the value from the keyboard then press the [Enter] key.

Note that only those values correspond one by one to variables of C language, CPU registers, I/O register or assembler symbols can be edited. In addition, read-only I/O register values cannot be edited.

Remark   This operation can be taken place while the program is in execution. See "(4) Display/modify the memory contents during program execution" for details on how to operate it.

You can use ASCII characters also to the values of the watch-expression.

Examples 1.   Entering via an ASCII character

In the [Value] area for the variable "ch", enter "'A'"

>> "0x41" will be written to the memory area allocated to "ch"

2.   Entering via a numeric value

In the [Value] area for the variable "ch", enter "0x41"

>> "0x41" will be written to the memory area allocated to "ch"

3.   Entering via an ASCII string

Set the display format of character array "str" to ASCII, and in the [Value] area, enter ""ABC""

>> "0x41, 0x42, 0x43, 0x00" will be written to the memory area allocated to "str"

Remarks 1.   If a number with fewer digits than the size of the variable is entered, the higher-order digits will be padded with zeroes.

2.   If a number with more digits than the size of the variable is entered, the higher-order digits will be masked.

3.   If the display format of a character array (type char or unsigned char) is set to ASCII, then the value can also be entered as a string (ASCII/Shift_JIS/EUC-JP/Unicode (UTF-8)).

(7) Display/modify the contents of watch-expressions during program execution

The Memory panel/Watch panel has the real-time display update function that can update/modify the display contents of the memory/watch-expression in real-time while executing the program.

Using the real-time display update function enables to display/modify the value of memory/watch-expression not only while the program is stopped, but also in execution.

See "(4)   Display/modify the memory contents during program execution" for details on how to operate it.

(8) Save the contents of watch-expressions

By selecting the [File] menu >> [Save Watch Data As...], the Save As dialog box can be opened, and all the contents of the watch-expression and its value can be saved in a text file (*.txt) or CSV file (*.csv).

When saving the contents to the file, all the values of the watch-expression are reacquired and save the latest values acquired.

If arrays, pointer type variables, structures/unions, and CPU registers (only those with the part name) are displayed expanded, the value of each expanded element is also saved. When they are not expanded, "+" mark is added on the top of the item and the value becomes blank.

Note that the values of read-protected I/O register are not re-read. If you want to save the latest values of those, select [Force Read Value] from the context menu then save the file.

Figure 2-147.   Output Example of Watch Data

```
Watch-expression     Value        Type(Byte Size)     Address     Memo
------------------------------------------------------------------
 Watch-expression     Value        Type(Byte Size)     Address     Memo
-Category name
  Watch-expression    Value        Type(Byte Size)     Address     Memo
     :                  :              :                 :           :
```

Remark   When the contents of the panel are overwritten with selecting the [File] menu >> [Save Watch Data], each watch panel (Watch1 to Watch4) is treated as a different panel.

2.10   Display Information on Function Call from Stack

This section describes how to show the information on function call from the stack.

The CubeSuite compiler (CA850/CX) pushes function-call information onto the stack, in accordance with the ANSI standard.

It is thus possible to learn the function call depth, the location of the caller, parameters, and other information by analyzing the function-call information.

This "function-call information" is called the call stack information; this term will be used in the rest of this document.

2.10.1   Display call stack information

Call stack information is displayed in the Call Stack panel below.

Select the [View] menu >> [Call Stack].

For details on the contents and function in each area, see the section for the Call Stack panel.

Caution   Nothing is displayed on this panel during execution of a program.

When the program is stopped, items in each area are displayed.

Figure 2-148.   Display Call Stack Information (Call Stack Panel)
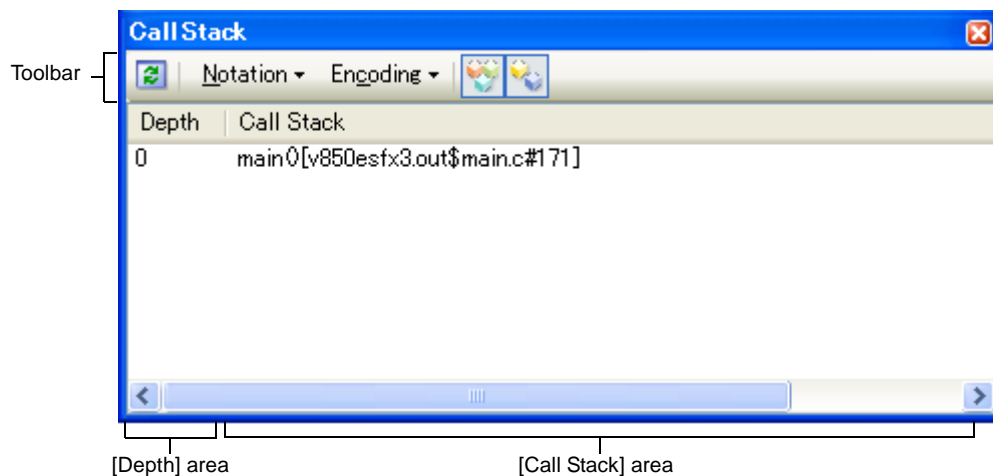
This section describes the following.

- (1) Change display format of values
- (2) Jump to the source line
- (3) Display local variables
- (4) Save the contents of call stack information

(1)Change display format of values

The display format of this panel can be changed using buttons below on the toolbar.

This item becomes invalid during execution of a program.

Notation	The following buttons to change the notation of a data value are displayed.
auto	Displays values on this panel in the default notation according to the type of variable (default).
Hex	Displays values on this panel in hexadecimal number.
Dec	Displays values on this panel in decimal number.
Oct	Displays values on this panel in octal number.
Bin	Displays values on this panel in binary number.
Encoding	The following buttons to change the encoding of character variables are displayed.
Asc	Displays character variables in ASCII code (default).
SJIS	Displays character variables in Shift_JIS code.
EUC	Displays character variables in EUC-JP code.
UTF	Displays character variables in UTF-8 code.

(2) Jump to the source line

The Editor panel is opened with moving the caret to the function call source line which the selected line designates when the line is double-clicked (if the panel is already opened, jump to the Editor panel).

Remark The Disassemble panel (Disassemble1) is opened with moving the caret to the function call source address which the currently selected line designates when selecting [Jump to Disassemble] from the context menu (if the panel is already opened, jump to the Disassemble panel (Disassemble1)).

(3) Display local variables

The Local Variables panel that displays the local variables which the currently selected line designates is opened when selecting [Jump to Local Variable at This Time] from the context menu.

(4) Save the contents of call stack information

By selecting the [File] menu >> [Save Call Stack Data As...], the Save As dialog box can be opened, and all the contents in the call stack information can be saved in a text file (*.txt) or CSV file (*.csv).

When saving to files, retrieve the latest information from the debug tool.

Figure 2-149. Output Example of Call Stack Information

```
Depth      Call stack
-----------------------------------------------------------------
 0         Call stack information
 1         Call stack information
 :                :
```

2.11   Collect Execution History of Programs [IECUBE][IECUBE2][Simulator]

This section describes how to collect the execution history of the program.

A history of program execution is generally called a trace; this term will be used in the remainder of this document.

It is nearly impossible to find the cause of runaway program execution from the memory contents, stack information, and the like after the runaway has occurred. The collected trace data, however, can be used to trace program execution up to the runaway directly, making this an effective tool for discovering hidden bugs.

> **Cautions 1.   [IECUBE]**
>
> **Some of the trace function, the real-time display update function (RRM function) and the coverage function are used on a mutually exclusive basis.**
>
> **2.   [IECUBE2]**
>
> **Some of the trace function, the timer function (except for the Run-Break timer) and the coverage function are used on a mutually exclusive basis.**
>
> **3.   [MINICUBE][MINICUBE2][E1]**
>
> **The trace function is not supported.**

2.11.1   Configure the trace operation

When tracing starts, the trace data that the execution process of the program currently in execution is recorded is collected (when the program is stopped, tracing automatically stops).

You need to configure the trace operation before using the trace function.

The setting method differs depending on the debug tool used.

(1) [IECUBE]

(2) [IECUBE2]

(3) [Simulator]

(1) [IECUBE]

This trace operation can be configured in the [Trace] category in the [Debug Tool Settings] tab in the Property panel.

Figure 2-150.   [Trace] Category [IECUBE]

(a) [Use for trace data]

Some of the trace function, the real-time display update function (RRM function) and the coverage function are used on a mutually exclusive basis. Accordingly, you need to specify a function used preferentially. Select [Trace] from the following drop-down list.

Trace	Uses the trace function preferentially.
	- Real-time display update function (RRM function)
	Cannot be used
	- Coverage function
	Cannot be used
RRM	Uses the real-time display update function (RRM function) preferentially (default).
	- Trace function
	Only trace branch instructions internally (the behavior is the same as when the [Select trace data] property is set to [Branch PC]). Trace-related events are also disabled (invalid).
	- Coverage function
	Cannot be used
Coverage	Uses the coverage function preferentially.
	- Trace function
	Only trace instructions internally (the behavior is the same as when the [Select trace data] property is set to [Branch PC]).
	Trace-related events are also disabled (invalid).
	- Coverage function
	Cannot be used

(b) [Select trace data]

This property appears only when the [Use for trace data] property is set to [Trace].

Specify a type of trace data to be collected from the following drop-down list.

Branch PC
All PC
Access Data
Branch PC + Access Data (default)
All PC + Access Data
Access PC + Access Data
Branch PC + Access PC + Access Data

Remarks 1.   Meaning of the type of trace data to be collected is as follows:

Branch PC	PC values of branch origin and branch destination instructions
All PC	PC values of all instructions
Access Data	Access address and access data
Access PC	PC values of instructions that caused access

2.   The DMA start point and end point are traced regardless of the trace condition.

(c) [Trace priority]

This property is valid only when the [Use for trace data] property is set to [Yes].

Specify which item should be given priority when using the trace function from the following drop-down list.

Speed priority	Traces giving priority to the real-time performance (default).
Data priority	Traces after stopping the execution pipeline of the CPU temporarily so that no data is missed.

(d) [Clear trace memory before running]

Specify from the drop-down list whether to clear (initialize) the trace memory before tracing starts in this property. Select [Yes] to clear the memory (default).

Remark   You can forcibly clear the trace memory when clicking the ☐ button in the toolbar in the Trace panel [IECUBE][IECUBE2][Simulator].

(e) [Operation after trace memory is full]

Specify the operation after the trace memory is full with the collected trace data from the following drop-down list.

Non stop and overwrite to trace memory	Continues overwriting the older trace data after the trace memory is full (default). When the [Clear trace memory before running] property is set to [Yes], at the time of a resumption, the trace data is collected after clearing the trace memory.
Stop trace	When the trace memory is full, the trace data stops writing (the program does not stop execution). When the [Clear trace memory before running] property is set to [No], the trace data is not collected even if the program is executed again.
Stop	When the trace memory is full, the trace data stops writing and the program stops execution. When the [Clear trace memory before running] property is set to [No], the program cannot be executed again even if it is executed.

(f) [Rate of frequency division of trace time tag]

Specify from the drop-down list the frequency division ratio of the counter to be used for time tag display (the [Time] item in the Trace panel [IECUBE][IECUBE2][Simulator]).

The drop-down list displays the following frequency division ratios (values in "()" indicate the resolution and the maximum measurement time when using a time tag counter of 32-bit and a external clock of 50 MHz).

1/1(20ns/1.4min)(default),	1/2(40ns/2.8min),	1/4(80ns/5.7min),
1/8(160ns/11.4min),	1/16(320ns/22.8min),	1/32(640ns/45.6min),
1/64(1280ns/1.5h),	1/128(2560ns/3.0h),	1/256(5120ns/6.0h),
1/512(10240ns/12.2h) ,	1/1024(20480ns/24.4h),	1/2048(40960ns/48.8h),
1/4096(81920ns/97.6h)		

(g) [Trace memory size[frames]]

Specify from the drop-down list the trace memory size (trace frame number) in this property.

The trace frame is a unit of the trace data. One trace frame is used for each operation in fetch/write/read.

The drop-down list displays the following trace frame numbers.

8K (default), 32K, 64K, 128K, 256K

(h) [Enable trace data complement]

Specify whether to perform complementary display when displaying the collected trace data in the Trace panel [IECUBE][IECUBE2][Simulator].

Complement display of instructions between branch instructions that cannot be traced by hardware can be performed.

Select [Yes] to perform complementary display (default).

Note that the trace memory is cleared automatically if you change the setting of this property.

(2)　[IECUBE2]

This trace operation can be configured in the [Trace] category in the [Debug Tool Settings] tab in the Property panel.

Figure 2-151.　[Trace] Category [IECUBE2]

(a)　[Use for trace data]

Some of the trace function, the timer function (except for the Run-Break timer) and the coverage function are used on a mutually exclusive basis. Accordingly, you need to specify a function used preferentially. Select [Trace] from the following drop-down list.

Trace	Uses the trace function preferentially. 　- Timer function 　　Cannot be used. Timer-related events (except for the Run-Break timer) are also disabled (invalid). 　- Coverage function 　　Cannot be used
Timer	Uses the timer function preferentially. 　- Trace function 　　Only trace branch instructions internally (the behavior is the same as when the [Select trace data] property is set to [Branch PC]). Trace-related events are also disabled (invalid). 　- Coverage function 　　Cannot be used
Coverage	Uses the coverage function preferentially. 　- Trace function 　　Only trace branch instructions internally (the behavior is the same as when the [Select trace data] property is set to [Branch PC]). Trace-related events are also disabled (invalid). 　- Timer function 　　Cannot be used. Timer-related events (except for the Run-Break timer) are also disabled (invalid).

(b)　[Select trace data]

This property appears only when the [Use for trace data] property is set to [Trace].

Specify a type of trace data to be collected from the following drop-down list.

Branch PC
Access Data

Branch PC + Access Data (default)
Access PC + Access Data
Branch PC + Access PC + Access Data

Remark   Meaning of the type of trace data to be collected is as follows:

Branch PC	PC values of branch origin and branch destination instructions
Access Data	Access address and access data
Access PC	PC values of instructions that caused access

(c) [DMA is traced]

Specify whether to trace the access data via DMA from the following drop-down list.

Select [Yes] to perform the DMA trace ([No] is selected by default).

(d) [Select DMA trace data]

This property appears only when the [DMA is traced] property is set to [Trace].

Specify a type of DMA trace data to be collected from the following drop-down list.

Access Type + Access Address + Data Size + Access Data
Access Type + Access Address + Data Size
Access Type + Channel Number
Access Type + Channel Number + Transfer Count
Access Type + Access Data + Channel Number + Transfer Count

Remark   Meaning of the type of DMA trace data to be collected is as follows:

Access Type	Type of the memory access (Read, Write, Read/Write)
Access Address	Target address of the memory access
Data Size	Access size
Access Data	Access data
Channel Number	DMA channel number
Transfer Count	DMA transfer count

(e) [Trace priority]

This property is valid only when the [Use for trace data] property is set to [Yes].

Specify which item should be given priority when using the trace function from the following drop-down list.

Speed priority	Traces giving priority to the real-time performance (default).
Data priority	Traces after stopping the execution pipeline of the CPU temporarily so that no data is missed.

(f) [Clear trace memory before running]

Specify from the drop-down list whether to clear (initialize) the trace memory before tracing starts in this property.

Select [Yes] to clear the memory (default).

Remark You can forcibly clear the trace memory when clicking the [ ] button in the toolbar in the Trace panel [IECUBE][IECUBE2][Simulator].

(g) [Operation after trace memory is full]
Specify the operation after the trace memory is full with the collected trace data from the following drop-down list.

Non stop and overwrite to trace memory	Continues overwriting the older trace data after the trace memory is full (default). When the [Clear trace memory before running] property is set to [Yes], at the time of a resumption, the trace data is collected after clearing the trace memory.
Stop trace	When the trace memory is full, the trace data stops writing (the program does not stop execution). When the [Clear trace memory before running] property is set to [No], the trace data is not collected even if the program is executed again.
Stop	When the trace memory is full, the trace data stops writing and the program stops execution. When the [Clear trace memory before running] property is set to [No], the program cannot be executed again even if it is executed.

(h) [Trace memory size[frames]]
Specify from the drop-down list the trace memory size (trace frame number) in this property.
The trace frame is a unit of the trace data. One trace frame is used for each operation in fetch/write/read.
The drop-down list displays the following trace frame numbers.
    8K (default), 32K, 64K, 128K, 256K

(i) [Enable trace data complement]
Specify whether to perform complementary display when displaying the collected trace data in the Trace panel [IECUBE][IECUBE2][Simulator].
Complement display of instructions between branch instructions that cannot be traced by hardware can be performed.
Select [Yes] to perform complementary display (default).
Note that the trace memory is cleared automatically if you change the setting of this property.

(3) [Simulator]
This trace operation can be configured in the [Trace] category in the [Debug Tool Settings] tab in the Property panel.

Figure 2-152.   [Trace] Category [Simulator]

(a) [Use trace function]
Specify whether to use trace function from the drop-down list in this property.
Select [Yes] to use the trace function ([No] is selected by default).

(b) [Clear trace memory before running]

Specify from the drop-down list whether to clear (initialize) the trace memory before tracing starts in this property. Select [Yes] to clear the memory (default).

Remark   You can forcibly clear the trace memory when clicking the 🪣 button in the toolbar in the Trace panel [IECUBE][IECUBE2][Simulator].

(c) [Operation after trace memory is full]

Specify the operation after the trace memory is full with the collected trace data from the following drop-down list.

Non stop and overwrite to trace memory	Continues overwriting the older trace data after the trace memory is full (default). When the [Clear trace memory before running] property is set to [Yes], at the time of a resumption, the trace data is collected after clearing the trace memory.
Stop trace	When the trace memory is full, the trace data stops writing (the program does not stop execution). When the [Clear trace memory before running] property is set to [No], the trace data is not collected even if the program is executed again.
Stop	When the trace memory is full, the trace data stops writing and the program stops execution. When the [Clear trace memory before running] property is set to [No], the program cannot be executed again even if it is executed.

(d) [Accumulate trace time]

Specify from the drop-down list whether to display the trace time with accumulated time in this property. Specify [Yew] to display trace time with accumulated time. Specify [No] to display the trace time with differential time (default).

(e) [Trace memory size[frames]]

Specify from the drop-down list the trace memory size (trace frame number) in this property.
The trace frame is a unit of the trace data. One trace frame is used for each operation in fetch/write/read.
Drop down list includes the following trace frame numbers.

    4K (default), 8K, 12K, 16K, 20K, 24K, 28K, 32K, 36K, 40K, 44K, 48K, 52K, 56K, 60K, 64K, 128K 192K, 256K, 320K, 384K, 448K, 512K, 576K, 640K, 704K, 768K, 832K, 896K, 960K, 1M, 2M, 3M

(f) [Rate of frequency division of trace time tag]

Specify from the drop-down list the frequency division ratio of the counter to be used for time tag display (the [Time] item in the Trace panel [IECUBE][IECUBE2][Simulator]).
The drop-down list displays the following frequency division ratio.

    1/1 (default), 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256, 1/512, 1/1K, 1/4K, 1/8K, 1/16K, 1/64K, 1/256K, 1/1M

2.11.2   Collect execution history until stop of the execution

In the debug tool, there is a function to collect the execution history from the start of program execution to the stop.
Therefore, the trace data collection is automatically started when the program starts executing and stopped when the program stops.

See "2.11.5 Display the collected execution history" for how to check the collected trace data.

Remark   This function is operated by the Unconditional Trace event, which is one of the built-in events set by default in the debug tool.

Consequently, if the Unconditional Trace event is set to Invalid state by clearing the check box in the Events panel, trace data linked to the start of program execution will not be collected (the Unconditional Trace event is set to Valid state by default).

Note that the Unconditional Trace event and the Trace event (see "2.11.3   Collect execution history in the arbitrary section" below) are events that are used on a mutually exclusive basis. Therefore, if the Trace event with Valid state is set, the Unconditional Trace event is automatically set to Invalid state.

2.11.3   Collect execution history in the arbitrary section

The execution history is collected as trace data only for the arbitrary section while the program is in execution by setting a Trace event (Start Tracing event/Stop Tracing event).

To use this function, follow the steps below.

Cautions 1.   Also see "**2.15.6   Notes for setting events**" **for details on Trace events (e.g. limits on the number of enabled events).**

2.   **[IECUBE][IECUBE2]**
If the [Use trace function] property in the [Trace] category on the [Debug Tool Settings] tab of the Property panel is set to other than [Trace], all of events related to trace that have currently been set become Invalid state.

3.   **[Simulator]**
Start Tracing events and Stop Tracing events cannot be set while a tracer is running.

(1)   Set a Start Tracing event/Stop Tracing event

Set a Start Tracing event/Stop Tracing event that starts/stops collecting the trace data in the Editor panel/ Disassemble panel.

(a)   How to set a Start Tracing event

Move the caret to the line/address[Note] to start collecting trace data then select [Trace Settings] >> [Start Tracing] from the context menu.

A Start Tracing event is set to the instruction at the start address of the line/address where the caret exists.

(b)   How to set a Stop Tracing event

Move the caret to the line/address[Note] to stop collecting trace data then select [Trace Settings] >> [Stop Tracing] from the context menu.

A Stop Tracing event is set to the instruction at the start address of the line/address where the caret exists.

Note   Start Tracing events/Stop Tracing events cannot be set at lines in the event area with the background color in gray (showing to the corresponding address of the line that no change is allowed).

Once a Start Tracing event/Stop Tracing event is set, the following event mark is displayed in the event area of the line/address that an event is set. Start Tracing events/Stop Tracing events are managed as a single event (the Trace event) in the Events panel (details about set Start Tracing events/Stop Tracing events can be checked by clicking the plus mark "+" next to the Trace event item).

Table 2-14.   Event Marks of the Start Tracing Event/Stop Tracing Event

Event Type	Event mark
Start Tracing	🏁
Stop Tracing	🏁

Figure 2-153.   Start Tracing Event/Stop Tracing Event Setting Example
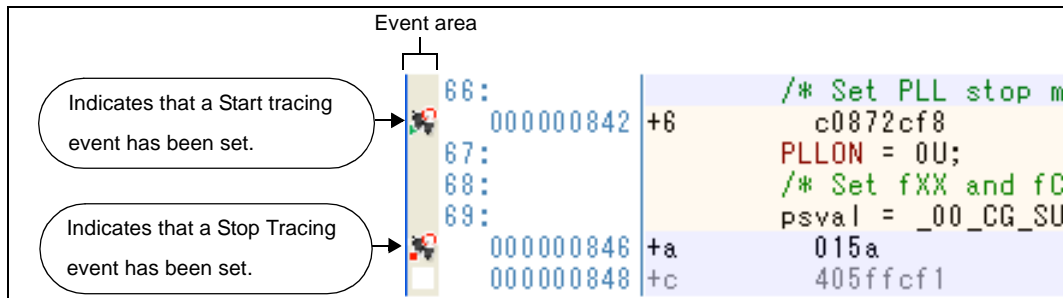
Remarks 1. When either of the Start Tracing event/Stop Tracing event is set as Valid state, the check box of the
Unconditional Trace event in the Events panel is automatically cleared, therefore, the trace data
collection does not automatically start with the start of the program execution (the tracer will not
move until the condition of the Start Tracing event that has been set is met).

   2. A Stop Tracing event is not indispensable for A Trace event.

   3. Event marks differ depending on the event state (see "2.15.1   Change the state of set events
(valid/invalid)").
   When an event is set at the point which other event is already set, the event mark (📁) is
displayed meaning more than one event is set at the point.

   4. **[Simulator]**
   When either of the Start Tracing event/Stop Tracing event is set to Valid state, the [Use trace
function] property in the [Trace] category on the [Debug Tool Settings] tab of the Property panel is
automatically set to [Yes] and the trace function will be enabled.

(2) Execute the program

Execute the program (see "2.7   Execute Programs").
If the instruction with the Start Tracing event/Stop Tracing event executes, the trace data collection starts/stops.
See "2.11.5   Display the collected execution history" for how to check the collected trace data.

(3) Delete the Start Tracing event/Stop Tracing event

To delete the set Start Tracing event/Stop Tracing event, select [Delete Event] from the context menu that appears
by right clicking on the event mark in the event area.

2.11.4   Collect execution history when the condition is met

The program execution history is collected when a condition is met.

(1) When an access to variables/I/O registers occurs

Set the Point Trace event to collect information about accesses to specified variables or I/O register as trace data.
Set a Point Trace event by one of the following operation.

Caution   Also see "2.15.6   Notes for setting events"for details on Point trace events (e.g. limits on the
number of enabled events).

Remark   [IECUBE2]
Accesses via DMA can be traced.

(a) When the access is to the variable/I/O register on the source text/disassembled text
Perform this operation in the Editor panel/Disassemble panel in which the source text/disassembly text is displayed.
Follow the operation listed below from the context menu after selecting the variable/I/O register.
Note, however, that only global variables, static variables inside functions, and file-internal static variables can be used.
By performing the following operation, it is interpreted as if a Point Trace event has been set at the target variable/I/O register, and it is managed in the Events panel (see "2.15   Manage Events" for details).

Access Type	Operation
Read/Write	Select [Trace Settings] >> [Record R/W Value].

Remark   Variables within the current scope can be specified.

(b) When the access is to the registered watch-expression
Follow the steps below in the Watch panel.
Follow the operation listed below from the context menu after selecting the registered watch-expression (see "2.9.6   Display/change watch-expressions").
Note, however, that only global variables, static variables inside functions, file-internal static variables, and I/O register can be used.
By performing the following operation, it is interpreted as if a Point Trace event has been set at the target watch-expression, and it is managed in the Events panel (see "2.15   Manage Events" for details).

Access Type	Operation
Read	Select [Trace Output] >> [Record Reading Value].
Write	Select [Trace Output] >> [Record Writing Value].
Read/Write	Select [Trace Output] >> [Record R/W Value].

Remark   A watch-expression within the current scope can be specified.
To target a watch-expression outside the current scope, select a watch-expression with a specified scope.

After the setting of the Point Trace event is complete, execute the program (see "2.7   Execute Programs").
If the conditions for a Point Trace event that you have set are met while the program is executing, that information is collected as trace data.
See "2.11.5   Display the collected execution history" for details on checking trace data.
To delete a Point Trace event, select the event name to delete in the Events panel opened when selecting the [View] menu >> [Event], then press the ✖ button in the toolbar in the same panel (see "2.15   Manage Events").

Figure 2-154.   Example of Point Trace Event Results View (When Using Simulator)
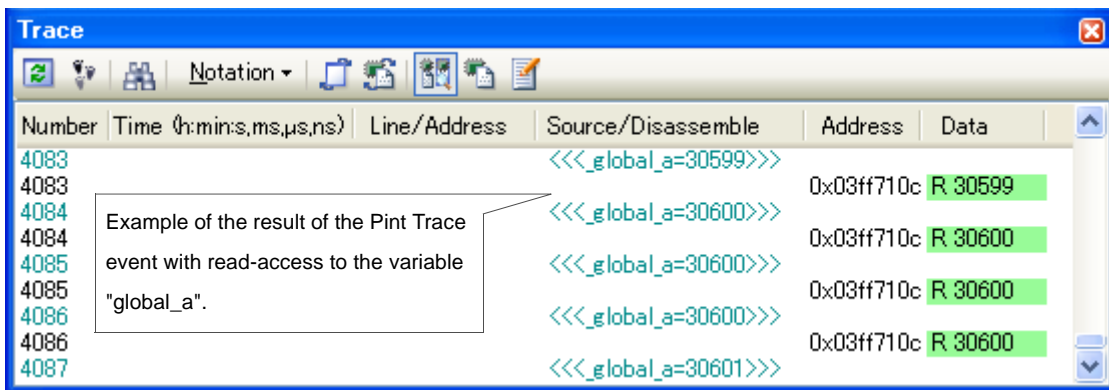
2.11.5   Display the collected execution history

The collected trace data is displayed in the Trace panel [IECUBE][IECUBE2][Simulator] below.

Select the [View] menu >> [Trace].

The trace data displays by mixing the disassembled text and source text by default, but it is also possible to display either one of these by selecting the Display mode.

For details on the contents and function in each area, see the section for the Trace panel [IECUBE][IECUBE2][Simulator].

Figure 2-155.   Display the Trace Data (Trace Panel)
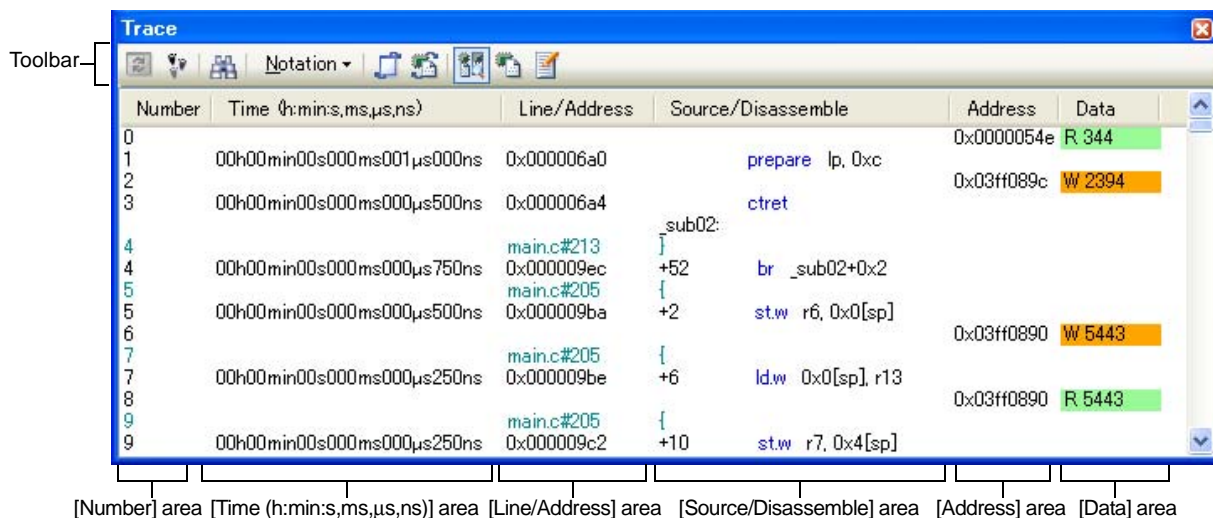
This section describes the following.

(1)   Change display mode
(2)   Change display format of values
(3)   Link with other panels

(1) Change display mode

Display mode can be changed to the purpose when clicking the buttons below in the toolbar.

These buttons become invalid during trace operation.

Table 2-15.   Display Modes of the Trace Panel

Button	Display Mode	Display Contents
	Mixed display mode	Displays the instruction (disassemble results), labels, source text (corresponding source line), point trace results, and break causes (default).
	Disassemble display mode	Displays the instruction (disassemble results), labels, point trace results, and break causes.
	Source display mode	Displays the source text (corresponding source line), and break causes. However, when a place where no debugging information is present is executed, "<No Debug Information>" is displayed.

Figure 2-156.   Example of Source Display Mode View (Trace Panel)
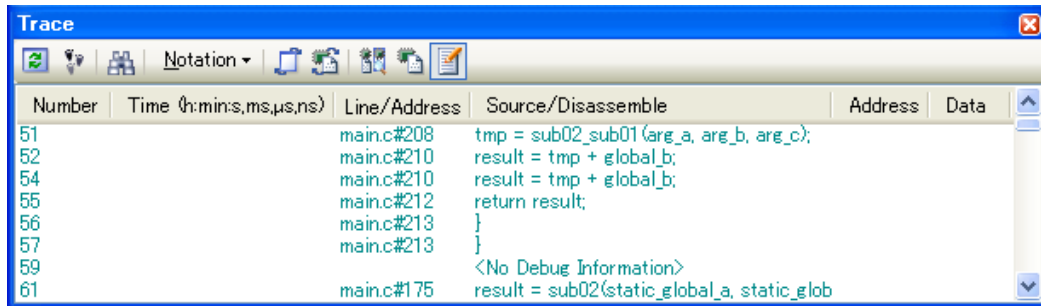
(2) Change display format of values

The display format of the [Line Number/Address]/[Address]/[Data] area can be changed using buttons below on the toolbar.

Note that the button is invalid while program is executed.

Notation	The following buttons to change the notation of a data value are displayed.
	Displays values on this panel in hexadecimal number (default).
	Displays values on this panel in decimal number.
	Displays values on this panel in octal number.
	Displays values on this panel in binary number.

(3) Link with other panels

Items in the trace panel can be linked to other panels using the currently selected line address as a pointer (window focus will not move).

Click the button on the toolbar to start linking to the Editor panel. Click the button on the toolbar to start linking to the Disassemble panel.

If the button is clicked again, the link is disconnected.

Remark   The Editor panel/Disassemble panel opens when selecting the [Jump to Source]/[Jump to Disassemble] from the context menu with moving the caret to the source line/address corresponding to the address of the currently selected line (focus is moved).

2.11.6   Clear the trace memory

To clear the collected trace data contents, click the [button] button on the toolbar.
Note that this button becomes invalid during trace operation.

Remark   When [Yes] is specified in the [Clear trace memory before running] property in the [Trace] category in the
[Debug Tool Settings] tab of the Property panel, the trace memory is cleared every time the program
executes.

2.11.7   Search the trace data

To search the collected trace data, click the [button] button to open the Trace Search dialog box
[IECUBE][IECUBE2][Simulator] (note that the search is disabled during the program is executing).
In this dialog box, follow the steps below.
When the tab on the tab selection area is selected, the trace data can be searched in instruction level/source level.
Note however, that if you search the trace data at the instruction level, the display mode must be set in the Trace panel
[IECUBE][IECUBE2][Simulator] to the Mixed display mode or Disassemble display mode.
When searching at the source level, the mode must be set to the Mixed display mode or Source display mode.

Figure 2-157.   Search the Trace Data (Trace Search Dialog Box)
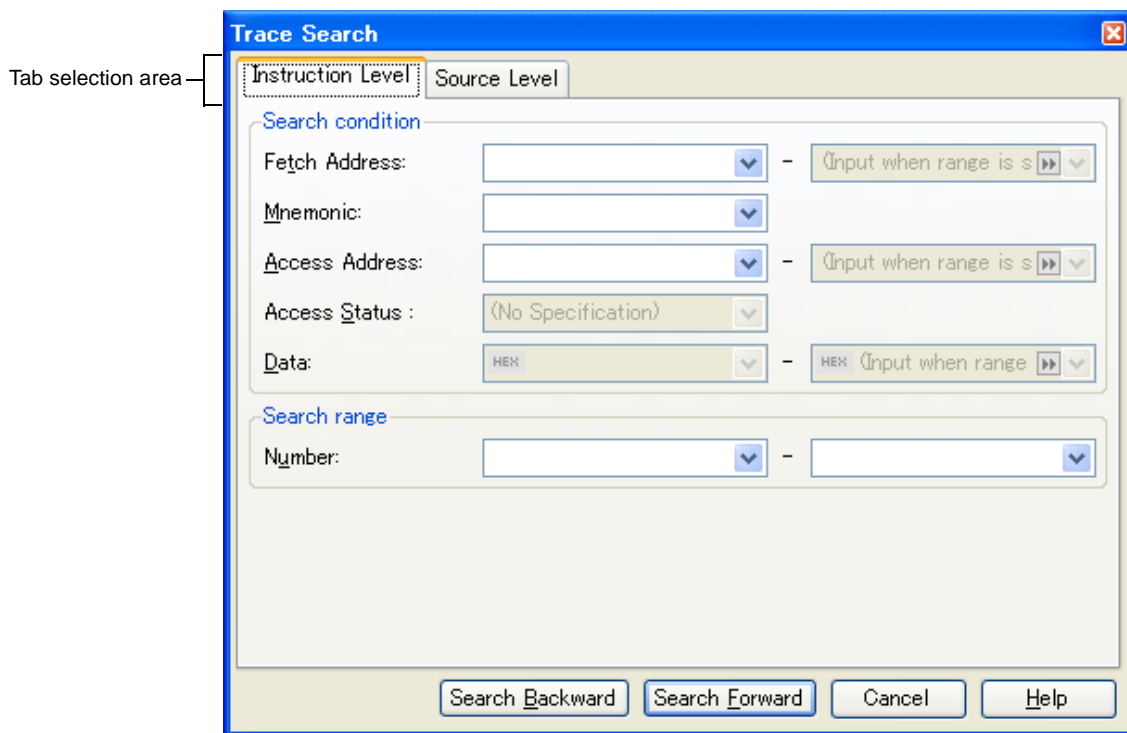
This section describes the following.

(1)   Search in the instruction level
(2)   Search in the source level

(1) Search in the instruction level
    Search the trace data in the instruction level.
    Select the [Instruction Level] tab and then follow the steps below.

Figure 2-158.   Search the Trace Data in the Instruction Level

(a) Specify [Fetch Address]
    Specify the fetch address if it is a required search parameter.
    You can either type the address expression directly into the text boxes, or select it from the input history via the
    drop-down list (up to 10 items).
    The fetch address can also be specified as a range. In this case, specify a range by specifying address
    expressions in both the left and right text boxes.
    If the right-hand text box is blank or contains the text [(Input when range is specified)], then the fixed address
    specified in the left-hand text box will be searched.
    Note that if an address value greater than the microcontroller address space is specified, the upper address
    value is masked.
    An address value greater than the value expressed within 32 bits cannot be specified.

(b) Specify [Mnemonic]
    Specify the mnemonic if it is a required search parameter.
    The specified character strings in this area is searched within the [Source/Disassemble] area of the Trace
    panel [IECUBE][IECUBE2][Simulator].
    You can either type a mnemonic directly into the text boxes, or select one from the input history via the drop-
    down list (up to 10 items).
    Searches are case-insensitive, and partial matches are also allowed.

(c) Specify [Access Address]

Specify the access address if it is a required search parameter.

You can either type the address value directly into the text boxes (in hexadecimal number), or select it from the input history via the drop-down list (up to 10 items).

The access address can also be specified as a range. In this case, specify a range by specifying address expressions in both the left and right text boxes.

If the right-hand text box is blank or contains the text [(Input when range is specified)], then the fixed address specified in the left-hand text box will be searched.

Note that if an address value greater than the microcontroller address space is specified, the upper address value is masked.

An address value greater than the value expressed within 32 bits cannot be specified.

(d) Specify [Access Status]

This item is only enable if a value for Specify [Access Address] is specified.

Select the access type (Read/Write, Read, Write, Vector Read and DMA) from drop-down list.

Select [(No Specification)] if you do not wish to limit access types.

(e) Specify [Data]

This item is only enable if a value for Specify [Access Address] is specified.

Specify the access data.

You can either type the data directly into the text boxes (in hexadecimal number), or select it from the input history via the drop-down list (up to 10 items).

The data can also be specified as a range. In this case, specify a range by specifying data in both the left and right text boxes.

If the right-hand text box is blank or contains the text [(Input when range is specified)], then the fixed data specified in the left-hand text box will be searched.

(f) Specify [Number]

Specify the range within the trace data to search via the number displayed in the [Number] area of the Trace panel [IECUBE][IECUBE2][Simulator].

Specify the starting number in the left text box, and the ending number in the right text box ("0" to "*last number*" are specified by default).

You can either type the numbers directly into the text boxes (in base-10 format), or select them from the input history via the drop-down list (up to 10 items).

If the left-hand text box is left blank, it is treated as if "0" were specified.

If the right-hand text box is left blank, it is treated as if the last number were specified.

(g) Click the [Search Backward]/[Search Forward] button

When the [Search Backward] button is clicked, search is taken place in the order from the large number to small and the search results are shown selected in the Trace panel [IECUBE][IECUBE2][Simulator].

When the [Search Backward] button is clicked, search is taken place in the order from the small number to large and the search results are shown selected in the Trace panel [IECUBE][IECUBE2][Simulator].

(2) Search in the source level

Search the trace data in the source level.

Select the [Source Level] tab.

Figure 2-159.   Search the Trace Data in the Source Level
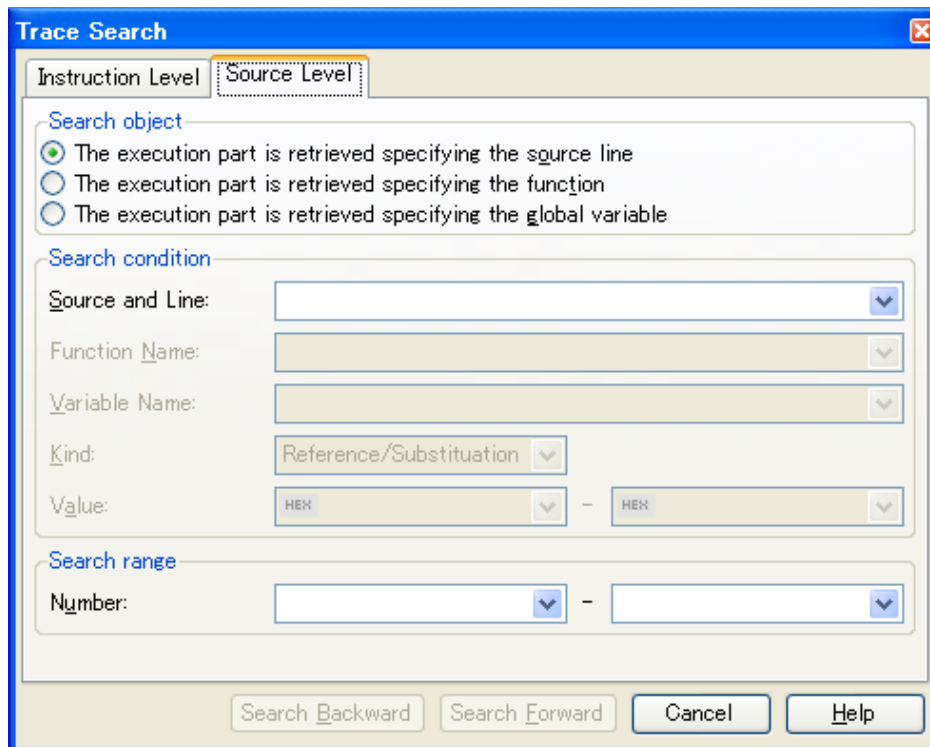
(a) Search with specifying the source line (default)

Select the [The execution part is retrieved specifying the source line] item in the [Search object] area and then follow the operation below.

<1> Specify [Source and Line]

The specified character strings in this area is searched within the [Line/Address] area of the Trace panel [IECUBE][IECUBE2][Simulator].

You can either type the character strings of the source line to be find directly into the text box, or select them from the input history via the drop-down list (up to 10 items).

Searches are case-insensitive, and partial matches are also allowed.

Examples 1.   main.c#40

2.   main.c

3.   main

<2> Specify [Number]

Specify the range within the trace data to search via the number displayed in the [Number] area of the Trace panel [IECUBE][IECUBE2][Simulator].

Specify the starting number in the left text box, and the ending number in the right text box ("0" to *last number*" are specified by default).

You can either type the numbers directly into the text boxes (in base-10 format), or select them from the input history via the drop-down list (up to 10 items).

If the left-hand text box is left blank, it is treated as if "0" were specified.

If the right-hand text box is left blank, it is treated as if the last number were specified.

<3>   Click the [Search Backward]/[Search Forward] button

When the [Search Backward] button is clicked, search is taken place in the order from the large number to small and the search results are shown selected in the Trace panel [IECUBE][IECUBE2][Simulator].

When the [Search Backward] button is clicked, search is taken place in the order from the small number to large and the search results are shown selected in the Trace panel [IECUBE][IECUBE2][Simulator].

(b)   Search with specifying the function name

Select the [The execution part is retrieved specifying the function] item in the [Search object] area and then follow the operation below.

<1>   Specify [Function Name]

You can either type the function name to be find directly into the text box, or select it from the input history via the drop-down list (up to 10 items).

Searches are case-insensitive, and only complete matches are retrieved.

<2>   Specify [Number]

Specify the range within the trace data to search via the number displayed in the [Number] area of the Trace panel [IECUBE][IECUBE2][Simulator].

Specify the starting number in the left text box, and the ending number in the right text box ("0" to *last number*" are specified by default).

You can either type the numbers directly into the text boxes (in base-10 format), or select them from the input history via the drop-down list (up to 10 items).

If the left-hand text box is left blank, it is treated as if "0" were specified.

If the right-hand text box is left blank, it is treated as if the last number were specified.

<3>   Click the [Search Backward]/[Search Forward] button

When the [Search Backward] button is clicked, search is taken place in the order from the large number to small and the search results are shown selected in the Trace panel [IECUBE][IECUBE2][Simulator].

When the [Search Backward] button is clicked, search is taken place in the order from the small number to large and the search results are shown selected in the Trace panel [IECUBE][IECUBE2][Simulator].

(c)   Search with specifying the global variable

Select the [The execution part is retrieved specifying the global variable] item in the [Search object] area and then follow the operation below.

<1>   Specify [Variable Name]

You can either type the variable name to be find directly into the text box, or select it from the input history via the drop-down list (up to 10 items).

Searches are case-insensitive, and only complete matches are retrieved.

<2>   Specify [Kind]

Select the access type ([Reference/Substitution], [Reference], or [Substitution]) from the drop-down list.

<3>   Specify [Value]

You can either type the accessed variable value directly into the text box, or select one from the input history via the drop-down list (up to 10 items).

The variable value can also be specified as a range. In this case, specify a range by specifying variable values in both the left and right text boxes.

If the right-hand text box is blank, then access locations with the fixed variable values specified in the left-hand text box will be searched for.

<4> Specify [Number]

Specify the range within the trace data to search via the number displayed in the [Number] area of the Trace panel [IECUBE][IECUBE2][Simulator].

Specify the starting number in the left text box, and the ending number in the right text box ("0" to "*last number*" are specified by default).

You can either type the numbers directly into the text boxes (in base-10 format), or select them from the input history via the drop-down list (up to 10 items).

If the left-hand text box is left blank, it is treated as if "0" were specified.

If the right-hand text box is left blank, it is treated as if the last number were specified.

<5> Click the [Search Backward]/[Search Forward] button

When the [Search Backward] button is clicked, search is taken place in the order from the large number to small and the search results are shown selected in the Trace panel [IECUBE][IECUBE2][Simulator].

When the [Search Backward] button is clicked, search is taken place in the order from the small number to large and the search results are shown selected in the Trace panel [IECUBE][IECUBE2][Simulator].

2.11.8   Save the contents of execution history

Contents of the collected trace data can be saved with range selection in text files (*.txt)/CSV files (*.csv).

When saving to the file, the latest information is acquired from the debug tool, and it is saved according with the display format on this panel.

The following Data Save dialog box can be opened by selecting the [File] menu >> [Save Trace Data As...].

In this dialog box, follow the steps below.

Figure 2-160.   Save Execution History (Data Save Dialog Box)

(1) Specify [File Name]

Specify the name of the file to save.

You can either type a filename directly into the text box (up to 259 characters), or select one from the input history via the drop-down list (up to 10 items).

You can also specify the file by clicking the [...] button, and selecting a file via the Select Data Save File dialog box.

(2) Specify [File Type]

Select the format in which to save the file from the following drop-down list.

The following file formats can be selected.

List Item	Format
Text files (*.txt)	Text format (default)
CSV (Comma-Separated Variables)(*.csv)	CSV format[Note]

Note The data is saved with entries separated by commas (,).

If the data contains commas, each entry is surrounded by double quotes "" in order to avoid illegal formatting.

(3) Specify [Save Range Number]

Specify the range of the number to save via "start number" and "end number".

Directly enter decimal number in each text box or select from the input history displayed in the drop-down list (up to 10 items).

When saving all the trace data, select the [All Trace Data] item in the drop-down list at the left (the right text box becomes invalid).

If a range is selected in the panel, that range is specified as the default. If there is no selection, then the range currently visible in the panel is specified.

(4) Click the [Save] button

Trace data is saved in the specified file with the specified format.

Figure 2-161.   Output Example of the Trace Data

```
Number     Time     Line Number/Address     Source/Disassemble     Address    Data
--------------------------------------------------------------------------------
Number     Time     Line Number/Address     Source/Disassemble     Address    Data
   :         :               :                       :                 :         :
```

2.12   Measure Execution Time of Programs

This section describes how to measure the execution time of the program.

2.12.1   Measure execution time until stop of the execution

In the debug tool, there is a function to measure the program execution time (Run-Break time) from the start to the stop.
Therefore, when the program starts its execution, the execution time is automatically measured.
You can check the result of the measurement by either one of the following.

Caution   **[Simulator]**
              To use this function, [Yes] must be specified with the [Use timer function] property in the [Timer]
              category in the [Debug Tool Settings] tab of the Property panel.

Remark   This function is operated by the Run-Break Timer event, which is one of the built-in events set by default in
              the debug tool.
              The Run-Break Timer event is always set to Valid state (changes not allowed).

(1) Check in the status bar
     After the program is stopped, the result of the measurement is displayed in the status bar on the Main window
     (when measurements have not been performed yet, "Unmeasurement" is displayed).

Figure 2-162.   Example of the Result of the Run-Break Timer Event (Status Bar)
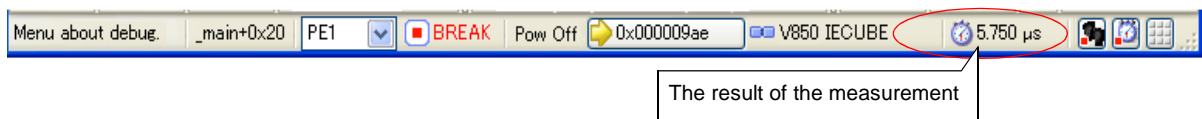
(2) Check on the Events panel
     After the program is stopped, the result of the measurement is displayed in the Events panel opened by selecting
     the [View] menu >> [Event], in event type as "Run-Break Timer".

Figure 2-163.   Example of the Result of the Run-Break Timer Event (Events Panel)
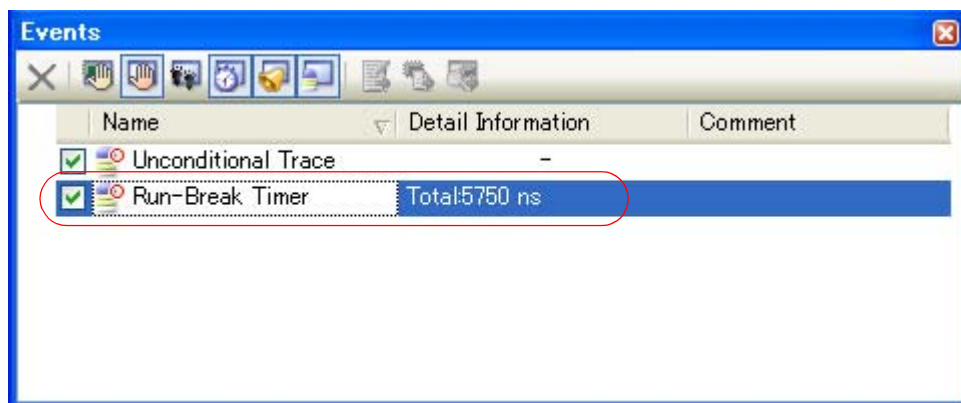
2.12.2   Measure execution time in the arbitrary section [IECUBE][IECUBE2][Simulator]

In the program execution process, the execution time in the arbitrary section can be measured by setting the Timer result event (Start Timer event/Stop Timer event).

To use this function, follow the steps below.

Cautions 1.   Also see "2.15.6   Notes for setting events" for details on Timer Result events (e.g. limits on the number of enabled events).

2.   [IECUBE2]
Some of the trace function, the timer function (except for the Run-Break timer) and the coverage function are used on a mutually exclusive basis. Therefore, to use the timer function, [Timer] must be specified with the [Use for trace data] property in the [Trace] category in the [Debug Tool Settings] tab of the Property panel.

3.   [MINICUBE][MINICUBE2][E1]
The Timer result event is not supported.

4.   [Simulator]
To use this function, [Yes] must be specified with the [Use timer function] property in the [Timer] category in the [Debug Tool Settings] tab of the Property panel.

(1) Set a Start Timer event/Stop Timer event

Set a Start Timer event/Stop Timer event in the Editor panel/Disassemble panel.

(a) How to set a Start Timer event

Move the caret to the line/address[Note] to start the timer then select [Timer Settings] >> [Start Timer] from the context menu. A Start Timer event is set to the instruction at the start address of the line/address where the caret exists.

(b) How to set a Stop Timer event

Move the caret to the line/address[Note] to stop the timer then select [Timer Settings] >> [Stop Timer] from the context menu. A Stop Timer event is set to the instruction at the start address of the line/address where the caret exists.

Note Start Timer events/Stop Timer events cannot be set at lines in the event area with the background color in gray (showing to the corresponding address of the line that no change is allowed).
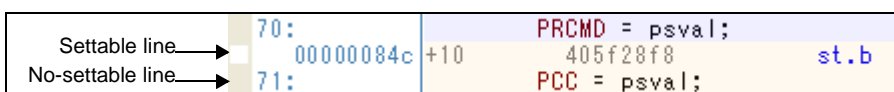
Once a Start Timer event/Stop Timer event is set, the following event marks are displayed in the event area of the line/address that an event is set. Start Timer events and Stop Timer events are managed as a single event (Timer Result event) in the Events panel (details about the set Start Timer event/Stop Timer event can be checked by clicking the plus mark "+" next to the Timer Result event item).

Table 2-16. Event Marks of the Start Timer Event/Stop Timer Event

Event Type	Event mark
Start Timer	
Stop Timer	

Figure 2-164.   Start Timer Event/Stop Timer Event Setting Example
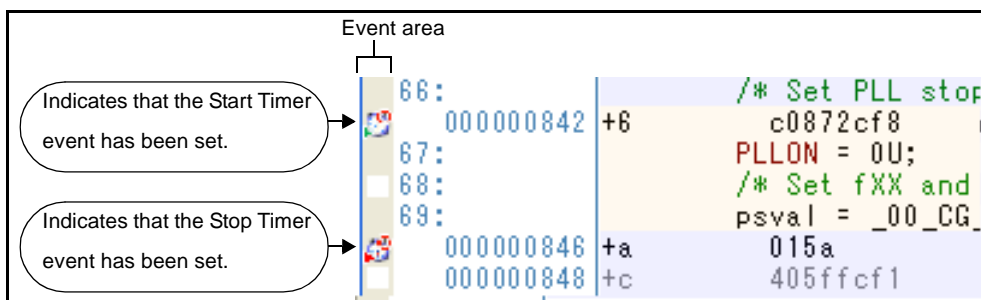
Remark   Event marks differ depending on the event state (see "2.15.1   Change the state of set events (valid/invalid)").

When an event is set at the point which other event is already set, the event mark (🗄) is displayed meaning more than one event is set at the point.

(2) Execute the program

Execute the program (see "2.7   Execute Programs").

If the instruction with the Start Timer event/Stop Timer event executes, the timer starts/stops.

After the program is stopped, the result of the measurement is displayed in the Events panel opened by selecting the [View] menu >> [Event], in event type as "Timer Result".

This "Timer Result" is the even type that is registered in the Events panel bundled in one event when both the Start Timer event and the Stop Timer event setting is completed.

Figure 2-165.   Timer Result Event (Start Timer Event/Stop Timer Event) Setting Example
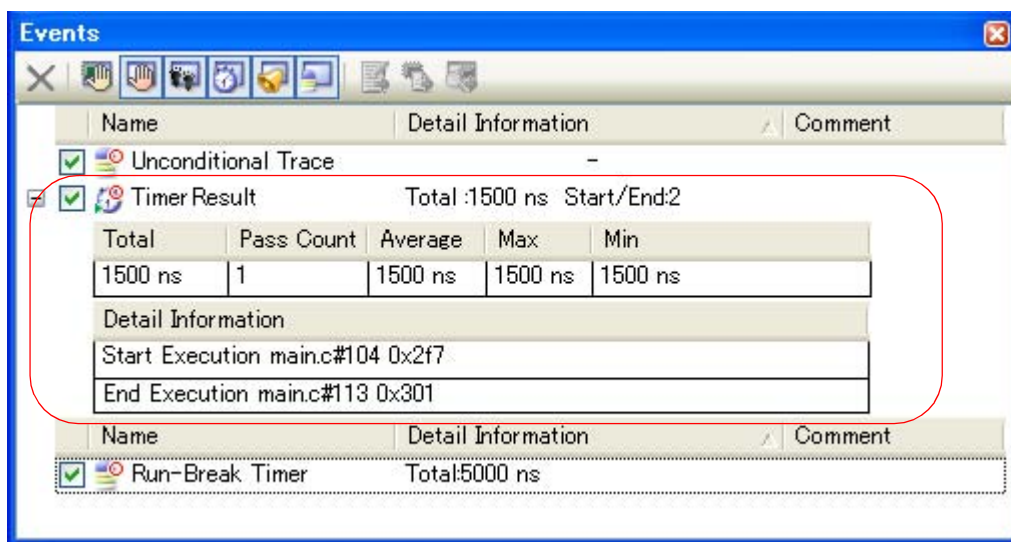
(3) Delete the Start Timer event/Stop Timer event

To delete the set Start Timer event/Stop Timer event, select [Delete Event] from the context menu that appears by right clicking on the event mark in the event area.

2.12.3   Measurable time ranges

The scope of time that can be measured via timers using Run-Break Timer events (see "2.12.1   Measure execution time until stop of the execution" for details) or Timer Result events (see "2.12.2   Measure execution time in the arbitrary section [IECUBE][IECUBE2][Simulator]" for details) is shown below.

If the maximum measurable length of time is exceeded, a timer-over break is generated, and program execution stops.

Table 2-17.   Measurable Time Ranges [V850E1][V850ES]

Debug Tool	Run-Break Timer Event		Timer Result Event	
IECUBE	Min.	20 ns	Min.	20 ns
	Max.	Approx. 195 hours 12 min. (time for 4K cycles) Overflow detection included	Max.	Approx. 195 hours 12 min. (time for 4K cycles) Maximum elapsed cycles: 65535 times Overflow detection included
MINICUBE E1 (JTAG)	Min.	200 ns		-
	Max.	Approx. 7 min. (time for TCK/DCK = 10 MHz) Overflow detection included		
MINICUBE2 E1 (Serial)	Min.	100 μs		-
	Max.	Approx. 119 hours 18 min. Overflow detection included		
Simulator	Depends on the clock frequency for timer/trace		Depends on the clock frequency for timer/trace	

Table 2-18.   Measurable Time Ranges [V850E2M]

Debug Tool	Run-Break Timer Event		Timer Result Event	
IECUBE2	Min.	5 ns	Min.	5 ns
	Max.	Approx. 24 hours (time for 4K cycles) Overflow detection included	Max.	Approx. 24 hours (time for 4K cycles) Elapsed cycles counter: 16-bit Overflow detection included
MINICUBE E1 (JTAG)	Min.	200 ns		-
	Max.	Approx. 7 min. (time for TCK/DCK = 10 MHz) Overflow detection included		
Simulator	Depends on the clock frequency for timer/trace		Depends on the clock frequency for timer/trace	

2.13   Measure Coverage [IECUBE][IECUBE2][Simulator]

This section describes the coverage measurement using the coverage function.

Although there are several types of coverage measurement, CubeSuite measures code coverage (C0 coverage) based on fetch-size for a source line/function and data coverage **[IECUBE2][Simulator]** based on access-size for a variable.

The coverage measurement area is as follows:

Table 2-19.   Coverage Measurement Area

Debug Tool		Code Coverage Measurement Area
V850E1 V850ES	IECUBE Simulator	Both area below - 1 MByte space of addresses 0x000000 to 0x0FFFFF(fixed measurement areas) - Any 1 MByte space other than the internal ROM area (see "2.13.1 Configure the coverage measurement")
V850E2M (Single-core only)	IECUBE2	4 MBytes space of the internal ROM
	Simulator	Both area below - 1 MByte space of addresses 0x000000 to 0x0FFFFF(fixed measurement areas) - Any 1 MByte space other than the internal ROM area (see "2.13.1 Configure the coverage measurement")

Cautions 1.   [IECUBE]

The coverage function can be used only when the coverage board is mounted on IECUBE. Some of the trace function, the real-time display update function (RRM function) and the coverage function are used on a mutually exclusive basis.

2.   [IECUBE2]

Some of the trace function, the timer function (except for the Run-Break timer) and the coverage function are used on a mutually exclusive basis.

3.   [MINICUBE][MINICUBE2][E1]

The coverage function is not supported.

Remark   C0 coverage: Instruction coverage (statement coverage)

For example, when all statements in a code have been executed at least once, the C0 coverage is 100%.

2.13.1   Configure the coverage measurement

You need to configure the code coverage measurement before using the coverage function.

The setting method differs depending on the debug tool used.

(1)   [IECUBE]
(2)   [IECUBE2]
(3)   [Simulator]

(1) [IECUBE]

Caution   The coverage function can be used only when the coverage board is mounted on IECUBE. Therefore, when no coverage boards are mounted on IECUBE used, the following configurations become invalid after connecting to the debug tool.

Some of the trace function, the real-time display update function (RRM function) and the coverage function are used on a mutually exclusive basis.

Therefore, first, specify [Coverage] with the [Use for trace data] property in the [Trace] category in the [Debug Tool Settings] tab of the Property panel to use the coverage function with priority.

Note that settings described below are invalid when this property is set to other than [Coverage].

Figure 2-166.   [Trace] Category (Setting for the Coverage Function)
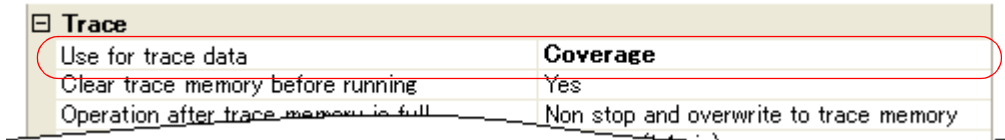
Then, configure the code coverage measurement function in the [Coverage] category in the [Debug Tool Settings] tab of the Property panel as follows:

Figure 2-167.   [Coverage] Category [IECUBE]
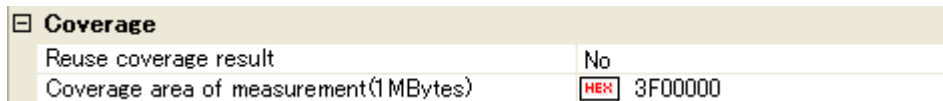
(a)   [Reuse coverage result]

Specify whether to restore the previous code coverage measurement result when connecting to the debug tool by automatically saving the acquired code coverage measurement result when disconnecting from the debug tool. Select [Yes] to restore the previous code coverage measurement result ([No] is selected by default).

(b)   [Coverage area of measurement(1MBytes)]

Specify the code coverage measurement area (see "Table 2-19.   Coverage Measurement Area").

Directly enter the start address of any 1 MByte space of addresses 0x100000 to 0x3FFFFFF in hexadecimal number (*the start address of the internal RAM area*] is specified by default).

(2)   [IECUBE2]

Some of the trace function, the timer function (except for the Run-Break timer) and the coverage function are used on a mutually exclusive basis.

Therefore, first, specify [Coverage] with the [Use for trace data] property in the [Trace] category in the [Debug Tool Settings] tab of the Property panel to use the coverage function with priority.

Note that settings described below are invalid when this property is set to other than [Coverage].

Figure 2-168.   [Trace] Category (Setting for the Coverage Function)
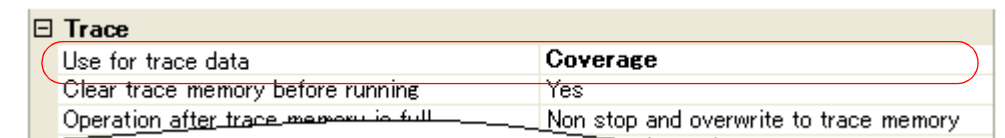
Then, configure the code coverage measurement function in the [Coverage] category in the [Debug Tool Settings] tab of the Property panel as follows:

Figure 2-169.   [Coverage] Category [IECUBE2]

(a) **[Reuse coverage result]**

Specify whether to restore the previous code coverage measurement result when connecting to the debug tool by automatically saving the acquired code coverage measurement result when disconnecting from the debug tool. Select [Yes] to restore the previous code coverage measurement result ([No] is selected by default).

(3) **[Simulator]**

You can configure the coverage measurement function in the [Coverage] category in the [Debug Tool Settings] tab of the Property panel as follows:

Figure 2-170.   [Coverage] Category [Simulator]

(a) **[Use coverage function]**

Specify whether to use the coverage function from the drop-down list.

Select [Yes] to use the coverage function ([No] is selected by default).

(b) **[Reuse coverage result]**

This property appears only when the [Use coverage function] property is set to [Yes].

Specify whether to restore the previous code coverage measurement result when connecting to the debug tool by automatically saving the acquired code coverage measurement result when disconnecting from the debug tool.

Select [Yes] to restore the previous code coverage measurement result ([No] is selected by default).

(c) **[Coverage area of measurement(1MBytes)]**

This property appears only when the [Use coverage function] property is set to [Yes].

Specify the code coverage measurement area (see "Table 2-19.   Coverage Measurement Area").

Directly enter the start address of any 1 Mbyte space other than the internal ROM area in hexadecimal number (*[the start address of the internal RAM area]* is specified by default).

2.13.2   Display the coverage measurement result

The code coverage measurement is automatically started when the program starts executing, and is stopped when the program stops.

(1) **Code coverage**

(a) **Displaying the code coverage ratio for each source text line/disassembled text line**

The code coverage ratio for each source text line/disassembled text line can be checked in the Editor panel/ Disassemble panel that is displaying the target program.

In each panel, background colors of the target source text lines/disassembled text lines are shown as "Table 2-21.", based on the coverage ratio obtained by using the expressions indicated in "Table 2-20.".

Note that the displaying of the code coverage ratio is not performed while executing a program or disconnecting from the debug tool.

All the code coverage measurement result can be cleared by selecting [Clear Coverage Information] from the context menu in the Editor panel/Disassemble panel (the color distinctions on the panels are also reset).

Table 2-20.   Expressions of Code Coverage Ratio

Panel	Expressions
Editor panel	"Number of bytes of code executed in the address range corresponding to the source text line" / "Total number of bytes of code in the address range corresponding to the source text line"
Disassemble panel	"Number of bytes of code executed in the address range corresponding to the disassembled text line" / "Total number of bytes of code in the address range corresponding to the disassembled text line"

Table 2-21.   View of Code Coverage Measurement Result (Default)

Code Coverage	Background Color
100 %	*Source text/disassembled text*
1 to 99 %	*Source text/disassembled text*
0 % (not yet executed)	*Source text/disassembled text*

Remarks 1.   The code coverage measurement result is automatically updated at a break in each panel.

   2.   Background colors above depend on the configuration in the [General - Font and Color] category of the Option dialog box.

   3.   Background colors above are not shown outside the coverage measurement area (see "Table 2-19.   Coverage Measurement Area").

   4.   When the downloaded lode module file is older than the source file currently being open, the displaying of the code coverage measurement result is not performed in the Editor panel.

Figure 2-171.   View of the Code Coverage Measurement Result (Editor Panel)
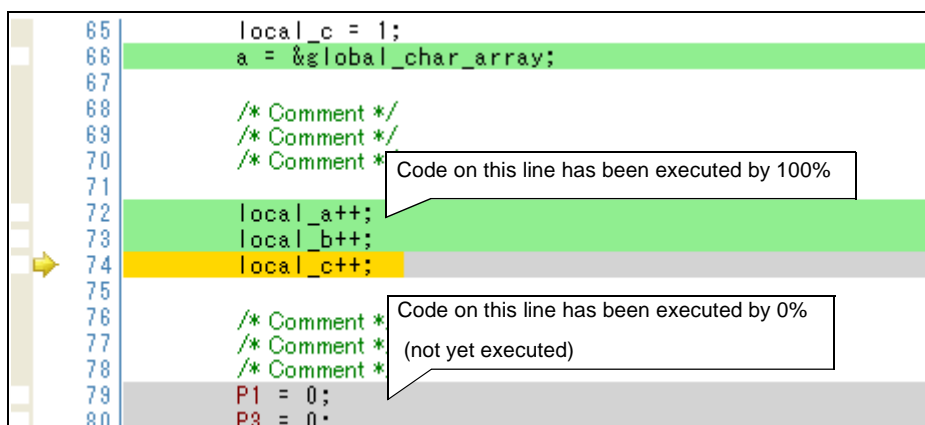
Figure 2-172.   View of the Code Coverage Measurement Result (Disassemble Panel)

(b) Displaying the code coverage ratio for each function

The code coverage ratio for each function can be checked by the [Code Coverage[%]] item in the Function List panel of the analyze tool. For details on "the code coverage ratio of the function", see "CubeSuite Analysis".

(2) Data coverage [IECUBE2][Simulator]

The data coverage ratio for each variable can be checked by the [Data Coverage[%]] item in the Variable List panel of the analyze tool. For details on "the data coverage ratio of the variable", see "CubeSuite Analysis".

2.14   Set an Action into Programs

This section describes how to set the specified action into the program.

2.14.1   Inset printf

By setting the Printf event that is one of "action events", the value of the specified variable expression can be output the Output panel by executing a printf command after temporary stopping the program in execution at an arbitrary position.
To use this function, follow the steps below.

Cautions 1.   **Also see "2.15.6   Notes for setting events" for details on Printf events (e.g. limits on the number of enabled events).**

2.   **Once a Printf event has been set, its variable expressions cannot be edited.**
 If you need to edit a variable expression, first delete the Printf event and then create a new one.

3.   **[Simulator]**
When [Yes] is specified with the [Execute instruction at breakpoint when break] property in the [Break] category on the [Debug Tool Settings] tab in the Property panel, all of Printf events currently being set are handled as Hardware Break events (printf commands will not be executed).

(1) Set a Printf event

Set a Printf event to the position where you want to execute the printf command in the Editor panel/Disassemble panel.
Select [Register Action Event...] from the context menu after moving the caret to the line/address[Note] to set a Printf event in the Editor panel/Disassemble panel to open the following the Action Events dialog box.
In this dialog box, follow the steps below.

Note   Printf events cannot be set at lines in the event area with the background color in gray (showing to the corresponding address of the line that no change is allowed).
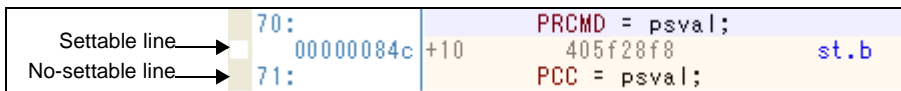
Figure 2-173.   Set Printf Event (Action Events Dialog Box)
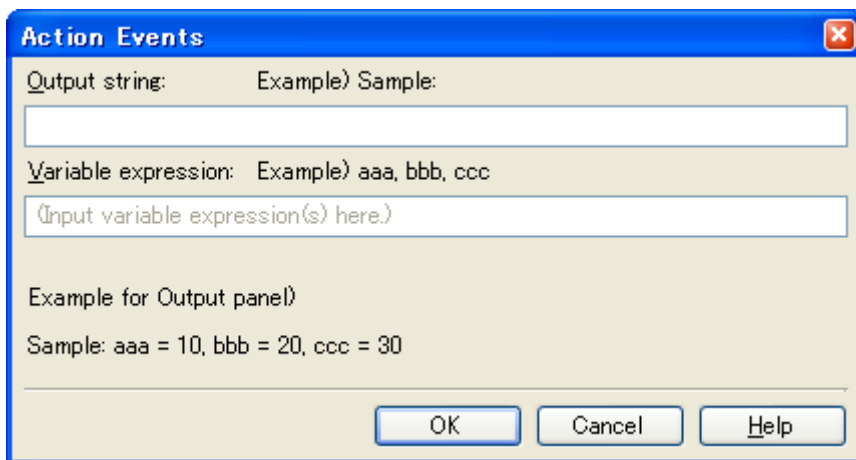
(a) Specify [Output string]

Directly enter from the keyboard the characters to add when output to the Output panel.
Characters must be in one line (spaces allowed).

(b) Specify [Variable expression]

Specify the variable expression for the Printf event to take place.

Type a variable expression directly into the text box (up to 1024 characters).

You can specify up to 10 variable expressions for a single Printf event by separating them with commas ",".

If this dialog box is opens with a variable expression selected in the Editor panel/Disassemble panel, the selected variable expression appears as the default.

For the basic input format that can be specified as variable expressions and the values output by Printf event, see "Table A-17. Relationship between Variable Expressions and Output Value (Printf Event)".

(c) Click the [OK] button

Set the Printf event to the line/address at the caret position in the Editor panel/Disassemble panel.

When the Printf event is set, the 🔔 mark is displayed in the event area on the Editor panel/Disassemble panel, and the set Printf event is managed in the Events panel (see "2.15 Manage Events").

(2) Execute the program

Execute the program (see "2.7 Execute Programs").

By executing the program, the program momentarily stops immediately before executing the instruction at the location where this event is set, and the value of the variable expression specified in this dialog box is output to the Output panel.

(3) Check the output result

The output result format from the Printf event in the [Debug Tool] tab of the Output panel are as follows:

Figure 2-174. Example of Output Result of Printf Event
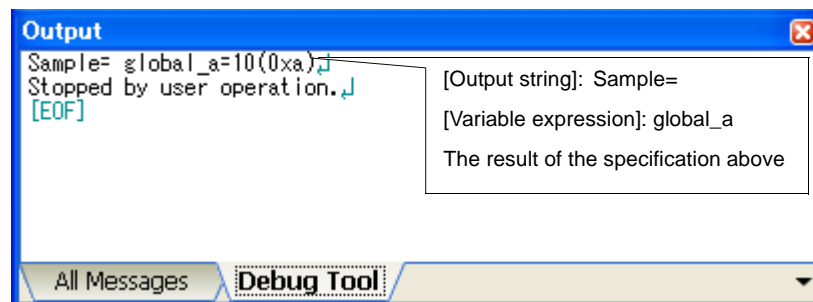
2.15   Manage Events

An event represents a certain status of the target system when debugging such as "Address 0x1000 is fetched" and "Data is written to address 0x2000".

In CubeSuite, these events are used as the action trigger of the debug function such as breakpoint, start/stop the tracing, and start/stop the timer.

This section describes how to manage those events.

Select the [View] menu >> [Event].

Events are all managed in the Events panel. In the Events panel, you can confirm the details of the currently set events in a list, and they can be deleted and changed enable/disable status.

For details on the contents and function in each area, see the section for the Events panel.

Figure 2-175.   Manage Events (Events Panel)
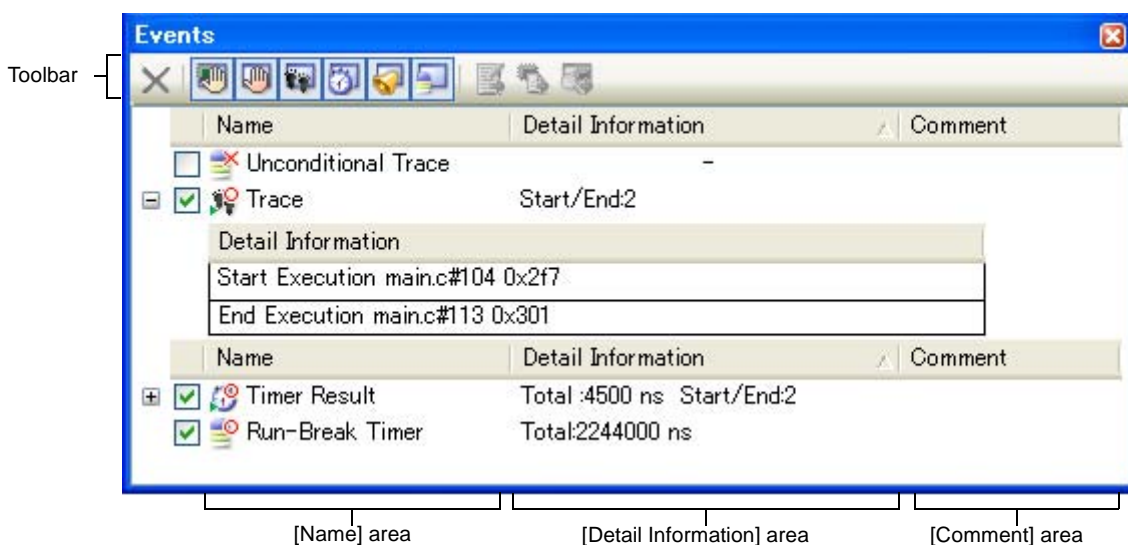
2.15.1   Change the state of set events (valid/invalid)

By changing the check on the check box of the event name, the setting state of the event can be changed (the Event mark is changed depending on the setting state of the event).

The following are types of the setting state of the event.

Figure 2-176.   Check Box of the Event
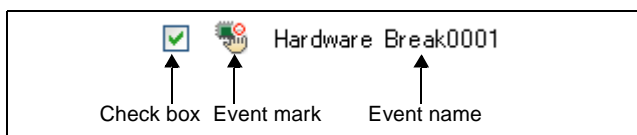
Table 2-22.   Setting State of the Event

☑	Valid state	Event occurs when the specified condition is met. It is possible to set the event to an invalid state by removing the check.
☐	Invalid state	Event does not occur when the specified condition is met. It is possible to set the event to a valid state by removing the check.
▣	Suspended State	The conditions that have been specified cannot be set with the program of the debugging target. It is not possible to operate the check box.

Remarks 1. Both of the Start Timer event and the Stop Timer event is must be set for the Timer Result event. Therefore, it is not possible to set a particular event to a valid state by only the setting of one of these (at the same time as both events are set, they are treated as grouped events as a Timer Result).

2. It is not possible to set the Run-Break Timer event to an invalid/suspended state.

3. The setting state of the event can be changed from the menu displayed by right clicking on the Event mark in the Editor panel/Disassemble panel.

4. The setting of the Unconditional Trace event and the Trace event to valid or invalid state is exclusively controlled. Therefore, the Unconditional Trace event, which is a built-in event, is valid state by default, but if either a Start Tracing event/Stop Tracing event is set, it automatically becomes invalid state, and the Trace event, which is a event name that is collectively called with a Start Tracing event and a Stop Tracing event, becomes valid state. Conversely, if the set Trace event is invalid state, the Unconditional Trace event automatically becomes valid state.

2.15.2   Display only particular event types

Click on the toolbar button to display only the particular event type.

	Displays events related to the Hardware Break.
[IECUBE][IECUBE2] [MINICUBE][MINICUBE2] [E1]	Displays events related to the Software Break.
[IECUBE][IECUBE2] [Simulator]	Displays events related to the trace.
[IECUBE][IECUBE2] [Simulator]	Displays events related to the timer.
	Displays events related to the action event (Printf event).
	Displays the built-in events (Unconditional Trace event and Run-Break Timer event).

2.15.3   Jump to the event address

Clicking the following buttons jumps to each panel which selected events address exist.

Note however, that when a Trace event/Timer Result event/ Unconditional Trace event/ Run-Break Timer event is selected, these button becomes invalid.

	Opens the Editor panel and jumps to the source line corresponding to the address where the selected event is being set.
	Opens the Disassemble panel and jumps to the disassemble results corresponding to the address where the selected event is being set.
	Opens the Memory panel and jumps to the source line corresponding to the address where the selected event is being set.

2.15.4   Delete events

To delete the set event, select the event and click the ✕ button in the toolbar.

Note that it is not possible to delete the built-in events (Unconditional Trace event and Run-Break Timer event).

Remarks 1. For the Break event of execution type, it is possible to delete the set event to click the event mark displayed in the Editor panel/Disassemble panel.

      2. To delete all of the set event, select [Select All from the context menu, then click the ☒ button (note, however, that it is not possible to delete the built-in events).

2.15.5 Write comment to events

The user can write comments for each event that has been set.

To input comments, click the [Comment] area after selecting the event to input comments, then input directly the desired text from the keyboard (the edit mode is cancelled by pressing the [Esc] key).

After editing the comments, complete the editing by pressing the [Enter] key or moving the focus to outside the edit region.

Up to 256 characters can be inputted for the comments, and this is saved as the settings of the user during use.

2.15.6 Notes for setting events

This section describes notes for setting each type of event.

(1) Maximum number of enabled events
(2) Event types that can be set and deleted during execution
(3) Other notes

(1) Maximum number of enabled events

The number of events that can be set to Valid state simultaneously is limited as follows:

Consequently, if enabling a new event would exceed the limit, you must first set some other event to Invalid state.

Table 2-23. Maximum Number of Enabled Events [V850E1][V850ES]

Event Type	Debug Tool to Use					
	IECUBE	MINICUBE E1 (JTAG)		MINICUBE2 E1 (Serial)		Simulator
		Nx85ET Core	Other than Nx85ET Core	With Debug Function	Without Debug Function	
Hardware Break (before execution)	2[Note 1]	2	2[Note 2]	2[Note 2]	-	64[Note 4]
Hardware Break (after execution)	8	8	0	0	-	
Hardware Break (access)	6	4	2[Note 2]	2[Note 2]	-	
Software Break	2000	2000 (For flash: 0 to 8[Note 3])				-
Trace (Start Tracing/Stop Tracing)	4 + 3[Note 5]	-	-	-	-	32[Note 5]
Point Trace	8 + 6	-	-	-	-	64
Timer Result (Start Timer/Stop Timer)	4 + 3[Note 5]	-	-	-	-	1
Action (Printf)	100[Note 6]					64[Note 7]

Table 2-24.   Maximum Number of Enabled Events [V850E2M]

Event Type	Debug Tool to Use			
	IECUBE	MINICUBE E1 (JTAG)		Simulator
		Nx85ET Core	Other than Nx85ET Core	
Hardware Break (before execution)	4	4	4	64[Note 4]
Hardware Break (after execution)	8	8	0	
Hardware Break (access)	6	6	0	
Software Break	8 (when TRAN is not incorporated) 40 - 200 (when TRAN is incorporated)			-
Trace (Start Tracing/Stop Tracing)	4 + 3 + 3[Note 5]	-	-	32[Note 5]
Point Trace	8 + 6 + 6	-	-	64
Timer Result (Start Timer/Stop Timer)	4 + 3 + 3[Note 5]	-	-	1
Action (Printf)	100[Note 6]			64[Note 7]

Notes 1.   1 when the flash emulation function is specified to valid ([Yes] is specified with the [Flash self-programing] property in the [Flash self Emulation] category in the [Flash Self Emulation Settings] tab [IECUBE] of the Property panel)

   2.   Hardware Break events (before execution) and Hardware Break events (access) are combined use

   3.   The number of settable break events that can be set in the flash memory area depends on the ROM collection function of the selected microcontroller.

   4.   The before break/after break can be specified in the Property panel.

   5.   Only one group can be set (however, two or more Start/Stop events can be set)

   6.   Combination with Software Break events

   7.   Combination with Hardware Break events

(2) Event types that can be set and deleted during execution
The following types of events can be set or deleted during program execution, or during tracer/timer execution.

Table 2-25.   Event Types That Can be Set and Deleted during Execution [V850E1][V850ES]

Event Type	Debug Tool to Use					
	IECUBE	MINICUBE E1 (JTAG)		MINICUBE2 E1 (Serial)		Simulator
		Nx85ET Core	Other than Nx85ET Core	With Debug Function	Without Debug Function	
Hardware Break (before execution)	△	△	△	△	—	▲
Hardware Break (after execution)	○	○	▲	▲	—	▲

Event Type	Debug Tool to Use					
	IECUBE	MINICUBE E1 (JTAG)		MINICUBE2 E1 (Serial)		Simulator
		Nx85ET Core	Other than Nx85ET Core	With Debug Function	Without Debug Function	
Hardware Break (access)	○	○	△	△	−	▲
Software Break	△	△	△	△	△	−
Trace (Start Tracing/Stop Tracing)	□	−	−	−	−	▲
Point Trace	□	−	−	−	−	▲
Timer Result (Start Timer/Stop Timer)	□	−	−	−	−	▲
Action (Printf)	△	△	△	△	−	▲

Table 2-26.   Event Types That Can be Set and Deleted during Execution [V850E2M]

Event Type	Debug Tool to Use			
	IECUBE2	MINICUBE E1 (JTAG)		Simulator
		Nx85ET Core	Other than Nx85ET Core	
Hardware Break (before execution)	△	△	△	▲
Hardware Break (after execution)	○	○	▲	▲
Hardware Break (access)	○	○	−	▲
Software Break	△	△	△	−
Trace (Start Tracing/Stop Tracing)	□	−	−	▲
Point Trace	□	−	−	▲
Timer Result (Start Timer/Stop Timer)	□	−	−	▲
Action (Printf)	△	△	△	▲

○ : Yes

△ : Yes, if the program execution is allowed to pause for events[Note]

□ : Yes, if the operation of tracer/timer is allowed to pause for events[Note]

▲ : Not while tracer or timer is executing

− : Cannot be set, or not supported

Note To enable this, specify [Yes] with the [Set event by stopping execution momentarily] property in the [Set Event While Running] category in the [Debug Tool Settings] tab of the Property panel.

(3) Other notes
- No events can be set to local variables.
- Events do not occur during step execution (including return execution) and program execution by selecting [Go to Here] from the context menu.

- If the location set for an existing event changes to midway in an instruction because the program to debug has been downloaded again, re-set the event using the following method.
    - If debugging information is available:
      The location setting of events is always moved to the beginning of the source text line.
    - If debugging information is not available:
      Depends on the [Automatic change method of event setting position] property in the [Download] category in the [Download File Settings] tab of the Property panel.

- If a change to internal ROM/RAM changes the location the event is set to a non-mapped area, then set events will not occur (they will also not change to Invalid state /Suspended State on the Events panel).

2.16   Use Hook Function

This section describes how to set hooks in the debug tool using the hook function.

Be setting a hook transaction, you can automatically change the values of the I/O register/CPU register before and after downloading a load module or after resetting the CPU.

Configure the hook transaction in the [Hook Transaction Settings] category in the [Hook Transaction Settings] tab of the Property panel.

Remark   By setting a I/O register by using the [Before download] property, for example, downloading can be executed at high speeds.
Downloading to the external RAM is also facilitated by using this function.

Figure 2-177.   [Hook Transaction Settings] Category

Table 2-27.   Properties in the [Hook Transaction Settings] Category

Property	Description
Before download	Perform the specified process immediately before downloading the load module file.
After download	Perform the specified process immediately after downloading the load module file.
After CPU reset under breaking	Perform the specified process immediately after resetting the CPU.
Before running	Perform the specified process immediately before starting program execution.
After breaking	Perform the specified process immediately after breaking program execution.

The properties in the [Hook Transaction Settings] category indicate the timing with which the hook process will be performed. "[]" indicates the current number of specified processes (no hook processes are configured by default).

Specify the target process in the property for which you want the hook process to be performed.

To specify a process, select the target property, then open the Text Edit dialog box by clicking the [...] button appears on the right edge of the field.

Figure 2-178.   Opening the Text Edit Dialog Box
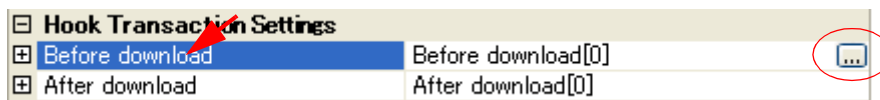
Figure 2-179. Use Hook Function (Text Edit Dialog Box)
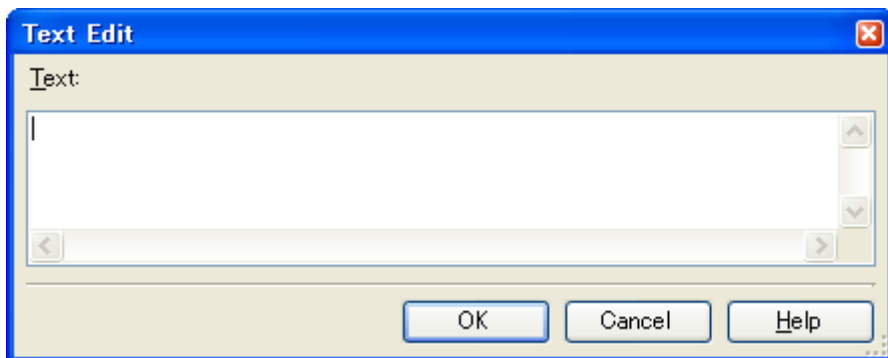
In this dialog box, directly enter the desired process from the key board.

The format for specifying processes is as follows. The value of the specified *I/O register*/CPU register will be overwritten with the specified value automatically at the corresponding time.

- *I/O register name*/*CPU register name* + space + *specified value*

Up to 64 characters for one process, and up to 128 processes for each property can be set (one line in the [Text] area in the Text Edit dialog box is equivalent to one processing).

After the specification of the process is complete, click the [OK] button to set the process to the Property panel.

Figure 2-180. Example of Hook Transaction
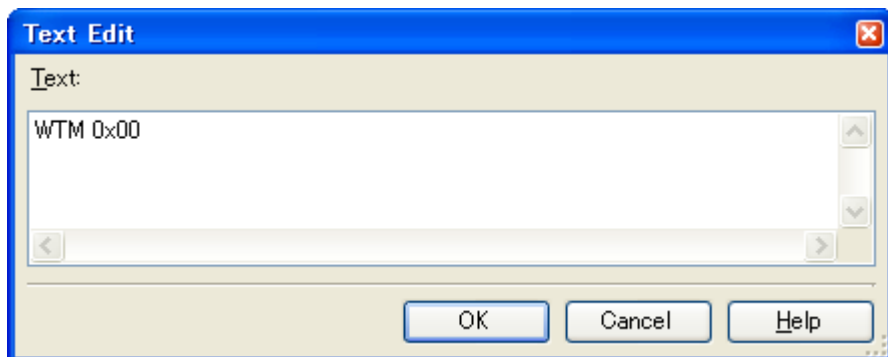
Remarks 1. When specifying hook processes, lines starting with a hash mark "#" will be treated as comments.

2. A tab character can be used instead of the space character.

2.17   Use the Simulator GUI [Simulator]

This section describes how to use the Simulator GUI.

Note that the function of the Simulator GUI described in this section is only supported when a microcontroller whose Simulator supports peripheral function simulations is selected.

Control the Simulator GUI via the Simulator GUI window below.

This window appears automatically by default after connecting to the debug tool when a microcontroller whose Simulator supports peripheral function simulations is selected and [Simulator] is selected as the debug tool to use.

Remark   The Simulator GUI window and windows opened from it cannot be docked to the CubeSuite Main window.

Figure 2-181.   Using the Simulator GUI (Simulator GUI Window)
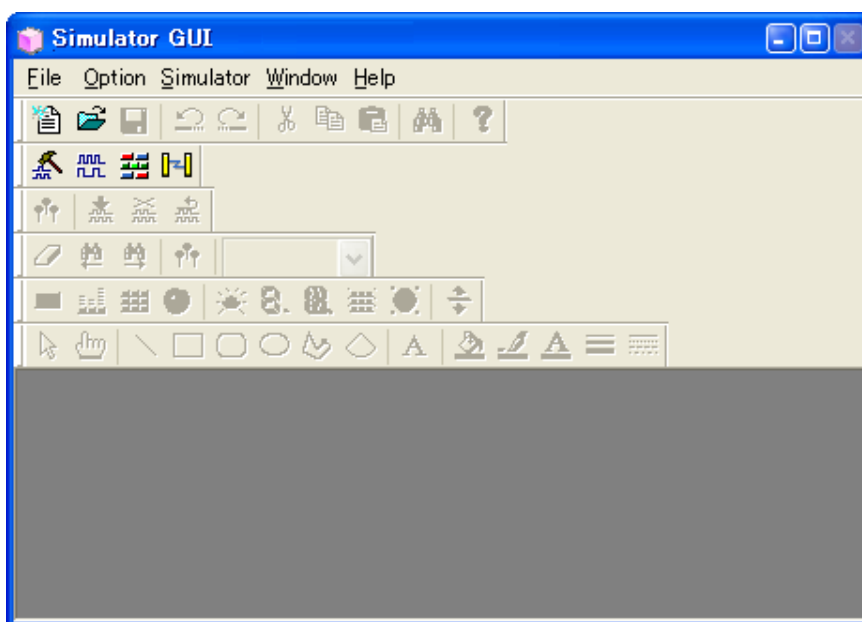
The setting of the display of the Simulator GUI window can be configured in the [Simulator GUI] category on the [Debug Tool Settings] tab of the Property panel as follows:

Configure the appropriate settings.

Caution   **After connecting to the debug tool, all the properties in this category will become invalid when a microcontroller whose Simulator does not support peripheral function simulations (instruction simulation version) is selected.**

Figure 2-182.   [Simulator GUI] Category

(1)   [Display Simulator GUI]

Specify whether to display the Simulator GUI window from the drop-down list.

Select [Yes] to use the function of the Simulator GUI (default).

When you do not need to use the Simulator GUI, select [No] to close the Simulator GUI window.

(2) [Display Simulator GUI on top of other windows]

This property appears only when the [Display Simulator GUI] property is set to [Yes].

Specify whether to display the Simulator GUI window in the forefront when program execution starts. Select [Yes] to display it in the forefront (default).

2.17.1   Check the I/O waveform of the microcontroller

It is possible to check the I/O waveform of the microcontroller by clicking the 🔲 button on the Simulator GUI window's toolbar and opening the Timing Chart window (shown below).

This window displays a timing chart of the input and output signals of the microcontroller's pins.

See the section on the Timing Chart window for details about controlling the window.

Figure 2-183.   Checking the I/O Waveform of the Microcontroller (Timing Chart Window)
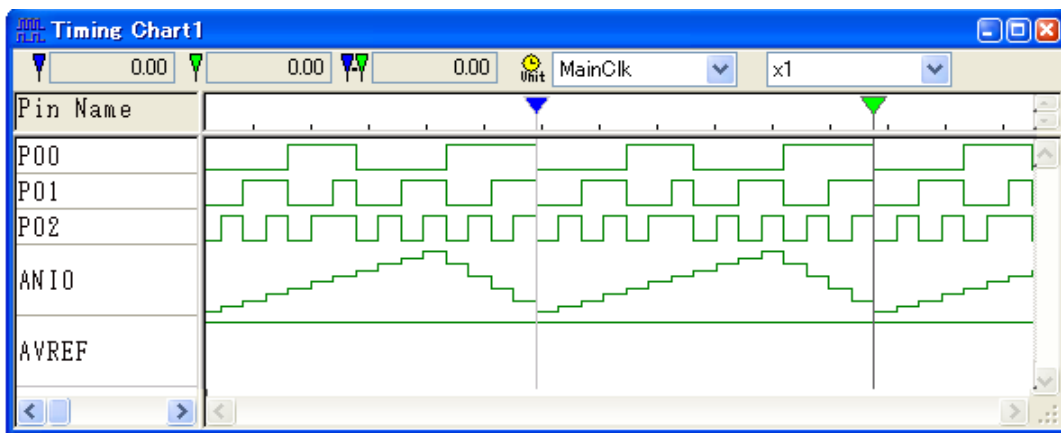
2.17.2   Input signals to the pins

To configure the input signal to a pin, click the 🔲 on the Simulator GUI window's toolbar. The following Signal Data Editor window opens.

You can use this window to set the input signal data for the input pin to a numerical value, to be input at an arbitrary time.

See the section on the Signal Data Editor window for details about controlling the window.

Figure 2-184.   Configuring the Input Signal to a Pin (Signal Data Editor Window)

2.17.3   Perform serial communication

To configure serial communication, click the 🔼 on the Simulator GUI window's toolbar. The following Serial window opens.

This window provides serial I/O features for communicating with the CPU's built-in serial interface. This enables you to input data to the microcontroller's serial receiver pin, and acquire output data from its serial transmitter pin.

See the section on the Serial window for details about controlling the window.

Figure 2-185.   Performing Serial Communication (Serial Window)
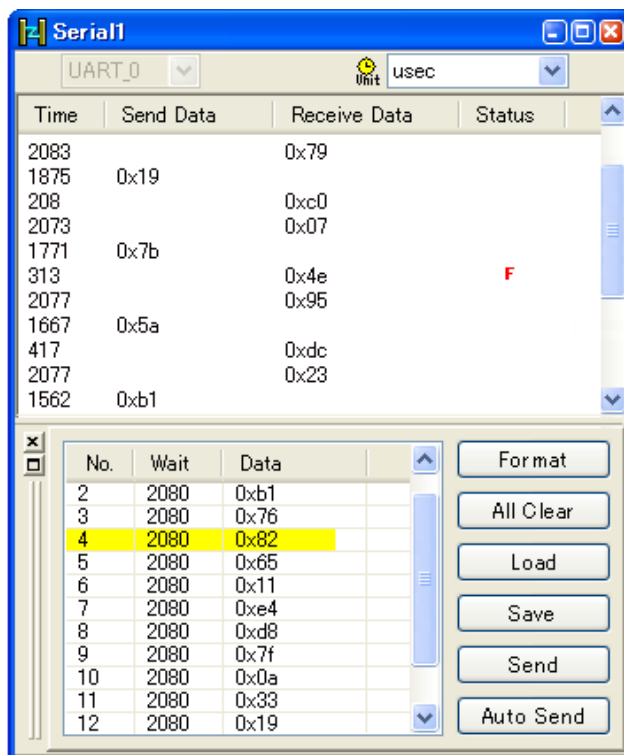
2.17.4   Use buttons, LEDs, level gauges, and other components

The Simulator GUI allows input manipulation and output display simulation by providing standard connected parts (buttons, LEDs, level gauges, etc.) in which the I/O block for peripheral I/O designed as a GUI interface.

To configure connected parts, click the 🔳 on the Simulator GUI window's toolbar. The following I/O Panel window opens.

This window enables you to configure the various connected parts, building a dummy target system.

See the section on the I/O Panel window for details about controlling the window.

Figure 2-186.   Configuring Connected Parts (I/O Panel Window)
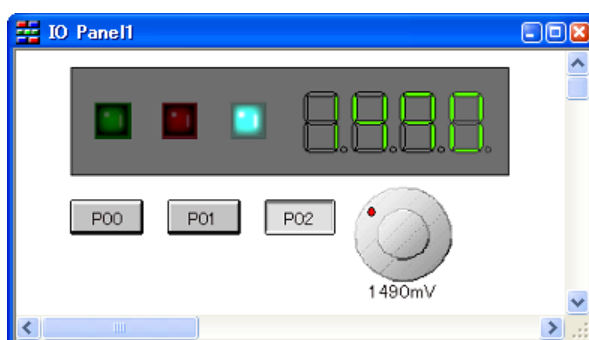
2.18 About Input Value

this section describes consideration to take when inputting values in each.

2.18.1 Input rule

Following is the rules for input to each panel/dialog box.

(1) Character set

Character sets that are allowed to input are as follows:

Table 2-28. List of Character Set

Character Set	Outline
ASCII	1- byte alphabets, numbers, symbols
Shift-JIS	2-byte alphabet, number, symbol, Hiragana, Katakana, Kanji and 1-byte Katakana.
EUC-JP	2-byte alphabet, number, symbol, Hiragana, Katakana, Kanji and 1-byte Katakana.
UTF-8	2-byte alphabet, number, symbol, Hiragana, Katakana, Kanji (include Chinese characters) and 1-byte Katakana.

(2) Number

Notations allowed when entering numbers are as follows:

Table 2-29. Notation List

Notation	Outline
Octal number	Start with 0 and continues with the numbers from 0 to 7.
Decimal	Start without 0 and continues with the numbers from 0 to 9.
Hexadecimal number	Start with 0x and continues with the numbers from 0 to 9 and alphabets a to f. (Case insensitive for alphabets) In the input area with the **HEX** mark, prefix 0x is not needed.

(3) Expression and operator

Expression represents constants, register name, I/O register name and symbols and those connected with operators.
When I/O register name, label name, function name and variable name are described as symbols, the address is operated as a value of symbols.
The basic input formats are as follows:

Table 2-30. Basic Input Format of Expressions

Expression	Description
Variable name of C language	Value of C language variable
Expression [*Index*]	Element of array
Expression Member name	Member of structures/unions
Expression->Member name	Member of structures/unions that pointer designates
Expression	Value of pointer variable
&*Expression*	Location address

Expression	Description
Name of CPU Register	Value of the CPU register
I/O register	I/O register value
Label, EQU symbol and immediate address	Values of label, EQU symbol and immediate address
Register name.bit	Register name.bit value
Label name.bit, EQU symbol.bit, immediate address.bit	Label name.bit value, EQU symbol. bit value, immediate addres.bit value
Bit symbol	Bit symbol value
Integer constant	Integer constant value
Floating constant	Floating point constant value
Character constant	Character constant value

From "Table 2-30.   Basic Input Format of Expressions", the following expressions with operator can be constructed.

Table 2-31.   Construction of Expressions with Operators

Expression	Description
(*Expression*)	Value of the parenthetical watch-expression
! *Expression*	Logical negation
~ *Expression*	Bit inversion
Expression ++	Backward increment
Expression --	Backward decrement
++ *Expression*	Forward increment
-- *Expression*	Forward decrement
(Data type) *Expression*	Type casting
size of *Expression*	Object size
size of (type name))	Type size
*Expression * Expression*	Product
Expression/ Expression	Quotient
Expression % Expression	Residue
Expression + Expression	Sum
Expression . Expression	Difference
Expression << Expression	Left shift
Expression >> Expression	Right shift
Expression < Expression	When it is "smaller", 1 is set, others 0 is set.
Expression <= Expression	When it is "smaller, or equal", 1 is set, others 0 is set.
Expression > Expression	When it is "bigger", 1 is set, others 0 is set.
Expression >= Expression	When it is "bigger, or equal", 1 is set, others 0 is set.

Expression	Description
Expression == *Expression*	When it is "equal", 1 is set, others 0 is set.
Expression != *Expression*	When it is not "equal", 1, others 0.
Expression & *Expression*	Logical multiplication by bits
Expression ^ *Expression*	Exclusive disjunction by bits
Expression | *Expression*	Logical sum by bits
Expression && *Expression*	Logical multiplication
Expression || *Expression*	Logical sum

2.18.2   Icons for invalid input

In some of the dialogs in CubeSuite, the 🔴 icons that represent that the input value is invalid when some invalid characters are input are shown at the relevant location and warns by changing the character color to red.

Remark   When the mouse cursor is hover over the 🔴 icon, information on the valid characters to input is pops up.

APPENDIX A   WINDOW REFERENCE

Appendix A provides detailed explanations of windows/panels/dialog boxes used for debugging with CubeSuite.

A.1   Description

Windows/panels/dialog boxes for debugging are listed below.

Table A-1.   Window/Panel/Dialog Box List

Window/Panel/Dialog Box Name	Description
Main window	Controls program execution. Various windows, panels and dialogs can be opened from this window.
Debug Manager panel [V850E2M]	Selects a core (PE*n*) to be debugged and displays the core status.
Project Tree panel	Selects the debug tool to use.
Property panel	Displays detailed information on the debug tool currently selected in the Project Tree panel, and enables the settings of the tool to be changed.
Editor panel	Enables text files to be viewed and edited, and is used to execute source level debug.
Memory panel	Displays and modifies memory values.
Disassemble panel	Displays the results of memory value disassemble and is used to execute line assemble and instruction level debug.
CPU Register panel	Displays the contents of CPU registers, and modifies register values.
IOR panel	Displays and modifies I/O register values.
Local Variables panel	Displays and modifies local variables.
Watch panel	Displays and modifies registered watch-expression values.
Call Stack panel	Displays call stack information on function calls.
Trace panel [IECUBE][IECUBE2][Simulator]	Displays trace data acquired from the debug tool.
Events panel	Displays detailed information on set events, switches the events between enabled and disabled, or deletes them.
Output panel	Displays messages output from the build tool/debug tool/plug-ins, or the results of batch searches carried out using the Search and Replace dialog box.
Memory Mapping dialog box	Sets the memory mapping.
Download Files dialog box	Selects files to be downloaded and sets the download conditions.
Flash Options Setting dialog box [V850E2M]	Configure options for the flash memory.
Text Edit dialog box	Inputs and modifies character strings.
Action Events dialog box	Sets action events.
Save Settings dialog box	Specifies the encoding and the new line code of the file being edited.
Memory Initialize dialog box	Initializes memory.
Memory Search dialog box	Searches memory.
Trace Search dialog box [IECUBE][IECUBE2][Simulator]	Searches trace data.

Window/Panel/Dialog Box Name	Description
Go to the Location dialog box	Moves the caret to the specified position.
Data Save dialog box	Saves the settings and other data displayed in the respective windows/panels/dialogs or saves upload data.
Progress Status dialog box	Displays the progress of the processing being executed.
Option dialog box	Makes settings for various environments.
Select Download File dialog box	Selects files to be downloaded.
Save As dialog box	Saves files or the contents of various windows/panels/dialogs.
Select Data Save File dialog box	Selects the file to save data.
Select Simulator Configuration File dialog box [Simulator]	Selects simulator configuration file.

Table A-2.   Simulator GUI Block-Dedicated Window/Dialog Box List

Window/Dialog Box Name	Description
Simulator GUI window	Opens and controls various simulator GUI's windows and dialog boxes.
Customize dialog box	Setting of window colors and fonts
Signal Data Editor window	Setting of input signal data
Loop dialog box	Setting of loop information for the Signal Data Editor window
Select Pin dialog box	Selection of display pins in the Signal Data Editor window and Timing Chart window
Timing Chart window	Timing chart display of input and output signals
Search Data dialog box	Detailed search of the Timing Chart window
I/O Panel window	Creation of dummy target system
Parts Button Properties dialog box	Setting of button pin connection information
Analog Button Properties dialog box	Setting of analog button pin connection information
Parts Key Properties dialog box	Setting of key matrix LED pin information
Parts Level Gauge Properties dialog box	Setting of level gauge pin connection information
Parts Led Properties dialog box	Setting of key matrix LED pin information
Parts Segment LED Properties dialog box	Setting of 7/14-segment LED pin connection information
Parts Matrix Led Properties dialog box	Setting of matrix LED pin connection information
Parts Buzzer Properties dialog box	Setting of buzzer pin connection information
Pull up/Pull down dialog box	Setting of pull-up/pull-down resistor pin connection information
Object Properties dialog box	Setting of connection information for figure, character, and bitmap pins
Parts List dialog box	List display of object pin connection status in the I/O Panel window
Serial window	Serial interface communication window
Format (UART) dialog box	Setting of serial format (UART)
Format (CSI) dialog box	Setting of serial format (CSI)

Main window

This window is automatically opened when CubeSuite is started up.

In this window, you can control the program execution and open panels for the debugging process.

Figure A-1.   Main Window
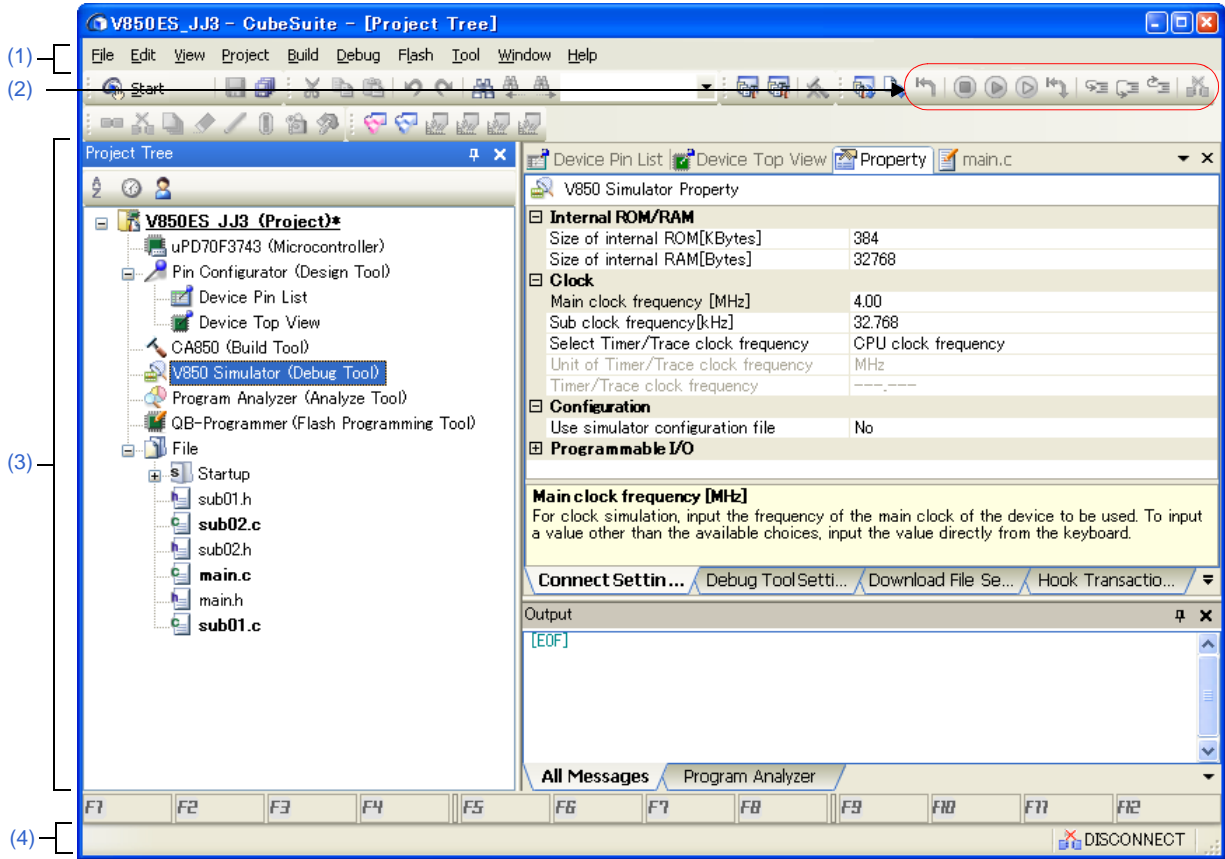
This section describes the following.

- [How to open]
- [Description of each area]

[How to open]

- From the Windows [Start] menu, select [All Programs] >> [NEC Electronics CubeSuite] >> [CubeSuite].

[Description of each area]

(1) Menubar

Menu items related to the debugging are as follows:

Remark   The items that can be selected in each menu can be customized using the User Setting dialog box.

(a) [View]

The [View] menu provides the following items and functions (default).

Debug Manager **[V850E2M]**	Opens the Debug Manager panel [V850E2M] This item is invalid when the selected microcontroller version does not support multi-core or while disconnecting from the debug tool.
Watch	The following cascade menus are displayed to open the Watch panel. These items are invalid while disconnecting from the debug tool.
Watch1	Opens the Watch panel (Watch1).
Watch2	Opens the Watch panel (Watch2).
Watch3	Opens the Watch panel (Watch3).
Watch4	Opens the Watch panel (Watch4).
Local Variable	Opens the Local Variables panel.
Call Stack	Opens the Call Stack panel.
Memory	The following cascade menus are displayed to open the Memory panel. These items are invalid while disconnecting from the debug tool.
Memory1	Opens the Memory panel (Memory1).
Memory2	Opens the Memory panel (Memory2).
Memory3	Opens the Memory panel (Memory3).
Memory4	Opens the Memory panel (Memory4).
IOR	Opens the IOR panel. This item is invalid while disconnecting from the debug tool.
CPU Register	Opens the CPU Register panel. This item is invalid while disconnecting from the debug tool.
Trace **[IECUBE][IECUBE2] [Simulator]**	Opens the Trace panel [IECUBE][IECUBE2][Simulator]. This item is invalid while disconnecting from the debug tool.
Disassemble	The following cascade menus are displayed to open the Disassemble panel. These items are invalid while disconnecting from the debug tool.
Disassemble1	Opens the Disassemble panel (Disassemble1).
Disassemble2	Opens the Disassemble panel (Disassemble2).
Disassemble3	Opens the Disassemble panel (Disassemble3).
Disassemble4	Opens the Disassemble panel (Disassemble4).
Event	Opens the Events panel. This item is invalid while disconnecting from the debug tool.

Show Current PC Location	Displays the current PC position in the Editor panel.
	This item is invalid while disconnecting from the debug tool.
Back to Last Cursor Position	Goes back to the position before jumping (see "(4) Jump to functions"/"(4) Move to the symbol defined location") to the defined location.
	This item is invalid while disconnecting from the debug tool.
Forward to Next Cursor Position	Forwards to the position before operating [Back to Last Cursor Position].
Tag Jump	Jumps to the corresponding line/column in the corresponding file if the information of a file name/line number/column number exists in the line at the caret position on the Editor panel/Output panel (see "(5) Jump to a desired line (tag jump)").

(b) [Debug]

The [Debug] menu provides the following items and functions (default).

Download	Downloads the specified file(s) into the debug tool for an active project.
	While disconnecting from the debug tool, the download is executed after connecting to the debug tool.
	This item becomes invalid during execution of a program.
Build & Download	Executes the build of a project and downloads the file into the debug tool for an active project.
	While disconnecting from the debug tool, the download is executed after connecting to the debug tool.
	However, when the build has failed, download will not be executed.
Connect to Debug Tool	Connects to the debug tool for an active project.
Upload...	Opens the Data Save dialog box to save the memory contents.
	This item becomes invalid during execution of a program or while disconnecting from the debug tool.
Disconnect from Debug Tool	Disconnects from the currently connected debug tool.
Stop	Forcibly stops the program currently being executed.
	This item becomes invalid during non-execution of a program or while disconnecting from the debug tool.
Go	Executes the program from the current PC position.
	Execution of the program will be stopped when the condition of a set break event is met.
	This item becomes invalid during execution of a program or while disconnecting from the debug tool.
Ignore break and go	Executes the program from the current PC position.
	Execution of the program continues, ignoring set beak events and action events.
	This item becomes invalid during execution of a program or while disconnecting from the debug tool.
Step In	Executes the program step by step[Note] from the current PC position (Step in execution).
	However, in the case of a function call, the program is stopped at the beginning of the function having been called.
	This item becomes invalid during execution of a program or while disconnecting from the debug tool.

Step Over	Executes the program step by step[Note] from the current PC position (Step over execution).
	In the case of a function call by the jarl instruction, all the source lines/instructions in the function are treated as one step and executed until the position where execution returns from the function (step-by-step execution will continue until the same nest is formed as when the jarl instruction has been executed).
	In the case of an instruction other than jarl, operation is the same as when [Step In] is selected.
	This item becomes invalid during execution of a program or while disconnecting from the debug tool.
Return Out	Executes the program until execution returns from the current function (or returns to the calling function)[Note](Return out execution).
	This item becomes invalid during execution of a program or while disconnecting from the debug tool.
CPU Reset	Resets the CPU (does not execute a program)
	This item becomes invalid during execution of a program or while disconnecting from the debug tool.
Restart	Resets the CPU and then executes the program from the reset address.
	This item becomes invalid during execution of a program or while disconnecting from the debug tool.

Note   Step execution can be carried out either in units of source lines or in units of instructions.
For details, see "2.7.3   Execute programs in steps".

(2) Debug toolbar

The debug toolbar includes the buttons that control the execution of programs.
The debug toolbar provides the following buttons and functions (default).

Remarks 1.   The buttons on the toolbar can be customized using the User Setting dialog box. Furthermore, a new toolbar can be created using the same dialog box.

2.   A Group of toolbar displayed can be selected with the context menu that is displayed by right-clicking on the toolbar.

	Executes the build of a project and downloads the file into the debug tool for an active project. While disconnecting from the debug tool, the download is executed after connecting to the debug tool. However, when the build has failed, download will not be executed. The function of this item is the same as that of [Build & Download] in the [Debug] menu.
	Downloads the specified file(s) into the debug tool for an active project. While disconnecting from the debug tool, the download is executed after connecting to the debug tool. The function of this item is the same as that of [Download] in the [Debug] menu.
	Resets the CPU (does not execute a program) This item becomes invalid during execution of a program or while disconnecting from the debug tool. The function of this item is the same as that of [CPU Reset] in the [Debug] menu.
	Forcibly stops the program currently being executed. This item becomes invalid during non-execution of a program or while disconnecting from the debug tool. The function of this item is the same as that of [Stop] in the [Debug] menu.
	Executes the program from the current PC position. Execution of the program will be stopped when the condition of a set break event is met. This item becomes invalid during execution of a program or while disconnecting from the debug tool. The function of this item is the same as that of [Go] in the [Debug] menu.

[icon]	Executes the program from the current PC position. Execution of the program continues, ignoring set beak events and action events. This item becomes invalid during execution of a program or while disconnecting from the debug tool. The function of this item is the same as that of [Ignore break and go] in the [Debug] menu.
[icon]	Resets the CPU and then executes the program from the reset address. This item becomes invalid during execution of a program or while disconnecting from the debug tool. The function of this item is the same as that of [Restart] in the [Debug] menu.
[icon]	Executes the program step by step[Note] from the current PC position (Step in execution). However, in the case of a function call, the program is stopped at the beginning of the function having been called. This item becomes invalid during execution of a program or while disconnecting from the debug tool. The function of this item is the same as that of [Step In] in the [Debug] menu.
[icon]	Executes the program step by step[Note] from the current PC position (Step over execution). In the case of a function call by the jarl instruction, all the source lines/instructions in the function are treated as one step and executed until the position where execution returns from the function (step-by-step execution will continue until the same nest is formed as when the jarl instruction has been executed). In the case of an instruction other than jarl, operation is the same as when the [icon] button is clicked. This item becomes invalid during execution of a program or while disconnecting from the debug tool. The function of this item is the same as that of [Step Over] in the [Debug] menu.
[icon]	Executes the program until execution returns from the current function (or returns to the calling function)[Note] (Return out execution). This item becomes invalid during execution of a program or while disconnecting from the debug tool. The function of this item is the same as that of [Return Out] in the [Debug] menu.
[icon]	Disconnects from the currently connected debug tool. The function of this item is the same as that of [Disconnect from Debug Tool] in the [Debug] menu.

Note Step execution can be carried out either in units of source lines or in units of instructions.
For details, see "2.7.3   Execute programs in steps".

(3) Panel display area

This area displays the various panels.
For details on the display content, see the sections describing the individual panels.

(4) Statusbar

Statusbar displays the following items of information.

Figure A-2.   Statusbar
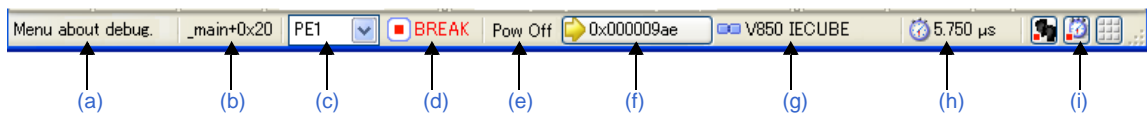
(a) Status message

This area displays the following messages and other information.
- A brief explanation of the selected menu item
- A message reporting that an invalid value has been input in the panel/dialog
- A message reporting that the specified character string has not been found as a result of a search using the Search and Replace dialog box
- A statement of the cause of the break when a break has occurred (see "2.8   Stop Programs (Break)")

(b) Focus panel status information

This area displays status information on the panel currently having the focus.

Note that nothing is displayed here for a panel that has no status information.

(c) Selection of debug target core [V850E2M]

This area is used to select a core (PE*n*) to be debugged.

Note that nothing is displayed here when the selected microcontroller version does not support multi-core or while disconnecting from the debug tool.

Remark   You can also select a core to be debugged with the Debug Manager panel [V850E2M].

(d) Running state

This area displays the state of the program with the following icons and character strings.

Note that nothing is displayed here when the debug tool is not connected.

State of Program	Display Contents
Running	▶ RUN
Stopped	■ BREAK
In step execution	▶❙ STEP

(e) CPU status

This area displays CPU statuses of the debug tool. When there is the possibility that the CPU is in two or more statuses, the corresponding display contents are displayed separated by "&".

Note that nothing is displayed here when the debug tool is not connected.

Debug Tool	Display Contents	CPU Status
IECUBE IECUBE2 MINICUBE E1 (JTAG)	Halt	In HALT mode
	StpIdle	Hardware STOP/Software STOP/IDLE mode
	Hold	In bus hold
	Wait	In wait state
	Reset	In reset state
	Pow Off	Power not supplied to the target
MINICUBE2 E1 (Serial)	Reset	In reset state
	Pow Off	Power not supplied to the target
Simulator	Halt	In HALT mode
	StopIdle	In STOP/IDLE mode
	Reset	In reset state

(f) Current PC position

This area displays the current PC position with a hexadecimal value. When this area is clicked, the caret moves to the current PC position on the Editor panel.

In addition, when the mouse pointer is placed over this area, a pop-up window appears to display the following information.

- Current PC: 0x *current PC value* (*source name#line count*[Note])

Note that nothing is displayed here when the debug tool is not connected.

Note   "*symbol name+offset value*" is displayed when acquisition of information is impossible.

(g) Connection state

This area displays the current state of connection with the debug tool using the following icons and character strings.

Connection State	Display Contents
Connected	🖥️ *Debug tool name*
Disconnected	❌ DISCONNECTED

(h) Run-Break Timer measurement result

This area displays the result of measurement by the Run-Break Timer event (the unit of value used differs depending on the measurement amount). See "2.12.1   Measure execution time until stop of the execution". Note that nothing is displayed here when the debug tool is not connected.

Condition	Display Contents
Un-measuring	Unmeasurement
Under measurement	Measuring
When a timer measurement overflow has occurred	OVERFLOW

(i) Debug tool state [IECUBE][IECUBE2][Simulator]

This area displays the current state of debug tool's functions using the following icons and character strings. When a function is stopped, clicking the appropriate icon enables the state to be switched between "Use" and "Not use"[Note].

Note that nothing is displayed here when the debug tool is not connected.

Function	Being Executed	Stopped (Use)	Not Use
Trace	👣	👣	👣
Timer	⏱️	⏱️	⏱️
Coverage	▦	▦	▦

Note [Simulator]

The result of switching will be reflected in the setting of [Yes]/[No] for the [Use trace function]/[Use timer function]/[Use coverage function] property of the [Trace]/[Timer]/[Coverage] category in the [Debug Tool Settings] tab of the Property panel.

Debug Manager panel [V850E2M]

When the selected microcontroller is the V850E2M multi-core product, this panel is used to select a core (PE: Processor Element) to be debugged and display the core status.

This panel appears only when connected to the debug tool.

Caution   This panel cannot be opened when the selected microcontroller is the single-core product.

Remarks 1.   Synchronous executions and synchronous breaks are performed in all PE in principle. Note, however, that step executions are performed only within a PE currently being selected.

2.   Breakpoints are automatically set to be valid in all PE.

Figure A-3.   Debug Manager Panel
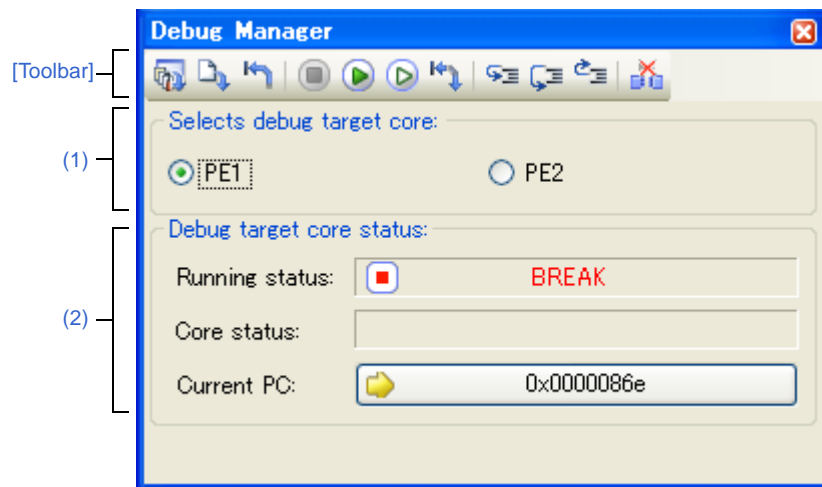
This section describes the following.
- [How to open]
- [Description of each area]
- [Toolbar]

[How to open]

- From the [View] menu, select [Debug Manager].

[Description of each area]

(1) [Selects debug target core] area
Select a core (PE*n*) to be debugged with a option button.
Note that this area becomes invalid during execution of a program.

Remark   You can also select a core to be debugged on the statusbar in the Main window.

(2) [Debug target core status] area
This area displays the status of the core currently being selected.

Remark   You can also confirm the information displayed in this area on the statusbar in the Main window.

(a) [Running status]

Displays the current state of the program with the following icons and character strings.

State of Program	Display Contents
Running	▶ RUN
Stopped	■ BREAK
In step execution	▶ⅠSTEP

(b) [Core status]

Displays the current core statuses of the debug tool. When there is the possibility that the core is in two or more statuses, the corresponding display contents are displayed separated by "&".

Debug Tool	Display Contents	Core Status
MINICUBE E1 (JTAG)	Halt	In HALT mode
	StpIdle	Hardware STOP/Software STOP/IDLE mode
	Hold	In bus hold
	Wait	In wait state
	Reset	In reset state
	Pow Off	Power not supplied to the target

(c) [Current PC]

Displays the current PC position with a hexadecimal value. When this button is clicked, the caret moves to the current PC position on the Editor panel.

[Toolbar]

The function of this toolbar is the same as that of the Debug toolbar on the Main window. For details on the function of each button, see "(2) Debug toolbar".

Project Tree panel

This panel is used to display the project components (microcontroller, build tool, debug tool, etc.) in a tree structure.

On this panel, You can select or change the debug tool to use.

Figure A-4.   Project Tree Panel
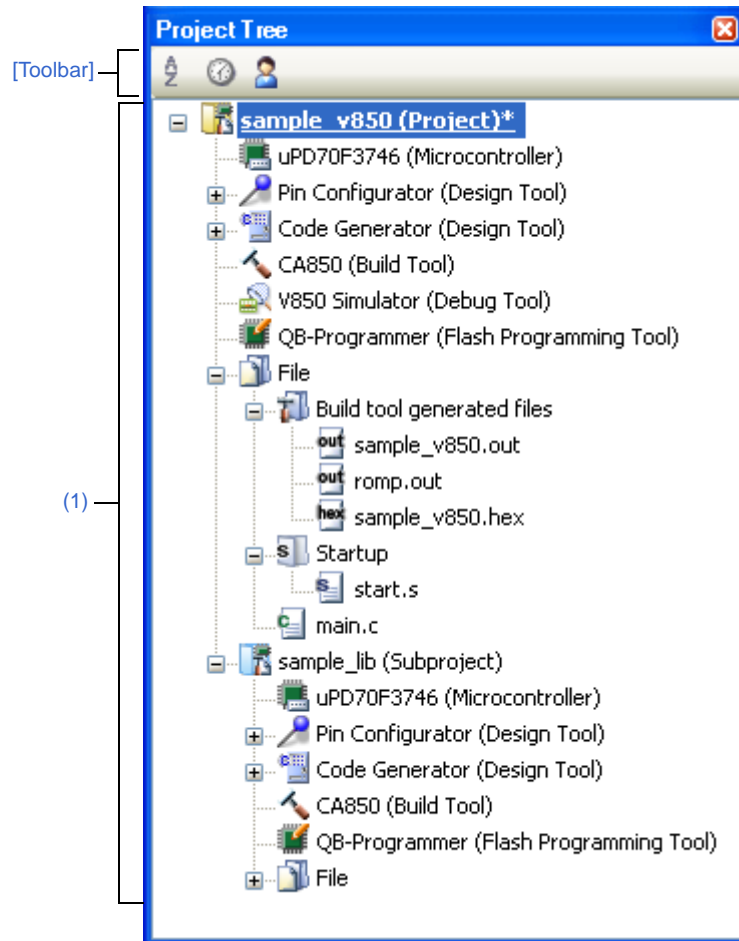
The following items are explained here.

- [How to open]
- [Description of each area]
- [Toolbar]
- [Context menu]

[How to open]

- From the [View] menu, select [Project Tree].

[Description of each area]

(1) Project tree area

Project components are displayed in tree view with the following given node.

Node	Description
Microcontroller type Debug tool name (Debug tool)	- *Microcontroller type*: The selected microcontroller type (V850 or V850E2M) is displayed. - *Debug tool name*: The debug tool (IECUBE, IECUBE2, MINICUBE, MINICUBE2(Serial), E1 (Serial), E1 (JTAG) or Simulator) used in the project is displayed. Simulator is selected by default.

Select the debug tool node to configure with the Property panel. If the Property panel is not being opened, double-click the node to open the corresponding Property panel.

[Toolbar]

[icons]	Sorts the category nodes and the file names into their name (character code) order. Clicking repeatedly toggles the sort order between ascending and descending.	
	[icon]	Indicates that the items are not being sorted into their name order (default).
	[icon]	Indicates that the items are being sorted in descending order.
	[icon]	Indicates that the items are being sorted in ascending order.
[icons]	Sorts the category nodes and the file names into their time-stamp order. Clicking repeatedly toggles the sort order between ascending and descending.	
	[icon]	Indicates that the items are not being sorted into their time-stamp order (default).
	[icon]	Indicates that the items are being sorted in descending order.
	[icon]	Indicates that the items are being sorted in ascending order.
[icons]	Sorts the category nodes and the file names into the order specified by user.	
	[icon]	Indicates that the items are not being sorted by user.
	[icon]	Indicates that the items are being sorted by user (default). You can arbitrarily customize the order of the items by dragging and dropping them.

[Context menu]

Using Debug Tool		The following cascade menus are displayed to select the debug tool to use. Note that the debug tools displayed in this menu differ depending on the microcontroller selected in the project (see "Table 2-1. Relationship between Types of Microcontroller and Connectable Debug Tools").
	Device name IECUBE	Uses IECUBE as the debug tool.
	Device name IECUBE2	Uses IECUBE2 as the debug tool.
	Device name MINICUBE	Uses MINICUBE as the debug tool.
	Device name MINICUBE2(Serial)	Uses MINICUBE2 as the debug tool.
	Device name E1 (Serial)	Uses E1 with serial communication method as the debug tool.
	Device name E1 (JTAG)	Uses E1 with JTAG communication method as the debug tool.
	Device name Simulator	Uses Simulator as the debug tool.
Property		Displays the selected category node's property in the Property panel.

Property panel

This panel is used to display and set the debug tool operation environment that is selected in the Project Tree panel.

Figure A-5. Property Panel (When IECUBE Is Selected)
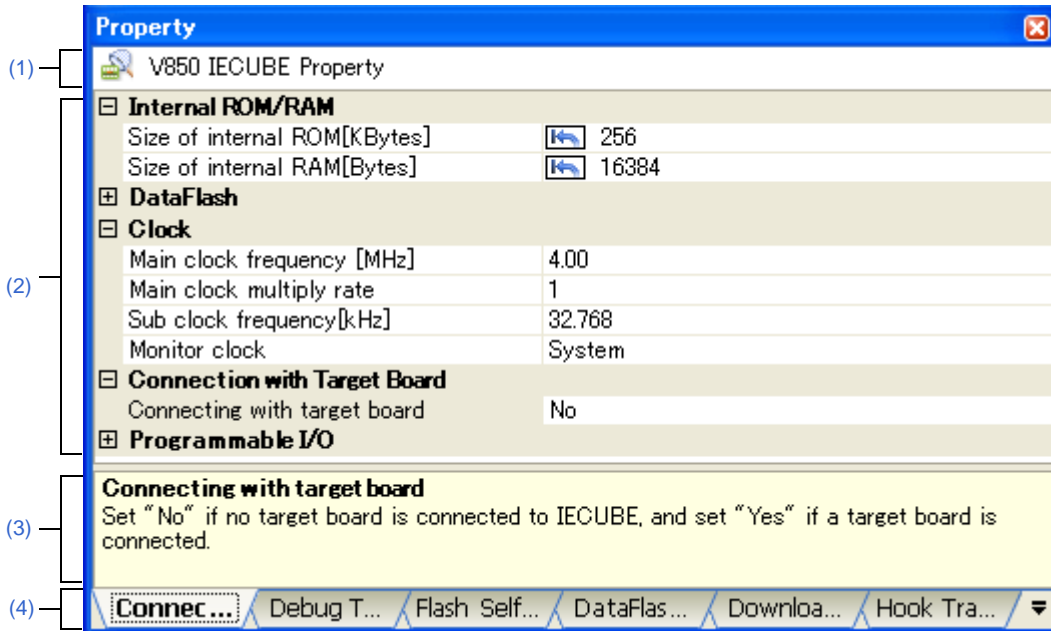
This section describes the following.
- [How to open]
- [Description of each area]
- [[Edit] menu (Property panel-dedicated items)]
- [Context menu]

[How to open]

- On the Project Tree panel, select the [*Microcontroller type Debug tool name* (Debug Tool)] node to use, and then select [Property] from the [View] menu or the context menu.
- On the Project Tree panel, double-click the [*Microcontroller type Debug tool name* (Debug Tool)] node to use.

Remark If this panel has been opened, the detailed information on the debug tool is displayed by selecting the [*Microcontroller type Debug tool name* (Debug Tool)] node on the Project Tree panel.

[Description of each area]

(1) Selected node area
In this area, the name of the selected debug tool on the Project Tree panel is displayed.

(2) Detailed information display/change area
In this area, the detailed information on the debug tool that is selected in Project Tree panel is displayed by category in the list. Also, you can directly change its settings.
The ⊟ mark is indicates all the items in the category are expanded. The ⊞ mark indicates all the items are shrink. You can expand/shrink the items by clicking these marks or double-clicking the category name.

Note that only the hexadecimal number is allowed in the text box if the `HEX` mark is displayed in the property configuration area.

For details on the information/how to setup in the category and property items contained in it, see the section explaining the corresponding tab.

(3) Property description area

In this area, brief description of the categories and properties selected in the detailed information display/change area is displayed.

(4) Tab selection area

Categories for the display of the detailed information are changed when each tab is selected.

In this panel, following tabs are contained (see the section explaining each tab for details on the display/setting on the tab).

- [Connect Settings] tab
- [Debug Tool Settings] tab
- [Flash Self Emulation Settings] tab [IECUBE]
- [DataFlash Emulation Settings] tab [IECUBE]
- [Download File Settings] tab
- [Flash Options Settings] tab [V850E2M]
- [Hook Transaction Settings] tab

[[Edit] menu (Property panel-dedicated items)]

Undo	Undoes the latest property value editing being done.
Cut	Deletes the selected character string(s) and copies them to the clipboard while editing the property value.
Copy	Copies the contents of the selected range to the clipboard as character string(s).
Paste	Pastes the contents of the clipboard to the property value while editing the property value.
Delete	Deletes the selected character string(s) while editing the property value.
Select All	Selects all the character strings in the selected property while editing the property value.

[Context menu]

[While not editing the property value]

Reset to Default	Restores the selected setting of the property item to default value.
Reset All to Default	Restores all the selected settings of the property items on the tab to default value.

[While editing the property value]

Undo	Undoes the latest property value editing being done.
Cut	Deletes the selected character string(s) and copies them to the clipboard while editing the property value.
Copy	Copies the contents of the selected range to the clipboard as character string(s).
Paste	Pastes the contents of the clipboard to the property value while editing the property value.
Delete	Deletes the selected character string(s) while editing the property value.
Select All	Selects all the character strings in the selected property while editing the property value.

[Connect Settings] tab

This tab is used to display the detailed information categorized by the following and the configuration can be changed.

(1)   [Internal ROM/RAM]
(2)   [DataFlash] [V850E1][V850ES] (except Simulator)
(3)   [Clock]
(4)   [Connection with Target Board] [IECUBE][IECUBE2][MINICUBE2][E1]
(5)   [Flash] [MINICUBE][MINICUBE2][E1]
(6)   [Configuration] [Simulator]
(7)   [Programmable I/O] [V850E1][V850ES]

Figure A-6.   Property Panel: [Connect Settings] Tab [IECUBE]
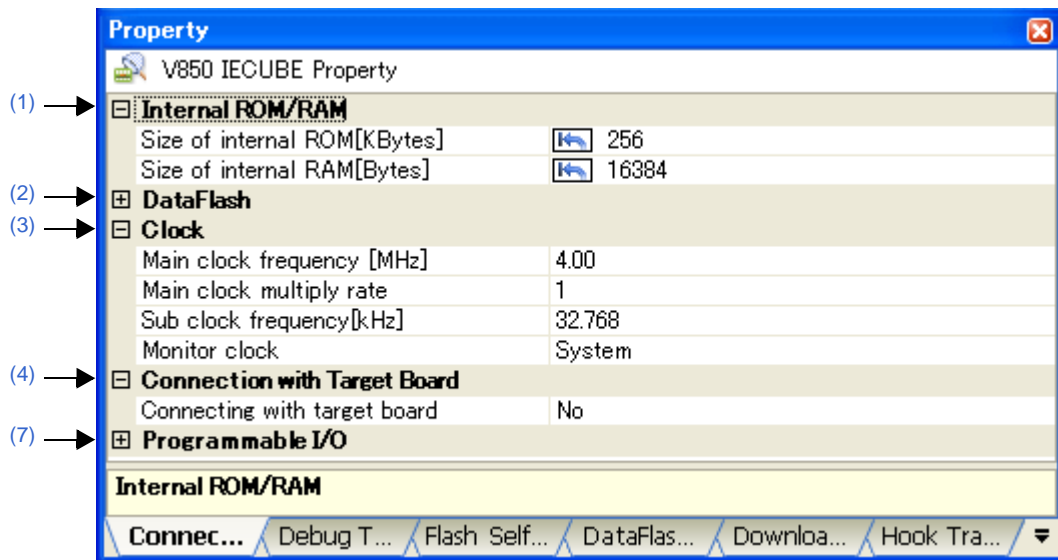
Figure A-7.   Property Panel: [Connect Settings] Tab [IECUBE2]
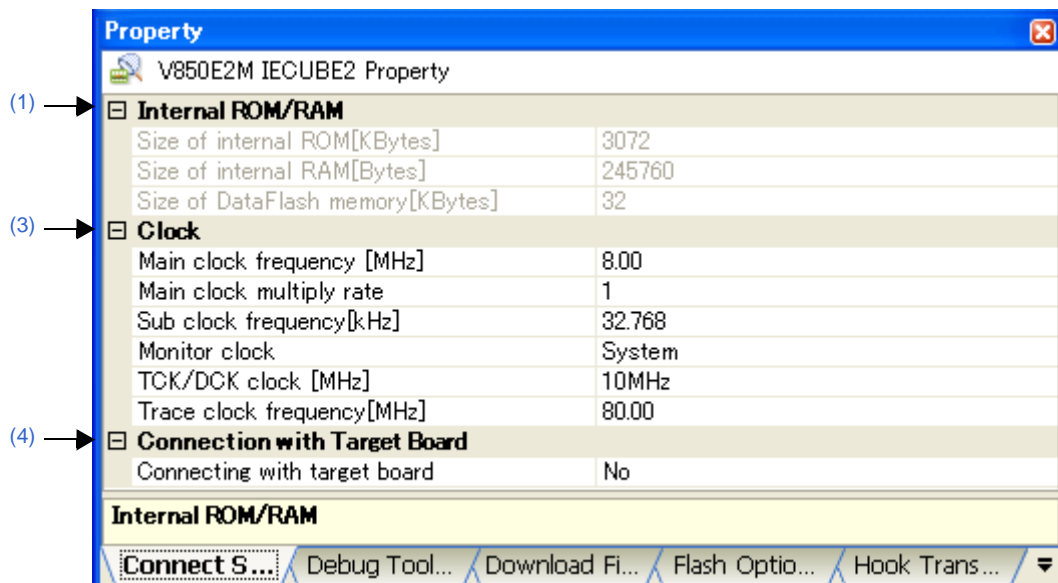
Figure A-8.   Property Panel: [Connect Settings] Tab [MINICUBE[V850E1][V850ES]]
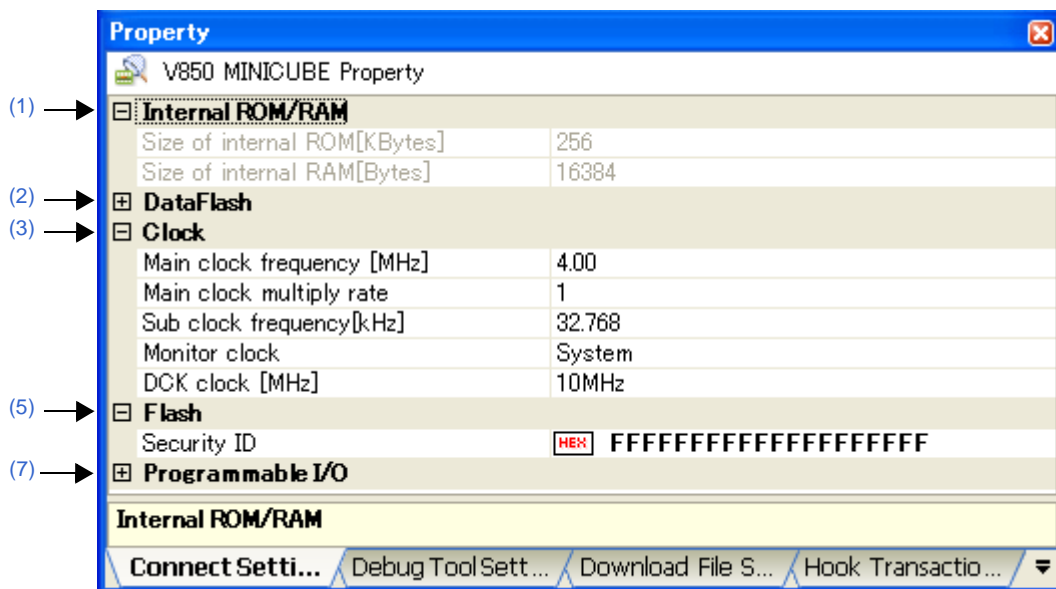
Figure A-9.   Property Panel: [Connect Settings] Tab [MINICUBE[V850E2M]]
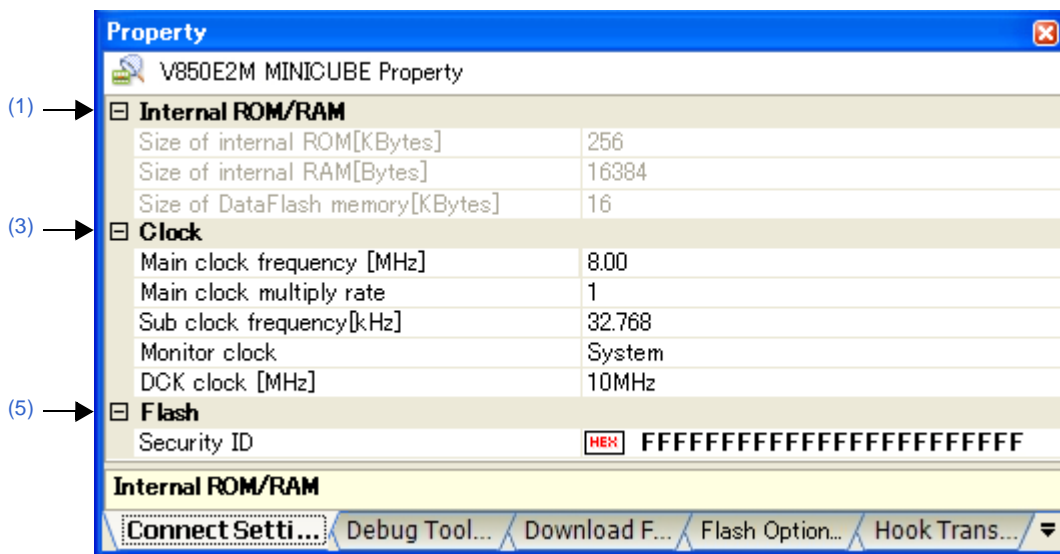
Figure A-10.   Property Panel: [Connect Settings] Tab [MINICUBE2]
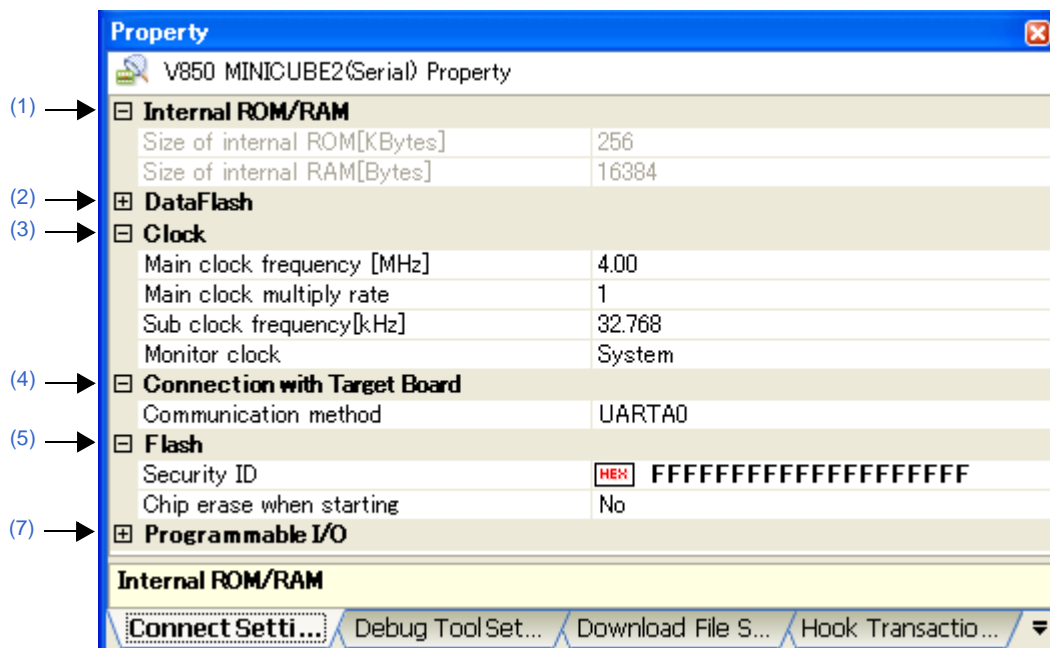
Figure A-11.   Property Panel: [Connect Settings] Tab [E1 (Serial)]
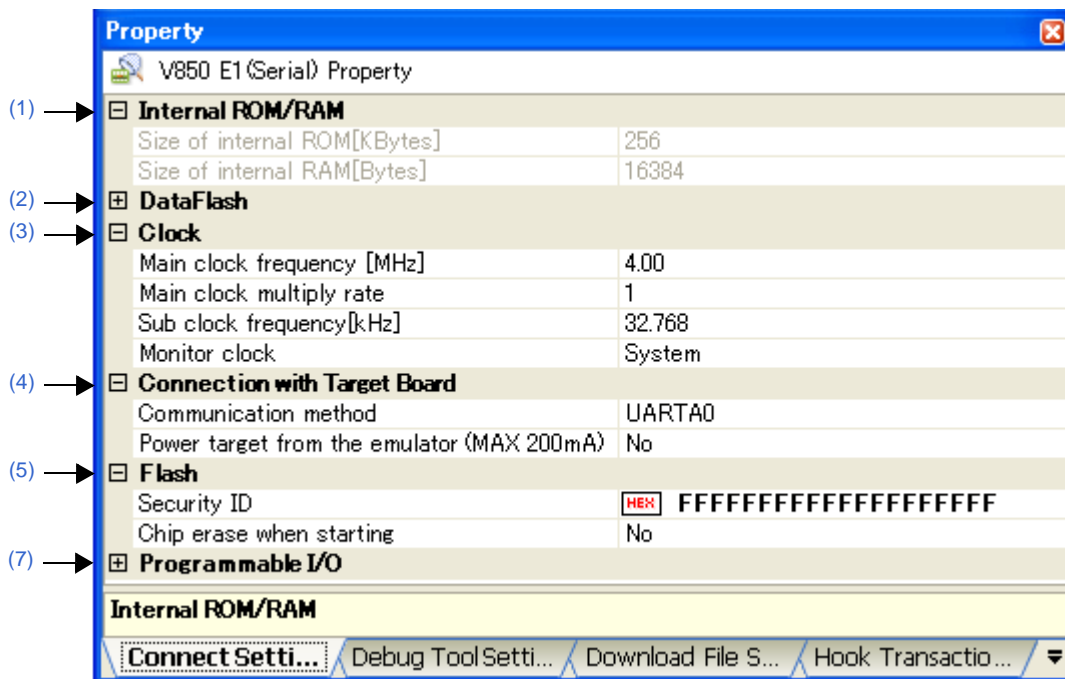
Figure A-12. Property Panel: [Connect Settings] Tab [E1 (JTAG)[V850E1][V850ES]]
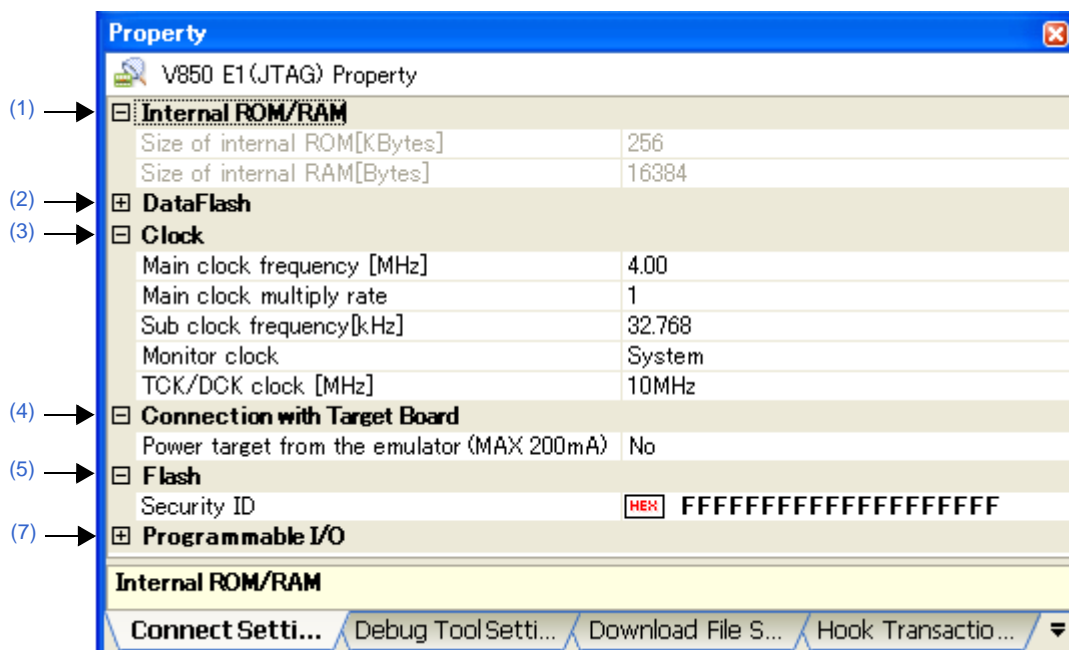
Figure A-13. Property Panel: [Connect Settings] Tab [E1 (JTAG)[V850E2M]]
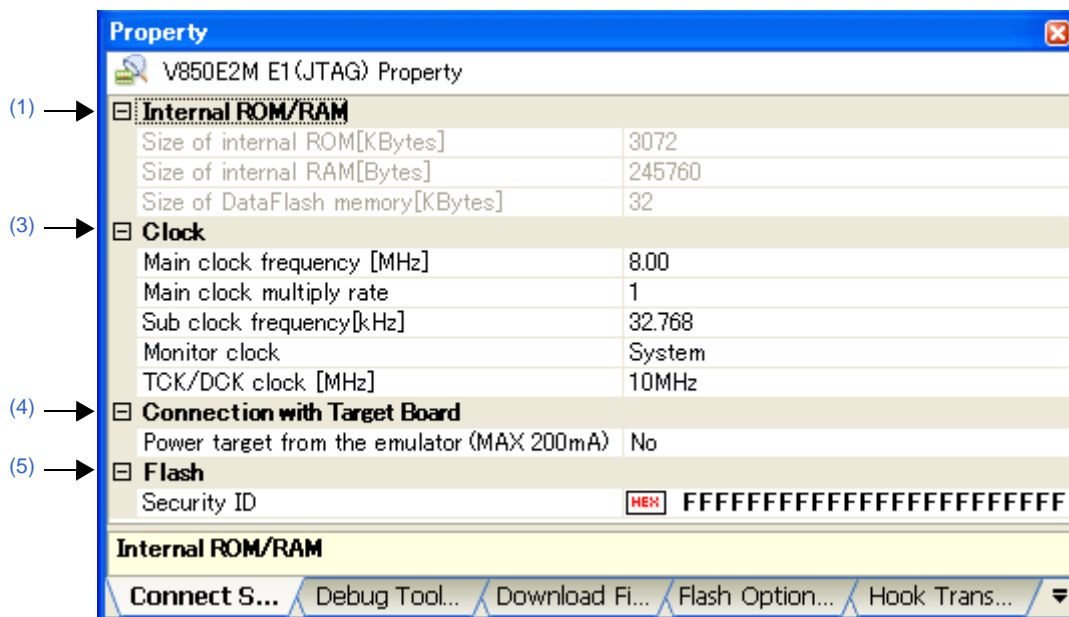
Figure A-14. Property Panel: [Connect Settings] Tab [Simulator[V850E1][V850ES]]
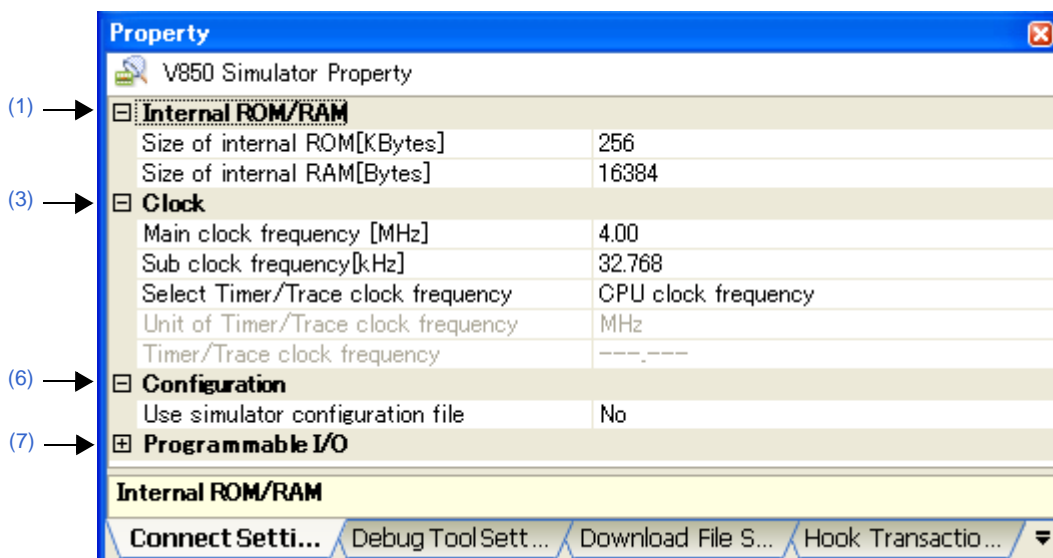
Figure A-15. Property Panel: [Connect Settings] Tab [Simulator[V850E2M]]
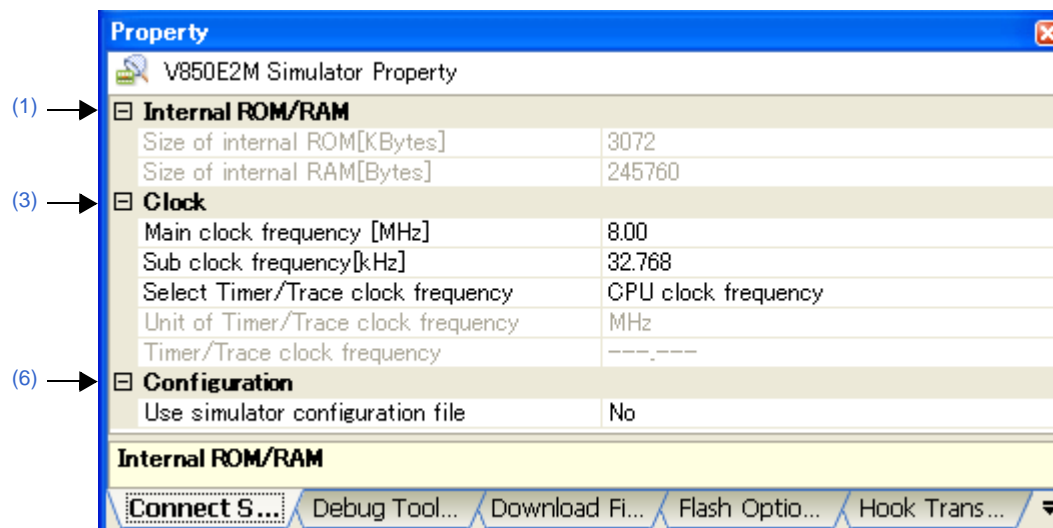
[Description of each category]

(1) [Internal ROM/RAM]

The detailed information on internal ROM/RAM is displayed and its configuration can be changed.

Cautions 1. You should be careful not to overlap the area with other memory mapping area.

2. [IECUBE]

CPU reset is generated automatically if you change the setting of the property in this category.

Size of internal ROM[KBytes]	Display and change the internal ROM size to emulate (simulate).	
	Default	- When the selected microcontroller is not a ROMless product Internal ROM size of the selected microcontroller[Note 1] - When the selected microcontroller is a ROMless product 0
	Modifying	**[IECUBE][Simulator[V850E1][V850ES]]** Select from the drop-down list[Note 2]. **[IECUBE2][MINICUBE][MINICUBE2][E1][Simulator[V850E2M]]** Changes not allowed
	Available values	**[IECUBE]** 0, 8, 32, 64, 128, 256, 512, 1024, *1024 + VSB Flash ROM size* (only when the selected microcontroller incorporates VSB Flash ROM) (unit: Kbytes) **[Simulator[V850E1][V850ES]]** 0, 8, 32, 64, 128, 256, 512, 1024 (unit: Kbytes)
Size of internal RAM[Bytes]	Display and change the internal RAM size to emulate (simulate).	
	Default	*Internal RAM size of the selected microcontroller*[Note 3]
	Modifying	**[IECUBE][Simulator[V850E1][V850ES]]** Select from the drop-down list. **[IECUBE][MINICUBE][MINICUBE2][E1][Simulator[V850E2M]]** Changes not allowed
	Available values	**[IECUBE]** 4096, 12288, 28672, 32768, 61440, *61440 + VSB RAM size* (only when the selected microcontroller incorporates VSB RAM) (unit: bytes) **[Simulator[V850E1][V850ES]]** 4096, 12288, 28672, 32768, 61440 (unit: bytes)
Size of DataFlash memory[KBytes] **[V850E2M]** (Other than Simulator)	Display the size of the data flash memory area of the selected microcontroller.	
	Default	*Data flash memory size of the selected microcontroller*
	Modifying	Changes not allowed

Notes 1. [V850E1]

When the selected microcontroller incorporates VSB Flash ROM, it displays the value including the size of VSB Flash ROM.

2. When the selected microcontroller is a ROMless product, the property value cannot be changed.

3. [V850E1]

When the selected microcontroller incorporates VSB RAM, it displays the value including the size of RAM.

(2) [DataFlash] [V850E1][V850ES] (except Simulator)

The detailed information on the data flash memory is displayed and its configuration can be changed.

Note that this category appears only when the selected microcontroller incorporates the data flash memory.

Use DataFlash memory area	Specify whether to use the data flash memory.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Uses the data flash memory.
		No	Does not use the data flash memory.
Size of DataFlash memory[KBytes]	Display the size of the data flash memory area of the selected microcontroller. This property appears only when the [Use DataFlash memory area] property is set to [Yes].		
	Default	*Data flash memory size of the selected microcontroller*	
	Modifying	Changes not allowed	
Chip select	Select the chip select used for mapping of the data flash memory. This property appears only when the [Use DataFlash memory area] property is set to [Yes].		
	Default	*Chip select of the selected microcontroller*	

	Modifying	**[IECUBE]**
		 - **[V850ES]**
		   Select from the drop-down list.
		 - **[V850E1]**
		   Changes not allowed
		[MINICUBE][MINICUBE2][E1]
		   Changes not allowed	

	Available values	*Chip select values that can be used by the selected microcontroller*[Note].	

Note   The selectable chip select values differ depending on the selected microcontroller.

(3) [Clock]

The detailed information on clocks is displayed and its configuration can be changed.

Main clock frequency[MHz]	Specify the main clock frequency (before multiplication) in MHz unit.	
	Default	- **[V850E1][V850ES]** 4.00 - **[V850E2M]** 8.00
	Modifying	Select from the drop-down list or directly enter from the keyboard.

	Available values	- Either one of the following from the drop-down list
		[V850E1][V850ES]
		    [IECUBE][MINICUBE][MINICUBE2][E1]
		    2.50, 4.00, 5.00, 6.00, 8.00, 13.50, 18.00 (unit: MHz)
		    [Simulator]
		    1.00, 2.00, 3.00, 3.57, 4.00, 4.19, 4.91, 5.00, 6.00, 8.00, 8.38, 10.00, 12.00, 16.00, 20.00, 25.00, 30.00, 32.00, 33.33, 34.00, 40.00, 48.00, 50.00, 64.00, 80.00 (unit: MHz)
		[V850E2M]
		    [IECUBE2][MINICUBE][E1 (JTAG)]
		    4.00, 5.00, 7.20, 8.00, 9.60, 10.00, 13.50, 16.00 (unit: MHz)
		    [Simulator]
		    1.00, 2.00, 3.00, 3.57, 4.00, 4.19, 4.91, 5.00, 6.00, 7.20, 8.00, 8.38, 9.60, 10.00, 12.00, 16.00, 20.00, 25.00, 30.00, 32.00, 33.33, 34.00, 40.00, 48.00, 50.00, 64.00, 80.00 (unit: MHz)
		- Directly enter the numbers ranged below
		0.001 to 999.999 (unit: MHz)

Main clock multiply rate **[IECUBE]** **[IECUBE2]** **[MINICUBE]** **[MINICUBE2]** **[E1]**	Specify the main clock frequency multiplier.	
	Default	1
	Modifying	Select from the drop-down list or directly enter from the keyboard.
	Available values	- Either one of the following from the drop-down list **[V850E1][V850ES]** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 **[V850E2M]** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20 - Directly enter the numbers ranged below Integer number between 1 and 99
Sub clock frequency[kHz]	Specify the sub clock frequency in kHz unit. This property appears only when the selected microcontroller supports a sub clock.	
	Default	**[IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]** *The minimum usable sub clock frequency* **[Simulator]** 32.768
	Modifying	Select from the drop-down list or directly enter from the keyboard.
	Available values	- Either one of the following from the drop-down list **[IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]** Sub clock frequency values that can be used by the selected microcontroller[Note] **[Simulator]** 32.768 - Directly enter the numbers ranged below 0.001 to 999.999 (unit: kHz)

Monitor clock [IECUBE] [IECUBE2] [MINICUBE] [MINICUBE2] [E1]	Specify a clock for monitor programs to operate while the program is stopped. This property appears only when the selected microcontroller supports a sub clock.		
	Default	System	
	Modifying	Select from the drop-down list.	
	Available values	System	Operates with main clock.
		User	Operates with the clock that the program specified.
TCK/DCK clock[MHz] [IECUBE2] [MINICUBE] [E1 (JTAG)]	Specify a clock supplied to the on-chip debug unit (DCU). Usually, however, [10MHz] must be selected. When [20MHz] is specified, it may not be possible to connect to the debug tool.		
	Default	10MHz	
	Modifying	Select from the drop-down list. Note that changes can be made only while disconnecting from the debug tool.	
	Available values	10MHz, 20MHz	
Trace clock frequency[MHz] [IECUBE2]	Specify the clock frequency for using trace function.		
	Default	80.00	
	Modifying	Directly enter from the keyboard.	
	Available values	1 to 999.999 (unit: MHz)	
Select Timer/Trace clock frequency [Simulator]	Specify the clock frequency for using timer/trace function.		
	Default	CPU clock frequency	
	Modifying	Select from the drop-down list.	
	Available values	CPU clock frequency	Uses the CPU clock frequency.
		Specify clock frequency	Specifies an arbitrary frequency (property items to specify become valid in the lower area).
Unit of Timer/Trace clock frequency [Simulator]	Specify the unit of the clock frequency for timer/trace. This property appears only when the [Select Timer/Trace clock frequency] property is set to [Specify clock frequency].		
	Default	MHz	
	Modifying	Select from the drop-down list.	
	Available values	MHz	The unit of the frequency is in MHz.
		KHz	The unit of the frequency is in kHz.

Timer/Trace clock frequency [Simulator]	The operation of this property differs depending on the specification with the [Select Timer/Trace clock frequency] property.

	- When [Specify clock frequency] is specified
	  Specify the clock frequency for timer/trace.
	- When [CPU clock frequency] is specified, displays the following (changes not allowed)
	  While disconnecting from the debug tool: [---_---]
	  While connecting to the debug tool: [*CPU clock frequency*]

	Default	- **[V850E1][V850ES]**
		  4.00
		- **[V850E2M]**
		  8.00

	Modifying	Directly enter from the keyboard.
	Available values	1 kHz to 999.999 MHz Unit is depending on the specification with the [Unit of Timer/Trace clock frequency] property.

Note The usable sub clock frequency values differ depending on the selected microcontroller.
If the frequency value which can be use is fixed, the drop-down list does not appear.

(4) [Connection with Target Board] [IECUBE][IECUBE2][MINICUBE2][E1]
The detailed information on the connection to the target board is displayed and its configuration can be changed.

Connecting with target board [IECUBE] [IECUBE2]	Specify if the target board is connected to IECUBE/IECUBE2 or not.	
	Default	No	
	Modifying	Select from the drop-down list. Note that changes can be made only while disconnecting from the debug tool.	
	Available values	Yes	Target board is connected.
		No	Target board is not connected.
Communication method [MINICUBE2] [E1 (Serial)]	Specify the port to which to connect MINICUBE2/E1 (Serial) when performing serial communication between MINICUBE2/E1 (Serial) and the device in the target system.	
	Default	Depends on the selected microcontroller.	
	Modifying	Select from the drop-down list. Note that changes can be made only while disconnecting from the debug tool.	
	Available values	Depends on the selected microcontroller[Note].	
Power target from the emulator (MAX 200mA) [E1]	Specify whether to supply power to the target system from E1.	
	Default	No	
	Modifying	Select from the drop-down list. Note that changes can be made only while disconnecting from the debug tool.	
	Available values	Yes	Supplies power to the target system.
		No	Does not supply power to the target system.

Supply voltage **[E1]**	Specify the power voltage supplied to the target system. This property appears only when the [Power target from the emulator (MAX 200mA)] property is set to [Yes].	
	Default	3.3V
	Modifying	Select from the drop-down list. Note that changes can be made only while disconnecting from the debug tool.
	Available values	3.3V, 5.0V

Note The type of the port in the drop-down list differs depending on the selected microcontroller.

(5) [Flash] [MINICUBE][MINICUBE2][E1]

The detailed information on the flash memory writing is displayed and its configuration can be changed.

Security ID	Specify a security ID for reading codes in the internal ROM or internal flash memoryNote. This property appears only when the selected microcontroller supports the ROM security function (on-chip debug security ID) for flash memory.		

	Default	- **[V850E1][V850ES]**
		  FFFFFFFFFFFFFFFFFFFF
		- **[V850E2M]**
		  FFFFFFFFFFFFFFFFFFFFFFFF	

	Modifying	Directly enter from the keyboard. Note that changes can be made only while disconnecting from the debug tool.	

	Available values	- **[V850E1][V850ES]**
		  20 digits hexadecimal number (10 bytes)
		- **[V850E2M]**
		  24 digits hexadecimal number (12 bytes)	

Chip erase when starting **[MINICUBE2]** **[E1 (Serial)]**	Specify whether to erase the chip when connecting to the debug tool.		
	Default	No	
	Modifying	Select from the drop-down list. Note that changes can be made only while disconnecting from the debug tool.	
	Available values	Yes	Erases the chip when connecting to the debug tool. Note that after connecting to the debug tool, this property automatically selects [No].
		No	Does not erase the chip when connecting to the debug tool.

Note When you change the setting of the property, a message appears prompting you to select whether or not to change the security ID of the build tool (except for the V850E2M core).
For details on the on-chip debug security ID, see MINICUBE/MINICUBE2/E1 User's Manual

(6) [Configuration] [Simulator]

The detailed information when customizing the simulator is displayed and its configuration can be changed.

Use simulator configuration file	Specify whether to use the simulator configuration file to perform user customization (adding of user models) of the simulator.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Uses the simulator configuration file.
		No	Does not use the simulator configuration file.
Simulator configuration file	Specify the simulator configuration file to be use. This property appears only when the [Use simulator configuration file] property is set to [Yes].		
	Default	*Blank*	
	Modifying	Directly enter from the keyboard, or specify with the Select Simulator Configuration File dialog box [Simulator] opened by clicking the [...] button. Note that changes can be made only while disconnecting from the debug tool.	

(7) [Programmable I/O] [V850E1][V850ES]

The detailed information on the programmable I/O area is displayed and its configuration can be changed. Change this setting to temporarily change values while debugging. To set a common value for the project, in the Property panel of the build tool, on the [Common Options] tab, in the [Device] category, set the value in [Programmable I/O area start address] property.

Use Programmable I/O area	Specify whether to use the programmable I/O area.		
	Default	No	
	Modifying	Select from the drop-down list. Note that changes can be made only when the selected microcontroller supports the programmable I/O area.	
	Available values	Yes	Uses the programmable I/O area.
		No	Does not use the programmable I/O area.
Programmable I/O area start address	The address is aligned to 16 Kbytes. This property appears only when the [Use Programmable I/O area] property is set to [Yes]. Specify the start address of the programmable I/O area.		
	Default	0	
	Modifying	Directly enter from the keyboard.	
	Available values	Address expressions within the address range of the programmable I/O area.	

[Debug Tool Settings] tab

This tab is used to display the detailed information categorized by the following and the configuration can be changed.

(1) [Memory]

(2) [Access Memory While Running]

(3) [Set Event While Running] [IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]

(4) [Break]

(5) [Fail-safe Break] [IECUBE]

(6) [Trace] [IECUBE][IECUBE2][Simulator]

(7) [Timer] [IECUBE][IECUBE2][Simulator]

(8) [Coverage] [IECUBE][IECUBE2][Simulator]

(9) [Mask for Input Signal] [IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]

(10) [External Flash Memory Download] [IECUBE2][MINICUBE][E1 (JTAG)]

(11) [Simulator GUI] [Simulator]

Figure A-16.   Property Panel: [Debug Tool Settings] Tab [IECUBE]
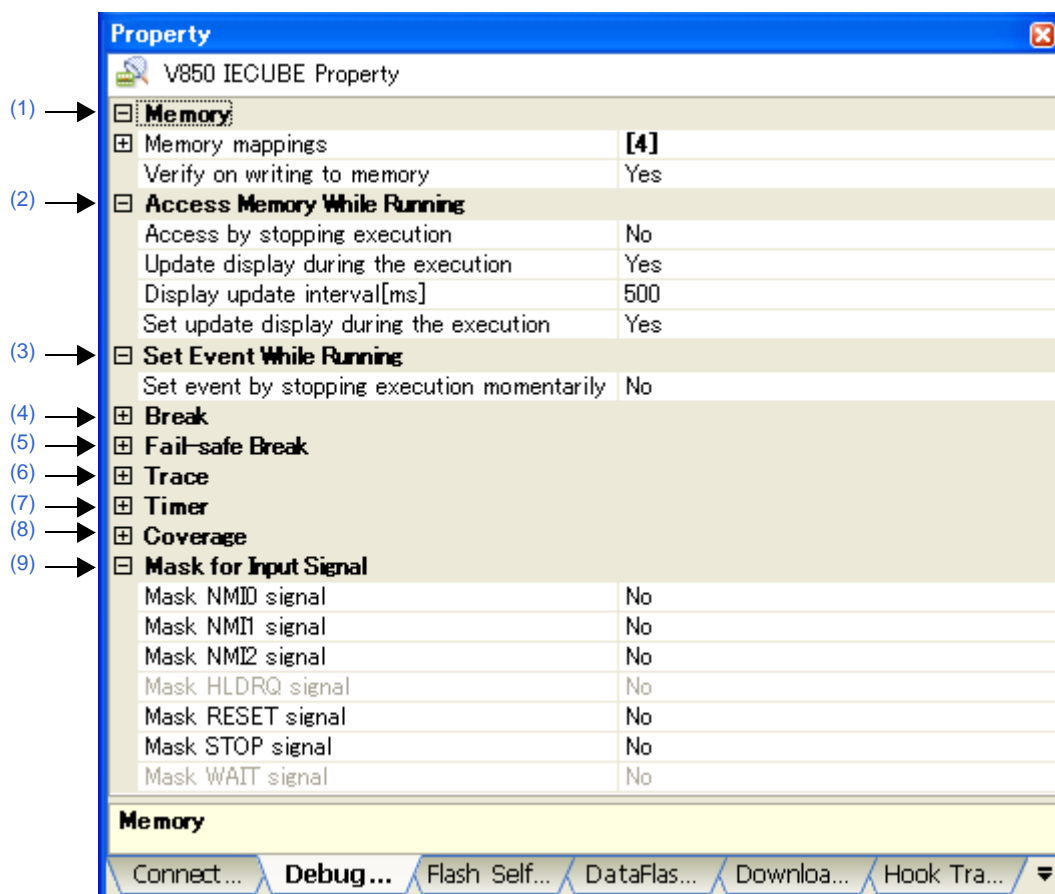
Figure A-17.   Property Panel: [Debug Tool Settings] Tab [IECUBE2]
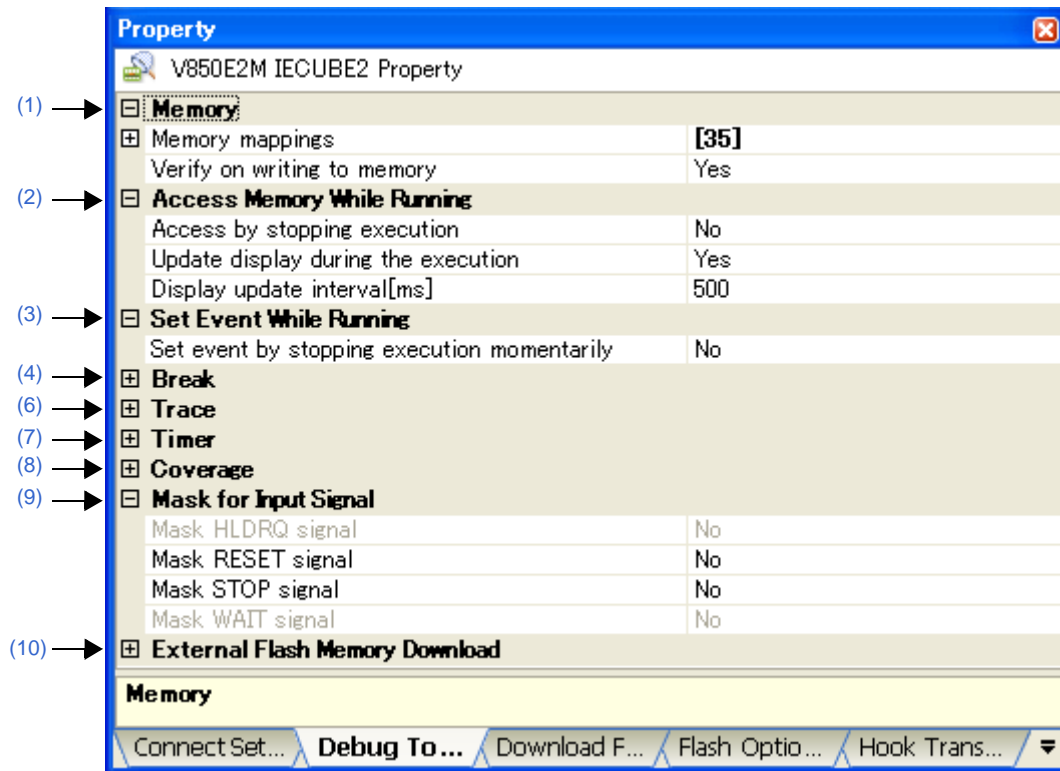
Figure A-18.   Property Panel: [Debug Tool Settings] Tab [MINICUBE[V850E1][V850ES]]
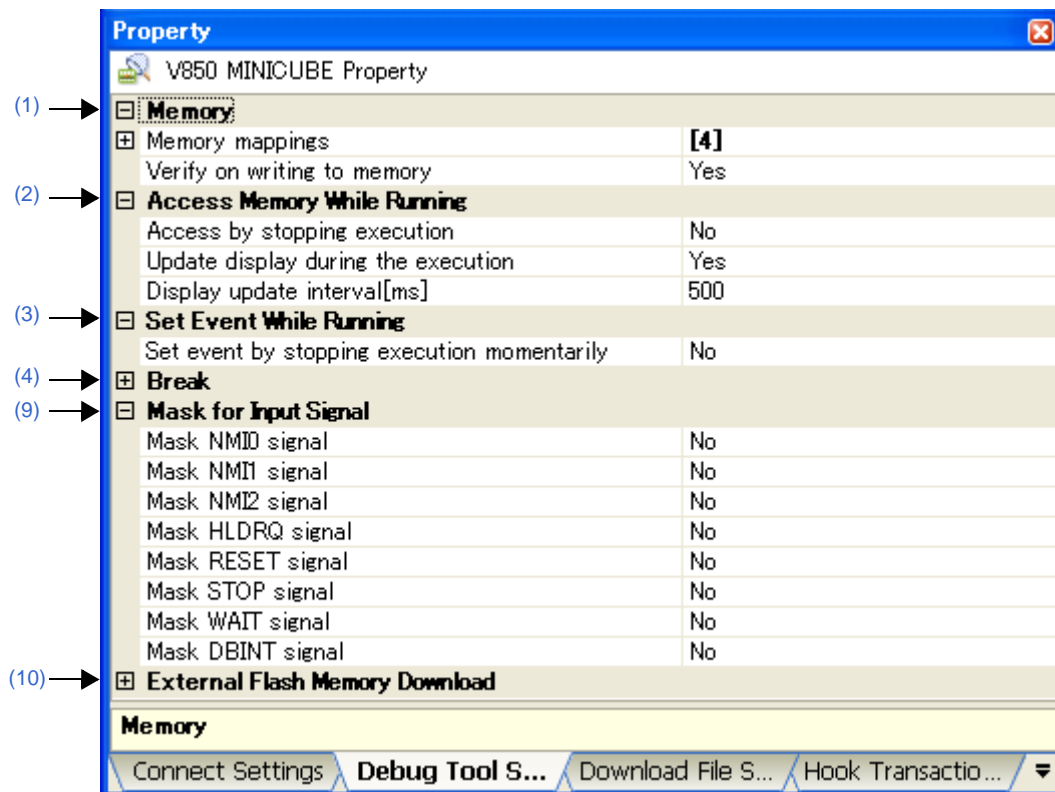
Figure A-19.   Property Panel: [Debug Tool Settings] Tab [MINICUBE[V850E2M]]
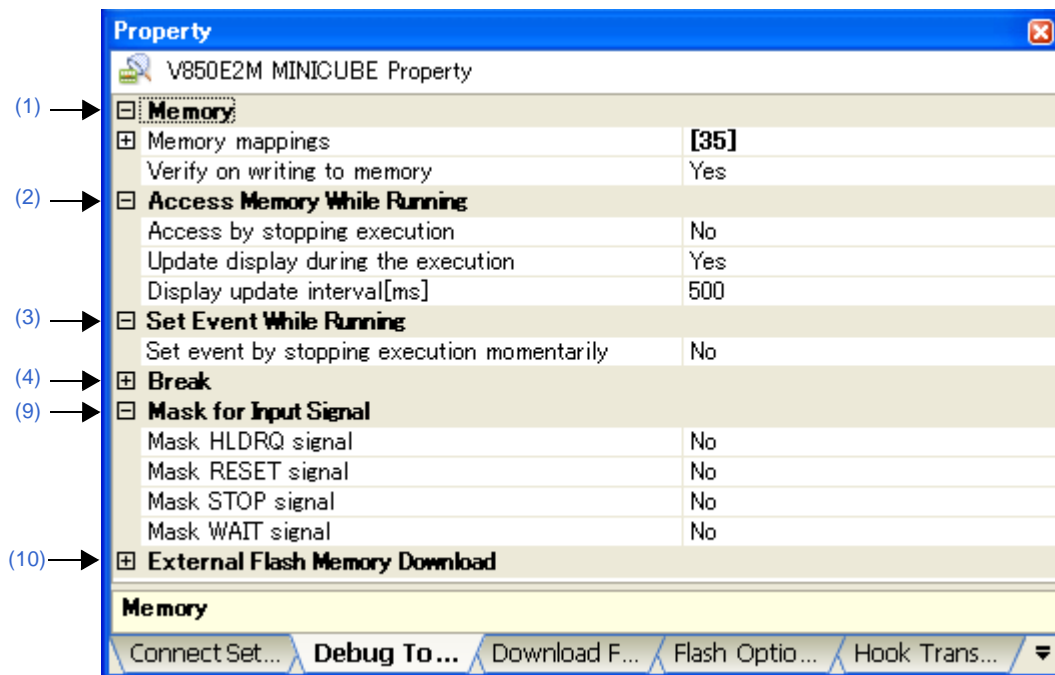
Figure A-20.   Property Panel: [Debug Tool Settings] Tab [MINICUBE2]
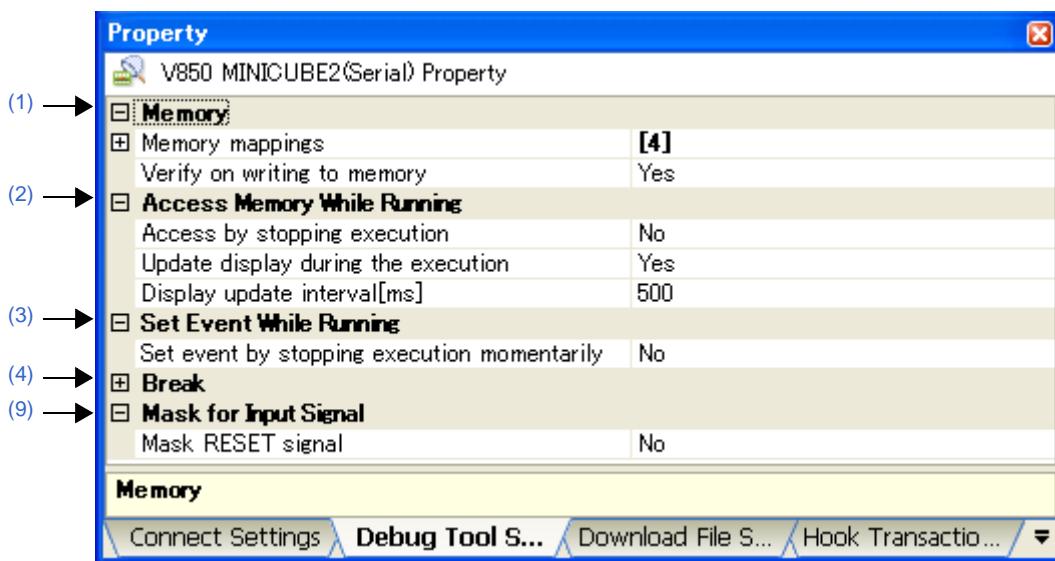
Figure A-21.   Property Panel: [Debug Tool Settings] Tab [E1 (Serial)]
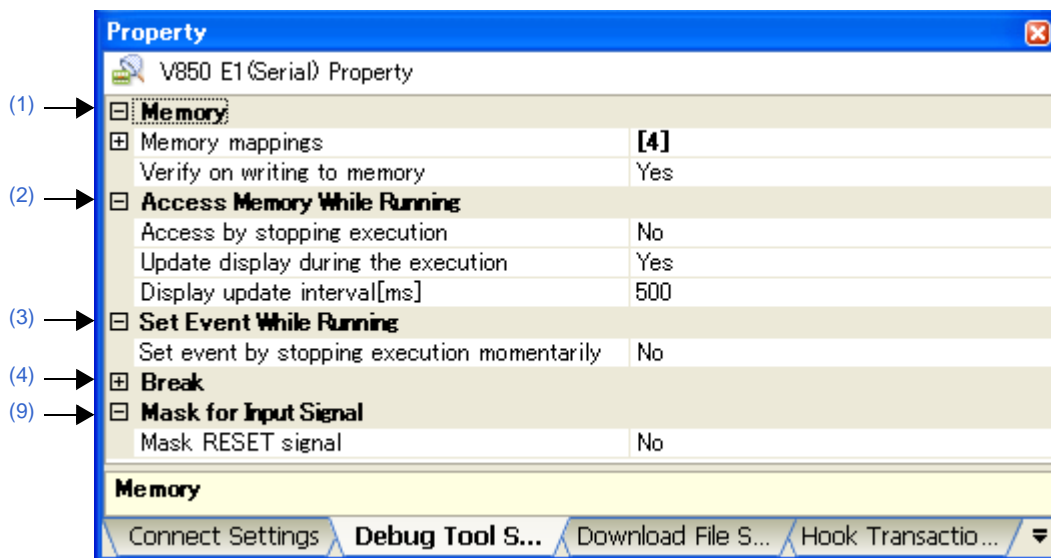
Figure A-22.   Property Panel: [Debug Tool Settings] Tab [E1 (JTAG)][V850E1][V850ES]]
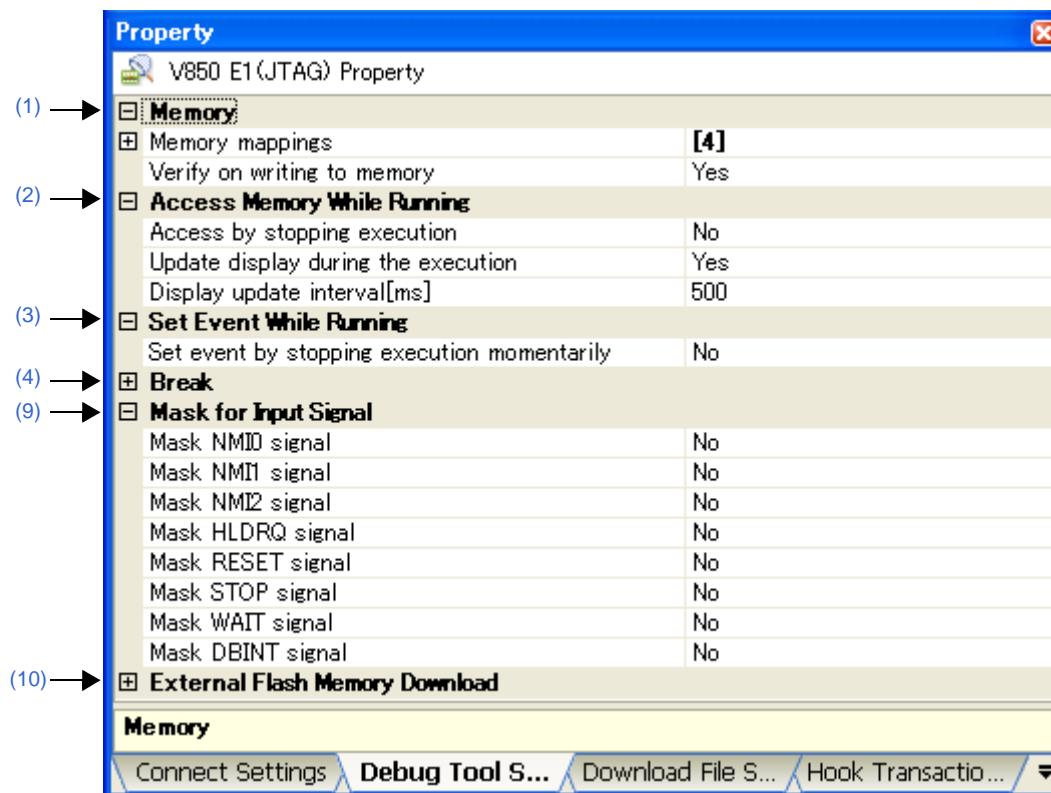
Figure A-23.   Property Panel: [Debug Tool Settings] Tab [E1 (JTAG)[V850E2M]]
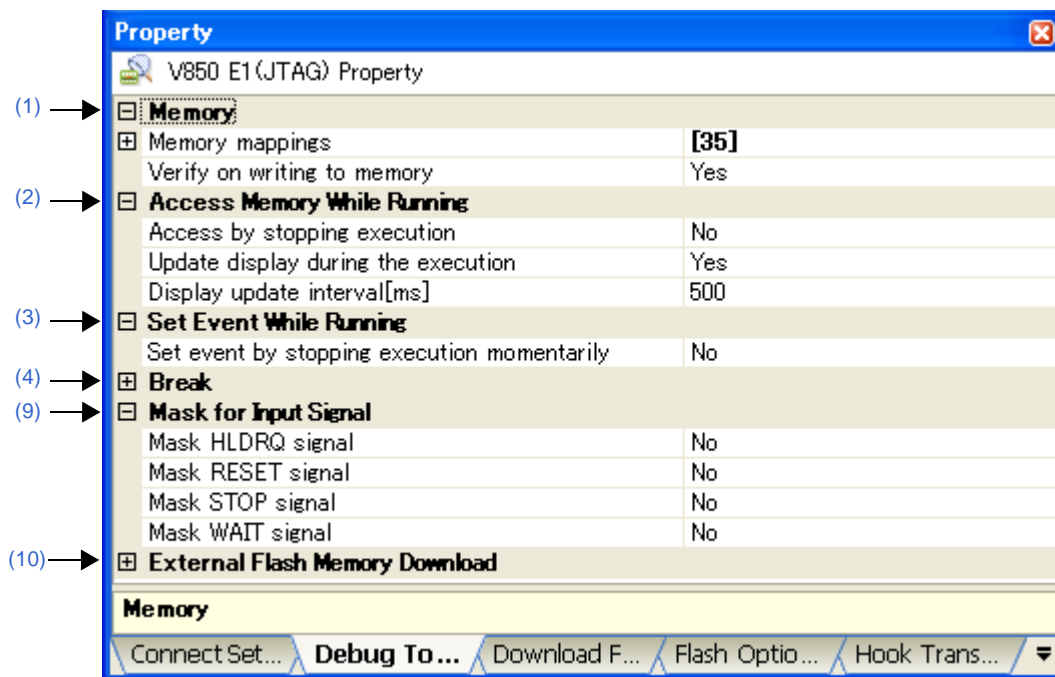
Figure A-24.   Property Panel: [Debug Tool Settings] Tab [Simulator[V850E1][V850ES]]
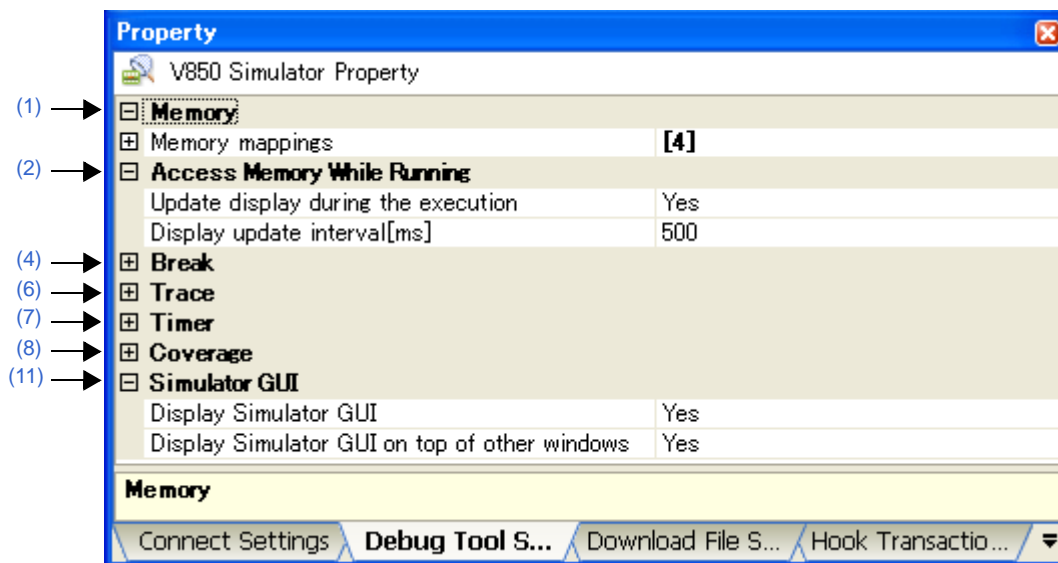
Figure A-25.   Property Panel: [Debug Tool Settings] Tab [Simulator[V850E2M]]
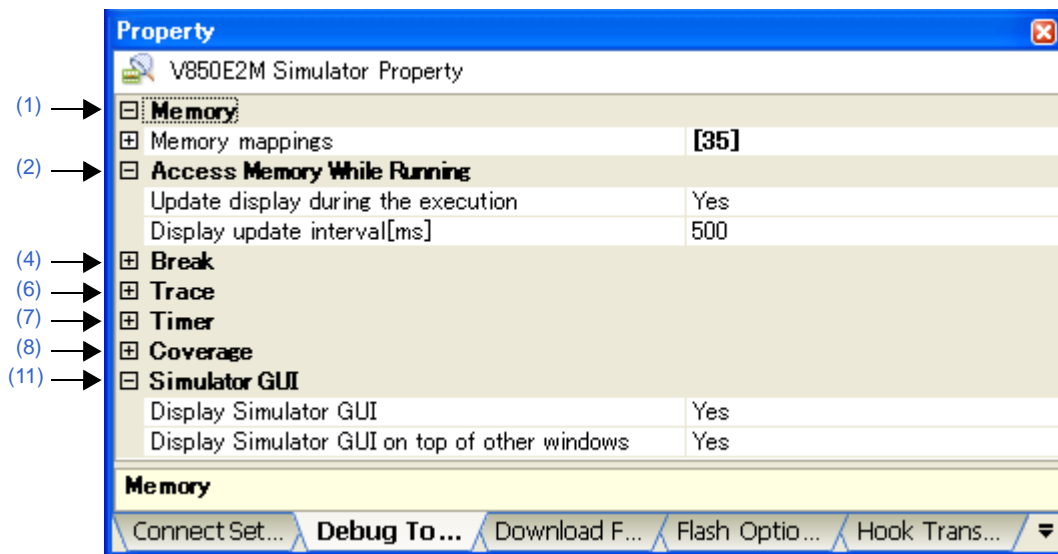
[Description of each category]

(1) [Memory]

The detailed information on memories is displayed and its configuration can be changed.

For details on memory types that are displayed, see the "Memory Mapping dialog box".

Memory mappings	Current memory mapping status is displayed by the types of memory area[Note].	
	Default	[*Total number of the memory mapping types*]
	Modifying	Specify with the Memory Mapping dialog box. The Memory Mapping dialog box is opened when clicking the [...] button appears at right by selecting the mapping value (you cannot change the mapping value on this panel).
	Display Contents	Displays the memory mapping status by the types of memory area. The following detailed information is displayed by clicking the "+" mark of each memory type. - Memory type - Start address - End address - Access width[bits]
Verify on writing to memory **[IECUBE]** **[IECUBE2]** **[MINICUBE]** **[MINICUBE2]** **[E1]**	Specify whether to perform a verify check when data has been written to memory. The timing to execute the verify check is as follows: - At when the program is downloaded - At when variables and data are changed in the Watch panel/Memory panel	
	Default	Yes
	Modifying	Select from the drop-down list.
	Available values	Yes	Executes the verify check.
		No	Does not execute the verify check.

Note   The type is of the memory mapping area registered in the device file.

(2) [Access Memory While Running]

The detailed information on memory accesses while executing a program (real-time display update function: see "(4) Display/modify the memory contents during program execution") is displayed and its configuration can be changed.

Access by stopping execution **[IECUBE]** **[IECUBE2]** **[MINICUBE]** **[MINICUBE2]** **[E1]**	**[IECUBE][IECUBE2]** 　Specify whether to allow access to the memory area where cannot be accessed while executing a program (target memory area//O register area). **[MINICUBE][MINICUBE2][E1]** 　Specify whether to allow access to the memory area while executing a program. Note that this property does not appear when the selected microcontroller version supports multi-core.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Temporary stops execution and read/write.
		No	Does not access to the memory during program execution.
Update display during the execution	Specify whether to update the display in the Watch panel/Memory panel during a program execution. **[IECUBE]** 　This property becomes invalid when the [Use for trace data] property in the [Trace] category is set to [Trace].		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Updates the display during program execution.
		No	Does not update the display during program execution.
Display update interval[ms]	Specify the interval in 100ms unit to update the contents in the Watch panel/Memory panel display while executing a program. This property appears only when the [Update display during the execution] property is set to [Yes].		
	Default	500	
	Modifying	Directly enter from the keyboard.	
	Available values	Integer number between 100 and 65500 (rounding up the fractions less than 100 ms).	
Set update display during the execution automatically **[IECUBE]**	Update the contents of the display while executing the program by setting the area shown in the Watch panel/Memory panel automatically as much as possible to the target range for the real-time display update function. This property appears only when the [Update display during the execution] property is set to [Yes].		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Automatically sets the real-time display update function.
		No	Does not set the real-time display update function.

Access by release HALT mode **[IECUBE]** **[IECUBE2]** **[MINICUBE]** **[E1 (JTAG)]**	Specify whether to allow access to the memory area/CPU registers/I/O registers by releasing the HALT mode. This property appears only when the [Access by stopping execution] property is set to [Yes].		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Releases the HALT mode to access them.
		No	Does not release the HALT mode to access them.

(3) [Set Event While Running] [IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]

The detailed information on the function of the event setting during program execution is displayed and its configuration can be changed.

Set event by stopping execution momentarily	Specify whether to forcibly pause the execution for events that cannot be set while executing the program or operating the tracer/timer. For details on the event types that are affected by this property, see "(2) Event types that can be set and deleted during execution".		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Sets these events by stopping the program execution or the tracer/timer operation momentarily.
		No	Does not allow to set these events during program execution or the tracer/timer operation.

(4) [Break]

The detailed information on break functions is displayed and its configuration can be changed.

First using type of breakpoint **[IECUBE]** **[IECUBE2]** **[MINICUBE]** **[MINICUBE2]** **[E1]**	Specify the type of the breakpoint to use with priority when setting it at the source line or the execution address with a one click operation of the mouse in the Editor panel/Disassemble panel.		
	Default	Software break	
	Modifying	Select from the drop-down list.	
	Available values	Software break	Sets software breakpoint with priority.
		Hardware break	Sets hardware breakpoint with priority.
Stop emulation of peripherals when stopping **[IECUBE]** **[IECUBE2]** **[MINICUBE]** **[E1 (JTAG)]**	Specify whether to terminate the peripheral emulation while stopping the program execution.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Terminates the peripheral emulation.
		No	Does not terminate the peripheral emulation.

Use open break function **[IECUBE] [IECUBE2] [MINICUBE] [E1 (JTAG)]**	Specify whether to use the open break function.		
	Default	No(Output signal)	
	Modifying	Select from the drop-down list.	
	Available values	Yes(Hi-Z)	The open break target pin becomes the Hi-Z state after the CPU is stopped.
		No(Output signal)	The open break target pin outputs the signal even after the CPU is stopped.
Execute instruction at breakpoint when break **[Simulator]**	Specify the timing to stop the program execution by breakpoints either after or before the execution of the instruction at the breakpoint.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Breaks after executing the instruction[Note].
		No	Breaks before executing the instruction.

Note When [Yes] is selected, all of the action events currently being set are handled as Hardware Break events (see "2.14 Set an Action into Programs").

(5) [Fail-safe Break] [IECUBE]

The detailed information on fail-safe break functions is displayed and its configuration can be changed.

Stop when wrote to Internal ROM	Specify whether to stop the execution right after writing to the internal ROM[Note]. This property is invalid when the selected microcontroller is a ROMless product.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Stops right after writing.
		No	Does not stop even after writing.
Stop when accessed to non-mapping Internal ROM	Specify whether to stop the execution right after accessing to the non-mapping area of the internal ROM[Note]. This property is invalid when the selected microcontroller is a ROMless product.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Stops right after accessing.
		No	Does not stop even after accessing.
Stop when accessed to non-mapping Internal RAM	Specify whether to stop the execution right after accessing to the non-mapping area of the internal RAM[Note].		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Stops right after accessing.
		No	Does not stop even after accessing.

Verify when wrote to Read-only Internal RAM	Specify whether to perform a verify check that detects a write operation to the write protected area in the internal RAM when the program execution is stopped[Note].		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Outputs messages when write operations have been detected.
		No	Does not check a write operation.
Stop when read from read protected I/O Register	Specify whether to stop the execution right after reading from the read protected area in the I/O register area.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Stops right after reading.
		No	Does not stop even after reading.
Stop when wrote to write protected I/O Register	Specify whether to stop the execution right after writing to the write protected area in the I/O register area.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Stops right after writing.
		No	Does not stop even after writing.
Stop when accessed to non-mapping I/O Register	Specify whether to stop the execution right after accessing to the non-mapping area in the I/O register area.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Stops right after accessing.
		No	Does not stop even after accessing.
Stop when wrote to write protected External memory	Specify whether to stop the execution right after writing to the write protected external memory.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Stops right after writing.
		No	Does not stop even after writing.
Stop when accessed to non-mapping External Memory	Specify whether to stop the execution right after accessing to the non-mapping area of the external memory.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Stops right after accessing.
		No	Does not stop even after accessing.

Note CPU reset is generated automatically if you change the setting of the property.

(6) [Trace] [IECUBE][IECUBE2][Simulator]

The detailed information on trace functions is displayed and its configuration can be changed.

Cautions 1.   [IECUBE]

Some of the trace function, the real-time display update function (RRM function) and the coverage function are used on a mutually exclusive basis.

2.   [IECUBE2]

Some of the trace function, the timer function (except for the Run-Break timer) and the coverage function are used on a mutually exclusive basis.

Note, however, that only the trace function can be used in this version.

Use for trace data [IECUBE] [IECUBE2]	[IECUBE] Some of the trace function, the real-time display update function (RRM function) and the coverage function are used on a mutually exclusive basis. Accordingly, specify a function used preferentially. [IECUBE2] Some of the trace function, the timer function (except for the Run-Break timer) and the coverage function are used on a mutually exclusive basis. Accordingly, specify a function used preferentially. See "2.11.1 Configure the trace operation" for details on this property.		
	Default	**[IECUBE]** RRM **[IECUBE2]** Trace	
	Modifying	Select from the drop-down list.	
	Available values **[IECUBE]**	Trace	Uses the trace function preferentially.
		RRM	Uses the real-time display update function (RRM function) preferentially.
		Coverage	Uses the coverage function preferentially.
	Available values **[IECUBE2]**	Trace	Uses the trace function preferentially.
		Timer	Uses the timer function preferentially.
		Coverage	Uses the coverage function preferentially.
Select trace data [IECUBE] [IECUBE2]	Select the type of the trace data to be acquired. This property appears only when the [Use for trace data] property is set to [Trace]. See "2.11.1 Configure the trace operation" for details on this property.		
	Default	Branch PC + Access Data	
	Modifying	Select from the drop-down list.	
	Available values	Branch PC	
		All PC **[IECUBE]**	
		Access Data	
		Branch PC + Access Data	
		All PC + Access Data **[IECUBE]**	
		Access PC + Access Data	
		Branch PC + Access PC + Access Data	

DMA is traced [IECUBE2]	Specify whether to perform the DMA trace (Direct Memory Access Trace).		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Performs the DMA trace.
		No	Does not perform the DMA trace.
Select DMA trace data [IECUBE2]	Select the type of the DMA trace data to be acquired. This property appears only when the [DMA is traced] property is set to [Yes]. See "2.11.1 Configure the trace operation" for details on this property.		
	Default	Access Type + Access Address + Data Size + Access Data	
	Modifying	Select from the drop-down list.	
	Available values	Access Type + Access Address + Data Size + Access Data	
		Access Type + Access Address + Data Size	
		Access Type + Channel Number	
		Access Type + Channel Number + Transfer Count	
		Access Type + Access Data + Channel Number + Transfer Count	
Trace priority [IECUBE] [IECUBE2]	Select which item should be given priority when collecting the trace data. This property appears only when the [Use for trace data] property is set to [Trace].		
	Default	Speed priority	
	Modifying	Select from the drop-down list.	
	Available values	Speed priority	Traces giving priority to the real-time performance.
		Data priority	Traces after stopping the execution pipeline of the CPU temporarily so that no data is missed.
Use trace function [Simulator]	Specify whether to use the trace function[Note 1].		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Uses trace functions.
		No	Does not use trace functions.
Clear trace memory before running	Specify whether to clear the trace memory before executing.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Clears the trace memory.
		No	Does not clear the trace memory.

Operation after trace memory is full	Specify the operation after the trace memory is full with the collected trace data.		
	Default	Non stop and overwrite to trace memory	
	Modifying	Select from the drop-down list.	
	Available values	Non stop and overwrite to trace memory	Continues overwriting the older trace data after the trace memory is full.
		Stop trace	Stops writing the trace data after the trace memory is full (the program execution will not be stopped).
		Stop	Stops writing the trace data after the trace memory is full and the program execution will be stopped.
Accumulate trace time **[Simulator]**	Specify whether to display the accumulated tracing time in the Trace panel [IECUBE][IECUBE2][Simulator].		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Displays the accumulated tracing time.
		No	Displays the trace time with differential value.
Rate of frequency division of trace time tag **[IECUBE]** **[Simulator]**	Specify the frequency division ratio of the counter to be used for time tag display (the [Time] item in the Trace panel [IECUBE][IECUBE2][Simulator]). Changing the frequency division ratio here changes the number of clocks necessary to count up a counter value which is displayed in the time tag.		
	Default	**[IECUBE]** 1/1(20ns/1.4min) **[Simulator]** 1/1	
	Modifying	Select from the drop-down list.	
	Available values	**[IECUBE]** 1/1(20ns/1.4min), 1/2(40ns/2.8min), 1/4(80ns/5.7min), 1/8(160ns/11.4min), 1/16(320ns/22.8min), 1/32(640ns/45.6min), 1/64(1280ns/1.5h), 1/128(2560ns/3.0h), 1/256(5120ns/6.0h), 1/512(10240ns/12.2h), 1/1024(20480ns/24.4h), 1/2048(40960ns/48.8h), 1/4096(81920ns/97.6h) (Values in "()" indicate the resolution, and the maximum measurement time when using a time tag counter of 32-bit and a external clock of 50 MHz). **[Simulator]** 1/1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256, 1/512, 1/1K, 1/4K, 1/8K, 1/16K, 1/64K, 1/256K, 1/1M	

Trace memory size[frames]	Specify the memory size for storing the trace data by the trace frame numbers[Note 2].	
	Default	**[IECUBE][IECUBE2]** 8K **[Simulator]** 4K
	Modifying	Select from the drop-down list.
	Available values	**[IECUBE]** 8K, 32K, 64K, 128K, 256K **[IECUBE2]** 8K, 32K, 64K, 128K, 256K, 512K **[Simulator]** 4K, 8K, 12K, 16K, 20K, 24K, 28K, 32K, 36K, 40K, 44K, 48K, 52K, 56K, 60K, 64K, 128K, 192K, 256K, 320K, 384K, 448K, 512K, 576K, 640K, 704K, 768K, 832K, 896K, 960K, 1M, 2M, 3M
Enable trace data complement **[IECUBE]** **[IECUBE2]**	Specify whether to perform complementary display when displaying the collected trace data[Note 3].	
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Performs complementary display of trace data.
		No	Does not perform complementary display of trace data.

Notes 1. This property is automatically set to [Yes] when selecting [Start Tracing]/[Stop Tracing] from the context menu in the Editor panel/Disassemble panel.

2. The trace frame is a unit for the trace data. Each fetch/write/read uses one trace frame.

3. Complement display of instructions between branch instructions that cannot be traced by hardware can be performed.
The trace memory is cleared automatically if you change the setting of this property.

(7) [Timer] [IECUBE][IECUBE2][Simulator]

The detailed information on timer functions is displayed and its configuration can be changed.

Caution [IECUBE2]

All of the properties in this category become invalid when the [Use for trace data] property in the [Trace] category is set to other than [Timer].

Use timer function **[Simulator]**	Specify whether to use the timer function.	
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Uses timer functions.
		No	Does not use timer functions.

Rate of frequency division of timer [IECUBE] [IECUBE2]	**[IECUBE]** Specify the frequency division ratio of the timer counter (32 bits, 50 MHz) used for timer measurement. **[IECUBE2]** Specify the frequency division ratio of the timer counter (32 bits, 200 MHz) used for timer measurement.		
	Default	**[IECUBE]** 1/1(20ns/1.4min) (Resolution/maximum measurement time are displayed in "()"). **[IECUBE2]** 1/1(5ns/21.4s) (Resolution/maximum measurement time are displayed in "()").	
	Modifying	Select from the drop-down list.	
	Available values	**[IECUBE]** 1/1(20ns/1.4min), 1/2(40ns/2.8min), 1/4(80ns/5.7min), 1/8(160ns/11.4min), 1/16(320ns/22.8min), 1/32(640ns/45.6min), 1/64(1280ns/1.5h), 1/128(2560ns/3.0h), 1/256(5120ns/6.0h), 1/512(10240ns/12.2h), 1/1024(20480ns/24.4h), 1/2048(40960ns/48.8h), 1/4096(81920ns/97.6h) **[IECUBE2]** 1/1(5ns/21.4s), 1/2(10ns/42.9s), 1/4(20ns/1.4min), 1/8(40ns/2.8min), 1/16(80ns/5.71min), 1/32(160ns/11.42min), 1/64(320ns/22.8min), 1/128(640ns/45.6min), 1/256(1280ns/1.5h), 1/512(2560ns/3.0.h), 1/1024(5120ns/6.0h), 1/2048(10240ns/ 12.2h), 1/4096(20480ns/24.4h	

(8) [Coverage] [IECUBE][IECUBE2][Simulator]

The detailed information on coverage functions is displayed and its configuration can be changed.

Cautions 1.   [IECUBE]
All of the properties in this category become invalid in any one of the following conditions:
- When no coverage board is mounted on IECUBE used
- When the [Use for trace data] property in the [Trace] category is set to other than
  [Coverage]
2.   [IECUBE2]
All of the properties in this category become invalid when the [Use for trace data] property in the [Trace] category is set to other than [Coverage].

Use coverage function [Simulator]	Specify whether to use the coverage function.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Use coverage functions
		No	Does not use coverage functions

Reuse coverage result	Specify whether to load/save the coverage measurement result when connecting to or disconnecting from the debug tool. **[IECUBE][IECUBE2]** This property appears only when the [Use for trace data] property is set to [Coverage]. **[Simulator]** This property appears only when the [Use coverage function] property is set to [Yes].		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Loads/saves the coverage measurement result.
		No	Does not load/save the coverage measurement result.
Coverage area of measurement(1MBytes) **[IECUBE]** **[Simulator]**	Specify the area that performs coverage measurement. Specify the start address of any 1 Mbyte space other than the internal ROM area. **[IECUBE]** This property appears only when the [Use for trace data] property is set to [Coverage]. **[Simulator]** This property appears only when the [Use coverage function] property is set to [Yes].		
	Default	*The start address of internal RAM area*	
	Modifying	Directly enter from the keyboard.	
	Available values	Address without the address range of the internal ROM area (symbols cannot be used).	

(9) [Mask for Input Signal] [IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]

The detailed information on the masking input signal is displayed and its configuration can be changed.

Mask NMI*n* signal **[IECUBE]** **[MINICUBE]** **[E1 (JTAG)]**	Specify whether to mask NMI*n* signal to prevent the signal input to emulators.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Masks NMI*n* signal
		No	Does not Mask NMI*n* signal
Mask HLDRQ signal **[IECUBE]** **[IECUBE2]** **[MINICUBE]** **[E1 (JTAG)]**	Specify whether to mask HLDRQ signal to prevent the signal input to emulators.		
	Default	No[Note]	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Masks HLDRQ signal
		No	Does not Mask HLDRQ signal
Mask RESET signal	Specify whether to mask RESET signal to prevent the signal input to emulators.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Masks RESET signal
		No	Does not Mask RESET signal

Mask STOP signal [IECUBE] [IECUBE2] [MINICUBE] [E1 (JTAG)]	Specify whether to mask STOP signal to prevent the signal input to emulators.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Masks STOP signal
		No	Does not Mask STOP signal
Mask WAIT signal [IECUBE] [IECUBE2] [MINICUBE] [E1 (JTAG)]	Specify whether to mask WAIT signal to prevent the signal input to emulators.		
	Default	No[Note]	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Masks WAIT signal.
		No	Does not mask WAIT signal.
Mask DBINT signal [MINICUBE[V850E1][V850ES]] [E1 (JTAG[V850E1][V850ES])]	Specify whether to mask DBINT signal to prevent the signal input to emulators. This property appears only when the selected microcontroller is other than the V850E2M core.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Masks DBINT signal
		No	Does not Mask DBINT signal

Note   [IECUBE][IECUBE2]
When the [Connecting with target board] property in the [Connection with Target Board] [IECUBE][IECUBE2][MINICUBE2][E1] category on the [Connect Settings] tab is set to [No], this property is fixed to [Yes] automatically after connecting to the debug tool (changes not allowed).

(10) [External Flash Memory Download] [IECUBE2][MINICUBE][E1 (JTAG)]
The detailed information on downloading to an external flash memory is displayed and its configuration can be changed.

External flash memory download	Specify whether to download load module files or hex files to an external flash memory.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Downloads to an external flash memory.
		No	Does not download to an external flash memory.
Information file	Specify the information file for the external flash memory. This property appears only when the [External flash memory download] property is set to [Yes].		
	Default	*Blank*	
	Modifying	Directly enter from the keyboard, or specify with the dialog box opened by clicking the [...] button.	
Start address	Specify the start address of the external flash memory. This property appears only when the [External flash memory download] property is set to [Yes].		
	Default	*Blank*	
	Modifying	Directly enter from the keyboard.	
	Available values	0x0 to 0xFFFFFFFF in hexadecimal number	

Data width	Specify the data width (access size) of the external flash memory. This property appears only when the [External flash memory download] property is set to [Yes].	
	Default	8
	Modifying	Select from the drop-down list.
	Available values	8, 16, 32
Number of parallel	Specify the number of external flash memories in parallel. This property appears only when the [External flash memory download] property is set to [Yes].	
	Default	1
	Modifying	Select from the drop-down list or directly enter from the keyboard.

	Available values	- Either one of the following from the drop-down list
		  1, 2, 4
		- Directly enter the numbers ranged below
		  Integer number between 1 and 99

Number of serial	Specify the number of external flash memories in serial. This property appears only when the [External flash memory download] property is set to [Yes].	
	Default	1
	Modifying	Select from the drop-down list or directly enter from the keyboard.

	Available values	- Either one of the following from the drop-down list
		  1, 2
		- Directly enter the numbers ranged below
		  Integer number between 1 and 99

(11) [Simulator GUI] [Simulator]

The detailed information on the Simulator GUI is displayed and its configuration can be changed.

Caution **After connecting to the debug tool, all the properties in this category will become invalid when a microcontroller whose Simulator does not support peripheral function simulations (instruction simulation version) is selected.**

Display Simulator GUI	Specify whether to display the Simulator GUI window.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Displays the Simulator GUI window.
		No	Does not display the Simulator GUI window.
Display Simulator GUI on top of other windows	Specify whether to display the Simulator GUI window in the forefront when program execution starts. This property appears only when the [Display Simulator GUI] property is set to [Yes].		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Displays it in the forefront.
		No	Does not display it in the forefront.

[Flash Self Emulation Settings] tab [IECUBE]

This tab is used to display the detailed information categorized by the following and the configuration can be changed.

(1)   [Flash Self Emulation]
(2)   [Macro Service Error]
(3)   [Security Flag Emulation]
(4)   [Reset Vector/Boot Swap]

Figure A-26.   Property Panel: [Flash Self Emulation Settings] Tab
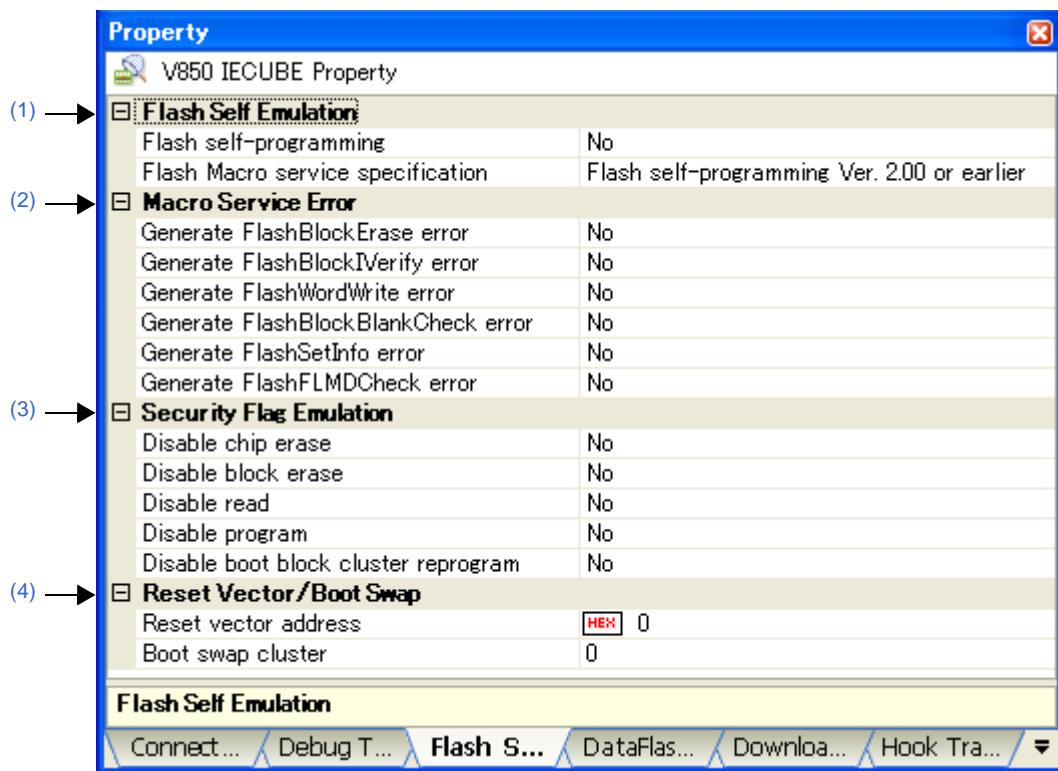
[Description of each category]

(1) [Flash Self Emulation]

The detailed information on flash self programming emulation functions is displayed and its configuration can be changed.

Flash self-programming	Specify whether to use the flash self programming emulation function.	
	Default	No
	Modifying	Select from the drop-down list.
	Available values	Yes	Uses the flash self programming emulation function.
		No	Does not use the flash self programming emulation function.

Flash Macro service specification	Select the specification type of the flash macro service. This property appears only when the flash memory process is Type1[Note].		
	Default	Flash self programming Ver.2.00 or earlier	
	Modifying	Select from the drop-down list.	
	Available values	Flash self programming Ver.2.00 or earlier	Specifies Ver.2.00 or earlier for the specification type.
		Flash self programming Ver.3.00 or later	Specifies Ver.3.00 or later for the specification type.

Note   The determination of the flash memory process is made when connected to the debug tool. Consequently, this will not be hidden when disconnected from the debug tool.

(2) [Macro Service Error]

The detailed information on flash macro service is displayed and its configuration can be changed.

Generate FlashBlockErase error	Specify whether to return error values in FlashBlockErase functions to emulate.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.
Generate FlashBlockIVerify error	Specify whether to return error values in FlashBlockIVerify functions to emulate.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.
Generate FlashWordWrite error	Specify whether to return error values in FlashWordWrite functions to emulate.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.
Generate FlashBlockBlankCheck error	Specify whether to return error values in FlashBlockBlankCheck functions to emulate.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.
Generate FlashSetInfo error	Specify whether to return error values in FlashSetInfo functions to emulate.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.

Generate FlashFLMDCheck error	Specify whether to return error values in FlashFLMDCheck functions to emulate.	
	Default	No
	Modifying	Select from the drop-down list.
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.

(3) [Security Flag Emulation]

The detailed information on the security flag emulation function is displayed and its configuration can be changed.

Disable chip erase	Specify whether to emulate to disable chip erase.	
	Default	No
	Modifying	Select from the drop-down list.
	Available values	Yes	Emulates to disable chip erase.
		No	Does not emulate to disable chip erase.
Disable block erase	Specify whether to emulate to disable block erase.	
	Default	No
	Modifying	Select from the drop-down list.
	Available values	Yes	Emulates to disable block erase.
		No	Does not emulate to disable block erase.
Disable read	Specify whether to emulate to disable reading. This property does not appear when the flash memory process is Type1[Note] and the specification type of the flash self programming is Ver.2.00 or earlier.	
	Default	No
	Modifying	Select from the drop-down list.
	Available values	Yes	Emulates to disable reading.
		No	Does not emulate to disable reading.
Disable program	Specify whether to emulate to disable writing.	
	Default	No
	Modifying	Select from the drop-down list.
	Available values	Yes	Emulates to disable writing.
		No	Does not emulate to disable writing.
Disable boot block cluster reprogram	Specify whether to emulate to disable rewrite boot area. This property appears only when the flash memory process is Type2 or Type3[Note].	
	Default	No
	Modifying	Select from the drop-down list.
	Available values	Yes	Emulates to disable rewrite boot area.
		No	Does not emulate to disable rewrite boot area.

Note The determination of the flash memory process is made when connected to the debug tool.

(4) [Reset Vector/Boot Swap]

The detailed information on reset vector and boot swap functions is displayed and its configuration can be changed.

Reset vector address	Specify the reset vector address.	
	This property appears only when the flash memory process is Type1 and the specification type of the flash self programming is Ver.3.00 or later[Note 1].	
	Default	0
	Modifying	Directly enter from the keyboard.
	Available values	0x0 to 0xFFFFFFFF in hexadecimal number
Boot swap cluster	Specify the target area of boot swap clusters.	
	The value set in this property is reflected to the target area in a boot block cluster[Note 2].	
	This property appears only when the flash memory process is Type4[Note 1].	
	Default	0
	Modifying	Directly enter from the keyboard.
	Available values	0 to 127 in decimal number

Notes 1. The determination of the flash memory process is made when connected to the debug tool. Consequently, this will not be hidden when disconnected from the debug tool.

2. Prohibition of boot area rewriting can be set to the target area of boot swap clusters by using the microcontroller's security function.

For the relationship between the value set in this property and the target are of boot block clusters, see "Table 2-2.　Relationship between Boot Swap Cluster and Boot Block Cluster".

[DataFlash Emulation Settings] tab [IECUBE]

This tab is used to display the detailed information categorized by the following and the configuration can be changed.

Note that this tab appears only when the selected microcontroller incorporates the data flash memory.

(1) [DataFlash Emulation]

(2) [EEPROM Library Error]

(3) [Writing Time / Erasing Time]

Figure A-27.   Property Panel: [DataFlash Emulation Settings] Tab
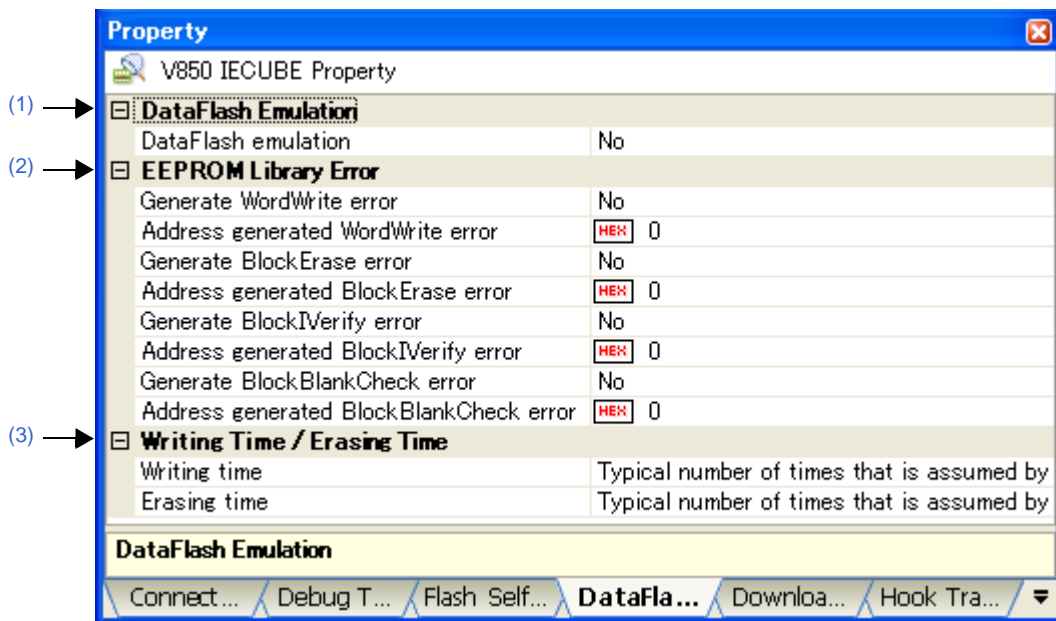
[Description of each category]

(1) [DataFlash Emulation]

The detailed information on the data flash emulation functions is displayed and its configuration can be changed.

DataFlash emulation	Specify whether to use the data flash emulation function.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Uses data flash emulation functions.
		No	Does not use data flash emulation functions.

(2) [EEPROM Library Error]

The detailed information on EEPROM library error is displayed and its configuration can be changed.

The error values that are not returned in normal emulation can forcibly be returned by setting the properties of this category.

Generate WordWrite error	Specify whether to return error values in WordWrite functions to emulate.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.
Address generated WordWrite error	Specify an address in the data flash memory, at which a WordWrite function error is to be generated. This property appears only when the [Generate WordWrite error] property is set to [Yes].		
	Default	0	
	Modifying	Directly enter from the keyboard.	
	Available values	0x0 to 0xFFFFFFFF in hexadecimal number	
Generate BlockErase error	Specify whether to return error values in BlockErase functions to emulate.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.
Address generated BlockErase error	Specify an address in the data flash memory, at which a BlockErase function error is to be generated. This property appears only when the [Generate BlockErase error] property is set to [Yes].		
	Default	0	
	Modifying	Directly enter from the keyboard.	
	Available values	0x0 to 0xFFFFFFFF in hexadecimal number	
Generate BlockIVerify error	Specify whether to return error values in BlockIVerify functions to emulate.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.
Address generated BlockIVerify error	Specify an address in the data flash memory, at which a BlockIVerify function error is to be generated. This property appears only when the [Generate BlockIVerify error] property is set to [Yes].		
	Default	0	
	Modifying	Directly enter from the keyboard.	
	Available values	0x0 to 0xFFFFFFFF in hexadecimal number	
Generate BlockBlankCheck error	Specify whether to return error values in BlockBlankCheck functions to emulate.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Forcibly returns error values.
		No	Does not return error values.

Address generated BlockBlankCheck error	Specify an address in the data flash memory, at which a BlockBlankCheck function error is to be generated. This property appears only when the [Generate BlockBlankCheck error] property is set to [Yes].	
	Default	0
	Modifying	Directly enter from the keyboard.
	Available values	0x0 to 0xFFFFFFFF in hexadecimal number

(3) [Writing Time / Erasing Time]

The detailed information on the time for writing to and erasing the data flash memory is displayed and its configuration can be changed.

Writing time	Select the delay time when writing to the data flash memory.		
	Default	Typical number of times that is assumed by flash macro specifications	
	Modifying	Select from the drop-down list.	
	Available values	No retry	Specifies "0" as the number of times of retry. The delay time is 0 (the writing speed is fastest).
		Typical number of times that is assumed by flash macro specifications	Specifies the typical number of times that is assumed by flash macro specifications.
		Maximum number of times that is assumed by flash macro specifications	Specifies the maximum number of times that is assumed by flash macro specifications.
		Retries for the maximum number of times specified	Specifies the maximum number of times of retry. The delay time is maximum (the writing speed is longest).
Erasing time	Select the delay time when erasing the data flash memory.		
	Default	Typical number of times that is assumed by flash macro specifications	
	Modifying	Select from the drop-down list.	
	Available values	No retry	Specifies "0" as the number of times of retry. The delay time is 0 (the erasing speed is fastest).
		Typical number of times that is assumed by flash macro specifications	Specifies the typical number of times that is assumed by flash macro specifications.
		Maximum number of times that is assumed by flash macro specifications	Specifies the maximum number of times that is assumed by flash macro specifications.
		Retries for the maximum number of times specified	Specifies the maximum number of times of retry. The delay time is maximum (the erasing speed is longest).

[Download File Settings] tab

This tab is used to display the detailed information categorized by the following and the configuration can be changed.
For details on the download function, see "2.5   Download/Upload Programs".

(1)   [Download]
(2)   [Debug Information]

Figure A-28.   Property Panel: [Download File Settings] Tab (Other than MINICUBE2/E1 (Serial))
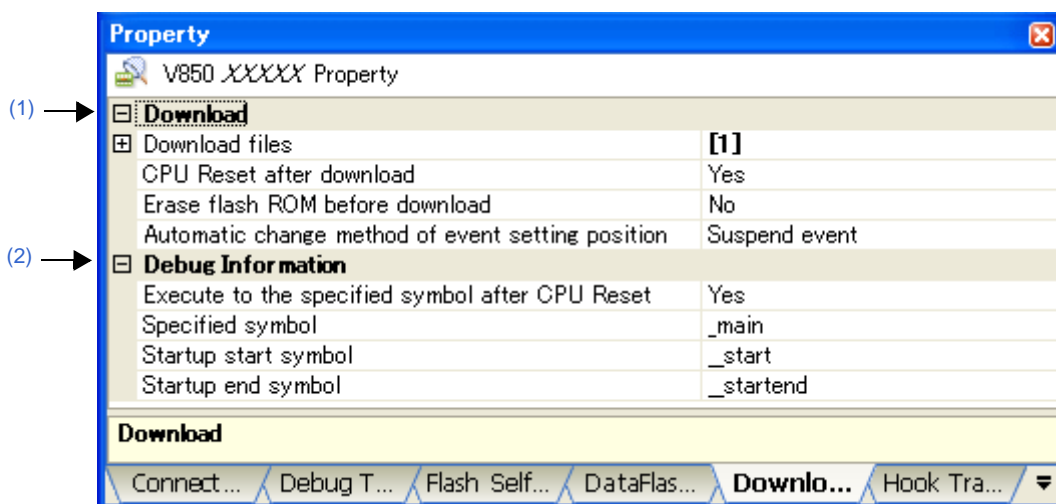
Figure A-29.   Property Panel: [Download File Settings] Tab [MINICUBE2]
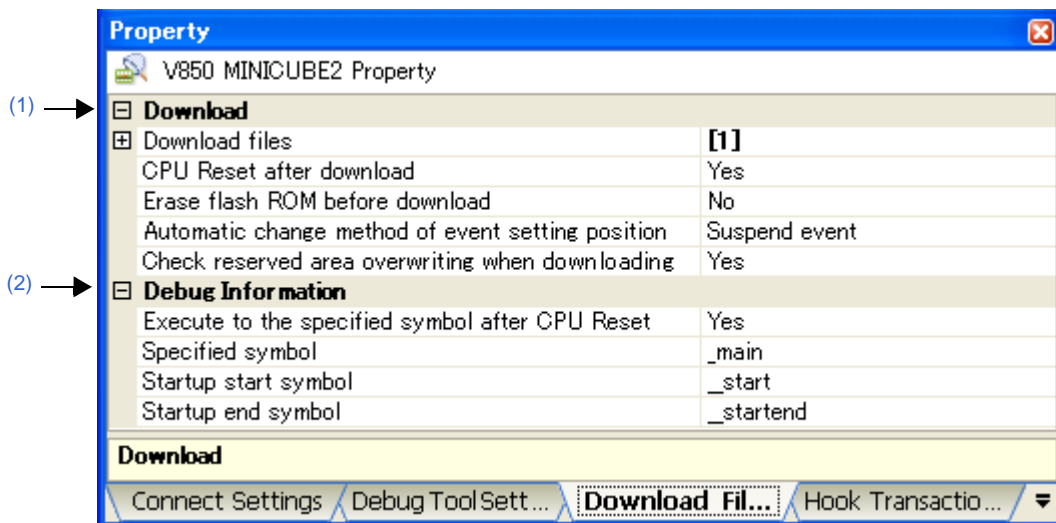
[Description of each category]

(1) [Download]

The detailed information on download is displayed and its configuration can be changed.

Download files	Specify the file to download[Note 1]. The names of files to be downloaded and the download conditions are listed in the lower area.		
	Default	[*Number of files to download*]	
	Modifying	Specify with the Download Files dialog box. The Download Files dialog box is opened when clicking the [...] button appears at right by selecting this property (you cannot specify the file to download on this panel).	
CPU Reset after download	Specify whether to reset the CPU after downloading.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Resets the CPU after downloading.
		No	Does not reset the CPU after downloading.
Erase flash ROM before download	Specify whether to erase the flash ROM before downloading.		
	Default	No	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Erases the flash ROM before downloading.
		No	Does not erase the flash ROM before downloading.
Automatic change method of event setting position	Specify how to perform the setting again if the file is downloaded again, and the location (address) set for the currently set event changes to midway in the instruction[Note 2].		
	Default	Suspend event	
	Modifying	Select from the drop-down list.	
	Available values	Move to the head of instruction	Sets the event to the top address of the instruction.
		Suspend event	Disables the event (suspended state).
Check reserved area overwriting when downloading **[MINICUBE2]** **[E1 (Serial)]**	Specify whether to output a message when overwriting to an area reserved for use by MINICUBE2/E1 (Serial) is attempted at the time of downloading.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Outputs a message when overwriting to an area reserved is attempted.
		No	Does not output a message when overwriting to an area reserved is attempted.

Notes 1.   Files specified as build targets in a main project or sub-project cannot be deleted from the target files to download (These files are automatically registered as download files by default).
See "Table 2-3.   Type of Files That Can be Downloaded" for downloadable file format.

2.   This property setting works only for the location setting of events without the debug information. The location setting of events with the debug information is always moved to the beginning of the source text line.

(2) **[Debug Information]**

The detailed information on debugging is displayed and its configuration can be changed.

Execute to the specified symbol after CPU Reset	Specify whether to execute the program to the specified symbol position after CPU reset.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Executes the program to the specified symbol position after CPU reset.
		No	Does not execute the program after CPU reset.
Specified symbol	Specify the position at which the program is stop after CPU reset. This property appears only when the [Execute to the specified symbol after CPU Reset] property is set to [Yes].		
	Default	_main	
	Modifying	Directly enter from the keyboard.	
	Available values	Address expression from 0 to the *"end address of the address space"*.	
Startup start symbol	Specify the start symbol of the text area of the startup routine.		
	Default	__start	
	Modifying	Directly enter from the keyboard.	
	Available values	Address expression from 0 to the *"end address of the address space"*.	
Startup end symbol	Specify the end symbol of the text area of the startup routine.		
	Default	__startend	
	Modifying	Directly enter from the keyboard.	
	Available values	Address expression from 0 to the *"end address of the address space"*.	

[Flash Options Settings] tab [V850E2M]

This tab is used to configure options for the flash memory incorporated in the V850E2M microcontroller.

To configure options, specify with the corresponding items on the Flash Options Setting dialog box [V850E2M] that is opened by clicking the [...] button appears at the right of the field by selecting the [Flash options] property in the [Flash Options] category on this tab (you cannot directly specify options on this tab).

Cautions 1.   You can configure options only while connecting to the debug tool.

2.   CPU reset may be generated automatically depending on the selected microcontroller when you change the configuration on this tab.

Figure A-30.   Property Panel: [Flash Options Settings] Tab
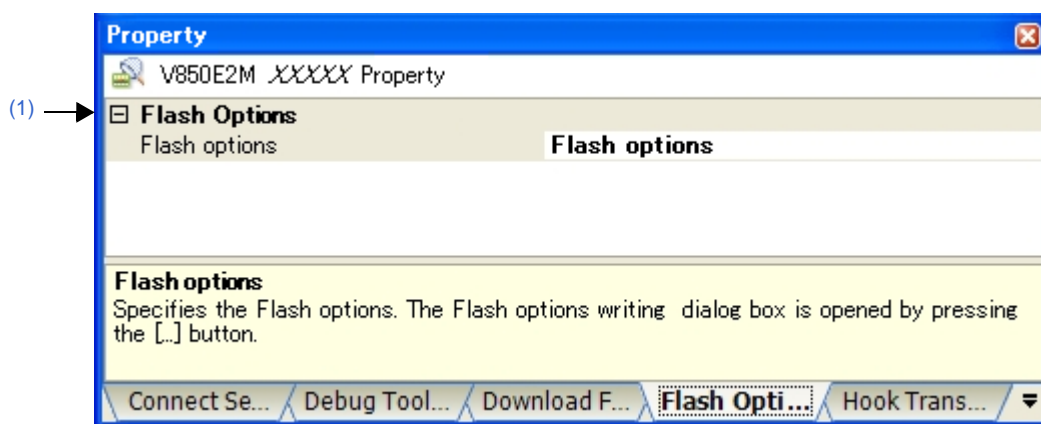
[Description of each category]

(1) [Flash Options]

The detailed information on the flash options is displayed and its configuration can be changed.

| Flash options	Click the [...] button appears at the right of the field by selecting this property, to open the Flash Options Setting dialog box [V850E2M].

[Hook Transaction Settings] tab

This tab is used to display the detailed information categorized by the following and the configuration can be changed. For details on the hook transaction, see "2.16   Use Hook Function".

(1)   [Hook Transaction Settings]

Figure A-31.   Property Panel: [Hook Transaction Settings] Tab
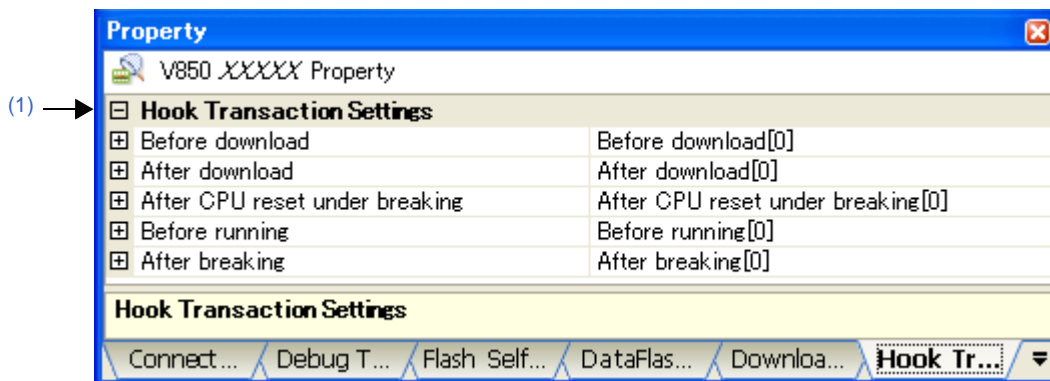
[Description of each category]

(1) [Hook Transaction Settings]

The detailed information on the hook transaction is displayed and its configuration can be changed.

Before download	Specify the process to proceed right before downloading. Automatically rewrite the I/O register/CPU register specified in this field right before downloading the load module file to the specified numbers.	
	Default	Before download[0] ("[]" is the current number of specified processes.)
	Modifying	Specify with the Text Edit dialog box. The Text Edit dialog box is opened when clicking the [...] button appears at right by selecting this property (you cannot specify the process on this panel).
	Available values	Format: - I/O register *name*/*CPU register name* + space + *specified value* Maximum characters:　　　　　64 (for one process) Maximum number of process: 128
After download	Specify the process to proceed right after downloading. Automatically rewrite the I/O register/CPU register specified in this field right after downloading the load module file to the specified numbers.	
	Default	After download[0] ("[]" is the current number of specified processes.)
	Modifying	Specify with the Text Edit dialog box. The Text Edit dialog box is opened when clicking the [...] button appears at right by selecting this property (you cannot specify the process on this panel).
	Available values	Format: - *I/O register name*/*CPU register name* + space + *specified value* Maximum characters:　　　　　64 (for one process) Maximum number of process: 128

After CPU reset under breaking	Specify the process to proceed right after CPU reset during break. Automatically rewrite the I/O register/CPU register specified in this field right after CPU reset to the specified numbers.	
	Default	After CPU reset under breaking[0] ("[]" is the current number of specified processes.)
	Modifying	Specify with the Text Edit dialog box. The Text Edit dialog box is opened when clicking the [...] button appears at right by selecting this property (you cannot specify the process on this panel).
	Available values	Format: - *I/O register name*/*CPU register name* + space + *specified value* Maximum characters: 64 (for one process) Maximum number of process: 128
Before running	Specify the process to proceed right before execution. Automatically rewrite the I/O register/CPU register specified in this field right before the execution of the program to the specified numbers.	
	Default	Before running[0] ("[]" is the current number of specified processes.)
	Modifying	Specify with the Text Edit dialog box. The Text Edit dialog box is opened when clicking the [...] button appears at right by selecting this property (you cannot specify the process on this panel).
	Available values	Format: - *I/O register name*/*CPU register name* + space + *specified value* Maximum characters: 64 (for one process) Maximum number of process: 128
After breaking	Specify the process to proceed right after the break. Automatically rewrite the I/O register/CPU register specified in this field right after the program break to the specified numbers.	
	Default	After breaking[0] ("[]" is the current number of specified processes.)
	Modifying	Specify with the Text Edit dialog box. The Text Edit dialog box is opened when clicking the [...] button appears at right by selecting this property (you cannot specify the process on this panel).
	Available values	Format: - *I/O register name*/*CPU register name* + space + *specified value* Maximum characters: 64 (for one process) Maximum number of process: 128

Editor panel

This panel is used to display and edit text files/source files.

Furthermore, the source level debugging (see "2.7.3 Execute programs in steps") and the code coverage measurement result display **[IECUBE][Simulator]** (see "2.13 Measure Coverage [IECUBE][IECUBE2][Simulator]") can be performed when connected to the debug tool and the downloaded source file is opened in this panel.

The file is opened by automatically distinguishing the encoding (Shift_JIS/EUC-JP/UTF-8) and the new line code (LF/CR) of the file, and when it is saved, the format are retained. The encoding and the new line code, however, can be changed via the Save Settings dialog box.

This panel can be multiply opened (up to 100 panels).

Remarks 1. When a project is closed, all of the Editor panels displaying a file being registered in the project are closed.

2. When a file is excluded from a project, the Editor panel displaying the file is closed.

3. A message is displayed when the downloaded lode module file is older than the source file to open.

Figure A-32. Editor Panel
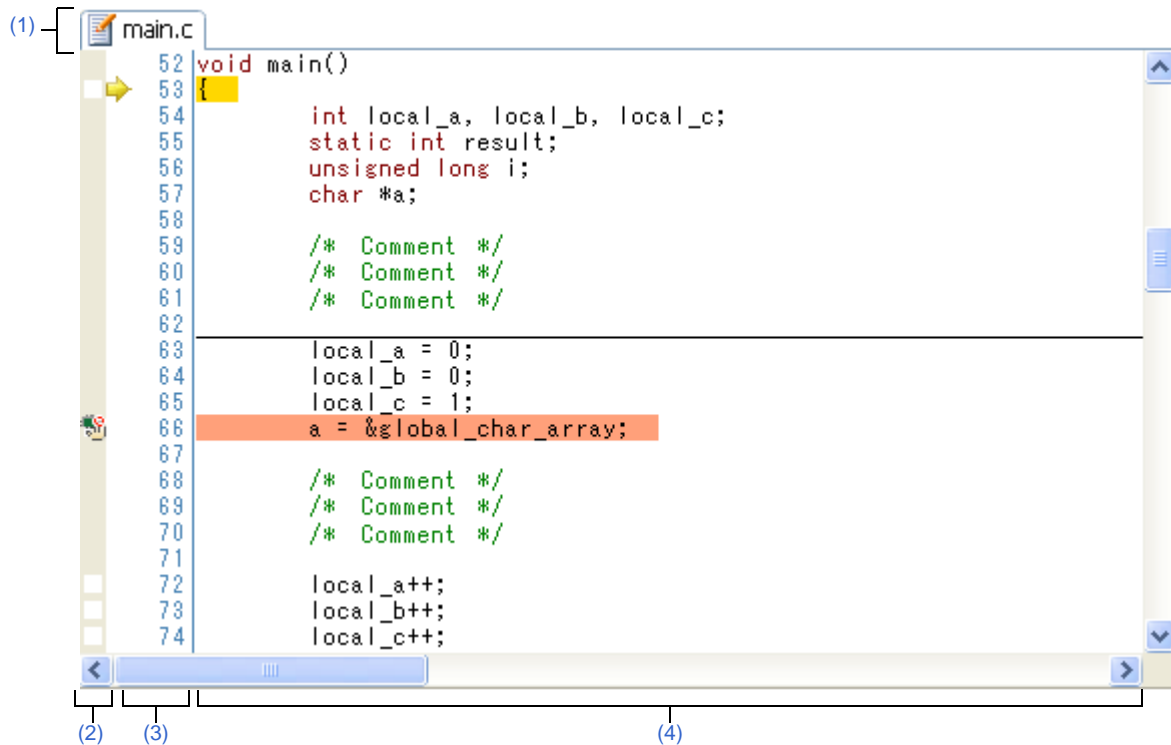
Figure A-33.   Editor Panel (When Code Coverage Measurement Result Is Displayed)
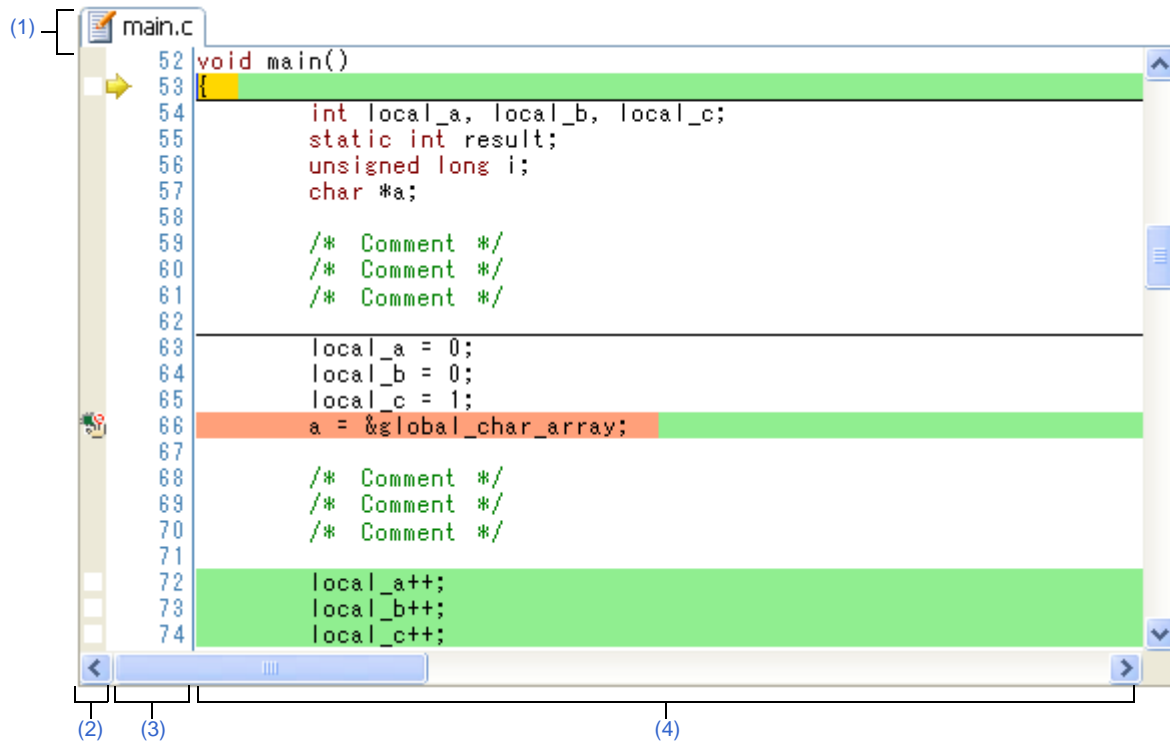
This section describes the following.

- [How to open]
- [Description of each area]
- [[File] menu (Editor panel-dedicated items)]
- [[Edit] menu (Editor panel-dedicated items)]
- [Context menu]

[How to open]

- On the Project Tree panel, double-click the file.
- Automatically opens after downloading the load module file with debug information.
- On the Project Tree panel, select the file and then select [Open] from the context menu.
- On the Project Tree panel, select [Add] >> [Add New File...] from the context menu and then create a text file/ source file.
- On the Disassemble panel/Call Stack panel/Trace panel [IECUBE][IECUBE2][Simulator]/Watch panel, select [Jump to Source] from the context menu.
- Automatically opens if there is a source text line correspond to the current PC value when the current PC value is forcibly changed or the program stops executing.

[Description of each area]

(1) Title bar

The name of the opened text file/source file is displayed.

Marks that are displayed at the end of the file name indicate as follows:

Mark	Description
*	The contents of the editing file is changed.
!	Update time and date of the source file opened are later than the one of the downloaded load module file in the case the text file, which is contained in the load module file downloaded in the connected debug tool as debug information, is opened.
(Uneditable)	The opened text file is write disabled.
ID number	The same text file is multiply opened.

(2) Event area

This area is valid only when connected to the debug tool and the downloaded source file is opened.

If the source text line can be converted to the corresponding address, events can be set to the line with background color in white (if the background color of the line is gray, events cannot be set to the line).

In addition, the Event mark corresponding to an event that has been currently set is displayed.

This area is provided with the following functions.

(a) Setting/deleting breakpoints

By clicking where you want to set a breakpoint with mouse, the breakpoints can be set easily.

The breakpoint is set to the instruction at the start address corresponding to the clicked line.

Once a breakpoint is set, an Event mark is displayed at the line that is set. In addition, the detailed information about the set breakpoint is reflected in the Events panel.

When this operation is performed at a place where any one of the event marks is already being displayed, that event is deleted and the setting of breakpoints cannot be done.

Note that the setting of events can be done only for those lines where the background color is shown in white.

See "2.8.2   Stop the program at the arbitrary position (breakpoint)" for details on how to set the breakpoint.

(b) Changes event status

Event status can be changed from the following menu displayed by right-clicking the event mark.

Enable Event	Changes the selected event state to a Valid state. Event occurs when the specified condition is met. When the event mark (🔧) which indicates that multiple events have been set is selected, all of the events that have been set are enabled.
Disable Event	Changes the selected event state to an Invalid state. Event does not occur when the specified condition is met. When the event mark (🔧) which indicates that multiple events have been set is selected, all of the events that have been set are disabled.
Delete Event	Deletes the selected event. When the event mark (🔧) which indicates that multiple events have been set is selected, all of the events that have been set are deleted.
View Event Detailed Setup	Opens the Events panel to display the detailed information of the selected event.

(c) Pop-up display

By superimposing the mouse cursor on the Event mark, the name of the event, the detailed information for the event and the comments added to the event are pop-up displayed.

When multiple events have been set in the applicable place, information for each event, up to a maximum of three events, is listed and displayed.

Remark   If the updated time and date of the downloaded load module file are older than the source file opened, this whole area is warned in blue (the color depends on the configuration in the [General - Font and Color] category of the Option dialog box).

Figure A-34.   Event Area Warned in Blue
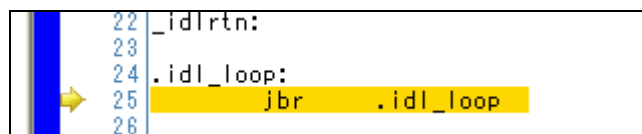
(3) Line number area

The line number of the opened text file/source file is displayed.

In addition, when connected to the debug tool and the downloaded source file is opened, the current PC mark ( ) that corresponds to the current PC position (PC register value) is displayed.

Note that the current PC mark is only displayed if the current PC value corresponds to the source text line, when the current PC position is modified or the state of the debug tool is changed from execution to stop.

(4) Characters area

The character strings of text files/source files are displayed and edited.

This area is provided with the following functions.

Remark   The following items can be customized by setting the Option dialog box.
- Display fonts
- Tab interval
- Display/hide/colors of control characters (control codes including a blank symbol)
- Colors of reserved words/comments

(a) Characters editing

Characters can be entered from the keyboard.

Various shortcut keys can be used to enhance the edit function.

(b) Tag jump

If the information of a file name, a line number and a column number exists in the line at the caret position, selecting [Tag Jump] from the context menu opens the file in the Editor panel and jumps to the corresponding line and the corresponding column (if the Editor panel is already opened, jump to the panel).

See "Table 2-6.   Operation of Tag Jump" for details on the operation of the tag jump.

(c) Current PC line display

When the current PC position (PC register value) corresponds to the source text lines, those lines are shown highlighted (the highlighting color depends on the configuration in the [General - Font and Color] category of the Option dialog box).

This function is only enabled when connected to the debug tool and the source file is opened.

(d) Lines with breakpoints display

Lines where the breakpoints are set are shown highlighted (the highlighting color depends on the configuration in the [General - Font and Color] category of the Option dialog box).

This function is only enabled when connected to the debug tool and the source file is opened.

(e) Code coverage measurement result display [IECUBE][Simulator]

When the coverage function is valid, lines corresponding to the specified coverage measurement area are shown highlighted based on the code coverage measurement result that is acquired by executing the program.

See "2.13   Measure Coverage [IECUBE][IECUBE2][Simulator]" for details on the coverage measurement.

This function is only enabled when connected to the debug tool and the source file is opened.

(f) Pop-up display of variables

By superimposing the mouse cursor on variables, variable values are pop-up displayed.

The display format of the variable value is same as "Table A-12.   Display Format of Watch-Expressions (Default)" depending on the type of the variable.

This function is only enabled when connected to the debug tool and the source file is opened.

(g) Setting of various events

Various events can be set to the addresses/lines where the caret currently exists by selecting [Bread Settings], [Trace Settings] or [Timer Settings] from the context menu.

The corresponding Event mark is displayed in the Event area by setting the event. In addition, the detailed information about the set event is reflected in the Events panel.

Note, however, that the setting of events can be done only for those lines where the background color is shown in white in the event area.

This function is only enabled when connected to the debug tool and the source file is opened.

See the following for details on how to set events.

- "2.8.3   Stop the program with the access to variables/I/O registers"
- "2.11.3   Collect execution history in the arbitrary section"
- "2.11.4   Collect execution history when the condition is met"
- "2.12.2   Measure execution time in the arbitrary section [IECUBE][IECUBE2][Simulator]"

Remark   A breakpoint can be set or deleted easily in the Event area as well (see "(a)   Setting/deleting breakpoints").

(h) Registering watch-expression

Variable names of C language, CPU registers, I/O registers, and assembler symbols can be registered in the Watch panel as watch-expressions.

See "(1)   Register a watch-expression" for details on how to operate it.

This function is only enabled when connected to the debug tool and the source file is opened.

(i) File monitor

The following function for monitoring is provided to manage source files.

- A message is displayed when the downloaded lode module file is older than the source file to open.
- If the contents of the currently displayed file are changed not with CubeSuite, a message is displayed to indicate whether to save the file. You can either select yes or no.

[[File] menu (Editor panel-dedicated items)]

The following items are exclusive for the [File] menu in the Editor panel (other items are common to all the panels).

Close *file name*	Closes the currently editing Editor panel. When the contents of the panel have not been saved, a confirmation message is displayed.
Save *file name*	Overwrites the contents of the currently editing Editor panel. Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save *file name* As...].
Save *file name* As...	Opens the Save As dialog box to newly save the contents of the currently editing Editor panel.
File name Save Option...	Opens the Save Settings dialog box to change the encoding and new line code of the file being opened in the currently editing Editor panel.
Page Setup...	Opens the Windows dialog box to set the page layout.
Print...	Opens the Windows dialog box for printing the contents of the currently editing Editor panel.

[[Edit] menu (Editor panel-dedicated items)]

The following items are exclusive for [Edit] menu in the Editor panel (other items are all invalid).

Undo	Cancels the previous operation and restores the characters and the caret position (up to 100 times).
Redo	Cancels the previous [Undo] operation and restores the characters and the caret position.
Cut	Cuts the selected character string and copies it to the clipboard.
Copy	Copies the contents of the selected range to the clipboard as character string(s).
Paste	Inserts (insert mode) or overwrites (overwrite mode) the characters that are copied on the clip board into the caret position.
Delete	Deletes one character at the caret position. When there is a selection area, all the characters in the area are deleted.
Select All	Selects all the characters from beginning to the end in the currently editing text file.
Find...	Opens the Search and Replace dialog box with selecting [Quick Search] tab. When there is a selection area, search is only taken place in the selection area.
Replace...	Opens the Search and Replace dialog box with selecting [Quick Replace] tab. When there is a selection area, replace is only taken place in the selection area.
Move...	Opens the Go to the Location dialog box to move the caret to the specified line.

[Context menu]

[Characters area/Line number area (while disconnecting from the debug tool)]

Jump to Function	Jumps to the function that is selected or at the caret position regarding the selected characters and the words at the caret position as functions (see "(4) Jump to functions").
Back To Last Cursor Position	Goes back to the position before operating [Jump to Function].
Forward To Next Cursor Position	Forwards to the position before operating [Back To Last Cursor Position].
Tag Jump	Jumps to the corresponding line and column in the corresponding file if the information of a file name, a line number and a column number exists in the line at the caret position (see "(b) Tag jump").
Cut	Deletes the selected character string(s) and copies them to the clipboard.
Copy	Copies the contents of the selected range to the clipboard as character string(s).
Paste	Inserts the contents of the clipboard into the caret position.
Open in New Panel	Opens a new Editor panel with the same contents as the current Editor panel (the title bar of the newly opened Editor panel displays the file name and ID number). The Editor panel can be opened up to 100 panels.

[Characters area/Line number area (while connecting to the debug tool)]

Register to Watch1	Registers a selected character string or a word at the caret position to the Watch panel (Watch1) as a watch-expression (the judgment of the word depends on current build tool). Note that this item becomes invalid when no corresponding address exists in the line at caret.
Register Action Event...	Opens the Action Events dialog box to set an action event to the corresponding address of the line at the caret position[Note]. Note that this item becomes invalid when no corresponding address exists in the line at caret.
Go to Here	Executes the program from the address indicated by the current PC value to the address corresponding to the line at the caret position[Note]. If the corresponding address of the line at the caret position does not exist, the program is executed to the corresponding address of the lower valid line. This item becomes invalid during execution of a program.
Set PC to Here	Sets the address of the line at the current caret position to the current PC value[Note]. Note that this item becomes invalid when no corresponding address exists in the line at caret, or during execution of a program.
Move…	Opens the Go to the Location dialog box to move the caret to the specified line.
Jump to Function	Jumps to the function that is selected or at the caret position regarding the selected characters and the words at the caret position as functions (see "(4) Jump to functions").
Back To Last Cursor Position	Goes back to the position before operating [Jump to Function].
Forward To Next Cursor Position	Forwards to the position before operating [Back To Last Cursor Position].
Tag Jump	Jumps to the corresponding line and column in the corresponding file if the information of a file name, a line number and a column number exists in the line at the caret position (see "(b) Tag jump").

Break Settings		The following cascade menus are displayed to set the break-related event.
		Note that events can be set only for lines for which events can be set (see "(2) Event area").
	Set Hardware Break	Sets a breakpoint (Hardware Break event) to the line at the caret position (see "2.8.2 Stop the program at the arbitrary position (breakpoint)")Note.
	Set Software Break **[IECUBE][IECUBE2 [MINICUBE][MINICUBE2] [E1]**	Sets a breakpoint (Software Break event) to the line at the caret position (see "2.8.2 Stop the program at the arbitrary position (breakpoint)")Note.
	Set Read Break to	Sets a break event with read access condition to the line at the caret or the selected variable (global variable/static variable inside functions/file-internal static variable)/I/O register (see "(1) Set a break event to a variable/I/O register").
	Set Write Break to	Sets a break event with write access condition to the line at the caret or the selected variable (global variable/static variable inside functions/file-internal static variable)/I/O register (see "(1) Set a break event to a variable/I/O register").
	Set R/W Break to	Sets a break event with read/write access condition to the line at the caret or the selected variable (global variable/static variable inside functions/file-internal static variable)/I/O register (see "(1) Set a break event to a variable/I/O register").
	Break Option	Opens the Property panel to set the break function.
Trace Settings **[IECUBE][IECUBE2] [Simulator]**		The following cascade menus are displayed to set the trace-related event.
		Note that events can be set only for lines for which events can be set (see "(2) Event area").
	Start Tracing	Sets a Start Tracing event to start collecting the trace data when the line at the caret is executed (see "2.11.3 Collect execution history in the arbitrary section")Note. **[Simulator]** In addition, the selecting of the [Use trace function] property in the [Trace] [IECUBE][IECUBE2][Simulator] category on the Property panel is automatically set to [Yes].
	Stop Tracing	Sets a Stop Tracing event to stop collecting the trace data when the line at the caret is executed (see "2.11.3 Collect execution history in the arbitrary section")Note. **[Simulator]** In addition, the selecting of the [Use trace function] property in the [Trace] [IECUBE][IECUBE2][Simulator] category on the Property panel is automatically set to [Yes].
	Record R/W Value	Sets a Point Trace event to record the access value as the trace data when the line at the caret or the selected variable (global variable/static variable inside functions/file-internal static variable)/I/O register is read/ write accessed (see "(1) When an access to variables/I/O registers occurs").
	Show Trace Result	Opens the Trace panel [IECUBE][IECUBE2][Simulator] and displays the acquired trace data.
	Trace Settings	Opens the Property panel to set the trace function.

Timer Settings **[IECUBE][IECUBE2]** **[Simulator]**		The following cascade menus are displayed to set the timer-related event (see "2.12.2 Measure execution time in the arbitrary section [IECUBE][IECUBE2][Simulator]"). Note that events can be set only for lines for which events can be set (see "(2) Event area").
	Start timer	Sets a Start Timer event to start measuring the execution time of the program when the line at caret is executed^{Note}. **[Simulator]** In addition, the selecting of the [Use timer function] property in the [Timer] [IECUBE][IECUBE2][Simulator] category on the Property panel is automatically set to [Yes].
	Stop timer	Sets a Stop Timer event to stop measuring the execution time of the program when the line at caret is executed^{Note}. **[Simulator]** In addition, the selecting of the [Use timer function] property in the [Timer] [IECUBE][IECUBE2][Simulator] category on the Property panel is automatically set to [Yes].
	View Result of Timer	Opens the Events panel and displays only timer-related events.
Clear Coverage Information		Clears all the coverage measurement results currently being stored in the debug tool. Note that this item appears only when the debug tool used supports the coverage function.
Jump to Disassemble		Opens the Disassemble panel and jumps to the address corresponding to the line at the caret^{Note}. Note that this item becomes invalid when no corresponding address exists in the line at caret.
Cut		Deletes the selected character string(s) and copies them to the clipboard.
Copy		Copies the contents of the selected range to the clipboard as character string(s).
Paste		Inserts the contents of the clipboard into the caret position.
Open in New Panel		Opens a new Editor panel with the same contents as the current Editor panel (the title bar of the newly opened Editor panel displays the file name and ID number).

Note A message is displayed if these items are selected when the downloaded lode module file is older than the opened source file.

[Event area] **[IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]**

Hardware Break First	The type of break that can be set by a one click operation of the mouse is set as a hardware breakpoint (this is reflected in the setting of the [First using type of breakpoint] property in the [Break] category on the Property panel).
Software Break First	The type of break that can be set by a one click operation of the mouse is set as a software breakpoint (this is reflected in the setting of the [First using type of breakpoint] property in the [Break] category on the Property panel).

Memory panel

This panel is used to display the contents of the memory and change the memory value (see "2.9.1   Display/change the memory").

Furthermore, the contents of data flash memory (including ID tag) can be displayed and changed when the selected microcontroller incorporates the data flash memory.

Up to a maximum of four of these panels can be opened. Each panel is identified by the names "Memory1", "Memory2", "Memory3", and "Memory4" on the title bar.

The display contents are automatically updated when the value of the memory changes after a program is executed (when the execution is done in steps, the display is updated after each step).

In addition, by enabling the Real-time display update function, it is also possible to update the display contents in real-time even while a program is being executed.

This panel appears only when connected to the debug tool.

Caution   **[V850E2M]**

                    CPU reset may be generated depending on the selected microcontroller if you change the memory value in the data flash area.

Figure A-35.   Memory Panel (When the Microcontroller without Data Flash Memory)
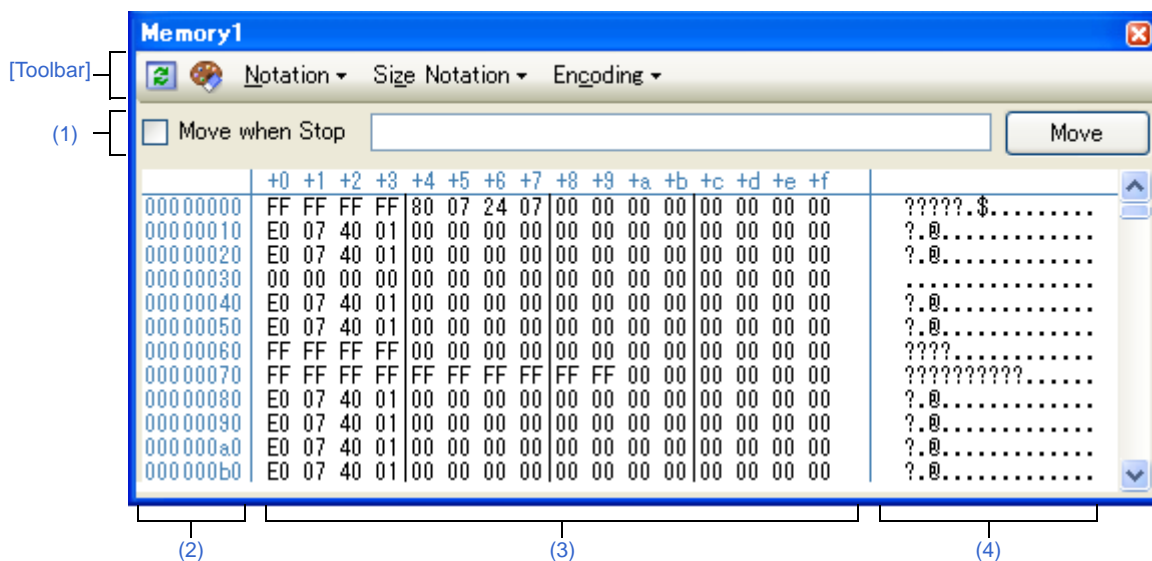
Figure A-36.   Memory Panel: [CPU Memory] Tab (When the Microcontroller with Data Flash Memory)

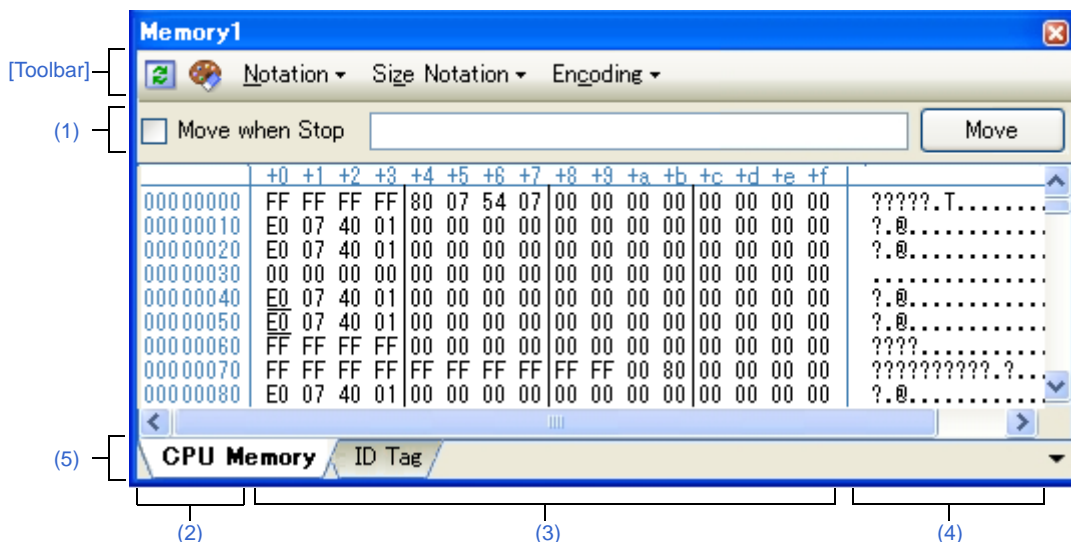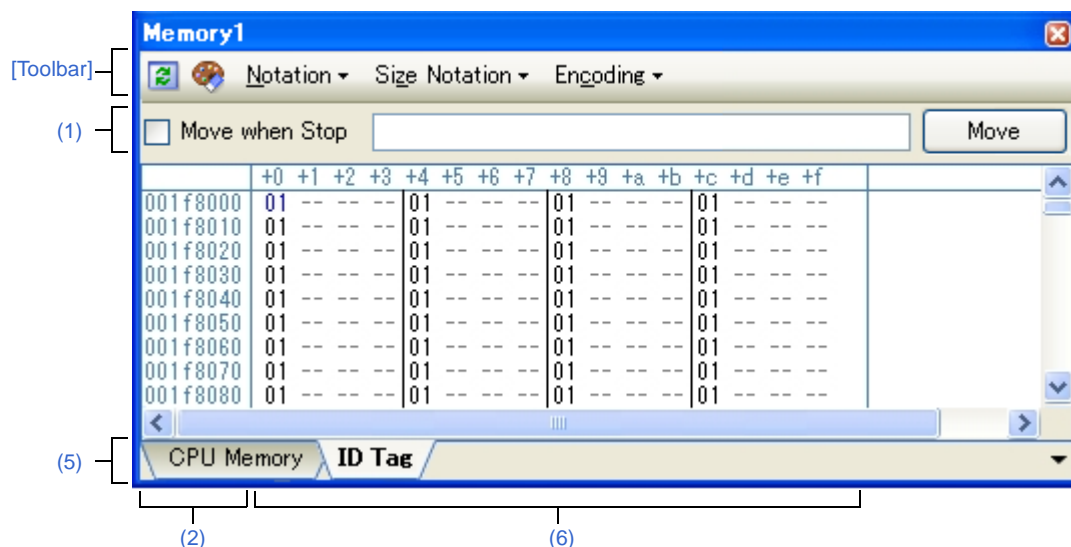**Figure A-37.   Memory Panel: [ID Tag] Tab (When the Microcontroller with Data Flash Memory)**

This section describes the following.

- [How to open]
- [Description of each area]
- [Toolbar]
- [[File] menu (Memory panel-dedicated items)]
- [[Edit] menu (Memory panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [Memory] >> [Memory *1-4*].

[Description of each area]

(1) Display position specification area

It is possible to specify the display start position of the memory contents by specifying an address expression.
The follow the steps below.

(a) Specify an address expression

Directly input the address expression of the memory value address to display in the text box. You can specify an input expression with up to 1024 characters. The result of the expression is treated as the display start position address.
Note that an address value greater than the microcontroller address space cannot be specified.
In addition, an address value greater than the value expressed within 32 bits cannot be specified.

(b) Specify automatic/manual evaluation of the address expression

The timing to change the display start position can be determined by specifying in the [Move when Stop] check box and the [Move] button.

[Move when Stop]	☑	The caret is moved to the address which is automatically calculated from the address expression after the program is stopped.
	☐	The address expression is not automatically evaluated after the program is stopped. Click the [Move] button to manually evaluate the address expression.
[Move] button		When the [Move when Stop] check box is not checked, click this button to evaluate the address expression and move the caret to the result address of the evaluation.

Remark　If the specified address expression is the symbol and its size can be recognized, everything from the start address to the end address of that symbol is displayed selected.

(2) Address area

The address of the memory is displayed (hexadecimal number notation fixing).
The address width corresponds to the one in memory space of the specified microcontroller in the project.
This area cannot be edited.

(3) Memory value area

The value of the memory mapped to the microcontroller is displayed and changed.
The display starts from address "0" by default.
Specification of the display notation (radix and size) for the memory value is performed by selecting the buttons on the toolbar or [Notation]/[Size Notation] from the context menu (it is hexadecimal number/ 8-bit width in the default condition).
The meanings of the marks and colors displayed as memory values are as follows (character colors and background colors depend on the configuration in the [General - Font and Color] category of the Option dialog box):

| Display Example (Default)			Description
00	Character color	Blue	Memory value that the user is changing Press the [Enter] key to write to the target memory.
	Background color	Standard color	
00 (Under line)	Character color	Standard color	Memory value of the address whose symbol has been defined (Registering watch-expression can be performed).
	Background color	Standard color	

Display Example (Default)			Description
00	Character color	Pink	Memory value for which the Real-time display update function is being operated
	Background color	Standard color	
00	Character color	Brown	Memory value that has been changed because of the execution of a program[Note]. To reset the highlighting, select the 🎨 button on the toolbar.
	Background color	Cream	
??	Character color	Gray	Area that are non-mapped, rewrite is restricted (such as I/O register area/I/O protection area), or memory which was not able to acquire a value.
	Background color	Standard color	
**	Character color	Standard color	A case where a area other than a real-time display update area is displayed and specified during execution of a program, or memory which was not able to acquire a value.
	Background color	Standard color	

Note   Just before execution of a program, only the memory value in the address range for which the Memory panel had been displayed becomes the target.
In addition, the value is not highlighted if it is same for before and after the execution of the program.

This area is provided with the following functions.

(a) Pop-up display

The following contents are pop-up displayed based on the nearest existing symbol forward from the address the mouse is designating when hovering the mouse cursor over the memory value.
Note that if there is no symbol information (the underlining is non-display), no pop-up display is done.

```
  variable      + 0x14
       |            |
  Symbol name   Offset value
```
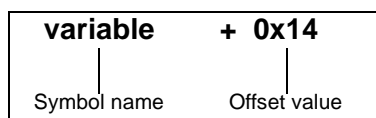
Symbol name	Indicates the name of the symbol.
Offset value	When a symbol has not been defined for the addresses, the offset value from the nearest symbol exists forward is displayed (hexadecimal number notation fixing).

(b) Real-time display update function

Using the real-time display update function enables to display/modify the value of the memory contents not only while the program is stopped, but also in execution.
See "(4)   Display/modify the memory contents during program execution" for details on the real-time display update function.

(c) Changing memory values

Directly edit from the keyboard after moving the caret to the memory value to be edited.
The color of the memory value changes when it is in editing. Press the [Enter] key to write the edited value to the target memory (if the [Esc] key is pressed before the [Enter] key is pressed, the editing is cancelled).
See "(3)   Modify the memory contents" for details on the method for changing the memory value.

(d) Searching/initializing memory value

The Memory Search dialog box is opened to search the memory contents in the specified address range by selecting [Find...] from the context menu.

In addition, the Memory Initialize dialog box is opened to change the memory contents collectively in the specified address range by selecting [Fill...] from the context menu.

(e) Copying and pasting

By selecting a range of memory values with the mouse, the contents of the range can be copied to the clipboard as a character string, and these contents can be pasted to the caret position.

These operations are performed by the selecting from the context menu or selecting from the [Edit] menu.

However, the paste operation is possible only when the character string to be pasted and the display notation (radix and size) of the area match.

If the display notation does not match, a message is displayed.

The character codes and character strings that can be handled by this area are as follows.

If character strings other than these are pasted, a message is displayed.

Character code	ASCII
Character string	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, A, B, C, D, E, F

(f) Registering watch-expression

A memory value with underline indicates that a symbol has been defined in the address, and its symbol can be registered as a watch-expression.

After selecting the memory value or placing the caret on the memory value, the symbol name of the address is registered in the Watch panel (Watch1) as a watch-expression by selecting [Register to Watch1] from the context menu.

Caution   A memory value without underline cannot be registered as a watch-expression.

(g) Saving the contents of memory values

The Data Save dialog box can be opened by selecting the [File] menu >> [Save Memory Data As...], and the contents of this panel can be saved in a text file (*.txt) or CSV file (*.csv).

See "(7) Save the memory contents" for details on the method for saving the contents of memory values.

(4) Character string area

Character codes that are converted from memory values are displayed.

Specification of the character code can be specified by clicking the button on the toolbar or selecting [Encoding] from the context menu item.

This area is provided with the following functions.

(a) Changing the character string

A character string can be changed only when [ASCII] has been selected as the character code.

Directly edit from the keyboard after moving the caret to the character string to be edited.

The color of the character string changes when it is in editing. Press the [Enter] key to write the edited value to the target memory (if the [Esc] key is pressed before the [Enter] key is pressed, the editing is cancelled).

(b) Copying and pasting

By selecting a range of character strings with the mouse, the contents of the range can be copied to the clipboard as a character string, and these contents can be pasted to the caret position.

These operations are performed by the selecting from the context menu or selecting from the [Edit] menu.

However, the paste operation is possible only when [ASCII] has been selected as the character code. If other than [ASCII] is selected, a message is displayed.

(5) Tab selection area [IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]

This area appears only when the selected microcontroller incorporates the data flash memory.

Select the tab that indicates the memory type.

The following tabs are displayed.

Tab Name	Description
CPU Memory	Displays the value of the memory mapped to the microcontroller.
ID Tag	Displays the value of the ID tag for the data flash memory.

(6) ID Tag area [IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]

This area appears only when the selected microcontroller incorporates the data flash memory.

The value of the ID tag for the data flash memory is displayed.

The ID tag is a bit assigned to the data flash memory and is used to detect power failure. When it is "0", it means that the data has been written normally.

The setting for this area can be changed only between "0" and "1".

[Toolbar]

The buttons below become invalid during execution of a program.

	Acquires the latest data from the debug tool, and updates the contents of this panel.
	Resets highlighting of values that have been changed by executing a program.
Notation	The following buttons to change the notation of memory values are displayed. The following cascade menus are displayed to specify the notation of memory values.
Hex	Displays memory values in hexadecimal number (default).
S Dec	Displays memory values in signed decimal number.
U Dec	Displays memory values in unsigned decimal number.
Oct	Displays memory values in octal number.
Bin	Displays memory values in binary number.
Size Notation	The following buttons to change the notation of sizes of memory values are displayed.
4	Displays memory values in 4-bit width.
8	Displays memory values in 8-bit width (default).
16	Displays memory values in 16-bit width.
32	Displays memory values in 32-bit width.
64	Displays memory values in 64-bit width.
Encoding	The following buttons to change the encoding of character strings are displayed.
SJIS	Displays character strings in ASCII code (default).
SJIS	Displays character strings in Shift_JIS code.
EUC	Displays character strings in EUC-JP code.
UTF	Displays character strings in UTF-8 code.

[[File] menu (Memory panel-dedicated items)]

The following items are exclusive for the [File] menu in the Memory panel (other items are common to all the panels).
Note that all these items are invalid during execution of a program.

Save Memory Data	Overwrites the contents of this panel to the previously saved text file (*.txt)/CSV file (*.csv) (see "(g) Saving the contents of memory values"). Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save Memory Data As...].
Save Memory Data As...	Opens the Data Save dialog box to newly save the contents of this panel to the specified text file (*.txt)/CSV file (*.csv) (see "(g) Saving the contents of memory values").

[[Edit] menu (Memory panel-dedicated items)]

The following items are exclusive for [Edit] menu in the Memory panel (other items are all invalid).
Note that all these items are invalid during execution of a program.

Copy	Copies the contents of the selected range to the clipboard as character string(s).
Paste	Pastes the character string(s) copied in the clipboard to the caret position. - To the memory value area: See "(e) Copying and pasting". - To the character string area: See "(b) Copying and pasting".
Find...	Opens the Memory Search dialog box. The search is operated either in the Memory value area or the Character string area, in which a caret is.

[Context menu]

Register to Watch1	Registers the symbol at the caret to the Watch panel (Watch1). At this time, since it is registered as a variable name, the symbol name that is displayed changes depending on the scope. Note that this item becomes invalid when no symbol has been defined in the address corresponding to the memory value at the caret position (see "(f) Registering watch-expression").
Find...	Opens the Memory Search dialog box. The search is operated either in the Memory value area or the Character string area, in which a caret is. This item becomes invalid during execution of a program.
Fill...	Opens the Memory Initialize dialog box.
Refresh	Acquires the latest data from the debug tool, and updates the contents of this panel. This item becomes invalid during execution of a program.
Copy	Copies the contents of the selected range to the clipboard as character string(s). This item becomes invalid during execution of a program.
Paste	Pasts the character string(s) copied in the clipboard to the caret position. This item becomes invalid during execution of a program. - To the memory value area: See "(e) Copying and pasting". - To the character string area: See "(b) Copying and pasting".

Notation		The following cascade menus are displayed to specify the notation of memory values. The items below become invalid during execution of a program.
	Hexadecimal	Displays memory values in hexadecimal number (default).
	Signed Decimal	Displays memory values in signed decimal number.
	Unsigned Decimal	Displays memory values in unsigned decimal number.
	Octal	Displays memory values in octal number.
	Binary	Displays memory values in binary number.
Size Notation		The following cascade menus are displayed to specify the notation of sizes of memory values. The items below become invalid during execution of a program.
	4 Bits	Displays memory values in 4-bit width.
	1 Byte	Displays memory values in 8-bit width (default).
	2 Bytes	Displays memory values in 16-bit width.
	4 Bytes	Displays memory values in 32-bit width.
	8 Bytes	Displays memory values in 64-bit width.
Encoding		The following cascade menus are displayed to specify the character code in the character string area. The items below become invalid during execution of a program.
	ASCII	Displays character strings in ASCII code (default).
	Shift_JIS	Displays character strings in Shift_JIS code.
	EUC-JP	Displays character strings in EUC-JP code.
	UTF-8	Displays character strings in UTF-8 code.
Highlight Accessed		Highlights memory values that have changed by execution of a program if this item is checked (default). This item becomes invalid during execution of a program.
Periodic Updating		The following cascade menus are displayed to set for the real-time display update function (see "(b) Real-time display update function"). This item becomes invalid during execution of a program.
	Periodic Updating Options	Opens the Property panel to set for the real-time display update function.

Disassemble panel

This panel is used to display the results of disassembling the contents of the memory (disassembled text), and execute line assembly (see "2.6.4　Perform line assembly").

Furthermore, the instruction level debugging (see "2.7.3　Execute programs in steps") and the code coverage measurement result display **[IECUBE][IECUBE2][Simulator]** (see "2.13　Measure Coverage [IECUBE2][IECUBE2][Simulator]") can be performed in this panel.

Up to a maximum of four of these panels can be opened. Each panel is identified by the names "Disassemble1", "Disassemble2", "Disassemble3" and "Disassemble4" on the title bar.

The source text in the source file corresponding to the code data can also be displayed by setting to the mixed display mode (default).

This panel appears only when connected to the debug tool.

Caution　**A step execution is performed in instruction level units when the focus is in this panel (see "2.7.3 Execute programs in steps").**

Figure A-38.　Disassemble Panel (When Mixed Display Mode Is Selected)
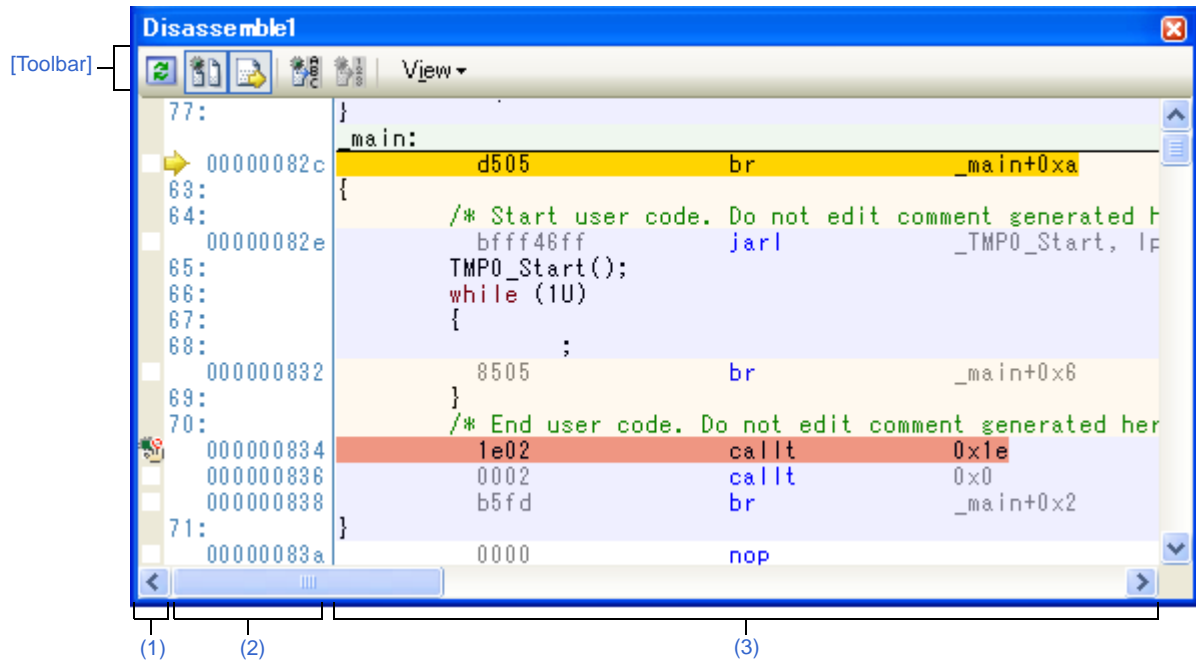

Figure A-39. Disassemble Panel (When Mixed Display Mode Is Not Selected)
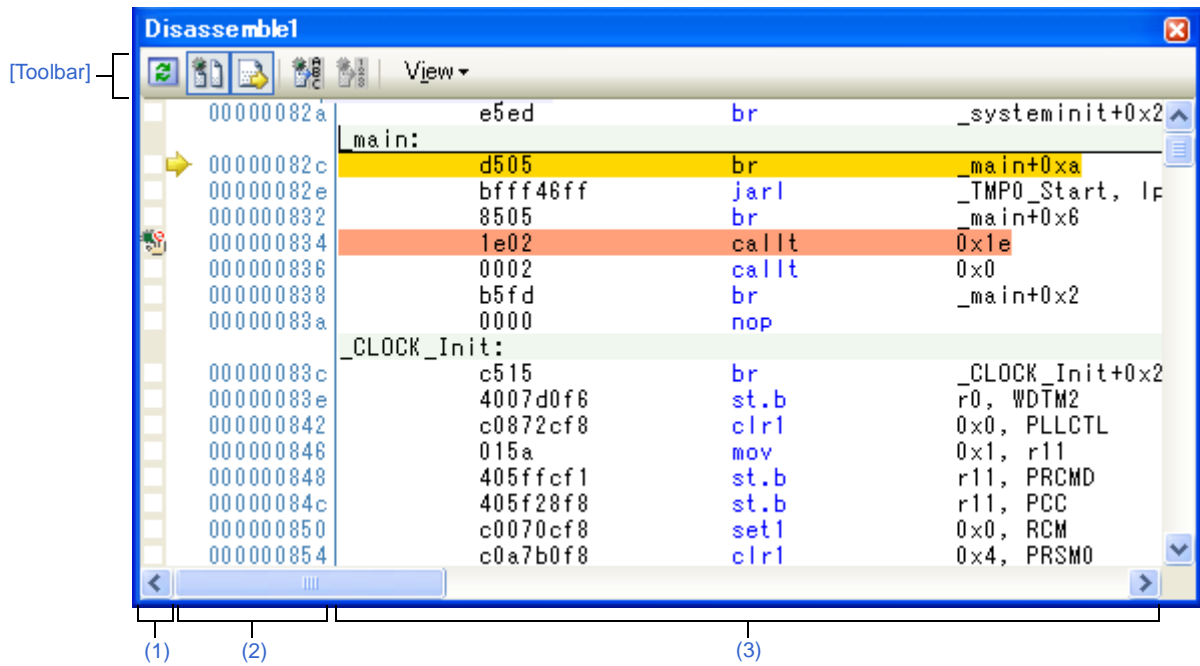
Figure A-40. Disassemble Panel (When Code Coverage Measurement Result Is Displayed)
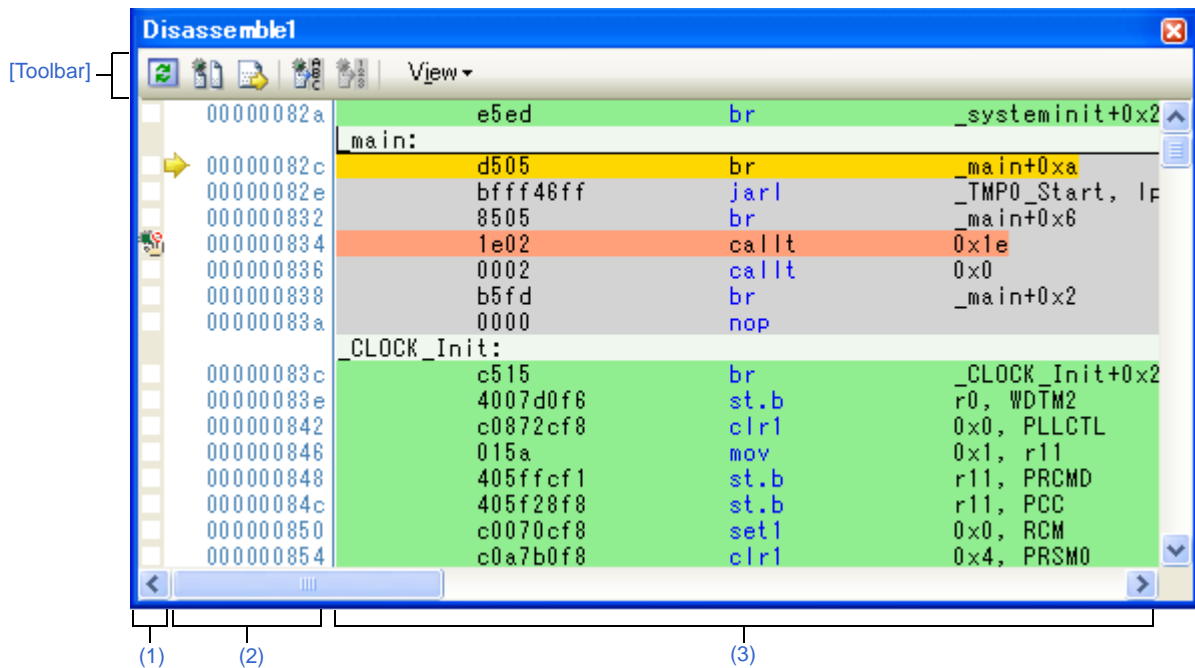
This section describes the following.

- [How to open]
- [Description of each area]
- [Toolbar]
- [[File] menu (Disassemble panel-dedicated items)]
- [[Edit] menu (Disassemble panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [Disassemble] >> [Disassemble *1 - 4*].

[Description of each area]

(1) Event area

The lines for which events can be set are shown with the background color in white (this mean that events cannot be set for those lines whose background color in gray).

In addition, the Event mark corresponding to an event that has been currently set is displayed.

This area is provided with the following functions.

(a) Setting/deleting breakpoints

By clicking where you want to set a breakpoint with the mouse, the breakpoint can be set easily.

The breakpoint is set to the instruction at the start address of the clicked line.

Once the breakpoint is set, the Event mark is displayed at the line that is set. In addition, the detailed information about the set breakpoint is reflected in the Events panel.

When this operation is performed at a place where any one of the event marks is already being displayed, that event is deleted and the setting of breakpoints cannot be done.

Note that the setting of events can be done only for those lines where the background color is shown in white.

See "2.8.2   Stop the program at the arbitrary position (breakpoint)" for details on how to set the breakpoint.

(b) Changes event status

Event status can be changed from the following menu displayed by right-clicking the event mark.

Enable Event	Changes the selected event state to a Valid state.
	Event occurs when the specified condition is met.
	When the event mark (🔳) which indicates that multiple events have been set is selected, all of the events that have been set are enabled.
Disable Event	Changes the selected event state to an Invalid state.
	Event does not occur when the specified condition is met.
	When the event mark (🔳) which indicates that multiple events have been set is selected, all of the events that have been set are disabled.
Delete Event	Deletes the selected event.
	When the event mark (🔳) which indicates that multiple events have been set is selected, all of the events that have been set are deleted.
View Event Detailed Setup	Opens the Events panel to display the detailed information of the selected event.

(c) Pop-up display

By superimposing the mouse cursor on the Event mark, the name of the event, the detailed information for the event and the comments added to the event are pop-up displayed.

When multiple events have been set in the applicable place, information for each event, up to a maximum of three events, is listed and displayed.

(2) Address area

The address per line to start disassembling is displayed (hexadecimal number notation fixing).

In addition, the current PC mark (🟨) that corresponds to the current PC position (PC register value) is displayed.

The address width corresponds to the one in memory space of the specified microcontroller in the project.

For the source text line in the mixed display mode, line numbers (*xxx:*) in the source file correspond to the start address are displayed.

This area is provided with the following functions.

(a) Pop-up display

By superimposing the mouse cursor on a address or line number, the following information is pop-up displayed.

Address	Format: *<Label name>* + *<Offset value>*
	Example1: main + 0x10
	Example2: sub function + 0x20
Source line number	Format: *<Load module name>*[Note]$*<File name>* # *<Line number>*
	Example1: test1.out$main.c#40
	Example2: main.c#100

Note *<Load module name>* is displayed only when multiple load modules have been downloaded to the debug tool.

(3) Disassemble area

The labels that have been defined at the start address, the code data, disassembled mnemonics and the corresponding source text, etc., are displayed as follows:

Figure A-41.   Display Contents of the Disassemble Area (In the Case of the Mixed Display Mode)
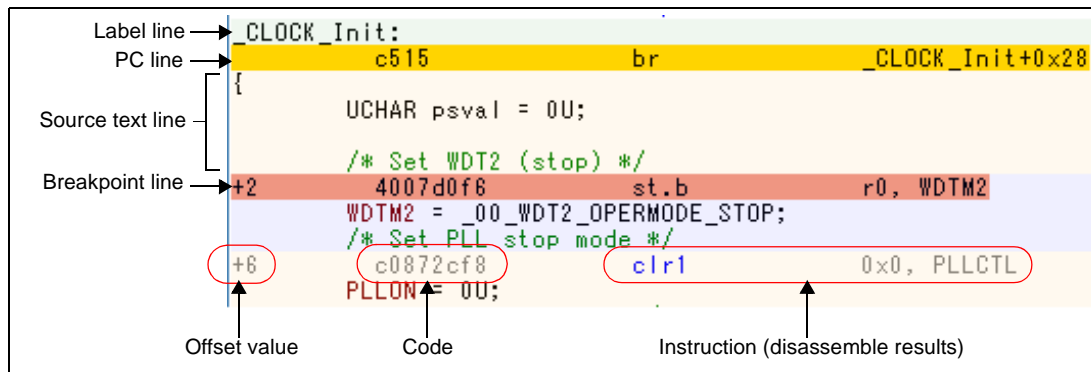
Label line	The label is displayed when a label is defined for the address, and its corresponding line is shown highlighted in lightgreen.
Offset value	The offset value from the nearest label is displayed when a label is defined for the address[Note 1].
Code	The code that is the target of disassembly is displayed in hexadecimal number.
Instruction	Instruction is displayed as the result of disassembling. The mnemonics are shown highlighted in blue.
Source text line	The source text corresponding to the code data is displayed[Note 2].
PC line	A line corresponding to an address of the current PC (PC register value) is shown highlighted[Note 3].
Breakpoint line	A line at which a breakpoint is set is shown highlighted[Note 3].

Notes 1.   Offset values are not displayed by default.
Offset values can be displayed by clicking the 🗔 button on the toolbar or selecting [Show Offset] from the context menu.

2.   The source text can be set to non-display by clicking the 🗔 button (toggle) on the toolbar or removing the check for [Mixed Display] from the context menu (this option is checked by default).

3.   The highlighting color depends on the configuration in the [General - Font and Color] category of the Option dialog box.

This area is provided with the following functions.

(a) Line assembly

Instructions and code displayed in this panel can be edited (line assembly).
See "2.6.4   Perform line assembly" for details on how to operate it.

(b) Program execution by instruction level

Execution can be controlled at the instruction level unit by step executing a program in a state where there is a focus on this panel.
See "2.7.3   Execute programs in steps" for details on how to operate it.

(c) Setting of various events

Various events can be set to the addresses/lines where the caret currently exists by selecting [Bread Settings], [Trace Settings] or [Timer Settings] from the context menu.
The corresponding Event mark is displayed in the Event area when an event is set. In addition, the detailed information about the set event is reflected in the Events panel.
+Note, however, that the setting of events can be done only for those lines where the background color is shown in white in the event area.
See the following for details on how to set events.
- "2.8.3   Stop the program with the access to variables/I/O registers"
- "2.11.3   Collect execution history in the arbitrary section"
- "2.11.4   Collect execution history when the condition is met"
- "2.12.2   Measure execution time in the arbitrary section [IECUBE][IECUBE2][Simulator]"

Remark   A breakpoint can be set or deleted easily in the Event area as well (see "(a)   Setting/deleting breakpoints").

(d) Registering watch-expression

Variable names of C language, CPU registers, I/O registers, and assembler symbols can be registered in the Watch panel as watch-expressions.
See "(1)   Register a watch-expression" for details on how to operate it.

(e) Moving to symbol definition place

By clicking the 🗔 button on the toolbar or selecting [Move to Symbol] from the context menu in a state where the caret has been moved to a instruction that has referenced a symbol, the caret position is moved to the address where the symbol at the caret position has been defined.
In addition, when following on this operation you click on the 🗔 button on the toolbar or select [Back to *Address*] from the context menu, the caret position is returned to the instruction that has referenced a symbol before the caret was moved (the address value of the instruction that has referenced a symbol is displayed in *Address*).

(f) Jump to source line and memory

By selecting [Jump to Source] from the context menu, the Editor panel is opened with moving the caret to the source line corresponding to the address at the current caret position (if the Editor panel is already opened, jump to the panel).

In addition, by similarly selecting [Jump to Memory], the Memory panel (Memory1) is opened with moving the caret to the memory value corresponding to the address at the current caret position (if the Memory panel (Memory1) is already opened, jump to the panel).

(g) Code coverage measurement result display [IECUBE][IECUBE2][Simulator]

When the coverage function is valid, lines corresponding to the specified coverage measurement area are shown highlighted based on the code coverage measurement result that is acquired by executing the program. See "2.13   Measure Coverage [IECUBE][IECUBE2][Simulator]" for details on the coverage measurement.

(h) Saving the contents of disassembled data

The Data Save dialog box can be opened by selecting the [File] menu >> [Save Disassemble Data As…], and the contents of this panel can be saved in a text file (*.txt) or CSV file (*.csv).

See "(5)   Save the disassembled text contents" for details on the method for saving the contents of disassembled data.

[Toolbar]

🔄		Acquires the latest data from the debug tool, and updates the contents of this panel. This item becomes invalid during execution of a program.
📄		Sets to the mixed display mode and displays the correspondence between the disassembled data and the source text (default).
➡️		Specifies the caret position so that it follows the current PC value.
🔤		Moves the caret to the define position of the selected symbol.
🔤		Moves the caret to the position (*address*) immediately before it is moved with the 🔤 button.
View		The following buttons to set the display contents in the disassemble area are displayed.
	🏷️	Displays the offset value of the label. The offset value from the nearest label is displayed when a label is defined for the address.
	🔤	Displays the address value in the format "symbol + offset value" (default). Note that when a symbol has been defined as the address value, only the symbol is displayed.
	🔤	Displays the name of the register by its function name (default).
	🔤	Displays the name of the register by its absolute name.

[[File] menu (Disassemble panel-dedicated items)]

The following items are exclusive for the [File] menu in the Disassemble panel (other items are common to all the panels).

Note that all these items are invalid during execution of a program.

Save Disassemble Data	Overwrites the contents of the disassembling to the previously saved text file (*.txt)/CSV file (*.csv) (see "(h) Saving the contents of disassembled data").
	Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save Disassemble Data As…].
Save Disassemble Data As…	Opens the Data Save dialog box to newly save the contents of the disassembling to the specified text file (*.txt)/CSV file (*.csv) (see "(h) Saving the contents of disassembled data").

[[Edit] menu (Disassemble panel-dedicated items)]

The following items are exclusive for the [Edit] menu in the Disassemble panel (other items are all invalid).

Copy	When a line is selected, copies the contents of the selected line to the clipboard as a character string.
	In the case of the edit mode, copies the selected character string to the clipboard.
Rename	Changes to the edit mode to edit the instruction/code at the caret position (see "2.6.4 Perform line assembly").
	This item becomes invalid during execution of a program.
Find…	Opens the Search and Replace dialog box with selecting the [Whole Search] tab.
Replace…	Opens the Search and Replace dialog box with selecting the [Whole Replace] tab.
Move…	Opens the Go to the Location dialog box to move the caret to the specified address.

[Context menu]

[Disassemble area and Address area]

Register to Watch1	Registers the selected character string or the word at the caret position to the Watch panel (Watch1) as a watch-expression (the judgment of the word depends on current build tool).
	At this time, since it is registered as a variable name, the symbol name that is displayed changes depending on the scope.
Register Action Event...	Opens the Action Events dialog box to set an action event to the address at the caret position.
Go to Here	Executes the program from the address indicated by the current PC value to the address corresponding to the line at the caret position.
	This item becomes invalid during execution of a program.
Set PC to Here	Sets the address of the line at the current caret position to the current PC value.
	This item becomes invalid during execution of a program.
Move…	Opens the Go to the Location dialog box to move the caret to the specified address.
Move to Symbol	Moves the caret to the define position of the selected symbol.
Back to *Address*	Moves the caret to the position (*address*) immediately before it is moved by [Move to Symbol].
	Note that this item becomes invalid when no symbol name is displayed in the address.

Break Settings		The following cascade menus are displayed to set the break-related event. Note that breakpoints can be set only for lines for which events can be set (see "(1) Event area").
	Set Hardware Break	Sets a breakpoint (Hardware Break event to the address at the caret position (see "2.8.2 Stop the program at the arbitrary position (breakpoint)").
	Set Software Break **[IECUBE][IECUBE2** **[MINICUBE][MINICUBE2]** **[E1]**	Sets a breakpoint (Software Break event) to the address at the caret position (see "2.8.2 Stop the program at the arbitrary position (breakpoint)").
	Set Read Break to	Sets a break event with read access condition to the address at the caret or the selected variable (global variable/static variable inside functions/file-internal static variable)/I/O register (see "(1) Set a break event to a variable/I/O register").
	Set Write Break to	Sets a break event with write access condition to the address at the caret or the selected variable (global variable/static variable inside functions/file-internal static variable)/I/O register (see "(1) Set a break event to a variable/I/O register").
	Set R/W Break to	Sets a break event with read/write access condition to the address at the caret or the selected variable (global variable/static variable inside functions/file-internal static variable)/I/O register (see "(1) Set a break event to a variable/I/O register").
	Break Option	Opens the Property panel to set the break function.
Trace Settings **[IECUBE][IECUBE2]** **[Simulator]**		The following cascade menus are displayed to set the trace-related event. Note that events can be set only for lines for which events can be set (see "(1) Event area").
	Start Tracing	Sets a Start Tracing event to start collecting the trace data when an instruction of an address at the caret position is executed (see "2.11.3 Collect execution history in the arbitrary section"). **[Simulator]** In addition, the selecting of the [Use trace function] property in the [Trace] [IECUBE][IECUBE2][Simulator] category on the Property panel is automatically set to [Yes].
	Stop Tracing	Sets a Stop Tracing event to stop collecting the trace data when an instruction of an address at the caret position is executed (see "2.11.3 Collect execution history in the arbitrary section"). **[Simulator]** In addition, the selecting of the [Use trace function] property in the [Trace] [IECUBE][IECUBE2][Simulator] category on the Property panel is automatically set to [Yes].
	Record R/W Value	Sets a Point Trace event to record the access value as the trace data when the address at the caret or the selected variable (global variable/static variable inside functions/file-internal static variable)/I/O register is read/ write accessed (see "(1) When an access to variables/I/O registers occurs").
	Show Trace Result	Opens the Trace panel [IECUBE][IECUBE2][Simulator] and displays the acquired trace data.
	Trace Settings	Opens the Property panel to set the trace function.

Timer Settings **[IECUBE][IECUBE2] [Simulator]**		The following cascade menus are displayed to set the timer-related event (see "2.12.2 Measure execution time in the arbitrary section [IECUBE][IECUBE2][Simulator]"). Note that events can be set only for lines for which events can be set (see "(1) Event area").
	Start timer	Sets a Start Timer event to start measuring the execution time of the program when an instruction of an address at the caret position is executed. **[Simulator]** In addition, the selecting of the [Use timer function] property in the [Timer] [IECUBE][IECUBE2][Simulator] category on the Property panel is automatically set to [Yes].
	Stop timer	Sets a Stop Timer event to stop measuring the execution time of the program when an instruction of an address at the caret position is executed. **[Simulator]** In addition, the selecting of the [Use timer function] property in the [Timer] [IECUBE][IECUBE2][Simulator] category on the Property panel is automatically set to [Yes].
	View Result of Timer	Opens the Events panel and displays only timer-related events.
Clear Coverage Information		Clears all the coverage measurement results currently being stored in the debug tool. Note that this item appears only when the debug tool used supports the coverage function.
Edit Disassemble		Changes to the edit mode to edit the instruction of the line at the caret position (see "2.6.4 Perform line assembly"). This item becomes invalid during execution of a program.
Edit Code		Changes to the edit mode to edit the code of the line at the caret position (see "2.6.4 Perform line assembly"). This item becomes invalid during execution of a program.
View		The following cascade menus to set the display contents in the disassemble area are displayed.
	Show Offset	Displays the offset value of the label. The offset value from the nearest label is displayed when a label is defined for the address.
	Show Symbol	Displays the address value in the format "symbol + offset value" (default). Note that when a symbol has been defined as the address value, only the symbol is displayed.
	Show Function Name	Displays the name of the register by its function name (default).
	Show Absolute Name	Displays the name of the register by its absolute name.
Mixed Display		Sets to the mixed display mode and displays the correspondence between the disassembled data and the source text (default).
Jump to Source		Opens the Editor panel and jumps to the source line corresponding to the address at the caret position in this panel.
Jump to Memory		Opens the Memory panel (Memory1) and jumps to the memory value corresponding to the address at the caret position in this panel.

[Event area] **[IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]**

Hardware Break First	The type of break that can be set by a one click operation of the mouse is set as a hardware breakpoint (this is reflected in the setting of the [First using type of breakpoint] property in the [Break] category on the Property panel).
Software Break First	The type of break that can be set by a one click operation of the mouse is set as a software breakpoint (this is reflected in the setting of the [First using type of breakpoint] property in the [Break] category on the Property panel).

CPU Register panel

This panel is used to display the contents of the CPU register (general-purpose registers and system registers) and change the CPU register values (see "2.9.2   Display/change the CPU register").
This panel appears only when connected to the debug tool.

Remark   When the separator line of each area in this panel is double-clicked, the width of the area changes to the shortest possible size that can display the contents of the area.

Figure A-42.   CPU Register Panel
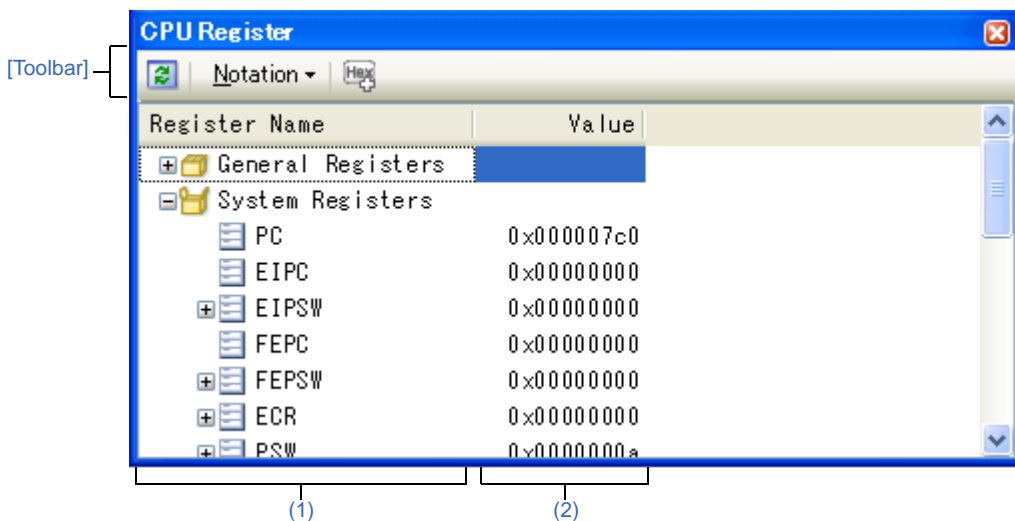
This section describes the following.
- [How to open]
- [Description of each area]
- [Toolbar]
- [[File] menu (CPU Register panel-dedicated items)]
- [[Edit] menu (CPU Register panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [CPU Register].

[Description of each area]

(1) [Register Name] area
The types of register are classified as categories (folders), and a list of the respective register names is displayed.
Note that neither category names nor register names can be edited and deleted.
The meanings of the icons are as follows:

	Indicates that the register name belonging to this category is displayed. When you double-click on the icon, or click on the "-" mark, the category is closed and the register name is hidden.
	Indicates that the register name belonging to this category is hidden. When you double-click on the icon, or click on the "+" mark, the category is opened and the register name is displayed.

▤	Indicates the name of the register. When you double-click on the icon, or click on the "+" or "-" marks, the name of the register part is displayed or hidden.
▤	Indicates the name of the register part.

Category names and register names displayed are as follows (number of "+" marks before register names indicates the depth of the display level):

Table A-3. Register Names in [General Registers] Category [V850E1][V850ES]

Register Name (Alias)	Bit Width	Register Name (Alias)	Bit Width
+ r0	32	+ r5(TP)	32
+ r1	32	+ r6 to + r29	32
+ r2	32	+ r30(EP)	32
+ r3(SP)	32	+ r31(LP)	32
+ r4(GP)	32		

Table A-4. Register Names in [General Registers] Category [V850E2M]

Register Name (Alias)	Bit Width	Register Name (Alias)	Bit Width
+ Integer	-	+ Single Precision Floating Point	-
++ r0	32	++ fp1 to + fp30	32
++ r1	32	++ fp31	32
++ r2	32	+ Double Precision Floating Point	-
++ r3(SP)	32	++ dp2	64
++ r4(GP)	32	++ dp2 to + dp28	64
++ r5(TP)	32	++ dp30	64
+ r6 to + r29	32		
++ r30(EP)	32		
++ r31(LP)	32		

Table A-5. Register Names in [System Registers] Category [V850E1][V850ES]

Register Name	Bit Width	Register Name	Bit Width
+ PC	32	+ CTPC	32
+ EIPC	32	+ CTPSW	32
+ EIPSW	32	++ NP	1
++ NP	1	++ EP	1
++ EP	1	++ ID	1
++ ID	1	++ SAT	1
++ SAT	1	++ CY	1
++ CY	1	++ OV	1
++ OV	1	++ S	1

Register Name	Bit Width	Register Name	Bit Width
++ S	1	++ Z	1
++ Z	1	+ DBPC	32
+ FEPC	32	+ DBPSW	32
+ FEPSW	32	++ NP	1
++ NP	1	++ EP	1
++ EP	1	++ ID	1
++ ID	1	++ SAT	1
++ SAT	1	++ CY	1
++ CY	1	++ OV	1
++ OV	1	++ S	1
++ S	1	++ Z	1
++ Z	1	+ CTBP	32
+ ECR	32		
+ PSW	32		
++ NP	1		
++ EP	1		
++ ID	1		
++ SAT	1		
++ CY	1		
++ OV	1		
++ S	1		
++ Z	1		

Table A-6.   Register Names in [System Registers] Category [V850E2M]

Register Name	Bit Width	Register Name	Bit Width
+ PC	32	++ Software Paging Bank	-
+ CPU Function Group	-	+++ MPM	32
++ Main Banks	-	+++ MPC	32
+++ EIPC	32	+++ TID	32
+++ EIPSW	32	+++ VMECR	32
+++ FEPC	32	+++ VMTID	32
+++ FEPSW	32	+++ VMADR	32
+++ ECR	32	+++ IPA0L	32
+++ PSW	32	+++ IPA0U	32
++++ PP	1	+++ IPA1L	32
++++ NPV	1	+++ IPA1U	32
++++ DMP	1	+++ IPA2L	32
++++ IMP	1	+++ IPA2U	32

Register Name	Bit Width	Register Name	Bit Width
++++ NP	1	+++ IPA3L	32
++++ EP	1	+++ IPA3U	32
++++ ID	1	+++ IPA4L	32
++++ SAT	1	+++ IPA4U	32
++++ CY	1	+++ DPA0L	32
++++ OV	1	+++ DPA0U	32
++++ S	1	+++ DPA1L	32
++++ Z	1	+++ DPA1U	32
+++ SCCFG	32	+++ DPA2L	32
+++ SCBP	32	+++ DPA2U	32
+++ EIIC	32	+++ DPA3L	32
+++ FEIC	32	+++ DPA3U	32
+++ CTPC	32	+++ DPA4L	32
+++ CTPSW	32	+++ DPA4U	32
+++ CTBP	32	+++ DPA5L	32
++ Exception Handler Switch Function Bank 0	-	+++ DPA5U	32
+++ SW_CTL	32	+ FPU Function Group	-
+++ SW_CFG	32	++ FPU Status Bank	-
+++ SW_BASE	32	+++ FPSR	32
++ Exception Handler Switch Function Bank 1	-	+++ FPEPC	32
+++ EH_CFG	32	+++ FPST	32
+++ EH_RESET	32	+++ FPCC	32
+++ EH_BASE	32	+++ FPCFG	32
+ Processor Protection Function Group	-	+++ FPEC	32
++ Processor Protection Violation Bank	-	+ User Group	-
+++ VSECR	32	++ User 0 Banks	-
+++ VSTID	32	+++ PSW	32
+++ VSADR	32	++++ PP	1
+++ VMECR	32	++++ NPV	1
+++ VMTID	32	++++ DMP	1
+++ VMADR	32	++++ IMP	1
+++ MCA	32	++++ NP	1
+++ MCS	32	++++ EP	1
+++ MCC	32	++++ ID	1
+++ MCR	32	++++ SAT	1
++ Processor Protection Setting Bank	-	++++ CY	1

Register Name	Bit Width	Register Name	Bit Width
+++ MPM	32	++++ OV	1
+++ MPC	32	++++ S	1
+++ TID	32	++++ Z	1
+++ IPA0L	32	+++ FPST	32
+++ IPA0U	32	+++ FPCC	32
+++ IPA1L	32	+++ FPCFG	32
+++ IPA1U	32	+++ CTPC	32
+++ IPA2L	32	+++ CTPSW	32
+++ IPA2U	32	++++ PP	1
+++ IPA3L	32	++++ NPV	1
+++ IPA3U	32	++++ DMP	1
+++ IPA4L	32	++++ IMP	1
+++ IPA4U	32	++++ NP	1
+++ DPA0L	32	++++ EP	1
+++ DPA0U	32	++++ ID	1
+++ DPA1L	32	++++ SAT	1
+++ DPA1U	32	++++ CY	1
+++ DPA2L	32	++++ OV	1
+++ DPA2U	32	++++ S	1
+++ DPA3L	32	++++ Z	1
+++ DPA3U	32	+++ CTBP	32
+++ DPA4L	32	+ Common Group	-
+++ DPA4U	32	++ EIWR	32
+++ DPA5L	32	++ FEWR	32
+++ DPA5U	32	++ BSEL	32
		+++ BNK	8
		+++ GRP	8

This area is provided with the following functions.

(a) Registering watch-expression

CPU registers/categories can be registered in the Watch panel as watch-expressions.

See "(1) Register a watch-expression" for details on how to operate it.

Remarks 1.   When you have registered a watch-expression with a category as the object, all of the CPU
registers belonging to that category are registered as watch-expressions.

2.   A scope specification is automatically added to a registered watch-expression.

(2) [Value] area

The values of each CPU register are displayed and changed.

The radix of a data value can be selected by the button on the toolbar or the context menu item. In addition, a display format adding the value in hexadecimal number constantly can also be selected as well.

The meanings of the colors of the CPU register values are as follows (character colors and background colors depend on the configuration in the [General - Font and Color] category of the Option dialog box):

Display Example (Default)			Description
0x0	Character color	Blue	The value of the CPU register that the user is changing
	Background color	Standard color	Press the [Enter] key to write to the target memory.
0x0	Character color	Brown	The value of the CPU register that has been changed because of the execution of a program
	Background color	Cream	The highlighting is rest by executing again the program.

This area is provided with the following functions.

(a) Changing the CPU register value

To edit the CPU register value, change the value directly from the keyboard after double-clicking on the value to be edited (press the [Enter] key to cancel the edit mode).

After you edit the value of the CPU register, it is written to the target memory of the debug tool by pressing the [Enter] key or moving the focus to outside the edit region.

(b) Saving the contents of the CPU register

The Save As dialog box can be opened by selecting the [File] menu >> [Save CPU Register Data As…], and all the contents of this panel can be saved in a text file (*.txt) or CSV file (*.csv).

See "(4)   Save the CPU register contents" for details on the method for saving the contents of the CPU register.

[Toolbar]

(refresh icon)	Acquires the latest data from the debug tool, and updates the contents of this panel. This item becomes invalid during execution of a program.
Notation	The following buttons to change the notation of a data value are displayed.
auto	Displays the value of the selected item (including sub-items) in the default notation (default).
Hex	Displays the value of the selected item (including sub-items) in hexadecimal number.
S Dec	Displays the value of the selected item (including sub-items) in signed decimal number.
U Dec	Displays the value of the selected item (including sub-items) in unsigned decimal number.
Oct	Displays the value of the selected item (including sub-items) in octal number.
Bin	Displays the value of the selected item (including sub-items) in binary number.
Asc	Displays the character string of the selected item (including sub-items) in ASCII code. If the character size is 2 bytes and above, it is displayed with the characters for each 1 byte arranged side-by-side.
Flo	Displays the value of the selected item in Float. Note that when the value is not 4-byte data, displays it in the default notation.
Dbl	Displays the value of the selected item in Double. Note that when the value is not 8-byte data, displays it in the default notation.
Hex	Adds the value in hexadecimal number enclosing with "()" at the end of the value.

[[File] menu (CPU Register panel-dedicated items)]

The following items are exclusive for the [File] menu in the CPU Register panel (other items are common to all the panels).

Note that all these items are invalid during execution of a program.

Save CPU Register Data	Overwrites the contents of this panel to the previously saved text file (*.txt)/CSV file (*.csv) (see "(b) Saving the contents of the CPU register"). Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save CPU Register Data As…].
Save CPU Register Data As…	Opens the Save As dialog box to newly save the contents of this panel to the specified text file (*.txt)/ CSV file (*.csv) (see "(b) Saving the contents of the CPU register").

[[Edit] menu (CPU Register panel-dedicated items)]

The following items are exclusive for [Edit] menu in the CPU Register panel (other items are all invalid).

Cut	Deletes the selected character string and copies it to the clipboard. This item becomes valid only when the character string is being edited.
Copy	Copies the selected character string to the clipboard during editing. If a line is selected, copies the register or the category to the clipboard. The copied item can be pasted to the Watch panel.
Paste	Pasts the character string copied in the clipboard to the caret position. This item becomes valid only when the character string is being edited.
Select All	Selects all the items of this panel.
Find…	Opens the Search and Replace dialog box with selecting the [Whole Search] tab.
Replace…	Opens the Search and Replace dialog box with selecting the [Whole Replace] tab.

[Context menu]

Register to Watch1	Registers the selected register or category to the Watch panel (Watch1).
Copy	Copies the selected character string to the clipboard during editing. If a line is selected, copies the register or the category to the clipboard. The copied item can be pasted to the Watch panel.

Notation		The following cascade menus to specify the notation of a data value are displayed.
	AutoSelect	Displays the value of the selected item (including sub-items) in the default notation (default).
	Hexadecimal	Displays the value of the selected item (including sub-items) in hexadecimal number.
	Signed Decimal	Displays the value of the selected item (including sub-items) in signed decimal number.
	Unsigned Decimal	Displays the value of the selected item (including sub-items) in unsigned decimal number.
	Octal	Displays the value of the selected item (including sub-items) in octal number.
	Binary	Displays the value of the selected item (including sub-items) in binary number.
	ASCII	Displays the character string of the selected item (including sub-items) in ASCII code. If the character size is 2 bytes and above, it is displayed with the characters for each 1 byte arranged side-by-side.
	Float	Displays the value of the selected item in Float. Note that when the value is not 4-byte data, displays it in the default notation.
	Double	Displays the value of the selected item in Double. Note that when the value is not 8-byte data, displays it in the default notation.
	Include Hexadecimal Value	Adds the value in hexadecimal number enclosing with "()" at the end of the value.

IOR panel

This panel is used to display the contents of the I/O register and change the I/O register values (see "2.9.3   Display/ change the I/O register").

This panel appears only when connected to the debug tool.

Caution   **The I/O register that cause the microcontroller to operate when it is read is read-protected and therefore cannot be read ("?" is displayed in the value).**
To read the value of read-protected I/O register, select [Force Read Value] from the context menu.

Remark   When the separator line of each area in this panel is double-clicked, the width of the area changes to the shortest possible size that can display the contents of the area.

Figure A-43.   IOR Panel
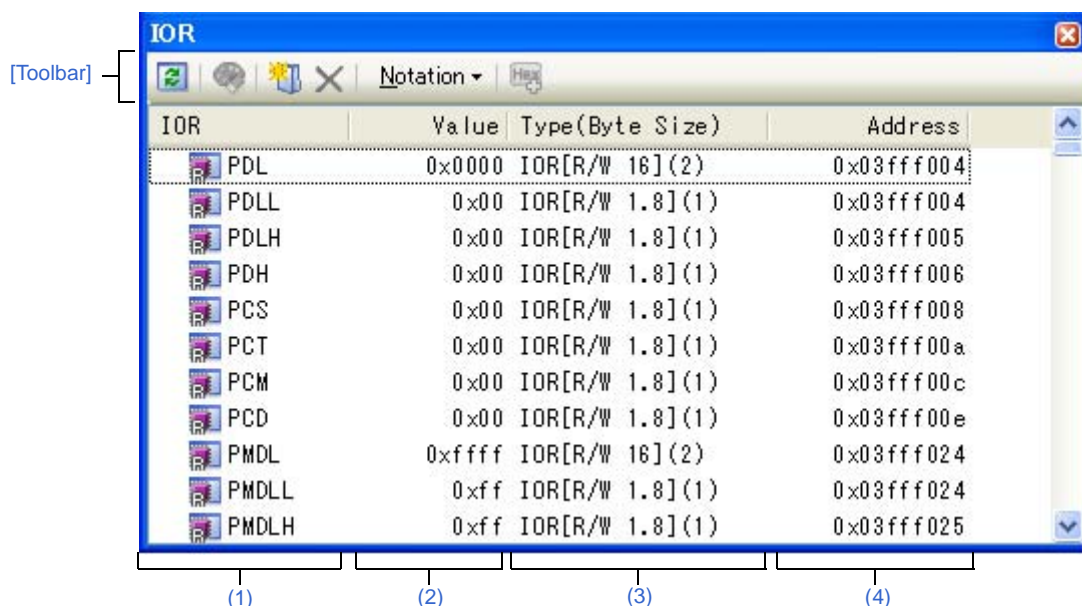
This section describes the following.

- [How to open]
- [Description of each area]
- [Toolbar]
- [[File] menu (IOR panel-dedicated items)]
- [[Edit] menu (IOR panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [IOR].

[Description of each area]

(1) [IOR] area

The types of I/O register are classified as categories (folders), and a list of the respective I/O register name is displayed.

The meanings of the icons are as follows:

📁	Indicates that the I/O register name belonging to this category is displayed. When you double-click on the icon, or click on the "-" mark, the category is closed and the I/O register name is hidden. Note that no categories exist by default. Perform Tree editing if you need a category.
📁	Indicates that the I/O register name belonging to this category is hidden. When you double-click on the icon, or click on the "+" mark, the category is opened and the I/O register name is displayed. Note that no categories exist by default. Perform Tree editing if you need a category.
🖼	Indicates the name of the I/O register.

Remark   The category names are sorted in character code order by clicking on the header part of this area (the I/O register names in the category are also similarly sorted).

This area is provided with the following functions.

(a) Tree editing

The each I/O register can be categorized (by folders) and displayed in the tree view.

To create a category, Click the 🖼 button on the toolbar or select [Create Category] from the context menu after moving the caret to a I/O register name to create a category, and then input a desired name from the keyboard (up to 1024 characters).

To delete a category, select the category then click the ✖ button on the toolbar or select [Delete] from the context menu. However, the categories that can be deleted are only the empty categories.

To rename the created category, select the category then do either one of the following.

- Click the name again, then directly rename the category name.
- Select the [Edit] menu >> [Rename], then directly rename the category name.
- Press the [F2] key, then directly rename the category name.

By directly dragging and dropping the I/O register in the created category, each I/O register is displayed in the categorized tree view.

Also, the display order of the categories and the I/O register names (upper or lower position) can be changed easily by drag and drop operation.

Cautions 1.   Categories cannot be created within categories.

2.   I/O registers cannot be added or deleted.

(b) Registering a watch-expression

Variable names of C language, CPU registers, I/O registers, and assembler symbols can be registered in the Watch panel as watch-expressions.

See "(1) Register a watch-expression" for details on how to operate it.

Remarks 1.   When you have registered a watch-expression with a category as the object, all of I/O registers belonging to that category are registered as watch-expressions.

2.   A scope specification is automatically added to a registered watch-expression.

(2) [Value] area

The value of I/O register is displayed and changed.

The radix of a data value can be selected by the button on the toolbar or the context menu item. In addition, a display format adding the value in hexadecimal number constantly can also be selected as well.

The meanings of the marks and colors displayed as I/O register values are as follows (character colors and background colors depend on the configuration in the [General - Font and Color] category of the Option dialog box):

Display Example (Default)			Description
0x0	Character color	Blue	The value of the I/O register that the user is changing (press the [Enter] key to write to the target memory).
	Background color	Standard color	
0x0	Character color	Brown	The value of the I/O register that has been changed because of the execution of a program To reset the highlighting, select the ![icon] button on the toolbar or [Reset Color] from the context menu.
	Background color	Cream	
?	Character color	Gray	The value of the I/O register that is a read-protected object[Note]
	Background color	Standard color	

Note   An I/O register for which the microcontroller ends up being activated by a read operation is shown.
To read the value of read-protected I/O register, select [Force Read Value] from the context menu.

Caution   **The timing for acquiring the values differs in the case of a 1 byte/2 bytes I/O register and that of 1 bit I/O registers that have been allocated to a 1 byte/2 bytes I/O register. Owing to this, there are also cases where the values differ even if the value of the same I/O register is displayed.**

Remark   The values are sorted in ascending order of the numerical values by clicking on the header part of this area.

This area is provided with the following functions.

(a) Changing I/O register values

To edit the I/O register value, change the value directly from the keyboard after double-clicking on the value to be edited (press the [Esc] key to cancel the edit mode).

After you edit the value of the I/O register, it is written to the register of the debug tool by pressing the [Enter] key, or moving the focus to outside the edit region.

Note that it is not possible to change the values of I/O registers during program execution.

You can use ASCII characters also to the values of I/O registers.

Examples 1.   When the numeric "0x41" is written to the I/O register "P0"
    >> "0x41" is written in the port "P0".
    2.   When the ASCII character "'A'" is written to the I/O register "P0"
    >> "0x41" is written in the port "P0".

Caution   **The value of the read-only I/O register cannot be edited.**

(b) Saving the contents of the I/O register

The Save As dialog box can be opened by selecting the [File] menu >> [Save *OR* Data As...], and all the contents of the I/O register can be saved in a text file (*.txt) or CSV file (*.csv).

See "(5) Save the I/O register contents" for details on the method for saving the contents of the I/O register.

(3) [Type (Byte Size)] area

The type information of each I/O register is displayed in the following formats.

- *<Type of I/O register>* [*<Access attribute>* *<All accessible sizes>*](*<Size>*)

Access attribute	One of the following is displayed as the access attribute.	
	R	Read only
	W	Write only
	R/W	Read/Write
All accessible sizes	All accessible sizes are demarcated by a comma and listed in order of the smallest size in bit units.	
Size	The size of the I/O register is displayed.	
	It is displayed by supplying the unit, in byte units in the event that it can be displayed in byte units, and in bit units in the event that it can be displayed on in bit units.	

Examples 1. 「The case of "IOR [R/W 1.8] (1 byte)"

An I/O register that is readable/writable and 1 bit accessible/8 bit accessible, and whose size is 1 byte

2. 「The case of "IOR [R/W 1] (1 bit)"

An I/O register that is readable/writable and 1 bit accessible, and whose size is 1 byte

Remark   The type information is sorted in the character code order by clicking on the header part of this area.

(4) [Address] area

The address that each I/O register is mapped is displayed (hexadecimal number notation fixing).

However, in the case of the bit register, it is displayed by providing a bit offset value like the following examples.

Examples 1.   The case of "0xFF40"

This is allocated to the address "OxFF40"

2.   The case of "0xFF40.4"

This is allocated to bit 4 of the address "0xFF40.4" (bit register)

Remark   The addresses are sorted in ascending order of numerical values by clicking on the header part of this area.

[Toolbar]

	Acquires the latest data from the debug tool, and updates the contents of this panel. Note that the values of read-protected I/O register are not re-read. This item becomes invalid during execution of a program.
	Resets highlighting of the selected I/O register whose value has been changed by executing a program. Note that this item becomes invalid during execution of a program.
	Adds a new category (folder). Directly input the category name in the text box. There are no restrictions on the number of categories that can be created anew (however, it is not possible to create a category inside a category). Note that this item becomes invalid during execution of a program.

✕		Deletes the selected character string(s).
		If an empty category is in a select state, its category is deleted (it is not possible to delete I/O registers).
Notation		The following buttons to change the notation of a data value are displayed.
	Hex	Displays the value of the selected item in hexadecimal number (default).
	S Dec	Displays the value of the selected item in signed decimal number.
	U Dec	Displays the value of the selected item in unsigned decimal number.
	Oct	Displays the value of the selected item in octal number.
	Bin	Displays the value of the selected item in binary number.
	Asc	Displays the value of the selected item in ASCII code.
	Hex+	Adds the value in hexadecimal number enclosing with "()" at the end of the value of the selected item.

[[File] menu (IOR panel-dedicated items)]

The following items are exclusive for the [File] menu in the IOR panel (other items are common to all the panels).
Note that all these items are invalid during execution of a program.

Save IOR Data	Overwrites the contents of this panel to the previously saved text file (*.txt)/CSV file (*.csv) (see "(b) Saving the contents of the I/O register").
	Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save IOR Data As...].
Save IOR Data As...	Opens the Save As dialog box to newly save the contents of this panel to the specified text file (*.txt)/CSV file (*.csv) (see "(b) Saving the contents of the I/O register").

[[Edit] menu (IOR panel-dedicated items)]

The following items are exclusive for [Edit] menu in the   IOR panel (other items are all invalid).

Cut	Deletes the selected character string(s) and copies them to the clipboard (it is not possible to cut I/O registers/categories).
Copy	Copies the contents of the selected range to the clipboard as character string(s).
	If the I/O register(s)/category(s) are selected, copies them to the clipboard.
	The copied item can be pasted to the Watch panel.
Paste	If texts are in editing, pastes the contents of the clipboard to the caret position (it is not possible to paste I/O registers/categories).
Delete	Deletes the selected character string(s).
	If an empty category is in a select state, its category is deleted (it is not possible to delete I/O registers).
Select All	If texts are in editing, selects all the character strings.
	If texts are not in editing, selects all the I/O registers/categories.
Rename	Edits the name of the selected category.
Find...	Opens the Search and Replace dialog box with selecting [Whole Search] tab.
Replace...	Opens the Search and Replace dialog box with selecting [Whole Replace] tab.
Move...	Opens the Go to the Location dialog box to move the caret to the specified I/O register.

[Context menu]

Register to Watch1	Registers the selected I/O register or category to the Watch panel (Watch1).
Refresh	Acquires the latest data from the debug tool, and updates the contents of this panel. Note that the values of read-protected I/O register are not re-read. This item becomes invalid during execution of a program.
Force Read Value	Forcibly reads once the value of the read-protected I/O register.
Move...	Opens the Go to the Location dialog box.
Create Category	Adds a new category (folder). Directly input the category name in the text box. There are no restrictions on the number of categories that can be created anew (however, it is not possible to create a category inside a category). Note that this item becomes invalid during execution of a program.
Copy	Copies the contents of the selected range to the clipboard as character string(s). If the I/O register(s)/category(s) are selected, copies them to the clipboard. The copied item can be pasted to the Watch panel.
Delete	Deletes the selected character string(s). If an empty category is in a select state, its category is deleted (it is not possible to delete I/O registers).
Notation	The following cascade menus are displayed to specify the notation.
Hexadecimal number	Displays the value of the selected item in hexadecimal number (default).
Signed Decimal	Displays the value of the selected item in signed decimal number.
Unsigned decimal number	Displays the value of the selected item in unsigned decimal number.
Octal	Displays the value of the selected item in octal number.
Binary	Displays the value of the selected item in binary number.
ASCII	Displays the value of the selected item in ASCII code.
Include Hexadecimal Value	Adds the value in hexadecimal number enclosing with "()" at the end of the value of the selected item.
Reset Color	Resets highlighting of the selected I/O register whose value has been changed by executing a program.

<div style="border:1px solid;">

Local Variables panel

</div>

This panel is used to display the contents of the local variable and change the local variable values (see "2.9.5 Display/change local variables").

This panel appears only when connected to the debug tool.

Cautions 1.   **Nothing is displayed on this panel during execution of a program. When the execution of a program is stopped, items in each area are displayed.**

         2.   **Due to compiler optimization, the data for the target variable may not be on the stack or in a register in blocks where that variable is not used. In this case, the target variable will not be displayed.**

Remark   When the separator line of each area in this panel is double-clicked, the width of the area changes to the shortest possible size that can display the contents of the area.

Figure A-44.   Local Variables Panel
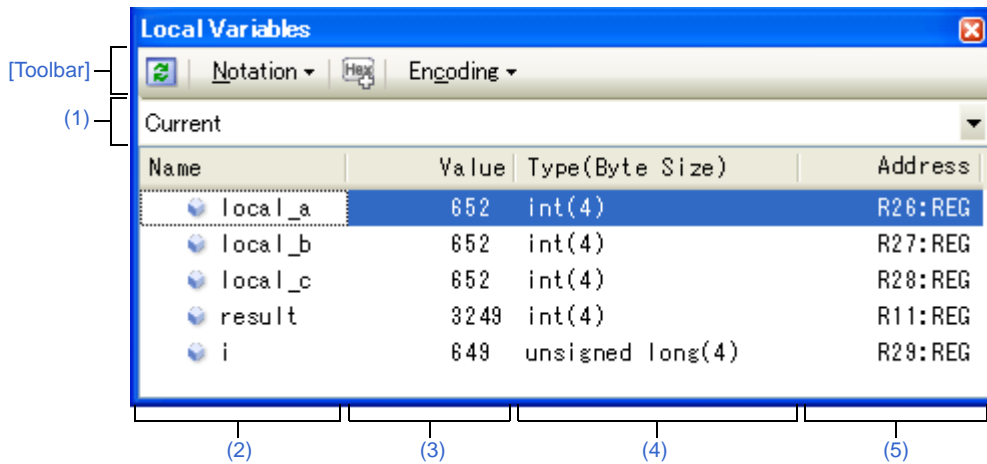
This section describes the following.

- [How to open]
- [Description of each area]
- [Toolbar]
- [[File] menu (Local Variables panel-dedicated items)]
- [[Edit] menu (Local Variables panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [Local Variable].

[Description of each area]

(1) Scope area

Select the scope of the local variable to be displayed from the following drop-down list.

Item	Operation
Current	Displays local variables in the scope of the current PC value.

Item	Operation
<Depth> <Function name() [*file name#line number*]>Note	Displays local variables in the scope of the calling function. After the program is executed, the scope that is selected is maintained as long as the selected scope exists.

Note The calling functions displayed in the Call Stack panel are displayed.

(2) [Name] area

The local variable name or function name is displayed.

The argument of the function is also displayed as the local variable.

In addition, the hierarchical structure is displayed in tree format for arrays, pointer variables, and structures or unions.

This area cannot be edited.

The meanings of the icons are as follows:

🔵	Indicates the variable. Auto, Internal Static, and Register variables are also displayedNote. In addition, the hierarchical structure is displayed in tree format for arrays, pointer variables, and structures or unions. If "+" mark exist at the top of the name, the next structure is expanded by clicking it (the mark changes to "-" after the expansion).	
	Array	All elements in the array
	Pointer variables	Variables that the pointer designates If the pointer designates a pointer, add "+" mark and expand it by clicking the mark. Note that if the pointer designates an unknown, "?" mark is displayed.
	Structures/Unions	All the member of structures/unions
🟡	Indicates the argument.	
🔴	Indicates the function.	

Note When Auto variables are used to display local variables, accurate values cannot be displayed at a prologue ("{") or epilogue ("}") of a function. (The Auto variable addresses are the relative addresses from the address pointed to by the stack pointer (SP), so their addresses are not determined until the SP value is determined in the function. The SP is manipulated via prologues or epilogues, so the accurate value cannot be displayed.

This area is provided with the following functions.

(a) Registering watch-expression

Variable names of C language can be registered in the Watch panel as watch-expressions.

See "(1) Register a watch-expression" for details on how to operate it.

Remark A scope specification is automatically added to a registered watch-expression.

(b) Jump to memory

By selecting [Jump to Memory] from the context menu, the Memory panel (Memory1) opens with moving the caret to the source line corresponding to the address where the selected local variable is disposed (if the Memory panel (Memory1) is already opened, jump to the panel).

(3) [Value] area

The value of the local variable is displayed and changed.

The notation of a data value can be selected by the button on the toolbar or the context menu item. In addition, a display format adding the value in hexadecimal number constantly can also be selected as well.

The meanings of the marks and colors displayed as the values of the local variables are as follows (character colors and background colors depend on the configuration in the [General - Font and Color] category of the Option dialog box):

Display Example (Default)			Description
0x0	Character color	Blue	The value of the local variable that the user is changing
	Background color	Standard color	Press the [Enter] key to write to the target memory.
0x0	Character color	Brown	The value of the local variable that have been changed because of the execution of a program[Note]
	Background color	Cream	The highlighting is rest by executing again the program.
?	Character color	Gray	When the value of the local variable cannot be acquired
	Background color	Standard color	

Note   Variables that the name stays same from the start point where the program started executing to the breakpoint and their values are changed are the target.

This area is provided with the following functions.

(a) Changing the local variable/argument value

To edit the local variable value or the argument value, select the value to edit, then change the value directly from the keyboard after clicking again on it (press the [Esc] key to cancel the edit mode).

After you edit the value of the local variable or the argument, it is written to the target memory of the debug tool by pressing the [Enter] key or moving the focus to outside the edit region.

See "(2) Modify the contents of local variables" for details on the method for changing the local variable/ argument value.

(b) Saving the contents of the local variable

The Save As dialog box can be opened by selecting the [File] menu >> [Save Local Variables Data As...], and all the contents of this panel can be saved in a text file (*.txt) or CSV file (*.csv).

See "(3) Save the contents of local variables" for details on the method for saving the contents of the local variable.

(4) [Type (Byte Size)] area

The type name of the local variable is displayed. The notation accords with the description of C language.

For an array, an element number is displayed in "[]". For a function, its size (number of bytes) is displayed in "()". This area cannot be edited.

(5) [Address] area

The address of the local variable is displayed. When a variable is assigned to the register, the name of the register is displayed.

This area cannot be edited.

[Toolbar]

The buttons below become invalid during execution of a program.

🔄	Acquires the latest data from the debug tool, and updates the contents of this panel.
Notation	The following buttons to specify the notation of values are displayed.
auto	Displays values on this panel in the default notation according to the type of variable (default).
Hex	Displays values on this panel in hexadecimal number.
Dec	Displays values on this panel in decimal number.
Oct	Displays values on this panel in octal number.
Bin	Displays values on this panel in binary number.
Dec	Displays array indexes on this panel in decimal number (default).
Hex	Displays array indexes on this panel in hexadecimal number.
Flo	Displays values on this panel in Float. Note that when the value is not 4-byte data, or has the type information, displays it in the default notation.
Dbl	Displays values on this panel in Double. Note that when the value is not 4-byte data, or has the type information, displays it in the default notation.
Hex	Adds the value in hexadecimal number enclosing with "()" at the end of the value.
Encoding	The following buttons to specify the encoding of character variables are displayed.
Asc	Displays character variables in ASCII code (default).
SJIS	Displays character variables in Shift_JIS code.
EUC	Displays character variables in EUC-JP code.
UTF	Displays character variables in UTF-8 code.

[[File] menu (Local Variables panel-dedicated items)]

The following items are exclusive for the [File] menu in the Local Variables panel (other items are common to all the panels).

Note that all these items are invalid during execution of a program.

Save Local Variables Data	Overwrites the contents of this panel to the previously saved text file (*.txt)/CSV file (*.csv) (see "(b) Saving the contents of the local variable"). Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save Local Variables Data As...].
Save Local Variables Data As...	Opens the Save As dialog box to newly save the contents of this panel to the specified text file (*.txt)/CSV file (*.csv) (see "(b) Saving the contents of the local variable").

[[Edit] menu (Local Variables panel-dedicated items)]

The following items are exclusive for [Edit] menu in the Local Variables panel (other items are all invalid).

Copy	Copies the contents of the selected line or the character string to the clipboard.
Select All	Selects all the items of this panel.
Rename	Changes to the edit mode to edit the selected local variable value (see "(2) Modify the contents of local variables"). This item becomes invalid during execution of a program.
Find...	Opens the Search and Replace dialog box with selecting the [Whole Search] tab.
Replace...	Opens the Search and Replace dialog box with selecting the [Whole Replace] tab.

[Context menu]

All the items from the context menu become invalid during execution of a program.

Register to Watch1	Registers the selected local variable to the Watch panel (Watch1).
Copy	Copies the contents of the selected line or the character string to the clipboard.
Notation	The following cascade menus to specify the notation of values are displayed.
AutoSelect	Displays values on this panel in the default notation according to the type of variable (default).
Hexadecimal	Displays values on this panel in hexadecimal number.
Decimal	Displays values on this panel in decimal number.
Octal	Displays values on this panel in octal number.
Binary	Displays values on this panel in binary number.
Decimal Notation for Array Index	Displays array indexes on this panel in decimal number (default).
Hexadecimal Notation for Array Index	Displays array indexes on this panel in hexadecimal number.
Float	Displays values on this panel in Float. Note that when the value is not 4-byte data, or has the type information, displays it in the default notation.
Double	Displays values on this panel in Double. Note that when the value is not 4-byte data, or has the type information, displays it in the default notation.
Include Hexadecimal Value	Adds the value in hexadecimal number enclosing with "()" at the end of the value.
Encoding	The following cascade menus to specify the encoding of character variables are displayed.
ASCII	Displays character variables in ASCII code (default).
Shift_JIS	Displays character variables in Shift_JIS code.
EUC-JP	Displays character variables in EUC-JP code.
UTF-8	Displays character variables in UTF-8 code.
Jump to Memory	Opens the Memory panel (Memory1) and jumps to the memory value corresponding to the address of the selected line in this panel.

Watch panel

This panel is used to display the contents of the registered watch-expressions and change their values (see "2.9.6 Display/change watch-expressions").

Up to a maximum of four of these panels can be opened. Each panel is identified by the names "Watch1", "Watch2", "Watch3", and "Watch4" on the title bar, and the watch-expressions can be registered/deleted/moved individually.

Watch-expressions can be registered in this panel as well as in the Editor panel, Disassemble panel, Memory panel, CPU Register panel, Local Variables panel or IOR panel.

When the panel is closed with registered watch-expressions, the panel closes but the information on the registered watch-expressions is retained. Therefore, if the same panel is opened again, it is opened with the watch-expressions registered.

The display contents are automatically updated when the value of the watch-expression changes after a program is executed (when the execution is done in steps, the display is updated after each step).

In addition, by enabling the Real-time display update function, it is also possible to update the display contents in real-time even while a program is being executed.

This panel appears only when connected to the debug tool.

Remark   When the separator line of each area in this panel is double-clicked, the width of the area changes to the shortest possible size that can display the contents of the area.

Figure A-45.   Watch Panel
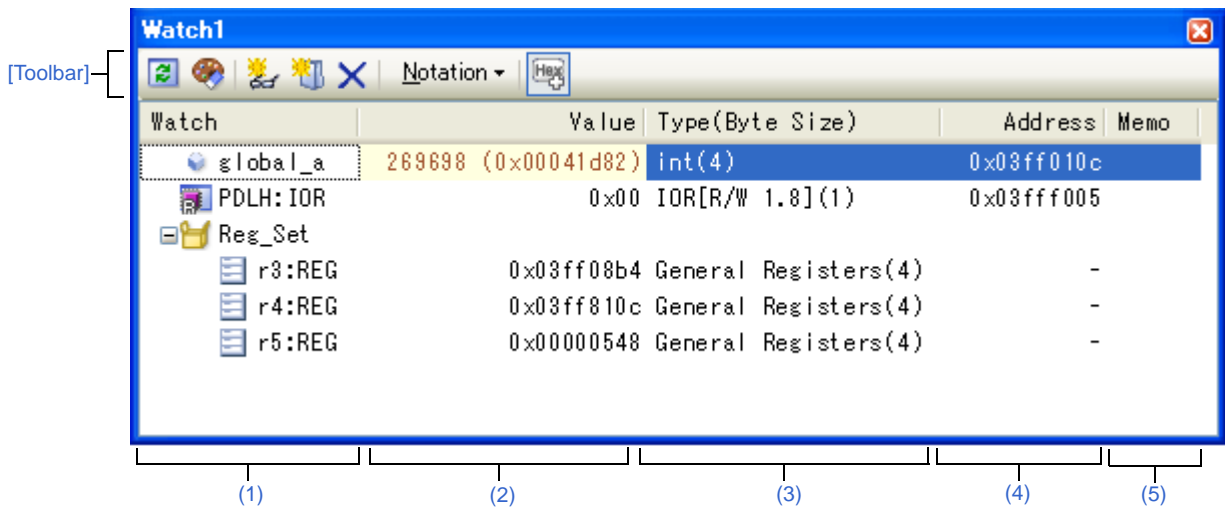
This section describes the following.

- [How to open]
- [Description of each area]
- [Toolbar]
- [[File] menu (Watch panel-dedicated items)]
- [[Edit] menu (Watch panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [Watch] >> [Watch *1 - 4*].

[Description of each area]

(1) [Watch] area

All the registered watch-expressions are displayed in a list.

Clicking the title of the list in this area sorts the watch-expressions in the list in alphabetical order.

Categories (folders) can be created to categorize the watch-expressions and display them in the tree view (see "(a) Tree editing").

The meanings of the icons are as follows:

	Indicates that the watch-expression belonging to this category is displayed. When you double-click on the icon, or click on the "-" mark, the category is closed and the watch-expression is hidden.
	Indicates that the watch-expression belonging to this category is hidden. When you double-click on the icon, or click on the "+" mark, the category is opened and the watch-expression is displayed.
	Indicates that the watch-expression is a variable. At the top of the watch-expression represents arrays, pointer type variables, and structures/unions, "+"/"-" mark is displayed. Click the mark to Expand/shrink display.
	Indicates that the watch-expression is a function.
	Indicates that the watch-expression is I/O register.
	Indicates that the watch-expression is CPU register. At the top of the watch-expression that has the lower level register (part of the register), "+"/"-" mark is displayed. Click the mark to Expand/shrink display.

This area is provided with the following functions.

(a) Tree editing

Watch-expressions can be categorized (by folders) and displayed in the tree view.

To create a category, click the [icon] button on the toolbar or select [Create Category] from the context menu after moving the caret to the position to create a category, and then input a desired name from the keyboard.

To delete a category, select the category then click the [icon] button on the toolbar or select [Delete] from the context menu.

To rename the created category, select the category then do either one of the following.
- Click the name again, then directly rename the category name.
- Select the [Edit] menu >> [Rename], then directly rename the category name.
- Press the [F2] key, then directly rename the category name.

By directly dragging and dropping the registered watch-expression in the created category, each category is displayed in the categorized tree view.

Also, the display order of the categories and the watch-expressions (upper or lower position) can be changed easily by drag and drop operation.

Cautions 1.   Categories cannot be created within categories.

        2.   Up to 64 categories can be created in one watch panel (if this restriction is violated, a message appears).

Remark   Drag and drop the watch-expressions/categories in other watch panel (Watch1 to Watch4) to copy them.

(b) Expand/shrink display

At the top of the watch-expression represents arrays, pointer type variables, structures/unions, and registers (with the name of the part), "+"/"-" mark is displayed. Click the mark to expand the contents ("+" mark is changed to "-" after the expansion).

Watch-Expression	Contents When Expanded
Array	All elements in the array Select [Encoding] >> [ASCII] from the context menu to display the value as a string (up to 256 characters). Note, however, that any characters that cannot be displayed in the encoding will be shown as periods ".".
Pointer type variable	Variables that the pointer designates
Structure/Union	All the member of structure/union

Register	Name of the bit/bit string that constructs register
	Example) ECR register
	   FECC register
	   EICC register

(c) Registering new watch-expression

There are three ways as follows to register new watch-expressions.

<1> Register from other panels

Do either one of the following to register watch-expressions in other panels.

- Drag and drop the target character string in this area in the desired watch panel (Watch1 to Watch4).
- Select [Register to Watch1] from the context menu after selecting the target character string or place the caret on either of the target character string (the target is automatically determined).
- Select the [Edit] menu >> [Paste] in this area in the desired watch panel (Watch1 to Watch4) after selecting the [Edit] menu >> [Copy] for the target character string.

The relationship between panels that can use this operation and targets that can be registered as watch-expressions is as follows:

Table A-7. Relationship between Panels and Targets That Can be Registered as Watch-Expressions

Panel Name	Targets That can be Registered as Watch-Expressions
Editor panel	Variable names of C language, CPU registers, I/O registers, and assembler symbols
Disassemble panel	Variable names of C language, CPU registers, I/O registers, and assembler symbols
CPU Register panel	CPU registers[Note]
Local Variables panel	Variable names of C language (local variables)
IOR panel	I/O registers[Note]

Note The scope-specification is automatically added to the registered watch-expression.

<2> Directly register in the Watch panel

Click the 🔆 button on the toolbar or select [Add New Watch] from the context menu in the desired watch panel (Watch1 to Watch4) to display an entry box for a new watch-expression in the bottom of this area.

Directly input a watch-expression from the keyboard in the [Watch] area in the entry box then press the [Enter] key.

The input format of the watch-expression is as follows:

Table A-8.   Input Format of Watch-Expression

Watch-Expression	Value to Display
Variable name of C language	Value of C language variable
Watch-expression[*Watch-expression*]	Element of array
Watch-expression Member name	Member of structures/unions
Watch-expression -> Member name	Member of structures/unions that pointer designates
*Watch-Expression	Value of pointer variable
CPU register name	Value of the CPU register
I/O register name	I/O register value
Label, EQU symbol and immediate address	Values of label, EQU symbol and immediate address
Bit symbol	Bit symbol value

Watch-expressions can be registered with specifying the scope. The scope specifications with watch-expression registration are as follows:

Table A-9.   Scope Specification of C language Used with Watch-Expression Registration

Scope Specification	Load Module File Name	Source File Name	Function Name	Variable Name
prog$file#func#var	prog	file	func	var
prog$file#var	prog	file	global	var
prog$var	prog	global	global	var
file#func#var	current	file	func	var
file#var	current	file	global	var
var	current	current	current	var

Table A-10.   Scope Specification of CPU Register with Watch-Expression Registration

Scope Specification	Register Bank	Name of CPU Register
r10:REG	-	r10

Table A-11.   Scope Specification of I/O register with Watch-Expression Registration

Scope Specification	Name of I/O register
P0:IOR	P0
P0	P0

Remarks 1.   If the same name exists either in C language variables, CPU registers or I/O register, and they are registered without specifying scopes, the symbol is resolved in the following order and the value is displayed.

Variable of C language > CPU registers > I/O register

If "$" is specified at the top of the watch-expression, the symbol is resolved in the following order and the value is displayed.

CPU registers > I/O register > Variable of C language

2.   If the same name exists in the local variable and the global variable, and their symbol names are only registered without specifying scopes, the symbol is resolved and the value is displayed based on the scope of the current PC value.

3.   [V850E2M]
System registers with the same name can be distinguished by the following scope specification.

Scope Specification	System Register Bank	System Register Name
PSW:SRBff00	CPU function group (0xff) / User 0 banks (0x00)	PSW
PSW:SRB0000	User group (0x00) / Main banks (0x00)	PSW

4.   When watch-expressions are registered from the IOR panel or the CPU Register panel, the scope specification is automatically added.

<3> Register from other application
Select a character string of a variable of C language, CPU register, I/O register or assembler symbol from a external editor then do either one of the following.

- Drag and drop the target character string in this area in the desired watch panel (Watch1 to Watch4).
- Select the [Edit] menu >> [Paste] in this area in the desired watch panel (Watch1 to Watch4) after copying the target character string.

Cautions 1.   Up to 128 watch-expressions can be registered in one watch panel (if this restriction is violated, a message appears).

2.   Due to compiler optimization, the data for the target variable may not be on the stack or in a register in blocks where that variable is not used. In this case, the target watch-expression value is displayed as "?".

Remarks 1.   Each watch-expression registered in each watch panel (Watch1 to Watch4) is managed in each panel and saved as the user information of the project.

2.   More than one watch-expression with the same name can be registered.

(d) Editing watch-expression
To edit the registered watch-expression, double-click the watch-expression to be edited to change the watch-expression to edit mode then directly edit from the keyboard (press the [Esc] key to cancel the edit mode). After editing the watch-expression, press the [Enter] key to complete the editing.

(e) Deleting watch-expression

To delete the registered watch-expression, select the watch-expression(s) to be deleted then click the ⊠ button on the toolbar or select [Delete] from the context menu.

(f) Setting of various events

Various events can be set to the selected watch-expression by selecting [Access Break] or [Trace Output] from the context menu.

If an access event is set, the mark of the watch-expression is changed (the event mark of a break event is displayed under the icon of the watch-expression in layers).

When an event is set, the detailed information about the set event is reflected in the Events panel.

Note that events are only set to the watch-expressions that are global variables, static variables inside functions, or file-internal static variables.

See the following for details on how to set events.

- "2.8.3   Stop the program with the access to variables/I/O registers"
- "2.11.4   Collect execution history when the condition is met"

(g) Jump to the address with memory definition

By selecting [Jump to Memory] from the context menu, the Memory panel (Memory1) opens with moving the caret to the address in which the selected watch-expression is defined (if the Memory panel (Memory1) is already opened, jump to the panel).

Note that this operation is disabled when more than one watch-expression is selected at the same time or the CPU register/I/O register is selected.

(2) [Value] area

The value of the registered watch-expression is displayed and changed (if the watch-expression is a function pointer, the function name is displayed in this area).

Notations and encodes can be selected by the button on the toolbar or the context menu item. In addition, a display format adding the value in hexadecimal number constantly can also be selected as well.

The default display format of the values is automatically decided depending on the type of the watch-expression.

Table A-12.   Display Format of Watch-Expressions (Default)

Type of Watch-Expression	Display Format
char, signed char, unsigned char	ASCII code with hexadecimal number
short, signed short, short int, signed short int, int, signed, signed int, long, signed long, long int, signed long int	Signed decimal number with hexadecimal number
unsigned short, unsigned short int, unsigned, unsigned int, unsigned long, unsigned long int	Unsigned decimal number with hexadecimal number
float	Float (when the size is 4-byte) with hexadecimal number
double, long double	Double (when the size is 8-byte) with hexadecimal number
Pointers to char, signed char, unsigned char	Characters Encoding: ASCII
Pointers to other than char, signed char, unsigned char	Hexadecimal number
Arrays of char, signed char, unsigned char types	Characters Encoding: ASCII
bit, boolean, _boolean	Unsigned decimal number with hexadecimal number
Enumeration type	Enumeration constant value with hexadecimal number

Type of Watch-Expression	Display Format
Label, address of immediate value, EQU symbol	Signed decimal number with hexadecimal number
bit symbol	Unsigned decimal number with hexadecimal number
Others	Hexadecimal number

The meanings of the marks and colors displayed as the values of watch-expressions are as follows (character colors and background colors depend on the configuration in the [General - Font and Color] category of the Option dialog box):

Display Example (Default)			Description
0x0	Character color	Blue	The value of the watch-expression that the user is changing
	Background color	Standard color	Press the [Enter] key to write to the target memory.
0x0	Character color	Pink	The value of the watch-expression that is displayed with the Real-time
	Background color	Standard color	display update function
0x0	Character color	Brown	The value of the watch-expression that has been changed because of the execution of a program
	Background color	Cream	To reset the highlighting, select the button on the toolbar or [Reset Color] from the context menu.
?	Character color	Gray	Variable that does not exist is registered as a watch-expression or the value of the watch-expression cannot be retrieved (variable is out of the scope)
	Background color	Standard color	

Remarks 1. The I/O register that cause the microcontroller to operate when it is read is read-protected and therefore cannot be read. To read the value of read-protected I/O register, select [Force Read Value] from the context menu.

2. Each watch-expression acquires the value in the order it was registered.
As the timing to acquire a value is different, the values displayed may be different if the same I/O register is registered more than once.

3. When a hexadecimal value is also given, then values in the specified notation and hexadecimal values are read separately. For this reason, the values with the specified notion and the hexadecimal values may differ due to the time lag between being read.

This area is provided with the following functions.

(a) Real-time display update function
Using the real-time display update function enables to display/modify the value of the watch-expression not only while the program is stopped, but also in execution.
See "(4) Display/modify the memory contents during program execution" for details on the real-time display update function.

(b) Changing values of watch-expressions
To edit the value of the watch-expression, change the value directly from the keyboard after double-clicking on the value to be edited (press the [Esc] key to cancel the edit mode).
After you edit the value of the watch-expression, it is written to the target memory of the debug tool by pressing the [Enter] key, or moving the focus to outside the edit region.
See "(6) Modify the contents of watch-expressions" for detail on how to change values of watch-expressions.

(c) Saving the contents of watch-expressions

By selecting the [File] menu >> [Save Watch Data As...], the Save As dialog box can be opened, and all the contents of this panel can be saved in a text file (*.txt) or CSV file (*.csv).

See "(8)   Save the contents of watch-expressions" for details on the method for saving the contents of watch-expressions.

(3) [Type (Byte Size)] area

The type information of watch-expressions with the following format is displayed.

Watch-Expression	Display Format	
Single CPU register	*<Types of CPU register>* (*<Size*[Note 1]*>*)	
Single I/O register	*<I/O register type>* (*<Access attribute>* *<Access type><Size*[Note 1]*>*)	
	Access attribute	R: Read only W: Write only R/W: Read/Write only
	Access type	1: Bit accessible 8: Byte accessible 16: Half word accessible 32: Word accessible
Unknown	?	
Others	*<Watch-expression type that follow the C compiler's determination*[Note 2]*>* (*<Size*[Note 1]*>*)	

Notes 1. The size of the watch-expression is displayed in bytes.

However, for bit I/O register or C language bit field, the size is displayed in bits and "bits" is added to the end of the number.

2. Types to be treated are displayed when compiling the watch-expression.

(4) [Address] area

The address that each watch-expression is mapped is displayed (hexadecimal number notation fixing).

If the watch-expression is single CPU register or is unknown, "-" or "?" is displayed instead.

Remark   When the watch-expression is the bit I/O register, the bit-offset value is also displayed.

Example   When the bit register is allocated to bit 4 of the address "0xFF40"
Display example:0xFF40.4

(5) [Memo] area

The user can write comments for the watch-expressions/categories.

Each comment for a watch-expression/category written in this area is saved individually as the user information of the project. Therefore, when any of the watch-expression/category is deleted, the comment corresponding to it is also deleted.

Note that when arrays or register are displayed expanded, the comment cannot be input for each element.

To edit the comment, input the character strings directly from the keyboard after double-clicking on the item to be edited (press the [Esc] key to cancel the edit mode).

Up to 256 character strings can be input (line feed code is ignored).

After editing the character strings, complete the editing by pressing the [Enter] key or moving the focus to outside the edit region.

[Toolbar]

🔄		Reacquires all the values of the registered watch-expression and updates the display. Note that read-protected I/O register values are not re-read. This item becomes invalid during execution of a program.
🎨		Resets highlighting of the selected watch-expression whose value has been changed by executing a program. This item becomes invalid during execution of a program.
📝		Registers a new watch-expression. Directly input the watch-expression in the text box (see "(c) Registering new watch-expression") Note that up to 128 watch-expressions can be registered in one watch panel.
📁		Adds a new category (folder). Directly input the category name in the text box. Note that up to 64 categories can be created in one watch panel (categories cannot be created in categories).
✖		Deletes the selected character string(s). If the watch-expression(s)/category(s) are selected, deletes them (except when the expanded item of the watch-expression is selected).
Notation		The following buttons to change the notation of a data value are displayed.
	auto	Displays the value of the selected watch-expression in the default notation (see "Table A-12. Display Format of Watch-Expressions (Default)") according the type of variable (default).
	Hex	Displays the value of the selected item in hexadecimal number.
	S Dec	Displays the value of the selected item in signed decimal number.
	U Dec	Displays the value of the selected item in unsigned decimal number.
	Oct	Displays the value of the selected item in octal number.
	Bin	Displays the value of the selected item in binary number.
	Asc	Displays the value of the selected item in ASCII code.
	Flo	Displays the value of the selected item in Float. Note that this item becomes valid only when the selected watch-expression value is 4-byte data.
	Dbl	Displays the value of the selected item in Double. Note that this item becomes valid only when the selected watch-expression value is 8-byte data.
Hex		Adds the value in hexadecimal number enclosing with "()" at the end of the value of the selected item (except the item displayed in hexadecimal number).

[[File] menu (Watch panel-dedicated items)]

The following items are exclusive for the [File] menu in the Watch panel (other items are common to all the panels).
Note that all these items are invalid during execution of a program.

Save Watch Data	Overwrites the contents of this panel to the previously saved text file (*.txt)/CSV file (*.csv) (see "(c) Saving the contents of watch-expressions"). Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save Watch Data As...].
Save Watch Data As...	Opens the Save As dialog box to newly save the contents of this panel to the specified text file (*.txt)/CSV file (*.csv) (see "(c) Saving the contents of watch-expressions").

[[Edit] menu (Watch panel-dedicated items)]

The following items are exclusive for [Edit] menu in the Watch panel (other items are all invalid).

Cut	Deletes the selected character string(s) and copies them to the clipboard.
	If the watch-expression(s)/category(s) are selected, deletes them (except when the expanded item of the watch-expression is selected).
Copy	Copies the contents of the selected range to the clipboard as character string(s).
	If the watch-expression(s)/category(s) are selected, copies them to the clipboard (except when the expanded item of the watch-expression is selected).
Paste	If texts are in editing, pastes the contents of the clipboard to the caret position.
	If texts are not in editing and the watch-expression(s) are copied in the clipboard, registers them to the caret position.
Delete	Deletes the selected character string(s).
	If the watch-expression(s)/category(s) are selected, deletes them (except when the expanded item of the watch-expression is selected).
Select All	If texts are in editing, selects all the character strings.
	If texts are not in editing, selects all the watch-expressions/categories.
Rename	Renames the selected watch-expression/category.
Find...	Opens the Search and Replace dialog box with selecting the [Whole Search] tab.
Replace...	Opens the Search and Replace dialog box with selecting the [Whole Replace] tab.

[Context menu]

Access Break	This item becomes valid only when the selected watch-expression is the global variable, the static variable inside functions, the file-internal static variable, or I/O register.
	The following cascade menus are displayed to set the access break event (see "(1) Set a break event to a variable/I/O register").
Set Read Break to	Sets a break event with read access condition to the selected watch-expression.
Set Write Break to	Sets a break event with write access condition to the selected watch-expression.
Set R/W Break to	Sets a break event with read/write access condition to the selected watch-expression.
Trace Output **[IECUBE][IECUBE2] [Simulator]**	This item becomes valid only when the selected watch-expression is the global variable, the static variable inside functions, the file-internal static variable, or I/O register.
	The following cascade menus are displayed to set the trace-related event (see "(1) When an access to variables/I/O registers occurs").
	The corresponding Event mark is displayed at the top of the watch-expression when an event is set.
Record Reading Value	Sets a Point Trace event to record the values in the trace memory when the selected watch-expression is accessed for read.
Record Writing Value	Sets a Point Trace event to record the values in the trace memory when the selected watch-expression is accessed for write.
Record R/W Value	Sets a Point Trace event to record the values in the trace memory when the selected watch-expression is accessed for read/write.
Trace	Opens the Trace panel [IECUBE][IECUBE2][Simulator] and displays the acquired trace data.

Periodic Updating		The following cascade menus are displayed to set for the real-time display update function (see "(a) Real-time display update function"). This item becomes invalid during execution of a program.
	Periodic Updating Options	Opens the Property panel to set for the real-time display update function.
Refresh		Reacquires all the values of the registered watch-expression and updates the display. Note that the values of read-protected I/O register are not re-read. This item becomes invalid during execution of a program.
Force Read Value		Forcibly reads once the values of the read-protected I/O register. This item becomes invalid during execution of a program.
Add New Watch		Registers a new watch-expression. Directly input the watch-expression in the text box (see "(c) Registering new watch-expression") Note that up to 128 watch-expressions can be registered in one watch panel.
Create Category		Adds a new category (folder). Directly input the category name in the text box. Note that up to 64 categories can be created in one watch panel (categories cannot be created in categories).
Delete		Deletes the selected character string(s). If the watch-expression(s)/category(s) are selected, deletes them (except when the expanded item of the watch-expression is selected).
Cut		Deletes the selected character string(s) and copies them to the clipboard. If the watch-expression(s)/category(s) are selected, deletes them (except when the expanded item of the watch-expression is selected).
Copy		Copies the contents of the selected range to the clipboard as character string(s). If the watch-expression(s)/category(s) are selected, copies them to the clipboard (except when the expanded item of the watch-expression is selected).
Paste		If texts are in editing, pastes the contents of the clipboard to the caret position. If texts are not in editing and the watch-expression(s) are copied in the clipboard, registers them to the caret position.

Notation		The following cascade menus are displayed to specify the notation.
	AutoSelect	Displays the value of the selected watch-expression in the default notation (see "Table A-12. Display Format of Watch-Expressions (Default)") according the type of variable (default).
	Hexadecimal number	Displays the value of the selected item in hexadecimal number.
	Signed Decimal	Displays the value of the selected item in signed decimal number.
	Unsigned decimal number	Displays the value of the selected item in unsigned decimal number.
	Octal	Displays the value of the selected item in octal number.
	Binary	Displays the value of the selected item in binary number.
	ASCII	Displays the value of the selected item in ASCII code.
	Include Hexadecimal Value	Adds the value in hexadecimal number enclosing with "()" at the end of the value of the selected item (except the item displayed in hexadecimal number).
	Float	Displays the value of the selected item in Float. Note that when the selected watch-expression value is not 4-byte data, or has the type information, displays it in the default notation (see "Table A-12. Display Format of Watch-Expressions (Default)").
	Double	Displays the value of the selected item in Double. Note that when the selected watch-expression value is not 8-byte data, or has the type information, displays it in the default notation (see "Table A-12. Display Format of Watch-Expressions (Default)").
	Decimal Notation for Array Index	Displays array indexes on this panel in decimal number (default).
	Hexadecimal Notation for Array Index	Displays array indexes on this panel in hexadecimal number.
Encoding		The following cascade menus are displayed to specify the character code.
	ASCII	Displays the value of the selected item in ASCII code (default).
	Shift_JIS	Displays the value of the selected item in Shift_JIS code.
	EUC-JP	Displays the value of the selected item in EUC-JP code.
	UTF-8	Displays the value of the selected item in UTF-8 code.
Size Notation		The following cascade menus are displayed to specify the size notation.
	1 Bytes	Displays the value of the selected item as 8-bit data.
	2 Bytes	Displays the value of the selected item as 16-bit data.
	4 Bytes	Displays the value of the selected item as 32-bit data.
Jump to Memory		Opens the Memory panel (Memory1) and jumps to the address which the selected watch-expression is defined (see "(g) Jump to the address with memory definition").
Reset Color		Resets highlighting of the selected watch-expression whose value has been changed by executing a program. This item becomes invalid during execution of a program.

Call Stack panel

This panel is used to display the call stack information for the function call (see "2.10.1   Display call stack information").
This panel appears only when connected to the debug tool.

Caution   **Nothing is displayed on this panel during execution of a program.**
When the execution of a program is stopped, items in each area are displayed.

Remark   When the separator line of each area in this panel is double-clicked, the width of the area changes to the
shortest possible size that can display the contents of the area.

Figure A-46.   Call Stack Panel
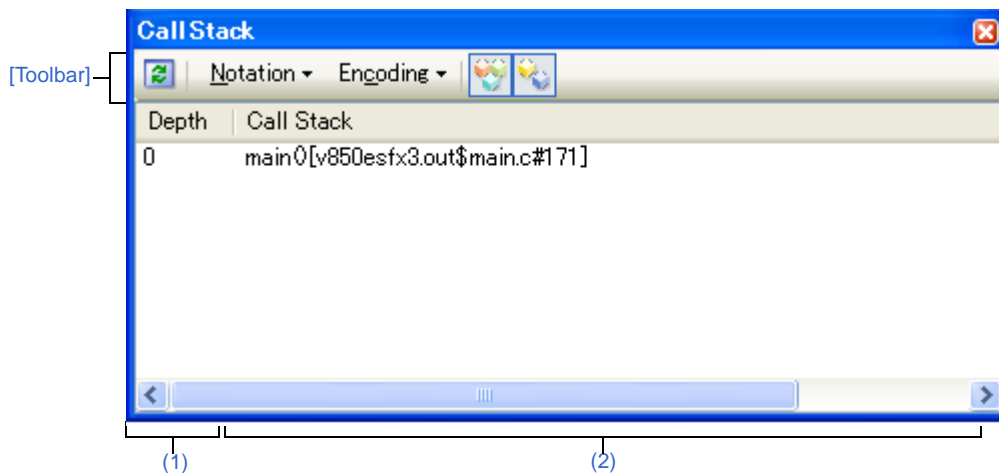
This section describes the following.
- [How to open]
- [Description of each area]
- [Toolbar]
- [[File] menu (Call Stack panel-dedicated items)]
- [[Edit] menu (Call Stack panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [Call Stack].

[Description of each area]

(1) [Depth] area
The depth of the call is displayed.
The line at the current PC position becomes 0 and incremented numbers from 1 is added to the calling function in
the order.

(2) [Call Stack] area
The current source position and the call stack information pushed on the stack (position of the calling function and
arguments of a each function, etc.) are displayed.
The display format in this area differs depending on the selection condition of the ⬛/⬛ button on the toolbar,
or of [Show Parameter]/[Show Module File Name] from the context menu.

Condition	Display Format
- Display arguments - Display module file name	*<Function>*(*<Argument>*=*<Argument Value*^{Note}*>*, ...)[*<Module file name>*$*<File name>*#*<Line number>*] (default)
- Display arguments - Do not display module file name	*<Function>*(*<Argument>*=*<Argument value*^{Note}*>*, ...)[*<File Name>*#*<Line number>*]
- Do not display arguments - Display module file name	*<Function>*()[*<Module file name>*$*<File name>*#*<Line number>*]
- Do not display arguments - Do not display module file name	*<Function>*()[*<File name>*#*<Line number>*]

Note　When the argument value is character string, up to 20 characters can be displayed.

Remark　Array arguments are passed as pointers rather than arrays (C language specification). For this reason, if the argument is an array, it is displayed as a pointer.

This area is provided with the following functions.

(a) Jump to source line and disassemble

By selecting [Jump to Source] from the context menu, the Editor panel is opened with moving the caret to the source line corresponding to the calling function at the current caret position (if the Editor panel is already opened, jump to the panel).

In addition, similarly by selecting [Jump to Disassemble], the Disassemble panel (Disasemmble1) is opened with moving the caret to the address corresponding to the calling function at the current caret position (if the Disassemble panel is already opened, jump to the panel (Disassemble1)).

Remark　It is possible to jump to the target source line by double-clicking on that line as well.

(b) Saving the contents of call stack information

By selecting the [File] menu >> [Save Call Stack Data As...], the Save As dialog box can be opened, and all the contents of this panel can be saved in a text file (*.txt) or CSV file (*.csv).

See "(4)　Save the contents of call stack information" for details on the method for saving the contents of call stack information.

[Toolbar]

The buttons below become invalid during execution of a program.

[icon]	Acquires the latest data from the debug tool, and updates the contents of this panel.
Notation	The following buttons to specify the notation of values are displayed.
[auto]	Displays values on this panel in the default notation according to the type of variable (default).
[Hex]	Displays values on this panel in hexadecimal number.
[Dec]	Displays values on this panel in decimal number.
[Oct]	Displays values on this panel in octal number.
[Bin]	Displays values on this panel in binary number.
Encoding	The following buttons to specify the encoding of character variables are displayed.
[Asc]	Displays character variables in ASCII code (default).
[SJIS]	Displays character variables in Shift_JIS code.
[EUC]	Displays character variables in EUC-JP code.
[UTF]	Displays character variables in UTF-8 code.
[icon]	Displays the call stack information with the module file name (default).
[icon]	Displays the call stack information with the parameters (arguments) of the function call (default).

[[File] menu (Call Stack panel-dedicated items)]

The following items are exclusive for the [File] menu in the Call Stack panel (other items are common to all the panels).
Note that all these items are invalid during execution of a program.

Save Call Stack Data	Overwrites the contents of this panel to the previously saved text file (*.txt)/CSV file (*.csv) (see "(b) Saving the contents of call stack information"). Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save Call Stack Data As...].
Save Call Stack Data As...	Opens the Save As dialog box to newly save the contents of this panel to the specified text file (*.txt)/CSV file (*.csv) (see "(b) Saving the contents of call stack information").

[[Edit] menu (Call Stack panel-dedicated items)]

The following items are exclusive for [Edit] menu in the Call Stack panel (other items are all invalid).

Copy	Copies the contents of the selected line to the clipboard.
Select All	Selects all the items of this panel.
Find...	Opens the Search and Replace dialog box with selecting the [Whole Search] tab.
Replace...	Opens the Search and Replace dialog box with selecting the [Whole Replace] tab.

[Context menu]

All the items from the context menu become invalid during execution of a program.

Copy	Copies the contents of the selected line to the clipboard.
Show Module File Name	Displays the call stack information with the module file name (default).
Show Parameter	Displays the call stack information with the parameters (arguments) of the function call (default).
Notation	The following cascade menus to specify the notation of values are displayed.
AutoSelect	Displays values on this panel in the default notation according to the type of variable (default).
Hexadecimal	Displays values on this panel in hexadecimal number.
Decimal	Displays values on this panel in decimal number.
Octal	Displays values on this panel in octal number.
Binary	Displays values on this panel in binary number.
Encoding	The following cascade menus to specify the encoding of character variables are displayed.
ASCII	Displays character variables in ASCII code (default).
Shift_JIS	Displays character variables in Shift_JIS code.
EUC-JP	Displays character variables in EUC-JP code.
UTF-8	Displays character variables in UTF-8 code.
Jump to Disassemble	Opens the Disassemble panel (Disassemble1) and jumps to the address corresponding to the calling function of the selected line in this panel.
Jump to Source	Opens the Editor panel and jumps to the source line corresponding to the calling function of the selected line in this panel.
Jump to Local Variable at This Time	Opens the Local Variables panel to display the local variable corresponding to the selected line.

Trace panel [IECUBE][IECUBE2][Simulator]

This panel is used to display trace data recording the execution history of the program (see "2.11   Collect Execution History of Programs [IECUBE][IECUBE2][Simulator]").

The trace data displays by mixing the disassembled text and source text by default, but it is also possible to display either one of these by selecting the Display mode.

After the execution of the program is stopped, the display position is automatically updated such that the latest trace data is displayed.

This panel appears only when connected to the debug tool.

Cautions 1.   [IECUBE]
Some of the trace function, the real-time display update function (RRM function) and the coverage function are used on a mutually exclusive basis.
2.   [IECUBE2]
Some of the trace function, the timer function (except for the Run-Break timer) and the coverage function are used on a mutually exclusive basis.

Remark   When the separator line of each area in this panel is double-clicked, the width of the area changes to the shortest possible size that can display the contents of the area.

Figure A-47.   Trace Panel
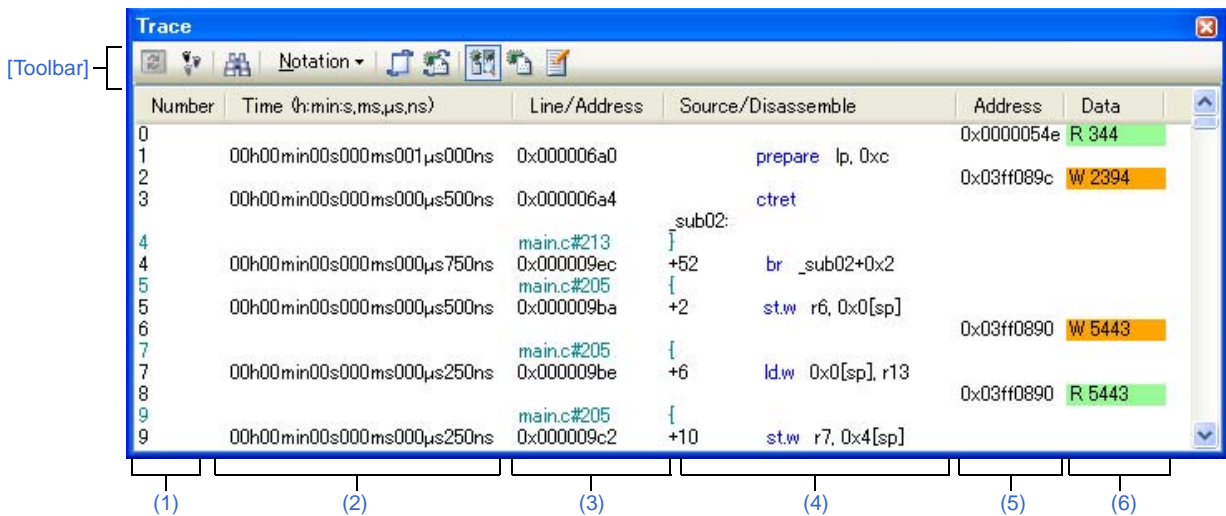
This section describes the following.
- [How to open]
- [Description of each area]
- [Toolbar]
- [[File] menu (Trace panel-dedicated items)]
- [[Edit] menu (Trace panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [Trace].
- On the Editor panel/Disassemble panel, select [Trace Settings] >> [Show Trace Result] from the context menu.

[Description of each area]

(1)   [Number] area

The trace number showing the position is displayed.

(2)   [Time (h:min:s,ms,μs,ns)] area

This area displays the time required from the execution start of the program to the execution start of an instruction of each frame or generation of memory access cause.

The time is displayed in units of "hours, minutes, seconds, milliseconds, microseconds and nanoseconds".

If overflow occurs, this area is displayed in invalid color (gray)

Remarks 1.   [IECUBE][IECUBE2]

The precision of the time depends on the setting of the [Rate of frequency division of trace time tag] property on the [Trace] category in the [Debug Tool Settings] tab of the Property panel.

2.   [Simulator]

The question of whether to set the time display as an integrated value or differential value depends on the setting of the [Accumulate trace time] property on the [Trace] category in the [Debug Tool Settings] tab of the Property panel.

(3)   [Line/Address] area

The address of the assemble code or the line number of a source file is displayed.

The notation of a data value can be selected by the button on the toolbar or the context menu item.

The display formats are as follows:

Type of Display Line	Display Format
Instruction (disassemble results)	*<Address>*
Source text	*<File name>#<Line number>*
Label	-
Point trace results	-
Break cause **[IECUBE2] [Simulator]**	-

Remark   Since the following execution histories are not displayed, the line numbers displayed are not consecutive numbers.
- CPU register access
- Operand access
- Invalid fetch

(4)   [Source/Disassemble] area

The collected trace data is displayed as follows.

The items displayed in this area differ depending on the selection of the display mode (see "(a)   Display mode").

Figure A-48.   Display Contents of the [Source/Disassemble] Area (Default)

Label	The label is displayed when a label is defined for the address.
Offset value	The offset value from the nearest label is displayed when a label is defined for the address.
Instruction (disassemble results)	The corresponding instructions are displayed as the result of disassembling when the Mixed display mode or Disassemble display mode is selected[Note 1]. The mnemonics are shown highlighted.
Source text	The corresponding source text is displayed when the Mixed display mode or Source display mode is selected. In addition, when the value of a variable[Note 2] or an I/O register that is accessed during execution of a source line can be analyzed, that value is displayed in the following format at the end of the source line. - <<<Variable name = Variable value>>> - <<<I/O register name = I/O register value>>> Example: a=b; <<<a=5>>> The results of the Point Trace are displayed as same as format above. However, when a place where no debugging information is present is executed, "<No Debug Information>" is displayed.
Break cause **[Simulator]**	The reason why the program has broken down is displayed.

Notes 1.   At a frame for which not all the trace data was fetched, "(LOST)" is displayed. In this case, the corresponding line is shown in error color (the error color depends on the configuration in the [General - Font and Color] category of the Option dialog box).

2.   When there is a memory access, a symbol will be interpreted as a variable and displayed only if a symbol is assigned to the accessed address. Note, however, that only variables of up to 4 bytes are supported. If multiplication or other code is processed by the standard libraries, the label of the SADDR area used by the standard library may be shown.

This area is provided with the following functions.

(a) Display mode

It is possible to select the following three display modes by selection of a button on the toolbar or the context menu.

Display Mode	Display Contents
Mixed display mode	Displays the instruction (disassemble results), labels, source text (corresponding source line), point trace results, and break causes (default).
Disassemble display mode	Displays the instruction (disassemble results), labels, point trace results, and break causes.
Source display mode	Displays the source text (corresponding source line) and break causes. However, when a place where no debugging information is present is executed, "<No Debug Information>" is displayed.

(b) Jumping to source line or disassemble

By selecting [Jump to Source] from the context menu, the Editor panel opens with moving the caret to the source line corresponding to the line at the current caret position (if the Editor panel is already opened, jump to the panel).

In addition, similarly by selecting [Jump to Disassemble], the Disassemble panel (Disasemmble1) is opened with moving the caret to the address corresponding to the fetch address of the line at the current caret position (if the Disassemble panel is already opened, jump to the panel (Disassemble1)).

(c) Linking with other panels

By clicking the ⬚/⬚ button on the toolbar, or selecting [Window Connecting] >> [Connect Source Window]/[Connect Disassemble Window] from the context menu, it is possible to link and display the corresponding places on the Editor panel/Disassemble panel, with the address of the caret position on this panel used as the pointer (no movement of the focus is done).

(d) Pop-up display

By superimposing the mouse cursor on a line, all the area (item) data corresponding to that line is displayed as a pop-up in tandem shape.

(e) Saving trace data

The Data Save dialog box can be opened by selecting the [File] menu >> [Save Trace Data As...], and the contents of this panel can be saved in a text file (*.txt) or CSV file (*.csv).

See "2.11.8   Save the contents of execution history" for details on the method for saving trace data.

(5) [Address] area

The target address of memory access is displayed.

However, in the event of access to I/O register, the I/O register name is displayed instead of the address (when a plurality is accessed these are displayed in the following lines).

The radix of a data value can be selected by the button on the toolbar or the context menu item.

Remark   [IECUBE2]

To display the target address of memory access via DMA, select [Yes] with the [DMA is traced] property in the [Trace] category on the [Debug Tool Settings] tab of the Property panel.

(6) [Data] area

The accessed data value and the access type at that time are displayed.

However, CPU register access is not displayed.

The notation of a data value can be selected by the button on the toolbar or the context menu item.

The display format of the data value and the access type are as follows (character colors and background colors depend on the configuration in the [General - Font and Color] category of the Option dialog box):

Display Example (Default)			Memory Access Type
R*Data value*	Character color	Standard color	Read access
	Background color	Palegreen	
W*Data value*	Character color	Standard color	Write access
	Background color	Orange	
RW*Data value*	Character color	Standard color	Read and write access
	Background color	Paleturquoise	
VECT*Data value*	Character color	Standard color	Vector read access
	Background color	Palegreen	

Remarks 1.   **[IECUBE]**

If an I/O register access or a memory access is generated via DMA, "DMA" will be shown immediately after the data value of the frame where the access started/ended.

2.   **[IECUBE2]**

To display the data value accessed via DMA, select [Yes] with the [DMA is traced] property in the [Trace] category on the [Debug Tool Settings] tab of the Property panel. In this case, the DMA trace data specified with the [Select DMA trace data] property in the category same as above is displayed in the following format:

> <*Access type*><*Data value*><DMA Channel number=*DMA Channel number*><Size=*Data size*><Transfer Count=*Transfer count number*>

[Toolbar]

🔃	Acquires the latest data from the debug tool, and updates the contents of this panel. This item becomes invalid during execution of a program.
✂	Clears the trace memory and the display of this panel (initialized). This item becomes invalid during execution of a program.
🔍	Opens the Trace Search dialog box [IECUBE][IECUBE2][Simulator].
Notation	The following buttons to change the notation of a data value are displayed. This item becomes invalid during execution of a program.
Hex	Displays values on this panel in hexadecimal number (default).
Dec	Displays values on this panel in decimal number.
Oct	Displays values on this panel in octal number.
Bin	Displays values on this panel in binary number.

📄		Links with the Editor panel.
📑		Links with the Disassemble panel.
📊		Sets to the Mixed display mode as the display mode (default). This item becomes invalid during execution of a program.
📄		Sets to the Disassemble display mode as the display mode. This item becomes invalid during execution of a program.
📝		Sets to the Source display mode as the display mode. This item becomes invalid during execution of a program.

[[File] menu (Trace panel-dedicated items)]

The following items are exclusive for the [File] menu in the Trace panel (other items are common to all the panels).

Note that all these items are invalid during execution of a program.

Save Trace Data	Overwrites the contents of this panel to the previously saved text file (*.txt)/CSV file (*.csv) (see "(e) Saving trace data"). Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save Trace Data As...].
Save Trace Data As...	Opens the Data Save dialog box to newly save the contents of this panel to the specified text file (*.txt)/CSV file (*.csv) (see "(e) Saving trace data").

[[Edit] menu (Trace panel-dedicated items)]

The following items are exclusive for [Edit] menu in the Trace panel (other items are all invalid).

Note that all these items are invalid during execution of a program.

Copy	Copies the contents of the selected line to the clipboard (multiple line selections impossible).
Find...	Opens the Trace Search dialog box [IECUBE][IECUBE2][Simulator].

[Context menu]

Clear Trace	Clears the trace memory and the display of this panel (initialized). This item becomes invalid during execution of a program.
Find...	Opens the Trace Search dialog box [IECUBE][IECUBE2][Simulator]. This item becomes invalid during execution of a program.
Copy	Copies the contents of the selected line to the clipboard (multiple line selections impossible). This item becomes invalid during execution of a program.
Mixed Display	Sets to the Mixed display mode as the display mode. This item becomes invalid during execution of a program.
Disassemble View	Sets to the Disassemble display mode as the display mode. This item becomes invalid during execution of a program.
Source View	Sets to the Source display mode as the display mode. This item becomes invalid during execution of a program.

Notation		The following cascade menus are displayed to specify the notation.
		This item becomes invalid during execution of a program.
	Hexadecimal number	Displays values on this panel in hexadecimal number (default).
	Decimal	Displays values on this panel in decimal number.
	Octal	Displays values on this panel in octal number.
	Binary	Displays values on this panel in binary number.
Window Connecting		The following cascade menus are displayed to link with other panels (see "(c) Linking with other panels").
	Connect Source Window	Links with the Editor panel.
	Connect Disassemble Window	Links with the Disassemble panel.
Jump to Disassemble		Opens the Disassemble panel (Disassemble1) and jumps to the fetch address corresponding to the selected line in this panel.
Jump to Source		Opens the Editor panel and jumps to the source line corresponding to the selected line in this panel.

Events panel

This panel is used to display the detailed information about the events that are set on the Editor panel/Disassemble panel/Watch panel. On this panel, you can change the setting state of the event between valid/invalid and delete the event (see "2.15   Manage Events").

This panel appears only when connected to the debug tool.

Remarks 1.   See "2.15.6   Notes for setting events" for more information about the setting of events.

         2.   This panel also manage the events that are set on the Function panel or Variable panel of the analyze tool (Program Analyzer).

         3.   When the separator line of each area in this panel is double-clicked, the width of the area changes to the shortest possible size that can display the contents of the area.

Figure A-49.   Events Panel
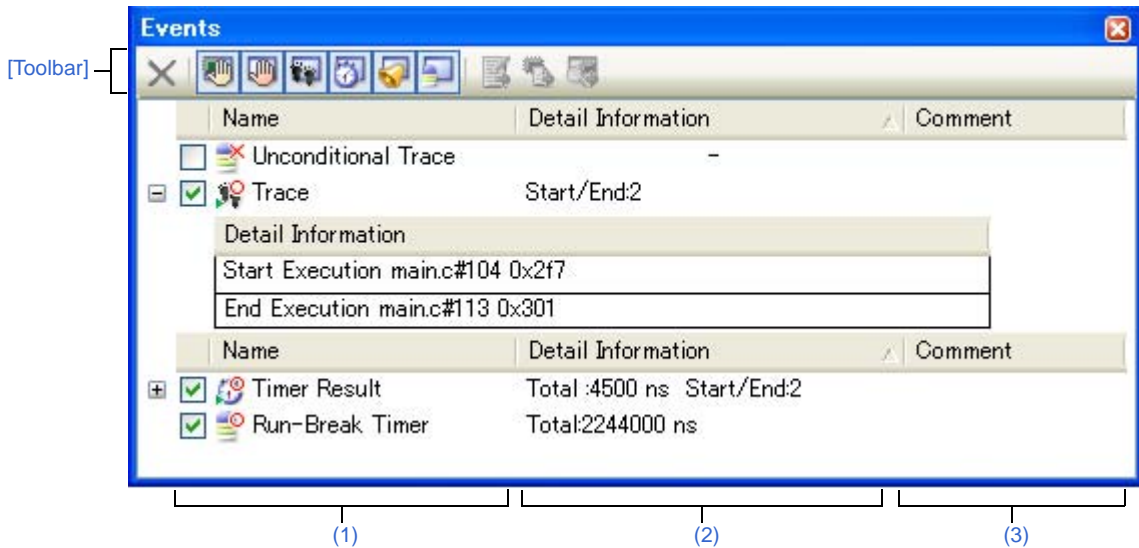
This section describes the following.

- [How to open]
- [Description of each area]
- [Toolbar]
- [[Edit] menu (Events panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [Event].
- **[IECUBE][IECUBE2][Simulator]**

     On the Editor panel/Disassemble panel, select [Timer Settings] >> [View Result of Timer] from the context menu.

[Description of each area]

(1) [Name] area

A list of the event names that have currently been set is displayed in the following format.
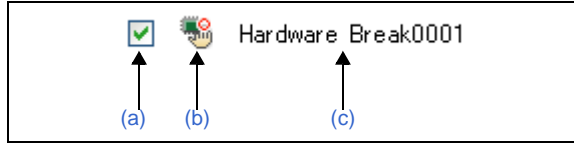
Remark It is possible to limit the event to be displayed by clicking the button on the toolbar (see "[Toolbar]").

(a) Check box

The setting state of the event is displayed/changed.

Note that the Event mark is changed depending on the setting state of the event.

☑	Valid state	Event occurs when the specified condition is met.
		It is possible to set the event to an invalid state by removing the check.
☐	Invalid state	Event does not occur when the specified condition is met.
		It is possible to set the event to a valid state by removing the check.
▣	Suspended state	The conditions that have been specified cannot be set with the program of the debugging target.
		It is not possible to operate the check box.

Remarks 1. Both of the Timer Start event and Timer Stop event is must be set for the Timer Result event. Therefore, it is not possible to set a particular event to a valid state by only the setting of one of these (at the same time as both events are set, they are treated as grouped events as a Timer Result).

2. It is not possible to set the Run-Break Timer event to an invalid/suspended state.

3. The setting of the Unconditional Trace event and the Trace event to valid or invalid state is exclusively controlled. Therefore, the Unconditional Trace event, which is a built-in event, is valid state by default, but if either a Start Tracing event/Stop Tracing event is set, it automatically becomes invalid state, and the Trace event, which is a event name that is collectively called with a Start Tracing event and a Stop Tracing event, becomes valid state. Conversely, if the set Trace event is invalid state, the Unconditional Trace event automatically becomes valid state.

(b) Event mark

The event mark shows the type of event, and in addition shows the current setting state.

The meanings of the marks displayed are as follows:

Table A-13.　Event Mark

Event Type	Valid State	Invalid State	Suspended State	Note
Hardware Break				-
Software Break **[IECUBE][IECUBE2] [MINICUBE][MINICUBE2] [E1]**				-
Unconditional Trace **[IECUBE][IECUBE2] [Simulator]**			None	-
Run-Break Timer		None	None	-
Trace **[IECUBE][IECUBE2] [Simulator]**				Used only in the Events panel
Start Tracing				Used only in the Editor panel/ Disassemble panel
Stop Tracing				
Timer Result **[IECUBE][IECUBE2] [Simulator]**				Used only in the Events panel
Start Timer				Used only in the Editor panel/ Disassemble panel
Stop Timer				
Point Trace **[IECUBE][IECUBE2] [Simulator]**				-
Printf (Action event)				-
Setting of two or more events	Note 1	Note 2	Note 3	Used only in the Editor panel/ Disassemble panel

Notes 1.　There is one or more event with valid state.

　　　　2.　There is no event with valid state and at least one event with invalid state.

　　　　3.　All the set events are suspended state

(c) Event name

The event type and ID number are displayed as the event name.

A number from 0001 is automatically provided as the ID number for each event (no renumbering of the ID number is done even in the event that an event that has been set once is deleted).

Event types that are displayed are as follows:

Table A-14.   Event Type

Event Type	Description
Hardware Break (Break[Note 1])	Breaks the program when the condition is met while the debug tool monitors the break condition all the time during program execution. -> See "2.8.2 Stop the program at the arbitrary position (breakpoint)" -> See "2.8.3 Stop the program with the access to variables/I/O registers"
Software Break (Break[Note 1]) **[IECUBE][IECUBE2] [MINICUBE][MINICUBE2] [E1]**	Breaks the program when the instruction, which an address code to break is rewritten for the break instruction, is executed. -> See "2.8.2 Stop the program at the arbitrary position (breakpoint)"
Unconditional Trace	Automatically collects the trace data with start of a program execution, and stops collecting the trace data with stop of the program execution. This event cannot be deleted because of the built-in event[Note 2] (this event is set to a Valid state by default). -> See "2.11.2 Collect execution history until stop of the execution"
Run-Break Timer	Automatically measures the execution time of a program with start of the program execution, and stops the measurement with stop of the program execution. This event cannot be deleted because of the built-in event[Note 2] (this event is set to a Valid state by default). -> See "2.12.1 Measure execution time until stop of the execution"
Trace **[IECUBE][IECUBE2] [Simulator]**	Starts/stops collecting the trace data when the condition specified with "Start Tracing" and "Stop Tracing" is met (this event is displayed when a Start Tracing event or a Stop Tracing event is set). -> See "2.11.3 Collect execution history in the arbitrary section"
Timer Result **[IECUBE][IECUBE2] [Simulator]**	Starts/stops measuring the execution time of a program when the condition specified with "Start Timer" and "Stop Timer" is met (this event is displayed when a Start Timer event or a Stop Timer event is set). -> See "2.12.2 Measure execution time in the arbitrary section [IECUBE][IECUBE2][Simulator]"
Point Trace **[IECUBE][IECUBE2] [Simulator]**	Records the information as the trace data when accessing the specified variable or I/O register during execution of a program. -> See "2.11.4 Collect execution history when the condition is met"
Printf	Executes printf command in software processing after temporary stopping a program in execution at an arbitrary position (action event). -> See "2.14.1 Inset printf"

Notes 1. A breakpoint that is set by a one click operation of the mouse is displayed "Break" (see "(2)   Set a breakpoint").

2. This is set in the debug tool by default.

Remark The following event types are also displayed when an event (breakpoint or break event) is set on the Function panel or the Variable panel of the analyze tool (Program Analyzer)

- In the case of a breakpoint at a function: "Break at start of function"
- In the case of a break event to a variable: "Access break to variable"

(2) [Detail Information] area

Detailed information about each event is displayed.

The contents of the information that is displayed differ depending on the event type as follows:

Table A-15.   Detailed Information with Event Type

Event Type	Display Contents[Note 1]		
Hardware Break (Condition: execution)	Format1	*<Condition to occur> <File name#Line number> <Address>*	
	Example	Before Execution main.c#39 0x100	
		After Execution sub.c#100 0x200	
		Before Execution - 0x300	
		Execution main.c#39 0x300 **[Simulator]**	
	Format2	*<Condition to occur> <Symbol + Offset> <Address>*	
	Example	Before Execution funcA + 0x10 0x100	
		After Execution funcB + 0x20 0x200	
		Before Execution - 0x300	
Hardware Break (Condition: access)	Format1	*<Condition to occur> <File name#Variable name> <Address(range)> <Comparison condition> <Comparison value>*	
	Example	Read main.c#variable1 0x100 - 0x101 == 0x5	
		Write sub.c#variable2 0x200 - 0x200 == 0x7	
		Read/Write sub2.c#variable3 0x300 - 0x303 == 0x8	
	Format2	*<Condition to occur> <File name#Function name#Variable name> <Address(range)> <Comparison condition> <Comparison value>*	
	Example	Read main.c#func1#variable1 0x100 - 0x101 == 0x10	
	Format3	*<Condition to occur> <Variable name> <Address(range)> <Comparison condition> <Comparison value>*	
	Example	Write variable1 0x100 - 0x101 == 0x10	
Software Break **[IECUBE][IECUBE2] [MINICUBE][MINICUBE2] [E1]**	Format1	*<Condition to occur> <File name#Line number> <Address>*	
	Example	Before Execution main.c#40 0x102	
		Before Execution sub.c#101 0x204	
	Format2	*<Condition to occur> <Symbol + Offset> <Address>*	
	Example	Before Execution funcA + 0x12 0x102	
Unconditional Trace **[IECUBE][IECUBE2] [Simulator]**	Format	-	
	Example	-	
Run-Break Timer	Format	Total: *<Total execution time>*	
	Example	Total: 1000ms	
		Total: OVERFLOW	

Event Type	Display Contents[Note 1]	
Trace **[IECUBE]** **[IECUBE2]** **[Simulator]** (Condition: execution)	Format	Total of Start/End: *<Total number of Start tracing/Stop tracing events>* [Note 2] <Start/End> *<Detailed information of Start tracing/Stop tracing>*
	Example	Total of Start/End: 4 - Start After Execution main.c#1000x300 - Start After Execution funcA + 0x1000x300 - End After Execution main.c#2000x100 - End After Execution funcA + 0x100x100
Timer Result **[IECUBE]** **[IECUBE2]** **[Simulator]** (Condition: execution)	Format	Total:*<Total execution time >* Total of Start/End: *<Total number of Start timer/Stop timer>*[Note 2] - *<Total execution time> <Pass Count> <Average> <Max> <Min>* - <Start/End> *<Detailed information of Start timer/Stop timer>*
	Example	Total: 10ms Total of Start/End: 4 - Total: 10ms Pass Count: 5 Average: 2ms Max: 4ms Min: 1ms - Start After Execution main.c#1000x300 - Start After Execution funcA + 0x300x100 - End After Execution main.c#1000x300 - End After Execution funcA + 0x500x100
Point Trace **[IECUBE]** **[IECUBE2]** **[Simulator]** (Condition: access)	Format1	*<Condition to occur> <Variable name> <Address(range)>*
	Example	Read variable1 0x100 - 0x101
	Format2	*<Condition to occur> <File name# Variable name> <Address(range)>*
	Example	Write sub.c#variable2 0x200 - 0x200
	Format3	*<Condition to occur> <File name#*Function name#*Variable name> <Address(range)>*
	Example	Read/Write sub.c#func1#variabl3 0x300 - 0x303
Printf (Action event)	Format	*<Condition to occur> <File name#Line number> <Address> <Setting of Printf event>*
	Example	Before Execution main.c#39 0x100 aaa, bbb, ccc
		After Execution sub.c#100 0x200 Result of aaa : aaa

Notes 1. Following are the details on the display format.

<Condition to occur>	Display one of the following conditions. **[IECUBE][IECUBE2][MINICUBE][MINICUBE2][E1]**
	Execution: Before Execution or After Execution
	Access: Read, Write, Read/Write
	[Simulator]
	Execution: Execution
	Access: Read, Write, Read/Write

<File name#Line number>	Shows the line number of the source. Display format is the same as the watch type scope specification expression. When multiple load module files are downloaded, *<Load module file name*$*File name#Line number>* is displayed.
	
	For those events set in the Disassemble panel, display *<Line number>* in the format *<Symbol + offset>* in the condition below.
	  - Line information exists and the specified position that the event is set not the top of the line information
	  - Line information does not exist and symbol information exists.
	Show *<Line number>* in "-" in the following condition.
	  - Line information and symbol information does not exist.

<Variable name>	Shows the variable name in the source file. Display format is the same as the watch type scope specification expression.
<Comparison condition>	Condition to compare (==) is shown. If the comparison value is not specified, comparison condition is not shown.
<Comparison value>	Comparison value is shown. If the comparison value is not specified, comparison condition is not shown.
<Address>	Address in the memory area is shown (only in hex number).
<Start/End>	Shows whether the contents of the detailed information is start event or the stop event.
<Pass Count>	Shows the measurement result of the pass count of the timer. If a timer overflow occurs (see "2.12.3 Measurable time ranges"), or if the illegal value was acquired, "OVERFLOW" is displayed.
<Total>	Shows the measurement result of the timer total execution time. The unit is either of ns/μs,/ms/s/min (if, however, the unit is in "min", a value in "s" unit also appears). If a timer overflow occurs (see "2.12.3 Measurable time ranges"), or if the illegal value was acquired, "OVERFLOW" is displayed.
<Average>	Shows the measurement result of average execution of the timer. The unit is either of ns/μs,/ms/s/min (if, however, the unit is in "min", a value in "s" unit also appears).
<Max>	Shows the measurement result of the maximum execution time of the timer. The unit is either of ns/μs,/ms/s/min (if, however, the unit is in "min", a value in "s" unit also appears).
<Min>	Shows the measurement result of the minimum execution time of the timer. The unit is either of ns/μs,/ms/s/min (if, however, the unit is in "min", a value in "s" unit also appears).
<Set print event>	Shows the variable expression and the character strings specified in the Action Events dialog box.

2. Click this line to display the detailed information of the lower lines.

(3) [Comment] area

The user can write comments for each event that has been set.

To input comments, click on this area, or select [Edit Comment] form the context menu after selecting the event in which you want to input comments, and then input directly the desired text from the keyboard (the edit mode is cancelled by pressing down the [Esc] key).

After editing the comments, complete the editing by pressing the [Enter] key or moving the focus to outside the edit region. Up to a maximum of 256 characters can be inputted for the comments, and this is saved as the settings of the user during use.

[Toolbar]

✕	Deletes the selected event. Note that it is not possible to delete the built-in events (Unconditional Trace event and Run-Break Timer event).
🔲	Displays events related to the Hardware Break (default).
🔲 **[IECUBE][IECUBE2]** **[MINICUBE][MINICUBE2]** **[E1]**	Displays events related to the Software Break (default).
🔲 **[IECUBE][IECUBE2]** **[Simulator]**	Displays events related to the trace (default).
🔲 **[IECUBE][IECUBE2]** **[Simulator]**	Displays events related to the timer (default).
🔲	Displays events related to the action event (Printf event) (default).
🔲	Displays events related to the built-in event (Unconditional Trace event/Run-Break Timer event) (default).
🔲	Opens the Editor panel and jumps to the source line corresponding to the address where the selected event^{Note} is being set.
🔲	Opens the Disassemble panel and jumps to the disassemble results corresponding to the address where the selected event^{Note} is being set.
🔲	Opens the Memory panel and jumps to the memory corresponding to the address where the selected event^{Note} is being set.

Note   Events other than Trace events, Timer Result events and built-in events (Unconditional Trace events/Run-Break Timer events) can be objects of this button.

[[Edit] menu (Events panel-dedicated items)]

The following items are exclusive for [Edit] menu in the Events panel (other items are all invalid).

Delete	Deletes the selected event. Note that it is not possible to delete the built-in events (Unconditional Trace event and Run-Break Timer event).
Select All	Selects all the events displayed on the panel.
Find...	Opens the Search and Replace dialog box with selecting [Whole Search] tab.
Replace...	Opens the Search and Replace dialog box with selecting [Whole Replace] tab.

[Context menu]

|	
Enable Event	Enables the selected event (valid state).
Disable Event	Disables the selected event (invalid state).
Delete	Deletes the selected event. Note that it is not possible to delete the built-in events (Unconditional Trace event and Run-Break Timer event).
Select All	Selects all the events of this panel.
View Select	The following cascade menus are displayed to limit the event type to be displayed. All of the items have been selected by default.
Hardware Break	Displays events related to the Hardware Break.
Software Break	Displays events related to the Software Break.
Timer Event	Displays events related to the timer.
Trace Event	Displays events related to the trace.
Action Event	Displays events related to the action event (Printf event).
Built-in Event	Displays events related to the built-in event (Unconditional Trace event/Run-Break Timer event).
Timer Settings	The following cascade menus are displayed to do the settings related to the timer. However, this item is valid only when a timer-related event has been selected.
Init Timer	Initializes the timer used by the selected event (except for the Run-Break Timer).
Nanosecond	Displays the Timer Results of the selected event in nanosecond units.
Microsecond	Displays the Timer Results of the selected event in microsecond units.
Millisecond	Displays the Timer Results of the selected event in millisecond units.
Second	Displays the Timer Results of the selected event in second units.
Minute	Displays the Timer Results of the selected event in minute units.
Jump to Memory	Opens the Memory panel (Memory1) and jumps to the memory corresponding to the address where the selected event[Note] is being set.
Jump to Disassemble	Opens the Disassemble panel (Disassemble1) and jumps to the disassemble results corresponding to the address where the selected event[Note] is being set.
Jump to Source	Opens the Editor panel and jumps to the source line corresponding to the address where the selected event[Note] is being set.
Edit Comment	Sets to the edit mode to input comments for the selected event. When comments are already present, all of that character string is set to a select state.

Note Events other than Trace events, Timer Result events and built-in events (Unconditional Trace events/Run-Break Timer events) can be objects of this button.

> **Output panel**

This panel is used to display operation logs for various components (debug tool, design tool, build tool, etc.) provided by CubeSuite, in addition to results of batch searches by the Search and Replace dialog box and a Printf event (see "2.14.1 Inset printf").

The messages are classified by the message origination tool and displayed on the individual tabs.

Figure A-50. Output Panel
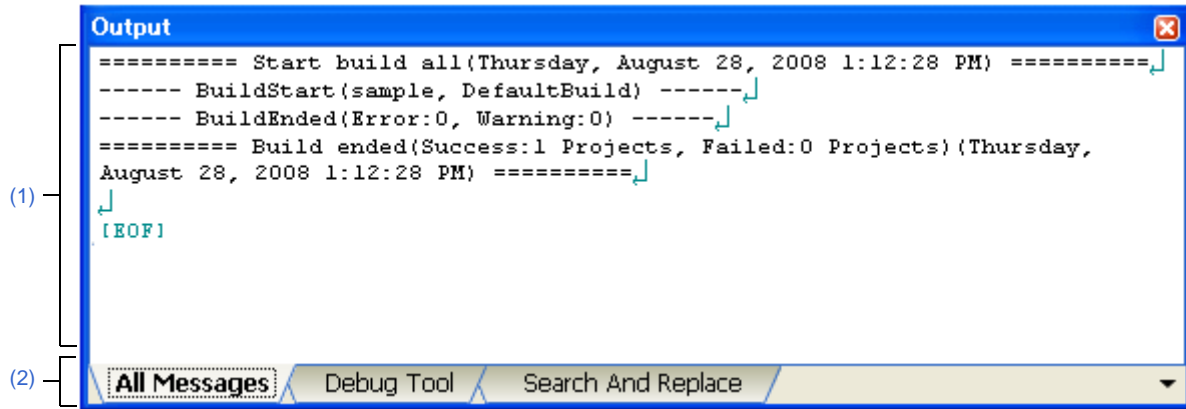
This section describes the following.

- [How to open]
- [Description of each area]
- [[File] menu (Output panel-dedicated items)]
- [[Edit] menu (Output panel-dedicated items)]
- [Context menu]

[How to open]

- From the [View] menu, select [Output].

[Description of each area]

(1) Message area

The output messages of each tool, search results and results by a Printf event are displayed.

In the case of search results (batch search), every time a search is performed, a new message will be displayed after the previous message is cleared (except for the [All Messages] tab).

The colors of message display differ with the type of message as shown below (character colors and background colors depend on the configuration in the [General - Font and Color] category of the Option dialog box).

Message Type	Display Example (Default)			Description
Normal message	AaBbCc	Character color	Black	Displayed with information notices
		Background color	White	
Warning message	AaBbCc	Character color	Blue	Displayed with warnings about operations
		Background color	Standard color	
Error message	AaBbCc	Character color	Red	Displayed when there is a critical error, or when execution is not possible due to a operational mistake
		Background color	Light gray	

This area is provided with the following functions.

(a) Tag jump

By double-clicking on the output message, the Editor panel is opened and the number of the corresponding line in the corresponding file is displayed.

This allows you to jump from error messages that are output when building, etc. to the corresponding error line in the source file.

(b) Help display

If there is a caret on the line where a warning message or error message is being displayed, you can select [Help for Message] from the context menu. You can also display help for that line's message by pressing the [F1] key.

(c) Saving a log

The Save As dialog box can be opened by selecting the [File] menu >> [Save Output-*tab name* As...], and the contents that are displayed on the currently selected tab can be saved in a text file (*.txt) (messages on deselected tabs will not be saved).

(2) Tab selection area

Select the tab that indicates the origin of message.

The following tabs are available for the debug tool.

Tab Name	Description
All Messages	Displays operation logs for all components (debug tool, design tool, build tool, etc.) provided by CubeSuite in order of output.
Debug Tool	Displays messages output from the debug tool. Display only operation logs for the debug tool out of those for various components (debug tool, design tool, build tool, etc.) provided by CubeSuite.
Search And Replace	Displays the batch search results from the Search and Replace dialog box.

Caution **Even if a new message is output on a deselected tab, tab selection will not automatically switch. In this case, "*" mark will be added in front of the tab name, indicating that a new message has been output.**

[[File] menu (Output panel-dedicated items)]

The following items are exclusive for the [File] menu in the Output panel (other items are common to all the panels). Note that all these items are invalid during execution of a program.

Save Output-*tab name*	Overwrites the contents that are displayed on the currently selected tab to the preciously saved text file (see "(c) Saving a log"). Note that when the file has never been saved or the file is write disabled, the same operation is applied as the selection in [Save Output-*tab name* As...]. This item becomes invalid while building.
Save Output-*file name* As...	Opens the Save As dialog box to newly save the contents that are displayed on the currently selected tab to the specified text file (*.txt) (see "(c) Saving a log").

[[Edit] menu (Output panel-dedicated items)]

The following items are exclusive for [Edit] menu in the Output panel (other items are all invalid).

Copy	Copies the contents of the selected range to the clipboard as character string(s).
Select All	Selects all the messages displayed on the currently selected tab.
Find...	Opens the Search and Replace dialog box with selecting [Quick Search] tab.
Replace...	Opens the Search and Replace dialog box with selecting [Whole Replace] tab.

[Context menu]

Copy	Copies the contents of the selected range to the clipboard as character string(s).
Select All	Selects all the messages displayed on the currently selected tab.
Clear	Deletes all the messages displayed on the currently selected tab.
Tag Jump	Opens the Editor panel and jumps to the number of the corresponding line in the corresponding file of the message at the caret position.
Stop Searching	Cancels the search currently being executed. This is invalid when a search is not being executed.
Help for Message	Displays help for the massage on the current caret position. This only applies to warning messages and error messages.

Memory Mapping dialog box

This dialog box is used to set the memory mapping for each type of memory.

Caution   If you are not connected to a debug tool, then only memory mapping areas added by user is displayed.
Connecting to a debug tool (see "2.4.1   Connect the debug tool to CubeSuite") will display details for each memory type.

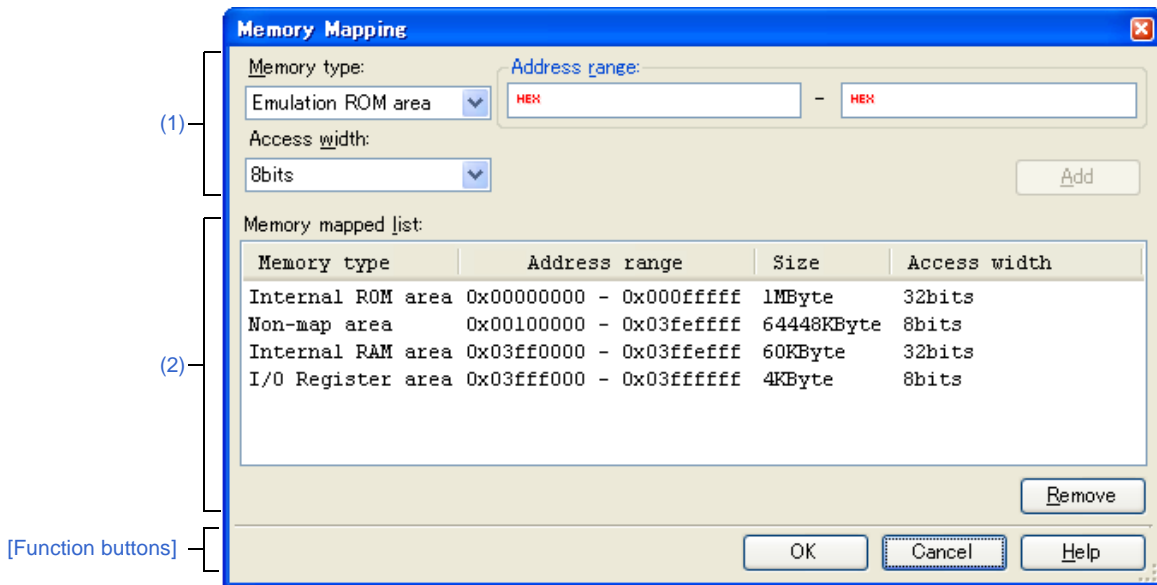**Figure A-51.   Memory Mapping Dialog Box [IECUBE[V850ES]][IECUBE2][MINICUBE][MINICUBE2][E1]**

Figure A-52.   Memory Mapping Dialog Box [IECUBE[V850E1]]
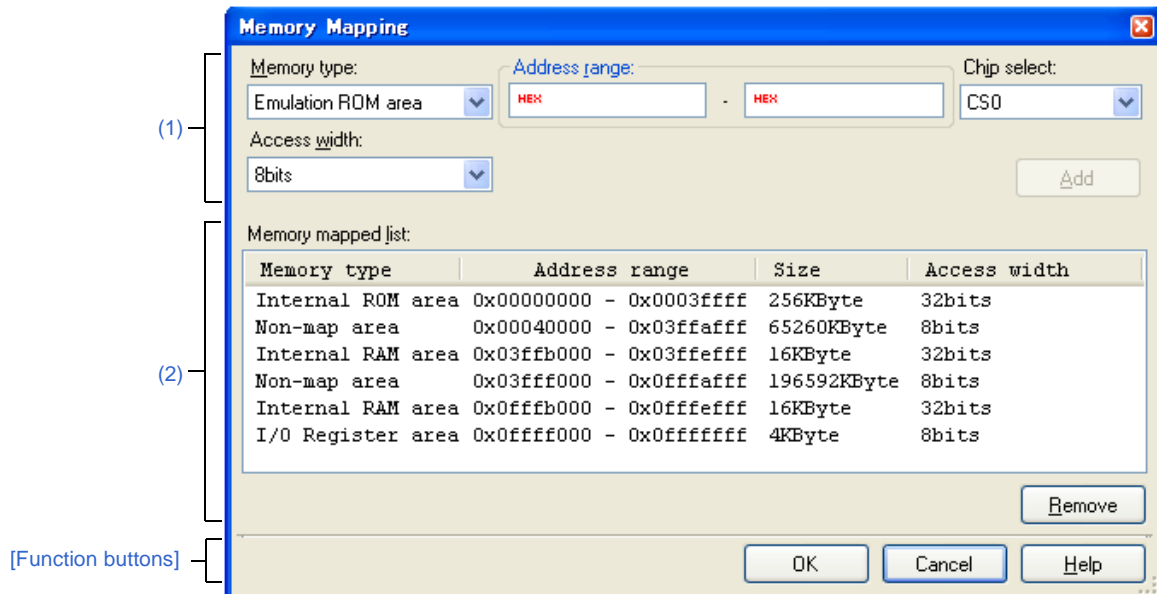
Figure A-53.   Memory Mapping Dialog Box [Simulator]
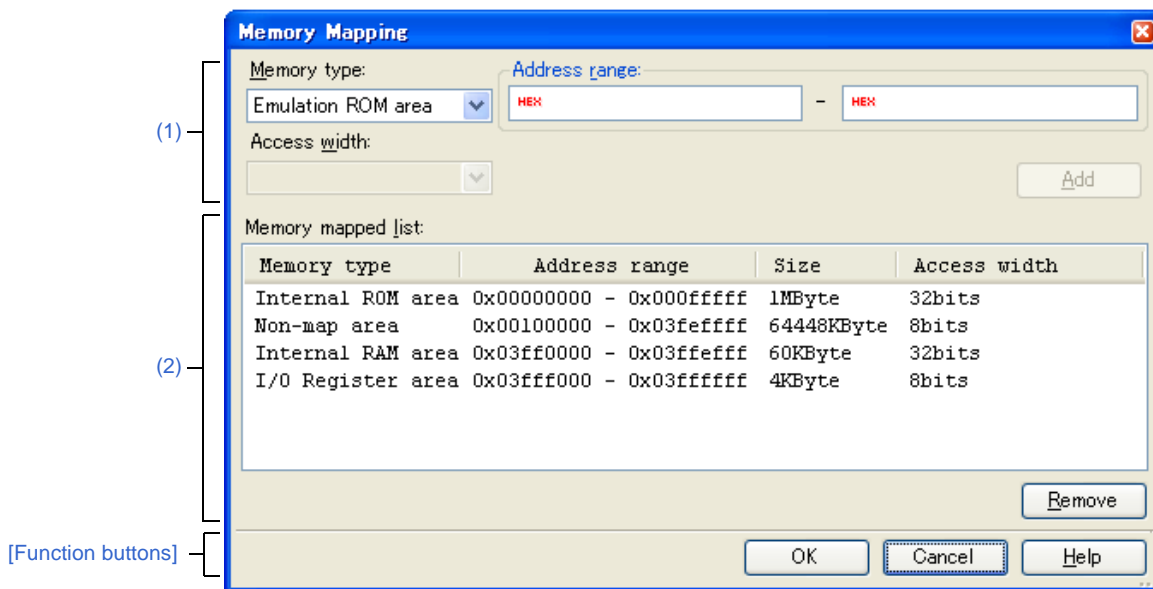
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the [Debug Tool Settings] tab of the Property panel, click the [...] button displayed by selecting one of the values of the [Memory mappings] property in the [Memory] category.

Caution   This dialog box cannot be opened during execution of a program.

[Description of each area]

(1) Added memory mapping specification area
Specify the information for a newly added memory mapping.

(a) [Memory type]
Select the memory type for the memory mapping to be added from the following drop-down list.
The item selected by default differs depending on the debug tool to use.

Emulation ROM area **[IECUBE][Simulator]**	Adds the Emulation ROM area. [IECUBE]: The IECUBE alternate ROM is used. [Simulator]: The simulator alternate ROM is used.
Emulation RAM area **[IECUBE][Simulator]**	Adds the Emulation RAM area. [IECUBE]: The IECUBE alternate RAM is used. [Simulator]: The simulator alternate RAM is used.
Target memory area	Adds the target memory area.
Target ROM area **[IECUBE]**	Adds the target ROM area. Areas specified as target ROM are subject to write protect for the fail-safe break.
I/O protection area	Adds the I/O protection area[Note].

Note   The I/O protected area is an area protected from access by the debug function.
The address range set to I/O protected area cannot be accessed from the Memory panel (i.e., it can only be accessed via executing the load module). I/O protected area mapping is only possible within the target memory area.

Remark   External memory area and guarded areas (areas where access is prohibited) are treated as non-mapped areas. For this reason, if a mapping overlaps a guarded area, then a message will inform the user that the mapping overlaps a non-mapped area.
See the manual of our microcontroller for details onfor details onfor details on the mapping information of external memory areas and guarded areas.

Mapping attributes and their sizes that can be set are as follows:

Table A-16.   Settable Mapping Attribute

Attribute	Debug Tool				
	IECUBE	IECUBE2	MINICUBE E1 (JTAG)	[MINICUBE2 E1 (Serial)	Simulator
Emulation ROM area	✓ Note	—	—	—	✓
Emulation RAM area	✓ Note	—	—	—	✓
Target memory area	✓	✓	✓	✓	✓
Target ROM area	✓	—	—	—	—
I/O protection area	✓	✓	✓	✓	✓

✓ : Valid (Mapping unit: 1 Mbyte for Emulator, 64Kbytes for Simulator))
— : Invalid

Note   Valid only with the option board.
The emulation ROM/RAM area are composed of 16 Mbytes.

(b)   [Address range]
Specify the start address and end address for the memory mapping to be added.
Directly input a hexadecimal number into the text box for each.
In the case of the following settings, however, new memory mappings cannot be added. Clicking the [Add] button in this area causes an error message to be displayed.

- If the specified address range duplicates a separate memory area when [Target memory area] is selected as the memory type
- If the specified address range cannot be put into a single target memory area when [I/O protection area] is selected as the memory type

(c)   [Chip select] [IECUBE[V850E1]]
This item is only valid when using V850E1 Core and when [Memory type] is set to [Emulation ROM] or [Emulation RAM].
Select from the drop-down list the chip select corresponding to the address specified with [Address range].

(d) [Access width] [IECUBE][MINICUBE][MINICUBE2][E1]

Select the access width of the memory mapping to be added from the following drop-down list (direct input is not possible).

In the case where [I/O protection area] is selected as the memory type, the access width must be set to the same value as the access width of the target memory area.

8bits	Sets the access width of the memory mapping to be added to 8 bits (default).
16bits	Sets the access width of the memory mapping to be added to 16 bits[Note].
32bits	Sets the access width of the memory mapping to be added to 32 bits[Note].

Note When being added to memory mapping, fractions in the address value specified as the ending address are automatically rounded up to multiples of 2 if [16bits] is selected, or multiples of 4 if [32bits] is specified.

(e) Button

Button	Function
Add	Adds the content specified in this area to memory mapping. The added memory mapping is displayed in the [Memory mapped list] area. The changes will not take effect until the [OK] button is clicked.

(2) [Memory mapped list] area

(a) List display

Information about the memory mapping added in the Added memory mapping specification area and the microcontroller's internal memory mapping is displayed. This area cannot be edited.

Memory type	Displays the following memory types. **[V850E1][V850ES]** - Internal ROM area[Note 1] - Internal RAM area - DataFlash area[Note 2] **[IECUBE][MINICUBE][MINICUBE2][E1]** - I/O register area - Target memory area - Target ROM area **[IECUBE]** - Emulation ROM area **[IECUBE][Simulator]** - Emulation RAM area **[IECUBE][Simulator]** - Non-map area - I/O protection area **[V850E2M]** - Internal ROM area[Note 1] - Internal RAM area - DataFlash area[Note 2] **[IECUBE2][MINICUBE][E1 (JTAG)]** - PBUS peripheral I/O area - CPU peripheral I/O area - Internal peripheral I/O area - HBUS peripheral I/O area - HBUS shared memory area - Back-up RAM area - Target memory area - Emulation ROM area **[Simulator]** - Emulation RAM area **[Simulator]** - Non-map area - I/O protection area
Address range	Displays the address range as *\<Start address\> - \<End address\>*. Display is fixed as "0x"-prefixed hexadecimal numbers.
Size	Displays size as a decimal number (unit: bytes/Kbytes[Note 3]).
Access width	Displays the access width (unit: bits).

Notes 1. This item does not appear when the selected microcontroller is a ROMless product.
In the case that internal emulation ROM area exists in the emulator, however, "Internal ROM area" will be displayed only when the [Size of internal ROM [KBytes]] property of the [Internal ROM/RAM] category is set to a value greater than "0" in the [Connect Settings] tab of the Property panel.

2. This item appears only when the selected microcontroller incorporates the data flash memory.

3. Only in the case of multiple of 1024, displays in kilobyte units.

(b) Button

Button	Function
Remove	Deletes the memory mapping selected in this area. The memory areas that can be deleted are the Target memory area, I/O protection area, Emulation ROM area **[IECUBE][Simulator]**, or the Emulation RAM area **[IECUBE][Simulator]** (the microcontroller's internal memory mapping cannot be deleted). In the case where you have attempted to delete a target memory area that is specified as an I/O protection area, however, a message will be displayed. The selected target memory area and the I/O protection area mapped to that area will both be deleted only if you click the [OK] button.

[Function buttons]

Button	Function
OK	Sets the currently specified memory mapping to the debug tool and closes this dialog box.
Cancel	Cancels memory mapping changes and closes this dialog box.
Help	Displays the help for this dialog box.

Download Files dialog box

This dialog box is used to select files for downloading and configure download conditions (see "2.5   Download/Upload Programs").

Note that files specified as build targets in a project (main project or sub-project) are automatically registered as download targets (they can be unregistered).

Figure A-54.   Download Files Dialog Box
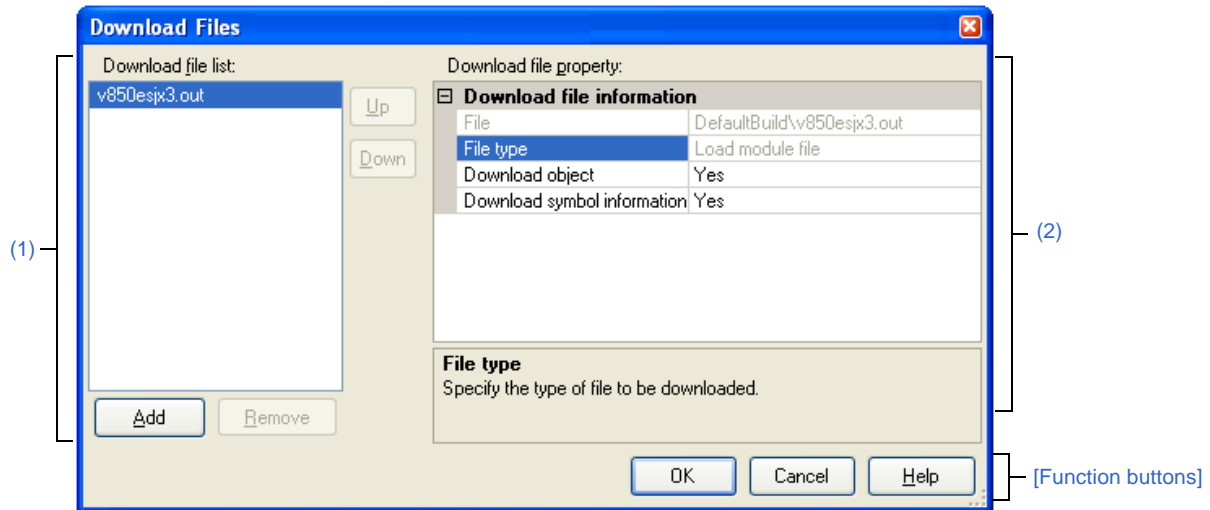
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the [Download File Settings] tab of the Property panel, click the [...] button displayed by selecting the [Download files] property in the [Download] category.

Caution   This dialog box cannot be opened during execution of a program.

[Description of each area]

(1) [Download file list] area

(a) List display
Displays a list of files to download. The names of files specified as build targets in a project (main project or sub-project) are displayed by default (they can be removed).
Files are downloaded in the order that they are displayed here.
To add a new file to be downloaded, click the [Add] button in this area, then in the [Download file property] area, specify the download conditions of the file to add.

(b) Button

Button	Function
Up	Moves the selected file up one row in the list. Clicking this for the top file in the list has no effect.
Down	Moves the selected file down one row in the list. Clicking this for the bottom file in the list has no effect.
Add	Adds an empty item "-" to the list, and selects it. Specify the download conditions of the file to add in the [Download file property] area. Note that this button will be disabled if 20 files have already been registered.
Delete	Deletes the selected file from the list. Note, however, that this button is disabled if the selected file is a project build target.

Remarks 1.   By superimposing the mouse cursor on a file name, the pass information of the file is pop-up displayed.

2.   By dragging a file name with the mouse, the display order in the list can be changed. Note, however, that the order of a project build target cannot be changed.

(2) [Download file property] area

(a) [Download file information]

This area is used to display or edit the download conditions of the file selected in the [Download file list] area.

It can also be used to specify the download conditions of new download files added via the [Add] button.

File	Specify the name of the file to download (up to 259 characters).	
	Default	*File name* (but it will be blank for newly added files)
	Modifying	Directly enter from the keyboard, or specify with the Select Download File dialog box opened by clicking the [...] button[Note 1] appears at right by selecting this item.
	Available values	See "Table 2-3. Type of Files That Can be Downloaded"
File type	Specify the type of the file to download.	
	Default	Load module file
	Modifying	Select from the drop-down list.
	Available values	Either of the following - Load module file - Hex file - Binary data file
Offset	This item appears only when the file to download is in hex format. Specify the offset from the address at which the file's download is to start.	
	Default	0
	Modifying	Directly enter from the keyboard.
	Available values	0x0 to 0xFFFFFFFF in hexadecimal number

Start address	This item appears only when the file to download is in binary format. Specify the address at which to start the file's download.		
	Default	0	
	Modifying	Directly enter from the keyboard.	
	Available values	0x0 to 0xFFFFFFFF in hexadecimal number	
Download object	This item appears only when the file to download is in load module format. Specify whether to download the object information from the specified file.		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Downloads object information.
		No	Does not download object information.
Download symbol information	This item appears only when the file to download is in load module format. Specify whether to download the symbol information from the specified file[Note 2].		
	Default	Yes	
	Modifying	Select from the drop-down list.	
	Available values	Yes	Downloads symbol information.
		No	Does not download symbol information.

Notes 1. When a file specified as build target in the project is selected in the [Download file list] area, or when the program is executing, the [...] button does not appear.

2. If the symbol information have not been downloaded, the source level debugging cannot be performed.

[Function buttons]

Button	Function
OK	Finishes the download file setup, and closes this dialog box.
Cancel	Cancels any changes to the download files, and closes this dialog box.
Help	Displays the help for this dialog box.

Flash Options Setting dialog box [V850E2M]

This dialog box is used to configure options for the flash memory incorporated in the V850E2M microcontroller.
This dialog box appears only when connected to the debug tool.

Caution   **CPU reset may be generated automatically depending on the selected microcontroller when you click the [Write] button after changing the configuration of this dialog box.**

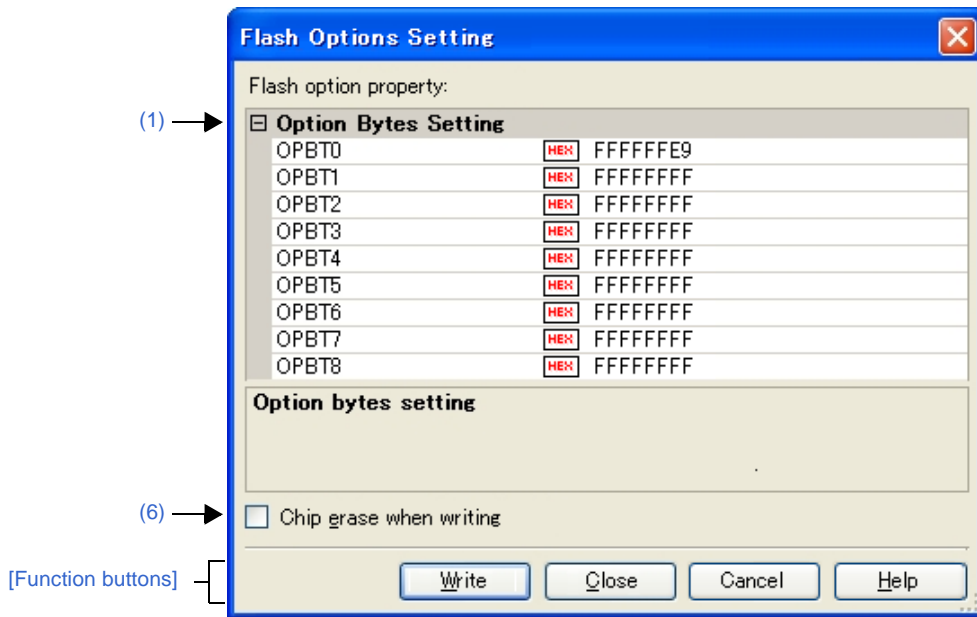**Figure A-55.   Flash Options Setting Dialog Box [IECUBE2][MINICUBE][E1 (JTAG)]**
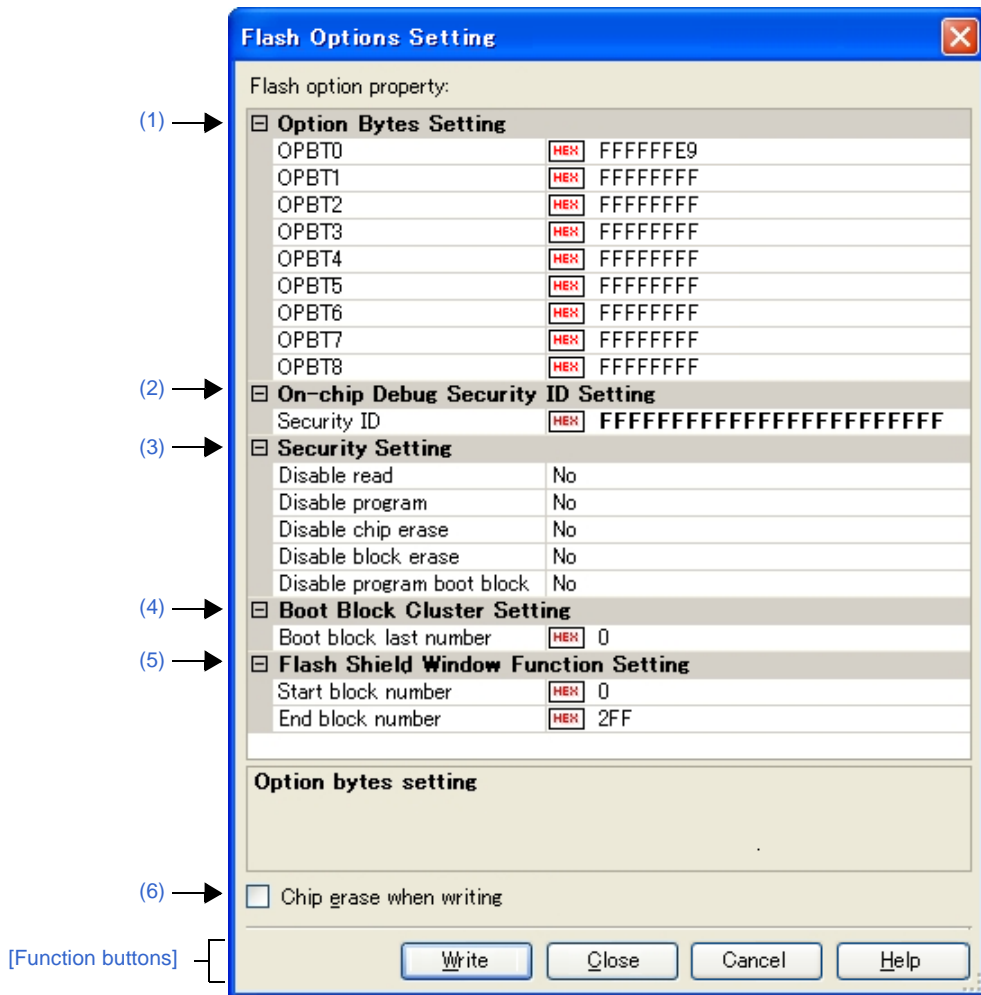
Figure A-56.   Flash Options Setting Dialog Box [Simulator]

This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the [Flash Options Settings] tab [V850E2M] tab of the Property panel, click the [...] button displayed by selecting the [Flash options] property in the [Flash Options] category.

[Description of each area]

(1) [Option Bytes Setting]

(a) [OPBT0] - [OPBT8]
Specify the option bytes.
Directly enter the value in hexadecimal number from 0x0 to 0xFFFFFFFF (the value before shipment of the microcontroller is specified by default).

(2) [On-chip Debug Security ID Setting] [IECUBE2][MINICUBE][E1 (JTAG)]

(a) [Security ID]
Specify the on-chip debug security ID.
Directly enter 24 digits hexadecimal number (12 bytes) ([FFFFFFFFFFFFFFFFFFFFFFFF] is specified by default). For details on the on-chip debug security ID, see User's Manual of the debug tool used.

(3) [Security Setting] [IECUBE2][MINICUBE][E1 (JTAG)]

Configure the security functions via each item below.

All items are set to [No] by default.

(a) [Disable read]

Specify whether to prohibit read command (reading).

Select [Yes] to prohibit read command.

(b) [Disable program]

Specify whether to prohibit program command (writing).

Select [Yes] to prohibit program command.

(c) [Disable chip erase]

Specify whether to prohibit chip erase command.

Select [Yes] to prohibit chip erase command.

(d) [Disable block erase]

Specify whether to prohibit block erase command.

Select [Yes] to prohibit block erase command.

(e) [Disable program boot block]

Specify whether to prohibit rewrite boot area.

Select [Yes] to prohibit to rewrite boot area.

(4) [Boot Block Cluster Setting] [IECUBE2][MINICUBE][E1 (JTAG)]

(a) [Boot block last number]

Specify the last block number of boot areas.

Directly enter the value in hexadecimal number ([0] is specified by default).

(5) [Flash Shield Window Function Setting] [IECUBE2][MINICUBE][E1 (JTAG)]

(a) [Start block number]

Specify the first block number of the flash shield window.

Directly enter the value in hexadecimal number from 0x0 to 0x7FF([0] is specified by default).

(b) [End block number]

Specify the end block number of the flash shield window.

Directly enter the value in hexadecimal number from 0x0 to 0x7FF([*End block number*] is specified by default).

(6) [Chip erase when writing]

☑	Executes the chip erase when clicking the [Write] button.
☐	Does not execute the chip erase when clicking the [Write] button (default).

[Function buttons]

Button	Function
Write	Writes the currently set values to the debug tool and sets them to the Property panel and the project. Then closes this dialog box. Note at this time that CPU reset may be generated automatically depending on the selected microcontroller.
Close	Sets the currently set values to the project and closes this dialog box.
Cancel	Closes this dialog box without setting.
Help	Displays the help for this dialog box.

Text Edit dialog box

This dialog box is used to input/modify character strings.

Figure A-57.   Text Edit Dialog Box
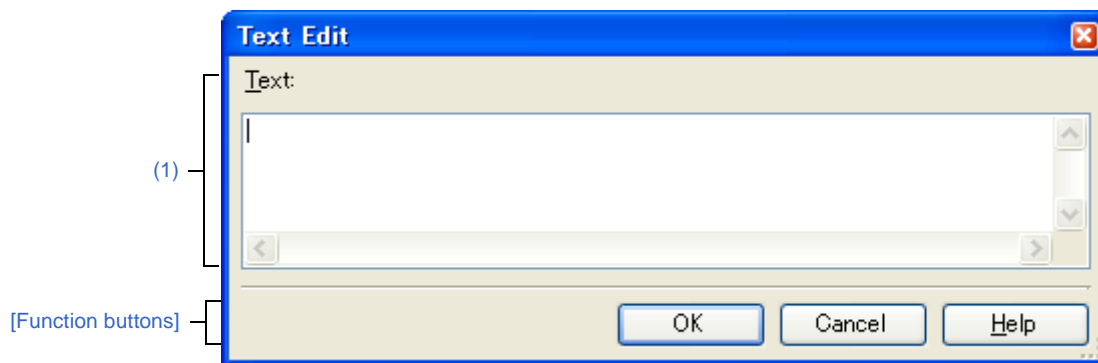
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the [Hook Transaction Settings] tab of the Property panel, click the [...] button displayed by selecting one of the property in the [Hook Transaction Settings] category.

[Description of each area]

(1) [Text] area
Input/modify character strings in this area.

[Function buttons]

Button	Function
OK	Sets the input character strings to the caller panel/dialog box and closes this dialog box.
Cancel	Closes this dialog box.
Help	Displays the help for this dialog box.

Action Events dialog box

This dialog box is used to configure Printf events as action events (see "2.14   Set an Action into Programs").

A Printf event momentarily stops the execution of the program at a specified location, and executes the printf command via software processing. When a Printf event is set, the program momentarily stops immediately before executing the command at the location where this event is set, and the value of the variable expression specified in this dialog box is output to the Output panel.

This dialog box appears only when connected to the debug tool.

> **Cautions 1.** **Also see "2.15.6   Notes for setting events" for details on Printf events (e.g. limits on the number of enabled events).**
>
> **2.** **Once a Printf event has been set, its variable expression cannot be edited.**
> **If you need to edit a variable expression, first delete the Printf event and then create a new one.**

Figure A-58.   Action Events Dialog Box
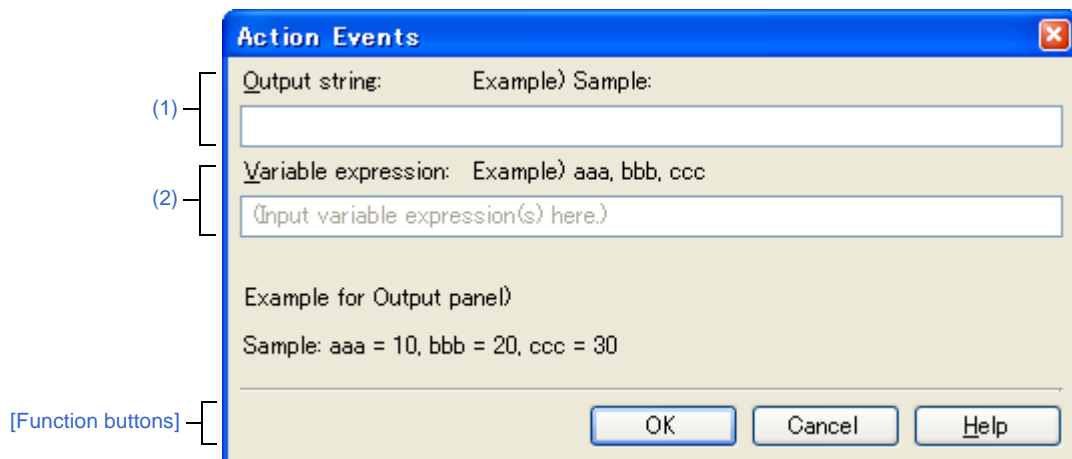
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Editor panel, move the caret to the line where you wish to set a Printf event, then select [Register Action Event...] from the context menu.
- On the Disassemble panel, move the caret to the address where you wish to set a Printf event, then select [Register Action Event...] from the context menu.

[Description of each area]

(1) [Output string] area
Type in the string to add to the Output panel directly via the keyboard (up to 1024 characters).
Note that the output string can only be one line (spaces allowed).

---

(2) [Variable expression] area

Specify the variable expression(s) for the Printf event.

Type a variable expression directly into the text box (up to 1024 characters).

You can specify up to 10 variable expressions for a single Printf event by separating them with commas (",").

If this dialog box is opens with a variable expression selected in the Editor panel /Disassemble panel, the selected variable expression appears as the default.

The basic input format that can be specified as variable expressions and the values output by Printf event are as follows:

Table A-17.   Relationship between Variable Expressions and Output Value (Printf Event)

Variable Expression	Output Value
Variable name of C language	Value of C language variable
Variable expression [*Variable expression*]	Element of array
Variable expression Member name	Member of structure/union
Variable expression -> Member name	Member of structure/union that pointer designates
**Variable expression*	Value of pointer variable
CPU register name	Value of the CPU register
I/O register name	I/O register value
Label, EQU symbol and immediate address	Values of label, EQU symbol and immediate address
Bit symbol	Bit symbol value

Caution   A variable expression including an arithmetic expression (e.g. "+"/"-") cannot be specified.

Remark   The output result format by the Printf event in the Output panel are as follows:

Figure A-59.   Output Result Format of Printf Event

```
Specified characters  Variable expression 1 = Value 1, Variable expression 2 = Value
2, Variable expression 3 = Value 3, ...
```

Specified characters	Characters specified with [Output string]
Variable expression 1 - 10	Characters specified with [Variable expression]
Value 1 - 10	Value of variable corresponds to " *Variable expression 1 - 10* " (Displayed as a decimal number followed by the hexadecimal equivalent in parentheses ()) Note that if the specified variable expression cannot be acquired, "?" is output. If the variable is of type char, then it is displayed as a character.

[Function buttons]

Button	Function
OK	Finishes configuring the Printf event, and sets it at the caret position in the Editor panel/ Disassemble panel.
Cancel	Cancels the Printf event setup and closes this dialog box.
Help	Displays the help for this dialog box.

Save Settings dialog box

This dialog box is used to specify the encoding and the new line code of the file being edited in the Editor panel.

Figure A-60.   Save Settings Dialog Box
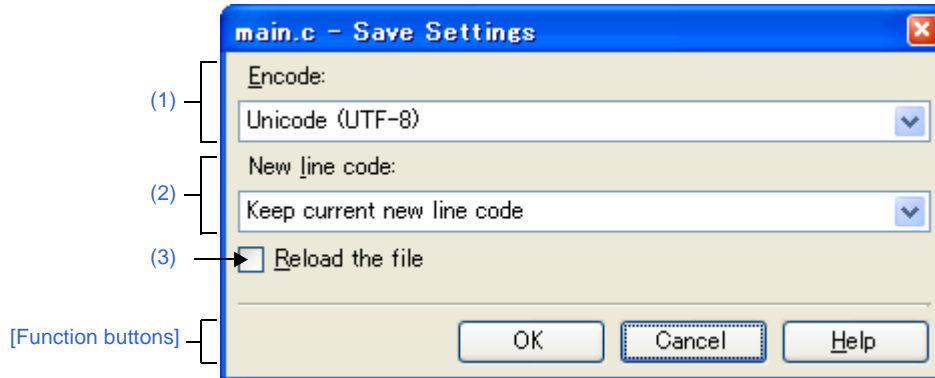
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Editor panel, select [*File name* Save Settings…] from the [File] menu.

[Description of each area]

(1) [Encode] area
Specify the encoding to be set from the drop-down list.
The items in the drop-down list appear in the following order.
Note that encodings that are either overlapped or non-supported by the OS currently being used do not appear.
- *A current encoding of the file* (default)
- *A regulation encoding of the OS currently being used*
- Unicode (UTF-8)
- Japanese (Shift-JIS)
- Japanese (JIS-Allow 1 byte Kana - SO/SI)
- Japanese (EUC)

(2) [New line code] area
Specify the new line code to be set from the drop-down list.
Either of the following can be selected.
- Keep current new line code
- Windows (CR LF)
- Macintosh (CR)
- Unix (LF)

---

(3) [Reload the file]

☑	Reloads the file with the specified encoding and new line code when the [OK] button is clicked.
☐	Does not reload the file when the [OK] button is clicked (default).

[Function buttons]

Button	Function
OK	Sets the specified encoding and new line code to the file and closes this dialog box. When the [Reload the file] check box is selected, reloads the file with the specified encoding and new line code, and closes this dialog box.
Cancel	Cancels the setting and closes this dialog box.
Help	Displays the help for this dialog box.

Memory Initialize dialog box

This dialog box is used to initialize memory (see "(6)   Modify the memory contents in batch (initialize)").

The memory area in the specified address range is repeatedly overwritten with the specified initial data pattern.

Caution   Initialization of memory cannot be performed for values of the ID tag for the data flash memory.

Figure A-61.   Memory Initialize Dialog Box
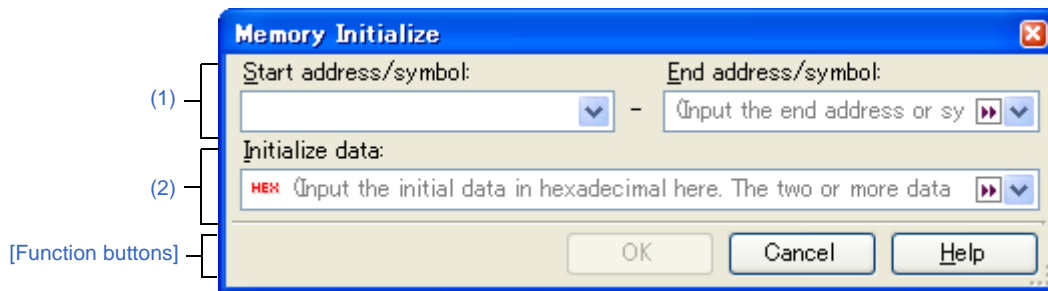
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Memory panel, select [Fill…] from the context menu.

[Description of each area]

(1) Range specification area

Specify the range of memory address to initialize via the [Start address/symbol] and [End address/symbol]. You can either type address expressions directly into the text boxes (up to 1024 characters), or select them from the input history via the drop-down list (up to 10 items).

The results of calculating the address expressions you have entered are treated as start and end addresses, respectively.

Note that address values greater than the microcontroller address space cannot be specified.

(2) [Initialize data] area

Specify the initial value(s) with which to overwrite the memory.

You can either type the initial value into the text box directly in hexadecimal number (the value need not start with "0x"), or select one from the input history via the drop-down list (up to 10 items).

You can specify more than one initial value. Specify up to 16 values of up to 4 bytes (8 characters) each, separated by spaces.

Each initial data value is parsed from the end of the string, with each two characters interpreted as a byte.

If the string has an odd number of characters, then the first character is interpreted as one byte.

Example Endianness specified by debug tool is little endian

Input Character String (Initialization Data)	How Data is Overwritten (in Bytes)
1	01
0 12	00 12
00 012 345	00 12 00 45 03
000 12 000345	00 00 12 45 03 00

[Function buttons]

Button	Function
OK	The memory area in the specified address range is repeatedly overwritten with the specified initial data pattern. If the end address is reached in the middle of the pattern, then writing ends at that point.
Cancel	Cancels the memory initialization and closes this dialog box.
Help	Displays the help for this dialog box.

Memory Search dialog box

This dialog box is used to search memory (see "(5)   Search the memory contents").

Search in either the Memory value area or Character string area where the caret was located in the Memory panel immediately before this dialog box opened.

Caution   Search for memory cannot be performed for values of the ID tag for the data flash memory.

Figure A-62.   Memory Search Dialog Box
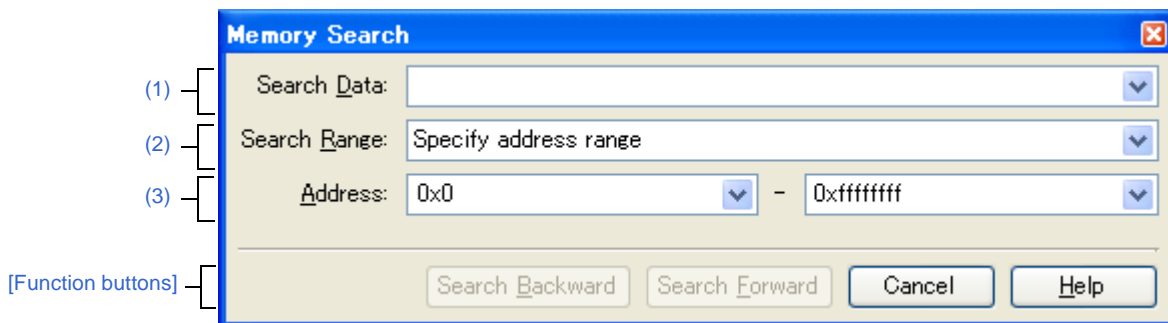
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Memory panel, select [Find…] from the context menu.

[Description of each area]

(1) [Search Data] area

Specify data to search.

You can either type a value directly into the text box (up to 256 bytes), or select one from the input history via the drop-down list (up to 10 items).

If the search is performed in the Memory value area of the Memory panel, the value must be entered in the same display format (notation and size) as that area.

If the search is performed in the Character string area, then the target of the search must be a string. The specified string is converted into the encoding format displayed in that area, and searched for.

If a memory value was selected immediately prior to opening this dialog box, then that value will appear as default.

(2) [Search Range] area

Select the range to search from the following drop-down list.

Specify address range	Searches in the address range specified in the [Address] area.
Memory mapping	Searches within the selected memory mapping range. This list item displays the memory mappings set in the Memory Mapping dialog box (except the non-mapped area). Display format: *<memory type> <address range> <size>*

---

(3) [Address] area

This item is only enabled if [Specify address range] is selected in the [Search Range] area.

Specify the range of memory address to search via the start and end addresses. You can either type address expressions directly into the text boxes (up to 1024 characters), or select them from the input history via the drop-down list (up to 10 items).

The results of calculating the address expressions you have entered are treated as start and end addresses, respectively.

Note, however, that the largest address that can be searched is the maximum address of the program space (0x03FFFFFF) (the mirror area cannot be searched).

In addition, an address value greater than the value expressed within 32 bits cannot be specified.

Remarks 1.   If the start address field is left blank, it is treated as if "0x0" were specified.

2.   If the end address field is left blank, then it is treated as if the maximum value in the microcontroller's address space were specified.

[Function buttons]

Button	Function
Search Backward	Searches upward within the range specified in the [Address] area or the [Search Range] area. The location found by the search is selected in the Memory panel.
	Note that if an illegal value is specified or while the program is being executed, a message will be appear, and the memory search will not be performed.
	If focus moves to this dialog box while the memory panel is hidden or another panel has focus, then this button will be disabled.
Search Forward	Searches downward within the range specified in the [Address] area or the [Search Range] area. The location found by the search is selected in the Memory panel.
	Note that if an illegal value is specified or while the program is being executed, a message will be appear, and the memory search will not be performed.
	If focus moves to this dialog box while the memory panel is hidden or another panel has focus, then this button will be disabled.
Cancel	Cancels the memory search and closes this dialog box.
Help	Displays the help for this dialog box.

Trace Search dialog box [IECUBE][IECUBE2][Simulator]

This dialog box is used to search trace data (see "2.11.7   Search the trace data").

The search can be performed at the instruction or source level.

Figure A-63.   Trace Search Dialog Box
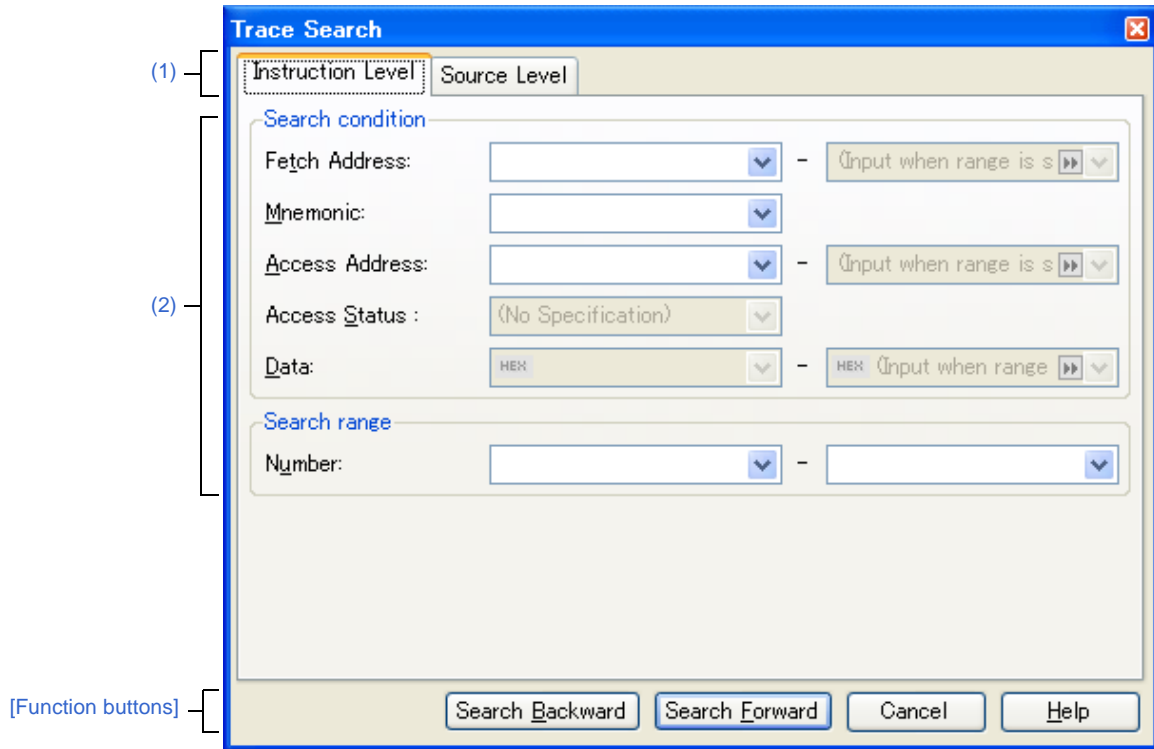
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Trace panel [IECUBE][IECUBE2][Simulator], select ⌖ button on the toolbar.
- On the Trace panel [IECUBE][IECUBE2][Simulator], select [Find…] from the context menu.

[Description of each area]

(1) Tab selection area

Select a tab to switch the level of the search.

This dialog box has the following two tabs.

- [Instruction Level] tab
- [Source Level] tab

---

(2) Search parameter setup area

Use this area to configure detailed search parameters.

for details on the window elements and how to configure the parameters for a particular tab, see the section for the tab in question.

[Function buttons]

Button	Function
Search Backward	Searches upward (in the direction of larger to smaller numbers) within the specified range. Search matches are selected in the Trace panel [IECUBE][IECUBE2][Simulator]. Note that if an illegal value is specified or while the program is being executed, a message will be appear, and the trace data search will not be performed. If focus moves to this dialog box while the Trace panel is hidden or another panel has focus, then this button will be disabled.
Search Forward	Searches forward (in the direction of smaller to larger numbers) within the specified range. Search matches are selected in the Trace panel [IECUBE][IECUBE2][Simulator]. Note that if an illegal value is specified or while the program is being executed, a message will be appear, and the trace data search will not be performed. If focus moves to this dialog box while the Trace panel is hidden or another panel has focus, then this button will be disabled.
Cancel	Cancels the trace data search and closes this dialog box.
Help	Displays the help for this dialog box.

[Instruction Level] tab

Use this tab to search the acquired trace data at the instruction level.

Caution   **If the Trace panel [IECUBE][IECUBE2][Simulator] is set to Source display mode, then performing an instruction level search via this tab will not perform the target search correctly. In order to perform an instruction level search, set the mode to Mixed display mode or Disassemble display mode.**

Figure A-64.   Trace Search Dialog Box: [Instruction Level] Tab
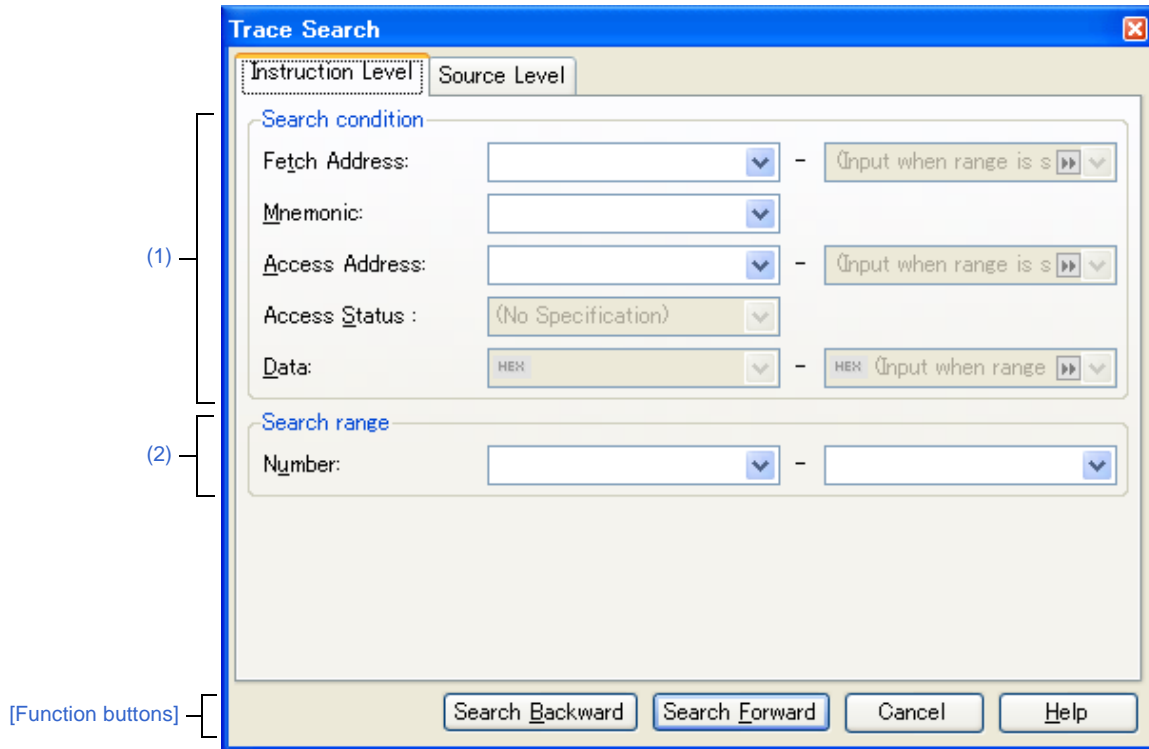
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Trace panel [IECUBE][IECUBE2][Simulator], select ⛏ button on the toolbar.
- On the Trace panel [IECUBE][IECUBE2][Simulator], select [Find…] from the context menu.

[Description of each area]

(1) [Search condition] area

(a) [Fetch Address]

Specify the fetch address if it is a required search parameter.

You can either type address expressions directly into the text boxes, or select them from the input history via the drop-down lists (up to 10 items).

The fetch address can also be specified as a range. In this case, specify a range by specifying address expressions in both the left and right text boxes.

If the right-hand text box is blank or contains the text [(Input value when range is specified)], then the fixed address specified in the left-hand text box will be searched.

Note that if an address value greater than the microcontroller address space is specified, the upper address value is masked.

In addition, an address value greater than the value expressed within 32 bits cannot be specified.

(b) [Mnemonic]

Specify the mnemonic if it is a required search parameter.

The specified character strings in this area are searched within the [Source/Disassemble] area of the Trace panel [IECUBE][IECUBE2][Simulator].

You can either type a mnemonic directly into the text boxes, or select one from the input history via the drop-down list (up to 10 items).

Searches are case-insensitive, and partial matches are also allowed.

(c) [Access Address]

Specify the access address if it is a required search parameter.

You can either type address expressions directly into the text boxes, or select them from the input history via the drop-down lists (up to 10 items).

The access address can also be specified as a range. In this case, specify a range by specifying address expressions in both the left and right text boxes.

If the right-hand text box is blank or contains the text [(Input value when range is specified)], then the fixed address specified in the left-hand text box will be searched.

Note that if an address value greater than the microcontroller address space is specified, the upper address value is masked.

In addition, an address value greater than the value expressed within 32 bits cannot be specified.

(d) [Access Status]

This item is only enabled if a value for [Access Address] is specified.

Select the access type from the following drop-down list.

Select [No Specification] if you do not wish to limit access types.

(No Specification)
Read/Write
Read
Write
Vector Read
DMA

(e) [Data]

This item is only enabled if a value for [Access Address] is specified.

Specify the access data.

You can either type the data directly into the text boxes (in hexadecimal number), or select it from the input history via the drop-down list (up to 10 items).

The data can also be specified as a range. In this case, specify a range by specifying data in both the left and right text boxes.

If the right-hand text box is blank or contains the text [(Input value when range is specified)], then the fixed data specified in the left-hand text box will be searched.

(2) [Search range] area

(a) [Number]

Specify the range within the trace data to search via the number displayed in the [Number] area of the Trace panel [IECUBE][IECUBE2][Simulator].

Specify the starting number in the left text box, and the ending number in the right text box ("0" to "last number" are specified by default).

You can either type the numbers directly into the text boxes (in base-10 format), or select them from the input history via the drop-down lists (up to 10 items).

If the left-hand text box is left blank, it is treated as if "0" were specified.

If the right-hand text box is left blank, it is treated as if the last number were specified.

[Function buttons]

Button	Function
Search Backward	Searches upward (in the direction of larger to smaller numbers) within the specified range.
	Search matches are selected in the Trace panel [IECUBE][IECUBE2][Simulator].
	Note that if an illegal value is specified, a message will be appear, and the trace data search will not be performed.
	If focus moves to this dialog box while the Trace panel is hidden or another panel has focus, then this button will be disabled.
Search Forward	Searches forward (in the direction of smaller to larger numbers) within the specified range.
	Search matches are selected in the Trace panel [IECUBE][IECUBE2][Simulator].
	Note that if an illegal value is specified, a message will be appear, and the trace data search will not be performed.
	If focus moves to this dialog box while the Trace panel is hidden or another panel has focus, then this button will be disabled.
Cancel	Cancels the trace data search and closes this dialog box.
Help	Displays the help for this dialog box.

[Source Level] tab

Use this tab to search the acquired trace data at the source level.

Caution   **If the Trace panel [IECUBE][IECUBE2][Simulator] is set to Disassemble display mode, then performing an source level search via this tab will not perform the target search correctly. In order to perform an source level search, set the mode to Mixed display mode or Source display mode.**

Figure A-65.   Trace Search Dialog Box: [Source Level] Tab
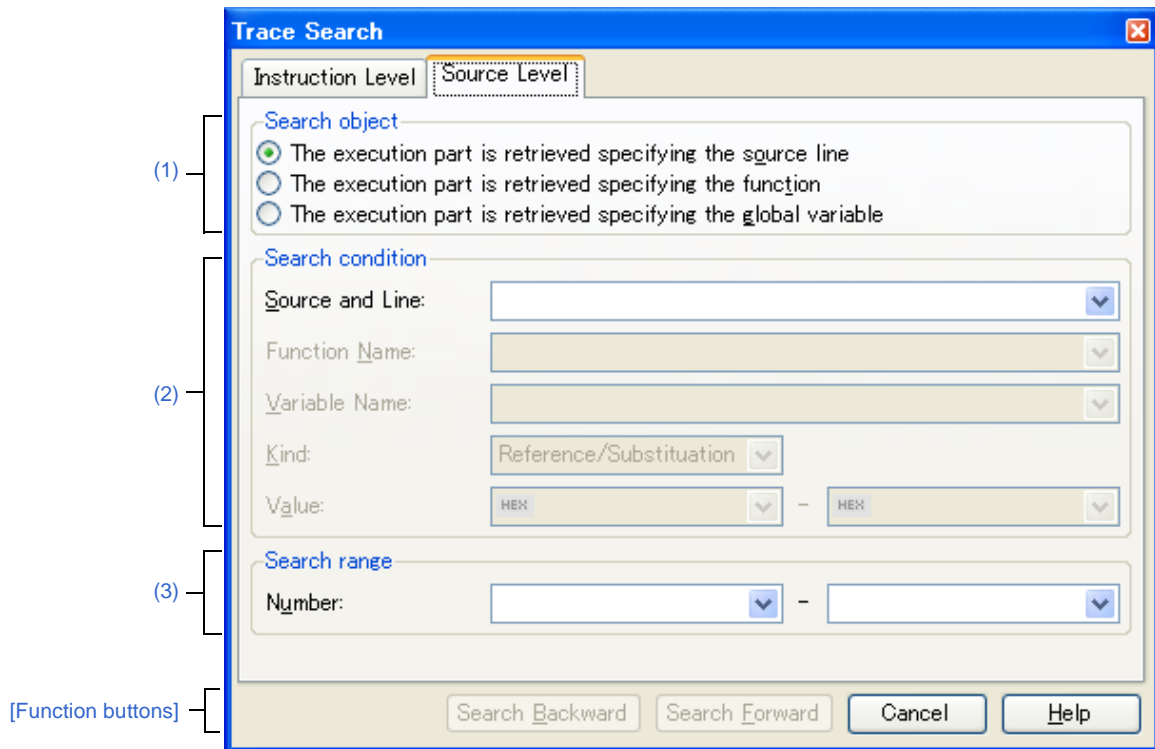
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Trace panel [IECUBE][IECUBE2][Simulator], select ⊞ button on the toolbar.
- On the Trace panel [IECUBE][IECUBE2][Simulator], select [Find…] from the context menu.

[Description of each area]

(1) **[Search object] area**

Select the search object from the following option buttons.

The execution part is retrieved specifying the source line	Finds the execution location in the specified source line (default). Only [Source and Line] will be enabled as a search parameter.
The execution part is retrieved specifying the function	Finds the execution location in the specified function. Only [Function Name] will be enabled as a search parameter.
The execution part is retrieved specifying the global variable	Finds the location at which the specified global variable was accessed. Only [Variable Name], [Kind] and [Value] will be enabled as a search parameters.

(2) **[Search condition] area**

(a) **[Source and Line]**

This item is only enabled if [The execution part is retrieved specifying the source line] is selected.

The specified character strings in this area are searched within the [Line/Address] area of the Trace panel [IECUBE][IECUBE2][Simulator].

You can either type the character strings of the source line to be find directly into the text box, or select them from the input history via the drop-down list (up to 10 items).

Searches are case-insensitive, and only complete matches are retrieved.

    Examples 1. main.c#40
            2. main.c
            3. main

(b) **[Function Name]**

This item is only enabled if [The execution part is retrieved specifying the function] is selected.

You can either type the function name to be find directly into the text box, or select it from the input history via the drop-down list (up to 10 items).

Searches are case-insensitive, and only complete matches are retrieved.

(c) **[Variable Name]**

This item is only enabled if [The execution part is retrieved specifying the global variable] is selected.

You can either type the variable name to be find directly into the text box, or select it from the input history via the drop-down list (up to 10 items).

Searches are case-insensitive, and only complete matches are retrieved.

(d) **[Kind]**

This item is only enabled if [The execution part is retrieved specifying the global variable] is selected.

Select the access type ([Reference/Substitution], [Reference], or [Substitution]) from the drop-down list.

(e) **[Value]**

This item is only enabled if [The execution part is retrieved specifying the global variable] is selected.

Specify the accessed variable value in hexadecimal number.

You can either type a variable value directly into the text box, or select one from the input history via the drop-down list (up to 10 items).

The variable value can also be specified as a range. In this case, specify a range by specifying variable values in both the left and right text boxes.

If the right-hand text box is blank, then access locations with the fixed variable values specified in the left-hand text box will be searched for.

(3) [Search range] area

(a) [Number]

Specify the range within the trace data to search via the number displayed in the [Number] area of the Trace panel [IECUBE][IECUBE2][Simulator].

Specify the starting number in the left text box, and the ending number in the right text box ("0" to "last number" are specified by default).

You can either type the numbers directly into the text boxes (in base-10 format), or select them from the input history via the drop-down lists (up to 10 items).

If the left-hand text box is left blank, it is treated as if "0" were specified.

If the right-hand text box is left blank, it is treated as if the last number were specified.

[Function buttons]

Button	Function
Search Backward	Searches upward (in the direction of larger to smaller numbers) within the specified range. Search matches are selected in the Trace panel [IECUBE][IECUBE2][Simulator]. Note that if an illegal value is specified, a message will be appear, and the trace data search will not be performed. If focus moves to this dialog box while the Trace panel is hidden or another panel has focus, then this button will be disabled.
Search Forward	Searches forward (in the direction of smaller to larger numbers) within the specified range. Search matches are selected in the Trace panel [IECUBE][IECUBE2][Simulator]. Note that if an illegal value is specified, a message will be appear, and the trace data search will not be performed. If focus moves to this dialog box while the Trace panel is hidden or another panel has focus, then this button will be disabled.
Cancel	Cancels the trace data search and closes this dialog box.
Help	Displays the help for this dialog box.

Go to the Location dialog box

This dialog box is used to move the caret to a specified position.

Figure A-66.   Go to the Location Dialog Box (for Editor Panel)
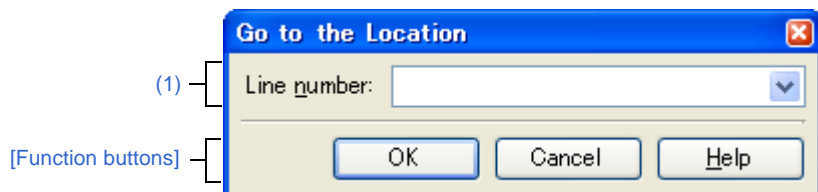
(1)

[Function buttons]

This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- From the [Edit] menu, select [Move to…].
- On the Editor panel, select [Move to…] from the context menu.
- On the Disassemble panel, select [Move to…] from the context menu.
- On the IOR panel, select [Move to…] from the context menu.

[Description of each area]

(1) [Line number], [Address/Symbol], or [IOR] area
Specify the location to which the caret is jump.
You can either type a location directly into the text box, or select one from the input history via the drop-down list (up to 10 items).
The data to specify various depending on the target panel, as follows:

Target Panel	Data Specified
Editor panel	Line number (base-10)
Disassemble panel	Address expression
IOR panel	I/O register name

[Function buttons]

Button	Function
OK	Moves the caret to the specified location, from the beginning of the area displayed in the target panel.
Cancel	Cancels the jump and closes this dialog box.
Help	Displays the help for this dialog box.

Data Save dialog box

This dialog box is used to save data displayed in the Disassemble panel, Memory panel, or Trace panel [IECUBE][IECUBE2][Simulator], and save uploaded data (see "2.5.3   Execute uploading").

This dialog box appears only when connected to the debug tool.

Figure A-67.   Data Save Dialog Box
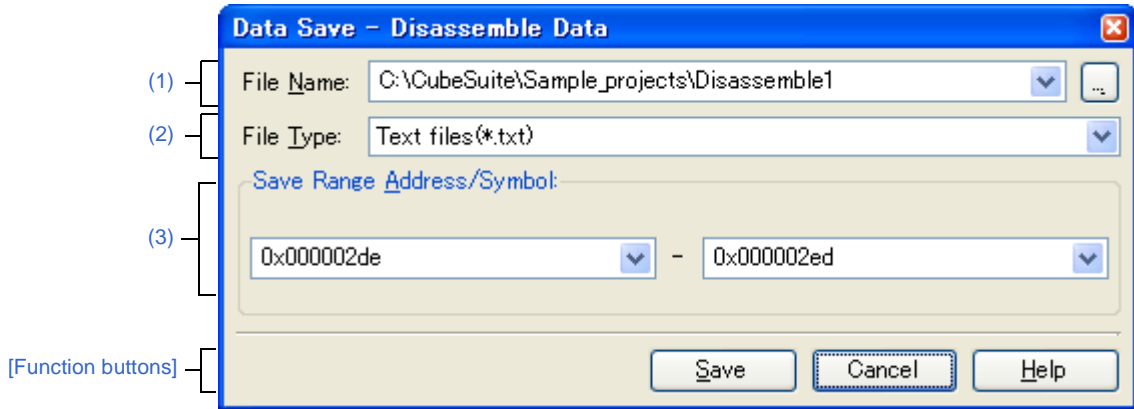
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- With the Disassemble panel in focus, select [Save Disassemble Data As…] form the [File] menu.
- With the Memory panel in focus, select [Save Memory Data As…] form the [File] menu.
- With the Trace panel [IECUBE][IECUBE2][Simulator] in focus, select [Save Trace Data As…] form the [File] menu.
- From the [Debug] menu, select [Upload…].

[Description of each area]

(1) [File Name] area
Specify the name of the file to save.
You can either type a filename directly into the text box (up to 259 characters), or select one from the input history via the drop-down list (up to 10 items).
You can also specify the file by clicking the […] button, and selecting a file via the Select Data Save File dialog box.

(2) [File Type] area
Select the format in which to save the file from the following drop-down list.
The available file formats will differ as follows depending on the type of data being saved.

(a) Saving the data displayed in a panel

Text files (*.txt)	Text format (default)
CSV (Comma-Separated Variables) (*.csv)	CSV format[Note]

Note The data is saved with entries separated by commas (,).

If the data contains commas, each entry is surrounded by double quotes ("") in order to avoid illegal formatting.

(b) Saving upload data

See "Table 2-5. Type of Files That Can be Uploaded".

(3) [Save Range *xxx*] area

Specify the range of data to save.

You can either type ranges directly into the text boxes, or select them from the input history via the drop-down lists (up to 10 items).

The method of specifying the ranges will differ as follows depending on the type of data to be saved.

Type of Data	Description
Disassemble panel	Specify the range of addresses to save via the start and end addresses. Ranges can be entered as base-16 numbers or as address expressions. When a range is selected in the panel, that range is specified by default. When there is no selection, then the range currently visible in the panel is specified.
Memory panel	Specify the range of memory to save via the start and end addresses. Ranges can be entered as base-16 numbers or as address expressions. When a range is selected in the panel, that range is specified by default. When there is no selection, then the range currently visible in the panel is specified.

Trace panel [IECUBE][IECUBE2][Simulator]	- Specifying a range to save
	  Specify the trace range to save via the start and end trace numbers[Note].
	  Ranges can only be entered as base-10 numbers.
	- Saving all trace data
	  From the drop-down list to the left, select [All Trace Data]. The text box to the right is disabled. All currently acquired trace data will be saved.
	The range currently visible in the panel is specified by default.

Upload data	Specify the range of memory to save via the start and end addresses. Ranges can be entered as base-16 numbers or as address expressions.

Note These are the numbers shown in the [Number] area of the Trace panel.

[Function buttons]

Button	Function
Save	Saves the data to a file with the specified filename, in the specified format.
Cancel	Cancels the save and closes this dialog box.
Help	Displays the help for this dialog box.

Progress Status dialog box

This dialog box is used to display the progress of long processes.

This dialog box closes automatically when the currently executing process completes.

Figure A-68.   Progress Status Dialog Box
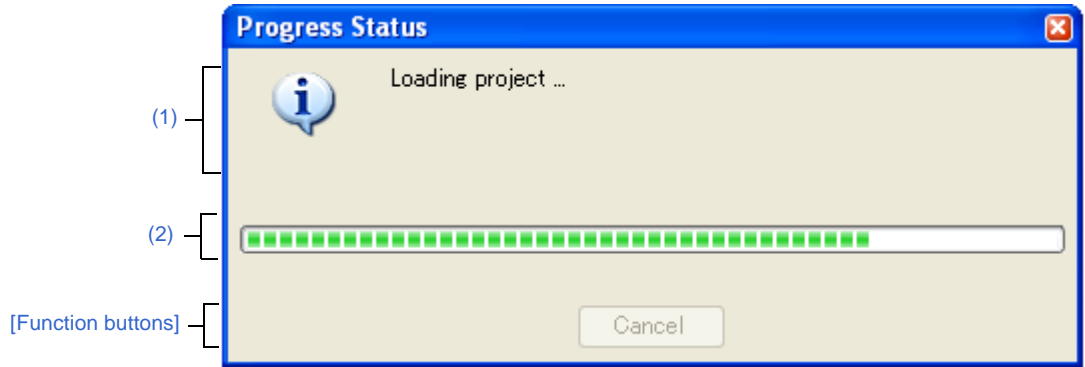
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- This dialog box appears automatically when a message is displayed during a long process.

[Description of each area]

(1)   Message display area

Displays messages during processing (cannot be edited).

(2)   Progress bar

The amount of progress mad toward completing the current progress is indicated by the length of the bar.

The dialog box will automatically close when the progress reaches 100% (the length of the bar reaches the right end).

[Function buttons]

Button	Function
Cancel	Cancels the currently executing process, and closes this dialog box. Note that this button will be disabled if the currently executing process cannot be interrupted.

Option dialog box

This dialog box is used to configure the CubeSuite environment.

All settings made via this dialog box are saved as preferences for the current user.

Figure A-69.   Option Dialog Box
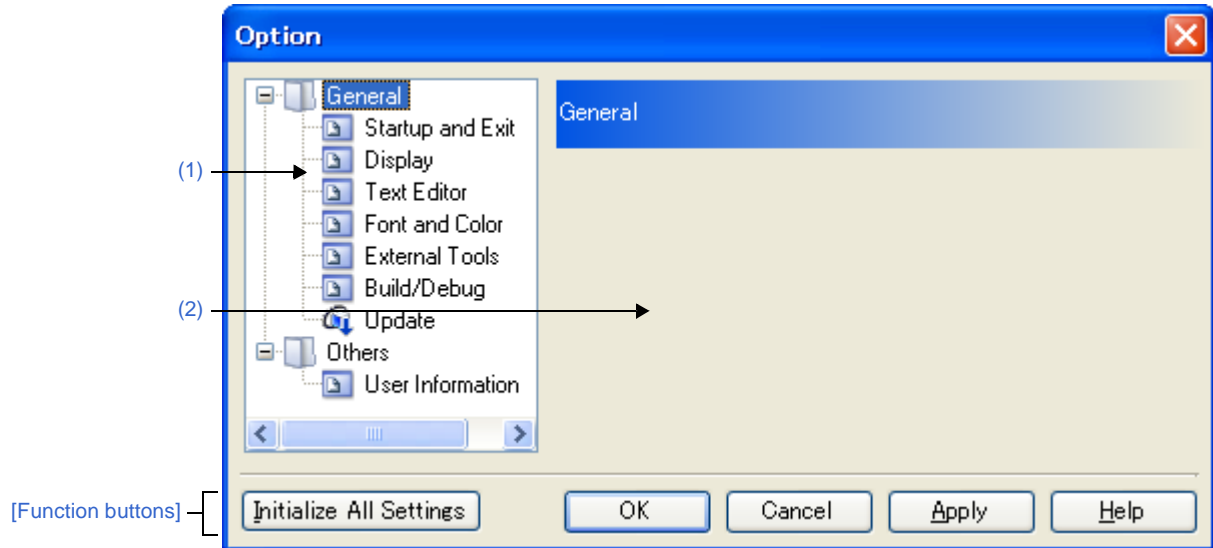
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- From the [Tool] menu, select [Option...].

[Description of each area]

(1) Category selection area

Select the items to configure from the following categories.

Category	Description
[General - Startup and Exit] category	Configure startup and shutdown.
[General - Display] category	Configure messages from the application.
[General - Text Editor] category	Configure the text editor.
[General - Font and Color] category	Configure the fonts and colors shown on each panel.
[General - External Tools] category	Configure the startup of external tools.
[General - Build/Debug] category	Configure building and debugging.
[General - Update] category	Configure updating.
[Others - User Information] category	Configure user information.

---

(2) Setting area

This area is used to configure the various options for the selected category.

For details about configuration for a particular category, see the section for the category in question.

[Function buttons]

Button	Function
Initialize All Settings	Restore all settings on this dialog box to their default values. Note, however, that newly added items in the [General - External Tools] category will not be removed.
OK	Apply all setting and closes this dialog box.
Cancel	Ignore the setting and closes this dialog box.
Apply	Apply all setting (does not close this dialog box).
Help	Display the help of this dialog box.

[General - Font and Color] category

Use this category to configure general settings relating to fonts and colors on each panel.

Figure A-70.   Option Dialog Box ([General - Font and Color] Category)
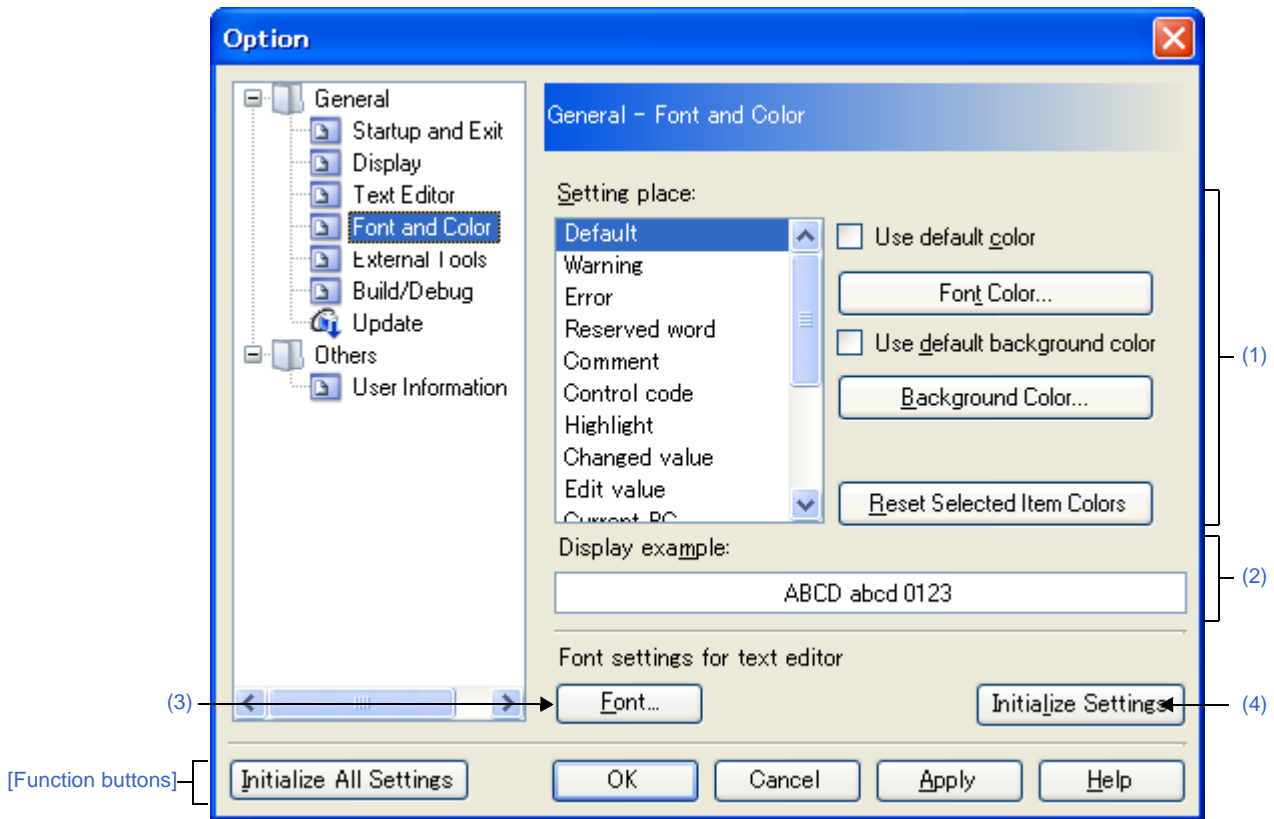
The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- From the [Tool] menu, select [Option...].

[Description of each area]

(1) Color options area

Use this area to configure the colors.

(a) [Setting place] area

Select a location from the list for which the color will be specified.

The relationships between the list items and default color settings are as follows:

Item	Example			Description
Default[Note]	AaBbCc	Font color	Black	The standard display color in all windows and panels.
		Background color	White	
Warning	AaBbCc	Font color	Blue	Warning messages are displayed in this color in the Output panel, and file names with warnings are displayed in this color in the Project Tree panel.
		Background color	Default color	
Error	AaBbCc	Font color	Red	Error messages are displayed in this color in the Output panel, and file names with errors are displayed in this color in the Project Tree panel.
		Background color	Whitesmoke	
Reserved word	AaBbCc	Font color	Maroon	The reserved words of your compiler/assembler are displayed in this color in the Editor panel.
		Background color	Default color	
Comment	AaBbCc	Font color	Green	Comments (in the case of a C source file, "/* ... */") are displayed in this color in the Editor panel.
		Background color	Default color	
String literal	AaBbCc	Font color	Gray	String literals are displayed in this color in the Editor panel.
		Background color	Default color	
Control code	AaBbCc	Font color	Darkcyan	Control characters are displayed in this color in the Editor panel.
		Background color	Default color	
Highlight	AaBbCc	Font color	White	Highlighted areas in plug-in products and the like are displayed in this color.
		Background color	Mediumvioletred	
Changed value	AaBbCc	Font color	Sienna	Values changed via the execution of a program are displayed in this color in the Memory panel, CPU Register panel, Local Variables panel, IOR panel and Watch panel.
		Background color	Lightyellow	
Edit value	AaBbCc	Font color	Blue	Values forcibly modified by user are displayed in this color in the Memory panel, CPU Register panel, Local Variables panel, IOR panel and Watch panel.
		Background color	Default color	
Current PC	AaBbCc	Font color	Black	The line with the current PC position is displayed in this color in the Editor panel and Disassemble panel.
		Background color	Gold	
Breakpoint	AaBbCc	Font color	Black	The line at which a breakpoint is set is displayed in this color in the Editor panel and Disassemble panel.
		Background color	Lightsalmon	
Update periodic	AaBbCc	Font color	Deeppink	Areas configured for real-time display updates are shown in this color in the Memory panel and Watch panel.
		Background color	Default color	

Item	Example			Description
Read or fetch	AaBbCc	Font color	Default color	Locations that have been read or fetched are displayed in this color in the Trace panel [IECUBE][IECUBE2][Simulator].
		Background color	Palegreen	
Write	AaBbCc	Font color	Default color	Locations that have been written are displayed in this color in the Trace panel [IECUBE][IECUBE2][Simulator].
		Background color	Orange	
Read and write	AaBbCc	Font color	Default color	Locations that have been read and written are displayed in this color in the Trace panel [IECUBE][IECUBE2][Simulator].
		Background color	Paleturquoise	
Coverage 100%	AaBbCc	Font color	Default color	The line for which code coverage is at 100 % is displayed in this color in the Editor panel and Disassemble panel.
		Background color	Lightgreen	
Coverage 1 - 99%	AaBbCc	Font color	Default color	The line for which code coverage is at 1 to 99 % is displayed in this color in the Editor panel and Disassemble panel.
		Background color	Lightpink	
Coverage 0%	AaBbCc	Font color	Default color	The line for which code coverage is at 0 % (not yet executed) is displayed in this color in the Editor panel and Disassemble panel.
		Background color	Lightgray	
Invalid	AaBbCc	Font color	Gray	Non-mapped areas in the Memory panel and filenames in the Project Tree panel that do not actually exist are displayed in this color.
		Background color	Default color	

Note   The [Default] text and background colors depends on the Windows settings of the host computer. Here, we use the Windows defaults, which are black text and white background.

(b) [Use default color]

☑	Display items selected via the [Setting place] area using the standard text color.
☐	Display items selected via the [Setting place] area with a user-defined text color. The [Font color...] button is enabled.

(c) [Use default background color]

☑	Display items selected via the [Setting place] area using the standard background color.
☐	Display items selected via the [Setting place] area with a user-defined background color. The [Background Color...] button is enabled.

(d) Buttons

Font Color...	The Edit Colors Dialog Box opens. Specify the text color of the item selected via the [Setting place] area.
	Note, however, that this button will be disabled if the [Use default color] check box is selected.
Background Color...	The Edit Colors Dialog Box opens. Specify the background color of the item selected via the [Setting place] area.
	Note, however, that this button will be disabled if the [Use default background color] check box is selected.
Reset Selected Item Colors	Reset the color information for the item selected via the [Setting place] area to the defaults.

Figure A-71.   Edit Colors Dialog Box
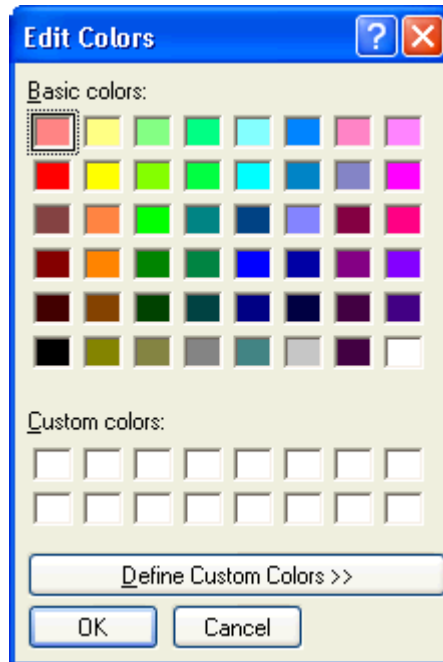
(2) [Display example] area

Display sample text using the color and font settings from the Color options area and the [Font settings for text editor] area.

By default the string "AaBbCc" is shown, but you can type an arbitrary string directly into the text box.

(3) [Font settings for text editor] area

Click the [Font...] button to open the Font Dialog Box and configure the fonts for your text editor.

Figure A-72. Font Dialog Box
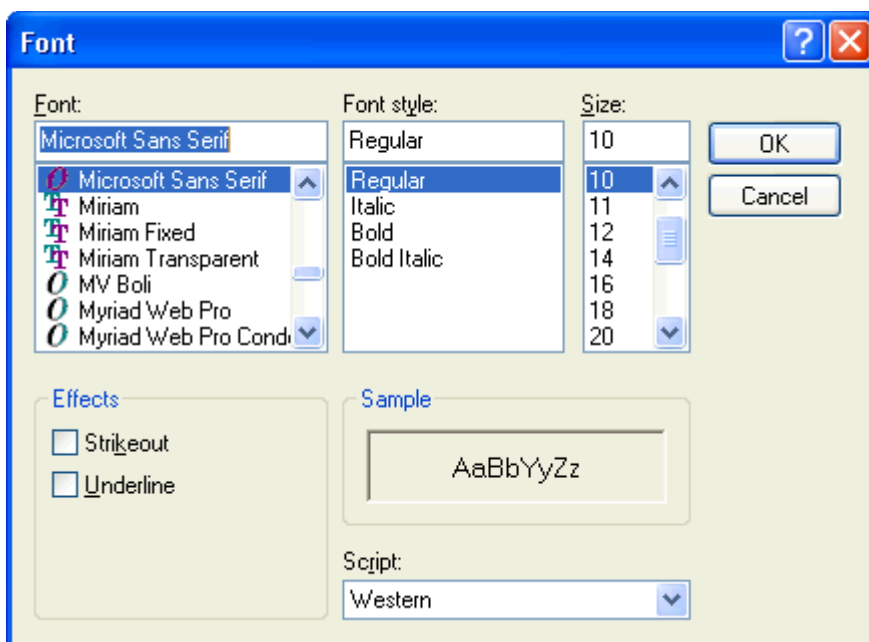
(4) Buttons

Initialize Settings	Returns all currently displayed setting to their default values.

[Function buttons]

Button	Function
Initialize All Settings	Restore all settings on this dialog box to their default values. Note, however, that newly added items in the [General - External Tools] category will not be removed.
OK	Apply all setting and closes this dialog box.
Cancel	Ignore the setting and closes this dialog box.
Apply	Apply all setting (does not close this dialog box).
Help	Display the help of this dialog box.

[General - Build/Debug] category

Use this category to configure general setting relating to building and debugging.

Figure A-73.   Option Dialog Box ([General - Build/Debug] Category)
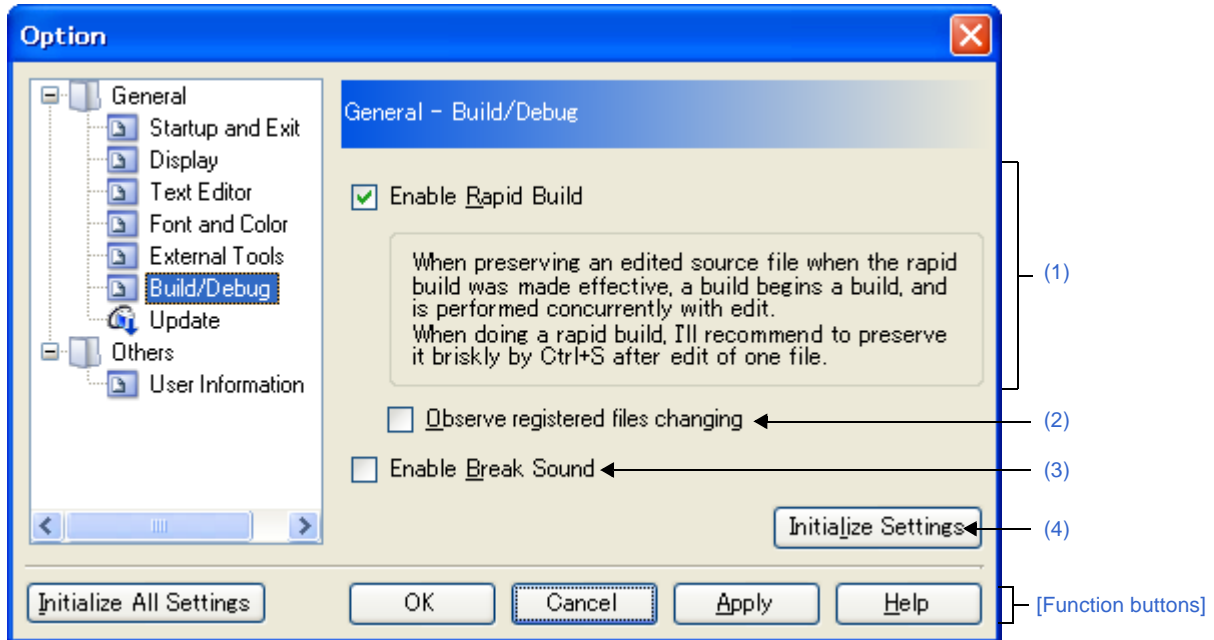
The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- From the [Tool] menu, select [Option...].

[Description of each area]

(1) [Enable Rapid Build]

☑	Enable the rapid build ^{Note} feature (default).
☐	Do not use the rapid build feature.

Note   This feature automatically begins a build when the source file being edited is saved.
Enabling this feature makes it possible to perform builds while editing source files.
If this feature is used, we recommend saving frequently after editing source files.

(2) [Observe registered files changing]

This item is only enabled if the [Enable Rapid Build] check box is selected.

☑	Start a rapid build when a source file registered in the project is edited or saved by an external text editor or the like.
☐	Do not start a rapid build when a source file registered in the project is edited or saved by an external text editor or the like (default).

Caution   **Files that are in the project folder and have been registered to the project can be monitored.**
                              The rapid build will not finish if this item is selected, and the files to be built have been registered for automatic editing or overwriting (e.g. by commands executed before or after the build).
                              If the rapid build does not finish, unselect this item, and stop the rapid build.

(3) [Enable Break Sound]

☑	Beep when the execution of a program is halted due to a break event (Hardware or Software break).
☐	Do not beep when the execution of a program is halted due to a break event (Hardware or Software break) (default).

(4) Buttons

Initialize Settings	Return all currently displayed setting to their default values.

[Function buttons]

Button	Function
Initialize All Settings	Restore all settings on this dialog box to their default values. Note, however, that newly added items in the [General - External Tools] category will not be removed.
OK	Apply all setting and closes this dialog box.
Cancel	Ignore the setting and closes this dialog box.
Apply	Apply all setting (does not close this dialog box).
Help	Display the help of this dialog box.

Select Download File dialog box

This dialog box is used to select a downloaded file.

Figure A-74. Select Download File Dialog Box
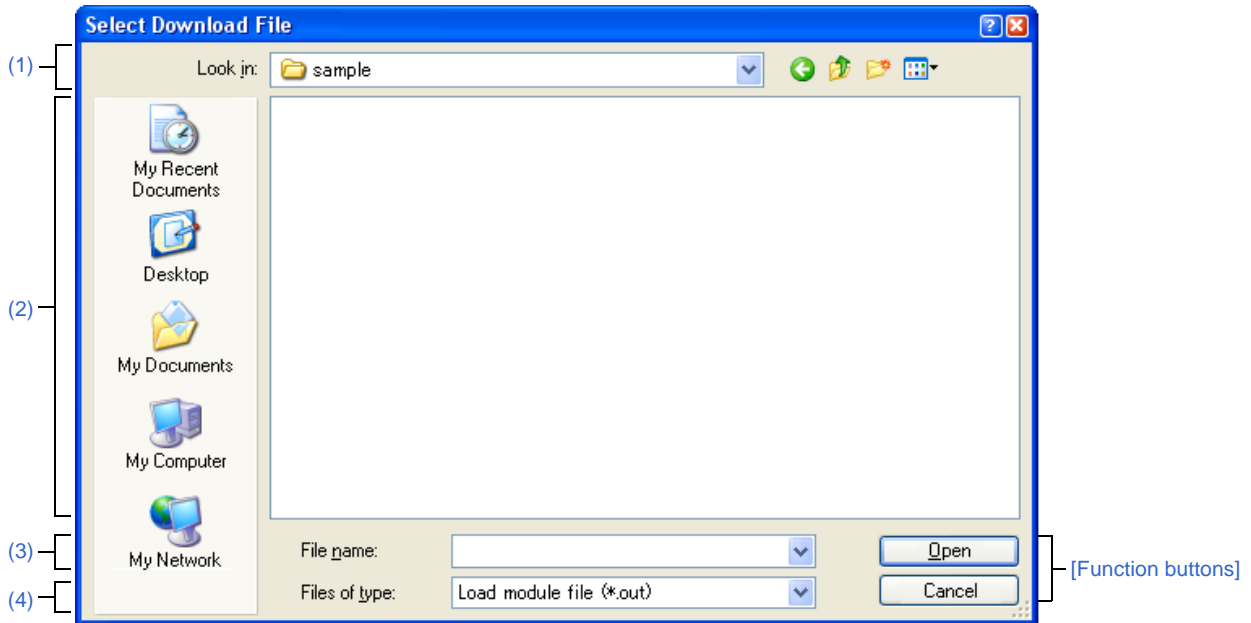
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the [Download file property] area in the Download Files dialog box, click the [...] button on the [File] item.

[Description of each area]

(1) [Look in] area

Select the folder in which the downloaded file is located from the drop-down list.

(2) List of files area

This area displays a list of files matching the conditions selected in the [Look in] and [Files of type] areas.

(3) [File name] area

Specify the name of the downloaded file.

---

(4) [Files of type] area

Select the type of the file from the following drop-down list.

Load module file (*.lmf) **[CX]**	Load module format (default)
Load module file (*.out) **[CA850]**	Load module format (default)
Hex file (*.hex)	Hex format
Binary data file (*.bin)	Binary format
All files (*.*)	All file formats

[Function buttons]

Button	Function
Open	Adds the specified file to the Download Files dialog box.
Cancel	Closes the dialog box.

> **Save As dialog box**

This dialog box is used to save the contents of the panel into a specified file.

Figure A-75.   Save As Dialog Box
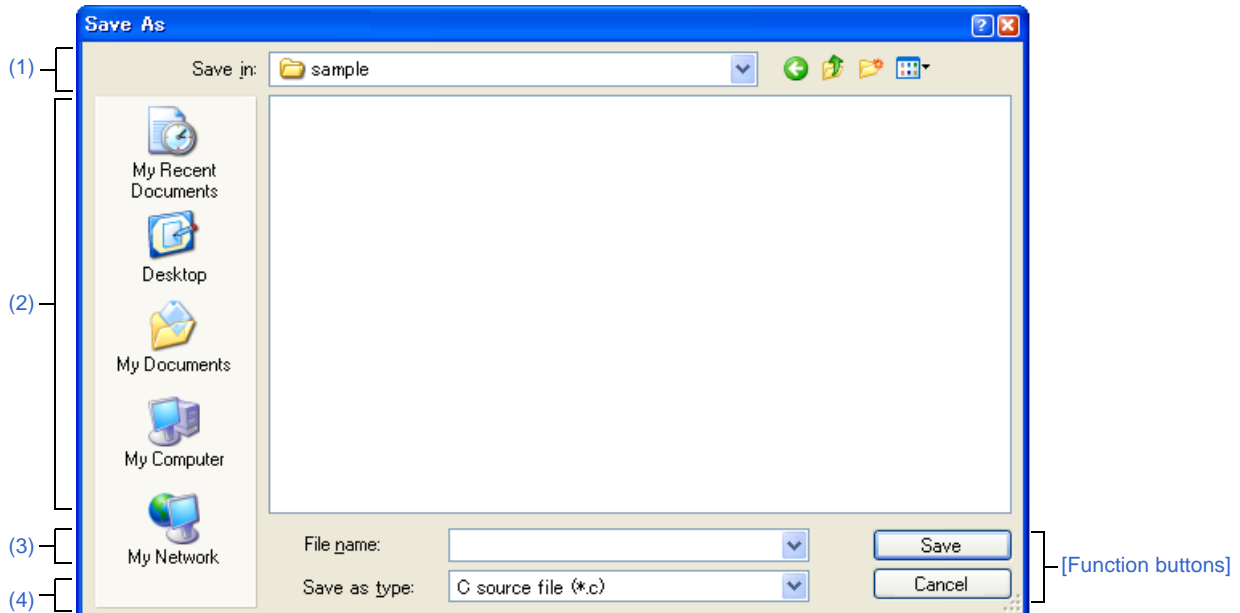
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- With the Editor panel in focus, select [Save *file name* As...] from the [File] menu.
- With the CPU Register panel in focus, select [Save CPU Register Data As...] from the [File] menu.
- With the Watch panel in focus, select [Save Watch Data As...] from the [File] menu.
- With the IOR panel in focus, select [Save IOR Data As...]. from the [File] menu.
- With the Call Stack panel in focus, select [Save Call Stack Data As...] from the [File] menu.
- With the Local Variables panel in focus, select [Save Local Variable Data As...] from the [File] menu.
- With the Output panel in focus, select [Save *tab name* As...] from the [File] menu.

[Description of each area]

(1)   [Save in] area

Select the folder in which the file is saved from the drop-down list.

(2)   List of files area

This area displays a list of files matching the conditions selected in the [Save in] and [Save as type] areas.

(3)   [File name] area

Specify the name of the file to save.

---

(4) [Save as type] area

(a) Editor panel

The following file types are displayed depending on the file being edited.

Text file (*.txt)	Text format
C source file (*.c)	C language source file
Header file (*.h;*.inc)	Header file
Assemble file (*.asm) **[CX]**	Assembly language source file
Assemble file (*.s)	Assembly language source file
Link directive file (*.dir;*.dr)	Link directive file
Section file (*.sf) **[CA850]**	Section file
Symbol information file (*.sfg) **[CX]**	Symbol information file
Map file (*.map)	Map file
Hex file (*.hex)	Hex file

(b) CPU Register panel/Watch panel/IOR panel/Call Stack panel/Local Variables panel

The following file types are displayed.

Select the format in which to save the file from the drop-down list.

Text file (*.txt)	Text format (default)
CSV (Comma-Separated Variables)(*.csv)	CSV format[Note]

Note   The data is saved with entries separated by commas (,).

If the data contains commas, each entry is surrounded by double quotes (" ") in order to avoid illegal formatting.

(c) Output panel

The following file types are displayed.

The contents can be saved only in text format.

Text file (*.txt)	Text format (default)

[Function buttons]

Button	Function
Save	Saves the file with the specified name.
Cancel	Closes the dialog box.

Select Data Save File dialog box

This dialog box is used to select a saved data file.

Figure A-76.   Select Data Save File Dialog Box
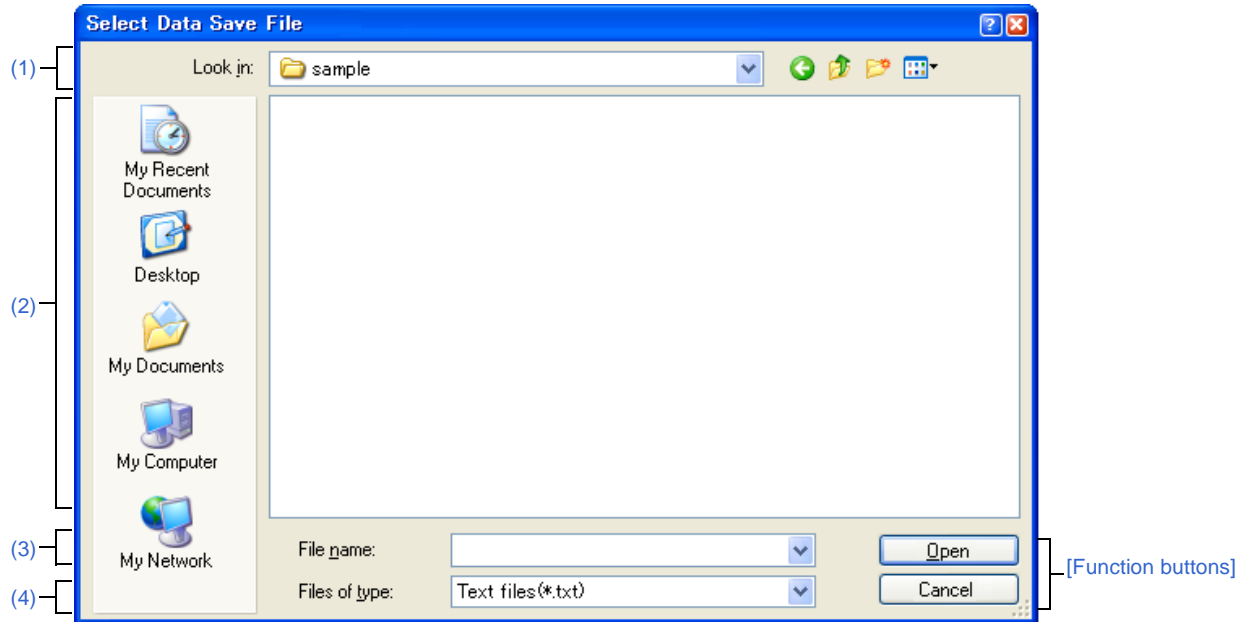
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the [File Name] area in the Data Save dialog box, click the [...] button.

[Description of each area]

(1) [Look in] area

Select the folder in which the file is located from the drop-down list.

(2) List of files area

This area displays a list of files matching the conditions selected in the [Look in] and [Files of type] areas.

(3) [File name] area

Specify the name of the file to save.

---

(4) [Files of type] area

Select the type of the file from the following drop-down list.

The available file formats will differ as follows depending on the type of data being saved.

(a) Saving the data displayed in a panel

Text files (*.txt)	Text format (default)
CSV (Comma-Separated Variables)(*.csv)	CSV format[Note]

Note The data is saved with entries separated by commas (,).

If the data contains commas, each entry is surrounded by double quotes (" ") in order to avoid illegal formatting.

(b) Saving upload data

See "Table 2-5. Type of Files That Can be Uploaded".

[Function buttons]

Button	Function
Open	Specifies the specified file in the Data Save dialog box.
Cancel	Closes the dialog box.

Select Simulator Configuration File dialog box [Simulator]

This dialog box is used to select the simulator configuration file to perform use customization (adding of user models) of the simulator.

Figure A-77.   Select Simulator Configuration File Dialog Box
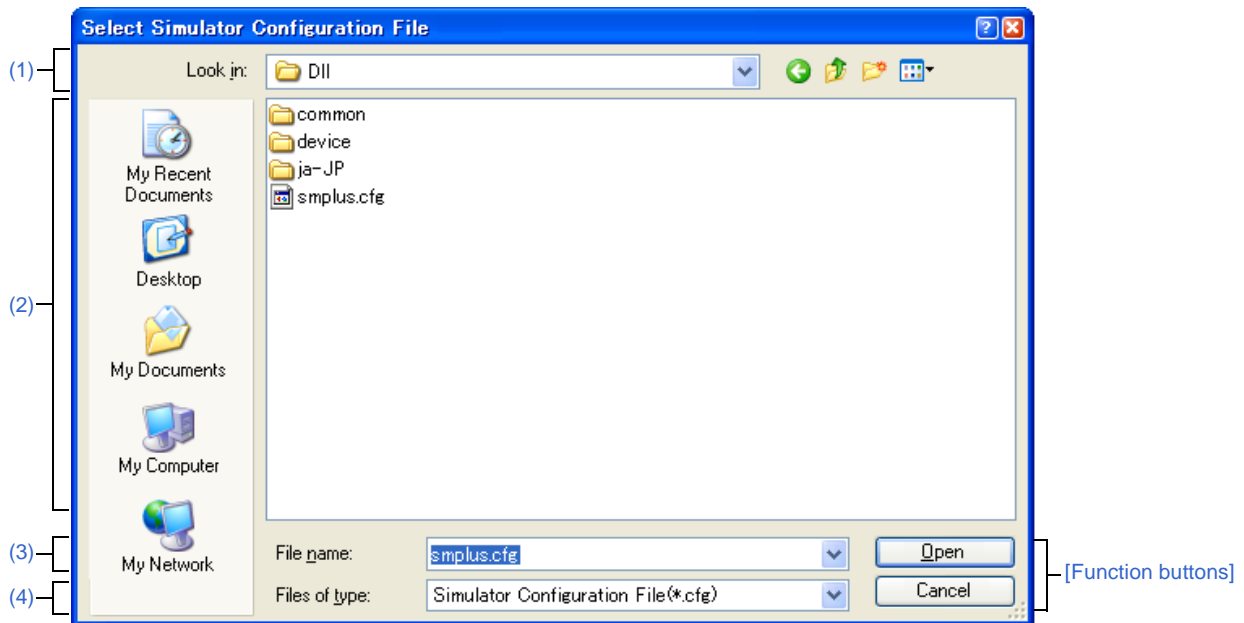
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the [Connect Settings] tab of the Property panel, click the [...] button displayed by selecting the [Simulator configuration file] property in the [Configuration] category.

[Description of each area]

(1) [Look in] area

Select the folder in which the simulator configuration file is located from the drop-down list.

(2) List of files area

This area displays a list of files matching the conditions selected in the [Look in] and [Files of type] areas.

(3) [File name] area

Specify the name of the simulator configuration file to be use.

(4) [Files of type] area

Select the type of the file from the drop-down list.
Note that it is fixed to "Simulator Configuration File(*.cfg)".

[Function buttons]

Button	Function
Open	Uses the specified simulator configuration file.
Cancel	Closes the dialog box.

Simulator GUI window

This window appears automatically by default after connecting to the debug tool when a microcontroller whose Simulator supports peripheral function simulations is selected and [Simulator] is selected as the debug tool to use (see "2.17 Use the Simulator GUI [Simulator]").

In Simulator GUI, other windows (Signal Data Editor window, Timing Chart window, I/O Panel window, and Serial window) are manipulated from this window.

Cautions 1.   **When a microcontroller whose Simulator does not support peripheral function simulations (instruction simulation version) is selected, this window cannot be opened.**

         2.   **This window and windows opened from it cannot be docked to the CubeSuite Main window.**

         3.   **The help for this window will not be displayed even if the [F1] key on the keyboard is pressed when no window opened from this window is opened.**

                   To open the help for this window, select the [Help] menu >> [Main Window] on this window.

         4.   **The [x] button on this window's titlebar is invalid (it is invalid even if using the Aero function of Windows Vista). If you wish to close this window, perform the configuration of the Property panel (see "2.17 Use the Simulator GUI [Simulator]").**

                   In addition, do not use the [Alt] + [F4] key to close this window.

Remark   The language of titlebar/menubar of this window and windows opened from it depends on the setting of [Regional and Language Options] in [Control Panel] of the host machine used (if this setting is set to [Japan]/[Japanese], their titlebar/menubar are displayed in Japanese).

Figure A-78. Simulator GUI Window
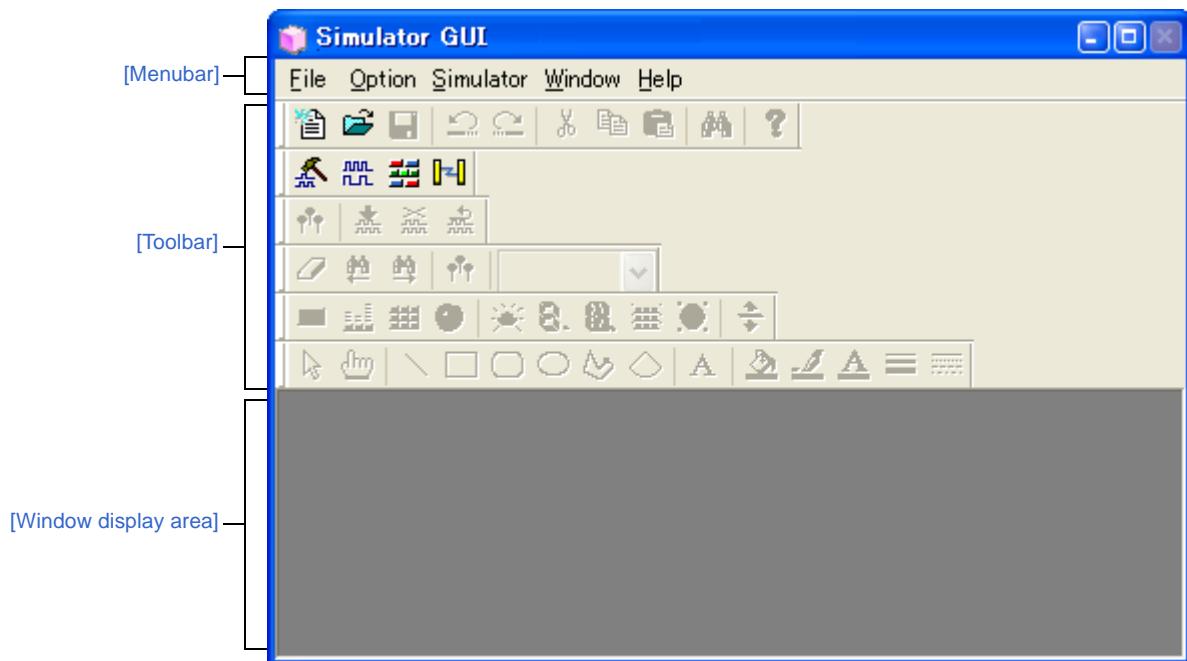
This section describes the following.

- [Menubar]
- [Toolbar]
- [Window display area]

[Menubar]

(1)　[File] menu

New File...	Opens a new window for the Simulator GUI window. Same operation as the 🗋 button.
Open...	Opens the files handled in the Simulator GUI window. Same operation as the 📂 button.
Close	Closes the window currently having the focus.
Save	Overwrites the contents of the window currently having the focus to the file handled in the Simulator GUI window. Same operation as the 💾 button.
Save As...	Saves the contents of the window currently having the focus to the specified file.

(2)　[Edit] menu

This menu varies depending on the window currently having the focus.

For details on this menu items, see "[Dedicated menu]" section in the Signal Data Editor window, Timing Chart window, I/O Panel window, or Serial window.

(3)　[View] menu

This menu varies depending on the window currently having the focus.

For details on this menu items, see "[Dedicated menu]" section in the Signal Data Editor window, Timing Chart window, I/O Panel window, or Serial window.

(4)　[Parts] menu

This menu is added when the I/O Panel window is opened.

For details on this menu items, see the [Parts] menu/[Parts] toolbar.

(5)　[Figure] menu

This menu is added when the I/O Panel window is opened.

For details on this menu items, see the [Figure] menu/[Figure] toolbar.

(6) [Option] menu

ToolBar		Switches on/off display of the toolbar corresponding to the cascade menu.
	Simulator Standard	Selects whether the [Simulator Standard] toolbar is displayed or not.
	Simulator Tools	Selects whether the [Simulator Tool] toolbar is displayed or not.
	Signal Data Editor	Selects whether the [Signal Data Editor] toolbar is displayed or not.
	Timing Chart	Selects whether the [Timing Chart] toolbar is displayed or not.
	Parts	Selects whether the [Parts] toolbar is displayed or not.
	Figure	Selects whether the [Figure] toolbar is displayed or not.
Customize...		Opens the Customize dialog box.

(7) [Simulator] menu

Signal Data Editor	Opens the Signal Data Editor window. Same operation as the button.
Timing Chart	Opens the Timing Chart window. Same operation as the button.
I/O Panel	Opens the I/O Panel window. Same operation as the button.
Serial	Opens the Serial window. Same operation as the button.

(8) [Window] menu

Close All	Closes all windows, except this window.
Cascade	Cascade display of the windows in this window.
Tile	Cascade display of the windows in this window.
Arrange Icons	Rearranges the icons in this window.

(9) [Help] menu

Main Window	Displays the help for this window.
Current Window	Displays the help for the current window.

[Toolbar]

(1) [Simulator Standard] toolbar
(2) [Simulator Tool] toolbar
(3) [Signal Data Editor] toolbar
(4) [Timing Chart] toolbar
(5) [Parts] toolbar
(6) [Figure] toolbar

(1) [Simulator Standard] toolbar

	Opens a new window for the Simulator GUI window.
	Opens the files handled in the Simulator GUI window.
	Overwrites the contents of the window currently having the focus to the file handled in the Simulator GUI window.
	Undoes the immediately preceding operation.
	Restores the status undone by the ⤺ button.
	Cuts the selected range and saves it to the clipboard.
	Copies the selected range and saves it to the clipboard.
	Pastes the clipboard contents.
	Opens the Search Data dialog box.
	Displays the contents of the help.

(2) [Simulator Tool] toolbar

	Opens the Signal Data Editor window.
	Opens the Timing Chart window.
	Opens the Serial window.
	Opens the I/O Panel window.

(3) [Signal Data Editor] toolbar

This toolbar can be used when the Signal Data Editor window has the focus.

For details on this toolbar, see the [Signal Data Editor toolbar].

(4) [Timing Chart] toolbar

This toolbar can be used when the Timing Chart window has the focus.

For details on this toolbar, see the [Timing Chart toolbar].

(5) [Parts] toolbar

This toolbar can be used when the I/O Panel window has the focus.

For details on this toolbar, see the [Parts] menu/[Parts] toolbar.

(6) [Figure] toolbar

This toolbar can be used when the I/O Panel window has the focus.

For details on this toolbar, see the [Figure] menu/[Figure] toolbar.

[Window display area]

This area is used to display various windows (Signal Data Editor window, Timing Chart window, I/O Panel window, or Serial window).

The displayed window can be changed in size or an icon can be created in this area.

> **Customize dialog box**

This dialog box is used to set or change the color and fonts for the Signal Data Editor window, Timing Chart window or Serial window.

Figure A-79.   Customize Dialog Box: [Color] Tab (For The Timing Chart Window)
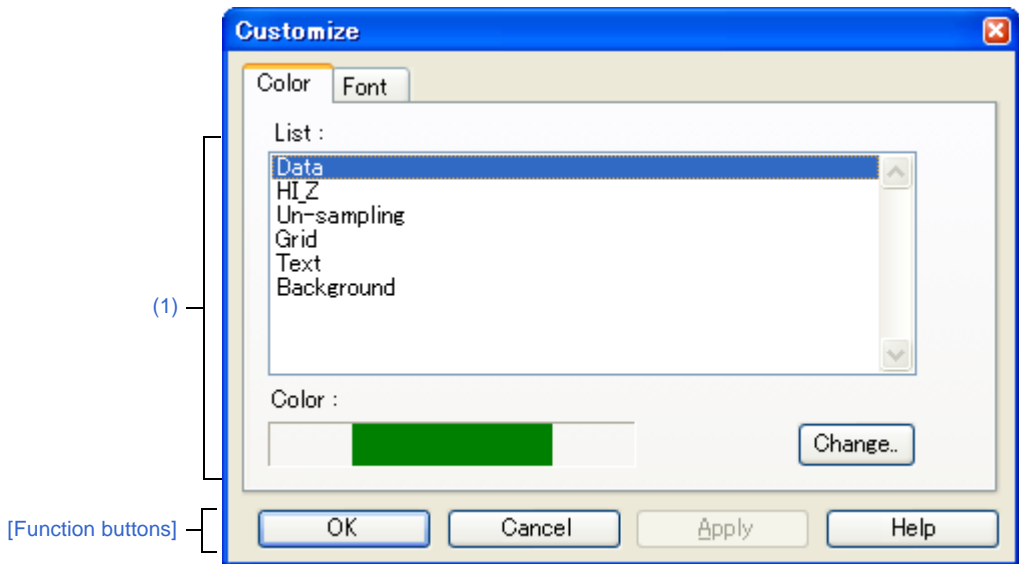
Figure A-80.   Customize Dialog Box: [Font] Tab
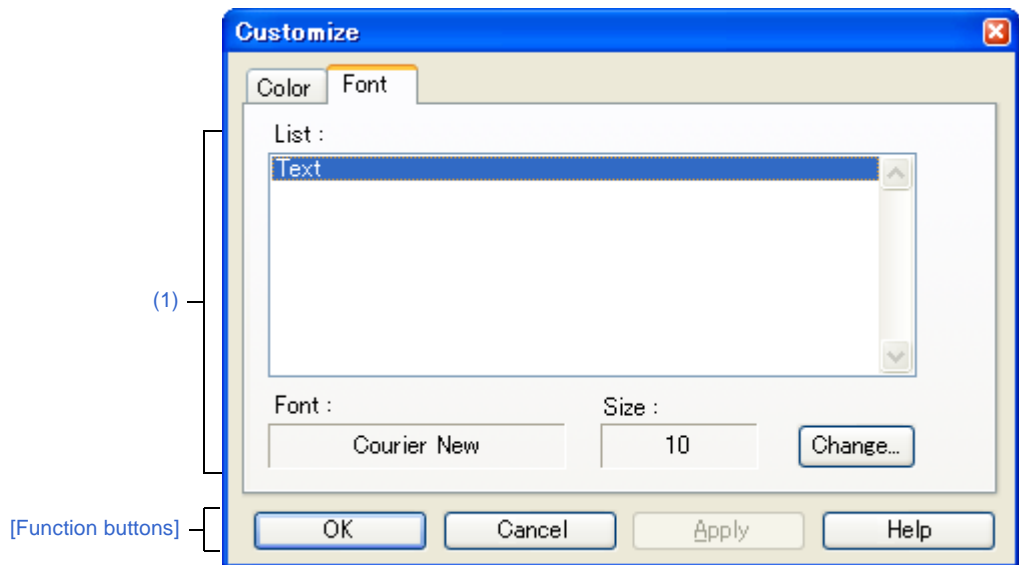
This section describes the following.
- [How to open]
- [[Color] tab]
- [[Font] tab]
- [Function buttons]

[How to open]

- With the Signal Data Editor window, Timing Chart window or Serial window in focus, select [Customize...] from the [Option] menu.

[[Color] tab]

(1) Color setting area

Set and change the color of each part in the window.

List	The parts for which color change is possible are displayed in list form.
	The parts displayed differ depending on the target window.
Color	The currently set color of the part is displayed when that part is selected from the list.
[Change...] button	The color currently set for the relevant part of each listed item can be changed.

[[Font] tab]

(1) Font setting area

Set and change the text font of each part in the window.

List	The parts for which font change is possible are displayed in list form.
Font	The current font name of the part is displayed, when that part is selected from the list.
Size	The current font size of the part is displayed, when that part is selected from the list.
[Change...] button	The font currently set for the relevant part of each listed item can be changed.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

Signal Data Editor window

This window is used to create and edit the signal data that is input to input pins.

The created signal data can be input to the input pin during simulation by selecting the [Edit] menu >> [Signal Input].

This data can also be saved to the signal data file (*.wvi) by selecting the [File] menu >> [Save]/[Save As...] or by saving the project file.

The saved file contents can be restored by selecting the [File] menu >> [Open...] or by loading the project file.

Cautions 1. **If the saved signal data file is opened or the project file is opened while Simulator GUI is running with a microcontroller different from the one used when the signal data file was created, the settings of pins that are not provided in the microcontroller will not be restored.**

2. **The main clock and sub clock cannot be input from this window. Set the main clock/sub clock oscillation frequency on the [Connect Settings] tab in the Property panel.**

3. **If inputting of signals is started during program break, the signals will actually be input when the program is resumed from the break.**

Remarks 1. The following data can be displayed or edited in this window:
- Newly created signal data
- Previously created signal data files
- File of signal data previously obtained by performing simulation and saving the results as output signal data

2. On the titlebar of this window, *"Project file name + Serial number* (from 0).wvi" is displayed when the project file has been loaded. However, after having loaded a project file of PM+, then if you save a file as the project file of CubeSuite, *"Project file name +* CS+ *Serial number* (from 0).wvi" is displayed on the titlebar.

Figure A-81.   Signal Data Editor Window
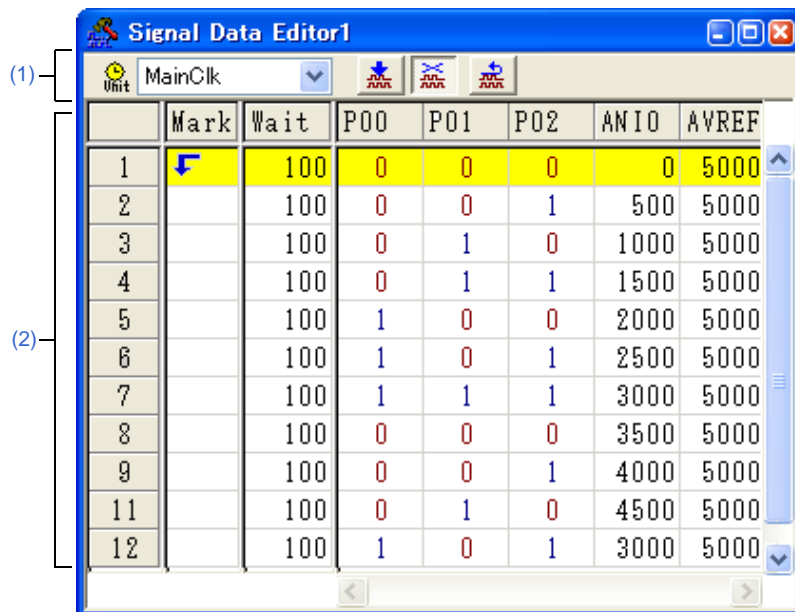
This section describes the following.
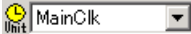
[How to open]

- Click the button button
- Select [Signal Data Editor] from the [Simulator] menu.

[Description of each area]

(1)   Information bar

It can be specified whether this area is displayed or not, by selecting the [View] menu >> [Information Bar].

![MainClk dropdown]	Select the unit of the wait time from the drop-down list. The wait time unit can be changed by selecting the [Edit] menu >> [Time unit].
![icon]	If this button is clicked while the program is running, signal input starts. If this button is clicked while the program is stopped, signal input starts automatically the next time the program execution is started.
![icon]	If this button is clicked while the program is running, signal input is stopped. If this button is clicked while the program is stopped, signal input does not start automatically even if the program execution is started.
![icon]	The current signal input line (line highlighted with yellow) is returned to the beginning.

(2)   Client area

Pin area	Displays the input pin names.
	The input pin to be used is selected via the Select Pin dialog box opened by clicking the ![button] button on the toolbar or selecting the [Edit] menu >> [Select Pin...].
	Data input to pins can be enabled/disabled by selecting the [Edit] menu >> [Pin Status].

Line number area	Displays line number.
	This area is used when performing editing in line units.
	Note that up to 1,048,576 (= 1M) lines can be specified as the signal data.

Mark area	Displays the loop information for the specified input value.
	The loop information is specified by selecting from the context menu or the [Edit] menu >> [Mark] in the relevant field.
	The following marks are displayed after the loop information has been specified.

	Loop start location (endless loop)
⌐n	Loop start location (with loop count)
↑	Loop end location

Wait area	Displays as "wait time" the timing at which the specified input value is input to the pins.
	The wait time is specified by directly writing numeric values in the relevant field.
	Note that numeric values (decimal code) from 0 to 4,294,967,295 can be specified (values that exceed 4,294,967,295 can be set by using one more line).
	The wait time unit can be changed by selecting the [Edit] menu >> [Time unit].

Data area	Displays the input value input to the pins.
	The input value is specified by directly writing numeric value in the relevant field.
	Note that the input rules differ as follows according to the pin type.

Digital pins	Any one of the following one character	
	0	LOW signal
	1	HIGH signal
	Z	Hi-Z signal (case insensitive)
Analog pins	A decimal value from 0 to 5000 (unit: mV)	

[Dedicated menu (Signal Data Editor window)]

(1) [Edit] menu

Undo	Cannot be selected.
Redo	Cannot be selected.
Cut	Cuts the selected range and saves it to the clipboard.
Copy	Copies the selected range and copies it to the clipboard.
Paste	Pastes the contents of the clipboard to the selected location.
Delete	Deletes the selected range.
Select All	Selects all display data.
Find	Cannot be selected.
Select Pin...	Opens the Select Pin dialog box. The pin(s) for which input signal data is to be created or edited is (are) selected in this dialog box.
Time unit	Selects the wait time unit.

	main clock	Main clock (default)
	usec	Microsecond
	msec	Millisecond

Pin Status		Selects the input status of the selected pin.
	Valid	Enables data input to the pin. (default)
	Invalid	Disables data input to the pin.
Mark		Sets a mark to the selected Mark area.
	Loop Start	Sets the loop start mark.
	Loop End	Sets the loop end mark.
	Loop Dialog	Opens the Loop dialog box. Sets the details of loop information.
Signal Input		Inputs signal data to the simulator.
	Start	Starts signal input.
	Stop	Stops signal input.
	Reset	Returns the current signal input line to the beginning.

(2) [View] menu

Information Bar	Selects whether the information bar is displayed or not.

(3) [Option] menu

Customize...	Opens the Customize dialog box.

[Signal Data Editor toolbar]

	Opens the Select Pin dialog box. The pin(s) for which input signal data is to be created or edited is (are) selected in this dialog box.
	If this button is clicked while the program is running, signal input starts. If this button is clicked while the program is stopped, signal input starts automatically the next time the program execution is started.
	If this button is clicked while the program is running, signal input is stopped. If this button is clicked while the program is stopped, signal input does not start automatically even if the program execution is started.
	The current signal input line (line highlighted with yellow) is returned to the beginning.

[Context menu]

The following context menus are available at each area in the Client area.

(1) Pin area

Valid	Enables data input to the pin (default).
Invalid	Disables data input to the pin.
Select Pin...	Opens the Select Pin dialog box. The pin(s) for which input signal data is to be created or edited is (are) selected in this dialog box.

(2) Line number

Cut	Cuts the selected range and saves it to the clipboard.
Copy	Copies the selected range and copies it to the clipboard.
Paste	Pastes the contents of the clipboard to the selected location.
Delete	Deletes the selected range.

(3) Mark area

Cut	Cuts the selected cell and saves it to the clipboard.
Copy	Copies the selected cell and copies it to the clipboard.
Paste	Pastes the contents of the clipboard to the selected location.
Delete	Deletes the selected cell.
Loop Start	Sets the loop start mark.
Loop End	Sets the loop end mark.
Loop Dialog	Opens the Loop dialog box. Sets the details of loop information.

(4) Wait area

Cut	Cuts the data in the selected cell and saves it to the clipboard. The data in the selected cell becomes 0.
Copy	Copies the data in the selected cell and copies it to the clipboard.
Paste	Pastes the contents of the clipboard to the selected location.
Delete	Deletes the data in the selected cell. The data in the selected cell becomes 0.

(5) Data area

Cut	Cuts the data in the selected cell and saves it to the clipboard. The data in the selected cell becomes "Z" (Hi-Z).
Copy	Copies the data in the selected cell and copies it to the clipboard.
Paste	Pastes the contents of the clipboard to the selected location.
Delete	Deletes the data in the selected cell. The data in the selected cell becomes "Z" (Hi-Z).
Start Signal Input	Starts signal input.
Stop Signal Input	Stops signal input.
Reset Signal Input	Returns the current signal input line to the beginning.

[Operation]

(1) Pin selection
(2) Creating signal data
(3) Data copy and paste
(4) Single-line editing
(5) Signal input
(6) Operation at CPU reset

(1) Pin selection

To create the signal data, it is first necessary to select the pin to be used.

Open the Select Pin dialog box by clicking the ▮ button on the toolbar or selecting the [Edit] menu >> [Select Pin...], and select the pin to be used. Once a pin is selected, its name is displayed in Pin area.

(2) Creating signal data

Create the signal data input to each pin.

(a) Setting of input value

In the Data area, specify the value that is input to each pin (see "Data area").

(b) Setting of input timing

In the Wait area, specify the timing at which the input value is input to each pin as "wait time" (see "Wait area").

(c) Setting of the loop information

When loop processing for the signal data specified in step (a) and (b) is needed, specify the loop information. To specify the loop information, select [Loop Start] from the context menu on the loop start position in the Mark area, and select [Loop End] on the loop stop position.

At this time, the loop count can be specified. In this case, specify the loop count via the Loop dialog box opened by selecting [Loop Dialog...] from the context menu.

The corresponding loop information marks are displayed if the setting of the loop information is completed (see "Mark area").

(3) Data copy and paste

The setting values in Mark area/Wait area/Data area can be copied and pasted.

However, copied data can be pasted only in the same area.

Copy	When one or more (a range of) cells are selected, these cells can be copied by selecting the [Edit] menu >> [Copy] (or pressing the [Ctrl] + [C] key).
Paste	When one or more (a range of) cells are selected, these cells can be pasted by selecting the [Edit] menu >> [Paste] (or pressing the [Ctrl] + [V] key). When multiple (a range of) cells are selected, the copied data is pasted to the cells repeatedly.

(4) Single-line editing

Single lines can be edited by selecting Line number area.

The method is the same as that described in "(3)   Data copy and paste".

Data pasted during line paste (insertion) is inserted immediately before the selected line.

(5) Signal input

Input the created signal data to the input pins of the simulator while simulation is executed.

At this time, the line whose signal is currently being input (current line) is highlighted in yellow during program breaks (this can be changed via [Inputted current line] item in the Customize dialog box), in order to display signal input progress.

There are the following types of signal data input operations:

Signal input start	Click the ▮ button, or select the [Edit] menu >> [Signal Input] >> [Start]. As a result, signal input starts from the current signal input line (highlighted line).
Signal input stop	Click the ▮ button, or select the [Edit] menu >> [Signal Input] >> [Stop]. This stops signal input.

| Signal reset | Click the 🔲 button, or select the [Edit] menu >> [Signal Input] >> [Reset]. |
| | This returns the current signal input line to the beginning. If signal reset was performed during signal input, input continues from the beginning. |

　Remark　The signal data input to the selected pin can be controlled by selecting the [Edit] menu >> [Pin Status] >> [Valid]/[Invalid].

(6) Operation at CPU reset

When CPU reset occurs, the current signal input line returns to the beginning.

If a CPU reset occurs during signal input, input continues from the beginning (same operation as the 🔲 button).

<div style="border:1px solid">

Loop dialog box

</div>

This dialog box is used to perform detailed settings (loop start/stop and loop count) related to the loop information in the Signal Data Editor window.

Figure A-82.　Loop Dialog Box
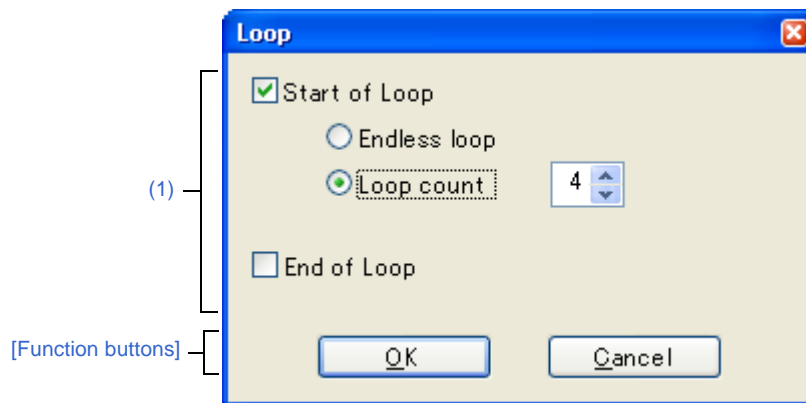
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

On the Signal Data Editor window, any one of the following:
- Double-click the Mark area.
- Select the Mark area, then select [Mark] >> [Loop Dialog...] from the [Edit] menu.

[Description of each area]

(1)　Loop information setting area

Start of Loop	Select this check box to set the Start of Loop.		
	Endless loop	Select this option button to set the Endless Loop.	
	Loop count	Select this option button to set the Loop Count. Sets the count value by spin button.	
		0	Skip loop
		1 to 99	Perform loop the specified count.
End of Loop	Select this check box to set the End of Loop.		

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.

Select Pin dialog box

This dialog box is used when selecting pins displayed in the Signal Data Editor window and the Timing Chart window.

The pin information set in this dialog box can be saved as a pin information file (*.pin) by clicking the [Save] button.

Moreover, the saved file contents can be restored by clicking the [Load] button.

Figure A-83. Select Pin Dialog Box
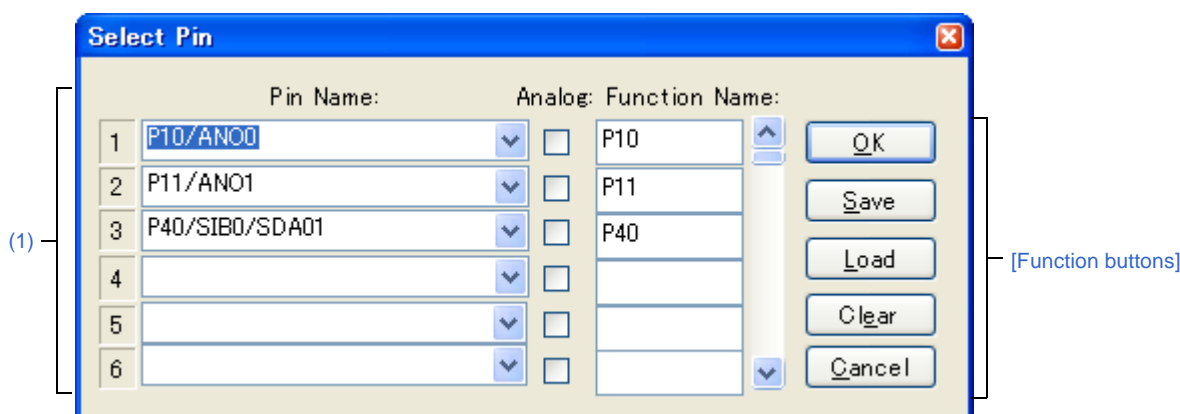
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

On the Signal Data Editor window/Timing Chart window, any one of the following:
- Click the 📊 button.
- Select [Select Pin...] from the [Edit] menu.

[Description of each area]

(1) Connection pins setting area

Up to 256 pins can be selected by using the scrollbar located on the right side of the pin name setting area.

Pin Name	This area is used to specify the pin name to be connected. The connection pins can be specified either via direct input or through selection from the drop-down list.
Analog	Select this check box to use the specified pin as an analog pin.
Function Name	This area is used to set a function name for the pin name. When a character string is input in this area, this character string is displayed as a function name in the pin name part. If nothing is specified, the pin name is displayed.

Remark   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

RENESAS

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box. The pin name (or display name) is applied in the Pin field of the window from where this dialog box was called up.
Save	Saves the display contents to the pin information file (*.pin).
Load	Loads the pin setting information of the specified file (*.pin).
Clear	Deletes the settings.
Cancel	Cancels the settings and closes this dialog box.

Timing Chart window

This window is used to display the output signals and input signals for pins in the form of a timing chart.

This window allows time measurement in main clock units.

The browsed signal data can be saved to the timing chart file (*.wvo) by selecting the [File] menu >> [Save]/[Save As...]. Moreover, the saved file contents can be restored by selecting [File] menu >> [Open...].

If the project file is saved, signal data is not saved but information of the set pins is saved. (Solely saving the project file is sufficient if saving of the measurement result is unnecessary.)

Cautions 1. **If the saved timing chart file is opened or the project file is opened while Simulator GUI is running with a microcontroller different from the one used when the timing chart file was created, the settings of pins that are not provided in the microcontroller will not be restored.**

2. **The main clock and sub clock waveforms cannot be displayed in this window. In addition, when using the external bus interface function, the waveforms of pins used for the external bus interface function cannot be displayed.**

Figure A-84.   Timing Chart Window
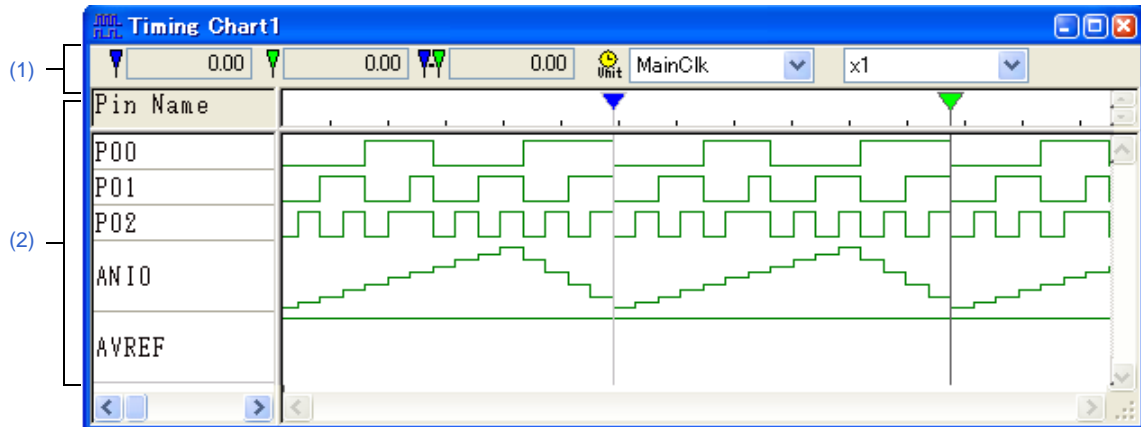
This section describes the following.
- [How to open]
- [Description of each area]
- [Dedicated menu (Timing Chart window)]
- [Timing Chart toolbar]
- [Context menu]
- [Operation]

[How to open]

- Click the 🔲 button.
- Select [Timing Chart] from the [Simulator] menu.

[Description of each area]

(1) Information bar

It can be specified whether this area is displayed or not, by selecting the [View] menu >> [Information Bar].

5553364.00	Clock/time count from simulation start until marker A location.
5554860.00	Clock/time count from simulation start until marker B location.
1496.00	Clock/time count between markers A and B. (Displayed as absolute value.)
MainClk	Select from the drop-down list the time unit for the location information of markers A and B. This item can also be set by using [Time unit] in the [Edit] menu or [Time unit] from the context menu.
x1/2	Selects the waveform data display magnification ratio from the drop-down list. If a part of waveform data may be lost as a result of changing the display magnification ratio, the confirmation dialog box is displayed.

Remarks 1.   Up to 4,294,967,262 clocks can be counted from the simulation start up to the marker position. When the count reaches the maximum value, the counter is cleared to 0 and starts counting again.

          2.   The menu for setting the magnify ratio appears dimmed during program execution, so changing the ratio is unavailable.

(2) Client area

Marker area

Pin Name area   Waveform area
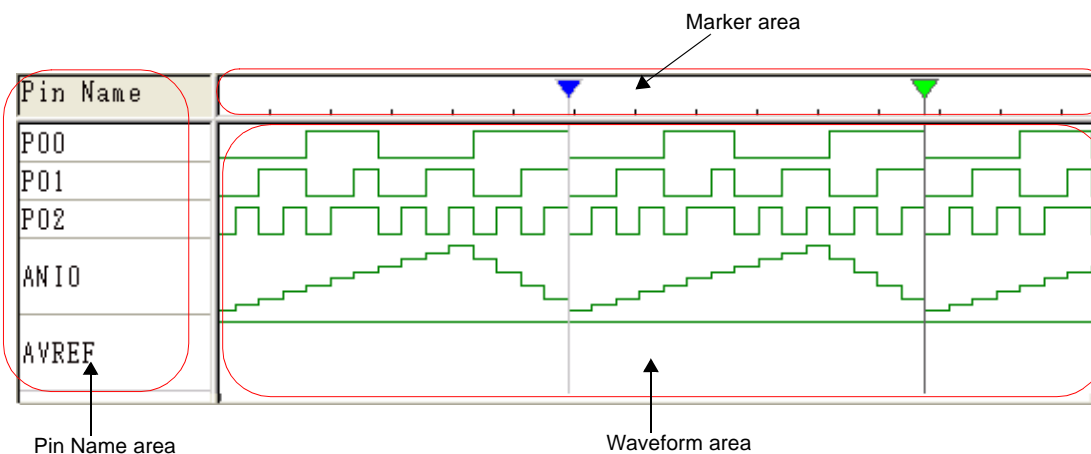
Pin Name area	Displays the names of the pins for which timing chart display is performed. Pin selection is performed by selecting the [Edit] menu >> [Select Pin...] to open the Select Pin dialog box.	
Marker area	Displays the 2 marker headers. These markers can be moved by dragging them with the mouse over this area.	
		Maker A
		Maker B
Waveform area	Performs timing chart display for the data of the pins specified in the Pin area. The following color distinctions are used according to the signal by default.	
	Green	The HIGH and LOW signals of the pins
	Red	High-impedance signals
	Blue	Unsampled signals

Remarks 1. When the buffer is full, the data will be overwritten by the latest data starting from the oldest data because the buffer storing the pin data is in a ring buffer format.

The upper limit of the buffer size is one of the following.
- The number of pin change points: 4,096
- The number of clocks: 2,147,483,631
- Horizontal draw width: 134,217,711 pixels

2. The colors and fonts can be changed via the Customize dialog box opened by selecting the [Option] menu >> [Customize...].

[Dedicated menu (Timing Chart window)]

(1) [Edit] menu

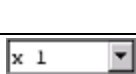|	
Clear	Deletes all the waveform data.
Find...	Opens the Search Data dialog box. Waveform data search is performed in this dialog box.
Search backward	Searches for the change point of the selected pin in the backward direction (toward the left).
Search forward	Searches for the change point of the selected pin in the forward direction (toward the right).
Select Pin...	Opens the Select Pin dialog box. The pin for which the waveform data is to be displayed is selected in this dialog box.
Time unit	Selects the time unit.
main clock	Main clock (default)
usec	Microsecond
msec	Millisecond

(2) [View] menu

Waveform	Switches on/off display of the Maker area and Waveform area.
Information Bar	Switches on/off display of the information bar.
Zoom	Selects the waveform data display magnification ratio from a cascade menu. If a part of waveform data may be lost as a result of changing the display magnification ratio, the confirmation dialog box is displayed.
x 1/32	Sets the magnification ratio to 1/32.
x 1/16	Sets the magnification ratio to 1/16.
x 1/8	Sets the magnification ratio to 1/8.
x 1/4	Sets the magnification ratio to 1/4.
x 1/2	Sets the magnification ratio to 1/2.
x 1	Sets the magnification ratio to 1.
x 2	Sets the magnification ratio to 2.
x 4	Sets the magnification ratio to 4.
x 8	Sets the magnification ratio to 8.
x 16	Sets the magnification ratio to 16.
x 32	Sets the magnification ratio to 32.

(3) [Option] menu

Customize...	Opens the Customize dialog box.

[Timing Chart toolbar]

[eraser icon]	Deletes all the waveform data.
[backward search icon]	Searches for the change point of the selected pin in the backward direction (toward the left).
[forward search icon]	Searches for the change point of the selected pin in the forward direction (toward the right).
[select pin icon]	Opens the Select Pin dialog box. The pin for which the waveform data is to be displayed is selected in this dialog box.
x 1 [drop-down]	Selects the waveform data display magnification ratio from the drop-down list. If a part of waveform data may be lost as a result of changing the display magnification ratio, the confirmation dialog box is displayed.

[Context menu]

The following context menus are available in the Client area.

Clear	Deletes all the waveform data.
Find...	Opens the Search Data dialog box. Waveform data search is performed in this dialog box.
Search backward	Searches for the change point of the selected pin in the backward direction (toward the left).
Search forward	Searches for the change point of the selected pin in the forward direction (toward the right).
Select Pin...	Opens the Select Pin dialog box. The pin for which the waveform data is to be displayed is selected in this dialog box.
Waveform	Switches on/off display of the Maker area and Waveform area.
Time unit	Selects the time unit.
Zoom	Selects the waveform data display magnification ratio. If a part of waveform data may be lost as a result of changing the display magnification ratio, the confirmation dialog box is displayed.
Move MakerA	Moves marker A to the cursor location. The same action can be accomplished by pressing the [Shift] key + left-clicking.
Move MakerB	Moves market B to the cursor location. The same action can be accomplished by pressing the [Ctrl] key + left-clicking.

[Operation]

(1) Pin selection
(2) Display of timing chart
(3) Clearing of timing chart
(4) Timing measurement for timing chart
(5) Data search function
(6) Operation at reset

(1) Pin selection

To display the timing chart, it is first necessary to select the pin to be displayed.

Open the Select Pin dialog box by selecting the [Edit] menu >> [Select Pin...] and select the pin to be edited. Once a pin is selected, its name is displayed in the Pin Name area.

(2) Display of timing chart

The waveforms of the selected pins are displayed in timing chart form through execution of the program.

Remark　The simulation speed can be increased by hiding the timing chart.

To hide the timing chart, select the [View] menu >> [Waveform] (deselect this item).

When the timing chart is hidden, the Maker area and Waveform area appear dimmed and "Display OFF" is displayed in the center.

(3) Clearing of timing chart

Timing chart display is cleared by selecting the [Edit] menu >> [Clear].

(4) Timing measurement for timing chart

The timing between two points can be measured by marking 2 locations using markers A and B.

The time of each marker and the time between markers are displayed in the information bar.

Each marker can be placed at the target position by dragging the marker head. Moreover, it can also be placed at the position of the current mouse cursor by selecting [Move MarkerA]/[Move MarkerB] from the context menu.

The marker that is clicked last becomes the selected marker and can be subjected to the Data search function.

Figure A-85.　Maker A And Maker B
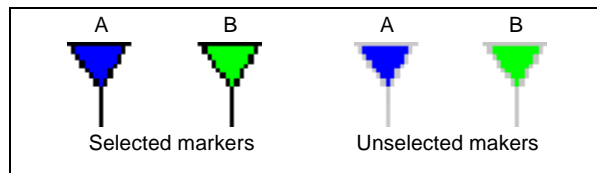
(5) Data search function

There are two data search functions for timing charts.

(a) Simple search

Simple search is a function used to search change points for one pin.

Select the name of the pin for which the search is to be performed in the Pin area and then select the [Edit] menu >> [Search backward] or [Search forward].

As a result, the selected marker moves to the data position at which the change point was detected.

(b) Detailed search

In the case of detailed search, search using a combination of data of multiple pins can be executed.

The search data is set in theSearch Data dialog box displayed by selecting the [Edit] menu >> [Find...].

The selected marker moves to the data position that was hit, similarly to simple search results.

(6) Operation at reset

If CPU reset or Simulator GUI reset occurs, the displayed waveforms of the timing chart are all cleared.

Search Data dialog box

This dialog box is used to search the signal data displayed in the Timing Chart window in detail.

Cautions 1.   Analog I/O signals cannot be searched.

       2.   This dialog box cannot be opened during program execution.

Figure A-86. Search Data Dialog Box
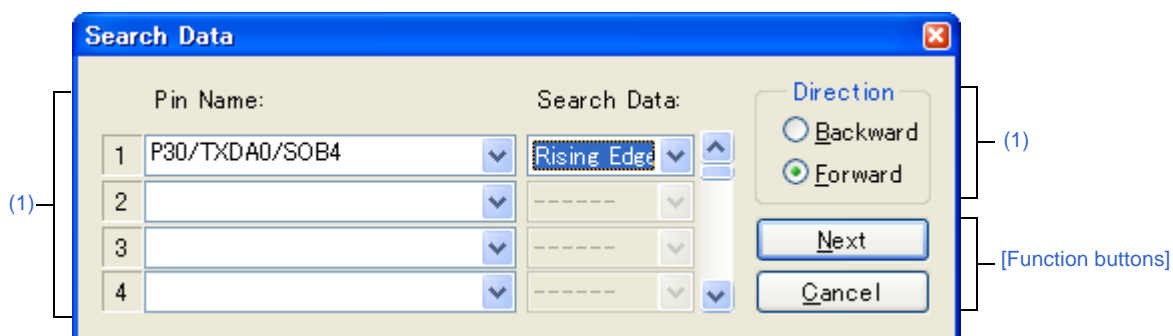
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

On the Timing Chart window, any one of the following:
- Click the 🔍 button.
- Select [Find...] from the [Edit] menu.

[Description of each area]

(1) Search pin setting area

When multiple search conditions have been specified, the signal data that meets all these search conditions is searched.

Up to 48 search conditions can be specified by using the scrollbar located on the right side.

Pin Name	This area is used to specify the pin name to be searched.
	The pins can be specified either via direct input or through selection from the drop-down list.
	Inputting noting makes an area off-limit to data search and input in the corresponding [Search Data] is disabled.

Search Data	Selects the data from drop-down list. The data is searched for the pin to be specified.	
	------	Don't care
	Rising Edge	Searches the rising edge of signal data.
	Falling Edge	Searches the falling edge of signal data.
	Rise/Fall Edge	Searches the rising/falling edge of the signal data.
	High	Searches the signal data that is HIGH.
	Low	Searches the signal data that is LOW.
	Hi Z	Searches the signal data that is high impedance.
Direction	Selects the data search direction by selecting one of the exclusive option buttons. When the [Next] button is clicked, the search is performed in the direction specified in this area.	
	Backward	Searches the data backward (data older than the current location).
	Forward	Searches the data forward (data newer than the current location). (default)

[Function buttons]

Button	Function
Next	Searches in the direction specified. When this button is clicked again following search completion, the next data is searched.
Cancel	Stops the data search and closes the dialog box.

I/O Panel window

This window is used to configure a dummy target system, and manipulate created connected parts.

A dummy target system can be constructed by creating and setting connected parts (figure objects and part objects) in this window. The connected parts for which settings have been performed can be moved to any location within the window, and you can manipulate them during simulation to control signal processing.

Information about parts that are placed in the window can be saved to the I/O panel file (*.pnl) by selecting the [File] menu >> [Save]/[Save As...] or by saving the project file.

The saved file contents can be restored by selecting the [File] menu >> [Open...] or by loading the project file.

Cautions 1. **If the saved I/O panel file is opened while Simulator GUI is running with a microcontroller different from the one used when the file was created, information of the parts connected to the pins that are not provided in the microcontroller will not be restored (the [Pin Name] item in the property dialog boxes for parts remains blank).**

2. **If inputting of signals is started (by an event such as clicking a button) during program break, the signal level will change in actuality when the program is resumed from the break.**

Remark   On the titlebar of this window, *"Project file name + Serial number* (from 0).pnl" is displayed when the project file has been loaded. However, after having loaded a project file of PM+, then if you save a file as the project file of CubeSuite, *"Project file name + CS+ Serial number* (from 0).pnl" is displayed on the titlebar.

Figure A-87.   I/O Panel Window
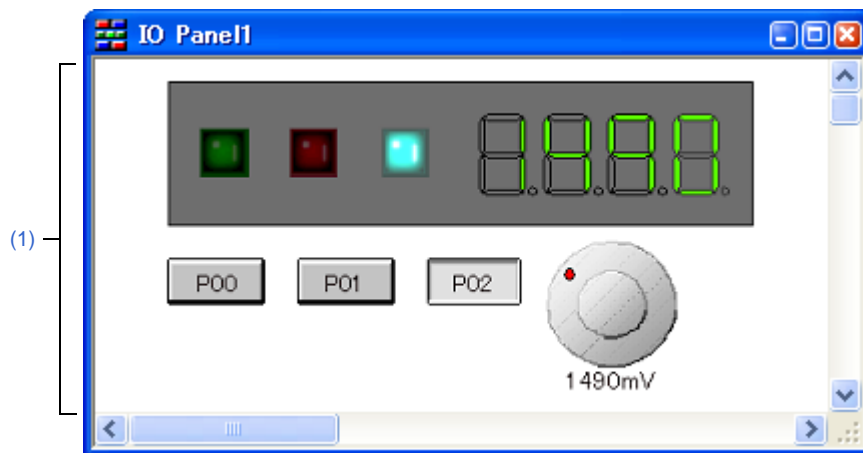
This section describes the following.

- [How to open]
- [Description of each area]
- [Dedicated menu/toolbar (I/O Panel window)]
- [Context menu]
- [Operation]

[How to open]

- Click the ⊞ button.
- Select [I/O Panel...] from the [Simulator] menu.

[Description of each area]

(1) Client area

This area is used to create and set connected parts (figure objects and part objects) in order to construct a dummy target system (see "[Operation]").

[Dedicated menu/toolbar (I/O Panel window)]

The menu items and buttons on the toolbar, which are used to perform operations related to this window, are described below.

(1) [Edit] menu

Select this menu to perform basic editing actions on created objects.

Undo	Undoes the immediately preceding operation, such as object move. Undo can restore up to 5 previous changes.
Redo	Restores the status undone by the [Undo] command.
Cut	Cuts the selected range and saves it to the clipboard.
Copy	Pastes the contents of the clipboard.
Paste	Pastes the contents of the clipboard.
Delete	Deletes the selected range.
Select All	Selects all the objects in the window.
Group	Groups the selected objects.
UnGroup	Ungroups the selected objects.
Bring to Front	Brings the selected object to the front.
Send to Back	Sends the selected object to the back.
Bring Forward	Brings the selected object one panel forward.
Send Backward	Sends the selected object one panel backward.

(2) [View] menu
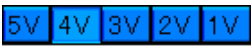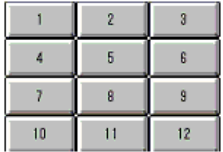
Select this menu to switch the toolbar/status bar display status in this window, or to show/hide various types of information in this window.

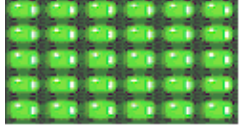|	
ToolBar	Switches on and off the display of two toolbars (Figure/Object).
StatusBar	Switches on and off display of the status bar.
Parts List...	Opens the Parts List dialog box. A list of all the figure/part objects in this window is displayed.
Grid	Shows/hides the window grid.
Properties	Opens the property dialog box of the selected figure/part object.

(3) [Parts] menu/[Parts] toolbar

This menu and toolbar are used to select connected parts (part objects) provided by Simulator GUI when newly creating or placing parts (see "(3)   Creating part objects").

In this menus, similar operations can be performed using the buttons on the [Parts] toolbar.
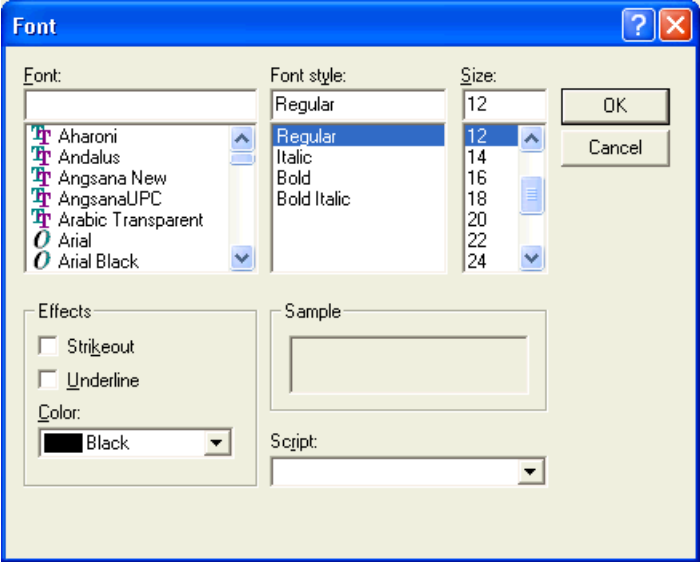
Menu Item	Button	Function
Button		Digital input switch
e.g.) P00		A button can be connected to any pin. A digital input value can be given to the connected pin by clicking the displayed button.
Analog Button		Analog input switches
e.g.) 5V 4V 3V 2V 1V		A button can be connected to any pin. An analog input value can be given to the connected pin by clicking the displayed button.
Key Matrix		A key matrix consists of multiple pins connected in a matrix array, wherein each contact represents a key, and clicking a key results in a specific state.
e.g.)		A key matrix can be connected to any pin, and data can be input using multiple keys.
Level Gauge		Used for inputting analog data such as power supply voltage. Any data within a given range can be set.
e.g.)		Any value within a specified range can be assigned to a pin connected to an A/D converter.
LED		Light Emitting Diode
e.g.)		An LED can be connected to any pin, and the output from the pin can be indicated by switching the LED on or off.
7-Segment LED		A product that consists of 7 LEDs configured to represent a numeric figure.
e.g.)		When the output from the pin assigned to the digit signal is active, the corresponding 7-segment LED switches on or off.
14-Segment LED		A product that consists of 14 LEDs configured to represent an alphabetic character.
e.g.)		When the output from the pin assigned to the digit signal is active, the corresponding 14-segment LED switches on or off.
Matrix LED		A product that consists of multiple LEDs arranged in a matrix array.
e.g.)		When the output from an assigned pin is active, the corresponding 14-segment LED switches on or off.
Buzzer		A buzzer connected to a pin indicates the output information from the connected pin with a bitmap.
e.g.)		
Pull up / Pull down...		Opens the Pull up/Pull down dialog box.
		Whether a pin is connected to a pull-up/down resistor can be specified via this dialog box.

(4) [Figure] menu/[Figure] toolbar

This menu and toolbar are used to set the operation mode of this window, and select connected parts (figure objects) when newly creating or placing parts (see "(2)   Creating figure objects").

In this menus, similar operations can be performed using the buttons on the [Figure] toolbar.

Menu Item	Button	Function
Select		Changes this window's operation mode to the Edit mode. The cursor shape changes into an arrow, enabling the edit of objects.
Simulation Mode		Changes this window's operation mode to the Simulation mode. The cursor shape changes into a hand, enabling simulation of input to connected parts (part objects).
Line		Changes the cursor shape into a cross (+), enabling the drawing of lines.
Rectangle		Changes the cursor shape into a cross (+), enabling the drawing of rectangles.
Rounded Rectangle		Changes the cursor shape into a cross (+), enabling the drawing of rectangles with rounded corners.
Ellipse		Changes the cursor shape into a cross (+), enabling the drawing of ellipses.
Polygon		Changes the cursor shape into a cross (+), enabling the drawing of polygons.
Fan-shaped		Changes the cursor shape into a cross (+), enabling the drawing of fan shapes.
Text		Changes the cursor shape into a cross (+), enabling the drawing of text.
Paste Bitmap...	-	Pastes the selected bitmap file in this window
Color of Line...		Opens the Color dialog box below. The color of the lines of the selected object can be changed to the selected color in this dialog box.
Color of Fill...		Opens the Color dialog box. The color used for the selected object can be changed to the selected color in this dialog box.

Menu Item	Button	Function
Font...	-	Opens the Font dialog box below The font of the selected object can be changed to the selected font in this dialog box.
Style of Line	≡	Selects the line style from a cascade menu. The lines of the selected object are changed.
16pt		Sets the line thickness to 16 pt.
12pt		Sets the line thickness to 12 pt.
8pt		Sets the line thickness to 8 pt.
4pt		Sets the line thickness to 4 pt.
2pt		Sets the line thickness to 2 pt.
1pt		Sets the line thickness to 1 pt.
No line		Does not draw lines.
Dotted Line Style		Selects the dotted line style from a cascade menu. The lines of the selected object are changed.
Solid Line		Draw solid lines.
Dash Line		Draw dashed lines.
Dotted Line		Draw dotted lines.
DashDot Line		Draw DashDot lines.
DashDotDot Line		Draw DashDotDot lines.

[Context menu]

In the Edit mode, the following context menus are available.

Copy	Copies the selected objects.
Paste	Pastes the contents of the clipboard.
Delete	Deletes the selected objects.
Group	Selects from a cascade menu.
Group	Groups the selected objects.
Ungroup	Ungroups the selected objects.
Order	Selects from a cascade menu.
Bring to Front	Brings the selected object to the front.
Send to Back	Sends the selected object to the back.
Bring Forward	Brings the selected object one panel forward.
Send Backward	Sends the selected object one panel backward.
Properties	Opens the property dialog box of the selected figure/part object.

[Operation]

How to create objects (figure objects and part objects) to construct a dummy target system, and manipulate them is as follows:

- (1) Edit mode
- (2) Creating figure objects
- (3) Creating part objects
- (4) Placing objects
- (5) Inputting text
- (6) List display of objects
- (7) The detailed setting for objects
- (8) Simulation mode

(1) Edit mode

Change the operation mode of this window to "edit mode" to create objects.

The edit mode can be set by any one of the following methods.

- Select the [Figure] menu >> [Select].
- Click the ⬚ button on the toolbar.
- Select the [Edit] menu >> [Select All].

(2) Creating figure objects

(a) Line

Select the [Figure] menu >> [Line] or click the ⬚ button on the toolbar.

-> The mouse cursor changes to a cross (+) shape and line drawing becomes possible.

Drag the cursor from the line start position to the end position.

-> The line start position and the line end position are connected by a straight line. (The line thickness and shape are the default settings.)

(b) Rectangle/Rounded rectangle/Fan shape

Select the [Figure] menu >> [Rectangle]/[Rounded Rectangle]/[Ellipse]/[Fan-shaped] or click the ▢ / ◯ /
◯ / ◇ button on the toolbar.

    -> The mouse cursor changes to a cross (+) shape and each drawings becomes possible.

Drag the mouse cursor from the top left corner to the bottom right corner of the drawing area (rectangular area).

    -> The corresponding figure is displayed in the drawing area, with the mouse position forming the lower
right corner.

Drop the figure to fix its size.

    -> Rectangles are drawn in the same size as the rectangular area, and other figure shapes are drawn in a
size that fits in the rectangular area. (The line thickness and shape are the default settings.)

(c) Polygon

Select the [Figure] menu >> [Polygon] or click the ◇ button on the toolbar.

    -> The mouse cursor changes to a cross (+) shape and polygon drawing becomes possible.

Click at each position marking an apex of the polygon.

    -> The apexes are linked by a straight line in the order in which they were clicked.

Drawing of the polygon is completed by double-clicking.

    -> The line thickness and shape are the default settings.

(d) Pasting a bitmap

You can use an arbitrary bitmap file as a figure object.

Select the [Figure] menu >> [Paste Bitmap], and then select the bitmap file (*.bmp) to be pasted.

    -> The corresponding bitmap file is pasted in the default position in this window.

(e) Changing the figure object style

The color or line style of the created figure object can be changed by any one of the following methods.

    - Specify with the [Style] tab of the Object Properties dialog box opened by double-clicking the figure object.

    - Select the [Figure] menu >> [Color of Line]/[Color of Fill]/[Style of Line]/[Dotted Line Style] or click the ✏ /
🎨 / ▤ / ▦ button on the toolbar.

(3) Creating part objects

You can create part objects by using connected parts provided by Simulator GUI.

(a) Selecting a part object

Select the part object to be created from the [Parts] menu or the toolbar.

    -> The mouse cursor changes to a cross (+) shape.

Click any location.

    -> The corresponding part object is created and placed with the clicked location as the top left corner
(default size).

(b) Changing the part object style

The style of the created part object can be changed via the [Style] tab of the corresponding property dialog box
opened by double-clicking the part object.

For details on the modifiable items, see the section of the property dialog box which is corresponding to the
part object (the items differ depending on the part object).

(4) Placing objects

(a) Grid display

A grid is displayed by selecting the [View] menu >> [Grid].

(b) Selecting objects

The selected types and methods are indicated below.

The selected object(s) is displayed surrounded by a tracker indicating its selected status.

- Single selection

Click the object to be selected.

- Multiple selections

Click the objects to be selected while pressing the [Shift] key.

- Range selection

Drag from the top left corner of the area including the object to be selected, and drop at the lower right corner.

- Select all

Select [Edit] menu>> [Select All].

(c) Moving objects

After selecting the object to be moved (multiple selections possible), then drag and drop it at the move destination.

Remark　Objects can be moved using the arrow keys on the keyboard.

If more than half of the part bitmap is hidden when the window size is reduced, however, the selected part cannot be moved by using the arrow keys.

Here, the part (14-segment LED in this case) can be moved using the arrow keys.

Here, the part (14-segment LED in this case) cannot be moved using the arrow keys.
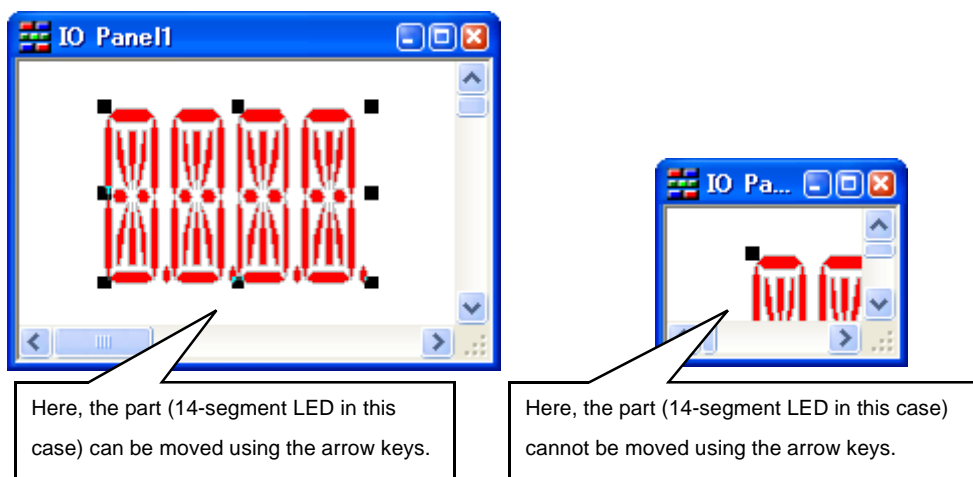
(d) Changing object size

After selecting the object whose size is to be changed, then drag the tracker displayed.

(e) Cut/Copy/Paste/Delete/Group/UnGroup an object

After selecting the object, select the corresponding item from the [Edit] menu.

(f) Changing object sequence (Bring to Front/Send to Back/Bring Forward/Send Backward)

After selecting the object, select the corresponding item from the [Edit] menu.

(5) Inputting text

Select [Figure] menu >> [Text] or click the A button on the toolbar.

-> The mouse cursor changes to a cross (+) shape.

Drag the mouse cursor from the top left corner of the character drawing area (rectangular area) to the lower right corner.

-> This rectangular area serves as the character drawing area.

Click in the character drawing area.

-> The cursor is displayed and character input becomes possible.

(6) List display of objects

Fugure objects and part objects created in this window can be displayed as a list by selecting the [View] menu >> [Parts List], in addition to display in this window.

(7) The detailed setting for objects

Each created object requires the detailed setting (pin connection information, etc.) in accordance with the target system to be used.

(a) Figure objects

The detailed setting can be performed via the [Object Connection] tab of the Object Properties dialog box opened by double-clicking the target figure object.

By connecting a object to an output pin, show/hide of the object can be switched depending on the output status of the connected pin.

(b) Part objects

The detailed setting can be performed via the [*xxx* Connection] tab of the property dialog box opened by double-clicking the target part object.

For details on the setable items, see the section of the property dialog box which is corresponding to the part object (the items differ depending on the part object).

(8) Simulation mode

The part objects whose settings have been completed can be manipulated during simulation (input values can be fed to the simulator). Accordingly you can check the I/O results from these displayed part objects in this window. Change the operation mode of this window to "simulation mode" to manipulate part objects.

The simulation mode can be set by any one of the following methods (the mouse cursor changes to a hand shape).

- Select the [Figure] menu >> [Simulation Mode].

- Click the 🖑 button on the toolbar.

Remark   For details on input operation, see the section of the property dialog box which is corresponding to the part object

---

Parts Button Properties dialog box

This property dialog box is used to set or change the pin connection information of buttons, which are one of the connection parts in the I/O Panel window.

Input to the simulator can be done from pin-connected buttons in the Simulation mode.

There are two types of button display styles, figure and bitmap. These styles can be changed on the [[Style] tab].

Figure A-88.   Parts Button Properties Dialog Box: [Button Connection] Tab
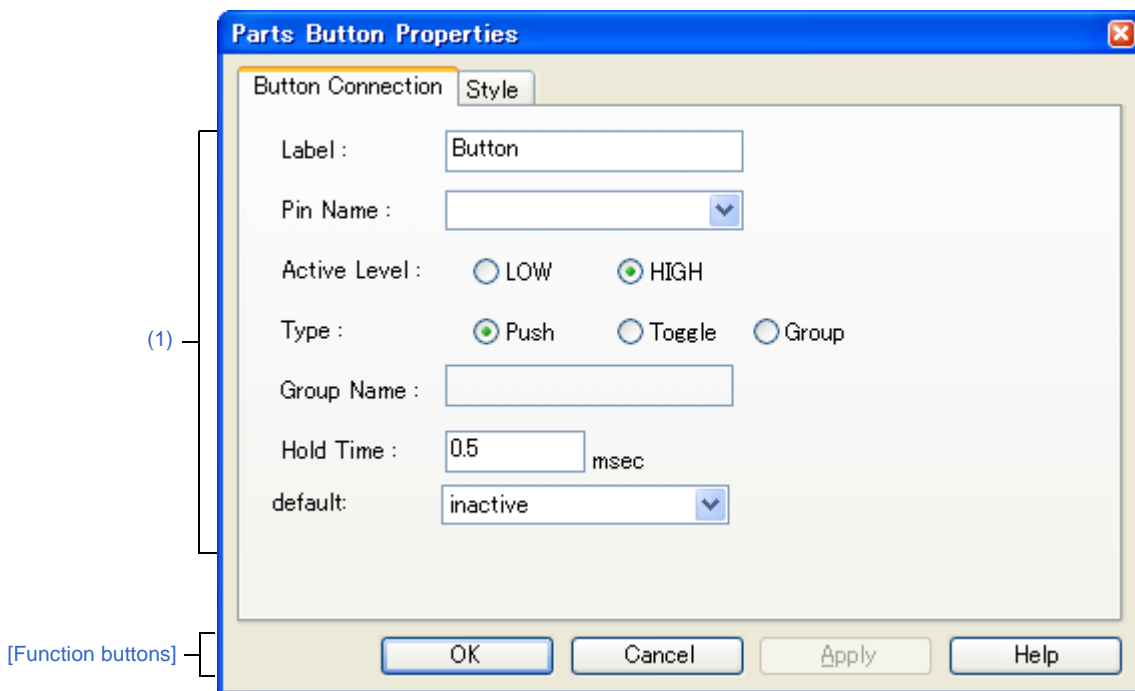
Figure A-89.   Parts Button Properties Dialog Box: [Style] Tab
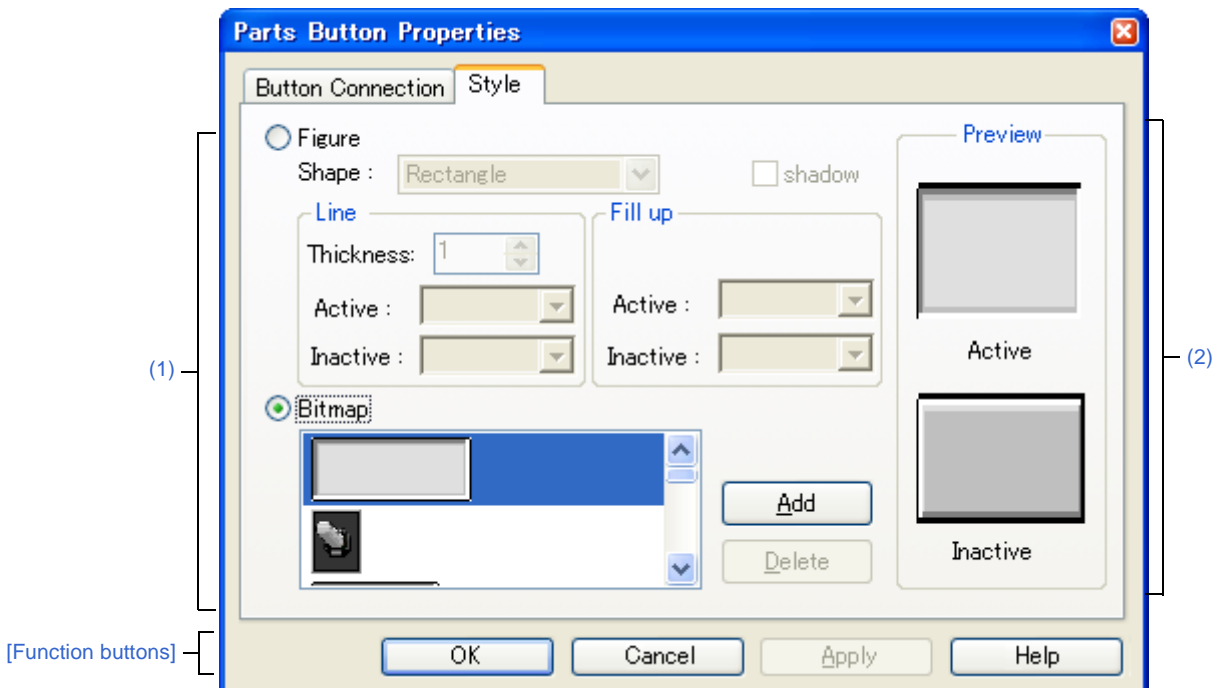
---

This section describes the following.

[How to open]

On the I/O Panel window, any one of the following:
- Double-click a part object "Button".
- Select [Properties...] form the context menu on a part object "Button".
- Select a part object "Button", and then select [Properties...] form the [View] menu.

[[Button Connection] tab]
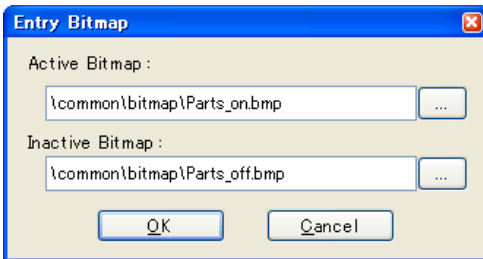
(1) Pin connection information setting area

Label	This area is used to specify the part name. The part name input here is displayed on the button. Moreover, it is also displayed in the Parts List dialog box as the label.	
Pin Name	This area is used to specify the pin name to be connected. The connection pins can be specified either via direct input or through selection from the drop-down list.	
Active Level	The active state is selected with a option button, as follows:	
	LOW	Sets the active level to LOW.
	HIGH	Sets the active level to HIGH (default).
Type	Button types are selected using option buttons, as follows:	
	Push	Makes the button a Push button (default). The [Hold Time] item must be specified.
	Toggle	Makes the button a Toggle button.
	Group	Makes the button a Group button. The [Group Name] item must be specified.
Group Name	This area is used to input the button's group name. Input to this area is possible only when [Group] has been selected in [Type].	
Hold Time	Specify the time (hold time) for which the input value is to be held (default: 0.5msec). The settable range is from 0.001 to 999 (msec). Input to this area is possible only when [Push] has been selected in [Type].	
default	Specify the button status after CPU reset.	
	Not change at reset.	Maintains the button status after CPU reset.
	inactive	Buttons are non-depressed after CPU reset (default).
	active	Buttons are depressed after CPU reset.

Remark　For the pin names that can be specified, see the user's manual of the microcontroller that is used.

[[Style] tab]

(1) Style information setting area

Figure	Select this option button to display the button with the following specified figure.		
	Shape	Select the figure shape. Two shapes can be selected: rectangle and ellipse.	
	shadow	Cannot be selected.	
	Line	Specify and change the figure line, as follows. You can change the color of figure line by clicking the pull-down button.	
		Thickness	Specifies the line thickness. Specification is made either using a spin button or through direct input. A value from 1 to 100 can be specified.
		Active	Specifies the color of the line during active display.
		Inactive	Specifies the color of the line during inactive display.
	Fill up	Specify and change the figure filling, as follows. You can change the color of figure filling by clicking the pull-down button.	
		Active	Specifies the fill color during active display.
		Inactive	Specifies the fill color during inactive display.
Bitmap	Select this option button to display the button with the following specified bitmap (default).		
	Selection list	Select a bitmap to be used from the selection list. The selectable bitmaps appear in the selection list.	
	[Add] button	Opens the Entry Bitmap dialog box below to add a new bitmap to the selection list. The bitmap file to be added can be specified either through file selection using the [...] button, or through direct input.	
	[Delete] button	Deletes the currently selected bitmap from the selection list. Note that only the bitmap that have been added by user can be deleted.	

(2) Preview area

This area displays the style of the button currently being specified.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

[Operation]

In the Simulation mode, by clicking the displayed button, data can be input to the pin connected to the button.

The input format differs depending on the button type (push, toggle, group).

Figure A-90.   Connected Parts Display Example (Button)

Push button	Clicking this button causes the active value to be loaded to the connected pin.
	The active value is held during the hold time, after which it changes to the original value.
Toggle button	Clicking this button loads the active value to the connected pin.
	The active value is held during the hold time, after which it changes to the original value.
Group button	Clicking this button loads the active value to the connected pin.
	The value of the Group button having the same group name returns to the original value.

Analog Button Properties dialog box

This property dialog box is used to set or change the pin connection information of analog buttons, which are one of the connection parts in the I/O Panel window.

Input to the simulator can be done from pin-connected analog buttons in the Simulation mode.

There are two types of analog button display styles, figure and bitmap. These styles can be changed on the [[Style] tab].

Figure A-91.   Analog Button Properties Dialog Box: [Analog Button Connection] Tab
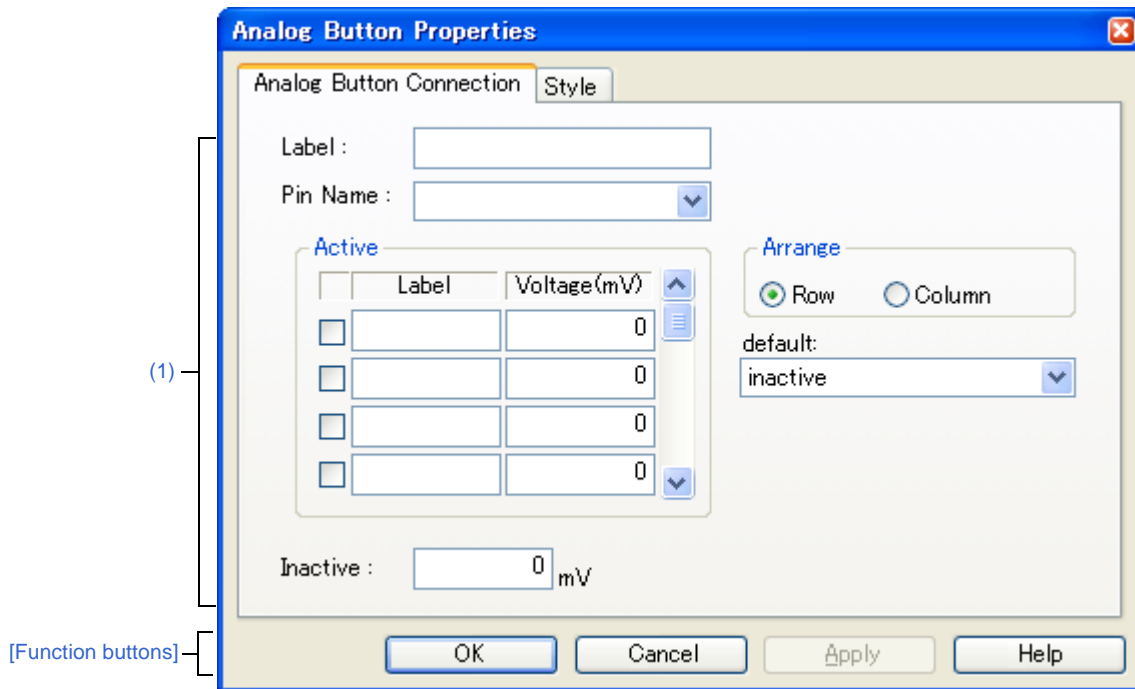
Figure A-92.   Analog Button Properties Dialog Box: [Style] Tab
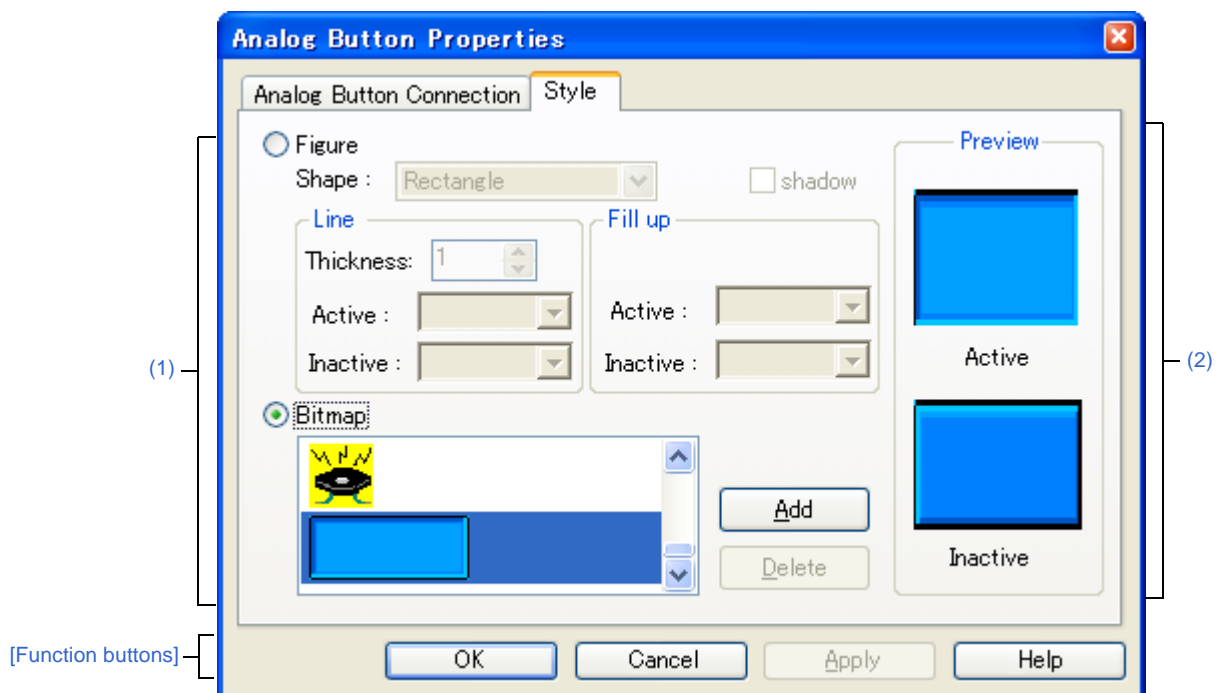
This section describes the following.

[How to open]

On the I/O Panel window, any one of the following:
- Double-click a part object "Analog Button".
- Select [Properties...] form the context menu on a part object "Analog Button".
- Select a part object "Analog Button", and then select [Properties...] form the [View] menu.

[[Analog Button Connection] tab]
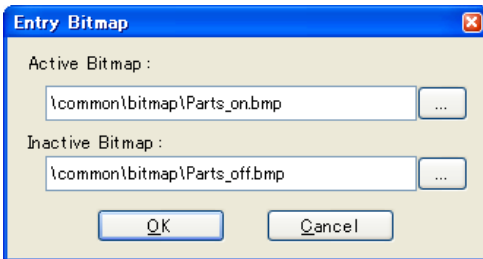
(1) Pin connection information setting area

Label	This area is used to specify the part name. The part name input here is displayed on the button. Moreover, it is also displayed in the Parts List dialog box as the label.	
Pin Name	This area is used to specify the pin name to be connected. The connection pins can be specified either via direct input or through selection from the drop-down list.	
Active	Specify the active state.	
	Check box	Analog buttons are created by the number of selected check boxes.
	Label	Directly input the name to be displayed on each analog button.
	Voltage(mV)	Directly input the voltage to be input when each analog button is clicked, in mV units.
Arrange	Specify the button arrangement using the option button. This setting is available if two or more analog buttons have been created in the Active area. This setting is ignored if there is only one analog button or no analog buttons have been created.	
	Row	Analog buttons will be arranged horizontally (default).
	Column	Analog buttons will be arranged vertically.
default	Maintains the analog button operation after CPU reset.	
	Not change at reset.	Maintains the analog button status immediately before CPU reset, after CPU reset.
	inactive	All of the analog buttons are non-depressed after CPU reset (default).
	Active is '*xxx*'*yyy*(mV)	The analog button specified with "'*xxx*'(*yyy*mV) " is depressed after CPU reset.
Inactive	Specify the input level when no analog buttons are depressed.	

Remark   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

[[Style] tab]

(1) Style information setting area

Figure	Select this option button to display the analog button with the following specified figure.		
	Shape	Select the figure shape. Two shapes can be selected: rectangle and ellipse.	
	shadow	Cannot be selected.	
	Line	Specify and change the figure line, as follows. You can change the color of figure line by clicking the pull-down button.	
		Thickness	Specifies the line thickness. Specification is made either using a spin button or through direct input. A value from 1 to 100 can be specified.
		Active	Specifies the color of the line during active display.
		Inactive	Specifies the color of the line during inactive display.
	Fill up	Specify and change the figure filling, as follows. You can change the color of figure filling by clicking the pull-down button.	
		Active	Specifies the fill color during active display.
		Inactive	Specifies the fill color during inactive display.
Bitmap	Select this option button to display the analog button with the following specified bitmap (default).		
	Selection list	Select a bitmap to be used from the selection list. The selectable bitmaps appear in the selection list.	
	[Add] button	Opens the Entry Bitmap dialog box below to add a new bitmap to the selection list. The bitmap file to be added can be specified either through file selection using the [...] button, or through direct input. Entry Bitmap Active Bitmap : \common\bitmap\Parts_on.bmp [...] Inactive Bitmap : \common\bitmap\Parts_off.bmp [...] OK Cancel	
	[Delete] button	Deletes the currently selected bitmap from the selection list. Note that only the bitmap that have been added by user can be deleted.	

(2) Preview area

This area displays the style of the analog button currently being specified.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

[Operation]

In the Simulation mode, while an analog button is depressed, the specified analog voltage value is input to the specified pin.

Clicking the depressed analog button again restores the button to its original state.

Figure A-93.   Connected Parts Display Example (Analog Button)
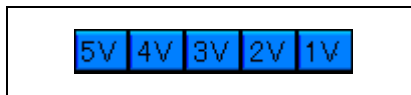
Parts Key Properties dialog box

This property dialog box is used to set or change the pin connection information of a key matrix, which is one of the connection parts in the I/O Panel window.

Input to the simulator can be done from pin-connected keys in the Simulation mode.

A key matrix consisting of input pins and output pins of up to 16 x 16 can be set.

Either figure or bitmap can be selected as the key matrix display style. These styles can be changed on the [[Style] tab].

Caution   **When connecting a key matrix to pins, also perform the Pull up/Pull down settings for the connection pins. When a key is pressed, the output value of the output pin connected to that key is input to the input pin connected to that key. The value when the key is not pressed is the value specified in the Pull up/Pull down dialog box.**
If the Pull up/Pull down settings are not performed, the input pin becomes he high-impedance state. Consequently, the operation of the function that is connected to the input pin becomes undefined.

Figure A-94.   Parts Key Properties Dialog Box: [Key Matrix Connection] Tab
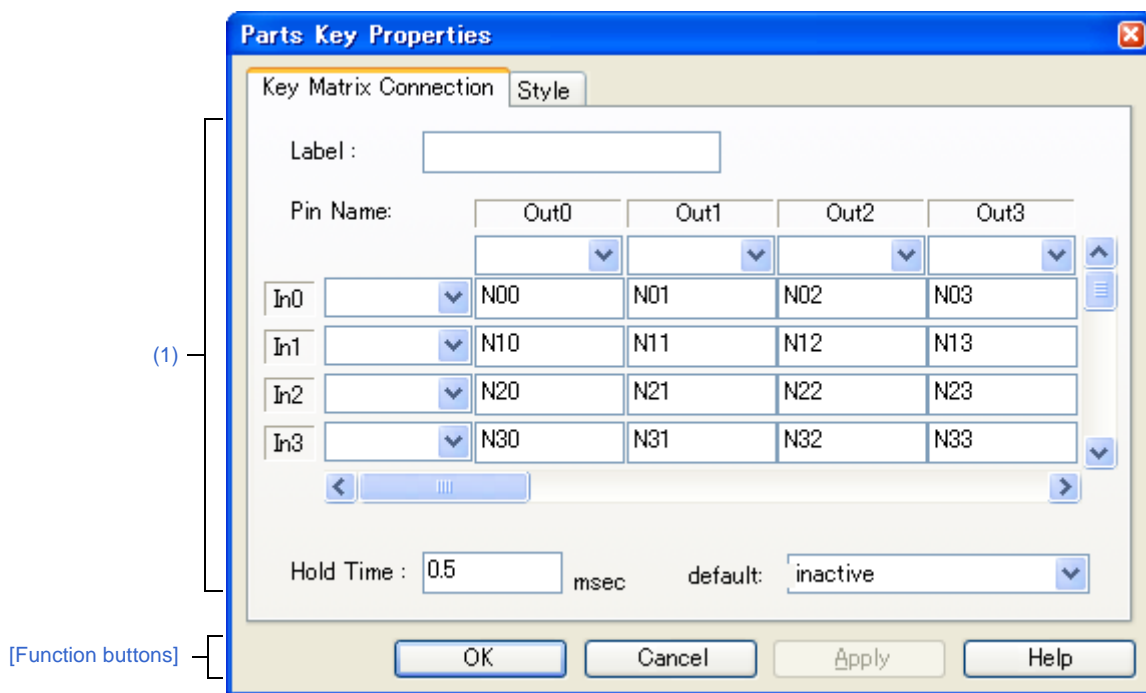
Figure A-95.   Parts Key Properties Dialog Box: [Style] Tab
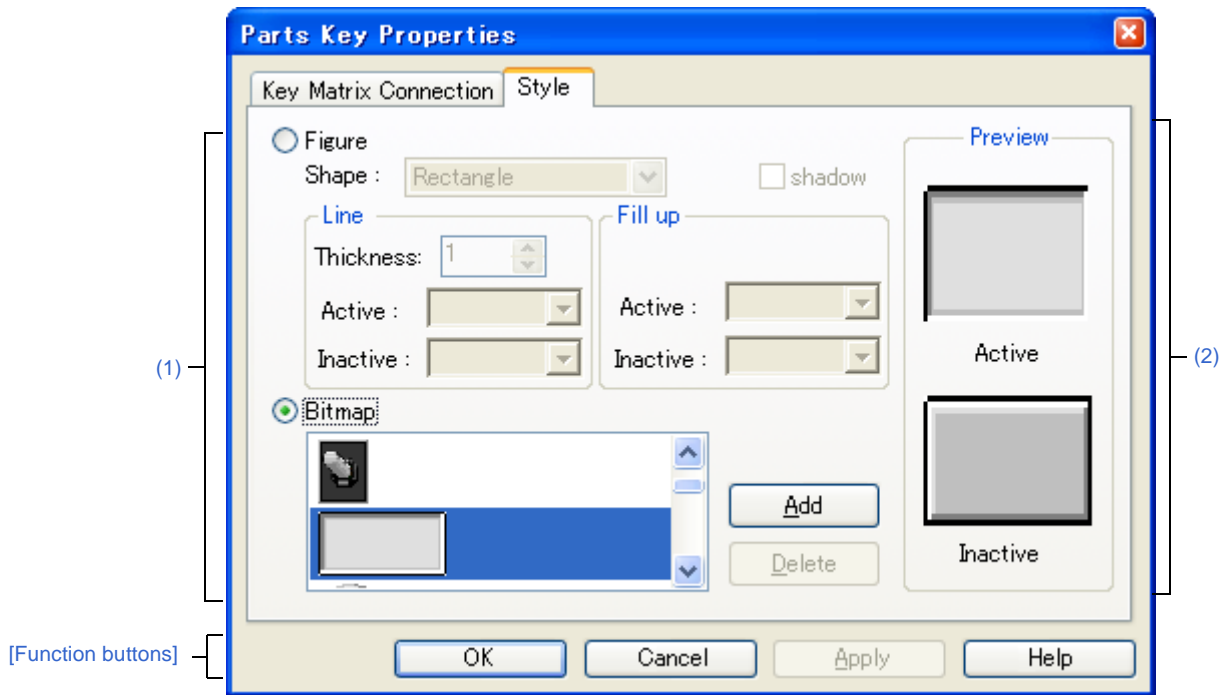
This section describes the following.

- [How to open]
- [[Key Matrix Connection] tab]
- [[Style] tab]
- [Function buttons]
- [Operation]

[How to open]

On the I/O Panel window, any one of the following:
- Double-click a part object "Key matrix".
- Select [Properties...] form the context menu on a part object "Key matrix".
- Select a part object "Key matrix", and then select [Properties...] form the [View] menu.

[[Key Matrix Connection] tab]

(1) Pin connection information setting area

Label	This area is used to specify the part name. The part name input here is also displayed in the Parts List dialog box as the label.

Pin Name	This area is used to specify the pin name to be connected (input pins and output pins). The connection pins can be specified either via direct input or through selection from the drop-down list. This area can be used to set 16 x 16 pins using the scrollbar.	
	In0 - In15	Specify input pins.
	Out0 - Out15	Specify output pins.
	N00 - Nff	This area is where the text strings displayed on the keys of the key matrix are specified. Text strings of any length can be specified. The default description string (N number) is not displayed on the keys.
Hold Time	Specify the time (hold time) for which the input value is to be held (default: 0.5msec). The settable range is from 0.001 to 999 (msec). Moreover, when multiple keys for input to the same input pin are pressed during the hold time, only the key that was clicked last is valid.	
default	Specify the key matrix operation after CPU reset.	
	Not change at reset.	The key matrix status does not change after CPU reset.
	inactive	No key matrix buttons are depressed after CPU reset (default).

Remark   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

[[Style] tab]

(1) Style information setting area

Figure	Select this option button to display the key matrix with the following specified figure.	
	Shape	Select the figure shape. Two shapes can be selected: rectangle and ellipse.
	shadow	Cannot be selected.
	Line	Specify and change the figure line, as follows. You can change the color of figure line by clicking the pull-down button.
		Thickness	Specifies the line thickness. Specification is made either using a spin button or through direct input. A value from 1 to 100 can be specified.
		Active	Specifies the color of the line during active display.
		Inactive	Specifies the color of the line during inactive display.
	Fill up	Specify and change the figure filling, as follows. You can change the color of figure filling by clicking the pull-down button.
		Active	Specifies the fill color during active display.
		Inactive	Specifies the fill color during inactive display.

Bitmap	Select this option button to display the key matrix with the following specified bitmap (default).	
	Selection list	Select a bitmap to be used from the selection list. The selectable bitmaps appear in the selection list.
	[Add] button	Opens the Entry Bitmap dialog box below to add a new bitmap to the selection list. The bitmap file to be added can be specified either through file selection using the [...] button, or through direct input.
	[Delete] button	Deletes the currently selected bitmap from the selection list. Note that only the bitmap that have been added by user can be deleted.

(2) Preview area

This area displays the style of the key matrix currently being specified.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

[Operation]

In the Simulation mode, the following operation can be done.
(1) Inputting multiple keys simultaneously
(2) Locking the key input value

(1) Inputting multiple keys simultaneously

To enter two keys, simultaneously press the key to be input and right-click the mouse to enter the wait status. Then, click the other key. This releases the wait status and enables simultaneous input of both keys. Multiple keys can be simultaneously input by setting the wait status for multiple keys, but if input is to be performed to the same input pin, the key that was input last is valid.

(2) Locking the key input value

To enter two keys, simultaneously press the key to be input and right-click the mouse to enter the wait status. Then, click the other key. This releases the wait status and enables simultaneous input of both keys. Multiple keys can be simultaneously input by setting the wait status for multiple keys, but if input is to be performed to the same input pin, the key that was input last is valid

Figure A-96.   Connected Parts Display Example (Key Matrix)

Parts Level Gauge Properties dialog box

This property dialog box is used to set or change the pin connection information of level gauge, which are one of the connection parts in the I/O Panel window.

Input to the simulator can be done from pin-connected level gauge in the Simulation mode.

Note that the connection pin must be an analog input pin.

There are two types of level gauge display styles, slide and dial. These styles can be changed on the [[Style] tab].

Figure A-97.   Parts Level Gauge Properties Dialog Box: [Level Gauge Connection] Tab
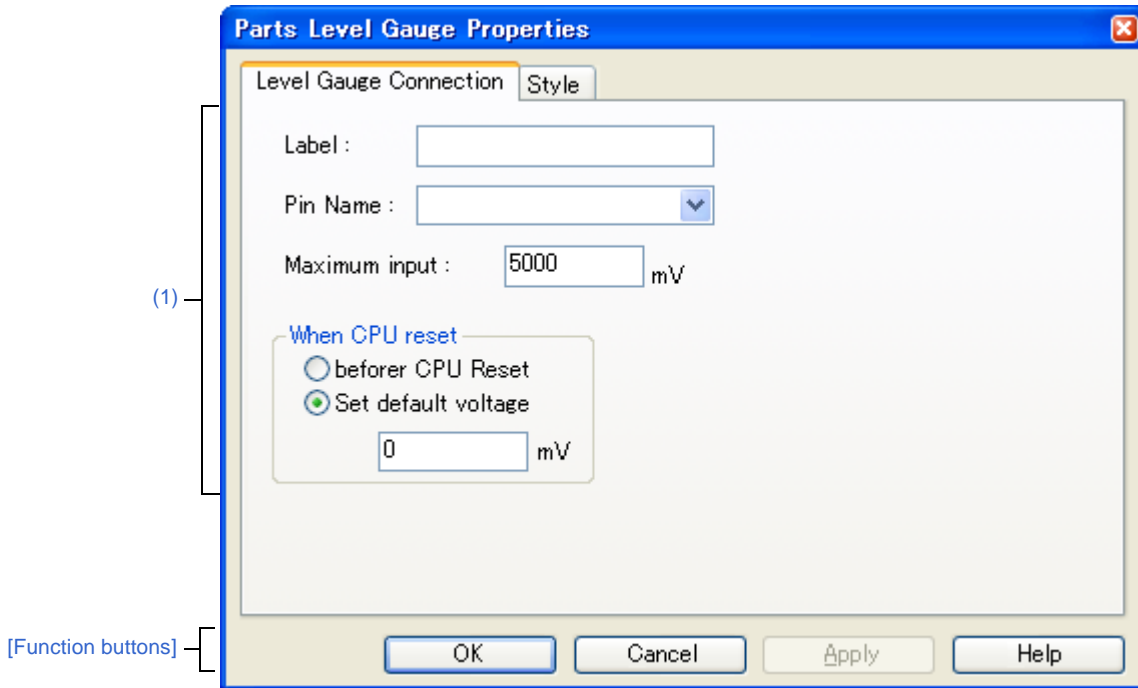
Figure A-98.   Parts Level Gauge Properties Dialog Box: [Style] Tab
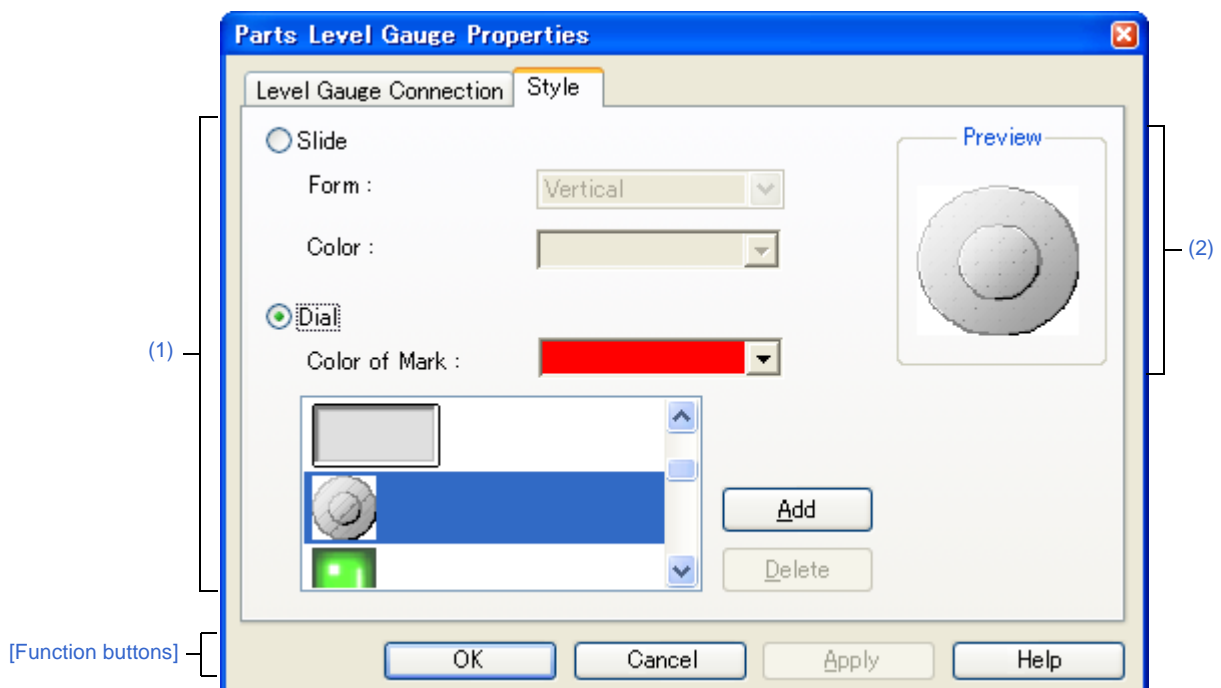
This section describes the following.

- [How to open]
- [Level Gauge Connection] tab]
- [[Style] tab]
- [Function buttons]
- [Operation]

[How to open]

On the I/O Panel window, any one of the following:
- Double-click a part object "Level Gauge".
- Select [Properties...] form the context menu on a part object "Level Gauge".
- Select a part object "Level Gauge", and then select [Properties...] form the [View] menu.

[Level Gauge Connection] tab]

(1) Pin connection information setting area
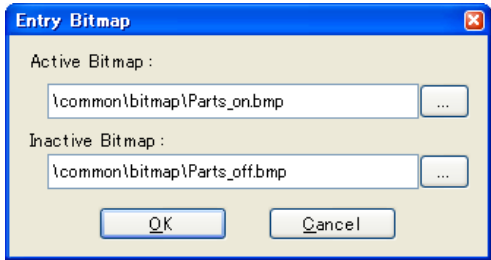
| Label	This area is used to specify the part name. The part name input here is also displayed in the Parts List dialog box as the label.	
Pin Name	This area is used to specify the pin name to be connected. The connection pins can be specified either via direct input or through selection from the drop-down list.	
Maximum input	This area is used to set the maximum level gauge input value (default: 5000mV). The settable range is from 0 to 65535. The operation range of the level gauge displayed in the I/O Panel window is determined by this specified value.	
When CPU reset	Specify the level gauge operation after CPU reset.	
	before CPU Reset	Maintains the level gauge status of immediately before CPU reset, after CPU reset.
	Set default voltage	The level gauge is set to the specified value after CPU reset (default). Specify the value in mV units. Input a value from 0 to "Maximum input:". (default: 0 mV)

Remark   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

[[Style] tab]

(1) Style information setting area

Slide	Select this option button to display the level gauge with the side-type.	
	Form	Select the slide direction (vertical or horizontal) from the drop-down list.
	Color	This area is used to specify or change the slide color. You can change the color by clicking the pull-down button.

Dial	Select this option button to display the level gauge with the dial-type (default).	
	Color of Mark	This area is used to specify or change the color of the mark indicating the operating point. You can change the color by clicking the pull-down button.
	Selection list	Select a bitmap to be used from the selection list. The selectable bitmaps appear in the selection list.
	[Add] button	Opens the Entry Bitmap dialog box below to add a new bitmap to the selection list. The bitmap file to be added can be specified either through file selection using the [...] button, or through direct input. **Entry Bitmap** Active Bitmap : \common\bitmap\Parts_on.bmp [...] Inactive Bitmap : \common\bitmap\Parts_off.bmp [...] OK Cancel
	[Delete] button	Deletes the currently selected bitmap from the selection list. Note that only the bitmap that have been added by user can be deleted.

(2) Preview area

This area displays the style of the level gauge currently being specified.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

[Operation]

In the Simulation mode, analog input from the level gauge is performed through manipulation of the displayed slider or dial.

The value specified in [Maximum input] is the maximum value that can be input.

Input is enabled by selecting the [Figure] menu >> [Simulation Mode].

Figure A-99.   Connected Parts Display Example (Level Gauge)

Dial-type level gauge	The displayed analog value changes as the operating point (red circle) on the dial is moved by dragging it with the mouse. When this analog value has become the value that is to be input, release the operating point. As a result, the displayed analog value is input. The operating point can also be moved by clicking the desired location on the dial.
Slide-type level gauge	The displayed analog value changes as the slider button is moved by dragging it with the mouse. When this analog value has become the value that is to be input, release the button. As a result, the displayed analog value is input. The button can also be moved by clicking the desired location on the slider.

Caution   **If you drag and drop the dial's operation point (red circle) or the slider's knob away from the level gauge, then the voltage displayed in the level gauge will change, but the voltage actually output by the level gauge will not change. Make sure to always drag and drop over the level gauge.**

Parts Led Properties dialog box

This property dialog box is used to set or change the pin connection information of LED, which are one of the connection parts in the I/O Panel window.

A pin-connected LED displays the information output from the simulator through lit/unlit display in the Simulation mode.

There are two types of LED display styles, figure and bitmap. These styles can be changed on the [[Style] tab].

Figure A-100.   Parts Led Properties Dialog Box: [LED Connection] Tab
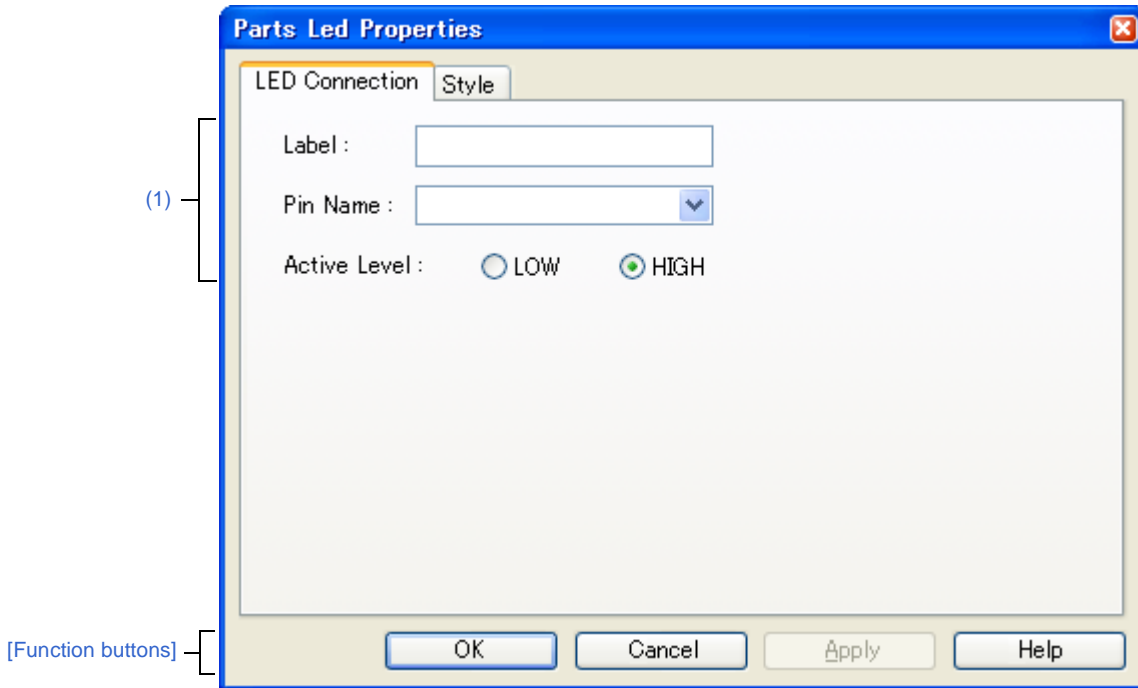
Figure A-101.   Parts Led Properties Dialog Box: [Style] Tab
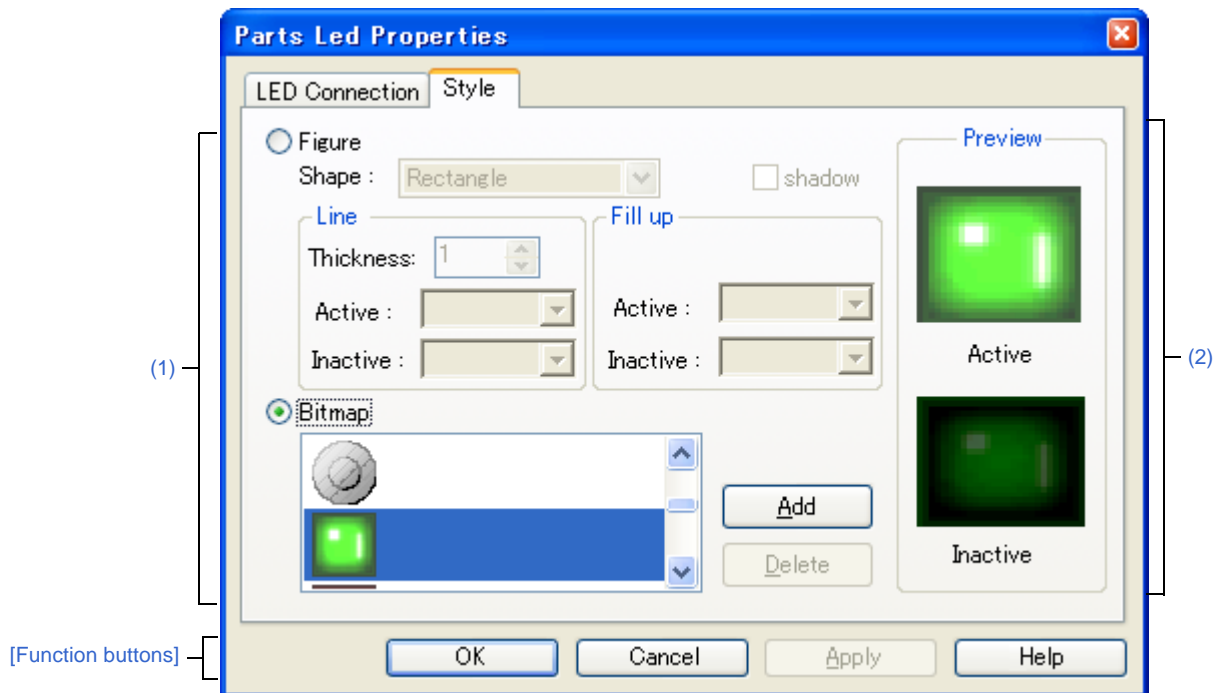
This section describes the following.

[How to open]

On the I/O Panel window, any one of the following:
- Double-click a part object "LED".
- Select [Properties...] form the context menu on a part object "LED".
- Select a part object "LED", and then select [Properties...] form the [View] menu.

[[LED Connection] tab]

(1)　Pin connection information setting area

Label	This area is used to specify the part name. The part name input here is also displayed in the Parts List dialog box as the label.	
Pin Name	This area is used to specify the pin name to be connected (output pin). The connection pins can be specified either via direct input or through selection from the drop-down list.	
Active Level	The active state is selected with a option button, as follows:	
	LOW	Sets the active level to LOW.
	HIGH	Sets the active level to HIGH (default).

Remark　For the pin names that can be specified, see the user's manual of the microcontroller that is used.

[[Style] tab]

(1)　Style information setting area

Figure	Select this option button to display the LED with the following specified figure.		
	Shape	Select the figure shape. Two shapes can be selected: rectangle and ellipse.	
	shadow	Cannot be selected.	
	Line	Specify and change the figure line, as follows. You can change the color of figure line by clicking the pull-down button.	
		Thickness	Specifies the line thickness. Specification is made either using a spin button or through direct input. A value from 1 to 100 can be specified.
		Active	Specifies the color of the line during active display.
		Inactive	Specifies the color of the line during inactive display.
	Fill up	Specify and change the figure filling, as follows. You can change the color of figure filling by clicking the pull-down button.	
		Active	Specifies the fill color during active display.
		Inactive	Specifies the fill color during inactive display.

Bitmap	Select this option button to display the LED with the following specified bitmap (default).	
	Selection list	Select a bitmap to be used from the selection list. The selectable bitmaps appear in the selection list.
	[Add] button	Opens the Entry Bitmap dialog box below to add a new bitmap to the selection list. The bitmap file to be added can be specified either through file selection using the [...] button, or through direct input.
	[Delete] button	Deletes the currently selected bitmap from the selection list. Note that only the bitmap that have been added by user can be deleted.

(2) Preview area

This area displays the style of the LED currently being specified.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

[Operation]

In the Simulation mode, the output status (active/inactive) of the connected pins is displayed in real-time using two types of bitmaps or figures.

Figure A-102.   Connected Parts Display Example (LED)

Parts Segment LED Properties dialog box

This property dialog box is used to set or change the pin connection information of 7-segment LED and 14-segment LED, which are one of the connection parts in the I/O Panel window.

A LED connected to pins displays the information output from the simulator in the Simulation mode.

The segment LED display styles can be changed on the [[Style] tab].

Figure A-103.   Parts Segment LED Properties Dialog Box: [SegmentLED Connection] Tab
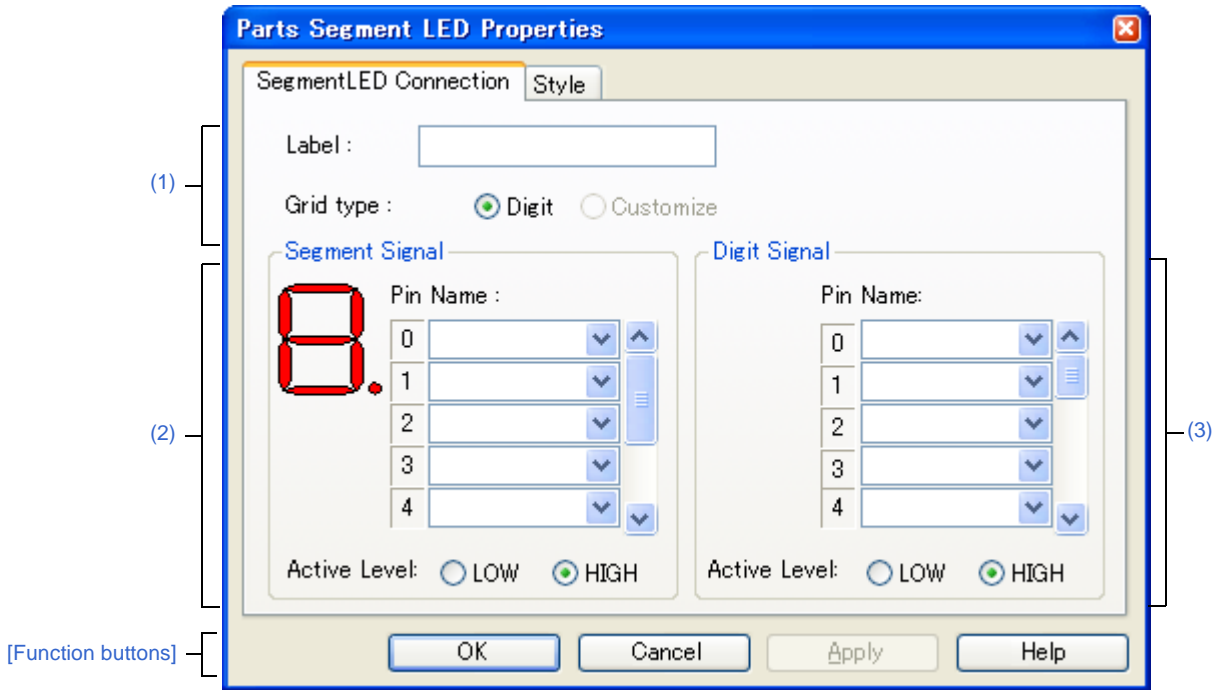
Figure A-104.   Parts Segment LED Properties Dialog Box: [Style] Tab
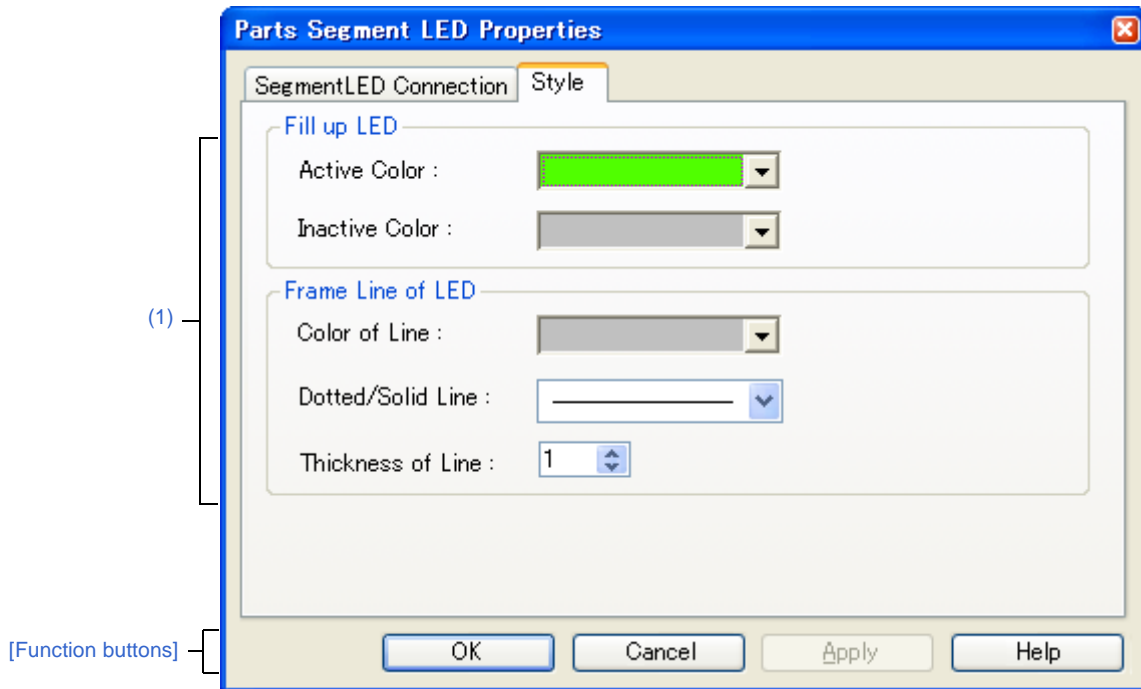
---

This section describes the following.

[How to open]

On the I/O Panel window, any one of the following:
- Double-click a part object "7-segment LED"/"14-segment LED".
- Select [Properties...] form the context menu on a part object "7-segment LED"/"14-segment LED".
- Select a part object "7-segment LED"/"14-segment LED", and then select [Properties...] form the [View] menu.

[[SegmentLED Connection] tab]

(1) Pin connection information setting area

| Label		This area is used to specify the part name. The part name input here is also displayed in the Parts List dialog box as the label.
Grid type		Select the grid signal allocation method from the following. As a result of the selection, the setting in the Digit signal setting area changes.
	Digit	Connects each segment LED to 1 grid pin. Digit signal settings are performed in the Digit signal setting area (default). 16 digits can be specified for the digit signal. A segment LED of up to 16 digits can be created with 1 segment LED part.
	Customize	Cannot be selected.

(2) Segment signal setting area

Segment Signal		This area is used to specify the pins (output pins) to be connected to the segment signals of the 7-segment LED/14-segment LED, as well as their active level.
	Figure	This area displays the bitmap of the 7-segment LED/14-segment LED at the top left. When "Pin Name:" is input, the corresponding location is indicated.
	Pin Name	The connection pins can be specified either via direct input or through selection from the drop-down list. The number of segment pins to be connected is 8 in the case of a 7-segment LED, and 15 in the case of a 14-segment LED. Connection to all the segment pins is possible by using the scroll bar on the right side.
	Active Level	The active status can be selected with a option button, as follows:
		LOW	Sets the active level to LOW.
		HIGH	Sets the active level to HIGH (default).

Remark   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

(3) Digit signal setting area

Digit Signal	This area is used to specify 7-segment LED/14-segment LED digits, grid signal connection pins (output pins), and their active level. The connection method changes as follows according to what is specified for [Grid type].
	- When [Digit] selected
	  Perform digit signal setting. The maximum number of digit pins that can be connected is 16. Connection to all the digit pins can be done by using the scroll bar on the right side of the area.
	- When [Customize] selected
	  Cannot be selected.		

	Pin Name	The connection pins can be specified either via direct input or through selection from the drop-down list. As the setting signal, specify the connection pins in a series from the lowermost digit.	
	Active Level	The active status can be selected with a option button, as follows:	
		LOW	Sets the active level to LOW.
		HIGH	Sets the active level to HIGH (default).

> **Remark**   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

[[Style] tab]

(1) Style information setting area

Full up LED	This area is used to set and change related to the filling of each cell of the object are performed. You can change the color by clicking the pull-down button.	
	Active Color	Specifies the fill color during active display.
	Inactive Color	Specifies the fill color during inactive display.
Frame Line of LED	This area is used to set and change related to the frame of each cell of the object are performed.	
	Color of Line	Specifies and changes the line color. You can change the color by clicking the pull-down button.
	Dotted/Solid Line	Specifies and changes the line shape (dotted/solid). The desired line shape can be selected from the drop-down list. The line shape can be specified only when the line thickness is "1" in [Thickness of Line].
	Thickness of Line	Specifies and changes the line thickness. The desired line thickness can be specified either via direct input or through selection from the spin button.A value in the range of 1 to 100 (decimal) can be specified.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

[Operation]

In the Simulation mode, as the results of 1 simulation, the output information of the connection pins is received and display is performed accordingly. When both the digit/grid signals and segment signals are active output, the segment LED of the corresponding digit/grid light.

Figure A-105.   Connected Parts Display Example (7-segment LED)
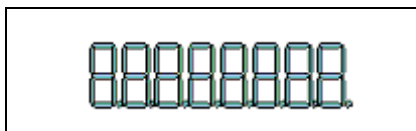
Parts Matrix Led Properties dialog box

This property dialog box is used to set or change the pin connection information of matrix LED, which are one of the connection parts in the I/O Panel window.

A pin-connected matrix LED displays the information output from the simulator through lit/unlit display in the Simulation mode.

There are two types of matrix LED display styles, figure and bitmap. These styles can be changed on the [[Style] tab].

Figure A-106.   Parts Matrix Led Properties Dialog Box: [Matrix LED Connection] Tab
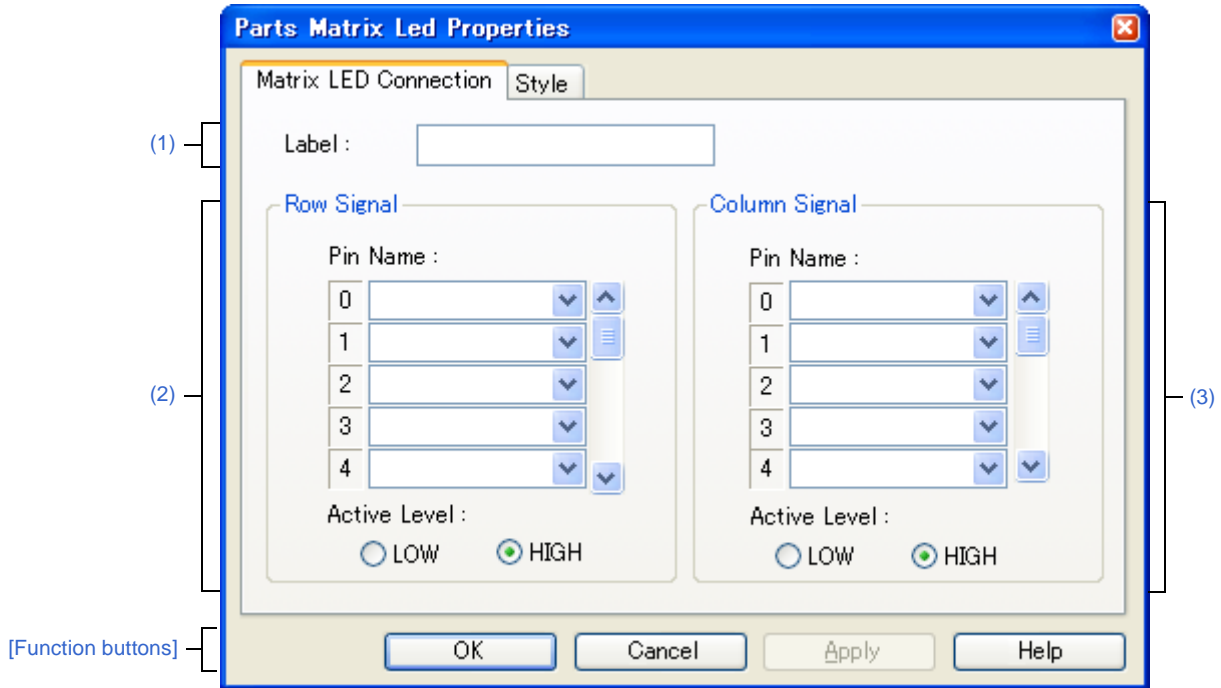
Figure A-107.   Parts Matrix Led Properties Dialog Box: [Style] Tab
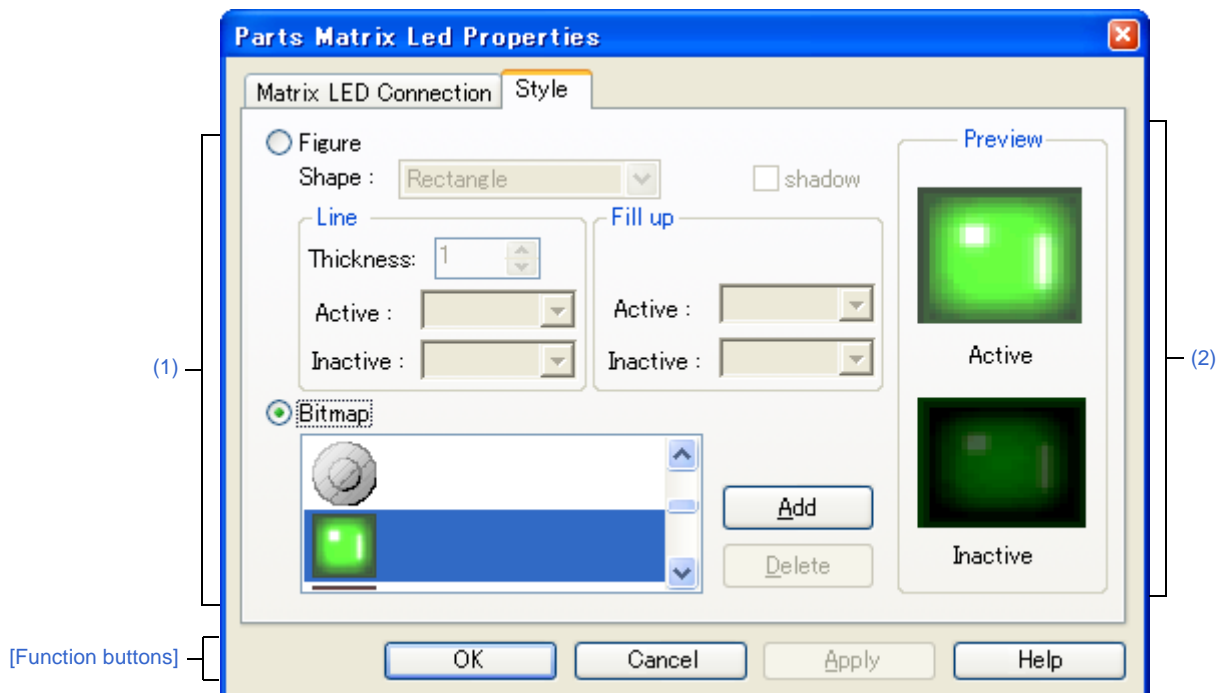
---

This section describes the following.

[How to open]

On the I/O Panel window, any one of the following:
- Double-click a part object "Matrix LED".
- Select [Properties...] form the context menu on a part object "Matrix LED".
- Select a part object "Matrix LED", and then select [Properties...] form the [View] menu.

[[Matrix LED Connection] tab]

(1) [Label]

Label	This area is used to specify the part name. The part name input here is also displayed in the Parts List dialog box as the label.

(2) Row direction signals setting area

Row Signal	This area is used to specify the output pins connected to the matrix LED row direction signals and their active level.		
	Pin Name	A maximum of 16 pins can be connected. Connection to all the row direction signals can be done by using the scrollbar located on the right side of the pin name input area.	
	Active Level	The active status can be selected with a option button, as follows:	
		LOW	Sets the active level to LOW.
		HIGH	Sets the active level to HIGH (default).

    Remark   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

(3) Column direction signals setting area

| Column Signal	This area is used to specify the output pins connected to the matrix LED column direction signals and their active level.		
	Pin Name	The connection pins can be specified either via direct input or through selection from the drop-down list.A maximum of 16 pins can be connected. Connection to all the column direction signals can be done by using the scrollbar located on the right side of the pin name input area.	
	Active Level	The active status can be selected with a option button, as follows:	
		LOW	Sets the active level to LOW.
		HIGH	Sets the active level to HIGH (default).

    Remark   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

[[Style] tab]

(1) Style information setting area

Figure	Select this option button to display the matrix LED with the following specified figure.		
	Shape	Select the figure shape. Two shapes can be selected: rectangle and ellipse.	
	shadow	Cannot be selected.	
	Line	Specify and change the figure line, as follows. You can change the color of figure line by clicking the pull-down button.	
		Thickness	Specifies the line thickness. Specification is made either using a spin button or through direct input. A value from 1 to 100 can be specified.
		Active	Specifies the color of the line during active display.
		Inactive	Specifies the color of the line during inactive display.
	Fill up	Specifiy and change the figure filling, as follows. You can change the color of figure filling by clicking the pull-down button.	
		Active	Specifies the fill color during active display.
		Inactive	Specifies the fill color during inactive display.
Bitmap	Select this option button to display the matrix LED with the following specified bitmap (default)		
	Selection list	Select a bitmap to be used from the selection list. The selectable bitmaps appear in the selection list.	
	[Add] button	Opens the Entry Bitmap dialog box below to add a new bitmap to the selection list. The bitmap file to be added can be specified either through file selection using the [...] button, or through direct input. Entry Bitmap Active Bitmap : \common\bitmap\Parts_on.bmp [...] Inactive Bitmap : \common\bitmap\Parts_off.bmp [...] OK Cancel	
	[Delete] button	Deletes the currently selected bitmap from the selection list. Note that only the bitmap that have been added by user can be deleted.	

(2) Preview area

This area displays the style of the matrix LED currently being specified.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

[Operation]

In the Simulation mode, as the results of 1 simulation, the output information of the connection pins is received and display is performed accordingly. When the matrix intersection of a row pin and column pin is active, the corresponding LED lights.

Figure A-108.   Connected Parts Display Example (Matrix LED)
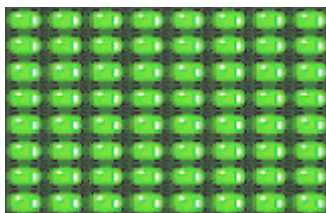
Parts Buzzer Properties dialog box

This property dialog box is used to set or change the pin connection information of a buzzer, which is one of the connection parts in the I/O Panel window.

The pin-connected buzzer displays the information output from the connected pins as a bitmap in the Simulation mode.

The buzzer output can be checked as "display".

There are two types of buzzer display styles, figure and bitmap. These styles can be changed on the [[Style] tab].

Figure A-109.   Parts Buzzer Properties Dialog Box: [Buzzer Connection] Tab
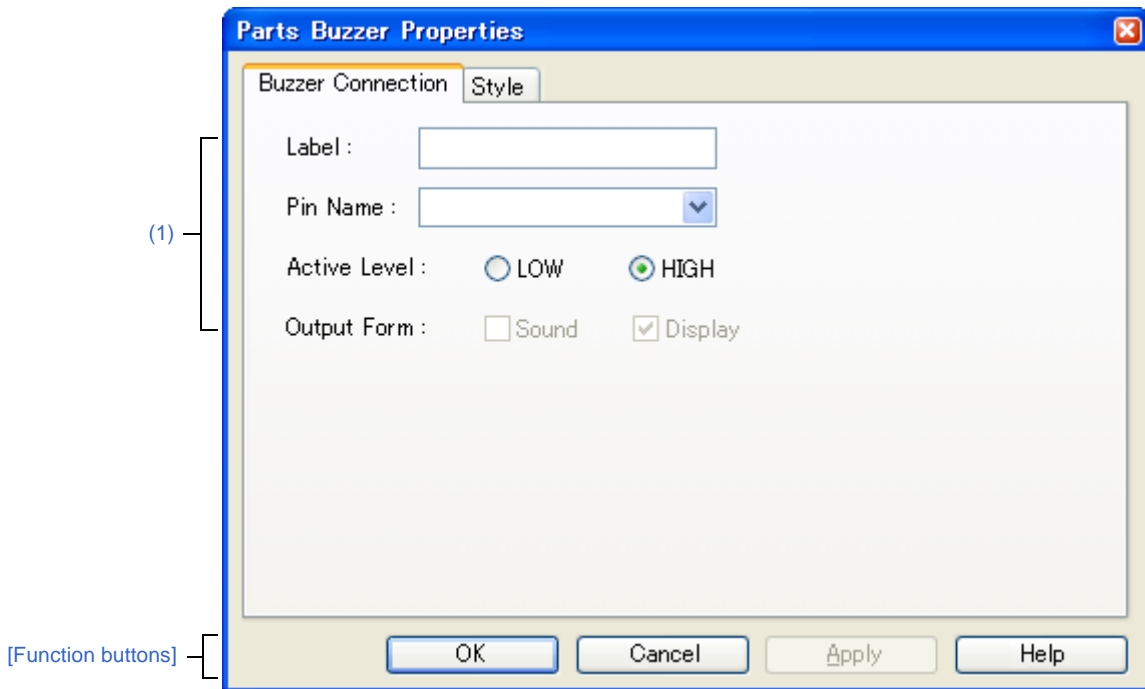
Figure A-110.   Parts Buzzer Properties Dialog Box: [Style] Tab

This section describes the following.
- [How to open]
- [[Buzzer Connection] tab]
- [[Style] tab]
- [Function buttons]
- [Operation]

[How to open]

On the I/O Panel window, any one of the following:
- Double-click a part object "Buzzer".
- Select [Properties...] form the context menu on a part object "Buzzer".
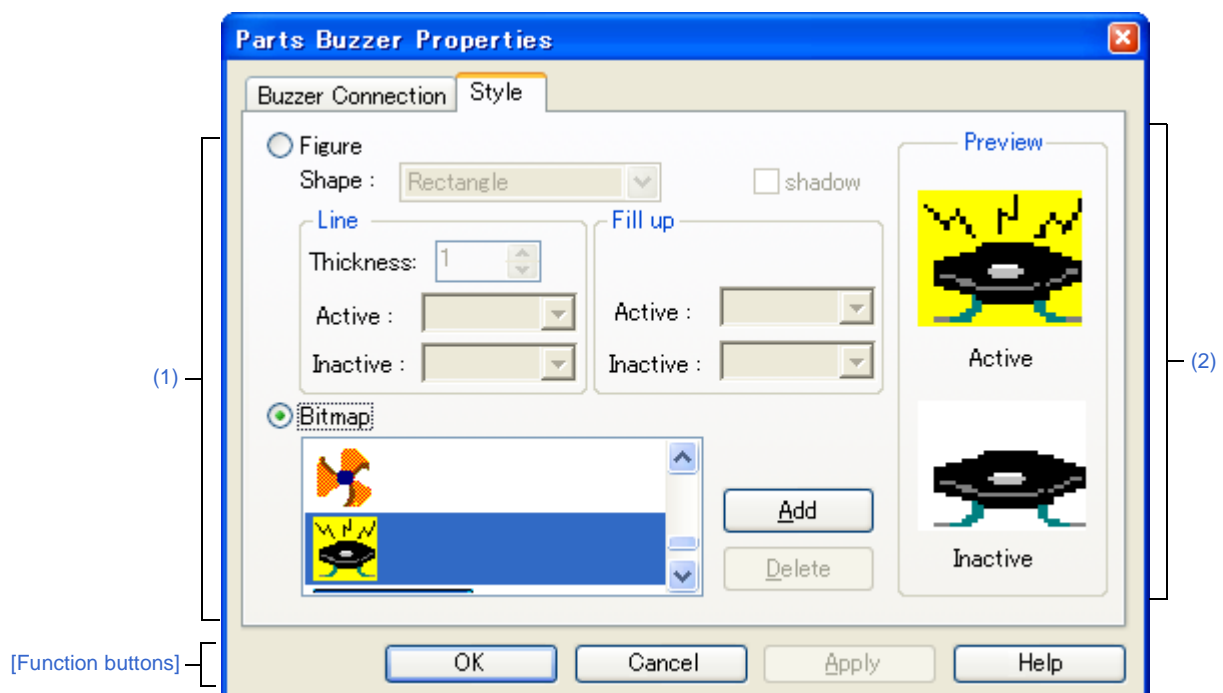- Select a part object "Buzzer", and then select [Properties...] form the [View] menu.

[[Buzzer Connection] tab]

(1) Pin connection information setting area

Label	This area is used to specify the part name. The part name input here is also displayed in the Parts List dialog box as the label.	
Pin Name	This area is used to specify the pin name to be connected (output pin). The connection pins can be specified either via direct input or through selection from the drop-down list.	
Active Level	The active state is selected with a option button, as follows:	
	LOW	Sets the active level to LOW.
	HIGH	Sets the active level to HIGH (default).
Output Form	This item is not allowed to change.	

Remark   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

[[Style] tab]

(1) Style information setting area

Figure	Select this option button to display the buzzer with the following specified figure.		
	Shape	Select the figure shape. Two shapes can be selected: rectangle and ellipse.	
	shadow	Cannot be selected.	
	Line	Specify and change the figure line, as follows. You can change the color of figure line by clicking the pull-down button.	
		Thickness	Specifies the line thickness. Specification is made either using a spin button or through direct input. A value from 1 to 100 can be specified.
		Active	Specifies the color of the line during active display.
		Inactive	Specifies the color of the line during inactive display.
	Fill up	Specify and change the figure filling, as follows. You can change the color of figure filling by clicking the pull-down button.	
		Active	Specifies the fill color during active display.
		Inactive	Specifies the fill color during inactive display.
Bitmap	Select this option button to display the buzzer with the following specified bitmap (default).		
	Selection list	Select a bitmap to be used from the selection list. The selectable bitmaps appear in the selection list.	
	[Add] button	Opens the Entry Bitmap dialog box below to add a new bitmap to the selection list. The bitmap file to be added can be specified either through file selection using the [...] button, or through direct input. **Entry Bitmap** Active Bitmap : \common\bitmap\Parts_on.bmp [...] Inactive Bitmap : \common\bitmap\Parts_off.bmp [...] OK Cancel	
	[Delete] button	Deletes the currently selected bitmap from the selection list. Note that only the bitmap that have been added by user can be deleted.	

(2) Preview area

This area displays the style of the buzzer currently being specified.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

[Operation]

In the Simulation mode, the active level output of the connected pins can be expressed as a bitmap.

The following bitmaps are displayed according to the pin's output value (active/inactive).

Figure A-111.   Connected Parts Display Example (Buzzer)
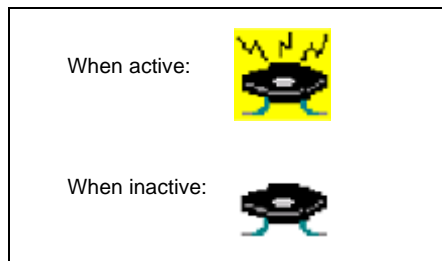
Pull up/Pull down dialog box

This dialog box is used to set or change the pin connection information of the pull-up/pull-down resistors, which are one of the connection parts of the I/O Panel window.

The setting method for these connection parts differs from those for other parts. The connection information of all the pins is managed as a group in this dialog box.

Figure A-112.   Pull up/Pull down Dialog Box
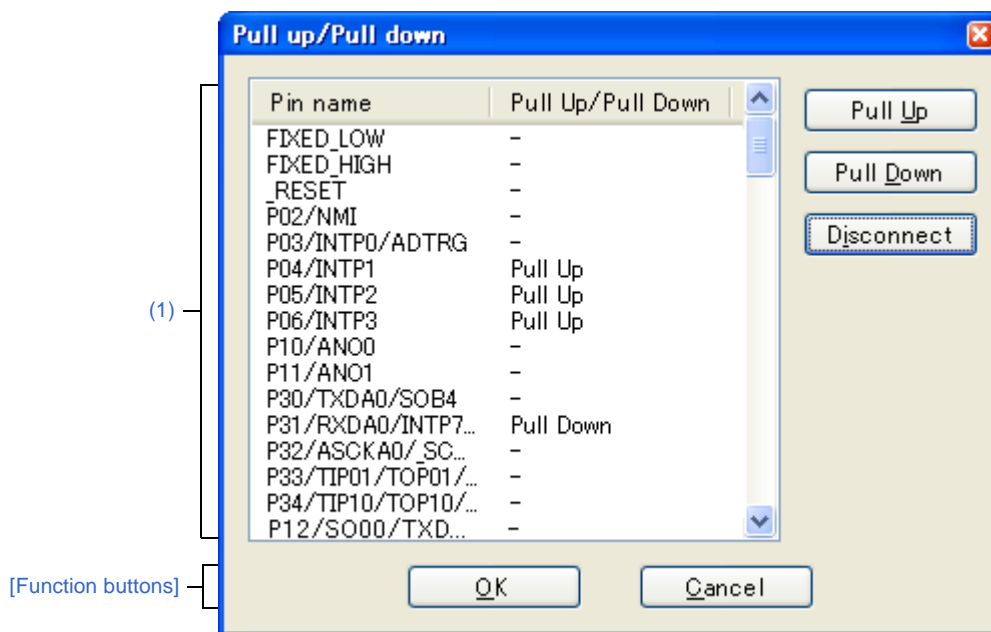
This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- With the I/O Panel window in focus, click the ⬍ button or select [Pull up/Pull down...] from the [Parts] menu.

[Description of each area]

(1) Connection information display area

Pin name	Displays the names of the pins that can be connected to pull-up/pull-down resistors.	
Pull Up/Pull Down	Displays the connected status of the pins.	
	Pull Up	Indicates pull-up resistor is connected.
	Pull Down	Indicates pull-down resistor connected.
	-	Indicates no pull-up/pull-down resistor connected.
Buttons	Changes the pin connection information of the pull-up/pull-down resistors.	
	[Pull Up]	Connects the pins selected to pull-up resistors. When connection is completed, "Pull Up" is displayed.
	[Pull Down]	Connects the pins selected to pull-down resistors. When connection is completed, "Pull Down" is displayed.
	[Disconnect]	Cancels the connected status of the pins selected. When connection is completed, "-" is displayed.

[Function buttons]

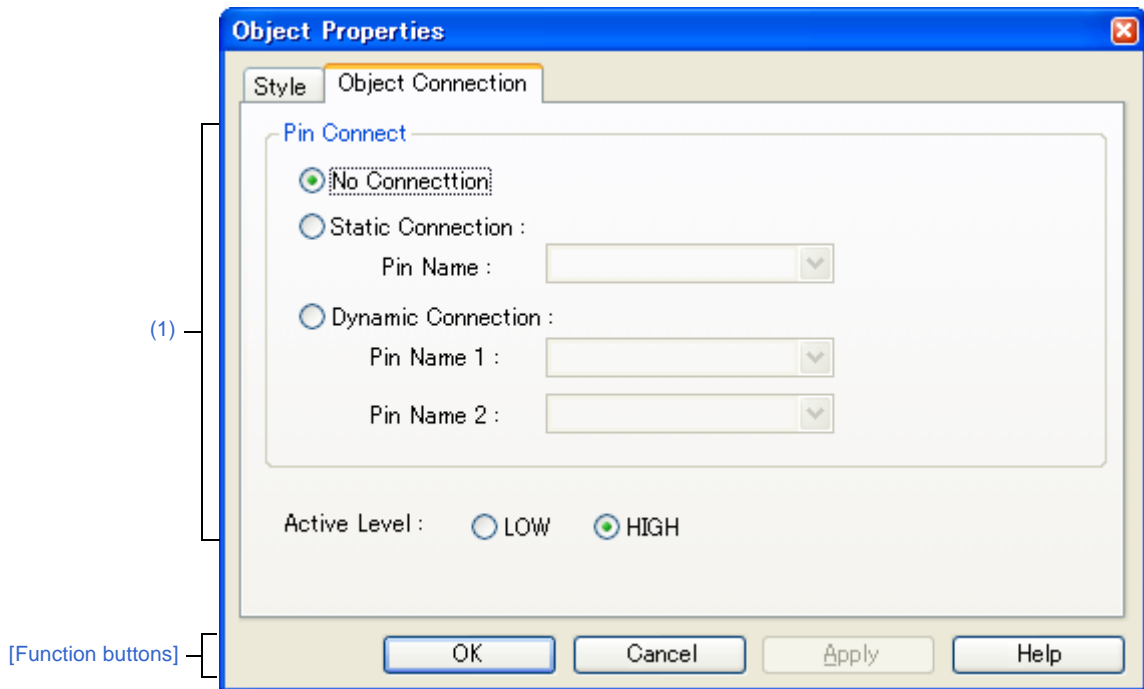| Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.

Object Properties dialog box

This property dialog box is used to set or change the connection information fed to the pins of figure object (including text and bitmap) of the I/O Panel window.
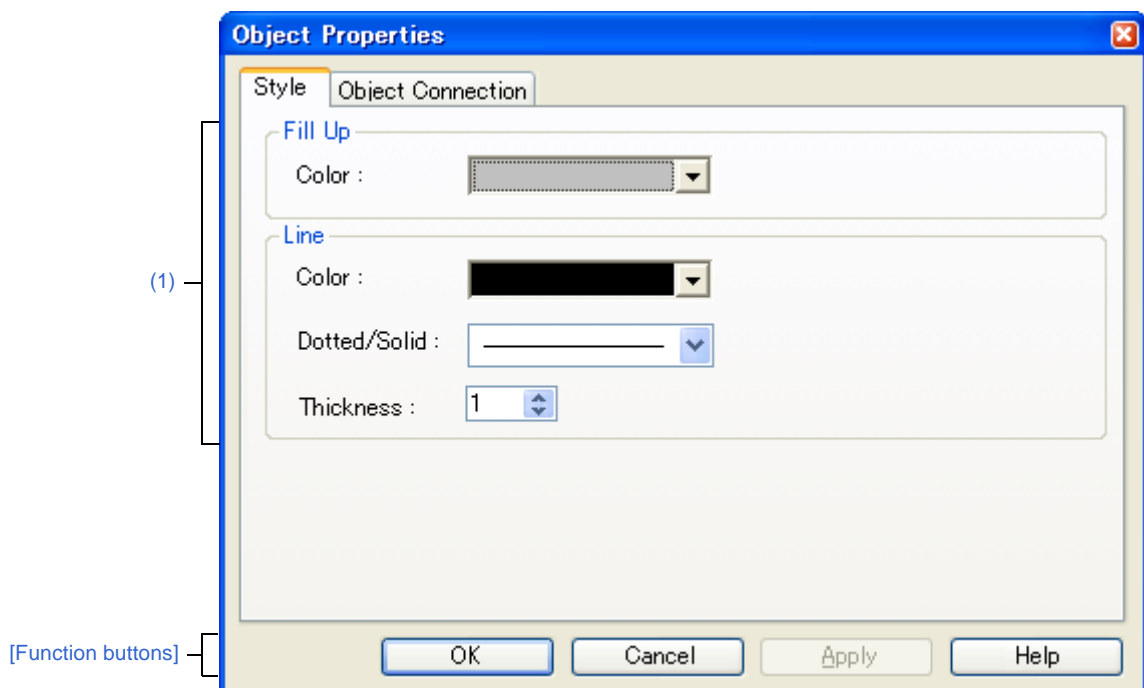
Show/hide can be switched for each pin-connected object by the output status of the connected pins in the Simulation mode.

The default status of each signal is active HIGH. Display styles can be changed on the [[Style] tab].

Figure A-113.   Object Properties Dialog Box: [Object Connection] Tab

Figure A-114.   Object Properties Dialog Box: [Style] Tab

This section describes the following.

- [How to open]
- [[Object Connection] tab]
- [[Style] tab]
- [Function buttons]

[How to open]

On the I/O Panel window, any one of the following:

- Double-click a figure object.
- Select [Properties...] form the context menu on a figure object.
- Select a figure object, and then select [Properties...] form the [View] menu.

[[Object Connection] tab]

(1) Pin connection information setting area

Pin Connect	Select the method for connecting objects and output pins by clicking the corresponding option buttons, and specify the output pin name. As a result of the connection, figure display is switched by the ON/OFF status of the connected output pin.	
	No Connection	The object and pin are not connected (default). Objects not connected to pins are always in the displayed status.
	Static Connection	Connects the figure to 1 output pin. The connection pin can be specified in [Pin Name] either via direct input or through selection from the drop-down list. During simulation, the object is displayed when the specified output signal data is active.
	Dynamic Connection	Connects the figure to 2 output pins. The connection pin can be specified in [Pin Name1] and [Pin Name2] either via direct input or through selection from the drop-down list. During simulation, the object is displayed when the specified output signal 1 data and the specified output signal 2 data are both active.
Active Level	The common active status of each output signal can be selected from the option buttons.	
	LOW	Sets the active level to LOW.
	HIGH	Sets the active level to HIGH (default).

Remark   For the pin names that can be specified, see the user's manual of the microcontroller that is used.

[[Style] tab]

(1) Style information setting area

Fill up	Specifies and changes settings related to each object filling[Note], as follows:
	
	The area to be filled differs according to the object.
	
	- Line
	  Not applicable
	
	- Rectangle, Ellipse, Rounded Rectangle
	  Inside area enclosed by contour lines
	
	- Polygon
	  Inside area enclosed by lines linking apexes
	
	- Text
	  Inside text box
	
	- Bitmap
	  Inside figure drawing area	

	Color	Specifies and changes settings of color You can change the color by clicking the pull-down button.

Line	This area is used to perform settings and changes related to the lines of objects.
	
	The definition of line for each type of object is provided below.
	
	- Line
	  All areas of the object
	
	- Rectangle,Ellipse,Rounded Rectangle
	  Contour line of the figure
	
	- Polygon
	  Lines that link the apexes of the polygon
	
	- Text
	  Contour lines of the text box
	
	- Bitmap
	  Contour lines of figure drawing area	

	Color	Specifies and changes the line color. You can change the color by clicking the pull-down button.
	Dotted/ Solid	Specifies and changes line shapes (dotted line/solid line). The desired line shape can be selected from the drop-down list. Note that this item can be changed only when [Thickness] is set to "1".
	Thickness	Specifies and changes the line thickness. The desired line thickness can be specified either via direct input or through selection from the spin button. A value in the range of 1 to 100 (decimal) can be specified.

Note   At this time, if the object that have been pasted from a bitmap file, it becomes invisible.

[Function buttons]

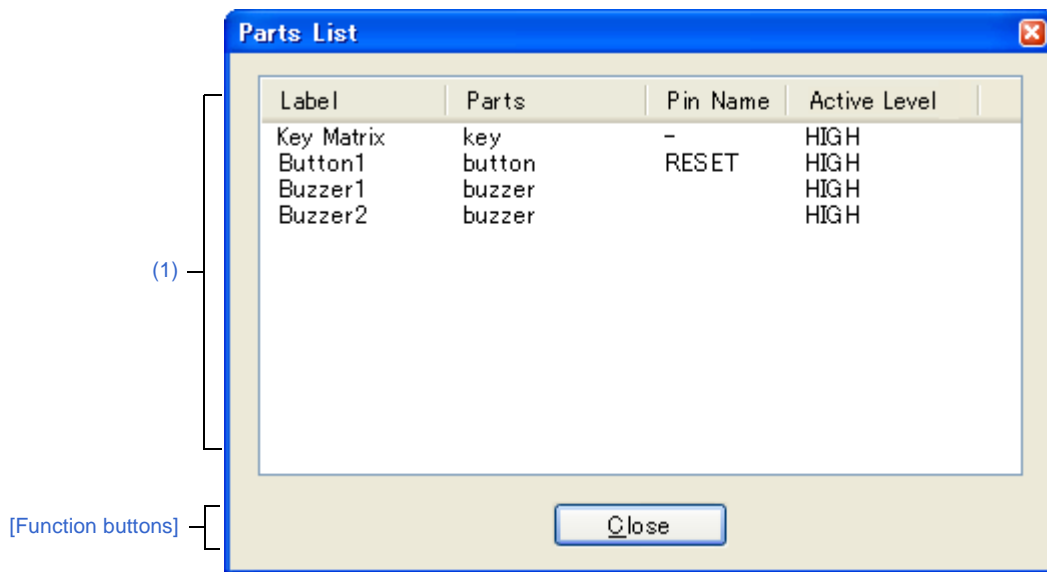| Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.
Apply	Cannot be selected.
Help	Displays the help for this dialog box.

Parts List dialog box

This dialog box displays all the figure objects created in the I/O Panel window as well as the pin connection status of part objects.

The pin connection settings for each object can be changed in the property dialog box, which can be opened by double-clicking the relevant object listed in this dialog box, or selecting the relevant object listed in this dialog box and then selecting the [View] menu >> [Properties...].

Figure A-115.   Parts List Dialog Box

This section describes the following.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- With the I/O Panel window in focus, select [Parts List...] form the [View] menu.

[Description of each area]

(1) Pin connection status display area

Label	Displays the label (name) attached to the object. Nothing is displayed for objects that do not have a label.	
Parts	Displays the part category.	
	rectangle	Straight line, rectangle, ellipse, rounded rectangle, fan shape
	polygon	Polygon
	text	Text
	bitmap	Bitmap
	button	Push button, pull button, group button
	analog button	Analog button
	key	Key matrix
	level gauge	Level gauge
	led	LED
	7segment led	7-segment LED
	14segment led	14-segment LED
	matrix led	Matrix LED
	buzzer	Buzzer
	groups	Grouped part
Pin Name	Displays the pins connected to parts. "-" is displayed for parts that are connected to multiple pins, and a blank is displayed for parts that are unconnected.	
Active Level	Displays the active value set for the part. "-" is displayed for parts that are connected to multiple pins, and a blank is displayed for parts that are unconnected.	

[Function buttons]

Button	Function
Close	Closes the this dialog box.

Serial window

This window is used to communicate with the serial interface provided in the CPU.

Since this window operates as the serial interface of the remote node of the CPU, transmission data from CPU turns into reception data in this window, and transmission data from this window turns into reception data in the CPU.

The following two types of files can be handled in this window.

The transmission/reception data displayed in the top area in this window can be saved to the serial log data file (*.log) (CSV format) by selecting the [File] menu >> [Save]/[Save As...].

Moreover, the transmission data created in the lower part of this window can be saved to the serial transmission data file (*.ser) (CSV format) by clicking the [Save] button (the contents can be restored by clicking the [Load] button).

Saving/restoring the created data can also be performed by saving/loading the project file. In this case, however, data is not saved as a CSV format text file but saved into the project file.

Cautions 1.   **If the saved serial transmission data file is opened or the project file is opened while Simulator GUI is running with a microcontroller different from the one used when the file was created, the settings of the serial interface that are not provided in the microcontroller will not be restored.**

2.   **Multiple instances of this window can be opened. After opening this window, select the serial interface to be verified in the Serial selection area.**

Figure A-116.   Serial Window
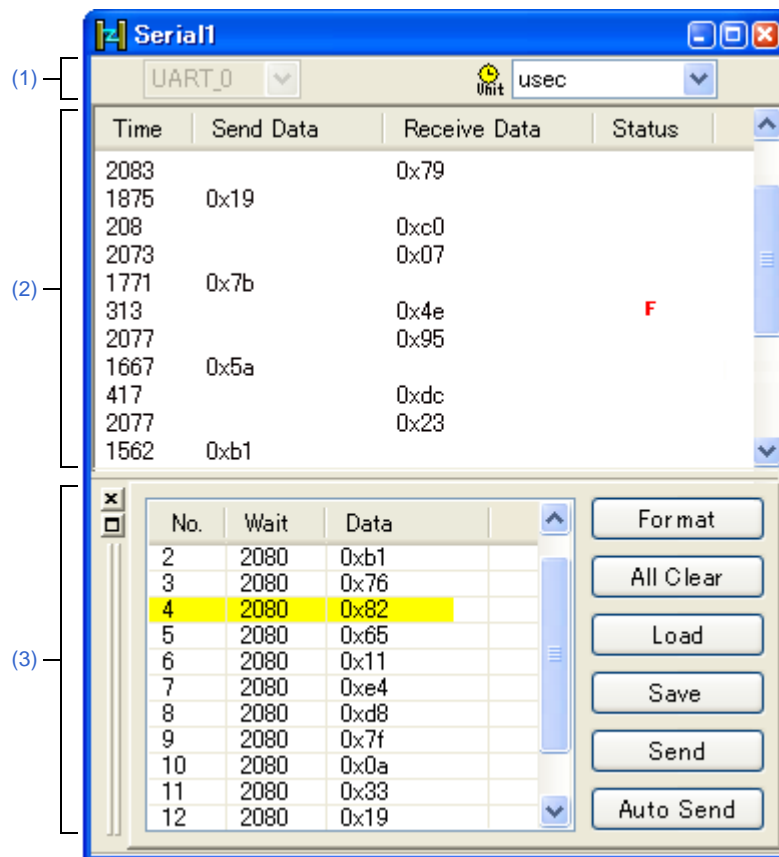
This section describes the following.

- [How to open]
- [Description of each area]
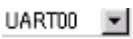- [Dedicated menu (Serial window)]
- [Context menu]

[How to open]

- Click the ![button] button.
- Select [Serial] from the [Simulator] menu.

[Description of each area]

(1) Serial selection area

Select the serial interface to be used.

| UART00 ▼ | Select the serial interface to be used from the list of provided serial interfaces (drop-down list)^Note. |
| Unit usec ▼ | Select from the drop-down list the time information unit applied to [Wait] in the Serial editor area.
The time information unit can be changed by selecting the [Edit] menu >> [Time unit]. |

Note Once selected, the serial interface cannot be changed.
If you wish to change the serial interface, open the another Serial window newly.

(2) Log display area

Displays the transmission/reception data.

The display timing is when all the bits constituting the data have been received or sent.

Only data from which the start bit, stop bit, and parity bit have been deleted is displayed in this area.

The notation of data can be changed by selecting the [View] menu>> [Numeric Representation] >> [Binary]/[Hexadecimal].

Log display is cleared upon debugger or simulator reset.

Time		Displays the time from the completion of reception/transmission of the previous data until completion of reception/transmission of the current data. The time information unit is specified by selecting the [Edit] menu >> [Time unit].
Send Data		Displays the data sent by this window (data received by the CPU).
Receive Data		Displays the data received by this window (data sent by the CPU).
Status		Displays the status during data reception. When an error occurs, one of the following marks is displayed. When everything is normal, nothing is displayed.
	P	Parity error (mismatching parity bit)
	F	Framing error (stop bit not detected)

(3) Serial editor area

This area is where the transmission data is created.

This area can be shown or hidden by selecting the [View] menu >> [Serial Editor].

No.	This is a number assigned sequentially from the beginning. It cannot be directly written.
	The maximum number is 9,999 lines.
Wait	Specifies the time from the completion of transmission of the immediately previous data until the start of transmission of next data.Valid during transmission using the [Auto send] button.
	The time information unit is the unit selected by selecting the [Edit] menu >> [Time unit].
	The wait value is input by placing the cursor in the Wait field to be edited and double-clicking.
	One wait value can be written per operation.
Data	This area is used to edit transmission data.
	Data can be directly input by placing the cursor in the data field and double-clicking.
	Data suffixed by "0x" is treated as hexadecimal data, and data suffixed by "0b" is treated as binary data. The default code is the hexadecimal code.
	If a bit length different from that specified in the Format (UART) dialog box or Format (CSI) dialog box is specified, data from the lower bit is valid.
	One data can be written per operation.
Button	Format	Opens the Format (UART) dialog box or Format (CSI) dialog box.
	All Clear	Clears all Serial editor area.
	Load	Loads the contents of the previously saved serial transmission data file (*.ser) and restores them in the Serial editor area.
		A file created for UART cannot be loaded as a file for CSI and vice versa.
	Save	Saves the contents of the Serial editor area to the specified serial transmission data file (*.ser).
	Send	Sends one of the data selected in the Serial editor area.
		The next data becomes selected upon completion of transmission.
		If no data is selected, the first data is sent.
	Auto Send	Makes the data selected in the Serial editor area the first data, and automatically transfers from the data to the bottom of the area. The data transmission time interval is the time specified for Wait.

Caution　The help for this window will not be displayed even if the [F1] key on the keyboard is pressed while the cursor is placed in this area.

**Remark　** When the CSI selected in this window is set to master mode, the clock must be supplied for reception. To perform reception, transmission of dummy data is therefore required.

[Dedicated menu (Serial window)]

(1) [Edit] menu

Insert		Inserts a new line immediately before the selected line.
Cut		Cuts the selected range and saves it to the clipboard.
Copy		Copies the selected range and copies it to the clipboard.
Paste		Pastes the contents of the clipboard to the selected location.
Delete		Deletes the selected range.
Time unit		Selects the time unit.
	main clock	Main clock (default)
	usec	Microsecond
	msec	Millisecond
Format...		Opens the Format (UART) dialog box or Format (CSI) dialog box.

(2) [View] menu

Serial Editor		Selects whether Serial editor area is displayed or not.
Numeric Representation		Changes the notation of the Log display area display method.
	Binary	Displays binary numbers.
	Hexadecimal	Displays hexadecimal numbers.

(3) [Option] menu

|	
Customize...	Opens the Customize dialog box.

[Context menu]

The following context menus are available in the Serial editor area.

Insert	Inserts a new line immediately before the selected line.
Cut	Cuts the selected range and saves it to the clipboard.
Copy	Copies the selected range and copies it to the clipboard.
Paste	Pastes the contents of the clipboard to the selected location.
Delete	Deletes the selected range.

Format (UART) dialog box

This dialog box is used to set the serial format for the asynchronous serial interface (UART).

Figure A-117.   Format (UART) Dialog Box
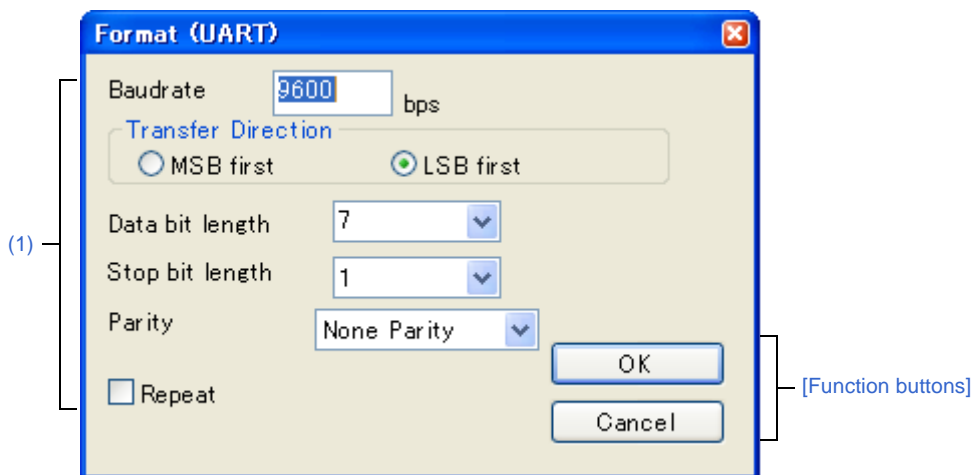
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

On the Serial window with the UART serial interface, any one of the following:
- Click the [Format] button.
- Select [Format...] from the [Edit] menu

[Description of each area]

(1) Serial format setting area

Baudrate	Directly input the serial baud rate value as an integer. (Unit: bps)
Transfer Direction	Select the transfer direction.
	MSB first	Sets MSB first as the transfer direction.
	LSB first	Sets LSB first as the transfer direction (default).
Data bit length	Select the bit length of the transmission data from the drop-down list, or specify it through direct input (default:7).
Stop bit length	Select the stop bit length from the drop-down list (default:1).
Parity	Select the parity information (none parity (default) /odd parity/even parity/0 parity).

Repeat	Select this item to repeat data transfer when the [Auto Send] button in the Serial window has been clicked.	
	☑	Following transmission of the last data during automatic transmission, returns to the beginning of the data and performs automatic transmission.
	☐	Following transmission of the last data during automatic transmission, stops transmission (default).

Remark   For the selectable range, see the user's manual of the microcontroller that is used.

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.

Format (CSI) dialog box

This dialog box is used to specify the serial format for the 3-wire serial interface (CSI).

Figure A-118.   Format (CSI) Dialog Box
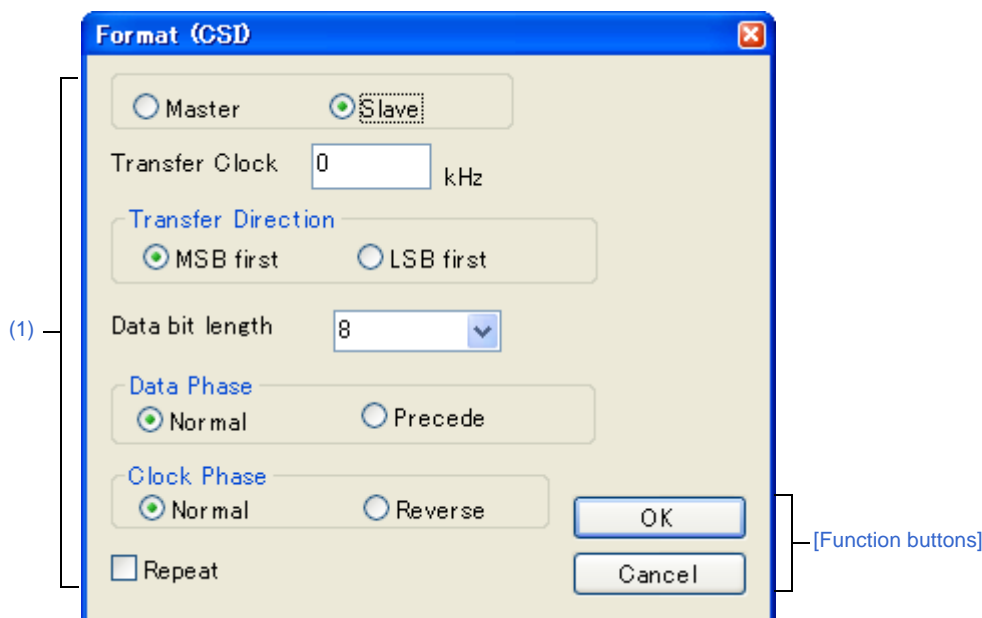
This section describes the following.
- [How to open]
- [Description of each area]
- [Function buttons]
- [Transmission/reception when 3-wire serial interface (CSI) is selected]

[How to open]

On the Serial window with the CSI serial interface, any one of the following:
- Click the [Format] button.
- Select [Format...] from the [Edit] menu
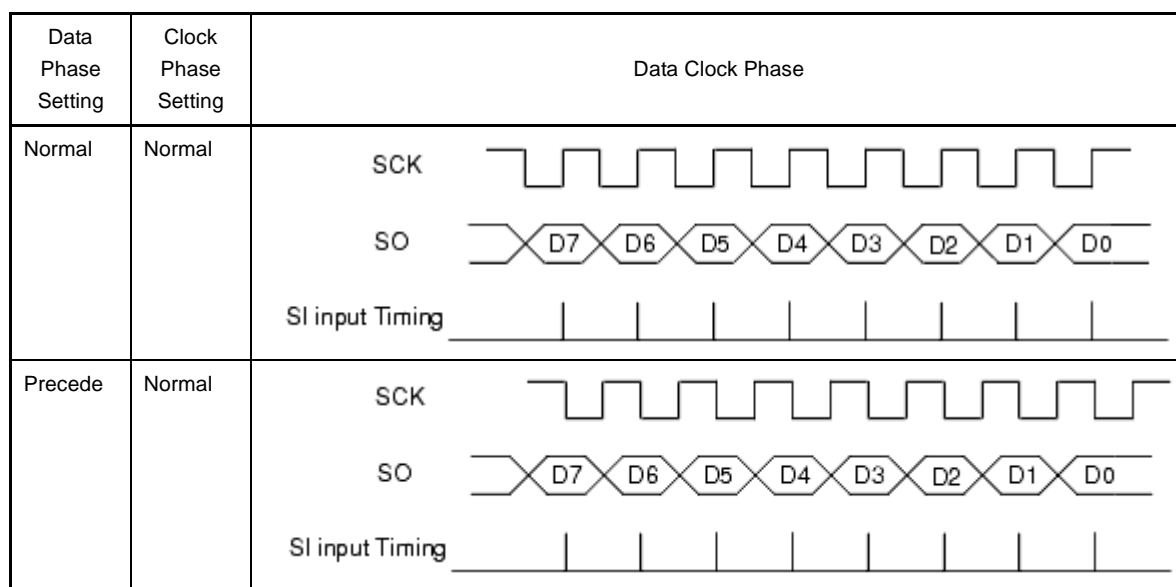
[Description of each area]

(1) Serial format setting area

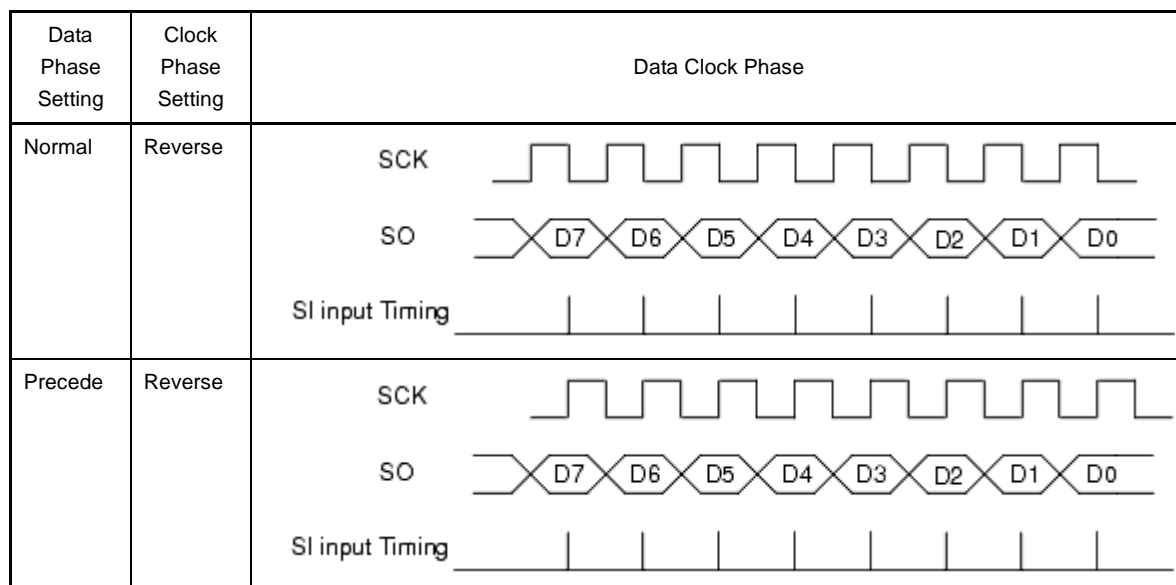| Master, Slave	Select the transfer mode.	
	Master	Operates this window as a master. Setting of [Transfer Clock] is required for generating the clock during communication.
	Slave	Operates this window as a slave (default). Communication is performed using the clock of the serial interface provided in the CPU.
Transfer Clock	Directly input the transfer clock value (unit: kHz). Values with decimals can also be set. This item must be set when master is selected.	

Transfer Direction	Select the transfer direction.	
	MSB first	Sets MSB first as the transfer direction (default).
	LSB first	Sets LSB first as the transfer direction.
Data bit length	Select the bit length of the transmission data from the drop-down list, or specify it through direct input (default: 8).	
Data Phase	Select the transmission/reception timing to set the data phase. The data clock phase is set in combination with "Clock Phase" as shown in "Table A-18.　Data Clock Phase Settings".	
	Normal	Transmits/receives data at the normal 3-wire serial transmission/reception timing (default).
	Precede	Transmits/receives data at a timing half a clock of the operation clock earlier than the normal 3-wire serial transmission/reception timing.
Clock Phase	Select the transmission/reception clock waveform to set the clock phase. The data clock phase is set in combination with "Data Phase" as shown in "Table A-18.　Data Clock Phase Settings".	
	Normal	Operates on the normal 3-wire serial clock. Transfer starts at the falling edge of the clock (default).
	Reverse	Operates on the reverse clock of the normal 3-wire serial clock. Transfer starts at the rising edge of the clock.
Repeat	Select this item to repeat data transfer when the [Auto Send] button in the Serial window has been clicked.	
	☑	Following transmission of the last data during automatic transmission, returns to the beginning of the data and performs automatic transmission.
	☐	Following transmission of the last data during automatic transmission, stops transmission.

Remark　For the selectable range, see the user's manual of the microcontroller that is used.

Table A-18.　Data Clock Phase Settings

Data Phase Setting	Clock Phase Setting	Data Clock Phase
Normal	Normal	
Precede	Normal	

Data Phase Setting	Clock Phase Setting	Data Clock Phase
Normal	Reverse	
Precede	Reverse	

[Function buttons]

Button	Function
OK	Validates the settings and closes this dialog box.
Cancel	Cancels the settings and closes this dialog box.

**[Transmission/reception when 3-wire serial interface (CSI) is selected]**

The Serial window when CSI serial interface is selected always operates in the transmission/reception mode regardless of whether [Master] or [Slave] is selected in this dialog box.

When [Master] is selected	Data is transmitted or received immediately after clicking the [Send] button or the [Auto Send] button in the Serial editor area.
When [Slave] is selected	Data becomes ready to be transmitted or received after clicking the [Send] button or the [Auto Send] button in the Serial editor area.
	Data starts to be transmitted or received when the CSI clock signal is received in the data transmission/reception ready status. This ready status is released when data transmission or reception is completed. (Even if the CSI clock is received, data is not transmitted or received in other than the data transmission/reception ready status.)

Accordingly, when only wanting to reception data in the Serial window when "Slave" is selected, execute as follows (Reception setting when CSI serial slave is selected):
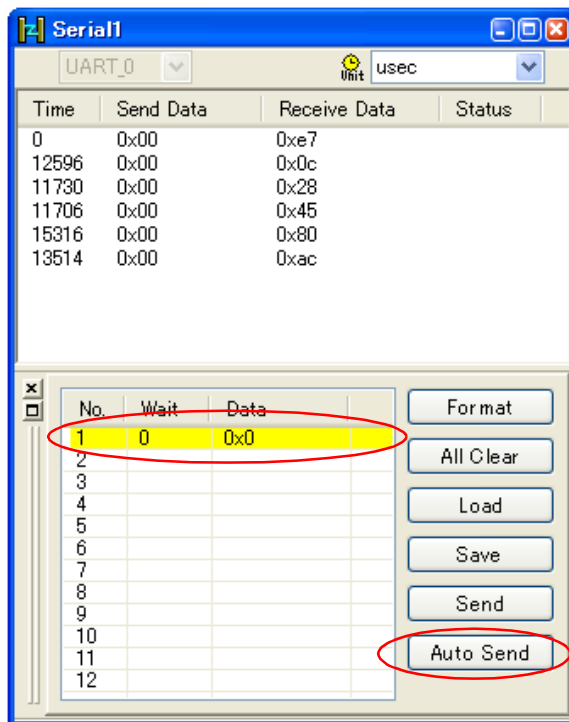
(1) Set [Repeat]

Set the [Repeat] check box in the [Format (CSI)] dialog box.
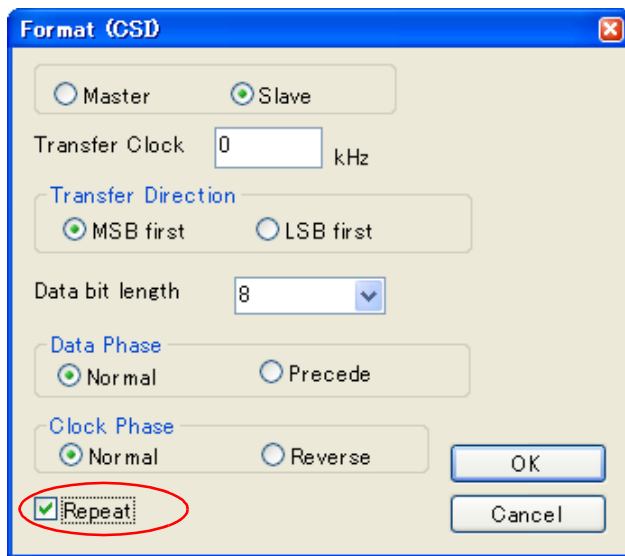
(2) Set Wait time

Set the Wait time to 0 as a dummy setting in the Serial window.

(3) Click the [Auto Send] button

APPENDIX B   USER OPEN INTERFACE

Appendix B provides detailed explanations of the user open interface that is one of the functions provided by Simulator GUI.

B.1   Overview

Simulator GUI provides two ways of creating an environment where a target system, as well as a CPU (CPU core + internal peripherals), can be simulated.

One is the I/O Panel window, via which a user-friendly simulation environment can be organized through GUI manipulation, by supplying standard components for connection and their manipulation environment.

The other is to create the simulation environment of the target system that uses the user open interface to be explained in this appendix. In this environment, functions that cannot be realized on the I/O Panel window can be used if the user programs an external user model.

Figure B-1.   Programming Image of User Model
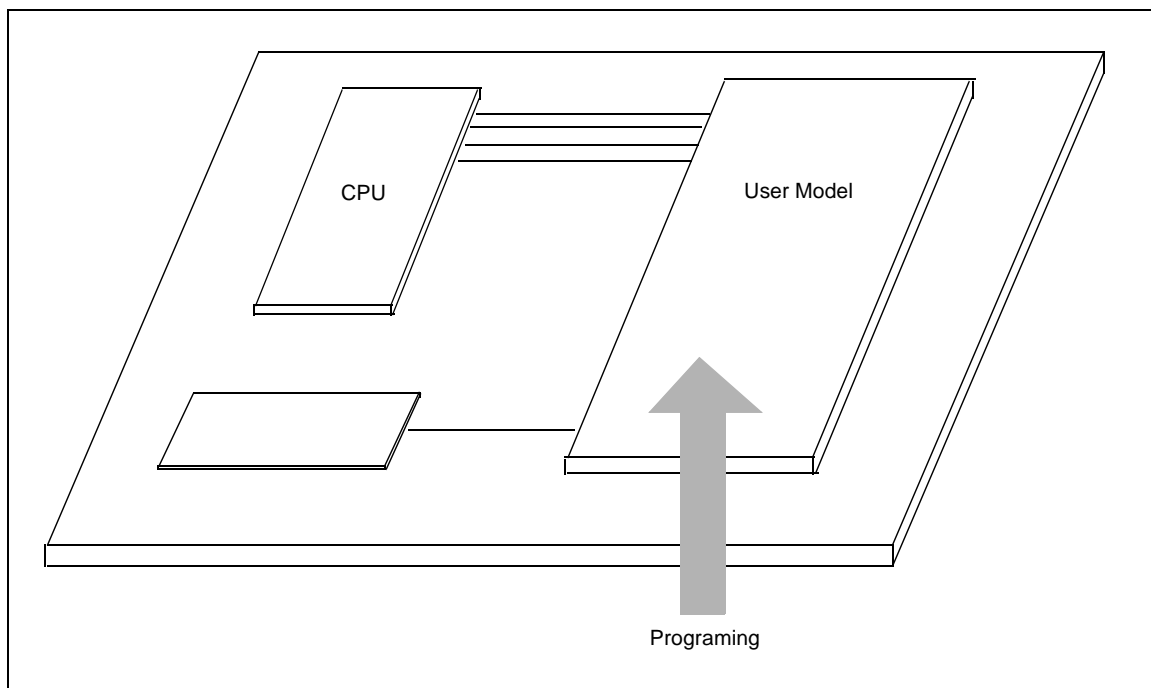
B.1.1   Types of interface functions

Simulator GUI's user open interface supplies the following types of interface functions (see "B.4   Supplied Interface Functions").

Table B-1.   Types of Functions Supplied by User Open Interface

Type	Description
Basic interface functions	Basic function of simulation - Initialization notification - Reset notification, etc.
Time interface functions	Cyclic timer function for time-series processing of the user model - Setting of timer - Clearing of timer - Notification of timer time, etc.
Pin interface functions	Pin I/O function - Signal output to pin - Notification of signal input to pin
External bus interface functions[Note]	Slave function of external bus - External bus read access notification - External bus write access notification, etc.
Serial interface functions	Serial transmission/reception function - Transmission of serial data - Notification of reception of serial data, etc.
Signal output unit interface functions	Function to output signals in accordance with signal data file - Signal output in accordance with signal data file, etc.

Note   When using the external bus interface function, the external memory area to use must be set to [Target memory area] with the [Memory Type] area of the Memory Mapping dialog box.

B.1.2   Interface methods

Simulator GUI's user open interface has the following interface methods.

(1) C language interface

The user open interface consists of a C language API (Application Program Interface) function set.
Therefore, program the user model in C language.

(2) Callback function method

The user open interface uses the callback function method as a means to call a program from the system.
The callback function method is that a program (user model) is called by the system (CPU) when it is necessary.
This method uses the pointer to the function which is defined on the program (user model). The system (CPU) calls the user program (user model) by using this pointer.
While the provided API functions call the system from the program, the callback function is used to call the program from the system, such as when inputting a signal to a pin.

(3) Event-driven method

The user open interface uses an event-driven method in which processing is described in accordance with occurrence of events.

Therefore, a callback function prepared on the user model side is called if an event such as initialization of simulation, resetting the CPU, signal output to a pin, or access to the external bus occurs on the Simulator GUI side. In addition, a time interface (= timer function) provided to perform time-series processing of a user model also calls a callback function prepared on the user model side when the specified time has elapsed.

B.1.3   Development environment

Use the following development tools to perform programming with the Simulator GUI's user open interface and create a DLL file.

- Microsoft Visual C++ (Ver. 6.00 or later)

B.2   Creating User Model

This section describes how to create a user model.

B.2.1   Program configuration

The following figure shows the program configuration when the Simulator GUI's user open interface is used to expand a system.

Figure B-2.   Program Configuration
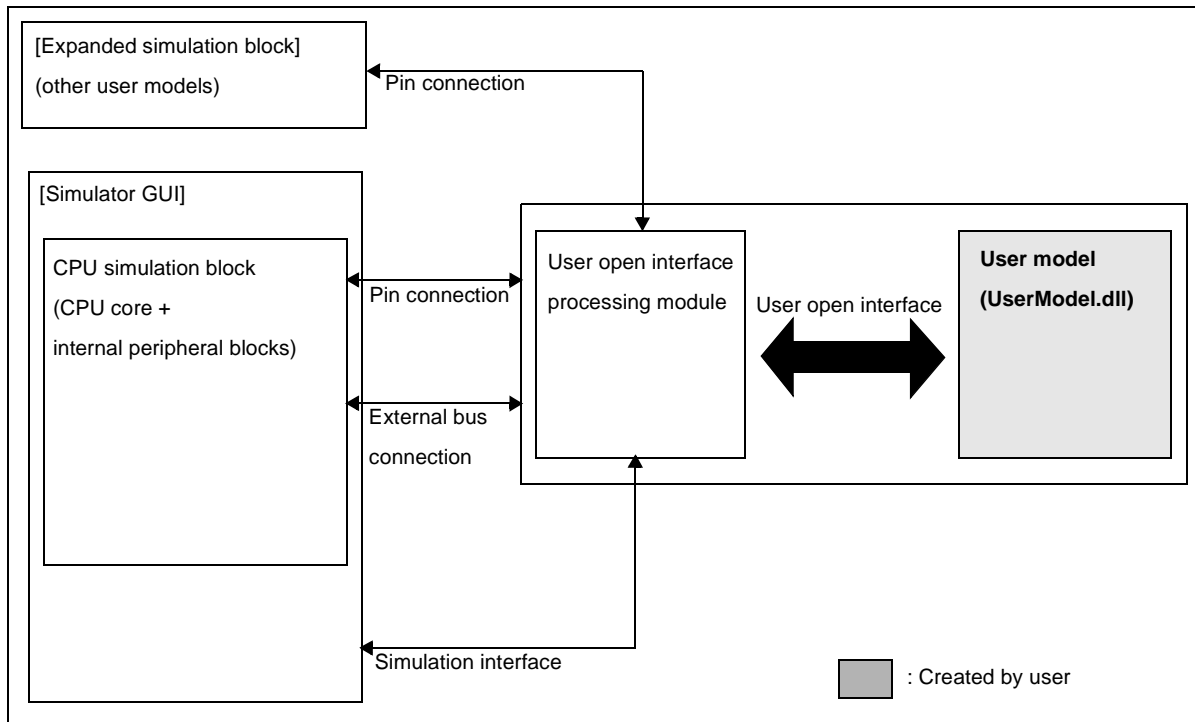
To expand a system, a user model must be created first.

Because the user model operates in conjunction with the simulation system, it interfaces with the user open interface processing module. This interface is the user open interface.

The user model generates resources such as pins and external bus slaves via the user open interface during configuration (processing to configure the simulator that is performed when Simulator GUI is started). By connecting the pins and external bus slaves to the pins and external bus masters of the CPU simulation block, signals can be input to or output from the pins of the CPU simulation block and the external bus can be accessed from the CPU simulation block.

The generated pins and external bus slaves can also be connected to the expansion simulation block (other user models), as well as to the CPU simulation block.

B.2.2   Outline of programming

The user model is programmed in the dynamic link library (DLL) format of WIN32.

The template of a program file is shown below.

Figure B-3.   Template of Program File
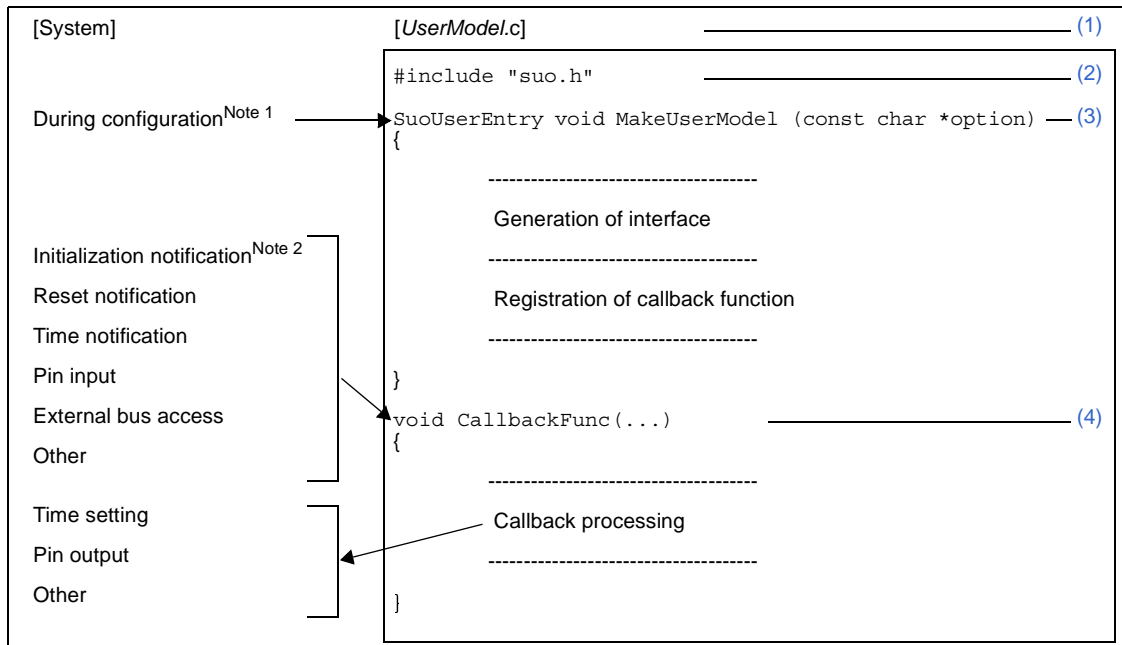
Notes 1. "Configuration" means simulator configuration processing that is executed when Simulator GUI is started.
    2. An initialization notification is reported only once, immediately after Simulator GUI is started when simulator configuration processing has been completed.

(1) File name

Indicates the file name of the user model to be created.

The file name can be determined freely (the suffix for a C language file is "*.c").

(2) Include file

Indicates a include file.

To use the user open interface, the system header file "suo.h" must be included.

(3) MakeUserModel function

Indicates the MakeUserModel function that is called from the system during configuration of Simulator GUI.

Note that the name of this function must be "MakeUserModel".

[Syntax]

```
SuoUserEntry void MakeUserModel(const char *option);
```

The following two types of processing are described in this function.

(a) Interface generation

Because Simulator GUI connects pins and buses during configuration processing when it is started, resources such as pins and buses that are to be connected during configuration must be generated.

To do this, call a function that generates an interface in the MakeUserModel function and generate an interface (see "B.4   Supplied Interface Functions"). The necessary resources will be also generated.

(b) Registering callback function

Callback functions can be registered as necessary.

Caution   When describing a callback function for initialization, be sure to register it at this time; otherwise callback will not function. This is because initialization notification is reported immediately after the MakeUserModel function is called.

(4) Callback function

Indicates a callback function.

Two or more callback functions, such as those for initialization notification, reset notification, time notification, pin input, and external bus access, can be created. Describe processing in accordance with the callback contents in the callback function (see "B.5   User-Defined Functions").

A callback function that has been created must be registered in advance so that it can be called from the system (see "B.4   Supplied Interface Functions"). The name of a callback function can be determined freely, and the format of the function differs depending on the type of callback.

B.2.3 Example of program file (*UserModel*.c)

```c
#include "suo.h"
#include <memory.h>


void Init(void);
void InputP00(SuoHandle handle, int pinValue);
void ReadBUS1(SuoHandle handle, unsigned long addr, int accessSize, unsigned char data[]);
void WriteBUS1(SuoHandle handle, unsigned long addr, int accessSize, const unsigned char data[]);


SuoHandle p00;
SuoHandle p01;
SuoHandle bus1;
unsigned char mem[0x100];


/* MakeUserModel */
SuoUserEntry void MakeUserModel(const char *option)
{
    SuoCreatePin("P00", &p00);
    SuoCreatePin("P01", &p01);
    SuoCreateExtbus("BUS1", 0x200000, 0x100, &bus1);

    SuoSetInitCallback(Init);
    SuoSetInputDigitalPinCallback(p00, InputP00);
    SuoSetReadExtbusCallback(bus1, ReadBUS1);
    SuoSetWriteExtbusCallback(bus1, WriteBUS1);
}


/* callbacks */
void Init(void)
{
    memset(mem, 0, 0x100);
}


void InputP00(SuoHandle handle, int pinValue)
{
    SuoOutputDigitalPin(p01, pinValue);
}


void ReadBUS1(SuoHandle handle, unsigned long addr, int accessSize, unsigned char data[])
{
    memcpy(data, &mem[addr-0x200000], accessSize);
}


void WriteBUS1(SuoHandle handle, unsigned long addr, int accessSize, const unsigned char data[])
{
    memcpy(&mem[addr-0x200000], data, accessSize);
}
```

B.2.4 **Compilation and linking**

Create a *UserModel*.dll by compiling and linking the created *UserModel*.c and suolink.cpp.

Figure B-4. Flow of Compilation and Linking
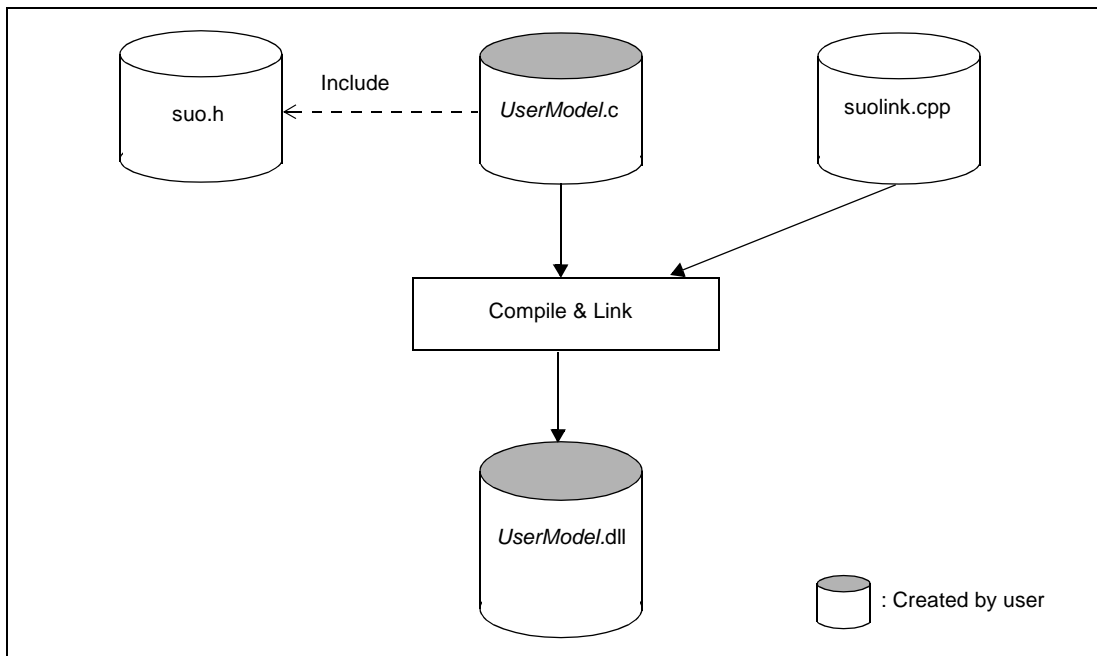
File Name	Description
suo.h	This is a system header file for the user open interface.
	This file is included by the program (*UserModel*.c) but is not compiled.
	Note that "suo.h" is stored in the following folder by default.
	- *Install-folder* \CubeSuite\DebugTools\DebugToolV850Simulator\useropen\sys
suolink.cpp	This is a file that performs dynamic link processing with the user open interface processing module of the system.
	Note that "suolink.cpp" is stored in the following folder by default.
	- *Install-folder* \CubeSuite\DebugTools\DebugToolV850Simulator\useropen\sys
UserModel.c	This is the source file of the user model to be created.
	The file name can be determined freely.
UserModel.dll	This is a binary file of the user model (DLL file).
	The file name can be determined freely.

Caution **To execute a DLL file in an environment in which Microsoft Visual C++ is not installed, the DLL file must be created using the release version.**

B.3   Embedding User Model

This section describes how to embed the created user model (*UserModel*.dll) in Simulator GUI.

To embed the user model in Simulator GUI, use a simulator configuration file (*.cfg).

Therefore, when using Simulator GUI embedding the created user model, you need to specify [Yes] with the [Use simulator configuration file] property in the [Configuration] category on the [Connect Settings] tab of the Property panel, and then specify the configuration file to be used with the [Simulator configuration file] property in the category same as above.

B.3.1   Description in simulator configuration file

Describe the user model generation processing, processing to connect pins and an external buses and so on in the simulator configuration file.

(1) User model generation processing
(2) Pin connection
(3) External bus connection
(4) Other processing

(1) User model generation processing

```
UserModel1 = Device("USEROPEN", "UserModel1.dll UserOption1");
```

(a) *UserModel1*

UserModel1 is a variable that indicates the generated user model. The variable name can be determined freely by user.

(b) Device function

The Device function is used to create a user model.

(c) ˝USEROPEN˝

"USEROPEN" is a user open interface processing module (system module).

(d) *UserModel1*.dll

UserModel1.dll is the binary file (DLL format) of the user model to be created in "B.2   Creating User Model". The file name can be determined freely by user.

Specify a relative path or absolute path from the folder where the simulator configuration file exists as the file path.

Caution   Do not use single-byte spaces for specifying the path name; otherwise, the user model will not be created.

(e) *UserOption1*

UserOption1 is an option character string for *UserModel1*.dll. This character string is passed to the "option" parameter of the MakeUserModel function as is.

(2) Pin connection

```
wire1 =  Wire(1);                                          --- (a)

wire1 += cpu.Port("PinName1");                             --- (b)

wire1 += UserModel1.Port("UserPinName1");                 --- (c)
```

(a) Generation of a wire

Generate a wire (= line that connects pins) by using the Wire function.

Be sure to specify "1" for the argument of the Wire function.

wire1 is a variable that indicates the generated wire. The variable name can be determined freely.

(b) Connection of the wire and CPU

Connect one end of the wire to a pin of the CPU.

Specify the name of the external CPU pin to be connected by using uppercase characters, as "*PinName1*" (note that lowercase characters cannot be used). Enclose the pin name between double quotation marks ("").

(c) Connection of the wire and user model

Connect the other end of the wire to a pin of the user model.

Specify the name of the user model pin to be connected as "*UserPinName1*" (pin name generated in the MakeUserModel function). Enclose the pin name between double quotation marks ("").

Add this line to connect two or more user model pins to the same wire.

(3) External bus connection

```
extbus1 =  BUS(n);                                         --- (a)

extbus1 += cpu.BusMasterIF("EXTBUS");                      --- (b)

extbus1 += UserModel1.BusSlaveIF("UserExtbusName1");      --- (c)
```

(a) Generation of a bus

Generate a bus by using the BUS function.

Argument *n* of the BUS function is the data bus bit width. This may be 8, 16, or 32.

extbus1 is a variable that indicates the generated bus. The variable name can be determined freely.

(b) Connection of the bus and CPU

Connect one end of the bus to the external bus master of the CPU.

Specify the external bus master "EXTBUS" for the argument.

(c) Connection of the bus and user model

Connect the other end of the bus to the external bus of the user model.

Specify the name of the external bus of the user model to be connected as "*UserExtbusName1*" (the external bus name generated in the MakeUserModel function). Enclose the external bus name between double quotation marks ("").

Add this line to connect two or more user model external buses.

(4) Other processing

In addition to the above, the formulaic connections for the main clock notification pin and the reset notification pin shown below is required to operate the user open interface.

```
clock1 =  Wire(1);                                                  --- (a)
clock1 += cpu.DebuggerPseudoPort("debugger_pseudo_pin_main_clkout");  --- (b)
clock1 += UserModel1.Port("gui_pseudo_pin_clock_notice");           --- (c)
reset1 =  Wire(1);                                                   --- (d)
reset1 += cpu.DebuggerPseudoPort("debugger_pseudo_pin_reset_notice"); --- (e)
reset1 += UserModel1.Port("gui_pseudo_pin_reset_notice");           --- (f)
```

(a) Generation of a wire

Generate a wire (= line that connects pins) by using the Wire function.

Be sure to specify "1" for the argument of the Wire function.

clock1 is a variable that indicates the generated wire. The variable name can be determined freely.

(b) Connection of the wire and main clock notification pin

Connect one end of the wire to Simulator GUI's main clock notification pin.

Specify "debugger_pseudo_pin_main_clkout" for the argument.

(c) Connection of the wire and user model

Connect the other end of the wire to a pin of the user model.

Specify "gui_pseudo_pin_clock_notice" for the argument.

(d) Generation of a wire

Generate a wire (= line that connects pins) by using the Wire function.

Be sure to specify "1" for the argument of the Wire function.

reset1 is a variable that indicates the generated wire. The variable name can be determined freely.

(e) Connection of the wire and reset notification pin

Connect one end of the wire to Simulator GUI's reset notification pin.

Specify "debugger_pseudo_pin_reset_notice" for the argument.

(f) Connection of the wire and user model

Connect the other end of the wire to a pin of the user model.

Specify "gui_pseudo_pin_reset_notice" for the argument.

B.3.2 Example of simulator configuration file

An example of the simulator configuration file is shown below.

In this example, the following connection processing is performed.

Type of Connection	CPU		User Model (SampleModel.dll)	
Pin	"P00/INTP0"	P00 pin	"P00"	Pin manipulating P00
	"P30/TXD1"	Serial output pin	"RXD"	Serial input pin
	"P31/RXD1"	Serial input pin	"TXD"	Serial output pin
External bus	"EXTBUS"	External bus master	"EXTBUS1"	External bus slave 1
	"EXTBUS"	External bus master	"EXTBUS2"	External bus slave 2

```
cpu = CPU('a');
# --------------------------------------------------------------
# SampleModel description
# --------------------------------------------------------------

# Generate SampleModel.dll
model = Device("USEROPEN", "SampleModel.dll -a -b");

# Connect PIN (CPU.P00-MODEL.P00)
wire_P00 =  Wire(1);
wire_P00 += cpu.Port("P00/INTP0");
wire_P00 += model.Port("P00");

# Connect PIN (CPU.TXD1-MODEL.RXD)
wire_RXD =  Wire(1);
wire_RXD += cpu.Port("P30/TXD1");
wire_RXD += model.Port("RXD");

# Connect PIN (CPU.RXD1-MODEL.TXD)
wire_TXD =  Wire(1);
wire_TXD += cpu.Port("P31/RXD1");
wire_TXD += model.Port("TXD");

# Connect BUS (CPU.EXTBUS-MODEL.EXTBUS1)
extbus =  BUS(32);
extbus += cpu.BusMasterIF("EXTBUS");
extbus += model.BusSlaveIF("EXTBUS1");
extbus += model.BusSlaveIF("EXTBUS2");

# Connect Pseudo PIN
wire_clock =  Wire(1);
wire_clock += cpu.DebuggerPseudoPort("debugger_pseudo_pin_main_clkout");
wire_clock += model.Port("gui_pseudo_pin_clock_notice");
wire_reset =  Wire(1);
wire_reset += cpu.DebuggerPseudoPort("debugger_pseudo_pin_reset_notice");
wire_reset += model.Port("gui_pseudo_pin_reset_notice");
```

B.4   Supplied Interface Functions

This section describes the interface functions supplied by Simulator GUI as the user open interface.

B.4.1   Overview

The supplied interface functions are listed below.

Table B-2.   List of Supplied Interface Functions

Type	Function Name	Outline of Function
Basic interface functions	SuoSetInitCallback	Registers initialization callback.
	SuoSetResetCallback	Registers reset callback.
	SuoGetMainClock	Acquires the cycle of the main clock for simulation.
Time interface functions	SuoCreateTimer	Generates timer interface.
	SuoGetTimerHandle	Acquires timer handle.
	SuoSetTimer	Sets cyclic timer.
	SuoKillTimer	Stops cyclic timer.
	SuoSetNotifyTimerCallback	Registers timer time notification callback.
Pin interface functions	SuoCreatePin	Generates pin interface.
	SuoGetPinHandle	Acquires pin interface handle.
	SuoOutputDigitalPin	Outputs digital pin value.
	SuoOutputAnalogPin	Outputs analog pin value.
	SuoOutputHighImpedance	Outputs high-impedance for the pin.
	SuoSetInputDigitalPinCallback	Registers digital pin value input callback.
	SuoSetInputAnalogPinCallback	Registers analog pin value input callback.
	SuoSetInputHighImpedanceCallback	Registers pin high-impedance state report callback.
External bus interface functionsNote	SuoCreateExtbus	Generates external bus interface.
	SuoGetExtbusHandle	Acquires external bus interface handle.
	SuoSetReadExtbusCallback	Registers external bus read access callback.
	SuoSetWriteExtbusCallback	Registers external bus write access callback.

Type	Function Name	Outline of Function
Serial interface functions	SuoCreateSerialUART	Generates serial interface (UART type).
	SuoCreateSerialCSI	Generates serial interface (CSI type).
	SuoGetSerialHandle	Acquires serial interface handle.
	SuoSetSerialParameterUART	Sets serial parameter (UART type).
	SuoSetSerialParameterCSI	Sets serial parameter (CSI type).
	SuoGetSerialParameterUART	Acquires serial parameter (UART type).
	SuoGetSerialParameterCSI	Acquires serial parameter (CSI type).
	SuoSendSerialData	Performs serial transmission (1 data).
	SuoSendSerialDataList	Performs serial transmission (more than one data).
	SuoSendSerialFile	Performs serial transmission (serial transmission data file).
	SuoSetNotifySentSerialCallback	Registers serial transmission end notification callback.
	SuoSetReceiveSerialCallback	Registers serial reception callback.
Signal output unit interface functions	SuoCreateWave	Generates signal output unit interface.
	SuoGetWaveHandle	Acquires signal output unit interface handle.
	SuoSendWaveFile	Performs transmission via signal output unit.
	SuoSetNotifySentWaveCallback	Registers signal output unit transmission end notification callback.

Note   When using the external bus interface function, the external memory area to use must be set to [Target memory area] with the [Memory Type] area of the Memory Mapping dialog box.

B.4.2   Basic interface functions

The basic interface functions that are supplied by Simulator GUI are as follows:

Function Name	Outline of Function
SuoSetInitCallback	Registers initialization callback.
SuoSetResetCallback	Registers reset callback.
SuoGetMainClock	Acquires the cycle of the main clock for simulation.

SuoSetInitCallback

Registers initialization callback.

Caution   A callback function is not executed unless this function is called in the MakeUserModel function.

[Syntax]

```
#include    "suo.h"
void    SuoSetInitCallback(SuoInitCallback func);
```

[Argument(s)]

Argument	Description
func	Pointer to the user-defined function that performs initialization processing (see "InitFunc")

[Return value]

None

[Description]

- This function registers the user-defined function that performs initialization processing.
- The function registered by this function is called only once, when Simulator GUI is started.
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include    "suo.h"
void InitFunc(void);


/* MakeUserModel */
SuoUserEntry void MakeUserModel(const char *option)
{
    .........
    SuoSetInitCallback(InitFunc);        /* Set initialize function */
}


/* Initialize function */
void InitFunc(void){
    .........
}
```

SuoSetResetCallback

Registers reset callback.

[Syntax]

```
#include    "suo.h"

void    SuoSetResetCallback(SuoResetCallback func);
```

[Argument(s)]

Argument	Description
func	Pointer to the user-defined function that performs reset processing (see "ResetFunc")

[Return value]

None

[Description]

- This function registers the user-defined function that performs reset processing.
- The registered function is called when the CPU is reset.
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include    "suo.h"

void ResetFunc(void);


void func1(void)
{
    .........
    SuoSetResetCallback(ResetFunc);        /* Set reset function */
}


/* Reset function */
void ResetFunc(void){
    .........
}
```

SuoGetMainClock

Acquires the cycle of the main clock for simulation.

Caution   This function cannot be called in the MakeUserModel function. It can only be called in a callback function.

[Syntax]

```
#include    "suo.h"
int     SuoGetMainClock(unsigned long* time);
```

[Argument(s)]

Argument	Description
time	Location where the main clock cycle value (unit: pS) is to be stored.

[Return value]

Macro	Description
SUO_NOERROR	Normal completionNormal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to acquire the cycle of the main clock for the simulation environment currently being executed.

[Example]

```
#include    "suo.h"
unsigned long time;


void func1(void)
{
    .........
    SuoGetMainClock(&time);        /* Get main clock */
}
```

B.4.3   Time interface functions

The time interface functions that are supplied by Simulator GUI are as follows:

Function Name	Outline of Function
SuoCreateTimer	Generates timer interface.
SuoGetTimerHandle	Acquires timer handle.
SuoSetTimer	Sets cyclic timer.
SuoKillTimer	Stops cyclic timer.
SuoSetNotifyTimerCallback	Registers timer time notification callback.

SuoCreateTimer

Generates timer interface.

Caution   This function can only be called in the MakeUserModel function. An error occurs if it is called at any other timing.

[Syntax]

```
#include    "suo.h"

int     SuoCreateTimer(const char* timerName, SuoHandle* handle);
```

[Argument(s)]

Argument	Description
timerName	Name of the timer
handle	Location where the handle of the timer interface is to be stored

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function generates a timer interface.
- The generated timer interface is associated with the name specified for *timerName*.
- If this function is successful, the handle of the generated timer interface can be obtained. The timer interface can then be controlled by specifying this handle.
- The handle can also be obtained by using the SuoGetTimerHandle function.

[Example]

```
#include    "suo.h"

SuoHandle hTim1;


SuoUserEntry void MakeUserModel(const char *option)
{
    .........
    SuoCreateTimer("TIM1", &hTim1);        /* Create "TIM1" */
}
```

SuoGetTimerHandle

Acquires timer handle.

[Syntax]

```
#include    "suo.h"

SuoHandle    SuoGetTimerHandle(const char* timerName);
```

[Argument(s)]

Argument	Description
timerName	Name of the timer

[Return value]

Macro	Description
Handle of the specified timer interface	Normal completion
NULL	Exit with error (abend)

[Description]

- This function is used to obtain the handle of the specified timer interface.
- Specify the name specified by the SuoCreateTimer function as *timerName* (if a different name is specified, NULL is returned).

[Example]

```
#include     "suo.h"

SuoHandle hTim1;


void func1(void)
{
    .........
    hTim1 = SuoGetTimerHandle("TIM1");        /* Get handle of "TIM1" */

}
```

SuoSetTimer

Sets cyclic timer.

Caution   This function cannot be called in the MakeUserModel function. It can only be called in a callback function.

[Syntax]

```
#include    "suo.h"
int    SuoSetTimer(SuoHandle handle, int timeUnit, unsigned long timeValue);
```

[Argument(s)]

Argument	Description
handle	Handle of the timer interface
timeUnit	Time unit (specify any of the following) - SUO_MAINCLK : Main clock cycle units - SUO_USEC : Microsecond units
timeValue	Timer cycle time

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function sets a cyclic timer for the specified timer interface.
- The cycle time is specified by the value of *timeValue* in units of *timeUnit* ("0" cannot be specified for *timeValue*.
- The timer starts operating immediately after this function is called. The timer continues operating until it is stopped by the SuoKillTimer function.
- If a timer notification function has been registered by the SuoSetNotifyTimerCallback function, the timer notification function is called in each cycle.
- If this function is called for the timer that is currently operating, the timer is reset and starts operating with the specified cycle time.

[Example]

```
#include    "suo.h"
SuoHandle hTim1;


void func1(void)
{
.........
    SuoSetTimer(hTim1, SUO_USEC, 20);     /* Invoke 20us cyclic timer */
}
```

SuoKillTimer

Stops cyclic timer.

Caution   This function cannot be called in the MakeUserModel function. It can only be called in a callback function.

[Syntax]

```
#include    "suo.h"
int     SuoKillTimer(SuoHandle handle);
```

[Argument(s)]

Argument	Description
handle	Handle of the timer interface

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function stops the cyclic timer of the specified timer interface.
- If the timer is operating, the timer is stopped. If the timer is stopped, nothing is done (in this case, an error does not occur).

[Example]

```
#include    "suo.h"
SuoHandle hTim1;


void func1(void)
{
    .........
    SuoKillTimer(hTim1);        /* Stop timer */
}
```

SuoSetNotifyTimerCallback

Registers timer time notification callback.

[Syntax]

```
#include    "suo.h"

int     SuoSetNotifyTimerCallback(SuoHandle handle, SuoNotifyTimerCallback func);
```

[Argument(s)]

Argument	Description
handle	Handle of the timer interface
func	Pointer to the user-defined function that reports the time of the timer (see "NotifyTimerFunc")

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function registers a user-defined function that performs processing when the time of the timer is reported.
- The registered function is called in every timer cycle of the specified timer interface.
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include    "suo.h"

void NotifyTimerFunc(SuoHandle handle);

SuoHandle hTim1;

void func1(void)
{
    .........
    SuoSetNotifyTimerCallback(hTim1, NotifyTimerFunc);    /* Set notify-timer function */
}


/* Notify-timer function */

void NotifyTimerFunc(SuoHandle handle)
{
    .........
}
```

B.4.4   Pin interface functions

The pin interface functions that are supplied by Simulator GUI are as follows:

Function Name	Outline of Function
SuoCreatePin	Generates pin interface.
SuoGetPinHandle	Acquires pin interface handle.
SuoOutputDigitalPin	Outputs digital pin value.
SuoOutputAnalogPin	Outputs analog pin value.
SuoOutputHighImpedance	Outputs high-impedance for the pin.
SuoSetInputDigitalPinCallback	Registers digital pin value input callback.
SuoSetInputAnalogPinCallback	Registers analog pin value input callback.
SuoSetInputHighImpedanceCallback	Registers pin high-impedance state report callback.

SuoCreatePin

Generates pin interface.

Caution   **This function can only be called in the MakeUserModel function. An error occurs if it is called at any other timing.**

[Syntax]

```
#include    "suo.h"

int     SuoCreatePin(const char* pinName, SuoHandle* handle);
```

[Argument(s)]

Argument	Description
pinName	Name of the pin
handle	Location where the handle of the pin interface is to be stored

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function generates a pin interface.
- The generated pin interface is associated with the name specified for *pinName*. Also, the pin specified as *pinName* is generated.
- If this function is successful, the handle of the generated pin interface can be obtained. The pin interface can then be controlled by specifying this handle.
  The handle can also be obtained by using the SuoGetPinHandle function.

[Example]

```
#include    "suo.h"

SuoHandle hPinP00;

SuoHandle hPinABC;


SuoUserEntry void MakeUserModel(const char *option)

{
    .........
    SuoCreatePin("P00", &hPinP00);        /* Create "P00" */
    SuoCreatePin("ABC", &hPinABC);        /* Create "ABC" */
}
```

SuoGetPinHandle

Acquires pin interface handle.

[Syntax]

```
#include    "suo.h"

SuoHandle    SuoGetPinHandle(const char* pinName);
```

[Argument(s)]

Argument	Description
pinName	Name of the pin

[Return value]

Macro	Description
Handle of the specified pin interface	Normal completion
NULL	Exit with error (abend)

[Description]

- This function is used to obtain the handle of the specified pin interface.
- Specify the name of a function specified by the SuoCreatePin function as *pinName* (if a different name is specified, NULL is returned).

[Example]

```
#include    "suo.h"

SuoHandle hPinP00;


void func1(void)
{
    .........
    hPinP00 = SuoGetPinHandle("P00");        /* Get handle of "P00" */
}
```

SuoOutputDigitalPin

Outputs digital pin value.

Caution　**This function cannot be called in the MakeUserModel function. It can only be called in a callback function.**

[Syntax]

```
#include    "suo.h"
int     SuoOutputDigitalPin(SuoHandle handle, int pinValue);
```

[Argument(s)]

Argument	Description
handle	Handle of the pin interface
pinValue	Digital output value (specify any of the following) 　- SUO_HIGH (=1):　　HIGH value 　- SUO_LOW (=0):　　LOW value

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function outputs a digital data signal specified with *pinValue* to the specified pin interface.
- To output an analog data signal, use the SuoOutputAnalogPin function.

[Example]

```
#include    "suo.h"
SuoHandle hPinP00;


void func1(void)
{
    .........
    SuoOutputDigitalPin(hPinP00, SUO_HIGH);        /* Output HIGH */
}
```

SuoOutputAnalogPin

Outputs analog pin value.

Caution　**This function cannot be called in the MakeUserModel function. It can only be called in a callback function.**

[Syntax]

```
#include    "suo.h"

int     SuoOutputAnalogPin(SuoHandle handle, double pinValue);
```

[Argument(s)]

Argument	Description
handle	Handle of the pin interface
pinValue	Analog output value (unit: V)

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function outputs an analog data signal specified with *pinValue* to the specified pin interface.
- Specify *pinValue* in V (volts), as floating-point data.
- To output a digital data signal, use the SuoOutputDigitalPin function.

[Example]

```
#include    "suo.h"

SuoHandle hPinP00;


void func1(void)
{
    .........
    SuoOutputAnalogPin(hPinP00, 3.5);        /* Output 3.5V */
}
```

SuoOutputHighImpedance

Outputs high-impedance for the pin.

Caution   This function cannot be called in the MakeUserModel function. It can only be called in a callback function.

[Syntax]

```
#include    "suo.h"

int     SuoOutputHighImpedance(SuoHandle handle);
```

[Argument(s)]

Argument	Description
handle	Handle of the pin interface

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to output high-impedance for the specified digital/analog pin interface.

[Example]

```
#include    "suo.h"

SuoHandle hPinP00;


void func1(void)
{
    .........
    SuoOutputHighImpedance(hPinP00);        /* Output High Impedance */
}
```

SuoSetInputDigitalPinCallback

Registers digital pin value input callback.

[Syntax]

```
#include    "suo.h"

int     SuoSetInputDigitalPinCallback(SuoHandle handle, SuoInputDigitalPinCallback func);
```

[Argument(s)]

Argument	Description
handle	Handle of the pin interface
func	Pointer to a user-defined function that performs digital pin input processing (see "InputDigitalPin-Func")

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8　Error numbers")

[Description]

- This function is used to register a user-defined function that performs digital pin input processing.
- The registered function is called when a signal is input to the specified pin.
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include    "suo.h"

void InputDigitalPinFunc(SuoHandle handle, int pinValue);

SuoHandle hPinP00;


void func1(void)

{

    .........

    SuoSetInputDigitalPinCallback(hPinP00, InputDigitalPinFunc); /* Set input-digital-pin function */

}


/* Input-digital-pin function */

void InputDigitalPinFunc(SuoHandle handle, int pinValue)

{

    .........

}
```

SuoSetInputAnalogPinCallback

Registers analog pin value input callback.

[Syntax]

```
#include    "suo.h"

int     SuoSetInputAnalogPinCallback(SuoHandle handle, SuoInputAnalogPinCallback func);
```

[Argument(s)]

Argument	Description
handle	Handle of the pin interface
func	Pointer to a user-defined function that performs analog pin input processing (see "InputAnalogPin-Func")

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to register a user-defined function that performs analog pin input processing.
- The registered function is called when a signal is input to the specified pin.
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include    "suo.h"

void InputAnalogPinFunc(SuoHandle handle, double pinValue);

SuoHandle hPinP00;


void func1(void)
{
    .........

    SuoSetInputAnalogPinCallback(hPinP00, InputAnalogPinFunc); /* Set input-analog-pin function */

}


/* Input-analog-pin function */

void InputAnalogPinFunc(SuoHandle handle, double pinValue)
{
    .........

}
```

SuoSetInputHighImpedanceCallback

Registers pin high-impedance state report callback.

[Syntax]

```
#include    "suo.h"

int  SuoSetInputHighImpedanceCallback(SuoHandle handle, SuoInputHighImpedanceCallback func);
```

[Argument(s)]

Argument	Description
handle	Handle of the pin interface
func	Pointer to a user-defined function that performs processing when all the connected pins enter the high-impedance state (see "InputHighImpedanceFunc")

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to register the user-defined function that performs processing when all the pins connected to digital/analog pins enter the high-impedance state.
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include    "suo.h"
void InputHighImpedanceFunc(SuoHandle handle);
SuoHandle hPinP00;

void func1(void)
{
    .........
    SuoSetInputHighImpedanceCallback(hPinP00,InputHighImpedanceFunc);

                                        /* Set input-high-impedance function */
}

/* Input-high-impedance function */
void InputHighImpedanceFunc(SuoHandle handle)
{
    .........
}
```

B.4.5   External bus interface functions

The external bus interface functions that are supplied by Simulator GUI are as follows:

Function Name	Outline of Function
SuoCreateExtbus	Generates external bus interface.
SuoGetExtbusHandle	Acquires external bus interface handle.
SuoSetReadExtbusCallback	Registers external bus read access callback.
SuoSetWriteExtbusCallback	Registers external bus write access callback.

Caution   When using the external bus interface function, the external memory area to use must be set to [Target memory area] with the [Memory Type] area of the Memory Mapping dialog box.

SuoCreateExtbus

Generates external bus interface.

Cautions 1.   This function can only be called in the MakeUserModel function. An error occurs if it is called at any other timing.

2.   When using the external bus interface function, the external memory area to use must be set to [Target memory area] with the [Memory Type] area of the Memory Mapping dialog box.

[Syntax]

```
#include    "suo.h"

int    SuoCreateExtbus(const char* extbusName, unsigned long addr, unsigned long size,
SuoHandle* handle);
```

[Argument(s)]

Argument	Description
extbusName	Name of the external bus
addr	The first address of the external memory area
size	Size of the external memory area
handle	Location where the handle of the external bus interface is to be stored

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to generate an external bus interface.
- The generated external bus interface is associated with the name specified for *extbusName*.
- If this function is successful, the handle of the generated external bus interface can be obtained. The external bus interface can then be controlled by specifying this handle.
  The handle can also be obtained by using the SuoGetExtbusHandle function.

[Example]

```
#include    "suo.h"
SuoHandle hExtbus1;


SuoUserEntry void MakeUserModel(const char *option)
{
    .........
    SuoCreateExtbus("EXTBUS1", 0x200000, 0x1000, &hExtbus1);    /* Create "EXTBUS1" */

}
```

SuoGetExtbusHandle

Acquires external bus interface handle.

[Syntax]

```
#include    "suo.h"

SuoHandle   SuoGetExtbusHandle(const char* extbusName);
```

[Argument(s)]

Argument	Description
extbusName	Name of the external bus

[Return value]

Macro	Description
Handle of the specified external bus interface	Normal completion
NULL	Exit with error (abend)

[Description]

- This function is used to obtain the handle of the specified external bus interface.
- Specify the name specified by the SuoCreateExtbus function as *extbusName* (if a different name is specified, NULL is returned).

[Example]

```
#include    "suo.h"

SuoHandle hExtbus1;


void func1(void)
{
    .........
    hExtbus1 = SuoGetExtbusHandle("EXTBUS1");    /* Get handle of "EXTBUS1" */
}
```

SuoSetReadExtbusCallback

Registers external bus read access callback.

[Syntax]

```
#include    "suo.h"

int    SuoSetReadExtbusCallback(SuoHandle handle, SuoReadExtbusCallback func);
```

[Argument(s)]

Argument	Description
handle	Handle of the external bus interface
func	Pointer to a user-defined function that performs read access processing of an external bus (see "ReadExtbusFunc")

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to register a user-defined function that performs read access processing of an external bus.
- The registered function is called if a read request is issued to the specified external bus (in the registered address range).
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include    "suo.h"

void ReadExtbusFunc(SuoHandle handle, unsigned long addr, int accessSize, unsigned char data[]);

SuoHandle hExtbus1;


void func1(void)
{
    .........

    SuoSetReadExtbusCallback(hExtbus1, ReadExtbusFunc);    /* Set read-external-bus function */
}


/* Read-external-bus function */

void ReadExtbusFunc(SuoHandle handle, unsigned long addr, int accessSize, unsigned char data[])
{
    .........
}
```

SuoSetWriteExtbusCallback

Registers external bus write access callback.

[Syntax]

```
#include    "suo.h"

int     SuoSetWriteExtbusCallback(SuoHandle handle, SuoWriteExtbusCallback func);
```

[Argument(s)]

Argument	Description
handle	Handle of the external bus interface
func	Pointer to a user-defined function that performs write access processing of an external bus (see "WriteExtbusFunc")

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to register a user-defined function that performs write access processing of an external bus.
- The registered function is called when a write request is issued to the specified external bus (in the registered address range).
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include    "suo.h"
void WriteExtbusFunc(SuoHandle handle, unsigned long addr, int accessSize, const unsigned char data[]);
SuoHandle hExtbus1;


void func1(void)
{
    .........
    SuoSetWriteExtbusCallback(hExtbus1, WriteExtbusFunc);  /* Set write-external-bus function */
}


/* Write-external-bus function */
void WriteExtbusFunc(SuoHandle handle, unsigned long addr, int accessSize, const unsigned char data[])
{
    .........
}
```

B.4.6   Serial interface functions

The serial interface functions that are supplied by Simulator GUI are as follows:

Function Name	Outline of Function
SuoCreateSerialUART	Generates serial interface (UART type).
SuoCreateSerialCSI	Generates serial interface (CSI type).
SuoGetSerialHandle	Acquires serial interface handle.
SuoSetSerialParameterUART	Sets serial parameter (UART type).
SuoSetSerialParameterCSI	Sets serial parameter (CSI type).
SuoGetSerialParameterUART	Acquires serial parameter (UART type).
SuoGetSerialParameterCSI	Acquires serial parameter (CSI type).
SuoSendSerialData	Performs serial transmission (1 data).
SuoSendSerialDataList	Performs serial transmission (more than one data).
SuoSendSerialFile	Performs serial transmission (serial transmission data file).
SuoSetNotifySentSerialCallback	Registers serial transmission end notification callback.
SuoSetReceiveSerialCallback	Registers serial reception callback.

SuoCreateSerialUART

Generates serial interface (UART type).

**Caution   This function can only be called in the MakeUserModel function. An error occurs if it is called at any
other timing.**

[Syntax]

```
#include    "suo.h"

int    SuoCreateSerialUART(const char* serialName, const char* pinNameTXD, const char*
pinNameRXD, SuoHandle* handle);
```

[Argument(s)]

Argument	Description
serialName	Name of the serial
pinNameTXD	Name of the transmit data pin used by the serial
pinNameRXD	Name of the receive data pin used by the serial
handle	Location where the handle of the serial interface is to be stored

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to generate a serial interface (UART type).
- The generated serial interface is associated with the name specified for *serialName*. In addition, pins specified as
  pinNameTXD and *pinNameRXD* are also generated.
- If this function is successful, the handle of the generated serial interface can be obtained. The serial interface can
  then be controlled by specifying this handle.
  The handle can also be obtained by using the SuoGetSerialHandle function.

[Example]

```
#include    "suo.h"
SuoHandle hUart1;


SuoUserEntry void MakeUserModel(const char *option)
{
    .........
    SuoCreateSerialUART("UART1", "TXD1", "RXD1", &hUart1);    /* Create "UART1" */

}
```

SuoCreateSerialCSI

Generates serial interface (CSI type).

Caution   **This function can only be called in the MakeUserModel function. An error occurs if it is called at any other timing.**

[Syntax]

```
#include   "suo.h"

int    SuoCreateSerialCSI(const char* serialName, const char* pinNameSO, const char*
pinNameSI, const char* pinNameSCK, SuoHandle* handle);
```

[Argument(s)]

Argument	Description
serialName	Name of the serial
pinNameSO	Name of the transmit data pin used by the serial
pinNameSI	Name of the receive data pin used by the serial
pinNameSCK	Name of the clock pin used by the serial
handle	Location where the handle of the serial interface is to be stored

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to generate a serial interface (CSI type).
- The generated serial interface is associated with the name specified for *serialName*. In addition, the pins specified as *pinNameSO*, *pinNameSI,* and *pinNameSCK* are also generated.
- If this function is successful, the handle of the generated serial interface can be obtained. The serial interface can then be controlled by specifying this handle.
  The handle can also be obtained by using the SuoGetSerialHandle function.

[Example]

```
#include    "suo.h"
SuoHandle hCsi1;


SuoUserEntry void MakeUserModel(const char *option)
{
    .........
    SuoCreateSerialCSI("CSI1", "SO1", "SI1", "SCK1", &hCsi1);    /* Create "CSI1" */

}
```

SuoGetSerialHandle

Acquires serial interface handle.

[Syntax]

```
#include    "suo.h"
SuoHandle    SuoGetSerialHandle(const char* serialName);
```

[Argument(s)]

Argument	Description
serialName	Name of the serial

[Return value]

Macro	Description
Handle of the specified serial interface	Normal completion
NULL	Exit with error (abend)

[Description]

- This function is used to obtain the handle of the specified serial interface.
- Specify the name specified by the SuoCreateSerialUART or SuoCreateSerialCSI function as *serialName* (if a different name is specified, NULL is returned).

[Example]

```
#include    "suo.h"
SuoHandle hSerial1;


void func1(void)
{
    .........
    hSerial1 = SuoGetSerialHandle("SERIAL1");    /* Get handle of "SERIAL1" */
}
```

SuoSetSerialParameterUART

Sets serial parameter (UART type).

Caution   This function cannot be called in the MakeUserModel function. It can only be called in a callback function.

[Syntax]

```
#include    "suo.h"

int     SuoSetSerialParameterUART(SuoHandle handle, const SuoSerialParameterUART* param);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
param	Location where the parameters of the serial interface (UART type) are to be stored Specify a pointer to SuoSerialParameterUART structure[Note].

Note The configuration of SuoSerialParameterUART structure is as follows:

```
typedef struct {
    unsigned long   baudrate;            /* Baud rate */
    int             direction;           /* Transfer direction */
    int             dataLength;          /* Data bit length */
    int             stopLength;          /* Stop bit length */
    int             parity;              /* Parity */
} SuoSerialParameterUART;
```

Parameter (UART Type)	Value	Description
Baud rate	*Baud rate value*	Unit: bps
Transfer direction	SUO_MSBFIRST	MSB first
	SUO_LSBFIRST	LSB first
Data bit length	1 to 32	-
Stop bit length	1 or 2	-
Parity	SUO_NONEPARITY	No parity
	SUO_ZEROPARITY	0 parity (During transmission: parity 0, During reception: no parity check)
	SUO_ODDPARITY	Odd parity
	SUO_EVENPARITY	Even parity

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to set parameters (UART type) related to the serial operation of the specified serial interface.
  The default values of the parameters are as follows:
  - Baud rate: 9600 bps
  - Transfer direction: LSB first
  - Data bit length: 7 bits
  - Stop bit length: 1 bit
  - Parity: None

[Example]

```
#include    "suo.h"
SuoHandle hUart1;


void func1(void)
{
    SuoSerialParameterUART param;
    .........
    param.baudrate   = 19200;                    /* 19200 bps */
    param.direction  = SUO_LSBFIRST;             /* LSB First */
    param.dataLength = 8;                        /* databit 8 bit */
    param.stopLength = 1;                        /* stopbit 1 bit */
    param.parity     = SUO_EVENPARITY;           /* even parity */
    SuoSetSerialParameterUART(hUart1, &param);   /* Set parameter of UART1 */
}
```

SuoSetSerialParameterCSI

Sets serial parameter (CSI type).

Caution   This function cannot be called in the MakeUserModel function. It can only be called in a callback function.

[Syntax]

```
#include   "suo.h"

int    SuoSetSerialParameterCSI(SuoHandle handle, const SuoSerialParameterCSI* param);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
param	Location where the parameters of the serial interface (CSI type) are to be stored Specify a pointer to SuoSerialParameterCSI structure[Note].

Note The configuration of SuoSerialParameterCSI structure is as follows:

```
typedef struct {
    int           mode;                  /* Operation mode */
    unsigned long  frequency;            /* Frequency of transfer clock */
    int           phase;                 /* Phase */
    int           direction;             /* Transfer direction */
    int           datalength;            /* Data bit length */
} SuoSerialParameterCSI;
```

Parameter (CSI Type)	Value	Description	
Operation mode	SUO_MASTER	Master operation	
	SUO_SLAVE	Slave operation	
Frequency of transfer clock	Frequency	Unit: Hz Note that "0" cannot be specified if master operation.	
Phase	0	Normal phase	See "Table B-3."
	SUO_PRECEDEDATA	Data output first	
	SUO_REVERSECLOCK	Clock reversal	
	SUO_PRECEDEDATA | SUO_REVERSECLOCK	Specifies both data output first and clock reversal.	
Transfer direction	SUO_MSBFIRS	MSB first	
	SUO_LSBFIRST	LSB first	
Data bit length	1 to 32	-	

Table B-3.   CSI Phase Types (SuoSetSerialParameterCSI Function)

Value of Phase	Phase
0	
SUO_PRECEDEDATA	
SUO_REVERSECLOCK	
SUO_PRECEDEDATA| SUO_REVERSECLOCK	

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to set parameters (CSI type) related to the serial operation of the specified serial interface.
  The default values of the parameters are as follows:
  - Operation mode :   Slave
  - Frequency of transfer clock : 0
  - Phase :   Normal phase
  - Transfer direction :   MSB first
  - Data bit length :   8 bits
- When operating as the master for CSI communication, transmission of dummy data is required for reception because the CSI is the communication mode that performs transmission/reception in response to the clock output from the master.

Remark   If the CSI pin waveform is checked in the Timing Chart window while CSI communication is not performed, an unexpected level will be monitored.

The expected level will be output after communication starts, so this issue does not affect the actual operation.

- SCK pin (in slave mode) : High level is output (which should be high impedance).
- SO pin :   High level is output (which should be low level).

[Example]

```
#include    "suo.h"


SuoHandle hCsi1;

void func1(void)

{

    SuoSerialParameterCSI param;

    .........

    param.mode         = SUO_SLAVE;              /* slave */

    param.frequency    = 1000000;                /* 1MHz */

    param.phase        = 0;                      /* normal */

    param.direction    = SUO_LSBFIRST;           /* LSB First */

    param.dataLength   = 8;                      /* databit 8bit */

    SuoSetSerialParameterCSI(hCsi1, &param);     /* Set parameter of CSI1 */

}
```

SuoGetSerialParameterUART

Acquires serial parameter (UART type).

Caution   This function cannot be called in the MakeUserModel function. It can only be called in a callback function.

[Syntax]

```
#include    "suo.h"
int     SuoGetSerialParameterUART(SuoHandle handle, SuoSerialParameterUART* param);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
param	Location where the parameters of the serial interface (UART type) are to be stored Specify a pointer to SuoSerialParameterUART structure[Note].

Note   For details on SuoSerialParameterUART structure, see the SuoSetSerialParameterUART function.

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

  - This function is used to obtain the parameters (UART type) related to serial operation of the specified serial interface.

[Example]

```
#include    "suo.h"
SuoHandle hUart1;

void func1(void)
{
    SuoSerialParameterUART param;
    .........
    SuoGetSerialParameterUART(hUart1, &param);    /* Get parameter of UART1 */
    .........
}
```

SuoGetSerialParameterCSI

Acquires serial parameter (CSI type).

Caution　　**This function cannot be called in the MakeUserModel function.　It can only be called in a callback function.**

[Syntax]

```
#include    "suo.h"

int    SuoGetSerialParameterCSI(SuoHandle handle, SuoSerialParameterCSI* param);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
param	Location where the parameters of the serial interface (UART type) are to be stored Specify a pointer to SuoSerialParameterCSI structureNote.

Note　For details on SuoSerialParameterCSI structure, see the SuoSetSerialParameterCSI function.

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8　Error numbers")

[Description]

- This function is used to obtain the parameters (CSI type) related to serial operation of the specified serial interface.

[Example]

```
#include    "suo.h"
SuoHandle hCsi1;


void func1(void)
{
    SuoSerialParameterCSI param;
    .........
    SuoGetSerialParameterCSI(hCsi1, &param);    /* Get parameter of CSI1 */
    .........
}
```

SuoSendSerialData

Performs serial transmission (1 data).

Cautions 1. **Be sure to use the SuoSendSerialDataList function (for serial transmission of multiple data units) when executing continuous UART transmission.**

2. **If this function is called using NotifySentSerialFunc function (serial transmission completion report callback function), transmission start is delayed for half the baud rate cycle of UART.**

3. **This function cannot be called in the MakeUserModel function. It can only be called in a call-back function.**

[Syntax]

```
#include    "suo.h"
int    SuoSendSerialData(SuoHandle handle, unsigned long data);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
data	Transmit data (1 data)

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to start transmitting one serial data.
- It takes time to complete transmitting the serial data. If you want to know the timing of transmission completion, set the transmission end notification function by using the SuoSetNotifySentSerialCallback function.
- If this function is called for a serial interface that is currently transmitting data, an error occurs.

[Example]

```
#include    "suo.h"
SuoHandle hSerial1;


void func1(void)
{
    .........
    SuoSendSerialData(hSerial1, 0x80);    /* Send 0x80 */
}
```

SuoSendSerialDataList

Performs serial transmission (more than one data).

Caution   This function cannot be called in the MakeUserModel function. It can only be called in a callback function.

[Syntax]

```
#include    "suo.h"
int     SuoSendSerialDataList(SuoHandle handle, long count, unsigned long dataList[]);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
count	Number of data to be transmitted (1 to 32767)
dataList[]	Transmit data Specify an array consisting of the number of data to be transmitted.

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to start transmitting two or more serial data.
- It takes time to complete transmission of the serial data. If you want to know the timing of transmission completion, set the transmission end notification function by using the SuoSetNotifySentSerialCallback function.
- If this function is called for a serial interface that is currently transmitting data, an error occurs.

[Example]

```
#include    "suo.h"
SuoHandle hSerial1;

void func1(void)
{
    unsigned long dataList[6] = {0x73, 0x65, 0x72, 0x69, 0x61, 0x6c};
    .........
    SuoSendSerialDataList(hSerial1, 6, dataList);        /* Send dataList */
}
```

SuoSendSerialFile

Performs serial transmission (serial transmission data file).

Caution   **This function cannot be called in the MakeUserModel function. It can only be called in a callback function.**

[Syntax]

```
#include    "suo.h"

int     SuoSendSerialFile(SuoHandle handle, const char* serialFile);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
serialFile	Name of the serial transmission data file that has been saved after being edited on the Serial window Note that if *serialFile* is specified by a relative path, it is treated as relative to the path of the user model (*UserModel.*dll).

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to start transmitting serial data described in a serial transmission data file (*.ser) that has been saved after being edited on the Serial window.
- It takes time to complete transmission of the serial data. If you want to know the timing of transmission completion, set the transmission end notification function by using the SuoSetNotifySentSerialCallback function.
- If this function is called for a serial interface that is currently transmitting data, an error occurs.

[Example]

```
#include    "suo.h"

SuoHandle hSerial1;


void func1(void)
{
    .........
    SuoSendSerialFile(hSerial1, "foo.ser");    /* Send serial data on "foo.ser" */
}
```

SuoSetNotifySentSerialCallback

Registers serial transmission end notification callback.

[Syntax]

```
#include    "suo.h"

int     SuoSetNotifySentSerialCallback(SuoHandle handle, SuoNotifySentSerialCallback func);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
func	Pointer to a user-defined function that performs processing when serial transmission is completed (see "NotifySentSerialFunc")

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to register a user-defined function that performs processing when serial transmission is completed.
- The registered function is called when one or more serial data specified to be transmitted have been completely transmitted.
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include   "suo.h"
void NotifySentSerialFunc(SuoHandle handle);
SuoHandle hSerial1;


void func1(void)
{
    .........
    SuoSetNotifySentSerialCallback(hSerial1, NotifySentSerialFunc);
                                        /* Set notify-sent-serial function */
}
/* Notify-sent-serial function */
void NotifySentSerialFunc(SuoHandle handle)
{
.........
}
```

SuoSetReceiveSerialCallback

Registers serial reception callback.

[Syntax]

```
#include    "suo.h"

int     SuoSetReceiveSerialCallback(SuoHandle handle, SuoReceiveSerialCallback func);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
func	Pointer to a user-defined function that performs processing when serial data is received (see "ReceiveSerialFunc")

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

   - This function is used to register a user-defined function that performs processing when serial data is received.
   - The registered function is called when one serial data has been received.
   - If NULL is specified for *func*, registration is canceled.

[Example]

```
#include    "suo.h"
void ReceiveSerialFunc(SuoHandle handle, unsigned long data, int status);
SuoHandle hSerial1;

void func1(void)
{
    .........
    SuoSetReceiveSerialCallback(hSerial1, ReceiveSerialFunc);
                                               /* Set receive-serial function */
}
/* Receive-serial function */
void ReceiveSerialFunc(SuoHandle handle, unsigned long data, int status)
{
    .........
}
```

B.4.7   Signal output unit interface functions

The signal output unit interface functions that are supplied by Simulator GUI are as follows:

Function Name	Outline of Function
SuoCreateWave	Generates signal output unit interface.
SuoGetWaveHandle	Acquires signal output unit interface handle.
SuoSendWaveFile	Performs transmission via signal output unit.
SuoSetNotifySentWaveCallback	Registers signal output unit transmission end notification callback.

SuoCreateWave

Generates signal output unit interface.

Caution **This function can only be called in the MakeUserModel function. An error occurs if it is called at any other timing.**

[Syntax]

```
#include    "suo.h"
int  SuoCreateWave(const char* waveName, int count, const char* pinNameList[], SuoHandle* handle);
```

[Argument(s)]

Argument	Description
waveName	Name of the signal output unit
count	Number of pins used by the signal output unit
pinNameList[]	Names of the pins used by the signal output unit Specify names in an array equivalent to the number of pins.
handle	Location where the handle of the signal output unit interface is to be stored

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to generate a signal output unit interface.
- The generated signal output unit interface is associated with the name specified for *waveName*. In addition, the pins specified by *count* and *pinNameList* are also generated.
- If this function is successful, the handle of the generated signal output unit interface can be obtained. The signal output unit interface can then be controlled by specifying this handle.
  The handle can also be obtained by using the SuoGetWaveHandle function.

[Example]

```
#include    "suo.h"
SuoHandle hWave1;


SuoUserEntry void MakeUserModel(const char *option)
{
    .........
    char* pinNameList[4] = {"P00", "P01", "P02", "P03"};
    SuoCreateWave("WAVE1", 4, pinNameList, &hWave1);        /* Create "WAVE1" */
}
```

SuoGetWaveHandle

Acquires signal output unit interface handle.

[Syntax]

```
#include    "suo.h"

SuoHandle   SuoGetWaveHandle(const char* waveName);
```

[Argument(s)]

Argument	Description
waveName	Name of the signal output unit

[Return value]

Macro	Description
Handle of the signal output unit interface	Normal completion
NULL	Exit with error (abend)

[Description]

- This function is used to obtain the handle of the specified signal output unit interface.
- Specify the name specified by the SuoCreateWave function as *waveName* (if a different name is specified, NULL is returned).

[Example]

```
#include    "suo.h"

SuoHandle hWave1;


void func1(void)
{
    .........
    hWave1 = SuoGetWaveHandle("WAVE1");        /* Get handle of "WAVE1" */
}
```

SuoSendWaveFile

Performs transmission via signal output unit.

Caution   **This function cannot be called in the MakeUserModel function. It can only be called in a callback function.**

[Syntax]

```
#include    "suo.h"

int     SuoSendWaveFile(SuoHandle handle, const char* waveFile);
```

[Argument(s)]

Argument	Description
handle	Handle of the signal output unit interface
waveFile	Name of the signal data file that has been saved after being edited on the Signal Data Editor window Note that if *waveFile* is specified by a relative path, it is treated as relative to the path of the user model (*UserModel.*dll).

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to start transmitting a signal value whose timing is described in a signal data file (*.wvi) that has been saved after being edited on the Signal Data Editor window.
- It takes time to complete transmitting the signal data file. If you want to know the timing of transmission completion, set the transmission end notification function by using the SuoSetNotifySentWaveCallback function.
- If this function is called for a signal output unit interface that is currently transmitting data, the data being transmitted is canceled and the newly specified data is transmitted.

[Example]

```
#include    "suo.h"

SuoHandle hWave1;


void func1(void)

{
    .........
    SuoSendWaveFile(hSerial1, "foo.wvi");       /* Send pin data on "foo.wvi" */

}
```

SuoSetNotifySentWaveCallback

Registers signal output unit transmission end notification callback.

[Syntax]

```
#include   "suo.h"

int     SuoSetNotifySentWaveCallback(SuoHandle handle, SuoNotifySentWaveCallback func);
```

[Argument(s)]

Argument	Description
handle	Handle of the signal output unit interface
func	Pointer to a user-defined function that performs processing when transmission by the signal output unit is completed (see "NotifySentWaveFunc")

[Return value]

Macro	Description
SUO_NOERROR	Normal completion
Error number	Exit with error (abend) (see "B.4.8 Error numbers")

[Description]

- This function is used to register a user function that performs processing when transmission by the signal output unit is completed.
- The registered function is called when all signal data specified to be transmitted have been completely transmitted.
- If NULL is specified for *func*, registration is canceled.

[Example]

```
#include   "suo.h"
void NotifySentWaveFunc(SuoHandle handle);
SuoHandle hWave1;


void func1(void)
{
    .........
    SuoSetNotifySentWaveCallback(hWave1, NotifySentWaveFunc);
                                        /* Set notify-sent-wave function */
}
/* Notify-sent-wave function */
void NotifySentWaveFunc(SuoHandle handle)
{
    .........
}
```

B.4.8　Error numbers

The meanings of error numbers (macro names) that are returned from the supplied interface function as return values are as follows:

Note that an error number is indicated by a macro name defined by the supplied header file (suo.h).

Table B-4.　List of Error Numbers (Macro Names)

Error number (Macro Name)	Description
SUO_NOERROR	Normal completion
SUO_CANTALLOC	Memory cannot be allocated.
SUO_ILLIFNAME	The interface name is not correct. NULL or "" is specified for the interface name. Or, an interface name that has not been generated is specified for a handle acquisition function.
SUO_ILLHANDLE	The handle is not correct. A handle other than that of the generated interface is specified.
SUO_ILLPARAM	The argument (parameter) is not correct. A value other than those that can be specified is specified as a argument (parameter).
SUO_CANTCALL	The function cannot be called. A function that can be called only by the MakeUserModel function is called by another function. Or, a function that can be called by a function other than the MakeUserModel function is called by the MakeUserModel function.
SUO_CONFLICTRES	The resources to be generated conflict. Two or more names that are the same as an interface name or pin name generated in the MakeUserModell function exist.
SUO_ILLFILENAME	The file name is not correct. NULL or a name including an invalid character is specified for a file name.
SUO_CANTOPENFILE	The signal data file cannot be opened. The signal data file does not exist, or is not permitted to be read.
SUO_ILLFILEFMT	[Serial transmission data file] 　The file cannot be opened. 　The file does not exist, is not permitted to be read, or the file name is not correct. [Signal data file] 　The file format is not correct. 　NULL or a name including an invalid character is specified for a file name.
SUO_ILLFILECONT	The file contents are not correct. The contents of data described in the file include a contradiction, or no data exists in the file.
SUO_ILLPINNAME	The pin name is not correct. NULL or "" is specified for the pin name.
SUO_ILLADDRRANGE	The address range is not correct. The address range is not valid.
SUO_UNDERSENDING	Already being transmitted. New transmission cannot be started because transmission is in progress.

B.5   User-Defined Functions

This section describes the user-defined functions that user creates.

The user-defined functions are listed below.

Table B-5.   List of User-Defined Functions

Function Name	Outline of Function
MakeUserModel	MakeUserModel entry function
InitFunc	Initialization callback function
ResetFunc	Reset callback function
NotifyTimerFunc	Timer time notification callback function
InputDigitalPinFunc	Digital pin input value callback function
InputAnalogPinFunc	Analog pin input value callback function
InputHighImpedanceFunc	Pin high-impedance state report callback function
ReadExtbusFunc	External bus read access callback function
WriteExtbusFunc	External bus write access callback function
NotifySentSerialFunc	Serial interface transmission end notification callback function
ReceiveSerialFunc	Serial interface reception callback function
NotifySentWaveFunc	Serial interface reception callback function

MakeUserModel

Creates the resources to be used as the entry function of the user model.

Caution   **Because MakeUserModel is a static entry function of the user model, this function name must be used.**

[Syntax]

```
#include    "suo.h"
SuoUserEntry    void MakeUserModel(const char *option);
```

[Argument(s)]

Argument	Description
option	Option character string specified in the simulator configuration file
	Note that if no option is specified in the simulator configuration file, NULL character (" ") is assumed.

[Return value]

None

[Description]

- This function must be used to generate the resources to be used with the user model. Any function other than this function cannot generate the resources.
- This function must be used to register a callback function as necessary. In particular, an initialization callback function must be registered by this function (because the initialization timing has passed even if a function is registered by a function other than this function).

[Example]

```
#include   "suo.h"
SuoHandle hTim1;
SuoHandle hPinP00;
SuoHandle hExtbus1;


void InitFunc(void);
void ResetFunc(void);


SuoUserEntry void MakeUserModel(const char *option)
{
    /* Create source */
    SuoCreateTimer("TIM1", &hTim1);                      /* Create "TIM1" */
    SuoCreatePin("P00", &hPinP00);                       /* Create "P00" */
    SuoCreateExtbus("EXTBUS1", 0x200000, 0x1000, &hExtbus1);  /* Create "EXTBUS1" */


    /* Set callbacks */
    SuoSetInitCallback(InitFunc);                        /* Set initialize function */
    SuoSetResetCallback(ResetFunc);                      /* Set reset function */
}
```

InitFunc

Performs the initialization processing as a callback function.

Caution   InitFunc is a place holder for a user-defined function name, so this function name does not have to be used.

[Syntax]

```
#include    "suo.h"
void    InitFunc (void);
```

[Argument(s)]

None

[Return value]

None

[Description]

  - InitFunc describes initialization processing.
  - Use the SuoSetInitCallback function to register InitFunc as a callback function.

ResetFunc

Performs the reset processing as a callback function.

Caution   ResetFunc is a place holder for a user-defined function name, so this function name does not have to be used.

[Syntax]

```
#include    "suo.h"
void    ResetFunc (void);
```

[Argument(s)]

None

[Return value]

None

[Description]

- ResetFunc describes the reset processing.
- Use the SuoSetResetCallback function to register ResetFunc as a callback function.

NotifyTimerFunc

Performs the processing when the timer time is reported, as a callback function.

Caution   NotifyTimerFunc is a place holder for a user-defined function name, so this function name does not have to be used.

[Syntax]

```
#include    "suo.h"
void    NotifyTimerFunc (SuoHandle handle);
```

[Argument(s)]

Argument	Description
handle	Handle of the timer interface

[Return value]

None

[Description]

- NotifyTimerFunc describes the processing when the timer time is reported.
- Use the SuoSetNotifyTimerCallback function to register NotifyTimerFunc as a callback function.

InputDigitalPinFunc

Performs the digital pin input processing, as a callback function.

Caution   InputDigitalPinFunc is a place holder for a user-defined function name, so this function name does not have to be used.

[Syntax]

```
#include   "suo.h"
void    InputDigitalPinFunc (SuoHandle handle, int pinValue);
```

[Argument(s)]

Argument	Description
handle	Handle of the pin interface
pinValue	Digital input value (specify any of the following) - SUO_HIGH (=1): HIGH value - SUO_LOW (=0): LOW value

[Return value]

None

[Description]

- InputDigitalPinFunc describes the digital pin input processing.
- Use the SuoSetInputDigitalPinCallback function to register InputDigitalPinFunc as a callback function.

InputAnalogPinFunc

Performs the analog pin input processing, as a callback function.

Caution   InputAnalogPinFunc is a place holder for a user-defined function name, so this function name does not have to be used.

[Syntax]

```
#include    "suo.h"
void    InputAnalogPinFunc (SuoHandle handle, double pinValue);
```

[Argument(s)]

Argument	Description
handle	Handle of the pin interface
pinValue	Value (analog value) input to the pin (unit: V)

[Return value]

None

[Description]

- InputAnalogPinFunc describes the analog pin input processing.
- Use the SuoSetInputAnalogPinCallback function to register InputAnalogPinFunc as a callback function.

InputHighImpedanceFunc

Performs the processing when all the pins connected to digital/analog pins enter the high-impedance state, as a callback function.

Caution   InputHighImpedanceFunc is a place holder for a user-defined function name, so this function name does not have to be used.

[Syntax]

```
#include    "suo.h"
void    InputHighImpedanceFunc (SuoHandle handle);
```

[Argument(s)]

Argument	Description
handle	Handle of the pin interface

[Return value]

None

[Description]

- InputHighImpedanceFunc is used to describe the processing when all the pins connected to digital/analog pins enter the high-impedance state.
- Use the SuoSetInputHighImpedanceCallback function to register InputHighImpedanceFunc as a callback function.

ReadExtbusFunc

Performs the read access processing of an external bus, as a callback function.

Caution   **ReadExtbusFunc is a place holder for a user-defined function name, so this function name does not have to be used.**

[Syntax]

```
#include    "suo.h"

void    ReadExtbusFunc (SuoHandle handle, unsigned long addr, int accessSize, unsigned char data[]);
```

[Argument(s)]

Argument	Description
handle	Handle of the external bus interface
addr	Address
accessSize	Access size
data[]	Data storage area As many data as the access size must be stored.

[Return value]

None

[Description]

- ReadExtbusFunc describes the read access processing of an external bus.
- Data must be stored in *data[]*.
- Use the SuoSetReadExtbusCallback function to register ReadExtbusFunc as a callback function.

WriteExtbusFunc

Performs the write access processing of an external bus, as a callback function.

Caution   WriteExtbusFunc is a place holder for a user-defined function name, so this function name does not have to be used.

[Syntax]

```
#include    "suo.h"

void    WriteExtbusFunc (SuoHandle handle, unsigned long addr, int accessSize, const unsigned char data[]);
```

[Argument(s)]

Argument	Description
handle	Handle of the external bus interface
addr	Address
accessSize	Access size
data[]	Data storage area As many data as the access size must be stored.

[Return value]

None

[Description]

- WriteExtbusFunc describes the write access processing of an external bus.
- Use the SuoSetWriteExtbusCallback function to register WriteExtbusFunc as a callback function.

NotifySentSerialFunc

Performs the processing when transmission by a serial interface has been completed, as a callback function.

Caution　　NotifySentSerialFunc is a place holder for a user-defined function name, so this function name does not have to be used.

[Syntax]

```
#include    "suo.h"
void    NotifySentSerialFunc (SuoHandle handle);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface

[Return value]

None

[Description]

- NotifySentSerialFunc describes the processing when transmission by a serial interface has been completed.
- Use the SuoSetNotifySentSerialCallback function to register NotifySentSerialFunc as a callback function.

ReceiveSerialFunc

Performs the processing during reception by a serial interface, as a callback function.

Caution   **ReceiveSerialFunc is a place holder for a user-defined function name, so this function name does not have to be used.**

[Syntax]

```
#include    "suo.h"

void    ReceiveSerialFunc (SuoHandle handle, unsigned long data, int status);
```

[Argument(s)]

Argument	Description
handle	Handle of the serial interface
data	Received serial data
status	Receive status (specify any of the following)
	- 0 : Normal reception
	- SUO_PARITYERR : Parity error (if parity bit does not match)
	- SUO_FRAMINGERR : Framing error (if stop bit is not detected)

[Return value]

None

[Description]

- ReceiveSerialFunc describes the processing during reception by a serial interface.
- Use the SuoSetReceiveSerialCallback function to register ReceiveSerialFunc as a callback function.

NotifySentWaveFunc

Performs the processing to be performed when transmission by a signal output unit has been completed, as a callback function.

Caution   NotifySentWaveFunc is a place holder for a user-defined function name, so this function name does not have to be used.

[Syntax]

```
#include    "suo.h"
void    NotifySentWaveFunc (SuoHandle handle);
```

[Argument(s)]

Argument	Description
handle	Handle of the signal output unit interface

[Return value]

None

[Description]

- NotifySentWaveFunc describes the processing to be performed when transmission by a signal output unit has been completed.
- Use the SuoSetNotifySentWaveCallback function to register NotifySentWaveFunc as a callback function.

B.6   Sample Program (Timer Model)

This section describes a sample program (timer model) of a user model created by using the Simulator GUI's user open interface.

B.6.1   Overview

Timer model is a sample program using a timer interface.

It outputs a value to a pin at fixed time intervals.

B.6.2   Configuration

The timer model generates the PIN pin and TIM timer.

The generated PIN pin is connected to the PDH pin of the CPU.

Figure B-5.   Timer Model Configuration
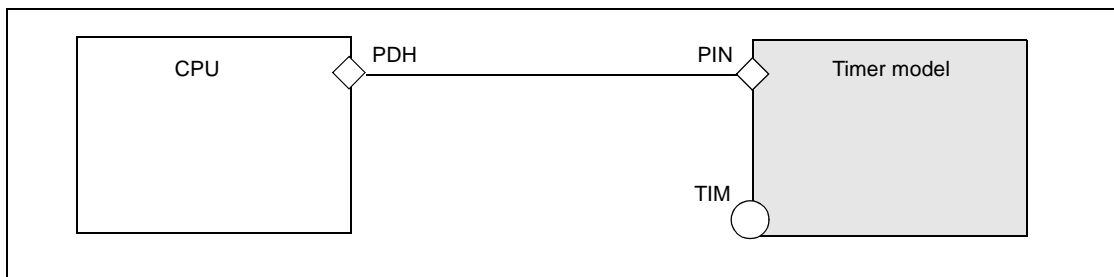
B.6.3   Operation

The timer model calculates a predetermined time by using the timer interface, and alternately outputs a low level and a high level to the PDH pin.

The output value and output time are as shown below.

Figure B-6.   Timer Model Operation
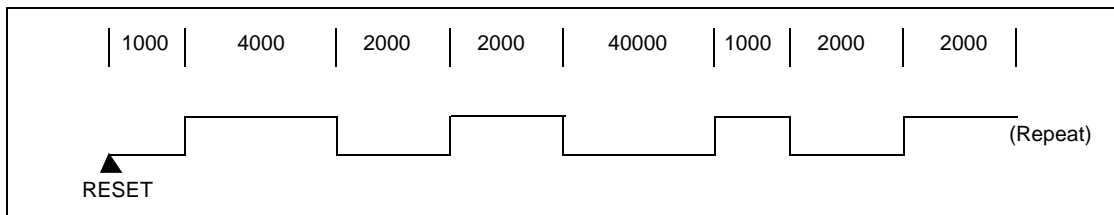
B.6.4   Project file

The following table shows the setting information of the Visual C++ project file of the Timer model.

Table B-6.   Setting Information of Timer Model

Information	Description
Project type	Win32 Dynamic-Link Library
Source file	suolink.c, uo_timer.c
Path of include file	Folder storing suo.h

B.6.5   Details of program

The following shows the sample programs of the timer model.

(1) Source file of timer model (uo_timer.c)

```c
#include <windows.h>
#include "suo.h"


/* Handle */
SuoHandle hTIM;
SuoHandle hPIN;


/* Wave-Table */
#define MAXWAVE 8
struct _WaveTable {
    unsigned longtime;              /* Wait Time [usec] */
    int pinValue;                   /* Pin Value (SUO_HIGH or SUO_LOW) */
} waveTable[MAXWAVE] = {
    1000, SUO_HIGH,
    4000, SUO_LOW,
    2000, SUO_HIGH,
    2000, SUO_LOW,
    4000, SUO_HIGH,
    1000, SUO_LOW,
    2000, SUO_HIGH,
    2000, SUO_LOW
};
int waveIndex;


/* Declare */
void Reset(void);
void NotifyTimer(SuoHandle handle);
void puterr(int error);


/* MakeUserModel */
void SuoUserEntry MakeUserModel(const char *option)
{
    int error;
    /* Create interface */
    if((error = SuoCreateTimer("TIM", &hTIM)) != SUO_NOERROR){
        puterr(error);
        return;
    }
```

```
    if((error = SuoCreatePin("PIN", &hPIN)) != SUO_NOERROR){
        puterr(error);
        return;
    }


    /* Set callback */
    SuoSetResetCallback(Reset);
    SuoSetNotifyTimerCallback(hTIM, NotifyTimer);
    }

/* Reset callback */
void Reset(void)
{
    int error;

    /* Initialize Wave-Tabel index */
    waveIndex = 0;

    /* Output LOW(initial value) to PIN */
    if((error = SuoOutputDigitalPin(hPIN, SUO_LOW)) != SUO_NOERROR){
        puterr(error);
        return;
    }


    /* Set wait time */
    if((error = SuoSetTimer(hTIM, SUO_USEC, waveTable[waveIndex].time)) != SUO_NOERROR){
        puterr(error);
        return;
    }
}


/* NotifyTimer callback */
void NotifyTimer(SuoHandle handle)
{
    int error;

    /* Output value to PIN */
    if((error = SuoOutputDigitalPin(hPIN, waveTable[waveIndex].pinValue)) != SUO_NOERROR){
        puterr(error);
        return;
    }
    /* Set next Wave-Tabel index */
    waveIndex++;
    if(waveIndex >= MAXWAVE){
        waveIndex = 0;
    }
```

```
    /* Set wait time */
    if((error = SuoSetTimer(hTIM, SUO_USEC, waveTable[waveIndex].time)) != SUO_NOERROR){
        puterr(error);
        return;
    }
}


/* Report error */
void puterr(int error)
{
    char message[80];
    wsprintf(message, "The user open interface error (0x%04x) occurred.", error);
    MessageBox(NULL, message, "ERROR", MB_OK|MB_ICONERROR);
}
```

(2) Simulator configuration file (smplus.cfg)

```
cpu = CPU('a');


#=========================================================================
# UO_TIMER description (CPU=uPD70F3261Y)
#=========================================================================


#---- Create UserOpen ----------------------------------------------------
uo_timer = Device("USEROPEN", "Release\uo_timer.dll");


#---- Pseudo Pin connection ----------------------------------------------
wire_clock =  Wire(1);
wire_clock += cpu.DebuggerPseudoPort("debugger_pseudo_pin_main_clkout");
wire_clock += uo_timer.Port("gui_pseudo_pin_clock_notice");
wire_reset =  Wire(1);
wire_reset += cpu.DebuggerPseudoPort("debugger_pseudo_pin_reset_notice");
wire_reset += uo_timer.Port("gui_pseudo_pin_reset_notice");


#---- PIN connection ----------------------------
# UO_TIMER.PIN <--> CPU.PDH0
wire_PIN =  Wire(1);
wire_PIN += uo_timer.Port("PIN");
wire_PIN += cpu.Port("PDH0");
```

(3) Source file of target program (lm_timer.c)

```
/* Target Program for UO_TIMER */

#pragma ioreg

void main( )
{
    unsigned char value;

    PMDH.0 = 1;                    /* set port-input mode */
    PMDH.1 = 0;                    /* set port-output mode */

    while( 1 ){
        value = PDH.0;         /* input signal from "PDH0" */
        PDH.1 = value;         /* output signal to "PDH1" */
    }
}
```

APPENDIX C   INDEX

Revision Record

Rev.	Date	Description	
		Page	**Summary**
1.00	Oct 01, 2010	**-**	First Edition issued

RENESAS

CubeSuite Ver.1.40