Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010 Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

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.
- 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. 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; anticrime 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 majorityowned subsidiaries.
- (Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



RX Family C/C++ Compiler Package

Application Notes: Compiler Usage Guide, Option Edition

This document explains the options for C/C++ Compiler V.1 for the RX family.

Table of contents

T1. Optimization options.		2
	zing on speed/size)	
	Insion	
	nization	
1.2 Detailed options to im	prove performance	7
1.2.1 External variable acce	ess optimization	8
1.2.20ptimization range sp	plitting	
1.2.3Enumeration type size	e	
1.2.4Switch statement exp	ansion method	14
2. Convenient options		
 Convenient options Debugging information 	n output mode	
2.1 Debugging informatio	n output mode	
2.1 Debugging information2.2 Pre-processor expansion	n output mode sion ttilization	
2.1 Debugging informatio2.2 Pre-processor expansion2.3 External variable vola	n output mode sion	
2.1 Debugging informatio2.2 Pre-processor expansion2.3 External variable vola2.4 Specifying bit field or	n output mode sion itilization	
2.1 Debugging informatio2.2 Pre-processor expansion2.3 External variable vola2.4 Specifying bit field or	n output mode sion tilization der	

1. Optimization options

The optimization options for compilers include two basic options that optimize on speed or size, and detailed options that set further details for each optimization item. 1.1 explains the basic options and related detailed options, and 1.2 explains detailed options that can increase performance in particular.

Note that the expansion code in assembly code as used in this document can be obtained by specifying "output=src" and "cpu=rx600".

When the "cpu" option is different, the expansion code in assembly language may also differ. Also, the expansion code in assembly language may change due to subsequent compiler improvements, so use this as a reference.

1.1 Basic options (optimizing on speed/size)

The compiler optimizations include those that reduce object size, and those that speed up execution.

To optimize on execution speed, specify the "speed" option. To optimize on size, specify the "size" option (default). The following optimizations are performed when each option is specified:

- When "speed" is specified
- Optimizations effective for size and execution speed, and optimizations that increase execution speed but may increase size • When "size" is specified

Optimizations effective for size and execution speed, and optimizations that decrease size but may decrease execution speed

Ideally, functions that demand execution speed and functions that demand size can be split into separate files, with size optimizations and speed optimizations selected for each file.

Format

```
speed
size : Default
```

[How to specify this option in High-Performance Embedded Workshop (herein as the Renesas IDE)]

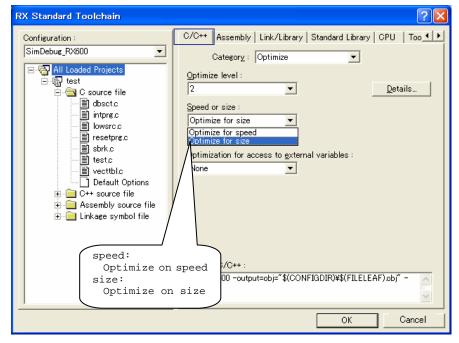


Figure1-1



Note 1:

The execution speed on production machines may change depending on causes unrelated to code generated by the compiler, such as memory architecture and interrupts. Therefore, specifying "speed" does not necessarily result in the fastest code. Confirm the effects of the various options presented in this document on the production machines.

Note 2:

The detailed options for setting details for the items optimized by the compiler include items for which the default values change depending on how the basic options are set. Table 1-1 shows the options for which the default values change.

No	Functio			Size	Speed				Reference
		optimize	max	2	0 or 1	max	2	0 or 1	
1	Automatic inline expa	ansion	inline=0	noinline	noinline	inline=500	inline=100	noinline	1.1.1
2	Loop expansion optir	nization	loop=1	loop=1	loop=1	loop=32	loop=2	loop=1	1.1.2

Table 1-1	List of	defaults
-----------	---------	----------

The following explains each option.

1.1.1 Automatic inline expansion

An option can be used to specify whether automatic inline expansion is performed for functions.

When the "inline" option is specified, automatic inline expansion is performed. "inline=<*number>*" can be used to specify that inline expansion is to be performed until the program size grows by a given percentage. For example, if "inline=50" is specified, inline expansion is performed until the program size grows by 50% (reaching 1.5 times its original size). If "inline=0" is specified, inline expansion is performed.

Automatic inline expansion prioritizes called functions by the smallest first.

Note that functions for which #pragma inline is specified are expanded inline regardless of the specification for automatic inline expansion. However, keep in mind that the size increase from inline expansion due to #pragma inline is included when the maximum size for automatic inline expansion is determined.

When the "noinline" option is specified, automatic inline expansion is not performed.

Keep in mind that inline expansion is not performed for functions if one of the following conditions is satisfied

- (1) The function has variable parameters.
- (2) The function is called using the address of the expanded function.

For details about inline expansion, see 1.1 Specifying inline function expansion in C/C++ Compiler Package for the RX Family Application Notes: Compiler Usage Guide, Extended Functionality Edition.

Format

inline [= <i><number></number></i>]	: Default when speed is specified, with a default number of 100
noinline	: Default when size is specified



[How to specify this option in the Renesas IDE]

RX Standard Toolchain	? 🗙
Configuration : SimDebug_RX600	C/C++ Assembly Link/Library Standard Library CPU Too Category: Optimize Optimize level: 2 Speed or size: Optimize for size Inter-module optimization Optimization for access to external variables: None
	Options C/C++ : -cpu=rx600 -output=obj="\$(CONFIGDIR)¥\$(FILELEAF).obj" - nologo
	OK Cancel

Figure1-2

N	Optimize details	
\square	Inline Global variables Miscellaneou	IS .
V	Automatic inline expansion : Custom	Automatic inline expansion: custom Specify the inline expansion size
	100	
	Inline file path :	
		<u>A</u> dd
		Remove
		OKCancel

Figure1-3



Inline expansion requires that the definitions of functions expanded at compile time be able to be referenced. Therefore, the function without the function definition that can be referred from the file to be compiled cannot be singled out for normal inline expansion. To progress the function that there is a definition in another file, specify the inter-file inline expansion (file_inline=<*file-name*>[,...]) option. Note that operation is not guaranteed if an extern function of the same name is defined in multiple files specified inline, across the files (a single chosen function definition is used).

Note that if the same file as the source file is specified for the inline expansion files, the following warning is output by the compiler.

C1315 (W) File_inline "file name" ignored by same file as source file

[How to specify this option in the Renesas IDE]

Select the "Compiler" tab, select "Optimization" for Category, choose Details (Figure 1-2), and then perform the following settings in the displayed dialog box.

Optimize details	? 🛛	
Inline Global variables Miscellaneous Automatic inline gxpansion : Custom 100 Inline file path : \$(PROJDIR)\#bc	Add <u>R</u> emove	Specify the inline expansion file
ОК	Cancel	

Figure1-4

Example:

```
Source code
<a.c>
void func(void)
{
   g();
}
<b.c>
#pragma inline (g)
void g(void)
{
   h();
}
a.c expansion when file_inline=b.c is specified
void func(void)
{
   h();
```



When an inter-file inline expansion file outside the current folder is specified, the inter-file inline expansion folder can be specified (file_inline_path=<*path-name*>[,...]) so that the path name for inter-file inline expansion files can be omitted.

Files subject to inter-file inline expansion are first searched for in the folder specified by the "file_inline_path" option, and then in the current folder.

[How to specify this option in the Renesas IDE]

RX Standard Toolchain	? 🛛
Configuration : SimDebug_RX600	C/C++ Assembly Link/Library Standard Library CPU Too ↓ Category : Source ▼ Show entries for : File inline path ▼ \$(PROJDIR) Add Insert Remove Move gp Move down
	Options C/C++ : -cpu=rx600 -file inline_path="\$(PROJDIR)" - output=obj="\$(CONFIGDIR)¥\$(FILELEAF).obj" -nologo OK Cancel

Figure1-5



1.1.2 Loop expansion optimization

An option can be used to specify whether loop expansion is performed.

When the "loop" option is specified, loop expansion is performed.

The maximum expansion count during loop expansion can be specified using "loop=<number-from-1-to-32>".

Format

loop[=<number>]

: Default when speed is specified is 2 : Default when size is specified is 1

[How to specify this option in the Renesas IDE]

To specify the maximum expansion count during loop expansion, select the "Compiler" tab, select "Optimization" for Category, choose Details (Figure 1-2), and then perform the following settings in the displayed dialog box.

Optimize details		? 🛛	
Inline Global variables Miscellaneous	Library <u>f</u> unction :		
Custom	Inline expansion Schedule instructions : Default	• •	
Switch statement : Auto	Divide the optimization range	ge: ▼	
Division by constant : Default	Use floa oint arithmeti instructions Default	•	
Approximate a floating-point constar			
Enable register declaration Not check the range in conversion be integer	etween floating point cou	e maximum Int durin	s custom, expansion g loop an be set.
	ОК	Cancel	

Figure1-6

1.2 Detailed options to improve performance

This chapter explains the options that can improve performance in particular, of the detailed operations used to set details for each optimization item.

No	Functionality	Option	Size	Execution speed	Reference
1	External variable access optimization	map/smap	Improves significantly	Improves significantly	1.2.1
2	Optimization range splitting	scope/noscope	May improve or worsen	May improve or worsen	1.2.2
3	Enumeration type size	auto_enum	Improves	Worsens	1.2.3
4	Switch statement expansion	case	May improve or worsen	May improve or worsen	1.2.4



1.2.1 External variable access optimization

When the "map" option and "smap" option are used, access to external variables is performed relatively from the base external variable. This removes the need for load processing for external variable addresses, which improves execution speed. Also, since address value load processing can be omitted, the program size can be reduced. External variable access optimization can improve both speed and size.

The "map" optimizations are performed using the external symbol allocation information generated by the optimization linkage editor. Therefore, compilation needs to be performed twice. (In the HEW environment, it is executed automatically.)

The "smap" optimizations perform external variable optimization only for external variables defined in compiled files. Since the external symbol allocation information generated by the linkage editor is not used, optimization can be performed with only one compilation.

The "map" optimizations are more effective at optimizing than the "smap" optimization, but since they require compilation twice, they increase the build time. "smap" optimizations are limited to external variables defined in files, but are performed in a single compilation, and can be performed when files such as library files are generated without address resolution.

Option	Build time	External symbol allocation information file generated by the optimization linkage editor	Optimization effect	Address resolution
map	Long	Required	Better than smap	Required
smap	Short	Not required	Worse than map	Not required

Table 1-3 Characteristics of map and smap

Format

Inter-module specification

map [= <*file-name*>]

Usage differs depending on the output option.

[For output=abs or mot] Specify only the "map" option.

[For output=obj or src]

Perform compilation once without specifying the "map" option, specify "map=<*file-name*>" during linkage, and create an external symbol allocation information file. When compiling the second time, specify the external symbol allocation information file (map=<*file-name*>).

When the definition order of external variables or static variables changes, the external symbol allocation information file will need to be regenerated.

Operation when a second compilation is performed is not guaranteed under the following conditions:

- When the specified option is anything other than the option specified for the first compilation and the "map" option

- When a source file different than that specified for the first compilation is specified

Intra-module specification

smap



[How to specify this option in the Renesas IDE]

When the setting for external variable access optimization is changed from [Inter-module] to anything else, or to [Inter-module] from anything else, a warning is displayed. This is because output of external symbol allocation information files by the linkage editor is automatically enabled/disabled.

RX Standard Toolchain	? 🗙
Configuration : SimDebug_RX600	C/C++ Assembly Link/Library Standard Library CPU Too ▲ ▶ Qptimize ▼ ●
	OK Cancel

Figure1-7



Example 1:

The following shows how to perform relative access of consecutively allocated variables from the same register when the variable allocation order is considered.

anocation order is considere	u.				
Source code					
int a,b;					
void f(void)					
{					
a=0;					
b=0;					
}					
5					
Accembly expansion	code (without map/smap	! Accembly	expansion	ode (with	<pre>map/smap specified):</pre>
specified): Code size			e of 18 byte		map/smap specified/
_f:	; function: f	f:	C OF TO DYC		; function: f
STACK	_f=4		.STACK	f=4	/ runeeron: r
L10:	_r-1	. L10:	. DIACK	_1-4	
MOV.L	#00000000H,R5		MOV.L	#000000	
MOV.L	#_a,R4		MOV.L MOV.L	#0000000 #_a,R4	JOH, КЗ
MOV.L MOV.L		:	MOV.L MOV.L		
	R5,[R4]			R5,[R4]	- 4 1
MOV.L	#_b,R4		MOV.L	R5,04H[H	<u></u>
MOV.L	R5,[R4]		RTS		AT TON 4
RTS		:	.SECTION	B,DATA,	ALIGN=4
.SECTION	B,DATA,ALIGN=4		.glb	_a	
.glb	_a	_a:		-	; static: a
_a:	; static: a	:	.blkl	1	
.blkl	1	:	.glb	_b	
.glb	_b	b:			; static: b
_p:	; static: b	•	.blkl	1	
.blkl	1	:			



1.2.2 Optimization range splitting

An option can be used to specify whether the function optimization range is split into multiple ranges and compiled or compiled without splitting.

When "scope" is specified, large functions may be split into optimization ranges and then compiled.

When "noscope" is specified and no optimization range splitting is performed, since optimization can be performed across the whole function, the size and execution speed will usually both improve. However, if the registers are not sufficient, performance may actually degrade.

Since this option may cause program code efficiency to either improve or worsen, try both options when tuning performance. Keep in mind that when compilation is performed without splitting the optimization range, compilation will take longer.

Format

scope	: Default when optimize=0 1 2 is specified
noscope	: Default when optimize=max is specified

Whether the optimization range has been split can be checked using information level messages, which can be enabled by specifying the "message" option.

If the optimization range is split, the following message is displayed.

C0101 (I) Optimizing range divided in function "function-name"

[How to specify this option in the Renesas IDE]

Select the "Compiler" tab, select "Optimization" for Category, choose Details (Figure 1-2), and then perform the following settings in the displayed dialog box.

Optimize details	? 🛛
Inline Global variables Miscellaneou	s
Loop expansion :	Library <u>f</u> unction :
Default 2	Inline expansion
	Default
Switch statement :	Divide the optimization range :
Auto	Disable
Division by constant :	Use floating point arithmetic instructions :
Default 👤	Default
Approximate a floating-point const	ant division
Enable register declaration	
Not check the range in conversion integer	between floating point number and
	OK Cancel

Figure1-8



1.2.3 Enumeration type size

types that can be assigned to the enumerator.

Enumeration type data declared using enum is handled using the minimum type in which the enumeration value will fit. If the "auto_enum" option is unspecified, the enumeration type size is processed as the signed int type. If the "auto_enum" option is specified, the type handled changed based on the values that can be assigned to the enumerator. Table 1-4 shows the values and

Enumerator			
Minimum value	Maximum value	Туре	
-128	127	signed char	
0	255	unsigned char	
-32768	32767	signed short	
0	65535	unsigned short	
None of the above		signed int	

Table 1-4 Relationship of values and types that can be assigned for enumeration types

Because of the reduced size of data, size performance is improved. This is especially effective when used for many enum type variables or structure members.

Format

auto_enum

[How to specify this option in the Renesas IDE]

RX Standard Toolchain	? 🛛
Configuration : SimDebug_RX600	C/C++ Assembly Link/Library Standard Library CPU Too ◀ ▶ QPU: Endian : Little Fast interrupt vector register : None ✓ Base register : RQM : None ✓ Addregs : Ox00000000 : None ✓ Mone ✓ Addregs : Ox00000000 : None ✓
	OK Cancel

Figure1-9



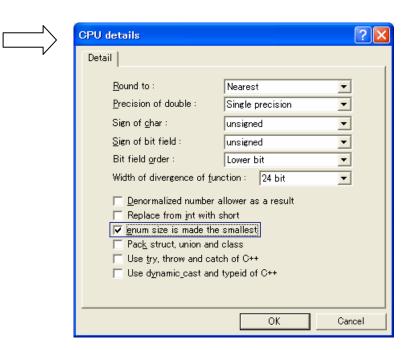


Figure1-10

Example:

Source code	
enum En {A_000 =0,A_001,A_002,A_003,A_END=255}; enum En x[3] = {A_000, A_001, A_END};	
Assembly expansion code (when auto_enum is	Assembly expansion code (when auto_enum is
unspecified)	specified)
_x: ; static: x	_x: ; static: x
.lword 0000000H, 0000001H,	.byte 00H, 01H,0FFH
00000FFH	



1.2.4 Switch statement expansion method

Processing to evaluate switch statements can be performed by selecting the "if-then method", which performs case value comparison, or the "Table method", which references a data table created from relative values of each case value. Sometimes, such as when there are few cases or the case value maximum and minimum are far apart, the "if-then method" is used even if "Table" method is selected to the "case" optional.

If the "case" option is not specified, the compiler will automatically select one of the following expansion methods:

- (1) If there are few cases or the case value maximum and minimum are far apart, the "if-then method" is used.
- (2) If (1) does not apply and the "case" option is specified, and "case" option specification is used.
- (3) If (1) and (2) do not apply, the "speed" option is specified, and around 10 or more case labels exist, the "Table method" is used.

During program execution, if a certain case value occurs often, execution speed is best when the corresponding case is coded first and the "if-then method" is specified. When execution does not skip to a given case value, execution speed is best when the "Table method" is specified.

Format

case = { ifthen | table | <u>auto</u>}

[How to specify this option in the Renesas IDE]

Select the "Compiler" tab, select "Optimization" for Category, choose Details (Figure 1-2), and then perform the following settings in the displayed dialog box.

Optimize details	? 🗙
Inline Global variables Miscellaneous]
Loop expansion :	Library <u>f</u> unction :
Default 2	Inline expansion Schedule instructions :
	Default
Switch statement :	Divide the optimization range :
Auto If then	Use floating point arithmetic
Table Detault	Default
Approximate a floating-point constan	nt division
Not check the range in conversion be integer	stween floating point number and
	OK Cancel

Figure1-11



2. Convenient options

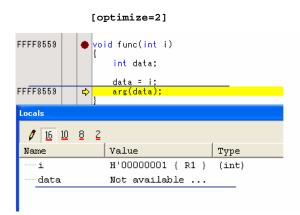
This chapter explains convenient options other than those to improve performance.

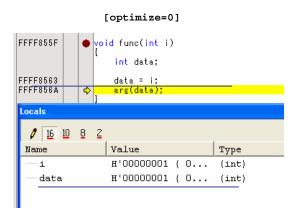
No	Functionality	Option	Reference
1	Debugging information output mode	optimize	2.1
2	Pre-processor expansion	preprocessor/noline	2.2
3	External variable volatilization	volatile	2.3
4	Specifying bit field order	bit_order	2.4
5	Specifying structure, shared structure, and class alignment count	pack	2.5

Table 2-1	List of conv	enient options

2.1 Debugging information output mode

When the "optimize=0" option is specified, information about local variables is always available to be referenced during debugging. Also, since optimizations related to per-statement deletion is completely suppressed, break points can be set for all each statements in the C source code. When this option is used to generate objects, object performance may suffer. We recommend using this temporarily during debugging.







Format

optimize = $\{ 0 | 1 | 2 | max \}$



[How to specify this option in the Renesas IDE]

RX Standard Toolchain	? 🛛
Configuration : SimDebug_FX600 C source file C source fil	C/C++ Assembly Link/Library Standard Library CPU Too Category: Optimize Optimize level: 2 Details 1 2 Max Inter-module optimization Optimization for access to external variables: None Optimization for access to external variables: None Optimise C/C++ :
	-cpu=rx600 -file inline path="\$(PROJDIR)" - output=obj="\$(CONFIGDIR)¥\$(FILELEAF) obj" -nologo

Figure2-2



2.2 Pre-processor expansion

This outputs the source program after pre-processor expansion. The code after pre-processor expansion is that after #include and #define are replaced. Since header file and other information has already been expanded in this file, it can be compiled on its own.

If *<file-name>* is not specified, a file with the extension p (when the input source file is a C program) or pp (when the input source program is a C++ program) is created, with the same name as the source file.

When "noline" is specified, #line output is suppressed during pre-processor expansion.

Format

output=prep [= <*file-name*>] noline

[How to specify this option in the Renesas IDE]

RX Standard Toolchain	? 🛛
Configuration : SimDebug_RX600	C/C++ Assembly Link/Library Standard Library CPU Too ↓ ▶ Qutput file type :
<u>)</u>	OK Cancel

Figure2-3

Example:

```
Source code
#define NUM 1
#define MESSAGE(num, name) {num, __DATE__, #name}
struct {
   int num ;
char* date ;
  char* string;
} data[] = {
       MESSAGE(NUM, aaaa),
};
Pre-processor expansion
#line 1 "C:¥¥Workspace_Evaluation_RX¥¥test1¥¥test1¥¥a.c"
struct {
   int
        num ;
   char* date ;
   char* string;
 data[] = {
}
       {1, "Jul 22 2009", "aaaa"},
```



2.3 External variable volatilization

When C source code is statically analyzed during compiler optimization processing, variable access order and count may be optimized as long as the result is not changed. However, when this optimization is performed on variables used in access to the I/O register or interrupt processing, operation may not be performed as intended. In this case, volatile declaration needs to be performed for such variables. When volatile declaration is performed, the access order and access count is performed as specified in the C source code.

The volatile declaration should be specified for variables deemed necessary, but sometimes it is difficult to check individual variables due to the misuse of past assets. In this case, try specifying "volatile". "volatile" can be specified to treat all external variables as having the volatile declaration.

Format

volatile novolatile

[How to specify this option in the Renesas IDE]

Select the "Compiler" tab, select "Optimization" for Category, choose Details (Figure 1-2), and then perform the following settings in the displayed dialog box.

Optimiz	ize details ?		
Inline	Global variables Miscellaneous		
🔽 🗹	reat global variables as volatile qualified		
	Propagate variables which are const qualified :		
Defa	ault		
	OK Cancel		

Figure2-4



Example:

Source co	ode					
int var	;;					
void fund	c(void)		1			
{			1			
var =	1;					
var =	0;					
}			1			
			1			
	hilinin (albana			+		
Optimization (when novolatile is specified)			Optimiza	tion (when	volatile is specified)	
int var	. •		int vai	~ ·		
void fund						
r voia rund	C(VOIQ)		l voia iun	void func(void)		
i var =	0.		1	var = 1;		
Var =	07			var = 1; var = 0;		
j	}			- 07		
			ſ			
Assembly	expansion (code (when novolatile is	Assembly	expansionco	de (when volatile is specified)	
specified	d)		:_func:		; function: func	
_func:		; function: func	-	.STACK	_func=4	
	.STACK	_func=4	L10:			
L10:			:	MOV.L	#_var,R4	
	MOV.L	#_var,R4	-	MOV.L	#0000001H,[R4]	
	MOV.L	#0000000H,[R4]	-	MOV.L	#00000000H,[R4]	
	RTS		:	RTS		
	.SECTION	B, DATA, ALIGN=4	:	.SECTION	B,DATA,ALIGN=4	
	.glb	_var	-	.glb	_var	
_var:		; static: var	_var:		; static: var	
	.blkl	1	-	.blkl	1	
			-			
			-			



2.4 Specifying bit field order

The sort order for bit fields can be changed. The ordering rules for bit fields may differ depending on the microcomputer. This functionality can be used to increase the portability for programs running on other microcomputers.

If "bit_order=left" is specified, members are allocated from the higher-order bits.

If "bit_order=right" is specified, members are allocated from the lower-order bits.

The bit field order can be specified even when #pragma bit_order is specified. When both the option and #pragma are specified at the same time, the #pragma specification takes precedence.

For details about this functionality, see 2.2 Specifying bit field order in C/C++ Compiler Package for the RX Family Application Notes: Compiler Usage Guide, Extended Functionality Edition.

Format

bit_order = { left | right }

[How to specify this option in the Renesas IDE]

Select the "CPU" tab, choose Details (Figure 1-9), and then perform the following settings in the displayed dialog box.

CPU details		?×
Detail		
<u>R</u> ound to : <u>P</u> recision of double : Sign of <u>c</u> har : <u>S</u> ign of bit field : Bit field <u>o</u> rder : Width of divergence of fur	Nearest	
Denormalized number allower bit Denormalized number allower as a result Replace from int with short enum size is made the smallest Pack struct, union and class Use try, throw and catch of C++ Use dynamic_cast and typeid of C++		
	OK Car	icel

Figure2-5



2.5 Specifying structure, shared structure, and class alignment count

Sometimes, the creation of free space needs to be avoided in structures (including shared structures or classes) used for communication programs. In such cases, "pack" can be specified as an option to set the alignment count of structure members to 1. This prevents free space from being created in structures with an alignment count of 1.

#pragma pack can also be used to set the structure alignment count to 1. When both this option and #pragma are specified at the same time, the #pragma specification takes precedence.

For details about this functionality, see 2.3 Specifying structure, shared structure, and class alignment count C/C++ Compiler Package for the RX Family Application Notes: Compiler Usage Guide, Extended Functionality Edition.

Format

pack unpack

[How to specify this option in the Renesas IDE]

Select the "CPU" tab, choose Details (Figure 1-9), and then perform the following settings in the displayed dialog box.

CPU details	?		
Detail			
, <u>R</u> ound to: <u>P</u> recision of double: Sign of <u>c</u> har: <u>S</u> ign of bit field: Bit field <u>o</u> rder:	Nearest		
Width of divergence of function : 24 bit Denormalized number allower as a result Replace from int with short enum size is made the smallest Pack struct, union and class Use try, throw and catch of C++ Use dynamic_cast and typeid of C++			
	OK Cancel		

Figure2-6



Web site and support <website and support>

Web site for Renesas Technology

http://japan.renesas.com/

Contact information

http://japan.renesas.com/inquiry

csc@renesas.com

Revision history<revision history,rh>

		Contents changed		
Rev.	Date issued	Page	Details	
1.00	2009.10.1		Initial edition	