User's Manual

RENESAS

CubeSuite Ver.1.40

Integrated Development Environment
User's Manual: Build for CX Compiler

Target Device
V850 Microcontroller

Renesas Electronics
www.renesas.com

Rev.1.00 Oct 2010

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

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
CHAPTER 3 BUILD OUTPUT LISTS
APPENDIX A WINDOW REFERENCE
APPENDIX B COMMAND REFERENCE
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:	$\overline{\text{XXX}}$ (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	This manual
	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	R20UT0264E
	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.

[MEMO]

[MEMO]

[MEMO]

TABLE OF CONTENTS

CHAPTER 1 GENERAL

This chapter explains the overview of the build tool (CX).

1.1 Overview

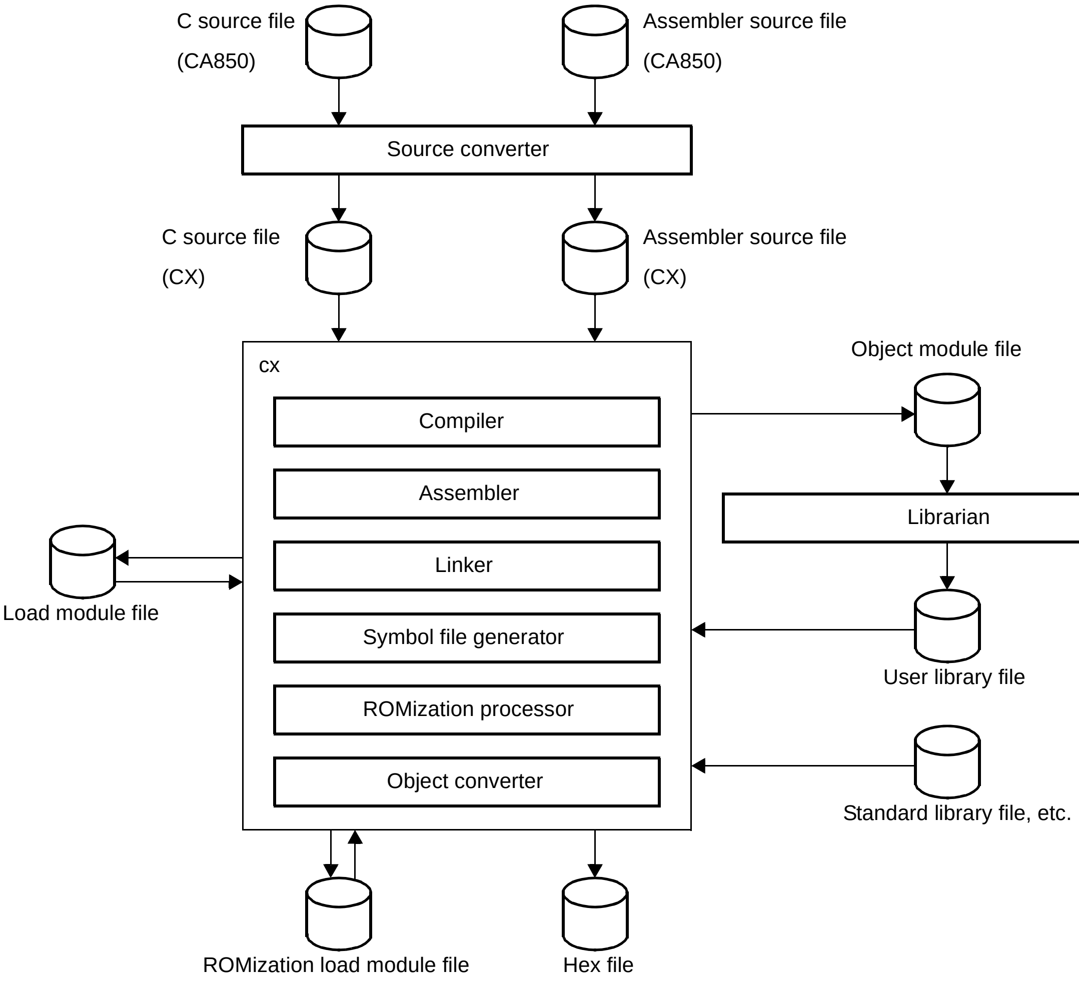
The build tool (CX) is comprised of components provided by CubeSuite. It enables various types of information to be configured via a GUI tool, enabling you to generate the load module file, ROMization load module file, hex file, and user library file from your source files, according to your objectives.

CX described by this manual consists of the following commands.

- cx
- Librarian
- Source Converter

The process flow of CX is shown below.

Figure 1-1. Process Flow of CX



1.2 Features

The features of the build tool (CX) are shown below.

- Optimization function
  You can generate efficient load module files by performing optimizations such as prioritizing code size or execution speed when compiling.

- Support for 8-byte types (double and long long)
  CX treats double as an 8-byte type.
  It also supports the long long type (signed / unsigned) specified by C99.

- Smart correction function
  If you want to correct a specific function, you can replace just the processing in that function without changing the contents of any other functions (code or addresses).

Remark See "CubeSuite Coding for CX Compiler" for details about the smart correction function.

- Linking continues after error
  If there is a memory overflow error, if CX can recover then it will continue linking, and display information about low memory.

CHAPTER 2 FUNCTIONS

This chapter describes the build procedure using CubeSuite and about the main build functions.

2.1 Overview

This section describes how to create a load module and user library.

Remark See "B.3 Source Converter" for details about conversion from CA850 files to CX files.

2.1.1 Create a load module

The procedure for creating a load module is shown below.

(1) Create or load a project

Create a new project, or load an existing one.

Remark See "CubeSuite Start" for details about creating a new project or loading an existing one.

(2) Set a build target project

Set a build target project (see "2.15 Make Settings for Build Operations").

Remarks 1. If there is no subproject in the project, the project is always active.

2. When setting a build mode, add the build mode (see "2.15.5 Add a build mode").

(3) Set build target files

Add or remove build target files and update the dependencies (see "2.3 Set Build Target Files").

Remarks 1. See "2.7.1 Add a user library" for the method of adding a user library to the project.

2. Also, you can set the link order of object module files and library files (see "2.15.1 Set the link order of files").

(4) Specify the output of a load module

Select the type of the load module to be generated (see "2.4 Set the Type of the Output File").

(5) Set build options

Set the options for the compiler, assembler, linker, and the like (see "2.5 Set Compile Options", "2.6 Set Assemble Options", "2.7 Set Link Options").

(6) Run a build

Run a build (see "2.16 Run a Build").

The following types of builds are available.

- Build (see "2.16.1 Run a build of updated files")
- Rebuild (see "2.16.2 Run a build of all files")
- Rapid build (see "2.16.3 Run a build in parallel with other operations")
- Batch build (see "2.16.4 Run builds in batch with build modes")

Remark If there are any commands you wish to run before or after the build process, on the Property panel, from the [Common Options] tab, in the [Others] category, set the [Commands executed before build processing] and [Commands executed after build processing] properties.

If there are any commands you wish to run before or after the build process at the file level, you can set them from the [Individual Compile Options] tab (for a C source file) and [Individual Assemble Options] tab (for an assembler source file).

(7) Save the project

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

Remark See "CubeSuite Start" for details about saving the project.

2.1.2 Create a user library

The procedure for creating a user library is shown below.

(1) Create or load a project

Create a new project, or load an existing one.
When you create a new project, set a library project.

Remark See "CubeSuite Start" for details about creating a new project or loading an existing one.

(2) Set a build target project

Set a build target project (see "2.15 Make Settings for Build Operations").

Remarks 1. If there is no subproject in the project, the project is always active.

2. When setting a build mode, add the build mode (see "2.15.5 Add a build mode").

(3) Set build target files

Add or remove build target files and update the dependencies (see "2.3 Set Build Target Files").

(4) Set build options

Set the options for the compiler, assembler, librarian, and the like (see "2.5 Set Compile Options", "2.6 Set Assemble Options", "2.10 Set Create Library Options").

Remark To create a library common to various devices, set the [Output common object module file for various devices] property in the [Output File Type and Path] category from the [Common Options] tab on the Property panel.

(5) Run a build

Run a build (see "2.16 Run a Build").
The following types of builds are available.

- Build (see "2.16.1 Run a build of updated files")
- Rebuild (see "2.16.2 Run a build of all files")
- Rapid build (see "2.16.3 Run a build in parallel with other operations")
- Batch build (see "2.16.4 Run builds in batch with build modes")

Remark If there are any commands you wish to run before or after the build process, on the Property panel, from the [Common Options] tab, in the [Others] category, set the [Commands executed before build processing] and [Commands executed after build processing] properties.
If there are any commands you wish to run before or after the build process at the file level, you can set them from the [Individual Compile Options] tab (for a C source file) and [Individual Assemble Options] tab (for an assembler source file).

(6) Save the project

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

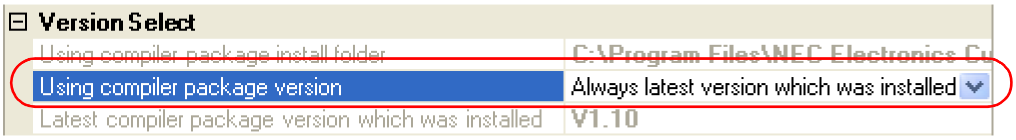
Remark See "CubeSuite Start" for details about saving the project.

2.2 Change the Build Tool Version

You can change the version of the build tool (compiler package) used in the project (main project or subproject).

Select the build tool node on the project tree and select the [Common Options] tab on the Property panel. Select [Always latest version which was installed] or the version in the [Using compiler package version] property in the [Version Select] category.

Figure 2-1. [Using compiler package version] Property



⊟ Version Select

Using compiler package install folder	C:\Program Files\NEC Electronics Cu
Using compiler package version	Always latest version which was installed
Latest compiler package version which was installed	V1.10

Remarks 1. When the build tool used in the main project and subprojects is the same, you can collectively change the build tool version by selecting all of the Build tool nodes and setting the property.

2. If you have selected a compiler package that has not been installed (e.g. if you open a project created in another execution environment), then that version is also displayed.

3. If the options change depending on the compiler package, then the display of the build tool's properties will change according to the selected version.

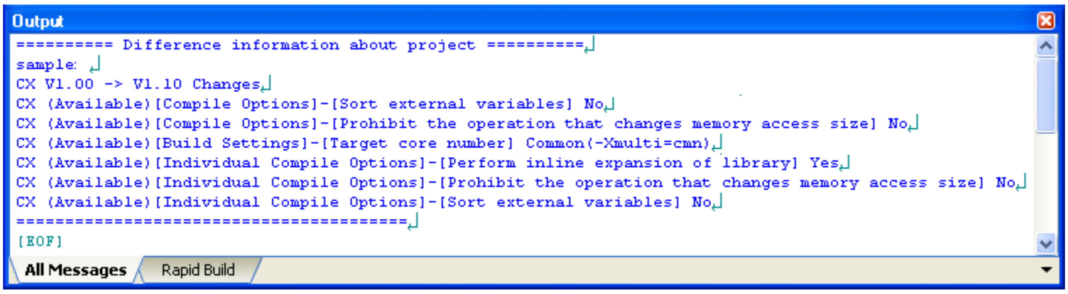
Properties that are hidden when the version is changed are saved in the project file's settings, and the values will be reproduced when the properties are displayed again.

Options are changed in accordance with the following rules. Information about changes is displayed in the Output panel.

- If you change from an older version to a newer version, the option settings will be inherited and converted (only if necessary).
- If you change from a newer version to an older version, only identical option settings will be inherited.

  Options that only exist in the older version will be set to the default values.

Figure 2-2. Output Image of Information about Changed Options



Output

```
========== Difference information about project ==========↵
sample: ↵
CX V1.00 -> V1.10 Changes↵
CX (Available)[Compile Options]-[Sort external variables] No↵
CX (Available)[Compile Options]-[Prohibit the operation that changes memory access size] No↵
CX (Available)[Build Settings]-[Target core number] Common(-Xmulti=cmn)↵
CX (Available)[Individual Compile Options]-[Perform inline expansion of library] Yes↵
CX (Available)[Individual Compile Options]-[Prohibit the operation that changes memory access size] No↵
CX (Available)[Individual Compile Options]-[Sort external variables] No↵
========================================↵
[EOF]
```

2.3 Set Build Target Files

Before running a build, you must add the build target files (such as C source file, assembler source file) to the project. This section explains operations on setting files in the project.

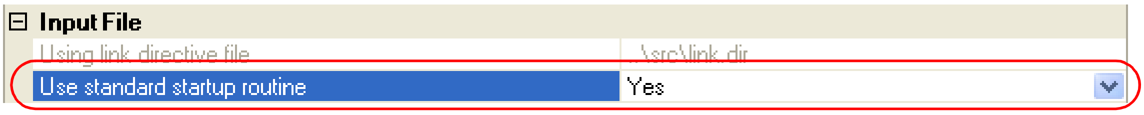
2.3.1 Set a startup routine

(1) Using the standard startup routine

Select the build tool node on the project tree and select the [Link Options] tab on the Property panel.

To use the standard startup routine (object module file provided with CX), select [Yes] in the [Use standard startup routine] property in the [Input File] category.

Figure 2-3. [Use standard startup routine] Property



⊟ Input File

Using link directive file .\src\link.dir

Use standard startup routine Yes

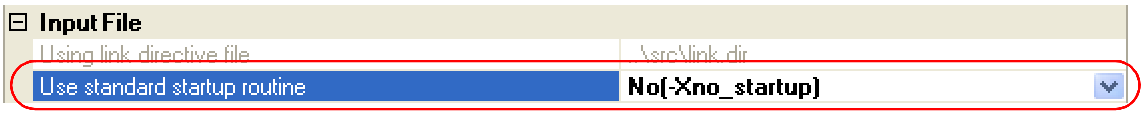
Remark In the [Output ROMized load module file] property in the [Output File] category from the [ROMize Options] tab, "cstart.obj" is linked when [Yes] is selected and "cstartN.obj" is linked when [No(-Xno_romize)] is selected.

(2) Using other than the standard startup routine

Select the build tool node on the project tree and select the [Link Options] tab on the Property panel.

To use other than the standard startup routine, select [No(-Xno_startup)] in the [Use standard startup routine] property in the [Input File] category (default: [Yes]).

Figure 2-4. [Use standard startup routine] Property



⊟ Input File

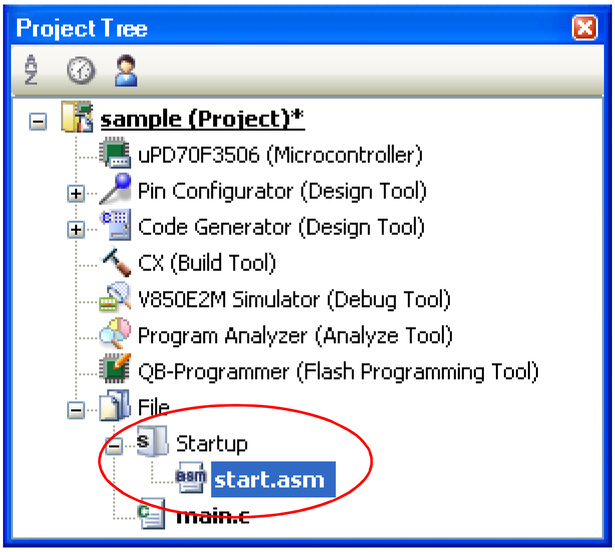
Using link directive file .\src\link.dir

Use standard startup routine No(-Xno_startup)

Next, add the startup file (the file that the startup routine is described) to the Startup node on the project tree.

Remark See "2.3.3 Add a file to a project" for the method of adding the file to the project tree.

Figure 2-5. Project Tree Panel (After Adding Startup File)


Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Startup
start.asm
main.c

Caution **The build target file added directly below the Startup node on the project tree is treated as the startup file.**

It is not treated as a startup file if it is added to the category below the Startup node.

When adding the startup file to the Startup node, if a startup file has already been added then only the latest startup file to be added is targeted by a build; any such files added prior to this one will not be targeted.

When setting the startup file that is not targeted by a build as the build target, if other startup files have also been added then the file will be targeted by the build, and the others will not be targeted.

Remark See "CubeSuite Coding for CX Compiler" for the method of creating the startup routine.

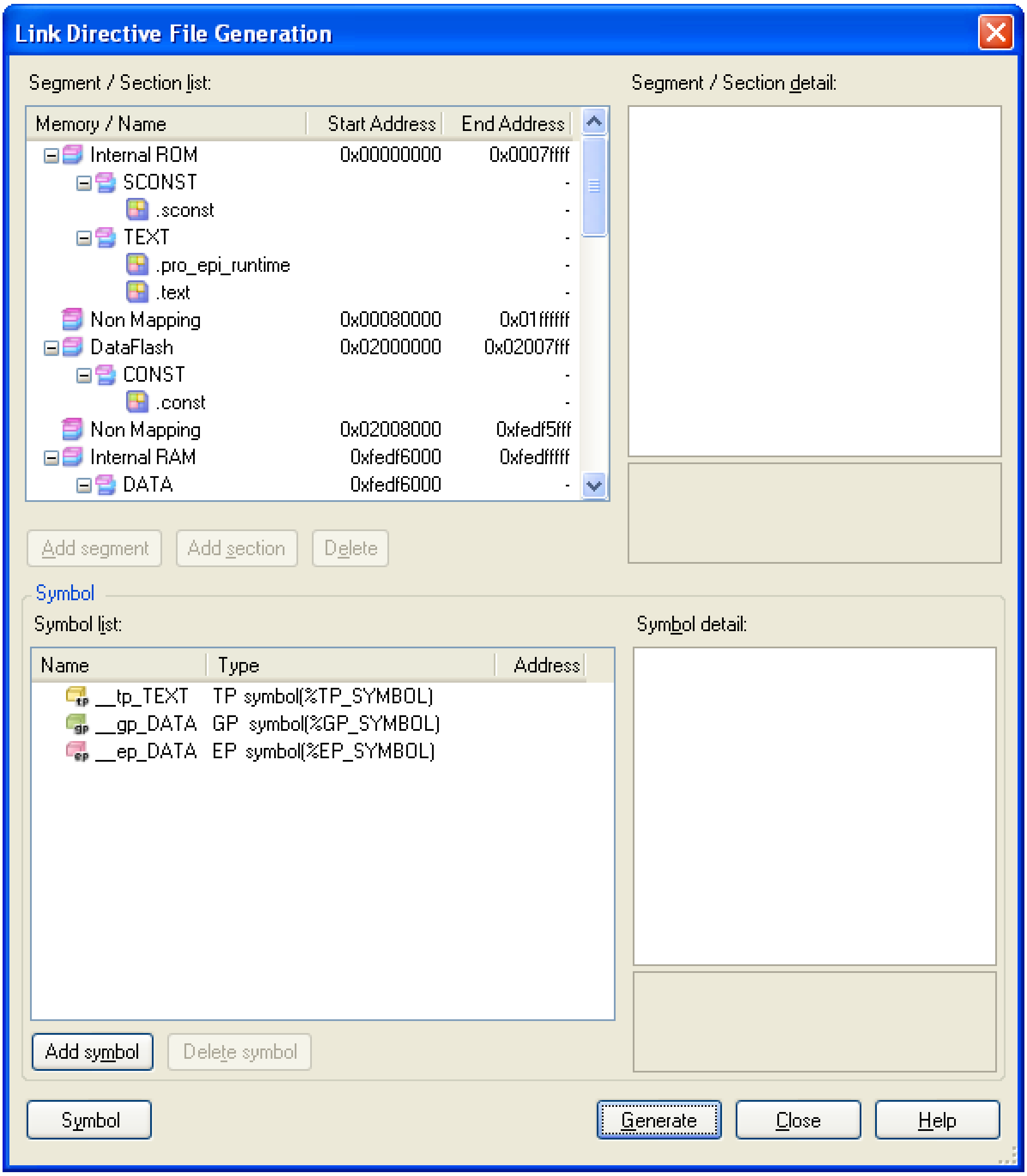
2.3.2 Automatically generate link directives

Although users can create a link directive file and add it to a project, it is also possible to generate it automatically in CubeSuite.

Remark See "CubeSuite Coding for CX Compiler" for details about link directives and creating a link directive file.

On the project tree, select the Build tool node, and then select [Create Link Directive File...] from the context menu. The Link Directive File Generation dialog box opens.

Figure 2-6. Link Directive File Generation Dialog Box



Memory / Name	Start Address	End Address
Internal ROM	0x00000000	0x0007ffff
SCONST		-
.sconst		-
TEXT		-
.pro_epi_runtime		-
.text		-
Non Mapping	0x00080000	0x01ffffff
DataFlash	0x02000000	0x02007fff
CONST		-
.const		-
Non Mapping	0x02008000	0xfedf5fff
Internal RAM	0xfedf6000	0xfedfffff
DATA	0xfedf6000	-

Name	Type	Address
__tp_TEXT	TP symbol(%TP_SYMBOL)	
__gp_DATA	GP symbol(%GP_SYMBOL)	
__ep_DATA	EP symbol(%EP_SYMBOL)	

Edit the segments/sections and symbols in the dialog box.

(1) Edit segments/sections

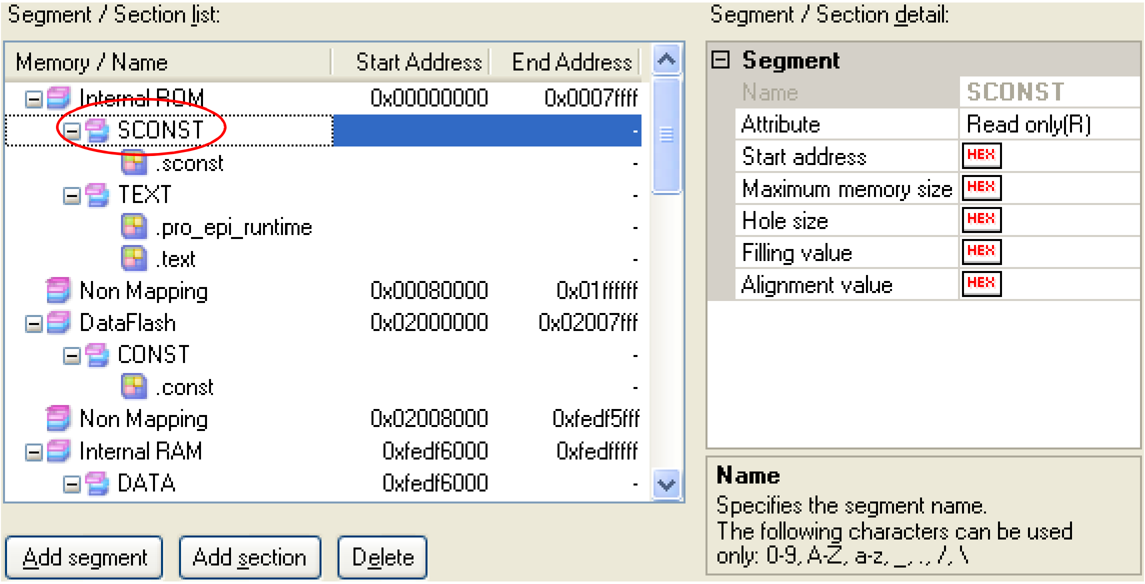
The [Segment / Section list] area displays the device memory allocation information, and a list of the currently configured segments and sections.

When a segment/section is selected from the list, detailed information on that segment/section is displayed in the [Segment/Section detail] area.

Edit the items in the [Segment / Section detail] area.

Remark Some items in reserved sections cannot be edited (items for which values are set automatically). See "APPENDIX A WINDOW REFERENCE", "Link Directive File Generation dialog box" for details about each item and how reserved sections are handled.

Figure 2-7. Segment Detail (When SCONST Is Selected)

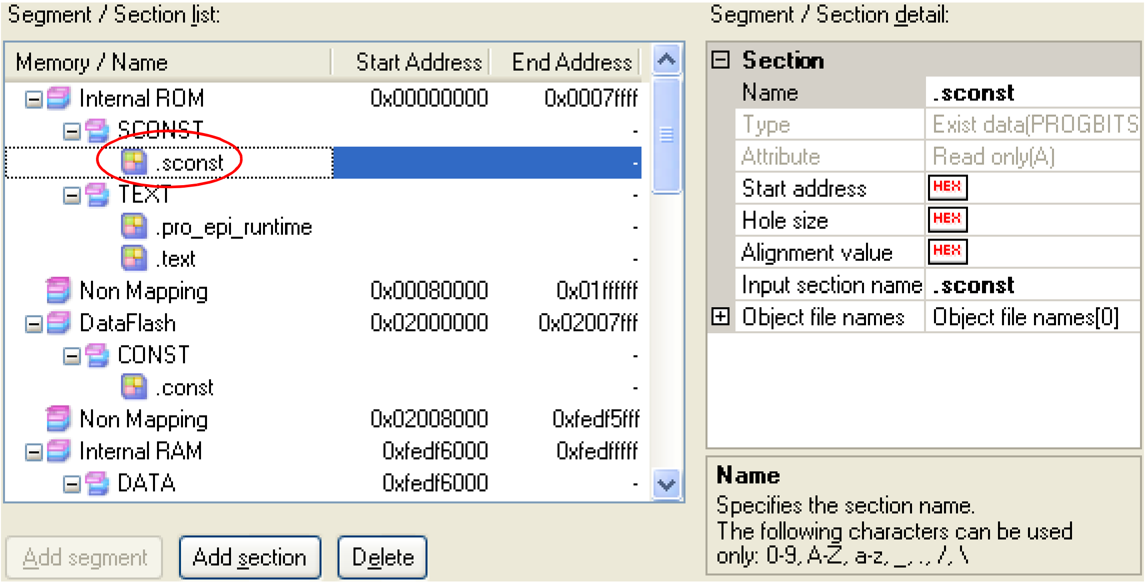


Memory / Name	Start Address	End Address
⊟ Internal ROM	0x00000000	0x0007ffff
⊟ SCONST		-
.sconst		-
⊟ TEXT		-
.pro_epi_runtime		-
.text		-
Non Mapping	0x00080000	0x01ffffff
⊟ DataFlash	0x02000000	0x02007fff
⊟ CONST		-
.const		-
Non Mapping	0x02008000	0xfedf5fff
⊟ Internal RAM	0xfedf6000	0xfedfffff
⊟ DATA	0xfedf6000	-

⊟ Segment	
Name	SCONST
Attribute	Read only(R)
Start address	HEX
Maximum memory size	HEX
Hole size	HEX
Filling value	HEX
Alignment value	HEX

Name
Specifies the segment name.
The following characters can be used only: 0-9, A-Z, a-z, _, ., /, \

Figure 2-8. Section Detail (When .sconst Is Selected)



Memory / Name	Start Address	End Address
⊟ Internal ROM	0x00000000	0x0007ffff
⊟ SCONST		-
.sconst		-
⊟ TEXT		-
.pro_epi_runtime		-
.text		-
Non Mapping	0x00080000	0x01ffffff
⊟ DataFlash	0x02000000	0x02007fff
⊟ CONST		-
.const		-
Non Mapping	0x02008000	0xfedf5fff
⊟ Internal RAM	0xfedf6000	0xfedfffff
⊟ DATA	0xfedf6000	-

⊟ Section	
Name	.sconst
Type	Exist data(PROGBITS
Attribute	Read only(A)
Start address	HEX
Hole size	HEX
Alignment value	HEX
Input section name	.sconst
⊞ Object file names	Object file names[0]

Name
Specifies the section name.
The following characters can be used only: 0-9, A-Z, a-z, _, ., /, \

Segments and sections can also be added.

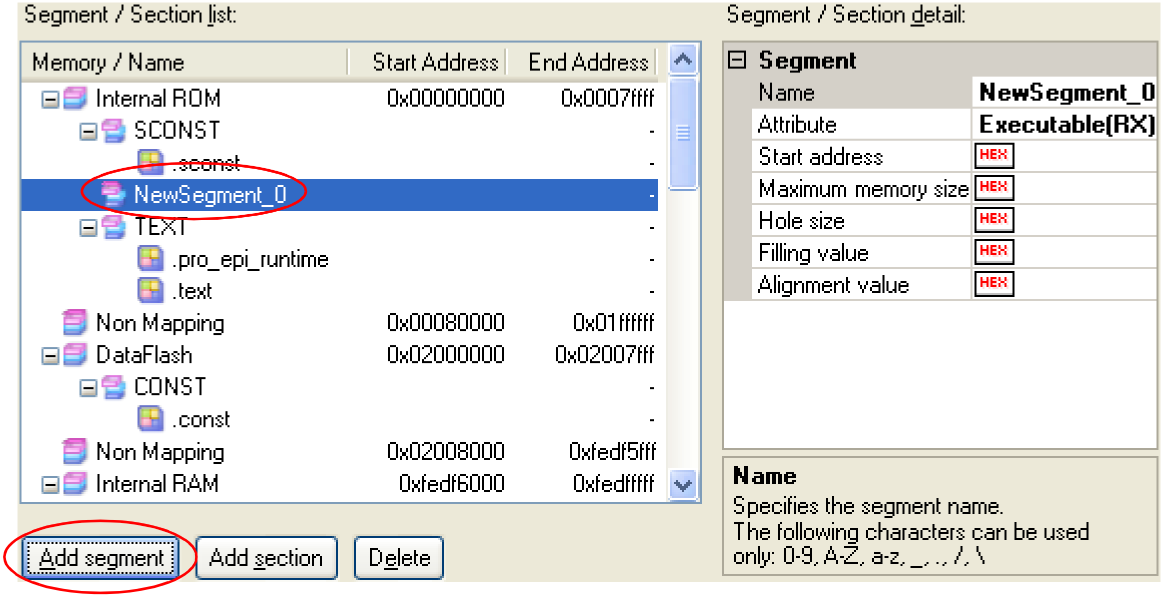
Click [Add Segment] to add a new segment "NewSegment_*XXX*" directly below the row selected in the list (*XXX*: 0 to 255 in decimal numbers).

Edit the items in the [Segment / Section detail] area.

By default, [Attribute] is set to [Executable(RX)] (if added to the internal ROM area or non mapping area), to [Read/ Write(RW)] (if added to the internal RAM area), or to [Read only(R)] (if added to the DataFlash area).

Caution When a section row is selected in the list, the [Add Segment] button is invalid.

Figure 2-9. Add Segment



Segment / Section list:

Memory / Name	Start Address	End Address
⊟ Internal ROM	0x00000000	0x0007ffff
⊟ SCONST		-
.sconst		-
NewSegment_0		-
⊟ TEXT		-
.pro_epi_runtime		-
.text		-
Non Mapping	0x00080000	0x01ffffff
⊟ DataFlash	0x02000000	0x02007fff
⊟ CONST		-
.const		-
Non Mapping	0x02008000	0xfedf5fff
⊟ Internal RAM	0xfedf6000	0xfedfffff

Segment / Section detail:

⊟ Segment	
Name	NewSegment_0
Attribute	Executable(RX)
Start address	HEX
Maximum memory size	HEX
Hole size	HEX
Filling value	HEX
Alignment value	HEX

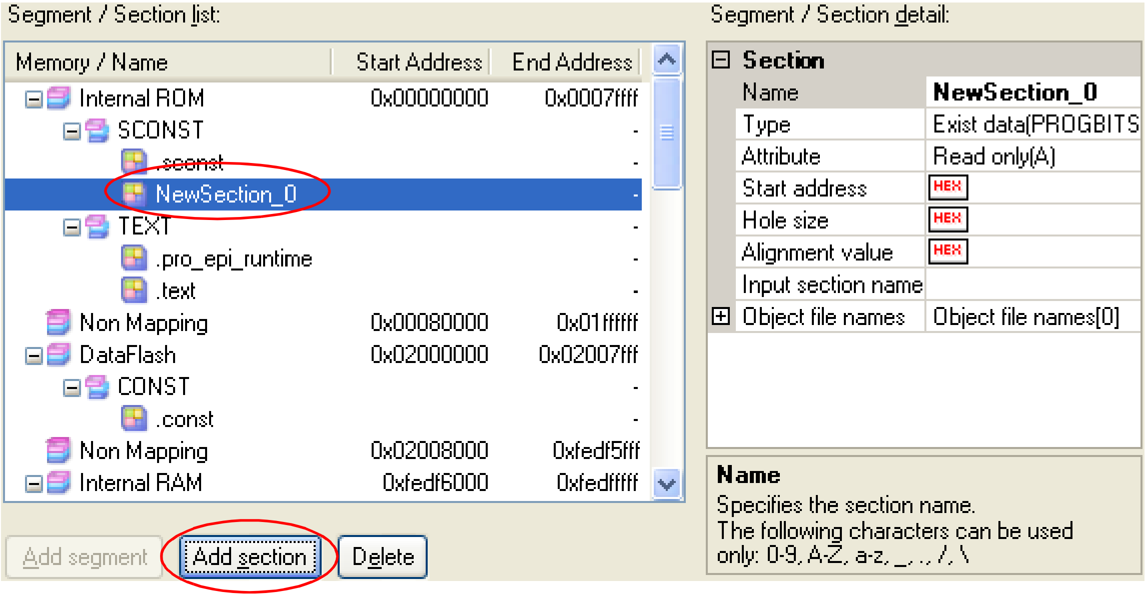
Name
Specifies the segment name.
The following characters can be used
only: 0-9, A-Z, a-z, _, ., /, \

Click [Add Section] to add a new section "NewSection_*XXX*" directly below the row selected in the list (*XXX*: 0 to 255 in decimal numbers).

Edit the items in the [Segment / Section detail] area.

By default, [Type] is set to [Exist data (PROGBITS)], and [Attribute] inherits the value of the parent segment.

Figure 2-10. Add Section



Segment / Section list:

Memory / Name	Start Address	End Address
⊟ Internal ROM	0x00000000	0x0007ffff
⊟ SCONST		-
.sconst		-
NewSection_0		-
⊟ TEXT		-
.pro_epi_runtime		-
.text		-
Non Mapping	0x00080000	0x01ffffff
⊟ DataFlash	0x02000000	0x02007fff
⊟ CONST		-
.const		-
Non Mapping	0x02008000	0xfedf5fff
⊟ Internal RAM	0xfedf6000	0xfedfffff

Segment / Section detail:

⊟ Section	
Name	NewSection_0
Type	Exist data(PROGBITS
Attribute	Read only(A)
Start address	HEX
Hole size	HEX
Alignment value	HEX
Input section name	
⊞ Object file names	Object file names[0]

Name
Specifies the section name.
The following characters can be used
only: 0-9, A-Z, a-z, _, ., /, \

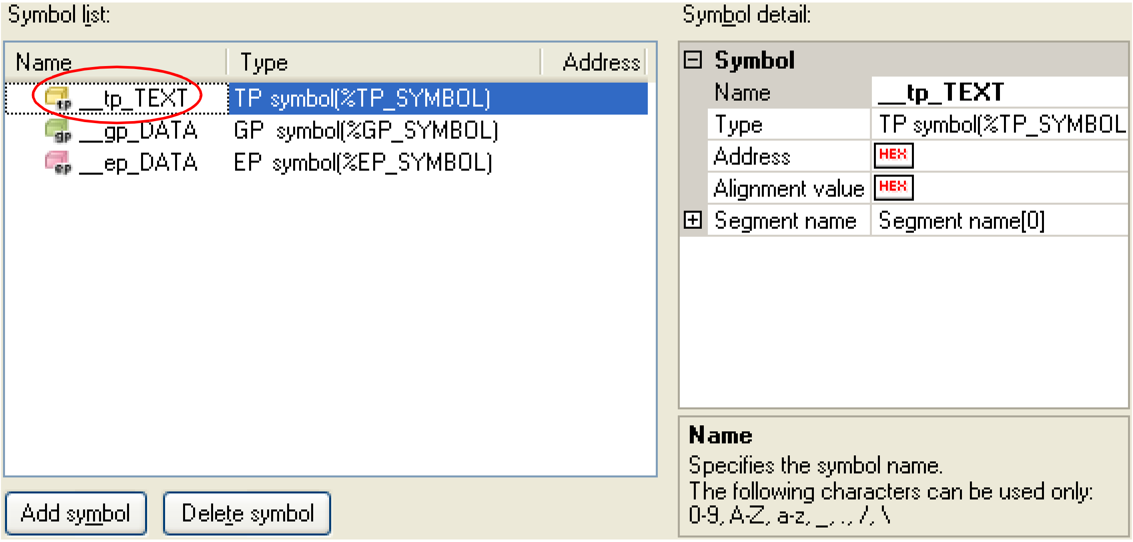
(2) Edit symbols

The [Symbol list] area displays the list of currently configured symbols.

When a symbol is selected from the list, detailed information on that symbol is displayed in the [Symbol detail] area.

Edit the items in the [Symbol detail] area.

Figure 2-11. Segment Detail (When _tp_TEXT Is Selected)



Symbol list:

Name	Type	Address
__tp_TEXT	TP symbol(%TP_SYMBOL)	
__gp_DATA	GP symbol(%GP_SYMBOL)	
__ep_DATA	EP symbol(%EP_SYMBOL)	

Symbol detail:

⊟ Symbol	
Name	__tp_TEXT
Type	TP symbol(%TP_SYMBOL
Address	HEX
Alignment value	HEX
⊞ Segment name	Segment name[0]

Name
Specifies the symbol name.
The following characters can be used only:
0-9, A-Z, a-z, _, ., /, \

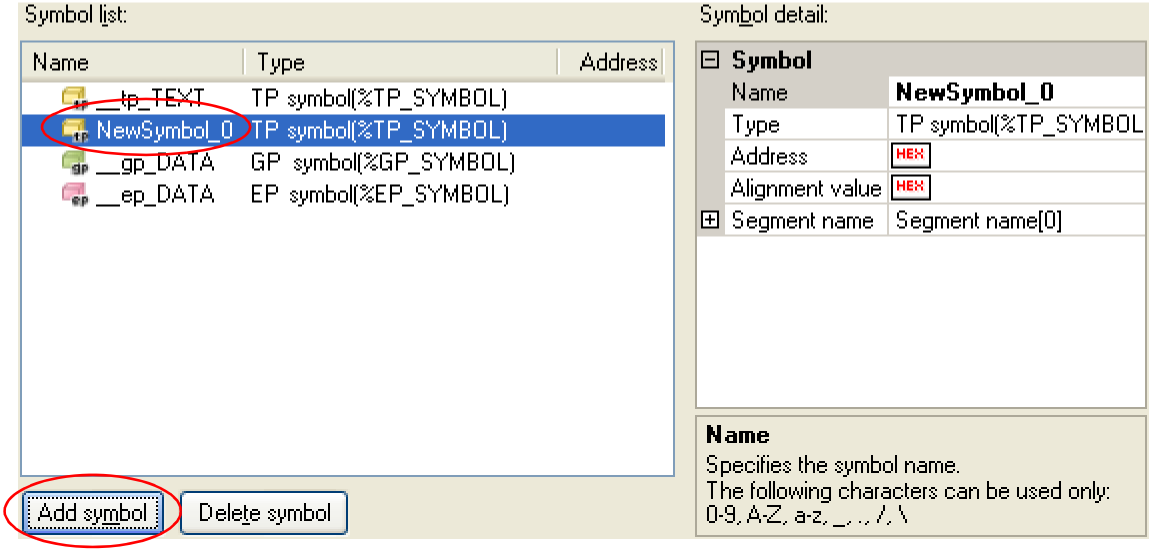
Symbols can also be added.

Click [Add symbol] to add a new symbol "NewSymbol_*XXX*" directly below the row selected in the list (*XXX*: 0 to 255 in decimal numbers).

Edit the items in the [Symbol detail] area.

By default, [Type] is set to [TP symbol(%TP_SYMBOL)].

Figure 2-12. Add Symbol



Symbol list:

Name	Type	Address
__tp_TEXT	TP symbol(%TP_SYMBOL)	
NewSymbol_0	TP symbol(%TP_SYMBOL)	
__gp_DATA	GP symbol(%GP_SYMBOL)	
__ep_DATA	EP symbol(%EP_SYMBOL)	

Symbol detail:

⊟ Symbol	
Name	NewSymbol_0
Type	TP symbol(%TP_SYMBOL
Address	HEX
Alignment value	HEX
⊞ Segment name	Segment name[0]

Name
Specifies the symbol name.
The following characters can be used only:
0-9, A-Z, a-z, _, ., /, \

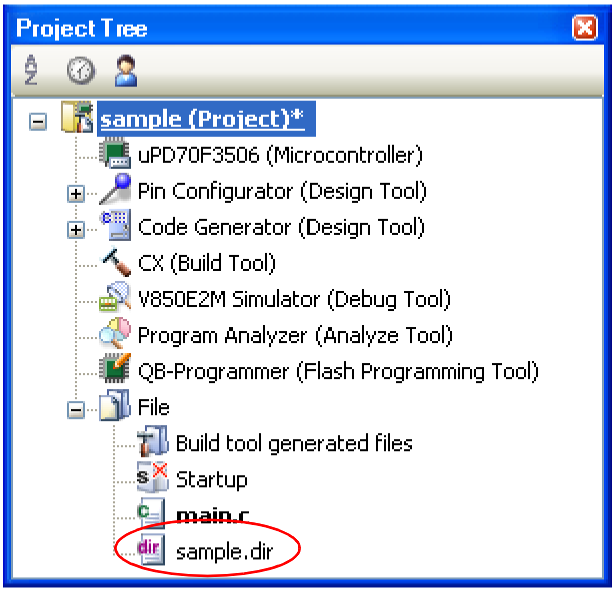
After editing the segments/sections and symbols, click the [Generate] button.

A link directive file (named *project-name*.dir) is generated based on the specified memory, segments, sections, and symbol allocation information, and then added to the project.

The link directive file is generated in the project folder.

The link directive file that has been generated is also shown on the project tree, under the File node.

Figure 2-13. Project Tree Panel (After Generating Link Directive File)



Project Tree

- sample (Project)*
  - uPD70F3506 (Microcontroller)
  - Pin Configurator (Design Tool)
  - Code Generator (Design Tool)
  - CX (Build Tool)
  - V850E2M Simulator (Debug Tool)
  - Program Analyzer (Analyze Tool)
  - QB-Programmer (Flash Programming Tool)
  - File
    - Build tool generated files
    - Startup
    - main.c
    - sample.dir

Caution **The generated link directive file will be a build target.**
If a link directive file has already been registered to the project, then the file will be removed from the build target.

2.3.3 Add a file to a project

Files can be added to a project by the following methods.

- Adding an existing file
- Creating and adding an empty file

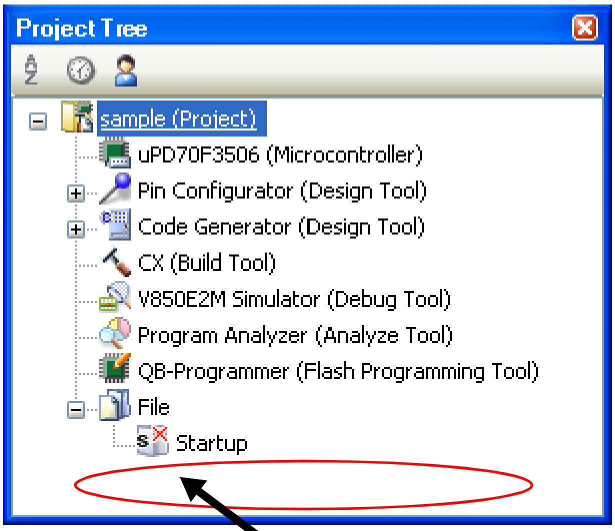
(1) Adding an existing file

(a) Add an individual files

Drag the file from Explorer or the like, and drop it onto the empty space below the project tree.

The file is added below the File node.

Figure 2-14. Project Tree Panel (File Drop Location)


Project Tree
sample (Project)
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Startup

Drop the file here

Caution To add other than the standard startup routine, drop the file onto the Startup node. See "2.3.1 Set a startup routine" for details about using other than the standard startup routine.

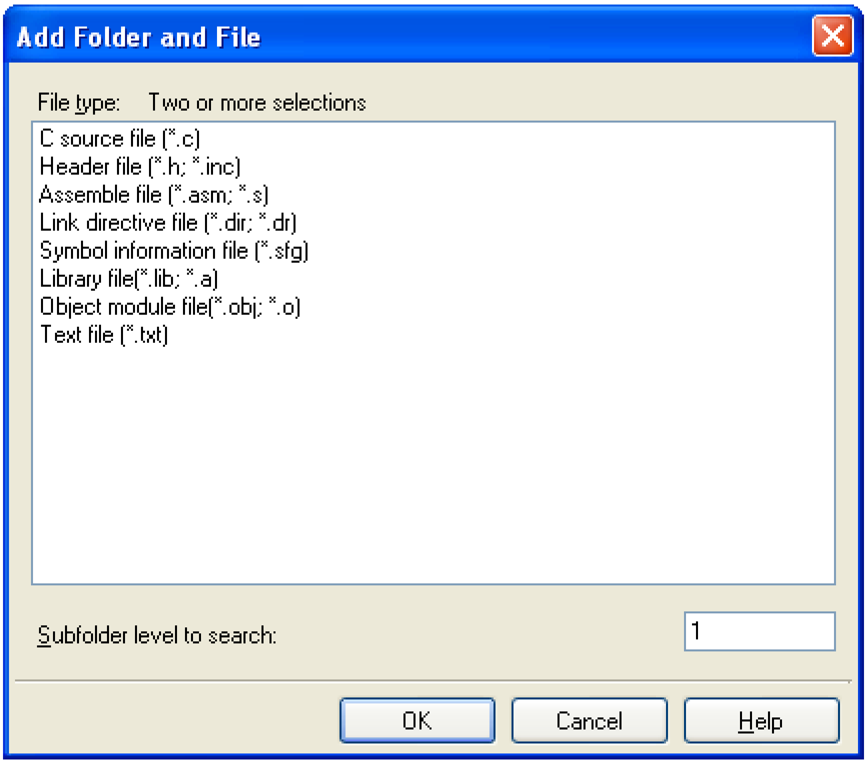
(b) Add a folder

Drag the folder from Explorer or the like, and drop it onto the empty space below the project tree. The Add Folder And File dialog box will open.

Remark You can also add multiple folders to the project at the same time by dragging multiple folders at same time and dropping them onto the project tree.

Caution When the folder with the name that is more than 200 characters is dropped, the folder is added to the project tree as a category with the name that 201st character and after are deleted.

Figure 2-15. Add Folder and File Dialog Box


Add Folder and File
File type: Two or more selections
C source file (*.c)
Header file (*.h; *.inc)
Assemble file (*.asm; *.s)
Link directive file (*.dir; *.dr)
Symbol information file (*.sfg)
Library file(*.lib; *.a)
Object module file(*.obj; *.o)
Text file (*.txt)
Subfolder level to search: 1
OK
Cancel
Help

In the dialog box, select the types of the files to be added to the project and specify the number of levels of the subfolder to be added to the project. And then click the [OK] button.

Remark You can select multiple file types by left clicking while holding down the [Ctrl] or [Shift] key. If nothing is selected, it is assumed that all types are selected.

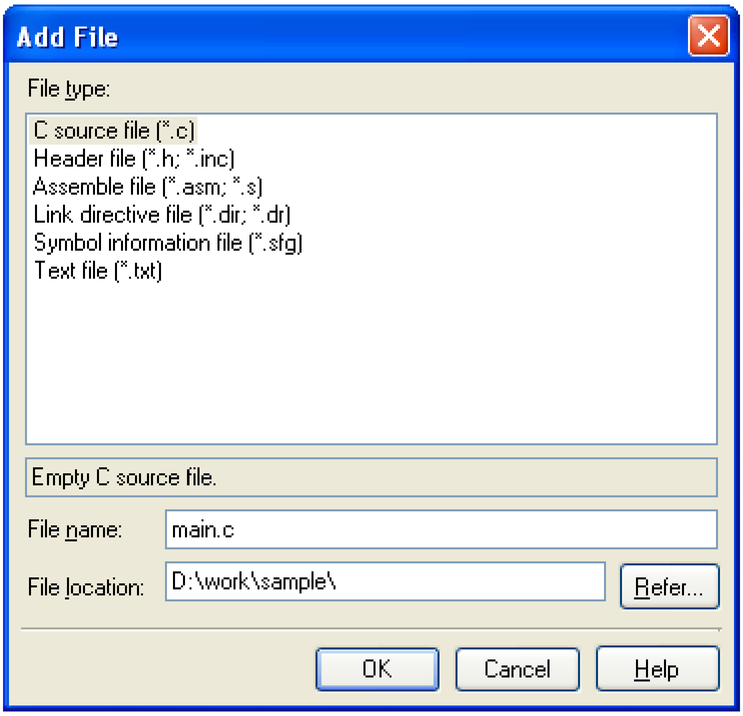
The folder is added below the File node.
Note that on the project tree, the folder is the category.

Remark When the category node created by the user exists, you can add a file below the node by dropping the file onto the node (see "2.3.6 Classify a file into a category" for a category node).

(2) Creating and adding an empty file

On the project tree, select either one of the Project node, Subproject node, or File node, and then select [Add] >> [Add New File...] from the context menu. The Add File dialog box will open.

Figure 2-16. Add File Dialog Box



Add File

File type:

C source file (*.c)
Header file (*.h; *.inc)
Assemble file (*.asm; *.s)
Link directive file (*.dir; *.dr)
Symbol information file (*.sfg)
Text file (*.txt)

Empty C source file.

File name: main.c

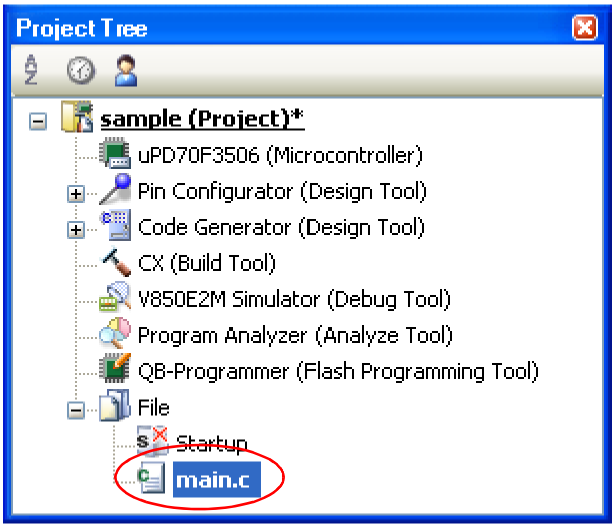
File location: D:\work\sample\

In the dialog box, specify the file to be created and then click the [OK] button.

The file is added below the File node.

The project tree after adding the file and folder will look like the one below.

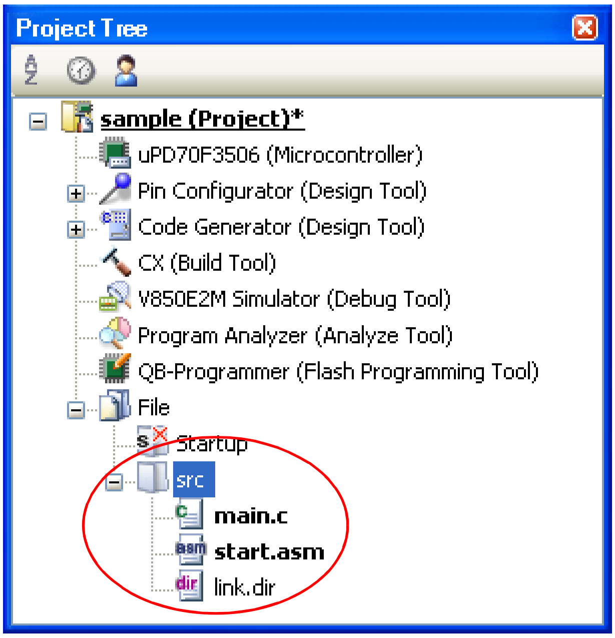
Figure 2-17. Project Tree Panel (After Adding File "main.c")



Project Tree

sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Startup
main.c

Figure 2-18. Project Tree Panel (After Adding Folder "src")


Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Startup
src
main.c
start.asm
link.dir

Remark The location of the file added below the File node depends on the current file display order setting. See “2.3.7 Change the file display order” for the method of changing the file display order.

Cautions 1. If the paths differ, you can add source files with the same name.
Note, however, that if the setting of the output file name is left as the default, the output files will have the same name, which will prevent the build from running correctly (for example, when adding D:\sample1\func.c and D:\sample2\func.c, the default output file name for these files is both func.obj).
The target file cannot open during debugging.
To avoid these problems, set the output file name for each of those files to a different name with the individual build options for the source files.
The changing the name of the C source file is made with the [Object module file name] property in the [Output File] category from the [Individual Compile Options] tab.
The changing the name of the assembler source file is made with the [Object module file name] property in the [Output File] category from the [Individual Assemble Options] tab.
See "2.11.2 Set build options at the file level" for how to set the individual build options.

2. If the file with the extension of "dr" or "dir" is added to the project, it is treated as the link directive file.
It is also treated as the link directive file if it is added below the Startup node.
When adding the link directive file to the project, if the link directive file has already been added then only the latest link directive file to be added is targeted by a build; any such files added prior to this one will not be targeted.
When setting the link directive file that is not targeted by a build as the build target, if other link directive files have also been added then the file will be targeted by the build, and the others will not be targeted.

3. Up to 5000 files can be added to the main project or subproject.

When a new file is added, an empty file is created in the location specified in the Add File dialog box.
By double clicking the file name on the project tree, you can open the Editor panel and edit the file.
The files that can be opened with the Editor panel are shown below.

- C source file (*.c)
- Assembler source file (*.asm, *.s)
- Header file (*.h, *.inc)
- Symbol information file (*.sfg)
- Link directive file (*.dir, *.dr)
- Link map file (*.map)
- Hex file (*.hex)
- Text file (*.txt)

Remarks 1. You can use one of the procedures below to open files other than those listed above in the Editor panel.

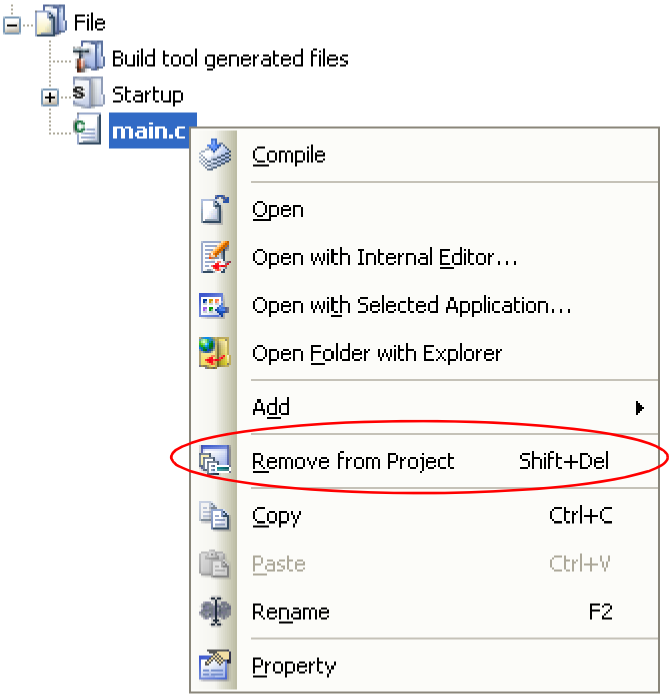
- Drag a file and drop it onto the Editor panel.
- Select a file and then select [Open with Internal Editor...] from the context menu.

2. When the environment is set to use an external editor on the Option dialog box, the file is opened with the external editor that has been set.
Other files are opened with the applications associated by the host OS.

2.3.4 Remove a file from a project

To remove the file added to a project, select the file to be removed from the project on the project tree and then select [Remove from Project] from the context menu.

Figure 2-19. [Remove from Project] Item


File
Build tool generated files
Startup
main.c
Compile
Open
Open with Internal Editor...
Open with Selected Application...
Open Folder with Explorer
Add
Remove from Project Shift+Del
Copy Ctrl+C
Paste Ctrl+V
Rename F2
Property

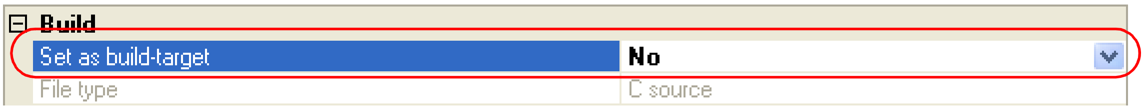
2.3.5 Remove a file from the build target

You can remove the specific file from the build target out of all the files added to the project.

Select the file to be removed from the build target on the project tree and select the [Build Settings] tab on the Property panel.

Select [No] on the [Set as build-target] property in the [Build] category.

Figure 2-20. [Set as build-target] Property



Build	
Set as build-target	No
File type	C source

Remark The files that can be applied this function are C source files, assembler source files, object module files, link directive files, symbol information files, and library files.

2.3.6 Classify a file into a category

You can create a category under the File node and classify files by the category. This makes it easier to view files added to the project on the project tree, and makes it easier to manage files according to function.

To create a category node, select either one of the Project node, Subproject node, or File node on the project tree, and then select [Add] >> [Add New File...] from the context menu.

Figure 2-21. [Add New Category] Item (For File Node)

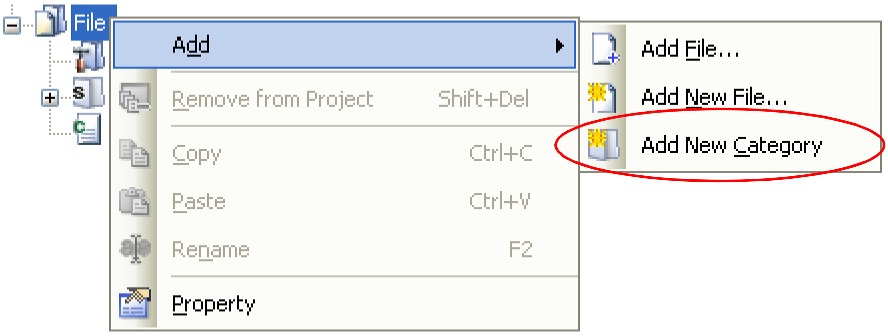
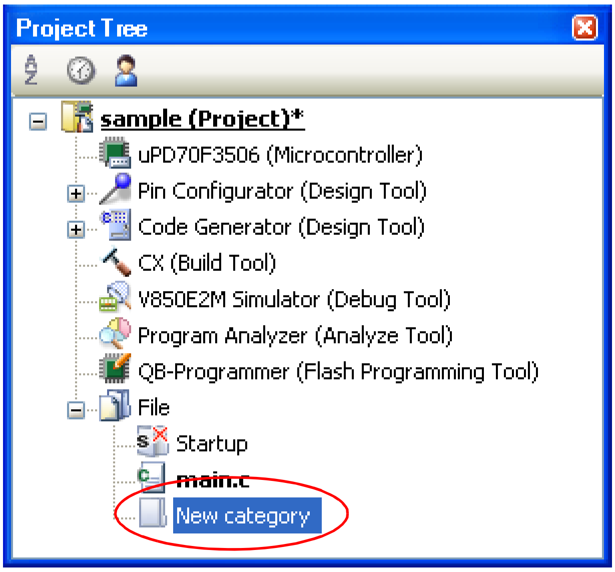


Figure 2-22. Project Tree Panel (After Adding Category Node)



Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Startup
main.c
New category

Remarks 1. The default category name is "New category".
To change the category name, you can use [Rename] from the context menu of the category node.

2. You can also add a category node with the same name as the existing category node.

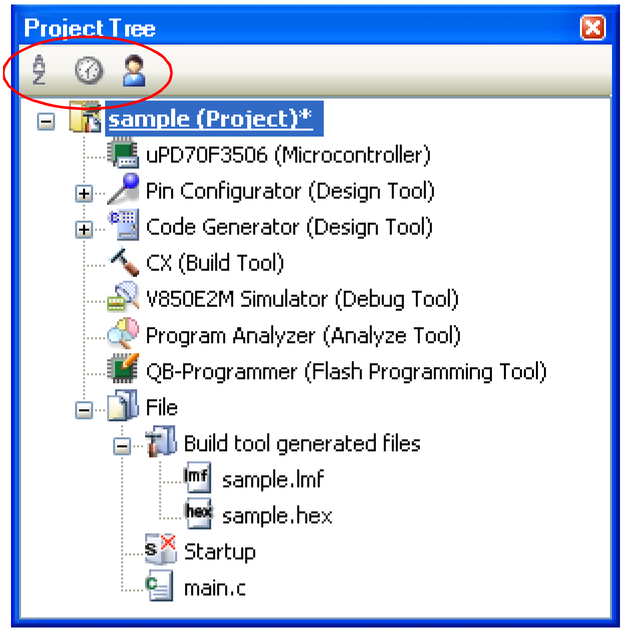
3. Categories can be nested up to 20 levels.

You can classify files into the created category node by dragging and dropping the file.

2.3.7 Change the file display order

You can change the display order of the files and category nodes by the buttons on the project tree.

Figure 2-23. Toolbar (Project Tree Panel)


Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Build tool generated files
sample.lmf
sample.hex
Startup
main.c

Select any of the buttons below on the toolbar of the Project Tree panel.

Button	Description
AZ AZ↓ ZA↓	Sorts category nodes and files in order of their names. AZ: Ascending order AZ↓: Descending order ZA↓: Ascending order
🕓 🕓↑ 🕓↓	Sorts category nodes and files in order of their timestamp. 🕓: Descending order 🕓↑: Ascending order 🕓↓: Descending order
👤	Displays category nodes and files in order of the user definition (default). You can change the display order of category nodes and files arbitrarily by dragging and dropping them.

2.3.8 Update file dependencies

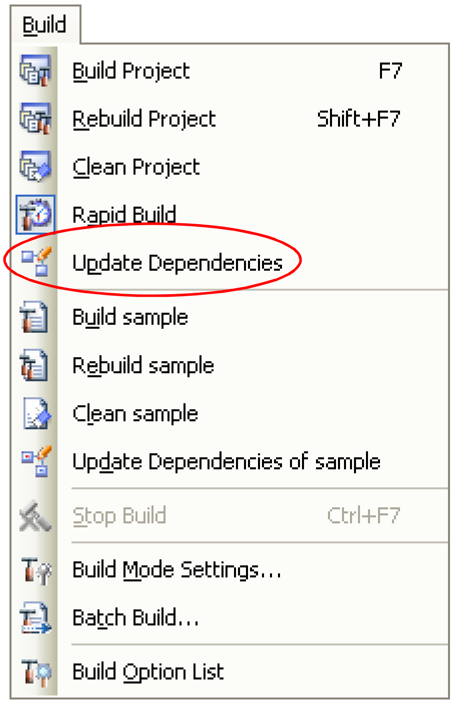
When you perform a change (changing include file paths, adding the include statement of the header file to the C source file and assembler source file, etc.) that effects the file dependencies in the compile option settings or assemble option settings, you must update the dependencies of the relevant files.

Updating file dependencies is performed for the entire project (main project and subprojects) or active project.

(1) For the entire project

From the [Build] menu, select [Update Dependencies].

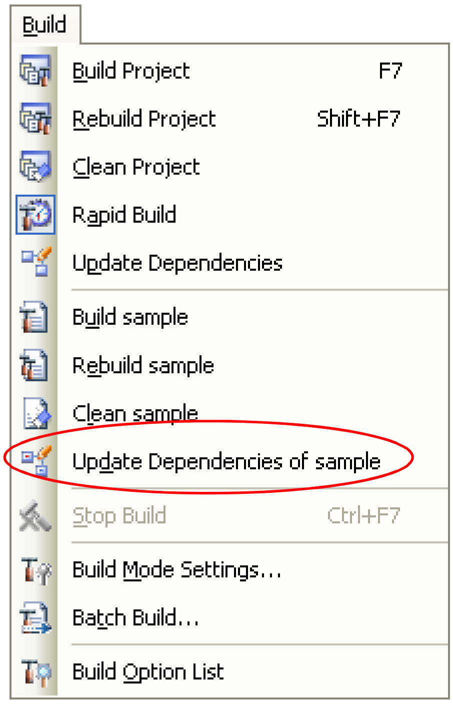
Figure 2-24. [Update Dependencies] Item



(2) For the active project

From the [Build] menu, select [Update Dependencies of *active project*].

Figure 2-25. [Update Dependencies of *active project*] Item



Remark If there are files being edited with the Editor panel when updating file dependencies, then all these files are saved.

Cautions 1. During checking of dependence relationships of include files with CubeSuite, condition statements such as #if and comments are ignored.
Therefore, include files not required for build are mistaken as required files (In the example below, header1.h and header5.h are judged as required for build).

```
#if         0
#include    "header1.h"     /* Dependence relationship judged to exist */
#else                       /* ! zero */
#include    "header2.h"     /* Dependence relationship to exist */
#endif

#define     AAA
#ifdef      AAA
#include    "header3.h"     /* Dependence relationship to exist */
#else
#include    "header4.h"     /* Dependence relationship to exist */
#endif

/*
#include    "header5.h"     /* Dependence relationship judged to exist */
*/
```

2. During checking of dependence relationships of include files with CubeSuite, include statements described after comments are ignored.

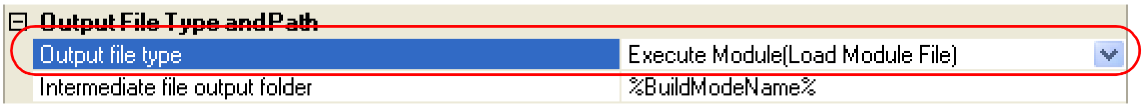
Therefore, include files required for build are mistaken as no-required files (In the example below, header6.h and header7.h are judged as no-required for build).

```
/* comment */   #include    "header6.h"     /* Dependence relationship judged not
to exist */


/*

comment

*/  #include    "header7.h"                 /* Dependence relationship judged not
to exist */
```

2.4 Set the Type of the Output File

Set the type of the file to be output as the product of the build.
Select the build tool node on the project tree and select the [Common Options] tab on the Property panel.
Select the file type in the [Output file type] property in the [Output File Type and Path] category.

Figure 2-26. [Output file type] Property



⊟ Output File Type and Path

Output file type	Execute Module(Load Module File)
Intermediate file output folder	%BuildModeName%

(1) When [Execute Module(Load Module File)] is selected (Default)

The load module file (non-ROMized) and hex file are generated.
The load module file will be the debug target.
The ROMized load module file will be the debug target when [Yes] in the [Output ROMized load module file] property in the [Output File] category from the [ROMize Options] tab category is selected.

(2) When [Execute Module(Non-ROMized Load Module File)] is selected

The load module file (non-ROMized), ROMized load module file and hex file are generated.
The load module file (non-ROMized) will be the debug target.

Remark This item is displayed only when [Yes] in the [Output ROMized load module file] property and [Yes] in the [Output Non-ROMized load module file] property in the [Output File] category from the [ROMize Options] tab is selected.

(3) When [Execute Module(Hex File)] is selected

The load module file (non-ROMized) and hex file are generated.
The hex file will be the debug target.

Caution For the library project, this property is always [Library] and cannot be changed.

2.4.1 Change the output file name

The names of the load module file, non-ROMized load module file, hex file, and library file output by the build tool are set to the following names by default.

"%ProjectName%" is an embedded macro. It is replaced to the project name.

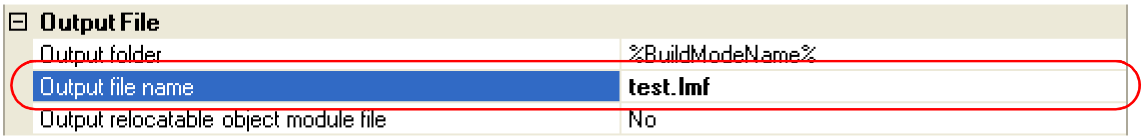
Load module file name: %ProjectName%.lmf
Non-ROMized load module file: %ProjectName%_NonROMize.lmf
Hex file name: %ProjectName%.hex
Library file name: lib%ProjectName%.lib

The method to change these file names is shown below.

(1) When changing the load module file name and non-ROMized load module file name

Select the build tool node on the project tree and select the [Link Options] tab on the Property panel.
Enter the file name to be changed to in the [Output file name] property in the [Output File] category.

Figure 2-27. [Output file name] Property



⊟ Output File	
Output folder	%BuildModeName%
Output file name	test.lmf
Output relocatable object module file	No

The non-ROMized load module file name is the name specified in the [Output file name] property with string "_NonROMize" added.

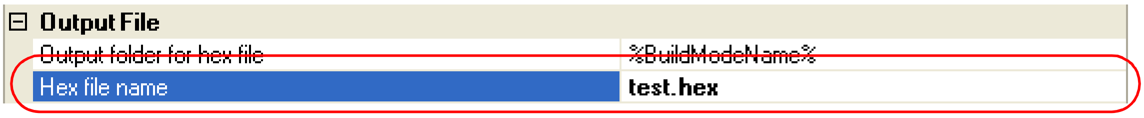
Remarks 1. You can also change the option in the same way with the [Output file name] property in the [Frequently Used Options(for Link)] category on the [Common Options] tab.

2. If the target is a multi-core CPU, the load module for the common module and the load module for core *n* are generated, and the final load module is then generated based on those (*n*: number of cores of the target CPU).
Load module for the common module: *input string without the extension* _cmn.lmf
Load module for the core *n*: *input string without the extension* _pen.lmf

(2) When changing the hex file name

Select the build tool node on the project tree and select the [Hex Output Options] tab on the Property panel.
Enter the hex file name to be changed to on the [Hex file name] property in the [Output File] category.

Figure 2-28. [Hex file name] Property

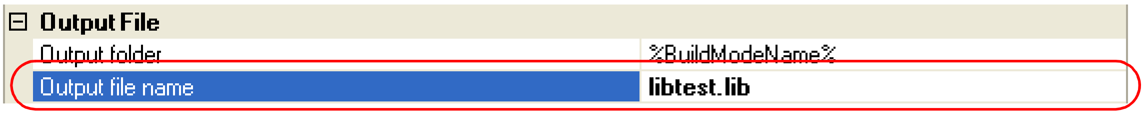


⊟ Output File	
Output folder for hex file	%BuildModeName%
Hex file name	test.hex

(3) When changing the library file name

Select the build tool node on the project tree and select the [Create Library Options] tab on the Property panel.
Enter the library file name to be changed to on the [Output file name] property in the [Output File] category.

Figure 2-29. [Output file name] Property



⊟ Output File

Output folder	%BuildModeName%
Output file name	libtest.lib

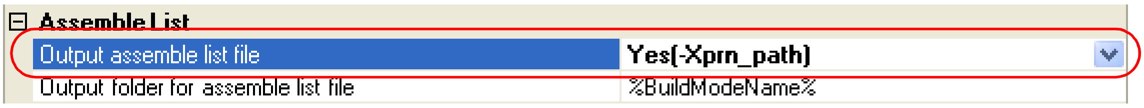
2.4.2 Output an assemble list

The assemble list (the code of the assemble result) is output to the assemble list file.

Select the build tool node on the project tree and select the [Compile Options] tab on the Property panel.

To output the assemble list file, select [Yes(-Xprn_path)] in the [Output assemble list file] property in the [Assemble List] category.

Figure 2-30. [Output assemble list file] Property



⊟ Assemble List

Output assemble list file	Yes(-Xprn_path)
Output folder for assemble list file	%BuildModeName%

When outputting the assemble list file, you can set the output folder and output file name.

(1) Set the output folder

Setting the output folder is made with the [Output folder for assemble list file] property by directly entering in the text box or by the [...] button.

"%BuildModeName%" is set by default.

"%BuildModeName%" is an embedded macro. It is replaced to the build mode name.

The file name will be the source file name with the extension replaced by ".prn".

Remark See "3.1 Assemble List File" for details about the assemble list file.

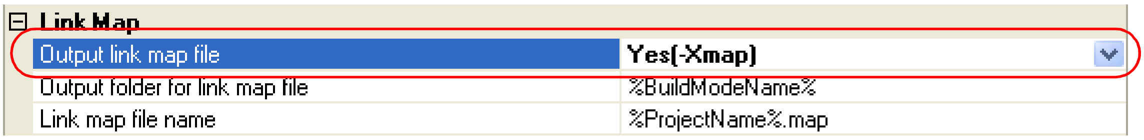
2.4.3 Output map information

The map information (the information of the link result) is output to the link map file.

Select the build tool node on the project tree and select the [Link Options] tab on the Property panel.

To output the link map file, select [Yes(-Xmap)] in the [Output link map file] property in the [Link Map] category.

Figure 2-31. [Output link map file] property



⊟ Link Map

Output link map file	Yes(-Xmap)
Output folder for link map file	%BuildModeName%
Link map file name	%ProjectName%.map

When outputting the link map file, you can set the output folder and output file name.

(1) Set the output folder

Setting the output folder is made with the [Output folder for link map file] property by directly entering in the text box or by the [...] button.

"%BuildModeName%" is set by default.

"%BuildModeName%" is an embedded macro. It is replaced to the build mode name.

(2) Set the output file name

Setting the output file is made with the [Link map file name] property by directly entering in the text box.

"%ProjectName%.map" is set by default.

"%ProjectName%" is an embedded macro. It is replaced to the project name.

Remark See "3.2 Link Map File" for details about the link map file.

2.4.4 Output symbol information

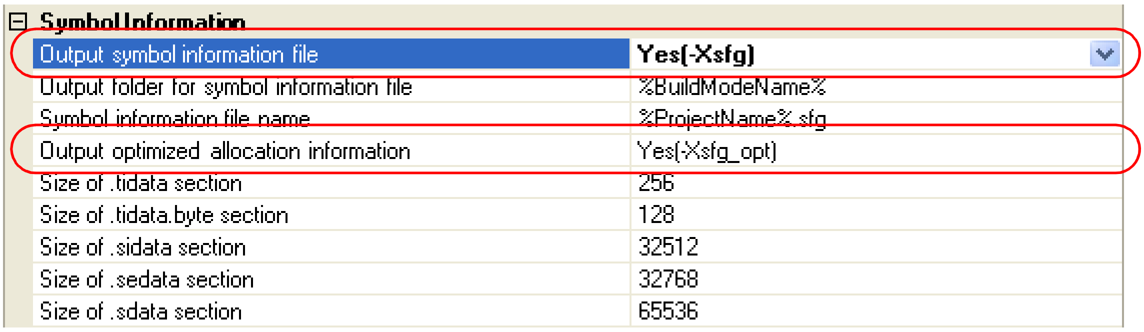
The symbol information (the allocation section information of variables) is output to the symbol information file.

Select the build tool node on the project tree and select the [Link Options] tab on the Property panel.

To output the symbol information file, select [Yes(-Xsfg)] in the [Output symbol information file] property in the [Symbol Information] category.

Select [Yes(-Xsfg_opt)] in the [Output optimized allocation information] property to output optimum allocation information for variables at the section level.

Figure 2-32. [Output symbol information file] and [Output optimized allocation information] Property



⊟ Symbol Information

Output symbol information file	Yes(-Xsfg)
Output folder for symbol information file	%BuildModeName%
Symbol information file name	%ProjectName%.sfg
Output optimized allocation information	Yes(-Xsfg_opt)
Size of .tidata section	256
Size of .tidata.byte section	128
Size of .sidata section	32512
Size of .sedata section	32768
Size of .sdata section	65536

When outputting the symbol information file, you can set the output folder and output file name.

(1) Set the output folder

Setting the output folder is made with the [Output folder for symbol information file] property by directly entering in the text box or by the [...] button.

"%BuildModeName%" is set by default.

"%BuildModeName%" is an embedded macro. It is replaced to the build mode name.

(2) Set the output file name

Setting the output file is made with the [Symbol information file name] property by directly entering in the text box.

"%ProjectName%.sfg" is set by default.

"%ProjectName%" is an embedded macro. It is replaced to the project name.

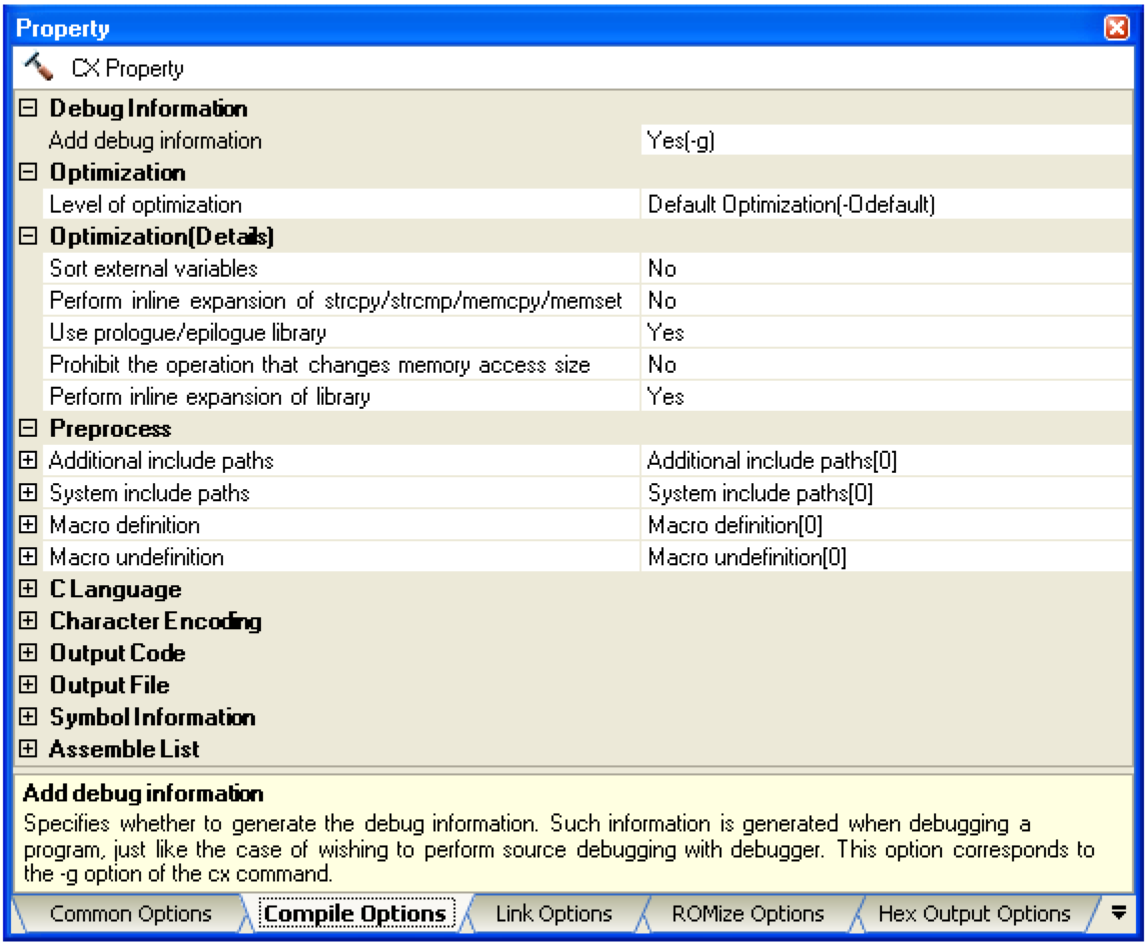
Remark See "3.3 Symbol Information File" for details about the symbol information file.

2.5 Set Compile Options

To set options for the compile phase, select the Build tool node on the project tree and select the [Compile Options] tab on the Property panel.

You can set the various compile options by setting the necessary properties in this tab.

Figure 2-33. Property Panel: [Compile Options] Tab



⊟ **Debug Information**	
Add debug information	Yes(-g)
⊟ **Optimization**	
Level of optimization	Default Optimization(-Odefault)
⊟ **Optimization(Details)**	
Sort external variables	No
Perform inline expansion of strcpy/strcmp/memcpy/memset	No
Use prologue/epilogue library	Yes
Prohibit the operation that changes memory access size	No
Perform inline expansion of library	Yes
⊟ **Preprocess**	
⊞ Additional include paths	Additional include paths[0]
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]
⊞ **C Language**	
⊞ **Character Encoding**	
⊞ **Output Code**	
⊞ **Output File**	
⊞ **Symbol Information**	
⊞ **Assemble List**	

Add debug information

Specifies whether to generate the debug information. Such information is generated when debugging a program, just like the case of wishing to perform source debugging with debugger. This option corresponds to the -g option of the cx command.

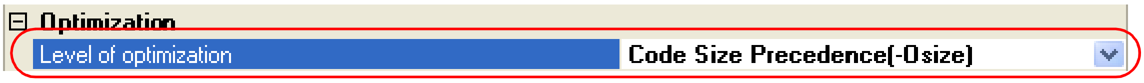
Remark Often used options have been gathered under the [Frequently Used Options(for Compile)] category on the [Common Options] tab.

2.5.1 Perform optimization with the code size precedence

Select the build tool node on the project tree and select the [Compile Options] tab on the Property panel.

To perform optimization with the code size precedence, select [Code Size Precedence(-Osize)] in the [Optimization Level] property in the [Optimization] category (default: [Default Optimization(-Odefault)]).

Figure 2-34. [Level of optimization] Property (Code Size Precedence)



⊟ **Optimization**	
Level of optimization	**Code Size Precedence(-Osize)**

Remarks 1. You can also set the option in the same way with the [Optimization Level] property in the [Frequently Used Options(for Compile)] category on the [Common Options] tab.

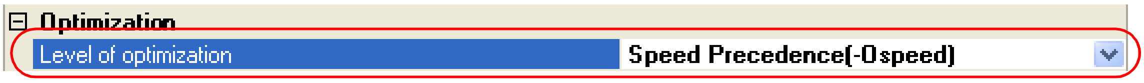
2. See "B.1.5 Optimization function" for details about the optimization function.

2.5.2 Perform optimization with the execution speed precedence

Select the build tool node on the project tree and select the [Compile Options] tab on the Property panel.

To perform optimization with the execution speed precedence, select [Speed Precedence(-Ospeed)] in the [Optimization Level] property in the [Optimization] category (default: [Default Optimization(-Odefault)]).

Figure 2-35. [Level of optimization] Property (Execution Speed Precedence)



⊟ Optimization

Level of optimization	Speed Precedence(-Ospeed)

Remarks 1. You can also set the option in the same way with the [Optimization Level] property in the [Frequently Used Options(for Compile)] category on the [Common Options] tab.

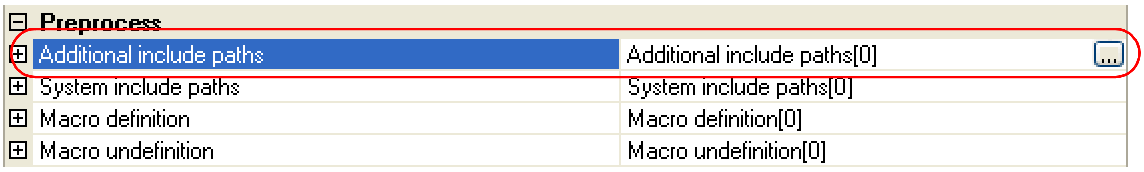
2. See "B.1.5 Optimization function" for details about the optimization function.

2.5.3 Add an include path

Select the build tool node on the project tree and select the [Compile Options] tab on the Property panel.

The include path setting is made with the [Additional include paths] property in the [Preprocess] category.

Figure 2-36. [Additional include paths] Property

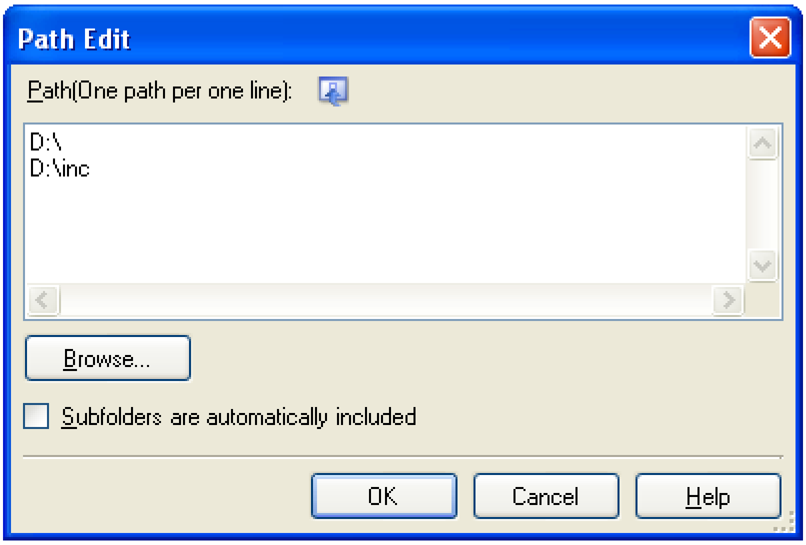


⊟ Preprocess

⊞ Additional include paths	Additional include paths[0]
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]

If you click the [...] button, the Path Edit dialog box will open.

Figure 2-37. Path Edit Dialog Box



Path Edit

Path(One path per one line):

```
D:\
D:\inc
```

Browse...

□ Subfolders are automatically included

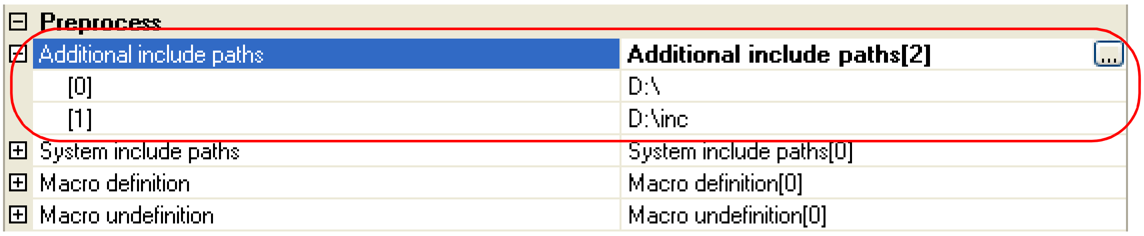
Enter the include path per line in [Path(One path per one line)].

You can specify up to 259 characters per line, up to 256 lines.

Remark You can also specify the include path by dragging and dropping from Explorer or the like, or by the [Browse...] button.
Select the [Subfolders are automatically included] check box before clicking the [Browse...] button to add all paths under the specified one (down to 5 levels) to [Path(One path per one line)].

If you click the [OK] button, the entered include paths are displayed as subproperties.

Figure 2-38. [Additional include paths] Property (After Adding Include Paths)



⊟ Preprocess	
⊟ Additional include paths	Additional include paths[2]
[0]	D:\
[1]	D:\inc
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]

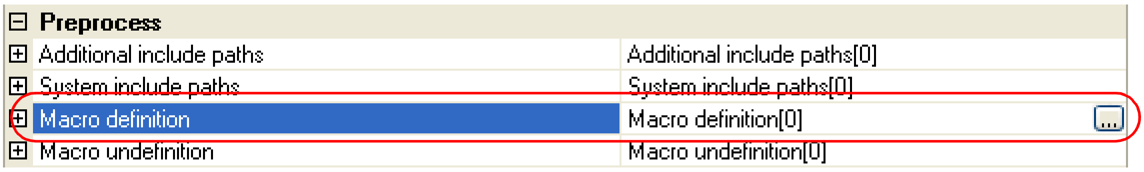
To change the include paths, you can use the [...] button or enter the path directly in the text box of the subproperty.
When the include path is added to the project tree, the path is added to the top of the subproperties automatically.

Remark You can also set the option in the same way with the [Additional include paths] property in the [Frequently Used Options(for Compile)] category on the [Common Options] tab.

2.5.4 Set a macro definition

Select the build tool node on the project tree and select the [Compile Options] tab on the Property panel.
The macro definition setting is made with the [Macro definition] property in the [Preprocess] category.

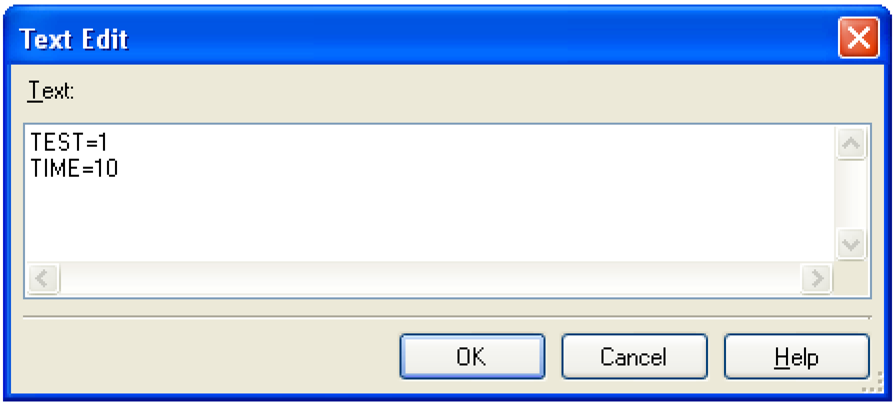
Figure 2-39. [Macro definition] Property



⊟ Preprocess	
⊞ Additional include paths	Additional include paths[0]
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]

If you click the [...] button, the Text Edit dialog box will open.

Figure 2-40. Text Edit Dialog Box

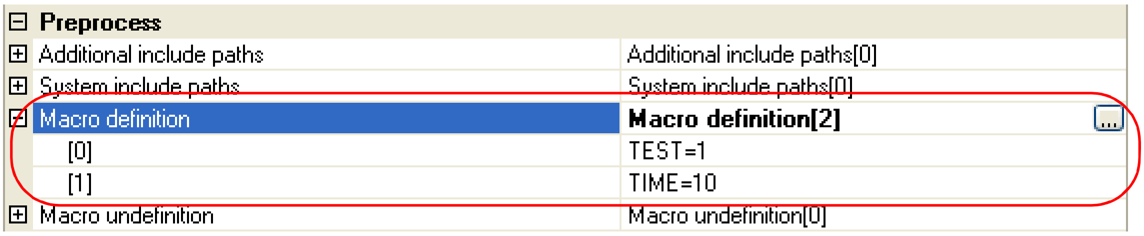


Text:

```
TEST=1
TIME=10
```

Enter the macro definition in the format of "*macro name=defined value*", with one macro name per line.
You can specify up to 256 characters per line, up to 256 lines.
The "*=defined value*" part can be omitted, and in this case, "1" is used as the defined value.
If you click the [OK] button, the entered macro definitions are displayed as subproperties.

Figure 2-41. [Macro definition] Property (After Setting Macros)



⊟ Preprocess	
⊞ Additional include paths	Additional include paths[0]
⊞ System include paths	System include paths[0]
⊟ Macro definition	Macro definition[2]
[0]	TEST=1
[1]	TIME=10
⊞ Macro undefinition	Macro undefinition[0]

To change the macro definitions, you can use the [...] button or enter the path directly in the text box of the subproperty.

Remark You can also set the option in the same way with the [Macro definition] property in the [Frequently Used Options(for Compile)] category on the [Common Options] tab.

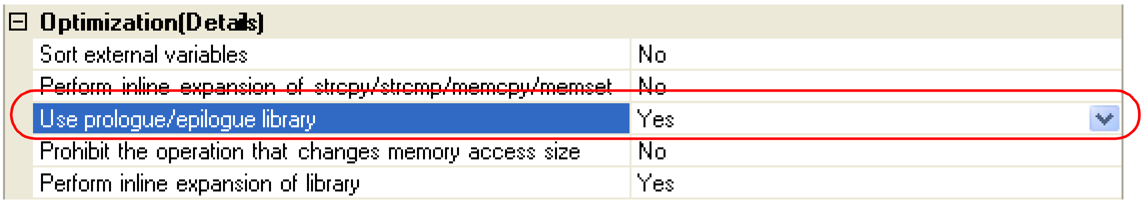
2.5.5 Reduce the code size (perform prologue/epilogue runtime calls)

It is possible to reduce the code size by performing a part of prologue/epilogue processing of the function based on runtime library function calls.

Select the build tool node on the project tree and select the [Compile Options] tab on the Property panel.

To perform prologue/epilogue processing of the function based on runtime library function calls, select [Yes] in the [Use prologue/epilogue library] property in the [Output Code] category.

Figure 2-42. [Use prologue/epilogue library] Property



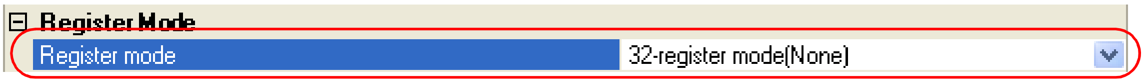
⊟ Optimization(Details)	
Sort external variables	No
Perform inline expansion of strcpy/strcmp/memcpy/memset	No
Use prologue/epilogue library	Yes
Prohibit the operation that changes memory access size	No
Perform inline expansion of library	Yes

2.5.6 Change the register mode

Select the build tool node on the project tree and select the [Common Options] tab on the Property panel.

Select the register mode to on the [Register mode] property in the [Register Mode] category.

Figure 2-43. [Register mode] Property



⊟ Register Mode	
Register mode	32-register mode(None)

You can select from the following register modes.

Register Mode	Working Registers	Registers for Register Variables
32-register mode(None) (default)	r10 to r19	r20 to r29
26-register mode(-Xreg_mode=26)	r10 to r16	r23 to r29

Register Mode	Working Registers	Registers for Register Variables
22-register mode(-Xreg_mode=22)	r10 to r14	r25 to r29
Universal register mode(-Xreg_mode=common)	r10 to r14	r25 to r29

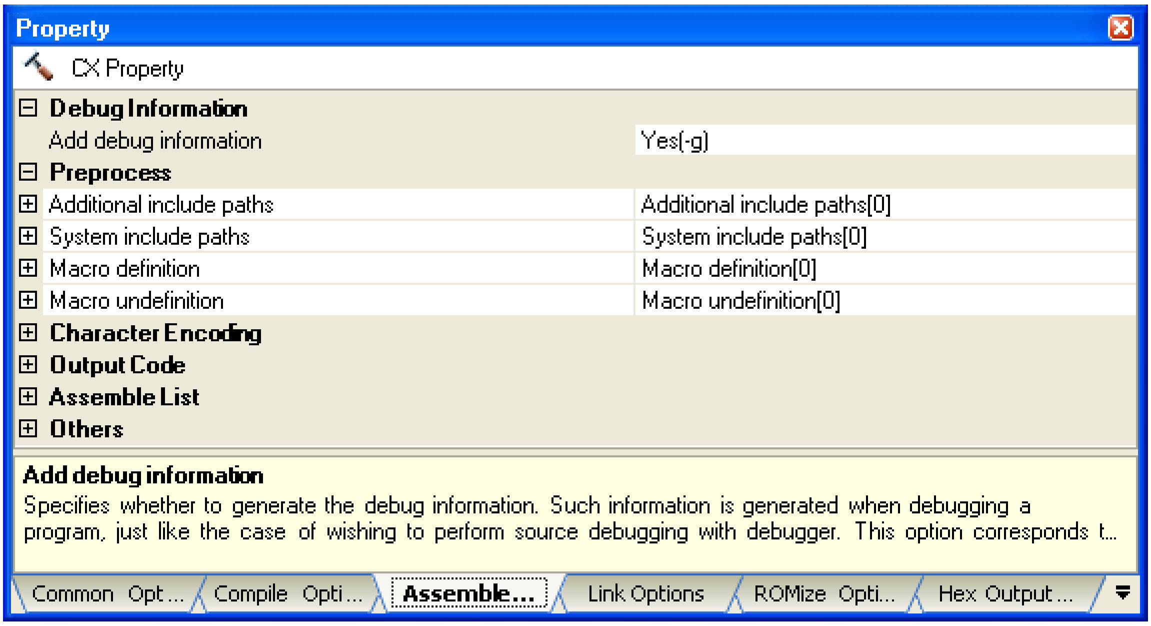
Remark See "CubeSuite Coding for CX Compiler" for details about the register mode.

2.6 Set Assemble Options

To set options for the assemble phase, select the Build tool node on the project tree and select the [Assemble Options] tab on the Property panel.

You can set the various assemble options by setting the necessary properties in this tab.

Figure 2-44. Property Panel: [Assemble Options] Tab



⊟ Debug Information

Add debug information	Yes(-g)

⊟ Preprocess

⊞ Additional include paths	Additional include paths[0]
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]

⊞ Character Encoding
⊞ Output Code
⊞ Assemble List
⊞ Others

Add debug information
Specifies whether to generate the debug information. Such information is generated when debugging a program, just like the case of wishing to perform source debugging with debugger. This option corresponds t...

Remark Often used options have been gathered under the [Frequently Used Options(for Assemble)] category on the [Common Options] tab.

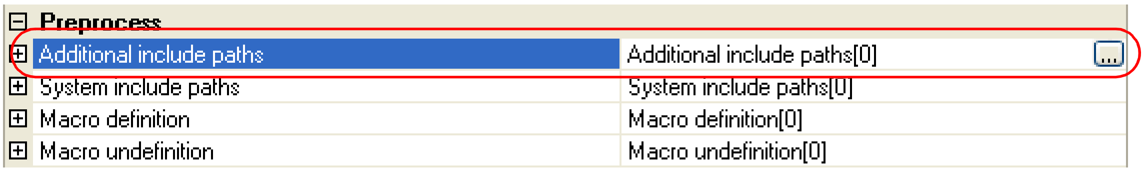
Caution This tab is displayed only when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.

2.6.1 Add an include path

Select the build tool node on the project tree and select the [Assemble Options] tab on the Property panel.

The include path setting is made with the [Additional include paths] property in the [Preprocess] category.

Figure 2-45. [Additional include paths] Property

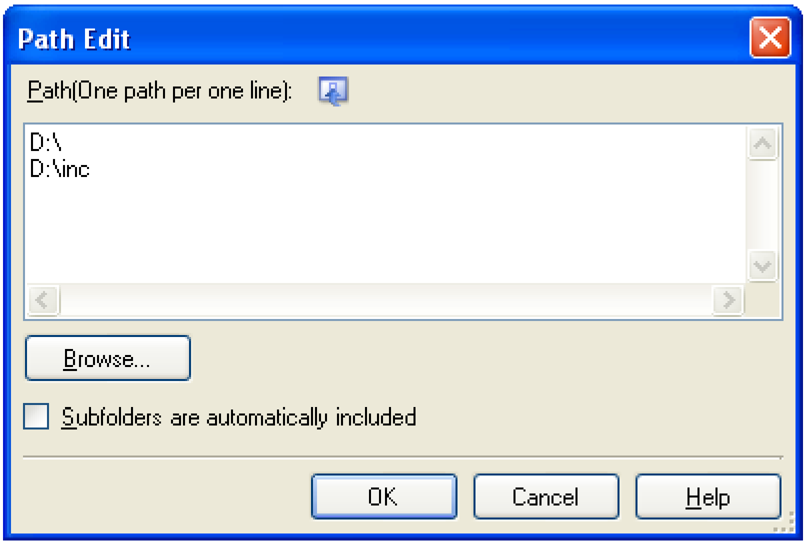


⊟ Preprocess

⊞ Additional include paths	Additional include paths[0] [...]
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]

If you click the [...] button, the Path Edit dialog box will open.

Figure 2-46. Path Edit Dialog Box


Path Edit
Path(One path per one line):
Browse...
Subfolders are automatically included
OK
Cancel
Help

Path(One path per one line):

```
D:\
D:\inc
```

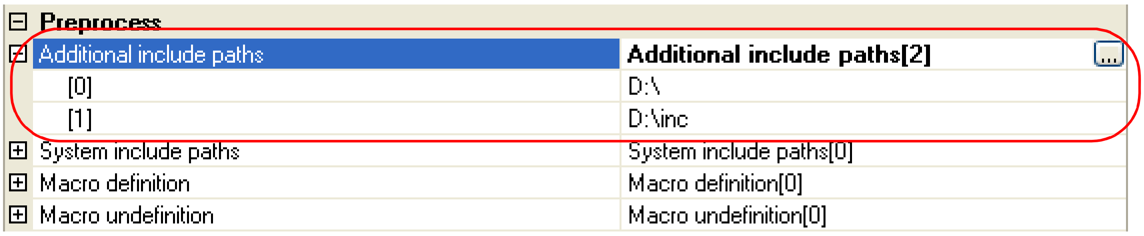
□ Subfolders are automatically included

Enter the include path per line in [Path(One path per one line)].
You can specify up to 259 characters per line, up to 256 lines.

Remark You can also specify the include path by dragging and dropping from Explorer or the like, or by the [Browse...] button.
Select the [Subfolders are automatically included] check box before clicking the [Browse...] button to add all paths under the specified one (down to 5 levels) to [Path(One path per one line)].

If you click the [OK] button, the entered include paths are displayed as subproperties.

Figure 2-47. [Additional include paths] Property (After Adding Include Paths)



⊟ Preprocess	
⊟ Additional include paths	Additional include paths[2]
[0]	D:\
[1]	D:\inc
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]

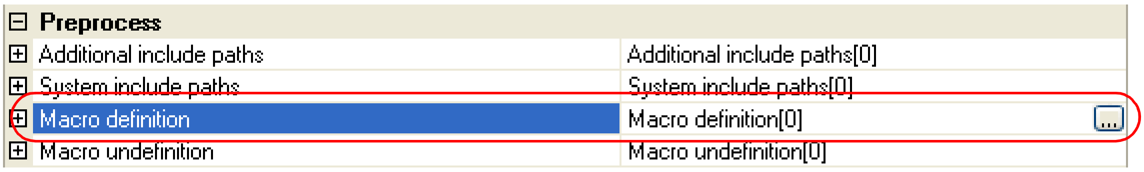
To change the include paths, you can use the [...] button or enter the path directly in the text box of the subproperty.
When the include path is added to the project tree, the path is added to the top of the subproperties automatically.

Remark You can also set the option in the same way with the [Additional include paths] property in the [Frequently Used Options(for Assemble)] category on the [Common Options] tab.

2.6.2 Set a macro definition

Select the build tool node on the project tree and select the [Assemble Options] tab on the Property panel.
The macro definition setting is made with the [Macro definition] property in the [Preprocess] category.

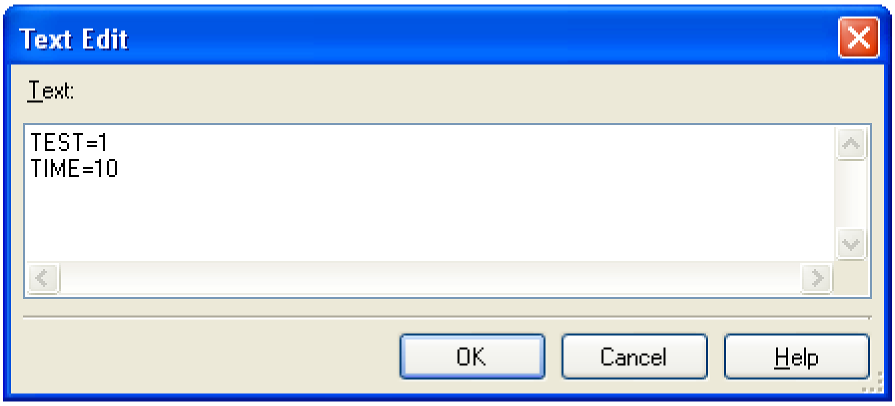
Figure 2-48. [Macro definition] Property



Preprocess	
Additional include paths	Additional include paths[0]
System include paths	System include paths[0]
Macro definition	Macro definition[0]
Macro undefinition	Macro undefinition[0]

If you click the [...] button, the Text Edit dialog box will open.

Figure 2-49. Text Edit Dialog Box



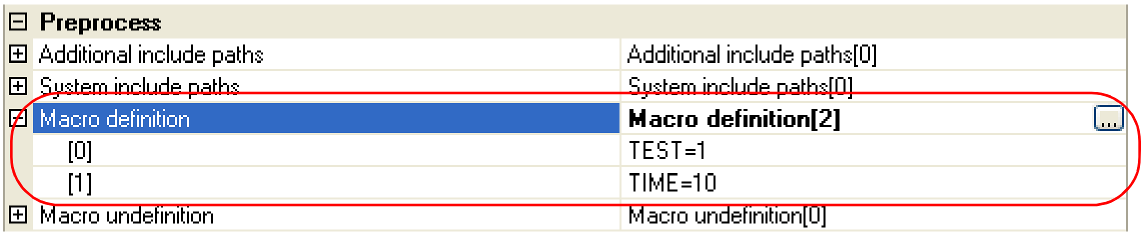
Text Edit

Text:

```
TEST=1
TIME=10
```

Enter the macro definition in the format of "*macro name=defined value*", with one macro name per line.
You can specify up to 256 characters per line, up to 256 lines.
The "=*defined value*" part can be omitted, and in this case, "1" is used as the defined value.
If you click the [OK] button, the entered macro definitions are displayed as subproperties.

Figure 2-50. [Macro definition] Property (After Setting Macros)



Preprocess	
Additional include paths	Additional include paths[0]
System include paths	System include paths[0]
Macro definition	**Macro definition[2]**
[0]	TEST=1
[1]	TIME=10
Macro undefinition	Macro undefinition[0]

To change the macro definitions, you can use the [...] button or enter the path directly in the text box of the subproperty.

Remark You can also set the option in the same way with the [Macro definition] property in the [Frequently Used Options(for Assemble)] category on the [Common Options] tab.

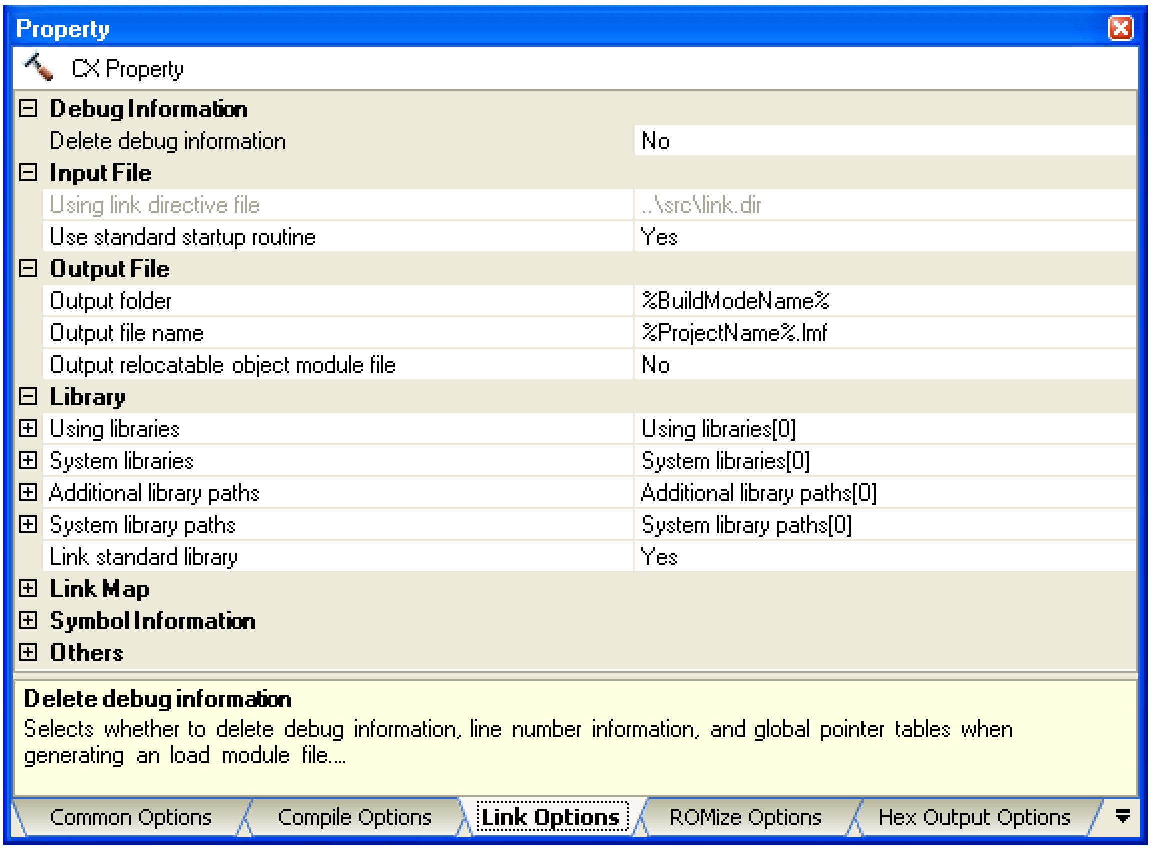
2.7 Set Link Options

To set options for the link phase, select the Build tool node on the project tree and select the [Link Options] tab on the Property panel.

You can set the various link options by setting the necessary properties in this tab.

Caution This tab is not displayed for the library project.

Figure 2-51. Property Panel: [Link Options] Tab



Property

CX Property

⊟ **Debug Information**	
Delete debug information	No
⊟ **Input File**	
Using link directive file	..\src\link.dir
Use standard startup routine	Yes
⊟ **Output File**	
Output folder	%BuildModeName%
Output file name	%ProjectName%.lmf
Output relocatable object module file	No
⊟ **Library**	
⊞ Using libraries	Using libraries[0]
⊞ System libraries	System libraries[0]
⊞ Additional library paths	Additional library paths[0]
⊞ System library paths	System library paths[0]
Link standard library	Yes
⊞ **Link Map**	
⊞ **Symbol Information**	
⊞ **Others**	

Delete debug information
Selects whether to delete debug information, line number information, and global pointer tables when generating an load module file....

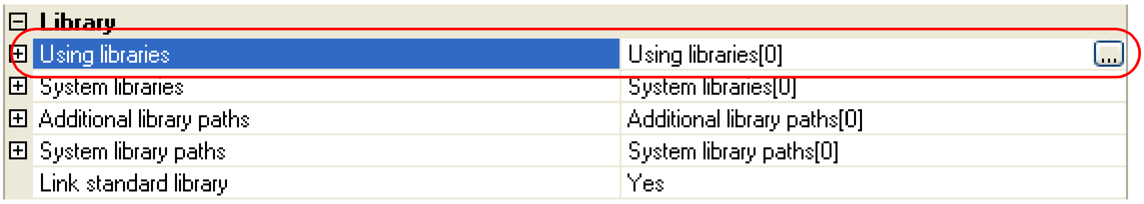
Remark Often used options have been gathered under the [Frequently Used Options(for Link)] category on the [Common Options] tab.

2.7.1 Add a user library

Select the build tool node on the project tree and select the [Link Options] tab on the Property panel.

Adding a user library is made with the [Using libraries] property in the [Library] category.

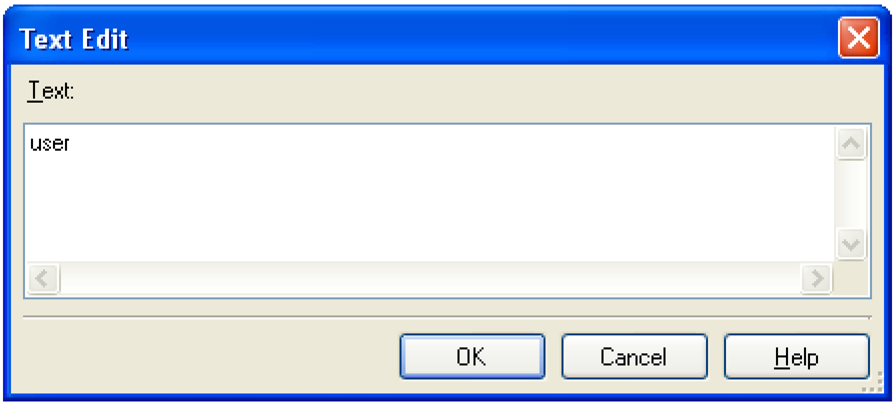
Figure 2-52. [Using libraries] Property



⊟ **Library**	
⊞ Using libraries	Using libraries[0] [...]
⊞ System libraries	System libraries[0]
⊞ Additional library paths	Additional library paths[0]
⊞ System library paths	System library paths[0]
Link standard library	Yes

If you click the [...] button, the Text Edit dialog box will open.

Figure 2-53. Text Edit Dialog Box



Text:

user

In the [Text], specify only the "*xxx*" part of library file name "lib*xxx*.lib" (example: if you specify "user", "libuser.lib" is assumed to be specified).

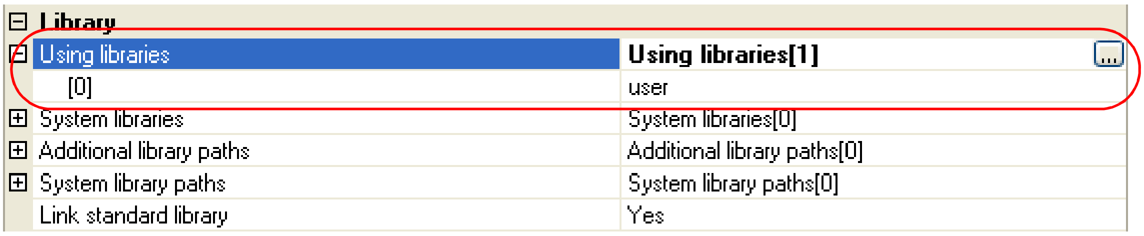
Add one item in one line.

You can specify up to 249 characters per line, up to 256 lines.

Remark "lib*xxx*.a" can also be specified.
lib*xxx*.lib is searched with higher priority. If it is not found, lib*xxx*.a is searched.

If you click the [OK] button, the entered library files are displayed as subproperties.

Figure 2-54. [Using libraries] Property (After Setting Library Files)



⊟ Library	
⊟ Using libraries	Using libraries[1]
[0]	user
⊞ System libraries	System libraries[0]
⊞ Additional library paths	Additional library paths[0]
⊞ System library paths	System library paths[0]
Link standard library	Yes

To change the library files, you can use the [...] button or enter the path directly in the text box of the subproperty.

Remark You can also set the option in the same way with the [Using libraries] property in the [Frequently Used Options(for Link)] category on the [Common Options] tab.

The library file is searched from the library path.

To add a library path, set the [Additional library paths] property.

Caution **The library file can also be linked by adding it directly to the project.**
In this case, the library file is not searched from the library path because it is linked directly via the absolute path.

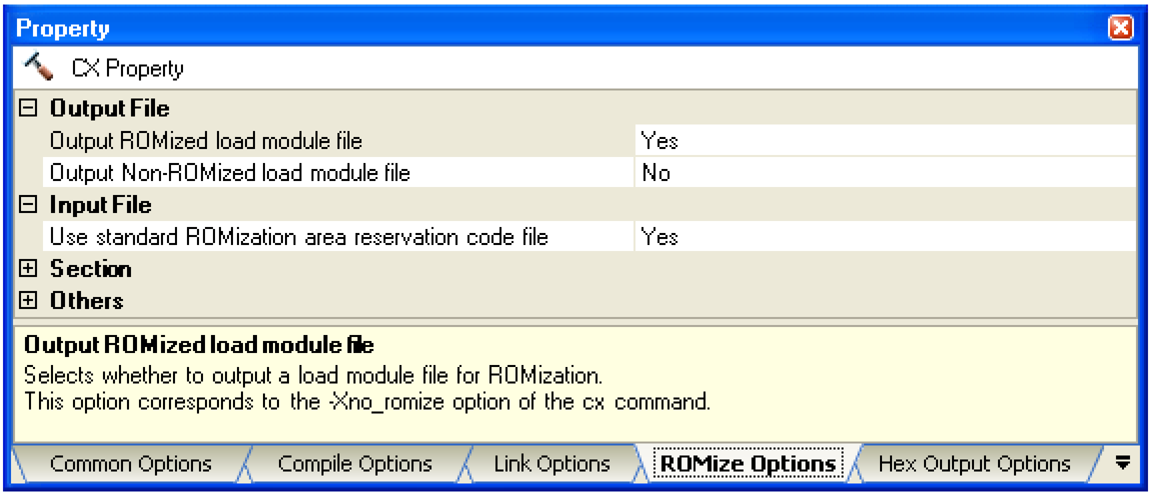
2.8 Set ROMize Options

To set options for the ROMize phase, select the Build tool node on the project tree and select the [ROMize Options] tab on the Property panel.

You can set the various ROMize options by setting the necessary properties in this tab.

Caution This tab is not displayed for the library project.

Figure 2-55. Property Panel: [ROMize Options] Tab



Property

CX Property

⊟ Output File

Output ROMized load module file	Yes
Output Non-ROMized load module file	No

⊟ Input File

Use standard ROMization area reservation code file	Yes

⊞ Section

⊞ Others

Output ROMized load module file

Selects whether to output a load module file for ROMization.

This option corresponds to the -Xno_romize option of the cx command.

2.8.1 Create a ROMized load module

The following procedure shows how to create a ROMized load module using the ROMization area reservation code file (rompcrt.obj) that is provided by default.

The ROMization processor is a tool that takes initial value information for variables in data-attribute sections as well as programs allocated to RAM and packs them into a single section.

By default, this section becomes the "rompsec section".

By allocating the rompsec section to ROM and calling the copy function, it is possible to deploy initial value information and programs into RAM.

Remark See "CubeSuite Coding for CX Compiler" for details about how to create the ROMized load module.

(1) Call a copy function

Call a copy function within the startup routine.

You can use the standard startup routine (cstart.obj) as-is, because it already contains a call to "_rcopy".

If you use "_rcopy2" or "_rcopy4" instead of "_rcopy" or specify the section to be copied, customize the standard startup routine.

Remarks 1. See "CubeSuite Coding for CX Compiler" for details about copy functions.

2. To use other than the standard startup routine, add the file to be used to the Startup node on the project tree.

See "2.3.1 Set a startup routine" for details about the setting of the startup routine.

(2) Set ROMize options

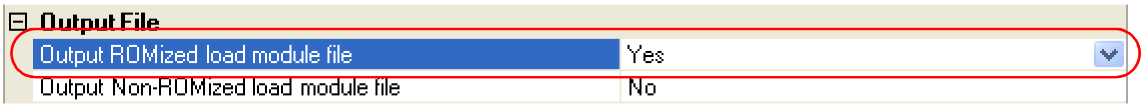
Specify to generate the ROMized load module by the ROMize option.

Select the build tool node on the project tree and select the [ROMize Options] tab on the Property panel.

(a) Set the output of the ROMized load module file

Select [Yes] in the [Output ROMized load module file] property in the [Output File] category.

Figure 2-56. [Output ROMized load module file] Property



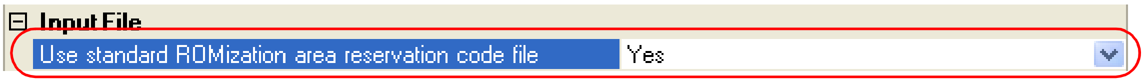
⊟ Output File

Output ROMized load module file	Yes
Output Non-ROMized load module file	No

(b) Set using the standard ROMization area reservation code file

Select [Yes] (default) in the [Use standard ROMization area reservation code file] property in the [Input file] category.

Figure 2-57. [Use standard ROMization area reservation code file] Property



⊟ Input File

Use standard ROMization area reservation code file	Yes

(3) Run a build

Run a build to generate the ROMization load module file.

At this time, files are linked according to the following sequence.

- Object module file of the startup routine (cstart.obj)
- Library file that stores copy functions (libc.lib)
- ROMization area reservation code file (rompcrt.obj)

Remark The ROMization area reservation code file should be linked last.
However, if [Yes(-Xrescan)] in the [Rescan library files] property in the [Others] category on the [Link Options] tab is specified, the library file is linked after the ROMization area reservation code file, and an error may be output during ROMization processing.
In such a case, explicitly secure the rompsec section area.
See "CubeSuite Coding for CX Compiler" for details.

CubeSuite performs hex output processing after ROMization processing. Therefore, a hex file is also generated. Load the generated hex file to the target using a ROM writer.

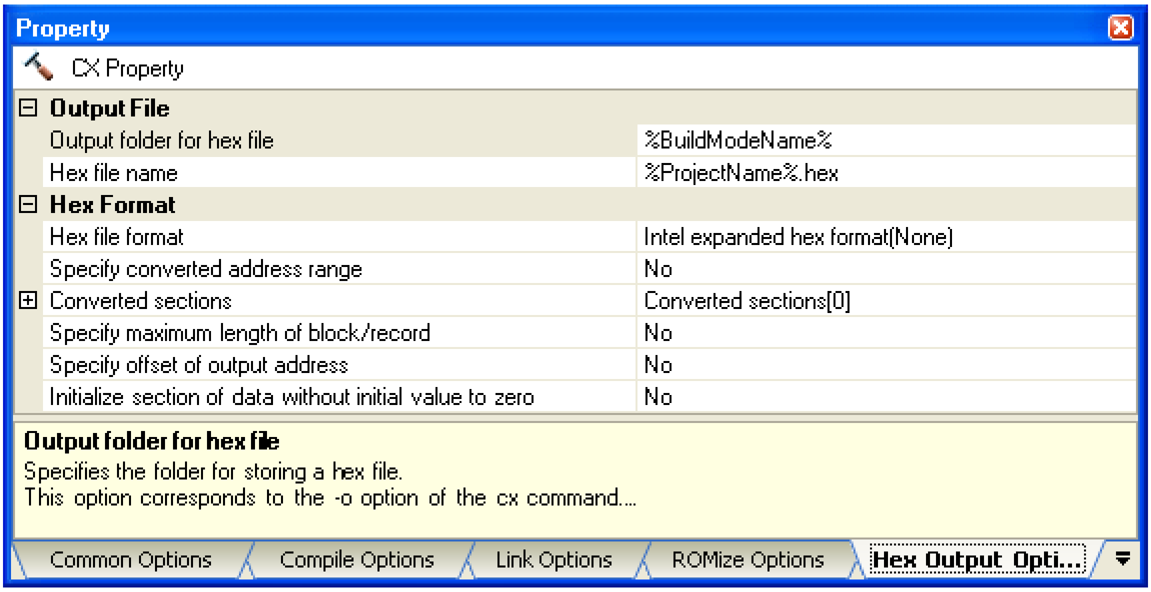
2.9 Set Hex Output Options

To set options for the hex output phase, select the Build tool node on the project tree and select the [Hex Output Options] tab on the Property panel.

You can set the various hex output options by setting the necessary properties in this tab.

Caution This tab is not displayed for the library project.

Figure 2-58. Property Panel: [Hex Output Options] Tab



⊟ **Output File**	
Output folder for hex file	%BuildModeName%
Hex file name	%ProjectName%.hex
⊟ **Hex Format**	
Hex file format	Intel expanded hex format(None)
Specify converted address range	No
⊞ Converted sections	Converted sections[0]
Specify maximum length of block/record	No
Specify offset of output address	No
Initialize section of data without initial value to zero	No

Output folder for hex file
Specifies the folder for storing a hex file.
This option corresponds to the -o option of the cx command....

Remark Often used options have been gathered under the [Frequently Used Options(for Hex Output)] category on the [Common Options] tab.

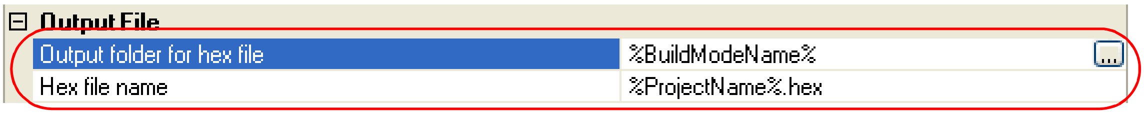
2.9.1 Set the output of a hex file

A hex file is generated by default after generating the ROMized load module.

Select the build tool node on the project tree and select the [Hex Output Options] tab on the Property panel.

The setting to output a hex file is made with the [Output File] category.

Figure 2-59. [Output File] Category



⊟ **Output File**	
Output folder for hex file	%BuildModeName%
Hex file name	%ProjectName%.hex

(1) Set the output folder

Setting the output folder is made with the [Output folder for hex file] property by directly entering in the text box or by the [...] button.

Up to 247 characters can be specified in the text box.

"%BuildModeName%" is set by default.

"%BuildModeName%" is an embedded macro. It is replaced to the build mode name.

(2) Set the output file name

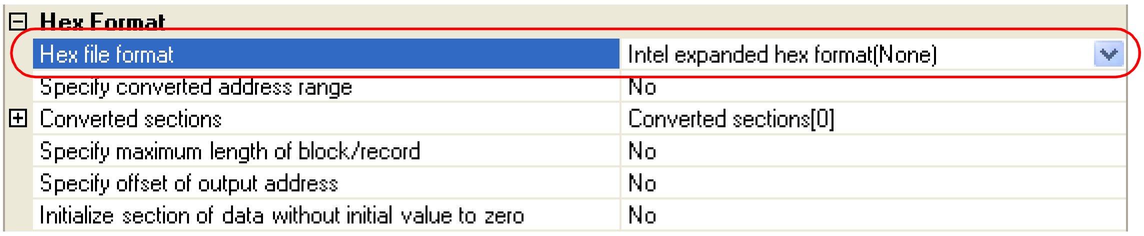
Setting the output file is made with the [Hex file name] property by directly entering in the text box.

Up to 259 characters can be specified in the text box.

"%ProjectName%.hex" is set by default.
"%ProjectName%" is an embedded macro. It is replaced to the project name.

You can also set the format of the hex file.
Select the format in the [Hex file format] property in the [Hex Format] category.

Figure 2-60. [Hex file format] Property



Hex Format	
Hex file format	Intel expanded hex format(None)
Specify converted address range	No
⊞ Converted sections	Converted sections[0]
Specify maximum length of block/record	No
Specify offset of output address	No
Initialize section of data without initial value to zero	No

You can select any of the formats below.

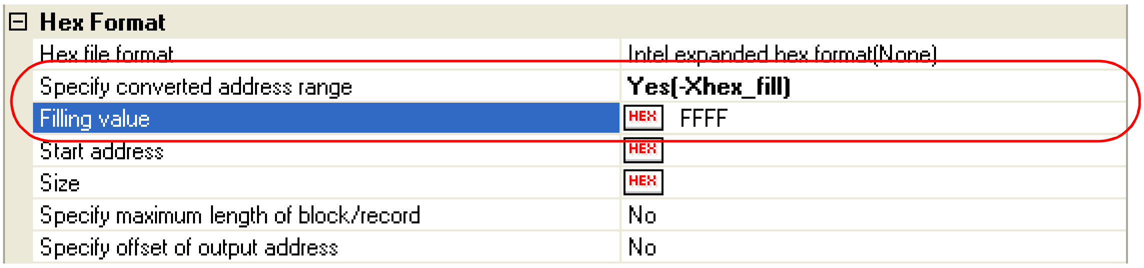
Format	Configuration
Intel expanded hex format(None)	Start address record, expanded address record, data record, and end record
Intel expanded hex format (32-bit address)(-Xhex_format=i)	Start linear address record, expanded linear start address record, start address record, expanded address record, data record, and end record
Motorola S type format(standard address)(-Xhex_format=S)	S0 record as a header record, S2 record as data record, and S8 record as end record
Motorola S type format(32-bit address)(-Xhex_format=s)	S0 record as a header record, S3 record as data record, and S7 record as end record
Expanded Tektronix hex format(-Xhex_format=T)	Data block, symbol block, and termination block

Remark See "3.4 Hex File" for details about the hex file.

2.9.2 Fill the vacant area

Select the build tool node on the project tree and select the [Hex Output Options] tab on the Property panel.
The setting to fill the vacant area is made with the [Hex Format] category.
If you select [Yes(-Xhex_fill)] in the [Specify converted address range] property, the [Filling value] property is displayed.

Figure 2-61. [Specify converted address range] and [Filling value] Property



Hex Format	
Hex file format	Intel expanded hex format(None)
Specify converted address range	Yes(-Xhex_fill)
Filling value	HEX FFFF
Start address	HEX
Size	HEX
Specify maximum length of block/record	No
Specify offset of output address	No

Enter the fill value for the vacant area directly in the text box.
The range that can be specified for the value is 00 to FFFF (2- or 4-digit hexadecimal number).
"FFFF" is set by default.

Set the address range of the area to be converted to a hex file in the [Start address] property and [Size] property.

The range that can be specified is 0 to FFFFFFFF (hexadecimal number) for the [Start address] property, and 1 to 100000000 (hexadecimal number) for the [Size] property.

Both are blank by default, so specify both values.

If either of them is blank, all the codes in the internal ROM area defined by the device file are converted into the hex format.

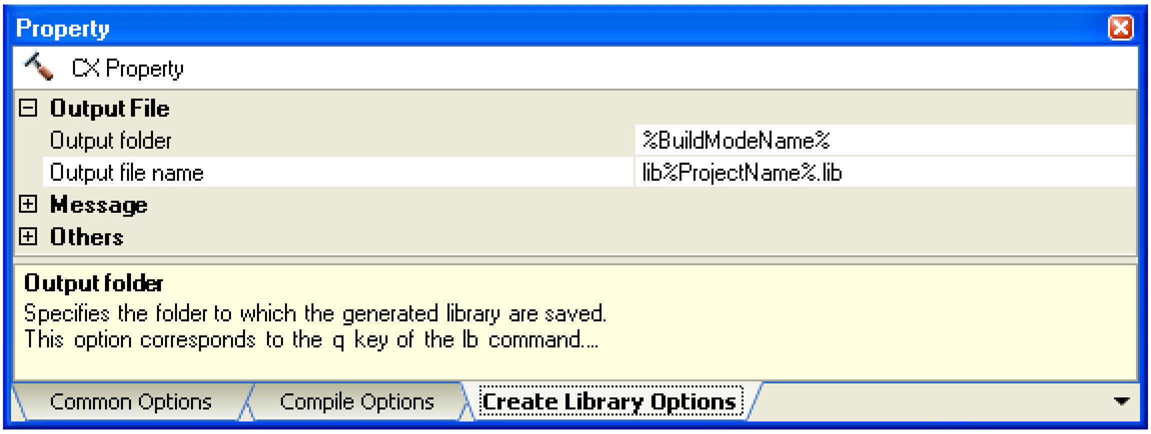
2.10 Set Create Library Options

To set options for the librarian, select the Build tool node on the project tree and select the [Create Library Options] tab on the Property panel.

You can set the various create library options by setting the necessary properties in this tab.

Caution This tab is displayed for the library project.

Figure 2-62. Property Panel: [Create Library Options] Tab



Property

CX Property

⊟ Output File

Output folder	%BuildModeName%
Output file name	lib%ProjectName%.lib

⊞ Message

⊞ Others

Output folder

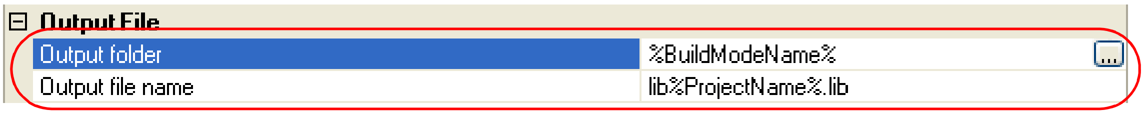
Specifies the folder to which the generated library are saved.
This option corresponds to the q key of the lb command....

2.10.1 Set the output of a library file

Select the build tool node on the project tree and select the [Create Library Options] tab on the Property panel.

The setting to output a library file is made with the [Output File] category.

Figure 2-63. [Output File] Category



⊟ Output File

Output folder	%BuildModeName%
Output file name	lib%ProjectName%.lib

(1) Set the output folder

Setting the output folder is made with the [Output folder] property by directly entering to the text box or by the [...] button.

Up to 247 characters can be specified in the text box.

"%BuildModeName%" is set by default.

"%BuildModeName%" is an embedded macro. It is replaced to the build mode name.

(2) Set the output file name

Setting the output file is made with the [Output file name] property by directly entering to the text box.

Up to 259 characters can be specified in the text box.

"lib%ProjectName%.lib" is set by default.

"%ProjectName%" is an embedded macro. It is replaced to the project name.

2.11 Set Build Options Separately

Build options are set at the project or file level.

Project level: See "2.11.1 Set build options at the project level"
File level: See "2.11.2 Set build options at the file level"

2.11.1 Set build options at the project level

To set options for build options for the project (main project or subproject), select the Build tool node on the project tree to display the Property panel.

Select the phase tab and set build options by setting the necessary properties.

Compile phase: [Compile Options] tab
Assemble phase: [Assemble Options] tab
Link phase: [Link Options] tab
ROMize phase: [ROMize Options] tab
Hex output phase: [Hex Output Options] tab
Create library phase: [Create Library Options] tab

2.11.2 Set build options at the file level

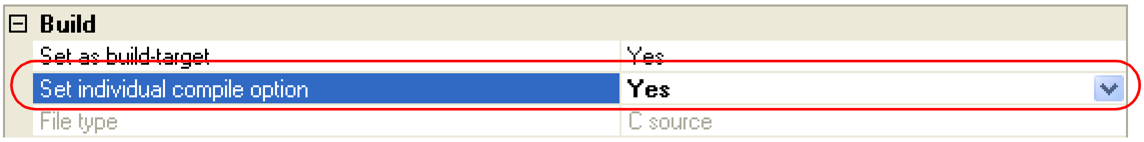
You can individually set compile and assemble options for each source file added to the project.

(1) When setting compile options for a C source file

Select the C source file on the project tree and select the [Build Settings] tab on the Property panel.

Select [Yes] in the [Set individual compile option] property in the [Build] category. The "Figure 2-65. Message Dialog Box" will open.

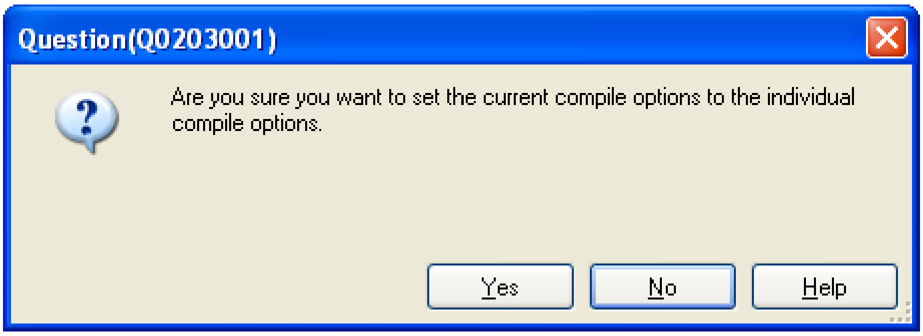
Figure 2-64. [Set individual compile option] Property



⊟ Build

Set as build-target	Yes
Set individual compile option	Yes
File type	C source

Figure 2-65. Message Dialog Box

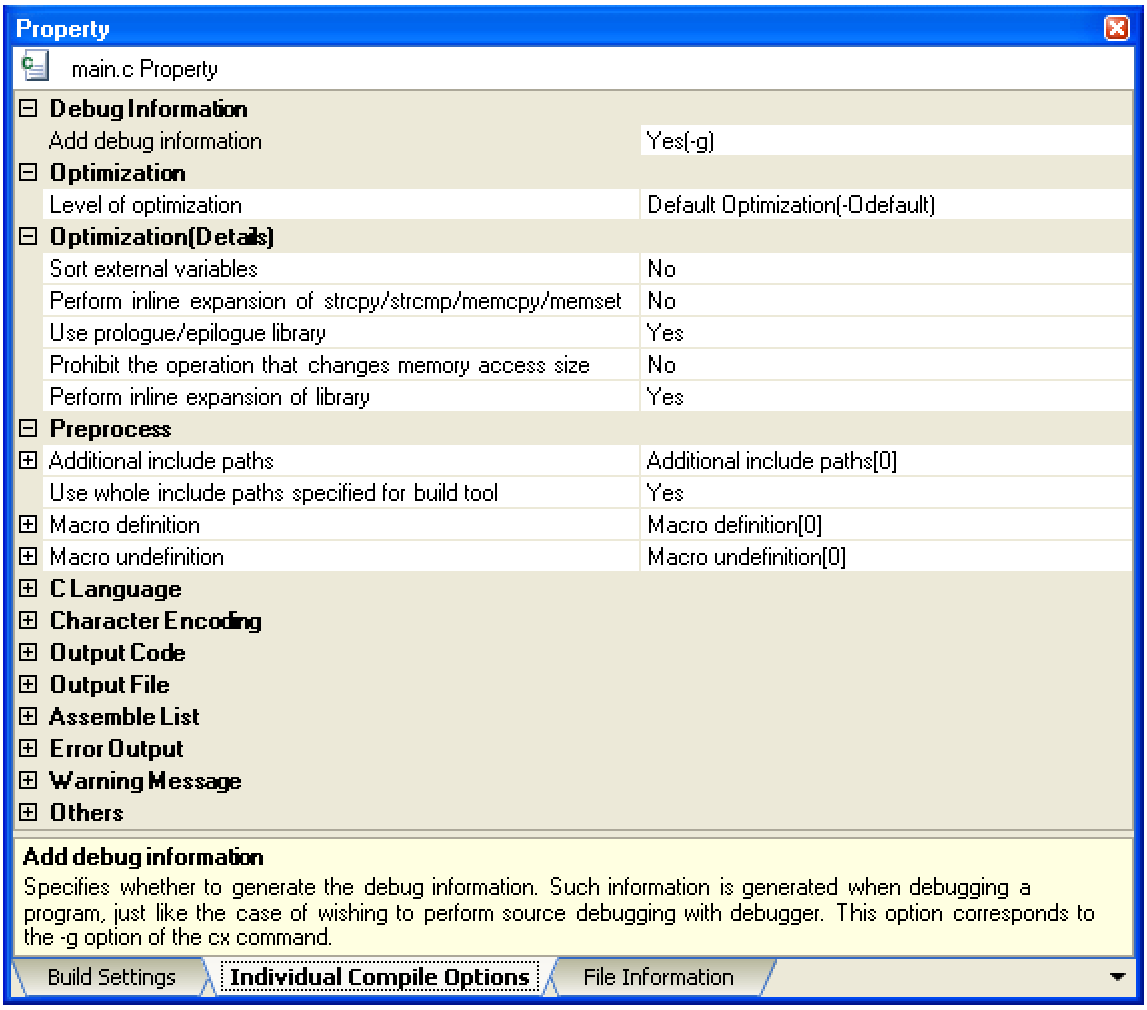


Question(Q0203001)

Are you sure you want to set the current compile options to the individual compile options.

Click [Yes] in the dialog box. The [Individual Compile Options] tab will be displayed.

Figure 2-66. Property Panel: [Individual Compile Options] Tab



⊟ **Debug Information**	
Add debug information	Yes(-g)
⊟ **Optimization**	
Level of optimization	Default Optimization(-Odefault)
⊟ **Optimization(Details)**	
Sort external variables	No
Perform inline expansion of strcpy/strcmp/memcpy/memset	No
Use prologue/epilogue library	Yes
Prohibit the operation that changes memory access size	No
Perform inline expansion of library	Yes
⊟ **Preprocess**	
⊞ Additional include paths	Additional include paths[0]
Use whole include paths specified for build tool	Yes
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]
⊞ **C Language**	
⊞ **Character Encoding**	
⊞ **Output Code**	
⊞ **Output File**	
⊞ **Assemble List**	
⊞ **Error Output**	
⊞ **Warning Message**	
⊞ **Others**	

Add debug information
Specifies whether to generate the debug information. Such information is generated when debugging a program, just like the case of wishing to perform source debugging with debugger. This option corresponds to the -g option of the cx command.

You can set compile options for the C source file by setting the necessary properties in this tab.

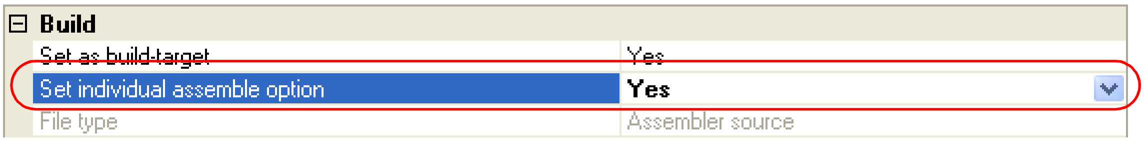
Note that this tab takes over the settings of the [Common Options] tab and [Compile Options] tab by default except the properties shown below.

- [Additional include paths] and [Use whole include paths specified for build tool] in the [Preprocess] category
- [Object module file name] in the [Output File] category

(2) When setting assemble options for an assembler source file

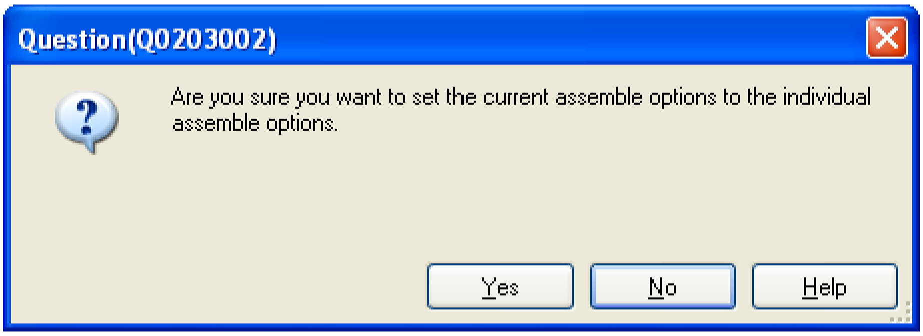
Select the assembler source file on the project tree and select the [Build Settings] tab on the Property panel. Select [Yes] in the [Set individual assemble option] property in the [Build] category. The "Figure 2-68. Message Dialog Box" will open.

Figure 2-67. [Set individual assemble option] Property



⊟ **Build**	
Set as build-target	Yes
Set individual assemble option	**Yes**
File type	Assembler source

Figure 2-68. Message Dialog Box

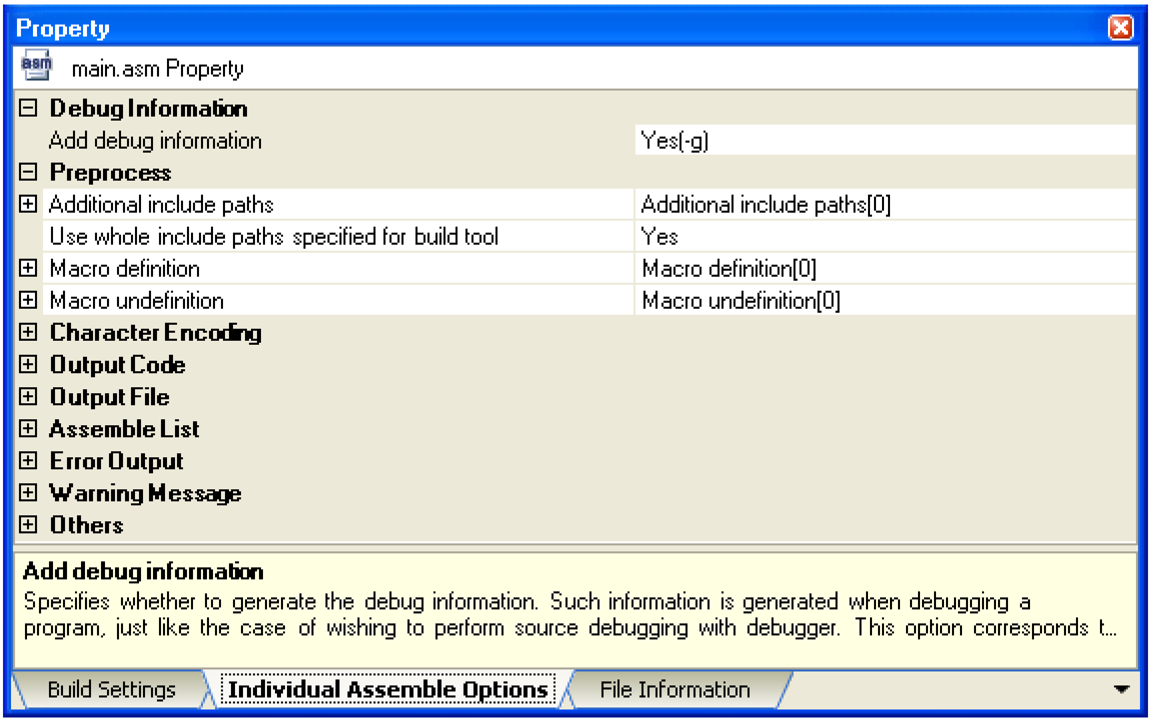


Question(Q0203002)

Are you sure you want to set the current assemble options to the individual assemble options.

Click [Yes] in the dialog box. The [Individual Assemble Options] tab will be displayed.

Figure 2-69. Property Panel: [Individual Assemble Options] Tab



Property

main.asm Property

⊟ **Debug Information**	
Add debug information	Yes(-g)
⊟ **Preprocess**	
⊞ Additional include paths	Additional include paths[0]
Use whole include paths specified for build tool	Yes
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]
⊞ **Character Encoding**	
⊞ **Output Code**	
⊞ **Output File**	
⊞ **Assemble List**	
⊞ **Error Output**	
⊞ **Warning Message**	
⊞ **Others**	

Add debug information
Specifies whether to generate the debug information. Such information is generated when debugging a program, just like the case of wishing to perform source debugging with debugger. This option corresponds t...

You can set assemble options for the assembler source file by setting the necessary properties in this tab.

Note that this tab takes over the settings of the [Common Options] tab and [Compile Options] tab/[Assemble Options] tab by default except the properties shown below.

- [Additional include paths] and [Use whole include paths specified for build tool] in the [Preprocess] category
- [Object module file name] in the [Output File] category

2.12 Prepare for Implementing Boot-Flash Relink Function

Depending on the system, in addition to the area which cannot be rewritten/replaced (boot area), there are occasions when you can use the area which can be rewritten/replaced (flash area), such as the flash or external ROM.

In these kinds of systems, when you wish to change the program in the flash area, a function called the "relink function" correctly performs function calls between the boot area and flash area without rebuilding the program in the boot area.

By creating load module files for the boot area and flash area, you can implement the relink function.

The method to implement the relink function is shown below.

Remark See "B.1.6 Boot-flash re-link function" for details about the relink function and how to implement it.

2.12.1 Prepare the build target files

(1) Specify the $ext_func control instruction

Describe the $ext_func control instruction in the source file for the boot area.

With the $ext_func control instruction, specify the ID value for the target function (the actual function exists in the flash area and is called from the boot area).

Remark In order to prevent description mistakes and inconsistencies between source files, it is recommend that you organize the $ext_func control instruction in a single file, and regardless of the boot area or flash area, include that file in all source files by using the $ext_func control instruction (or #include directive when describing in C language).

(2) Prepare the startup routines

Prepare the startup routines for the projects for both the boot area and flash area.

Each startup routine must perform the following processing.

- Setting tp, gp, and ep values in the boot area
- Calling the _rcopy function to initialize the RAM area to be used for the boot area
- Branching from the boot area to the startup routine of the flash area
- Calling the _rcopy function to initialize the RAM area to be used for the flash area
- Moving to the processing of the flash area

Remarks 1. If tp, gp, and ep are not used in the boot area, the values may be set in the flash area.
If ROMization processing is not performed, the _rcopy function call is not required.

2. Use the same address values in the boot area and flash area for the tp, gp, and ep values.
These values may be different, but in this case the values must be set each time control has been transferred between an instruction code in the boot area and one in the flash area.

(3) Prepare the link directive files

Prepare link directive files for the projects for both the boot area and flash area.

Remark You can use the same link directive file with the boot area and flash area, but since the description will become complicated, it is recommended that the separate link directive file for each area is used.

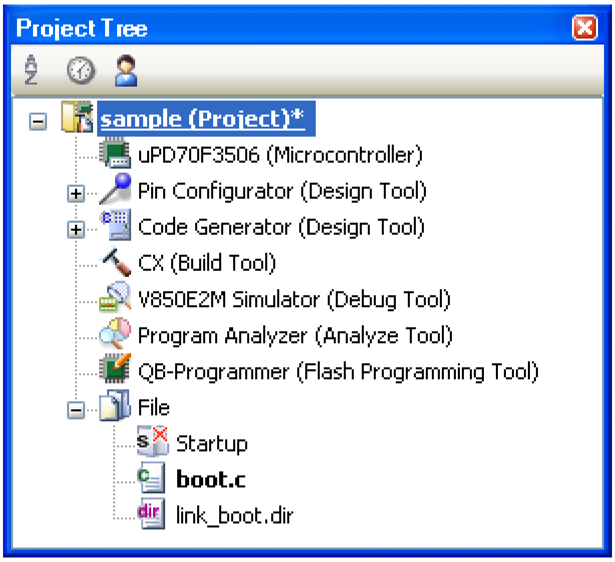
2.12.2 Set the boot area project

(1) Create the boot area project

Create the project for the boot area and add the build target files to the project.

Add the startup routine directly below the Startup node.

Figure 2-70. Boot Area Project


Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Startup
boot.c
link_boot.dir

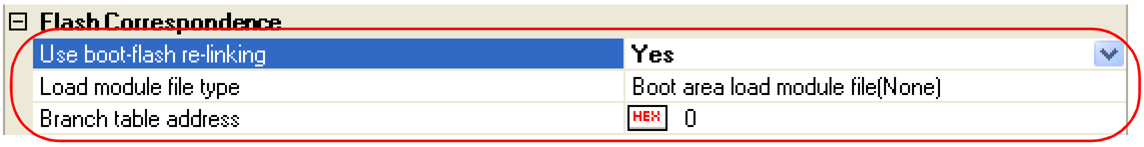
(2) **Set the build options for the boot area project**

Select the build tool node on the project tree and select the [Common Options] tab on the Property panel.

The settings for the build options are made in the [Flash Correspondence] category.

Select [Yes] in the [Use boot-flash re-linking] property. The [Load module file type] property and [Branch table address] property will be displayed.

Figure 2-71. [Flash Correspondence] Category in Boot Area



Flash Correspondence

Use boot-flash re-linking	Yes
Load module file type	Boot area load module file(None)
Branch table address	HEX 0

Select [Boot area load module file(None)] (default) in the [Load module file type] property.

Specify the start address of the branch table (address in the flash area) in the [Branch table address] property.

The range that can be specified for the value is 0 to FFFFFFFF (hexadecimal number).

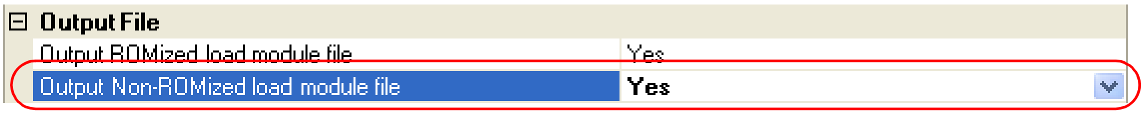
"0" is set by default.

Caution The load module file before ROMization processing must be specified for the flash area project. CubeSuite performs ROMization processing by default. If ROMization processing is performed, the load module file before ROMization processing must be output.

Select the [ROMize Options] tab and [Yes] in the [Output Non-ROMized load module file] property in the [Output File] category.

The file name is the load module file name with string "_NonROMize" added.

Figure 2-72. [Output Non-ROMized load module file] Property



Output File

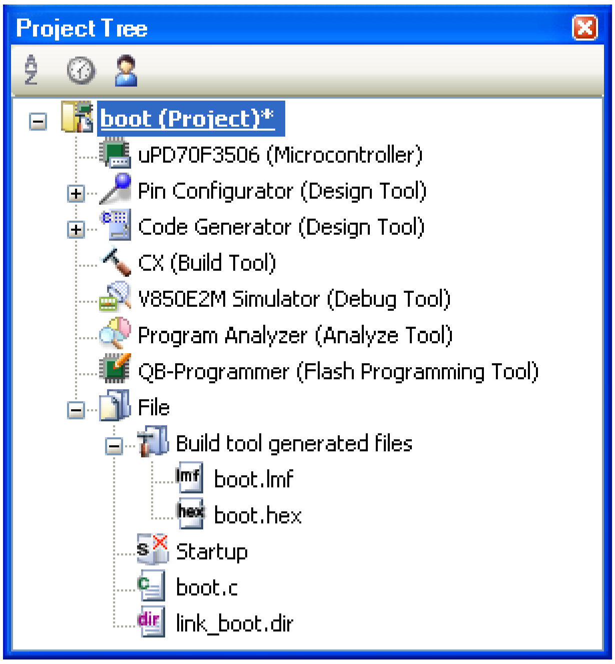
Output ROMized load module file	Yes
Output Non-ROMized load module file	Yes

(3) Run a build of the boot area project

When you run a build of the boot area project, the load module file is generated.

The hex file is also generated.

Figure 2-73. Generated Files for Boot Area



Project Tree
boot (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Build tool generated files
boot.lmf
boot.hex
Startup
boot.c
link_boot.dir

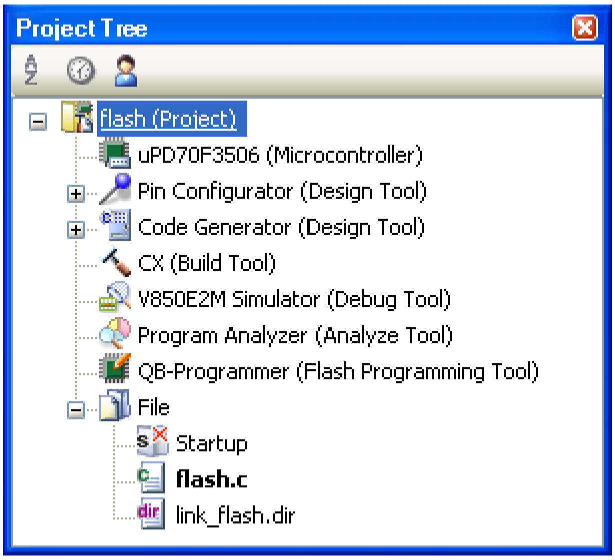
2.12.3 Set the flash area project

(1) Create the flash area project

Create the project for the boot area and add the build target files to the project.

Add the startup routine directly below the Startup node.

Figure 2-74. Flash Area Project



Project Tree
flash (Project)
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Startup
flash.c
link_flash.dir

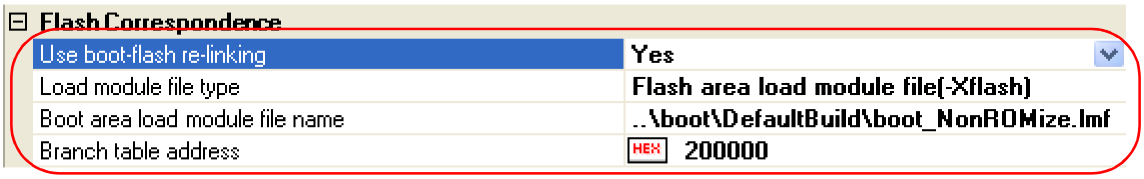
(2) Set the build options for the flash area project

Select the build tool node on the project tree and select the [Common Options] tab on the Property panel.

The settings for the build options are made in the [Flash Correspondence] category.

Select [Yes] in the [Use boot-flash re-linking] property. The [Load module file type] property and [Branch table address] property will be displayed.

Figure 2-75. [Flash Correspondence] Category in Flash Area



⊟ Flash Correspondence

Use boot-flash re-linking	Yes
Load module file type	Flash area load module file(-Xflash)
Boot area load module file name	..\boot\DefaultBuild\boot_NonROMize.lmf
Branch table address	HEX 200000

Select [Flash area load module file(-Xflash)] in the [Load module file type] property. The [Boot area load module file name] property will be displayed.

Specify the load module file for the boot area.

Caution Specify the load module file before ROMization processing in this property.
An error will occur if the load module file after ROMization processing is specified.

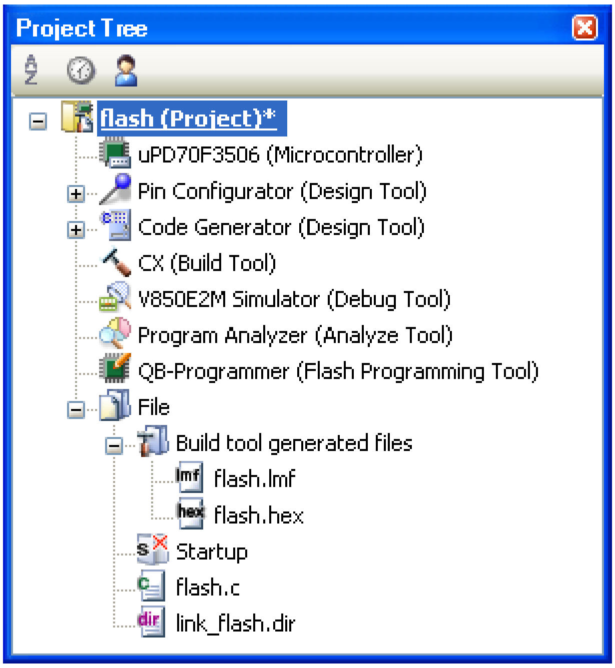
Specify the start address of the branch table (same as the address specified in the boot area project) in the [Branch table address] property.

(3) Run a build of the flash area project

When you run a build of the flash area project, the load module file which implements the re-link function is created.

The boot area hex file (the same content as the file generated in "2.12.2 Set the boot area project") and flash area hex file are also generated.

Figure 2-76. Generated Files for Flash Area



Project Tree

```
⊟ flash (Project)*
    uPD70F3506 (Microcontroller)
  ⊞ Pin Configurator (Design Tool)
  ⊞ Code Generator (Design Tool)
    CX (Build Tool)
    V850E2M Simulator (Debug Tool)
    Program Analyzer (Analyze Tool)
    QB-Programmer (Flash Programming Tool)
  ⊟ File
    ⊟ Build tool generated files
        flash.lmf
        flash.hex
      Startup
      flash.c
      link_flash.dir
```

2.13 Allocate Variables to Optimum Section

To allocate the variables to the optimum section, the symbol information file (a text-format file that allocation information for the variable defined in the C source file is described) can be used.

The variables can be allocated to the optimum section without editing the C source file by generating the symbol information file and referencing the file when compiling.

The procedures for performing this operation are described below.

- Generating the symbol information file automatically and relocating the variables
- Editing and using the auto-generated symbol information file

Confirm that the build has completed successfully and the load module file has been generated before using this function.

Remark See "B.1.4 Symbol information file" for detail about the symbol information file.

(1) Generating the symbol information file automatically and relocating the variables

Below is the procedure for generating the symbol information file automatically and referring that file to allocate the variables, via one build.

(a) Set the generation of the symbol information file

Select the build tool node on the project tree and select the [Link Options] tab on the Property panel.

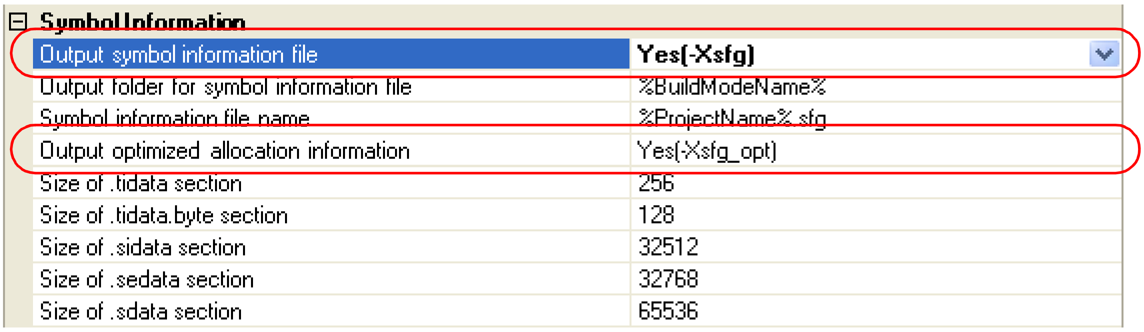
Select [Yes(-Xsfg)] in the [Output symbol information file] property in the [Symbol Information] category. The empty symbol information file will be generated and added to the project (it will be also shown in the Files node of the project tree).

The output destination is the file set in the [Output folder for symbol information file] property and [Symbol information file name] property.

Select [Yes(-Xsfg_opt)] in the [Output optimized allocation information] property.

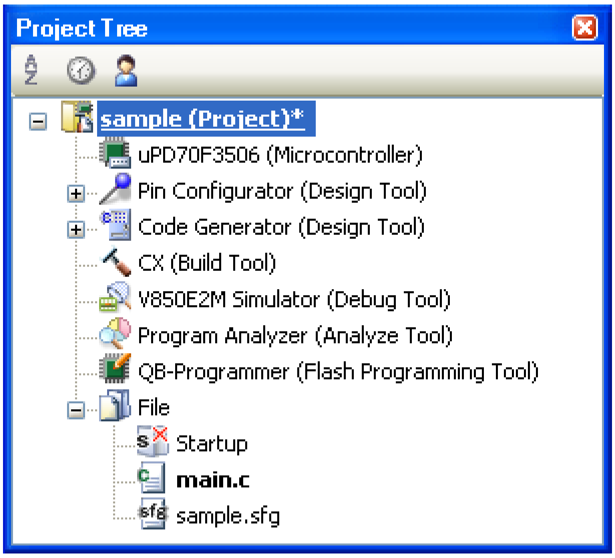
Remark If a symbol information file with the same name already exists, the build will be configured to use it.

Figure 2-77. [Output symbol information file] and [Output optimized allocation information] Property



⊟ Symbol Information	
Output symbol information file	Yes(-Xsfg)
Output folder for symbol information file	%BuildModeName%
Symbol information file name	%ProjectName%.sfg
Output optimized allocation information	Yes(-Xsfg_opt)
Size of .tidata section	256
Size of .tidata.byte section	128
Size of .sidata section	32512
Size of .sedata section	32768
Size of .sdata section	65536

Figure 2-78. Project Tree Panel (After Generating Symbol Information File)



Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Startup
main.c
sample.sfg

The settings of the output folder and file of the symbol information file are can be changed.

<1> Set the output folder

Setting the output folder is made with the [Output folder for symbol information file] property by directly entering in the text box or by the [...] button.
Up to 247 characters can be specified in the text box.
"%BuildModeName%" is set by default.
"%BuildModeName%" is an embedded macro. It is replaced to the build mode name.

<2> Set the output file name

Setting the output file is made with the [Symbol information file name] property by directly entering in the text box.
Up to 259 characters can be specified in the text box.
"%ProjectName%.sfg" is set by default.
"%ProjectName%" is an embedded macro. It is replaced to the project name.
If this property is changed, the empty symbol information file is generated and added to the project (it will be also shown in the Files node of the project tree).

(b) Run a build of the project

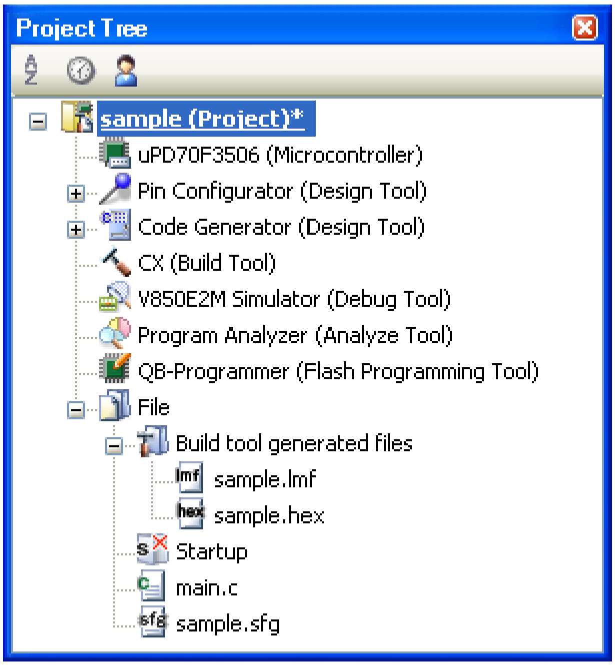
Run a build of the project.
The symbol information file is generated. And then it is inputted into CX automatically and a rebuild is executed again.

Remarks 1. The symbol information file in "(a) Set the generation of the symbol information file" is overwritten by running a build.

2. Since the objects are generated anew using the symbol information file, the second build will be a rebuild.

If the build completes successfully, the load module file is generated with the variables allocated.

Figure 2-79. Project Tree Panel (After Generating Load Module File)


Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Build tool generated files
sample.lmf
sample.hex
Startup
main.c
sample.sfg

(2) Editing and using the auto-generated symbol information file

Users can edit the symbol information file.

Below is the procedure for editing the generated symbol information file in "(1) Generating the symbol information file automatically and relocating the variables" by the user and referring that file to allocate variables.

(a) Editing the symbol information file

Edit the symbol information file generated automatically in "(1) Generating the symbol information file automatically and relocating the variables".

Describe the symbol information file according to the following format.

Describe the variables in the order of priority, from highest use frequency.

```
[tidata.byte]:Size_section-sizebytes
[tidata.word]:Size_section-sizebytes
[sidata]:Size_section-sizebytes
[sedata]:Size_section-sizebytes
[sdata]:Size_section-sizebytes
global-variable,reference-count,byte-size
static-variable-in-file, reference-count,byte-size,file-name-including-path
static-variable-in-function,reference-count,byte-size,file-name-including-
path,function-name
```

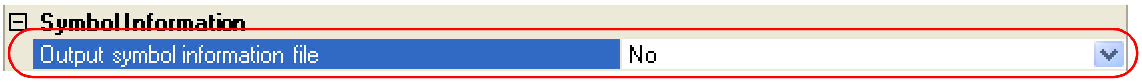
Remark See "3.3 Symbol Information File" for detail about the format of the symbol information file.

(b) Set the generation of the symbol information file

Select the build tool node on the project tree and select the [Link Options] tab on the Property panel.

Select [No] in the [Output symbol information file] property.

Figure 2-80. [Output symbol information file] Property



Symbol Information

Output symbol information file: No

(c) **Run a build of the project**

Run a build of the project.

The load module file is generated with the variables allocated according to the specified contents in the symbol information file.

Caution If the file with the extension of "sfg" is added to the project, it is treated as a symbol information file. It is also treated as a symbol information file if it is added below the Startup node.

When adding a symbol information file to the project, if a symbol information file has already been added, then only the latest symbol information file to be added is targeted by a build; any such files added prior to this one will not be targeted.

When setting the symbol information file that is not targeted by a build as a build target, if other symbol information files have also been added then the file will be targeted by the build, and the others will not be targeted.

2.14 Create a Multi-Core Load Module

CubeSuite is able to create a single load module from a program that runs on multiple cores (multi-core).
The procedures for performing this operation are described below.

Remark See "CubeSuite Coding for CX Compiler" for details about multi-core programming.

(1) Create a link directive file

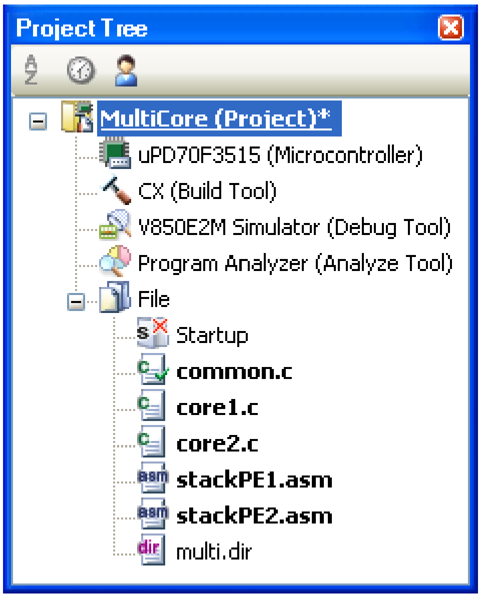
The link directive file must be created together with each target system.
Create the file with reference to the sample link directive file below.

CubeSuite install folder\CubeSuite\CX\Vx.xx\smp\850\multi_smp.dir

(2) Create a project

Create a project and add build target files (source file, link directive file, etc.) to the project.

Figure 2-81. Project for Multi-Core

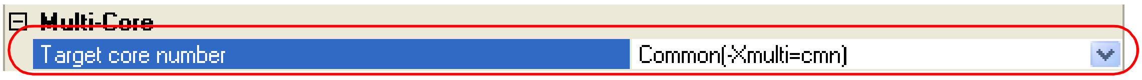


Remark To use other than the standard startup routine, add the file to the Startup node (see "2.3.1 Set a startup routine").

(3) Set build options

Select each core file on the project tree and select the [Build Settings] tab on the Property panel.
Select the corresponding core number ([Core *n* (-Xmulti=pe*n*)]) in the [Target core number] property in the [Multi-Core] category (*n*: number of cores on the target CPU).

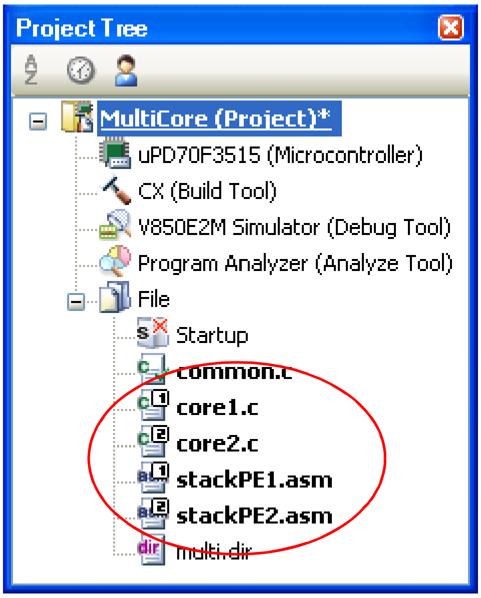
Figure 2-82. [Target core number] Property



Target core number: Common(-Xmulti=cmn)

The icon of the file specifying the core number changes to an icon with the core number added.

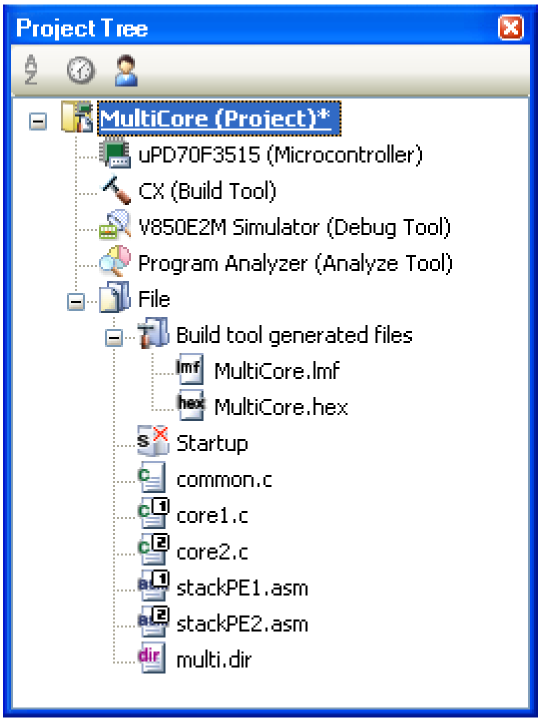
Figure 2-83. Project for Multi-Core (After Setting Build Options)



(4) Run a build of the project

Run a build to generate the load module file for multi-core.

Figure 2-84. Project for Multi-Core (After Running Build)



2.15 Make Settings for Build Operations

This section explains operations on a build.

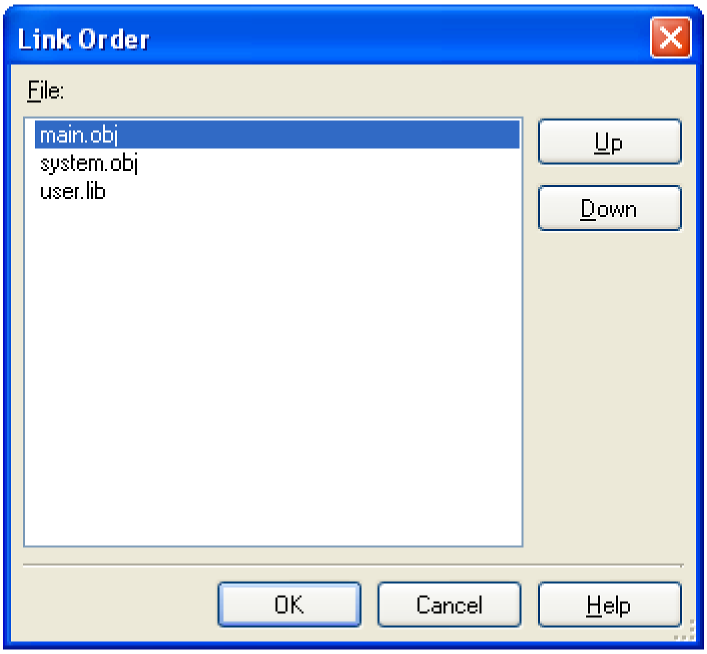
- Set the link order of files
- Change the file build order of subprojects
- Display a list of build options
- Change the file build target project
- Add a build mode
- Change the build mode
- Delete a build mode
- Set the current build options as the standard for the project

2.15.1 Set the link order of files

The link order of object module files and library files is decided automatically, but you can also set the order.

On the project tree, select the Build tool node, and then select [Set Link Order...] from the context menu. The Link Order dialog box will open.

Figure 2-85. Link Order Dialog Box



The names of the following files are listed in [File] in the order that the files are input to the linker.

- Object module files which are generated from the source files added to the selected main project or subproject
- Object module files which are added directly to the project tree of the selected main project or subproject
- Library files which are added directly to the project tree of the selected main project or subproject

Remark The default order is the order that the files are added to the project.
Object module files generated from newly added source files and newly added object module files are added after the last object module file in the list.
Newly added library files are added to the end of the list.

By changing the display order of the files, you can set the input order of the files to the linker.

To change the display order, use the [Up] and [Down] buttons, or drag and drop the file names.

After changing the display order, click the [OK] button.

2.15.2 Change the file build order of subprojects

Builds are run in the order of subproject, main project, but when there are multiple subprojects added, the build order of subprojects is their display order on the project tree.

To change the display order of the subprojects on the project tree, drag the subproject to be moved and drop it on the desired location.

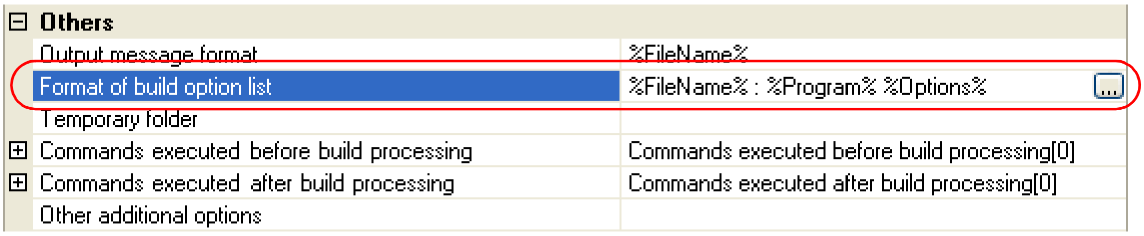
2.15.3 Display a list of build options

You can display the list of build options set currently on the Property panel for the project (main project and subproject).

Select [Build Options List] from the [Build] menu. The current settings of the options for the project will be displayed on the [Build Tool] tab from the Output panel in the build order.

Remark You can change the display format of the build option list.
Select the build tool node on the project tree and select the [Common Options] tab on the Property panel.
Set the [Format of build option list] property in the [Others] category.

Figure 2-86. [Format of build option list] Property



⊟ Others	
Output message format	%FileName%
Format of build option list	%FileName% : %Program% %Options%
Temporary folder	
⊞ Commands executed before build processing	Commands executed before build processing[0]
⊞ Commands executed after build processing	Commands executed after build processing[0]
Other additional options	

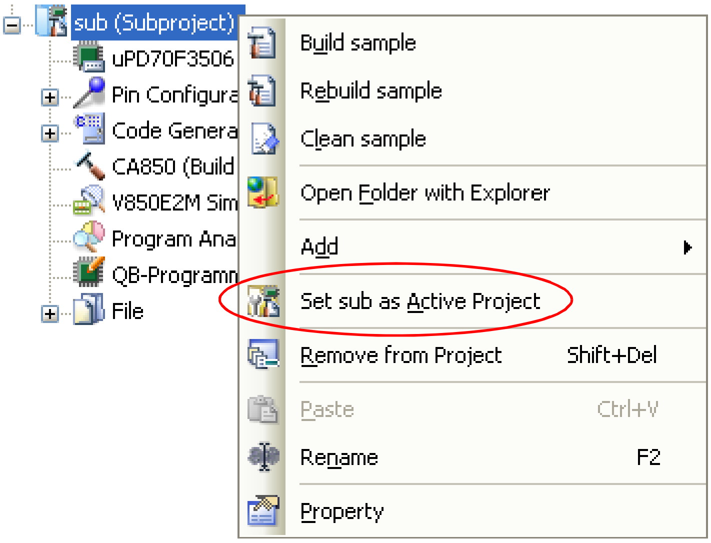
"%FileName% : %Program% %Options%" is set by default.
"%FileName%", "%Program%", and "%Options%" are embedded macros. They are replaced to the file name being built, program name under execution, and command line option under build execution.

2.15.4 Change the file build target project

When running a build that targets a specific project (main project or subproject), you must set that project as the "active project".

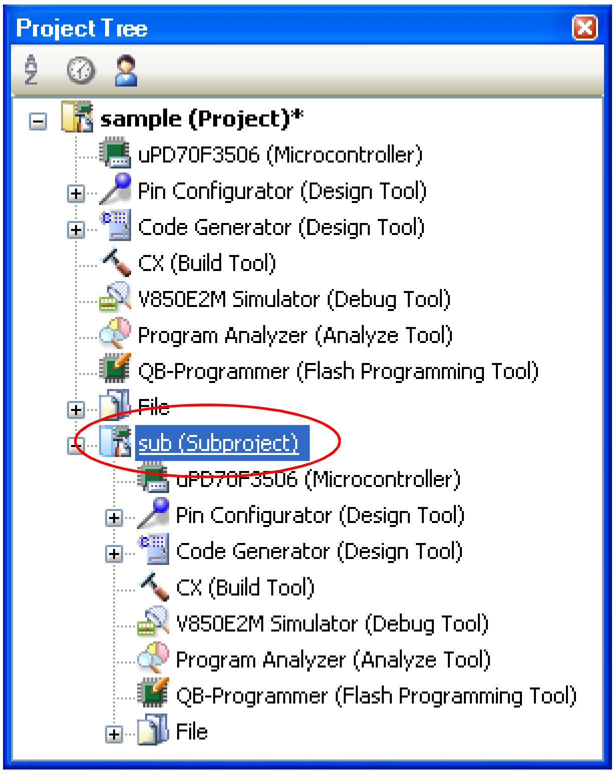
To set the active project, select the main project or subproject to be set as the active project on the project tree and select [Set *selected subproject* as Active Project] from the context menu.

Figure 2-87. [Set *selected project* as Active Project] Item



When a project is set as the active project, that project is underlined.

Figure 2-88. Active Project



Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
sub (Subproject)
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File

Remarks 1. Immediately after creating a project, the main project is the active project.

2. When you remove a subproject that set as the active project from a project, the main project will be the active project.

2.15.5 Add a build mode

When you wish to change the build options and macro definitions according to the purpose of the build, you can collectively change those settings.

Build options and macro definition settings are organized into what is called "build mode", and by changing the build mode, you eliminate the necessity of changing the build options and macro definition settings every time.

The build mode prepared by default is only "DefaultBuild". Add a build mode according to the purpose of the build.

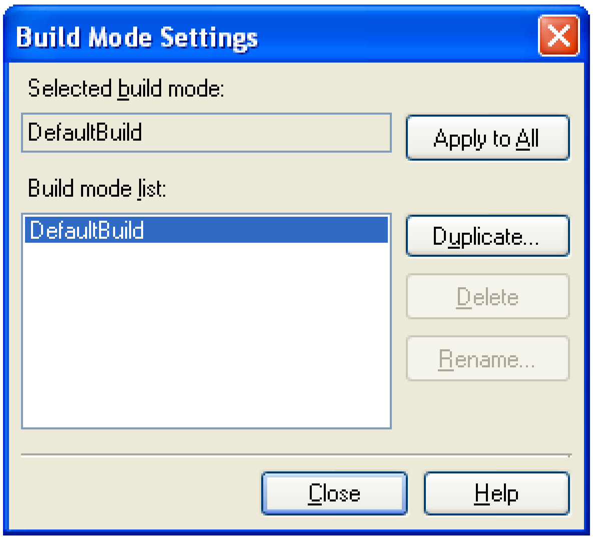
The method to add a build mode is shown below.

(1) Create a new build mode

Creating a new build mode is performed with duplicating an existing build mode.

Select [Build Mode Settings...] from the [Build] menu. The Build Mode Settings dialog box will open.

Figure 2-89. Build Mode Settings Dialog Box

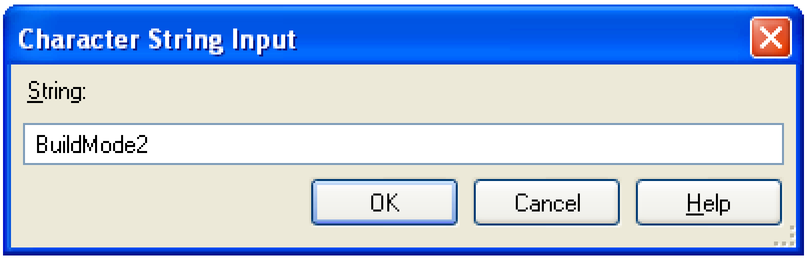


Selected build mode: DefaultBuild

Build mode list: DefaultBuild

Select the build mode to be duplicated from the build mode list and click the [Duplicate...] button. The Character String Input dialog box will open.

Figure 2-90. Character String Input Dialog Box

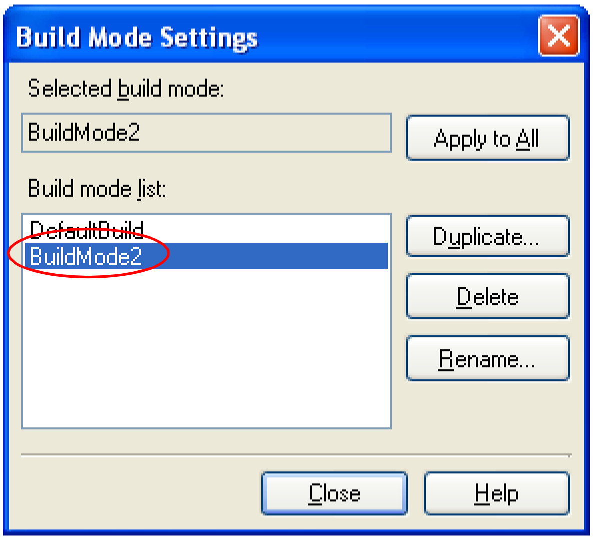


String: BuildMode2

In the dialog box, enter the name of the build mode to be created and then click the [OK] button. The build mode with that name will be duplicated.

The created build mode is added to the build modes of the main project and all the subprojects which belong to the project.

Figure 2-91. Build Mode Settings Dialog Box (After Adding Build Mode)



(2) Change the build mode

Change the build mode to the newly created build mode (see "2.15.6 Change the build mode").

(3) Change the setting of the build mode

Select the build tool node on the project tree and change the build options and macro definition settings on the Property panel.

Remark Creating a build mode is regarded a project change.
When closing the project, you will be asked to confirm whether or not to save the build mode.

2.15.6 Change the build mode

When you wish to change the build options and macro definitions according to the purpose of the build, you can collectively change those settings.

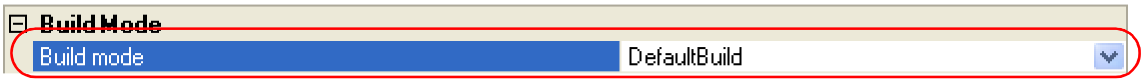
Build options and macro definition settings are organized into what is called "build mode", and by changing the build mode, you eliminate the necessity of changing the build options and macro definition settings every time.

(1) When changing the build mode for the main project or subprojects

Select the Build tool node of the target project on the project tree and select the [Common Options] tab on the Property panel.

Select the build mode to be changed to on the [Build Mode] property in the [Build Mode] category.

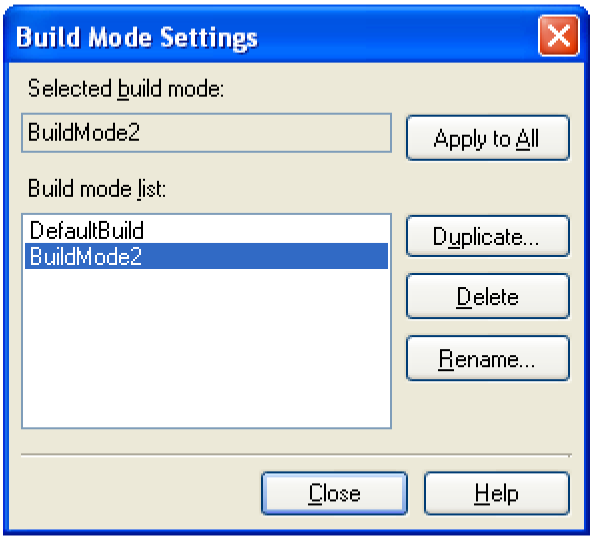
Figure 2-92. [Build mode] Property



(2) When changing the build mode for the entire project

Select [Build Mode Settings...] from the [Build] menu. The Build Mode Settings dialog box will open.

Figure 2-93. Build Mode Settings Dialog Box



Select the build mode to be changed from the build mode list. The selected build mode will be displayed in [Selected build mode].

Click the [Apply to All] button. The build mode for the main project and all the subprojects which belong to the project will be changed to the build mode selected in the dialog box.

Caution For projects that the selected build mode does not exist, the build mode is duplicated from "DefaultBuild" with the selected build mode name, and the build mode is changed to the duplicated build mode.

Remarks 1. The build mode prepared by default is only "DefaultBuild".
See "2.15.5 Add a build mode" for the method of adding a build mode.

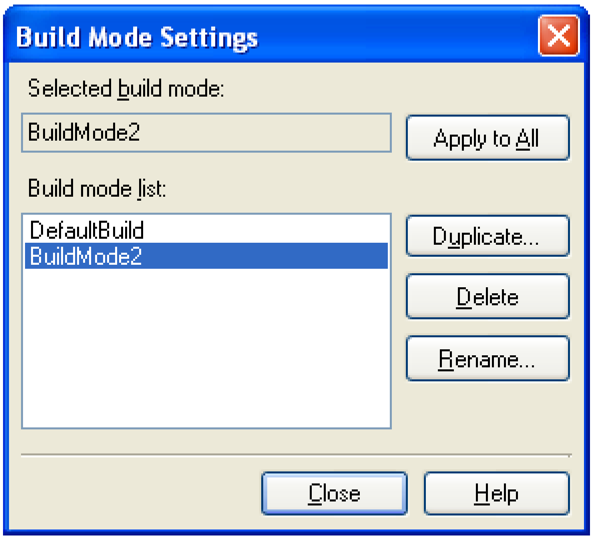
2. You can change the name of the build mode by selecting the build mode from the build mode list and clicking the [Rename...] button.
However, you cannot change the name of "DefaultBuild".

2.15.7 Delete a build mode

Deleting a build mode is performed with the Build Mode Settings dialog box.

Select [Build Mode Settings...] from the [Build] menu. The dialog box will open.

Figure 2-94. Build Mode Settings Dialog Box



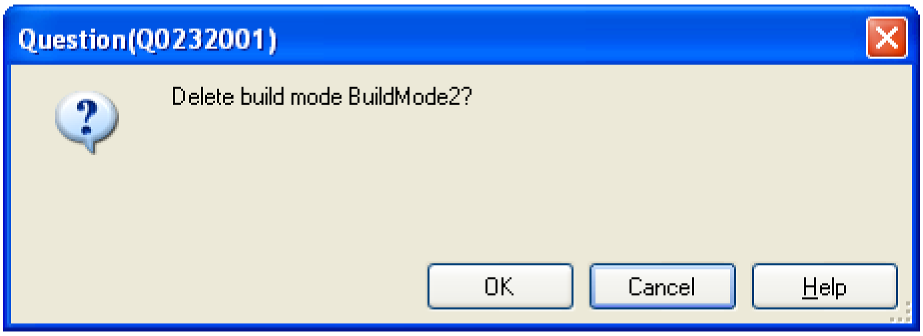
Build Mode Settings

Selected build mode: BuildMode2

Build mode list:
- DefaultBuild
- BuildMode2

Select the build mode to be deleted from the build mode list and click the [Delete] button. The Message dialog box below will open.

Figure 2-95. Message Dialog Box



Question(Q0232001)

Delete build mode BuildMode2?

To continue with the operation, click the [OK] button in the dialog box.

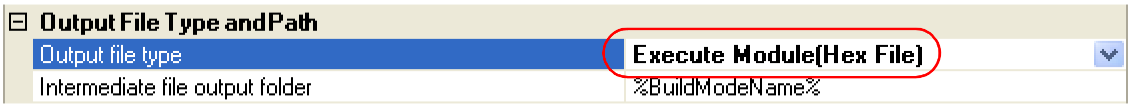
The selected build mode is deleted from the project.

Caution You cannot delete "DefaultBuild".

2.15.8 Set the current build options as the standard for the project

On the Property panel, if you add a change to the settings for the standard build options, the value of the property will be displayed in boldface.

Figure 2-96. Property Panel (After Changing Standard Build Option)

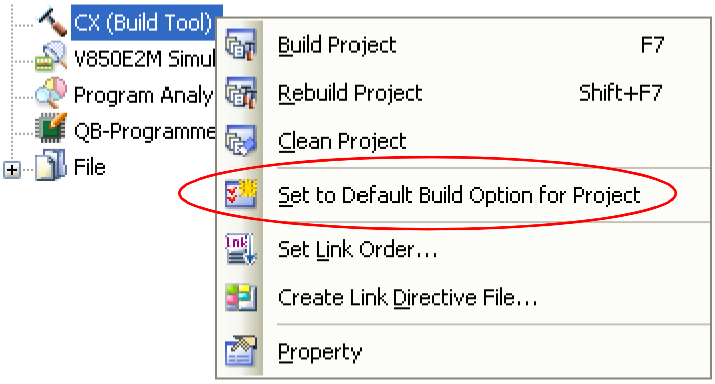


⊟ **Output File Type and Path**

Output file type	**Execute Module(Hex File)**
Intermediate file output folder	%BuildModeName%

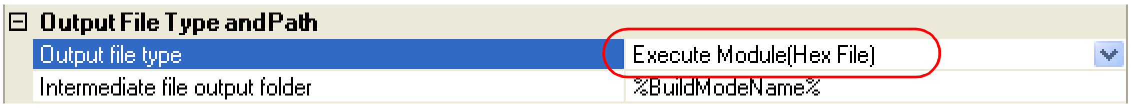
To make the build options for the currently selected project (main project or subproject) the standard build options (remove the boldface), select the Build tool node on the project tree and select [Set to Default Build Option for Project] from the context menu.

Figure 2-97. [Set to Default Build Option for Project] Item



The values of the properties after setting them as the standard build option are as shown below.

Figure 2-98. Property Panel (After Setting Standard Build Option)



⊟ **Output File Type and Path**

Output file type	Execute Module(Hex File)
Intermediate file output folder	%BuildModeName%

Caution When the main project is selected, only the main project settings are made.Even if subprojects are added, their settings are not made.

2.16 Run a Build

This section explains operations related to running a build.

(1) Build types

The following types of builds are available.

Table 2-1. Build Types

Type	Description
Build	Out of build target files, runs a build of only updated files. -> See "2.16.1 Run a build of updated files".
Rebuild	Runs a build of all build target files. -> See "2.16.2 Run a build of all files".
Rapid build	Runs a build in parallel with the change of the build setting. -> See "2.16.3 Run a build in parallel with other operations".
Batch build	Runs builds in batch with the build modes that the project has. -> See "2.16.4 Run builds in batch with build modes".

Remarks 1. Builds are run in the order of subproject, main project.
Subprojects are built in the order that they are displayed on the project tree (see "2.15.2 Change the file build order of subprojects").

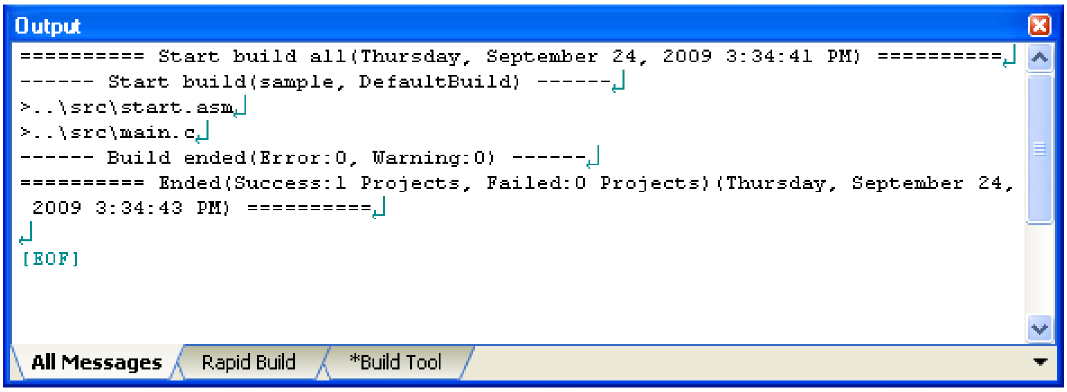
2. If there are files being edited with the Editor panel when running a build, rebuild, or batch build, then all these files are saved.

(2) Displaying execution results

The execution results of the build (output messages of the build tool) are displayed in each tab on the Output panel.

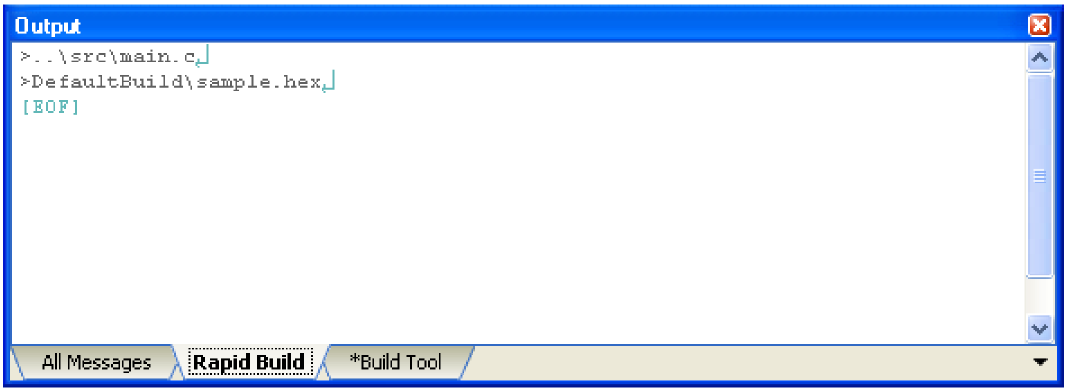
- Build, rebuild, or batch build
  -> [All Messages] tab and [Build Tool] tab
- Rapid build
  -> [Rapid Build] tab

Figure 2-99. Build Execution Results (Build, Rebuild, or Batch Build)



```
========== Start build all(Thursday, September 24, 2009 3:34:41 PM) ==========↲
------ Start build(sample, DefaultBuild) ------↲
>..\src\start.asm↲
>..\src\main.c↲
------ Build ended(Error:0, Warning:0) ------↲
========== Ended(Success:1 Projects, Failed:0 Projects)(Thursday, September 24,
 2009 3:34:43 PM) ==========↲
↲
[EOF]
```

Figure 2-100. Build Execution Results (Rapid Build)



```
>..\src\main.c
>DefaultBuild\sample.hex
[EOF]
```

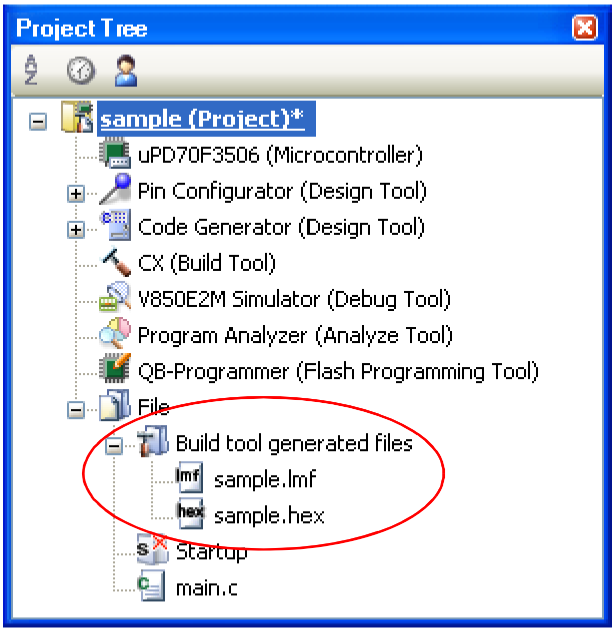
Remarks 1. The text in the [Rapid Build] tab becomes dimmed.

2. When a file name or line number can be obtained from the output messages, if you double click on the message, you can jump to the relevant line in the file.

3. If you press the [F1] key when the cursor is on a line displaying the warning or error message, you can display the help related to that line's message.

Files generated by the build tool appear under the Build tool generated files node on the Project Tree panel.

Figure 2-101. Build Tool Generated Files


Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Tool)
File
Build tool generated files
sample.lmf
sample.hex
Startup
main.c

Remark Files displayed under the Build tool generated files node are as follows.

- For other than the library project

  Load module file (*.lmf)

  Link map file (*.map)

  Hex file (*.hex)

  Error message file (*.err)

- For the library project

  Library file (*.lib)

Error message file (*.err)

Caution The Build tool generated files node is created during build.
This node will no longer appear if you reload the project after building.

2.16.1 Run a build of updated files

Out of build target files, a build of only updated files is run (hereafter referred to as "build").

Running a build is performed for the entire project (main project and subprojects) or active project (see "2.15.4 Change the file build target project").

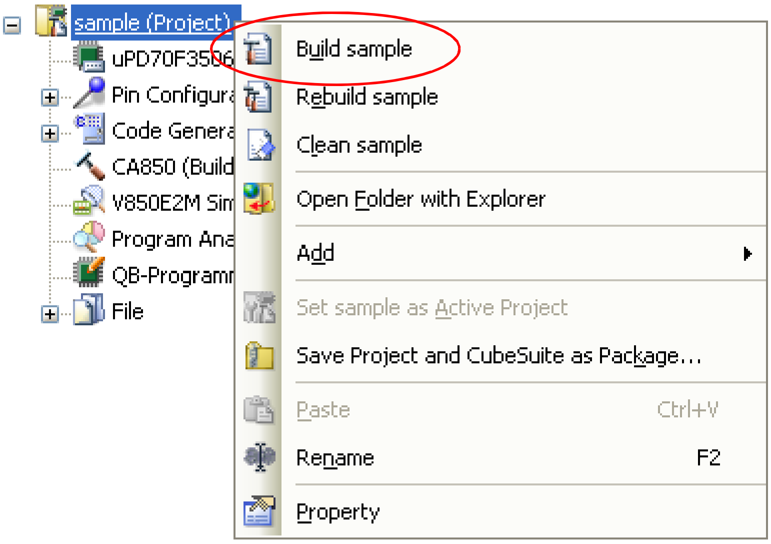
(1) When running a build of the entire project

Click on the toolbar.

(2) When running a build of the active project

Select the project, and then select [Build *active project*] from the context menu.

Figure 2-102. [Build *active project*] Item



Remark If the build of the included source file is not run after editing the header file and running the build, update the file dependencies (see "2.3.8 Update file dependencies").

2.16.2 Run a build of all files

A build of all build target files is run (hereafter referred to as "rebuild").

The cross reference file is deleted.

Running a rebuild is performed for the entire project (main project and subprojects) or active project (see "2.15.4 Change the file build target project").

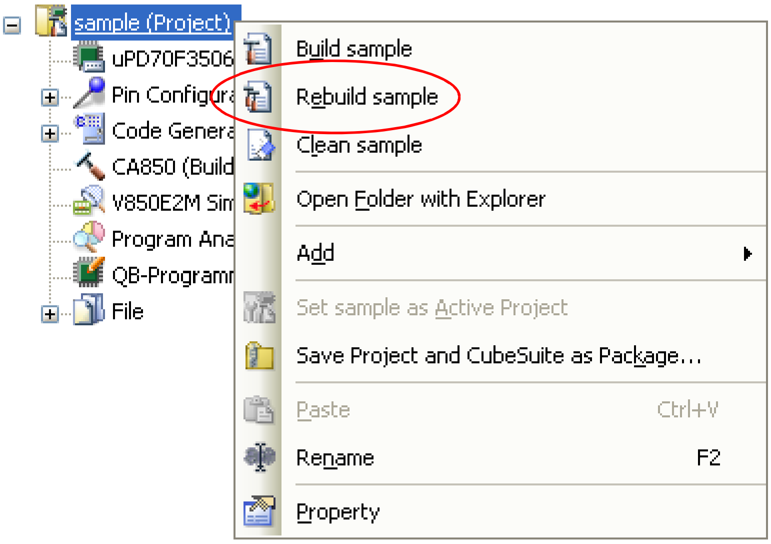
(1) When running a rebuild of the entire project

Click on the toolbar.

(2) When running a rebuild of the active project

Select the project, and then select [Rebuild *active project*] from the context menu.

Figure 2-103. [Rebuild *active project*] Item



2.16.3 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 (hereafter referred to as "rapid build").

- When the C source file, assembler source file, header file, link directive file, symbol information file, object module file, or library file that has been added to the project is 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 the properties of the build tool or build target files are 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.

Figure 2-104. [Rapid Build] Item (When Rapid Build Is Valid)

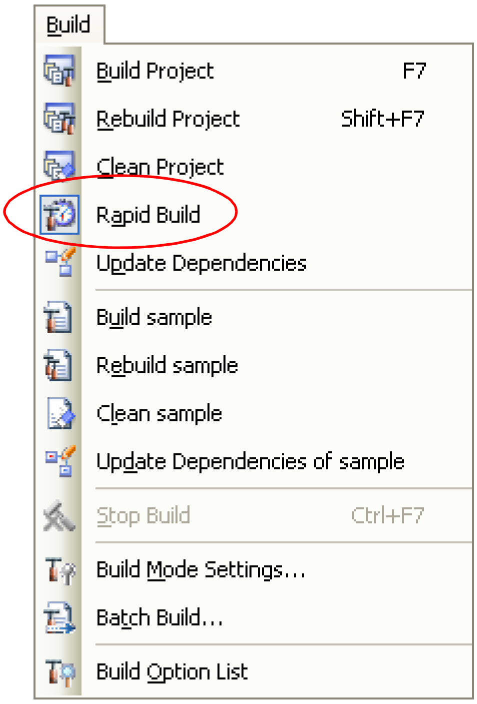
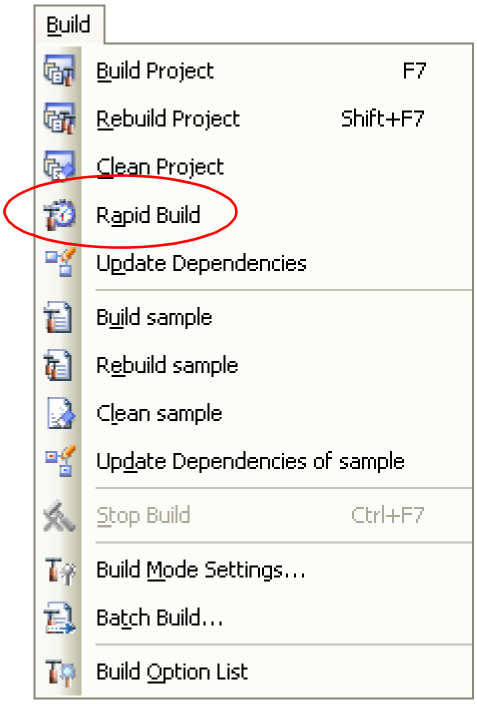


Figure 2-105. [Rapid Build] Item (When Rapid Build Is Invalid)



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.

Caution This function will also be enabled in an external editor if the [Observe registered files changing] checkbox is selected in the [Build/Debug] category of the Option dialog box.

2.16.4 Run builds in batch with build modes

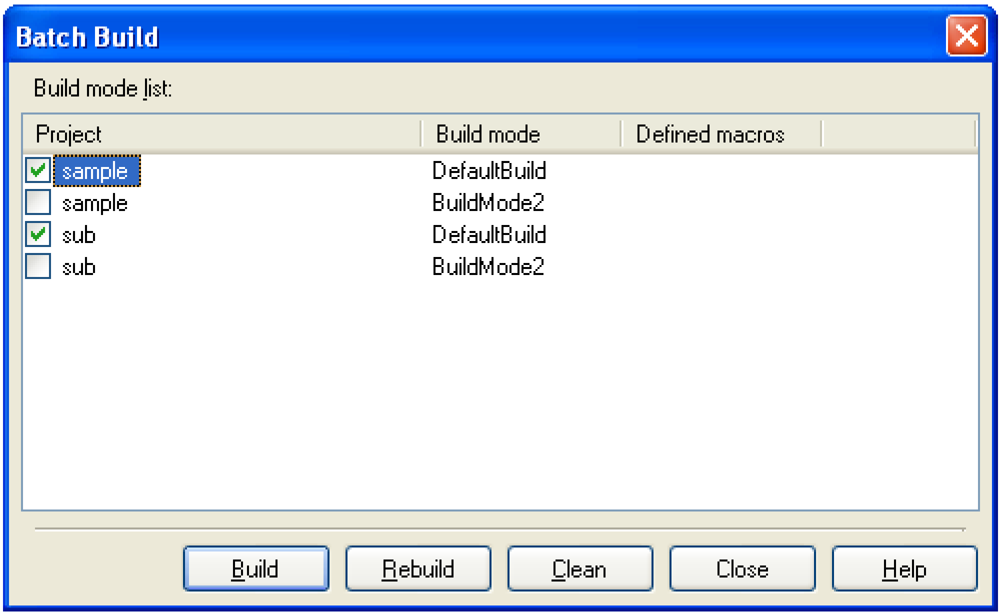
You can run builds, rebuilds and cleans in batch with the build modes that the project (main project and subproject) has (hereafter referred to as "batch build").

Remark See the sections below for a build, rebuild, and clean.

- Build: See "2.16.1 Run a build of updated files".
- Rebuild: See "2.16.2 Run a build of all files".
- Clean: See "2.16.9 Delete intermediate files and generated files".

Select [Batch Build] from the [Build] menu. The Batch Build dialog box will open.

Figure 2-106. Batch Build Dialog Box



Batch Build

Build mode list:

Project	Build mode	Defined macros
☑ sample	DefaultBuild	
□ sample	BuildMode2	
☑ sub	DefaultBuild	
□ sub	BuildMode2	

In the dialog box, the list of the combinations of the names of the main project and subprojects in the currently opened project and their build modes and macro definitions is displayed.

Select the check boxes for the combinations of the main project and subprojects and build modes that you wish to run a batch build, and then click the [Build], [Rebuild], or [Clean] button.

Remark The batch build order follows the project build order, the order of the subprojects, main project.
When multiple build modes are selected for a single main project or subproject, after running builds of the subproject with all the selected build modes, the build of the next subproject or main project is run.

2.16.5 Compile/assemble/link multiple files simultaneously

If there are multiple build target files, you can compile, assemble, and link all files simultaneously with a single cx command call (this features is called "simultaneous build").

An image of calling the cx command is shown below.

Example When build target files are aaa.c, bbb.c, and ccc.c

- When a build is run simultaneously

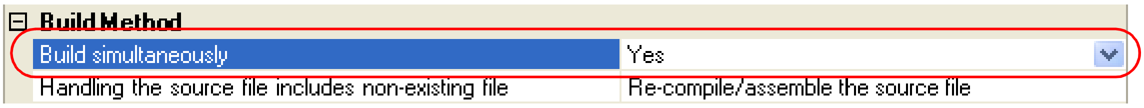
```
>cx -CF3746 aaa.c bbb.c ccc.c <- "a.lmf" is generated.
```

- When a build is not run simultaneously

```
>cx -CF3746 aaa.c <- "aaa.obj" is generated.
>cx -CF3746 bbb.c <- "bbb.obj" is generated.
>cx -CF3746 ccc.c <- "ccc.obj" is generated.
>cx -CF3746 aaa.obj bbb.obj ccc.obj <- "a.lmf" is generated.
```

Whether to run a build simultaneously is made with the property.
Select the build tool node on the project tree and select the [Common Options] tab on the Property panel.
Select [Yes] in the [Build simultaneously] property in the [Build Method] category.

Figure 2-107. [Build simultaneously] Property



⊟ Build Method

Build simultaneously	Yes
Handling the source file includes non-existing file	Re-compile/assemble the source file

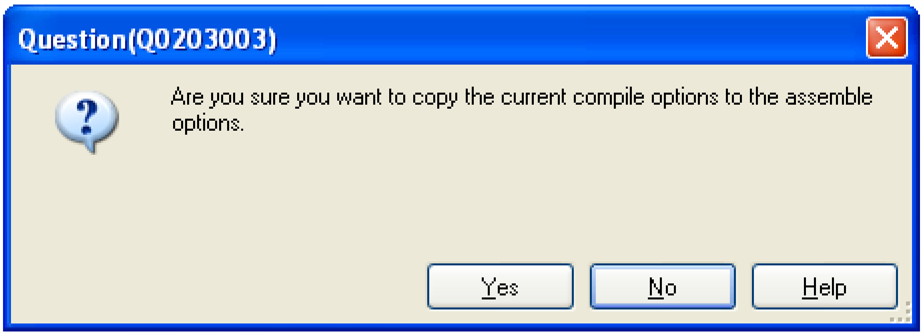
At this time, the [Assemble Options] tab is hidden.
The assembler source file (except the file with the individual build options) is assembled using the settings of the [Compile Options] tab.

Remarks 1. The build of the files with the individual build options and files to be executed prior to the build is run simultaneously.
A build of the file that is not targeted for a simultaneous build is run separately.

2. If the source file is older than the generated object module file or related properties and project or the like, the object module file will be used for the build instead of the source file.

3. If the [Build simultaneously] property is changed from [Yes] to [No], the following message dialog box will open.

Figure 2-108. Message Dialog Box



Question(Q0203003)

Are you sure you want to copy the current compile options to the assemble options.

Click [Yes] in the dialog box. The settings of the [Compile Options] tab will be copied to the [Assemble Options] tab.

If [No] is clicked, the state before the [Assemble Options] tab was hidden will be displayed.

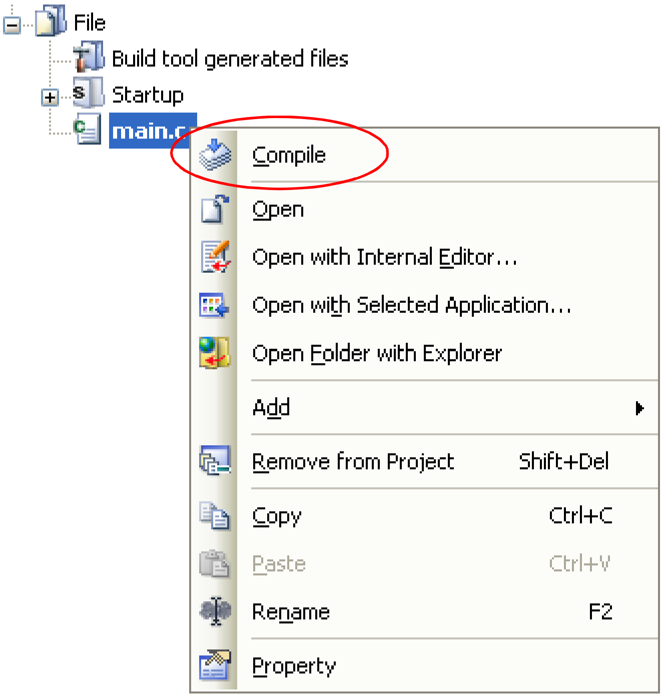
2.16.6 Compile/assemble individual files

You can just compile or assemble for each source file added to the project.

(1) When compiling a C source file

Select the C source file on the project tree and select the [Compile] from the context menu.

Figure 2-109. [Compile] Item

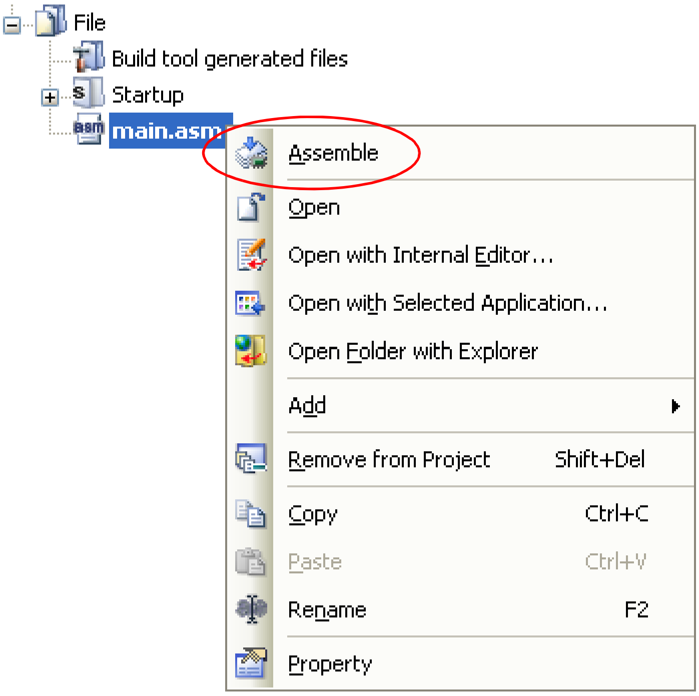


File
Build tool generated files
Startup
main.c
Compile
Open
Open with Internal Editor...
Open with Selected Application...
Open Folder with Explorer
Add
Remove from Project Shift+Del
Copy Ctrl+C
Paste Ctrl+V
Rename F2
Property

(2) **When assembling an assembler source file**

Select the assembler source file on the project tree and select the [Assemble] from the context menu.

Figure 2-110. [Assemble] Item


File
Build tool generated files
Startup
main.asm
Assemble
Open
Open with Internal Editor...
Open with Selected Application...
Open Folder with Explorer
Add
Remove from Project Shift+Del
Copy Ctrl+C
Paste Ctrl+V
Rename F2
Property

2.16.7 Stop running a build

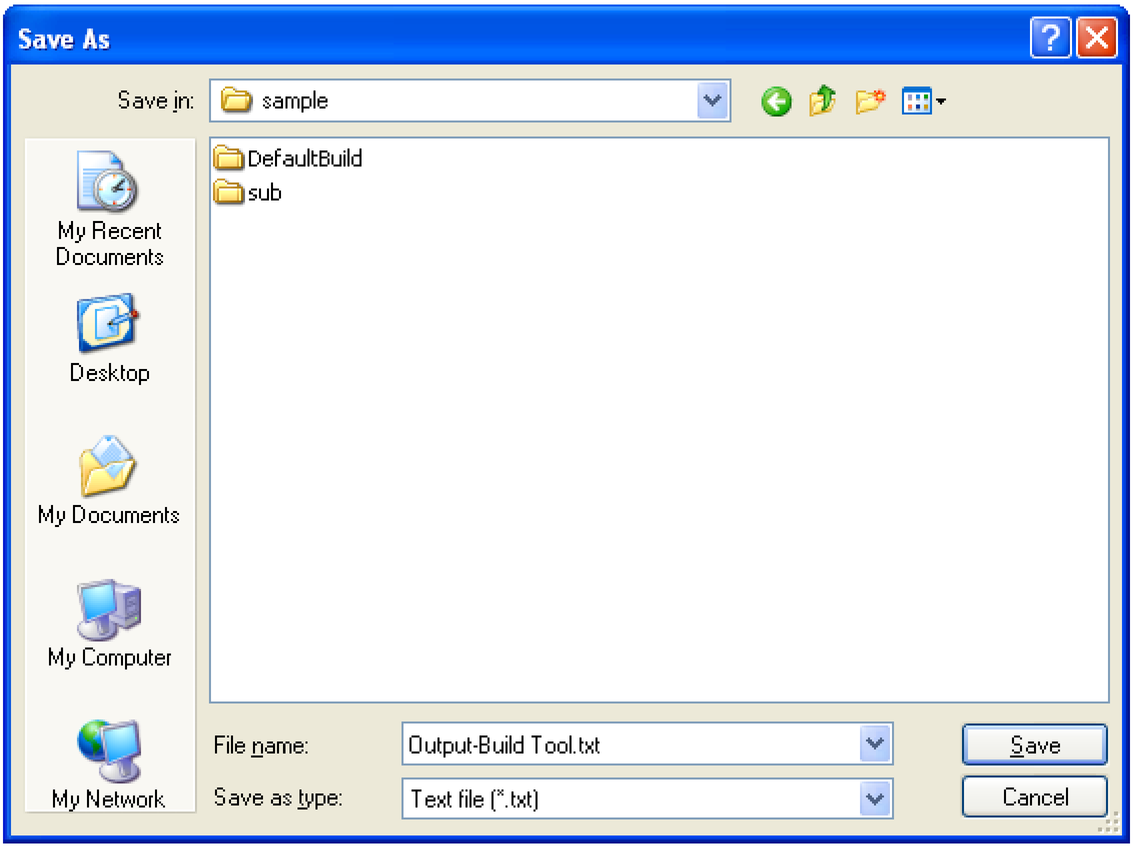
To stop running a build, rebuild, or batch build, click on the toolbar.

2.16.8 Save the build results to a file

You can save the execution results of the build (output messages of the build tool) that displayed on the Output panel.

Select the [Build Tool] tab on the panel, and then select [Save Output - Build Tool As...] from the [File] menu. The Save As dialog box will open.

Figure 2-111. Save As Dialog Box



Save in: sample

DefaultBuild
sub

File name: Output-Build Tool.txt

Save as type: Text file (*.txt)

In the dialog box, specify the file to be saved and then click the [Save] button.

2.16.9 Delete intermediate files and generated files

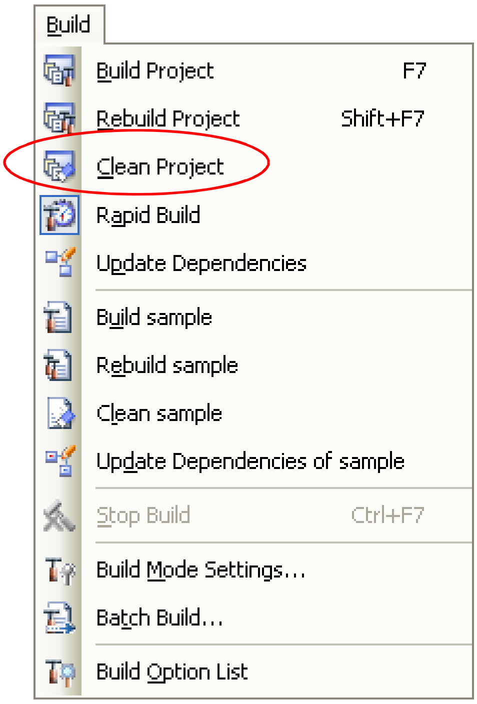
You can delete all the intermediate files and generated files output by running a build (hereafter referred to as "clean").

Running a clean is performed for the entire project (main project and subprojects) or active project (see "2.15.4 Change the file build target project").

(1) When running a clean of the entire project

From the [Build] menu, select [Clean Project].

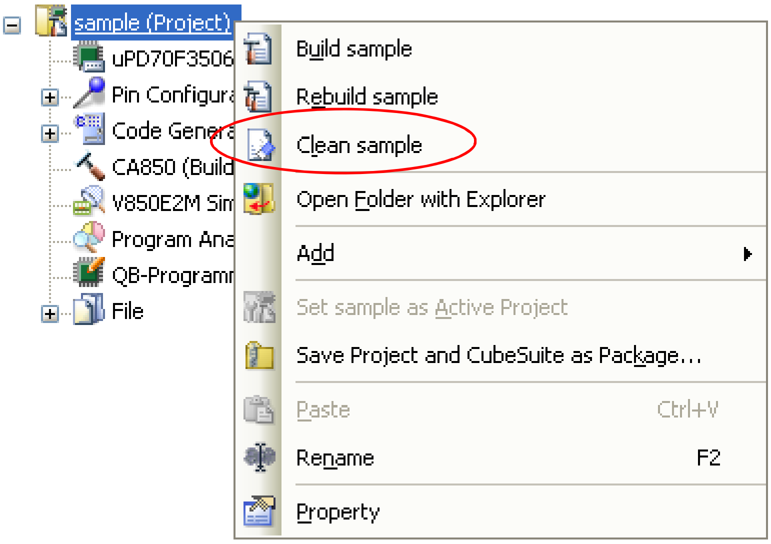
Figure 2-112. [Clean Project] Item



(2) When running a clean of the active project

Select the project, and then select [Clean *active project*] from the context menu.

Figure 2-113. [Clean *active project*] Item



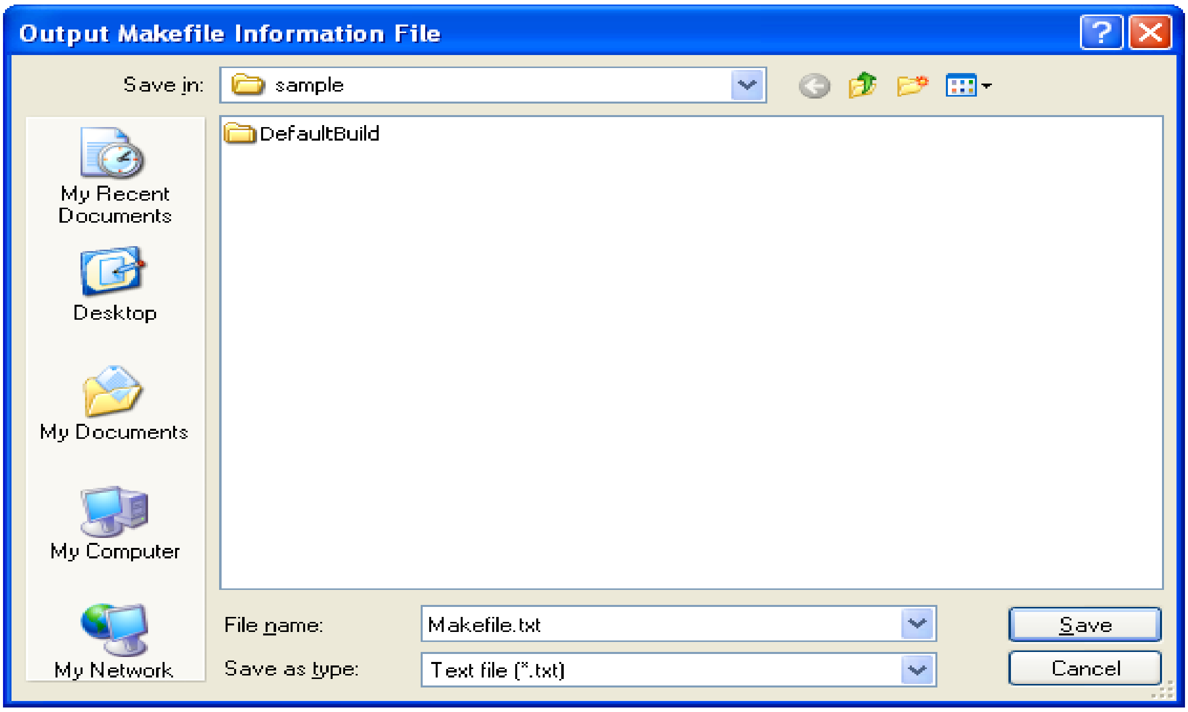
2.17 Output Information to Create a Makefile

You can obtain information for creating a Makefile from the current project, and output it to a file.

You can also use the output file as a Makefile by customizing it.

On the project tree, select the Build tool node, and then select [Output Makefile Information File...] from the context menu. The Output Makefile Information File dialog box opens.

Figure 2-114. Output Makefile Information File Dialog Box



File name: Makefile.txt

Save as type: Text file (*.txt)

In the dialog box, specify the file to be output and then click the [Save] button.

Remarks 1. The file is output into the project folder with name "Makefile.txt" by default.

2. The outpu file is not a target for clean.

The macros used in the output file are shown below.

Table 2-2. Type of Command Macros

Type of Macro	Command
CC = *path name*	C compiler
LB = *path name*	Librarian

Table 2-3. Type of Flag Macros

Type of Macro	Corresponding Command
CFLAGS = *option*	Compiler
ASFLAGS = *option*	Assembler
LDFLAGS = *option*	Linker

Type of Macro	Corresponding Command
LDFLAGS_CORE = *option*	Linker (only used when linking each core in cases where the project uses a device supporting multi-core)
ROMPFLAGS = *option*	ROMization processor
LBFLAGS = *option*	Librarian
DFFLAGS = *device specification option*	Device type

Remark The above macros cannot be used with files for which individual options are specified.

Table 2-4. Type of File Macros

Type of Macro	Description
OBJS = *Relative paths from the project of object files generated during compilation/assembly; it is passed during link/library processing*	Passed to the dependency file of link/library processing command, and command
OBJS_*core number* = *Relative paths from the project of object files generated during compilation/assembly; it is passed during link/library processing*	Passed to the dependency file of link/library processing command, and command (only used when linking each core in cases where the project uses a device supporting multi-core) (*core number*: "CMN" for processing of common core; "PE*core number*" for processing of each core)
LIBS = *Relative paths from the project of library files registered to the project; it is passed during link/library processing*	Passed to link and library processing commands (e.g. -l option)
STDLIBS = *Relative paths from the project of standard library files; it is passed during link/library processing*	Passed to link and library processing commands (e.g. -l option) (Used in cases that cannot be expressed by LIBS only via the command-line format)

Remark If simultaneous build is enabled, then the OBJS and OBJS_*core number* macros will also include the sources that are the targets of simultaneous compilation/assembly.

2.18 Using Stack Usage Tracer

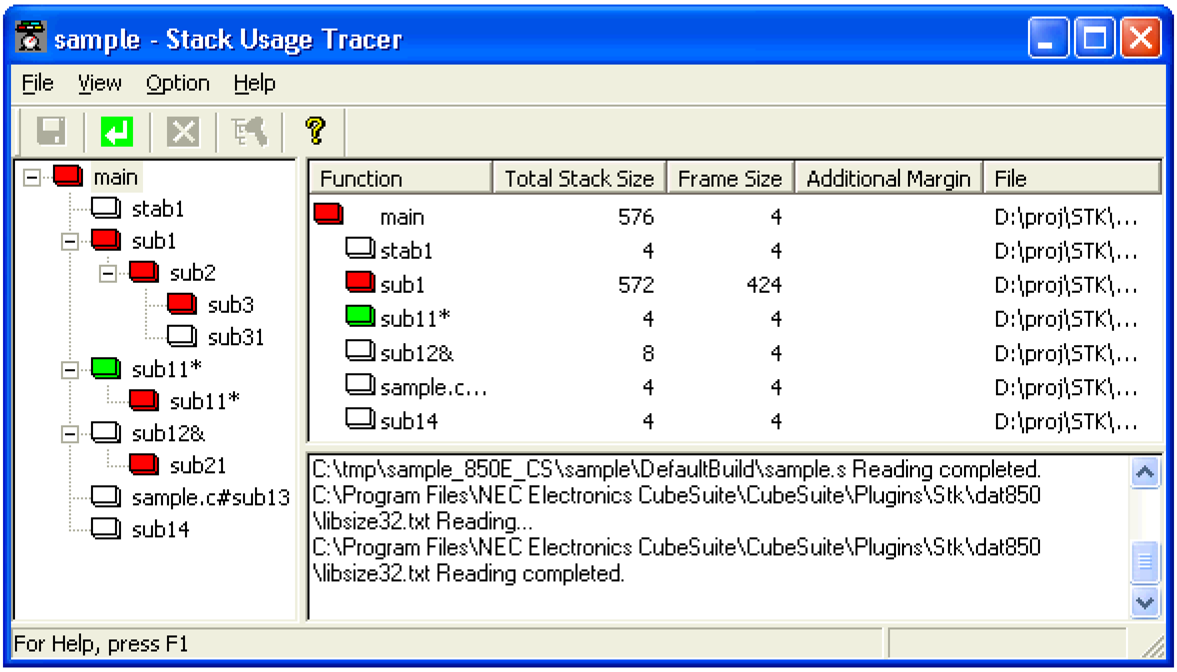
The stack usage tracer performs a static analysis, and displays the functions called by a function in a tree format, as well as stack information for each function (function name, total stack size, frame size, additional margin, and file name) in list format.

2.18.1 Starting and exiting

To start the stack usage tracer, from the Main window, select the [Tool] menu >> [Startup Stack Usage Tracer].

After the stack usage tracer finishes starting up, it will display the function call relationship and stack information for each function in the tree display area/list display area of the Stack Usage Tracer window.

Figure 2-115. Starting Up Stack Usage Tracer



Function	Total Stack Size	Frame Size	Additional Margin	File
🟥 main	576	4		D:\proj\STK\...
□ stab1	4	4		D:\proj\STK\...
🟥 sub1	572	424		D:\proj\STK\...
🟩 sub11*	4	4		D:\proj\STK\...
□ sub12&	8	4		D:\proj\STK\...
□ sample.c...	4	4		D:\proj\STK\...
□ sub14	4	4		D:\proj\STK\...

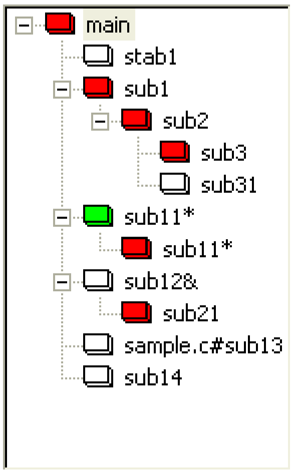
```
C:\tmp\sample_850E_CS\sample\DefaultBuild\sample.s Reading completed.
C:\Program Files\NEC Electronics CubeSuite\CubeSuite\Plugins\Stk\dat850
\libsize32.txt Reading...
C:\Program Files\NEC Electronics CubeSuite\CubeSuite\Plugins\Stk\dat850
\libsize32.txt Reading completed.
```

To exit the stack usage tracer, from the Stack Usage Tracer window, select [File] menu >> [Exit skcx].

2.18.2 Check the call relationship

You can check the function-call relationship in the tree display area of the Stack Usage Tracer window.

Figure 2-116. Tree Display Area



```
⊟ 🟥 main
     □ stab1
  ⊟ 🟥 sub1
     ⊟ 🟥 sub2
          🟥 sub3
          □ sub31
  ⊟ 🟩 sub11*
       🟥 sub11*
  ⊟ □ sub12&
       🟥 sub21
     □ sample.c#sub13
     □ sub14
```

Remark The table below shows the meaning of the icon displayed to the left of the string representing the function name.

The display priority for icons is from High: 🟥 to Low: □ .

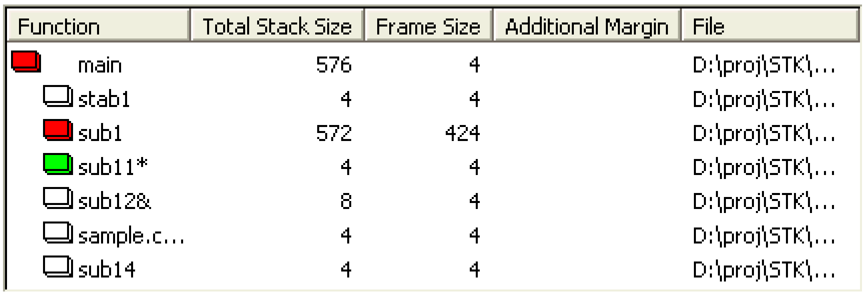
🟥	The function directly called by a given function with the largest total stack size
🟦	Information (additional margin, recursion depth, or callee functions) has been modified via the Adjust Stack Size dialog box or a stack size specification file
🟩	Recursive function
🟨	The stack usage tracer has not acquired any stack information for this function
□	Other than the above

2.18.3 Check the stack information

You can check the stack information (function name, total stack size, frame size, additional margin, and file name) from the list display area of the Stack Usage Tracer window.

- Total stack size (including stack size of callee functions)
- Frame size (not including stack size of callee functions)
- Additional margin (value mandatorily added to frame size)

Figure 2-117. List Display Area



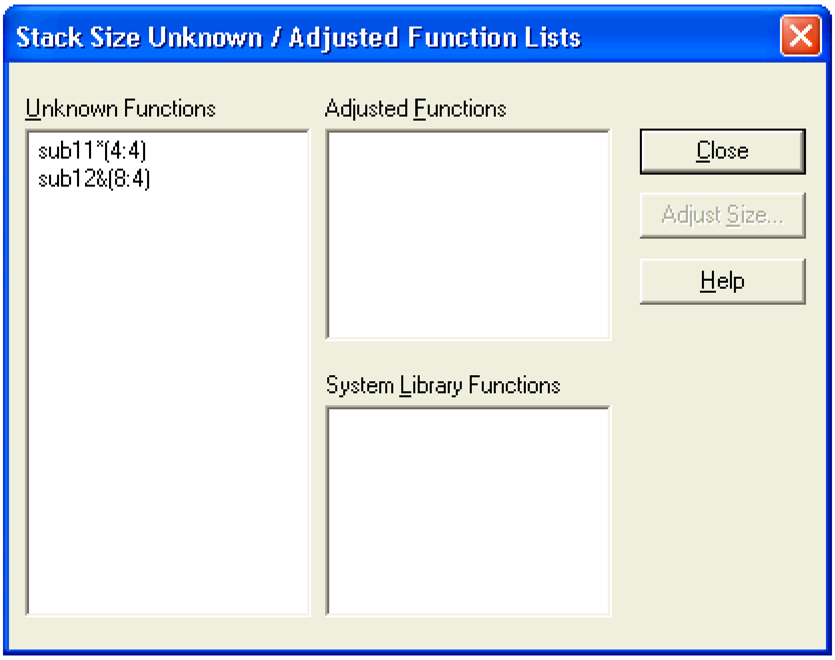
Function	Total Stack Size	Frame Size	Additional Margin	File
🟥 main	576	4		D:\proj\STK\...
□ stab1	4	4		D:\proj\STK\...
🟥 sub1	572	424		D:\proj\STK\...
🟩 sub11*	4	4		D:\proj\STK\...
□ sub12&	8	4		D:\proj\STK\...
□ sample.c...	4	4		D:\proj\STK\...
□ sub14	4	4		D:\proj\STK\...

Remark If you make changes to the project that will affect the total stack size while the stack usage tracer is running (e.g. you edit the files in your project so that the total stack size changes), then after rebuilding the project, click ↵ to update the display.

2.18.4 Check unknown functions

You can check functions for which the stack usage tracer could not obtain stack information in the Stack Size Unknown / Adjusted Function Lists dialog box, under [Unknown Functions].

Figure 2-118. Stack Size Unknown / Adjusted Function Lists Dialog Box



sub11*(4:4)
sub12&(8:4)

Remark Functions will appear under [Unknown Functions] in the following circumstances.

- The frame size could not be measured.
- A recursive function for which the recursion depth has not been set in the Adjust Stack Size dialog box.
- The function includes indirect function calls which are not set as callee functions in the Adjust Stack Size dialog box.

2.18.5 Change the frame size

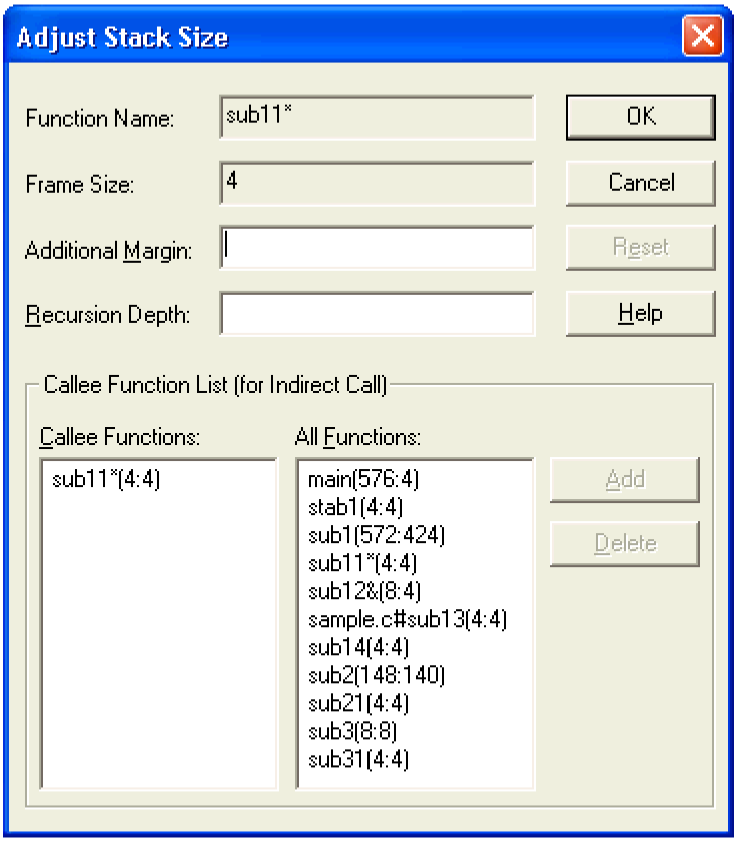
You can dynamically change the frame size of functions for which the stack usage tracer was not able to obtain stack information, or for functions that you intentionally want to modify, using the Adjust Stack Size dialog box or a stack size specification file.

(1) Using the Adjust Stack Size dialog box

The procedure for using the Adjust Stack Size dialog box is as follows.

- Select the desired item in the tree display area of the Stack Usage Tracer window, then click toolbar >> .
  The Adjust Stack Size dialog box opens.

Figure 2-119. Adjust Stack Size Dialog Box



Function Name: sub11*

Frame Size: 4

Additional Margin:

Recursion Depth:

Callee Functions:

sub11*(4:4)

All Functions:

main(576:4)
stab1(4:4)
sub1(572:424)
sub11*(4:4)
sub12&(8:4)
sample.c#sub13(4:4)
sub14(4:4)
sub2(148:140)
sub21(4:4)
sub3(8:8)
sub31(4:4)

- After setting [Additional Margin], [Recursion Depth], and [Callee Functions], click the [OK] button.

(2) Using a stack size specification file

Below is the procedure for using a stack size specification file.

- Create a stack size specification file

  Write the functions in the stack size specification file that you would like to set dynamically, using the following format.

  function name [, ADD=additional margin] [, RECTIME=recursion depth] [, CALL=callee function] ...

```
# Set the frame size of function "_flib" written in assembly
# language to 50
[flib], ADD=50

# Set the frame size of function "sub2" written in C to 100
sub2, ADD=100

#Set the recursion depth of recursive function "sub3" written
# in C to 123
sub3, RECTIME=123
```

- From the Stack Usage Tracer window, select [File] menu >> [Load Stack Size Specification File...]. The Open dialog box opens. Specify the stack size specification file, then click [Open].

CHAPTER 3 BUILD OUTPUT LISTS

This chapter explains the format and other aspects of files output by a build via CX.

3.1 Assemble List File

This section explains the assemble list file.

The assemble list is the list-formatted version of the code that is output when the source has been compiled and assembled.

It can be used to check the code resulting from compilation and assembly.

(1) How to output

The methods for outputting the assemble list file are shown below.

(a) CubeSuite

Select the build tool node on the Project Tree panel and select the [Compile Options] tab on the Property panel.

To output the assemble list file, select [Yes(-Xprn_path)] in the [Output assemble list file] property in the [Assemble List] category.

Specify the output destination in the [Output folder for assemble list file] property.

The file name will be the source file name with the extension replaced by ".prn".

When [Yes(-Xpass_source)] in the [Output comment to assembler source file] property in the [Output Code] category is selected, the C source program that corresponds to the assembler source program is output as comments in the assemble list file.

(b) Command line

When the -Xprn_path option is specified, the assemble list file is output under the source file name with the extension replaced by ".prn".

Also, the file name can be specified by the parameter of the -Xprn_path option.

If the -Xpass_source option is specified at the same time, the C source program that corresponds to the assembler source program is output as comments in the assemble list file.

(2) Output example

When the C source file shown below is compiled and the output assembler source file is assembled, the assemble list file shown below is output.

- C source file

```
void main(void) {
    int     a;
}
```

- Assemble list file

```
(1)     (2)          (3)       (4)               (5)
        :
A-X- 00000000                 16           .file    “a.c“
A-X- 00000000                 17           .align   4
A-X- 00000000                 18           #@BF
A-X- 00000000                 19           .func    _main, .F2-.F2.end, 20
A-X- 00000000                 20           .public _main
A-X- 00000000                 21   _main:
A-X- 00000000 0C8A            22           jbr      .L4
A-X- 00000002                 23   .L5:
A-X- 00000002 4001            24           mov      r0, r10
A-X- 00000004 E3CF0000        25           ld.w     -4+.F2[sp], lp
A-X- 00000008 6444            26           add      .F2, sp
A-X- 0000000A 1F18            27           jmp      [lp]     --1
A-X- 0000000C                 28   .L4:
A-X- 0000000C                 29           sub      .F2, sp
A-X- 0000000C 2440                 -- mov  0x4, r1
A-X- 0000000E 6108                 -- sub  r1, sp
A-X- 00000010 E3DF0000        30           st.w     lp, -4+.F2[sp]
A-X- 00000014 EE8B            31           jbr      .L5
A-X- 00000016                 32           #@FUNC_ARG
A-X- 00000016                 33           .F2      .set   0x4
A-X- 00000016                 34           .A2      .set   0x0
A-X- 00000016                 35           .T2      .set   0x0
A-X- 00000016                 36           #@EF
        :
```

Item Number	Description
(1)	Section attribute This is the section attribute of the section storing the code generated for the source program of the corresponding line. The section attributes and their meanings are as follows. A: Section occupying the memory W: Section that can be written X: Section that can be executed G: Section allocated to the memory area that can be referenced by using global pointer (gp) and 16-bit displacement
(2)	Location counter value This is the location counter value for the beginning of the code generated for the source program of the corresponding line.
(3)	Code This is the code (machine language instruction or data) generated for the source program of the corresponding line. Each byte is expressed as 2-digit hexadecimal number.

Item Number	Description
(4)	Line number This is the number of the line. This is expressed in a decimal number.
(5)	Source program This is the source program of the line. If instruction expansion is performed for the instruction at that line, the disassembly of the array of machine language instructions generated after the instruction expansion is displayed after "--".

3.2 Link Map File

This section explains the link map file.

The link map has information of the link result. It can be referenced for information such as the section's allocation addresses.

(1) How to output

The methods for outputting the link map file are shown below.

(a) CubeSuite

Select the build tool node on the Project Tree panel and select the [Link Options] tab on the Property panel.
To output the link map file, select [Yes(-Xmap)] in the [Output link map file] property in the [Link Map] category.
Specify the output destination in the [Output folder for link map file] property and [Link map file name] property.
It is also shown on the project tree, under the Build tool generated files node.

(b) Command line

When the -Xmap option is specified, the link map file is output to the same folder as the load module file under the load module file name with the extension replaced by ".map".
Also, the file name can be specified by the parameter of the -Xmap option.

The link map file to be output differs depending on whether the -Xno_romize option is specified as follows.

- When the -Xno_romize option is specified
  The link map file is output after link processing.
- When the -Xno_romize option is not specified
  The link map file is output after ROMization processing.

(2) Output example

When the -Xno_romize option is specified and the following objects are linked, the link map file shown below is output.

- test.obj (object module file specified by the user)
- cstartN.obj (startup routine)
- libc.lib (standard library)

```
            ********** MEMORY ALLOCATION MAP **********
  (1)          (2)                (3)          (4)          (5)
OUTPUT      SEGMENT           VIRTUAL     SIZE(16)     SIZE(10)
SEGMENT     ATTRIBUTE         ADDRESS

INT         RX                0x00000000  0x0000007a        122
TEXT        RX                0x00000080  0x00000262        610
DATA        RW                0x0fffc000  0x00000008          8

            ********** LINK EDITOR ALLOCATION MAP **********
  (6)          (7)                (8)          (9)          (10)
OUTPUT      INPUT             VIRTUAL     SIZE         INPUT
SECTION     SECTION           ADDRESS                  FILE

RESET                         0x00000000  0x00000004
            RESET             0x00000000  0x00000004   D:\lib\cstartN.obj

SECURITY_ID                   0x00000070  0x0000000a
            SECURITY_ID       0x00000070  0x0000000a   *(nil)*

.pro_epi_runtime              0x00000080  0x000001e0
            .pro_epi_runtime  0x00000080  0x000001e0   callt.obj(D:\lib\libc.lib)

.text                         0x00000260  0x00000082
            .text             0x00000260  0x00000076   D:\lib\cstartN.obj
            .text             0x000002d6  0x0000000a   D:\debug\test.obj
            .text             0x000002e0  0x00000002   hdwinit.obj(D:\lib\libc.lib)

.data                         0x0fffc000  0x00000008
            .data             0x0fffc000  0x00000008   D:\lib\cstartN.obj

.sbss                         0x0fffc008  0x00000000
            .sbss             0x0fffc008  0x00000000   D:\lib\cstartN.obj

.bss                          0x0fffc008  0x00000000
            .bss              0x0fffc008  0x00000000   D:\lib\cstartN.obj
```

Item Number	Description
(1)	Output segment This is the name of the output segment configuring the load module file to be generated. The name of the output segment is not stored the load module file to be generated.

Item Number	Description
(2)	Segment attribute The segment attributes and their meanings are as follows. R: Readable W: Writable X: Executable
(3)	Address This is the start address of the output segment.
(4)	Size (hexadecimal number) This is the size of the memory including the alignment condition between the sections and the align hole.
(5)	Size (decimal number) This is the size of the memory including the alignment condition between the sections and the align hole.
(6)	Output section This is the name of the output section configuring the load module file to be generated.
(7)	Input section This is the name of the input section allocated to each output section.
(8)	Address This is the start address of the output section and input section.
(9)	Size This is the start address of the output section and input section.
(10)	Input file This is the name of the object module file that the input section belongs to. "*(nil)*" is displayed for sections generated by linking, and for .symtab, .strtab, .shstrtab, and the like generated by assembly. If the identifier indicating the object is tentatively defined by external linkage (for assembly language, an area that is secured by a .comm quasi directive), this will be an area common to all files, and "*(Common)*" or "*(GpCommon)*" will be displayed for that section. If the object module file to which the input section belongs is the object module file in the library file, the library file name is displayed in format "(object module file name (library file name)".

3.3 Symbol Information File

This section explains the symbol information file.

The symbol information file is a text-format file that allocation information for a variable (global variable, static variable in a file, and static variable in a function) and function defined in a C source file is described.

(1) How to output

The methods for outputting the symbol information file are shown below.

(a) CubeSuite

Select the build tool node on the Project Tree panel and select the [Link Options] tab on the Property panel.

To output the symbol information file, select [Yes(-Xsfg)] in the [Output symbol information file] property in the [Symbol Information] category.

Specify the output destination in the [Output folder for symbol information file] property and [Symbol information file name] property.

It is also shown on the project tree, under the Files node.

(b) Command line

When the -Xsfg option is specified, the symbol information file is output to the same folder as the load module file under the load module file name with the extension replaced by ".sfg".

Also, the file name can be specified by the parameter of the -Xsfg option.

(2) Output example

When the C source file is compiled, the symbol information file shown below is output.

```
(1)## Section and Variable Information ##
(2)// Global-Variable             : Name, ReferenceCount, ByteSize
(2)// Static-Variable in file     : Name, ReferenceCount, ByteSize, "FileName"
(2)// Static-Variable in function : Name, ReferenceCount, ByteSize, "FileName", FunctionName

(3)[tidata.byte]:Size_128bytes
(11)func_static_UC,(12)5,(13)1,(14)"D:\develop\tp\xx_2.c",(15)sub2
(11)func_static_UC,(12)4,(13)1,(14)"..\tp\xx_1.c",(15)main
(4)ti_data_byte,(5)1,(6)1
(4)ti_bss_byte,(5)1,(6)1
(11)@func_static_unused_UC,(12)0,(13)1,(14)"D:\develop\tp\xx_2.c",(15)sub2

(3)[tidata.word]:Size_128bytes
(4)global_int_module_xx1_symbol,(5)6,(6)4
(4)global_const_int_module_xx1_symbol,(5)5,(6)4
(11)func_static_UL,(12)5,(13)4,(14)"D:\develop\tp\xx_2.c",(15)sub2
(7)file_static_int_module_xx2,(8)5,(9)4,(10)"D:\develop\tp\xx_2.c"
(11)func_static_UL,(12)4,(13)4,(14)"..\tp\xx_1.c",(15)main
(7)file_static_int_module_xx1,(8)4,(9)4,(10)"..\tp\xx_1.c"
(4)global_const_int_module_xx2_symbol,(5)4,(6)4
(7)static_module_xx1_symbol,(8)1,(9)2,(10)"..\tp\xx_1.c"
(4)si_data,(5)1,(6)4
(4)si_bss,(5)1,(6)4
```

```
(4)se_data,(5)1,(6)4
(4)se_bss,(5)1,(6)4
(4)struct_xx2_sub2,(5)1,(6)24
(11)@func_static_unused_UC,(12)0,(13)4,(14)"D:\develop\tp\xx_2.c",(15)sub2
(4)@struct_xx2_unused,(5)0,(6)24
(4)@global_const_int_module_xx1_unusedsymbol,(5)0,(6)4

(3)[sidata]:Size_32512bytes

(3)[sedata]:Size_32768bytes

(3)[sdata]:Size_65536bytes


(16)## Function Information ##
(2)// FunctionName, StartAddress, FunctionByteSize, ReferenceCount, StackByteSize, "FileName"

(17)sub1,(18)0x00000210,(19)24,(20)2,(21)0,(22)"..\tp\xx_1.c"
(17)sub2,(18)0x00000080,(19)174,(20)2,(21)0,(22)"D:\tp\xx_2.c"
(17)@unusedfunc2,(18)0x0000012E,(19)12,(20)0,(21)0,(22)"D:\tp\xx_2.c"
(17)@main,(18)0x0000013A,(19)214,(20)0,(21)0,(22)"..\tp\xx_1.c"
(17)@unusedfunc1,(18)0x00000228,(19)12,(20)0,(21)0,(22)"..\tp\xx_1.c"
```

Item Number	Description
(1)	Start of section allocation information of variable The compiler refers the section allocation information of variables described on the following line and beyond.
(2)	Comment
(3)	Section name and section size The name and size of the variable allocation section is output in the following format: "[*section-name*]:Size_*decimal-byte-size*bytes" This is output only when the -Xsfg_opt option is specified.
(4)	Variable name This is the name of the global variable. If the number of references is 0, "@" is prepended, indicating that it is the unused variable.
(5)	Number of references This is the number of references of the global variable.
(6)	Size This is the byte size of the global variable.
(7)	Variable name This is the name of the static variable in the file. If the number of references is 0, "@" is prepended, indicating that it is the unused variable.
(8)	Number of references This is the number of references of the static variable in the file.
(9)	Size This is the byte size of the static variable in the file.

Item Number	Description
(10)	File name The name of the file in which the static variable is defined is output with its path.
(11)	Variable name This is the name of the static variable in the function. If the number of references is 0, "@" is prepended, indicating that it is the unused variable.
(12)	Number of references This is the number of references of the static variable in the function.
(13)	Size This is the byte size of the static variable in the function.
(14)	File name The name of the file in which the static variable in the function is defined is output with its path.
(15)	Function name This is the name of the function that the static variable is defined.
(16)	Start of function information
(17)	Function name This is the name of the function. If the number of references is 0, "@" is prepended, indicating that it is the unused function.
(18)	Start address This is the start address that the function is defined.
(19)	Function size This is the byte size of the function.
(20)	Number of references This is the number of references of the function.
(21)	Stack size This is the byte size of the amount of stack used by the function.
(22)	File name The name of the file in which the function is defined is output with its path.

3.4 Hex File

This section explains the hex file.

The hex file is the file to which a load module file is converted in hex format.

The following hex formats can be specified.

- Intel expanded hex format
- Motorola S type hex format
- Expanded Textronix hex format

(1) How to output

With CX, the hex file is output by default.

The method for setting the hex format is shown below.

(a) CubeSuite

Select the build tool node on the Project Tree panel and select the [Hex Output Options] tab on the Property panel.

Select the hex format in the [Hex file format] property in the [Hex Format] category.

It is also shown on the project tree, under the Build tool generated files node.

(b) Command line

Specify the format by the parameter of the -Xhex_format option.

The specification format is shown below.

```
-Xhex_format=format
```

The items that can be specified as *format* are shown below.

I	Intel expanded hex format (up to 1 MB)
i	Intel expanded hex format (32-bit address) (up to 4 GB)
S	Motorola S type hex format (standard address) (up to 16 MB)
s	Motorola S type hex format (32-bit address) (up to 4 GB)
T	Expanded Tektronix hex format (up to 4 GB)

3.4.1 Intel expanded hex format

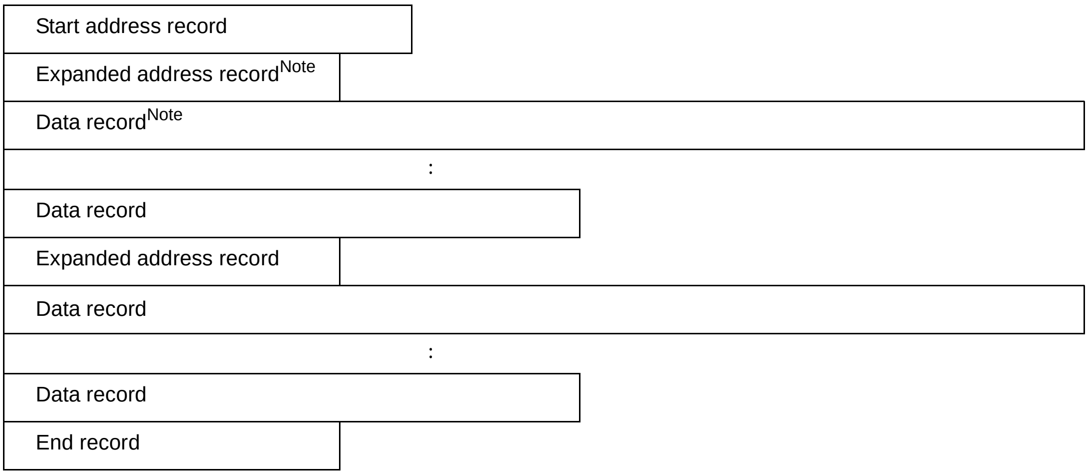
The Intel expanded hex format file (20 bits) consists of four records[Note]: start address record, expanded address record, data record, and end record.

The Intel expanded hex format file (32 bits) consists of six records[Note]: start linear address record, expanded linear start address record, start address record, expanded address record, data record, and end record.

Note Each record is output in ASCII code.

The file configuration in Intel expanded hex format is shown below.

Figure 3-1. File Configuration in Intel Expanded Hex Format



Note The expanded address record and data record are repeated.

Each record consists of the following fields.

```
:   XX   XXXX   XX   DD......DD  SS   NL
(1) (2)  (3)    (4)      (5)     (6)  (7)
```

Item Number	Description
(1)	Record mark
(2)	Number of bytes This is the number of bytes expressed as 2-digit hexadecimal number of (5).
(3)	Location address
(4)	Record type 05: Start linear address record 04: Expanded linear address record 03: Start address record 02: Expanded address record 00: Data record 01: End record
(5)	Code This is each byte of code expressed as 2-digit hexadecimal number.

Item Number	Description
(6)	Checksum This is the value expressed as a 2-digit hexadecimal number in the record (other than :, SS, and NL) sequentially subtracted from initial value 0 and that lower 1 byte expressed as a 2-digit hexadecimal number.
(7)	Newline (\n)

- Start linear address record (32-bit address)

  This indicates the linear address.

```
:    04   0000   05   XXXXXXXX   SS   NL
(1)  (2)  (3)    (4)     (5)
```

Item Number	Description
(1)	Record mark
(2)	Fixed at 04
(3)	Fixed at 0000
(4)	Fixed at 05
(5)	Linear address value

- Expanded linear address record (32-bit address)

  This indicates the upper 16-bit address at bits 32 to 16.

```
:    02   0000   04   0000   SS   NL
(1)  (2)  (3)    (4)   (5)
```

Item Number	Description
(1)	Record mark
(2)	Fixed at 02
(3)	Fixed at 0000
(4)	Fixed at 04
(5)	Upper 16-bit address at bits 32 to 16

Note The lower 16 bits are used as the location address of the data record.

- Start address record

  This indicates the entry point address.

```
:    04   0000   03   PPPP   OOOO   SS   NL
(1)  (2)  (3)    (4)   (5)    (6)
```

Item Number	Description
(1)	Record mark
(2)	Fixed at 04
(3)	Fixed at 0000
(4)	Fixed at 03
(5)	Paragraph value of entry point address[Note]
(6)	Offset value of entry point address

Note The address is calculated by (paragraph value << 4) + offset value.

- Expanded address record

This indicates the paragraph value of the entry address[Note].

Note This is output if the segment is renewed at the beginning of the segment (when the data record is output) or when the offset value of the data record's load address exceeds the maximum value of 0xffff.

```
:   02  0000  02  PPPP  SS  NL
(1) (2) (3)    (4) (5)
```

Item Number	Description
(1)	Record mark
(2)	Fixed at 02
(3)	Fixed at 0000
(4)	Fixed at 02
(5)	Paragraph value of segment

- Data record

This indicates the value of the code.

```
:   XX  XXXX  00  DD......DD  SS  NL
(1) (2) (3)    (4)      (5)
```

Item Number	Description
(1)	Record mark
(2)	Number of bytes[Note]
(3)	Location address
(4)	Fixed at 00
(5)	Code This is each byte of code expressed as 2-digit hexadecimal number.

Note This is limited to the range of 0x1 to 0xff (the minimum value for the number of bytes of the code indicated by one data record is 1 and the maximum value is 255).

Example

```
:   04  0100  00  3C58E01B  6C  NL
(1) (2) (3)   (4)    (5)    (6) (7)
```

Item Number	Description
(1)	Record mark
(2)	Number of bytes of 3C58E01B expressed as 2-digit hexadecimal number
(3)	Location address
(4)	Record type 00
(5)	Each byte of code expressed as 2-digit hexadecimal number
(6)	Checksum The lower 1 byte of two's complement E6C of 04 + 01 + 00 + 00 + 3C + 58 + E0 + 1B = 194 is expressed as a 2-digit hexadecimal number.
(7)	Newline (\n)

- End record

This indicates the end of the code.

```
:   00  0000  01  FF  NL
(1) (2) (3)   (4) (5)
```

Item Number	Description
(1)	Record mark
(2)	Fixed at 00
(3)	Fixed at 0000
(4)	Fixed at 01
(5)	Fixed at FF

Remark The allocation address in the Intel hex format is 2 bytes (16 bits).
Therefore, only a 64 KB space can be directly specified.
To extend this area, the Intel extended hex format adds the extension address of 16 bits so that the space up to 1 M byte (20 bits) can be used.
Specifically, the record type that specifies the 16-bit extension address is added.
This extension address is shifted 4 bits and added to the allocation address to express a 20-bit address.
To indicate FFFFFH, for example, F000H is set as the extension address, and FFFFH is specified as the location address.
In the Intel extended hex format, only 0 to FFFFFH can be addressed.
To express 100000H, another object format must be used.

With CX, a message will be output if the rule of this format is violated with this address and size.
In the Intel extended hex format, a value that can be expressed is 20 bits, or 1 M byte (0x100000).

```
W0562022:The start address of convert area exceeds the maximum value of the
address that can be expressed in the Intel expanded hex format.
```

If the message "W0562022" is output, the start address of the area to be converted into the hex format exceeds 1 M byte.

```
W0562020:The address of convert area exceeds the maximum value of the address that can
be expressed in the Intel expanded hex format.
```

If the message "W0562020" is output, the address to be converted into the hex format exceeds 1 M byte (20 bits).
The above error will be occur in the following cases even if 1 M byte is not exceeded.

Examples 1. The offset that starts from the address specified by the -Xhex_offset option is not used
-> The absolute address is stored in the hex format.

2. The section is allocated in the vicinity of the upper limit of the address that can be expressed by 20 bits
-> The start address fits in 20 bits, but 20 bits are exceeded in the middle of the section.

If these two patterns are satisfied, the message "W0562020" is output even if the area to be converted is as small as 4 bytes.

3.4.2 Motorola S type hex format

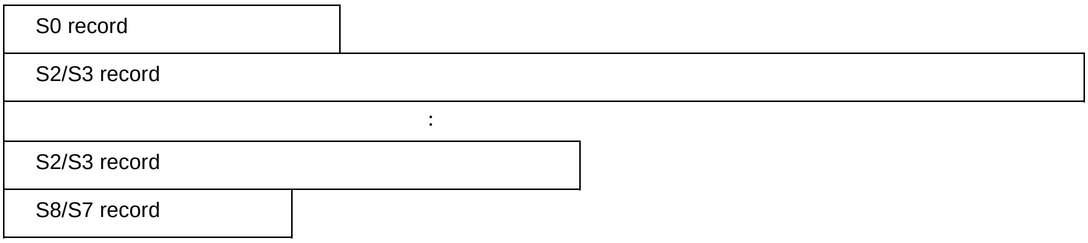
The Motorola S type hex format file consists of five records[Note 1]: S0 record as the header record, S2/S3 records as the data record, and S8/S7 record as the end records[Note 2].

Notes 1. Each record is output in ASCII code.

2. The Motorola S type hex formats are divided into two types: (24-bit) standard address and 32-bit address types. The format of the standard address type consists of S0, S2, and S8 records, and the format of the 32-bit address type consists of S0, S3, and S7 records.

The file configuration in the Motorola S type hex format is shown below.

Figure 3-2. File Configuration of Motorola S Type Hex Format



Each record consists of the following fields.

```
Sx   XX   YY......YY   SS   NL
(1)  (2)      (3)      (4)  (5)
```

Item Number	Description
(1)	Record type
(2)	Record length This is the number of bytes expressed as 2-digit hexadecimal number of (3) + number of bytes expressed by SS.[Note]
(3)	Field
(4)	Checksum This is lower 1 byte expressed as a 2-digit hexadecimal number of one’s complement of total of number of bytes in the record (other than Sx, SS, and NL) expressed as a 2-digit hexadecimal number.
(5)	Newline (\n)

Note This is 1.

- S0 record

  This indicates the file name.

```
S0   XX   XX......XX   SS   NL
(1)  (2)      (3)
```

Item Number	Description
(1)	Fixed at S0
(2)	Record length
(3)	File name This is the specified file name indicated in ASCII code.

- S2 record

This indicates the value of the code.

```
S2  XX  YYYYYY  ZZ......ZZ  SS  NL
(1) (2)   (3)        (4)
```

Item Number	Description
(1)	Fixed at S2
(2)	Record length
(3)	Load address 24 bits (0x0 to 0xffffff)
(4)	Code Each byte of code expressed as 2-digit hexadecimal number

- S3 record

This indicates the value of the code.

```
S3  XX  YYYYYY  ZZ......ZZ  SS  NL
(1) (2)   (3)        (4)
```

Item Number	Description
(1)	Fixed at S3
(2)	Record length
(3)	Load address 32 bits (0x0 to 0xffffffff)
(4)	Code Each byte of code expressed as 2-digit hexadecimal number

- S7 record

This indicates the entry point address.

```
S7  XX  YYYYYY  SS  NL
(1) (2)   (3)
```

Item Number	Description
(1)	Fixed at S7
(2)	Record length
(3)	Entry point address 32 bits (0x0 to 0xffffffff)

- S8 record

This indicates the entry point address.

```
S8  XX  YYYYYY  SS  NL
(1) (2)   (3)
```

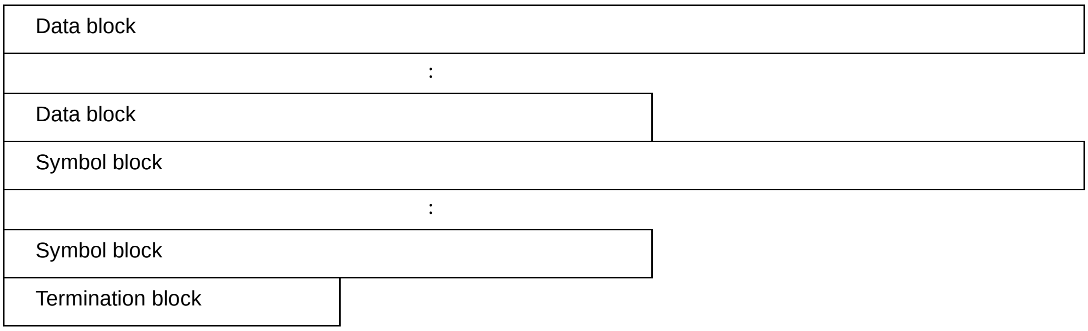
Item Number	Description
(1)	Fixed at S8
(2)	Record length
(3)	Entry point address 24 bits (0x0 to 0xffffff)

3.4.3 Expanded Textronix hex format

The Expanded Textronix hex format file consists of three blocks: data block, symbol block, and termination block.

The file configuration in Expanded Textronix hex format is shown below.

Figure 3-3. File Configuration of Expanded Tektronix Hex Format



Data block

:

Data block

Symbol block

:

Symbol block

Termination block

Each block consists of the following fields.

```
%    XX    X   SS   FF[FF......]   NL
(1)  (2)  (3)  (4)       (5)       (6)
```

Item Number	Description
(1)	Header character
(2)	Block length This is the number of characters in the block other than % and NL.
(3)	Type of block 6: Data block 3: Symbol block 8: Termination block
(4)	Checksum This is the remainder expressed as a 2-digit hexadecimal number that results from dividing total value[Note] of characters in blocks other than %, SS, and NL, by 256.
(5)	The specification differs depending on each block.
(6)	Newline (\n)

Note The value for each character is determined as follows:

0 to 9: 0 to 9, A to Z: 10 to 35, $: 36, %: 37, .: 38, -: 39, a to z: 40 to 65

- Data block

This indicates the value of the code.

```
%    XX   6    SS   XXXX   DD......DD   NL
(1)  (2)  (3)  (4)   (5)        (6)
```

Item Number	Description
(1)	Header character
(2)	Block length
(3)	Fixed at 6
(4)	Checksum
(5)	Number of digits in the load address and the load address 2 to 17 bytes
(6)	Code Each byte of code expressed as 2-digit hexadecimal number

Example

```
%   15  6   1C  3 100 020202020202  NL
(1) (2) (3) (4)  (5)       (6)
```

Item Number	Description
(1)	Header character
(2)	Block length
(3)	Type of block 6
(4)	Checksum This is the remainder expressed as a 2-digit hexadecimal number that results from dividing 1 + 5 + 6 + 3 + 1 + 0 + 0 + 0 + 2 + 0 + 2 + 0 + 2 + 0 + 2 + 0 + 2 + 0 + 2 = 28 by 256.
(5)	Number of digits in the load address is 3, and the load address is 100.
(6)	Each byte of code expressed as 2-digit hexadecimal number

- Symbol block

This indicates the value of the symbol.

```
%   XX  3   SS  XX...XX XX...XX XX...XX NL
(1) (2) (3) (4)   (5)      (6)      (7)
```

Item Number	Description
(1)	Header character
(2)	Block length
(3)	Fixed at 3
(4)	Checksum
(5)	Number of characters of the section name and the section name 2 to 17 bytes
(6)	Section definition field (SEDF)Note 1 5 to 35 bytes

Item Number	Description
(7)	Symbol definition field (SYDF)[Note 2] 5 to 35 bytes

Notes 1. Section definition field

One section definition field must exist in each section. A section definition field can be followed by or can follow any of symbol definition fields.

```
0   XX...XX XX...XX
(1)   (2)     (3)
```

Item Number	Description
(1)	Fixed at 0 This indicates that this field is the section definition field.
(2)	Number of digits in the base address of the section and the base address of the section 2 to 17 bytes
(3)	Number of digits in the length of the section and the length of the section 2 to 17 bytes

2. Symbol definition field

```
T   XX...XX XX...XX
(1)   (2)     (3)
```

Item Number	Description
(1)	Type of symbol 1: Global address (symbol having binding class GLOBAL and type other than ABS) 2: Global scalar (symbol having binding class GLOBAL and type ABS) 3: Global code address 4: Global code address 5: Local address (symbol having binding class GLOBAL and type other than ABS) 6: Local scalar (symbol having binding class GLOBAL and type ABS) 7: Global code address 8: Global code address
(2)	Number of characters of the symbol and the symbol 2 to 17 bytes
(3)	Number of digits of the symbol and the symbol 2 to 17 bytes

Examples 1.

```
%   37  3   60  8SVCSTUFF 02402C6 22CR1D14OPEN25014READ25815WRITE260   NL
(1) (2) (3) (4)     (5)       (6)                     (7)
```

Item Number	Description
(1)	Header character
(2)	Block length
(3)	Type of block 3
(4)	Checksum
(5)	The number of the characters of the section name is 8 and the section name is SVCSTUFF.
(6)	Section definition field The number of digits in the base address of the section is 2, the base address of the section is 40, the number of digits in the length of the section is 2, and the length of the section is C6.
(7)	Symbol definition field 22CR1D/14OPEN250/14READ258/15WRITE260

2.

```
%   37  3   C8  8SVCSTUFF   15CLOSE26814EXIT27029BUFLENGTH28013BUF278 NL
(1) (2) (3) (4)    (5)                         (6)
```

Item Number	Description
(1)	Header character
(2)	Block length
(3)	Type of block 3
(4)	Checksum
(5)	The number of the characters of the section name is 8 and the section name is SVCSTUFF.
(6)	Symbol definition field 15CLOSE268/14EXIT270/29BUFLENGTH280/13BUF278

- Termination block

  This indicates the entry point address.

```
%   XX  8   SS  YY......YY  NL
(1) (2) (3) (4)     (5)
```

Item Number	Description
(1)	Header character
(2)	Block length
(3)	Fixed at 8
(4)	Checksum
(5)	Number of digits in the entry point address and the entry point address 2 to 17 bytes

Example

```
%    08   8    1A   280    NL
(1)  (2)  (3)  (4)  (5)
```

Item Number	Description
(1)	Header character
(2)	Block length
(3)	Type of block 8
(4)	Checksum
(5)	The number of digits in the entry point address is 2, and the entry point address is 80.

APPENDIX A WINDOW REFERENCE

This appendix explains windows/panels/dialog boxes used in build process.

A.1 Description

The following lists the windows/panels/dialog boxes used in build process.

Table A-1. List of Windows/Panels/Dialog Boxes

Window/Panel/Dialog Box Name	Function Description
Main window	This is the first window to be opened when CubeSuite is launched.
Project Tree panel	This panel is used to display the project components in tree view.
Property panel	This panel is used to display the detailed information on the Build tool node, file, or category node that is selected on the Project Tree panel and change the settings of the information.
Editor panel	This panel is used to display and edit text files and source files.
Output panel	This panel is used to display the message that is output from the build tool.
Add File dialog box	This dialog box is used to create a new file and add it to the project.
Add Folder And File dialog box	This dialog box is used to add existing files and folder hierarchies to the project.
Character String Input dialog box	This dialog box is used to input and edit characters in one line.
Text Edit dialog box	This dialog box is used to input and edit texts in multiple lines.
Path Edit dialog box	This dialog box is used to edit or add the path or file name including the path.
System Include Path Order dialog box	This dialog box is used to refer the system include paths specified for the compiler and set their specified sequence.
File Save Settings dialog box	This dialog box is used to set the encoding and newline code of the file that is editing on the Editor panel.
Link Directive File Generation dialog box	This dialog box is used to generate a link directive file.
Object File Select dialog box	This dialog box is used to select the object file and set it to the area that this dialog box is called from.
Segment Select dialog box	This dialog box is used to select the segment and set it to the area that this dialog box is called from.
Link Order dialog box	This dialog box is used to refer files to be input to the linker and configure these link orders.
Output Makefile Information File dialog box	This dialog box is used to output a text file described information for creating a Makefile.
Build Mode Settings dialog box	This dialog box is used to add and delete build modes and configure the current build mode in batch.
Batch Build dialog box	This dialog box is used to do build, rebuild and clean process in batch with the build mode that the project has.
Go to the Location dialog box	This dialog box is used to move the caret to the designated location.
Progress Status dialog box	This dialog box is used to show how the process has been progressed.
Option dialog box	This dialog box is used to configure the CubeSuite environment.
Add Existing File dialog box	This dialog box is used to select existing files to add to the project.

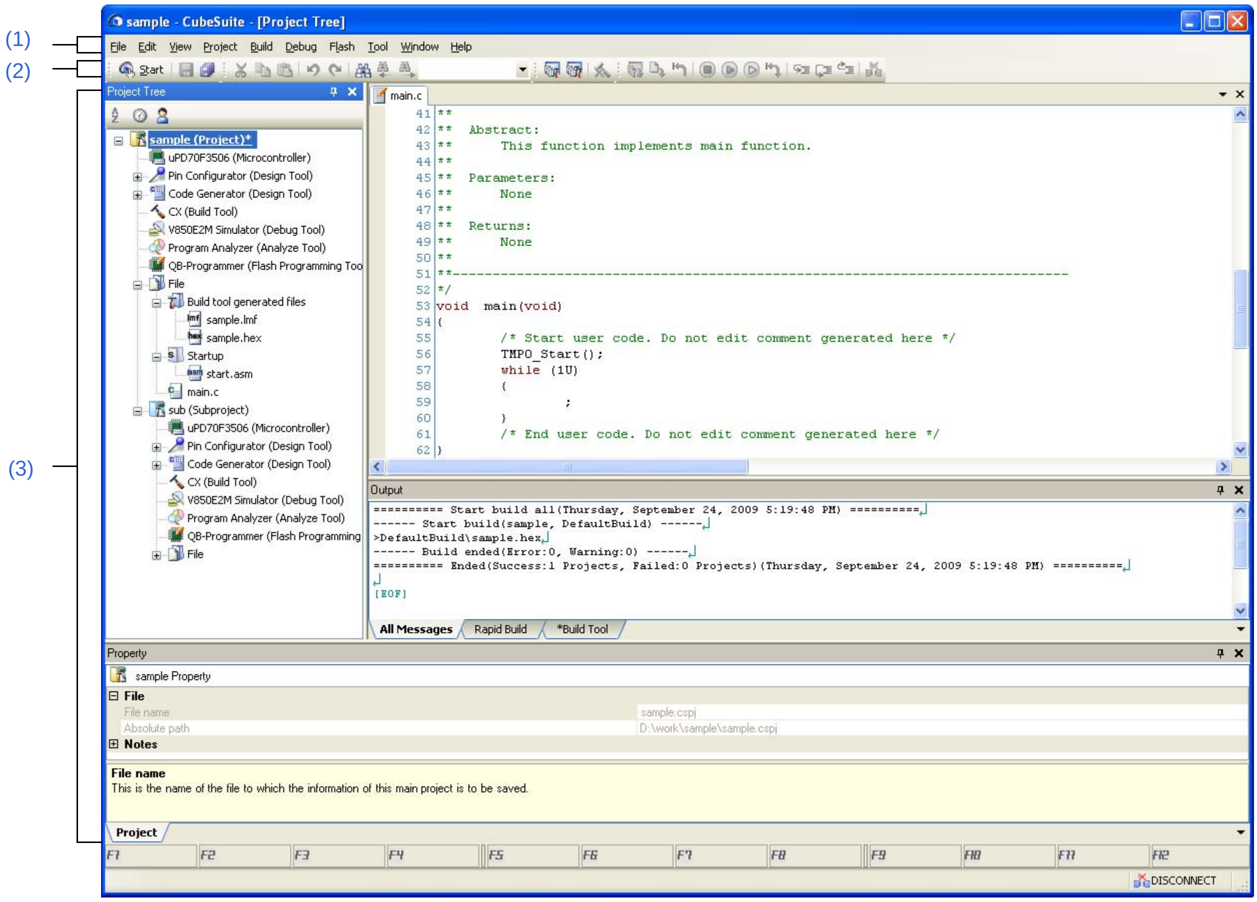
Window/Panel/Dialog Box Name	Function Description
Browse For Folder dialog box	This dialog box is used to select the folder and set it to the area that this dialog box is called from.
Specify Boot Area Load Module File dialog box	This dialog box is used to select the boot area load module file and set it to the area that this dialog box is called from.
Specify Far Jump File dialog box	This dialog box is used to select the Far Jump file and set it to the area that this dialog box is called from.
Specify ROMization Area Reservation Code File dialog box	This dialog box is used to select the ROMization area reservation code file and set it to the area that this dialog box is called from.
Save As dialog box	This dialog box is used to save the editing file or contents of each panel to a file with a name.
Open with Program dialog box	This dialog box is used to select the application to open the file.
Stack Usage Tracer window	This is the first window to be opened when the stack usage tracer is launched.
Stack Size Unknown / Adjusted Function Lists dialog box	This dialog box is used to display a list of functions for which the stack usage tracer could not obtain stack information; functions for which information was changed intentionally, and functions for which the stack usage tracer forcibly set an additional margin.
Adjust Stack Size dialog box	This dialog box is used to change the information for the selected function.
Open dialog box	This dialog box is used to open an existing stack size specification file.

Main window

This is the first window to be opened when CubeSuite is launched.

This window is used to control the user program execution and open panels for the build process.

Figure A-1. Main Window


(1)
(2)
(3)
sample - CubeSuite - [Project Tree]
File Edit View Project Build Debug Flash Tool Window Help
Start
Project Tree
sample (Project)*
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming Too
File
Build tool generated files
sample.lmf
sample.hex
Startup
start.asm
main.c
sub (Subproject)
uPD70F3506 (Microcontroller)
Pin Configurator (Design Tool)
Code Generator (Design Tool)
CX (Build Tool)
V850E2M Simulator (Debug Tool)
Program Analyzer (Analyze Tool)
QB-Programmer (Flash Programming
File
main.c
Output
All Messages
Rapid Build
*Build Tool
Property
sample Property
Project
F1
F2
F3
F4
F5
F6
F7
F8
F9
F10
F11
F12
DISCONNECT

```
**
**   Abstract:
**       This function implements main function.
**
**   Parameters:
**       None
**
**   Returns:
**       None
**
**-----------------------------------------------------------------------------
*/
void  main(void)
{
        /* Start user code. Do not edit comment generated here */
        TMPO_Start();
        while (1U)
        {
                ;
        }
        /* End user code. Do not edit comment generated here */
}
```

```
========== Start build all(Thursday, September 24, 2009 5:19:48 PM) ==========↲
------ Start build(sample, DefaultBuild) ------↲
>DefaultBuild\sample.hex↲
------ Build ended(Error:0, Warning:0) ------↲
========== Ended(Success:1 Projects, Failed:0 Projects)(Thursday, September 24, 2009 5:19:48 PM) ==========↲
↲
[EOF]
```

⊟ File

File name	sample.cspj
Absolute path	D:\work\sample\sample.cspj

⊞ Notes

File name
This is the name of the file to which the information of this main project is to be saved.

The following items are explained here.

- [How to open]
- [Description of each area]

[How to open]

- Select Windows® [start] >> [All programs] >> [NEC Electronics CubeSuite] >> [CubeSuite]

[Description of each area]

(1) Menu bar

The menus used in build process are displayed.

(a) [Project]

The [Project] menu shows menu items to operate the project and others.

Add New Subproject...	Closes the current project and opens the Create Project dialog box to create a new project. If the currently open project or file has been modified but it has not been saved yet, a confirmation message is displayed to ask you whether you want to save it.
Open Project...	Closes the current project and opens the Open Project dialog box to open the existing project. If the currently open project or file has been modified but it has not been saved yet, a confirmation message is displayed to ask you whether you want to save it.
Favorite Projects	Displays a cascading menu to use to open or save your favorite project.
1*path*	Opens your favorite project registered with [Favorite Projects] >> [1 Register to Favorite Project]. If no project has been registered, "Favorite Projects" is displayed.
2*path*	Opens your favorite project registered with [Favorite Projects] >> [2 Register to Favorite Project]. If no project has been registered, "Favorite Projects" is displayed.
3*path*	Opens your favorite project registered with [Favorite Projects] >> [3 Register to Favorite Project]. If no project has been registered, "Favorite Projects" is displayed.
4*path*	Opens your favorite project registered with [Favorite Projects] >> [4 Register to Favorite Project]. If no project has been registered, "Favorite Projects" is displayed.
1 Register to Favorite Project	The current project path is added to [1*path*] in [Favorite Projects].
2 Register to Favorite Project	The current project path is added to [2*path*] in [Favorite Projects].
3 Register to Favorite Project	The current project path is added to [3*path*] in [Favorite Projects].
4 Register to Favorite Project	The current project path is added to [4*path*] in [Favorite Projects].
Add	Displays the cascading menu to add subprojects to the project.
Add Subproject...	Opens the Add Existing Subproject dialog box to add an existing subproject to the project.
Add New Subproject...	Opens the Create Project dialog box to add a new subproject to the project.
Add File...	Opens the Add Existing File dialog box to add the selected file to the project.
Add New File...	Opens the Add File dialog box to create a file with the selected file type and add to the file to the project. The added file can be opened with the application corresponds to the file extension.
Add New Category	Adds a new category node to the root of the File node. This allows the category name to be changed. The default category name is "New category". You can also add a category node with the same name as an existing category node. Note that this menu is disabled when the build tool is in operation.

Sets *selected project or subproject* to an active project.	Sets the selected project or subproject to an active project.
Close Project	Closes the current project. If the currently open project or file has been modified but it has not been saved yet, a confirmation message is displayed to ask you whether you want to save it.
Save Project	Saves the setting information of the current project to the project file.
Save Project As...	Opens the Save Project As dialog box to save the setting information of the current project to the project file with another name.
Remove from Project	Removes the selected project or subproject from the project. The subproject files or the file themselves are not deleted from the file system.
Save Project and CubeSuite as Package...	Saves a set of the CubeSuite and the project by copying them in a folder.

(b) [Build]

The [Build] menu shows menu items for the build process and others.

Build Project	Runs a build of the project. A build of a subproject is also run when it is added in the project. Note that this menu is disabled when the build tool is in operation.
Rebuild Project	Runs a rebuild of the project. A rebuild of a subproject is also run when it is added in the project. Note that this menu is disabled when the build tool is in operation.
Clean Project	Runs a clean of the project. A clean of a subproject is also run when it is added in the project. Note that this menu is disabled when the build tool is in operation.
Rapid build	Toggles the rapid build function between enabled (default) and disabled.
Update Dependencies	Updates the dependency of a file in the project to build. The dependency of a file in the subproject to be build is also updated when the subproject is added to the project.
Build *active project*	Runs a build of the active project. If the active project is the main project, a build of its subproject is not run. Note that this menu is disabled when the build tool is in operation.
Rebuild *active project*	Runs a rebuild of the active project. If the active project is the main project, a rebuild of its subproject is not run. Note that this menu is disabled when the build tool is in operation.
Clean *active project*	Runs a clean of the active project. If the active project is the main project, a clean of its subproject is not run. Note that this menu is disabled when the build tool is in operation.
Update Dependencies of *active project*	Updates the dependency of a file in the active project to build.
Stop Build	Cancels the build, rebuild, batch build and clean operation.
Build Mode Settings...	Opens the Build Mode Settings dialog box to modify and add to the build mode.
Batch Build...	Opens the Batch Build dialog box to run a batch build.
Build Option List	Lists the currently set build options in the Output panel.

(2) Toolbar

The buttons used in build process are displayed.

(a) Build toolbar

Build toolbar shows buttons used in build process.

	Runs a build of the project. A build of a subproject is also run when it is added in the project. Note that this button is disabled when the build tool is in operation.
	Runs a rebuild of the project. A rebuild of a subproject is also run when it is added in the project. Note that this button is disabled when the build tool is in operation.
	Cancels the build, rebuild, batch build and clean operation.

(3) Panel display area

The following panels are displayed in this area.

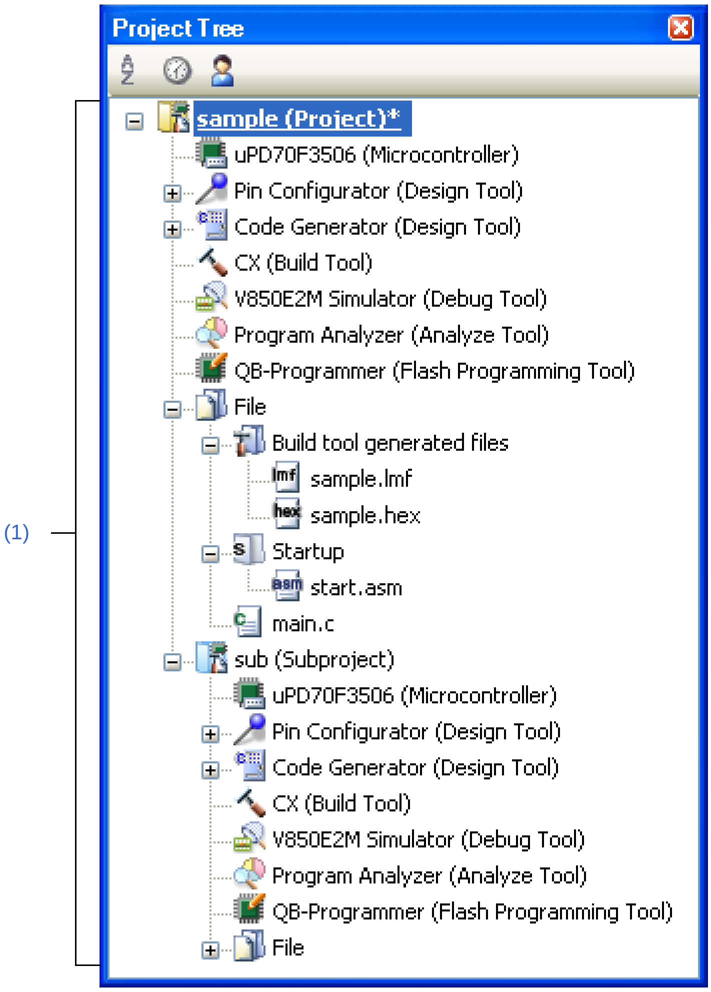
- Project Tree panel
- Property panel
- Editor panel
- Output panel

See the each panel section for details of the contents of the display.

Project Tree panel

This panel is used to displays the project components such as the build tool, source files, etc. in tree view.

Figure A-2. Project Tree Panel



- sample (Project)*
  - uPD70F3506 (Microcontroller)
  - Pin Configurator (Design Tool)
  - Code Generator (Design Tool)
  - CX (Build Tool)
  - V850E2M Simulator (Debug Tool)
  - Program Analyzer (Analyze Tool)
  - QB-Programmer (Flash Programming Tool)
  - File
    - Build tool generated files
      - sample.lmf
      - sample.hex
    - Startup
      - start.asm
    - main.c
  - sub (Subproject)
    - uPD70F3506 (Microcontroller)
    - Pin Configurator (Design Tool)
    - Code Generator (Design Tool)
    - CX (Build Tool)
    - V850E2M Simulator (Debug Tool)
    - Program Analyzer (Analyze Tool)
    - QB-Programmer (Flash Programming Tool)
    - File

The following items are explained here.

- [How to open]
- [Description of each area]
- [[Edit] menu (only available for the Project Tree panel)]
- [Context menu]

[How to open]

- From the [View] menu, select [Project Tree].

[Description of each area]

(1) Project tree area

This area displays the project components in tree view with the following given nodes.

Node	Description
Project name (Project) (hereafter referred to as "Project node")	The project name.
Build tool name (Build tool) (hereafter referred to as "Build tool node")	The build tool (CX) used in the project.
File (hereafter referred to as "File node")	The following files that are added to the project are displayed directly below this node. - C source file (*.c) - Assembler source file (*.asm, *.s) - Header file (*.h, *.inc) - Object module file (*.obj, *.o) - Library file (*.lib) - Link directive file (*.dir, *.dr) - Symbol information file (*.sfg) - Other file (*.doc, *.xml, etc.)
Build tool generated files (hereafter referred to as "Build tool generated files node")	The following files generated by the build tool appear directly below the node generated during the build. - For other than library projects Load module file (*.lmf) Link map file (*.map) Hex file (*.hex) Error message file (*.err) - For library projects Library file (*.lib) Error message file (*.err) Files displayed below this node cannot be renamed, deleted, or moved. This node is always generated below the File node. This node will no longer appear if you reload the project after building.
Startup (hereafter referred to as "Startup node")	The node for adding other than the standard startup routine to the project. This node is always displayed below the File node.
Category name (hereafter referred to as "category node")	Categories that the user creates to categorize files (see "2.3.6 Classify a file into a category"). This node is always created below the File node.
Subproject name (Subproject) (hereafter referred to as "Subproject node")	Subprojects added to the project.

When each component (the node or file) is selected, the detailed information (property) is displayed on the Property panel. You can change the settings.

Remark When multiple components are selected, only the tab that is common to all the components is displayed.
When multiple files are selected and the values of their common properties are different, then the corresponding value fields are displayed blank.

This area has the following functions.

(a) Adding a file

You can add a file by one of the following procedures.
The files are added below the File node.

<1> Adding an existing file

- Select either one of the Project node, Subproject node, File node or a file. Then select [Add] >> [Add File...] from the [File] menu. The Add Existing File dialog box will open. Select the file to be added.
- Select either one of the Project node, Subproject node, File node. Then select [Add] >> [Add File...] from the context menu. The Add Existing File dialog box will open. Select the file to be added.
- Copy the file using such as Explorer and then point the mouse to this area. Select [Paste] from the [Edit] menu.
- Drag a file using such as Explorer and drop it onto the location in this area where you want to add it to.

Remark If you drag the file using such as Explorer and drop it onto the empty space below the lower project tree, it is assumed that you have dropped it onto the Main project.

<2> Adding a new file

- Select either one of the Project node, Subproject node, File node or a file. Then select [Add] >> [Add New File...] from the [File] menu. The Add File dialog box will open. Designate the file to be created.
- Select either one of the Project node, Subproject node, File node. Then select [Add] >> [Add New File...] from the context menu. The Add File dialog box will open. Designate the file to be created.

Remark A blank file is created at the location designated in the Add File dialog box.

(b) Removing a file from the project

You can remove a file from the project by one of the following procedures.
The file itself is not deleted from the file system.

- Select the file you want to remove from the project and then select [Remove from Project] from the [Project] menu.
- Select the file you want to remove from the project and then select [Remove from Project] from the context menu.

(c) Moving a file

You can move a file by the following procedure.
The file is moved below the File node.

- Drag the file you want to move and then drop it onto the destination.

Remarks 1. If you drop the file in the main project or subproject, the individual options set in the file are retained.

2. If you drop the file between the different projects, or onto the main project or subproject in same project, the file is copied, not moved.
Note that the individual option set in the file are not retained.

(d) Adding a category

You can add a category node by one of the following procedures.
The category node is added below the File node.

- Select [Add New Category] from the [Project] menu.
- Select [Add New Category] from the context menu of either one of the Project node, Subproject node, or File node.

Remarks 1. The default category name is "New category".
2. You can also add a category node with the same name as the existing category node.

(e) Moving a category

You can move a category node by the following procedure.
The category node is moved below the File node.

- Drag the category node you want to move and then drop it onto the destination.

Remarks 1. If you drop the category node in the main project or subproject, the individual options set in the file in the category node are retained.
2. If you drop the category node between the different projects, or onto the main project or subproject in same project, the category node is copied, not moved.
Note that the individual option set in the file in the category node are not retained.

(f) Adding a folder

You can add a folder using such as Explorer by the following procedure.
The folder is added below the File node.
The folder is added as a category.

- Drag the folder using such as Explorer and drop it onto the destination. The Add Folder And File dialog box will open. Specify the file types and subfolder levels in the folder to be added.

Caution You cannot drag and drop a folder and file onto this area simultaneously.

(g) Editing the display of the subproject build order

Subprojects are displayed in order of build from the top.
Therefore, the build order can be changed by changing the display order of subprojects.
Builds of the project are run from the subprojects then the main project.

(h) Setting the standard build options

On the Property panel, if you add a change to the settings for the standard build options, the value of the property will be displayed in boldface.
You can change the current build options to the standard setting (cancel boldface) by the following procedure.

- Select the Build tool node and then select [Set to Default Build Option for Project] in the context menu.

Remark The setting of the standard build options takes effect to the whole project (main project and subprojects).

(i) Sorting files and categories

You can sort files and category nodes in order of the file name, time stamp, or user definition by the following procedure.

- Select one of the buttons on the toolbar.

The description of each button is shown below.

👤 is selected by default.

Button	Description
AZ AZ↓ ZA↓	Sorts category nodes and files in order of their names. AZ : Ascending order AZ↓ : Descending order ZA↓ : Ascending order
🕐 🕐↑ 🕐↓	Sorts category nodes and files in order of their timestamp. 🕐 : Descending order 🕐↑ : Ascending order 🕐↓ : Descending order
👤	Displays category nodes and files in order of the user definition (default). You can change the display order of category nodes and files arbitrarily by dragging and dropping them.

(j) Displaying the editing file

When the file added to the project is edited in the Editor panel and the file is not saved once, the file name is followed by "*".

If the file is saved, "*" will be deleted.

The file that is saved	main.c
The file that is not saved after editing	main.c*

(k) Displaying the source file in boldface that the individual build option is set

The source file icon whose option is different from the project general options (individual compile options or individual assemble options) is changed to the different one from the normal icon.

The file with project general options	main.c
The file with the individual build options	✓ main.c

(l) Highlighting the file with the read-only attribute

The read-only file added to the project is displayed in italic.

The file without the read-only attribute	main.c
The file with the read-only attribute	*main.c*

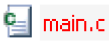
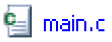
(m) Highlighting the file that does not exist

The file that is added to the project but does not exist is grayed out and its icon is dimmed.

The file that exists	main.c
The file that does not exist	*main.c*

(n) Highlighting the build-target file

<1> The file which an error occurred during building (rapid building), rebuilding, compiling or assembling is highlighted as the example below.

The file without errors or warnings	main.c
The file with an error	 main.c
The file with a warning	 main.c

Remarks 1. The file with both an error and a warning is highlighted in red.

2. The highlight is canceled when the build option (general option or individual option) or the build mode is changed.

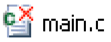
<2> The names of the following files are displayed in boldface.

- The source file that has not been compiled after editing
- The source file after a clean has been executed
- The source file after the build tool option has been changed
- The source file after the build mode has been changed

Remark The file names are all displayed in boldface right after the project is opened. The boldface display is canceled after a build is executed.

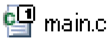
(o) Highlighting the non build-target file

The file that is set as non build-target is highlighted as shown in the example below.

Build-target file	main.c
Non build-target file	 main.c

(p) Highlighting the file that core-number is specified

In a project using a device supporting multi-core, the icon of the file specifying the core number will change to an icon with the core number added.

Ordinary file	main.c
File specifying core number 1	 main.c

Remarks 1. Core numbers 1 to 4 are supported.

2. Files that are not build targets are not targeted by this function.

(q) Highlighting the project that has been changed

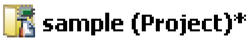
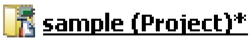
The file component that is added to the project and the property of the project component are changed, the project name is followed by "*" and is displayed in boldface.

The boldface is canceled when the project is saved.

The project that has not been changed	sample (Project)
The project that has been changed	**sample (Project)***

(r) Highlighting the active project

The active project is underlined.

Non-active project	 sample (Project)*
Active project	 sample (Project)*

(s) Running the editor

The file with the specific extension is opened in the Editor panel. When an external editor is set to be used in the Option dialog box, the file is opened with the external editor that has been set. Other files are opened with the applications associated by the host OS.

Caution The file with the extension that is not associated with the host OS is not displayed.

You can open the editor by one of the following procedures.

- Double click the file.
- Select a file and then select [Open] from the context menu.
- Select a file and then press the [Enter] key.

The files that can be opened with the Editor panel are shown below.

- C source file (*.c)
- Assembler source file (*.asm, *.s)
- Header file (*.h, *.inc)
- Symbol information file (*.sfg)
- Link directive file (*.dir, *.dr)
- Link map file (*.map)
- Hex file (*.hex)
- Text file (*.txt)

Remark You can use one of the procedures below to open files other than those listed above in the Editor panel.

- Drag a file and drop it onto the Editor panel.
- Select a file and then select [Open with Internal Editor...] from the context menu.

[[Edit] menu (only available for the Project Tree panel)]

Copy	Copies the selected file or category node to the clipboard. When the file name or category name is in editing, the characters of the selection are copied to the clipboard. Note that this menu is enabled when a file or category is selected.
Paste	Inserts the contents of the clipboard directly below the selected node on the project tree. When the file name or category name is in editing, the contents of the clipboard are inserted. Note that this menu is disabled when the contents of the clipboard exist in the same project, when multiple files and category nodes are selected, and when the build tool is in operation.

Rename	You can rename the selected project, subproject, file, or category node.Press the [Enter] key to confirm editing. Press the [ESC] key to cancel editing. When the file is selected, the actual file name is also changed. When the selected file is added to other project, that file name is also changed. Note that this menu is enabled when a project, subproject, file, or category node is selected. Note that this menu is disabled when the build tool is in operation.

[Context menu]

(1) When the Project node is selected

Build *active project*	Runs a build of the active project. If the active project is the main project, a build of its subproject is not run. Note that this menu is disabled when the build tool is in operation.
Rebuild *active project*	Runs a rebuild of the active project. If the active project is the main project, a rebuild of its subproject is not run. Note that this menu is disabled when the build tool is in operation.
Clean *active project*	Runs a clean of the active project. If the active project is the main project, a clean of its subproject is not run. Note that this menu is disabled when the build tool is in operation.
Open Folder with Explorer	Opens the folder that contains the project file of the selected project with Explorer.
Add	Displays the cascading menu to add subprojects and files to the project.
Add Subproject...	Opens the Add Existing Subproject dialog box to add the selected subproject to the project.
Add New Subproject...	Opens the Create Project dialog box to add the created subproject to the project.
Add File...	Opens the Add Existing File dialog box to add the selected file to the project.
Add New File...	Opens the Add File dialog box to create a file with the selected file type and add to the file to the project. The added file can be opened with the application corresponds to the file extension.
Add New Category	Adds a category node directly below the File node. This allows the category name to be changed. Up to 200 characters can be specified. The default category name is "New category". You can also add a category node with the same name as the existing category node. This menu is disabled when the build tool is in operation and when categories are nested 20 levels.
Set *selected project* as Active Project	Sets the selected project to the active project.
Save Project and CubeSuite as Package...	Saves a set of the CubeSuite and the project by copying them in a folder.
Paste	This menu is always disabled.
Rename	You can rename the selected project.
Property	Displays the selected project's property on the Property panel.

(2) When the Subproject node is selected

Build *active project*	Runs a build of the active project. Note that this menu is disabled when the build tool is in operation.
Rebuild *active project*	Runs a rebuild of the active project. Note that this menu is disabled when the build tool is in operation.
Clean *active project*	Runs a clean of the active project. Note that this menu is disabled when the build tool is in operation.
Open Folder with Explorer	Opens the folder that contains the subproject file of the selected subproject with Explorer.
Add	Displays the cascading menu to add subprojects, files, and category nodes to the project.
Add Subproject...	Opens the Add Existing Subproject dialog box to add the selected subproject to the project. The subproject cannot be added to another subproject.
Add New Subproject...	Opens the Create Project dialog box to add the created subproject to the project. The subproject cannot be added to another subproject.
Add File...	Opens the Add Existing File dialog box to add the selected file to the project.
Add New File...	Opens the Add File dialog box to create a file with the selected file type and add to the file to the project. The added file can be opened with the application corresponds to the file extension.
Add New Category	Adds a category node directly below the File node. This allows the category name to be changed. Up to 200 characters can be specified. The default category name is "New category". You can also add a category node with the same name as the existing category node. This menu is disabled when the build tool is in operation and when categories are nested 20 levels.
Set *selected subproject* as Active Project	Sets the selected subproject to the active project.
Remove from Project	Removes the selected subproject from the project. The subproject file itself is not deleted from the file system. When the selected subproject is the active project, it cannot be removed from the project. Note that this menu is disabled when the build tool is in operation.
Paste	This menu is always disabled.
Rename	You can rename the selected subproject.
Property	Displays the selected subproject's property on the Property panel.

(3) When the Build tool node is selected

Build Project	Runs a build the selected project (main project or subproject). A build of a subproject is also run when it is added in the project. Note that this menu is disabled when the build tool is in operation.
Rebuild Project	Runs a rebuild the selected project (main project or subproject). A rebuild of a subproject is also run when it is added in the project. Note that this menu is disabled when the build tool is in operation.

Clean Project	Runs a clean of the selected project (main project or subproject). A clean of a subproject is also run when it is added in the project. Note that this menu is disabled when the build tool is in operation.
Set to Default Build Option for Project	Sets the current build options as the standard for the project. When a subproject is added, its setting is not made. When the build option that is different from the standard option is set, its property is displayed in boldface.
Set Link Order...	Opens the Link Order dialog box to display object module files and library files to and set their link orders. Note that this menu is disabled when the build tool is in operation.
Output Makefile Information File...	Opens Output Makefile Information File dialog box to output a text file described information for creating a Makefile. Note that this menu is disabled when the build tool is in operation.
Create Link Directive File...	Opens the Link Directive File Generation dialog box to create the link directive file.
Property	Displays the selected build tool's property on the Property panel.

(4) When the Files node is selected

Add	Shows the cascading menu to add files and category nodes to the project.
Add File...	Opens the Add Existing File dialog box to add the selected file to the project. The file is added directly below this node. The added file can be opened with the application corresponds to the file extension.
Add New File...	Opens the Add File dialog box to create a file with the selected file type and add to the file to the project. The file is added directly below this node. The added file can be opened with the application corresponds to the file extension.
Add New Category	Adds a category node directly below this node. This allows the category name to be changed. Up to 200 characters can be specified. The default category name is "New category". You can also add a category node with the same name as the existing category node. This menu is disabled when the build tool is in operation and when categories are nested 20 levels.
Remove from Project	This menu is always disabled.
Copy	This menu is always disabled.
Paste	Inserts the contents of the clipboard directly below this node. However, this menu is disabled when the contents of the clipboard exist in the same project.
Rename	This menu is always disabled.
Property	Displays this node's property on the Property panel.

(5) When a file is selected

Compile	Compiles the selected C source file. Note that this menu is only displayed when a C source file (except a non build-target file) is selected. Note that this menu is disabled when the build tool is in operation.

Assemble	Assembles the selected assembler source file. Note that this menu is only displayed when an assembler source file (except a non build-target file) is selected. Note that this menu is disabled when the build tool is in operation.
Open	Opens the selected file with the application corresponds to the file extension (see "(s) Running the editor"). Note that this menu is disabled when multiple files are selected.
Open with Internal Editor...	Opens the selected file with the Editor panel. Note that this menu is disabled when multiple files are selected.
Open with Selected Application...	Opens the Open with Program dialog box to open the selected file with the designated application. Note that this menu is disabled when multiple files are selected.
Open Folder with Explorer	Opens the folder that contains the selected file with Explorer.
Add	Shows the cascading menu to add files and category nodes to the project.
Add File...	Opens the Add Existing File dialog box to add the selected file to the project. The file is added to the same level as the selected file.
Add New File...	Opens the Add File dialog box to create a file with the selected file type and add to the file to the project. The file is added to the same level as the selected file. The added file can be opened with the application corresponds to the file extension.
Add New Category	Adds a new category node at the same level as the selected file. You can rename the category. Up to 200 characters can be specified. The default category name is "New category". You can also add a category node with the same name as the existing category node. This menu is disabled when the build tool is in operation and when categories are nested 20 levels.
Remove from Project	Removes the selected file from the project. The file itself is not deleted from the file system. Note that this menu is disabled when the build tool is in operation.
Copy	Copies the selected file to the clipboard. When the file name is in editing, the selected characters are copied to the clipboard.
Paste	This menu is always disabled.
Rename	You can rename the selected file. The actual file is also renamed. When the selected file is added to other project, that file name is also changed.
Property	Displays the selected file's property on the Property panel.

(6) When the Build tool generated files node is selected

Property	Displays this node's property on the Property panel.

(7) When the Startup node is selected

Add	Shows the cascading menu to add files and category nodes to the project.

Add File...	Opens the Add Existing File dialog box to add the selected file to the project. The file is added directly below this node. The added file can be opened with the application corresponds to the file extension.
Add New File...	Opens the Add File dialog box to create a file with the selected file type and add to the file to the project. The file is added directly below this node. The added file can be opened with the application corresponds to the file extension.
Add New Category	Adds a category node directly below this node. This allows the category name to be changed. Up to 200 characters can be specified. The default category name is "New category". You can also add a category node with the same name as the existing category node. This menu is disabled when the build tool is in operation and when categories are nested 20 levels.
Remove from Project	This menu is always disabled.
Copy	This menu is always disabled.
Paste	Inserts the contents of the clipboard directly below this node. However, this menu is disabled when the contents of the clipboard exist in the same project.
Rename	This menu is always disabled.
Property	Displays this node's property on the Property panel.

(8) When a category node is selected

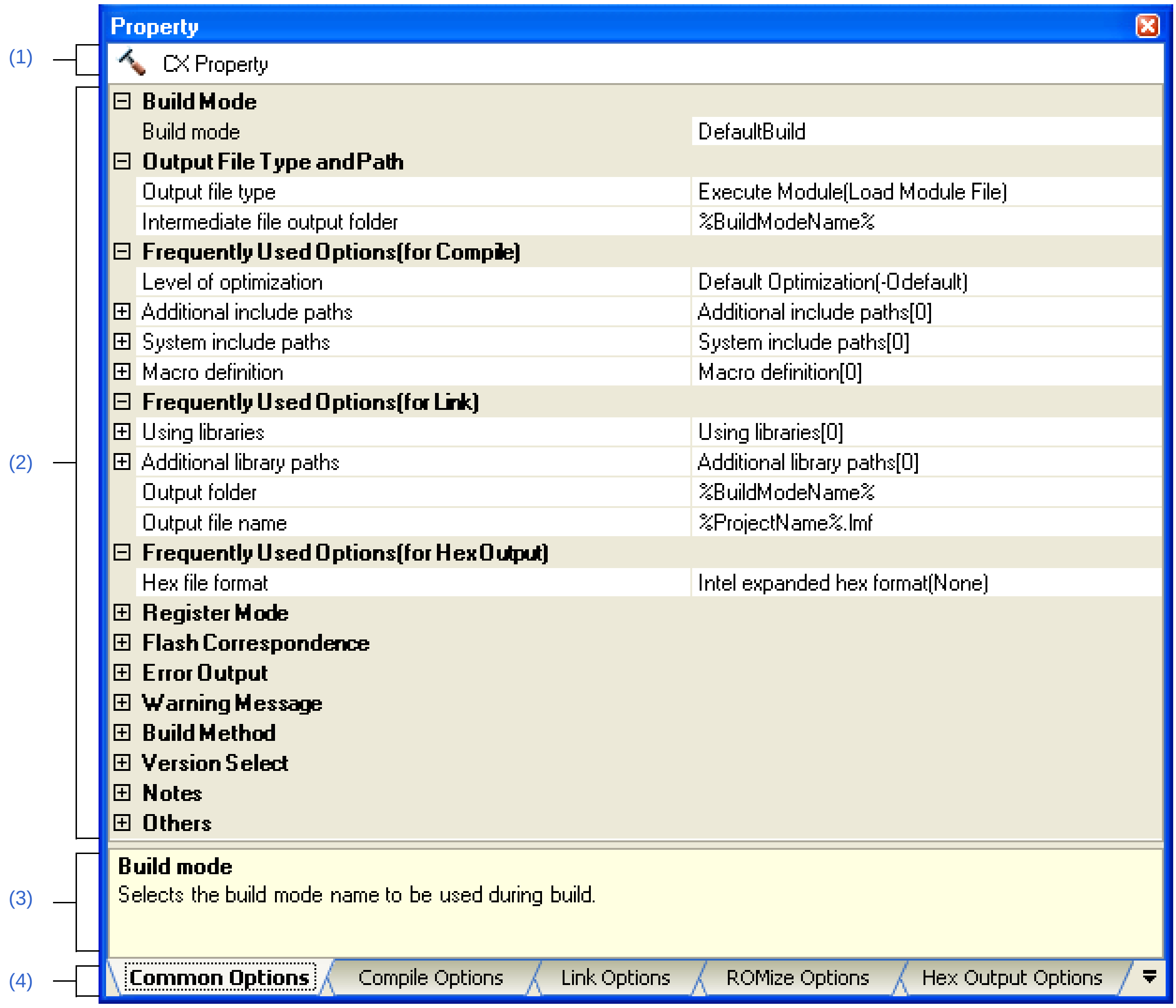
Add	Shows the cascading menu to add files and category nodes to the project.
Add File...	Opens the Add Existing File dialog box to add the selected file to the project. The file is added directly below this node. The added file can be opened with the application corresponds to the file extension.
Add New File...	Opens the Add File dialog box to create a file with the selected file type and add to the file to the project. The file is added directly below this node. The added file can be opened with the application corresponds to the file extension.
Add New Category	Adds a category node directly below this node. This allows the category name to be changed. Up to 200 characters can be specified. The default category name is "New category". You can also add a category node with the same name as the existing category node. This menu is disabled when the build tool is in operation and when categories are nested 20 levels.
Remove from Project	Removes the selected category node from the project. Note that this menu is disabled when the build tool is in operation.
Copy	Copies the selected category node to the clipboard. When the category name is in editing, the characters of the selection are copied to the clipboard.
Paste	Inserts the contents of the clipboard directly below this node. However, this menu is disabled when the contents of the clipboard exist in the same project. When the category name is in editing, the contents of the clipboard are inserted.
Rename	You can rename the selected category node.

Property	Displays the selected category node's property on the Property panel.

Property panel

This panel is used to display the detailed information on the Build tool node, file, or category node that is selected on the Project Tree panel by every category and change the settings of the information.

Figure A-3. Property Panel



CX Property

⊟ **Build Mode**	
Build mode	DefaultBuild
⊟ **Output File Type and Path**	
Output file type	Execute Module(Load Module File)
Intermediate file output folder	%BuildModeName%
⊟ **Frequently Used Options(for Compile)**	
Level of optimization	Default Optimization(-Odefault)
⊞ Additional include paths	Additional include paths[0]
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊟ **Frequently Used Options(for Link)**	
⊞ Using libraries	Using libraries[0]
⊞ Additional library paths	Additional library paths[0]
Output folder	%BuildModeName%
Output file name	%ProjectName%.lmf
⊟ **Frequently Used Options(for Hex Output)**	
Hex file format	Intel expanded hex format(None)
⊞ **Register Mode**	
⊞ **Flash Correspondence**	
⊞ **Error Output**	
⊞ **Warning Message**	
⊞ **Build Method**	
⊞ **Version Select**	
⊞ **Notes**	
⊞ **Others**	

Build mode
Selects the build mode name to be used during build.

The following items are explained here.

- [How to open]
- [Description of each area]
- [[Edit] menu (only available for the Property panel)]
- [Context menu]

[How to open]

- On the Project Tree panel, select the Build tool node, file, or category node, and then select [Property] from the [View] menu or [Property] from the context menu.

Remark When either one of the Build tool node, file, or category node on the Project Tree panel is selected while the Property panel has been opened, the detailed information of the selected item is displayed.

[Description of each area]

(1) Selected node area

This area displays the name of the selected node on the Project Tree panel.

When multiple nodes are selected, this area is blank.

(2) Detailed information display/change area

In this area, the detailed information on the Build tool node, file, or category node that is selected on the Project Tree panel is displayed by every category in the list. And the settings of the information can be changed directly.

Mark ⊟ indicates that all the items in the category are expanded. Mark ⊞ indicates that all the items are collapsed. You can expand/collapse the items by clicking these marks or double clicking the category name.

Mark HEX indicates that only a hexadecimal number is allowed to input in the text box.

See the section on each tab for the details of the display/setting in the category and its contents.

(3) Property description area

This area displays the brief description of the category and item selected in the detailed information display/change area.

(4) Tab selection area

Categories for the display of the detailed information are changed by selecting a tab.

In this panel, the following tabs are contained (see the section on each tab for the details of the display/setting on the tab).

(a) When the Build tool node is selected on the Project Tree panel

- [Common Options] tab
- [Compile Options] tab
- [Assemble Options] tab
- [Link Options] tab
- [ROMize Options] tab
- [Hex Output Options] tab
- [Create Library Options] tab

(b) When a file is selected on the Project Tree panel

- [Build Settings] tab (for C source file, assembler source file, object module file, link directive file, symbol information file, and library file)
- [Individual Compile Options] tab (for C source file)
- [Individual Assemble Options] tab (for assembler source file)
- [File Information] tab

(c) When the category node, Files node, Build tool generated files node, or Startup node is selected on the Project Tree panel

- [Category Information] tab

Remark When multiple components are selected on the Project Tree panel, only the tab that is common to all the components is displayed.

If the value of the property is modified, that is taken effect to the selected components all of which are common to all.

[[Edit] menu (only available for the Property panel)]

Undo	Cancels the previous edit operation of the value of the property.
Cut	While editing the value of the property, cuts the selected characters and copies them to the clipboard.
Copy	Copies the selected characters of the property to the clipboard.
Paste	While editing the value of the property, inserts the contents of the clipboard.
Delete	While editing the value of the property, deletes the selected characters.
Select All	While editing the value of the property, selects all the characters of the selected property.

[Context menu]

Undo	Cancels the previous edit operation of the value of the property.
Cut	While editing the value of the property, cuts the selected characters and copies them to the clipboard.
Copy	Copies the selected characters of the property to the clipboard.
Paste	While editing the value of the property, inserts the contents of the clipboard.
Delete	While editing the value of the property, deletes the selected characters.
Select All	While editing the value of the property, selects all the characters of the selected property.
Reset to Default	Restores the configuration of the selected item to the default configuration of the project. For the [Individual Compile Options] tab and [Individual Assemble Options] tab, restores to the configuration of the general option.
Reset All to Default	Restores all the configuration of the current tab to the default configuration of the project. For the [Individual Compile Options] tab and [Individual Assemble Options] tab, restores to the configuration of the general option.

[Common Options] tab

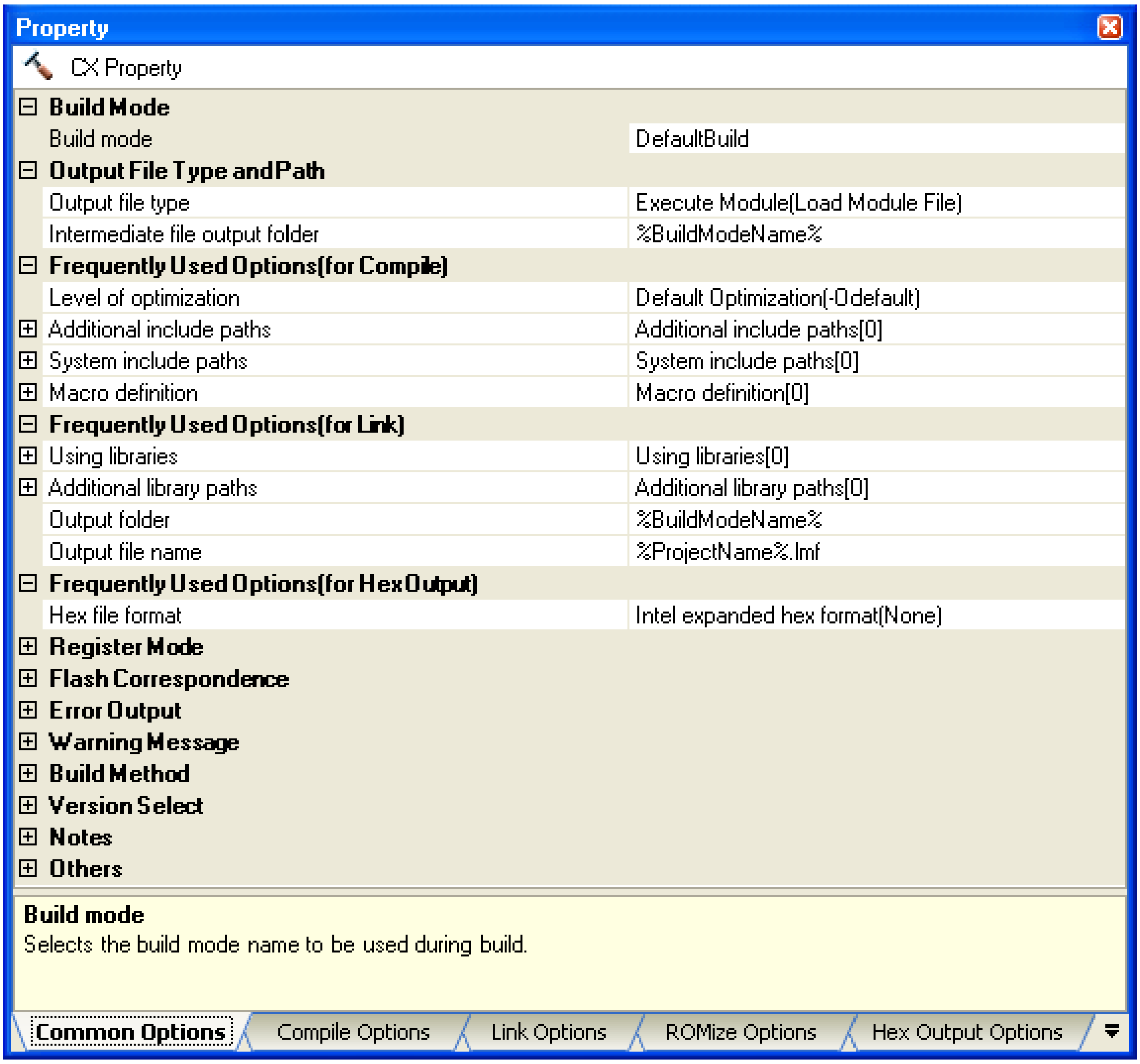
This tab shows the detailed information on the build tool categorized by the following and the configuration can be changed.

(1) [Build Mode]
(2) [Output File Type and Path]
(3) [Frequently Used Options(for Compile)]
(4) [Frequently Used Options(for Assemble)]
(5) [Frequently Used Options(for Link)]
(6) [Frequently Used Options(for Hex Output)]
(7) [Register Mode]
(8) [Flash Correspondence]
(9) [Error Output]
(10) [Warning Message]
(11) [Build Method]
(12) [Version Select]
(13) [Notes]
(14) [Others]

Remark If the property in the [Frequently Used Options] category is changed, the value of the property having the same name contained in the corresponding tab will be changed accordingly.

Category from [Common Options] Tab	Corresponding Tab
[Frequently Used Options(for Compile)] category	[Compile Options] tab
[Frequently Used Options(for Assemble)] category	[Assemble Options] tab
[Frequently Used Options(for Link)] category	[Link Options] tab
[Frequently Used Options(for Hex Output)] category	[Hex Output Options] tab

Figure A-4. Property Panel: [Common Options] Tab



⊟ **Build Mode**	
Build mode	DefaultBuild
⊟ **Output File Type and Path**	
Output file type	Execute Module(Load Module File)
Intermediate file output folder	%BuildModeName%
⊟ **Frequently Used Options(for Compile)**	
Level of optimization	Default Optimization(-Odefault)
⊞ Additional include paths	Additional include paths[0]
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊟ **Frequently Used Options(for Link)**	
⊞ Using libraries	Using libraries[0]
⊞ Additional library paths	Additional library paths[0]
Output folder	%BuildModeName%
Output file name	%ProjectName%.lmf
⊟ **Frequently Used Options(for Hex Output)**	
Hex file format	Intel expanded hex format(None)
⊞ **Register Mode**	
⊞ **Flash Correspondence**	
⊞ **Error Output**	
⊞ **Warning Message**	
⊞ **Build Method**	
⊞ **Version Select**	
⊞ **Notes**	
⊞ **Others**	

Build mode
Selects the build mode name to be used during build.

[Description of each category]

(1) [Build Mode]

The detailed information on the build mode is displayed and the configuration can be changed.

Build mode	Select the build mode to be used during a build. Note that this property is not applied to [Reset All to Default] from the context menu.		
	Default	DefaultBuild	
	How to change	Select from the drop-down list.	
	Restriction	DefaultBuild	Runs a build with the default build mode that is set when a new project is created.
		Build mode that is added to the project (other than DefaultBuild)	Runs a build with the build mode that is added to the project (other than DefaultBuild).

(2) [Output File Type and Path]

The detailed information on output file types and paths is displayed and the configuration can be changed.

Output file type	Select the type of the file to be generated during a build. The file type set here will be the debug target. For other than the library project, only [Execute Module(Load Module File)], [Execute Module(Non-ROMized Load Module File)], and [Execute Module(Hex File)] are displayed. For the library project, only [Library] is displayed.		
	Default	- For other than the library project Execute Module(Load Module File) - For the library project Library	
	How to change	Select from the drop-down list.	
	Restriction	Execute Module(Load Module File)	Generates a load module file and hex file during a build. The load module file will be the debug target.
		Execute Module(Non-ROMized Load Module File)	Generates a load module file and hex file during a build. When ROMization processing is performed, the non-ROMized load module file will be the debug target. This item is displayed only when [Yes] in the [Output ROMized load module file] property and [Yes] in the [Output Non-ROMized load module file] property in the [Output File] category from the [ROMize Options] tab is selected.
		Execute Module(Hex File)	Generates a load module file and hex file during a build. The hex file will be the debug target.
		Library	Generates a library file during a build.
Output common object module file for various devices	Select whether to output the object module file common to the various devices. This corresponds to the -Xcommon option of the cx command. This property is displayed only for the library project.		
	Default	No(specific device)(None)	
	How to change	Select from the drop-down list.	
	Restriction	Yes(V850E/ES core common)(-Xcommon=v850e)	Outputs the object module file compatible with models having the instruction set architecture superior to V850E (V850E, V850E2, or V850E2V3).
		Yes(V850E2 core common)(-Xcommon=v850e2)	Outputs the object module file compatible with models having the instruction set architecture superior to V850E2 (V850E2 or V850E2V3).
		Yes(V850E2V3 architecture common)(-Xcommon=v850e2v3)	Outputs the object module file compatible with models having the V850E2V3 instruction set architecture.
		No(specific device)(None)	The object module file having information specific to the specified device is output.

Intermediate file output folder	Specify the folder which the intermediate file is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xobj_path option of the cx command.	
	Default	%BuildModeName%
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.
	Restriction	Up to 247 characters

(3) [Frequently Used Options(for Compile)]

The detailed information on frequently used options during compilation is displayed and the configuration can be changed.

Level of optimization	Select the level of the optimization for compiling. This corresponds to the -O option of the cx command.		
	Default	Default Optimization(-Odefault)	
	How to change	Select from the drop-down list.	
	Restriction	Default Optimization(-Odefault)	Performs optimization that debugging is not affected (optimization of expressions and register allocation, and the like).
		Code Size Precedence(-Osize)	Performs optimization with the object size precedence. Regards reducing the ROM/RAM capacity as important and performs the maximum optimization that is effective for general programs.
		Speed Precedence(-Ospeed)	Performs optimization with the execution speed precedence. Regards shortening the execution speed as important and performs the maximum optimization that is effective for general programs.
		Debug Precedence(-Onothing)	Performs optimization with the debug precedence. Regards debugging as important and suppresses all optimization including default optimization.
		Detail Setting	The properties are added in the [Optimization(Details)] category from the [Compile Options] tab. The optimization items selected in the category has the precedence for the optimization.

Additional include paths	Specify the additional include paths during compiling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The specified include path is searched with higher priority than the standard include file folder of CX. The reference point of the path is the project folder. When this property is omitted, only the standard folder of CX is searched. This corresponds to the -I option of the cx command. The specified include path is displayed as the subproperty. When the include path is added to the project tree, the path is added to the top of the subproperties. Uppercase characters and lowercase characters are not distinguished for the include paths.	
	Default	Additional include paths[*number of defined items*]
	How to change	Edit by the Path Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 259 characters Up to 256 items can be specified.
System include paths	Change the specified order of the include paths which the system set during compiling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The system include path is searched with lower priority than the additional include path. The reference point of the path is the project folder. This corresponds to the -I option of the cx command. The include path is displayed as the subproperty.	
	Default	System include paths[*number of defined items*]
	How to change	Edit by the System Include Path Order dialog box which appears when clicking the [...] button.
	Restriction	Changes not allowed (Only the specified order of the include paths can be changed.)
Macro definition	Specify the name of the macro to be defined. Specify in the format of "*macro name=defined value*", with one macro name per line. The "=*defined value*" part can be omitted, and in this case, "1" is used as the defined value. This corresponds to the -D option of the cx command. The specified macro is displayed as the subproperty.	
	Default	Macro definition[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 256 characters Up to 256 items can be specified.

(4) [Frequently Used Options(for Assemble)]

The detailed information on frequently used options during assembling is displayed and the configuration can be changed.

This category is displayed when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.

Additional include paths	Specify the additional include paths during assembling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The specified include path is searched with higher priority than the standard include file folder of CX. The reference point of the path is the project folder. When this property is omitted, only the standard folder of CX is searched. This corresponds to the -I option of the cx command. The specified include path is displayed as the subproperty. When the include path is added to the project tree, the path is added to the top of the subproperties. Uppercase characters and lowercase characters are not distinguished for the include paths.	
	Default	Additional include paths[*number of defined items*]
	How to change	Edit by the Path Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 259 characters Up to 256 items can be specified.
System include paths	Change the specified order of the include paths which the system set during assembling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The system include path is searched with lower priority than the additional include path. The reference point of the path is the project folder. This corresponds to the -I option of the cx command. The include path is displayed as the subproperty.	
	Default	System include paths[*number of defined items*]
	How to change	Edit by the System Include Path Order dialog box which appears when clicking the [...] button.
	Restriction	Changes not allowed (Only the specified order of the include paths can be changed.)
Macro definition	Specify the name of the macro to be defined. Specify in the format of "*macro name=defined value*", with one macro name per line. The "=*defined value*" part can be omitted, and in this case, "1" is used as the defined value. This corresponds to the -D option of the cx command. The specified macro is displayed as the subproperty.	
	Default	Macro definition[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 256 characters Up to 256 items can be specified.

(5) [Frequently Used Options(for Link)]

The detailed information on frequently used options during linking is displayed and the configuration can be changed.

This category is not displayed for the library project.

Using libraries	Specify the library file (lib*xxx*.lib or lib*xxx*.a) to be used other than the standard library. lib*xxx*.lib is searched with higher priority. If it is not found, lib*xxx*.a is searched. Specify only the "*xxx*" part (example: if you specify "user", "libuser.lib" is assumed to be specified). Add one file in one line. The library files are searched from the library path. This corresponds to the -l option of the cx command. The specified library file is displayed as the subproperty.	
	Default	Using libraries[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 249 characters Up to 256 items can be specified.
Additional library paths	Specify the search folder of the library file to be used other than the standard libraries. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. If a relative path is specified, the reference point of the path is the project folder. This corresponds to the -L option of the cx command. The specified library path is displayed as the subproperty.	
	Default	Additional library paths[*number of defined items*]
	How to change	Edit by the Path Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 259 characters Up to 256 items can be specified.
Output folder	Specify the folder which the load module to be generated is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -o option of the cx command.	
	Default	%BuildModeName%
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.
	Restriction	Up to 247 characters

Output file name	Specify the file name of the load module to be generated. The extension other than ".lmf" cannot be specified. If the extension is omitted, ".lmf" is automatically added. If the target is a multi-core CPU, the load module for the common module and the load module for core *n* are generated, and the final load module is then generated based on those (*n*: number of cores of the target CPU). Load module for the common module: *input string without the extension* _cmn.lmf Load module for the core *n*: *input string without the extension* _pen.lmf The following macro name is available as an embedded macro. %ProjectName%: Replaces with the project name. This corresponds to the -o option of the cx command.	
	Default	%ProjectName%.lmf
	How to change	Directly enter in the text box.
	Restriction	Up to 259 characters

(6) [Frequently Used Options(for Hex Output)]

The detailed information on frequently used options during hex outputting is displayed and the configuration can be changed.

This category is not displayed for the library project.

Hex file format	Select the format of the hex file to be output. This corresponds to the -Xhex_format option of the cx command.		
	Default	Intel expanded hex format(None)	
	How to change	Select from the drop-down list.	
	Restriction	Intel expanded hex format(None)	Specifies the Intel expanded hex format as the format of the hex file.
		Intel expanded hex format (32-bit address)(-Xhex_format=i)	Specifies the Intel expanded hex format (32-bit address) as the format of the hex file.
		Motorola S type format(standard address)(-Xhex_format=S)	Specifies the Motorola S type format (standard address) as the format of the hex file.
		Motorola S type format(32-bit address)(-Xhex_format=s)	Specifies the Motorola S type format (32-bit address) as the format of the hex file.
		Expanded Tektronix hex format(-Xhex_format=T)	Specifies the expanded Tektronix hex format as the format of the hex file.

(7) [Register Mode]

The detailed information on register modes is displayed and the configuration can be changed.

Register mode	Select the register mode (number of registers used by the Compiler)[Note] of the software register bank function. This corresponds to the -Xreg_mode option of the cx command.		
	Default	32-register mode(None)	
	How to change	Select from the drop-down list.	
	Restriction	32-register mode(None)	Sets the register mode to 32.
		26-register mode(-Xreg_mode=26)	Sets the register mode to 26.
		22-register mode(-Xreg_mode=22)	Sets the register mode to 22.
		Universal register mode(-Xreg_mode=common)	Sets the register mode to 22. Use this item to generate the object module file that does not depend on the register mode.

Note The register modes provided by CX are shown below.

Register Mode	Working Registers	Registers for Register Variables
common	r10 to r14	r25 to r29
22-register mode	r10 to r14	r25 to r29
26-register mode	r10 to r16	r23 to r29
32-register mode	r10 to r19	r20 to r29

(8) [Flash Correspondence]

The detailed information on the flash correspondence is displayed and the configuration can be changed.

This category is not displayed for the library project.

Use boot-flash re-linking	Select whether to use the boot-flash re-linking function. This must be specified for both the flash area and the boot area.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Uses the boot-flash re-linking function.
		No	Does not use the boot-flash re-linking function.
Load module file type	Select the type of the load module file to be generated. This corresponds to the -Xflash option of the cx command. This property is displayed only when [Yes] in the [Use boot-flash re-linking] property is selected.		
	Default	Boot area load module file(None)	
	How to change	Select from the drop-down list.	
	Restriction	Boot area load module file(None)	Generates the load module file for the boot area.
		Flash area load module file(-Xflash)	Generates the load module file for the flash area. Specify the boot area load module file in the [Boot area load module file name] property.

Boot area load module file name	Specify the boot area load module file name. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). Be sure to specify this property when [Flash area load module file(-Xflash)] in the [Load module file type] property is selected. This corresponds to the -Xflash option of the cx command. This property is displayed only when [Flash area load module file(-Xflash)] in the [Load module file type] property is selected.	
	Default	Blank
	How to change	Directly enter in the text box or edit by the Specify Boot Area Load Module File dialog box which appears when clicking the [...] button.
	Restriction	Up to 259 characters
Branch table address	Specify the start address of the branch table. Specify the same address for both the flash area and the boot area. If an odd number is specified, it is revised to an even number. This corresponds to the -Xflash_ext_table option of the cx command. This property is displayed only when [Yes] in the [Use boot-flash re-linking] property is selected.	
	Default	0 (hexadecimal number)
	How to change	Directly enter in the text box.
	Restriction	0 to FFFFFFFF (hexadecimal number)

(9) [Error Output]

The detailed information on the error output is displayed and the configuration can be changed.

Output error message file	Select whether to output the error message file. This corresponds to the -Xerror_file option of the cx command. Error messages are displayed on the Output panel regardless of this property's .		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xerror_file)	Outputs the error message file.
		No	Does not output the error message file.

Error message file output folder	Specify the folder which the error message file is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xerror_file option of the cx command. This property is displayed only when [Yes(-Xerror_file)] in the [Output error message file] property is selected.	
	Default	%BuildModeName%
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.
	Restriction	Up to 247 characters
Error message file name	Specify the error message file name. The extension can be freely specified. The following macro name is available as an embedded macro. %ProjectName%: Replaces with the project name. If this is blank, it is assumed that "%ProjectName%.err" has been specified. This corresponds to the -Xerror_file option of the cx command. This property is displayed only when [Yes(-Xerror_file)] in the [Output error message file] property is selected.	
	Default	%ProjectName%.err
	How to change	Directly enter in the text box.
	Restriction	Up to 259 characters

(10) [Warning Message]

The detailed information on warning messages is displayed and the configuration can be changed.

Displayed warning message	Specify the warning message number to be displayed regardless of the warning level. If multiple message numbers are specified, delimit them with "," (comma) (example: 02042,02107). Also, the range can be set using "-" (hyphen) (example: 02222-02554,02699-02782). If the same number is specified in the [Undisplayed warning message] property and this property, the number specified in this property takes precedence. This corresponds to the -Xwarning option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 2048 characters

Undisplayed warning message	Specify the warning message number to not be displayed regardless of the warning level. If multiple message numbers are specified, delimit them with "," (comma) (example: 02042,02107). Also, the range can be set using "-" (hyphen) (example: 02222-02554,02699-02782). If the same number is specified in the [Displayed warning message] property and this property, the number specified in the [Displayed warning message] property takes precedence. This corresponds to the -Xno_warning option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 2048 characters

(11) [Build Method]

The detailed information on the build method is displayed and the configuration can be changed.

Build simultaneously	Select whether to geneate the load module file by compiling/assembling/linking multiple files simultaneously. The files with the individual build options and files to be executed prior to the build are excluded from runing a build simultaneously. See "2.16.5 Compile/assemble/link multiple files simultaneously" for details about runing a build simultaneously.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Compiles, assembles, and links multiple files simultaneously. The assembler source file (except the file with the individual build options) is assembled using the settings of the [Compile Options] tab.
		No	Compiles, assembles, and links for each file.
Handling the source file includes non-existing file	Selects whether to recompile/assemble the source file if there are no files that include it.		
	Default	Re-compile/assemble the source file	
	How to change	Select from the drop-down list.	
	Restriction	Re-compile/assemble the source file	Recompiles/assembles the source file if there are no files that include it.
		Ignore re-compiling/ assembling the source file	Does not recompile/assemble the source file even if there are no files that include it.

(12) [Version Select]

The detailed information on the build tool version is displayed and the configuration can be changed.

Using compiler package install folder	The folder in which the compiler package to be used is installed is displayed.	
	Default	*Install folder name*
	How to change	Changes not allowed

Using compiler package version	Select the version of the compiler package to be used. This setting is common to all the build modes.		
	Default	Always latest version which was installed	
	How to change	Select from the drop-down list.	
	Restriction	Always latest version which was installed	Uses the latest version in the installed compiler packages.
		Versions of the installed compiler packages	Uses the selected version in the compiler package.
Latest compiler package version which was installed	The version of the compiler package to be used when [Always latest version which was installed] is selected in the [Using compiler package version] property is displayed. This setting is common to all the build modes. This property is displayed only when [Always latest version which was installed] in the [Using compiler package version] property is selected.		
	Default	*The latest version of the installed compiler packages*	
	How to change	Changes not allowed	

(13) [Notes]

The detailed information on notes is displayed and the configuration can be changed.

Memo	Add memos to the build tool. Add one item in one line. This setting is common to all the build modes. The specified memo is displayed as the subproperty.	
	Default	Memo[*number-of-items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 256 characters Up to 256 items can be specified.

(14) [Others]

Other detailed information on the build tool is displayed and the configuration can be changed.

Output message format	Specify the format of the message under build execution. This applies to the messages output by the build tool to be used, and commands added by plugins. It does not apply to the output messages of commands specified in the [Commands executed before build processing] or [Commands executed after build processing] property. The following macro names are available as embedded macros. %Program%: Replaces with the program name under execution. %Options%: Replaces with the command line option under build execution. %FileName%: Replaces with the file name being compile/assemble or making link. If this is blank, "%Program% %Options%" will be set automatically.		
	Default	%FileName%	
	How to change	Directly enter in the text box (up to 256 characters) or select from the drop-down list.	
	Restriction	%FileName%	Displays the file name in the output message.
		%FileName%: %Options%	Displays the file name and command line options in the output message.
		%Program% %Options%	Displays the program name and command line options in the output message.
Format of build option list	Specify the display format of the build option list (see "2.15.3 Display a list of build options"). This applies to the options of the build tool to be used, and commands added by plugins. It does not apply to the options of commands specified in the [Commands executed before build processing] or [Commands executed after build processing] property. The following macro names are available as embedded macros. %Program%: Replaces with the program name under execution. %Options%: Replaces with the command line option under build execution. %FileName%: Replaces with the file name being compile/assemble or making link. If this is blank, "%FileName% : %Program% %Options%" will be set automatically.		
	Default	%FileName% : %Program% %Options%	
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.	
	Restriction	Up to 256 characters	
Temporary folder	Specify the folder to which the temporary files generated by the cx command under execution are saved. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). If this is blank, the temporary folder is determined in the following sequence. (1) Folder specified by environmental variable TEMP (2) Folder specified by environmental variable TMP (3) Folder of the main project or subproject This corresponds to the -Xtemp_path option of the cx command.		
	Default	Blank	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 200 characters	

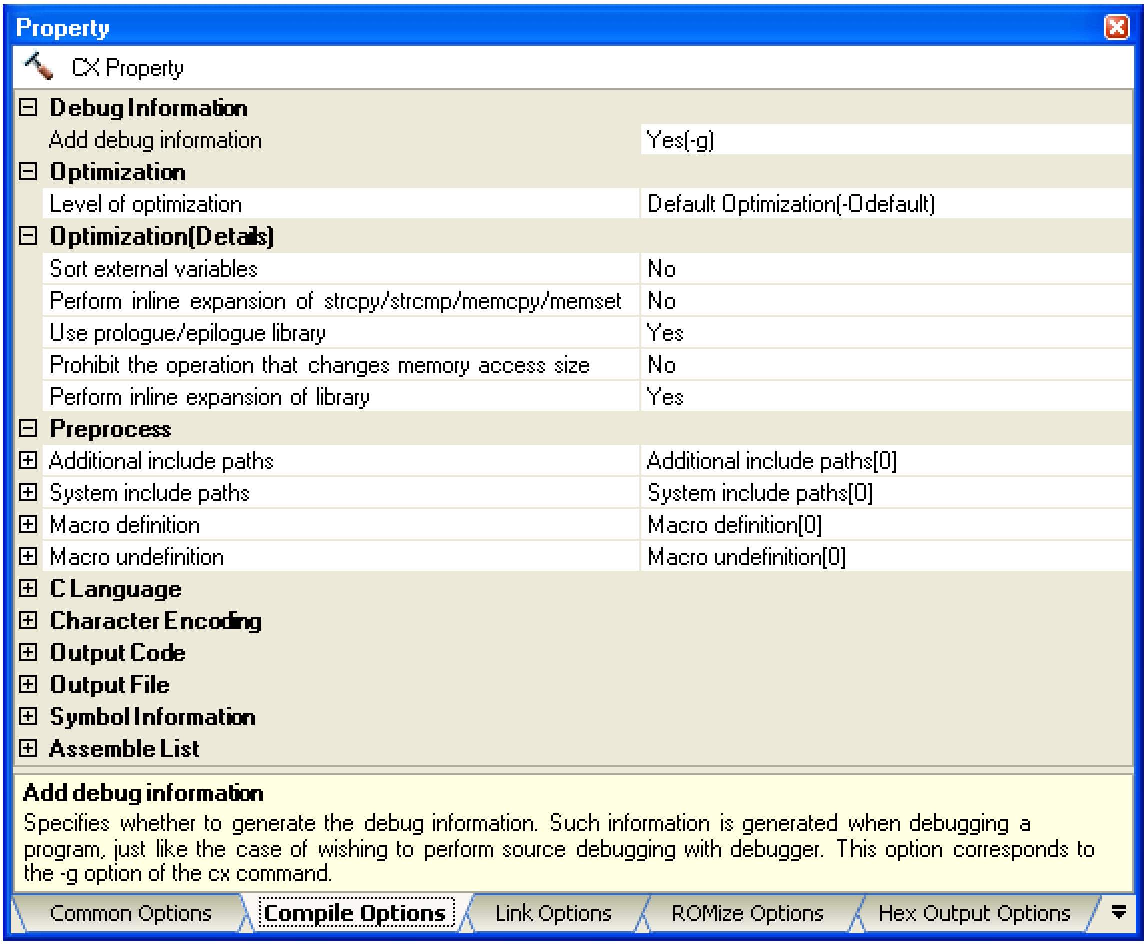
Commands executed before build processing	Specify the command to be executed before build processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. The specified command is displayed as the subproperty.	
	Default	Commands executed before build processing[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.
Commands executed after build processing	Specify the command to be executed after build processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. The specified command is displayed as the subproperty.	
	Default	Commands executed after build processing[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.
Other additional options	Input the option to be added additionally. The options set here are added at the end of the cx options group.	
	Default	Blank
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 259 characters

[Compile Options] tab

This tab shows the detailed information on the compile phase categorized by the following and the configuration can be changed.

(1) [Debug Information]
(2) [Optimization]
(3) [Optimization(Details)]
(4) [Preprocess]
(5) [C Language]
(6) [Character Encoding]
(7) [Output Code]
(8) [Output File]
(9) [Symbol Information]
(10) [Assemble List]
(11) [External Variable Register]
(12) [Others]

Figure A-5. Property Panel: [Compile Options] Tab



⊟ **Debug Information**	
Add debug information	Yes(-g)
⊟ **Optimization**	
Level of optimization	Default Optimization(-Odefault)
⊟ **Optimization(Details)**	
Sort external variables	No
Perform inline expansion of strcpy/strcmp/memcpy/memset	No
Use prologue/epilogue library	Yes
Prohibit the operation that changes memory access size	No
Perform inline expansion of library	Yes
⊟ **Preprocess**	
⊞ Additional include paths	Additional include paths[0]
⊞ System include paths	System include paths[0]
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]
⊞ **C Language**	
⊞ **Character Encoding**	
⊞ **Output Code**	
⊞ **Output File**	
⊞ **Symbol Information**	
⊞ **Assemble List**	

Add debug information
Specifies whether to generate the debug information. Such information is generated when debugging a program, just like the case of wishing to perform source debugging with debugger. This option corresponds to the -g option of the cx command.

[Description of each category]

(1) [Debug Information]

The detailed information on debug information is displayed and the configuration can be changed.

Add debug information	Select whether to generate the debug information. It is possible to perform source debugging with the debugger by outputting information for source debugging to the output file. This corresponds to the -g option of the cx command.		
	Default	Yes(-g)	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-g)	Generates the debug information.
		No	Does not generate the debug information.

(2) [Optimization]

The detailed information on the optimization is displayed and the configuration can be changed.

Level of optimization	Select the level of the optimization for compiling. This corresponds to the -O option of the cx command.		
	Default	Default Optimization(-Odefault)	
	How to change	Select from the drop-down list.	
	Restriction	Default Optimization(-Odefault)	Performs optimization that debugging is not affected (optimization of expressions and register allocation, and the like).
		Code Size Precedence(-Osize)	Performs optimization with the object size precedence. Regards reducing the ROM/RAM capacity as important and performs the maximum optimization that is effective for general programs.
		Speed Precedence(-Ospeed)	Performs optimization with the execution speed precedence. Regards shortening the execution speed as important and performs the maximum optimization that is effective for general programs.
		Debug Precedence(-Onothing)	Performs optimization with the debug precedence. Regards debugging as important and suppresses all optimization including default optimization.
		Detail Setting	The properties are added in the [Optimization(Details)] category. The optimization items selected in the category has the precedence for the optimization.

(3) [Optimization(Details)]

The detailed information on the optimization is displayed and the configuration can be changed.

Maximum number of loop expansions	Specify the maximum number of times to expand the loops such as "for" and "while". If 0 or 1 is specified, expansion is suppressed. If this is blank, it is assumed that "4" has been specified. This corresponds to the -O unroll option of the cx command. This property is displayed only when [Detail Setting] in the [Level of optimization] property is selected.		
	Default	0	
	How to change	Directly enter in the text box.	
	Restriction	0 to 999 (decimal number) or blank	
Remove unused static functions	Select whether to remove the static functions which are not called. This corresponds to the -Odelete_static_func option of the cx command. This property is displayed only when [Detail Setting] in the [Level of optimization] property is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Odelete_static_func=on)	Removes the unused static functions which are not called.
		No	Does not remove the unused static functions which are not called.
Perform inline expansion	Specify whether to perform inline expansion at the location calling functions. This corresponds to the -Oinline option of the cx command. This property is displayed only when [Detail Setting] in the [Level of optimization] property is selected.		
	Default	Yes(Only Specified Functions)(None)	
	How to change	Select from the drop-down list.	
	Restriction	Yes(Only Specified Functions)(None)	Performs inline expansion at the location calling the function for which #pragma inline is specified.
		Yes(Auto-detect)(-Oinline=2)	Distinguishes the function that is the target of inline expansion automatically and expands it.
		No(-Oinline=0)	Does not specify inline expansion of the function.
Perform pipeline optimization	Select whether to improve the program's execution performance by reordering instructions at the machine-language level. This corresponds to the -Opipeline option of the cx command. This property is displayed only when the project uses a device with the V850E2V3 architecture and when [Detail Setting] in the [Level of optimization] property in the [Optimization] category is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Opipeline=on)	Performs pipeline optimization.
		No	Does not specify pipeline optimization.

Sort external variables	Select whether to sort external variables. This property is for reducing the RAM capacity. This corresponds to the -Xsort_var option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xsort_var)	Rearranges external variables allocated to a section other than const/sconst sequentially, starting from the largest alignment size.
		No	Does not specify sorting external variables.
Perform inline expansion of strcpy/strcmp/ memcpy/memset	Selsect whether to perform inline expansion of functions "strcpy()", "strcmp()", "memcpy()", and "memset()" calls, with regarding the alignment conditions of the array (including character strings) and the structure as 4 bytes. This improves the execution speed of the program to be generated, but it increases the code size. This corresponds to the -Xinline_strcpy option of the cx command. This property is displayed only when [No] in the [Structure packing] property in the [Output Code] category is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xinline_strcpy)	Performs inline expansion of functions "strcpy()", "strcmp()", "memcpy()", and "memset()" calls.
		No	Does not specify inline expansion of functions "strcpy()", "strcmp()", "memcpy()", and "memset()" calls.
Use prologue/epilogue library	Specify whether to perform prologue/epilogue processing of functions through runtime library calls. This corresponds to the -Xpro_epi_runtime option of the cx command. This property is displayed only when [Speed Precedence(-Ospeed)] in the [Level of optimization] property in the [Optimization] category is selected.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Performs prologue/epilogue processing of the function through runtime library calls.
		No(-Xpro_epi_runtime=off)	Does not perform prologue/epilogue processing of functions through runtime library calls.
Prohibit the operation that changes memory access size	Select whether to prohibit replacing the load and store instructions with 1-bit manipulation instructions (set1, clr1, tst1, and not1). This corresponds to the -Xkeep_access_size option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xkeep_access_size)	Prohibits replacing the load and store instructions with 1-bit manipulation instructions.
		No	1-bit manipulation instructions may be generated instead of load and store instructions.

Perform inline expansion of library	Specify whether to perform inline expansion at the location calling library functions. This corresponds to the -Xcall_lib option of the cx command. This property is displayed only for the project using a device with the V850E2V3 architecture.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Generates a sqrtf.s instruction instead of a call to the sqrtf function.
		No(-Xcall_lib)	Always generates a call to the sqrtf function.

(4) [Preprocess]

The detailed information on preprocessing is displayed and the configuration can be changed.

Additional include paths	Specify the additional include paths during compiling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The specified include path is searched with higher priority than the standard include file folder of CX. The reference point of the path is the project folder. When this property is omitted, only the standard folder of CX is searched. This corresponds to the -I option of the cx command. The specified include path is displayed as the subproperty. When the include path is added to the project tree, the path is added to the top of the subproperties. Uppercase characters and lowercase characters are not distinguished for the include paths.	
	Default	Additional include paths[*number of defined items*]
	How to change	Edit by the Path Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 259 characters Up to 256 items can be specified.
System include paths	Change the specified order of the include paths which the system set during compiling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The system include path is searched with lower priority than the additional include path. The reference point of the path is the project folder. This corresponds to the -I option of the cx command. The include path is displayed as the subproperty.	
	Default	System include paths[*number of defined items*]
	How to change	Edit by the System Include Path Order dialog box which appears when clicking the [...] button.
	Restriction	Changes not allowed (Only the specified order of the include paths can be changed.)

Macro definition	Specify the name of the macro to be defined. Specify in the format of "*macro name=defined value*", with one macro name per line. The "=*defined value*" part can be omitted, and in this case, "1" is used as the defined value. This corresponds to the -D option of the cx command. The specified macro is displayed as the subproperty.		
	Default	Macro definition[*number of defined items*]	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 256 characters Up to 256 items can be specified.	
Macro undefinition	Specify the macro name to be undefined. Specify in the format of "*macro name*", with one macro name per line. This corresponds to the -U option of the cx command. The specified macro is displayed as the subproperty.		
	Default	Macro undefinition[*number of defined items*]	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 256 characters Up to 256 items can be specified.	
Output C source comments to preprocessed file	Select whether to output the comments of the C source to the preprocessed file. This corresponds to the -Xpreprocess option of the cx command. This property is displayed only when [Yes(-P)] in the [Output preprocessed source file] property in the [Output File] category is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xpreprocess=comment)	Outputs the comments of the C source to the preprocessed file.
		No	Does not output the comments of the C source to the preprocessed file.
Output line number information to preprocessed file	Select whether to output the line number information of the C source to the preprocessed file. This corresponds to the -Xpreprocess option of the cx command. This property is displayed only when [Yes(-P)] in the [Output preprocessed source file] property in the [Output File] category is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xpreprocess= line)	Outputs the line number information of the C source to the preprocessed file.
		No	Does not output the line number information of the C source to the preprocessed file.

(5) [C Language]

The detailed information on C language is displayed and the configuration can be changed.

Compile strictly according to ANSI standards	Select whether to process as making C source program comply strictly with the ANSI standard and output an error or warning for a specification that violates the standard. This corresponds to the -Xansi option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xansi)	Processes as making C source program comply strictly with the ANSI standard and outputs an error or warning for a specification that violates the standard.
		No	Compatibility with the conventional C language specifications is conferred and processing continues after warning is output.
Sign of char	Select whether char type without a type specifier (signed or unsigned) are handled as signed or unsigned. This corresponds to the -Xchar option of the cx command.		
	Default	signed	
	How to change	Select from the drop-down list.	
	Restriction	signed	Handles char type without a type specifier as signed.
		unsigned(-Xchar=unsigned)	Handles char type without a type specifier as unsigned.
Enumeration type	Select which integer type the enumeration type handles. This corresponds to the -Xenum_type option of the cx command.		
	Default	signed int(None)	
	How to change	Select from the drop-down list.	
	Restriction	signed int(None)	The enumeration type is handled as int type.
		auto(-Xenum_type=auto)	Treats each enumerated type as the smallest integer type capable of expressing all the enumerators in that type.
Treat tentative definition as definition	Select whether to treat tentative definitions of variables as definitions. This corresponds to the -Xdef_var option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xdef_var)	Handles tentative definition of variables as definition.
		No	Does not handle tentative definition of variables as definition.

(6) [Character Encoding]

The detailed information on character encoding is displayed and the configuration can be changed.

Character encoding	Select the character code to be used for Japanese/Chinese comments and character strings in the source file. This corresponds to the -Xcharacter_set option of the cx command.		
	Default	SJIS(None)	
	How to change	Select from the drop-down list.	
	Restriction	SJIS(None)	Interprets the Japanese character code in the source file as SJIS.
		EUC(-Xcharacter_set=euc_jp)	Interprets the Japanese character code in the source file as EUC.
		UFT-8(-Xcharacter_set=utf8)	Interprets the Japanese character code in the source file as UFT-8.
		Big5(-Xcharacter_set=big5)	Interprets the Chinese character code in the source file as Traditional Chinese.
		GB2312(-Xcharacter_set=gb2312)	Interprets the Chinese character code in the source file as Simplified Chinese.
		No-process(-Xcharacter_set=none)	Does not process the Japanese character code in the source file.

(7) [Output Code]

The detailed information on output codes is displayed and the configuration can be changed.

Structure packing	Select whether to perform structure packing. The specified alignment can be used without aligning structure members according to the type of each member. This corresponds to the -Xpack option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	1 byte(-Xpack=1)	Aligns structure members on a 1-byte boundary.
		2 bytes(-Xpack=2)	Aligns structure members on a 2-byte boundary.
		4 bytes(-Xpack=4)	Aligns structure members on a 4-byte boundary.
		8 bytes(-Xpack=8)	Aligns structure members on a 8-byte boundary.
		No	Does not perform structure packing.
Output comment to assembler source file	Select whether to output a C source program as a comment to the assembler source file to be output. This corresponds to the -Xpass_source option of the cx command. This property is displayed only when [Yes(-Xasm_path)] in the [Output assembler source file] property in the [Output File] category or [Yes(-Xprn_path)] in the [Output assemble list file] property is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xpass_source)	Outputs a C source program as a comment to the assembler source file.
		No	Does not output a C source program as a comment to the assembler source file.

Output code of switch statement	Select the code output mode for switch statements in programs. This corresponds to the -Xswitch option of the cx command.		
	Default	Auto(None)	
	How to change	Select from the drop-down list.	
	Restriction	Auto(None)	The cx selects the optimum output format.
		if-else(-Xswitch=ifelse)	Outputs the switch statements in the same format as the if-else statement along a string of case statements in programs. Select this item if the case statements are written in the order of frequency or if only a few labels are used. Because the case statements are compared starting from the top, unnecessary comparison can be reduced and the execution speed can be increased if the case statement that most often matches is written first.
		Binary search(-Xswitch=binary)	Outputs the code in the binary search format for switch statements in programs. Searches for a matching case statement by using a binary search algorithm. If this item is selected when many labels are used, any case statement can be found at almost the same speed.
		Table jump(-Xswitch=table)	Outputs the code in the table jump format for switch statements in programs. References a table indexed on the values in the case statements, and selects and processes case labels from the switch statement values. The code will branch to all the case statements with about the same speed. However, if case values are not used in succession, an unnecessary area will be created.
Label size of switch table	Select the size per label of the branch table for the case labels in switch statements. This corresponds to the -Xword_case option of the cx command. This property is displayed only when [Auto(None)] or [Table jump(-Xswitch=table)] in the [Output code of switch statement] property is selected. If [Auto(None)] is selected in the [Output code of switch statement] property, and the cx command automatically selects [if-else(-Xswitch=ifelse)] or [Binary search(-Xswitch=binary)], the setting of this property will be disabled (set to 2 bytes).		
	Default	2 bytes(None)	
	How to change	Select from the drop-down list.	
	Restriction	2 bytes(None)	Generates one 2-byte branch table per case label in a switch statement.
		4 bytes(-Xword_case)	Generates one 4-byte branch table per case label in a switch statement. Select this item when a compile error occurs because the switch statement is long.

Size threshold of sdata/ sbss section allocation(Bytes)	Specify the upper limit size of the data length allocated to the .sdata/.sbss sections. Data specified the .sdata or .sbss section by the "#pragma section" directive is allocated to that section regardless of the size. This corresponds to the -Xsdata option of the cx command.		
	Default	Blank	
	How to change	Directly enter in the text box.	
	Restriction	0 to 65535 (decimal number)	
Size threshold of sconst section allocation(Bytes)	Specify the upper limit size (bytes) for allocating const attribute data and character string literals to the const section. This corresponds to the -Xsconst option of the cx command.		
	Default	32767	
	How to change	Directly enter in the text box.	
	Restriction	0 to 32767 (decimal number)	
Floating-point calculating type	Select whether to generate runtime library call instructions for floating-point calculations, or to generate floating-point instructions for the floating point unit (FPU). This corresponds to the -Xfloat [V850E2V3] option of the cx command. This property is displayed only for the project using a device with the V850E2V3 architecture having an FPU.		
	Default	Auto(None)	
	How to change	Select from the drop-down list.	
	Restriction	Auto(None)	If the target device does not have FPU, floating-point calculation instructions are not generated and runtime library call instructions are generated. If the target device has FPU, floating-point calculation instructions are generated.
		Software Calculating(-Xfloat=soft)	Generates runtime library call instructions for floating-point calculations.
		FPU Calculating(-Xfloat=fpu)	Generates floating-point calculation instructions of FPU (floating-point unit) for floating-point calculations.
Generate div/divu instructions	Select whether to generate the div and divu instructions instead of the divq and divqu instructions for division. Although the divq and divqu instructions are fast, the number of execution cycles will differ depending on the values of the operands. This corresponds to the -Xdiv [V850E2V3] option of the cx command. This property is displayed only for the project using a device with the V850E2V3 architecture having an FPU.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xdiv)	Generates the div and divu instructions for division.
		No	Generates the divq and divqu instructions for division.

Use 32-bit branch instruction	Select whether to use the far jump function for the jarl and jr instructions. By using the far jump function, it is assumed that the jarl and jr instructions are jarl32 and jr32 instructions, and assembling is performed. This corresponds to the -Xasm_far_jump [V850E2][V850E2V3] option of the cx command. This property is displayed only when the project uses a device with the V850E2 core and V850E2V3 architecture and when [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xasm_far_jump)	Assumes that the jarl and jr instructions are jarl32 and jr32 instructions, and performs assembling.
		No	Performs assembly as a jarl or jr instruction.
Far Jump file names	Specify the Far Jump file name. The Far Jump file outputs the code that uses the jmp instruction (V850E/ES core) or jarl32 and jr32 instruction (V850E2 core or V850E2V3 architecture) for branch instructions of functions described in the file. The cx command outputs an error if the function is in a range that cannot be branched to by the jarl or jr directive (±2MB or more), in which case the Far Jump file is used to recompile. Use the extension ".fjp". This corresponds to the -Xfar_jump option of the cx command. The specified Far Jump file name is displayed as the subproperty.		
	Default	Far Jump file names[*number of set items*]	
	How to change	Click the [...] button to open the Path Edit dialog box. -> Edit by the Specify Far Jump File dialog box which appears when clicking the [Browse...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 259 characters Up to 5000 items can be specified. However, only the file name specified last is valid.	

(8) [Output File]

The detailed information on output files is displayed and the configuration can be changed.

Output assembler source file	Select whether to output the assembler source file of the compile result for the C source. This corresponds to the -Xasm_path option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xasm_path)	Outputs the assembler source file of the compile result for the C source.
		No	Does not output the assembler source file of the compile result for the C source.

Output folder for assembler source file	Specify the folder which the assembler source file is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. The assembler source file is saved under the C source file name with the extension replaced by ".asm". If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xasm_path option of the cx command. This property is displayed only when [Yes(-Xasm_path)] in the [Output assembler source file] property is selected.		
	Default	%BuildModeName%	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 247 characters	
Output preprocessed source file	Select whether to output the execution result of preprocessing for the source file to a file. The file is output to the folder specified in the [Intermediate file output folder] property. The file is output under the source file name with the extension replaced by ".i". This corresponds to the -P option of the cx command. This property is displayed only when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-P)	Outputs the execution result of preprocessing for the source file to a file.
		No	Does not output the execution result of preprocessing for the source file to a file.

(9) [Symbol Information]

The detailed information on the symbol information is displayed and the configuration can be changed.

Using symbol information file	Select the name of the symbol information file to be used during a build. The effective symbol information file to be added to the project is searched and used. This corresponds to the -Xsymbol_file option of the cx command.	
	Default	*The name of the symbol information file that is added to the project*
	How to change	Changes not allowed

(10) [Assemble List]

The detailed information on the assemble list is displayed and the configuration can be changed.

Output assemble list file	Select whether to output the assemble list file. This corresponds to the -Xprn_path option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xprn_path)	Outputs the assemble list file.
		No	Does not output the assemble list file.
Output folder for assemble list file	Specify the folder which the assemble list file is output. The assemble list file is output under the source file name with the extension replaced by ".prn". If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xprn_path option of the cx command. This property is displayed only when [Yes(-Xprn_path)] in the [Output assemble list file] property is selected.		
	Default	%BuildModeName%	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 247 characters	

(11) [External Variable Register]

The detailed information on the external variable registers is displayed and the configuration can be changed. This category is not displayed when [32-register mode(None)] or [Universal register mode(-Xreg_mode=common)] in the [Register mode] property in the [Register Mode] category from the [Common Options] tab is selected.

External variable assigned to the r15 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr15 option of the cx command. This property is not displayed when [26-register mode(-Xreg_mode=26)] in the [Register mode] property in the [Register Mode] category from the [Common Options] tab is selected.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters
External variable assigned to the r16 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr16 option of the cx command. This property is not displayed when [26-register mode(-Xreg_mode=26)] in the [Register mode] property in the [Register Mode] category from the [Common Options] tab is selected.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters

External variable assigned to the r17 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr17 option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters
External variable assigned to the r18 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr18 option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters
External variable assigned to the r19 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr19 option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters
External variable assigned to the r20 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr20 option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters
External variable assigned to the r21 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr21 option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters
External variable assigned to the r22 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr22 option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters
External variable assigned to the r23 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr23 option of the cx command. This property is not displayed when [26-register mode(-Xreg_mode=26)] in the [Register mode] property in the [Register Mode] category from the [Common Options] tab is selected.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters

External variable assigned to the r24 register	Specify the external variables (symbol name excluding "_") assigned to the register. This corresponds to the -Xr24 option of the cx command. This property is not displayed when [26-register mode(-Xreg_mode=26)] in the [Register mode] property in the [Register Mode] category from the [Common Options] tab is selected.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 1022 characters

(12) [Others]

Other detailed information on compilation is displayed and the configuration can be changed.

This category is displayed only when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.

Commands executed before compile processing	Specify the command to be executed before compile processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %InputFile%: Replaces with the absolute path of the file to be compiled. %CompiledFile%: Replaces with the absolute path of the output file under compiling. The specified command is displayed as the subproperty.	
	Default	Commands executed before compile processing[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.
Commands executed after compile processing	Specify the command to be executed after compile processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %InputFile%: Replaces with the absolute path of the file to be compiled. %CompiledFile%: Replaces with the absolute path of the output file under compiling. The specified command is displayed as the subproperty.	
	Default	Commands executed after compile processing[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.

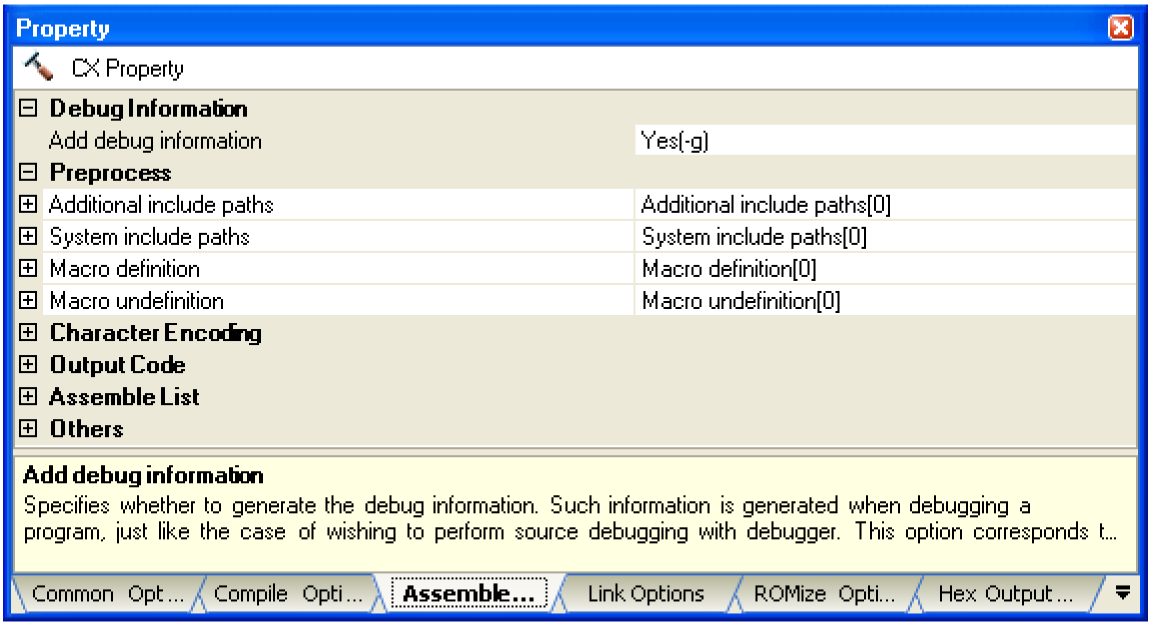
Other additional options	Input the compile option to be added additionally. The options set here are added at the end of the compile options group.	
	Default	Blank
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 259 characters

[Assemble Options] tab

This tab shows the detailed information on the assemble phase categorized by the following and the configuration can be changed.

(1) [Debug Information]
(2) [Preprocess]
(3) [Character Encoding]
(4) [Output Code]
(5) [Assemble List]
(6) [Others]

Caution This tab is displayed only when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.



Property

CX Property

⊟ Debug Information
Add debug information: Yes(-g)
⊟ Preprocess
⊞ Additional include paths: Additional include paths[0]
⊞ System include paths: System include paths[0]
⊞ Macro definition: Macro definition[0]
⊞ Macro undefinition: Macro undefinition[0]
⊞ Character Encoding
⊞ Output Code
⊞ Assemble List
⊞ Others

Add debug information
Specifies whether to generate the debug information. Such information is generated when debugging a program, just like the case of wishing to perform source debugging with debugger. This option corresponds t...

Figure A-6. Property Panel: [Assemble Options] Tab

[Description of each category]

(1) [Debug Information]

The detailed information on debug information is displayed and the configuration can be changed.

Add debug information	Select whether to generate the debug information. It is possible to perform source debugging with the debugger by outputting information for source debugging to the output file. This corresponds to the -g option of the cx command.		
	Default	Yes(-g)	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-g)	Generates the debug information.
		No	Does not generate the debug information.

(2) [Preprocess]

The detailed information on preprocessing is displayed and the configuration can be changed.

Additional include paths	Specify the additional include paths during assembling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The specified include path is searched with higher priority than the standard include file folder of CX. The reference point of the path is the project folder. When this property is omitted, only the standard folder of CX is searched. This corresponds to the -I option of the cx command. The specified include path is displayed as the subproperty. When the include path is added to the project tree, the path is added to the top of the subproperties. Uppercase characters and lowercase characters are not distinguished for the include paths.	
	Default	Additional include paths[*number of defined items*]
	How to change	Edit by the Path Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 259 characters Up to 256 items can be specified.
System include paths	Change the specified order of the include paths which the system set during assembling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The system include path is searched with lower priority than the additional include path. The reference point of the path is the project folder. This corresponds to the -I option of the cx command. The include path is displayed as the subproperty.	
	Default	System include paths[*number of defined items*]
	How to change	Edit by the System Include Path Order dialog box which appears when clicking the [...] button.
	Restriction	Changes not allowed (Only the specified order of the include paths can be changed.)
Macro definition	Specify the name of the macro to be defined. Specify in the format of "*macro name=defined value*", with one macro name per line. The "=*defined value*" part can be omitted, and in this case, "1" is used as the defined value. This corresponds to the -D option of the cx command. The specified macro is displayed as the subproperty.	
	Default	Macro definition[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 256 characters Up to 256 items can be specified.

Macro undefinition	Specify the macro name to be undefined. Specify in the format of "*macro name*", with one macro name per line. This corresponds to the -U option of the cx command. The specified macro is displayed as the subproperty.	
	Default	Macro undefinition[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 256 characters Up to 256 items can be specified.

(3) [Character Encoding]

The detailed information on character encoding is displayed and the configuration can be changed.

Character encoding	Select the character code to be used for Japanese comments and character strings in the source file. This corresponds to the -Xcharacter_set option of the cx command.		
	Default	SJIS(None)	
	How to change	Select from the drop-down list.	
	Restriction	SJIS(None)	Interprets the Japanese character code in the source file as SJIS.
		EUC(-Xcharacter_set=euc_jp)	Interprets the Japanese character code in the source file as EUC.
		UFT-8(-Xcharacter_set=utf8)	Interprets the Japanese character code in the source file as UFT-8.
		Big5(-Xcharacter_set=big5)	Interprets the Chinese character code in the source file as Traditional Chinese.
		GB2312(-Xcharacter_set=gb2312)	Interprets the Chinese character code in the source file as Simplified Chinese.
		No-process(-Xcharacter_set=none)	Does not process the Japanese character code in the source file.

(4) [Output Code]

The detailed information on output codes is displayed and the configuration can be changed.

Size threshold of sdata/sbss section allocation(Bytes)	Specify the upper limit size of the data length allocated to the .sdata/.sbss sections. Data specified the .sdata or .sbss section by the "#pragma section" directive is allocated to that section regardless of the size. This corresponds to the -Xsdata option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0 to 65535 (decimal number)

Use 32-bit branch instruction	Select whether to use the far jump function for the jarl and jr instructions. By using the far jump function, it is assumed that the jarl and jr instructions are jarl32 and jr32 instructions, and assembling is performed. This corresponds to the -Xasm_far_jump [V850E2][V850E2V3] option of the cx command. This property is displayed only for the project using a device with the V850E2 core and V850E2V3 architecture.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xasm_far_jump)	Assumes that the jarl and jr instructions are jarl32 and jr32 instructions, and performs assembling.
		No	Performs assembly as a jarl or jr instruction.

(5) [Assemble List]

The detailed information on the assemble list is displayed and the configuration can be changed.

Output assemble list file	Select whether to output the assemble list file. This corresponds to the -Xprn_path option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xprn_path)	Outputs the assemble list file.
		No	Does not output the assemble list file.
Output folder for assemble list file	Specify the folder which the assemble list file is output. The assemble list file is output under the source file name with the extension replaced by ".prn". If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xprn_path option of the cx command. This property is displayed only when [Yes(-Xprn_path)] in the [Output assemble list file] property is selected.		
	Default	%BuildModeName%	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 247 characters	

(6) [Others]

Other detailed information on assembly is displayed and the configuration can be changed.

This category is displayed only when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.

Commands executed before assemble processing	Specify the command to be executed before assemble processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %InputFile%: Replaces with the absolute path of the file to be assembled. %AssembledFile%: Replaces with the absolute path of the output file under assembling. The specified command is displayed as the subproperty.	
	Default	Commands executed before assemble processing[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.
Commands executed after assemble processing	Specify the command to be executed after assemble processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %InputFile%: Replaces with the absolute path of the file to be assembled. %AssembledFile%: Replaces with the absolute path of the output file under assembling. The specified command is displayed as the subproperty.	
	Default	Commands executed after assemble processing[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.
Other additional options	Input the assemble option to be added additionally. The options set here are added at the end of the assemble options group.	
	Default	Blank
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 259 characters

[Link Options] tab

This tab shows the detailed information on the link phase categorized by the following and the configuration can be changed.

(1) [Debug Information]
(2) [Input File]
(3) [Output File]
(4) [Library]
(5) [Link Map]
(6) [Symbol Information]
(7) [Others]

Caution This tab is not displayed for the library project.

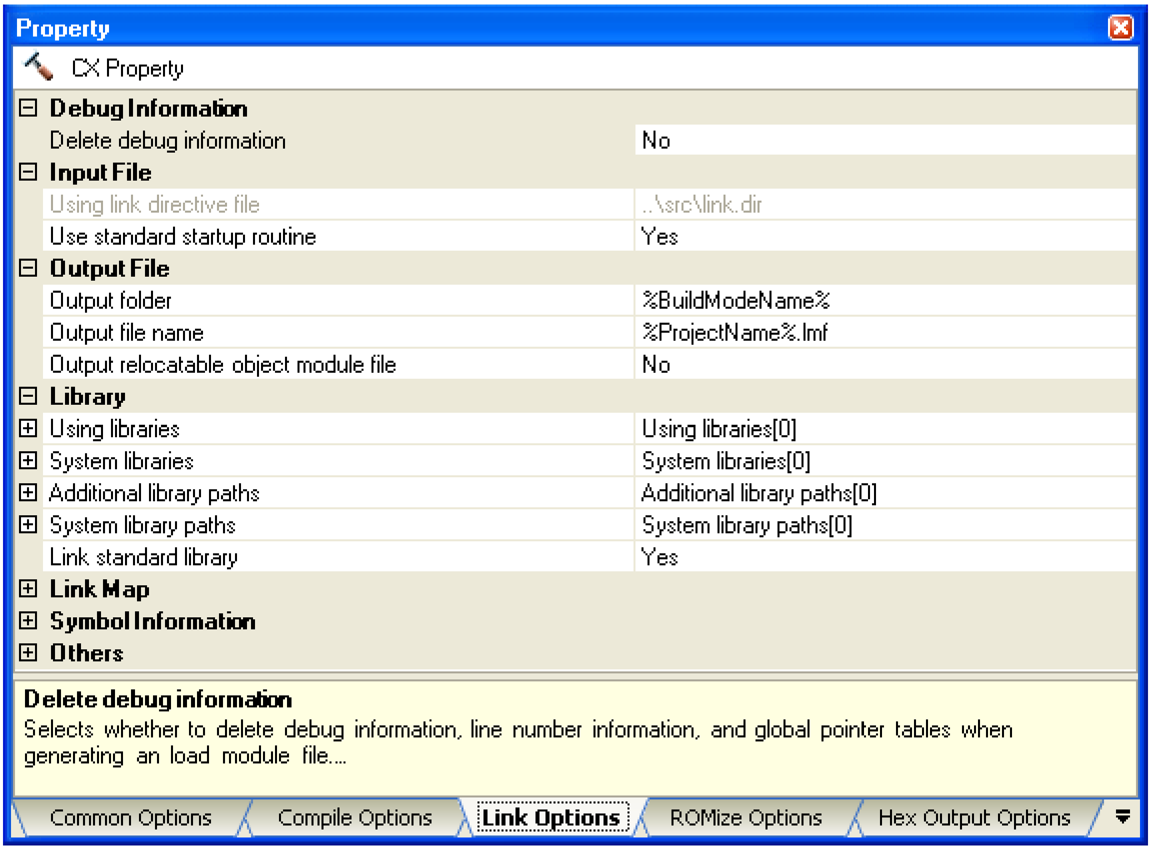


Figure A-7. Property Panel: [Link Options] Tab

⊟ **Debug Information**	
Delete debug information	No
⊟ **Input File**	
Using link directive file	..\src\link.dir
Use standard startup routine	Yes
⊟ **Output File**	
Output folder	%BuildModeName%
Output file name	%ProjectName%.lmf
Output relocatable object module file	No
⊟ **Library**	
⊞ Using libraries	Using libraries[0]
⊞ System libraries	System libraries[0]
⊞ Additional library paths	Additional library paths[0]
⊞ System library paths	System library paths[0]
Link standard library	Yes
⊞ **Link Map**	
⊞ **Symbol Information**	
⊞ **Others**	

Delete debug information
Selects whether to delete debug information, line number information, and global pointer tables when generating an load module file....

[Description of each category]

(1) [Debug Information]

The detailed information on debug information is displayed and the configuration can be changed.

Delete debug information	Select whether to delete debug information, line number information, and global pointer tables when generating the load module file. This corresponds to the -Xstrip option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xstrip)	Deletes debug information, line number information, and global pointer tables when generating the load module file.
		No	Does not delete debug information, line number information, and global pointer tables when generating the load module file.

(2) [Input File]

The detailed information on input files is displayed and the configuration can be changed.

Using link directive file	The name of the link directive file to be used for linking is displayed. An effective link directive file to be added to the project is searched and used. This corresponds to the -Xlink_directive option of the cx command.		
	Default	*The name of the link directive file that is added to the project*	
	How to change	Changes not allowed	
Use standard startup routine	Select whether to link the object module file (cstart.obj or cstartN.obj) provided with CX in which the standard startup routine is written. In the [Output ROMized load module file] property in the [Output File] category from the [ROMize Options] tab, "cstart.obj" is linked when [Yes] is selected and "cstartN.obj" is linked when [No(-Xno_romize)] is selected. This corresponds to the -Xno_startup option of the cx command. This property is not displayed when the build target file is added directly below the Startup node on the project tree or when any C source file is added under the File node.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Links the standard startup routine (cstart.obj or cstartN.obj).
		No(-Xno_startup)	Does not link the standard startup routine.

(3) [Output File]

The detailed information on output files is displayed and the configuration can be changed.

Output folder	Specify the folder which the load module to be generated is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -o option of the cx command.		
	Default	%BuildModeName%	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 247 characters	
Output file name	Specify the file name of the load module to be generated. The extension other than ".lmf" cannot be specified. If the extension is omitted, ".lmf" is automatically added. If the target is a multi-core CPU, the load module for the common module and the load module for core *n* are generated, and the final load module is then generated based on those (*n*: number of cores of the target CPU). Load module for the common module: *input string without the extension* _cmn.lmf Load module for the core *n*: *input string without the extension* _pe*n*.lmf The following macro name is available as an embedded macro. %ProjectName%: Replaces with the project name. This corresponds to the -o option of the cx command.		
	Default	%ProjectName%.lmf	
	How to change	Directly enter in the text box.	
	Restriction	Up to 259 characters	
Output relocatable object module file	Select whether to generate a relocatable object module file. This corresponds to the -Xrelinkable_object option of the cx command. This property is displayed only when [No] in the [Use boot-flash re-linking] property in the [Flash Correspondence] category from the [Common Options] tab is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xrelinkable_object)	Generates the relocatable object module file.
		No	Generates the executable object module file.

(4) [Library]

The detailed information on the library is displayed and the configuration can be changed.

Using libraries	Specify the library file (lib*xxx*.lib or lib*xxx*.a) to be used other than the standard library. lib*xxx*.lib is searched with higher priority. If it is not found, lib*xxx*.a is searched. Specify only the "*xxx*" part (example: if you specify "user", "libuser.lib" is assumed to be specified). Add one file in one line. The library file is searched from the library path. This corresponds to the -l option of the cx command. The specified library file is displayed as the subproperty.	
	Default	Using libraries[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 249 characters Up to 256 items can be specified.
System libraries	The name of the library file which the system uses is displayed. The system library file is searched with lower priority than the library file to be used. This corresponds to the -l option of the cx command. The library file name is displayed as the subproperty.	
	Default	System libraries[*number of defined items*]
	How to change	Changes not allowed
Additional library paths	Specify the search folder of the library file to be used other than the standard libraries. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. If a relative path is specified, the reference point of the path is the project folder. This corresponds to the -L option of the cx command. The specified library path name is displayed as the subproperty.	
	Default	Additional library paths[*number of defined items*]
	How to change	Edit by the Path Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 259 characters Up to 256 items can be specified.
System library paths	The folder to search the system library file is displayed. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. If a relative path is displayed, the reference point of the path is the project folder. This corresponds to the -L option of the cx command. The library path name is displayed as the subproperty.	
	Default	System library paths[*number of defined items*]
	How to change	Changes not allowed

Link standard library	Select whether to link the standard library. This corresponds to the -Xno_stdlib option of the cx command.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Links the standard library.
		No(-Xno_stdlib)	Does not link the standard library.

(5) [Link Map]

The detailed information on the link map is displayed and the configuration can be changed.

Output link map file	Select whether to output the link map file. This corresponds to the -Xmap option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xmap)	Outputs the link map file.
		No	Does not output the link map file.
Output folder for link map file	Specify the folder which the link map file is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xmap option of the cx command. This property is displayed only when [Yes(-Xmap)] in the [Output link map file] property is selected.		
	Default	%BuildModeName%	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 247 characters	
Link map file name	Specify the link map file name. The extension other than ".map" cannot be specified. If the extension is omitted, ".map" is automatically added. The following macro name is available as an embedded macro. %ProjectName%: Replaces with the project name. If this is blank, it is assumed that "%ProjectName%.map" has been specified. This corresponds to the -Xmap option of the cx command. This property is displayed only when [Yes(-Xmap)] in the [Output link map file] property is selected.		
	Default	%ProjectName%.map	
	How to change	Directly enter in the text box.	
	Restriction	Up to 259 characters	

(6) [Symbol Information]

The detailed information on the symbol information is displayed and the configuration can be changed.

Output symbol information file	Select whether to output the symbol information file during a build. This corresponds to the -Xsfg option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xsfg)	Outputs the symbol information file.
		No	Does not output the symbol information file.
Output folder for symbol information file	Specify the folder which the symbol information file is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xsfg option of the cx command. This property is displayed only when [Yes(-Xsfg)] in the [Output symbol information file] property is selected.		
	Default	%BuildModeName%	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 247 characters	
Symbol information file name	Specify the symbol information file name. The extension other than ".sfg" cannot be specified. If the extension is omitted, ".sfg" is automatically added. The following macro name is available as an embedded macro. %ProjectName%: Replaces with the project name. If this is blank, the file is output under the project file name with the extension replaced by ".sfg". This corresponds to the -Xsfg option of the cx command. This property is displayed only when [Yes(-Xsfg)] in the [Output symbol information file] property is selected.		
	Default	%ProjectName%.sfg	
	How to change	Directly enter in the text box.	
	Restriction	Up to 259 characters	

Output optimized allocation information	Select whether to sort variables in the order starting from highest use frequency for each section, determine whether they can be allocated to the .tidata.byte, .tidata.word, .sidata, .sedata, or .sdata section, and output the optimum allocation information to the symbol information file. This corresponds to the -Xsfg_opt option of the cx command. This property is displayed only when [Yes(-Xsfg)] in the [Output symbol information file] property is selected.		
	Default	Yes(-Xsfg_opt)	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xsfg_opt)	Sorts variables in the order starting from highest use frequency for each section and outputs the optimum allocation information to the symbol information file.
		No	Sorts variables in the order starting from highest use frequency and outputs the allocation information to the symbol information file.
Size of .tidata section	Specify the upper size variables allocated to the .tidata section. This corresponds to the -Xsfg_size_tidata option of the cx command. This property is displayed only when [Yes(-Xsfg_opt)] in the [Output optimized allocation information] property is selected.		
	Default	256 (decimal number)	
	How to change	Directly enter in the text box.	
	Restriction	0 to 256 (decimal number)	
Size of .tidata.byte section	Specify the upper size variables allocated to the .tidata.byte section. This corresponds to the -Xsfg_size_tidata_byte option of the cx command. This property is displayed only when [Yes(-Xsfg_opt)] in the [Output optimized allocation information] property is selected.		
	Default	128 (decimal number)	
	How to change	Directly enter in the text box.	
	Restriction	0 to 128 (decimal number)	
Size of .sidata section	Specify the upper size variables allocated to the .sidata section. This corresponds to the -Xsfg_size_sidata option of the cx command. This property is displayed only when [Yes(-Xsfg_opt)] in the [Output optimized allocation information] property is selected.		
	Default	32512 (decimal number)	
	How to change	Directly enter in the text box.	
	Restriction	0 to 32512 (decimal number)	
Size of .sedata section	Specify the upper size variables allocated to the .sedata section. This corresponds to the -Xsfg_size_sedata option of the cx command. This property is displayed only when [Yes(-Xsfg_opt)] in the [Output optimized allocation information] property is selected.		
	Default	32768 (decimal number)	
	How to change	Directly enter in the text box.	
	Restriction	0 to 32768 (decimal number)	

Size of .sdata section	Specify the upper size variables allocated to the .sdata section. This corresponds to the -Xsfg_size_sdata option of the cx command. This property is displayed only when [Yes(-Xsfg_opt)] in the [Output optimized allocation information] property is selected.		
	Default	65536 (decimal number)	
	How to change	Directly enter in the text box.	
	Restriction	0 to 65536 (decimal number)	

(7) [Others]

Other detailed information on linking is displayed and the configuration can be changed.

Entry symbol	Specify the symbol to be set as the entry point address of the load module file to be generated. If this is blank, the entry point address is determined in the following sequence: (1) Address of symbol "__start" (2) If "__start" does not exist, the start address of the text attribute section that is allocated to the lowest address area in the load module file to be generated. (3) Address 0 This corresponds to the -Xentry_address option of the cx command.		
	Default	Blank	
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.	
	Restriction	Up to 1022 characters	
Check register mode	Select whether to check all input object module files for mixing different register modes and display detailed information. This corresponds to the -Xregmode_info option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xregmode_info)	Checks for mixing different register modes and displays detailed information.
		No	Checks for mixing different register modes and does not display detailed information.
Force linking against error	Select whether to continue link processing when the memory overflows. Information about low memory is displayed on the Output panel. Even if forced linking is performed, the final file will not be generated. This corresponds to the -Xforce_link option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xforce_link)	Continues link processing when the memory overflows.
		No	Terminates link processing when the memory overflows.

Display GP information	Select whether to display the information used as a yardstick in the value setting in [Size threshold of sdata/sbss section allocation(Bytes)] property in the [Output Code] category from the [Compile Options] tab on the Output panel. This corresponds to the -Xsdata_info option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xsdata_info)	Displays the information used as a yardstick in the value setting in [Size threshold of sdata/sbss section allocation(Bytes)] property.
		No	Does not display the information used as a yardstick in the value setting in [Size threshold of sdata/sbss section allocation(Bytes)] property.
Link in 2-pass mode	Select whether to perform linking in the 2-pass mode. The 2-pass mode is slower than the 1-pass mode, but it is able to process larger sized files. The filling value for holes can also be specified. This corresponds to the -Xtwo_pass_link option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xtwo_pass_link)	Performs linking in the 2-pass mode.
		No	Performs linking in the 1-pass mode.
Filling value of holes	Specify the filling value for align holes between sections of the load module file to be generated. If this is blank, it is assumed that "0000" (hexadecimal number) has been specified. This corresponds to the -Xalign_fill option of the cx command. This property is displayed only when [Yes(-Xtwo_pass_link)] in the [Link in 2-pass mode] property is selected.		
	Default	0000 (hexadecimal number)	
	How to change	Directly enter in the text box.	
	Restriction	0000 to FFFF (hexadecimal number of 4 digits or fewer)	
Ignore illegal relocation	Select whether to continue linking outputting warning messages instead of errors if the following illegalities are found during relocation processing. - The result of address calculation of an unresolved external reference is illegal - The relationship with the section to be allocated is illegal This corresponds to the -Xignore_address_error option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xignore_address_error)	Continues linking outputting warning messages if an illegality is found during relocation processing.
		No	An error occurs if an illegality is found during relocation processing.

Check all multi-defined symbols	Select whether to output a message for all multi-defined external symbols and stop link processing. This corresponds to the -Xmultiple_symbol option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xmultiple_symbol)	Outputs a message for all multi-defined external symbols and stops link processing.
		No	Outputs a message for the first multi-defined external symbol and stops link processing.
Check illegality of external symbol	Select whether to check if the size and alignment conditions of an external symbol are invalid when linking it. This corresponds to the -Xlink_check_off option of the cx command.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Checks if the size and alignment conditions of an external symbol are invalid when linking it.
		No(-Xlink_check_off=symbol)	Does not check if the size and alignment conditions of an external symbol are invalid when linking it.
Check illegality of undefined external symbol	Select whether to check if the size and alignment conditions of an undefined external symbol are invalid when linking it. This corresponds to the -Xlink_check_off option of the cx command.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Checks if the size and alignment conditions of an undefined external symbol are invalid when linking it.
		No(-Xlink_check_off=undefined)	Does not check if the size and alignment conditions of an undefined external symbol are invalid when linking it.
Check allocation for internal ROM area	Select whether to check for the allocation to the internal ROM area. Select [No(-Xlink_check_off=irom)] when the ROM-less mode is used. This corresponds to the -Xlink_check_off option of the cx command.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Checks for the allocation to the internal ROM area.
		No(-Xlink_check_off=irom)	Does not check the allocation to the internal ROM area.

Rescan library files	Select whether to rescan the library file specified in the [Using libraries] and [System libraries] properties in the [Library] category. When this property is specified, symbols that are unresolved through the link sequence of the library can be prevented. This corresponds to the -Xrescan option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xrescan)	Rescans the library file to be used.
		No	Does not rescan the library file to be used.
Commands executed before link processing	Specify the command to be executed before link processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %LinkedFile%: Replaces with the absolute path of the output file under link processing. The specified command is displayed as the subproperty. This property is displayed only when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.		
	Default	Commands executed before link processing[*number of defined items*]	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 1023 characters Up to 64 items can be specified.	
Commands executed after link processing	Specify the command to be executed after link processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %LinkedFile%: Replaces with the absolute path of the output file under link processing. The specified command is displayed as the subproperty. This property is displayed only when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.		
	Default	Commands executed after link processing[*number of defined items*]	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 1023 characters Up to 64 items can be specified.	

Other additional options	Input the link option to be added additionally. The options set here are added at the end of the link options group. This property is displayed only when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.	
	Default	Blank
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 259 characters

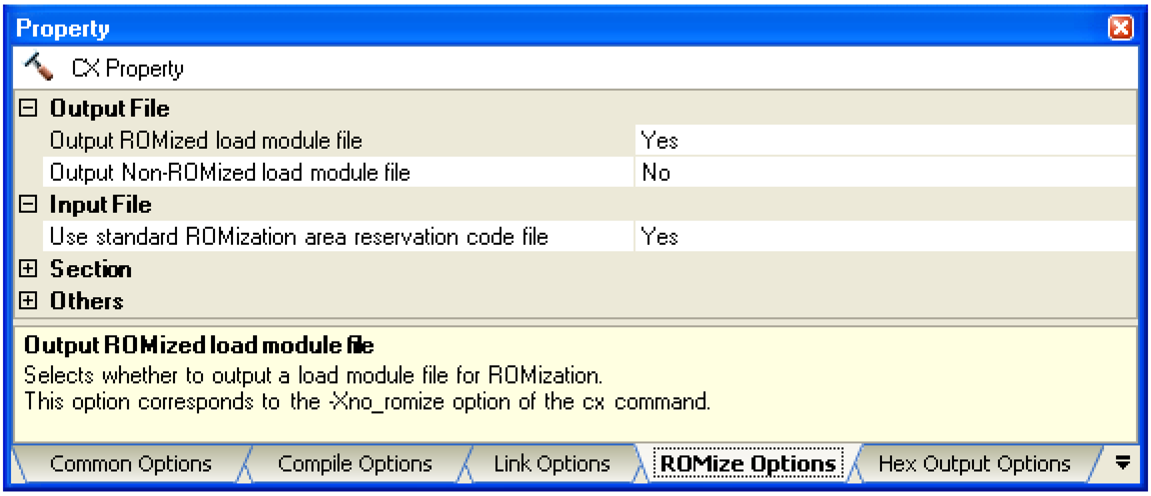
[ROMize Options] tab

This tab shows the detailed information on the ROMize phase categorized by the following and the configuration can be changed.

(1) [Output File]
(2) [Input File]
(3) [Section]
(4) [Others]

Caution This tab is not displayed for the library project.

Figure A-8. Property Panel: [ROMize Options] Tab



Property

CX Property

⊟ Output File

Output ROMized load module file	Yes
Output Non-ROMized load module file	No

⊟ Input File

Use standard ROMization area reservation code file	Yes

⊞ Section

⊞ Others

Output ROMized load module file
Selects whether to output a load module file for ROMization.
This option corresponds to the -Xno_romize option of the cx command.

[Description of each category]

(1) [Output File]

The detailed information on output files is displayed and the configuration can be changed.

<table>
<tr><td rowspan="4">Output ROMized load module file</td><td colspan="3">Select whether to output the ROMized load module file.
This corresponds to the -Xno_romize option of the cx command.</td></tr>
<tr><td>Default</td><td colspan="2">Yes</td></tr>
<tr><td>How to change</td><td colspan="2">Select from the drop-down list.</td></tr>
<tr><td>Restriction</td><td>Yes
No(-Xno_romize)</td><td>Outputs the ROMized load module file.
Does not output the ROMized load module file.</td></tr>
</table>

Output Non-ROMized load module file	Select whether to output the load module file before ROMization processing. This corresponds to the -Xlink_output option of the cx command. This property is displayed only when [Yes] in the [Output ROMized load module file] property is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Outputs the load module file before ROMization processing. The file is output under the file name specified in the [Output file name] property in the [Output File] category from the [Link Options] tab with string "_NonROMize" added.
		No	Does not output the load module file before ROMization processing.

(2) [Input File]

The detailed information on input files is displayed and the configuration can be changed.

Use standard ROMization area reservation code file	Select whether to use the standard ROMization area reservation code file (rompcrt.obj). This property is displayed only when [Yes] in the [Output ROMized load module file] property in the [Output File] category is selected.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Uses the standard ROMization area reservation code file.
		No	Does not use the standard ROMization area reservation code file. Make the ROMization area reservation code file, and specify the file for the [ROMization area reservation code file name] property.
ROMization area reservation code file name	Specify the name of the ROMization area reservation code file. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). If this is blank, a link error will occur. Be sure to specify this property. This corresponds to the -Xrompcrt option of the cx command. This property is displayed only when [No] in the [Use standard ROMization area reservation code file] property is selected.		
	Default	Blank	
	How to change	Directly enter in the text box or edit by the Specify ROMization Area Reservation Code File dialog box which appears when clicking the [...] button.	
	Restriction	Up to 259 characters	

(3) [Section]

The detailed information on the section is displayed and the configuration can be changed.

This category is displayed only when [Yes] in the [Output ROMized load module file] property in the [Output File] category is selected.

Start symbol of rompsec section	Specify the symbol to be used as the start address of the rompsec section to be generated. If this is blank, it is assumed that "__S_romp" has been specified. This corresponds to the -Xrompsec_start option of the cx command.		
	Default	Blank	
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.	
	Restriction	Up to 1022 characters	
Generate load module file has rompsec section only	Select whether to generate the load module file that includes only a rompsec section; no text-attribute section is included in the file to be generated. This corresponds to the -Xrompsec_only option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xrompsec_only)	Generates the load module file that has only the rompsec section.
		No	Includes the section with the text attribute in the load module file to be generated.
Data sections included in rompsec section	Specify the data section included in the rompsec section. Add one section in one line. Each of the specified data sections is included in the rompsec section, in descending order. This corresponds to the -Xrompsec_data option of the cx command. The specified section name is displayed as the subproperty.		
	Default	Data sections included in rompsec section[*number of specified items*]	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 1022 characters Up to 1024 items can be specified.	
Text sections included in rompsec section	Specify the text section included in the rompsec section. Add one section in one line. Each of the specified text sections is included in the rompsec section, in descending order. This corresponds to the -Xrompsec_text option of the cx command. The specified section name is displayed as the subproperty.		
	Default	Data sections included in text section[*number of specified items*]	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 1022 characters Up to 1024 items can be specified.	

(4) [Others]

Other detailed information on ROMization is displayed and the configuration can be changed.

This category is displayed only when [Yes] in the [Output ROMized load module file] property in the [Output File] category is selected.

Ignore ROMization error	Select whether to check the peripheral allocation error of the internal ROM for the rompsec section and duplicate addresses of the input file and output file. This corresponds to the -Xromize_check_off option of the cx command.		
	Default	No(None)	
	How to change	Select from the drop-down list.	
	Restriction	Yes(Ignore all)(-Xromize_check_off=rom_less,address)	Does not check the peripheral allocation error of the internal ROM and duplicate addresses.
		Yes(Ignore location error around internal ROM)(-Xromize_check_off=rom_less)	Does not check the peripheral allocation error of the internal ROM for the rompsec section. Specify this item when the ROM-less mode is used.
		Yes(Ignore overlapping address)(-Xromize_check_off=address)	Does not check the duplicate addresses of the input file and output file.
		No(None)	Checks the peripheral allocation error of the internal ROM and duplicate addresses.

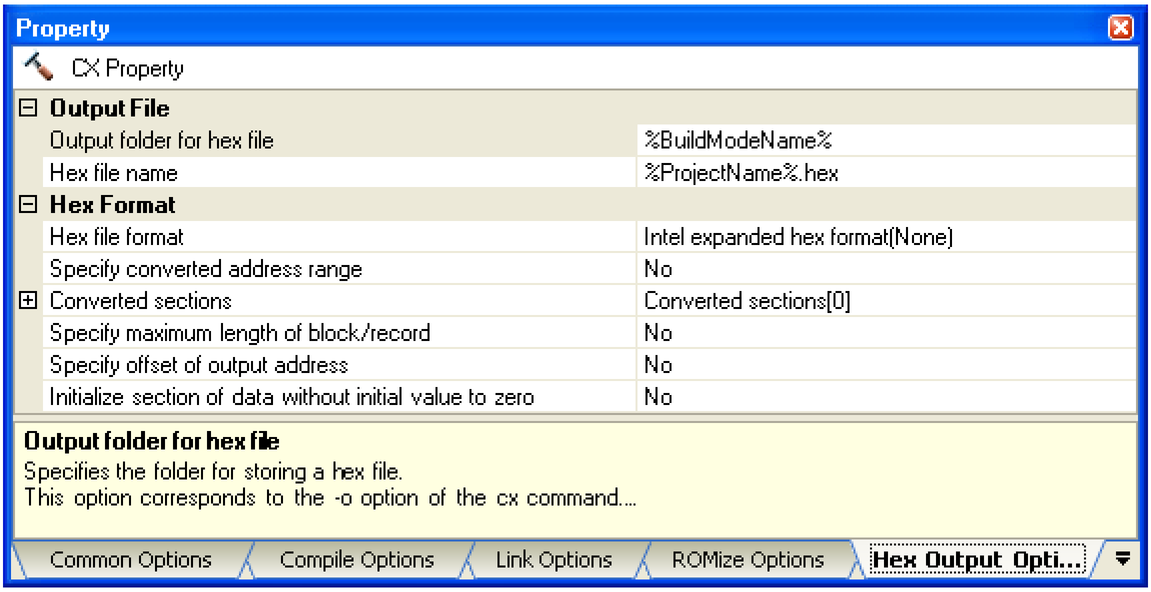
[Hex Output Options] tab

This tab shows the detailed information on the Hex output phase categorized by the following and the configuration can be changed.

(1) [Output File]
(2) [Hex Format]
(3) [Symbol Table]
(4) [Others]

Caution This tab is not displayed for the library project.

Figure A-9. Property Panel: [Hex Output Options] Tab



⊟ **Output File**

Output folder for hex file	%BuildModeName%
Hex file name	%ProjectName%.hex

⊟ **Hex Format**

Hex file format	Intel expanded hex format(None)
Specify converted address range	No
⊞ Converted sections	Converted sections[0]
Specify maximum length of block/record	No
Specify offset of output address	No
Initialize section of data without initial value to zero	No

Output folder for hex file
Specifies the folder for storing a hex file.
This option corresponds to the -o option of the cx command....

[Description of each category]

(1) [Output File]

The detailed information on output files is displayed and the configuration can be changed.

Output folder for hex file	Specify the folder which the hex file is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xhex option of the cx command.	
	Default	%BuildModeName%
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.
	Restriction	Up to 247 characters
Hex file name	Specify the hex file name. The extension can be freely specified. The following macro name is available as an embedded macro. %ProjectName%: Replaces with the project name. If this is blank, it is assumed that "%ProjectName%.hex" has been specified. This corresponds to the -Xhex option of the cx command.	
	Default	%ProjectName%.hex
	How to change	Directly enter in the text box.
	Restriction	Up to 259 characters

(2) [Hex Format]

The detailed information on the hex format is displayed and the configuration can be changed.

Hex file format	Select the format of the hex file to be output. This corresponds to the -Xhex_format option of the cx command.		
	Default	Intel expanded hex format(None)	
	How to change	Select from the drop-down list.	
	Restriction	Intel expanded hex format(None)	Specifies the Intel expanded hex format as the format of the hex file.
		Intel expanded hex format (32-bit address)(-Xhex_format=i)	Specifies the Intel expanded hex format (32-bit address) as the format of the hex file.
		Motorola S type format(standard address)(-Xhex_format=S)	Specifies the Motorola S type format (standard address) as the format of the hex file.
		Motorola S type format(32-bit address)(-Xhex_format=s)	Specifies the Motorola S type format (32-bit address) as the format of the hex file.
		Expanded Tektronix hex format(-Xhex_format=T)	Specifies the expanded Tektronix hex format as the format of the hex file.
Specify converted address range	Select whether to specify the address range to be converted to the hex file. This corresponds to the -Xhex_fill option of the cx command. This property is not displayed when [Expanded Tektronix hex format(-Xhex_format=T)] in the [Hex file format] property is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xhex_fill)	Specifies the address range to be converted to the hex file.
		No	Does not specify the address range to be converted to the hex file.
Filling value	Specify the filling value of the unused areas under the case of converting to the hex file. If this is blank, it is assumed that "FF" (hexadecimal number) has been specified. This corresponds to the -Xhex_fill option of the cx command. This property is not displayed when [Expanded Tektronix hex format(-Xhex_format=T)] in the [Hex file format] property is selected and [No] in the [Specify converted address range] property is selected.		
	Default	FFFF (hexadecimal number)	
	How to change	Directly enter in the text box.	
	Restriction	00 to FFFF (2- or 4-digit hexadecimal number)	
Start address	Specify the start address of the area to be converted to the hex file. If this property is specified, the [Size] property must be specified. If either of them is blank, all the codes in the internal ROM area defined by the device file are converted into the hex format. This corresponds to the -Xhex_fill option of the cx command. This property is not displayed when [Expanded Tektronix hex format(-Xhex_format=T)] in the [Hex file format] property is selected and [No] in the [Specify converted address range] property is selected.		
	Default	Blank	
	How to change	Directly enter in the text box.	
	Restriction	0 to FFFFFFFF (hexadecimal number)	

Size	Specify the size of the area to be converted to the hex file. If this property is specified, the [Start address] property must be specified. If either of them is blank, all the codes in the internal ROM area defined by the device file are converted into the hex format. This corresponds to the -Xhex_fill option of the cx command. This property is not displayed when [Expanded Tektronix hex format(-Xhex_format=T)] in the [Hex file format] property is selected and [No] in the [Specify converted address range] property is selected.		
	Default	Blank	
	How to change	Directly enter in the text box.	
	Restriction	1 to 100000000 (hexadecimal number)	
Converted sections	Specify the section to be converted to the hex file. Add one section in one line. When this property is omitted, all the sections with the section type other than "NOBITS" and section attribute "A" are converted to the hex file. This corresponds to the -Xhex_section option of the cx command. The specified section is displayed as the subproperty. This property is displayed when [Expanded Tektronix hex format(-Xhex_format=T)] in the [Hex file format] property is selected or [No] in the [Specify converted address range] property is selected.		
	Default	Converted sections[*number of set items*]	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 1022 characters Up to 1024 items can be specified.	
Specify maximum length of block/record	Select whether to specify the maximum length of block/record. This corresponds to the -Xhex_block_size option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xhex_block_size)	Specifies the maximum length of block/record.
		No	Regards the default value as the maximum length of block/record.

Maximum length of block/record	Specify the maximum length of block/record. This corresponds to the -Xhex_block_size option of the cx command. This property is displayed only when [Yes(-Xhex_block_size)] in the [Specify maximum length of block/record] property is selected.		
	Default	This differs depending on the item selected in the [Hex file format] property. - When [Intel expanded hex format(None)] or [Intel expanded hex format (32-bit address)(-Xhex_format=i)] is selected and [Reset to Default] from the context menu of this property is selected 32 - When [Motorola S type format(standard address)(-Xhex_format=S)] is selected and [Reset to Default] from the context menu of this property is selected 80 - When [Motorola S type format(32-bit address)(-Xhex_format=s)] is selected and [Reset to Default] from the context menu of this property is selected 80 - When [Expanded Tektronix hex format(-Xhex_format=T)] is selected and [Reset to Default] from the context menu of this property is selected 255 Note that if a change to the [Hex file format] property causes the value set in this property to be outside the specifiable range, then it is set immediately to the above default.	
	How to change	Directly enter in the text box.	
	Restriction	This differs depending on the item selected in the [Hex file format] property. - When [Intel expanded hex format(None)] or [Intel expanded hex format (32-bit address)(-Xhex_format=i)] is selected 1 to 255 (decimal number), or 01 to FF (hexadecimal number) - When [Motorola S type format(standard address)(-Xhex_format=S)] is selected 1 to 251 (decimal number), or 01 to FB (hexadecimal number) - When [Motorola S type format(32-bit address)(-Xhex_format=s)] is selected 1 to 250 (decimal number), or 01 to FA (hexadecimal number) - When [Expanded Tektronix hex format(-Xhex_format=T)] is selected 16 to 255 (decimal number), or 10 to FF (hexadecimal number)	
Specify offset of output address	Select whether to specify the offset of the address to be output. This corresponds to the -Xhex_offset option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xhex_offset)	Specifies the offset of the address to be output.
		No	Does not specify the offset of the address to be output.

Offset of output address	Specify the offset of the address to be output. This corresponds to the -Xhex_offset option of the cx command. This property is displayed only when [Yes(-Xhex_offset)] in the [Specify offset of output address] property is selected.		
	Default	0 (hexadecimal number)	
	How to change	Directly enter in the text box.	
	Restriction	0 to FFFFFFFE (hexadecimal number)	
Initialize section of data without initial value to zero	Select whether to initialize the section of the data without the initial value (the section with section type "NOBITS" and section attribute "A") to zero during the conversion to the hex file. This corresponds to the -Xhex_null option of the cx command. This property is displayed when [Expanded Tektronix hex format(-Xhex_format=T)] in the [Hex file format] property is selected or [No] in the [Specify converted address range] property is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xhex_null)	Initializes the section of the data without the initial value to zero.
		No	Does not initialize the section of the data without the initial value to zero.

(3) [Symbol Table]

The detailed information on the symbol table is displayed and the configuration can be changed.

This category is displayed only when [Expanded Tektronix hex format(-Xhex_format=T)] in the [Hex file format] property in the [Hex Format] category is selected.

Convert symbol table	Select whether to convert the symbol table during the conversion to the hex file. This corresponds to the -Xhex_symtab option of the cx command.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(Convert global and local symbols)(-Xhex_symtab=all)	Converts global symbols and local symbols.
		Yes(Convert global symbols)(-Xhex_symtab=global)	Converts only global symbols.
		No	Does not convert the symbol table.

(4) [Others]

Other detailed information on the hex output is displayed and the configuration can be changed.

This category is displayed only when [Yes] in the [Specify converted address range] property in the [Hex Format] category is selected.

Warn internal ROM overflow	Select whether to display the warning message when the area to be converted to the hex file overflows from the internal ROM area. This corresponds to the -Xhex_rom_less option of the cx command.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Displays the warning message when the area to be converted to the hex file overflows from the internal ROM area.
		No(-Xhex_rom_less)	Does not display the warning message when the area to be converted to the hex file overflows from the internal ROM area. If this item is specified, the [Start address] property and [Size] property must be specified.

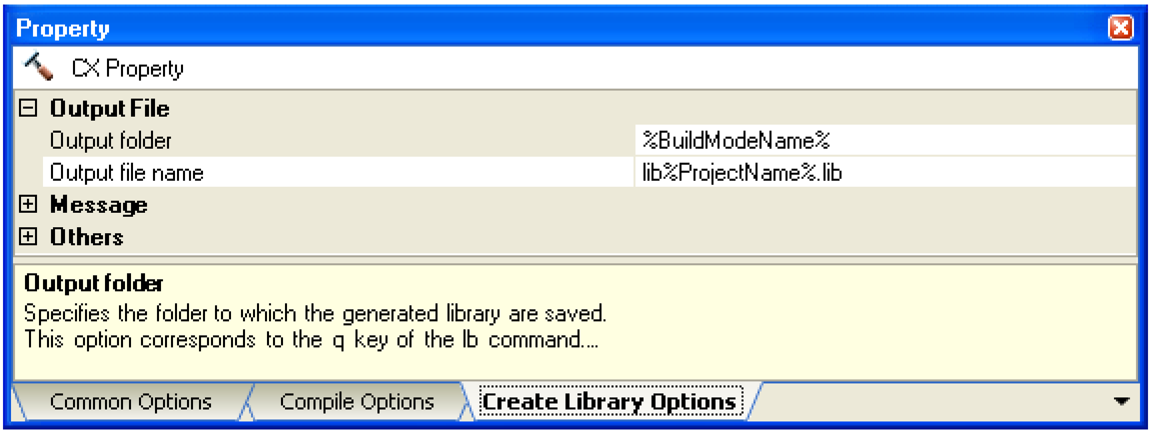
[Create Library Options] tab

This tab shows the detailed information on the create library phase categorized by the following and the configuration can be changed.

(1) [Output File]
(2) [Message]
(3) [Others]

Caution This tab is displayed for the library project.

Figure A-10. Property Panel: [Create Library Options] Tab



Property

CX Property

⊟ Output File

Output folder	%BuildModeName%
Output file name	lib%ProjectName%.lib

⊞ Message

⊞ Others

Output folder

Specifies the folder to which the generated library are saved.
This option corresponds to the q key of the lb command....

[Description of each category]

(1) [Output File]

The detailed information on output files is displayed and the configuration can be changed.

Output folder	Specify the folder which the library file to be generated is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the r key of the librarian.	
	Default	%BuildModeName%
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.
	Restriction	Up to 247 characters

Output file name	Specify the name of the library file to be generated. The extension other than ".lib" cannot be specified. If the extension is omitted, ".lib" is automatically added. The following macro name is available as an embedded macro. %ProjectName%: Replaces with the project name. If this is blank, it is assumed that "lib%ProjectName%.lib" has been specified. This corresponds to the r key of the librarian.	
	Default	lib%ProjectName%.lib
	How to change	Directly enter in the text box.
	Restriction	Up to 259 characters

(2) [Message]

The detailed information on messages is displayed and the configuration can be changed.

Verbose mode	Select whether to display the execution status of the librarian[Note] on the Output panel. This corresponds to the v option of the librarian.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes(v)	Displays the execution status of the librarian.
		No	Does not display the execution status of the librarian.

Note The meaning of the output format of the execution status is shown below.

Output Format	Meaning
q - *file-name*	Create a new library file, or add a member

(3) [Others]

Other detailed information on creating a library is displayed and the configuration can be changed.

Commands executed before library generate processing	Specify the command to be executed before library generation processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %LibraryFile%: Replaces with the absolute path of the output file under the library generation processing. The specified command is displayed as the subproperty.	
	Default	Commands executed before library generate processing[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.

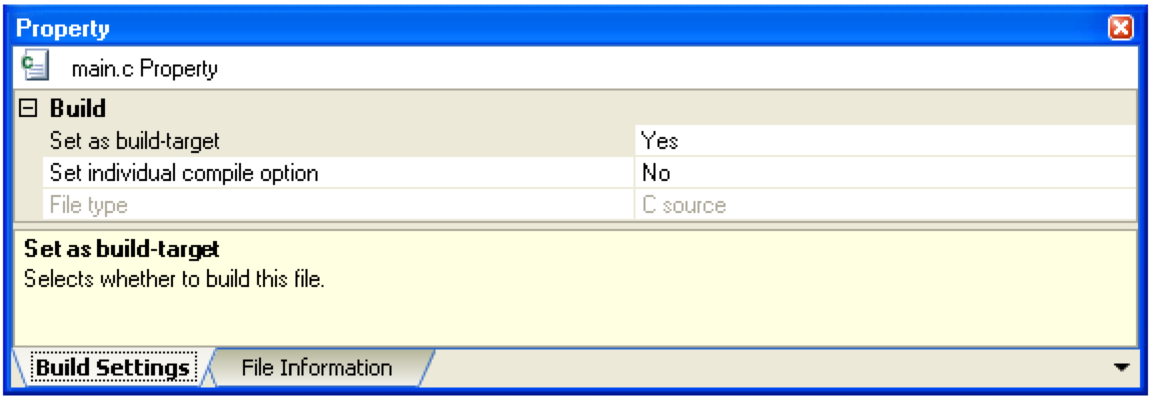
Commands executed after library generate processing	Specify the command to be executed after library generation processing. Use the call instruction to specify a batch file (example: call a.bat). The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %LibraryFile%: Replaces with the absolute path of the output file under the library generation processing. The specified command is displayed as the subproperty.	
	Default	Commands executed after library generate processing[*number of defined items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.
Other additional options	Input the create library options to be added additionally. The options set here are added at the end of the create library options group.	
	Default	Blank
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 259 characters

[Build Settings] tab

This tab shows the detailed information on each C source file, assembler source file, object module file, link directive file, symbol information file, and library file categorized by the following and the configuration can be changed.

(1) [Build]
(2) [Multi-Core]

Figure A-11. Property Panel: [Build Settings] Tab (When Selecting C Source File)

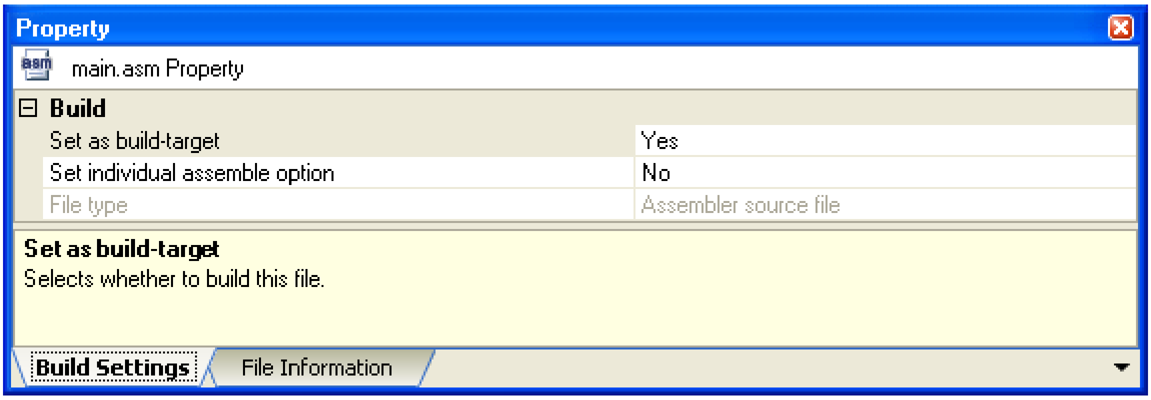


⊟ Build

Set as build-target	Yes
Set individual compile option	No
File type	C source

Set as build-target
Selects whether to build this file.

Figure A-12. Property Panel: [Build Settings] Tab (When Selecting Assembler Source File)

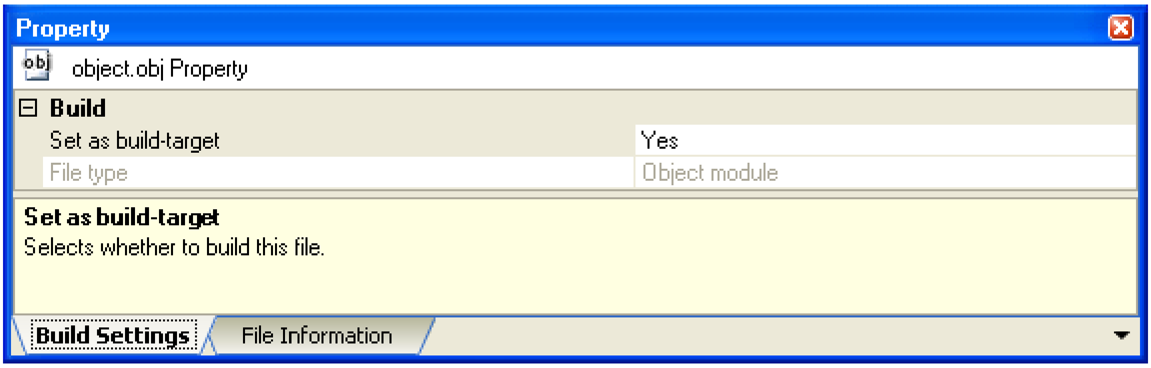


⊟ Build

Set as build-target	Yes
Set individual assemble option	No
File type	Assembler source file

Set as build-target
Selects whether to build this file.

Figure A-13. Property Panel: [Build Settings] Tab (When Selecting Object Module File)

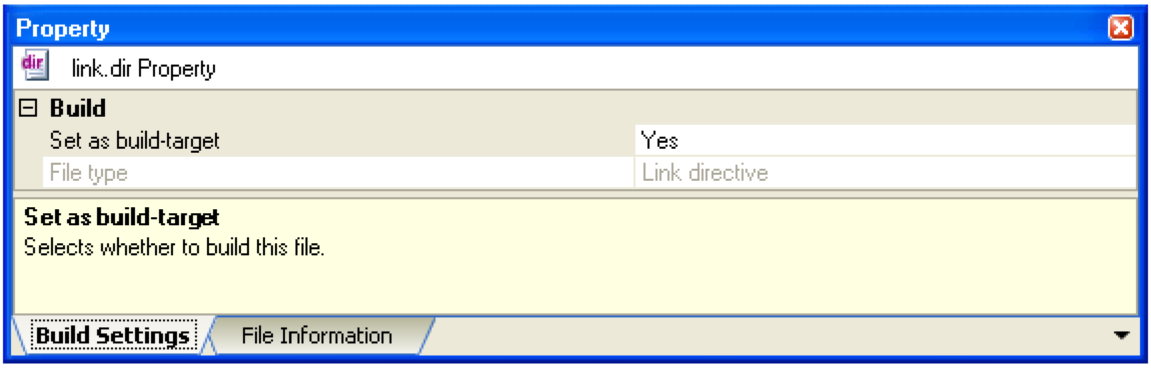


⊟ Build

Set as build-target	Yes
File type	Object module

Set as build-target
Selects whether to build this file.

Figure A-14. Property Panel: [Build Settings] Tab (When Selecting Link Directive File)



Property

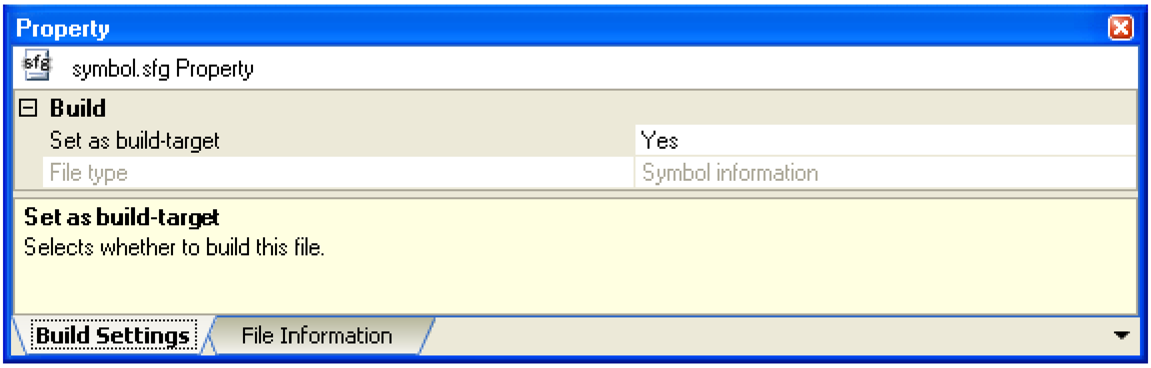
link.dir Property

⊟ Build

Set as build-target	Yes
File type	Link directive

Set as build-target
Selects whether to build this file.

Figure A-15. Property Panel: [Build Settings] Tab (When Selecting Symbol Information File)



Property

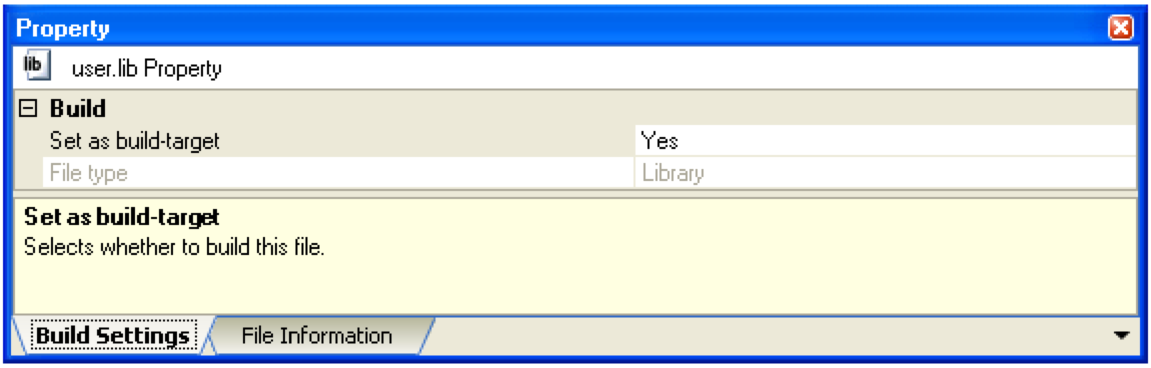
symbol.sfg Property

⊟ Build

Set as build-target	Yes
File type	Symbol information

Set as build-target
Selects whether to build this file.

Figure A-16. Property Panel: [Build Settings] Tab (When Selecting Library File)



Property

user.lib Property

⊟ Build

Set as build-target	Yes
File type	Library

Set as build-target
Selects whether to build this file.

[Description of each category]

(1) [Build]

The detailed information on the build is displayed and the configuration can be changed.

Set as build-target	Select whether to run a build of the selected file.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Runs a build of the selected file.
		No	Does not run a build of the selected file.

Set individual compile option	Select whether to set the compile option that differs from the project settings to the selected C source file. This property is displayed only when a C source file is selected on the project tree and [Yes] in the [Set as build-target] property from this tab is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Sets the option that differs from the project settings to the selected C source file.
		No	Does not set the option that differs from the project settings to the selected C source file.
Set individual assemble option	Select whether to set the assemble option that differs from the project settings to the selected assembler source file. This property is displayed only when the assembler source file is selected on the project tree and [Yes] in the [Set as build-target] property from this tab is selected.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Sets the option that differs from the project settings to the selected assembler source file.
		No	Does not set the option that differs from the project settings to the selected assembler source file.
File type	The type of the selected file is displayed.		
	Default	C source (when the C source file is selected) Assembler source (when the assembler source file is selected) Object module (when the object module file is selected) Link directive (when the link directive file is selected) Symbol information (when the symbol information file is selected) Library (when the library file is selected)	
	How to change	Changes not allowed	

(2) [Multi-Core]

The detailed information on the multi-core is displayed and the configuration can be changed.

Note that this category is displayed only when a C source file, assembler source file, object module file, or library file is selected in the project using devices supporting multi-core.

Target core number	Specify the target core number. [Core *n* (-Xmulti=pe*n*)]) shows the number of cores. (*n*: number of cores on the target CPU) This corresponds to the -Xmulti option of the cx command.		
	Default	Common(-Xmulti=cmn)	
	How to change	Select from the drop-down list.	
	Restriction	Common(-Xmulti=cmn)	The code and data in the selected file will be available to all cores.
		Core *n*(-Xmulti=pe*n*)	The code and data in the selected file will be available to the specified core only.

[Individual Compile Options] tab

This tab shows the detailed information on a C source file categorized by the following and the configuration can be changed.

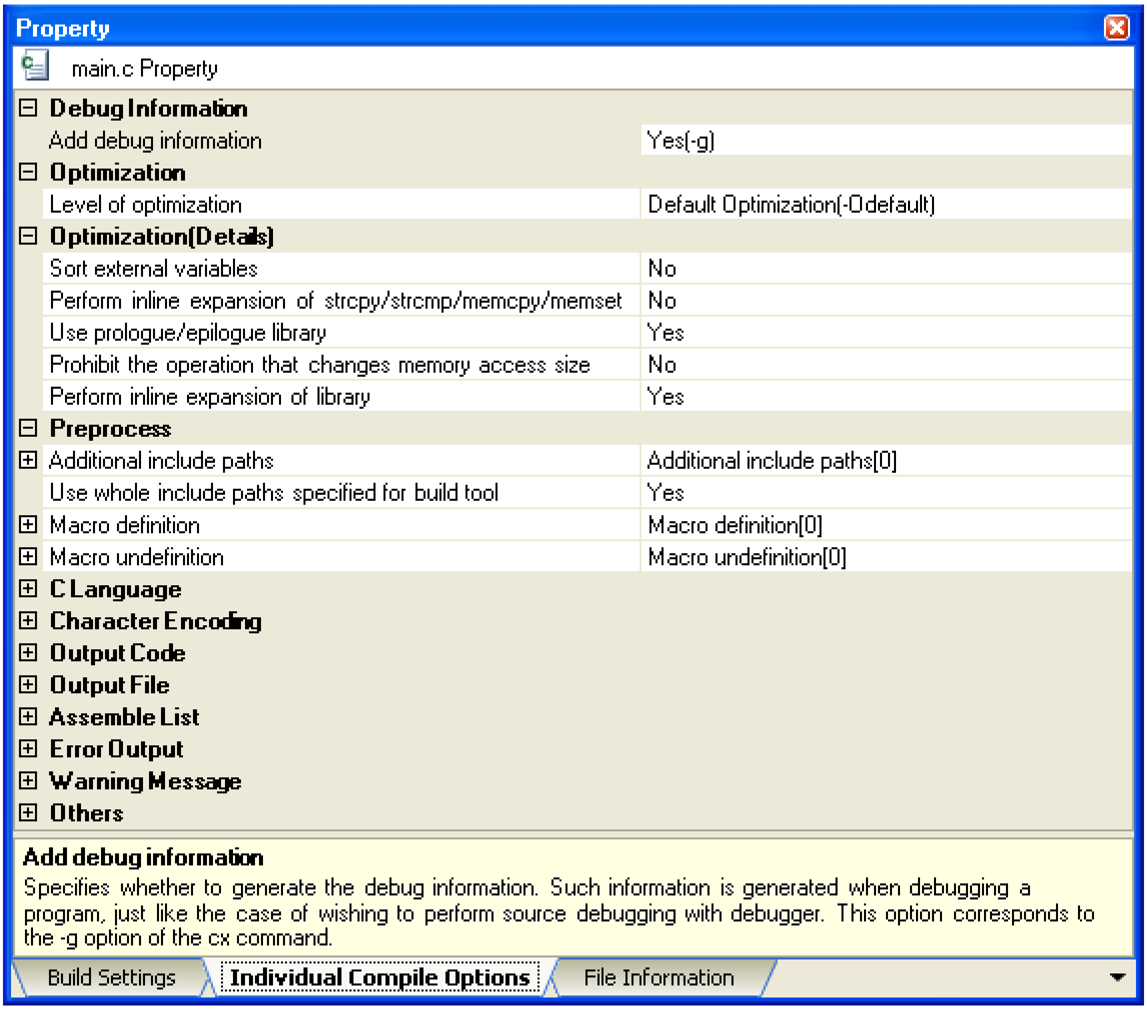
Note that this tab takes over the settings of the [Common Options] tab and [Compile Options] tab.

When the settings are changed from these tabs, the properties are displayed in boldface.

(1) [Debug Information]
(2) [Optimization]
(3) [Optimization(Details)]
(4) [Preprocess]
(5) [C Language]
(6) [Character Encoding]
(7) [Output Code]
(8) [Output File]
(9) [Assemble List]
(10) [Error Output]
(11) [Warning Message]
(12) [Others]

Remark This tab is displayed only when [Yes] in the [Set individual compile option] property in the [Build] category from the [Build Settings] tab is selected.

Figure A-17. Property Panel: [Individual Compile Options] Tab


Property

main.c Property

⊟ **Debug Information**

Add debug information	Yes(-g)

⊟ **Optimization**

Level of optimization	Default Optimization(-Odefault)

⊟ **Optimization(Details)**

Sort external variables	No
Perform inline expansion of strcpy/strcmp/memcpy/memset	No
Use prologue/epilogue library	Yes
Prohibit the operation that changes memory access size	No
Perform inline expansion of library	Yes

⊟ **Preprocess**

⊞ Additional include paths	Additional include paths[0]
Use whole include paths specified for build tool	Yes
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]

⊞ **C Language**

⊞ **Character Encoding**

⊞ **Output Code**

⊞ **Output File**

⊞ **Assemble List**

⊞ **Error Output**

⊞ **Warning Message**

⊞ **Others**

Add debug information

Specifies whether to generate the debug information. Such information is generated when debugging a program, just like the case of wishing to perform source debugging with debugger. This option corresponds to the -g option of the cx command.

[Description of each category]

(1) [Debug Information]

The detailed information on debug information is displayed and the configuration can be changed.

Add debug information	Select whether to generate the debug information. It is possible to perform source debugging with the debugger by outputting information for source debugging to the output file. This corresponds to the -g option of the cx command.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-g)	Generates the debug information.
		No	Does not generate the debug information.

(2) [Optimization]

The detailed information on the optimization is displayed and the configuration can be changed.

Level of optimization	Select the level of the optimization for compiling. This corresponds to the -O option of the cx command.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Default Optimization(-Odefault)	Performs optimization that debugging is not affected (optimization of expressions and register allocation, and the like).
		Code Size Precedence(-Osize)	Performs optimization with the object size precedence. Regards reducing the ROM/RAM capacity as important and performs the maximum optimization that is effective for general programs.
		Speed Precedence(-Ospeed)	Performs optimization with the execution speed precedence. Regards shortening the execution speed as important and performs the maximum optimization that is effective for general programs.
		Debug Precedence(-Onothing)	Performs optimization with the debug precedence. Regards debugging as important and suppresses all optimization including default optimization.
		Detail Setting	The properties are added in the [Optimization(Details)] category. The optimization items selected in the category has the precedence for the optimization.

(3) [Optimization(Details)]

The detailed information on the optimization is displayed and the configuration can be changed.

Maximum number of loop expansions	Specify the maximum number of times to expand the loops such as "for" and "while". If 0 or 1 is specified, expansion is suppressed. If this is blank, it is assumed that "4" has been specified. This corresponds to the -Ounroll option of the cx command. This property is displayed only when [Detail Setting] in the [Level of optimization] property is selected.		
	Default	*Configuration of the compile option*	
	How to change	Directly enter in the text box.	
	Restriction	0 to 999 (decimal number) or blank	
Remove unused static functions	Select whether to remove the static functions which are not called. This corresponds to the -Odelete_static_func option of the cx command. This property is displayed only when [Detail Setting] in the [Level of optimization] property is selected.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Odelete_static_func=on)	Removes the unused static functions which are not called.
		No	Does not remove the unused static functions which are not called.

Perform inline expansion	Specify whether to perform inline expansion at the location calling functions. This corresponds to the -Oinline option of the cx command. This property is displayed only when [Detail Setting] in the [Level of optimization] property is selected.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(Only Specified Functions)(None)	Performs inline expansion at the location calling the function for which #pragma inline is specified.
		Yes(Auto-detect)(-Oinline=2)	Distinguishes the function that is the target of inline expansion automatically and expands it.
		No(-Oinline=0)	Does not specify inline expansion of the function.
Perform pipeline optimization	Select whether to improve the program's execution performance by reordering instructions at the machine-language level. This corresponds to the -Opipeline option of the cx command. This property is displayed only when the project uses a device with the V850E2V3 architecture and when [Detail Setting] in the [Level of optimization] property in the [Optimization] category is selected.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Opipeline=on)	Performs pipeline optimization.
		No	Does not specify pipeline optimization.
Sort external variables	Select whether to sort external variables. This property is for reducing the RAM capacity. This corresponds to the -Xsort_var option of the cx command.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xsort_var)	Rearranges external variables allocated to a section other than const/sconst sequentially, starting from the largest alignment size.
		No	Does not specify sorting external variables.

Perform inline expansion of strcpy/strcmp/ memcpy/memset	Selsect whether to perform inline expansion of functions "strcpy()", "strcmp()", "memcpy()", and "memset()" calls, with regarding the alignment conditions of the array (including character strings) and the structure as 4 bytes. This improves the execution speed of the program to be generated, but it increases the code size. This corresponds to the -Xinline_strcpy option of the cx command. This property is displayed only when [No] in the [Structure packing] property in the [Output Code] category from the [Compile Options] tab is selected.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xinline_strcpy)	Performs inline expansion of functions "strcpy()", "strcmp()", "memcpy()", and "memset()" calls.
		No	Does not specify inline expansion of functions "strcpy()", "strcmp()", "memcpy()", and "memset()" calls.
Use prologue/epilogue library	Specify whether to perform prologue/epilogue processing of functions through runtime library calls. This corresponds to the -Xpro_epi_runtime option of the cx command. This property is displayed only when [Speed Precedence(-Ospeed)] in the [Level of optimization] property in the [Optimization] category is selected.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Performs prologue/epilogue processing of the function through runtime library calls.
		No(-Xpro_epi_runtime=off)	Does not perform prologue/epilogue processing of functions through runtime library calls.
Prohibit the operation that changes memory access size	Select whether to prohibit replacing the load and store instructions with 1-bit manipulation instructions (set1, clr1, tst1, and not1). This corresponds to the -Xkeep_access_size option of the cx command.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xkeep_access_size)	Prohibits replacing the load and store instructions with 1-bit manipulation instructions.
		No	1-bit manipulation instructions may be generated instead of load and store instructions.
Perform inline expansion of library	Specify whether to perform inline expansion at the location calling library functions. This corresponds to the -Xcall_lib option of the cx command. This property is displayed only for the project using a device with the V850E2V3 architecture.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Generates a sqrtf.s instruction instead of a call to the sqrtf function.
		No(-Xcall_lib)	Always generates a call to the sqrtf function.

(4) [Preprocess]

The detailed information on preprocessing is displayed and the configuration can be changed.

Additional include paths	Specify the additional include paths during compiling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The specified include path is searched with higher priority than the standard include file folder of CX. The reference point of the path is the project folder. When this property is omitted, only the standard folder of CX is searched. This corresponds to the -I option of the cx command. The specified include path is displayed as the subproperty. Uppercase characters and lowercase characters are not distinguished for the include paths.		
	Default	Additional include paths[*number of defined items*]	
	How to change	Edit by the Path Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 259 characters Up to 256 items can be specified.	
Use whole include paths specified for build tool	Select whether to compile using the include path specified in the [Additional include paths] property in the [Preprocess] category from the [Compile Options] tab of the build tool to be used. The include paths are added by the following procedure. - Paths specified in the [Additional include paths] property from this tab - Paths specified in the [Additional include paths] property from the [Compile Options] tab - Paths displayed in the [System include paths] property from the [Compile Options] tab This corresponds to the -I option of the cx command.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Compiles using the include path specified in the property of the build tool to be used.
		No	Does not use the include path specified in the property of the build tool to be used.
Macro definition	Specify the name of the macro to be defined. Specify in the format of "*macro name*=*defined value*", with one macro name per line. The "=*defined value*" part can be omitted, and in this case, "1" is used as the defined value. This corresponds to the -D option of the cx command. The specified macro is displayed as the subproperty.		
	Default	*Configuration of the compile option*	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 256 characters Up to 256 items can be specified.	

Macro undefinition	Specify the macro name to be undefined. Specify in the format of "*macro name*", with one macro name per line. This corresponds to the -U option of the cx command. The specified macro is displayed as the subproperty.		
	Default	*Configuration of the compile option*	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 256 characters Up to 256 items can be specified.	
Output C source comments to preprocessed file	Select whether to output the comments of the C source to the preprocessed file. This corresponds to the -Xpreprocess option of the cx command. This property is displayed only when [Yes(-P)] in the [Output preprocessed source file] property in the [Output File] category is selected.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xpreprocess=comment)	Outputs the comments of the C source to the preprocessed file.
		No	Does not output the comments of the C source to the preprocessed file.
Output line number information to preprocessed file	Select whether to output the line number information of the C source to the preprocessed file. This corresponds to the -Xpreprocess option of the cx command. This property is displayed only when [Yes(-P)] in the [Output preprocessed source file] property in the [Output File] category is selected.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xpreprocess= line)	Outputs the line number information of the C source to the preprocessed file.
		No	Does not output the line number information of the C source to the preprocessed file.

(5) [C Language]

The detailed information on C language is displayed and the configuration can be changed.

Compile strictly according to ANSI standards	Select whether to process as making C source program comply strictly with the ANSI standard and output an error or warning for a specification that violates the standard. This corresponds to the -Xansi option of the cx command.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xansi)	Processes as making C source program comply strictly with the ANSI standard and outputs an error or warning for a specification that violates the standard.
		No	Compatibility with the conventional C language specifications is conferred and processing continues after warning is output.

(6) [Character Encoding]

The detailed information on character encoding is displayed and the configuration can be changed.

Character encoding	Select the character code to be used for Japanese comments and character strings in the source file. This corresponds to the -Xcharacter_set option of the cx command.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	SJIS(None)	Interprets the Japanese character code in the source file as SJIS.
		EUC(-Xcharacter_set=euc_jp)	Interprets the Japanese character code in the source file as EUC.
		UFT-8(-Xcharacter_set=utf8)	Interprets the Japanese character code in the source file as UFT-8.
		Big5(-Xcharacter_set=big5)	Interprets the Chinese character code in the source file as Traditional Chinese.
		GB2312(-Xcharacter_set=gb2312)	Interprets the Chinese character code in the source file as Simplified Chinese.
		No-process(-Xcharacter_set=none)	Does not process the Japanese character code in the source file.

(7) [Output Code]

The detailed information on output codes is displayed and the configuration can be changed.

Output comment to assembler source file	Select whether to output a C source program as a comment to the assembler source file to be output. This corresponds to the -Xpass_source option of the cx command. This property is displayed only when [Yes(-Xasm_path)] in the [Output assembler source file] property in the [Output File] category or [Yes(-Xprn_path)] in the [Output assemble list file] property is selected.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xpass_source)	Outputs a C source program as a comment to the assembler source file.
		No	Does not output a C source program as a comment to the assembler source file.

Output code of switch statement	Select the code output mode for switch statements in programs. This corresponds to the -Xswitch option of the cx command.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Auto(None)	The cx selects the optimum output format.
		if-else(-Xswitch=ifelse)	Outputs the switch statements in the same format as the if-else statement along a string of case statements in programs. Select this item if the case statements are written in the order of frequency or if only a few labels are used. Because the case statements are compared starting from the top, unnecessary comparison can be reduced and the execution speed can be increased if the case statement that most often matches is written first.
		Binary search(-Xswitch=binary)	Outputs the code in the binary search format for switch statements in programs. Searches for a matching case statement by using a binary search algorithm. If this item is selected when many labels are used, any case statement can be found at almost the same speed.
		Table jump(-Xswitch=table)	Outputs the code in the table jump format for switch statements in programs. References a table indexed on the values in the case statements, and selects and processes case labels from the switch statement values. The code will branch to all the case statements with about the same speed. However, if case values are not used in succession, an unnecessary area will be created.
Label size of switch table	Select the size per label of the branch table for the case labels in switch statements. This corresponds to the -Xword_case option of the cx command. This property is displayed only when [Auto(None)] or [Table jump(-Xswitch=table)] in the [Output code of switch statement] property is selected. If [Auto(None)] is selected in the [Output code of switch statement] property, and the cx command automatically selects [if-else(-Xswitch=ifelse)] or [Binary search(-Xswitch=binary)], the setting of this property will be disabled (set to 2 bytes).		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	2 bytes(None)	Generates one 2-byte branch table per case label in a switch statement.
		4 bytes(-Xword_case)	Generates one 4-byte branch table per case label in a switch statement. Select this item when a compile error occurs because the switch statement is long.

Floating-point calculating type	Select whether to generate runtime library call instructions for floating-point calculations, or to generate floating-point instructions for the floating point unit (FPU). This corresponds to the -Xfloat [V850E2V3] option of the cx command. This property is displayed only for the project using a device with the V850E2V3 architecture having an FPU.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Auto(None)	If the target device does not have FPU, floating-point calculation instructions are not generated and runtime library call instructions are generated. If the target device has FPU, floating-point calculation instructions are generated.
		Software Calculating(-Xfloat=soft)	Generates runtime library call instructions for floating-point calculations.
		FPU Calculating(-Xfloat=fpu)	Generates floating-point calculation instructions of FPU (floating-point unit) for floating-point calculations.
Generate div/divu instructions	Select whether to generate the div and divu instructions instead of the divq and divqu instructions for division. Although the divq and divqu instructions are fast, the number of execution cycles will differ depending on the values of the operands. This corresponds to the -Xdiv [V850E2V3] option of the cx command. This property is displayed only for the project using a device with the V850E2V3 architecture having an FPU.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xdiv)	Generates the div and divu instructions for division.
		No	Generates the divq and divqu instructions for division.
Far Jump file names	Specify the Far Jump file name. The Far Jump file outputs the code that uses the jmp instruction (V850E/ES core) or jarl32 and jr32 instruction (V850E2 core or V850E2V3 architecture) for branch instructions of functions described in the file. The cx command outputs an error if the function is in a range that cannot be branched to by the jarl or jr directive (± 2MB or more), in which case the Far Jump file is used to recompile. Use the extension ".fjp". This corresponds to the -Xfar_jump option of the cx command. The specified Far Jump file name is displayed as the subproperty.		
	Default	*Configuration of the compile option*	
	How to change	Click the [...] button to open the Path Edit dialog box. -> Edit by the Specify Far Jump File dialog box which appears when clicking the [Browse...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 259 characters Up to 5000 items can be specified. However, only the file name specified last is valid.	

(8) [Output File]

The detailed information on output files is displayed and the configuration can be changed.

Object module file name	Specify the name of the object module file generated after compilation. The extension other than ".obj" cannot be specified. If the extension is omitted, ".obj" is automatically added. If this is blank, the file name will be the source file name with the extension replaced by ".obj". This corresponds to the -o option of the cx command.		
	Default	Blank	
	How to change	Directly enter in the text box.	
	Restriction	Up to 259 characters	
Output assembler source file	Select whether to output the assembler source file of the compile result for the C source. This corresponds to the -Xasm_path option of the cx command.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xasm_path)	Outputs the assembler source file of the compile result for the C source.
		No	Does not output the assembler source file of the compile result for the C source.
Output folder for assembler source file	Specify the folder which the assembler source file is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. The assembler source file is saved under the C source file name with the extension replaced by ".asm". If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xasm_path option of the cx command. This property is displayed only when [Yes(-Xasm_path)] in the [Output assembler source file] property is selected.		
	Default	*Configuration of the compile option*	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 247 characters	

Output preprocessed source file	Select whether to output the execution result of preprocessing for the source file to a file. The file is output to the folder specified in the [Intermediate file output folder] property. The file is output under the source file name with the extension replaced by ".i". This corresponds to the -P option of the cx command. This property is displayed only when [No] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-P)	Outputs the execution result of preprocessing for the source file to a file.
		No	Does not output the execution result of preprocessing for the source file to a file.

(9) [Assemble List]

The detailed information on the assemble list is displayed and the configuration can be changed.

Output assemble list file	Select whether to output the assemble list file. This corresponds to the -Xprn_path option of the cx command.		
	Default	*Configuration of the compile option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xprn_path)	Outputs the assemble list file.
		No	Does not output the assemble list file.
Output folder for assemble list file	Specify the folder which the assemble list file is output. The assemble list file is output under the source file name with the extension replaced by ".prn". If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xprn_path option of the cx command. This property is displayed only when [Yes(-Xprn_path)] in the [Output assemble list file] property is selected.		
	Default	*Configuration of the compile option*	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 247 characters	

(10) [Error Output]

The detailed information on the error output is displayed and the configuration can be changed.

Output error message file	Select whether to output the error message file to the Output panel. This corresponds to the -Xerror_file option of the cx command.		
	Default	*Configuration of the common option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xerror_file)	Outputs the error message file.
		No	Does not output the error message file.
Error message file output folder	Specify the folder which the error message file is output. If a relative path is specified, the reference point of the path is the main project or subproject folder. If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different). The following macro name is available as an embedded macro. %BuildModeName%: Replaces with the build mode name. If this is blank, it is assumed that the project folder has been specified. This corresponds to the -Xerror_file option of the cx command. This property is displayed only when [Yes(-Xerror_file)] in the [Output error message file] property is selected.		
	Default	*Configuration of the common option*	
	How to change	Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.	
	Restriction	Up to 247 characters	
Error message file name	Specify the error message file name. The extension can be freely specified. The following macro name is available as an embedded macro. %ProjectName%: Replaces with the project name. If this is blank, it is assumed that "%ProjectName%.err" has been specified. This corresponds to the -Xerror_file option of the cx command. This property is displayed only when [Yes(-Xerror_file)] in the [Output error message file] property is selected.		
	Default	*Configuration of the common option*	
	How to change	Directly enter in the text box.	
	Restriction	Up to 259 characters	

(11) [Warning Message]

The detailed information on warning messages is displayed and the configuration can be changed.

Displayed warning message	Specify the warning message number to be displayed regardless of the warning level. If multiple message numbers are specified, delimit them with "," (comma) (example: 02042,02107). Also, the range can be set using "-" (hyphen) (example: 02222-02554,02699-02782). If the same number is specified on the [Undisplayed warning message] property and this property, the number specified on this property takes precedence. This corresponds to the -Xwarning option of the cx command.	
	Default	*Configuration of the common option*
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 2048 characters
Undisplayed warning message	Specify the warning message number to not be displayed regardless of the warning level. If multiple message numbers are specified, delimit them with "," (comma) (example: 02042,02107). Also, the range can be set using "-" (hyphen) (example: 02222-02554,02699-02782). If the same number is specified on the [Displayed warning message] property and this property, the number specified on the [Displayed warning message] property takes precedence. This corresponds to the -Xno_warning option of the cx command.	
	Default	*Configuration of the common option*
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 2048 characters

(12) [Others]

Other detailed information on compilation is displayed and the configuration can be changed.

Commands executed before compile processing	Specify the command to be executed before compile processing. The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %InputFile%: Replaces with the absolute path of the file to be compiled. %CompiledFile%: Replaces with the absolute path of the output file under compiling. The specified command is displayed as the subproperty.	
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected Blank - Other than above *Configuration of the compile option*
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.

Commands executed after compile processing	Specify the command to be executed after compile processing. The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %InputFile%: Replaces with the absolute path of the file to be compiled. %CompiledFile%: Replaces with the absolute path of the output file under compiling. The specified command is displayed as the subproperty.	
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected Blank - Other than above *Configuration of the compile option*
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.
Other additional options	Input the compile option to be added additionally. The options set here are added at the end of the compile options group.	
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected Blank - Other than above *Configuration of the compile option*
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 259 characters

[Individual Assemble Options] tab

This tab shows the detailed information on an assemble source file categorized by the following and the configuration can be changed.

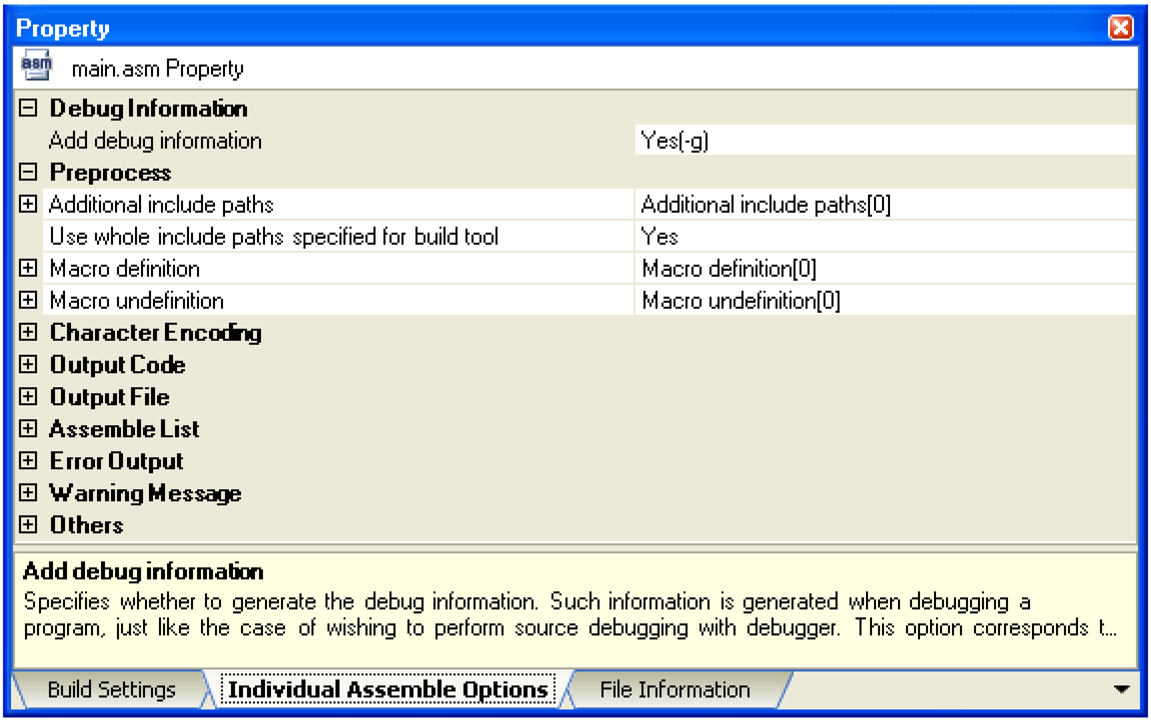
Note that this tab takes over the settings of the [Common Options] tab, [Compile Options] tab, and [Assemble Options] tab.

When the settings are changed from these tabs, the properties are displayed in boldface.

(1) [Debug Information]
(2) [Preprocess]
(3) [Character Encoding]
(4) [Output Code]
(5) [Output File]
(6) [Assemble List]
(7) [Error Output]
(8) [Warning Message]
(9) [Others]

Remark This tab is displayed only when [Yes] in the [Set individual assemble option] property in the [Build] category from the [Build Settings] tab is selected.

Figure A-18. Property Panel: [Individual Assemble Options] Tab



⊟ **Debug Information**	
Add debug information	Yes(-g)
⊟ **Preprocess**	
⊞ Additional include paths	Additional include paths[0]
Use whole include paths specified for build tool	Yes
⊞ Macro definition	Macro definition[0]
⊞ Macro undefinition	Macro undefinition[0]
⊞ **Character Encoding**	
⊞ **Output Code**	
⊞ **Output File**	
⊞ **Assemble List**	
⊞ **Error Output**	
⊞ **Warning Message**	
⊞ **Others**	

Add debug information
Specifies whether to generate the debug information. Such information is generated when debugging a program, just like the case of wishing to perform source debugging with debugger. This option corresponds t...

[Description of each category]

(1) [Debug Information]

The detailed information on debug information is displayed and the configuration can be changed.

Add debug information	Select whether to generate the debug information. It is possible to perform source debugging with the debugger by outputting information for source debugging to the output file. This corresponds to the -g option of the cx command.		
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected *Configuration of the compile option* - Other than above *Configuration of the assemble option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-g)	Generates the debug information.
		No	Does not generate the debug information.

(2) [Preprocess]

The detailed information on preprocessing is displayed and the configuration can be changed.

Additional include paths	Specify the additional include paths during assembling. The following macro names are available as embedded macros. %BuildModeName%: Replaces with the build mode name. %ProjectName%: Replaces with the project name. %CubeSuitePath%: Replaces with the absolute path of the CubeSuite install folder. The specified include path is searched with higher priority than the standard include file folder of CX. The reference point of the path is the project folder. When this property is omitted, only the standard folder of CX is searched. This corresponds to the -I option of the cx command. The specified include path is displayed as the subproperty. Uppercase characters and lowercase characters are not distinguished for the include paths.	
	Default	Additional include paths[*number of defined items*]
	How to change	Edit by the Path Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 259 characters Up to 256 items can be specified.

Use whole include paths specified for build tool	Select whether to assemble using the include path specified in the [Additional include paths] property in the [Preprocess] category from the [Assemble Options] tab of the build tool to be used. The setting of the [Compile Options] tab is used when [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected. When [Yes] in the [Build simultaneously] property is selected, the include paths are added by the following procedure. - Paths specified in the [Additional include paths] property from this tab - Paths specified in the [Additional include paths] property from the [Compile Options] tab - Paths displayed in the [System include paths] property from the [Compile Options] tab When [No] in the [Build simultaneously] property is selected, the include paths are added by the following procedure. - Paths specified in the [Additional include paths] property from this tab - Paths specified in the [Additional include paths] property from the [Assemble Options] tab - Paths displayed in the [System include paths] property from the [Assemble Options] tab This corresponds to the -I option of the cx command.		
	Default	Yes	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Assembles using the include path specified in the property of the build tool to be used.
		No	Does not use the include path specified in the property of the build tool to be used.
Macro definition	Specify the name of the macro to be defined. Specify in the format of "*macro name=defined value*", with one macro name per line. The "=*defined value*" part can be omitted, and in this case, "1" is used as the defined value. This corresponds to the -D option of the cx command. The specified macro is displayed as the subproperty.		
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected *Configuration of the compile option* - Other than above *Configuration of the assemble option*	
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.	
	Restriction	Up to 256 characters Up to 256 items can be specified.	

Macro undefinition	Specify the macro name to be undefined. Specify in the format of "*macro name*", with one macro name per line. This corresponds to the -U option of the cx command. The specified macro is displayed as the subproperty.	
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected *Configuration of the compile option* - Other than above *Configuration of the assemble option*
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 256 characters Up to 256 items can be specified.

(3) [Character Encoding]

The detailed information on character encoding is displayed and the configuration can be changed.

Character encoding	Select the character code to be used for Japanese comments and character strings in the source file. This corresponds to the -Xcharacter_set option of the cx command.		
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected *Configuration of the compile option* - Other than above *Configuration of the assemble option*	
	How to change	Select from the drop-down list.	
	Restriction	SJIS(None)	Interprets the Japanese character code in the source file as SJIS.
		EUC(-Xcharacter_set=euc_jp)	Interprets the Japanese character code in the source file as EUC.
		UFT-8(-Xcharacter_set=utf8)	Interprets the Japanese character code in the source file as UFT-8.
		Big5(-Xcharacter_set=big5)	Interprets the Chinese character code in the source file as Traditional Chinese.
		GB2312(-Xcharacter_set=gb2312)	Interprets the Chinese character code in the source file as Simplified Chinese.
		No-process(-Xcharacter_set=none)	Does not process the Japanese character code in the source file.

(4) [Output Code]

The detailed information on output codes is displayed and the configuration can be changed.

Use 32-bit branch instruction	Select whether to use the far jump function for the jarl and jr instructions. By using the far jump function, it is assumed that the jarl and jr instructions are jarl32 and jr32 instructions, and assembling is performed. This corresponds to the -Xasm_far_jump [V850E2][V850E2V3] option of the cx command. This property is displayed only for the project using a device with the V850E2 core and V850E2V3 architecture.		
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected *Configuration of the compile option* - Other than above *Configuration of the assemble option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xasm_far_jump)	Assumes that the jarl and jr instructions are jarl32 and jr32 instructions, and performs assembling.
		No	Performs assembly as a jarl or jr instruction.

(5) [Output File]

The detailed information on output files is displayed and the configuration can be changed.

Object module file name	Specify the name of the object module file generated after assembling. The extension other than ".obj" cannot be specified. If the extension is omitted, ".obj" is automatically added. If this is blank, the file name will be the source file name with the extension replaced by ".obj". This corresponds to the -o option of the cx command.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	Up to 259 characters

(6) [Assemble List]

The detailed information on the assemble list is displayed and the configuration can be changed.

Output assemble list file	Select whether to output the assemble list file. This corresponds to the -Xprn_path option of the cx command.		
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected *Configuration of the compile option* - Other than above *Configuration of the assemble option*	
	How to change	Select from the drop-down list.	
	Restriction	Yes(-Xprn_path)	Outputs the assemble list file.
		No	Does not output the assemble list file.

<table>
<tr><td rowspan="4">Output folder for assemble list file</td><td colspan="2">Specify the folder which the assemble list file is output.
The assemble list file is output under the source file name with the extension replaced by ".prn".
If a relative path is specified, the reference point of the path is the main project or subproject folder.
If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different).
The following macro name is available as an embedded macro.
%BuildModeName%: Replaces with the build mode name.
If this is blank, it is assumed that the project folder has been specified.
This corresponds to the -Xprn_path option of the cx command.
This property is displayed only when [Yes(-Xprn_path)] in the [Output assemble list file] property is selected.</td></tr>
<tr><td>Default</td><td>- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected
<i>Configuration of the compile option</i>
- Other than above
<i>Configuration of the assemble option</i></td></tr>
<tr><td>How to change</td><td>Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.</td></tr>
<tr><td>Restriction</td><td>Up to 247 characters</td></tr>
</table>

(7) [Error Output]

The detailed information on the error output is displayed and the configuration can be changed.

<table>
<tr><td rowspan="5">Output error message file</td><td colspan="3">Select whether to output the error message file to the Output panel.
This corresponds to the -Xerror_file option of the cx command.</td></tr>
<tr><td>Default</td><td colspan="2"><i>Configuration of the common option</i></td></tr>
<tr><td>How to change</td><td colspan="2">Select from the drop-down list.</td></tr>
<tr><td rowspan="2">Restriction</td><td>Yes(-Xerror_file)</td><td>Outputs the error message file.</td></tr>
<tr><td>No</td><td>Does not output the error message file.</td></tr>
<tr><td rowspan="4">Error message file output folder</td><td colspan="3">Specify the folder which the error message file is output.
If a relative path is specified, the reference point of the path is the main project or subproject folder.
If an absolute path is specified, the reference point of the path is the main project or subproject folder (unless the drives are different).
The following macro name is available as an embedded macro.
%BuildModeName%: Replaces with the build mode name.
If this is blank, it is assumed that the project folder has been specified.
This corresponds to the -Xerror_file option of the cx command.
This property is displayed only when [Yes(-Xerror_file)] in the [Output error message file] property is selected.</td></tr>
<tr><td>Default</td><td colspan="2"><i>Configuration of the common option</i></td></tr>
<tr><td>How to change</td><td colspan="2">Directly enter in the text box or edit by the Browse For Folder dialog box which appears when clicking the [...] button.</td></tr>
<tr><td>Restriction</td><td colspan="2">Up to 247 characters</td></tr>
</table>

Error message file name	Specify the error message file name. The extension can be freely specified. The following macro name is available as an embedded macro. %ProjectName%: Replaces with the project name. If this is blank, it is assumed that "%ProjectName%.err" has been specified. This corresponds to the -Xerror_file option of the cx command. This property is displayed only when [Yes(-Xerror_file)] in the [Output error message file] property is selected.	
	Default	*Configuration of the common option*
	How to change	Directly enter in the text box.
	Restriction	Up to 259 characters

(8) [Warning Message]

The detailed information on warning messages is displayed and the configuration can be changed.

Displayed warning message	Specify the warning message number to be displayed regardless of the warning level. If multiple message numbers are specified, delimit them with "," (comma) (example: 02042,02107). Also, the range can be set using "-" (hyphen) (example: 02222-02554,02699-02782). If the same number is specified on the [Undisplayed warning message] property and this property, the number specified on this property takes precedence. This corresponds to the -Xwarning option of the cx command.	
	Default	*Configuration of the common option*
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 2048 characters
Undisplayed warning message	Specify the warning message number to not be displayed regardless of the warning level. If multiple message numbers are specified, delimit them with "," (comma) (example: 02042,02107). Also, the range can be set using "-" (hyphen) (example: 02222-02554,02699-02782). If the same number is specified on the [Displayed warning message] property and this property, the number specified on the [Displayed warning message] property takes precedence. This corresponds to the -Xno_warning option of the cx command.	
	Default	*Configuration of the common option*
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 2048 characters

(9) [Others]

Other detailed information on assembly is displayed and the configuration can be changed.

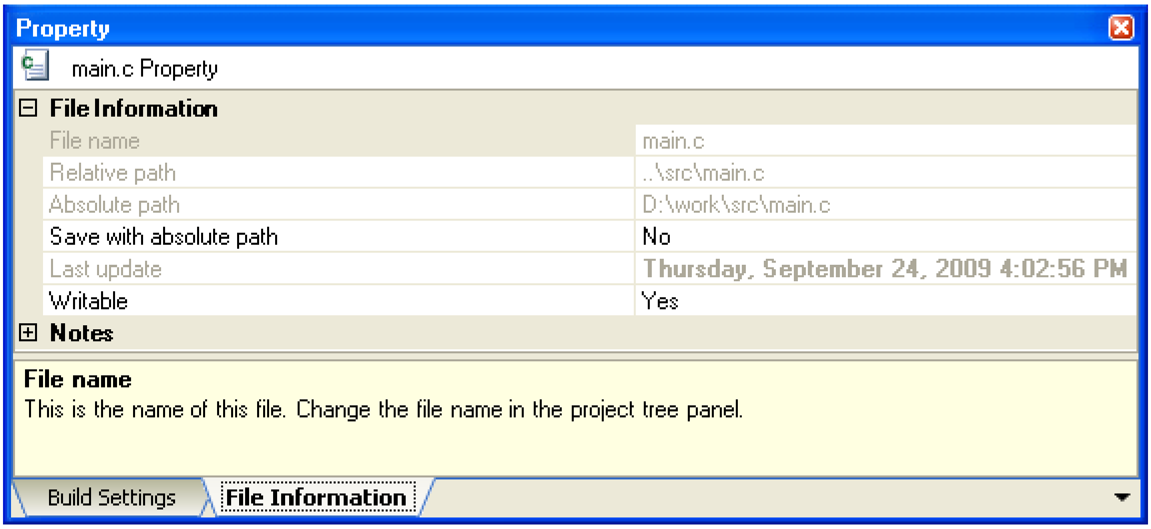
Commands executed before assemble processing	Specify the command to be executed before assemble processing. The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %InputFile%: Replaces with the absolute path of the file to be assembled. %AssembledFile%: Replaces with the absolute path of the output file under assembling. The specified command is displayed as the subproperty.	
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected Blank - Other than above *Configuration of the assemble option*
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.
Commands executed after assemble processing	Specify the command to be executed after assemble processing. The following macro names are available as embedded macros. %ProjectFolder%: Replaces with the absolute path of the project folder. %OutputFolder%: Replaces with the absolute path of the output folder. %OutputFile%: Replaces with the absolute path of the output file. %InputFile%: Replaces with the absolute path of the file to be assembled. %AssembledFile%: Replaces with the absolute path of the output file under assembling. The specified command is displayed as the subproperty.	
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected Blank - Other than above *Configuration of the assemble option*
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 1023 characters Up to 64 items can be specified.
Other additional options	Input the assemble option to be added additionally. The options set here are added at the end of the assemble options group.	
	Default	- When [Yes] in the [Build simultaneously] property in the [Build Method] category from the [Common Options] tab is selected Blank - Other than above *Configuration of the assemble option*
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 259 characters

[File Information] tab

This tab shows the detailed information on each file categorized by the following and the configuration can be changed.

(1) [File Information]
(2) [Notes]

Figure A-19. Property Panel: [File Information] Tab



File name	main.c
Relative path	..\src\main.c
Absolute path	D:\work\src\main.c
Save with absolute path	No
Last update	Thursday, September 24, 2009 4:02:56 PM
Writable	Yes

File name
This is the name of this file. Change the file name in the project tree panel.

[Description of each category]

(1) [File Information]

The detailed information on the file is displayed and the configuration can be changed.

File name	The file name is displayed. Change the file name on the project tree.		
	Default	*file name*	
	How to change	Changes not allowed	
Relative path	The relative path of the file from the project folder is displayed.		
	Default	*The relative path of the file from the project folder*	
	How to change	Changes not allowed	
Absolute path	The absolute path of the file is displayed.		
	Default	*The absolute path of the file*	
	How to change	Changes not allowed	
Save with absolute path	Select whether to save the file location with the absolute path.		
	Default	No	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Saves the file location with the absolute path.
		No	Saves the file location with the relative path.

Last update	The time and date on which this file was changed last is displayed.		
	Default	*File updated time and date*	
	How to change	Changes not allowed	
Writable	Select whether to enable writing to the file.		
	Default	Yes (when the file is write enabled) No (when the file is not write enabled)	
	How to change	Select from the drop-down list.	
	Restriction	Yes	Enables the file to write.
		No	Does not enable the file to write.

(2) [Notes]

The detailed information on notes is displayed and the configuration can be changed.

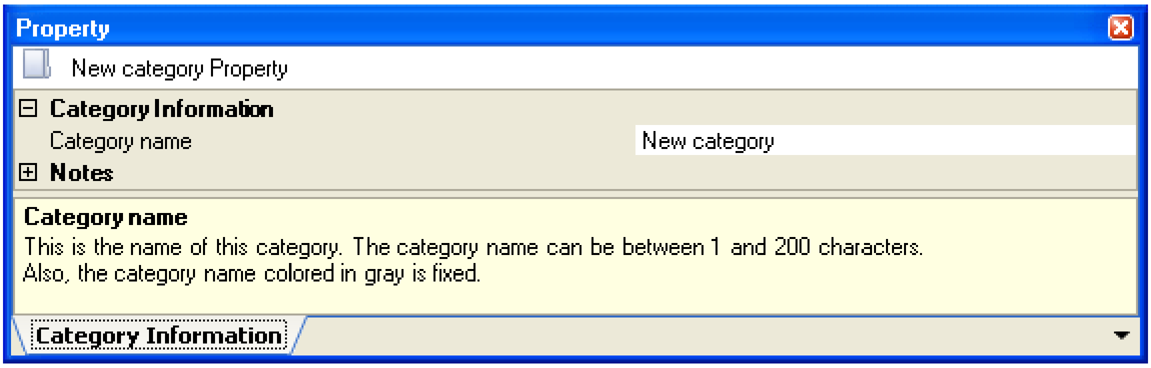
Memo	Add memos to the file. Add one item in one line. The specified memo is displayed as the subproperty.	
	Default	Memo[*number of items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 256 characters Up to 256 items can be specified.

[Category Information] tab

This tab shows the detailed information on the category node (the category that the user added), Files node, Build tool generated files node, and Startup node categorized by the following and the configuration can be changed.

(1) [Category Information]
(2) [Notes]

Figure A-20. Property Panel: [Category Information] Tab



Category name
This is the name of this category. The category name can be between 1 and 200 characters.
Also, the category name colored in gray is fixed.

[Description of each category]

(1) [Category Information]

The detailed information on the category is displayed and the configuration can be changed.

Category name	Specify the name of the category to categorize files. This property of the Files node, Build tool generated files node, and Startup node is displayed in gray and you cannot change the attribute.	
	Default	*Category name of files*
	How to change	Directly enter in the text box.
	Restriction	1 to 200 characters

(2) [Notes]

The detailed information on notes is displayed and the configuration can be changed.
This category of the Files node, Build tool generated files node, and Startup node is not displayed.

Memo	Add memos to the category of files. Add one item in one line. The specified memo is displayed as the subproperty.	
	Default	Memo[*number of items*]
	How to change	Edit by the Text Edit dialog box which appears when clicking the [...] button. For the subproperty, you can enter directly in the text box.
	Restriction	Up to 256 characters Up to 256 items can be specified.

Editor panel

This panel is used to display and edit text files and source files.

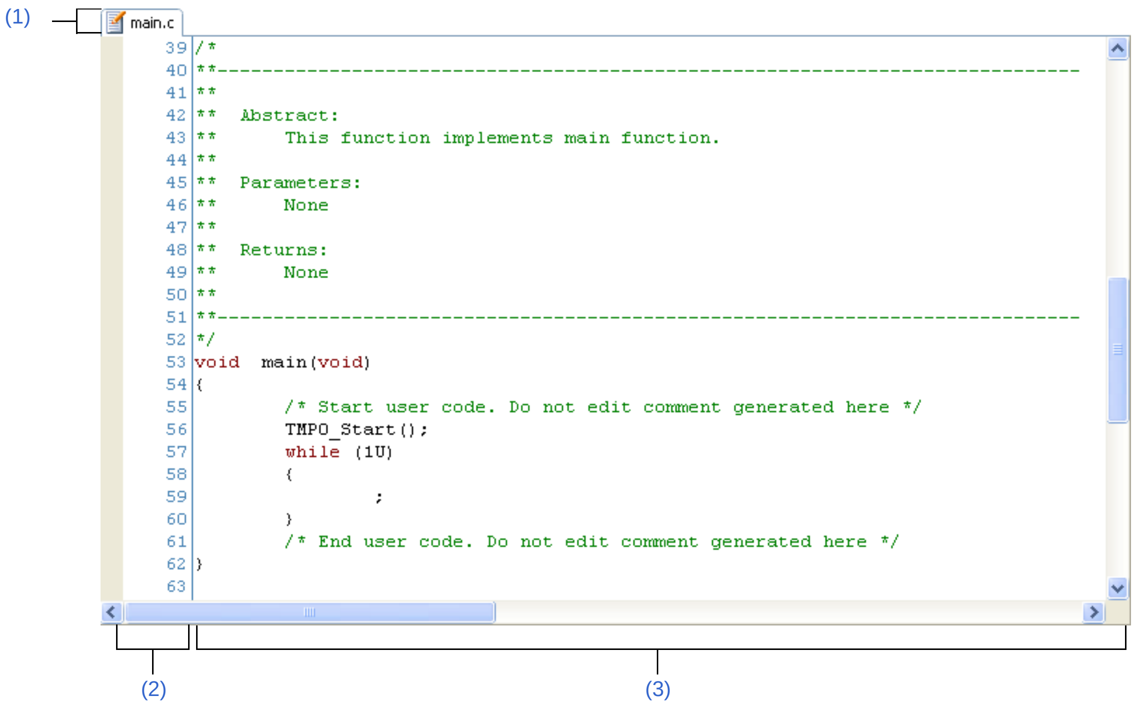
The file is opened by automatically distinguishing the encoding (Shift_JIS/EUC-JP/UTF-8) and line feed code of the file and the encoding is retained when it is saved.

If the encoding and newline code is specified in the File Save Settings dialog box, however, then the file is saved in accordance with those settings.

This panel can be multiply opened (max:100 panels).

Remark A message is shown when the downloaded load module file is older than the source file to be opened.

Figure A-21. Editor Panel



```
/*
**-------------------------------------------------------------------------------
**
**  Abstract:
**      This function implements main function.
**
**  Parameters:
**      None
**
**  Returns:
**      None
**
**-------------------------------------------------------------------------------
*/
void  main(void)
{
        /* Start user code. Do not edit comment generated here */
        TMP0_Start();
        while (1U)
        {
                ;
        }
        /* End user code. Do not edit comment generated here */
}
```

The following items are explained here.

- [How to open]
- [Description of each area]
- [[File] menu (only available for the Editor panel)]
- [[Edit] menu (only available for the Editor panel)]
- [Context menu]

[How to open]

- On the Project Tree panel, double click a file.
- On the Project Tree panel, select a source file, and then select [Open] from the context menu.
- On the Project Tree panel, select a file and then select [Open with Internal Editor...] from the context menu.
- On the Project Tree panel, select [Add] >> [Add New File...] from the context menu, and then create a text file or source file.

[Description of each area]

(1) Title bar

The name of the open text file or 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.
(Uneditable)	The opened text file is write disabled.
ID number	The same text file is multiply opened.

(2) Line number area

This area displays the line number of the opened text file or source file.

(3) Characters area

This area displays character strings of text files and source files and you can edit it.

This area has the following functions.

(a) Character editing

Characters can be entered from the keyboard.

Various shortcut keys can be used to enhance the edit function.

(b) File monitor

The following function for monitoring is provided to manage source files.

- 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.

Remark The following items can be customized by setting the Option dialog box.

- Display fonts
- Tab interval
- Display, hide, and colors of control characters (control codes including a blank symbol)
- Colors of reserved words and comments

[[File] menu (only available for the Editor panel)]

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 the Editor panel. When the contents of the panel have not been saved, a confirmation message is shown.
Save *file name*	Overwrites the contents of the currently editing the 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...].
file name Save Settings...	This dialog box is used to open the File Save Settings dialog box to set the encoding and newline code of the file that is editing on this panel.

Save *file name* As...	Opens the Save As dialog box to newly save the contents of the currently editing the Editor panel.
Page Setup...	Opens the Page Setup dialog box of Windows.
Print...	Opens the Print dialog box of Windows for printing the contents of the currently editing the Editor panel.

[[Edit] menu (only available for the Editor panel)]

The following items are exclusive for the [Edit] menu in the Editor panel (other items are all invalid).

Undo	Cancels the previous operation on the Editor panel and restores the characters and the caret position (max 100 times).
Redo	Cancels the previous [Undo] operation on the Editor panel and restores the characters and the caret position.
Cut	Cuts the selected characters and copies them to the clip board.
Copy	Copies the selected characters to the clipboard.
Paste	Insert (insert mode) or overwrite (overwrite mode) the characters that are copied on the clip board into the caret position. When the contents of the clipboard are not recognized as characters, the operation is invalid.
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 the beginning to the end in the currently editing text file.
Find...	Opens the Search and Replace dialog box with the [Quick Search] tab target. When there is a selection area, search is only taken place in the selection area.
Replace...	Opens the Search and Replace dialog box with the [Quick Replace] tab target. When there is a selection area, replace is only taken place in the selection area.
Move To...	Opens the Go to the Location dialog box to move the caret to the designated line.

[Context menu]

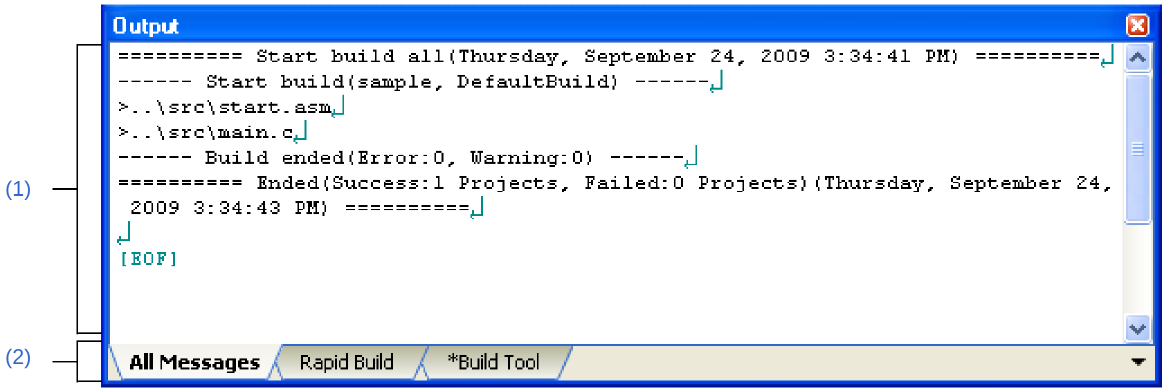
[Characters area/Line number area]

Jump To Function	Jumps to the function regarding the selected characters and the words at the caret position as a function name, variable name, or label in C language. If a single line contains multiple statements, then it may not be possible to jump to the correct location.
Back To Last Cusor Position	Goes back to the position before the cusor is jumped.
Forward To Next Cusor Position	Forwards to the position before operating [Back To Last Cusor Position].
Tag Jump	Jumps to the caret line in the editor indicated by the message (file, line, and column).
Cut	Cuts the selected characters and copies them to the clip board.
Copy	Copies the selected characters to the clip board.
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.

Output panel

This panel is used to display the message that is output from the build tool.
Messages are shown individually on the tab categorized by the output tool.

Figure A-22. Output Panel



```
========== Start build all(Thursday, September 24, 2009 3:34:41 PM) ==========↲
------ Start build(sample, DefaultBuild) ------↲
>..\src\start.asm↲
>..\src\main.c↲
------ Build ended(Error:0, Warning:0) ------↲
========== Ended(Success:1 Projects, Failed:0 Projects)(Thursday, September 24,
 2009 3:34:43 PM) ==========↲
↲
[EOF]
```

The following items are explained here.

- [How to open]
- [Description of each area]
- [[File] menu (only available for the Output panel)]
- [[Edit] menu (only available for the Output panel)]
- [Context menu]

[How to open]

- From the [View] menu, select [Output].

[Description of each area]

(1) Message area

This area displays messages output from each tool.
In build result display, a new message is displayed deleting the previous message every time build is done (but not the [All Messages] tab).

Remark Up to 500000 lines of messages can be displayed.
If 500001 lines or more of messages are output, then the excess lines are deleted, oldest first.

The message colors differ as follows depends on the type of the output message (the character color/background color is set in the [General - Font and Color] category in the Option dialog box).

Message Type	Example (Default)			Description
Normal message	AaBbCc	Character color	Black	Information on something.
		Background color	White	
Warning message	AaBbCc	Character color	Blue	Warning for the operation.
		Background color	Normal color	

Message Type	Example (Default)			Description
Error message	AaBbCc	Character color	Red	Fatal error or operation disabled because of an error in operation.
		Background color	Light gray	

This area has the following functions.

(a) Tag jump

When the output message is double-clicked, or the [Enter] key is pressed with the caret over the message, the Editor panel appears and the destination line number of the file is displayed.

You can jump to the line of the source file that generated the error from the error message output when building.

(b) Display help

If you select the [Help for Message] from the context menu or press the [F1] key when the cursor is on a line displaying the warning or error message, the help related to that line's message is displayed.

(c) Save log

The contents displayed on the currently selected tab can be saved in a text file (*.txt) by selecting [Save Output - *tab name* As...] from the [File] menu to open the Save As dialog box (messages on the tab that is not selected will not be saved).

(2) Tab selection area

Select tabs that messages are output from.

Tabs that are displayed are as follows.

Tab Name	Description
All Messages	Displays all the messages by order of output (except while executing a rapid build).
Rapid build	Displays the message output from the build tool by running a rapid build.
Build Tool	Displays the message output from the build tool by running a build, rebuild, or batch build.

Caution Even if a new message is output on a deselected tab, tab selection will not automatically switch. In this case, a ※ mark will be added in front of the tab name, indicating that a new message has been output.

[[File] menu (only available for the Output panel)]

The following items are exclusive for the [File] menu in the Output panel (other items are common to all the panels).

Save Output - *tab name*	Saves the contents on the currently selecting tab in the previously saved text file (*.txt) (see "(c) Save log"). When this item is selected for the first time after launching the program, the operation is equivalent to when selecting [Save Output - *tab name* As...].
Save Output - *tab name* As...	Opens the Save As dialog box to save the contents on the currently selecting tab in the designated text file (*.txt) (see "(c) Save log").

[[Edit] menu (only available for the Output panel)]

The following items are exclusive for the [Edit] menu in the Output panel (other items are all invalid).

Copy	Copies the selected characters to the clipboard.
Select All	Selects all the messages displayed on this panel.
Find...	Opens the Search and Replace dialog box with the [Quick Search] tab target.
Replace...	Opens the Search and Replace dialog box with the [Whole Replace] tab target.

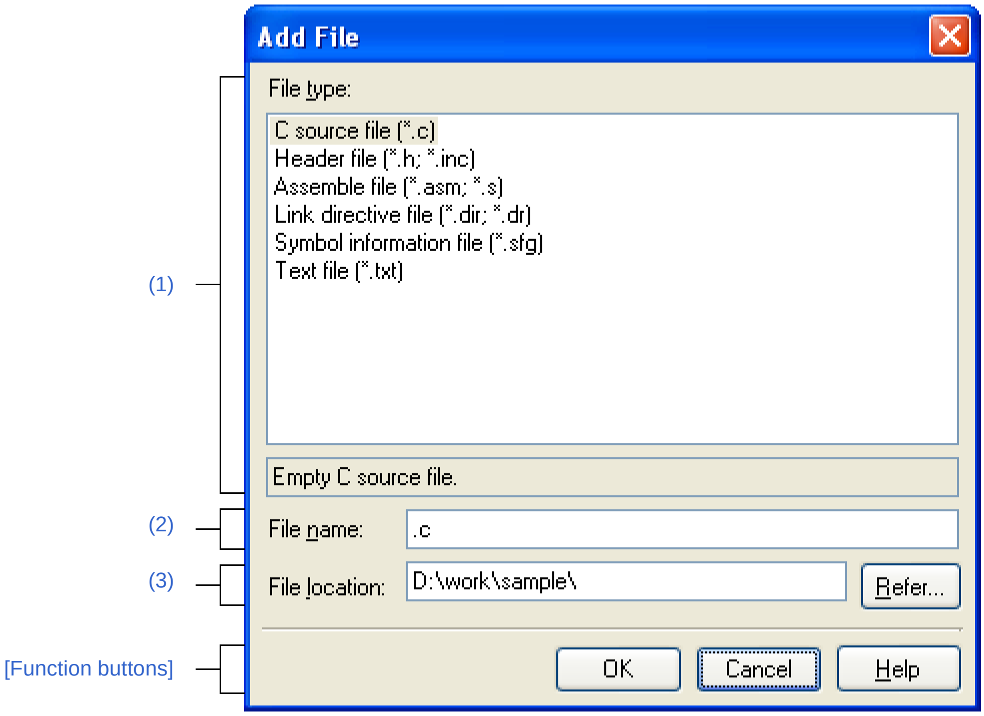
[Context menu]

Copy	Copies the selected characters to the clipboard.
Select All	Selects all the messages displayed on this panel.
Clear	Deletes all the messages displayed on this panel.
Tag jump	If there is information of the file name, line, and column on the caret line, jumps to that location.
Help for Message	Displays the help related to the message on the current caret position. This only applies to warning messages and error messages.

Add File dialog box

This dialog box is used to create a new file and add it to the project.

Figure A-23. Add File Dialog Box



C source file (*.c)
Header file (*.h; *.inc)
Assemble file (*.asm; *.s)
Link directive file (*.dir; *.dr)
Symbol information file (*.sfg)
Text file (*.txt)

Empty C source file.

File name: .c

File location: D:\work\sample\

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- From the [File] menu, select [Add] >> [Add New File...].
- On the Project Tree panel, select either one of the Project node, Subproject node, File node, or Category node, and then select [Add] >> [Add New File...] from the context menu.

[Description of each area]

(1) [File type] area

Select the type of the file to be created.

When the file type is selected, the description is displayed at the lower box.

The file types to be displayed are as follows.

- C source file (*.c)
- Header file (*.h; *.inc)
- Assemble file (*.asm; *.s)
- Link directive file (*.dir; *.dr)
- Symbol information file (*.sfg)
- Text file (*.txt)

(2) [File name] area

Directly enter the name of the file to be created.
".txt" is displayed by default.

Remark If any extension is not designated, the one selected in the [File type] area will be added. Also, if the extension different from the one selected in the [File type] area is designated, the one selected in the [File type] area will be added (for example, if you designate "aaa.txt" as the file name and select "C source file (*.c)" as the file type, the file is named as "aaa.txt.c").

(3) [File location] area

Designate the location to be created the file by directly entering its path or selecting from the [Refer...] button.
The path of the project folder is displayed by default.

(a) Button

Refer···	Opens the Browse For Folder dialog box. If a folder is selected, the path will be added in the text box.

Remarks 1. If the text box is blank, it is assumed that the project folder is designated.
2. If the relative path is designated, the reference point of the path is the project folder.

Remark Up to 259 characters (path and file name combined) can be specified in the [File name] area and [File location] area. When the input violates any restriction, the following messages will be shown on the [File name] area in the tooltip.

Message	Description
The file name including the path is too long. Make it within 259 characters.	The file name with the path is more than 259 characters.
The specified path contains a folder that does not exist.	The path contains a folder that does not exist.
The file name or path name is invalid. The following characters cannot be used: \, /, :, *, ?, ", <, >, |	The file name with the invalid path is designated. The following characters cannot be used for the file name and folder name: \, /, :, *, ?, ", <, >, |

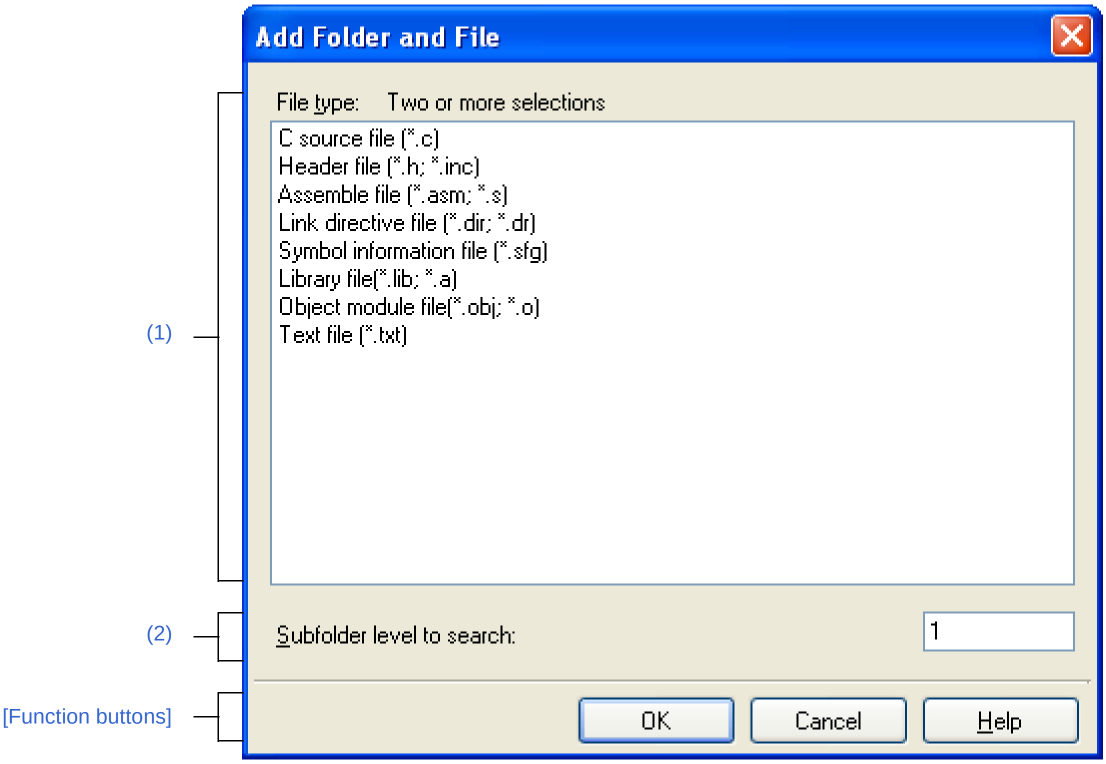
[Function buttons]

Button	Function
OK	Creates the file with the entered file name, adds it to the project, and opens with the Editor panel. And then closes this dialog box.
Cancel	Does not create a file and closes this dialog box.
Help	Displays the help of this dialog box.

Add Folder And File dialog box

This dialog box is used to add existing files and folder hierarchies to the project.
The folder is added as a category.

Figure A-24. Add Folder and File Dialog Box



Add Folder and File

File type: Two or more selections

C source file (*.c)
Header file (*.h; *.inc)
Assemble file (*.asm; *.s)
Link directive file (*.dir; *.dr)
Symbol information file (*.sfg)
Library file(*.lib; *.a)
Object module file(*.obj; *.o)
Text file (*.txt)

Subfolder level to search: 1

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- Drag the folder from Explorer or the like, and drop it on the Project Tree panel.

[Description of each area]

(1) [File type] area

Select the type of the file to be added to the project.
You can select multiple file types by left clicking while holding down the [Ctrl] or [Shift] key.
If nothing is selected, it is assumed that all types are selected.
The file types to be displayed are as follows.

- C source file (*.c)
- Header file (*.h; *.inc)
- Assemble file (*.asm; *.s)
- Link directive file (*.dir; *.dr)
- Symbol information file (*.sfg)

- Library file (*.lib; *.a)
- Object module file (*.obj; *.o)
- Text file (*.txt)

(2) [Subfolder level to search] area

Directly enter the number of levels of the subfolder to be added to the project.
"1" is displayed by default.

Remark Up to 10 (decimal number) can be specified.
When the input violates any restriction, the following messages will be shown in the tooltip.

Message	Description
Fewer than 0 or more than 10 values cannot be specified.	More than 10 subfolder levels have been specified.
Specify in decimal.	A number in other than decimal or a string has been specified.

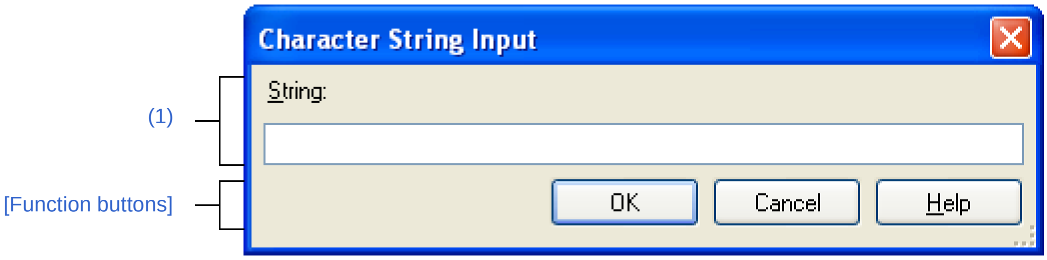
[Function buttons]

Button	Function
OK	Adds the folder that was dragged and dropped and the files in that folder to the project. And then closes this dialog box.
Cancel	Does not add the folder and files, and then closes this dialog box.
Help	Displays the help of this dialog box.

Character String Input dialog box

This dialog box is used to input and edit characters in one line.

Figure A-25. Character String Input Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Property panel, select the following properties, and then click the [...] button.
  - From the [Common Options] tab, [Displayed warning message] and [Undisplayed warning message] in the [Warning Message] category, [Format of build option list] and [Other additional options] property in the [Others] category
  - From the [Compile Options] tab, [Other additional options] in the [Others] category
  - From the [Assemble Options] tab, [Other additional options] in the [Others] category
  - From the [Link Options] tab, [Entry symbol] and [Other additional options] in the [Others] category
  - From the [ROMize Options] tab, [Start symbol of rompsec section] in the [Section] category
  - From the [Create Library Options] tab, [Other additional options] in the [Others] category
  - From the [Individual Compile Options] tab, [Other additional options] in the [Others] category
  - From the [Individual Assemble Options] tab, [Other additional options] in the [Others] category
- In the Link Directive File Generation dialog box, select a segment or section in the [Segment / Section list] area, and then click the [...] button on [Name] in the [Segment / Section detail] area.
- In the Link Directive File Generation dialog box, select a section in the [Segment / Section list] area, and then click the [...] button on [Input section name] in the [Segment / Section detail] area.
- In the Link Directive File Generation dialog box, select a symbol in the [Symbol list] area, and then click the [...] button on [Segment name] in the [Symbol detail] area.
- In the Link Directive File Generation dialog box, select a symbol in the [Symbol list] area, and then click the [...] button on [Segment name] in the [Base symbol name] area.

[Description of each area]

(1) [String] area

Input characters in one line.

By default, the current value of the area that this dialog box is called from is reflected to this area.

You cannot start a new line.

Remark Up to 32767 characters can be entered.
When the input violates any restriction, the following messages will be shown in the tooltip.

Message	Description
More than *maximum number of restriction in the property that called this dialog box* characters cannot be specified.	The numbers of input characters exceeds the maximum number of restriction in the property that called this dialog box.

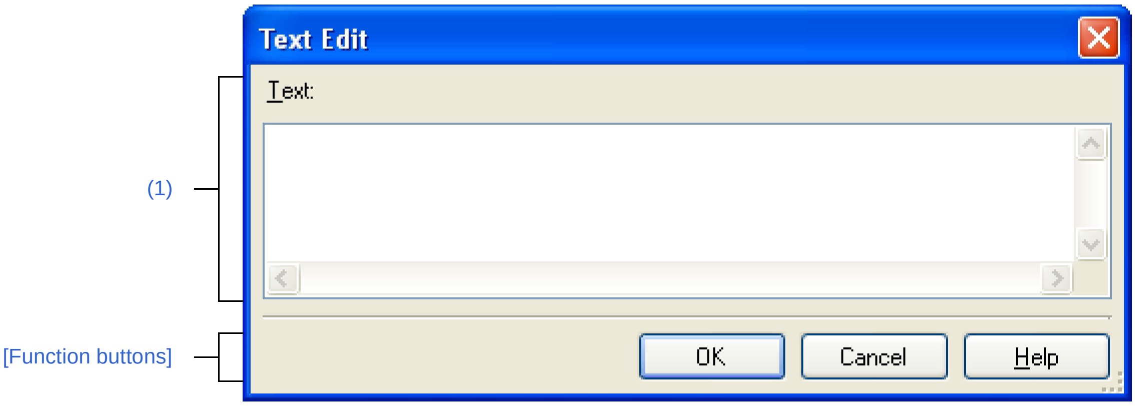
[Function buttons]

Button	Function
OK	Reflects the entered characters to the area that this dialog box is called from and closes this dialog box.
Cancel	Does not reflect the entered characters to the area that this dialog box is called from and closes this dialog box.
Help	Displays the help of this dialog box.

Text Edit dialog box

This dialog box is used to input and edit texts in multiple lines.

Figure A-26. Text Edit Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Property panel, select the following properties, and then click the [...] button.
  - From the [Common Options] tab, [Macro definition] in the [Frequently Used Options(for Compile)] category, [Macro definition] in the [Frequently Used Options(for Assemble)] category, [Using libraries] in the [Frequently Used Options(for Link)] category, [Memo] in the [Notes] category, and [Commands executed before build processing], [Commands executed after build processing] in the [Others] category
  - From the [Compile Options] tab, [Macro definition], [Macro undefinition] in the [Preprocess] category, and [Commands executed before compile processing], [Commands executed after compile processing] in the [Others] category
  - From the [Assemble Options] tab, [Macro definition] in the [Preprocess] category, and [Commands executed before assemble processing], [Commands executed after assemble processing] in the [Others] category
  - From the [Link Options] tab, [Using libraries] in the [Library] category, and [Commands executed before link processing], [Commands executed after link processing] in the [Others] category
  - From the [ROMize Options] tab, [Data sections included in rompsec section], [Text sections included in rompsec section] in the [Section] category
  - From the [Hex Output Options] tab, [Converted sections] in the [Hex Format] category
  - From the [Create Library Options] tab, [Commands executed before library generate processing], [Commands executed after library generate processing] in the [Others] category
  - From the [Individual Compile Options] tab, [Macro definition], [Macro undefinition] in the [Preprocess] category, and [Commands executed before compile processing], [Commands executed after compile processing] in the [Others] category
  - From the [Individual Assemble Options] tab, [Macro definition] in the [Preprocess] category, and [Commands executed before assemble processing], [Commands executed after assemble processing] in the [Others] category
  - From the [File Information] tab, [Memo] in the [Notes] category
  - From the [Category Information] tab, [Memo] in the [Notes] category

[Description of each area]

(1) [Text] area

Edit texts in multiple lines.

By default, the current value of the area that this dialog box is called from is reflected in this area.

Remark Up to 65535 lines and 65535 characters can be entered.
When the input violates any restriction, the following message will be shown in the tooltip.

Message	Description
More than *maximum number of restriction in the property that called this dialog box* characters cannot be specified. The current number of characters is displayed between brackets at the beginning of the line in excess of the limit.	The numbers of input characters exceeds the maximum number of restriction in the property that called this dialog box.

[Function buttons]

Button	Function
OK	Reflects the entered text to the text box that opened this dialog box and closes this dialog box.
Cancel	Does not reflect the entered text to the text box that opened this dialog box and closes this dialog box.
Help	Displays the help of this dialog box.

Path Edit dialog box

This dialog box is used to edit or add the path or file name including the path.

Figure A-27. Path Edit Dialog Box (When Editing Path)

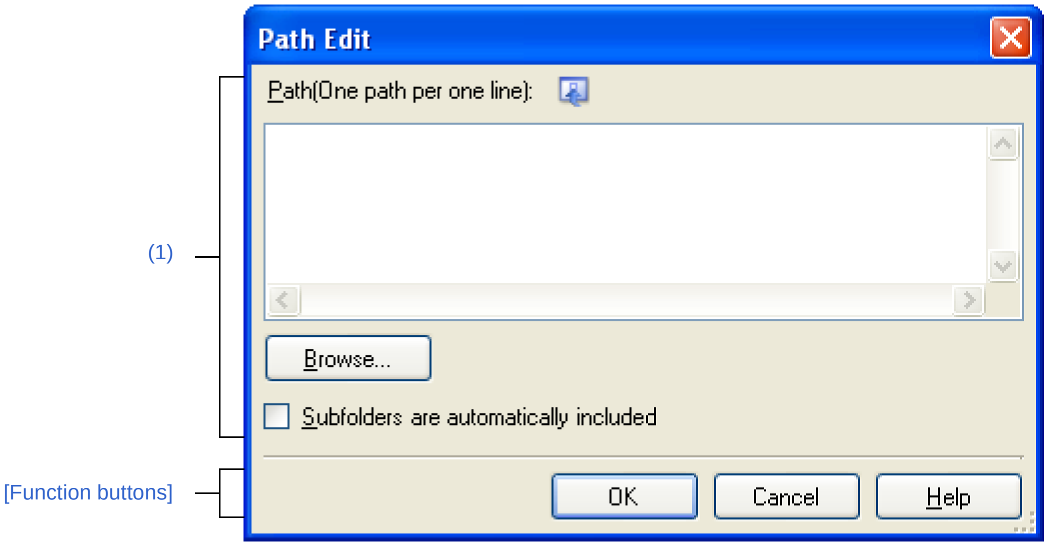
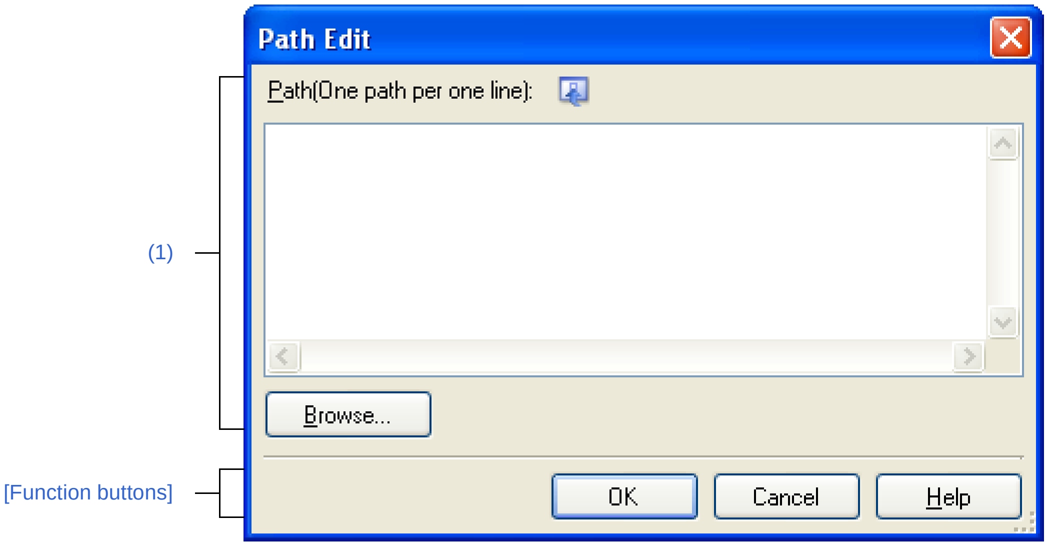

Path Edit
Path(One path per one line):
Browse...
□ Subfolders are automatically included
OK
Cancel
Help
(1)
[Function buttons]

Figure A-28. Path Edit Dialog Box (When Editing File Name Including Path)



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Property panel, select the following properties, and then click the [...] button.
  - From the [Common Options] tab, [Additional include paths] in the [Frequently Used Options(for Compile)] category, [Additional include paths] in the [Frequently Used Options(for Assemble)] category, and [Additional library paths] in the [Frequently Used Options(for Link)] category
  - From the [Compile Options] tab, [Additional include paths] in the [Preprocess] category, and [Far Jump file names] in the [Output Code] category
  - From the [Assemble Options] tab, [Additional include paths] in the [Preprocess] category
  - From the [Link Options] tab, [Additional library paths] in the [Library] category
  - From the [Individual Compile Options] tab, [Additional include paths] in the [Preprocess] category
  - From the [Individual Assemble Options] tab, [Additional include paths] in the [Preprocess] category

[Description of each area]

(1) Path edit area

Edit or add the path or file name including the path.

(a) [Path(One path per one line)]

Edit or add the path or file name including the path by directly entering it.
The path or file name including the path can be designated in multiple lines. Designate the path or file name including the path at a line.
By default, the current contents of the text box that opened this dialog box are reflected in this area.

The path can be added by one of the following methods.

- Click the [Browse...] button, and then select the folder in the Browse For Folder dialog box.
- Drag and drop the folder using such as Explorer.

The file names including the path can be added by one of the following methods.

- Click the [Browse...] button, and then select the file in the Specify Far Jump File dialog box.
- Drag and drop the file using such as Explorer.

Caution **If an extremely long absolute path is specified as a relative path, an error could occur when clicking the [OK] button. In this case, designate the absolute path.**

Remark Up to 10000 lines can be entered. Up to the maximum characters that are limited by the Windows OS can be entered.
When the input violates any restriction, the following messages will be shown in the tooltip.

Message	Description
Specify a path.	The field is empty.
The path is too long. Specify a path with a number of characters equal to or fewer than *maximum number of restriction in the property that called this dialog box*.	The file name including the path exceeds the maximum number of restriction in the property that called this dialog box.
The specified path contains a folder that does not exist.	The path contains a folder that does not exist.
The file name or path name is invalid. The following characters cannot be used: \, /, :, *, ?, ", <, >, |	The file name with the invalid path is designated. The following characters cannot be used for the file name and folder name: \, /, :, *, ?, ", <, >, |

Message	Description
More than *maximum number of paths or files specified by the caller* lines cannot be specified.	The number of paths or files which have been entered exceeds the maximum number of restriction in the property that called this dialog box.

(b) Button

Browse...	- When adding the path Opens the Browse For Folder dialog box. If a folder is selected, the path will be added in [Path(One path per one line)]. - When adding the file name including the path Opens the Specify Far Jump File dialog box. If a file is selected, the file will be added in [Path(One path per one line)].

(c) [Subfolders are automatically included]

Select this check box and then click the [Browse...] button to specify the path. The path will be added, including subfolders, to [Path(One path per one line)] (up to five levels deep).

[Function buttons]

Button	Function
OK	Reflects the entered path to the text box that opened this dialog box and closes this dialog box.
Cancel	Does not reflect the entered path to the text box that opened this dialog box and closes this dialog box.
Help	Displays the help of this dialog box.

System Include Path Order dialog box

This dialog box is used to refer the system include paths specified for the compiler and set their specified sequence.

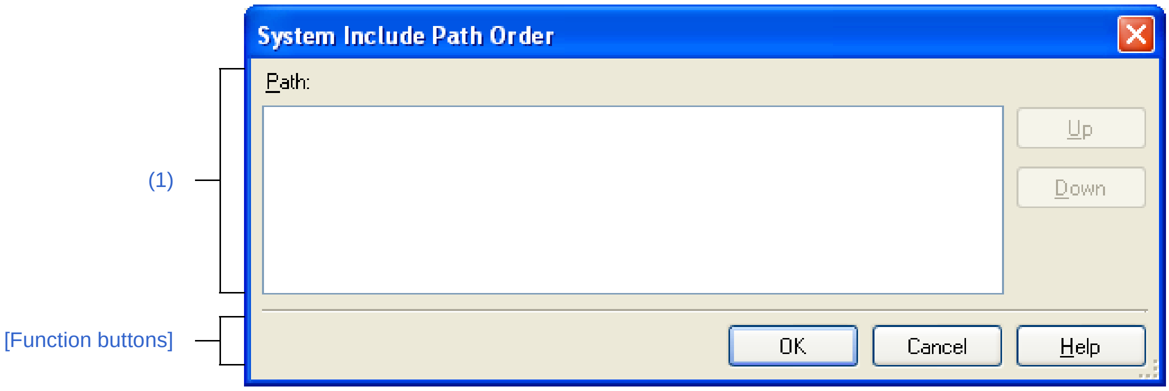


Figure A-29. System Include Path Order Dialog Box

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Property panel, select the following properties, and then click the [...] button.
  - From the [Common Options] tab, [System include paths] in the [Frequently Used Options(for Compile)] category, and [System include paths] in the [Frequently Used Options(for Assemble)] category
  - From the [Compile Options] tab, [System include paths] in the [Preprocess] category
  - From the [Assemble Options] tab, [System include paths] in the [Preprocess] category

[Description of each area]

(1) Path list display area

This area displays the list of the system include paths specified for the compiler.

(a) [Path]

This area displays the list of the system include paths in the specified sequence for the compiler.
The default order is the order that the files are registered to the project.
By changing the display order of the paths, you can set the specified order of the paths to the compiler.
To change the display order, use the [Up] and [Down] buttons, or drag and drop the path names.

Remarks 1. Move the mouse cursor over a file name to display a tooltip with the absolute path of that file.

2. Newly added system include paths are added next to the last path of the list.

3. When the path names are dragged and dropped, the multiple path names which are next to each other can be selected together.

(b) Button

Up	Moves the selected path to up.

Down	Moves the selected path to down.

Remark Note that above buttons are disabled when any path is not selected.

[Function buttons]

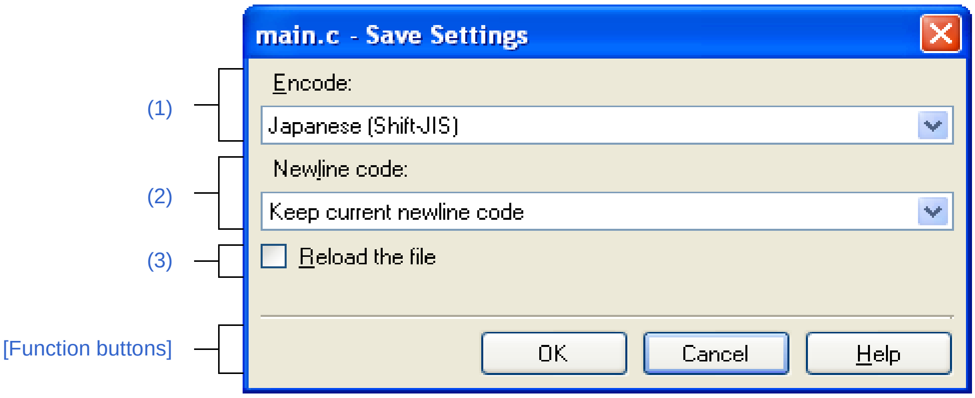
Button	Function
OK	Sets the specified order of the paths to the compiler as the display order in the Path list display area and closes this dialog box.
Cancel	Cancels the specified order of the paths and closes the dialog box.
Help	Displays the help of this dialog box.

File Save Settings dialog box

This dialog box is used to set the encoding and newline code of the file that is being edited on the Editor panel.

Remark The target file name is displayed on the title bar.

Figure A-30. File Save Settings Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- Focus the Editor panel, and then select [*file name* Save Settings...] from the [File] menu.

[Description of each area]

(1) [Encode]

Select the encoding to be set from the drop-down list.

The items of the drop-down list are displayed according to the following sequence.

Note that the same encoding and encoding which are not supported by the current OS will not be displayed.

- *Encoding of the current file* (default)
- *Default encoding of the current OS*
- *Encoding of code page 932 (SJIS)*
- *Encoding of code page 50222 (JIS)*
- *Encoding of code page 51932 (EUC)*
- *Encoding of code page 65001 (UTF8)*

(2) [Newline code]

Select the newline code to be set from the drop-down list.

You can select any of items below.

- Keep current newline code
- Windows (CR LF)
- Macintosh (CR)
- Unix (LF)

"Keep current newline code" is selected by default.
After the newline code is changed, the set newline code is selected by default.

(3) [Reload the file]

Use this check box to select whether to reload the file with the selected encoding and newline code when the [OK] button is clicked.
The check box is not selected by default.

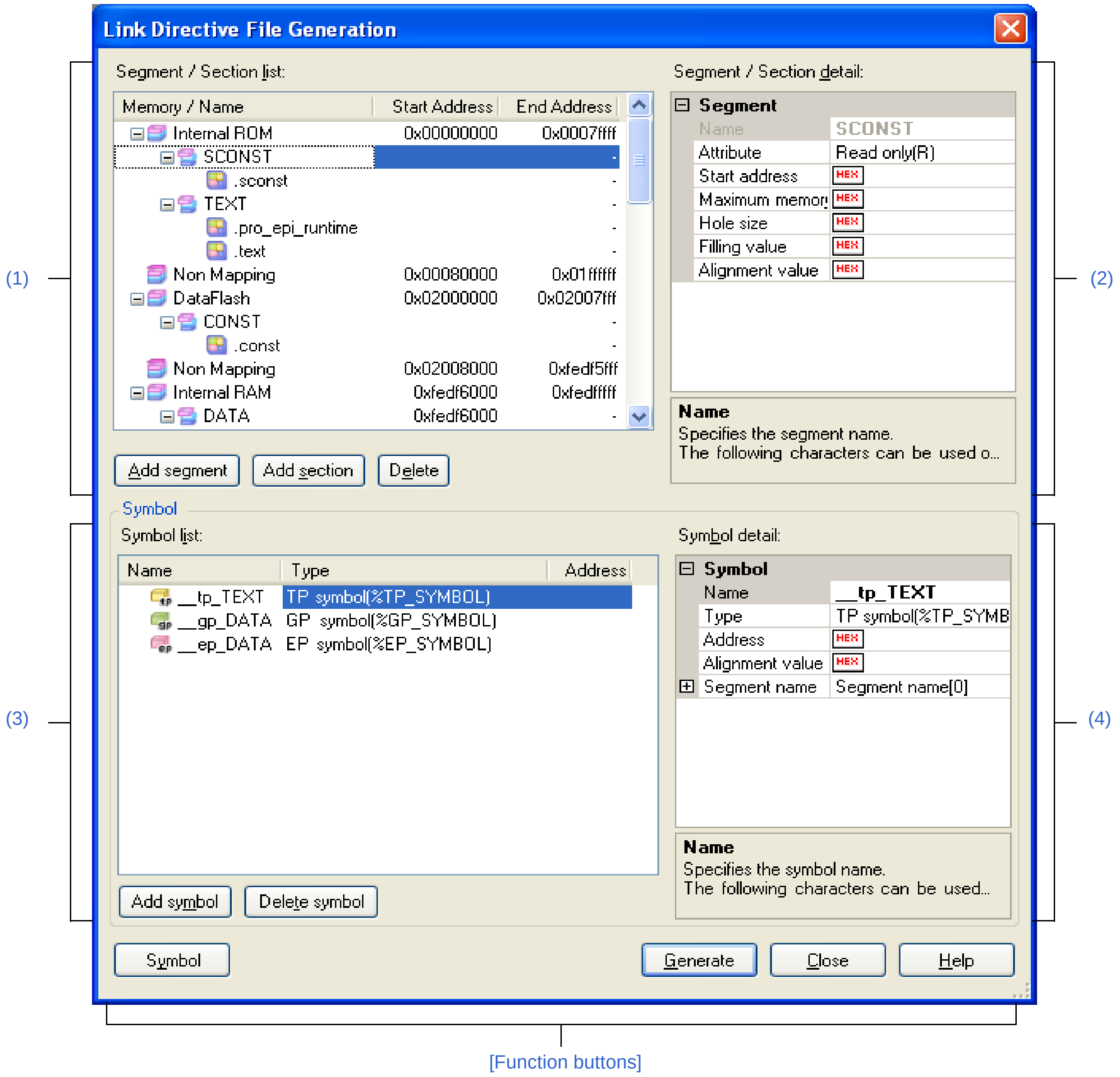
[Function buttons]

Button	Function
OK	Sets the selected encoding and newline code to the target file and closes this dialog box. If [Reload the file] is selected, sets the selected encoding and newline code to the target file and reloads the file.And then closes this dialog box.
Cancel	Cancels the settings of the encoding and newline code and closes the dialog box.
Help	Displays the help of this dialog box.

Link Directive File Generation dialog box

This dialog box is used to generate a link directive file based on the specified memory, segments, sections, and symbol allocation information.

Figure A-31. Link Directive File Generation Dialog Box



Link Directive File Generation

Segment / Section list:

Memory / Name	Start Address	End Address
Internal ROM	0x00000000	0x0007ffff
SCONST		-
.sconst		-
TEXT		-
.pro_epi_runtime		-
.text		-
Non Mapping	0x00080000	0x01ffffff
DataFlash	0x02000000	0x02007fff
CONST		-
.const		-
Non Mapping	0x02008000	0xfedf5fff
Internal RAM	0xfedf6000	0xfedfffff
DATA	0xfedf6000	-

Segment / Section detail:

Segment	
Name	SCONST
Attribute	Read only(R)
Start address	HEX
Maximum memor	HEX
Hole size	HEX
Filling value	HEX
Alignment value	HEX

Name
Specifies the segment name.
The following characters can be used o...

Symbol

Symbol list:

Name	Type	Address
__tp_TEXT	TP symbol(%TP_SYMBOL)	
__gp_DATA	GP symbol(%GP_SYMBOL)	
__ep_DATA	EP symbol(%EP_SYMBOL)	

Symbol detail:

Symbol	
Name	__tp_TEXT
Type	TP symbol(%TP_SYMB
Address	HEX
Alignment value	HEX
Segment name	Segment name[0]

Name
Specifies the symbol name.
The following characters can be used...

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Project Tree panel, select the Build tool node, and then select [Create Link Directive File...] from the context menu.

[Description of each area]

(1) [Segment / Section list] area

This area displays the device memory allocation information and a list of the currently configured segments and sections.

(a) [Memory / Name]

This area displays the names of the memory area, segments, and sections.
For the memory area, the name of the corresponding memory area as shown below is displayed.

- Internal ROM
- Non Mapping
- Internal RAM
- DataFlash

This item can be edited directly for the segments and sections.If a segment name and section name is changed, the value of [Name] in the [Segment / Section detail] area is also changed.

Caution Some segment and section names in reserved sections cannot be edited. See the remark of the [Segment / Section detail] area for details.

(b) [Start Address]

This area displays the start addresses of the memory area, segments, and sections.
This item can be edited directly for the segments and sections. If the start address is changed, the value of [Start Address] in the [Segment / Section detail] area is also changed.

(c) [End Address]

This area displays the end addresses of the memory area.
A dash (-) appears in segment and section rows.

(d) Button

Add segment	Adds a new segment directly below the row selected in the list. The segment name is "NewSegment_*XXX*" by default (*XXX*: 0 to 255 in decimal numbers). Make detailed segment settings in [Segment / Section detail] area. This button is invalid when a section row is selected or when 256 segments have been registered to the list.
Add section	Adds a new section directly below the row selected in the list. The section name is "NewSection_*XXX*" by default (*XXX*: 0 to 255 in decimal numbers). Make detailed section settings in [Segment / Section detail] area. This button is invalid when 256 sections are registered in the list.
Delete	Deletes the segment or section that is selected in the list. If a segment is deleted, the section included in the segment is also deleted.

This area has the following functions.

- Expand/collapse a row view
  You can expand/collapse each low view by double clicking the row or clicking ⊞ or ⊟ at the beginning of the row.

- Move a segment or section row
  You can move segment or section rows by dragging and dropping them.

Remark If a segment is moved, the section included in the segment is also moved.

- Copy a segment or section
  After selecting a segment or section, press the [Ctrl] + [C] key to copy it, then the [Ctrl] + [V] key to paste it. The copy of the row is pasted immediately below the row that is selected when the [Ctrl] + [V] key is pressed. "Copy_" is added to the head of the name of the copy of the segment or section.

Remarks 1. If a segment is copied, the section included in the segment is also copied.

2. The start address of the copy of the segment or section is blank.

3. If the copy cannot be performed due to the attributes of the segment being copied to, an error will occur.

(2) [Segment / Section detail] area

This area displays detailed information on the segment or section selected in the [Segment / Section list] area and you can edit it.

(a) Detailed information of segments

Name	Specify the segment name. The following characters can be used only: 0-9, A-Z, a-z, _, ., /, \.		
	Default	NewSegment_*XXX* (*XXX*: 0 to 255 in decimal numbers)	
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.	
	Restriction	Up to 1022 characters	
Attribute	Select the attribute of the segment. If a segment contains a reserved section, then this is only available if the segment attributes can also be set according to the section attributes. In this case, the attributes that cannot be set are not appeared in the drop-down list.		
	Default	- When adding the segment to the internal ROM area or non mapping area Executable(RX) - When adding the segment to the internal ROM area Read/Write(RW) - When adding the segment to the DataFlash area Read only(R)	
	How to change	Select from the drop-down list.	
	Restriction	Executable(RX)	Makes the segment readable and executable.
		Read only(R)	Makes the segment readable.
		Read/Write(RW)	Makes the segment readable and writable.
		All enable (RWX)	Makes the segment readable, writable, and executable.

Start address	Specify the start address to allocate the segment. If this field is blank, the segment is allocated in the behind of the previous segment by the link function of the compiler.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0x0 to 0xFFFFFFFF (hexadecimal number)
Maximum memory size	Specify the maximum memory size of the segment. If this field is blank, the size is considered as 0x100000 bytes by the link function of the compiler. An error will occur if the specified maximum memory size is exceeded.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0x0 to 0xFFFFFFFF (hexadecimal number)
Hole size	Specify the hole size between segments. If this field is blank, the size is considered as 0x0 (byte) by the link function of the compiler.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0x0 to 0xFFFFFFFF (hexadecimal number)
Filling value	Specify the filling value for a hole between segments. If this field is blank, the value is considered as 0x0000 by the link function of the compiler.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0x0000 to 0xFFFF (hexadecimal number)
Alignment value	Specify the alignment conditions of the segment. When the odd number value is specified, it changes to the even number value by automatically adding one. If this field is blank, the value is considered as 0x8 by the link function of the compiler.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0x0 to 0xFF (hexadecimal number)

(b) Detailed information of sections

Name	Specify the section name. The following characters can be used only: 0-9, A-Z, a-z, _, ., /, \.	
	Default	NewSection_*XXX* (*XXX*: 0 to 255 in decimal numbers)
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 1022 characters

Type	Select the type of the section. Select [Exist data(PROGBITS)] when a object file contains sections with actual values (.text, .data, etc.). Select [No data(NOBITS)] when a object file contains sections without actual values (.bss, .sbss, etc.).		
	Default	Exist data (PROGBITS)	
	How to change	Select from the drop-down list.	
	Restriction	Exist data (PROGBITS)	Sets the section with a default value.
		No data (NOBITS)	Sets the section without a default value.
Attribute	Select the attribute of the section.		
	Default	- When the attribute of the parent segment is [Executable(AX)] Executable(AX) - When the attribute of the parent segment is [Read only(A)] Read only(A) - When the attribute of the parent segment is [Read/Write(AW)] Read/Write(AW) - When the attribute of the parent segment is [All enable (AWX)] All enable (AWX)	
	How to change	Select from the drop-down list.	
	Restriction	Executable(AX)	Sets a section that occupies a memory and enables to execute. This item is not displayed when the attribute of the parent segment is [Read only(R)].
		Read only(A)	Sets a section that occupies a memory.
		Read/Write(AW)	Sets a section that occupies a memory and enables to write. This item is displayed only when the attribute of the parent segment is [Read/Write(RW)] or [All enable (RWX)].
		GP with 1 instruction(AWG)	Sets a section assigned within a memory range that enables it to occupy a memory, write to it, and reference it using a global pointer (gp) and 16-bit displacement. This item is displayed only when the attribute of the parent segment is [Read/Write(RW)] or [All enable (RWX)].
		All enable (AWX)	Sets a section that occupies a memory and enables to write and execute. This item is displayed only when the attribute of the parent segment is [All enable (RWX)].
Start address	Specify the start address to allocate the section. If this field is blank, the section is allocated in the behind of the previous section by the linker.		
	Default	Blank	
	How to change	Directly enter in the text box.	
	Restriction	0x0 to 0xFFFFFFFF (hexadecimal number)	

Hole size	Specify the hole size between sections. If this field is blank, the size is considered as 0x0 (byte) by the linker.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0x0 to 0xFFFFFFFF (hexadecimal number)
Alignment value	Specify the alignment conditions of the section. When the odd number value is specified, it changes to the even number value by automatically adding one. If this field is blank, the value is considered as 0x4 by the linker. However, if the section name is ".tidata.byte" or ".tibss.byte", the odd number value can be specified. If this field is blank, the value is considered as 0x1 by the linker.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0x0 to 0xFF (hexadecimal number)
Input section name	Specify the input section name. The following characters can be used only: 0-9, A-Z, a-z, _, ., /, \.	
	Default	Blank
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.
	Restriction	Up to 1022 characters
Object file name	Specify the name of the object file including the input section. The specified object file name is displayed as the subproperty.	
	Default	Object file name[*number of set items*]
	How to change	Edit by the Object File Select dialog box which appears when clicking the [...] button.

Remark Reserved sections are handled as follows.

- If a section defined in the compiler as a reserved section is specified by [Name] or [Input section name], the [Types] and [Attribute] cannot be edited, and then their values are set automatically.
  The combinations of reserved section names and values set automatically are shown below.

Reserved Section Name	Type	Attribute
.pro_epi_runtime	Exist data (PROGBITS)	Executable(AX)
.text	Exist data (PROGBITS)	Executable(AX)
.data	Exist data (PROGBITS)	Read/Write(AW)
.sedata	Exist data (PROGBITS)	Read/Write(AW)
.sidata	Exist data (PROGBITS)	Read/Write(AW)
.tidata	Exist data (PROGBITS)	Read/Write(AW)
.tidata.byte	Exist data (PROGBITS)	Read/Write(AW)
.tidata.word	Exist data (PROGBITS)	Read/Write(AW)
.bss	No data (NOBITS)	Read/Write(AW)

Reserved Section Name	Type	Attribute
.sebss	No data (NOBITS)	Read/Write(AW)
.sibss	No data (NOBITS)	Read/Write(AW)
.tibss	No data (NOBITS)	Read/Write(AW)
.tibss.byte	No data (NOBITS)	Read/Write(AW)
.tibss.word	No data (NOBITS)	Read/Write(AW)
.sdata	Exist data (PROGBITS)	GP with 1 instruction(AWG)
.sbss	Exist data (PROGBITS)	GP with 1 instruction(AWG)
.const	Exist data (PROGBITS)	Read only(A)
.sconst	Exist data (PROGBITS)	Read only(A)

- The linker limits the reserved sections below to the names of segments where they can be assigned.

Section Name	Segment Name
.sidata, .sibss, .tidata, .tibss, .tidata byte, .tibss.byte, .tidata.word, .tibss.word	SIDATA
.sedata, .sebss	SEDATA
.sconst	SCONST

If one of these section names is specified for [Name], the name of the parent segment is referenced. Although these sections cannot be moved within a segment, they can be moved to other segments.

- For the following reserved sections, the linker creates a fixed correspondence between the output and input section names. For this reason, even if the input section name is omitted, the linker will assign it automatically.
  .pro_epi_runtime, .tidata, .tibss, .tidata.byte, .tibss.byte, .tidata.word, .sidata, .sibss, .sedata, .sebss

(3) [Symbol list] area

This area displays the list of currently configured symbols.

(a) [Name]

This area displays the symbol name.
This item can be edited directly.If the symbol name is changed, the value of [Name] in the [Symbol detail] area is also changed.

(b) [Type]

This area displays the type of the symbol.
This item can be edited directly.If the type is changed, the value of [Type] in the [Symbol detail] area is also changed.

(c) [Address]

This area displays the start address to allocate the symbol.
This item can be edited directly. If the address is changed, the value of [Address] in the [Symbol detail] area is also changed.

(d) Button

Add symbol	Adds a new symbol directly below the row selected in the list. The symbol name is "NewSymbol_*XXX*" by default (*XXX*: 0 to 255 in decimal numbers). Make detailed symbol settings in [Symbol detail] area. This button is invalid when 256 symbols are registered in the list.
Delete symbol	Deletes the section that is selected in the list.

This area has the following functions.

- Move a symbol row
  You can move symbol rows by dragging and dropping them.

(4) [Symbol detail] area

This area displays detailed information on the symbol selected in the [Symbol list] area and you can edit it.

Name	Specify the symbol name. The following characters can be used only: 0-9, A-Z, a-z, _, ., /, \.		
	Default	NewSymbol_*XXX* (*XXX*: 0 to 255 in decimal numbers)	
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.	
	Restriction	Up to 1022 characters	
Type	Select the type of the symbol.		
	Default	TP symbol(%TP_SYMBOL)	
	How to change	Select from the drop-down list.	
	Restriction	TP symbol(%TP_SYMBOL)	Sets the TP symbol as the type of the symbol.
		GP symbol(%GP_SYMBOL)	Sets the GP symbol as the type of the symbol.
		EP symbol(%EP_SYMBOL)	Sets the EP symbol as the type of the symbol.
Base symbol name	Specify the base symbol (TP symbol that is used when the GP symbol value is defined) from among the TP symbol that already exists. If a base symbol name is specified, the offset value from the TP symbol value will be the GP symbol value. The following characters can be used only: 0-9, A-Z, a-z, _, ., /, \. This property is displayed only when [GP symbol(%GP_SYMBOL)] in the [Type] property is selected.		
	Default	Blank	
	How to change	Directly enter in the text box or edit by the Character String Input dialog box which appears when clicking the [...] button.	
	Restriction	Up to 1022 characters	

Address	Specify the symbol to allocate the section. If this field is blank, the address is considered automatically by the linker.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0x0 to 0xFFFFFFFF (hexadecimal number)
Alignment value	Specify the alignment conditions of the symbol. When the odd number value is specified, it changes to the even number value by automatically adding one. If this field is blank, the value is considered as 0x4 by the linker.	
	Default	Blank
	How to change	Directly enter in the text box.
	Restriction	0x0 to 0xFF (hexadecimal number)
Segment name	Specify the segment name that will be referenced by TP and GP symbol values. The specified segment name is displayed as the subproperty. This property is not displayed when [EP symbol(%EP_SYMBOL)] in the [Type] property is selected.	
	Default	Segment name[*number of set items*]
	How to change	Edit by the Segment Select dialog box which appears when clicking the [...] button.

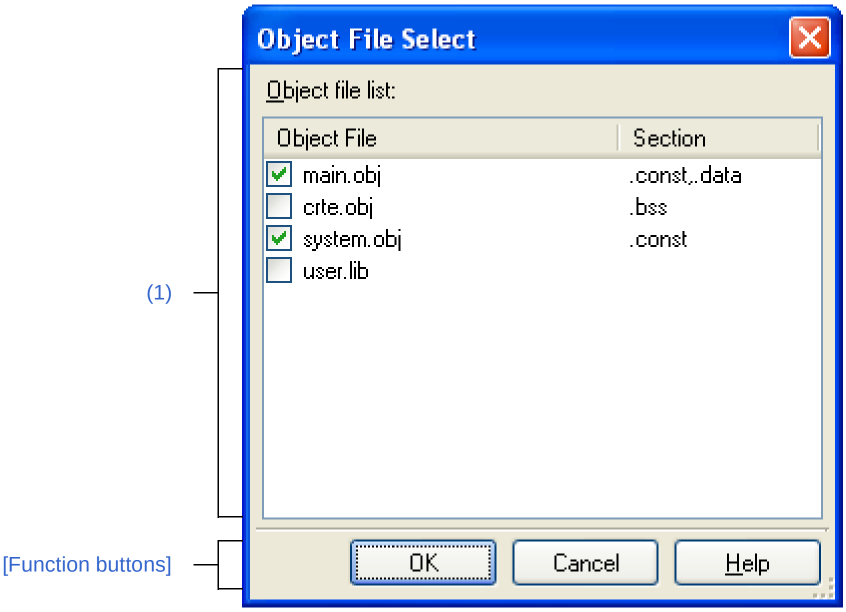
[Function buttons]

Button	Function
Symbol	Toggles the [Symbol list] area and [Symbol detail] area between visible and hidden.
Generate	A link directive file (named *project-name*.dir) is generated based on the specified memory, segments, sections, and symbol allocation information, and then added to the project. The link directive file is generated in the project folder. The link directive file that has been generated is also shown on the project tree, under the File node. The generated link directive file will be a build target. If a link directive file has already been registered to the project, then the file will be removed from the build target.
Close	Closes this dialog box.
Help	Displays the help of this dialog box.

Object File Select dialog box

This dialog box is used to specify the object file among object files and library files added to the project and set it to the area that this dialog box is called from.

Figure A-32. Object File Select Dialog Box



Object File	Section
☑ main.obj	.const,.data
□ crte.obj	.bss
☑ system.obj	.const
□ user.lib	

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- In the Link Directive File Generation dialog box , select a section in the [Segment / Section list] area, and then click the [...] button on [Object file name] in the [Segment / Section detail] area.

[Description of each area]

(1) [Object file list] area

This area displays a list of object files and library files added to the project that opened the Link Directive File Generation dialog box, and the sections that specify them in the Link Directive File Generation dialog box.

(a) [Object File]

This area displays the following file name list.

Select files to set to [Object file name] in the [Segment / Section detail] area in the Link Directive File Generation dialog box that opened this dialog box, via check boxes.

- The object module files generated from the source files added to the project
- The object module files added directly to the project tree
- The library files added directly to the project tree

Remarks 1. Move the mouse cursor over a file name to display a tooltip with the absolute path of that file.

2. In the Link Directive File Generation dialog box that opened this dialog box, in the [Segment / Section detail] area, if an object file is already set in [Object file name], the check box for that object file will be selected by default.

(b) [Section]

This area displays the section that specifies the corresponding object file in the Link Directive File Generation dialog box.
If an object file is specified from multiple sections, they are displayed separated by commas.
If the section that specifies the object file does not exist, this field is blank.

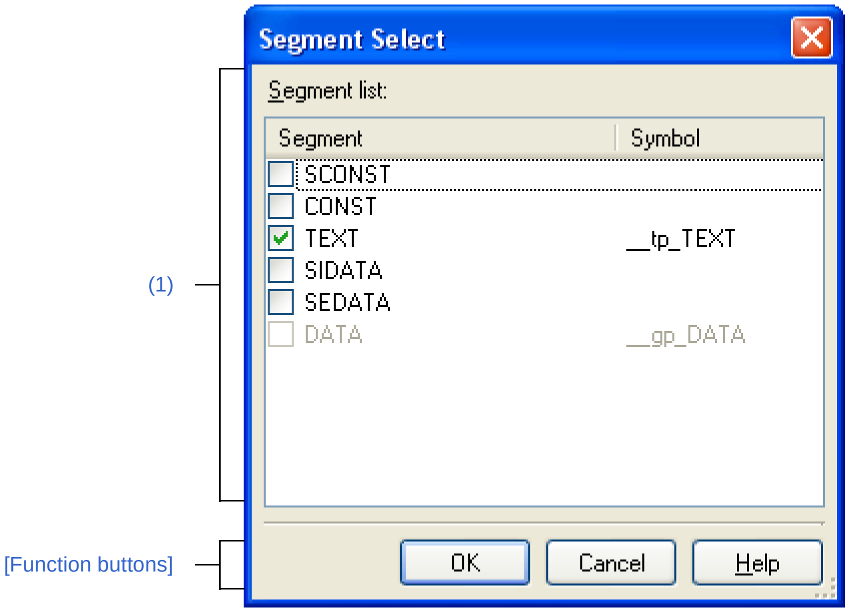
[Function buttons]

Button	Function
OK	Closes this dialog box and sets the selected file to [Object file name] in the [Segment / Section detail] area in the Link Directive File Generation dialog box.
Cancel	Cancels the file selecting and closes the dialog box.
Help	Displays the help of this dialog box.

Segment Select dialog box

This dialog box is used to select the segment among the segments currently set in the Link Directive File Generation dialog box and set it to the area that this dialog box is called from.

Figure A-33. Segment Select Dialog Box



Segment	Symbol
□ SCONST	
□ CONST	
☑ TEXT	__tp_TEXT
□ SIDATA	
□ SEDATA	
□ DATA	__gp_DATA

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- In the Link Directive File Generation dialog box , select a symbol in the [Symbol list] area, and then click the [...] button on [Segment name] in the [Symbol detail] area.

[Description of each area]

(1) [Segment list] area

This area displays the list of currently set segments in the Link Directive File Generation dialog box and symbols that specify them.

(a) [Segment]

This area displays a list of segment names currently set in the Link Directive File Generation dialog box. Select segments to set to [Segment name] in the [Symbol detail] area in the Link Directive File Generation dialog box that opened this dialog box, via check boxes.

Remarks 1. Move the mouse cursor over a file name to display a tooltip with the absolute path of that file.

2. In the Link Directive File Generation dialog box that opened this dialog box, in the [Symbol detail] area, if a segment is already set in [Segment name], the check box for that segment will be selected by default.

3. The check box for the segment that specifies a symbol other than the one that opened this dialog box will be disabled.

(b) [Symbol]

This area displays the symbol specifying the displayed segment.
If the symbol that specifies the segment does not exist, this field is blank.

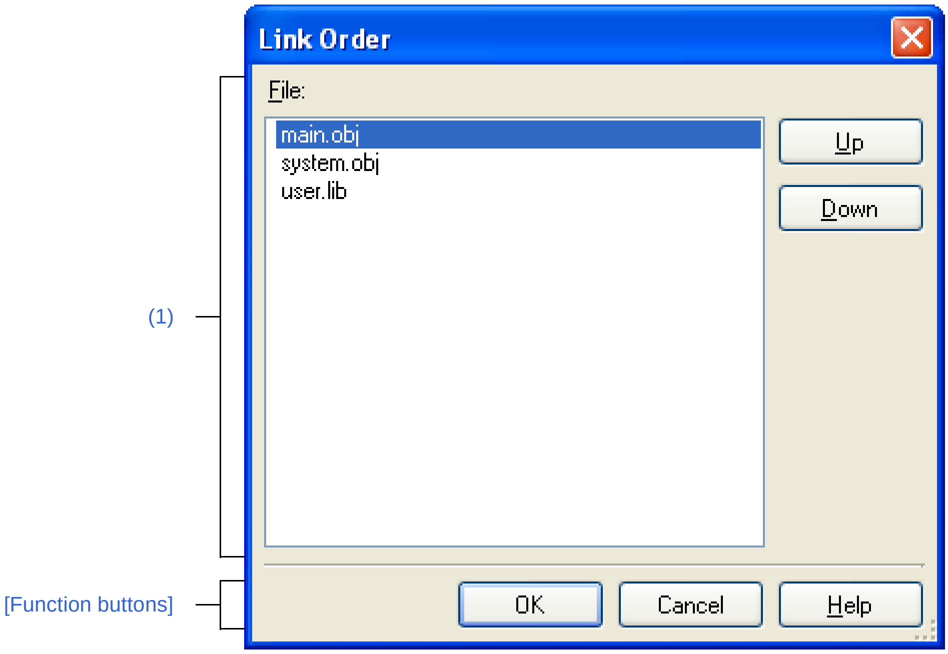
[Function buttons]

Button	Function
OK	Closes this dialog box and sets the selected segment to [Segment name] in the [Symbol detail] area in the Link Directive File Generation dialog box.
Cancel	Cancels the file selecting and closes the dialog box.
Help	Displays the help of this dialog box.

Link Order dialog box

This dialog box is used to refer object module files and library files to be input to the linker and configure these link orders.

Figure A-34. Link Order Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Project Tree panel, select the Build tool node, and then select [Set Link Order...] from the context menu.

[Description of each area]

(1) File list display area

This area displays the file list to be input to the linker.

(a) [File]

The names of the following files are listed in the order that the files are input to the linker.

- Object module files which are generated from the source files added to the selected main project or subproject
- Object module files which are added directly to the project tree of the selected main project or subproject
- Library files which are added directly to the project tree of the selected main project or subproject

The default order is the order that the files are added to the project.

By changing the display order of the files, you can set the input order of the files to the linker.

To change the display order, use the [Up] and [Down] buttons, or drag and drop the file names.

Remarks 1. When the mouse cursor is hovered over a file name, the path of the file appears in a popup. If the file is on the same drive as the project file, then it appears as the relative path; if it is on the different drive, then it appears as the absolute path.

2. Object module files which are generated from newly added source files and newly added object module files are added after the last object module file in the list. Newly added library files are added to the end of the list.

3. When the file is dragged and dropped, the multiple files that are next to each other can be selected together.

(b) Button

Up	Moves the selected file to up.
Down	Moves the selected file to down.

Remark Note that above buttons are disabled when any file is not selected.

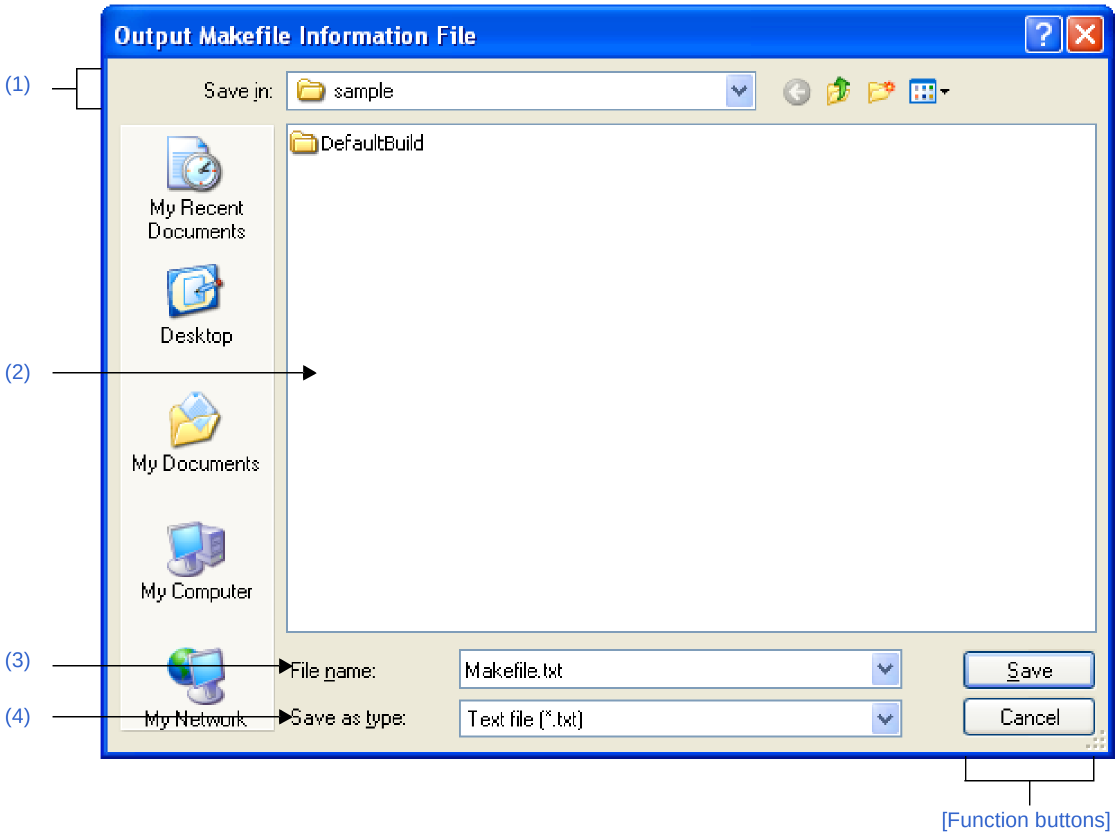
[Function buttons]

Button	Function
OK	Sets the input order of the files to the linker as the display order in the File list display area and closes this dialog box.
Cancel	Cancels the link order settings and closes the dialog box.
Help	Displays the help of this dialog box.

Output Makefile Information File dialog box

This dialog box is used to output a text file described information for creating a Makefile.

Figure A-35. Output Makefile Information File Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Project Tree panel, select the Build tool node, and then select [Output Makefile Information File...] from the context menu.

[Description of each area]

(1) [Save in] area

Specify the folder for outputting the Makefile information file.

The project folder is selected by default.

(2) File list area

This area displays the list of the files which match to the selections in the [Save in] and [Save as type].

(3) [File name] area

Specify the Makefile information file name to be output.

"Makefile.txt" is displayed by default.

(4) [Save as type] area

Designate the type of the Makefile information file to be output.

Text file(*.txt)	Text format (default)
All files(*.*)	All the formats

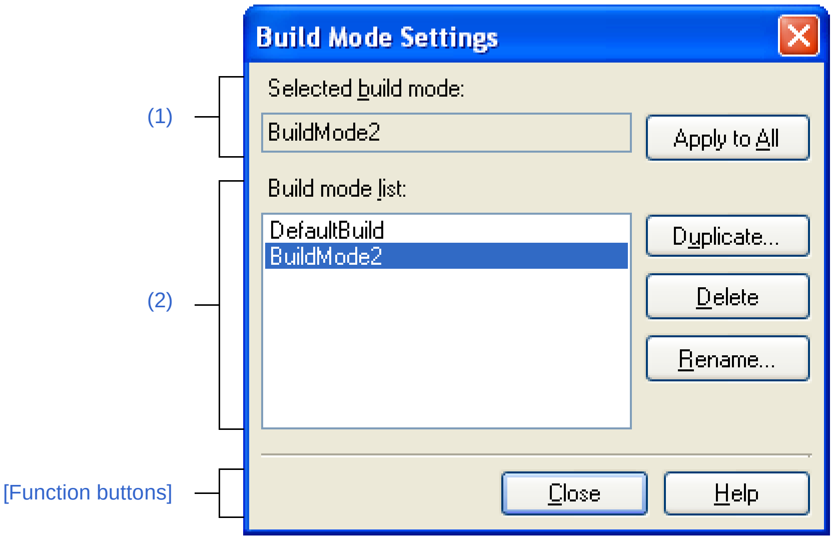
[Function buttons]

Button	Function
Save	Saves the Makefile information file as the designated name.
Cancel	Closes this dialog box.

Build Mode Settings dialog box

This dialog box is used to add and delete build modes and configure the current build mode in batch.

Figure A-36. Build Mode Settings Dialog Box



Selected build mode: BuildMode2

Build mode list:

DefaultBuild
BuildMode2

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- From the [Build] menu, select [Build Mode Settings...].

[Description of each area]

(1) [Selected build mode] area

This area displays the build mode selected in the [Build mode list] area.

(a) Button

Apply to All	Set the build mode of the main project and all subprojects of the currently opened project to the currently displayed build mode.

(2) [Build mode list] area

This area displays all build modes that exist in the currently opening project (main project and subproject) in a list. The current build modes of all projects are same, the build mode is selected by default.If they are not same, "DefaultBuild" will be selected.

The build mode that exists only in part of the main project and subproject is shown with the mark "*".

Note that the "DefaultBuild" is the default build mode and is always displayed at the top.

(a) Button

Duplicate...	Duplicates the selected build mode. The Character String Input dialog box opens. And the build mode is duplicated with the name entered in the dialog box and added to the main project and all the subprojects in the currently opening project. When the build mode with "*" mark does not exist in the main project or subproject and duplicate the build mode, "DefaultBuild" will be duplicated. Up to 20 build modes can be added.
Delete	Deletes the selected build mode. Note that "DefaultBuild" cannot be deleted.
Rename...	Renames the selected build mode. Rename the build mode with entered name in the opening the Character String Input dialog box.

Caution When duplicating or renaming the build mode, the existing build mode name cannot be used.

Remarks 1. Up to 127 characters can be specified as a build mode name. When the input violates any restriction, the following messages are shown in the tooltip.

Message	Description
A build mode with the same name already exists.	The entered build mode name already exists.
More than 127 characters cannot be specified.	Build mode name is too long (more than 128 characters).
The build mode name is invalid. The following characters cannot be used: \, /, :, *, ?, “, <, >, |	Invalid build mode name is entered. The characters (\, /, :, *, ?, “, <, >, |) cannot be used because the build mode name is used for the folder name.

2. Up to 20 build modes can be added. When the input violates any restriction, the following messages are shown in the tooltip.

Message	Description
The maximum number of build modes that can be set per project/subproject is 20.	The number of build modes exceed 20.

[Function buttons]

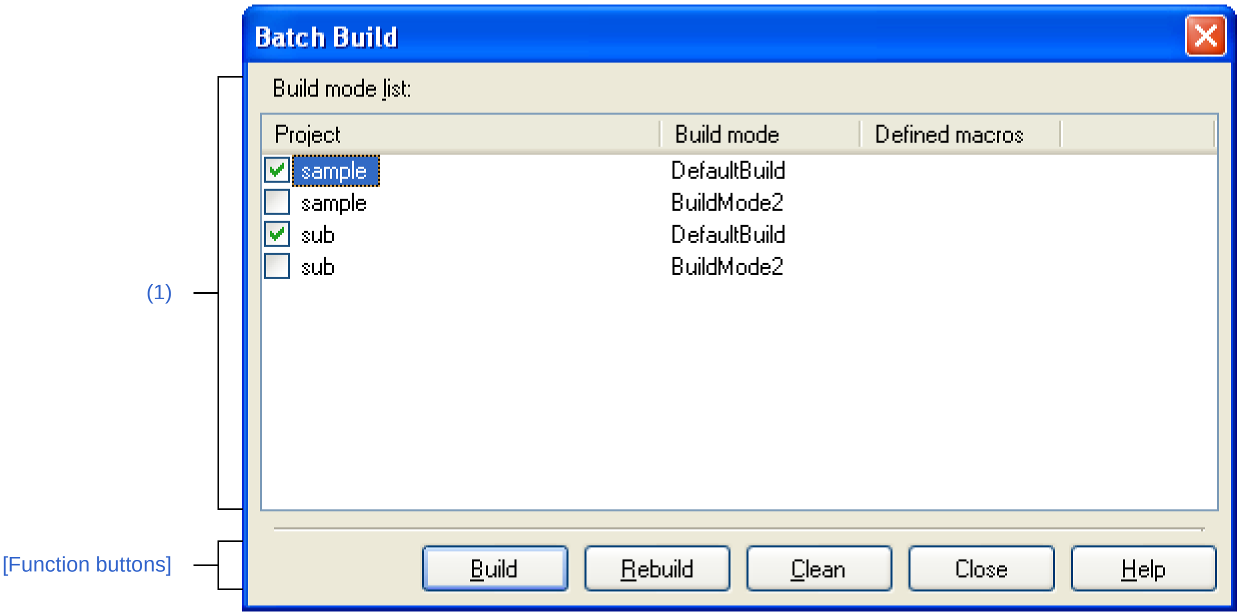
Button	Function
Close	Closes this dialog box.
Help	Displays the help of this dialog box.

Batch Build dialog box

This dialog box is used to run builds, rebuilds and cleans in batch with the build modes that the project (main project and subproject) has.

Remark The batch build order follows the project build order, the order of the subprojects, main project.
When multiple build modes are selected for a single main project or subproject, after running builds of the subproject with all the selected build modes, the build of the next subproject or main project is run.

Figure A-37. Batch Build Dialog Box



Build mode list:

Project	Build mode	Defined macros
☑ sample	DefaultBuild	
□ sample	BuildMode2	
☑ sub	DefaultBuild	
□ sub	BuildMode2	

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- From the [Build] menu, select [Batch Build...].

[Description of each area]

(1) [Build mode list] area

This area displays the list of the combinations of the names of the main project and subprojects in the currently opened project, their build modes, and their macro definitions.

(a) [Project]

This area displays the main project and subprojects in the currently opened project.
Select the check boxes for the combinations of the main project and subprojects and their build modes which you wish to run a build.
When this dialog box is opened for the first time after the project is created, all the check boxes are unchecked. From the second time, the previous setting is retained.

(b) [Build mode]

This area displays the build modes which the main project and subprojects have.

(c) [Defined macros]

For the combination of the main project and subprojects and their build modes, the defined macros which have been set in the [Compile Options] tab and the [Assemble Options] tab on the Property panel are separated with "|" and displayed.

The defined macro in the compile option comes before the one in assemble option. They are separated with ", " and displayed.

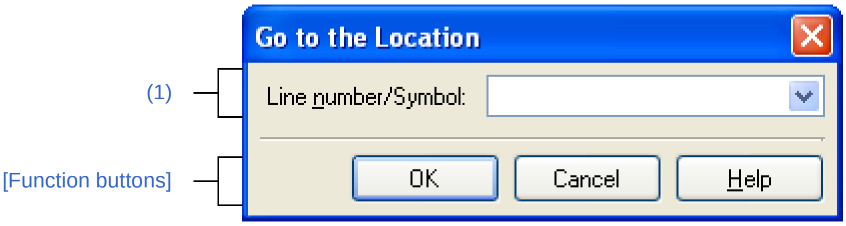
[Function buttons]

Button	Function
Build	Closes this dialog box and runs builds of the selected projects in the respective build modes. The execution result of the builds is displayed on the Output panel. After the builds are complete, the build mode configuration restores to the one before this dialog box was opened. If any project is not selected, this button will be disabled.
Rebuild	Closes this dialog box and runs rebuilds of the selected projects in the respective build modes. The execution result of the rebuilds is displayed on the Output panel. After the rebuilds are complete, the build mode configuration restores to the one before this dialog box was opened. If any project is not selected, this button will be disabled.
Clean	Closes this dialog box and deletes the files which are built in the respective build modes set for the selected projects. The execution result of the cleans is displayed on the Output panel. After the cleans are complete, the build mode configuration restores to the one before this dialog box was opened. If any project is not selected, this button will be disabled.
Close	Closes this dialog box.
Help	Displays the help of this dialog box.

Go to the Location dialog box

This dialog box is used to move the caret to the designated location.

Figure A-38. Go to the Location Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- From the [Edit] menu, select [Move To...].

[Description of each area]

(1) [Line number/Symbol] area

Specify the line number (decimal number) or symbol name of the location to which the caret is moved.
You can directly enter the characters into the text box or select from the input history in the drop down list (maximum numbers of the history: 10).

[Function buttons]

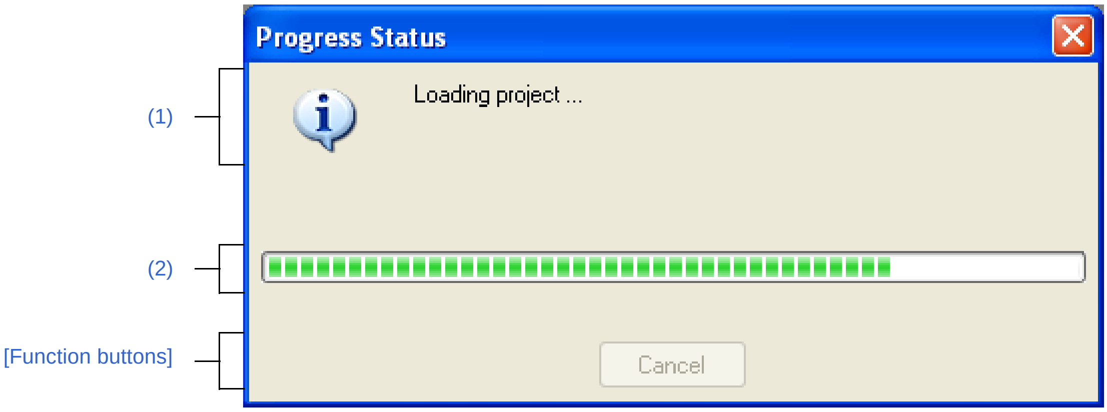
Button	Function
OK	Displays the designated location at the middle of the target panel display and moves the caret there.
Cancel	Cancels the criteria and closes this dialog box.
Help	Displays the help of this dialog box.

Progress Status dialog box

This dialog box is used to show how the process has been progressed when the time consuming process is taken place.

This dialog box automatically closes when the process in progress is done.

Figure A-39. Progress Status Dialog Box



Loading project ...

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- This dialog box automatically opens when a message is output while the time consuming process is in progress.

[Description of each area]

(1) Message display area

This area displays messages output while process is in progress (edit not allowed).

(2) Progress bar

The progress bar shows the current progress of the process in progress with the bar length.

When the process is 100% done (the bar gets to the right end), this dialog box automatically closed.

[Function buttons]

Button	Function
Cancel	Cancels the process in progress and closes this dialog box. Note that this button will be disabled if the process termination is impossible.

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.

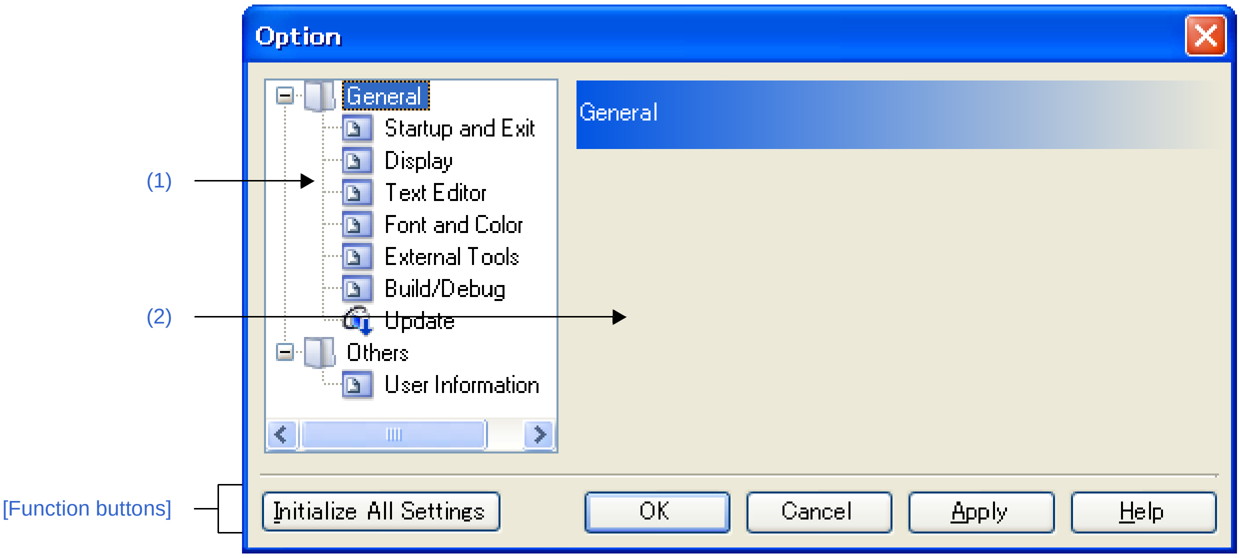


Figure A-40. Option Dialog Box

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) 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 update.
[Other - User Information] category	Configure user information.

Remark See "CubeSuite Start" for details about categories other than [General - Build/Debug].

(2) Settings

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.

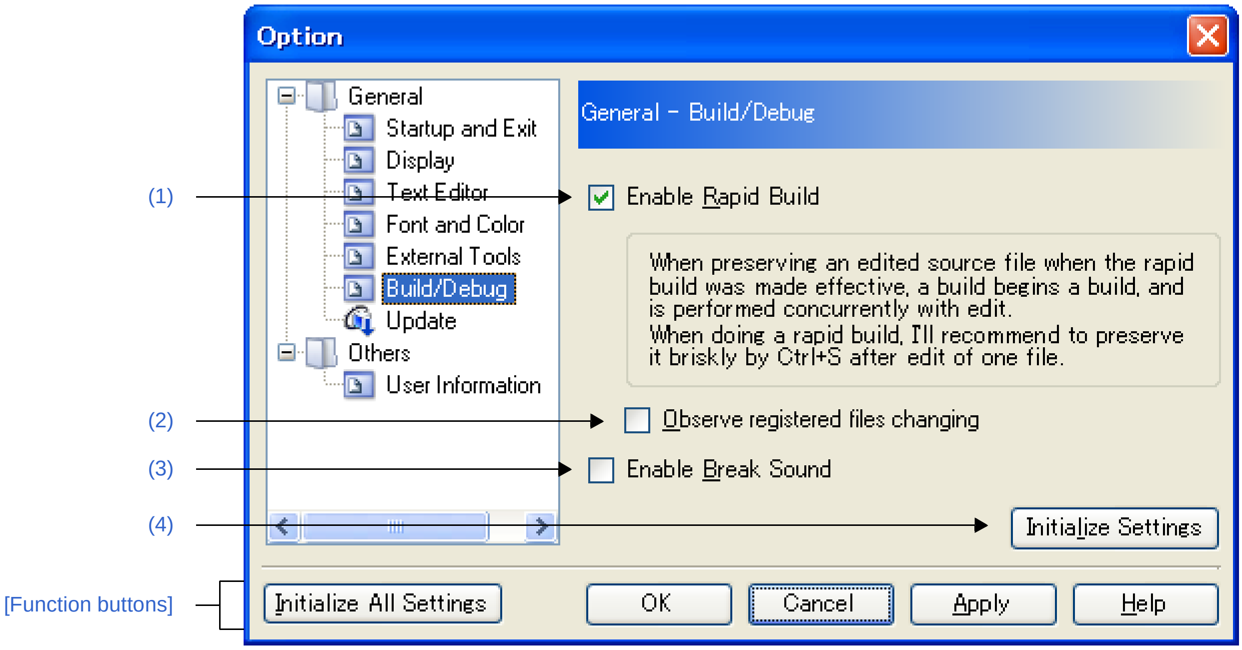
[Function buttons]

Button	Function
Initialize All Settings	Restores 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	Applies all setting and closes this dialog box.
Cancel	Ignores the setting and closes this dialog box.
Apply	Applies all setting (does not close this dialog box).
Help	Displays the help of this dialog box.

[General - Build/Debug] category

Use this category to configure general setting relating to building and debugging.

Figure A-41. Option Dialog Box ([General - Build/Debug] Category)



☑ Enable Rapid Build

When preserving an edited source file when the rapid build was made effective, a build begins a build, and is performed concurrently with edit.
When doing a rapid build, I'll recommend to preserve it briskly by Ctrl+S after edit of one file.

□ Observe registered files changing

□ Enable Break Sound

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]

☑	Enables the rapid build [Note] feature (default).
□	Does 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.

☑	Starts a rapid build when a source file registered in the project is edited or saved by an external text editor or the like.
□	Does 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).

Remark This item is only enabled if the [Enable Rapid Build] check box is selected.

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]

☑	Beeps when the execution of a user program is halted due to a break event (hardware or software break).
□	Does not beep when the execution of a user program is halted due to a break event (hardware or software break) (default).

(4) Buttons

Initialize Settings	Returns all currently displayed setting to their default values.

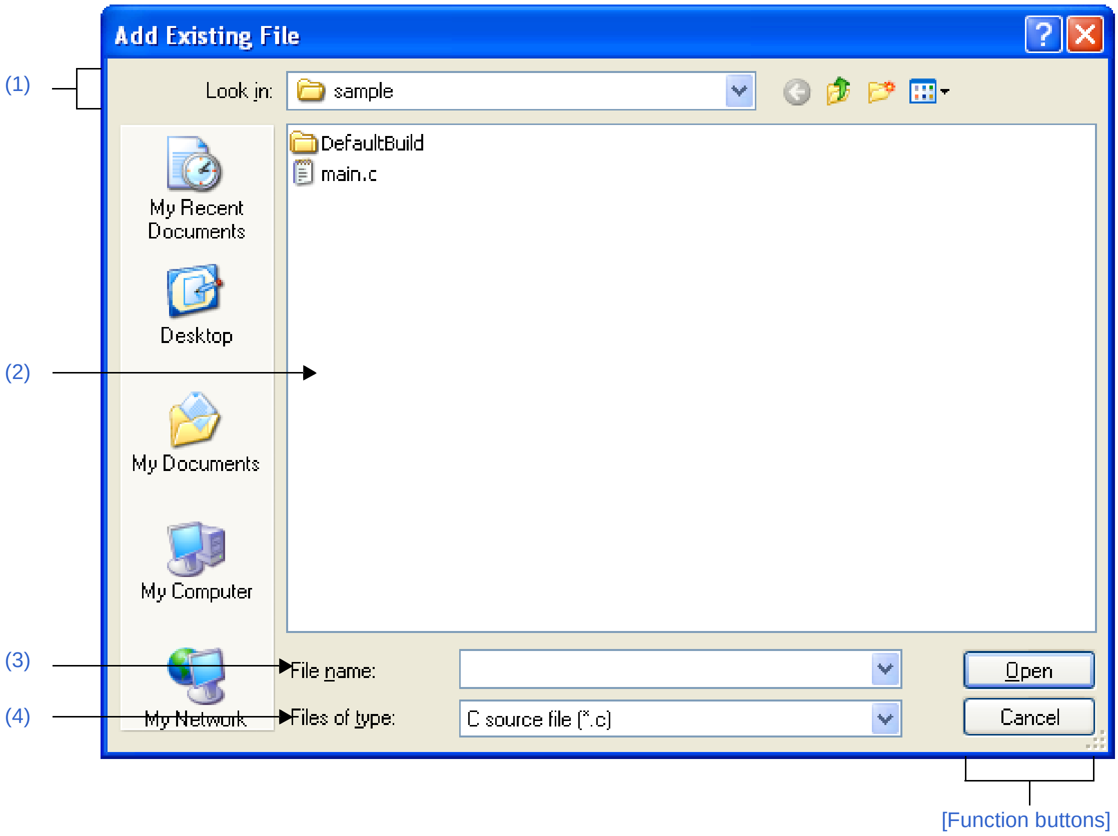
[Function buttons]

Button	Function
Initialize All Settings	Restores 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	Applies all setting and closes this dialog box.
Cancel	Ignores the setting and closes this dialog box.
Apply	Applies all setting (does not close this dialog box).
Help	Displays the help of this dialog box.

Add Existing File dialog box

This dialog box is used to select existing files to add to the project.

Figure A-42. Add Existing File Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- From the [File] menu, select [Add] >> [Add File...].
- On the Project Tree panel, select either one of the Project node, Subproject node, File node, or file, and then select [Add] >> [Add File...] from the context menu.

[Description of each area]

(1) [Look in] area

Select the folder that the file to be added to the project exists.

The project folder is selected by default.

(2) File list area

This area displays the list of the files which match to the selections in the [Look in] and [Files of type].

(3) [File name] area

Designate the name of the file to be added to the project.

(4) [Files of type] area

Designate the type of the file to be added to the project.

C source file (*.c)	C source file
Header file(*.h; *.inc)	Header file
Assemble file(*.asm; *.s)	Assembler source file
Link directive file(*.dir; .*dr)	Link directive file
Symbol information file(*.sfg)	Symbol information file
Library file(*.lib; *.a)	Library file
Object module file(*.obj; *.o)	Object module file
Text file(*.txt)	Text format
All Files(*.*)	All the formats (default)

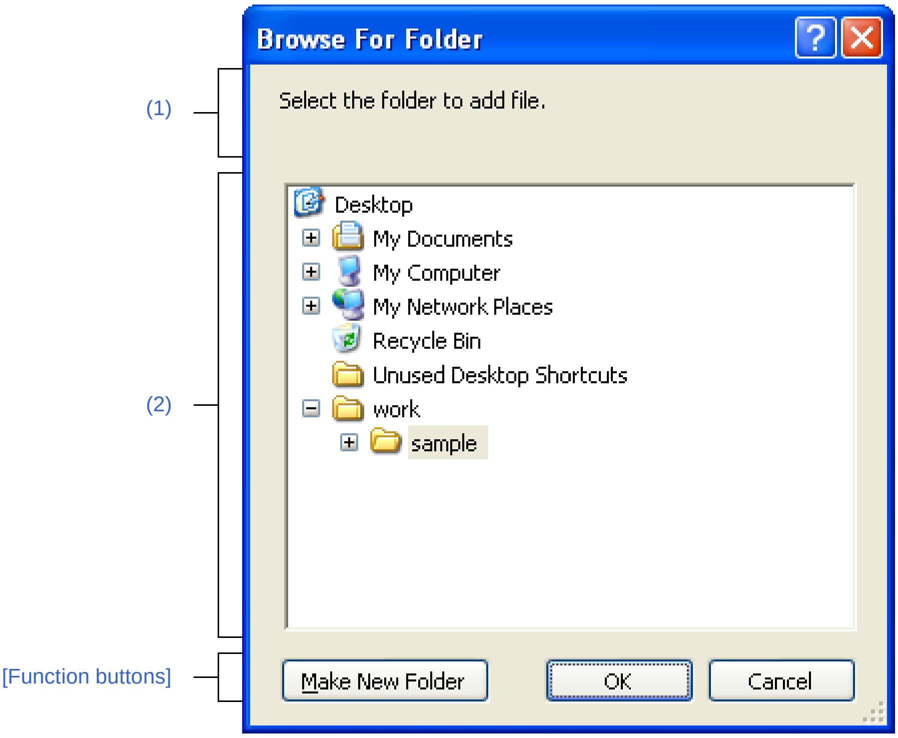
[Function buttons]

Button	Function
Open	Adds the designated file to the project.
Cancel	Closes this dialog box.

Browse For Folder dialog box

This dialog box is used to select the folder and set it to the area that this dialog box is called from.

Figure A-43. Browse For Folder Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- In the Add File dialog box, click the [Refer...] button in the [File location] area.
- In the Path Edit dialog box, click the [Browse...] button in the path edit area.
- On the Property panel, select the following properties, and then click the [...] button.
  - From the [Common Options] tab, [Intermediate file output folder] in the [Output File Type and Path] category, [Output folder] in the [Frequently Used Options(for Link)] category, [Error message file output folder] in the [Error Output] category, and [Temporary folder] in the [Others] category
  - From the [Compile Options] tab, [Output folder for assembler source file] in the [Output File] category and [Output folder for assemble list file] in the [Assemble List] category
  - From the [Link Options] tab, [Output folder] in the [Output File] category, [Output folder for link map file] in the [Link Map] category, and [Output folder for symbol information file] in the [Symbol Information] category
  - From the [Hex Output Options] tab, [Output folder for hex file] in the [Output File] category
  - From the [Create Library Options] tab, [Output folder] in the [Output File] category
  - From the [Individual Compile Options] tab, [Output folder for assembler source file] in the [Output File] category, [Output folder for assemble list file] in the [Assemble List] category, and [Error message file output folder] in the [Error Output] category
  - From the [Individual Assemble Options] tab, [Error message file output folder] in the [Error Output] category

[Description of each area]

(1) Message area

This area displays the message related to the folder to be selected in this dialog box.

(2) Folder location area

Select the folder to be set to the area that this dialog box is called from.

The folder selected by default differs depending on the area that this dialog box is called from.

(a) Add File dialog box

The folder has been set to the area that this dialog box is called from is selected.

When the area has been blank or the path which does not exist has been set, the project folder is selected by default.

(b) Path Edit dialog box and Property panel

The project folder is selected.

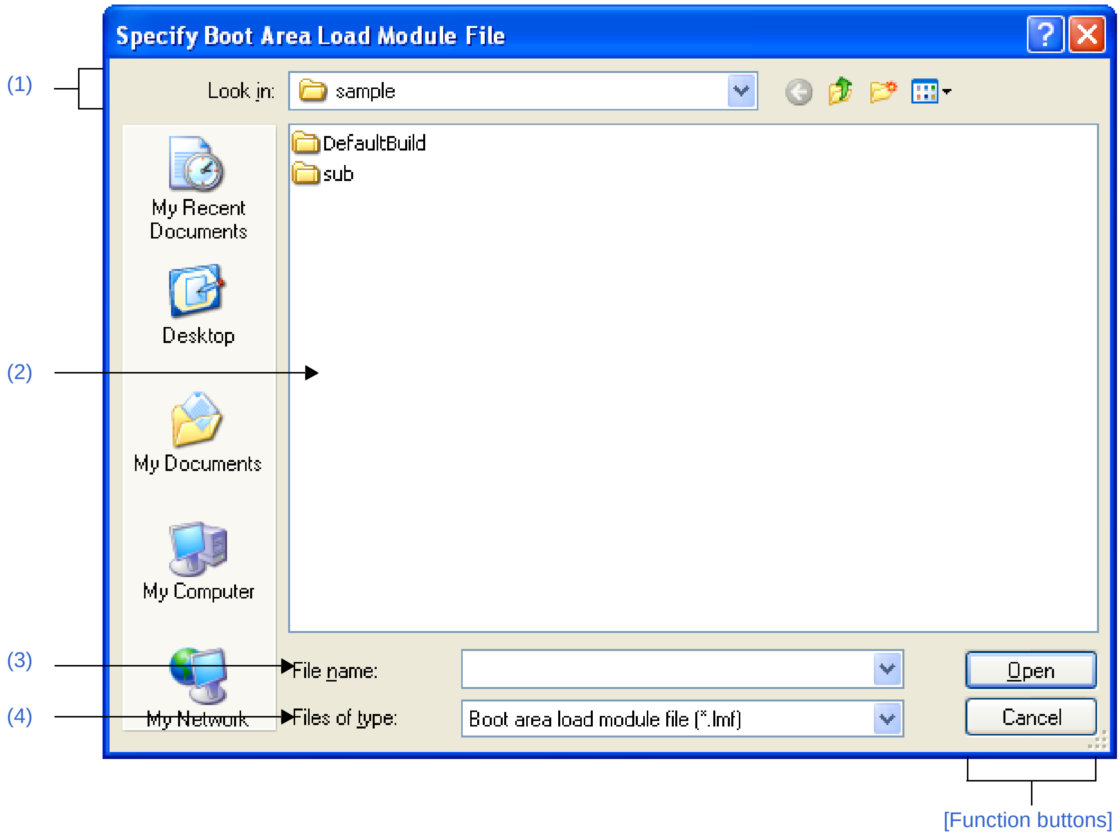
[Function buttons]

Button	Function
Make New Folder	Creates a new folder directly below the root of the selected folder. The default folder name is "New Folder".
OK	Sets the designated folder path to the area that this dialog box is called from.
Cancel	Closes this dialog box.

Specify Boot Area Load Module File dialog box

This dialog box is used to select the boot area load module file and set it to the area that this dialog box is called from.

Figure A-44. Specify Boot Area Load Module File Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Property panel, select the following property, and then click the [...] button.
    - From the [Common Options] tab, [Boot area load module file name] in the [Flash Correspondence] category

[Description of each area]

(1) [Look in] area

Select the folder where the file to be set to the area that this dialog box is called from exists.

The project folder is selected by default.

(2) File list area

This area displays the list of the files which match to the selections in the [Look in] and [Files of type].

(3) [File name] area

Select the name of the file to be set to the area that this dialog box is called from.

(4) [Files of type] area

Select the type of the file to be set to the area that this dialog box is called from.

Boot area load module file(*.lmf)	Boot area load module file (default)
All Files(*.*)	All the formats

[Function buttons]

Button	Function
Open	Sets the designated file to the area that this dialog box is called from.
Cancel	Closes this dialog box.

Specify Far Jump File dialog box

This dialog box is used to select the Far Jump file and set it to the area that this dialog box is called from.

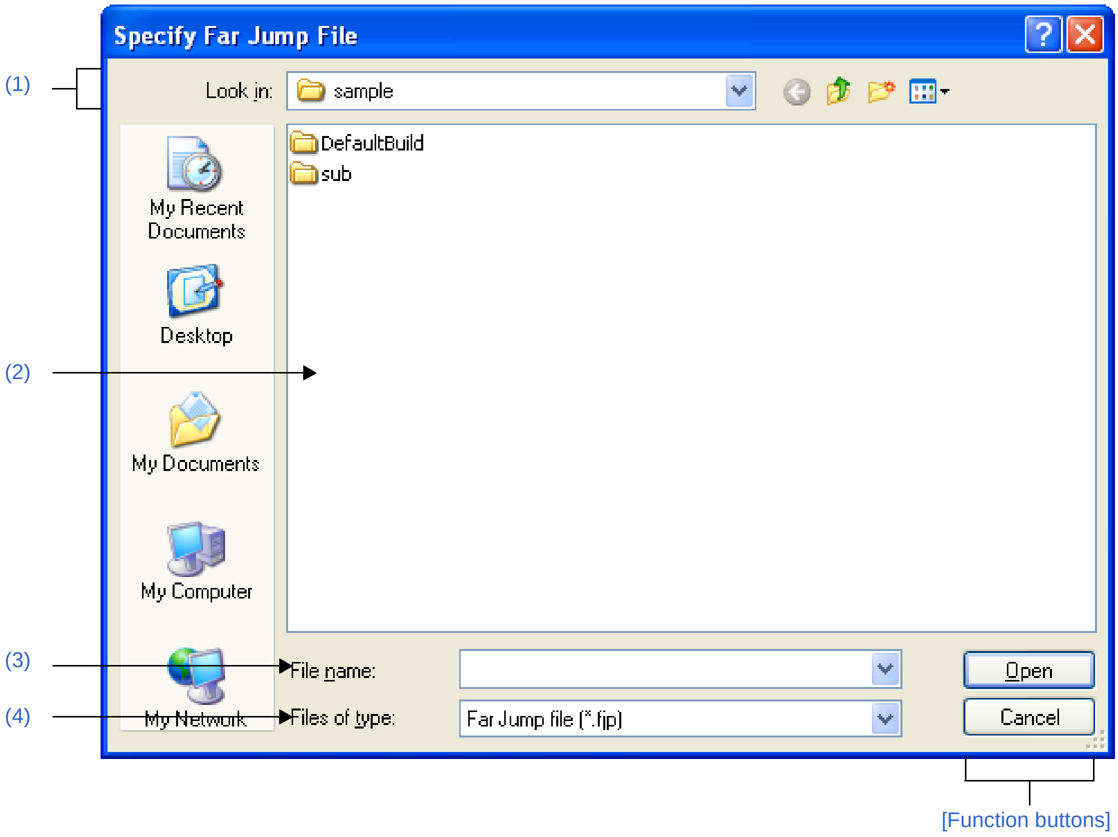


Figure A-45. Specify Far Jump File Dialog Box

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Property panel, from the [Compile Options] tab, in the [Output Code] category, after selecting the [Far Jump file names] property, open the Path Edit dialog box by clicking the [...] button.
  And then click the [Browse...] button in the dialog box.

[Description of each area]

(1) [Look in] area

Select the folder where the file to be set to the area that this dialog box is called from exists.
The project folder is selected by default.

(2) File list area

This area displays the list of the files which match to the selections in the [Look in] and [Files of type].

(3) [File name] area

Select the name of the file to be set to the area that this dialog box is called from.

(4) [Files of type] area

Select the type of the file to be set to the area that this dialog box is called from.

Far Jump file(*.fjp)	Far Jump file

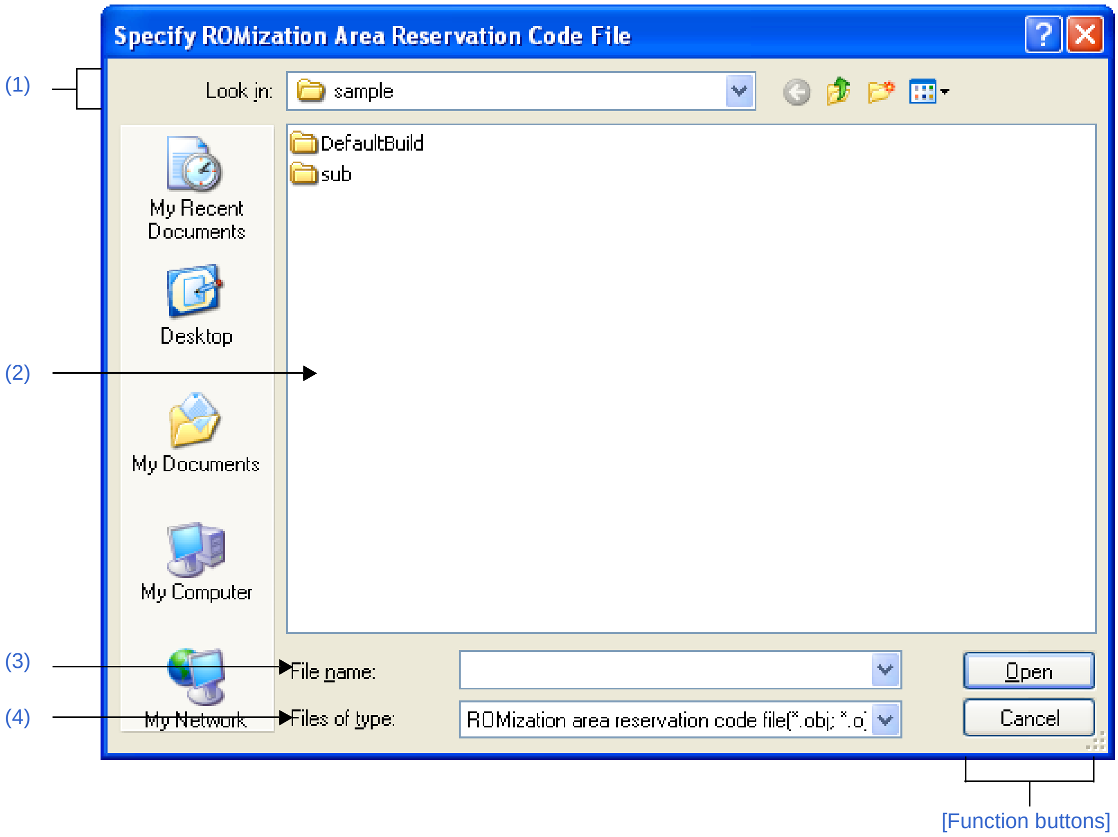
[Function buttons]

Button	Function
Open	Sets the designated file to the area that this dialog box is called from.
Cancel	Closes this dialog box.

Specify ROMization Area Reservation Code File dialog box

This dialog box is used to select the ROMization area reservation code file and set it to the area that this dialog box is called from.

Figure A-46. Specify ROMization Area Reservation Code File Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Property panel, select the following property, and then click the [...] button.
    - From the [ROMize Options] tab, [ROMization area reservation code file name] in the [Input File] category

[Description of each area]

(1) [Look in] area

Select the folder where the file to be set to the area that this dialog box is called from exists.

The project folder is selected by default.

(2) File list area

This area displays the list of the files which match to the selections in the [Look in] and [Files of type].

(3) [File name] area

Select the name of the file to be set to the area that this dialog box is called from.

(4) [Files of type] area

Select the type of the file to be set to the area that this dialog box is called from.

ROMization area reservation code file(*.obj; *.o)	ROMization area reservation code file (default)
ROMization area reservation code file(*.asm; *.s)	ROMization area reservation code file

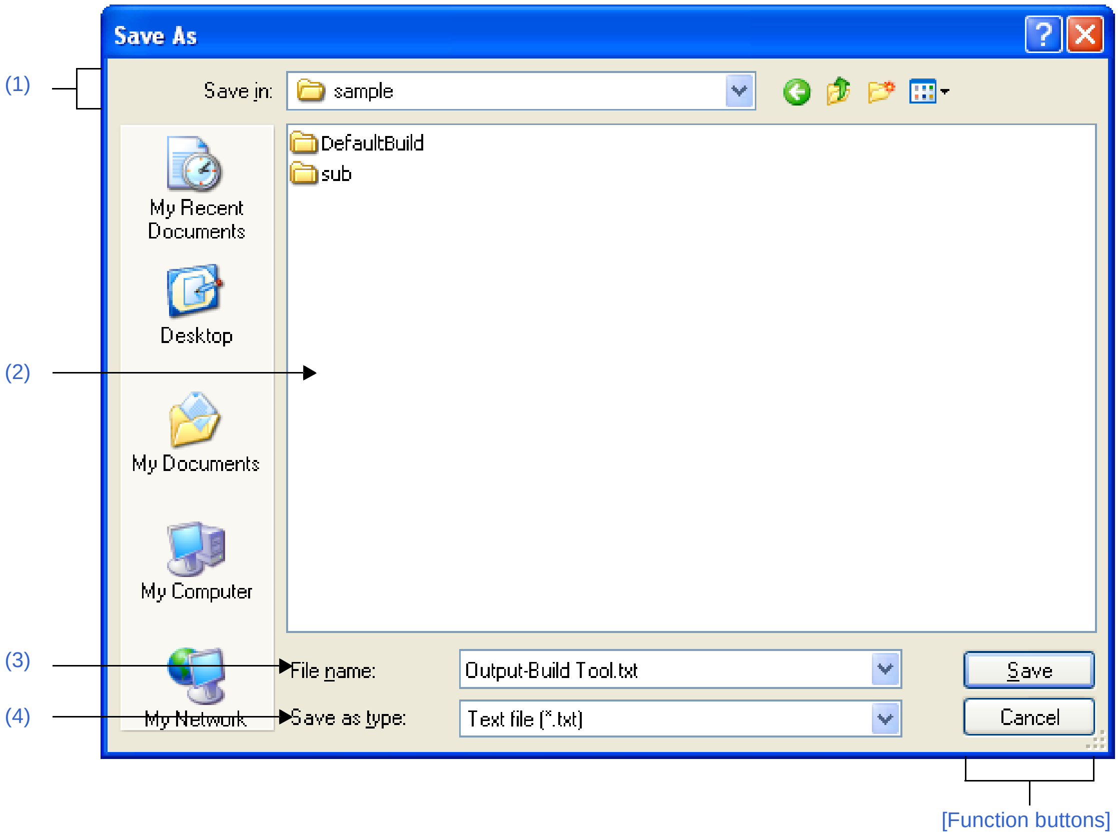
[Function buttons]

Button	Function
Open	Sets the designated file to the area that this dialog box is called from.
Cancel	Closes this dialog box.

Save As dialog box

This dialog box is used to save the editing file or contents of each panel to a file with a name.

Figure A-47. Save As Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- Focus the Editor panel, and then select [Save *file name* As...] from the [File] menu.
- Focus the Output panel, and then select [Save *tab name* As...] from the [File] menu.

[Description of each area]

(1) [Save in] area

Select the folder to save the panel contents in the file.

The following folders are selected by default.

(a) In the Editor panel

The folder that currently editing file is saved is selected.

(b) In the Output panel

The project folder is selected when the file is saved for the first time. The previously selected file is selected after the second time.

(2) File list area

This area displays the list of the files which match to the selections in the [Save in] area and [Save as type] area.

(3) [File name] area

Designate the name of the file to be saved.

(4) [Save as type] area

(a) In the Editor panel

This area displays any of the following file types depend on the type of the currently editing file.

Text file(*.txt)	Text format
C source file(*.c)	C source file
Header file(*.h; *.inc)	Header file
Assemble file(*.asm)	Assembler source file
Assemble file(*.s)	Assembler source file
Link directive file(*.dir; *.dr)	Link directive file
Symbol information file(*.sfg)	Symbol information file
Map file(*.map)	Map file
Hex file(*.hex)	Hex file

(b) In the Output panel

This area displays the following file type.

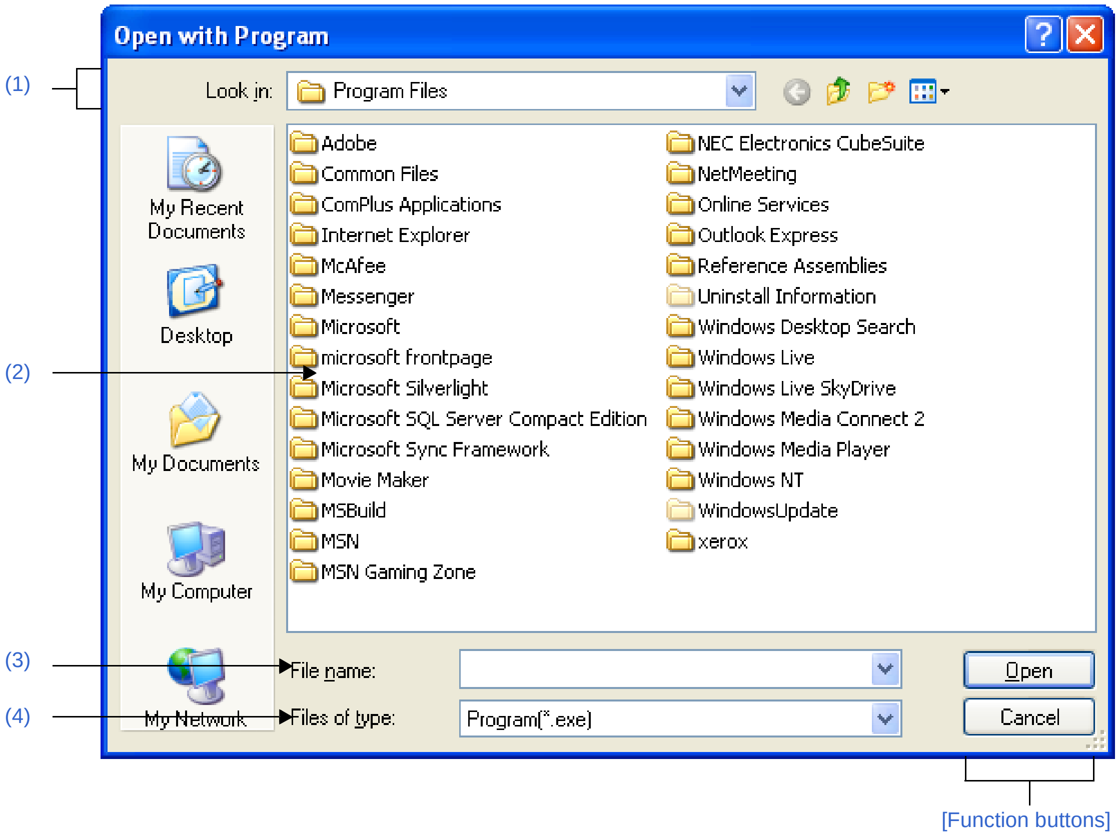
Text file(*.txt)	Text format

[Function buttons]

Button	Function
Save	Saves the file as the designated name.
Cancel	Closes this dialog box.

Open with Program dialog box

This dialog box is used to select the application to open the file selected in Project Tree.



Open with Program

Look in: Program Files

File name:

Files of type: Program(*.exe)

Figure A-48. Open with Program Dialog Box

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Project Tree panel, select a file, and then select [Open with Selected Application...] from the context menu.

[Description of each area]

(1) [Look in] area

Select the folder where the application to open the file is stored.

Program folder (for Windows XP, "C:\Program Files") is selected by default.

(2) File list area

This area displays the list of the files which match to the selections in the [Look in] and [Files of type].

(3) [File name] area

Specify the executable file name of the application to open the file.

(4) [Files of type] area

Specify the executable file type of the application to open the file.

Program(*.exe)	Executable format (default)
All Files (*.*)	All the formats

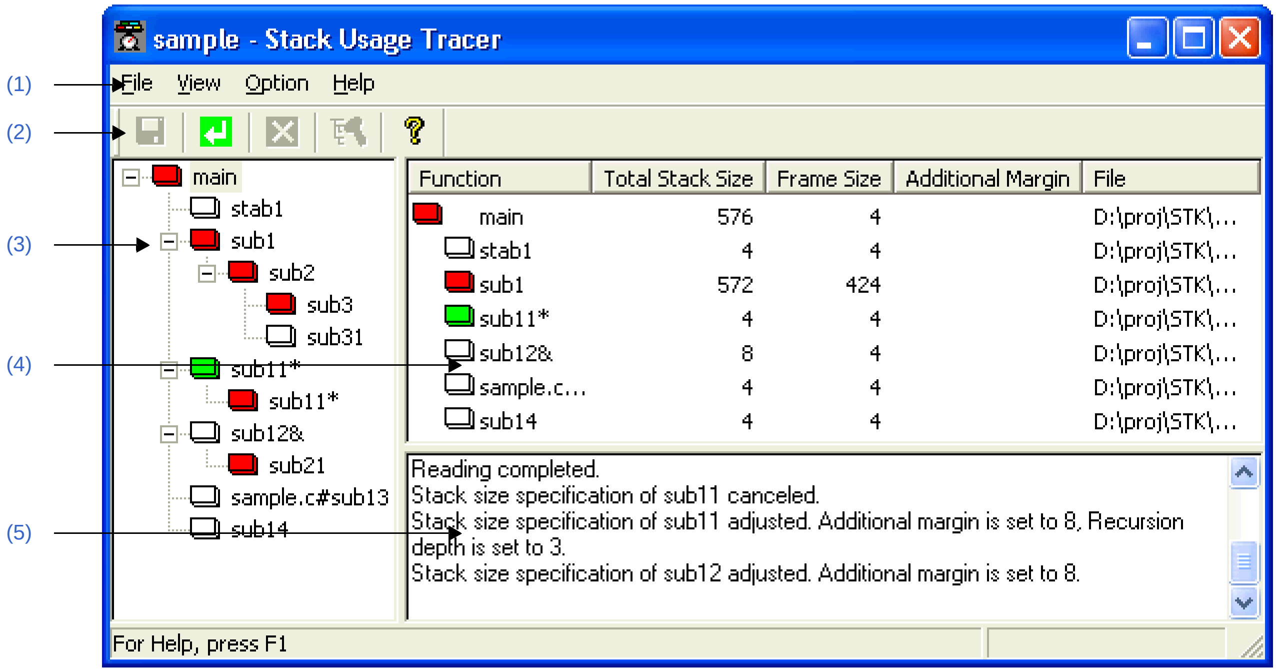
[Function buttons]

Button	Function
Open	Opens the file with the specified application.
Cancel	Closes this dialog box.

Stack Usage Tracer window

This is the first window to be opened when the stack usage tracer is launched.
Use this window to check or modify the amount of stack used on a per-function basis.

Figure A-49. Stack Usage Tracer Window



Function	Total Stack Size	Frame Size	Additional Margin	File
🟥 main	576	4		D:\proj\STK\...
□ stab1	4	4		D:\proj\STK\...
🟥 sub1	572	424		D:\proj\STK\...
🟩 sub11*	4	4		D:\proj\STK\...
□ sub12&	8	4		D:\proj\STK\...
□ sample.c...	4	4		D:\proj\STK\...
□ sub14	4	4		D:\proj\STK\...

```
Reading completed.
Stack size specification of sub11 canceled.
Stack size specification of sub11 adjusted. Additional margin is set to 8, Recursion
depth is set to 3.
Stack size specification of sub12 adjusted. Additional margin is set to 8.
```

The following items are explained here.

- [How to open]
- [Description of each area]
- [Caution]

[How to open]

- From the [Tool] menu, select [Startup Stack Usage Tracer].

[Description of each area]

(1) Menu bar

This area consists of the following menu items.

(a) [File] menu

Save Call Chain with Maximum Stack from Selected Function...	Opens the Save As dialog box for saving the call chain with the greatest total stack size (including the stack size of callee functions) of the function selected in the tree display area / list display area to an output result file. Functions in the same manner as the 💾 button.
Save All Call Chains from Selected Function...	Opens the Save As dialog box for saving all call chains of the function selected in the tree display area / list display area to an output result file.
Save Call Chain with Maximum Stack from Every Root...	Opens the Save As dialog box for saving the call chain of the function displayed in the tree display area with the largest total stack size to an output result file.

Save All Call Chains from Every Root...	Opens the Save As dialog box for saving all call chains of all functions displayed in the tree display area to an output result file.
Load Stack Size Specification File...	Opens the Open dialog box for loading a stack size specification file.
Save Stack Size Specification File...	Opens the Save As dialog box for saving the results of the operations made in the Adjust Stack Size dialog box (e.g. changes to function information) to a stack size specification file.
Exit skcx	Closes this window.

Remark The output result file can only be saved in text format (*.txt) or CSV format (*.csv).

(b) [View] menu

Recalculate Stack Size	Recalculates the total stack size. Functions in the same manner as the ↵ button.	
Stop	Forcibly stop the action of the stack usage tracer (e.g. recalculating the total stack size). Functions in the same manner as the ☒ button.	
Sort List by	Changes the function display order in the list display area.	
	Function Name	Sort by function name.
	Icon Type	Sort by icon display priority (High: 🟥 to Low: □).
	Stack Size	Sort by total stack size.
	Frame Size	Sort by frame size.
	Additional Margin	Sort by additional margin.
	File Name	Sort by file name.

(c) [Option] menu

Stack Size Unknown / Adjusted Function Lists...	Opens the Stack Size Unknown / Adjusted Function Lists dialog box to display a list of functions with unknown frame size, functions for which information (additional margin, recursion depth, or callee functions) has been modified, and functions for which the stack usage tracer has forcibly set an additional margin.
Adjust Stack Size...	Opens the Adjust Stack Size dialog box to change the information (additional margin, recursion depth, and callee functions) for the function selected in the tree display area / list display area. This dialog box is used to change the information (additional margin, recursion depth, and callee functions) for the selected function. Functions in the same manner as the 🔨 button.
Reset Function	Resets the information (additional margin, recursion depth, and callee functions) for the selected function to the default values. This button will be grayed out if all the information for the selected function has the default values.
Reset All Functions	Resets the information (additional margin, recursion depth, and callee functions) for all functions to the default values. This button will be grayed out if all the information for all functions has the default values.

(d) [Help] menu

skcx Help	Displays the help of this window. Functions in the same manner as the ? button.
About skcx...	Opens the Version Information dialog box of the stack usage tracer.

(2) Toolbar

This area consists of the following buttons.

💾	Opens the Save As dialog box for saving the call chain with the greatest total stack size (including the stack size of callee functions) of the function selected in the tree display area / list display area to an output result file. Functions in the same manner as when [Save Call Chain with Maximum Stack from Selected Function...] is selected from the [File] menu.
↵	Recalculates the total stack size. Function in the same manner as when [Recalculate Stack Size] is selected from the [View] menu.
☒	Forcibly stop the action of the stack usage tracer (e.g. recalculating the total stack size). Functions in the same manner as when [Stop] is selected from the [View] menu.
🛠	Opens the Adjust Stack Size dialog box to change the information (additional margin, recursion depth, and callee functions) for the function selected in the tree display area / list display area. Functions in the same manner as when [Adjust Stack Size...] is selected from the [Option] menu.
?	Displays the help of this window. Functions in the same manner as when [skcx Help] is selected from the [Help] menu.

(3) Tree display area

The calling relationship of the functions is shown in tree format.

The table below shows the meaning of the icon displayed to the left of the string representing the function name.

🟥	The function directly called by a given function with the largest total stack size
🟦	Information (additional margin, recursion depth, or callee functions) has been modified via the Adjust Stack Size dialog box or a stack size specification file
🟩	Recursive function
🟨	The stack usage tracer has not acquired any stack information for this function
□	Other than the above

Remark The display priority for icons is from High: 🟥 to Low: □ .

(a) Context menu

Select a function in this area, and then right click with the mouse. The context menu described below appears.

Adjust Stack Size...	Opens the Adjust Stack Size dialog box to change the information (additional margin, recursion depth, and callee functions) for the selected function.

(4) List display area

Display the stack information for a single function (function name, total stack size, frame size, additional margin, and file name) in list format.

Function	Displays the function name. Note that this area will only display functions from level 1 (the selected function) and level 2 (functions called directly by the selected function).
Total Stack Size	Displays the total stack size (including the stack size of callee functions; in bytes).
Frame Size	Displays the frame size (not including the stack size of callee functions; in bytes).
Additional Margin	Displays the value to mandatorily added to frame size (in bytes).
File	Displays the file name.

The table below shows the meaning of the icon displayed to the left of the string representing the function name.

🟥	The function directly called by a given function with the largest total stack size
🟦	Information (additional margin, recursion depth, or callee functions) has been modified via the Adjust Stack Size dialog box or a stack size specification file
🟩	Recursive function
🟨	The stack usage tracer has not acquired any stack information for this function
⬜	Other than the above

(a) Context menu

Select a function in this area, and then right click with the mouse. The context menu described below appears.

Adjust Stack Size...	Opens the Adjust Stack Size dialog box to change the information (additional margin, recursion depth, and callee functions) for the selected function.	
Sort List by	Changes the function display order in the list display area.	
	Function Name	Sort by function name.
	Icon Type	Sort by icon display priority (High: 🟥 to Low: ⬜).
	Stack Size	Sort by total stack size.
	Frame Size	Sort by frame size.
	Additional Margin	Sort by additional margin.
	File Name	Sort by file name.

(5) Message display area

Display operation logs of the stack usage tracer.

[Caution]

- Assembly files
  The stack usage tracer calculates total stack size by collecting information from the assembly files output by the compiler as intermediate files, with debugging information added. As a consequence, in order to obtain stack information at the function level using the stack usage tracer, it is necessary to configure the compiler options to output "Assembly files with debugging information".

- Timing of static analysis
  The stack usage tracer performs static analysis upon startup, and displays the calling relationship between functions and function-level stack information in its main window. Consequently, changes to the calling relationship between functions or function-level stack information (e.g. adding files, changing compiler options, or modifying the source code) will not be reflected in this window.

- Functions analyzed
  The stack usage tracer only analyzes functions contained in assembly files with debugging information output by the compiler as intermediate files, and in library files provided by the build tool. Consequently, functions in assembly files written by the user and library files created by the user are not analyzed. For this reason, the information for these files must be set using the Adjust Stack Size dialog box.

- Icon display colors
  Display priorities (High: 🟥 to Low: □) are assigned to icons displayed in the tree display area/list display area in the window. Consequently, you must be aware that even if the 🟥 icon (function called directly by same function with greatest total stack size) is displayed, information with relatively low priority, such as the 🟨 icon (frame size unknown) will be hidden by the GUI.

- Determining the maximum stack size
  When the stack usage tracer searches for the path with the largest stack size, it assumes that functions that are not analyzed have a stack size of zero. Consequently, when determining the maximum stack size, you must make sure that there are no functions under [Unknown Functions] in the Stack Size Unknown / Adjusted Function Lists dialog box.

- Tree display for recursive functions
  The window's tree display area only displays up to the second call of a recursive function. Consequently, the third and subsequent calls are hidden.

- Library functions bsearch, exit, and qsort
  The stack usage tracer treats bsearch, exit, and qsort as unknown functions, even if they are in a library file provided by the build tool. Consequently, if you are using these functions, you must set the relevant information (e.g. recursion depth and callee functions) in the Adjust Stack Size dialog box.

- Callee functions
  The stack usage tracer only allows the following types of "callee functions" to be added in the Adjust Stack Size dialog box: functions contained in C source files, and functions that are explicitly called (not called using a pointer). Consequently, the [All Functions] section of the Adjust Stack Size dialog box only displays functions meeting the above conditions.

- Functions called by multiple functions

  The stack usage tracer treats the stack information of functions called by multiple functions as unique. Consequently, it is not possible to change the stack information for such functions depending on which function is calling them.

Example If you select function sub called by func1 in the tree display area and open the Adjust Stack Size dialog box, the changes are reflected in sub called by func2 as well.

```
int     sub ( int i );
void    func1 ( void );
void    func2 ( void );

void main ( void ) {
    func1 ( );
    func2 ( );
}
int sub ( int i ) {
    i++;
    return ( i );
}

void func1 ( void ) {
    int ret, i = 0;
    ret = sub ( i );
}

void func2 ( void ) {
    int ret, i = 100;
    ret = sub ( i );
}
```

- ASM statements in C source

  If C source contains ASM statements, the stack usage tracer may output the following message: "W0594132 : Illegal format in file (path name : line number)". If this occurs, fix the problem by disabling the code in question using #if declarations or the like, or commenting it out.

- Calls to indirectly recursive functions

  If a recursion path consists of multiple functions, the stack size may be calculated incorrectly.

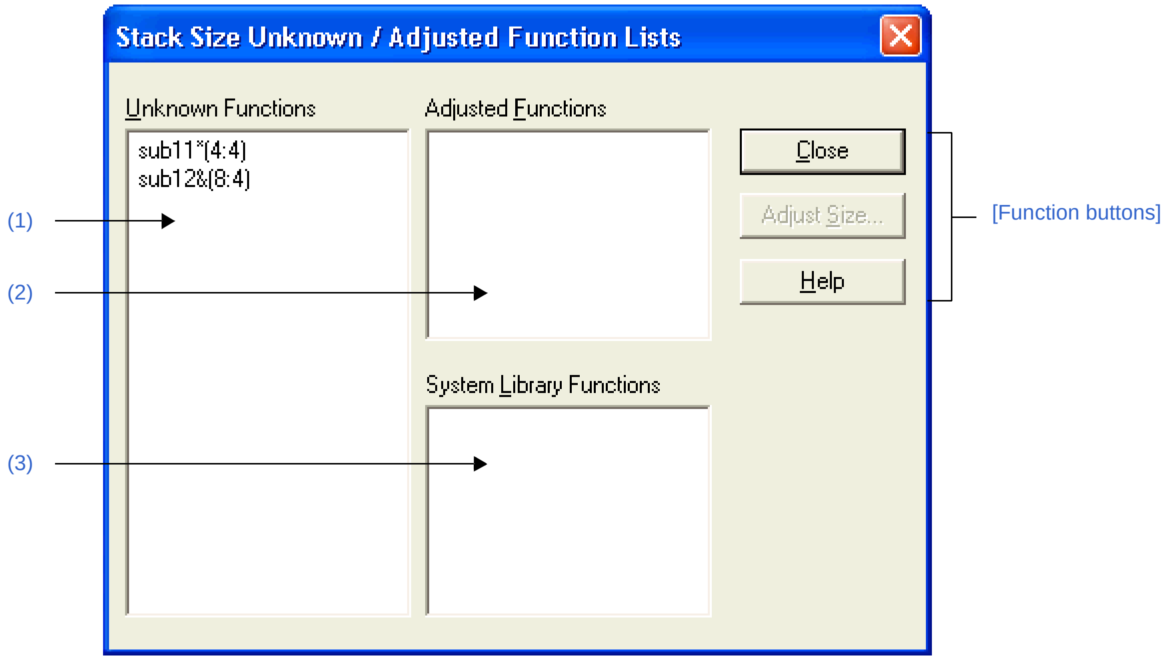
Example Assuming that the frame size of recursive functions "func_rec1/func_rec2" is 8 bytes, if the recursion depth of "func_rec1/func_rec2" is set to 3 in the Adjust Stack Size dialog box, then although the stack size of func1 will be calculated correctly as "(8 + 24) * 3", the stack size of func2 will be calculated as "8 * 3", ignoring calls to func_rec1.

```
void    func_rec1 ( int i );
void    func_rec2 ( int i );
void    func1 ( void );
void    func2 ( void );

void main ( void ) {
    func1 ( );
    func2 ( );
}
void func_rec1 ( int i ) {
    func_rec2 ( i );
}

void func_rec2 ( int i ) {
    if ( i ) {
        func_rec1 ( i - 1 );
    }
}

void func1 ( void ) {
    func_rec1 ( 2 );
}

void func2 ( void ) {
    func_rec2 ( 2 );
}
```

Stack Size Unknown / Adjusted Function Lists dialog box

This dialog box is used to display a list of functions for which the stack usage tracer could not obtain stack information; functions for which information (additional margin, recursion depth, and callee functions) was changed intentionally, and functions for which the stack usage tracer forcibly set an additional margin.

Figure A-50. Stack Size Unknown / Adjusted Function Lists Dialog Box



sub11*(4:4)
sub12&(8:4)

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Stack Usage Tracer window, select the [Stack Size Unknown / Adjusted Function Lists...] from the [Option] menu.

[Description of each area]

(1) [Unknown Functions]

Display a list of "unknown functions" -- functions for which the stack usage tracer could not obtain stack information. This area generally displays unknown functions in the following format.

function name (total stack size : frame size)

Remarks 1. If the unknown function is written in assembly language, then the underscore (_) pre-appended to the symbol name is deleted, and the name is surrounded by square brackets ([ ]); this is displayed as the function name.

2. If the unknown function is a recursive function, then an asterisk (*) is appended to the end of the function name.

3. If the unknown function includes functions called indirectly using function pointers, then an ampersand (&) is appended to the end of the function name.
4. If the unknown function is a static function, then "file name#" is appended to the end of the function name.

(2) [Adjusted Functions]

Display a list of functions for which information (additional margin, recursion depth, or callee functions) has been modified intentionally via the Adjust Stack Size dialog box or a stack size specification file. This area generally displays modified ("adjusted") functions in the following format.

function name (total stack size : frame size : additional margin)

Remarks 1. If the adjusted function is written in assembly language, then the underscore (_) pre-appended to the symbol name is deleted, and the name is surrounded by square brackets ([ ]); this is displayed as the function name.

2. If the adjusted function is a recursive function, then an asterisk (*) is appended to the end of the function name.

3. If the adjusted function includes functions called indirectly using function pointers, then an ampersand (&) is appended to the end of the function name.

4. If the adjusted function is a static function, then "file name#" is appended to the end of the function name.

5. If the only action performed in the Adjust Stack Size dialog box was adding "callee functions", then the display format of this area will be as follows.

function name (total stack size : frame size)

(3) [System Library Functions]

Display a list of automatically configured system library functions for which the frame size is unknown, and the stack usage tracer has forcibly set an additional margin. This area generally displays modified system library functions in the following format.

function name (total stack size : ? : additional margin)

Remarks 1. The underscore (_) pre-appended to the symbol name is deleted, and the name is surrounded by square brackets ([ ]); this is displayed as the function name.

2. An appropriate frame size is added to corresponding system library functions in the stack usage tracer's database as additional margin.

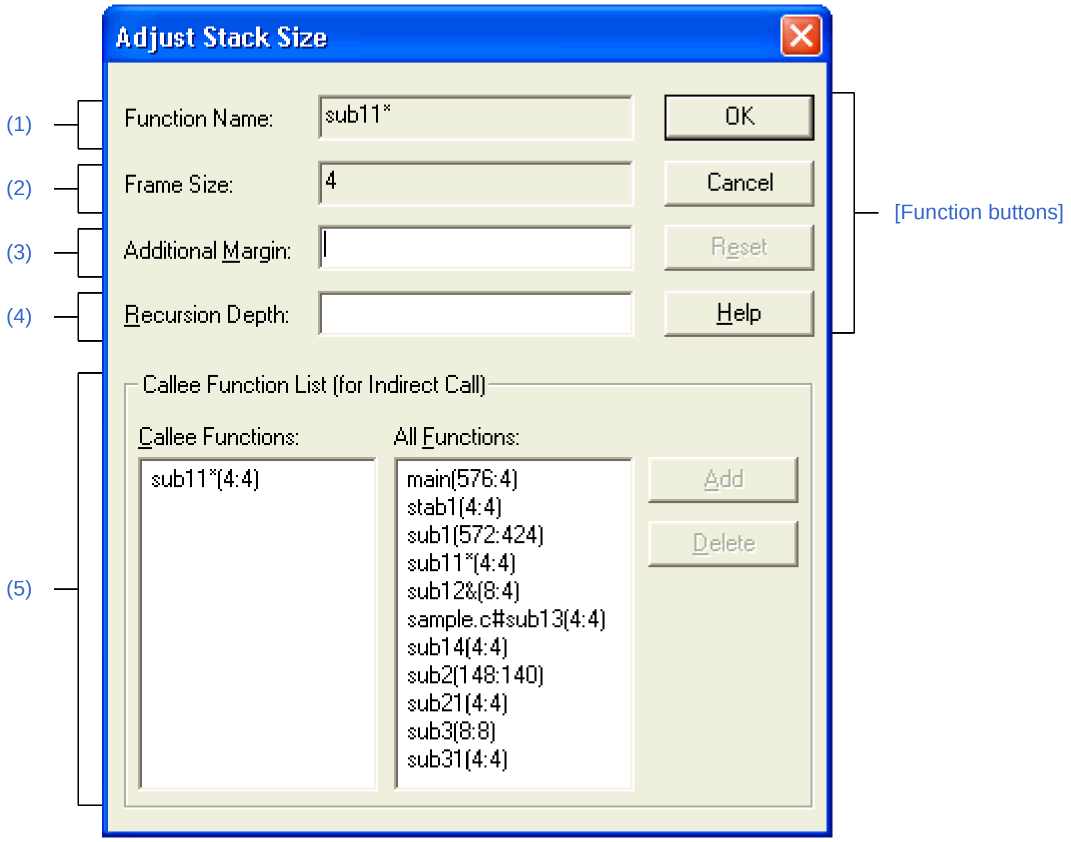
[Function buttons]

Button	Function
Close	Closes this dialog box.
Adjust Size...	Opens the Adjust Stack Size dialog box to change the information (additional margin, recursion depth, and callee functions) for the function selected in the [Unknown Functions]/[Adjusted Functions]/[System Library Functions].
Help	Displays the help of this dialog box.

Adjust Stack Size dialog box

This dialog box is used to change the information (additional margin, recursion depth, and callee functions) for the selected function.

Figure A-51. Adjust Stack Size Dialog Box



Function Name: sub11*

Frame Size: 4

Additional Margin:

Recursion Depth:

Callee Functions:

sub11*(4:4)

All Functions:

main(576:4)
stab1(4:4)
sub1(572:424)
sub11*(4:4)
sub12&(8:4)
sample.c#sub13(4:4)
sub14(4:4)
sub2(148:140)
sub21(4:4)
sub3(8:8)
sub31(4:4)

The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the tree display area/list display area of the Stack Usage Tracer window, select a function, and then select [Adjust Stack Size...] from the [Option] menu.
- On the tree display area/list display area of the Stack Usage Tracer window, select a function, and then click the button from toolbar.
- On the tree display area/list display area of the Stack Usage Tracer window, select a function, and then select [Adjust Stack Size...] from the context menu.
- On the [Unknown Functions]/[Adjusted Functions]/[System Library Functions] of the Stack Size Unknown / Adjusted Function Lists dialog box, select a function, and then click the [Adjust Size...] button.

[Description of each area]

(1) [Function Name]

Display the function name of the selected function.

Remarks 1. If the selected function is written in assembly language or it is a system library function, then the underscore (_) pre-appended to the symbol name is deleted, and the name is surrounded by square brackets ([ ]); this is displayed as the function name.

2. If the selected function is a recursive function, then an asterisk (*) is appended to the end of the function name.

3. If the selected function includes functions called indirectly using function pointers, then an ampersand (&) is appended to the end of the function name.

4. If the selected function is a static function, then "file name#" is appended to the end of the function name.

(2) [Frame Size]

Display the frame size (not including the stack size of callee functions; in bytes) of the selected function.

Remark If the frame size is not known, then a question mark (?) is displayed; if it is over the maximum limit, then "SIZEOVER" is displayed.

(3) [Additional Margin]

Specify the value to forcibly add to the selected function (in bytes), either as a decimal number, or as a hexadecimal number starting with "0x" or "0X".

(4) [Recursion Depth]

Specify the recursion depth, either as a decimal number, or as a hexadecimal number starting with "0x" or "0X".

Remark If the selected function is not a recursive function, then this item will be grayed out.

(5) [Callee Function List (for Indirect Call)] area

(a) [Callee Functions]

Display a list of "callee" functions called by the selected function (functions called indirectly using a function pointer or the like).

This area generally displays callee functions in the following format.

function name (total stack size : frame size : additional margin)

Remarks 1. If the callee function is written in assembly language or it is a system library function, then the underscore (_) pre-appended to the symbol name is deleted, and the name is surrounded by square brackets ([ ]); this is displayed as the function name.

2. If the callee function is a recursive function, then an asterisk (*) is appended to the end of the function name.

3. If the callee function includes functions called indirectly using function pointers, then an ampersand (&) is appended to the end of the function name.

4. If the callee function is a static function, then "file name#" is appended to the end of the function name.

5. Functions added intentionally from [All Functions] by clicking the [Add] button are shown with a plus sign (+) appended to the end of the function name.

(b) [All Functions]

Display a list of functions that can be added as functions called by the selected function ("callee functions").

This area generally displays functions that can be added in the following format.

function name (total stack size : frame size : additional margin)

Remarks 1. If the function that can be added is written in assembly language or it is a system library function, then the underscore (_) pre-appended to the symbol name is deleted, and the name is surrounded by square brackets ([ ]); this is displayed as the function name.

2. If the function that can be added is a recursive function, then an asterisk (*) is appended to the end of the function name.

3. If the function that can be added includes functions called indirectly using function pointers, then an ampersand (&) is appended to the end of the function name.

4. If the function that can be added is a static function, then "file name#" is appended to the end of the function name.

(c) Button area

Add	Adds the function selected in [All Functions] to [Callee Functions]. If no function is selected in [All Functions], then this button will be grayed out.
Delete	Deletes the function selected in [Callee Functions] from [Callee Functions]. If no function is selected in [Callee Functions], then this button will be grayed out.

Remark Functions can only be deleted from [Callee Functions] if the function name ends with a plus sign (+) (functions added from [All Functions] intentionally by clicking [Add]).

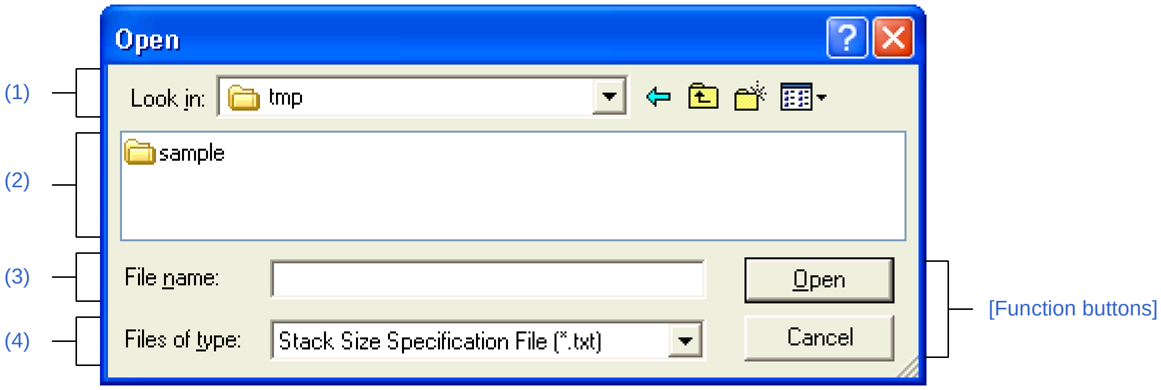
[Function buttons]

Button	Function
OK	Reflects the settings in the Stack Usage Tracer window / save them to the project file (*.prj), then close the dialog.
Cancel	Ignores the setting and closes this dialog box.
Reset	Resets the information (additional margin, recursion depth, and callee functions) for the selected function to the default values. This button will be grayed out if all the information for the selected function has the default values.
Help	Displays the help of this dialog box.

Open dialog box

This dialog box is used to open an existing stack size specification file.

Figure A-52. Open Dialog Box



The following items are explained here.

- [How to open]
- [Description of each area]
- [Function buttons]

[How to open]

- On the Stack Usage Tracer window, select [Load Stack Size Specification File...] from the [File] menu.

[Description of each area]

(1) [Look in] area

Select the folder containing the stack size specification file you wish to open.

(2) List of files

This area displays a list of files matching the conditions selected in [Look in] area and [Files of type] area.

(3) [File name] area

Specify the file name of the stack size specification file to open.

(4) [Files of type] area

Select the type of file to open.

Stack Size Specification File (*.txt)	Text format

[Function buttons]

Button	Function
Open	Opens the specified file.
Cancel	Ignores the setting and closes this dialog box.

APPENDIX B COMMAND REFERENCE

This appendix describes the detailed specifications of each command included in the build tool.

B.1 cx

The cx generates files executable on the target system from source programs described in C language or assembly language.

When using the cx, a single driver controls all phases from the processor to the object converter.

Processing of each phase is shown below.

(1) Compiler

Performs processing of preprocess directives, comment processing, and optimization for a C source program and then generates an assembler source file.

(a) Preprocessor

Performs processing of preprocess directives in a C source program.

When the -P option is specified, a preprocessed file is output.

Remark The -P option is not specified by default.

(b) Syntax analyzer

Performs syntax analysis processing of a C source program and then converts the program into an internal data expression of the compiler.

(c) Common optimization module

Performs optimization for internal data expressions converted from the C source program.

(d) Code generation module

Converts an internal data expression into an assembler source program.

(2) Assembler

Converts an assembler source program into machine language instructions and then generates a relocatable object module file.

(3) Linker

Links an object module file, link directive file, library file and then generates an executable object file (load module file) on the target system.

(4) Symbol file generator

Generates a symbol information file.

This runs when the -Xsfg option is specified.

If the generated symbol information file is specified by the -Xsymbol_file option, the compiler can use the file.

Remarks 1. The -Xsfg and -Xsymbol_file options are not specified by default.

2. When the symbol information file is generated, you need to specify the -Xcube_suite_info option at the same time.

3. When the symbol information file is generated, you need to specify the -Xcube_suite_info option at the same time.

(5) ROMization processor

Performs ROMization processing for a load module file.

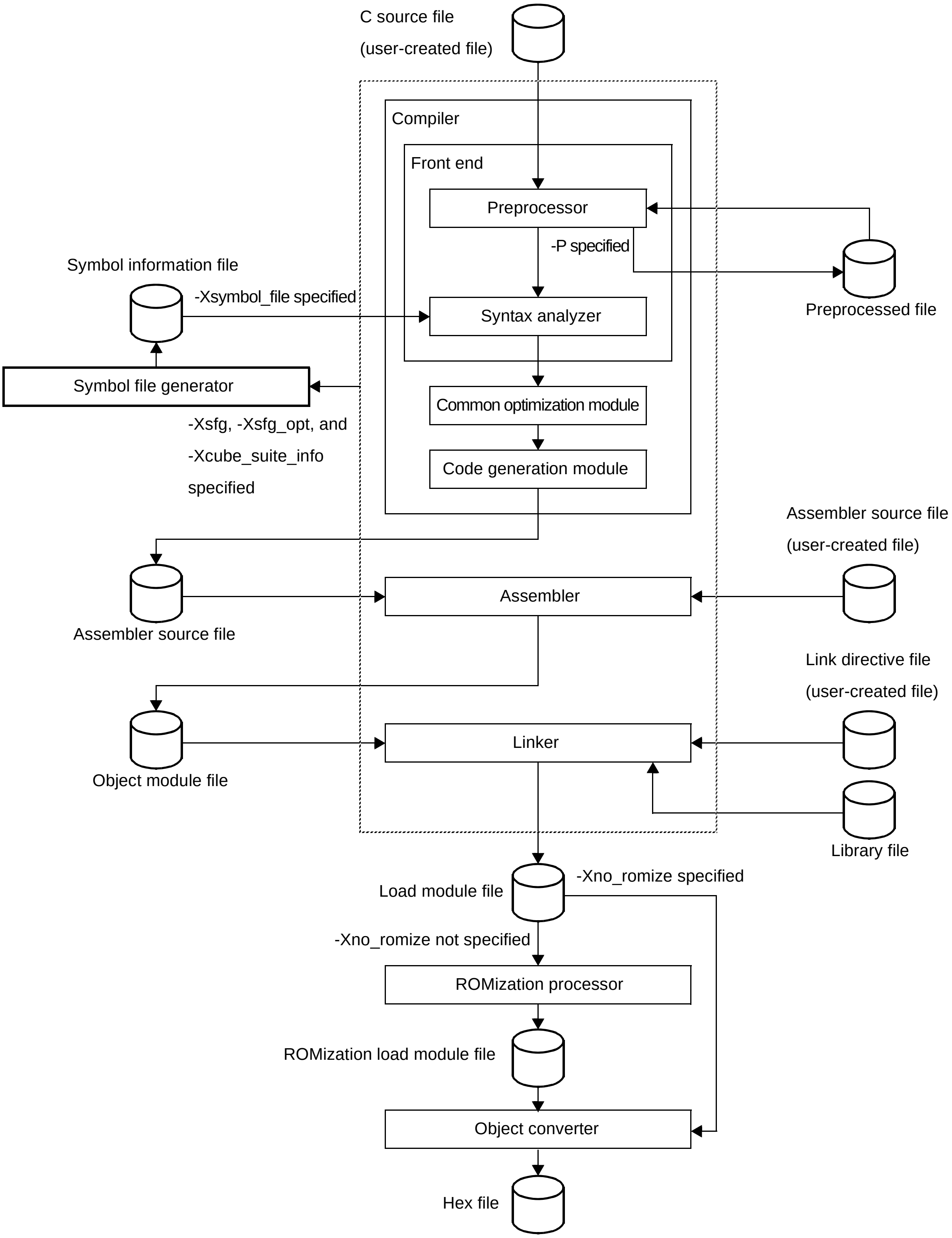
This does not run when the -Xno_romize option is specified.

Remark By default, the -Xno_romize option is not specified. The cx generates a load module file after ROMization processing.

(6) Object converter

Performs hex conversion processing for a load module file and then generates a hex file.

Figure B-1. Operation Flow of cx



B.1.1 I/O files

The I/O files of the cx are shown below.

Table B-1. I/O Files of cx

File Type	Extension	I/O	Description
C source file	.c	I	Source file described in C language This is created by the user.
Preprocessed file	.i[Note 1]	O	File which the execution result of preprocessing for the input file is output This is an ASCII image file. This file is output when the -P option is specified.
Assembler source file	.asm[Note 1]	O	Assembly language file which generated from C source file by compilation
	asm .s	I	Source file which described in assembly language This is created by the user.
Header file	free	I	File which is referred by source files This file is described in C language or assembly language. This is created by the user. The extension is free, but the following is recommended. - #include directive: .h - $include control instruction: .inc
Symbol information file[Note 2]	.sfg[Note 1]	I/O	File which specifies the section to which a variable defined in the C source file is allocated This file is output when the -Xsfg option is specified.
Object module file	.obj[Note 1]	I/O	ELF-format file including machine-language information, relocation information relating to machine-language allocation addresses, and symbol information
	.o	I	
Assemble list file[Note 2]	.prn[Note 1]	O	List file which has information of the assemble result This file is output when the -Xprn_path option is specified.
Link directive file	free	I	File which memory map information when linking is described This is created by the user. The extension is free, but ".dir" is recommended.
Library file	.lib .a	I	File in which two or more object module files are included
Load module file	.lmf[Note 1]	I/O	ELF-format file of the object code of the link result This is the input file when a hex file is output.
Hex file[Note 2]	.hex[Note 1]	O	This file is the load module file converted into hex format.
Link map file[Note 2]	.map[Note 1]	O	This is the list file that has information of the link result. This file is output when the -Xmap option is specified.
Information file for CubeSuite	free	I/O	Information file which CubeSuite uses. The extension is free, but ".cref" is recommended. This file is output when the -Xcube_suite_info option is specified.
Error message file	free	O	File which contains error messages This file is output when the -Xerror_file option is specified.

File Type	Extension	I/O	Description
Command file	free	I	File which contains the parameters of the execution program This is created by the user.

Notes 1. The extension of the output file can be changed by specifying the option.
2. See "CHAPTER 3 BUILD OUTPUT LISTS" for details about output files.

B.1.2 Method for manipulating

This section explains how to manipulate the cx.

(1) Startup from the command line

Enter the following on the command line.

```
>cx[Δoption]...Δfile-name[Δfile-name or option]...
```

[ ]: Can be omitted
...: Pattern in proceeding [ ] can be repeated
Δ: One or more spaces

- Uppercase characters and lowercase characters are distinguished for options.
- When numerical values are specified as parameters of an option, decimal or hexadecimal numbers which starts with "0x" ("0X") can be specified.
  Uppercase characters and lowercase characters are not distinguished for the alphabet of hexadecimal numbers.
  However, only hexadecimal numbers can be specified as the parameter of the -Xhex_fill option.
- When a file name is specified as a parameter of an option, it can include the path (absolute path or relative path).
  When a file name without the path or a relative path is specified, the reference point of the path is the current folder.
- When a parameter of an option includes a space (such as a path name), enclose the parameter in a pair of double quotation marks (" ").
- A file name supported by Windows can be specified.
  However, "@" cannot be used at the beginning of a file name because it is regarded as the command file specification.
  "-" and "+" cannot be also used at the beginning of a file name because they are regarded as the option specification.
- The length that can be specified for a file name depends on Windows (up to 259 characters).
- Uppercase characters and lowercase characters are not distinguished for the alphabet of a file name.
- When the -P option is specified, one file can be specified as input.
  An error will occur if two or more files are specified.
  Otherwise, two or more files can be specified as input.
  Files which have different types (C source file and assembler source file or object module file, and the like) can be mixed.
  Therefore, two or more files with the same name (even if they are in different folders) cannot be specified.
  When two or more files are specified, even if there is an error in one file, processing of the remaining files will continue if processing is possible.
- The generated object module file is not deleted after linking.

The examples of operations on the command line are shown below.

Remark See "B.1.3 Option" for details about each option.

(a) Performing all phases from compilation to linking at once

<1> "file1.c", "file2.asm", and "file3.obj" are read, and then load module file "a.lmf" is generated. Hex file "a.hex" is also generated.

```
>cx -CF3746 file1.c file2.asm file3.obj
```

(b) Performing phases from compilation to assembly, and linking separately

<1> "file1.c" and "file2.asm" are read, and then object module file "file1.obj" and "file2.obj" are generated.

```
>cx -CF3746 -c file1.c file2.asm
```

<2> "file1.obj", "file2.obj", and "file3.obj" are linked, and then load module file "a.lmf" is generated. Hex file "a.hex" is also generated.

```
>cx -CF3746 file1.obj file2.obj file3.obj
```

(c) Performing compilation, assembly, and linking separately

<1> "file1.c" is read, and then object module file "file.obj" is generated.

```
>cx -CF3746 -c file1.c
```

<2> "file2.asm" is read, and then object module file "file2.obj" is generated.

```
>cx -CF3746 -c file2.asm
```

<3> "file1.obj", "file2.obj", and "file3.obj" are linked, and then load module file "a.lmf" is generated. Hex file "a.hex" is also generated.

```
>cx -CF3746 file1.obj file2.obj file3.obj
```

(2) Startup from a command file

A command file is a file that options and file names specified for the cx command are described.

The cx command treats the contents of a command file as if they were command-line arguments.

Use a command file when the arguments will not fit on the command line, or when same options are specified repeatedly each time the command is executed.

(a) Cautions about description of a command file

- The arguments to be specified can be coded over several lines.
  However, you cannot start a new line within the name of the option or file.
- A command file cannot be nested.
- The character code contents of a command file cannot be specified by using the -Xcharacter_set option.
  If you use characters other than ASCII in the command file, use the UTF-8 file with BOM.
- The following characters are treated as special characters.
  These special characters themselves are not included in the command line of the cx command and deleted.

" (double quotation mark)	The character string until the next double quotation mark is treated as a contiguous character string.
# (sharp)	If this is specified at the beginning of a line, characters on that line before the end of the line are interpreted as a comment.
^ (circumflex)	The character immediately following this is not treated as a special character.

(b) Example of command file specification

Create command file cfile using an editor.

```
-CF3746          ... Specifies the device
-Utest           ... Deletes the definition of macro "test"
-oobject.obj     ... Specifies the object module file name
-c               ... Specifies the execution until the assemble phase
file.c           ... Specifies the file to be compiled
```

When the cx command is executed on the command line, specify command file "cfile" by command file specification option "@".

```
>type cfile      <- displays the contents of cfile
-CF3746
-Utest
-oobject.obj
-c
file.c
>cx @cfile       <- Same operation as "cx -CF3746 -Utest -oobject.obj -c file.c"
```

(3) Set options in CubeSuite

This section describes how to set cx options from CubeSuite.

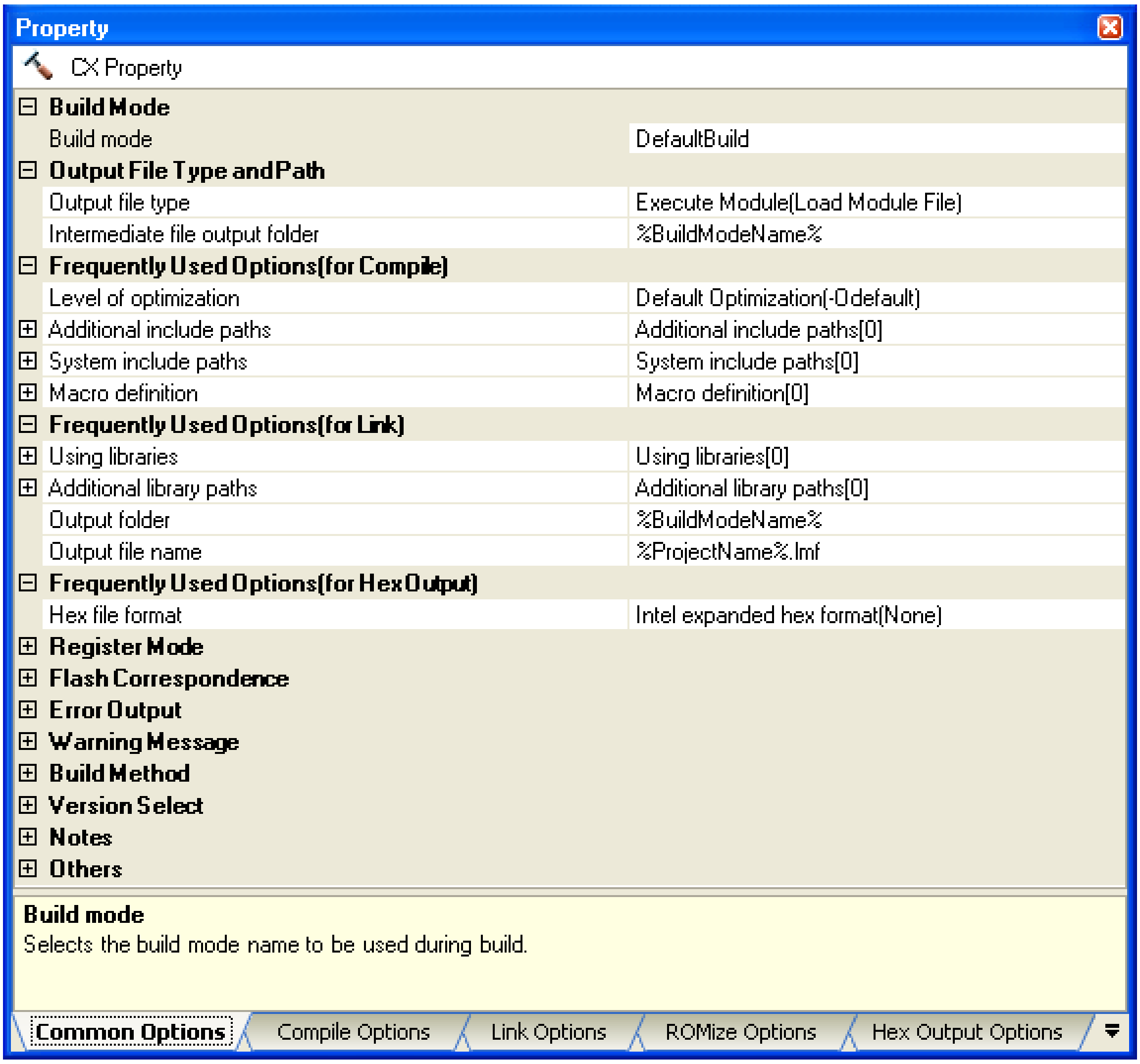
On the CubeSuite's Project Tree panel, select the Build Tool node. Next, select the [View] menu >> [Property].

The Property panel opens.

Next, select the [Common Options] tab/[Compile Options] tab/[Assemble Options] tab/[Link Options] tab/[ROMize Options] tab/[Hex Output Options] tab.

You can set the various cx options by setting the necessary properties in this tab.

Figure B-2. Property Panel



Build Mode	
Build mode	DefaultBuild
Output File Type and Path	
Output file type	Execute Module(Load Module File)
Intermediate file output folder	%BuildModeName%
Frequently Used Options(for Compile)	
Level of optimization	Default Optimization(-Odefault)
Additional include paths	Additional include paths[0]
System include paths	System include paths[0]
Macro definition	Macro definition[0]
Frequently Used Options(for Link)	
Using libraries	Using libraries[0]
Additional library paths	Additional library paths[0]
Output folder	%BuildModeName%
Output file name	%ProjectName%.lmf
Frequently Used Options(for Hex Output)	
Hex file format	Intel expanded hex format(None)
Register Mode	
Flash Correspondence	
Error Output	
Warning Message	
Build Method	
Version Select	
Notes	
Others	

Build mode
Selects the build mode name to be used during build.

B.1.3 Option

This section explains cx options for each phase.

Compile phase	->	See "(1) Compile options"
Assemble phase	->	See "(2) Assemble options"
Link phase	->	See "(3) Link options"
ROMize phase	->	See "(4) ROMize options"
Hex output phase	->	See "(5) Hex output options"

(1) Compile options

The types and explanations for options of the compile phase are shown below.

Table B-2. Compile Options

Classification	Option	Description
Version/help display specification	-V	This option displays the version information of the cx.
	-h	This option displays the descriptions of the cx options.
Output file specification	-o	This option specifies the output file name.
	-Xobj_path	This option specifies where the object module file generated during compilation is to be saved.
	-Xasm_path	This option specifies where an assembler source file generated during compilation is to be saved.
	-Xprn_path	This option specifies where an assemble list file is to be saved.
	-Xtemp_path	This option specifies the temporary folder.
	-Xlink_output	This option saves the load module file before ROMization processing.
Source debugging control	-g	This option outputs information for source debugging.
	-Xkeep_access_size	This option prohibits changing the memory access size.
Device specification	-C	This option specifies the target device.
	-Xcommon	This option specifies that an object module file common to the various devices is generated.
	-Xdev_path	This option specifies the folder to search device files.
Processing interrupt specification	-P	This option executes only preprocessing for the input file.
	-S	This option does not execute processing after assembling.
	-c	This option does not execute processing after linking.
Preprocessor control	-D	This option defines preprocessor macros.
	-U	This option deletes the preprocessor macro definition by the -D option.
	-I	This option specifies the folder to search include files.
	-Xpreprocess	This option controls outputting the result of preprocessing.
C language control	-Xansi	This option processes as making C source program comply strictly with the ANSI standard.
	-Xchar	This option specifies signed for char type.
	-Xenum_type	This option specifies which integer type the enumeration type handles.
	-Xdef_var	This option handles tentative definition of variables as definition.
Variable information specification	-Xsymbol_file	This option uses a symbol information file.
Japanese/Chinese character control	-Xcharacter_set	This option specifies the Japanese/Chinese character code.

Classification	Option	Description
Optimization specification	-O	This option specifies the optimization level or the details of each optimization items.
	-Xsort_var	This option sorts external variables.
	-Xinline_strcpy	This option performs inline expansion of standard library functions "strcpy", "strcmp", "memcpy", and "memset" calls.
	-Xpro_epi_runtime	This option specifies whether or not to perform prologue/epilogue processing of the function through runtime library calls.
	-Xcall_lib	This option suppresses inline expansion of the library function.
Generated code control	-Xpack	This option performs the structure packing.
	-Xpass_source	This option outputs a C source program as a comment to the assembler source file.
	-Xswitch	This option specifies a mode in which the code of a switch statement is to be output.
	-Xword_case	This option generates a 4-byte branch table.
	-Xr	This option specifies the register that external variables are allocated.
	-Xreg_mode	This option specifies the register mode.
	-Xsdata	This option specifies the maximum size of data allocated to the .sdata or .sbss section.
	-Xsconst	This option specifies that constant data is allocated to the .sconst section.
	-Xfloat [V850E2V3]	This option controls generating floating-point calculation instructions.
	-Xfar_jump	This option controls outputting far jump.
	-Xdiv [V850E2V3]	This option generates the div and divu instructions for division.
Assembler control specification	-Xasm_far_jump [V850E2][V850E2V3]	This option controls outputting far jump for an assembler source file.
Library link control	-l	This option specifies the library file to be used during linking.
	-L	This option specifies the folder to search library files.
	-Xno_stdlib	This option suppresses linking the standard library.
	-Xno_startup	This option suppresses linking the startup routine.
	-Xstartup	This option specifies the startup routine.

Classification	Option	Description
Link control	-Xlink_directive	This option specifies the link directive file.
	-Xmap	This option outputs the link map file.
	-Xentry_address	This option specifies the entry point address.
	-Xrelinkable_object	This option generates the relocatable object module file.
	-Xregmode_info	This option outputs detailed information when different register modes are mixed.
	-Xforce_link	This option continues link processing when the internal ROM/RAM overflows.
	-Xsdata_info	This option outputs information that can be used as a yardstick for the parameter of the -Xsdata option to the standard output.
	-Xtwo_pass_link	This option performs linking in the 2-pass mode.
	-Xignore_address_error	This option continues link processing if an illegality is found during relocation processing when linking.
	-Xmultiple_symbol	This option outputs an error message for all multi-defined external symbols.
	-Xlink_check_off	This option suppresses checking when linking.
	-Xalign_fill	This option specifies the filling value of align holes.
	-Xrescan	This option rescans the library file specified by the -l option.
	-Xstrip	This option generates the load module file from which the debug information, line number information, and global pointer table have been removed.
	-Xflash	This option generates the load module file for the flash area.
	-Xflash_ext_table	This option specifies the start address value of the branch table for the boot-flash re-link function.
ROMization control	-Xno_romize	This option suppresses ROMization processing.
	-Xrompcrt	This option specifies the ROMization area reservation code file.
	-Xrompsec_start	This option specifies the start address of the rompsec section.
	-Xrompsec_data	This option specifies the data section included in the rompsec section.
	-Xrompsec_text	This option specifies the text section included in the rompsec section.
	-Xrompsec_only	This option generates the load module file that has only the rompsec section.
	-Xromize_check_off	This option omits error checking under ROMization.

Classification	Option	Description
Hex output control	-Xhex	This option specifies the hex file name.
	-Xhex_only	This option executes only hex output.
	-Xhex_format	This option specifies the format of the hex file to be output.
	-Xhex_fill	This option specifies fill processing of the hex file.
	-Xhex_section	This option converts the codes in the specified section in hex format and outputs them.
	-Xhex_block_size	This option specifies the maximum length of the block.
	-Xhex_offset	This option specifies the offset of the address to be output.
	-Xhex_null	This option generates as many null characters as the size of the section of data without an initial value.
	-Xhex_symtab	This option converts the symbol table and outputs it.
	-Xhex_rom_less	This option does not use the information of the internal ROM area when the hex file is filled.
Multi-core support specification	-Xmulti	This option specifies the generation of a subprogram of a multi-core program.
	-Xmulti_link	This option specifies the linking of multi-core subprograms.
Information file output control	-Xcube_suite_info	This option outputs the CubeSuite information file.
	-Xno_cube_suite_info	This option suppresses the output of the CubeSuite information file.
	-Xsfg	This option generates a symbol information file.
	-Xsfg_opt	This option outputs the optimum allocation information.
	-Xsfg_size_tidata	This option specifies the size of .tidata section.
	-Xsfg_size_tidata_byte	This option specifies the size of .tidata.byte section.
	-Xsfg_size_sidata	This option specifies the size of .sidata section.
	-Xsfg_size_sedata	This option specifies the size of .sedata section.
	-Xsfg_size_sdata	This option specifies the size of .sdata section.
Error output control	-Xerror_file	This option outputs error messages to a file.
Warning message output control	-Xwarning	This option outputs the specified warning message.
	-Xno_warning	This option suppresses outputting warning messages of the specified number.
Phase individual option specification	-Xasm_option	This option specifies the file to be assembled.
	-Xlk_option	This option specifies the file to be linked.
Command file specification	@	This option specifies a command file.

Table B-3. Mark Used in Option Descriptions

[V850E2]	This option is only for devices with the V850E2 core and V850E2 instruction set architecture.
[V850E2V3]	This option is only for devices with the V850E2 core and V850E2V3 instruction set architecture.

Version/help display specification

The version/help display specification options are as follows.

- -V
- -h

-V

This option displays the version information of the cx.

[Specification format]

```
-V
```

- Interpretation when omitted
  Compilation is performed without displaying the version information of the cx.

[Detailed description]

- This option outputs the version information of the cx to the standard error output.
  It does not execute compilation.

[Example of use]

- To output the version information of the cx to the standard error output, describe as:

```
>cx -CF3746 -V
```

-h

This option displays the descriptions of the cx options.

[Specification format]

```
-h
```

- Interpretation when omitted
  The descriptions of the cx options are not displayed.

[Detailed description]

- This option outputs the descriptions of the cx options to the standard error output.
  It does not execute compilation.

[Example of use]

- To output the descriptions of the cx options to the standard error output, describe as:

```
>cx -CF3746 -h
```

Output file specification

The output file specification options are as follows.

- -o
- -Xobj_path
- -Xasm_path
- -Xprn_path
- -Xtemp_path
- -Xlink_output

-o

This option specifies the output file name.

[Specification format]

```
-ofile
```

- Interpretation when omitted
  The file is output to the current folder.
  - If this option is specified with the -P option
    The output file name will be the input file name with the extension replaced by ".i".
  - If this option is specified with the -S option
    The output assembler source file name will be the source file name with the extension replaced by ".asm".
  - If this option is specified with the -c option
    The output object module file name will be the source file name with the extension replaced by ".obj".
  - Other than above
    The output load module file name is "a.lmf".

[Detailed description]

- This option specifies the output file name as *file*.
- If *file* already exists, it will be overwritten.
- This option is valid when processing is interrupted by specifying the -P, -S, or -c option.
  - If this option is specified with the -P option
    It is assumed that is the name of the file containing the results of preprocessing performed on the input file has been specified as *file*.
  - If this option is specified with the -S option
    It is assumed that an assembler source file name has been specified as *file*.
  - If this option is specified with the -c option
    It is assumed that an object module file name has been specified as *file*.
  - Other than above
    It is assumed that a load module file name has been specified as *file*.
- An error will occur if two or more files are output.
- An error will occur if *file* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Individual Compile Options] tab, [Object module file name] in the [Output File] category

[Example of use]

- To output the load module file with "sample.lmf" as the file name, describe as:

```
>cx -CF3746 -osample.lmf main.c
```

-Xobj_path

This option specifies where the object module file generated during compilation is to be saved.

[Specification format]

```
-Xobj_path[=path]
```

- Interpretation when omitted
  The object module file is saved under the source file name with the extension replaced by ".obj" to the current folder.
  However, the object module file will not be saved if one source file is specified as input and the -c option is not specified.

[Detailed description]

- This option saves the object module file generated during compilation to *path*.
  - If an existing file is specified as *path*
    If one object module file is output, it will be saved with *path* as the file name.
    If two or more object module files are output, an error will occur.
  - If an existing folder is specified as *path*
    The object module file is saved under the source file name with the extension replaced by ".obj" to *path*.
  - If the specified folder or file as *path* does not exist
    An error will occur.
  - If “=*path*“ is omitted
    The object module file is saved under the source file name with the extension replaced by ".obj" to the current folder.
- If two or more files with the same name (even if they are in different folders) are specified as source files, then a warning is output, and an object module file is only saved for the last source file to be specified.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Intermediate file output folder] in the [Output File Type and Path] category

[Example of use]

- To save the object module file generated during compilation with "sample.obj" as a file name, describe as:

```
>cx -CF3746 -Xobj_path=sample.obj main.c
```

-Xasm_path

This option specifies where an assembler source file generated during compilation is to be saved.

[Specification format]

```
-Xasm_path[=path]
```

- Interpretation when omitted
  An assembler source file will not be output (except when specifying the -S option).

[Detailed description]

- This option saves the assembler source file generated during compilation to *path*.
  - If an existing file is specified as *path*
    If one assembler source file is output, it will be saved with *path* as the file name.
    If two or more assembler source files are output, an error will occur.
  - If an existing folder is specified as *path*
    The assembler source file is saved under the C source file name with the extension replaced by ".asm" to *path*.
  - If the specified folder or file as *path* does not exist
    An error will occur.
  - If "=*path*" is omitted
    The assembler source file is saved under the C source file name with the extension replaced by ".asm" to the current folder.
- If two or more files with the same name (even if they are in different folders) are specified as source files, then a warning is output, and an assembler source file is only saved for the last source file to be specified.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Output assembler source file] in the [Output File] category
  - From the [Compile Options] tab, [Output folder for assembler source file] in the [Output File] category
  - From the [Individual Compile Options] tab, [Output assembler source file] in the [Output File] category
  - From the [Individual Compile Options] tab, [Output folder for assembler source file] in the [Output File] category

[Example of use]

- To save the assembler source file generated during compilation with "sample.asm" as a file name, describe as:

```
>cx -CF3746 -Xasm_path=sample.asm main.c
```

-Xprn_path

This option specifies where an assemble list file is to be saved.

[Specification format]

```
-Xprn_path[=path]
```

- Interpretation when omitted
  An assemble list file will not be output.

[Detailed description]

- This option outputs an assemble list file when assembling and saves it to *path*.
  - If an existing file is specified as *path*
    If one assemble list file is output, it will be saved with *path* as the file name.
    If two or more assemble list files are output, an error will occur.
  - If an existing folder is specified as *path*
    The assemble list file is saved under the source file name with the extension replaced by ".prn" to *path*.
  - If the specified folder or file as *path* does not exist
    An error will occur.
  - If "=*path*" is omitted
    The assemble list file is saved under the source file name with the extension replaced by ".prn" to the current folder.
- If two or more files with the same name (even if they are in different folders) are specified as source files, then a warning is output, and an assemble list file is only saved for the last source file to be specified.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Output assemble list file] in the [Assemble List] category
  - From the [Compile Options] tab, [Output folder for assemble list file] in the [Assemble List] category
  - From the [Individual Compile Options] tab, [Output assemble list file] in the [Assemble List] category
  - From the [Individual Compile Options] tab, [Output folder for assemble list file] in the [Assemble List] category

[Example of use]

- To save the assemble list file that is output when assembling with "sample.prn" as a file name, describe as:

```
>cx -CF3746 -Xprn_path=sample.prn main.c
```

-Xtemp_path

This option specifies the temporary folder.

[Specification format]

```
-Xtemp_path=path
```

- Interpretation when omitted
  The temporary folder is determined according to the following sequence.

(1) Folder specified by environmental variable TEMP

(2) Folder specified by environmental variable TMP

(3) Current folder

[Detailed description]

- This option specifies *path* as the folder for generating temporary files that are used internally.
- If *path* does not exist, a warning will be output and the temporary folder will be determined.

(1) Folder specified by environmental variable TEMP

(2) Folder specified by environmental variable TMP

(3) Current folder

- An error will occur if *path* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Temporary folder] in the [Others] category

[Example of use]

- To specify folder "D:\tmp" as the temporary folder, describe as:

```
>cx -CF3746 -Xtemp_path=D:\tmp main.c
```

-Xlink_output

This option saves the load module file before ROMization.

[Specification format]

```
-Xlink_output=file
```

- Interpretation when omitted
  The load module file will not be saved before ROMization.

[Detailed description]

- This option saves the load module file as *file* before ROMization.
- An error will occur if *file* is omitted.
- If this option and the -Xno_romize option are specified at the same time, a warning will be output and this option will be ignored.

[Example of use]

- To save the load module file with "sample.lmf" as the file name before ROMization, describe as:

```
>cx -CF3746 -Xlink_output=sample.lmf main.c
```

Source debugging control

The source debugging control options are as follows.

- -g
- -Xkeep_access_size

-g

This option outputs information for source debugging.

[Specification format]

```
-g
```

- Interpretation when omitted
  Information for source debugging will not be output.

[Detailed description]

- This option outputs information for source debugging to the output file.
- Source debugging can be performed by specifying this option.
- If this option and an optimization option are specified at the same time, the ease of debugging could be affected. See "(2) Effects of optimization on debugging" for details.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Add debug information] in the [Debug Information] category
  - From the [Individual Compile Options] tab, [Add debug information] in the [Debug Information] category

[Example of use]

- To output information for source debugging to the output file, describe as:

```
>cx -CF3746 -g main.c
```

-Xkeep_access_size

This option prohibits changing the memory access size.

[Specification format]

```
-Xkeep_access_size
```

- Interpretation when omitted
  1-bit manipulation instructions (set1, clr1, tst1, and not1) may be generated instead of load and store instructions.

[Detailed description]

- This option prohibits replacing load and store instructions (except for byte access) with 1-bit manipulation instructions (set1, clr1, tst1, and not1).
- This option is enabled when read or write events are set for variables during debugging.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Prohibit the operation that changes memory access size] in the [Optimization(Details)] category
  - From the [Individual Compile Options] tab, [Prohibit the operation that changes memory access size] in the [Optimization(Details)] category

[Example of use]

- To prohibit replacing load and store instructions with 1-bit manipulation instructions, describe as:

```
>cx -CF3746 -Xkeep_access_size main.c
```

Device specification

The device specification options are as follows.

- -C
- -Xcommon
- -Xdev_path

-C

This option specifies the target device.

[Specification format]

```
-Cdevice
```

- Interpretation when omitted
  If the -Xcommon option is specified, the behavior is in accordance with that specification.
  Otherwise, an error will occur (except when specifying the -V, -h, or -P option).
  An error will occur when linking is performed.

[Detailed description]

- This option specifies the target device.
- See the user's manual of each device file about device types that can be specified as *device*.
- If *device* does not exist (the corresponding device file does not exist), an error will occur.
- An error will occur if *device* is omitted.
- This option cannot be omitted when linking is performed.

[Example of use]

- To specify uPD70F3746 as the target device, describe as:

```
>cx -CF3746 main.c
```

-Xcommon

This option specifies that an object module file common to the various devices is generated.

[Specification format]

```
-Xcommon=series
```

- Interpretation when omitted
  If the -C option is specified, the behavior is in accordance with that specification.
  Otherwise, an error will occur.

[Detailed description]

- This option specifies that an object module file common to the various devices is generated.
- When this option is specified, only instructions are used in the instruction set architecture of the target, and common magic number *series* that supports the instruction set architecture is embed into the object module file.
- The items that can be specified as *series* are shown below.
  An abort error will occur if any other item is specified.

v850e	It is possible to link to the model with the instruction set architecture superior to V850E (V850E, V850E2, or V850E2V3), which is specified as the target device.
v850e2	It is possible to link to the model with the instruction set architecture superior to V850E2 (V850E2 or V850E2V3), which is specified as the target device.
v850e2v3	It is possible to link to the model with the V850E2V3 instruction set architecture, which is specified as a target device. If the instruction set architecture of the target device is V850E2V3, we recommend specifying this item to get the best performance.

- An error will occur if *series* is omitted.
- If this option and the -C option are specified at the same time, the processing is shown below.

Parameter *series* of -Xcommon	Device Specified by -C		
	V850E	V850E2	V850E2V3
v850e	Normal processing	Replaces with -Xcommon=v850e2 (Outputs a warning)	Replaces with -Xcommon=v850e2v3 (Outputs a warning)
v850e2	Normal processing (Outputs a warning)	Normal processing	Replaces with -Xcommon=v850e2v3 (Outputs a warning)
v850e2v3	Normal processing (Outputs a warning)	Normal processing (Outputs a warning)	Normal processing

If the instruction set architecture of the device specified by the -C option is the one specified by this option, both options are processed.
If the instruction set architecture of the device specified by the -C option is inferior to the one specified by this option (V850E2V3 > V850E2 > V850E), a warning is output and both options are processed.

If the instruction set architecture of the device specified by the -C option is superior to the one specified by this option, a warning is output and parameter *series* of this option is replaced by the instruction set architecture of the device specified by the -C option.

- This option is equivalent to the following property in CubeSuite.
    - From the [Common Options] tab, [Output common object module file for various devices] in the [Output File Type and Path] category

[Example of use]

- To embed the magic number common to models with the instruction set architectures superior to V850E into the object module file to be generated, describe as:

```
>cx -Xcommon=v850e -c main.c
```

-Xdev_path

This option specifies the folder to search device files.

[Specification format]

```
-Xdev_path=path
```

- Interpretation when omitted
  The device file is searched from the standard device file folder.

[Detailed description]

- This option searches a device file from folder *path*.
- If the folder as *path* does not exist, or if the device file specified by the -C option cannot be found, a warning will be output and the standard device file folder[Note] will be searched.
  If it cannot be found, an error will occur.

Note The device file is searched in the following order.

[V850E2V3]

1. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device\V850E2\Devicefile
2. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device\V850\Devicefile
3. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device_Custom\ Devicefile

[V850E/V850E2]

1. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device\V850\Devicefile
2. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device_Custom\ Devicefile

- An error will occur if *path* is omitted.

[Example of use]

- To search a device file from folder C:\NECTools32\dev, describe as:

```
>cx -CF3746 -Xdev_path=C:\NECTools32\dev main.c
```

Processing interrupt specification

The processing interrupt specification options are as follows.

- -P
- -S
- -c

-P

This option executes only preprocessing for the input file.

[Specification format]

```
-P
```

- Interpretation when omitted
  Processing is continued after preprocessing.
  The preprocessed file are not output.

[Detailed description]

- This option executes only preprocessing for the input file and outputs the results to a file.
- An error will occur if two or more input files are specified.
- The output file name will be the input file name with the extension replaced by ".i".
- The output file name can be specified by specifying this option and the -o option.
- The contents of the output file can be controlled by specifying the -Xpreprocess option.
- If this option and the -Xchar option are specified at the same time, the -Xchar option will be invalid.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Output preprocessed source file] in the [Output File] category
  - From the [Individual Compile Options] tab, [Output preprocessed source file] in the [Output File] category

[Example of use]

- To execute only preprocessing for the input file and output the results to file "main.i", describe as:

```
>cx -CF3746 -P main.c
```

-S

This option does not execute processing after assembling.

[Specification format]

```
-S
```

- Interpretation when omitted
  Processing is continued after assembling.

[Detailed description]

- This option does not execute processing after assembling.
- The assembler source file is output under the source file name with the extension replaced by ".asm".
- The output file name can be specified by specifying this option and the -o option.

[Example of use]

- To output assembler source file "main.asm" without executing any processing after the assembling, describe as:

```
>cx -CF3746 -S main.c
```

-c

This option does not execute processing after linking.

[Specification format]

```
-c
```

- Interpretation when omitted
  Processing is continued after linking.

[Detailed description]

- This option does not execute processing after linking.
- The object module file is output under the source file name with the extension replaced by ".obj".
- The output file name can be specified by specifying this option and the -o option.

[Example of use]

- To output object module file "main.obj" without executing any processing after the linking, describe as:

```
>cx -CF3746 -c main.c
```

Preprocessor control

The preprocessor control options are as follows.

- -D
- -U
- -I
- -Xpreprocess

-D

This option defines preprocessor macros.

[Specification format]

```
-Dname[=def][name[=def]]...
```

- Interpretation when omitted

  None

[Detailed description]

- This option defines *name* as a preprocessor macro.
- This is equivalent to adding "#define *name* def" at the beginning of the C source program.
- If *name* contains characters that are allowed in an assembler symbol, but which cannot be used in a preprocessor macro ("@", ".", and "~"), a warning will be displayed, and it is defined as an assembler symbol only.
- An error will occur if *name* is omitted.
- If “=*def*“ is omitted, *def* is regarded as 1.
- This option can be specified more than once.
- If both this option and -U option are specified for the same preprocessor macro, the option specified last will be valid.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Macro definition] in the [Frequently Used Options(for Compile)] category
  - From the [Compile Options] tab, [Macro definition] in the [Preprocess] category
  - From the [Individual Compile Options] tab, [Macro definition] in the [Preprocess] category

[Example of use]

- To define "sample=256" as a preprocessor macro, describe as:

```
>cx -CF3746 -Dsample=256 main.c
```

-U

This option deletes the preprocessor macro definition by the -D option.

[Specification format]

```
-Uname[,name]...
```

- Interpretation when omitted
  None

[Detailed description]

- This option deletes the definition of preprocessor macro *name* by the -D option.
- This is equivalent to adding "#undef *name*" at the beginning of the C source program.
- An error will occur if *name* is omitted.
- This option cannot delete the definition by describing "#define *name def*".
- This option can be used to undefine C language macros that have been defined already, but it cannot undefine the following macros: __LINE__, __FILE__, __DATE__, __TIME__, or __CX__.
  An error will occur if these are specified for *name*.
- This option can be specified more than once.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Macro definition] in the [Preprocess] category
  - From the [Individual Compile Options] tab, [Macro definition] in the [Preprocess] category

[Example of use]

- To delete the definition of preprocessor macro "test" by the -D option, describe as:

```
>cx -CF3746 -Utest main.c
```

-I

This option specifies the folder to search include files.

[Specification format]

```
-Ipath[,path]...
```

- Interpretation when omitted
  The include file is searched from the standard include file folder.

[Detailed description]

- This option specifies the folder to search include files that are read by preprocessor directive "#include" as *path*.
  Include files are searched according to the following sequence.

(1) Folder with source files (When files are specified by using " ")

(2) Path specified by the -I option

(3) Standard include file folder[Note]

Note *CubeSuite install folder*\Cubesuite\CX\V*x.xx*\inc

- If *path* does not exist, a warning will be output.
- An error will occur if *path* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Additional include paths] in the [Frequently Used Options(for Compile)] category
  - From the [Common Options] tab, [System include paths] in the [Frequently Used Options(for Compile)] category
  - From the [Compile Options] tab, [Additional include paths] in the [Preprocess] category
  - From the [Compile Options] tab, [System include paths] in the [Preprocess] category
  - From the [Individual Compile Options] tab, [Additional include paths] in the [Preprocess] category
  - From the [Individual Compile Options] tab, [Use whole include paths specified for build tool] in the [Preprocess] category

[Example of use]

- To search include files from the current folder, folder D:\include, the standard folder in that order, describe as:

```
>cx -CF3746 -ID:\include main.c
```

-Xpreprocess

This option controls outputting the result of preprocessing.

[Specification format]

```
-Xpreprocess=string[,string]
```

- Interpretation when omitted
  The comments and line number information of the C source are not output to the preprocessed file.

[Detailed description]

- This option outputs the comments and line number information of the C source to the preprocessed file.
- This option is valid only when the -P option is specified.
  If the -P option is not specified, this option will be ignored.
- The items that can be specified as *string* are shown below.
  An abort error will occur if any other item is specified.

comment	Outputs the comments of the C source.
line	Outputs line number information[Note].

Note The format of the line number information is shown below.

```
#line line-number “file-name“
```

- *line-number* is a decimal number, and the maximum value is the maximum number of unsigned int.
- In the full path of *file-name*, "\\" is converted to "\", and "" to '\"'.
  Other than printable characters (including spaces) are output as *3-digit octal number* (e.g. "\\%03o").
  Line feed characters are converted to "\\n".
- If an input source file contains the preprocessor directive '#*number* "*string*"' or '#line *number* "*string*"', then *number* is used as *line-number*, and *string* as *file-name*.

- An error will occur if *string* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Output C source comments to preprocessed file] in the [Preprocess] category
  - From the [Compile Options] tab, [Output line number information to preprocessed file] in the [Preprocess] category
  - From the [Individual Compile Options] tab, [Output C source comments to preprocessed file] in the [Preprocess] category
  - From the [Individual Compile Options] tab, [Output line number information to preprocessed file] in the [Preprocess] category

[Example of use]

- To output the comments and line number information of the C source to the preprocessed file, describe as:

```
>cx -CF3746 -Xpreprocess=comment,line -P main.c
```

The following example is equivalent to the example above.

```
>cx -CF3746 -Xpreprocess=comment -Xpreprocess=line -P main.c
```

C language control

The C language control options are as follows.

- -Xansi
- -Xchar
- -Xenum_type
- -Xdef_var

-Xansi

This option processes as making C source program comply strictly with the ANSI standard.

[Specification format]

```
-Xansi
```

- Interpretation when omitted
  Compatibility with the conventional C language specifications is conferred and processing continues after warning is output.

[Detailed description]

- This option processes as making C source program comply strictly with the ANSI standard[Note] and outputs an error or warning for a specification that violates the standard.
- When this option is specified, macro name "__STDC__" is defined as the macro that the value is 1.
- Processing when compiling in strict adherence to the language specification is as follows.
  - Bit fields
    An error will occur if a type other than an int type is specified in a bit field.
    If this option is not specified, a warning will be output and specifying a type other than an int type will be enabled.
  - #line-number
    An error will occur.
    If this option is not specified, "#line-number" will be handled in the same way as "#line line-number".
  - Argument of function for which #pragma inline is specified
    If the type of the return value or parameter is different but type conversion is possible between the specified function call and definition, an error will occur.
    If this option is not specified, the type of the return value is converted to the type at the call side, the parameters are converted to the type of the function definition, and inline expansion is performed.
  - Binary constant
    An error will occur.
    If this option is not specified, a string that consists of "0b" or "0B" followed by one or more "0" or "1" is treated as a binary constant.
- This option is equivalent to the following property in CubeSuite.
  - From the [Individual Compile Options] tab, [Compile strictly according to ANSI standards] in the [C Language] category
  - From the [Individual Compile Options] tab, [Compile strictly according to ANSI standards] in the [C Language] category

Note This is a standard specified by ISO/IE C9899:1990 (C90).
Although the cx also accepts some of the specifications added by ISO/IE C9899:1999 (C99), if this option is specified, code in violation of the standard will cause an error.

[Example of use]

- To process as making C source program comply strictly with the ANSI standard and output an error or warning for a specification that violates the standard, describe as:

```
>cx -CF3746 -Xansi main.c
```

-Xchar

This option specifies signed for char type.

[Specification format]

```
-Xchar=string
```

- Interpretation when omitted
  To handle char type that does not have signed as signed, describe as:

[Detailed description]

- This option specifies whether char type that does not have signed is handled as signed or unsigned.
- The items that can be specified as *string* are shown below.
  An abort error will occur if any other item is specified.

signed	Handled as signed
unsigned	Handled as unsigned

- An error will occur if *string* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Sign of char] in the [C Language] category

[Example of use]

- To handle char type that does not have signed as signed, describe as:

```
>cx -CF3746 -Xchar=signed main.c
```

-Xenum_type

This option specifies which integer type the enumeration type handles.

[Specification format]

```
-Xenum_type=string
```

- Interpretation when omitted
  The enumeration type is handled as signed int.

[Detailed description]

- This option specifies which integer type the enumeration type handles.
- The items that can be specified as *string* are shown below.
  An abort error will occur if any other item is specified.

auto	Each enumerated type is treated as the smallest integer type capable of expressing all the enumerators in that type.

- An error will occur if *string* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Enumeration type] in the [C Language] category

[Example of use]

- To treat each enumerated type as the smallest integer type capable of expressing all the enumerators in that type, describe as:

```
>cx -CF3746 -Xenum_type=schar main.c
```

-Xdef_var

This option handles tentative definition of variables as definition.

[Specification format]

```
-Xdef_var
```

- Interpretation when omitted
  Tentative definition of variables is not handled as definition.

[Detailed description]

- This option handles tentative definition of variables as definition.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Treat tentative definition as definition] in the [C Language] category

[Example of use]

- To handle tentative definition of variables as definition, describe as:

```
>cx -CF3746 -Xdef_var main.c
```

Variable information specification

The variable information specification option is as follows.

- -Xsymbol_file

-Xsymbol_file

This option uses a symbol information file.

[Specification format]

```
-Xsymbol_file=file
```

- Interpretation when omitted
  A symbol information file is not read.

[Detailed description]

- This option reads symbol information file *file* describing the variable allocation section and uses the information while compiling.
- See "B.1.4 Symbol information file" for details.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Using symbol information file] in the [Symbol Information] category

[Example of use]

- To read symbol information file symbol and use the information while compiling, describe as:

```
>cx -CF3746 -Xsymbol_file=symbol.sfg main.c
```

Japanese/Chinese character control

The Japanese/Chinese character control option is as follows.

- -Xcharacter_set

-Xcharacter_set

This option specifies the Japanese/Chinese character code.

[Specification format]

```
-Xcharacter_set=code
```

- Interpretation when omitted
  The Japanese character code is handled as SJIS.

[Detailed description]

- This option specifies the character code to be used for Japanese/Chinese comments and character strings in the source file.
- The items that can be specified as *code* are shown below.
  An abort error will occur if any other item is specified.
  Operation is not guaranteed if the specified character code differs from the character code of the source file.

none	Does not process the Japanese and Chinese character code
euc_jp	EUC (Japanese)
sjis	SJIS
utf8	UTF-8
big5	Traditional Chinese
gb2312	Simplified Chinese

- An error will occur if *code* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Character encoding] in the [Character Encoding] category
  - From the [Individual Compile Options] tab, [Character encoding] in the [Character Encoding] category

[Example of use]

- To specify EUC as the character code to be used for Japanese comments and character strings in the input file, describe as:

```
>cx -CF3746 -Xcharacter_set=euc_jp main.c
```

Optimization specification

The optimization specification options are as follows.

- -O
- -Xsort_var
- -Xinline_strcpy
- -Xpro_epi_runtime
- -Xcall_lib

-O

This option specifies the optimization level or the details of each optimization items.

[Specification format]

```
-O[level]
-O[item[=value][,item[=value]]...
```

- Interpretation when omitted
  Only optimization that debugging is not affected is performed (It is the same result as when -Odefault option is specified).

[Detailed description]

- This option specifies the optimization level or the details of each optimization items.
  See "B.1.5 Optimization function" for details.
- The items that can be specified as *level* are shown below.
  An abort error will occur if any other item is specified.

nothing	Optimization with debugging precedence Regards debugging as important and suppresses all optimization including default optimization.
default	Default Performs optimization that debugging is not affected (optimization of expressions and register allocation, and the like)
size	Optimization with the object size precedence Regards reducing the ROM/RAM capacity as important and performs the maximum optimization that is effective for general programs.
speed	Optimization with the execution speed precedence Regards shortening the execution speed as important and performs the maximum optimization that is effective for general programs.

- If *level* is omitted, it is assumed that "size" has been specified.
- The items that can be specified as *item* and *value* are shown below.
  An abort error will occur if any other item is specified.

Optimization Item (*item*)	Parameter (*value*)	Description
unroll	0 to 4294967295 (Integer value)	Loop expansion When 0 or 1 is specified, the expansion is suppressed. If 2 or more is specified, the number of executions is converted into *N* loops (*N* is a constant), and a loop including code expanded *value* times. If the code size after expansion is too great or if the number of times of execution of the loop is too few, the number of times of expansion may decrease, or the loop may not be expanded at all. In addition, a loop having a complicated structure, such as having inner loops, may not be expanded. If *value* is omitted, it is assumed that 4 has been specified. If the -Ospeed option is specified, this item is assumed that the -Ounroll=4 option is specified.
inline	0 to 2 (Integer value)	Inline expansion for functions *value* signifies the level of the expansion. 0: Suppresses all inline expansion including the function for which "#pragma inline" is specified. 1: Performs inline expansion for only a function for which "#pragma inline" is specified. 2: Distinguishes a function that is the target of expansion automatically and expands it. However, if 1 or 2 is specified, the function that is specified by "#pragma inline" may not be expanded according to the content of the function and the status of compilation. If *value* is omitted, it is assumed that 2 has been specified. If the -Ospeed option is specified, this item is assumed that the -Oinline=2 option is specified. If the -Ospeed or -Oinline option is not specified, this item is assumed that the -Oinline=1 option is specified. If the -Onothing option is specified, this item is assumed that the -Oinline=0 option is specified.
delete_static_func	on or off	Deleting unused static functions If *value* is omitted, it is assumed that "on" has been specified. This item is valid when the -Onothing option is not specified.
pipeline [V850E2V3]	on or off	Pipeline optimization If *value* is omitted, it is assumed that "on" has been specified. This item is always valid when the -Ospeed option is specified.

- If *item* is omitted, it is assumed that this option has specified "-Osize".
- If this option is specified more than once for the same *item*, the option specified last will be valid.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Level of optimization] in the [Frequently Used Options(for Compile)] category
  - From the [Compile Options] tab, [Level of optimization] in the [Optimization] category
  - From the [Compile Options] tab, [Maximum number of loop expansions] in the [Optimization(Details)] category
  - From the [Compile Options] tab, [Remove unused static functions] in the [Optimization(Details)] category
  - From the [Compile Options] tab, [Perform inline expansion] in the [Optimization(Details)] category
  - From the [Compile Options] tab, [Perform pipeline optimization] in the [Optimization(Details)] category

- From the [Individual Compile Options] tab, [Level of optimization] in the [Optimization] category
- From the [Individual Compile Options] tab, [Maximum number of loop expansions] in the [Optimization(Details)] category
- From the [Individual Compile Options] tab, [Remove unused static functions] in the [Optimization(Details)] category
- From the [Individual Compile Options] tab, [Perform inline expansion] in the [Optimization(Details)] category
- From the [Individual Compile Options] tab, [Perform pipeline optimization] in the [Optimization(Details)] category

[Example of use]

- To perform optimization with the object size precedence, describe as:

```
>cx -CF3746 -Osize main.c
```

-Xsort_var

This option sorts external variables.

[Specification format]

```
-Xsort_var
```

- Interpretation when omitted
  External variables are not sorted.

[Detailed description]

- This option rearranges external variables allocated to a section other than const or sconst sequentially, starting from the largest alignment size.
- This option is for reducing the RAM capacity.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Sort external variables] in the [Optimization(Details)] category
  - From the [Individual Compile Options] tab, [Sort external variables] in the [Optimization(Details)] category

[Example of use]

- To rearrange external variables allocated to a section other than const or sconst sequentially, starting from the largest alignment size, describe as:

```
>cx -CF3746 -Xsort_var main.c
```

-Xinline_strcpy

This option performs inline expansion of standard library functions "strcpy", "strcmp", "memcpy", and "memset" calls.

[Specification format]

```
-Xinline_strcpy
```

- Interpretation when omitted
  Inline expansion of standard library functions "strcpy", "strcmp", "memcpy", and "memset" calls is not performed.

[Detailed description]

- This option performs inline expansion of standard library functions "strcpy", "strcmp", "memcpy", and "memset" calls.
- This option can not be specified together with the -Xpack option.
- Inline expansion of strcpy is performed only when the second argument is a character string.
- If this option is specified, arrays and character strings are allocated automatically to 4-byte boundary area.
- This improves the execution speed of the program to be generated, but it increases the code size.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Perform inline expansion of strcpy/strcmp/memcpy/memset] in the [Optimization(Details)] category
  - From the [Individual Compile Options] tab, [Perform inline expansion of strcpy/strcmp/memcpy/memset] in the [Optimization(Details)] category

[Example of use]

- To perform inline expansion of standard library functions "strcpy", "strcmp", "memcpy", and "memset" calls, describe as:

```
>cx -CF3746 -Xinline_strcpy main.c
```

-Xpro_epi_runtime

This option specifies whether or not to perform prologue/epilogue processing of the function through runtime library calls.

[Specification format]

```
-Xpro_epi_runtime[=on|=off]
```

- Interpretation when omitted
  Prologue/epilogue processing of the function through runtime library calls is performed.
  However, processing through runtime library calls is not performed when the -Ospeed option is specified.

[Detailed description]

- This option specifies whether or not to perform prologue/epilogue processing of the function through runtime library calls.
- If "on" is specified, prologue/epilogue processing of the function is performed through runtime library calls.
- If "=on" or "=off" is omitted, it is assumed that "=on" has been specified.
- An abort error will occur if other than "=on" or "=off" is specified.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Use prologue/epilogue library] in the [Optimization(Details)] category
  - From the [Individual Compile Options] tab, [Use prologue/epilogue library] in the [Optimization(Details)] category

[Example of use]

- To perform prologue/epilogue processing of the function through runtime library calls, describe as:

```
>cx -CF3746 -Xpro_epi_runtime=on main.c
```

-Xcall_lib

This option suppresses inline expansion of the library function.

[Specification format]

```
-Xcall_lib
```

- Interpretation when omitted
  If the conditions below are met at the same time, then at locations where there is a call to the library function sqrtf, a sqrtf.s instruction will be generated in the place of the function call instruction.
  In other cases, a function call instruction is generated.
  - One of the optimization options -O, -Osize, or -Ospeed is specified.
  - The source file containing the call to sqrtf includes standard header file "math.h", or contains a declaration of the sqrtf prototype.
  - Any one of the conditions below is met.
    - The -C option specifies a device with a V850E2V3 FPU, and the -Xfloat=soft option is not specified.
    - The -Xcommon=v850e2v3 and -Xfloat=fpu options are specified at the same time.

Caution If the sqrtf.s instruction is generated directly, then unlike the case of a library function call, no exception processing will be performed.
It is therefore necessary to specify this option if you wish to perform exception processing in the same manner as the library function.
Specifically, the following points differ.
- Even if the argument is a negative real number, macro EDOM will not be set in global variable errno.
- Error processing function matherrf (matherr) will not be enabled.

[Detailed description]

- At the location of the call to library function sqrtf, a call to each library function is always generated.
- Use this option to generate a library function call instruction without expanding to the instruction directly, regardless of whether the conditions under [Interpretation when omitted] are met.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Perform inline expansion of library] in the [Optimization(Details)] category
  - From the [Individual Compile Options] tab, [Perform inline expansion of library] in the [Optimization(Details)] category

[Example of use]

- At the location of the call to library function sqrtf, a call to each library function is always generated.

```
>cx -CF3746 -Xcall_lib main.c
```

Generated code control

The generated code control options are as follows.

- -Xpack
- -Xpass_source
- -Xswitch
- -Xword_case
- -Xr
- -Xreg_mode
- -Xsdata
- -Xsconst
- -Xfloat [V850E2V3]
- -Xfar_jump
- -Xdiv [V850E2V3]

-Xpack

This option performs the structure packing.

[Specification format]

```
-Xpack=num
```

- Interpretation when omitted
  The structure packing is not performed.

[Detailed description]

- This option performs the structure packing.
- If this option is specified, struct members will not be aligned by their member types, but rather code will be generated with alignment packed to the specified *num* bytes.
- 1, 2, 4, or 8 can be specified as *num*.
  An abort error will occur if any other item is specified.
- An error will occur if *num* is omitted.
- This option can not be specified together with the -Xinline_strcpy option.
- If this option is specified when the structure packing is specified by the #pragma directive in the C source, the value specified by this option is applied to all structures until the first #pragma directive appears.
  After that, the value of the #pragma directive is applied.
  Even after the #pragma directive has appeared, however, the value specified by the option is applied if the default value is specified (if the value of the packing by the #pragma directive).
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Structure packing] in the [Output Code] category

[Example of use]

- To generate code with struct member alignment packed to 1 byte, describe as:

```
>cx -CF3746 -Xpack=1 main.c
```

-Xpass_source

This option outputs a C source program as a comment to the assembler source file.

[Specification format]

```
-Xpass_source
```

- Interpretation when omitted
  The C source program is not output as a comment to the assembler source file.

[Detailed description]

- This option outputs a C source program as a comment to the assembler source file.
- The output comments are for reference only and may not correspond exactly to the code.
  Additionally, non-executed lines may not be output as comments (e.g. type declarations and labels).
  For example, comments concerning global variables, local variables, function declarations, etc., may be output to incorrect positions.
  By specifying the optimization options, the code may be deleted and only the comment may remain.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Output comment to assembler source file] in the [Output Code] category
  - From the [Individual Compile Options] tab, [Output comment to assembler source file] in the [Output Code] category

[Example of use]

- To output a C source program as a comment to the assembler source file, describe as:

```
>cx -CF3746 -Xpass_source main.c
```

-Xswitch

This option specifies the format in which the code of switch statements is to be output.

[Specification format]

```
-Xswitch=type
```

- Interpretation when omitted
  The cx selects the optimum output format for each switch statement.

[Detailed description]

- This option specifies the format in which the code of switch statements is to be output.
- The items that can be specified as *type* are shown below.
  An abort error will occur if any other item is specified.

ifelse	Outputs the code in the same format as the if-else statement along a string of case statements. Select this item if the case statements are written in the order of frequency or if only a few labels are used. Because the case statements are compared starting from the top, unnecessary comparison can be reduced and the execution speed can be increased if the case statement that most often matches is written first.
binary	Outputs the code in the binary search format. Searches for a matching case statement by using a binary search algorithm. If this item is selected when many labels are used, any case statement can be found at almost the same speed.
table	Outputs the code in a table jump format. References a table indexed on the values in the case statements, and selects and processes case labels from the switch statement values. The code will branch to all the case statements with about the same speed. However, if case values are not used in succession, an unnecessary area will be created. If the difference between the maximum and minimum values of the case labels exceeds 8192, then this option is ignored, and the optimum output format for each switch statement is selected automatically.

- An error will occur if *type* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Output code of switch statement] in the [Output Code] category
  - From the [Individual Compile Options] tab, [Output code of switch statement] in the [Output Code] category

[Example of use]

- To output a code for the switch statement in the binary search format, describe as:

```
>cx -CF3746 -Xswitch=binary main.c
```

-Xword_case

This option generates a 4-byte branch table.

[Specification format]

```
-Xword_case
```

- Interpretation when omitted
  One 2-byte branch table per case label in a switch statement is generated.

[Detailed description]

- This option generates one 4-byte branch table per case label in a switch statement.
- Specify this option when a compile error occurs because the switch statement is long.
- If this option is specified together with the -Xswitch=ifelse or -Xswitch=binary option, this option will be invalid.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Label size of switch table] in the [Output Code] category
  - From the [Individual Compile Options] tab, [Label size of switch table] in the [Output Code] category

[Example of use]

- To generate one 4-byte branch table per case label in a switch statement, describe as:

```
>cx -CF3746 -Xword_switch main.c
```

-Xr

This option specifies the register that external variables are allocated.

[Specification format]

```
-Xrnum=sym
```

- Interpretation when omitted
  Variables are allocated to registers optimally in the whole program.

[Detailed description]

- This option allocates external variable *sym* to register r*num*.
- Specify the register that is vacated by specifying the -Xreg_mode option as *num*.
  The items that can be specified as *num* are shown below.
  An abort error will occur if any other item is specified.

Register Mode	Register That Can Be Specified (*num*)	
	Working Registers	Registers for Register Variables
22	15 to 19	20 to 24
26	17 to 19	20 to 22
32	None	None

- Specify the external variable name (variable name is described in C language) as *sym*.
  A volatile variable, variable using address operator, aggregate, array, variable having internal linkage, and peripheral I/O register cannot be specified.
  An abort error will occur if these variables are specified.
  If *sym* does not exist, it will be ignored.
- An error will occur if *num* or *sym* is omitted.
- This option can be specified for the external variable with the size of 8 bytes. In this case, only an even number can be specified as *num*.
  An error will occur if an odd number is specified.
- If the default value is set when declaring *sym*, r*num* is not initialized automatically.
  Therefore, you need to assign the value as *sym* explicitly by using an assignment statement or set the value to r*sym* in the startup routine.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [External variable assigned to the r15 register] in the [External Variable Register] category
  - From the [Compile Options] tab, [External variable assigned to the r16 register] in the [External Variable Register] category
  - From the [Compile Options] tab, [External variable assigned to the r17 register] in the [External Variable Register] category
  - From the [Compile Options] tab, [External variable assigned to the r18 register] in the [External Variable Register] category
  - From the [Compile Options] tab, [External variable assigned to the r19 register] in the [External Variable Register] category

- From the [Compile Options] tab, [External variable assigned to the r20 register] in the [External Variable Register] category
- From the [Compile Options] tab, [External variable assigned to the r21 register] in the [External Variable Register] category
- From the [Compile Options] tab, [External variable assigned to the r22 register] in the [External Variable Register] category
- From the [Compile Options] tab, [External variable assigned to the r23 register] in the [External Variable Register] category
- From the [Compile Options] tab, [External variable assigned to the r24 register] in the [External Variable Register] category

[Example of use]

- To allocate external variable "arg" to register "r18" (when using the 22-register mode), describe as:

```
>cx -CF3746 -Xreg_mode=22 -Xr18=arg main.c
```

-Xreg_mode

This option specifies the register mode.

[Specification format]

```
-Xreg_mode=mode
```

- Interpretation when omitted
  The 32-register mode object module file is generated.

[Detailed description]

- This option generates the object module file for the specified register mode.
- This option limits the number of registers used by the cx to 32 (the 32-register mode), 26 (the 26-register mode), or 22 (the 22-register mode or register mode "common") and embeds the magic number into the object module file.
- Use register mode "common" to generate the object module file that does not depend on register modes.
- The items that can be specified as *mode* are shown below.
  An abort error will occur if any other item is specified.

Register Mode (*mode*)	Working Registers	Registers for Register Variables
common	r10 to r14	r25 to r29
22	r10 to r14	r25 to r29
26	r10 to r16	r23 to r29
32	r10 to r19	r20 to r29

- An error will occur if *mode* is omitted.
- When "-Xreg_mode=common" is specified, the behavior of "setjmp" and "longjmp" is the same as when "-Xreg_mode=32" is specified.
  For this reason, even if the values of r20 to r24 are changed after calling "setjmp", after "longjmp" is called, they will be returned to their values before the setjmp call.
- When "-Xreg_mode=common" is specified, the behavior of runtime functions is the same as when "-Xreg_mode=32" is specified.
  For this reason, when an exception is raised, even if the values of r15 to r19 are changed inside "matherrf" or "matherrd", the runtime functions will not be reflected in the calling program.
- This option generates the code using the register that can be used for a C source file.
- The same register mode must be specified for all source files.
  The different register mode cannot be specified for each source file.
  If there are object module files with different register modes, a warning will be output during linking and link processing will be continued.
- By specifying this option, the register mode of the software register bank function can be changed.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Register mode] in the [Register Mode] category

[Example of use]

- To generate the 22-register mode object module file, describe as:

```
>cx -CF3746 -Xreg_mode=22 main.c
```

-Xsdata

This option specifies the maximum size of data allocated to the .sdata or .sbss section.

[Specification format]

```
-Xsdata=num
```

- Interpretation when omitted
  All data is allocated to the .sdata section or .sbss section.
  However, static variables with a const modifier are allocated to the .const section.

[Detailed description]

- This option allocates data of less than *num* bytes to the .sdata or .sbss section.
- Data specified the .sdata or .sbss section by the "#pragma section" directive is allocated to that section regardless of the size.
- This option is not applicable for arrays of incomplete types (arrays whose size is indeterminate within the file).
- Specify 0 to 65535 as *num*.
  An abort error will occur if a value outside the range is specified.
- An error will occur if *num* is omitted.
- The yardstick for the value to be specified as *num* can be output by the -Xsdata_info option.
- If a different option is specified for each file, a code of a different method of placing and referencing variables may be generated and an error or warning may be output during linking.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Size threshold of sdata/sbss section allocation(Bytes)] in the [Output Code] category

[Example of use]

- To allocate data of less than 16 bytes to the .sdata or .sbss section, describe as:

```
>cx -CF3746 -Xsdata=16 main.c
```

-Xsconst

This option specifies that constant data is allocated to the .sconst section.

[Specification format]

```
-Xsconst[=num]
```

- Interpretation when omitted
  All const data is allocated to the .const section.

[Detailed description]

- This option allocates const data of less than *num* bytes (const attribute data and character string literals) to the .sconst section.
- Only data that a section is not specified is valid.
- This option is not applicable for arrays of incomplete types (arrays whose size is indeterminate within the file).
- Specify 0 to 32767 as *num*.
  An abort error will occur if a value outside the range is specified.
- If *num* is omitted, all const data is allocated to the .sconst section.
- If a different option is specified for each file, a code of a different method of placing and referencing variables may be generated and an error or warning may be output during linking.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Size threshold of sconst section allocation(Bytes)] in the [Output Code] category

[Example of use]

- To allocate const data of less than 16 bytes to the .sconst section, describe as:

```
>cx -CF3746 -Xsconst=16 main.c
```

-Xfloat [V850E2V3]

This option controls generating floating-point calculation instructions.

[Specification format]

```
-Xfloat=type
```

- Interpretation when omitted
  If the target device specified by the -C option has FPU, floating-point calculation instructions are generated. Additionally, the FPU math support library will be linked.
  In other cases (the target device specified by the -C option does not have an FPU, or the -C option is not specified and the -Xcommon option is specified), floating-point calculation instructions are not generated and runtime library call instructions are generated. Additionally, the FPU math support library will not be linked.

[Detailed description]

- This option controls generating floating-point calculation instructions.
- Specify this option if the target device has an FPU.
- The items that can be specified as *type* are shown below.
  An abort error will occur if any other item is specified.

soft	Generates runtime library call instructions for floating-point calculations. Additionally, the FPU math support library will not be linked.
fpu	Generates floating-point calculation instructions of FPU (floating-point unit) for floating-point calculations. Additionally, the FPU math support library will be linked. If this item is specified for the target device (including devices other than V850E2V3) that does not have the FPU, a warning will be output and this item will be ignored.

- An error will occur if *type* is omitted.
- If the target device does not have an FPU, then linking an object module file containing FPU instructions in CX version 1.10 or later will cause an error. Linking an object module file generated by CX version 1.01 or earlier in this case will cause a warning to be displayed regardless of whether there are FPU instructions (object module files generated with CA850 will be linked as is).
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Floating-point calculating type] in the [Output Code] category
  - From the [Individual Compile Options] tab, [Floating-point calculating type] in the [Output Code] category

[Example of use]

- To generate runtime library call instructions for floating-point calculations, describe as:

```
>cx -CF3746 -Xfloat=soft main.c
```

-Xfar_jump

This option controls outputting far jump.

[Specification format]

```
-Xfar_jump=file
```

- Interpretation when omitted
  The jarl or jr directive is generated for the branch to the function.

[Detailed description]

- This option generates the code that uses the jump instruction (V850E) or the jarl32 and jr32 instruction (V850E2 and V850E2V3) for the branch to functions (including interrupt functions described by C language) specified in far jump calling function list file *file*.
- ".fjp" is recommended as the extension of *file*.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.
- An error will occur during linking if the function is in the range that cannot be branched to by the jarl or jr directive (±2MB or more). In this case, recompile by using this option.
- If this option is specified more than once, the option specified last will be valid.
- The example of the output code is shown below.
  - C source

```
         far func();     /* "jarl  far func, lp" is output by default. */
```

  - Output assembler source (V850E)

```
         movea    far func, tp, r10
         movea    .BB.LABEL.15, tp, lp
         jmp      [r10]
.BB.LABEL.15:
```

  - Output assembler source (V850E2 and V850E2V3)

```
         jarl32   far func, lp
```

- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Far Jump file names] in the [Output Code] category
  - From the [Individual Compile Options] tab, [Far Jump file names] in the [Output Code] category

Remark Cautions about are the format of the far jump calling function list file as follows.
- Describe with one function name per line.
  If two or more function name is described, the first name will be valid.
- Describe the function name (label name in an assembler source) by prefixing "_" to that in C language.
  However, the following formats can be specified instead of function names.

Format	Meaning
{all function}	All functions are called.
{all interrupt}	All interrupt functions are called.

- Rutime routines can be specified instead of functions.
  In this case, describe the runtime routine name as-is as the function name.
- A space and tab can be inserted before and after function names.
- Only ASCII characters can be used.
  Therefore, characters that cannot be used as a function name in C language are ignored.
- Comments cannot be inserted.
- Up to 1023 characters can be specified per line (including a space and tab).

The example when specifying functions is shown below.

```
_func_led
_func_beep
_func_motor
    :
_func_switch
```

[Example of use]

- To generate the code that uses jmp directive to branch to the function specified in func.fjp, describe as:

```
>cx -CF3746 -Xfar_jump=func.fjp main.c
```

-Xdiv [V850E2V3]

This option generates the div and divu instructions for division.

[Specification format]

```
-Xdiv
```

- Interpretation when omitted
  The divq and divqu instructions are generated for division.

[Detailed description]

- This option generates the div and divu instructions instead of the divq and divqu instructions for division.
- Although the divq and divqu instructions are fast, the number of execution cycles will differ depending on the values of the operands.
  For this reason, specify this option if it is necessary to maintain a constant number of execution cycles at all times (e.g. in order to guarantee real-time performance).
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Generate div/divu instructions] in the [Output Code] category
  - From the [Individual Compile Options] tab, [Generate div/divu instructions] in the [Output Code] category

[Example of use]

- To generate the div and divu instructions for division, describe as:

```
>cx -CF3746 -Xdiv main.c
```

Assembler control specification

The assembler control specification option is as follows.

- -Xasm_far_jump [V850E2][V850E2V3]

-Xasm_far_jump [V850E2][V850E2V3]

This option controls outputting far jump for an assembler source file.

[Specification format]

```
-Xasm_far_jump
```

- Interpretation when omitted
  Assembly is performed as a jarl or jr instruction.

[Detailed description]

- For an assembler source file, this option assumes that all jarl and jr instructions described in the source are jarl32 and jr32 instructions, and assembling is performed.
- If you wish to control individual instructions, add jarl22/jarl32 or jr22/jarl22 to the source.
- This option does not affect the jump instruction.
- If this option is specified for a C source file, that will be ignored without displaying a warning.
- This option is equivalent to the following property in CubeSuite.
  - From the [Compile Options] tab, [Use 32-bit branch instruction] in the [Output Code] category

[Example of use]

- To assume that all jarl and jr instructions described in the source are jarl32 and jr32 instructions, and perform assembling, describe as:

```
>cx -CF3746 -Xasm_far_jump main.c
```

Library link control

The library link control options are as follows.

- -l
- -L
- -Xno_stdlib
- -Xno_startup
- -Xstartup

-l

This option specifies the library file to be used during linking.

[Specification format]

```
-lstring[,string]...
```

- Interpretation when omitted
  Only the standard library, mathematical library, and standard startup routine are linked.
  When the -Xno_romize option is not specified, the ROMization area reservation code file is linked.

[Detailed description]

- This option specifies library file lib*string*.lib to be used during linking.
- When the cx resolves an unresolved external symbol reference after linking all object module files, the library file (lib*string*.lib) is referenced.
- An error will occur if *string* is omitted.
- If this option and the -L option are specified at the same time, the library file is searched from the folder specified by the -L option.
  If the -L option is not specified, the library file is searched from the standard folder[Note].
- If the library file specified by this option cannot be found, a message will not output and link processing will be continued.
- If two or more library files are specified, they are searched in the order which they are specified.
- The cx links the standard library (libc.lib) and standard startup routine (cstart.obj) as well as the library specified by this option automatically.
  When the -Xno_romize option is not specified, the ROMization area reservation code file (rompcrt.obj) is also linked automatically.
  Use the -Xno_stdlib and -Xno_startup option to suppress this.

Note *CubeSuite install folder*\Cubesuite\CX\V*x.xx*\lib\850e

[Example of use]

- To specify library file "libsmp.lib" to be used during linking, describe as:

```
>cx -CF3746 -lsmp main.c
```

-L

This option specifies the folder to search library files.

[Specification format]

```
-Lpath[,path]...
```

- Interpretation when omitted
  The library file is searched from the standard library file folder.

[Detailed description]

- This option searches libraries from folder *path*, the standard folder[Note] in that order.
- If *path* does not exist, a warning will be output.
- An error will occur if *path* is omitted.

Note *CubeSuite install folder*\Cubesuite\CX\V*x.xx*\lib\850e

[Example of use]

- To search libraries from folder "lib", the standard folder in that order, describe as:

```
>cx -CF3746 -Llib main.c
```

-Xno_stdlib

This option suppresses linking the standard library.

[Specification format]

```
-Xno_stdlib
```

- Interpretation when omitted
  The standard library is linked.

[Detailed description]

- This option does not link the standard library.

[Example of use]

- Not to link the standard library, describe as:

```
>cx -CF3746 -Xno_stdlib main.c
```

-Xno_startup

This option suppresses linking the startup routine.

[Specification format]

```
-Xno_startup
```

- Interpretation when omitted
  The startup routine is linked.

[Detailed description]

- This option does not link the startup routine (default: cstart.obj, when the -Xno_romize option is specified: cstartN.obj).
- Specify this option to handle the file including startup routine in the same way as other source files without preparing the special file as the startup routine.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Use standard startup routine] in the [Input File] category

[Example of use]

- Not to link the startup routine, describe as:

```
>cx -CF3746 -Xno_startup main.c
```

-Xstartup

This option specifies the startup routine.

[Specification format]

```
-Xstartup=file3
```

- Interpretation when omitted
  The startup routine is linked.

[Detailed description]

- This option links *file* as the startup routine instead of the standard startup routine (default: cstart.obj, when the -Xno_romize option is specified: cstartN.obj).
- Specify the object module file as *file*.
- An error will occur if *file* does not exist.
- An error will occur if *num* is omitted.

[Example of use]

- To link "start.obj" as the startup routine, describe as:

```
>cx -CF3746 -Xstartup=start.obj main.c
```

Link control

The link control options are as follows.

- -Xlink_directive
- -Xmap
- -Xentry_address
- -Xrelinkable_object
- -Xregmode_info
- -Xforce_link
- -Xsdata_info
- -Xtwo_pass_link
- -Xignore_address_error
- -Xmultiple_symbol
- -Xlink_check_off
- -Xalign_fill
- -Xrescan
- -Xstrip
- -Xflash
- -Xflash_ext_table

-Xlink_directive

This option specifies the link directive file.

[Specification format]

```
-Xlink_directive=file
```

- Interpretation when omitted
  The default link directive file is used.

[Detailed description]

- This option performs linking according to the link directive in link directive file *file*.
- ".dir" is recommended as the extension of *file*.
- An error will occur if *file* is omitted.
- If this option is specified more than once, the option specified last will be valid, and the option specified first will be ignored.
- See "CubeSuite Coding for CX Compiler" for details about the link directive file.

[Example of use]

- To perform linking according to the link directive in link directive file "link.dir", describe as:

```
>cx -CF3746 -Xlink_directive=link.dir main.c
```

-Xmap

This option outputs the link map file.

[Specification format]

```
-Xmap[=file]
```

- Interpretation when omitted
  The link map file is not output.

[Detailed description]

- This option outputs link map file *file*.
- The contents of the link map file are shown below.
  - Allocation of input sections which are included in the specified object module file to the memory space
  - Allocation of the output sections which composes the load module file to be generated by linking input sections to the memory space
  - Address information of symbols
- If *file* already exists, it will be overwritten.
- If only the file name is specified for *file*, the file is output under the specified file name to the same folder as the load module file.
- If *file* is omitted, the file is output to the same folder as the load module file under the load module file name with the extension replaced by ".map".
- The link map file to be output differs depending on whether the -Xno_romize option is specified as follows.
  - When the -Xno_romize option is specified
    The link map file is output after link processing.
  - When the -Xno_romize option is not specified
    The link map file is output after ROMization processing.

[Example of use]

- To output link map file "smp1.map" after link processing, describe as:

```
>cx -CF3746 -Xmap=smp.map main.c -Xno_romize
```

- To output link map file "smp2.map" after ROMization processing, describe as:

```
>cx -CF3746 -Xmap=smp.map main.c
```

-Xentry_address

This option specifies the entry point address.

[Specification format]

```
-Xentry_address=symbol
```

- Interpretation when omitted
  The entry point address value for the load module file to be generated is determined according to the following rules.
- If symbol "__start" exists, it is used.
- If "__start" does not exist, the start address of the text attribute section that is allocated to the lowest address area in the load module file to be generated is used.
- If the text attribute section does not exist, "0" is used.

[Detailed description]

- This option regards the value of symbol *symbol* as the entry point address value (it is used when hex conversion is performed) for the load module file to be generated.
- An error will occur if *symbol* does not exist.
- An error will occur if *symbol* is omitted.

[Example of use]

- To regard the value of symbol "__my_start" as the entry point address value for the load module file to be generated, describe as:

```
>cx -CF3746 -Xentry_address=__my_start main.c
```

-Xrelinkable_object

This option generates the relocatable object module file.

[Specification format]

```
-Xrelinkable_object
```

- Interpretation when omitted
  The cx tries to generate the executable object module file.
  If an unresolved external reference remains after linking, an error is output and linking is stopped.
  At this time, the load module file is not generated.

[Detailed description]

- This option generates the relocatable object module file.
- If this option is specified, a message is not output and linking is completed normally if an unresolved external reference remains after linking.
- If the object module file generated by the cx is specified as the target for relocation, use this option to generate the target object module file for relocation.
- If this option is specified, only the types and attributes in the segment directive section in the link directive will be valid and the others are ignored.
- If this option is specified, any reserved symbol is not created.
- This option must be specified together with the -Xno_startup and -Xno_romize options.

[Example of use]

- To generate the relocatable object module file, describe as:

```
>cx -CF3746 -Xrelinkable_object main.c -Xno_startup -Xno_romize
```

-Xregmode_info

This option outputs detailed information when different register modes are mixed.

[Specification format]

```
-Xregmode_info
```

- Interpretation when omitted
  Detailed information is not output when different register modes are mixed.

[Detailed description]

- This option outputs detailed information and specifies the input object module file that causes a warning when different register modes are mixed for all input object module files.
- No information will be output if the register modes match.

[Example of use]

- To output detailed information when different register modes are mixed for all input object module files, describe as:

```
>cx -CF3746 -Xregmode_info file1.c file2.c
```

-Xforce_link

This option continues link processing when the internal ROM/RAM overflows.

[Specification format]

```
-Xforce_link
```

- Interpretation when omitted
  An error is output and link processing is completed when the internal ROM/RAM overflows.

[Detailed description]

- This option outputs a warning and continues link processing when the internal ROM/RAM overflows.
- The output file is not generated.
- The excess of the size is output to the standard error output.

[Example of use]

- To output a warning and continue link processing when the internal ROM/RAM overflows, describe as:

```
>cx -CF3746 -Xforce_link main.c
```

-Xsdata_info

This option outputs information that can be used as a yardstick for the parameter of the -Xsdata option to the standard output.

[Specification format]

```
-Xsdata_info
```

- Interpretation when omitted
  Information that can be used as a yardstick for the parameter of the -Xsdata option is not output to the standard output.

[Detailed description]

- This option outputs information that can be used as a yardstick for the parameter of the -Xsdata option (it specifies the maximum size of data allocated to the sdata/sbss section) to the standard output.
- When using the numerical value indicated by *OK*, data with a size less than that value is allocated to the sdata/sbss area.

[Example of use]

- To output information that can be used as a yardstick for the parameter of the -Xsdata option to the standard output, describe as:

```
>cx -CF3746 -Xsdata_info main.c
```

-Xtwo_pass_link

This option performs linking in the 2-pass mode.

[Specification format]

```
-Xtwo_path_link
```

- Interpretation when omitted
  Linking is performed in the 1-pass mode.

[Detailed description]

- This option performs linking in the 2-pass mode.
- The 2-pass mode is slower than the 1-pass mode, but it is able to process larger sized files.

[Example of use]

- To perform linking in the 2-pass mode, describe as:

```
>cx -CF3746 -Xtwo_path_link main.c
```

-Xignore_address_error

This option continues link processing if an illegality is found during relocation processing when linking.

[Specification format]

```
-Xignore_address_error
```

- Interpretation when omitted
  An error occurs if an illegality is found during relocation processing when linking.

[Detailed description]

- If any of the following illegalities is found during relocation processing when linking, this option outputs a warning instead of an error and continues link processing.
  - The result of address calculation of an unresolved external reference is illegal
  - The relationship with the section to be allocated is illegal

  Specifically, it indicates one of the following:
  - There is no GP symbol for GP relative relocation (LOCAL / GLOBAL / EXTERN)
  - The result of PC22/26 relocation is a branch to an odd address
  - The result of PC relative relocation is outside the allowed bounds (symbol/no symbol)
  - The result of other than PC-relative relocation is outside the allowed bounds (symbol/no symbol)
- The value of address calculation judged as an illegality is not assigned to the unresolved external reference judged as an error and the original value remains.

[Example of use]

- To output a warning and continue link processing if the result of address calculation of an unresolved external reference is illegal during relocation processing when linking, describe as:

```
>cx -CF3746 -Xignore_address_error main.c
```

-Xmultiple_symbol

This option outputs an error message for all multi-defined external symbols.

[Specification format]

```
-Xmultiple_symbol
```

- Interpretation when omitted
  An error message is output for the first multi-defined external symbol and link processing is stopped.

[Detailed description]

- This option outputs an error message for all multi-defined external symbols and file names and stops link processing.

[Example of use]

- To output an error message for all multi-defined external symbols and stop link processing, describe as:

```
>cx -CF3746 -Xmultiple_symbol main.c
```

-Xlink_check_off

This option suppresses checking when linking.

[Specification format]

```
-Xlink_check_off=string[,string]
```

- Interpretation when omitted
  When the external symbol is linked, a warning will be output and link processing will be continued if the size of the symbol is checked and the difference is detected.
  At this time, the symbol size of the file in which the symbol is defined is valid.
  When the undefined symbol is linked, a warning will be output and link processing will be continued if the size and alignment conditions of the symbol is checked and the difference is detected.
  If the application allocation overlaps the addresses of the internal ROM area, a warning will be output.

[Detailed description]

- This option suppresses checking when linking.
- The items that can be specified as *string* are shown below.
  An error will occur if any other item is specified.

symbol	Does not check the size and alignment conditions of the external symbol when it is linked.
undefined	Does not check the size and alignment conditions of the undefined external symbol when it is linked.
irom	Does not check the allocation to the internal ROM area. Therefore, if the application allocation overlaps the addresses of the internal ROM area, a warning will not be output.

- An error will occur if *string* is omitted.
- When the application is created in the ROM-less mode, it is assumed that "irom" has been specified.

Caution Checking the overflow of the internal ROM is not supported when the single-chip mode is selected.
Invalidate checking the overflow of the internal ROM by specifying the -Xlink_check_off=irom option and check the overflow on the link map.

[Example of use]

- Not to check the size and alignment conditions of the external symbol when it is linked, describe as:

```
>cx -CF3746 -Xlink_check_off=symbol main.c
```

-Xalign_fill

This option specifies the filling value of align holes.

[Specification format]

```
-Xalign_fill=value
```

- Interpretation when omitted
  The filling value 0x0000 is used for align holes between sections of the load module file to be generated.

[Detailed description]

- This option specifies filling value *value* of align holes between sections of the load module file to be generated.
- The range that can be specified for *value* is 0x0000 to 0xFFFF.
  An abort error will occur if a value outside the range is specified.
- If the specified value is less than 4 digits, the higher digits will be filled with 0.
- An error will occur if *value* is omitted.
- When this option is specified, linking must be performed in the 2-pass mode by specifying the -Xtwo_path_link option.
  An error will occur if the -Xtwo_path_link option is not specified.

[Example of use]

- To specify 0xFFFF as the filling value of align holes between sections of the load module file to be generated, describe as:

```
>cx -CF3746 -Xalign_fill=0xFFFF -Xtwo_path_link main.c
```

-Xrescan

This option rescans the library file specified by the -l option.

[Specification format]

```
-Xrescan
```

- Interpretation when omitted
  The library file specified by the -l option is not rescanned.

[Detailed description]

- This option rescans the library file specified by the -l option.
- When this option is specified, symbols that are unresolved through the link sequence of the library can be prevented.

[Example of use]

- To rescan library file "libtest1.lib" and "libtest2.lib", describe as:

```
>cx -CF3746 -Xrescan main.c -ltest1 -ltest2
```

-Xstrip

This option generates the load module file from which the debug information, line number information, and global pointer table have been removed.

[Specification format]

```
-Xstrip
```

- Interpretation when omitted
  If the debug information, line number information and global pointer table exist in the load module file to be generated, they are not removed.

[Detailed description]

- This option generates the load module file from which the debug information, line number information and global pointer table have been removed.

[Example of use]

- To generate the load module file from which the debug information, line number information and global pointer table have been removed, describe as:

```
>cx -CF3746 -Xstrip main.c
```

-Xflash

This option generates the load module file for the flash area.

[Specification format]

```
-Xflash=file
```

- Interpretation when omitted
  When the boot-flash re-link function is used, the load module file for the boot area is generated.
  When the boot-flash re-link function is not used, normal link processing is performed.

[Detailed description]

- This option generates the load module file for the flash area when the boot-flash re-link function is used.
  At this time, symbol information of load module file for the boot area *file* is referred and link processing is performed.
- Specify the load module file for the boot area that is generated using the boot-flash re-link function as *file*.
  The load module file specified here must be the file before ROMization processing (generated with the -Xno_romize or -Xlink_output option specified).
- An error will occur if a nonexistent file is specified as *file*.
- An error will occur if *file* is omitted.
- This option must be specified together with the -Xflash_ext_table option.

[Example of use]

- To generate load module file for the boot area "boot.lmf" with 0x200 as the start address of the branch table for the flash area, describe as:

```
>cx -CF3746 -Xflash_ext_table=0x200 -Xlink_output=boot.lmf boot1.c boot2.c -
Xlink_directive=boot.dir
```

To create the branch table at address 0x200 and generate load module file for the flash area "flash.lmf", describe as:
At this time, symbol information of load module file for the boot area "boot.lmf" is referred and link processing is performed.

```
>cx -CF3746 -Xflash_ext_table=0x200 -Xflash=boot.lmf -oflash.lmf flash1.c flash2.c  -
Xlink_directive=flash.dir
```

-Xflash_ext_table

This option specifies the start address value of the branch table for the boot-flash re-link function.

[Specification format]

```
-Xflash_ext_table=address
```

- Interpretation when omitted
  The load module file for the boot-flash re-link function is not generated and normal link processing is performed.

[Detailed description]

- This option generates the load module file for the boot-flash re-link function using address *address* as the start address value of the branch table.
  See "B.1.6 Boot-flash re-link function" for details about the boot-flash re-link function.
- Specify generating the load module file for the boot area or flash area depending on whether the -Xflash option is specified.
  - When the load module file for the boot area is generated (when the -Xflash option is not specified)
    The branch to the flash area is processed.
    At this time, the process is the branch to the branch table that is created at the address specified by this option.
  - When the load module file for the flash area is generated (when the -Xflash option is specified)
    The branch table that has the branch instruction to the original branch destination is created at the address specified by this option.
- The range that can be specified for *address* is 0x00000000 to 0xFFFFFFFF.
  An abort error will occur if a value outside the range is specified.
- If the specified value is less than 8 digits, the higher digits will be filled with 0.
- If an odd value is specified as *address*, it is corrected to an even number, and then a warning will be output and processing will continued.
- An error will occur if *address* is omitted.
- *address* must be the same as the value that is used when the load module file for the boot area and flash area is generated.
  An operation fault will occur if a different value is specified.
  No error checking is done.
- *address* must be within the ROM for the flash area.
  No error checking is done because it is not possible to determine which area contains the specified address.
- By specifying this option, section ".ext_table" that has a size of "(maximum ID value Note + 1) * (entry size of branch table)" bytes and starts with *address* is automatically created when the load module file for the flash area is generated.
  Although this section does not require an alignment specification in the link directive file, you must leave enough space for alignment.

Note This is the value specified by the .ext_func quasi directive in the assembler source file.

- When the load module file for the boot area is generated, the load module file before ROMization processing must be saved.
  This is because this file is specified by the -Xflash option when the load module file for the flash area is generated.
  Consequently, if this option is not specified with the -Xflash option at the same time, it must be specified with the -

Xlink_output option (if ROMization is performed) or the -Xno_romize option (if ROMization is not performed) at the same time.

- This option can not be specified together with the -Xrelinkable_object option.
  An operation fault will occur if the load module file that has been generated using the -Xrelinkable_object option is input.

[Example of use]

- To generate load module file for the boot area "boot.lmf" with 0x200 as the start address of the branch table, describe as:

```
>cx -CF3746 -Xflash_ext_table=0x200 -Xlink_output=boot.lmf boot.c
```

ROMization control

The ROMization control options are as follows.

- -Xno_romize
- -Xrompcrt
- -Xrompsec_start
- -Xrompsec_data
- -Xrompsec_text
- -Xrompsec_only
- -Xromize_check_off

-Xno_romize

This option suppresses ROMization processing.

[Specification format]

```
-Xno_romize
```

- Interpretation when omitted
  ROMization processing is performed.
  The following are linked.
  - ROMization area reservation code file (except when specifying the -Xno_stdlib option)
  - Startup routine (cstart.obj) that contains copy function "_rcopy" call (except when specifying the -Xstartup or -Xno_startup option)

[Detailed description]

- This option suppresses ROMization processing.
  The ROMization area reservation code file is not linked.
  The startup routine (cstartN.obj) that does not contain copy function "_rcopy" call is linked (except when specifying the -Xstartup or -Xno_startup option).
- ROMization is a process whereby data that will be expanded in RAM is copied to ROM, and a routine is added to copy this data from ROM into RAM when the application first starts.
- If this option is specified in the application that does not have data with a default value, the code can be reduced.

[Example of use]

- To suppress ROMization processing, describe as:

```
>cx -CF3746 -Xno_romize main.c
```

-Xrompcrt

This option specifies the ROMization area reservation code file.

[Specification format]

```
-Xrompcrt=file
```

- Interpretation when omitted
  When the -Xno_romize option is not specified, the standard ROMization area reservation code file (rompcrt.obj) is linked at the end of input files.

[Detailed description]

- This option links *file* instead of the standard ROMization area reservation code file at the end of input files.
- Specify the object module file as *file*.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.
- If this option is specified together with the -Xno_romize option, this option will be invalid.

[Example of use]

- To link rompack.obj instead of the standard ROMization area reservation code file at the end of input files, describe as:

```
>cx -CF3746 -Xrompcrt=rompack.obj main.c
```

-Xrompsec_start

This option specifies the start address of the rompsec section.

[Specification format]

```
-Xrompsec_start=label
```

- Interpretation when omitted
  The value of label "__S_romp" is regarded as the start address of the rompsec section to be created.

[Detailed description]

- This option performs hex output only for the specified section.
- The value of label *label* is regarded as the start address of the rompsec section to be created.
- If *label* is not in the load module file, or if this option is specified more than once, the option specified last will be valid, and the option specified first will be ignored.
- An error will occur if *label* is omitted.

[Example of use]

- To regard the value of label "romp_start" as the start address of the rompsec section to be created, describe as:

```
>cx -CF3746 -Xrompsec_start=romp_start main.c
```

-Xrompsec_data

This option specifies the data section included in the rompsec section.

[Specification format]

```
-Xrompsec_data=section[,section]...
```

- Interpretation when omitted
  All sections with the data or sdata attribute and the sections allocated to the internal instruction RAM are included in the rompsec section.

[Detailed description]

- This option specifies the data section included in the rompsec section.
- The contents of section *section* and the corresponding address and size information are included in the rompsec section.
- This option is related to the section with the data or sdata attribute.
- If *section* does not exist in the load module file, an error will be output and processing will be stopped.
- *section* cannot include a space.
- An error will occur if *section* is omitted.
- If this option is specified more than once, each data section will be included in the rompsec section in the order which they are specified.

[Example of use]

- To include section "data1" and "data2" in the rompsec section, describe as:

```
>cx -CF3746 -Xrompsec_data=data1,data2 main.c
```

-Xrompsec_text

This option specifies the text section included in the rompsec section.

[Specification format]

```
-Xrompsec_text=section[,section]...
```

- Interpretation when omitted
  Each section allocated to the internal instruction RAM is included in the rompsec section.

[Detailed description]

- This option specifies the text section included in the rompsec section.
- The contents of section *section* and the corresponding address and size information are included in the rompsec section.
- This option is related to the section with the text or const attribute.
- The section that can be specified as *section* is the section with the text or const attribute.
  If the section that has any other attribute is specified, a warning will be output and processing will be stopped.
- If *section* does not exist in the load module file, an error will be output and processing will be stopped.
- *section* cannot include a space.
- An error will occur if *section* is omitted.
- If this option is specified more than once, the section will be included in the rompsec section in the order which they are specified.
- If the particular section is specified by using this option for the input file linked specifying the device file with the internal instruction RAM, the section that is not specified and is allocated to internal instruction RAM will not be included in the rompsec section and also be deleted from the output file.

[Example of use]

- To include section "text1" and "text2" in the rompsec section, describe as:

```
>cx -CF3746 -Xrompsec_text=text1,text2 main.c
```

-Xrompsec_only

This option generates the load module file that has only the rompsec section.

[Specification format]

```
-Xrompsec_only
```

- Interpretation when omitted
  The section with the text attribute is included in the load module file to be generated.

[Detailed description]

- This option generates the load module file that has only the rompsec section; no section with the text attribute is included in the file.

[Example of use]

- To generate the load module file that has only the rompsec section, describe as:

```
>cx -CF3746 -Xrompsec_only main.c
```

-Xromize_check_off

This option omits error checking under ROMization.

[Specification format]

```
-Xromize_check_off=string[,string]
```

- Interpretation when omitted
  A peripheral allocation error of the internal ROM is not checked for the rompsec section.
  The duplicate address of the input file and output file is checked.

[Detailed description]

- This option omits error checking under ROMization.
- The items that can be specified as *string* are shown below.
  An error will occur if any other item is specified.

rom_less	Does not check a peripheral allocation error of the internal ROM for the rompsec section. Specify this item when the ROM-less mode is used. Checking the overflow of the internal ROM is not supported when 1 is selected as the single-chip mode. Disable checking the overflow of the internal ROM by specifying this item.
address	Does not check the duplicate addresses of the input file and output file.

- An error will occur if *string* is omitted.

[Example of use]

- Not to check a peripheral allocation error of the internal ROM for the rompsec section, describe as:

```
>cx -CF3746 -Xromize_check_off=rom_less main.c
```

Hex output control

The hex output control options are as follows.

- -Xhex
- -Xhex_only
- -Xhex_format
- -Xhex_fill
- -Xhex_section
- -Xhex_block_size
- -Xhex_offset
- -Xhex_null
- -Xhex_symtab
- -Xhex_rom_less

-Xhex

This option specifies the hex file name.

[Specification format]

```
-Xhex=file
```

- Interpretation when omitted
  The hex file is output to the same folder as the load module file under the load module file name with the extension replaced by ".hex" (the load module file name that ".hex" is added if the extension of the file name is not ".lmf").

[Detailed description]

- This option outputs the hex file by the name of *file* after linking.
- If only the file name is specified for *file*, the file is output to the same folder as the load module file under the specified file name.
- An error will occur if *file* is omitted.

[Example of use]

- To output hex file "sample.hex" after linking, describe as:

```
>cx -CF3746 -Xhex=sample.hex main.c
```

-Xhex_only

This option executes only hex output.

[Specification format]

```
-Xhex_only[=file]
```

- Interpretation when omitted
  Processing is performed in the usual order of linking, ROMization, and hex output.

[Detailed description]

- This option generates the hex file from the load module file specified as the input file and outputs it by the name of *file*.
- If only the file name is specified for *file*, the file is output to the same folder as the load module file under the specified file name.
- If *file* is omitted, the file is output to the same folder as the load module file under the load module file name with the extension replaced by ".map".
- Use this option when only hex output is executed after generating the load module file.
  An error will occur if a load module file is not specified as the input file.

[Example of use]

- To generate hex file "sample.hex" from load module file "a.lmf", describe as:

```
>cx -CF3746 -Xhex_only=sample.hex a.lmf
```

-Xhex_format

This option specifies the format of the hex file to be output.

[Specification format]

```
-Xhex_format=format
```

- Interpretation when omitted
  The format of the hex file to be output is regarded as the Intel expanded hex format (32-bit address) (It is the same result as when the -Xhex_format=i option is specified).

[Detailed description]

- This option specifies the format of the hex file to be output.
- The items that can be specified as *format* are shown below.

I	Intel expanded hex format (up to 1 MB)
i	Intel expanded hex format (32-bit address) (up to 4 GB)
S	Motorola S type hex format (standard address) (up to 16 MB)
s	Motorola S type hex format (32-bit address) (up to 4 GB)
T	Expanded Tektronix hex format (up to 4 GB)

- An error will occur if *format* is omitted.

[Example of use]

- To regard the format of the hex file to be output as the Motorola S type hex format (standard address), describe as:

```
>cx -CF3746 -Xhex_format=S main.c
```

-Xhex_fill

This option specifies fill processing of the hex file.

[Specification format]

```
-Xhex_fill
-Xhex_fill=value
-Xhex_fill=value,start,size
-Xhex_fill=start,size
```

- Interpretation when omitted
  Fill processing is not performed.

[Detailed description]

- This option converts all codes in the area from address *start* to size *size* into hex format and outputs them.
  The unused area in the specified area is filled with *value*.
- The range that can be specified for *value* is 0x00 to 0xFFFF.
- Specify *value* in hexadecimal.
- *value* can be specified in 1-byte or 2-byte units.
  If the specified value is less than 2 or 4 digits, the higher bits will be filled with 0.
- If *value* is omitted, it is assumed that 0xFF has been specified.
- The range that can be specified for *start* is 0x00 to 0xFFFFFFFF.
  Even if *start* is within the above range, an error will occur if the section does not exist.
- The range that can be specified for *size* is 0x01 to 0x100000000.
- Specify *start* and *size* in hexadecimal.
- If *start* and *size* are omitted, all the codes in the internal ROM area defined by the device file are converted in hex format and output.
- This option can not be specified together with the -Xhex_format=T option.
- When the information of the internal ROM area defined by the device file is not used, specify this option together with the -Xhex_rom_less option.
  In this case, parameter *start* and *size* of this option must be specified.

[Example of use]

- To convert all codes in the area from address 0x1000 to size 0x2000 into hex format and outputs them, describe as:
  The unused area in the area is filled with 0x55.

```
>cx -CF3746 -Xhex_fill=0x55,0x1000,0x2000 main.c
```

-Xhex_section

This option converts the codes in the specified section in hex format and outputs them.

[Specification format]

```
-Xhex_section=section[,section]...
```

- Interpretation when omitted
  All sections which have the section type other than NOBITS and section attribute A are converted in hex format and output them.

[Detailed description]

- This option converts the codes in section *section* in hex format and outputs them.
- An error will occur if *section* does not exist.
- An error will occur if *section* is omitted.
- This option can not be specified together with the -Xhex_fill option.

[Example of use]

- To convert the codes in section "sec" in hex format and output them, describe as:

```
>cx -CF3746 -Xhex_section=sec main.c
```

-Xhex_block_size

This option specifies the maximum length of the block.

[Specification format]

```
-Xhex_block_size=num
```

- Interpretation when omitted
  The default value defined for each hex format is regards as the maximum block length.

[Detailed description]

- This option regards the value specified for *num* as the maximum block length (or, in the case of the Intel expanded hex format or Motorola S type hex format, the number of bytes of the code indicated in one data record).
- The range that can be specified for *num* differs depending on the hex format.
  The range that can be specified for *num* are shown below for each hex format.
  If the specified value is less than the minimum value, a warning will be output and it is corrected to the default value.
  If the specified value exceeds the maximum value, a warning will be output and it is corrected to the maximum value.
  An error will occur if 0 is specified.

Hex Format	Range of *num*	Default Value
Intel expanded	1 to 255 (0x01 to 0xFF)	32 (0x20)
Intel expanded (32-bit address)	1 to 255 (0x01 to 0xFF)	32 (0x20)
Motorola S type (standard address)	1 to 251 (0x01 to 0xFB)	80 (0x50)
Motorola S type (32-bit address)	1 to 250 (0x01 to 0xFA)	80 (0x50)
Extended Tektronix	16 to 255 (0x10 to 0xFF)	255 (0xFF)

- An error will occur if *num* is omitted.

[Example of use]

- To specify 255 as the maximum length of the block, describe as:

```
>cx -CF3746 -Xhex_block_size=255 main.c
```

-Xhex_offset

This option specifies the offset of the address to be output.

[Specification format]

```
-Xhex_offset=num
```

- Interpretation when omitted
  It is regarded that the address to be output does not have the offset.
  The address is output from address 0.

[Detailed description]

- This option outputs the original address, offset by *num*.
- The range that can be specified for *num* is 0x0 to 0xFFFFFFFE.
  An abort error will occur if a value outside the range is specified.
- An error will occur if *num* is omitted.

[Example of use]

- To regard the offset of the address to be output as 0x10000, describe as:

```
>cx -CF3746 -Xhex_offset=0x10000 main.c
```

-Xhex_null

This option generates as many null characters as the size of the section of data without an initial value.

[Specification format]

```
-Xhex_null
```

- Interpretation when omitted
  All codes are converted in hex format, and the unused area is filled with 0xFFFF and output.

[Detailed description]

- This option generates as many null characters (\0) as the size of the section with the section type NOBITS and section attribute A (section for data for which no initial value is specified, such as the .bss and .sbss section).
- This option can not be specified together with the -Xhex_fill option.

[Example of use]

- To generate as many null characters as the size of the section with the section type NOBITS and section attribute A, describe as:

```
>cx -CF3746 -Xhex_null main.c
```

-Xhex_symtab

This option converts the symbol table and outputs it.

[Specification format]

```
-Xhex_symtab=string
```

- Interpretation when omitted
  The symbol table is not output.

[Detailed description]

- This option converts the symbol table and outputs it.
- The items that can be specified as *string* are shown below.
  An error will occur if any other item is specified.

global	Converts only global symbols.
all	Converts local symbols as well.

- An error will occur if *string* is omitted.
- This option is valid only when the -Xhex_format=T option is specified.
- This option can not be specified together with the -Xhex_fill option.

[Example of use]

- To convert the symbol table and output it, describe as:

```
>cx -CF3746 -Xhex_symtab -Xhex_format=T main.c
```

-Xhex_rom_less

This option does not use the information of the internal ROM area when the hex file is filled.

[Specification format]

```
-Xhex_rom_less
```

- Interpretation when omitted
  When this option is omitted and the -Xhex_fill option is specified and parameter *start* and *size* are omitted, the internal ROM area defined by the device file is filled.

[Detailed description]

- This option does not use the information of the internal ROM area defined by the device file when the hex file is filled.
- Specify this option when the -Xhex_fill option is specified and the information of the internal ROM area defined by the device file is not used.
- This option must be specified together with the -Xhex_fill option.
  Besides, parameter *start* and *size* of the -Xhex_fill option must be specified.
  An error will occur if the -Xhex_fill option and parameter *start* and *size* is omitted.
- A warning will not be output even if parameter *start* and *size* of the -Xhex_fill option exceeds the internal ROM area.

[Example of use]

- Not to use the information of the internal ROM area defined by the device file when the hex file is filled, describe as:

```
>cx -CF3746 -Xhex_rom_less -Xhex_fill=0xff,0x00,1000 main.c
```

Multi-core support specification

The multi-core suppor specification options are as follows.

- -Xmulti
- -Xmulti_link

-Xmulti

This option specifies the generation of a subprogram of a multi-core program.

[Specification format]

```
-Xmulti=type
```

- Interpretation when omitted
  A single-core program is generated.

[Detailed description]

- This option specifies the subprogram type when generating a subprogram of a multi-core program.
- If this option is specified for a target that is not a multi-core CPU, a warning will be output and this option will be ignored.
- The items that can be specified as *type* are shown below.
  An error will occur if any other item is specified.

pe*n*	Generates a program for core *n*. The value that can be specified for *n* is an integer in the range of 1 to the number of cores on the target CPU. An error will occur if any other value is specified.
cmn	Generate the common module. r0 relative instructions for acquiring variable access and function addresses are generated (handled in the same way as if a "#pragma nopic" directive is described at the top of the file). An error will occur if a "#pragma pic" directive is described. An error will occur if the data be allocated to a section with relocation attributes other than data, const, or sconst (e.g. sdata) is specified.

- When this option is specified, code corresponding to *type* will be generated during compilation, symbol references that can be resolved within the scope of the subprogram will be resolved, and then linking will be performed, and a relinkable load module file will be created.
  At this time, the default startup routine and standard libraries will not be linked.
  Additionally, ROMization processing and generation of a hex file will not be performed.
- When this option is specified, the following preprocessor macros are set automatically.

Type	Preprocessor Macro to Define (Value is 1)
pe*n*	__MULTI_CORE__, MULTI_PE*n*__
cmn	__MULTI_CORE__, MULTI_CMN__

- Specifying this option affects the following options.
  - Implicitly enabled options

```
-Xno_startup, -Xno_stdlib, -Xno_romize, -Xrelinkable_object
```

  - Ignored options

```
-l, -L, -Xstartup, -Xmap, -Xno_romize, -Xrompcrt, -Xrompsec_start, -Xrompsec_data,
-Xrompsec_text, -Xrompsec_only, -Xromize_check_off, -Xhex, -Xhex_only, -Xhex_format,
-Xhex_fill, -Xhex_section, -Xhex_block_size, -Xhex_offset, -Xhex_null, -Xhex_symtab,
-Xhex_rom_less
```

- To generate library files for each of the subprograms, specify this option at the same time as the -c option, and create an object module file.

Caution Do not create a library file for object module files created with different -Xmulti options specified.

- This option is equivalent to the following property in CubeSuite.
  - From the [Build Settings] tab, [Target core number] in the [Multi-Core] category

[Example of use]

- To generate a program for core 1, describe as:

```
>cx -CF3515 -Xmulti=pe1 file_pe1_1.c file_pe1_2.c -ope1.lmf
```

-Xmulti_link

This option specifies the linking of multi-core subprograms.

[Specification format]

```
-Xmulti_link
```

- Interpretation when omitted
  It will not be treated the linking of multi-core subprograms.

[Detailed description]

- This option links each generated subprograms when generating a multi-core program.
- When this option is specified, each subprogram, library file, and object module file are input, the addresses between the subprogram are resolved, and the multi-core program is generated.
  At this time, the startup routine and libraries supporting multi-core will be linked.
  Additionally, ROMization processing and generation of a hex file will be performed at the same time.
- An error will occur if a C source file or assembler source file is specified as input.
  If another type of file is specified, then if a necessary file is not specified, and there are remaining unresolved gp, ep, or tp relative symbol references, then an error will occur during linking.
- If this option is specified together with the -Xmulti option, an error will occur.

[Example of use]

- To link multi-core subprograms: pe1.lmf, pe2.lmf, and cmn.lmf, describe as:

```
>cx -CF3515 -Xmulti_link pe1.lmf pe2.lmf cmn.lmf -otarget.lmf -lmulti_lib
```

Information file output control

The information file output control options are as follows.

- -Xcube_suite_info
- -Xno_cube_suite_info
- -Xsfg
- -Xsfg_opt
- -Xsfg_size_tidata
- -Xsfg_size_tidata_byte
- -Xsfg_size_sidata
- -Xsfg_size_sedata
- -Xsfg_size_sdata

-Xcube_suite_info

This option outputs the CubeSuite information file.

[Specification format]

```
-Xcube_suite_info=file
```

- Interpretation when omitted
  The CubeSuite information file is not output.

[Detailed description]

- This option analyzes input files statically and outputs the information that the IDE, editor, etc. of the CubeSuite use to *file*.
- ".cref" is recommended as the extension of *file*.
- If *file* already exists, it will be updated.
  - When the input is a C source file
    If the information about the C source file exists in *file*, it will be deleted and replaced with new contents.
    If the information does not exist, it will be added into *file*.
  - When the input is an assembler source file
    If the information about the assembler source file exists in *file*, it will be deleted and replaced with new contents.
    If the information does not exist, it will be added into *file*.
    If the information about the original C source file exists of the assembler source file in *file*, it will be deleted.
    The original C source file is acquired from the .FILE quasi instruction in the assembler source file.
  - When the input is an object module file
    If the information about the source file of the object module file exists in *file*, only information determined during linking in it will be deleted and replaced with new contents.
    If the information does not exist, nothing will be output.
- An error will occur if *file* is omitted.
- If two or more files are input and if C source files, assembler source files, and object module files are mixed, each input file will be updated as above and the information will be output to *file*.
- The information about a library file is not output.
  Information for static functions that are not called (unless -Odelete_static_func=off is specified) and information for static variables in unused files are not output.

- It is recommend that this option is specified identically for all source files in the given application.

[Example of use]

- To output the CubeSuite information file as file name "info.cref", describe as:

```
>cx -CF3746 -Xcube_suite_info=info.cref main.c
```

-Xno_cube_suite_info

This option suppresses the output of the CubeSuite information file.

[Specification format]

```
-Xno_cube_suite_info
```

- Interpretation when omitted
  If the -Xcube_suite_info option is specified, then the CubeSuite information file will be output.
  If the -Xcube_suite_info option is not specified, then the CubeSuite information file will not be output.

[Detailed description]

- If this option and the -Xcube_suite_info option are specified at the same time, the -Xcube_suite_info option will be invalid, regardless of the order in which they are specified.
  If the file specified by the -Xcube_suite_info option exists, then it will be deleted.
- The -Xcube_suite_info option is always added when started from CubeSuite. Specify this option if you want to suppress the addition (e.g. to speed up compilation).
- If this option is specified, then some of the CubeSuite functions will not be available.

[Example of use]

- To suppress the output of the CubeSuite information file, describe as:

```
>cx -CF3746 -Xcube_suite_info=info.cref main.c -Xno_cube_suite_info
```

-Xsfg

This option generates a symbol information file.

[Specification format]

```
-Xsfg[=file]
```

- Interpretation when omitted
  A symbol information file is not generated.

[Detailed description]

- This option generates symbol information file *file* based on the static analysis result of the C source file.
- ".sfg" is recommended as the extension of *file*.
- If *file* already exists, it will be overwritten.
- If *file* is omitted, the file is output to the same folder as the load module file under the load module file name with the extension replaced by ".map".
- This option must be specified together with the -Xcube_suite_info option to use the information of the CubeSuite information file.
- It is recommend that this option is specified identically for all source files in the given application.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Output symbol information file] in the [Symbol Information] category
  - From the [Link Options] tab, [Output folder for symbol information file] in the [Symbol Information] category
  - From the [Link Options] tab, [Symbol information file name] in the [Symbol Information] category

[Example of use]

- To output the symbol information file as file name "symbol.sfg", describe as:

```
>cx -CF3746 -Xsfg=symbol.sfg -Xcube_suite_info=info.cref main.c
```

-Xsfg_opt

This option outputs the optimum allocation information.

[Specification format]

```
-Xsfg_opt
```

- Interpretation when omitted
  Allocation information is output with variables are sorted in the order starting from highest use frequency.

[Detailed description]

- Variables are sorted in the order starting from highest use frequency for each section, and the optimum allocation information is output, so that they can be allocated within the sizes of the .tidata.byte, .tidata.word, .sidata, .sedata, and .sdata sections.
- The size of each section can be specified by the -Xsfg_size_tidata, -Xsfg_size_tidata_byte, -Xsfg_size_sidata, -Xsfg_size_sedata, and -Xsfg_size_sdata options.
  When these option are not specified, the size of each section are shown below.

Section Name	Size (byte)
.tidata.byte	128
.tidata.word	128
.sidata	32512
.sedata	32768
.sdata	65536

- This option is valid only when the -Xsfg option is specified.
  If the -Xsfg option is not specified, this option will be ignored.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Output optimized allocation information] in the [Symbol Information] category

[Example of use]

- To sort variables in the order starting from highest use frequency for each section and output the optimum allocation information, describe as:

```
>cx -CF3746 -Xsfg_opt -Xsfg main.c
```

-Xsfg_size_tidata

This option specifies the size of .tidata section.

[Specification format]

```
-Xsfg_size_tidata=num
```

- Interpretation when omitted
  The frequency of variable use is calculated, limiting the size of variables allocated to the .tidata section to 128 bytes.

[Detailed description]

- This option calculates the frequency of variable use, limiting the size of variables allocated to the .tidata section to *num* bytes.
- Specify 0 to 256 as *num*.
  An error will occur if a value outside the range is specified.
- An error will occur if *num* is omitted.
- The .tidata section is 256 bytes by default. It is internally divided into the .tidata.byte section (maximum 128 bytes by default) and .tidata.word section.
  .tidata section = .tidata.byte section + .tidata.word section
  Variables are selected until the total size of the .tidata.byte and tidata.word sections reaches 256 bytes.
  However, the selection is stopped when the size of the .tidata.byte section reaches the specified size or 256 bytes.
- If the value smaller than the size of the .tidata.byte section, it is assumed that the .tidata and .tidata.byte sections have the same size.
  Therefore, the size of the .tidata.word section in this case is 0.
- This option is valid only when the -Xsfg_opt option is specified.
  If the -Xsfg_opt option is not specified, this option will be ignored.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Size of .tidata section] in the [Symbol Information] category

[Example of use]

- To calculate the frequency of variable use, limiting the size of variables allocated to the .tidata section to 128 bytes, describe as:

```
>cx -CF3746 -Xsfg_size_tidata=128 -Xsfg_opt main.c
```

-Xsfg_size_tidata_byte

This option specifies the size of .tidata.byte section.

[Specification format]

```
-Xsfg_size_tidata_byte=num
```

- Interpretation when omitted
  The frequency of variable use is calculated, limiting the size of variables allocated to the .tidata.byte section to 128 bytes.

[Detailed description]

- This option calculates the frequency of variable use, limiting the size of variables allocated to the .tidata.byte section to *num* bytes.
- Specify 0 to 128 as *num*.
  An error will occur if a value outside the range is specified.
- An error will occur if *num* is omitted.
- This option is valid only when the -Xsfg_opt option is specified.
  If the -Xsfg_opt option is not specified, this option will be ignored.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Size of .tidata.byte section] in the [Symbol Information] category

[Example of use]

- To calculate the frequency of variable use, limiting the size of variables allocated to the .tidata.byte section to 64 bytes, describe as:

```
>cx -CF3746 -Xsfg_size_tidata_byte=64 -Xsfg_opt main.c
```

-Xsfg_size_sidata

This option specifies the size of .sidata section.

[Specification format]

```
-Xsfg_size_sidata=num
```

- Interpretation when omitted
  The frequency of variable use is calculated, limiting the size of variables allocated to the .sidata section to 32512 bytes.

[Detailed description]

- This option calculates the frequency of variable use, limiting the size of variables allocated to the .sidata section to *num* bytes.
- Specify 0 to 32512 as *num*.
  An error will occur if a value outside the range is specified.
- An error will occur if *num* is omitted.
- This option is valid only when the -Xsfg_opt option is specified.
  If the -Xsfg_opt option is not specified, this option will be ignored.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Size of .sidata section] in the [Symbol Information] category

[Example of use]

- To calculate the frequency of variable use, limiting the size of variables allocated to the .sidata section to 32000 bytes, describe as:

```
>cx -CF3746 -Xsfg_size_sidata=32000 -Xsfg_opt main.c
```

-Xsfg_size_sedata

This option specifies the size of .sedata section.

[Specification format]

```
-Xsfg_size_sedata=num
```

- Interpretation when omitted
  The frequency of variable use is calculated, limiting the size of variables allocated to the .sedata section to 32768 bytes.

[Detailed description]

- This option calculates the frequency of variable use, limiting the size of variables allocated to the .sedata section to *num* bytes.
- Specify 0 to 32768 as *num*.
  An error will occur if a value outside the range is specified.
- An error will occur if *num* is omitted.
- This option is valid only when the -Xsfg_opt option is specified.
  If the -Xsfg_opt option is not specified, this option will be ignored.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Size of .sedata section] in the [Symbol Information] category

[Example of use]

- To calculate the frequency of variable use, limiting the size of variables allocated to the .sedata section to 16384 bytes, describe as:

```
>cx -CF3746 -Xsfg_size_sedata=16384 -Xsfg_opt main.c
```

-Xsfg_size_sdata

This option specifies the size of .sdata section.

[Specification format]

```
-Xsfg_size_sdata=num
```

- Interpretation when omitted
  The frequency of variable use is calculated, limiting the size of variables allocated to the .sdata section to 65536 bytes.

[Detailed description]

- This option calculates the frequency of variable use, limiting the size of variables allocated to the .sdata section to *num* bytes.
- Specify 0 to 65536 as *num*.
  An error will occur if a value outside the range is specified.
- An error will occur if *num* is omitted.
- This option is valid only when the -Xsfg_opt option is specified.
  If the -Xsfg_opt option is not specified, this option will be ignored.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Size of .sdata section] in the [Symbol Information] category

[Example of use]

- To calculate the frequency of variable use, limiting the size of variables allocated to the .sdata section to 32768 bytes, describe as:

```
>cx -CF3746 -Xsfg_size_sdata=32768 -Xsfg_opt main.c
```

Error output control

The error output control options are as follows.

- -Xerror_file

-Xerror_file

This option outputs error messages to a file.

[Specification format]

```
-Xerror_file=file
```

- Interpretation when omitted
  Error messages are output to only the standard error output.

[Detailed description]

- This option outputs error messages to the standard error output and file *file*.
- If *file* already exists, it will be overwritten.
- An error will occur if *file* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Output error message file] in the [Error Output] category
  - From the [Common Options] tab, [Error message file output folder] in the [Error Output] category
  - From the [Common Options] tab, [Error message file name] in the [Error Output] category
  - From the [Individual Compile Options] tab, [Output error message file] in the [Error Output] category
  - From the [Individual Compile Options] tab, [Error message file output folder] in the [Error Output] category
  - From the [Individual Compile Options] tab, [Error message file name] in the [Error Output] category

[Example of use]

- To output error messages to the standard error output and file "err", describe as:

```
>cx -CF3746 -Xerror_file=err main.c
```

Warning message output control

The warning message output control options are as follows.

- -Xwarning
- -Xno_warning

-Xwarning

This option outputs the specified warning message.

[Specification format]

```
-Xwaning=num[,num]...
-Xwaning=num1-num2
```

- Interpretation when omitted
  Severe warning messages are output (same as when "-Xwarning_level=1" is specified).

[Detailed description]

- This option outputs the specified warning message.
- Specify the error numbers as *num*, *num1*, and *num2*.
  If the error number that does not exist, it will be ignored.
- An error will occur if *num*, *num1*, or *num2* is omitted.
- If *num1-num2* is specified, it is assumed that error numbers within the range have been specified.
- The error number specified by this option is the rightmost 5 digits of the 7-digit number following the "W".
  See "CubeSuite Message" for error numbers.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Displayed warning message] in the [Warning Message] category
  - From the [Individual Compile Options] tab, [Displayed warning message] in the [Warning Message] category

[Example of use]

- To output warning message "W0566002", describe as:

```
>cx -CF3746 -Xwaning=66002 main.c
```

-Xno_warning

This option suppresses outputting warning messages of the specified number.

[Specification format]

```
-Xno_waning=num[,num]...
-Xno_waning=num1-num2
```

- Interpretation when omitted
  Severe warning messages are output (same as when "-Xwarning_level=1" is specified).

[Detailed description]

- This option suppresses outputting warning messages of the specified number.
- Specify the error numbers as *num*, *num1*, and *num2*.
  If the error number that does not exist, it will be ignored.
- An error will occur if *num*, *num1*, or *num2* is omitted.
- If *num1*-*num2* is specified, it is assumed that error numbers within the range have been specified.
- The error number specified by this option is the rightmost 5 digits of the 7-digit number following the "W".
  See "CubeSuite Message" for error numbers.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Undisplayed warning message] in the [Warning Message] category
  - From the [Individual Compile Options] tab, [Undisplayed warning message] in the [Warning Message] category

[Example of use]

- To suppress outputting warning message "W0566002", describe as:

```
>cx -CF3746 -Xno_waning=66002 main.c
```

Phase individual option specification

The phase individual option specification options are as follows.

- -Xasm_option
- -Xlk_option

-Xasm_option

This option specifies the file to be assembled.

[Specification format]

```
-Xasm_option=file[,file]...
```

- Interpretation when omitted
  Only ".asm" and ".s" are recognized as the assembly target.

[Detailed description]

- This option specifies so that a file that is not recognized as an assembler source file (a file other than ".asm" and ".s") is the assembly target.
- Specify the file to be assembled as *file*.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.

[Example of use]

- To assemble file "assemble.test", describe as:

```
>cx -CF3746 -Xasm_option=assemble.test
```

-Xlk_option

This option specifies the file to be linked.

[Specification format]

```
-Xlk_option=file[,file]...
```

- Interpretation when omitted
  Only ".obj", ".lib", and ".lmf" are recognized as the link target.

[Detailed description]

- This option specifies so that a file that is not recognized as a link target (a file other than ".obj", ".lib", and ".lmf") is the link target.
- Specify the file to be linked as *file*.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.

[Example of use]

- To link file "link.test", describe as:

```
>cx -CF3746 -Xlk_option=link.test
```

Command file specification

The command file specification option is as follows.

- @

@

This option specifies a command file.

[Specification format]

```
@file
```

- Interpretation when omitted
  Only the options and file names specified on the command line are recognized.

[Detailed description]

- This option handles *file* as a command file.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.
- See "(2) Startup from a command file" for details about a command file.

[Example of use]

- To handle "command" as a command file, describe as:

```
>cx @command
```

(2) Assemble options

The types and explanations for options of the assemble phase are shown below.

Table B-4. Assemble Options

Classification	Option	Description
Version/help display specification	-V	This option displays the version information of the cx.
	-h	This option displays the descriptions of the cx options.
Output file specification	-o	This option specifies the output file name.
	-Xobj_path	This option specifies where the object module file generated during assembling is to be saved.
	-Xprn_path	This option specifies where an assemble list file is to be saved.
Source debugging control	-g	This option outputs information for source debugging.
Device specification	-C	This option specifies the target device.
	-Xcommon	This option specifies that an object module file common to the various devices is generated.
	-Xdev_path	This option specifies the folder to search device files.
Preprocessor control	-D	This option defines assembler symbols.
	-U	This option deletes the assembler symbol definition by the -D option.
	-I	This option specifies the folder to search include files.
Japanese/Chinese character control	-Xcharacter_set	This option specifies the Japanese/Chinese character code.
Generated code control	-Xsdata	This option specifies the maximum size of data allocated to the .sdata or .sbss section.
Assembler control specification	-Xasm_far_jump [V850E2][V850E2V3]	This option controls outputting far jump for an assembler source file.
Link control	-Xflash	This option generates the load module file for the flash area.
Multi-core support specification	-Xmulti	This option specifies the generation of a subprogram of a multi-core program.
Error output control	-Xerror_file	This option outputs error messages to a file.
Warning message output control	-Xwarning	This option outputs the specified warning message.
	-Xno_warning	This option suppresses outputting warning messages of the specified number.
Command file specification	@	This option specifies a command file.

Table B-5. Mark Used in Option Descriptions

[V850E2]	This option is only for devices with the V850E2 core and V850E2 instruction set architecture.
[V850E2V3]	This option is only for devices with the V850E2 core and V850E2V3 instruction set architecture.

Version/help display specification

The version/help display specification options are as follows.

- -V
- -h

-V

This option displays the version information of the cx.

[Specification format]

```
-V
```

- Interpretation when omitted
  Assembling is performed without displaying the version information of the cx.

[Detailed description]

- This option outputs the version information of the cx to the standard error output.
  It does not execute assembling.

[Example of use]

- To output the version information of the cx to the standard error output, describe as:

```
>cx -CF3746 -V
```

-h

This option displays the descriptions of the cx options.

[Specification format]

```
-h
```

- Interpretation when omitted
  The descriptions of the cx options are not displayed.

[Detailed description]

- This option outputs the descriptions of the cx options to the standard error output.
  It does not execute compilation.
- This option outputs the descriptions of the cx options to the standard error output.
  It does not execute assembling.

[Example of use]

- To output the descriptions of the cx options to the standard error output, describe as:

```
>cx -CF3746 -h
```

Output file specification

The output file specification options are as follows.

- -o
- -Xobj_path
- -Xprn_path

-o

This option specifies the output file name.

[Specification format]

```
-ofile
```

- Interpretation when omitted
  The file is output to the current folder.
  - If this option is specified with the -c option
    The output object module file name will be the source file name with the extension replaced by ".obj".
  - Other than above
    The output load module file name is "a.lmf".

[Detailed description]

- This option specifies the output file name as *file*.
- If *file* already exists, it will be overwritten.
- This option is valid when processing is interrupted by specifying the -c option.
  - If this option is specified with the -c option
    It is assumed that an object module file name has been specified as *file*.
  - Other than above
    It is assumed that a load module file name has been specified as *file*.
- An error will occur if two or more files are output.
- An error will occur if *file* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Individual Assemble Options] tab, [Object module file name] in the [Output File] category

[Example of use]

- To output the load module file with "sample.lmf" as the file name, describe as:

```
>cx -CF3746 -osample.lmf main.asm
```

-Xobj_path

This option specifies where the object module file generated during assembling is to be saved.

[Specification format]

```
-Xobj_path[=path]
```

- Interpretation when omitted
  The object module file is saved under the source file name with the extension replaced by ".obj" to the current folder.
  However, the object module file will not be saved if one source file is specified as input and the -c option is not specified.

[Detailed description]

- This option saves the object module file generated during assembling to *path*.
  - If an existing file is specified as *path*
    If one object module file is output, it will be saved with *path* as the file name.
    If two or more object module files are output, an error will occur.
  - If an existing folder is specified as *path*
    The object module file is saved under the source file name with the extension replaced by ".obj" to *path*.
  - If the specified folder or file as *path* does not exist
    An error will occur.
  - If “=*path*“ is omitted
    The object module file is saved under the source file name with the extension replaced by ".obj" to the current folder.
- If two or more files with the same name (even if they are in different folders) are specified as source files, then a warning is output, and an object module file is only saved for the last source file to be specified.

[Example of use]

- To save the object module file generated during assembling with "sample.obj" as a file name, describe as:

```
>cx -CF3746 -Xobj_path=sample.obj main.asm
```

-Xprn_path

This option specifies where an assemble list file is to be saved.

[Specification format]

```
-Xprn_path[=path]
```

- Interpretation when omitted
  An assemble list file will not be output.

[Detailed description]

- This option outputs an assemble list file when assembling and saves it to *path*.
  - If an existing file is specified as *path*
    If one assemble list file is output, it will be saved with *path* as the file name.
    If two or more assemble list files are output, an error will occur.
  - If an existing folder is specified as *path*
    The assemble list file is saved under the source file name with the extension replaced by ".prn" to *path*.
  - If the specified folder or file as *path* does not exist
    An error will occur.
  - If “=*path*“ is omitted
    The assemble list file is saved under the source file name with the extension replaced by ".prn" to the current folder.
- If two or more files with the same name (even if they are in different folders) are specified as source files, then a warning is output, and an assemble list file is only saved for the last source file to be specified.
- This option is equivalent to the following property in CubeSuite.
  - From the [Assemble Options] tab, [Output assemble list file] in the [Assemble List] category
  - From the [Assemble Options] tab, [Output folder for assemble list file] in the [Assemble List] category
  - From the [Individual Assemble Options] tab, [Output assemble list file] in the [Assemble List] category
  - From the [Individual Assemble Options] tab, [Output folder for assemble list file] in the [Assemble List] category

[Example of use]

- To save the assemble list file that is output when assembling with "sample.prn" as a file name, describe as:

```
>cx -CF3746 -Xprn_path=sample.prn main.asm
```

Source debugging control

The source debugging control options are as follows.

- -g

-g

This option outputs information for source debugging.

[Specification format]

```
-g
```

- Interpretation when omitted
  Information for source debugging will not be output.

[Detailed description]

- This option outputs information for source debugging to the output file.
- Source debugging can be performed by specifying this option.
- This option is equivalent to the following property in CubeSuite.
  - From the [Assemble Options] tab, [Add debug information] in the [Debug Information] category
  - From the [Individual Assemble Options] tab, [Add debug information] in the [Debug Information] category

[Example of use]

- To output information for source debugging to the output file, describe as:

```
>cx -CF3746 -g main.asm
```

Device specification

The device specification options are as follows.

- -C
- -Xcommon
- -Xdev_path

-C

This option specifies the target device.

[Specification format]

```
-Cdevice
```

- Interpretation when omitted
  If the -Xcommon option is specified, the behavior is in accordance with that specification.
  Otherwise, an error will occur (except when specifying the -V, -h, or -P option).
  An error will occur when linking is performed.

[Detailed description]

- This option specifies the target device.
- See the user's manual of each device file about device types that can be specified as *device*.
- If *device* does not exist (the corresponding device file does not exist), an error will occur.
- An error will occur if *device* is omitted.
- This option cannot be omitted when linking is performed.

[Example of use]

- To specify uPD70F3746 as the target device, describe as:

```
>cx -CF3746 main.asm
```

-Xcommon

This option specifies that an object module file common to the various devices is generated.

[Specification format]

```
-Xcommon=series
```

- Interpretation when omitted
  If the -C option is specified, the behavior is in accordance with that specification.
  Otherwise, an error will occur.

[Detailed description]

- This option specifies that an object module file common to the various devices is generated.
- When this option is specified, only instructions are used in the instruction set architecture of the target, and common magic number *series* that supports the instruction set architecture is embed into the object module file.
- The items that can be specified as *series* are shown below.
  An abort error will occur if any other item is specified.

v850e	It is possible to link to the model with the instruction set architecture superior to V850E (V850E, V850E2, or V850E2V3), which is specified as the target device.
v850e2	It is possible to link to the model with the instruction set architecture superior to V850E2 (V850E2 or V850E2V3), which is specified as the target device.
v850e2v3	It is possible to link to the model with the V850E2V3 instruction set architecture, which is specified as a target device. If the instruction set architecture of the target device is V850E2V3, we recommend specifying this item to get the best performance.

- An error will occur if *series* is omitted.
- If this option and the -C option are specified at the same time, the processing is shown below.

Parameter *series* of -Xcommon	Device Specified by -C		
	V850E	V850E2	V850E2V3
v850e	Normal processing	Replaces with -Xcommon=v850e2 (Outputs a warning)	Replaces with -Xcommon=v850e2v3 (Outputs a warning)
v850e2	Normal processing (Outputs a warning)	Normal processing	Replaces with -Xcommon=v850e2v3 (Outputs a warning)
v850e2v3	Normal processing (Outputs a warning)	Normal processing (Outputs a warning)	Normal processing

If the instruction set architecture of the device specified by the -C option is the one specified by this option, both options are processed.
If the instruction set architecture of the device specified by the -C option is inferior to the one specified by this option (V850E2V3 > V850E2 > V850E/ES), a warning is output and both options are processed.

If the instruction set architecture of the device specified by the -C option is superior to the one specified by this option, a warning is output and parameter *series* of this option is replaced by the instruction set architecture of the device specified by the -C option.

[Example of use]

- To embed the magic number common to models with the instruction set architectures superior to V850E into the object module file to be generated, describe as:

```
>cx -Xcommon=v850e -c main.asm
```

-Xdev_path

This option specifies the folder to search device files.

[Specification format]

```
-Xdev_path=path
```

- Interpretation when omitted
  The device file is searched from the standard device file folder.

[Detailed description]

- This option searches a device file from folder *path*.
- If the folder as *path* does not exist, or if the device file specified by the -C option cannot be found, a warning will be output and the standard device file folder[Note] will be searched.
  If it cannot be found, an error will occur.

Note The device file is searched in the following order.

[V850E2V3]
1. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device\V850E2\Devicefile
2. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device\V850\Devicefile
3. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device_Custom\ Devicefile

[V850E/V850E2]
1. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device\V850\Devicefile
2. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device_Custom\ Devicefile

- An error will occur if *path* is omitted.

[Example of use]

- To search a device file from folder C:\NECTools32\dev, describe as:

```
>cx -CF3746 -Xdev_path=C:\NECTools32\dev main.asm
```

Preprocessor control

The preprocessor control options are as follows.

- -D
- -U
- -I

-D

This option defines assembler symbols.

[Specification format]

```
-Dname[=def][name[=def]]...
```

- Interpretation when omitted
  None

[Detailed description]

- This option defines *name* as an assembler symbol.
- This is equivalent to adding ".SET *name* def" at the beginning of the assembler source program.
- If *name* contains characters that are allowed in an assembler symbol, but which cannot be used in a preprocessor macro ("@", ".", and "~"), a warning will be displayed, and it is defined as an assembler symbol only.
- An error will occur if *name* is omitted.
- If “=*def*“ is omitted, *def* is regarded as 1.
- This option can be specified more than once.
- If both this option and -U option are specified for the same assembler symbol, the option specified last will be valid.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Macro definition] in the [Frequently Used Options(for Assemble)] category
  - From the [Assemble Options] tab, [Macro definition] in the [Preprocess] category
  - From the [Individual Assemble Options] tab, [Macro definition] in the [Preprocess] category

[Example of use]

- To define "sample=256" as an assembler symbol, describe as:

```
>cx -CF3746 -Dsample=256 main.asm
```

-U

This option deletes the assembler symbol definition by the -D option.

[Specification format]

```
-Uname[,name]...
```

- Interpretation when omitted
  None

[Detailed description]

- This option deletes the definition of assembler symbol *name* by the -D option.
- An error will occur if *name* is omitted.
- This option cannot delete the definition by describing ".SET *name def*".
- This option can be specified more than once.
- This option is equivalent to the following property in CubeSuite.
  - From the [Assemble Options] tab, [Macro undefinition] in the [Preprocess] category
  - From the [Individual Assemble Options] tab, [Macro undefinition] in the [Preprocess] category

[Example of use]

- To delete the definition of assembler symbol "test" by the -D option, describe as:

```
>cx -CF3746 -Utest main.asm
```

-I

This option specifies the folder to search include files.

[Specification format]

```
-Ipath[,path]...
```

- Interpretation when omitted
  The include file is searched from the standard include file folder.

[Detailed description]

- This option specifies the folder to search include files that are read by assembler control instruction "$INCLUDE/ $BINCLUDE" as *path*.
  Include files are searched according to the following sequence.

(1) Path specified by the -I option

(2) Folder with source file

(3) Folder of original C source file

(4) Current folder

- If *path* does not exist, a warning will be output.
- An error will occur if *path* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Additional include paths] in the [Frequently Used Options(for Assemble)] category
  - From the [Common Options] tab, [System include paths] in the [Frequently Used Options(for Assemble)] category
  - From the [Assemble Options] tab, [Additional include paths] in the [Preprocess] category
  - From the [Assemble Options] tab, [System include paths] in the [Preprocess] category
  - From the [Individual Assemble Options] tab, [Additional include paths] in the [Preprocess] category
  - From the [Individual Assemble Options] tab, [Use whole include paths specified for build tool] in the [Preprocess] category

[Example of use]

- To search include files from the current folder, folder D:\include, the standard folder in that order, describe as:

```
>cx -CF3746 -ID:\include main.asm
```

Japanese/Chinese character control

The Japanese/Chinese character control option is as follows.

- -Xcharacter_set

-Xcharacter_set

This option specifies the Japanese/Chinese character code.

[Specification format]

```
-Xcharacter_set=code
```

- Interpretation when omitted
  The Japanese character code is handled as SJIS.

[Detailed description]

- This option specifies the character code to be used for Japanese comments and character strings in the source file.
- The items that can be specified as *code* are shown below.
  An abort error will occur if any other item is specified.
  Operation is not guaranteed if the specified character code differs from the character code of the source file.

none	Does not process the Japanese and Chinese character code
euc_jp	EUC (Japanese)
sjis	SJIS
utf8	UTF-8
big5	Traditional Chinese
gb2312	Simplified Chinese

- An error will occur if *code* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Assemble Options] tab, [Character encoding] in the [Character Encoding] category
  - From the [Individual Assemble Options] tab, [Character encoding] in the [Character Encoding] category

[Example of use]

- To specify EUC as the character code to be used for Japanese comments and character strings in the input file, describe as:

```
>cx -CF3746 -Xcharacter_set=euc_jp main.asm
```

Generated code control

The generated code control options are as follows.

- -Xsdata

-Xsdata

This option specifies the maximum size of data allocated to the .sdata or .sbss section.

[Specification format]

```
-Xsdata=num
```

- Interpretation when omitted
  All data is allocated to the .sdata section or .sbss section.
  However, static variables with a const modifier are allocated to the .const section.

[Detailed description]

- This option allocates data of less than *num* bytes to the .sdata or .sbss section.
- Data specified the .sdata or .sbss section by the "#pragma section" directive is allocated to that section regardless of the size.
- This option is not applicable for arrays of incomplete types (arrays whose size is indeterminate within the file).
- Specify 0 to 65535 as *num*.
  An abort error will occur if a value outside the range is specified.
- An error will occur if *num* is omitted.
- The yardstick for the value to be specified as *num* can be output by the -Xsdata_info option.
- If a different option is specified for each file, a code of a different method of placing and referencing variables may be generated and an error or warning may be output during linking.
- This option is equivalent to the following property in CubeSuite.
  - From the [Assemble Options] tab, [Size threshold of sdata/sbss section allocation(Bytes)] in the [Output Code] category

[Example of use]

- To allocate data of less than 16 bytes to the .sdata or .sbss section, describe as:

```
>cx -CF3746 -Xsdata=16 main.asm
```

Assembler control specification

The assembler control specification option is as follows.

- -Xasm_far_jump [V850E2][V850E2V3]

-Xasm_far_jump [V850E2][V850E2V3]

This option controls outputting far jump for an assembler source file.

[Specification format]

```
-Xasm_far_jump
```

- Interpretation when omitted
  Assembly is performed as a jarl or jr instruction.

[Detailed description]

- For an assembler source file, this option assumes that all jarl and jr instructions described in the source are jarl32 and jr32 instructions, and assembling is performed.
- If you wish to control individual instructions, add jarl22/jarl32 or jr22/jarl22 to the source.
- This option does not affect the jump instruction.
- If this option is specified for a C source file, that will be ignored without displaying a warning.
- This option is equivalent to the following property in CubeSuite.
  - From the [Assemble Options] tab, [Use 32-bit branch instruction] in the [Output Code] category
  - From the [Individual Assemble Options] tab, [Use 32-bit branch instruction] in the [Output Code] category

[Example of use]

- To assume that all jarl and jr instructions described in the source are jarl32 and jr32 instructions, and perform assembling, describe as:

```
>cx -CF3746 -Xasm_far_jump main.asm
```

Link control

The link control option is as follows.

- -Xflash

-Xflash

This option generates the load module file for the flash area.

[Specification format]

```
-Xflash=file
```

- Interpretation when omitted
  When the boot-flash re-link function is used, the load module file for the boot area is generated.
  When the boot-flash re-link function is not used, normal link processing is performed.

[Detailed description]

- This option generates the load module file for the flash area when the boot-flash re-link function is used.
  At this time, symbol information of load module file for the boot area *file* is referred and link processing is performed.
- Specify the load module file for the boot area that is generated using the boot-flash re-link function as *file*.
  The load module file specified here must be the file before ROMization processing (generated with the -Xno_romize or -Xlink_output option specified).
- An error will occur if a nonexistent file is specified as *file*.
- An error will occur if *file* is omitted.
- This option must be specified together with the -Xflash_ext_table option.

[Example of use]

- To generate load module file for the boot area "boot.lmf" with 0x200 as the start address of the branch table for the flash area, describe as:

```
>cx -CF3746 -Xflash_ext_table=0x200 -Xlink_output=boot.lmf boot1.asm boot2.asm -
Xlink_directive=boot.dir
```

To create the branch table at address 0x200 and generate load module file for the flash area "flash.lmf", describe as:
At this time, symbol information of load module file for the boot area "boot.lmf" is referred and link processing is performed.

```
>cx -CF3746 -Xflash_ext_table=0x200 -Xflash=boot.lmf -oflash.lmf flash1.asm flash2.asm  -
Xlink_directive=flash.dir
```

Multi-core support specification

The multi-core suppor specification options are as follows.

- -Xmulti

-Xmulti

This option specifies the generation of a subprogram of a multi-core program.

[Specification format]

```
-Xmulti=type
```

- Interpretation when omitted
  A single-core program is generated.

[Detailed description]

- This option specifies the subprogram type when generating a subprogram of a multi-core program.
- If this option is specified for a target that is not a multi-core CPU, a warning will be output and this option will be ignored.
- The items that can be specified as *type* are shown below.
  An error will occur if any other item is specified.

pe*n*	Generates a program for core *n*. The value that can be specified for *n* is an integer in the range of 1 to the number of cores on the target CPU. An error will occur if any other value is specified.
cmn	Generate the common module. An error will occur if a "#pragma pic" directive is described. An error will occur if the data be allocated to a section with relocation attributes other than data, const, or sconst (e.g. sdata) is specified.

- When this option is specified, the following preprocessor macros are set automatically.

type	Preprocessor Macro to Define (Value is 1)
pe*n*	__MULTI_CORE__, MULTI_PE*n*__
cmn	__MULTI_CORE__, MULTI_CMN__

- When this option is specified, the section name is automatically converted during assembling (unless the section name is explicitly specified in the source).
  That is, the suffix corresponding to type (.pen or .cmn) is added after the default section name.
  In each object module file, the section defined in the file and the section of information indicating correspondence of *type* is output.
- Specifying this option affects the following options.
  - Implicitly enabled options

```
-Xno_startup, -Xno_stdlib, -Xno_romize, -Xrelinkable_object
```

- Ignored options

```
-l, -L, -Xstartup, -Xmap, -Xno_romize, -Xrompcrt, -Xrompsec_start, -Xrompsec_data,
-Xrompsec_text, -Xrompsec_only, -Xromize_check_off, -Xhex, -Xhex_only, -Xhex_format,
-Xhex_fill, -Xhex_section, -Xhex_block_size, -Xhex_offset, -Xhex_null, -Xhex_symtab,
-Xhex_rom_less
```

- To generate library files for each of the subprograms, specify this option at the same time as the -c option, and create an object module file.

Caution Do not create a library file for object module files created with different -Xmulti options specified.

[Example of use]

- To generate a program for core 1, describe as:

```
>cx -CF3515 -Xmulti=pe1 file_pe1_1.asm file_pe1_2.asm -ope1.lmf
```

Error output control

The error output control options are as follows.

- -Xerror_file

-Xerror_file

This option outputs error messages to a file.

[Specification format]

```
-Xerror_file=file
```

- Interpretation when omitted
  Error messages are output to only the standard error output.

[Detailed description]

- This option outputs error messages to the standard error output and file *file*.
- If *file* already exists, it will be overwritten.
- An error will occur if *file* is omitted.
- An error will occur if *file* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Individual Assemble Options] tab, [Output error message file] in the [Error Output] category
  - From the [Individual Assemble Options] tab, [Error message file output folder] in the [Error Output] category
  - From the [Individual Assemble Options] tab, [Error message file name] in the [Error Output] category

[Example of use]

- To output error messages to the standard error output and file "err", describe as:

```
>cx -CF3746 -Xerror_file=err main.asm
```

Warning message output control

The warning message output control options are as follows.

- -Xwarning
- -Xno_warning

-Xwarning

This option outputs the specified warning message.

[Specification format]

```
-Xwaning=num[,num]...
-Xwaning=num1-num2
```

- Interpretation when omitted
  Severe warning messages are output (same as when "-Xwarning_level=1" is specified).

[Detailed description]

- This option outputs the specified warning message.
- Specify the error numbers as *num*, *num1*, and *num2*.
  If the error number that does not exist, it will be ignored.
- An error will occur if *num*, *num1*, or *num2* is omitted.
- If *num1*-*num2* is specified, it is assumed that error numbers within the range have been specified.
- The error number specified by this option is the rightmost 5 digits of the 7-digit number following the "W".
  See "CubeSuite Message" for error numbers.
- This option is equivalent to the following property in CubeSuite.
  - From the [Individual Assemble Options] tab, [Displayed warning message] in the [Warning Message] category

[Example of use]

- To output warning message "W0566002", describe as:

```
>cx -CF3746 -Xwaning=66002 main.asm
```

-Xno_warning

This option suppresses outputting warning messages of the specified number.

[Specification format]

```
-Xno_waning=num[,num]...
-Xno_waning=num1-num2
```

- Interpretation when omitted
  Severe warning messages are output (same as when "-Xwarning_level=1" is specified).

[Detailed description]

- This option suppresses outputting warning messages of the specified number.
- Specify the error numbers as *num*, *num1*, and *num2*.
  If the error number that does not exist, it will be ignored.
- An error will occur if *num*, *num1*, or *num2* is omitted.
- If *num1*-*num2* is specified, it is assumed that error numbers within the range have been specified.
- The error number specified by this option is the rightmost 5 digits of the 7-digit number following the "W".
  See "CubeSuite Message" for error numbers.
- This option is equivalent to the following property in CubeSuite.
  - From the [Individual Assemble Options] tab, [Undisplayed warning message] in the [Warning Message] category

[Example of use]

- To suppress outputting warning message "W0566002", describe as:

```
>cx -CF3746 -Xno_waning=66002 main.asm
```

Command file specification

The command file specification option is as follows.

- @

@

This option specifies a command file.

[Specification format]

```
@file
```

- Interpretation when omitted
  Only the options and file names specified on the command line are recognized.

[Detailed description]

- This option handles *file* as a command file.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.
- See "(2) Startup from a command file" for details about a command file.

[Example of use]

- To handle "command" as a command file, describe as:

```
>cx @command
```

(3) Link options

The types and explanations for options of the link phase are shown below.

Table B-6. Link Options

Classification	Option	Description
Version/help display specification	-V	This option displays the version information of the cx.
	-h	This option displays the descriptions of the cx options.
Output file specification	-o	This option specifies the output file name.
	-Xtemp_path	This option specifies the temporary folder.
Source debugging control	-g	This option outputs information for source debugging.
Device specification	-C	This option specifies the target device.
	-Xdev_path	This option specifies the folder to search device files.
Library link control	-l	This option specifies the library file to be used during linking.
	-L	This option specifies the folder to search library files.
	-Xno_stdlib	This option suppresses linking the standard library.
Link directive file specification	-Xlink_directive	This option specifies the link directive file.
Force linking to continue	-Xforce_link	This option continues link processing when the internal ROM/RAM overflows.
Entry point address specification	-Xentry_address	This option specifies the entry point address.
Link map file output specification	-Xmap	This option outputs the link map file.
Generating object module file control	-Xrelinkable_object	This option generates the relocatable object module file.
Control checking for mixing register modes	-Xreg_mode	This option checks whether the specified register mode is mixed.
	-Xregmode_info	This option outputs detailed information when different register modes are mixed.
Specify checking for mixing of device common objects	-Xcommon	This option checks for mixing of the device common object module file to be generated and the device specified by the -C option.
sdata/sbss information output specification	-Xsdata_info	This option outputs information that can be used as a yardstick for the parameter of the -Xsdata option to the standard output.
2-pass mode link specification	-Xtwo_pass_link	This option performs linking in the 2-pass mode.
Relocation resolution error processing control	-Xignore_address_error	This option continues link processing if an illegality is found during relocation processing when linking.
Symbol multiple definition error output specification	-Xmultiple_symbol	This option outputs an error message for all multi-defined external symbols.

Classification	Option	Description
Link-time check suppress specification	-Xlink_check_off	This option suppresses checking when linking.
Filling value specification	-Xalign_fill	This option specifies the filling value of align holes.
Library file rescan specification	-Xrescan	This option rescans the library file specified by the -l option.
Debug information section output suppress specification	-Xstrip	This option generates the load module file from which the debug information, line number information, and global pointer table have been removed.
Boot-flash re-link function control	-Xflash_ext_table	This option specifies the start address value of the branch table for the boot-flash re-link function.
	-Xflash	This option generates the load module file for the flash area.
Non-ROMized load module file save specification	-Xlink_output	This option saves the load module file before ROMization processing.
Multi-core support specification	-Xmulti	This option specifies the generation of a subprogram of a multi-core program.
	-Xmulti_link	This option specifies the linking of multi-core subprograms.
Error output control	-Xerror_file	This option outputs error messages to a file.
Warning message output control	-Xwarning	This option outputs the specified warning message.
	-Xno_warning	This option suppresses outputting warning messages of the specified number.
Phase individual option specification	-Xlk_option	This option specifies the file to be linked.
Command file specification	@	This option specifies a command file.

Version/help display specification

The version/help display specification options are as follows.

- -V
- -h

-V

This option displays the version information of the cx.

[Specification format]

```
-V
```

- Interpretation when omitted
  Linking is performed without displaying the version information of the cx.

[Detailed description]

- This option outputs the version information of the cx to the standard error output.
  It does not execute linking.

[Example of use]

- To output the version information of the cx to the standard error output, describe as:

```
>cx -CF3746 -V
```

-h

This option displays the descriptions of the cx options.

[Specification format]

```
-help
```

- Interpretation when omitted
  The descriptions of the cx options are not displayed.

[Detailed description]

- This option outputs the descriptions of the cx options to the standard error output.
  It does not execute linking.

[Example of use]

- To output the descriptions of the cx options to the standard error output, describe as:

```
>cx -CF3746 -h
```

Output file specification

The output file specification options are as follows.

- -o
- -Xtemp_path

-o

This option specifies the output file name.

[Specification format]

```
-ofile
```

- Interpretation when omitted
  The file is output to the current folder.
  The output load module file name is "a.lmf".

[Detailed description]

- This option specifies the output file name as *file*.
- If *file* already exists, it will be overwritten.
- It is assumed that a load module file name has been specified as *file*.
- An error will occur if two or more files are output.
- An error will occur if *file* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Output folder] in the [Frequently Used Options(for Link)] category
  - From the [Common Options] tab, [Output file name] in the [Frequently Used Options(for Link)] category
  - From the [Link Options] tab, [Output folder] in the [Output File] category
  - From the [Link Options] tab, [Output file name] in the [Output File] category

[Example of use]

- To output the load module file with "sample.lmf" as the file name, describe as:

```
>cx -CF3746 -osample.lmf main.obj
```

-Xtemp_path

This option specifies the temporary folder.

[Specification format]

```
-Xtemp_path=path
```

- Interpretation when omitted
  The temporary folder is determined according to the following sequence.

(1) Folder specified by environmental variable TEMP

(2) Folder specified by environmental variable TMP

(3) Current folder

[Detailed description]

- This option specifies *path* as the folder for generating temporary files that are used internally.
- If *path* does not exist, a warning will be output and the temporary folder will be determined.

(1) Folder specified by environmental variable TEMP

(2) Folder specified by environmental variable TMP

(3) Current folder

- An error will occur if *path* is omitted.

[Example of use]

- To specify folder "D:\tmp" as the temporary folder, describe as:

```
>cx -CF3746 -Xtemp_path=D:\tmp main.obj
```

Source debugging control

The source debugging control option is as follows.

- -g

-g

This option outputs information for source debugging.

[Specification format]

```
-g
```

- Interpretation when omitted
  Information for source debugging will not be output.

[Detailed description]

- This option outputs information for source debugging to the output file.
- Source level debugging can be performed by specifying this option.

[Example of use]

- To output information for source debugging to the output file, describe as:

```
>cx -CF3746 -g main.obj
```

Device specification

The device specification options are as follows.

- -C
- -Xdev_path

-C

This option specifies the target device.

[Specification format]

```
-Cdevice
```

- Interpretation when omitted
  If the -Xcommon option is specified, the behavior is in accordance with that specification.
  Otherwise, an error will occur (except when specifying the -V, -h, or -P option).
  An error will occur when linking is performed.

[Detailed description]

- This option specifies the target device.
- See the user's manual of each device file about device types that can be specified as *device*.
- If *device* does not exist (the corresponding device file does not exist), an error will occur.
- An error will occur if *device* is omitted.
- This option cannot be omitted when linking is performed.

[Example of use]

- To specify uPD70F3746 as the target device, describe as:

```
>cx -CF3746 main.obj
```

-Xdev_path

This option specifies the folder to search device files.

[Specification format]

```
-Xdev_path=path
```

- Interpretation when omitted
  The device file is searched from the standard device file folder.

[Detailed description]

- This option searches a device file from folder *path*.
- If the folder as *path* does not exist, or if the device file specified by the -C option cannot be found, a warning will be output and the standard device file folder[Note] will be searched.
  If it cannot be found, an error will occur.

Note The device file is searched in the following order.

[V850E2V3]
1. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device\V850E2\Devicefile
2. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device\V850\Devicefile
3. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device_Custom\ Devicefile

[V850E/V850E2]
1. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device\V850\Devicefile
2. *CubeSuite install folder*\CubeSuite\CX\V*x.xx*\..\..\Device_Custom\ Devicefile

- An error will occur if *path* is omitted.

[Example of use]

- To search a device file from folder C:\NECTools32\dev, describe as:

```
>cx -CF3746 -Xdev_path=C:\NECTools32\dev main.obj
```

Library link control

The library link control options are as follows.

- -l
- -L
- -Xno_stdlib

-l

This option specifies the library file to be used during linking.

[Specification format]

```
-lstring[,string]...
```

- Interpretation when omitted
  Only the standard library, mathematical library, and standard startup routine are linked.
  When the -Xno_romize option is not specified, the ROMization area reservation code file is linked.

[Detailed description]

- This option specifies library file lib*string*.lib to be used during linking.
- When the cx resolves an unresolved external symbol reference after linking all object module files, the library file (lib*string*.lib) is referenced.
- An error will occur if *string* is omitted.
- If this option and the -L option are specified at the same time, the library file is searched from the folder specified by the -L option.
  If the -L option is not specified, the library file is searched from the standard folder[Note].
- If the library file specified by this option cannot be found, a message will not output and link processing will be continued.
- If two or more library files are specified, they are searched in the order which they are specified.
- The cx links the standard library (libc.lib) and standard startup routine (cstart.obj) as well as the library specified by this option automatically.
  When the -Xno_romize option is not specified, the ROMization area reservation code file (rompcrt.obj) is also linked automatically.
  Use the -Xno_stdlib and -Xno_startup option to suppress this.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Using libraries] in the [Frequently Used Options(for Link)] category
  - From the [Link Options] tab, [Using libraries] in the [Library] category
  - From the [Link Options] tab, [System libraries] in the [Library] category

Note *CubeSuite install folder*\Cubesuite\CX\V*x.xx*\lib\850e

[Example of use]

- To specify library file "libsmp.lib" to be used during linking, describe as:

```
>cx -CF3746 -lsmp main.obj
```

-L

This option specifies the folder to search library files.

[Specification format]

```
-Lpath[,path]...
```

- Interpretation when omitted
  The library file is searched from the standard library file folder.

[Detailed description]

- This option searches libraries from folder *path*, the standard folder[Note] in that order.
- If *path* does not exist, a warning will be output.
- An error will occur if *path* is omitted.
- This option is equivalent to the following property in CubeSuite.
    - From the [Common Options] tab, [Additional library paths] in the [Frequently Used Options(for Link)] category
    - From the [Link Options] tab, [Additional library paths] in the [Library] category
    - From the [Link Options] tab, [System library paths] in the [Library] category

Note *CubeSuite install folder*\Cubesuite\CX\V*x.xx*\lib\850e

[Example of use]

- To search libraries from folder "lib", the standard folder in that order, describe as:

```
>cx -CF3746 -Llib main.obj
```

-Xno_stdlib

This option suppresses linking the standard library.

[Specification format]

```
-Xno_stdlib
```

- Interpretation when omitted
  The standard library is linked.

[Detailed description]

- This option does not link the standard library.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Link standard library] in the [Library] category

[Example of use]

- Not to link the standard library, describe as:

```
>cx -CF3746 -Xno_stdlib main.obj
```

Link directive file specification

The link directive file specification option is as follows.

- -Xlink_directive

-Xlink_directive

This option specifies the link directive file.

[Specification format]

```
-Xlink_directive=file
```

- Interpretation when omitted
  The default link directive file is used.

[Detailed description]

- This option performs linking according to the link directive in link directive file *file*.
- ".dir" is recommended as the extension of *file*.
- An error will occur if *file* is omitted.
- If this option is specified more than once, the option specified last will be valid, and the option specified first will be ignored.
- See "CubeSuite Coding for CX Compiler" for details about the link directive file.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Using link directive file] in the [Input File] category

[Example of use]

- To perform linking according to the link directive in link directive file "link.dir", describe as:

```
>cx -CF3746 -Xlink_directive=link.dir main.obj
```

Force linking to continue

The option forces linking to continue is as follows.

- -Xforce_link

-Xforce_link

This option continues link processing when the internal ROM/RAM overflows.

[Specification format]

```
-Xforce_link
```

- Interpretation when omitted
  An error is output and link processing is completed when the internal ROM/RAM overflows.

[Detailed description]

- This option outputs a warning and continues link processing when the internal ROM/RAM overflows.
- The output file is not generated.
- The excess of the size is output to the standard error output.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Force linking against error] in the [Others] category

[Example of use]

- To output a warning and continue link processing when the internal ROM/RAM overflows, describe as:

```
>cx -CF3746 -Xforce_link main.obj
```

Entry point address specification

The entry point address specification option is as follows.

- -Xentry_address

-Xentry_address

This option specifies the entry point address.

[Specification format]

```
-Xentry_address=symbol
```

- Interpretation when omitted
  The entry point address value for the load module file to be generated is determined according to the following rules.
- If symbol "__start" exists, it is used.
- If "__start" does not exist, the start address of the text attribute section that is allocated to the lowest address area in the load module file to be generated is used.
- If the text attribute section does not exist, "0" is used.

[Detailed description]

- This option regards the value of symbol *symbol* as the entry point address value (it is used when hex conversion is performed) for the load module file to be generated.
- An error will occur if *symbol* does not exist.
- An error will occur if *symbol* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Entry symbol] in the [Others] category

[Example of use]

- To regard the value of symbol "__my_start" as the entry point address value for the load module file to be generated, describe as:

```
>cx -CF3746 -Xentry_address=__my_start main.obj
```

Link map file output specification

The link map file output specification option is as follows.

- -Xmap

-Xmap

This option outputs the link map file.

[Specification format]

```
-Xmap[=file]
```

- Interpretation when omitted
  The link map file is not output.

[Detailed description]

- This option outputs link map file *file*.
- The contents of the link map file are shown below.
  - Allocation of input sections which are included in the specified object module file to the memory space
  - Allocation of the output sections which composes the load module file to be generated by linking input sections to the memory space
  - Address information of symbols
- If *file* already exists, it will be overwritten.
- If only the file name is specified for *file*, the file is output to the same folder as the load module file under the specified file name.
- If *file* is omitted, the file is output to the same folder as the load module file under the load module file name with the extension replaced by ".map".
- The link map file to be output differs depending on whether the -Xno_romize option is specified as follows.
  - When the -Xno_romize option is specified
    The link map file is output after link processing.
  - When the -Xno_romize option is not specified
    The link map file is output after ROMization processing.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Output link map file] in the [Link Map] category
  - From the [Link Options] tab, [Output folder for link map file] in the [Link Map] category
  - From the [Link Options] tab, [Link map file name] in the [Link Map] category

[Example of use]

- To output link map file "smp1.map" after link processing, describe as:

```
>cx -CF3746 -Xmap=smp.map main.obj -Xno_romize
```

- To output link map file "smp2.map" after ROMization processing, describe as:

```
>cx -CF3746 -Xmap=smp.map main.obj
```

Generating object module file control

The generating object module file control option is as follows.

- -Xrelinkable_object

-Xrelinkable_object

This option generates the relocatable object module file.

[Specification format]

```
-Xrelinkable_object
```

- Interpretation when omitted
  The cx tries to generate the executable object module file.
  If an unresolved external reference remains after linking, an error is output and linking is stopped.
  At this time, the load module file is not generated.

[Detailed description]

- This option generates the relocatable object module file.
- If this option is specified, a message is not output and linking is completed normally if an unresolved external reference remains after linking.
- If the object module file generated by the cx is specified as the target for relocation, use this option to generate the target object module file for relocation.
- If this option is specified, only the types and attributes in the segment directive section in the link directive will be valid and the others are ignored.
- If this option is specified, any reserved symbol is not created.
- This option must be specified together with the -Xno_startup and -Xno_romize options.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Output relocatable object module file] in the [Output File] category

[Example of use]

- To generate the relocatable object module file, describe as:

```
>cx -CF3746 -Xrelinkable_object main.obj -Xno_startup -Xno_romize
```

Control checking for mixing register modes

The option controls checking for mixing register modes are as follows.

- -Xreg_mode
- -Xregmode_info

-Xreg_mode

This option checks whether the specified register mode is mixed.

[Specification format]

```
-Xreg_mode=mode
```

- Interpretation when omitted
  The check for mixing of 32 register modes is performed.

[Detailed description]

- This option checks whether register mode *mode* specified as the compile option or assemble option is mixed.
- The items that can be specified as *mode* are shown below.
  An error will occur if any other item is specified.

Register Mode (*mode*)	Working Registers	Registers for Register Variables
common	r10 to r14	r25 to r29
22	r10 to r14	r25 to r29
26	r10 to r16	r23 to r29
32	r10 to r19	r20 to r29

- If "common" is specified as *mode*, mix checking is not performed.
- An error will occur if *mode* is omitted.
- If there are object module files with different register modes, a warning will be output and link processing will be continued.

[Example of use]

- To perform the check for mixing of 22 register modes, describe as:

```
>cx -CF3746 -Xreg_mode=22 main.obj
```

-Xregmode_info

This option outputs detailed information when different register modes are mixed.

[Specification format]

```
-Xregmode_info
```

- Interpretation when omitted
  Detailed information is not output when different register modes are mixed.

[Detailed description]

- This option outputs detailed information and specifies the input object module file that causes a warning when different register modes are mixed for all input object module files.
- No information will be output if the register modes match.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Check register mode] in the [Others] category

[Example of use]

- To output detailed information when different register modes are mixed for all input object module files, describe as:

```
>cx -CF3746 -Xregmode_info file1.obj file2.obj
```

Specify checking for mixing of device common objects

The option specifies checking for mixing of device common objects is as follows.

- -Xcommon

-Xcommon

This option checks for mixing of the device common object module file to be generated and the device specified by the -C option.

[Specification format]

```
-Xcommon=series
```

- Interpretation when omitted
  Linking is performed in accordance with the specification of the -C option.

[Detailed description]

- This option checks whether common magic number *series* specified as the compile option or assemble option and the series number of the device specified by the -C option are mixed.
- The items that can be specified as *series* are shown below.
  An error will occur if any other item is specified.

v850e	It is possible to link to the model with the instruction set architecture superior to V850E (V850E, V850E2, or V850E2V3), which is specified as the target device.
v850e2	It is possible to link to the model with the instruction set architecture superior to V850E2 (V850E2 or V850E2V3), which is specified as the target device.
v850e2v3	V850E2V3 common It is possible to link to the model with the V850E2V3 instruction set architecture, which is specified as a target device. If the instruction set architecture of the target device is V850E2V3, we recommend specifying this item to get the best performance.

- An error will occur if *series* is omitted.

[Example of use]

- To check whether the common magic number and device "uPD70F3746" specified by the -C option are mixed, describe as:

```
>cx -CF3746 -Xcommon=v850e main.obj
```

sdata/sbss information output specification

The sdata/sbss information output specification option is as follows.

- -Xsdata_info

-Xsdata_info

This option outputs information that can be used as a yardstick for the parameter of the -Xsdata option to the standard output.

[Specification format]

```
-Xsdata_info
```

- Interpretation when omitted
  Information that can be used as a yardstick for the parameter of the -Xsdata option is not output to the standard output.

[Detailed description]

- This option outputs information that can be used as a yardstick for the parameter of the -Xsdata option (it specifies the maximum size of data allocated to the sdata/sbss section) to the standard output.
- When using the numerical value indicated by *OK*, data with a size less than that value is allocated to the sdata/sbss area.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Display GP information] in the [Others] category

[Example of use]

- To output information that can be used as a yardstick for the parameter of the -Xsdata option to the standard output, describe as:

```
>cx -CF3746 -Xsdata_info main.obj
```

2-pass mode link specification

The 2-pass mode link specification option is as follows.

- -Xtwo_pass_link

-Xtwo_pass_link

This option performs linking in the 2-pass mode.

[Specification format]

```
-Xtwo_path_link
```

- Interpretation when omitted
  Linking is performed in the 1-pass mode.

[Detailed description]

- This option performs linking in the 2-pass mode.
- The 2-pass mode is slower than the 1-pass mode, but it is able to process larger sized files.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Link in 2-pass mode] in the [Others] category

[Example of use]

- To perform linking in the 2-pass mode, describe as:

```
>cx -CF3746 -Xtwo_path_link main.obj
```

Relocation resolution error processing control

The relocation resolution error processing control option is as follows.

- -Xignore_address_error

-Xignore_address_error

This option continues link processing if an illegality is found during relocation processing when linking.

[Specification format]

```
-Xignore_address_error
```

- Interpretation when omitted
  An error occurs if an illegality is found during relocation processing when linking.

[Detailed description]

- If any of the following illegalities is found during relocation processing when linking, this option outputs a warning instead of an error and continues link processing.
  - The result of address calculation of an unresolved external reference is illegal
  - The relationship with the section to be allocated is illegal

  Specifically, it indicates one of the following:
  - There is no GP symbol for GP relative relocation (LOCAL / GLOBAL / EXTERN)
  - The result of PC22/26 relocation is a branch to an odd address
  - The result of PC relative relocation is outside the allowed bounds (symbol/no symbol)
  - The result of other than PC-relative relocation is outside the allowed bounds (symbol/no symbol)
- The value of address calculation judged as an illegality is not assigned to the unresolved external reference judged as an error and the original value remains.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Ignore illegal relocation] in the [Others] category

[Example of use]

- To output a warning and continue link processing if the result of address calculation of an unresolved external reference is illegal during relocation processing when linking, describe as:

```
>cx -CF3746 -Xignore_address_error main.obj
```

Symbol multiple definition error output specification

The symbol multiple definition error output specification option is as follows.

- -Xmultiple_symbol

-Xmultiple_symbol

This option outputs an error message for all multi-defined external symbols.

[Specification format]

```
-Xmultiple_symbol
```

- Interpretation when omitted
  An error message is output for the first multi-defined external symbol and link processing is stopped.

[Detailed description]

- This option outputs an error message for all multi-defined external symbols and file names and stops link processing.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Check all multi-defined symbols] in the [Others] category

[Example of use]

- To output an error message for all multi-defined external symbols and stop link processing, describe as:

```
>cx -CF3746 -Xmultiple_symbol main.obj
```

Link-time check suppress specification

The link-time check suppress specification option is as follows.

- -Xlink_check_off

-Xlink_check_off

This option suppresses checking when linking.

[Specification format]

```
-Xlink_check_off=string[,string]
```

- Interpretation when omitted
  When the external symbol is linked, a warning will be output and link processing will be continued if the size of the symbol is checked and the difference is detected.
  At this time, the symbol size of the file in which the symbol is defined is valid.
  When the undefined symbol is linked, a warning will be output and link processing will be continued if the size and alignment conditions of the symbol is checked and the difference is detected.
  If the application allocation overlaps the addresses of the internal ROM area, a warning will be output.

[Detailed description]

- This option suppresses checking when linking.
- The items that can be specified as *string* are shown below.
  An error will occur if any other item is specified.

symbol	Does not check the size and alignment conditions of the external symbol when it is linked.
undefined	Does not check the size and alignment conditions of the undefined external symbol when it is linked.
irom	Does not check the allocation to the internal ROM area. Therefore, if the application allocation overlaps the addresses of the internal ROM area, a warning will not be output.

- An error will occur if *string* is omitted.
- When the application is created in the ROM-less mode, it is assumed that "irom" has been specified.

Caution Checking the overflow of the internal ROM is not supported when the single-chip mode is selected.
Invalidate checking the overflow of the internal ROM by specifying the -Xlink_check_off=irom option and check the overflow on the link map.

- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Check illegality of external symbol] in the [Others] category
  - From the [Link Options] tab, [Check illegality of undefined external symbol] in the [Others] category
  - From the [Link Options] tab, [Check allocation for internal ROM area] in the [Others] category

[Example of use]

- Not to check the size and alignment conditions of the external symbol when it is linked, describe as:

```
>cx -CF3746 -Xlink_check_off=symbol main.obj
```

Filling value specification

The filling value specification option is as follows.

- -Xalign_fill

-Xalign_fill

This option specifies the filling value of align holes.

[Specification format]

```
-Xalign_fill=value
```

- Interpretation when omitted
  The filling value 0x0000 is used for align holes between sections of the load module file to be generated.

[Detailed description]

- This option specifies filling value *value* of align holes between sections of the load module file to be generated.
- The range that can be specified for *value* is 0x0000 to 0xFFFF.
  An abort error will occur if a value outside the range is specified.
- If the specified value is less than 4 digits, the higher digits will be filled with 0.
- An error will occur if *value* is omitted.
- When this option is specified, linking must be performed in the 2-pass mode by specifying the -Xtwo_path_link option.
  An error will occur if the -Xtwo_path_link option is not specified.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Filling value of holes] in the [Others] category

[Example of use]

- To specify 0xFFFF as the filling value of align holes between sections of the load module file to be generated, describe as:

```
>cx -CF3746 -Xalign_fill=0xFFFF -Xtwo_path_link main.obj
```

Library file rescan specification

The library file rescan specification option is as follows.

- -Xrescan

-Xrescan

This option rescans the library file specified by the -l option.

[Specification format]

```
-Xrescan
```

- Interpretation when omitted
  The library file specified by the -l option is not rescanned.

[Detailed description]

- This option rescans the library file specified by the -l option.
- When this option is specified, symbols that are unresolved through the link sequence of the library can be prevented.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Rescan library files] in the [Others] category

[Example of use]

- To rescan library file "libtest1.lib" and "libtest2.lib", describe as:

```
>cx -CF3746 -Xrescan main.obj -ltest1 -ltest2
```

Debug information section output suppress specification

The debug information section output suppress specification option is as follows.

- -Xstrip

-Xstrip

This option generates the load module file from which the debug information, line number information, and global pointer table have been removed.

[Specification format]

```
-Xstrip
```

- Interpretation when omitted
  If the debug information, line number information and global pointer table exist in the load module file to be generated, they are not removed.

[Detailed description]

- This option generates the load module file from which the debug information, line number information and global pointer table have been removed.
- This option is equivalent to the following property in CubeSuite.
  - From the [Link Options] tab, [Delete debug information] in the [Debug Information] category

[Example of use]

- To generate the load module file from which the debug information, line number information and global pointer table have been removed, describe as:

```
>cx -CF3746 -Xstrip main.obj
```

Boot-flash re-link function control

The boot-flash re-link function control options are as follows.

- -Xflash_ext_table
- -Xflash

-Xflash_ext_table

This option specifies the start address value of the branch table for the boot-flash re-link function.

[Specification format]

```
-Xflash_ext_table=address
```

- Interpretation when omitted
  The load module file for the boot-flash re-link function is not generated and normal link processing is performed.

[Detailed description]

- This option generates the load module file for the boot-flash re-link function using address *address* as the start address value of the branch table.
  See "B.1.6 Boot-flash re-link function" for details about the boot-flash re-link function.
- Specify generating the load module file for the boot area or flash area depending on whether the -Xflash option is specified.
  - When the load module file for the boot area is generated (when the -Xflash option is not specified)
    The branch to the flash area is processed.
    At this time, the process is the branch to the branch table that is created at the address specified by this option.
  - When the load module file for the flash area is generated (when the -Xflash option is specified)
    The branch table that has the branch instruction to the original branch destination is created at the address specified by this option.
- The range that can be specified for *address* is 0x00000000 to 0xFFFFFFFF.
  An abort error will occur if a value outside the range is specified.
- If the specified value is less than 8 digits, the higher digits will be filled with 0.
- If an odd value is specified as *address*, it is corrected to an even number, and then a warning will be output and processing will continued.
- An error will occur if *address* is omitted.
- *address* must be the same as the value that is used when the load module file for the boot area and flash area is generated.
  An operation fault will occur if a different value is specified.
  No error checking is done.
- *address* must be within the ROM for the flash area.
  No error checking is done because it is not possible to determine which area contains the specified address.
- By specifying this option, section ".ext_table" that has a size of "(maximum ID value [Note] + 1) * (entry size of branch table)" bytes and starts with *address* is automatically created when the load module file for the flash area is generated.
  Although this section does not require an alignment specification in the link directive file, you must leave enough space for alignment.

Note This is the value specified by the .ext_func quasi directive in the assembler source file.

- When the load module file for the boot area is generated, the load module file before ROMization processing must be saved.
  This is because this file is specified by the -Xflash option when the load module file for the flash area is generated. Consequently, if this option is not specified with the -Xflash option at the same time, it must be specified with the -Xlink_output option (if ROMization is performed) or the -Xno_romize option (if ROMization is not performed) at the same time.
- This option can not be specified together with the -Xrelinkable_object option.
  An operation fault will occur if the load module file that has been generated using the -Xrelinkable_object option is input.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Branch table address] in the [Flash Correspondence] category

[Example of use]

- To generate load module file for the boot area "boot.lmf" with 0x200 as the start address of the branch table, describe as:

```
>cx -CF3746 -Xflash_ext_table=0x200 -Xlink_output=boot.lmf boot.obj
```

-Xflash

This option generates the load module file for the flash area.

[Specification format]

```
-Xflash=file
```

- Interpretation when omitted
  When the boot-flash re-link function is used, the load module file for the boot area is generated.
  When the boot-flash re-link function is not used, normal link processing is performed.

[Detailed description]

- This option generates the load module file for the flash area when the boot-flash re-link function is used.
  At this time, symbol information of load module file for the boot area *file* is referred and link processing is performed.
- Specify the load module file for the boot area that is generated using the boot-flash re-link function as *file*.
  The load module file specified here must be the file before ROMization processing (generated with the -Xno_romize or -Xlink_output option specified).
- An error will occur if a nonexistent file is specified as *file*.
- An error will occur if *file* is omitted.
- This option must be specified together with the -Xflash_ext_table option.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Load module file type] in the [Flash Correspondence] category
  - From the [Common Options] tab, [Boot area load module file name] in the [Flash Correspondence] category

[Example of use]

- To generate load module file for the boot area "boot.lmf" with 0x200 as the start address of the branch table for the flash area, describe as:

```
>cx -CF3746 -Xflash_ext_table=0x200 -Xlink_output=boot.lmf boot1.obj boot2.obj -
Xlink_directive=boot.dir
```

  To create the branch table at address 0x200 and generate load module file for the flash area "flash.lmf", describe as:
  At this time, symbol information of load module file for the boot area "boot.lmf" is referred and link processing is performed.

```
>cx -CF3746 -Xflash_ext_table=0x200 -Xflash=boot.lmf -oflash.lmf flash1.obj flash2.obj  -
Xlink_directive=flash.dir
```

Non-ROMized load module file save specification

The non-ROMized load module file save specification option is as follows.

- -Xlink_output

-Xlink_output

This option saves the load module file before ROMization processing.

[Specification format]

```
-Xlink_output=file
```

- Interpretation when omitted
  The load module file is not saved before ROMization processing.

[Detailed description]

- This option saves the load module file with *file* as the file name before ROMization processing.
- The load module file before ROMization processing is normally deleted, but you can specify this option if you wish to save it.
- ".lmf" is recommended as the extension of *file*.
- An error will occur if *file* is omitted.
- If this option and the -Xno_romize option are specified at the same time, a warning will be output and this option will be ignored.
- This option is equivalent to the following property in CubeSuite.
  - From the [ROMize Options] tab, [Output Non-ROMized load module file] in the [Output File] category

[Example of use]

- To save the load module file with "sample.lmf" as the file name before ROMization processing, describe as:

```
>cx -CF3746 -Xlink_output=sample.lmf main.c
```

Multi-core support specification

The multi-core suppor specification options are as follows.

- -Xmulti
- -Xmulti_link

-Xmulti

This option specifies the generation of a subprogram of a multi-core program.

[Specification format]

```
-Xmulti=type
```

- Interpretation when omitted
  A single-core program is generated.

[Detailed description]

- This option specifies the subprogram type when generating a subprogram of a multi-core program.
- If this option is specified for a target that is not a multi-core CPU, a warning will be output and this option will be ignored.
- The items that can be specified as *type* are shown below.
  An error will occur if any other item is specified.

pe*n*	Generates a program for core *n*. The value that can be specified for *n* is an integer in the range of 1 to the number of cores on the target CPU. An error will occur if any other value is specified.
cmn	Generate the common module. When the input is a C source file, r0 relative instructions for acquiring variable access and function addresses are generated (handled in the same way as if a "#pragma nopic" directive is described at the top of the file). An error will occur if a "#pragma pic" directive is described. An error will occur if the data be allocated to a section with relocation attributes other than data, const, or sconst (e.g. sdata) is specified in the C source file and assembler source file.

- When this option is specified, code corresponding to *type* will be generated during compilation and assembling, symbol references that can be resolved within the scope of the subprogram will be resolved, and then linking will be performed, and a relinkable load module file will be created.
  At this time, the default startup routine and standard libraries will not be linked.
  Additionally, ROMization processing and generation of a hex file will not be performed.
- When this option is specified, the following preprocessor macros are set automatically.

Type	Preprocessor Macro to Define (Value is 1)
pe*n*	__MULTI_CORE__, MULTI_PE*n*__
cmn	__MULTI_CORE__, MULTI_CMN__

- When this option is specified, the section name is automatically converted during assembling (unless the section name is explicitly specified in the source).
  That is, the suffix corresponding to type (.pen or .cmn) is added after the default section name.
  In each object module file, the section defined in the file and the section of information indicating correspondence of *type* is output.
- Specifying this option affects the following options.
  - Implicitly enabled options

```
-Xno_startup, -Xno_stdlib, -Xno_romize, -Xrelinkable_object
```

  - Ignored options

```
-l, -L, -Xstartup, -Xmap, -Xno_romize, -Xrompcrt, -Xrompsec_start, -Xrompsec_data,
-Xrompsec_text, -Xrompsec_only, -Xromize_check_off, -Xhex, -Xhex_only, -Xhex_format,
-Xhex_fill, -Xhex_section, -Xhex_block_size, -Xhex_offset, -Xhex_null, -Xhex_symtab,
-Xhex_rom_less
```

- To generate library files for each of the subprograms, specify this option at the same time as the -c option, and create an object module file.

Caution Do not create a library file for object module files created with different -Xmulti options specified.

[Example of use]

- To generate a program for core 1, describe as:

```
>cx -CF3515 -Xmulti=pe1 file_pe1_1.c file_pe1_2.c -ope1.lmf
```

-Xmulti_link

This option specifies the linking of multi-core subprograms.

[Specification format]

```
-Xmulti_link
```

- Interpretation when omitted
  It will not be treated the linking of multi-core subprograms.

[Detailed description]

- This option links each generated subprograms when generating a multi-core program.
- When this option is specified, each subprogram, library file, and object module file are input, the addresses between the subprogram are resolved, and the multi-core program is generated.
  At this time, the startup routine and libraries supporting multi-core will be linked.
  Additionally, ROMization processing and generation of a hex file will be performed at the same time.
- An error will occur if a C source file or assembler source file is specified as input.
  If another type of file is specified, then if a necessary file is not specified, and there are remaining unresolved gp, ep, or tp relative symbol references, then an error will occur during linking.
- If this option is specified together with the -Xmulti option, an error will occur.

[Example of use]

- To link multi-core subprograms: pe1.lmf, pe2.lmf, and cmn.lmf, describe as:

```
>cx -CF3515 -Xmulti_link pe1.lmf pe2.lmf cmn.lmf -otarget.lmf -lmulti_lib
```

Error output control

The error output control option is as follows.

- -Xerror_file

-Xerror_file

This option outputs error messages to a file.

[Specification format]

```
-Xerror_file=file
```

- Interpretation when omitted
  Error messages are output to only the standard error output.

[Detailed description]

- This option outputs error messages to the standard error output and file *file*.
- If *file* already exists, it will be overwritten.
- An error will occur if *file* is omitted.

[Example of use]

- To output error messages to the standard error output and file "err", describe as:

```
>cx -CF3746 -Xerror_file=err main.obj
```

Warning message output control

The warning message output control options are as follows.

- -Xwarning
- -Xno_warning

-Xwarning

This option outputs the specified warning message.

[Specification format]

```
-Xwaning=num[,num]...
-Xwaning=num1-num2
```

- Interpretation when omitted
  Severe warning messages are output (same as when "-Xwarning_level=1" is specified).

[Detailed description]

- This option outputs the specified warning message.
- Specify the error numbers as *num*, *num1* and *num2*.
  If the error number that does not exist, it will be ignored.
- An error will occur if *num*, or *num1* and *num1* is omitted.
- If *num1-num2* is specified, it is assumed that error numbers within the range have been specified.
- The error number specified by this option is the rightmost 5 digits of the 7-digit number following the "W".
  See "CubeSuite Message" for error numbers.

[Example of use]

- To output warning message "W0566002", describe as:

```
>cx -CF3746 -Xwaning=66002 main.obj
```

-Xno_warning

This option suppresses outputting warning messages of the specified number.

[Specification format]

```
-Xno_waning=num[,num]...
-Xno_waning=num1-num2
```

- Interpretation when omitted
  Severe warning messages are output (same as when "-Xwarning_level=1" is specified).

[Detailed description]

- This option suppresses outputting warning messages of the specified number.
- Specify the error numbers as *num*, *num1* and *num2*.
  If the error number that does not exist, it will be ignored.
- An error will occur if *num*, or *num1* and *num1* is omitted.
- If *num1-num2* is specified, it is assumed that error numbers within the range have been specified.
- The error number specified by this option is the rightmost 5 digits of the 7-digit number following the "W".
  See "CubeSuite Message" for error numbers.

[Example of use]

- To suppress outputting warning message "W0566002", describe as:

```
>cx -CF3746 -Xno_waning=66002 main.obj
```

Phase individual option specification

The phase individual option specification option is as follows.

- -Xlk_option

-Xlk_option

This option specifies the file to be linked.

[Specification format]

```
-Xlk_option=file1[,file2]...
```

- Interpretation when omitted
  Only ".obj", ".lib", and ".lmf" are recognized as the link target.

[Detailed description]

- This option specifies so that a file that is not recognized as a link target (a file other than ".obj", ".lib", and ".lmf") is the link target.
- Specify the file to be linked as *file*.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.

[Example of use]

- To link file "link.test", describe as:

```
>cx -CF3746 -Xlk_option=link.test
```

Command file specification

The command file specification option is as follows.

- @

@

This option specifies a command file.

[Specification format]

```
@file
```

- Interpretation when omitted
  Only the options and file names specified on the command line are recognized.

[Detailed description]

- This option handles *file* as a command file.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.
- See "(2) Startup from a command file" for details about a command file.

[Example of use]

- To handle "command" as a command file, describe as:

```
>cx @command
```

(4) ROMize options

The types and explanations for options of the ROMize phase are shown below.

Table B-7. ROMize Options

Classification	Option	Description
ROMization processing suppress specification	-Xno_romize	This option suppresses ROMization processing.
ROMization area reservation code file specification	-Xrompcrt	This option specifies the ROMization area reservation code file.
rompsec section control	-Xrompsec_start	This option specifies the start address of the rompsec section.
	-Xrompsec_data	This option specifies the data section included in the rompsec section.
	-Xrompsec_text	This option specifies the text section included in the rompsec section.
	-Xrompsec_only	This option generates the load module file that has only the rompsec section.
Control checking for errors under ROMization	-Xromize_check_off	This option omits error checking under ROMization.

ROMization processing suppress specification

The ROMization processing suppress specification option is as follows.

- -Xno_romize

-Xno_romize

This option suppresses ROMization processing.

[Specification format]

```
-Xno_romize
```

- Interpretation when omitted
  ROMization processing is performed.
  The following are linked.
- ROMization area reservation code file (except when specifying the -Xno_stdlib option)
- Startup routine (cstart.obj) that contains copy function "_rcopy" call (except when specifying the -Xstartup or -Xno_startup option)

[Detailed description]

- This option suppresses ROMization processing.
  The ROMization area reservation code file is not linked.
  The startup routine (cstartN.obj) that does not contain copy function "_rcopy" call is linked (except when specifying the -Xstartup or -Xno_startup option).
- ROMization is a process whereby data that will be expanded in RAM is copied to ROM, and a routine is added to copy this data from ROM into RAM when the application first starts.
- If this option is specified in the application that does not have data with a default value, the code can be reduced.
- This option is equivalent to the following property in CubeSuite.
  - From the [ROMize Options] tab, [Output ROMized load module file] in the [Output File] category

[Example of use]

- To suppress ROMization processing, describe as:

```
>cx -CF3746 -Xno_romize main.c
```

ROMization area reservation code file specification

The ROMization area reservation code file specification option is as follows.

- -Xrompcrt

-Xrompcrt

This option specifies the ROMization area reservation code file.

[Specification format]

```
-Xrompcrt=file
```

- Interpretation when omitted
  When the -Xno_romize option is not specified, the standard ROMization area reservation code file (rompcrt.obj) is linked at the end of input files.

[Detailed description]

- This option links *file* instead of the standard ROMization area reservation code file at the end of input files.
- Specify the object module file as *file*.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.
- If this option is specified together with the -Xno_romize option, this option will be invalid.
- This option is equivalent to the following property in CubeSuite.
  - From the [ROMize Options] tab, [ROMization area reservation code file name] in the [Input File] category

[Example of use]

- To link rompack.obj instead of the standard ROMization area reservation code file at the end of input files, describe as:

```
>cx -CF3746 -Xrompcrt=rompack.obj main.c
```

rompsec section control

The rompsec section control options are as follows.

- -Xrompsec_start
- -Xrompsec_data
- -Xrompsec_text
- -Xrompsec_only
- -Xromize_check_off

-Xrompsec_start

This option specifies the start address of the rompsec section.

[Specification format]

```
-Xrompsec_start=label
```

- Interpretation when omitted
  The value of label "__S_romp" is regarded as the start address of the rompsec section to be created.

[Detailed description]

- This option performs hex output only for the specified section.
- The value of label *label* is regarded as the start address of the rompsec section to be created.
- If *label* is not in the load module file, or if this option is specified more than once, the option specified last will be valid, and the option specified first will be ignored.
- An error will occur if *label* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [ROMize Options] tab, [Start symbol of rompsec section] in the [Section] category

[Example of use]

- To regard the value of label "romp_start" as the start address of the rompsec section to be created, describe as:

```
>cx -CF3746 -Xrompsec_start=romp_start main.c
```

-Xrompsec_data

This option specifies the data section included in the rompsec section.

[Specification format]

```
-Xrompsec_data=section[,section]...
```

- Interpretation when omitted
  All sections with the data or sdata attribute and the sections allocated to the internal instruction RAM are included in the rompsec section.

[Detailed description]

- This option specifies the data section included in the rompsec section.
- The contents of section *section* and the corresponding address and size information are included in the rompsec section.
- This option is related to the section with the data or sdata attribute.
- If *section* does not exist in the load module file, an error will be output and processing will be stopped.
- *section* cannot include a space.
- An error will occur if *section* is omitted.
- If this option is specified more than once, each data section will be included in the rompsec section in the order which they are specified.
- This option is equivalent to the following property in CubeSuite.
  - From the [ROMize Options] tab, [Data sections included in rompsec section] in the [Section] category

[Example of use]

- To include section "data1" and "data2" in the rompsec section, describe as:

```
>cx -CF3746 -Xrompsec_data=data1,data2 main.c
```

-Xrompsec_text

This option specifies the text section included in the rompsec section.

[Specification format]

```
-Xrompsec_text=section[,section]...
```

- Interpretation when omitted
  Each section allocated to the internal instruction RAM is included in the rompsec section.

[Detailed description]

- This option specifies the text section included in the rompsec section.
- The contents of section *section* and the corresponding address and size information are included in the rompsec section.
- This option is related to the section with the text or const attribute.
- The section that can be specified as *section* is the section with the text or const attribute.
  If the section that has any other attribute is specified, a warning will be output and processing will be stopped.
- If *section* does not exist in the load module file, an error will be output and processing will be stopped.
- *section* cannot include a space.
- An error will occur if *section* is omitted.
- If this option is specified more than once, the section will be included in the rompsec section in the order which they are specified.
- If the particular section is specified by using this option for the input file linked specifying the device file with the internal instruction RAM, the section that is not specified and is allocated to internal instruction RAM will not be included in the rompsec section and also be deleted from the output file.
- This option is equivalent to the following property in CubeSuite.
  - From the [ROMize Options] tab, [Text sections included in rompsec section] in the [Section] category

[Example of use]

- To include section "text1" and "text2" in the rompsec section, describe as:

```
>cx -CF3746 -Xrompsec_text=text1,text2 main.c
```

-Xrompsec_only

This option generates the load module file that has only the rompsec section.

[Specification format]

```
-Xrompsec_only
```

- Interpretation when omitted
  The section with the text attribute is included in the load module file to be generated.

[Detailed description]

- This option generates the load module file that has only the rompsec section; no section with the text attribute is included in the file.
- This option is equivalent to the following property in CubeSuite.
  - From the [ROMize Options] tab, [Generate load module file has rompsec section only] in the [Section] category

[Example of use]

- To generate the load module file that has only the rompsec section, describe as:

```
>cx -CF3746 -Xrompsec_only main.c
```

Control checking for errors under ROMization

The option controls checking for errors under ROMization is as follows.

- -Xromize_check_off

-Xromize_check_off

This option omits error checking under ROMization.

[Specification format]

```
-Xromize_check_off=string[,string]
```

- Interpretation when omitted
  A peripheral allocation error of the internal ROM is not checked for the rompsec section.
  The duplicate address of the input file and output file is checked.

[Detailed description]

- This option omits error checking under ROMization.
- The items that can be specified as *string* are shown below.
  An error will occur if any other item is specified.

rom_less	Does not check a peripheral allocation error of the internal ROM for the rompsec section. Specify this item when the ROM-less mode is used. Checking the overflow of the internal ROM is not supported when 1 is selected as the single-chip mode. Disable checking the overflow of the internal ROM by specifying this item.
address	Does not check the duplicate addresses of the input file and output file.

- An error will occur if *string* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [ROMize Options] tab, [Ignore ROMization error] in the [Others] category

[Example of use]

- Not to check a peripheral allocation error of the internal ROM for the rompsec section, describe as:

```
>cx -CF3746 -Xromize_check_off=rom_less main.c
```

(5) Hex output options

The types and explanations for options of the hex output phase are shown below.

Table B-8. Hex Output Options

Classification	Option	Description
Hex file output specification	-Xhex	This option specifies the hex file name.
	-Xhex_only	This option executes only hex output.
Hex conversion control	-Xhex_format	This option specifies the format of the hex file to be output.
	-Xhex_fill	This option specifies fill processing of the hex file.
	-Xhex_section	This option converts the codes in the specified section in hex format and outputs them.
	-Xhex_block_size	This option specifies the maximum length of the block.
	-Xhex_offset	This option specifies the offset of the address to be output.
	-Xhex_null	This option generates as many null characters as the size of the section of data without an initial value.
	-Xhex_symtab	This option converts the symbol table and outputs it.
	-Xhex_rom_less	This option does not use the information of the internal ROM area when the hex file is filled.

Hex file output specification

The hex file output specification options are as follows.

- -Xhex
- -Xhex_only

-Xhex

This option specifies the hex file name.

[Specification format]

```
-Xhex=file
```

- Interpretation when omitted
  The hex file is output to the same folder as the load module file under the load module file name with the extension replaced by ".hex" (the load module file name that ".hex" is added if the extension of the file name is not ".lmf").

[Detailed description]

- This option outputs the hex file by the name of *file* after linking.
- If only the file name is specified for *file*, the file is output to the same folder as the load module file under the specified file name.
- An error will occur if *file* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Hex Output Options] tab, [Output folder for hex file] in the [Output File] category
  - From the [Hex Output Options] tab, [Hex file name] in the [Output File] category

[Example of use]

- To output hex file "sample.hex" after linking, describe as:

```
>cx -CF3746 -Xhex=sample.hex main.c
```

-Xhex_only

This option executes only hex output.

[Specification format]

```
-Xhex_only[=file]
```

- Interpretation when omitted
  Processing is performed in the usual order of linking, ROMization, and hex output.

[Detailed description]

- This option generates the hex file from the load module file specified as the input file and outputs it by the name of *file*.
- If only the file name is specified for *file*, the file is output to the same folder as the load module file under the specified file name.
- If *file* is omitted, the file is output to the same folder as the load module file under the load module file name with the extension replaced by ".map".
- Use this option when only hex output is executed after generating the load module file.
  An error will occur if a load module file is not specified as the input file.

[Example of use]

- To generate hex file "sample.hex" from load module file "a.lmf", describe as:

```
>cx -CF3746 -Xhex_only=sample.hex a.lmf
```

Hex conversion control

The hex conversion control options are as follows.

- -Xhex_format
- -Xhex_fill
- -Xhex_section
- -Xhex_block_size
- -Xhex_offset
- -Xhex_null
- -Xhex_symtab
- -Xhex_rom_less

-Xhex_format

This option specifies the format of the hex file to be output.

[Specification format]

```
-Xhex_format=format
```

- Interpretation when omitted
  The format of the hex file to be output is regarded as the Intel expanded hex format (32-bit address) (It is the same result as when the -Xhex_format=i option is specified).

[Detailed description]

- This option specifies the format of the hex file to be output.
- The items that can be specified as *format* are shown below.

I	Intel expanded hex format (up to 1 MB)
i	Intel expanded hex format (32-bit address) (up to 4 GB)
S	Motorola S type hex format (standard address) (up to 16 MB)
s	Motorola S type hex format (32-bit address) (up to 4 GB)
T	Expanded Tektronix hex format (up to 4 GB)

- An error will occur if *format* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Common Options] tab, [Hex file format] in the [Frequently Used Options(for Hex Output)] category
  - From the [Hex Output Options] tab, [Hex file format] in the [Hex Format] category

[Example of use]

- To regard the format of the hex file to be output as the Motorola S type hex format (standard address), describe as:

```
>cx -CF3746 -Xhex_format=S main.c
```

-Xhex_fill

This option specifies fill processing of the hex file.

[Specification format]

```
-Xhex_fill
-Xhex_fill=value
-Xhex_fill=value,start,size
-Xhex_fill=start,size
```

- Interpretation when omitted
  Fill processing is not performed.

[Detailed description]

- This option converts all codes in the area from address *start* to size *size* into hex format and outputs them.
  The unused area in the specified area is filled with *value*.
- The range that can be specified for *value* is 0x00 to 0xFFFF.
- Specify *value* in hexadecimal.
- *value* can be specified in 1-byte or 2-byte units.
  If the specified value is less than 2 or 4 digits, the higher bits will be filled with 0.
- If *value* is omitted, it is assumed that 0xFF has been specified.
- The range that can be specified for *start* is 0x00 to 0xFFFFFFFF.
  Even if *start* is within the above range, an error will occur if the section does not exist.
- The range that can be specified for *size* is 0x01 to 0x100000000.
- Specify *start* and *size* in hexadecimal.
- If *start* and *size* are omitted, all the codes in the internal ROM area defined by the device file are converted in hex format and output.
- This option can not be specified together with the -Xhex_format=T option.
- When the information of the internal ROM area defined by the device file is not used, specify this option together with the -Xhex_rom_less option.
  In this case, parameter *start* and *size* of this option must be specified.
- This option is equivalent to the following property in CubeSuite.
  - From the [Hex Output Options] tab, [Specify converted address range] in the [Hex Format] category
  - From the [Hex Output Options] tab, [Filling value] in the [Hex Format] category
  - From the [Hex Output Options] tab, [Start address] in the [Hex Format] category
  - From the [Hex Output Options] tab, [Size] in the [Hex Format] category

[Example of use]

- To convert all codes in the area from address 0x1000 to size 0x2000 into hex format and outputs them, describe as:
  The unused area in the area is filled with 0x55.

```
>cx -CF3746 -Xhex_fill=0x55,0x1000,0x2000 main.c
```

-Xhex_section

This option converts the codes in the specified section in hex format and outputs them.

[Specification format]

```
-Xhex_section=section[,section]...
```

- Interpretation when omitted
  All sections which have the section type other than NOBITS and section attribute A are converted in hex format and output them.

[Detailed description]

- This option converts the codes in section *section* in hex format and outputs them.
- An error will occur if *section* does not exist.
- An error will occur if *section* is omitted.
- This option can not be specified together with the -Xhex_fill option.
- This option is equivalent to the following property in CubeSuite.
  - From the [Hex Output Options] tab, [Converted sections] in the [Hex Format] category

[Example of use]

- To convert the codes in section "sec" in hex format and outputs them, describe as:

```
>cx -CF3746 -Xhex_section=sec main.c
```

-Xhex_block_size

This option specifies the maximum length of the block.

[Specification format]

```
-Xhex_block_size=num
```

- Interpretation when omitted
  The default value defined for each hex format is regards as the maximum block length.

[Detailed description]

- This option regards the value specified for *num* as the maximum block length (or, in the case of the Intel expanded hex format or Motorola S type hex format, the number of bytes of the code indicated in one data record).
- The range that can be specified for *num* differs depending on the hex format.
  The range that can be specified for *num* are shown below for each hex format.
  If the specified value is less than the minimum value, a warning will be output and it is corrected to the default value.
  If the specified value exceeds the maximum value, a warning will be output and it is corrected to the maximum value.
  An error will occur if 0 is specified.

Hex Format	Range of *num*	Default Value
Intel expanded	1 to 255 (0x01 to 0xFF)	32 (0x20)
Intel expanded (32-bit address)	1 to 255 (0x01 to 0xFF)	32 (0x20)
Motorola S type (standard address)	1 to 251 (0x01 to 0xFB)	80 (0x50)
Motorola S type (32-bit address)	1 to 250 (0x01 to 0xFA)	80 (0x50)
Extended Tektronix	16 to 255 (0x10 to 0xFF)	255 (0xFF)

- An error will occur if *num* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Hex Output Options] tab, [Specify maximum length of block/record] in the [Hex Format] category
  - From the [Hex Output Options] tab, [Maximum length of block/record] in the [Hex Format] category

[Example of use]

- To specify 255 as the maximum length of the block, describe as:

```
>cx -CF3746 -Xhex_block_size=255 main.c
```

-Xhex_offset

This option specifies the offset of the address to be output.

[Specification format]

```
-Xhex_offset=num
```

- Interpretation when omitted
  It is regarded that the address to be output does not have the offset.
  The address is output from address 0.

[Detailed description]

- This option outputs the original address, offset by *num*.
- The range that can be specified for *num* is 0x0 to 0xFFFFFFFE.
  An abort error will occur if a value outside the range is specified.
- An error will occur if *num* is omitted.
- This option is equivalent to the following property in CubeSuite.
  - From the [Hex Output Options] tab, [Specify offset of output address] in the [Hex Format] category
  - From the [Hex Output Options] tab, [Offset of output address] in the [Hex Format] category

[Example of use]

- To regard the offset of the address to be output as 0x10000, describe as:

```
>cx -CF3746 -Xhex_offset=0x10000 main.c
```

-Xhex_null

This option generates as many null characters as the size of the section of data without an initial value.

[Specification format]

```
-Xhex_null
```

- Interpretation when omitted
  All codes are converted in hex format, and the unused area is filled with 0xFFFF and output.

[Detailed description]

- This option generates as many null characters (\0) as the size of the section with the section type NOBITS and section attribute A (section for data for which no initial value is specified, such as the .bss and .sbss section).
- This option can not be specified together with the -Xhex_fill option.
- This option is equivalent to the following property in CubeSuite.
  - From the [Hex Output Options] tab, [Initialize section of data without initial value to zero] in the [Hex Format] category

[Example of use]

- To generate as many null characters as the size of the section with the section type NOBITS and section attribute A, describe as:

```
>cx -CF3746 -Xhex_null main.c
```

-Xhex_symtab

This option converts the symbol table and outputs it.

[Specification format]

```
-Xhex_symtab=string
```

- Interpretation when omitted
  The symbol table is not output.

[Detailed description]

- This option converts the symbol table and outputs it.
- The items that can be specified as *string* are shown below.
  An error will occur if any other item is specified.

global	Converts only global symbols.
all	Converts local symbols as well.

- An error will occur if *string* is omitted.
- This option is valid only when the -Xhex_format=T option is specified.
- This option can not be specified together with the -Xhex_fill option.
- This option is equivalent to the following property in CubeSuite.
  - From the [Hex Output Options] tab, [Convert symbol table] in the [Symbol Table] category

[Example of use]

- To convert the symbol table and outputs it, describe as:

```
>cx -CF3746 -Xhex_symtab -Xhex_format=T main.c
```

-Xhex_rom_less

This option does not use the information of the internal ROM area when the hex file is filled.

[Specification format]

```
-Xhex_rom_less
```

- Interpretation when omitted
  When this option is omitted and the -Xhex_fill option is specified and parameter *start* and *size* are omitted, the internal ROM area defined by the device file is filled.

[Detailed description]

- This option does not use the information of the internal ROM area defined by the device file when the hex file is filled.
- Specify this option when the -Xhex_fill option is specified and the information of the internal ROM area defined by the device file is not used.
- This option must be specified together with the -Xhex_fill option.
  Besides, parameter *start* and *size* of the -Xhex_fill option must be specified.
  An error will occur if the -Xhex_fill option and parameter *start* and *size* is omitted.
- A warning will not be output even if parameter *start* and *size* of the -Xhex_fill option exceeds the internal ROM area.
- This option is equivalent to the following property in CubeSuite.
  - From the [Hex Output Options] tab, [Warn internal ROM overflow] in the [Others] category

[Example of use]

- Not to use the information of the internal ROM area defined by the device file when the hex file is filled, describe as:

```
>cx -CF3746 -Xhex_rom_less -Xhex_fill=0xff,0x00,1000 main.c
```

(6) Specifying multiple options

This section describes the operation when two or more options are specified at the same time.

(a) Priority

- The following options disable other options.

-V/-h	All options will be invalid.
-P	Since execution is terminated at preprocessing, options related to the following processing will be invalid.
-S	Since execution is terminated at code generation processing, options related to the following processing will be invalid.
-c	Since execution is terminated at assemble processing, options related to the following processing will be invalid.
-Xswitch	If an item other than table is specified as the parameter, the -Xword_case option will be invalid.
-Xno_romize	Other ROMize options and the -Xlink_output option will be invalid.
-Xhex_only	Options other than Hex output options will be invalid. Among hex output options, -Xhex will be invalid.
-Xhex_format	If "T" is specified as the parameter, the -Xhex_fill option will be invalid.
-Xhex_fill	The -Xhex_section, -Xhex_symtab, and -Xhex_null options will be invalid.

- If options are specified by the following combinations, the option specified last will be valid with displaying a warning.
  - -P, -S, -c
  - -D, -U (When their symbol names are same.)
  - -Onothing, -Odefault, -Osize, -Ospeed
  - -Xstartup, -Xno_startup

Depending on the order of specified options, the following options will be invalid.

- -O*item*[Note] that is specified before -Onothing, -Odefault, or -Osize
- -O*item*[Note] or -Xpro_epi_runtime that is specified before -Ospeed

Note -O*item*: -Ounroll, -Oinline, -Odelete_static_func, -Opipeline

(b) Dependencies

The behavior of the following options varies depending on what other options are specified.

-C	An error will occur normally if the -C option is not specified. However, an error will not occur if any of the -Xcommon, -V, -h, or -P options is specified.
-Xcommon	The behavior may differ, if the -C option is specified at the same time. See "-Xcommon" for details.
-Xpreprocess	This option will be invalid if the -P option is not specified at the same time. At this time, a warning will not be output.
-Xalign_fill	An error will occur if the -Xtwo_pass_link option is not specified at the same time.
-Xhex_fll	If the -Xhex_rom_less option is specified at the same time, parameters *start* and *size* must be specified. An error will occur if these parameters are not specified.
-Xflash_ext_table	An error will occur if the any of -Xflash, -Xlink_output, -Xno_romize options (one or more) is not specified at the same time.
-Xflash	An error will occur if the -Xflash_ext_table option is not specified at the same time.
-Xsfg	An error will occur if the -Xcube_suite_info option is not specified at the same time.
-Xsfg_opt	This option will be invalid if the -Xsfg option is not specified at the same time.
-Xsfg_size_tidata -Xsfg_size_tidata_byte -Xsfg_size_sidata -Xsfg_size_sedata -Xsfg_size_sdata	This option will be invalid if the -Xsfg_opt option is not specified at the same time.
-o	If the -P, -S, or -c option is specified at the same time, the preprocessed file, assembler source file, or object module file
-g	If the -O option is specified at the same time, debug information may not be correct.

B.1.4 Symbol information file

The symbol information file is a text-format file that allocation information for a variable (global variable, static variable in a file, and static variable in a function) and function defined in a C source file is described.

(1) Function related to symbol information file

The cx provides the following two functions about the symbol information file.

- Referencing the symbol information file
  It is possible to specify the allocation of variables from external files (change the allocated section) without editing the C source file by referencing the symbol information file when compiling.
  The symbol information file to be referred can be edited and created by the user and it can be also generated by using the cx.

- Generating the symbol information file
  The symbol information file that allocation information for a variable and function defined in the C source file is described is generated automatically.
  You can edit the generated symbol information file, if necessary.

Caution If the symbol information file is referred when compiling, the cx will only use the variable allocation information.
If a symbol information file is generated automatically using the cx, function allocation information will also be output, but it is not possible to specify function allocation using this information.

(2) Information output to symbol information file

If a symbol information file is generated automatically using the cx, the optimum allocation information is output for the variables and functions defined in the C source file.

- The following are output: global variables; static variables in the file; static variables in functions; and functions.
- Information about the number of references and size of variables and functions, and the like is output in the order starting from highest use frequency.
- The optimum allocation information for each section can be output by specifying the -Xsfg_opt option.
  Variables are allocated so that they can be allocated within the sizes of the .tidata.byte, .tidata.word, .sidata, .sedata, and .sdata sections, in order starting from highest use frequency.
  The size of each section can be specified by the -Xsfg_size_tidata, -Xsfg_size_tidata_byte, -Xsfg_size_sidata, -Xsfg_size_sedata, and -Xsfg_size_sdata options.

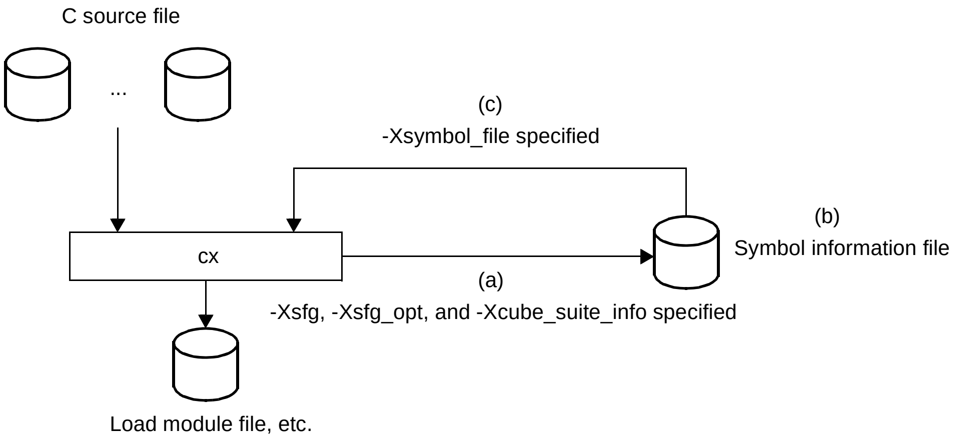
Remarks 1. The frequency of variable use is calculated as the number of references per byte from the number of references at the assembler source level, and the size (in bytes).
The frequency of function use is calculated as the number of references per byte from the number of references in the C source file, and the size (in bytes).

2. See "Information file output control" for details about each option.

(3) How to use the symbol information file

You must execute the cx command two times to generate a symbol information file using the cx and refer that file. On the first execution, a symbol information file is generated, and on the second execution, the symbol information file is referred.

Figure B-3. Flow of Processing Using Symbol Information File



The method for manipulating on the command line is shown below.

Remark See "2.13 Allocate Variables to Optimum Section" for the method manipulating in CubeSuite.

(a) Generating the symbol information file

Execute the cx command with the -Xsfg option to generate the symbol information file.

Example Symbol information file "symbol.sfg" is generated.

```
>cx -CF3746 -Xsfg=symbol.sfg -Xsfg_opt -Xcube_suite_info=info.cref file.c
```

Cautions 1. Specify the -Xsfg_opt option to output optimum allocation information for variables at the section level.

2. The symbol information file is generated based on the static analysis result of the C source file.
Therefore, when the -Xsfg option is specified, you need to specify the -Xcube_suite_info option at the same time.

3. A symbol information file cannot be generated if linking is unsuccessful, because it refers link-time information.

Remark The size of each section can be specified by the -Xsfg_size_tidata, -Xsfg_size_tidata_byte, -Xsfg_size_sidata, -Xsfg_size_sedata, and -Xsfg_size_sdata options.
See "Information file output control" for details about each option.

(b) Editing the symbol information file

The symbol information file is a text-format. Therefore, you can change the allocation information (change the allocated sections) by editing the file with a text editor or the like.
Edit the symbol information file, if necessary.

Remark See "3.3 Symbol Information File" for the format of the symbol information file.

(c) Referencing the symbol information file

Specify the generated symbol information file using the -Xsymbol_file option and then execute the cx command again. Compiling is performed according to the contents of the specified symbol information file.

Example Compiling is performed according to the contents of symbol information file "symbol.sfg".

```
>cx -CF3746 -Xsymbol_file=symbol.sfg file.c
```

B.1.5 Optimization function

This section explains the optimization function that the cx executes.
The target of optimization is a C source file.

(1) Outline

The cx performs optimization for the following two purposes.

- Faster execution speed (speeding code generation)
- Reduction of the object size (reducing the ROM/RAM capacity that the generated codes use)

Although most optimization items improve both of the above, some optimizations improve one at the expense of the other (e.g. it might increase execution speed while increasing ROM size).
Additionally, some optimizations will complicate the correspondence between C source lines and machine-language instructions. This will make debugging harder, because it could prevent break points from being set, or cause a variable's value to be referred or set in a location different from the location in the C source.
Therefore, the cx provides the following four optimization levels.

Table B-9. Optimization Level

Optimization Level	Description
Debug precedence	Regards debugging as important and suppresses all optimization including default optimization.
Default	Performs optimization that debugging is not affected (optimization of expressions and register allocation, and the like).
Object size precedence	Regards reducing the ROM/RAM capacity as important and performs the maximum optimization that is effective for general programs.
Execution speed precedence	Regards shortening the execution speed as important and performs the maximum optimization that is effective for general programs.

Optimization items which the cx executes are shown below.

Table B-10. Optimization Item

Item	Description
Optimization of expressions	Performs calculation of constants and deformation of expressions.
Optimization of control flow	Rearranges instructions to reduce branch instructions.
Propagation of copy	Replaces the referred variable with the value stored in the variable or other variable reference.
Recognition of common expressions	Holds the calculated value and reuses when same calculation is performed repeatedly.
Deletion of unnecessary instructions	Deletes the operation and assignment which result is not used.
Loop expansion	Expands a loop such a "for" or "while" for the number of times specified.
Loop optimization	Deletes a loop that is executed 0 or 1 time and performs optimization of expressions including a loop control variable.
Moving loop invariants	Calculations whose values do not change within a loop are moved outside the loop.

Item	Description
Inline expansion for functions	Expands a function at the location calling it. The calling and called functions must be defined within the same source file. If inter-file optimization is also specified, however, they can be defined in different source files.
Deleting static variables in the unused file	Deletes static variables in the unused file. Does not delete static variables in the unused function.
Deleting unused static functions	Deletes unused static functions.
Inline expansion for standard library functions	Performs inline expansion of standard library functions "strcpy", "strcmp", "memcpy", and "memset" calls.
Prologue/epilogue processing of a function based on runtime library calls	Instead of generating instruction rows to perform function prologue/epilogue processing, generates instructions to call the runtime library to perform equivalent processing. Although this reduces the code size, it will make execution somewhat slower.
Pipeline optimization [V850E2V3]	Reorders instructions at the machine-language level in order to use the pipeline effectively, and get the best performance from the V850E2V3 architecture.

Remark See "Optimization specification" for optimization options.

(2) Effects of optimization on debugging

Note with caution that optimization can have the following kinds of effects when debugging.

- As a result of deformation of an expression by optimization (propagation of copy and recognition of common part expression), "variable reference" does not take place where the read/write event of a variable appears in the C source program, and the event may not occur as expected by the user.
- When a statement has been made common, deleted, or rearranged, step execution and breakpoints may not be set as intended by the user.
- The live range of a variable (range in which the variable can be referenced in the program) and position of a variable (position on a register or memory) may be changed.
- Breakpoints cannot be set for statements that have been deleted.
- Transfer, splitting, or merging of statements may have rearranged the sequence of executable instructions[Note], so that lines between the lines which have been rearranged may be handled as a single line for which break points and step execution may no longer be possible.

Note The address of an executable instruction within a line of source code may be smaller than the address of an executable instruction in a previous line or may be greater than the address of an executable instruction in a subsequent line.

- If the sequence of executable instructions for if-else statements has been rearranged or if loop expansion has caused a sequence of executable instructions to be rearranged, step execution may no longer be possible.
- The entire function is regarded as the valid range (scope) for all automatic variables.
  However, if the variables have been allocated to registers, they can be deleted or otherwise rendered invisible by optimization even when they are within the scope.
  This can occur when the variables are being used as "local variables" within the scope or have been assigned as local variables as a result of optimization.

Example

```
void f(void)
{
    int      a;       /* Valid within function */
         :
    /* address1 */
        :           /* "a" is used only within the range from address1 to address2.
*/
    /* address2 */
         :
}
```

In the above example, the scope of "a" is the entire function f().
However, use of "a" is limited to section between address1 and address2.In this case, if "a" is allocated to a register and optimization causes it to be deleted from the stack frame, "a" will become invisible outside of the section between address1 and address2.
This phenomenon occurs in order to make more efficient use of registers by making the register where "a" has been allocated (except for the section between address1 and address2) available for the allocation of other variables.

- During compilation, the processing of debug information uses a large amount of memory and therefore can cause an "out of memory" condition to occur.
- Sections that have been performed inline expansion are treated as a single unit, and cannot be stepped into.
- When a variable value is referenced, a temporary value from an ongoing calculation may be obtained instead of the correct value.
- If part of an array, structure elements, or pointer variables of user-defined types is allocated to registers, then variable display and editing on the Watch panel and the like of the debug tool will be illegal.

(3) Caution about optimization

The Caution about optimization is shown below.

- Optimization and register allocation are not performed across assembler code (__asm declarations or code between "#pragma asm" and "#pragma endasm" statements) and following embedded functions in a C program.
  __DI, __EI, __set_il, __nop, __halt, __ldsr, __stsr, __ldgr, __stgr
- Unused static functions and static variables in the file are deleted by default.
  Therefore, deleting unused static functions can be suppressed by specifying the -Odelete_static_func option.
- Functions for which "#pragma inline" is specified will not necessarily be performed inline expansion even if -Oinline=1 or 2 is specified.
  In the C language, the "#pragma inline" specification does not guarantee inline expansion; instead, it serves as a hint to the compiler. As with the "register" / "__inline" keywords, it is possible that the function will not be expanded, depending on the contents and compilation status.

B.1.6 Boot-flash re-link function

(1) Outline

Some systems are equipped with flash area or detachable ROM.

To upgrade the version of the program, the contents of the flash area may be rewritten or the detachable ROM may be replaced with a new ROM.

When changing the program even partially, basically the project itself is reorganized or rebuilt.

However, it would be convenient if the allocation to be upgraded was limited to the flash area or external ROM and if it was not necessary to reorganize the project.

The boot area is fixed to the internal ROM. If a function is called between the flash area to be rewritten and the boot area, and if the start address of the function is changed as a result of modifying the function in the flash area, the function cannot be called correctly.

The "boot-flash re-link function" (hereafter referred to as the "re-link function") is used to prevent this and enable functions to be called correctly.

This function is realized as follows.

(a) A "branch table" where instructions to branch to the functions in the flash area are written is prepared in the flash area.

(b) When a function in the flash area is called from the boot area, execution jumps to the branch table in the flash area, and then the instruction used to branch to the intended function is executed and jump occurs.

This mechanism can be realized by the user. If the "re-link function" is used, this can be done relatively easily.

To use this function, however, the functions to be called in the flash area must be determined when the boot area is created.

This mechanism is used to call a function from the boot area even if the function is modified in the flash area.

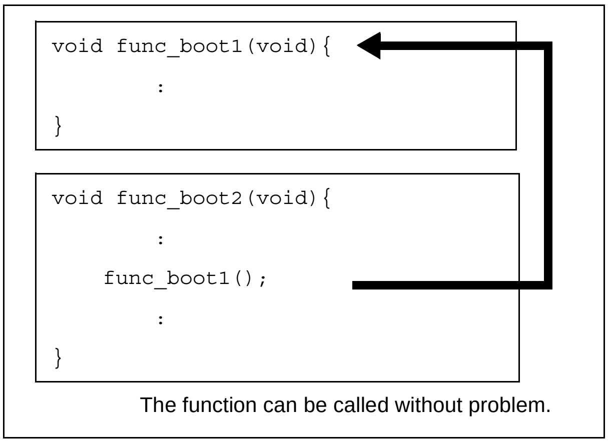
(2) Image of re-link function

A function is called as shown below when the re-link function is used.

(a) To call function in the boot area from the boot area

The function can be called without problem because addresses have been resolved before they are programmed to the boot area.

Figure B-4. In Boot Area



```
void func_boot1(void){
         :
}
```

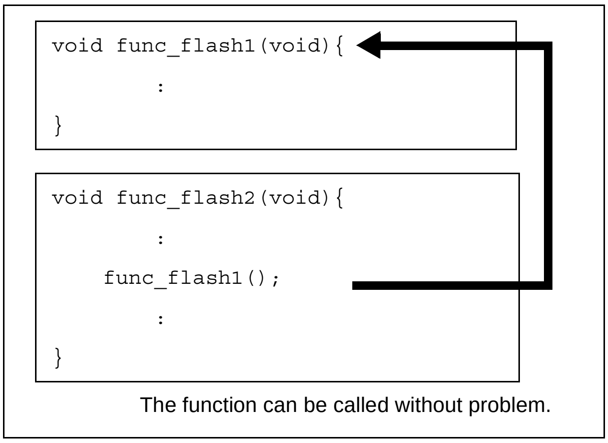
```
void func_boot2(void){
         :
    func_boot1();
         :
}
```

The function can be called without problem.

(b) To call function in the flash area from the flash area

The function can be called without problem because addresses have been resolved in the flash area.

Figure B-5. In Flash Area



```
void func_flash1(void){
        :
}
```

```
void func_flash2(void){
        :
    func_flash1();
        :
}
```

The function can be called without problem.

(c) To call function in the flash area from the boot area

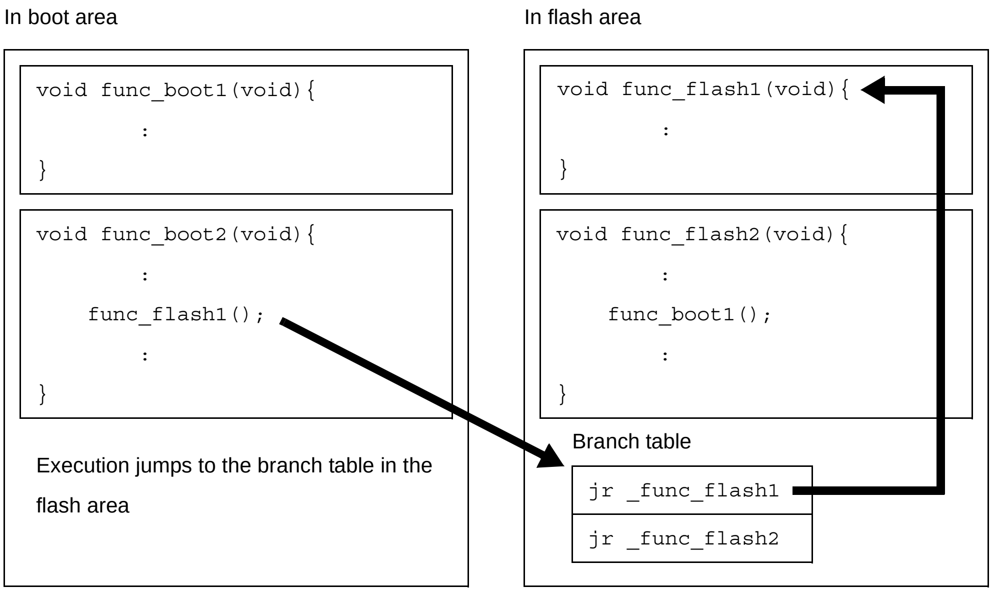
When a function in the flash area is called from the boot area, the address of the function cannot be specified from the boot area because the function size, etc., have been changed in the flash area.

In other words, a function in the flash area cannot be directly called.

To solve this, execution jumps to the branch table in the flash area.

Next, execute the jump instruction from that table to the relevant function and jump to the intended function.

Figure B-6. From Boot Area to Flash Area



```
void func_boot1(void){
        :
}
```

```
void func_boot2(void){
        :
    func_flash1();
        :
}
```

Execution jumps to the branch table in the flash area

```
void func_flash1(void){
        :
}
```

```
void func_flash2(void){
        :
    func_boot1();
        :
}
```

Branch table

```
jr _func_flash1
jr _func_flash2
```

In the same manner as functions, this is relevant to referencing external variables.

A global variable defined in the flash area cannot be referenced from the boot area.

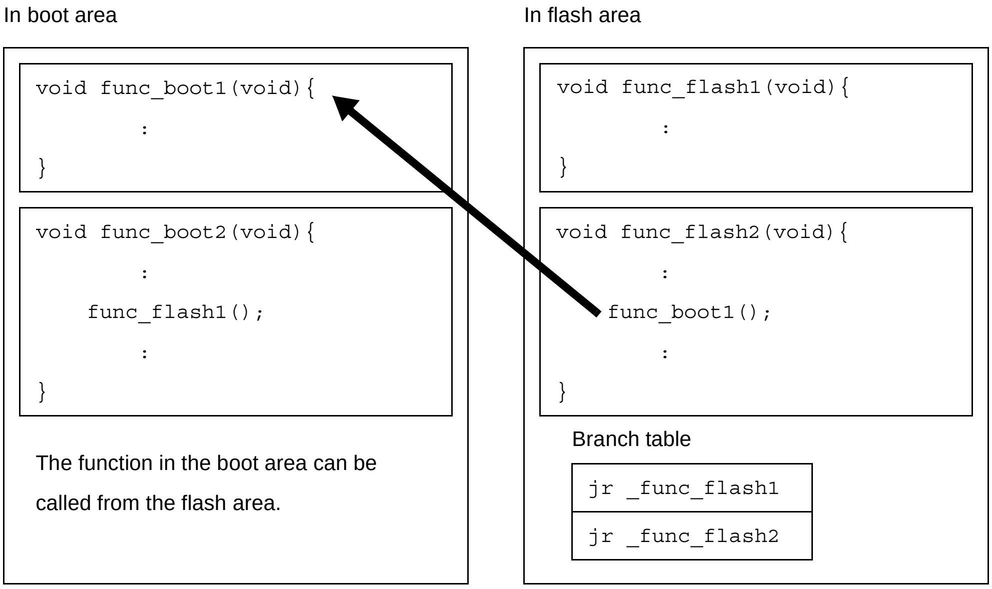
Therefore, an external variable of the same name can be defined in both the boot area and flash area.

Each of these external variables is referenced only from the respective areas.

(d) To call function in the boot area from the flash area

When a function in the boot area is called from the flash area, the contents of the boot area are not changed. Therefore, a function in the boot area can be directly called from the flash area.

Figure B-7. From Flash Area to Boot Area



In the same manner as functions, this is relevant to referencing external variables.

A global variable defined in the boot area cannot be referenced from the flash area.

(3) Realizing re-link function

To realize the re-link function, a boot area and flash area must be separately created.

This means that only the flash area is modified after the boot area has been created (after a program has been stored in ROM).

When creating a project with CubeSuite, therefore, divide the projects as follows.

- Project to be allocated to the boot area
- Project to be allocated to the flash area (project that may be modified in the future)

In addition, separately prepare a startup routine and link directive file for each area.

(a) Specify the $ext_func control instruction

When calling a function in the flash area from the boot area, specify the name of the function to be called (label name) and ID number to the boot area by using the $ext_func control instruction.

The format of the $ext_func control instruction is shown below.

```
$ext_func function-name, ID-value
```

When a function name in the flash area is specified in the boot area by using the $ext_func control instruction, a branch table (ext_table) is created.

Specify the address of this branch table (the first symbol is "__ext_table_head") using the -Xflash_ext_table option when the load module file for the boot area is generated and when the load module file for the flash area is generated.

It is recommended that allocating the branch table to the beginning of the flash area due to the relationship with the link directive file format.

If the internal ROM is a flash area, however, the INT segment goes first, so allocate it after this.

Specify the ID value in a positive number.
The different ID value must not be specified for the same function name or the same ID value must not be specified for the different function names.

The size of the branch table (byte) is calculated as shown below.
(maximum-ID-value + 1) * entry-size-of-branch-table
The address of the branch table must be set with consideration for this.
If this is not taken into account, the branch table could overlap with another section, causing an error.

When execution branches to the body of a function, the actual function address is obtained by referencing the offset of the ID value from the beginning of the created branch table, and then execution branches.

For example, when functions "func_flash0", "func_flash1", and "func_flash2" are allocated to the flash area and they are called from the boot area, describe as follows in the boot area.

- When describing in C language

```
void dummy() {
#pragma asm
$ext_func _func_flash0, 0
$ext_func _func_flash1, 1
$ext_func _func_flash2, 2
#pragma endasm
}
```

- When describing in assembly language

```
$ext_func _func_flash0, 0
$ext_func _func_flash1, 1
$ext_func _func_flash2, 2
```

It is recommended to describe these $ext_func control instructions in one file and include this file in all source files by using the $include control instruction (or #include directive when describing in C language), in order to prevent missing descriptions or the occurrence of contradictions, i.e., to prevent the error of specifying the different ID values for the same function name or specifying the same ID value for the different function names.

An image of re-link function is shown below.

Assembly Source Described By User	Assembler Image After Linking
[ext_table.inc] $ext_func _func_flash0, 0 $ext_func _func_flash1, 1 $ext_func _func_flash2, 2	

Assembly Source Described By User	Assembler Image After Linking

[boot.asm]
        $include(ext_table.inc)
        .extern _func_flash0
        .extern _func_flash1
        .extern _func_flash2
        jarl   _func_flash0, lp
        jarl   _func_flash1, lp
        jarl   _func_flash2, lp
	[boot.lmf]
	        .extern __ext_table_head
	        jarl __ext_table_head+0x4*0,lp
	        jarl __ext_table_head+0x4*1,lp
	        jarl __ext_table_head+0x4*2,lp

[flash.asm]
        $include(ext_table.inc)
        .public _func_flash0
        .public _func_flash2
_func_flash0:
            :
        jmp   [lp]
 	
        .public _func_flash1
_func_flash1:
            :
        jmp   [lp]
 	
_func_flash2:
            :
        jmp   [lp]
	[flash.obj]
	#(branch table)
	.ext_table .cseg text
	        .public __ext_table_head
	        .extern _func_flash0
	        .extern _func_flash1
	        .extern _func_flash2
	__ext_table_head:
	        jr _func_flash0
	        jr _func_flash1
	        jr _func_flash2

	#(function body)
	        .public _func_flash0
	_func_flash0:
	            :
	        jmp [lp]
	
	        .public _func_flash1
	_func_flash1:
	            :
	        jmp [lp]
	
	        .public _func_flash2
	_func_flash2:
	            :
	        jmp [lp]

If the $ext_func control instruction is specified as shown above, a table is created with symbol "ext_table", and the first symbol of this table is "__ext_table_head".
Code "jarl__flash0, lp" in the boot area is an offset from "__ext_table_head", and obtains the address of "_func_flash0" and jumps to the function body by the jarl instruction.

(b) Prepare the startup routines

Prepare startup routines for both the boot area and flash area.
Each startup routine must perform the following processing.

- Setting tp, gp, and ep values in the boot area
- Calling the _rcopy function to initialize the RAM area to be used for the boot area

- Branching from the boot area to the startup routine of the flash area
- Calling the _rcopy function to initialize the RAM area to be used for the flash area
- Moving to the processing of the flash area

Remarks 1. If tp, gp, and ep are not used in the boot area, the values may be set in the flash area.
If ROMization processing is not performed, the _rcopy function call is not required.

2. Use the same address values in the boot area and flash area for the tp, gp, and ep values.
These values may be different, but in this case the values must be set each time control has been transferred between an instruction code in the boot area and one in the flash area.

(c) Prepare the link directive files

Prepare link directive files for the projects for both the boot area and flash area.
The following points should be noted when describing a link directive file.

- Even if the address of a section placed in the boot area overlaps in the boot area and flash area, an error cannot be output when linking because the projects are different.
  In other words, the addresses can overlap.
  However, that only the last RAM area to be written is valid. For this reason, for the RAM area that must be referenced simultaneously in the boot area and flash area, the addresses must be specified so that they do not overlap.
- Use the same address values in the boot area and flash area for the tp, gp, and ep values.
  These values may be different, but in this case the values must be set each time control has been transferred between an instruction code in the boot area and one in the flash area.
- Allocation to the branch table (ext_table) does not have to be described in the link directive file.
  It is automatically allocated to an address specified by the -Xflash_ext_table option.
  However, the following points must be noted.
  - If a vacant area of the size of the branch table is at the address specified by the -Xflash_ext_table option, the link directive file is allocated as is.
    The other segments are not affected.
    This is the most ideal case.
  - If a vacant area of the size of the branch table is not at the address specified by the -Xflash_ext_table option, an error will occur.
    This applies, for example, if a code has been already allocated to the address specified by the -Xflash_ext_table option in a TEXT segment for which an address is specified.
    The example is as follows.

Address specification of the branch table

```
-Xflash_ext_table=0x300
```

Link directive file (part)

```
TEXT : !LOAD ?RX V0x400{
        .pro_epi_runtime = $PROGBITS ?AX;
        .text            = $PROGBITS ?AX;
};
```

(Size of TEXT segment is 0x100 bytes or more)

An error occurs during linking because the branch table cannot be allocated to address 0x400.

Change the value specified by the -Xflash_ext_table option.

- If another segment is allocated to the address specified by the -Xfalsh_ext_table option before the re-link function is used but the address of that segment is not specified in the link directive file, the branch table will be allocated to the address specified by the -Xfalsh_ext_table option and the original segment will be moved behind the branch table.
  However, If the segment overlaps a segment for which an address is specified as a result of moving, an error occurs.

Address specification of the branch table

```
-Xflash_ext_table=0x300
```

Link directive file (part)

```
TEXT : !LOAD ?RX{
        .pro_epi_runtime = $PROGBITS ?AX;
        .text            = $PROGBITS ?AX;
};
```

(It is assumed that the TEXT segment is allocated from address 0x300, continuing from the segment prior to the TEXT segment)

At this time, the branch table is allocated to address 0x300 because no address is specified for the TEXT segment, and the TEXT segment is allocated behind the branch table.

(d) Specify the $ext_ent_size control instruction

When an actual function is called from the branch table in the flash memory, jr branch instructions are generated as follows by default.

```
__ext_table_head:
        jr      _func_flash0
        jr      _func_flash1
        jr      _func_flash2
```

However, the jr instruction can branch only within a 22-bit range (±1MB).
To branch in the entire 32-bit space, additionally specify the $ext_ent_size control instruction.
The format of the $ext_ent_size directive is shown below.

```
$ext_ent_size entry size
```

The value that can be specified as the entry size is 4 or 8 (V850Ex core).
The entry size means the instruction size necessary for one branch processing.
The default entry size is 4. A 4-byte instruction is allocated as follows.

```
jr      _flash_func0    -- 4-byte instruction
```

If 8 is specified, a total of 8 bytes of instructions will be allocated, as follows.

```
mov     #_flash_func0, r1       -- 6-byte instruction
jmp     [r1]                    -- 2-byte instruction
```

It is assumed that there is only one entry size for all source.
An abort error will occur if a different value is specified for each source.

(e) Specify the $ext_func control instruction for the library

If a library function is called from the boot area or flash area, the library is linked to the object on the calling side.
For example, even if a library is linked to the flash area, the same library is linked to the boot area if the same library function is called from the boot area.
When a library function is called, therefore, a function does not have to be specified by the $ext_func control instruction for the library function because branching does not take place between the boot area and flash area.
However, in a special case where the library linked to the boot area branches to a function in the flash area, a function must be specified by the $ext_func control instruction.
A function does not have to be specified by the $ext_func control instruction for the library attached to the CX.

(f) Specify the $ext_func control instruction for the interrupt handler

Describe the part that calls an interrupt handler in the area where the address of the interrupt handler exists.
In the following case, an interrupt handler function name must also be specified by the $ext_func control instruction.

- When the interrupt handler address is in the boot area
- When the interrupt handler body is in the flash area

Assembler Source Described By User	Assembler Image After Linking
[ext_table.inc] $ext_func _int_flash0, 0	
[boot.asm] $include(ext_table.inc) .extern _int_flash0 .cseg TEXT jr _int_flash0	[boot.lmf] .cseg TEXT jr __ext_table_head+0x4*0,lp
[flash.asm] $include(ext_table.inc) .public _int_flash0 _int_flash0: : reti	[flash.lmf] #(branch table) .cseg TEXT .public __ext_table_head .extern _int_flash0 __ext_table_head: jr _int_flash0
	#(handler body) .public _int_flash0 _int_flash0: : reti

(g) Method for manipulating

The method for manipulating on the command line is shown below.

Remark See "2.12 Prepare for Implementing Boot-Flash Relink Function" for the method manipulating in CubeSuite.

<1> Generate the load module file for the boot area

Create startup routine for the boot area "cstartN_b.obj".

```
>cx -CF3746 cstartN_b.asm
```

Link startup routine "cstartN_b.obj" and link directive file "directive_b.dir" to generate load module file for the boot area "boot.lmf".

The start address of the branch table for the flash area is regarded as 0x300.

ROMization processing is also performed by default.

```
>cx -CF3746 -Xstartup=cstartN_b.obj boot.asm -Xlink_output=boot.lmf -
Xflash_ext_table=0x300 -Xlink_directive=directive_b.dir -oromp_b.lmf
```

If ROMization processing is not performed, specify as follows.

```
>cx -CF3746 -Xstartup=cstartN_b.obj boot.asm -Xno_romize -oboot.lmf -
Xflash_ext_table=0x300 -Xlink_directive=directive_b.dir
```

<2> Generate the load module file for the flash area

Create startup routine for the flash area "cstartN_f.obj".

```
>cx -CF3746 cstartN_f.asm
```

Link startup routine "cstartN_f.obj" and link directive file "directive_f.dir" to generate load module file for the flash area "flash.lmf".

At this time, symbol information of load module file for the boot area "boot.lmf" is referred and link processing is performed.

The branch table is created at address 0x300.

ROMization processing is also performed by default.

```
>cx -CF3746 -Xno_startup -Xstartup=cstartN_f.obj flash.asm -Xlink_output=flash.lmf
-Xflash=boot.lmf -Xflash_ext_table=0x300 -Xlink_directive=directive_f.dir -
oromp_f.lmf
```

If ROMization processing is not performed, specify as follows.

```
>cx -CF3746 -Xno_startup -Xstartup=cstartN_f.obj flash.asm -Xno_romize -oflash.lmf
-Xflash=boot.lmf -Xflash_ext_table=0x300 -Xlink_directive=directive_f.dir
```

Cautions 1. The load module file for the boot area must be the file before ROMization processing.

2. The address specified by the -Xflash_ext_table option must be the same as the value that is used in the boot area and flash area.

B.1.7 Cautions

This section explains cautions about the cx command.

(1) How to use the -Xsdata_info option

This section describes how to use the -Xsdata_info option.

With CubeSuite, on the Property panel, from the [Link Options] tab, in the [Other] category, set the [Display GP information] property to [Yes(-Xsdata_info)].

(a) Function

Information that can be used as a yardstick for parameter *num* of the -Xsdata option is output to the standard output.

With CubeSuite, it is output on the Output panel.

The -Xsdata option allocates data of less than *num* bytes to the .sdata or .sbss section.

The cx outputs codes in compliance with the following rule for the data allocated to the sdata, sbss, data, and bss areas.

The cx first tries to allocate the data to the sdata-attribute section or sbss-attribute section, which are areas that can be accessed with a single instruction from the gp register (data with an initial value is allocated to the sdata-attribute section and data without an initial value is allocated to the sbss-attribute section).

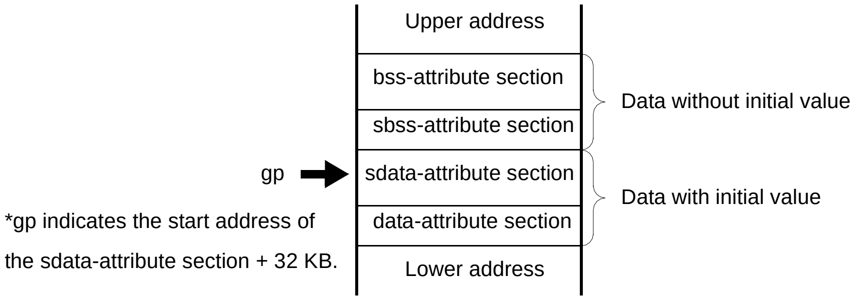
Because these areas are accessed by a code that uses gp and a 16-bit displacement for access, data can be allocated only in a range of ±32 KB from gp.

If the data does not fit in these areas, the cx tries to allocate the data to the data-attribute section or bss-attribute section, which are areas that can be accessed with two instructions from the gp register (data with an initial value is allocated to the data-attribute section and data without an initial value is allocated to the bss-attribute section).

In these areas, the address of the access area is first generated, and a code using gp and a 32-bit displacement for access is generated.

Consequently, the entire 4 GB space can be accessed.

Figure B-8. Memory Allocation Image of gp Offset Reference Section



*gp indicates the start address of the sdata-attribute section + 32 KB.

Therefore, the execution efficiency and object efficiency are enhanced if more data is allocated to the sdata-attribute or sbss-attribute section, which can be accessed with a single instruction.

To allocate data, the user can intentionally specify the allocation location by using the #pragma section directive in the case of a C source or by using the .section quasi directive in the case of an assembler source.

If a threshold value of the size of the data to be allocated to the sdata-attribute or sbss-attribute section is prepared and if data of a size less than the threshold value can be allocated to the sdata-attribute or sbss-attribute section, more data can be allocated without having to modify the source program.

This specification is made by the -Xsdata option.

The value specified as *num* of this option is the data size, so it would be convenient to have information that can be used as a yardstick.

This information is output by the -Xsdata_info option.

If the -Xsdata_info option is specified, information that can be used as a yardstick for *num* of the -Xsdata option.

(b) Explanation of output information

The examples of the output information when this option is specified when an executable object module file is generated (when the -Xrelinkable_object option is not specified), and when this option is specified when a relocatable object module file is generated (when the -Xrelinkable_object option is specified) are shown below.

Examples 1. Output Information for the executable load module file

```
        ******** LINK EDITOR GP INFORMATION ********
   (1)       (2)         (3)          (4)           (5)        (6)
GP SYMBOL   SECTION     SECTION     SECTION         GP
NAME        NAME        SIZE(REAL)  SIZE(ASSUMED)   NUMBER
_gp_DATA
            .sdata      0x000af10
                                    0x00002000      4           *OK*
                                    0x00003450      8           *OK*
                                    0x00004430      12          *OK*
                                    0x000050a8      16          *OK*
                                    0x00007b40      20          *OK*
                                    0x0000a010      24
                                    0x0000af10      32
            .sbss       0x00012050
                                    0x00000050      4           *OK*
                                    0x00002050      16          *OK*
                                    0x00007050      512         *OK*
                                    0x00010050      1024
```

2. Output Information for the relocatable object module file

```
        ******** LINK EDITOR GP INFORMATION ********
   (1)       (2)          (3)          (4)           (5)        (6)
GP SYMBOL   SECTION     SECTION     SECTION         GP
NAME        NAME        SIZE(REAL)  SIZE(ASSUMED)   NUMBER
*(NOT AVAILABLE)
            .sdata      0x000af10
                                    0x00002000      4           *OK*
                                    0x00003450      8           *OK*
                                    0x00004430      12          *OK*
                                    0x000050a8      16          *OK*
                                    0x00007b40      20          *OK*
                                    0x0000a010      24
                                    0x0000af10      32
            .sbss       0x00012050
                                    0x00000050      4           *OK*
                                    0x00002050      16          *OK*
            *GpCommon*  0x00010000
                                    0x00005000      512         *OK*
                                    0x00010000      1024
```

Item Number	Description
(1)	Name of global pointer symbol This is the name of the global pointer symbol used for linking. If the object module file is relocatable, "*(NOT AVAILABLE)*" is displayed.
(2)	Section name This is the name of the sdata-attribute section or sbss-attribute section to which data are allocated. Because a relocatable object module file cannot determine allocation of an undefined external symbol to a section, the cx internally generates a virtual section "*GpCommon*" and temporarily allocates the data to this section.
(3)	Actual size of section This is the actual size of the section that is considered for use as the area for the hole generated by data alignment.
(4)	Assumed size of section This is the size of the section that is assumed if compilation is performed with the -Xsdata option (with the value shown in (5) as *num*). Because the calculation of this size assumes an alignment condition of more than 4 bytes without taking the actual alignment condition into consideration, the value shown in this column does not necessarily agree with the actual size of the created section.
(5)	Value of *num* of the -Xsdata option assumed This is the value of *num* of the -Xsdata option during compilation and assembling, which is assumed as a result of calculating the "assumed size of section" shown (4).

Item Number	Description
(6)	Judgement result This is the result of the judgment[Note] as to whether or not the size of the section is within a range of 15 bits (0x0 to 0x7fff) if compilation is performed with the -Xsdata option with the value shown in (5) (specified as *num*). If the size is within this range, "*OK*" is displayed; if it is not, nothing is displayed.

Note Usually the sections to which data is allocated are allocated from the lower address in the order of data/sdata/sbss/bss attribute sections in the cx. The global pointer (gp) is assumed to be set in the startup module, etc. so as to indicate the start address of the sdata-attribute section + 32 KB. Therefore, if the result is OK in this judgement, the sdata/sbss attribute sections are assumed to be allocated to a memory range that can be referenced using 16-bit displacement.

(c) Cautions

The information output by the -Xsdata_info option is only a yardstick, and the judgment result may not be correct, such as in the following cases:

- When allocation of a section that creates a hole is specified by a link directive, etc.
- When a direct address is specified for a global pointer symbol
- When data is allocated to the .sdata/.sbss section by the #pragma section directive

(2) Library file

An library file is created by linking two or more object module files with the librarian.
The cx searches library files for unresolved external references[Note 1] after linking all object module files and links only the necessary object module files.

The library file can be also specified in the mapping directive of the link directive.
If the library file is also specified in the mapping directive, it is searched for unresolved external references at that time and only the necessary object module files[Note 2] are linked.

Notes 1. The library file includes a symbol table of the symbols belonging to the object module file, and the library file is repeatedly searched as long as unresolved external references remain unresolved.

2. Object module file that defines the referenced symbol.

(3) Reserved symbols

During link processing, the cx generates reserved symbols with the values of the start address of each output section, the first address beyond the end of each output section (4-byte aligned address), and the first address beyond the end of the executable load module file.
If the user defines a symbol having the same name as any of these reserved symbols, the cx uses the defined symbol, and does not create its own symbol.

A symbol having a name made by prefixing "__s" to the name of the output section is used as a reserved symbol that has the start address of a section as a value.
If this section name begins with ".", "." is taken out and "__s" is prefixed to make it a symbol name.
The reserved symbols with the values of the first address exceeding the end of a section (4-byte aligned address) consist of the output section name, prefaced with "__e".
If this section name begins with ".", "." is taken out and "__e" is prefixed to make it a symbol name.
"__end" is used as the reserved symbol with the value of the first address beyond the end of the executable load module file that was generated (4-byte aligned address).

The default link directive used by the cx uses the following reserved sections as output sections.

```
.text, .pro_epi_runtime, .data, .sdata, .sbss, .bss, .sconst, .const, .sedata, .sebss, .sidata, .sibss,.tidata, .tidata.byte,
.tidata.word, .tibss, .tibss.byte, .tibss.word
```

Therefore, the cx normally creates the following reserved symbols.

```
__end, __ebss, __econst, __edata, __epro_epi_runtime, __esbss, __esconst, __esdata, __esebss, __esedata, __esibss,
__esidata, __etext, __etibss, __etibss.byte, __etibss.word, __etidata, __etidata.byte, __etidata.word, __sbss, __sconst,
__sdata, __spro_epi_runtime, __ssbss, __ssconst, __ssdata, __ssebss, __ssedata, __ssibss, __ssidata, __stext,
__stibss, __stibss.byte, __stibss.word, __stidata, __stidata.byte, __stidata.word
```

Caution Of the above symbols, only those for which a section exists in the executable file after link processing are generated.
The cx behaves as if no section exists if a section that is actually allocated does not exist even if a mapping directive is described in the link directive file.

(4) When it may not be allocated to the expected sections

Even if a link directive file specifies an object module file or library file to be allocated to a section, the object module file or library file may not be allocated to the expected sections, depending on how the file name is described.

In such cases, relink using the link directive file with the exact same name displayed in the map file, including the path and file name.

(5) main function

If linking is performed without creating a main function, an error message may be output to indicate that the _main symbol is an undefined symbol.

This may occur when the user links the default startup routine (cstart.obj or cstartN.obj) rather than a user-specified startup routine, or when "cstart.asm" or "cstartN.asm" is used as they are assembled and linked.

The error is due to code "jarl _main, lp" that is written following "cstart.asm" and "cstartN.asm".

If the main function is not needed, overwrite this code then use the reassembled object module file as the startup routine.

In the case of an application that uses the real-time OS, main function does not exist normally.

Use the startup routine provided as a sample of the real-time OS.

(6) Prologue/epilogue runtime library

The prologue/epilogue runtime library must be allocated to the special-purpose .pro_epi_runtime section.

If it is not allocated there, the following message will be output and linking will be stopped.

```
F0560657:Section "section" must be specified in link directive.
```

If a link directive file has been specified, describe the mapping directive before the .text section.

```
.pro_epi_runtime = $PROGBITS ?AX .pro_epi_runtime;
.text = $PROGBITS ?AX;
```

If the .pro_epi_runtime section is placed after the .text section, it overlaps the allocation position of the default operation of the section that is packed during ROMization. Therefore, allocating the .pro_epi_runtime section before the .text section is recommended.

If a link directive file has not been specified, link before the .text section.

Other cautions are shown below.

- The prologue/epilogue runtime libraries are included in standard library "libc.lib".
- Unlike ordinary sections, the .pro_epi_runtime section has a fixed input section name and only the special-purpose section is allocated.
- The prologue/epilogue runtime libraries use the callt instruction when a device of the V850Ex/V850E2 core is used.
  Set CTBP in the startup routine.

(7) Programmable peripheral I/O register

For an application program that uses programmable peripheral I/O register functions, the .bpc section (which is a reserved section) is output when assembling.
If there is the .bpc section in an input object module file when linking, the value specified as BPC is checked.
If values do not match between input object module files, the cx outputs an error message like the following and suspends link processing.

```
F0560114:Input files have different BPC value.
0x00001234      file1.obj
0x00001234      file2.obj
0x00001235      file3.obj
*(none)*        file4.obj
```

In the above case, there is an error because the value set in file3.obj is different.
Object that does not reference the programmable peripheral I/O register is not checked.
As in file4.obj above, "*(none)*" is displayed.
If there are no errors in checking BPC values, a .bpc section is generated with section type "SHT_PROGBITS", section attribute "none", and section size 0x4.
The start address of the programmable peripheral I/O register area, which is the BPC value shifted a preset number of bits, is stored in the .bpc section.

Example If the BPC value is specified as "0x1234" when using the V850E/IA1, the start address of the programmable peripheral I/O register area is the value shifted 14 bits to the left, or "0x48d0000".In this case, the information in the .bpc section is as follows.

```
.bpc
  Address    00  01  02  03  04  05  06  07  -  08  09  0A  0B  0C  0D  0E  0F
0x00000000 : 00  00  8d  04                  -                               ...
```

- The processing above is performed without question when generating a relocatable object module file and when generating an executable object module file.
- The .bpc section is a special reserved section for information and is never loaded into memory.
  Therefore, it need not be specified in a link directive like a normal section.

(8) Debug information

When displaying debug information for types specified with a different name via typedef, the original type may not match the type described in the source file (it will be a different type with the same size and signed/unsigned specification).

- If the original type is unsigned or unsigned int, then it will be unsigned long.

Example

```
typedef unsigned int UI;    /* In the debug information, the original type in UI
will be unsigned long. */
```

- If the original type is signed, signed int, or signed long, then it will be long.

Example

```
typedef signed int SI;      /* In the debug information, the original type in SI
will be long. */
```

- If the original type is signed short, then it will be short.

Example

```
typedef signed short SS;    /* In the debug information, the original type in SS
will be short. */
```

- If the original type is signed char, then it will be char.

Example

```
typedef signed char SC;     /* In the debug information, the original type in SC
will be char. */
```

- If the original type is char and the -Xchar=unsigned option is specified, then it will be unsigned char.

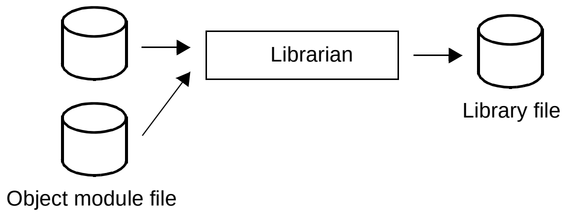
Example

```
typedef char SC;    /* In the debug information, the original type in SC will be
unsigned char. */
```

B.2 Librarian

The librarian couples specified relocatable object module files and creates a library file.
Therefore, this utility is used to combine two or more object module files to create a "library".
In the CX, "lb" is the librarian.

Figure B-9. Operation Flow of Librarian



Details about the functions of the librarian are shown below.

- Performing the library formation for object modules
  The cx creates one file for one output module.
  Therefore, if many modules exist, the number of files will increase.
  For this reason, the cx provides a function that two or more modules are combined into a file.
  This function is called the module library formation. A file which is organized as a library is called a library file.

  The library file created by the librarian can be specified as an input file to the cx.
  If a library file is specified, the cx searches the necessary object module file from the specified library file and links only the object module file that can be found.
  Therefore, by creating a library file consisting of modules common to many programs, users can make file management and operation efficient and easy when performing modular programming.

- Editing a library file
  The librarian provides the following edit functions for a library file.
  - Addition of an object module file to a library file
  - Deletion of an object module file from a library file
  - Movement of an object module file in a library file
  - Replacement of object module files in a library file
  - Retrieval of an object module file from a library file

B.2.1 I/O files

The I/O files of the librarian are shown below.

Table B-11. I/O Files of Librarian

File Type	Extension	I/O	Description
Object module file	.obj	I	Binary file including machine-language information, relocation information relating to machine-language allocation addresses, and symbol information
Library file	.lib	O	File in which two or more object module files are included

Remark Extension above is the default. It can be freely changed.

B.2.2 Method for manipulating

This section explains how to manipulate the librarian.

(1) Startup from the command line

Enter the following on the command line.

```
>lb[Δerror-output-control-option]Δkey[option][Δmember-nameNote]Δlibrary-file-name[Δmember-
name or file-name]...
```

[ ]: Can be omitted
...: Pattern in proceeding [ ] can be repeated
Δ: One or more spaces

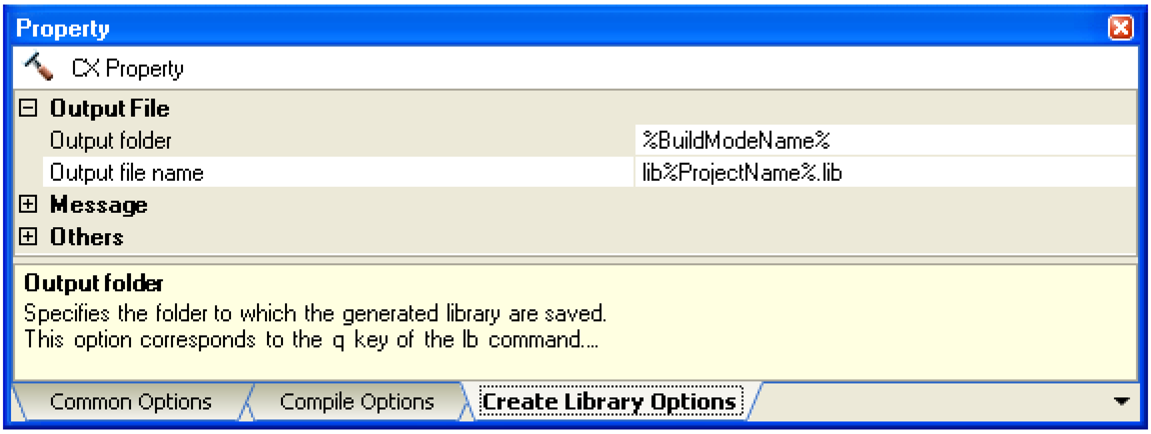
Note When an object module file is linked within a library file, it is called a member.
Each member has the same name as its original file name.

Remark You must specify one key when you runs the librarian. An option can be omitted.

(2) Set options in CubeSuite

This section describes how to set create library options from CubeSuite.
On the CubeSuite's Project Tree panel, select the Build Tool node. Next, select the [View] menu >> [Property].
The Property panel opens.
Next, select the [Create Library Options] tab.
You can set the various create library options by setting the necessary properties in this tab.

Figure B-10. Property Panel: [Create Library Options] Tab



⊟ **Output File**	
Output folder	%BuildModeName%
Output file name	lib%ProjectName%.lib
⊞ **Message**	
⊞ **Others**	

Output folder
Specifies the folder to which the generated library are saved.
This option corresponds to the q key of the lb command....

When activating the librarian from the command line, combine object module files to create a library file.
In addition, detailed operations can be performed, such as manipulation of library file objects.
By contrast, when using CubeSuite to create a library file, compile and assemble source files, and then combine the object module files to be generated into a library file.
Operations cannot be performed for a complete library file via CubeSuite.
The user should keep this difference in mind when choosing between command-line activation and activation via CubeSuite.

B.2.3 Key/Option

The types and explanations for keys and options of the librarian are shown below.

Table B-12. Create Library Key

Classification	Key	Description
Version display specification	V	This key displays the version information of the librarian.
Member delete specification	d	This key deletes a member from the specified library file.
Member move specification	m	This key moves the specified member in the specified library file to the end of the file.
	ma	This key moves the specified member in the specified library file to the position immediately after the specified member.
	mb	This key moves the specified member in the specified library file to the position immediately before the specified member.
Member add specification	q	This key adds the specified object module file to the end of the specified library file.
Member replace control	r	This key replaces the specified object module file with the member having the same name in the specified library file.
	ra	This key replaces the specified object module file with the member having the same name in the specified library file, and then moves the object module file to the position immediately after the specified member.
	ru	When the specified object module file has been updated more recently than the member having the same name in the specified library file, this key replaces the member with the specified object module file.
Member output specification	t	This key outputs the names of the members which exist in the specified library file to the standard output.
Object generate control	x	This key extracts the member which exists in the specified library file and generates the object module file with the same name.

Table B-13. Create Library Options

Classification	Option	Description
Message output suppress specification	c	This option does not output messages.
Execution status output specification	v	This option outputs the execution status of the librarian.
Command file specification	@	This option specifies a command file.
Error output control	+err_file	This option adds and saves error messages to the specified file.
	-err_file	This option overwrites and saves error messages to the specified file.

Version display specification

The version display specification key is as follows.

- V

V

This key displays the version information of the librarian.

[Specification format]

```
V
```

- Interpretation when omitted
  The version information of the librarian is not output.

[Detailed description]

- This key outputs the version information of the librarian to the standard error output and terminates processing.

[Example of use]

- To output the version information of the librarian to the standard error output, describe as:

```
>lb V
```

Member delete specification

The member delete specification key is as follows.

- d

d

This key deletes a member from the specified library file.

[Specification format]

```
d
```

- Interpretation when omitted
  None

[Detailed description]

- This key deletes a member from the specified library file.

[Example of use]

- To delete member "sub.obj" from library file "libmain.lib", describe as:

```
>lb d libmain.lib sub.obj
```

Member move specification

The member move specification keys are as follows.

- m
- ma
- mb

m

This key moves the specified member in the specified library file to the end of the file.

[Specification format]

```
m
```

- Interpretation when omitted
  Members are not moved.

[Detailed description]

- This key moves the specified member in the specified library file to the end of the file.

[Example of use]

- To move member "sub.obj" in library file "libmain.lib" to the end of the file, describe as:

```
>lb m libmain.lib sub.obj
```

ma

This key moves the specified member in the specified library file to the position immediately after the specified member.

[Specification format]

```
ma member
```

- Interpretation when omitted
  Members are not moved.

[Detailed description]

- This key moves the specified member in the specified library file to the position immediately after member *member*.
- If *member* is omitted, processing will be stopped.

[Example of use]

- To move member "sub.obj" in library file "libmain.lib" to the position immediately after member "main.obj", describe as:

```
>lb ma main.obj libmain.lib sub.obj
```

mb

This key moves the specified member in the specified library file to the position immediately before the specified member.

[Specification format]

```
mb member
```

- Interpretation when omitted
  Members are not moved.

[Detailed description]

- This key moves the specified member in the specified library file to the position immediately before member *member*.
- If *member* is omitted, processing will be stopped.

[Example of use]

- To move member "sub.obj" in library file "libmain.lib" to the position immediately before member "main.obj", describe as:

```
>lb mb main.obj libmain.lib sub.obj
```

Member add specification

The member add specification key is as follows.

- q

q

This key adds the specified object module file to the end of the specified library file.

[Specification format]

```
q
```

- Interpretation when omitted
  None

[Detailed description]

- This key adds the specified object module file to the end of the specified library file.
- If the specified library file does not exist, a new library file that contains the specified object module file will be created.
- There is no checking as to whether or not the member with the same name as the specified object module file exists.
  If the member with the same name exists, the library file contains multiple members with the same name, and the oldest member will be selected during linking.
  Be sure to delete the old library file in order to prevent mixing the members with the same name when a new library file is created.
  Otherwise, it is recommended to use the r key.
- Use the r key to replace the member with the member having the same name.

[Example of use]

- To add object module file "sub.obj" to the end of library file "libmain.lib", describe as:

```
>lb q libmain.lib sub.obj
```

Member replace control

The member replace control keys are as follows.

- r
- ra
- ru

r

This key replaces the specified object module file with the member having the same name in the specified library file.

[Specification format]

```
r
```

- Interpretation when omitted
  Members are not replaced.

[Detailed description]

- This key replaces the specified object module file with the member having the same name in the specified library file.
- If the member with the same name as the specified object module file does not exist in the specified library file, the object module file is added to the end of the library file.
- If the specified library file does not exist, a new library file that contains the specified object module file will be created.
- This option is equivalent to the following property in CubeSuite.
  - From the [Create Library Options] tab, [Output folder] in the [Output File] category
  - From the [Create Library Options] tab, [Output file name] in the [Output File] category

[Example of use]

- To replace "sub.obj" with the member having the same name in library file "libmain.lib", describe as:

```
>lb r libmain.lib sub.obj
```

ra

This key replaces the specified object module file with the member having the same name in the specified library file, and then moves the object module file to the position immediately after the specified member.

[Specification format]

```
ra member
```

- Interpretation when omitted
  Members are not replaced.

[Detailed description]

- This key replaces the specified object module file with the member having the same name in the specified library file, and then moves the object module file to the position immediately after member *member*.
- If the member with the same name as the specified object module file does not exist in the specified library file, the object module file is added to the end of the library file.
- If *member* is omitted, processing will be stopped.

[Example of use]

- To replace "sub.obj" with the member having the same name in library file "libmain.lib", and then move "sub.obj" to the position immediately after member "main.obj", describe as:

```
>lb ra main.obj libmain.lib sub.obj
```

ru

When the specified object module file has been updated more recently than the member having the same name in the specified library file, this key replaces the member with the specified object module file.

[Specification format]

```
ru
```

- Interpretation when omitted
  Members are not replaced.

[Detailed description]

- When the specified object module file has been updated more recently than the member having the same name in the specified library file, this key replaces the member with the specified object module file.
- If the member with the same name as the specified object module file does not exist in the specified library file, the object module file is added to the end of the library file.
- If the specified library file does not exist, a new library file that contains the specified object module file will be created.

[Example of use]

- When the specified object module file has been updated more recently than the member having the same name in the specified library file, to replace the member with the specified object module file, describe as:

```
>lb ru libarc.lib sub.obj
```

Member output specification

The member output specification key is as follows.

- t

t

This key outputs the names of the members which exist in the specified library file to the standard output.

[Specification format]

```
t
```

- Interpretation when omitted
  Members are not output.

[Detailed description]

- When the member name is specified, this key outputs the name of that member which exists in the specified library file to the standard output.
- When the member name is not specified, this key outputs the names of all members which exist in the specified library file to the standard output.

[Example of use]

- To output the names of all members which exist in library file "libmain.lib" to the standard output, describe as:

```
>lb t libmain.lib
```

Object generate control

The object generate control key is as follows.

- x

x

This key extracts the member which exists in the specified library file and generates the object module file with the same name.

[Specification format]

```
x
```

- Interpretation when omitted
  Members are not extracted.

[Detailed description]

- If the member name is specified and the member exists in the specified library file, this key extracts that member and generates the object module file with the same name.
- If the member name is not specified, this key extracts all members which exist in the specified library file and generates the object module files with the same name.
  The contents of the library file are not changed.

[Example of use]

- To extract member "sub.obj" which exists in library file "libmain.lib" and generate "sub.obj", describe as:

```
>lb x libmain.lib sub.obj
```

Message output suppress specification

The message output suppress specification option is as follows.

- c

c

This option does not output messages.

[Specification format]

```
c
```

- Interpretation when omitted
  Messages are output.

[Detailed description]

- This option does not output messages.

[Example of use]

- Not to output messages, describe as:

```
>lb tc libmain.lib
```

Execution status output specification

The execution status output specification option is as follows.

- v

v

This option outputs the execution status of the librarian.

[Specification format]

```
v
```

- Interpretation when omitted
  The execution status of the librarian is not output.

[Detailed description]

- This option outputs the execution status of the librarian using format "[a|d|q|m|r|x] - *file*".

a - *file*	Add
d - *file*	Delete
q - *file*	Create new, or add
m - *file*	Move
r - *file*	Replace
x - *file*	Extract

- This option is equivalent to the following property in CubeSuite.
  - From the [Create Library Options] tab, [Verbose mode] in the [Output File] category

[Example of use]

- Member "sub.obj" is deleted from library file "libmain.lib".
  At this time, to output the execution status of the librarian, describe as:

```
>lb dv libmain.lib sub.obj
```

Command file specification

The command file specification option is as follows.

- @

@

This option specifies a command file.

[Specification format]

```
@file
```

- Interpretation when omitted
  It is assumed that a command file is not specified.

[Detailed description]

- This option handles *file* as a command file.
- See "(2) Startup from a command file" for details about a command file.

[Example of use]

- To handle "command" as a command file, describe as:

```
>lb @command
```

Error output control

The error output control option is as follows.

- +err_file
- -err_file

+err_file

This option adds and saves error messages to the specified file.

[Specification format]

```
+err_file=file
```

- Interpretation when omitted
  An error message file is not created.

[Detailed description]

- This option adds and saves error messages to file *file*.
- This option must be specified at the beginning of the command line (before the key).

[Example of use]

- Member "sub.obj" is deleted from library file "libmain.lib".
  At this time, to add and save error messages to file "err", describe as:

```
>lb +err_file=err d libmain.lib sub.obj
```

-err_file

This option overwrites and saves error messages to the specified file.

[Specification format]

```
-err_file=file
```

- Interpretation when omitted
  An error message file is not created.

[Detailed description]

- This option overwrites and saves error messages to file *file*.
- This option must be specified at the beginning of the command line (before the key).

[Example of use]

- Member "sub.obj" is deleted from library file "libmain.lib".
  At this time, to overwrite and save error messages to the file "err", describe as:

```
>lb -err_file=err d libmain.lib sub.obj
```

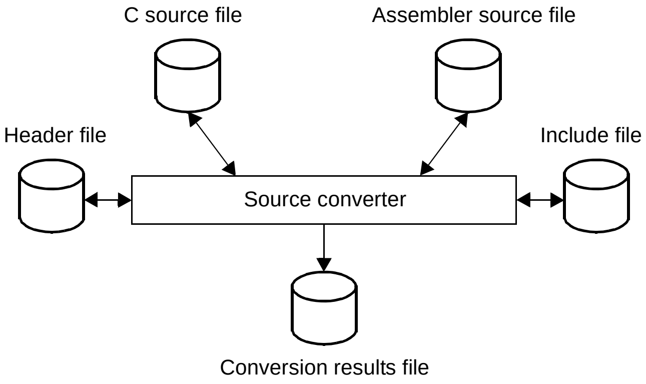
B.3 Source Converter

The source converter converts source files (C source file, assembler source files, etc.) created for CA850 into formats that can be used by CX.

The converted files are output retaining the code for CA850 as a comment.

Note that an option can be specified to output the conversion results to a text file.

Figure B-11. Operation Flow of Source Converter



B.3.1 I/O files

The I/O files of the source converter are shown below.

Table B-14. I/O Files of Source Converter

File Type	Extension	I/O	Description
C source file	.c	I/O	The source converter enters C source files that contain code for CA850, and outputs files containing code for CX.
Header file	.h	I/O	The source converter enters CA850 header files and outputs files containing code for CX.
Assembler source file	.s	I/O	The source converter enters CA850 assembler source files and outputs files containing code for CX.
Include file	.inc	I/O	The source converter enters CA850 include files and outputs files containing code for CX.
Conversion results file	free	O	This is a text file that the conversion results is output. This file is output when the -r option is specified.

B.3.2 Method for manipulating

This section explains how to manipulate the source converter.

With CubeSuite, you can perform this conversion processing when creating a new project with CX as the build tool by reusing the project with CA850 as the build tool.

See "CubeSuite Start" for details.

Caution The option cannot be set in CubeSuite.
Manipulate on the command line to set the option.

(1) Startup from the command line

Enter the following on the command line.

```
>cnv850[Δfile-name][Δoption]...
```

[ ]: Can be omitted
...: Pattern in proceeding [ ] can be repeated
Δ: One or more spaces

- When a file name (including a parameter of an option) is specified, it can include the path (absolute path or relative path).
  When a file name without the path or a relative path is specified, the reference point of the path is the current folder.
- When a file name (including a parameter of an option) includes a space (such as a path name), enclose the parameter in a pair of double quotation marks (" ").
- The length that can be specified for a file name (including option parameters) is 259 characters, including the path.
- Uppercase characters and lowercase characters are not distinguished for the alphabet of a file name.
- An error will occur if two or more file names are specified.
  When two or more files are specified as input, use the -l option.
- An error will occur if the specified file (including a parameter of an option) does not exist.
  However, an error will not occur if the file specified as the parameter of the -o option does not exist.
  If the file specified by the parameter of the -l option does not exist, a warning will be output and the processing of that file will be skipped.
- Uppercase characters and lowercase characters are distinguished for options.
- If the option that is not listed in "Table B-15. Source Convert Options" is specified, the specified option will be ignored without displaying a warning.

(2) Content displayed in conversion results

(a) Outline

The overview of conversion results is output to the standard error output.

The output format is shown below.

```
file-name
    result
```

file-name: This is the image of command line specification.

result: One of the following is displayed.

- If a location requiring conversion is detected

```
Converted(total-
number)(numberΔdeleted,ΔnumberΔinserted,ΔnumberΔchanged,ΔnumberΔinformation)
```

total-number: The total number of converted locations

number: The number of converted locations for each information type (deletion, insertion, change, and information)

- If no locations requiring conversion is detected

```
None
```

- When an error occurs

```
Error
```

Remark When two or more files are specified using the -l option, the outline will be displayed for each file.

(b) Detail

Details of the conversion results are displayed in the format below for each location requiring conversion. The detail is output to the standard output by default. When the -r option is specified, it is output to a file.

```
file-name(line-number):Δmessage-number:[information-type]Δcontents
```

file-name: This is the image of command line specification of the converted file name.

line-number: This is the number of the converted lines.

message-number: This is the number of output message.

information-type: One of the following is displayed.

Information Type	Meaning
Delete	The target line has been deleted.
Insert	The target line has been inserted.
Change	The target line has been changed.
Info	The user must convert the corresponding line manually.

contents: This is the contents of output message.

B.3.3 Option

The types and explanations for options of the source converter are shown below.

Table B-15. Source Convert Options

Classification	Option	Description
Version/help display specification	-V	This option displays the version information of the source converter.
	-h	This option displays the descriptions of the source converter options.
Character code specification	-c	This option specifies the Japanese character code.
List file specification	-l	This option specifies the list file that I/O file names are described.
Output file specification	-o	This option specifies the output file name.
Conversion results file specification	-r	This option specifies the conversion results file name.
File type specification	-t	This option specifies the type of the input file.

Version/help display specification

The version/help display specification options are as follows.

- -V
- -h

-V

This option displays the version information of the source converter.

[Specification format]

```
-V
```

- Interpretation when omitted
  Conversion is performed without displaying the version information of the source converter.

[Detailed description]

- This option outputs the version information of the source converter to the standard error output.
  It does not execute conversion.
- If this option and other option are specified at the same time, other option will be ignored without displaying a warning.

[Example of use]

- To output the version information of the source converter to the standard error output, describe as:

```
>cnv850 -V
```

-h

This option displays the descriptions of the source converter options.

[Specification format]

```
-h
```

- Interpretation when omitted
  The descriptions of the source converter options are not displayed.

[Detailed description]

- This option outputs the descriptions of the source converter options to the standard error output.
  It does not execute conversion.
- If this option and other option (except -V) are specified at the same time, other option will be ignored without displaying a warning.
- If this option and the -V option are specified at the same time, the -V option will be valid.

[Example of use]

- To output the descriptions of the source converter options to the standard error output, describe as:

```
>cnv850 -h
```

Character code specification

The character code specification option is as follows.

- -c

-c

This option specifies the Japanese character code.

[Specification format]

```
-c=code
```

- Interpretation when omitted
  The Japanese character code is handled as SJIS.

[Detailed description]

- This option specifies the character code to be used for Japanese comments and character strings in the input file.
- The items that can be specified as *code* are shown below.
  If any other item is specified, that will be ignored without displaying a warning.

none	Does not process the Japanese character code (process as ASCII)
euc	EUC (Japanese)
sjis	SJIS

- An error will occur if *code* is omitted.

[Example of use]

- To specify EUC as the character code to be used for Japanese comments and character strings in the input file, describe as:

```
>cnv850 main.c -c=euc
```

List file specification

The list file specification option is as follows.

- -l

-l

This option specifies the list file that I/O file names are described.

[Specification format]

```
-l=file
```

- Interpretation when omitted
  If a file name is specified on the command line, the file will be regarded as the input file.
  However, in this case, the -o option must be specified.
  If a file name is not specified on the command line, nothing will be performed.

[Detailed description]

- This option specifies *file* as the list file that input file names and output file names are described.
  The format of the list file is shown below.

```
[-tΔ][-cΔ]input-file-nameΔoutput-file-name
```

  - The input and output file names that can be specified are the same as those that can be specified on the command line.
    The rules for specification are also the same as for the command line.
  - If the input file does not exist, a warning will be output and the processing of that file will be skipped.
  - If the -c option is specified in the list file, then if it differs from the -c option and the parameter specified on the command line, a warning will be displayed, and the specification in the list file will be enabled.
  - Japanese character code is only allowed in the list file if it is encoded as UTF-8 (with BOM).
  - Only "CR + LF" is a valid line-break code.

- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.
- If this option is specified, the input file name specified on the command line will be ignored with displaying a warning.

[Example of use]

- To specify list file "io.lst", describe as:

```
>cnv850 -l=io.lst
```

Output file specification

The output file specification option is as follows.

- -o

-o

This option specifies the output file name.

[Specification format]

```
-o=file3
```

- Interpretation when omitted
  If a file name is specified on the command line, this option cannot be omitted.
  If this option is omitted, an error will occur (except when specifying the -h or -V option).

[Detailed description]

- This option specifies the name of the conversion output file as *file*.
- If *file* already exists, it will be overwritten without displaying a warning.
- An error will occur if *file* does not exist.
- An error will occur if *file* is omitted.

[Example of use]

- To output the converted file with name "main_cnv.c", describe as:

```
>cnv850 main.c -o=main_cnv.c
```

Conversion results file specification

The conversion results file specification option is as follows.

- -r

-r

This option specifies the conversion results file name.

[Specification format]

```
-r=file
```

- Interpretation when omitted
  The conversion results are output to the standard output.

[Detailed description]

- This option outputs the conversion results to file *file*.
- An error will occur if *file* is omitted.

[Example of use]

- To output the conversion results to file "result", describe as:

```
>cnv850 main.c -r=result
```

File type specification

The file type specification option is as follows.

- -t

-t

This option specifies the type of the input file.

[Specification format]

```
-t=type
```

- Interpretation when omitted
  The file type is determined by the extension of the input file.

[Detailed description]

- This option processes the input file as the type specified by *type*.
- The items that can be specified as *type* are shown below.
  If any other item is specified, that will be ignored without displaying a warning, and the input file is output as-is, without conversion.

c	C source file or C header file
asm	Assembler source file or assembler include file

- An error will occur if *type* is omitted.

[Example of use]

- To process the input file as a C source file, describe as:

```
>cnv850 sample.test -t=c
```

B.3.4 Conversion Specification

This section describes details of conversions in C source files and assembler source files.

(1) C source file

(a) Internal expression of data

In CA850, the double type is 32 bits, the same as the float type, but in CX a double is 64 bits.

In CA850 source, type declarations of "double" are converted to "float".

CA850 source	`double`
After conversion for CX	`float`

(b) Device specification

Although you can specify devices in C source in CA850, you cannot do so in CX. In CX, device-specification directives are deleted.

To specify a device in CX, you must specify the -C option.

CA850 source	`#pragma cpu` *`device-nane`*
After conversion for CX	

(c) Data section allocations

The data section-specification format for CA850 is shown below.

```
#pragma section section-type ["section-name"] begin
Variable declarations/definitions
#pragma section section-type ["section-name"] end
```

The data section-specification format for CX is shown below.

```
#pragma section attribute-specification-char ["section-name"]
Variable declarations/definitions
```

Section types and attribute specification characters are the same, and only the format differs, so they are converted to CX format. Since the end specification line is not needed, it is deleted.

CA850 source	`#pragma section` *`section-type`* `"`*`section-name`*`" begin`
	`#pragma section` *`section-type`* `"`*`section-name`*`" end`

After conversion for CX	`#pragma section` *`attribute-specification-char`* `"`*`section-name`*`"`
	`#pragma section default`

(d) Access to peripheral function registers using register names

<1> How to access

The "#pragma ioreg" format for using register names is the same in CA850 and CX, so no conversion is performed.

<2> Bit access

I/O register bit accesses in CA850 have been deleted in CX. In order to replace this, it is replaced by bit-field type declarations and macros.

If there is bit access in the I/O register, then a type declaration and macro are output at the top of the file, and the access portion is changed to a macro call.

Since the type of the I/O register is unknown to the source converter, an 8, 16, or 32-bit field is created depending on the position of the bit. In the example below, 3 and 5 fit into 8 bits, so an 8-bit field is created.

CA850 source	`i = PAHL.3;`
	`PAHL.5 = 0;`

After conversion for CX	`#ifndef __BIT8`
	`typedef struct {`
	`unsigned int b0:1;`
	`unsigned int b1:1;`
	`unsigned int b2:1;`
	`unsigned int b3:1;`
	`unsigned int b4:1;`
	`unsigned int b5:1;`
	`unsigned int b6:1;`
	`unsigned int b7:1;`
	`} __Bits8;`
	
	`#define __BIT8(name,bit)(((__Bits8*)&name)->b##bit)`
	`#endif`
	
	`i = __BIT8(PAHL,3);`
	`__BIT8(PAHL,5) = 0;`

(e) Assembler instruction format

No conversion is performed, because CA850 and CX use the same assembler instruction format.

When there are differences, however, as with pseudo instructions, the user must perform the conversion manually.

(f) Specification of inline expansion

No conversion is performed, because CA850 and CX use the same inline-expansion specification format.

(g) Function specification of name-specified sections

No conversion is performed, because CA850 and CX use the same method for specifying functions to name-specified sections.

(h) Interrupt level settings

No conversion is performed, because CA850 and CX use the same interrupt-level settings.

(i) Coding format to disable interrupts

No conversion is performed, because CA850 and CX use the same method to disable interrupts.

(j) Coding format for interrupt/exception handlers

The format for interrupt handlers in CA850 is shown below.

```
#pragma interrupt interrupt-request-name function-name [allocation-method]
__interrupt function-definition or function-declaration
__multi_interrupt function-definition or function-declaration
```

The format for interrupt handlers in CX is shown below.

```
#pragma interrupt interrupt-request-name function-name [allocation-method] [option]
```

The function specifiers __interrupt and __multi_interrupt have been discontinued, and multiplex interrupts are specified via an option. For this reason, these specifiers are deleted.
"#pramga interrupt" is converted to the CX specification.If the function has the function specifier __multi_interrupt, then the "multi" option will be set in #pragma interrupt.
If there is no #pragma interrupt declaration, and only the function specifier has been coded, then "#pragma interrupt NO_VECT" is output.

CA850 source	1. #pragma interrupt *interrupt-request-name function-name allocation-method* __interrupt *function-definition* 2. #pragma interrupt *interrupt-request-name function-name allocation-method* __interrupt_interrupt *function-definition* 3. __interrupt *function-definition*
After conversion for CX	1. #pragma interrupt *interrupt-request-name function-name allocation-method* *Function-definition* 2. #pragma interrupt *interrupt-request-name function-name allocation-method* multi *Function-definition* 3. #pragma interrupt NO_VECT *function-name* *Function-definition*

Note that with CA850, the user needed to code an ei instruction inside multiplex interrupt functions, but CX now automatically generates an ei instruction in the function prologue. This must be changed manually by the user.
Additionally, in #define statements, __interrupt and __multi_interrupt in macro definitions cannot be converted, so the user must do this manually.

(k) RTOS supported functions

No conversion is performed, because CA850 and CX use the same RTOS supported function format.

(l) Coding format using embedded-function instructions

No conversion is performed, because CA850 and CX embedded-function instructions are upward compatible.

(m) Structure packing function

No conversion is performed, because CA850 and CX use the same structure-packing function format.

(n) Calls to __rcopy function

The ROMization processing of CA850 and CX differs. For this reason, if a RAM area data copy function (_rcopy / _rcopy1 / _rcopy2 / _rcopy4) is called in a C source file (e.g. the main function), an error will occur in CX. A warning is therefore displayed for locations where these copy functions are referenced (no conversion is performed).

(2) Assembler source file

The conversion specification for assembler source files is shown below.

Table B-16. Conversion Specification for Assembler Source Files

Before Conversion	After Conversion	Information Type	Remark
`;`	Line break	Change	Multi statement
`--`	`;`	Change	Comment
`hi1`	`highw1`	Change	
`hi`	`highw`	Change	
`lo`	`loww`	Change	
`.text`	`.cseg text`	Change	
`.const`	`.cseg const`	Change	
`.sconst`	`.cseg sconst`	Change	
`.bss`	`.dseg bss`	Change	
`.data`	`.dseg data`	Change	
`.previous`	Not converted	Information	Even if the target line is deleted, the section cannot be switched correctly. For this reason, a message to change the section is output.
`.sbss`	`.dseg sbss`	Change	
`.sdata`	`.dseg sdata`	Change	
`.sebss`	`.dseg sebss`	Change	
`.sedata`	`.dseg sedata`	Change	
`.sibss`	`.dseg sibss`	Change	
`.sidata`	`.dseg sidata`	Change	
`.tibss`	`.dseg tibss`	Change	
`.tibss.byte`	`.dseg tibss.byte`	Change	
`.tibss.word`	`.dseg tibss.word`	Change	
`.tidata`	`.dseg tidata`	Change	

Before Conversion	After Conversion	Information Type	Remark
`.tidata.byte`	`.dseg tidata.byte`	Change	
`.tidata.word`	`.dseg tidata.word`	Change	
`.section ".pro_epi_runtime", text`	`".pro_epi_runtime" .cseg text`	Change	
`.section ".text", text`	`.cseg text`	Change	
`.section ".data", data`	`.dseg data`	Change	
`.section ".sedata", data`	`.dseg sedata`	Change	
`.section ".sidata", data`	`.dseg sidata`	Change	
`.section ".tidata", data`	`.dseg tidata`	Change	
`.section ".tidata.byte", data`	`.dseg tidata.byte`	Change	
`.section ".tidata.word", data`	`.dseg tidata.word`	Change	
`.section ".bss", bss`	`.dseg bss`	Change	
`.section ".sebss", bss`	`.dseg sebss`	Change	
`.section ".sibss", bss`	`.dseg sibss`	Change	
`.section ".tibss", bss`	`.dseg tibss`	Change	
`.section ".tibss.byte", bss`	`.dseg tibss.byte`	Change	
`.section ".tibss.word", bss`	`.dseg tibss.word`	Change	
`.section ".sdata", sdata`	`.dseg sdata`	Change	
`.section ".sbss", sbss`	`.dseg sbss`	Change	
`.section ".const", const`	`.cseg const`	Change	
`.section ".sconst", const`	`.cseg sconst`	Change	
`.section ".version", comment`	`.version .vseg`	Change	
`.section "name", text`	`"name" .cseg text`	Change	"name" is the name of an arbitrary section.
`.section "name", const`	`"name" .cseg const`	Change	As above
`.section "name", bss`	`"name" .dseg bss`	Change	As above
`.section "name", data`	`"name" .dseg data`	Change	As above
`.section "name", sbss`	`"name" .dseg sbss`	Change	As above
`.section "name", sdata`	`"name" .dseg sdata`	Change	As above
`.section "name", comment`	`"name" .vseg`	Change	As above
`.section "name"`	`.cseg text`	Change	This is the case when no allocation attribute is specified.
`.vdbstrtab`	Not converted	Information	Even if the target line is deleted, the section's code cannot be deleted. For this reason, a message to delete the section itself is output.
`.vdebug`	Not converted	Information	As above

Before Conversion	After Conversion	Information Type	Remark
.vline	Not converted	Information	As above
.ext_ent_size	$ext_ent_size	Change	
.ext_func	$ext_func	Change	
.file	Not converted	Not applicable	
.frame	.func	Change	Sets the stack size to 0.
.set	.set	Change	Changes the symbol specification method.
.size	Delete	Delete	
.align	Not converted	Not applicable	
.org	Not converted	Not applicable	
.byte val	.db val	Change	
.byte bit : val	Not converted	Information	
.float	Not converted	Not applicable	
.hword val	.dhw val	Change	
.hword bit : val	Not converted	Information	
.lcomm label, size, byte	.align byte label: .ds (size)	Added	
.shword val	.dshw val	Change	
.shword bit : val	Not converted	Information	
.space	.ds	Change	
.str	.db	Change	
.word val	.dw val	Change	
.word bit : val	Not converted	Information	
.comm	Not converted	Not applicable	
.extern	Not converted	Not applicable	
.globl	.public	Change	
.binclude	$binclude	Change	
.include	$include	Change	
.irepeat	.irp	Change	
.repeat	.rept	Change	
.else	$else	Change	
.elseif	$elseif	Change	
.elseifn	$elseifn	Change	
.endif	$endif	Change	
.if	$if	Change	
.ifdef	$ifdef	Change	
.ifn	$ifn	Change	

Before Conversion	After Conversion	Information Type	Remark
`.ifndef`	`$ifndef`	Change	
`.exitm`	Not converted	Not applicable	
`.exitma`	Not converted	Not applicable	
`.endm`	Not converted	Not applicable	
`.local`	Not converted	Not applicable	
`.macro`	`.macro`	Change	Changes the method of specifying the macro name and formal parameters.
`.option asm`	Delete	Delete	
`.option az_info_j`	Delete	Delete	
`.option az_info_r`	Delete	Delete	
`.option az_info_ri`	Delete	Delete	
`.option c`	Delete	Delete	
`.option callt`	`$callt`	Change	
`.option cpu`	`$processor`	Change	
`.option data`	`$data`	Change	
`.option ep_label`	`$ep_label`	Change	
`.option macro`	`$macro`	Change	
`.option mask_reg`	Delete	Delete	
`.option new_fcall`	Delete	Delete	
`.option no_ep_label`	`$no_ep_label`	Change	
`.option nomacro`	`$nomacro`	Change	
`.option nooptimize`	Delete	Delete	
`.option novolatile`	Delete	Delete	
`.option nowarning`	`$nowarning`	Change	
`.option optimize`	Delete	Delete	
`.option sdata`	`$sdata`	Delete	
`.option volatile`	Delete	Delete	
`.option warning`	`$warning`	Change	
`.option reg_mode 5 5`	`$reg_mode 22`	Change	
`.option reg_mode 7 7`	`$reg_mode 26`	Change	
`.option reg_mode 10 10`	`$reg_mode 32`	Change	
`.option reg_mode x x`	Not converted	Information	

APPENDIX C INDEX

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Oct 01, 2010	**-**	First Edition issued

CubeSuite Ver.1.40 User’s Manual: Build (CX Compiler)

Publication Date: Rev.1.00 Oct 1, 2010

Published by: Renesas Electronics Corporation

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

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

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

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

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

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

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

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

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

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

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

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

Colophon 1.0

CubeSuite Ver.1.40

Renesas Electronics Corporation