RL78ファミリ用 Cコンパイラ CC-RL コーディングテクニック

CC-RL V1.02.00

2015年12月24日　R20UT3569JJ0100

ツールビジネス事業本部　ツール技術部
ルネサス システムデザイン株式会社

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

アジェンダ

- はじめに　ページ　3
- コーディング・テクニック　ページ　4
- メモリモデル　ページ 16
- 変数／関数情報ファイルの利用　ページ 19

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

はじめに

本資料は、CC-RL Cコンパイラを使用して、最適化オプションを指定した後にコード・サイズをさらに小さくする、または、実行速度を高速化するためのコーディング・テクニックについて説明します。

なお、各項目の例に記述された削減量はその例に関するものであり、他に適用した場合の削減量は個々のケースにより若干異なります。

本資料で記載してある例の出力アセンブラは、ミディアム・モデル、サイズ優先最適化（-Osize）を指定してコンパイルしたものです。その他の最適化（デフォルトの最適化およびスピード優先最適化など）を指定した場合には、結果が異なりますので、注意してください。

本資料は、次のツール、バージョンで説明しています。

- RL78ファミリ用Cコンパイラ　CC-RL　V1.02.00
- 統合開発環境 e² studio V4.2.0.012
- 統合開発環境 CS+ V3.03.00

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

コーディング・テクニック

© 2015 Renesas System Design Co., Ltd. All rights reserved.

コーディングテクニックによる効果

コーディングテクニック適用による、出力コードのコードサイズ、実行速度への効果

項目	コードサイズへの効果	実行速度への効果
変数のサイズ	○	○
符号なし変数	○	○
saddr領域	○	○
callt関数	○	×
構造体メンバのアライメント	○	△
ビットフィールドと1バイト変数	○	○

○：効果あり、△：効果なし、×：性能低下あり

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

変数のサイズ

変数は可能な限り小さいサイズの型を使用してください。

RL78ファミリが小さいサイズの型を得意としたデバイスであるためです。

（例）

- Cソースプログラム

変更前	変更後

void func(void)
{
　　signed int i;
　　for(i=0; i<10; i++)
　　　　__nop();
}
	void func(void)
	{
	　　signed char i;
	　　for(i=0; i<10; i++)
	　　　　__nop();
	}

- 出力アセンブラ

変更前		変更後	
movw ax, #0x000A	3	**mov a, #0x0A**	2
.BB@LABEL@1_1:		.BB@LABEL@1_1:	
nop	1	nop	1
addw ax, #0xFFFF	3	**dec a**	1
bnz $.BB@LABEL@1_1	2	bnz $.BB@LABEL@1_1	2
ret	1	ret	1
	10バイト		7バイト

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

符号なし変数

負数を扱わないデータは、全てunsignedを付けてください。

RL78ファミリが unsigned を得意としたデバイスであるためです。

（例）

- Cソースプログラム

変更前	変更後
signed int data0,data1; if(data0 > 10) data1++;	**unsigned** int data0,data1; if(data0 > 10) data1++;

- 出力アセンブラ

変更前		変更後	
movw ax, !LOWW(_data0)	3	movw ax, !LOWW(_data0)	3
xor a, #0x80	2		
cmpw ax, #0x800B	3	cmpw ax, #0x000B	3
skc	2	skc	2
incw !LOWW(_data1)	3	incw !LOWW(_data1)	3
	13バイト		11バイト

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

saddr領域（1/2）

使用頻度の高い外部変数および関数内static変数は、__saddr修飾子、#pragma saddr宣言等を使用してください。

saddr領域に割り当てることで、より良いコードとなります。

特に、1ビットのビットフィールドは__saddr修飾子、#pragma saddr宣言の効果が大きくなる傾向にあります。

変数／関数情報ファイルでもsaddr領域への変数の指定が可能です。

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

saddr領域（2/2）

（例）

- Cソースプログラム

変更前	変更後

typedef struct {
unsigned char b0:1;
unsigned char b1:1;
unsigned char b2:1;
unsigned char b3:1;
unsigned char b4:1;
unsigned char b5:1;
unsigned char b6:1;
unsigned char b7:1;
} BITF;
BITF data0, data1;
 	
data0.b4 = data1.b1;
	typedef struct {
	unsigned char b0:1;
	unsigned char b1:1;
	unsigned char b2:1;
	unsigned char b3:1;
	unsigned char b4:1;
	unsigned char b5:1;
	unsigned char b6:1;
	unsigned char b7:1;
	} BITF;
	__saddr BITF data0, data1;
	
	data0.b4 = data1.b1;

- 出力アセンブラ

変更前			変更後		
movw	**hl,#LOWW (_data1)**	3			
mov1	CY,[hl].1	2	mov1	CY,_data1.1	3
movw	**hl,#LOWW (_data0)**	3			
mov1	[hl].4,CY	2	mov1	_data0.4,CY	3
		10バイト			6バイト

© 2015 Renesas System Design Co., Ltd. All rights reserved.

callt関数（1/2）

関数呼出し頻度の高い関数は、__callt修飾子、#pragma callt宣言等を使用してください。

calltテーブル領域 [80H - BFH] にコールする関数のアドレスを格納し、直接関数をコールするよりも短いコードで関数をコールすることが可能です。

（例）

- Cソースプログラム

変更前	変更後

void func_sub(void)
{
;
}
void func()
{
func_sub();
;
func_sub();
}
	__callt void func_sub(void)
	{
	;
	}
	void func()
	{
	func_sub();
	;
	func_sub();
	}

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

callt関数（2/2）

（例）

- 出力アセンブラ

変更前		変更後	
		.SECTION .callt0,CALLT0 @_func_sub: .DB2 _func_sub	2
.SECTION .textf,TEXTF _func: call !!_func_sub	4	.SECTION .textf,TEXTF _func: callt [@_func_sub]	2
call !!_func_sub	4	callt [@_func_sub]	2
	8バイト		6バイト

注意

- 呼び出しのための関数のアドレスのテーブルを生成します（.callt0）。
- そのため、1回しか呼び出されない関数の場合には、コードサイズ削減の効果はありません。
- CALLT命令はCALL命令よりも実行クロック数は多くなります。
- 変数／関数情報ファイルでもCALLT命令で呼び出す関数の宣言の指定が可能です。

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

構造体メンバのアライメント（1/2）

RL78ファミリは奇数番地からワード・データのリード／ライトができないため、デフォルトオプションでは2バイト以上のメンバが偶数番地に配置されるようアライン・データを挿入します。

したがって、構造体のメンバはアライメントを考慮し、無駄な隙間を作らないように配置してください。

（例）

- Cソースプログラム

変更前	変更後
struct { signed char a; signed int b; signed char c; struct { signed int d; signed int e; } f; } data;	struct { signed char a; signed char c; signed int b; struct { signed int d; signed int e; } f; } data;

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

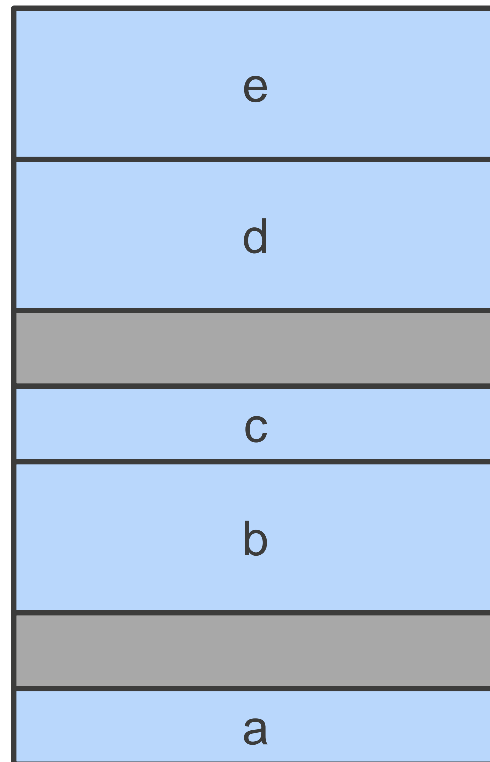
構造体メンバのアライメント（2/2）

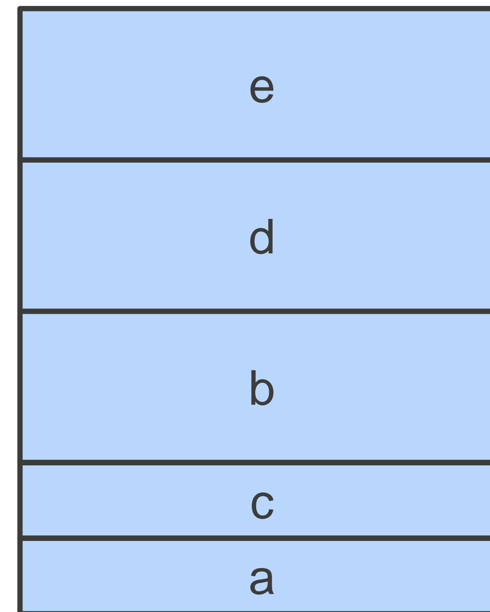
（例）

▪ メモリ配置

（アドレス上位）

（アドレス下位）





アラインデータ領域

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

ビットフィールドと1バイト変数（1/2）

ビットフィールドのメンバのサイズが2ビット以上の場合は、ビットフィールド（2bit以上）は使用せずに、char型に変更してください。

ただし、使用するRAMのメモリ領域サイズは増加します。

（例）

- Cソースプログラム

変更前	変更後

struct {
unsigned char b0:1;
unsigned char b1:2;
} data;
unsigned char dummy;
 	
if(**data.b1**){
dummy++;
}
	unsigned char data;
	
	unsigned char dummy;
	
	if(**data**){
	dummy++;
	}

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

ビットフィールドと1バイト変数（2/2）

（例）

- 出力アセンブラ

変更前		変更後	
mov a, #0x06	2	**cmp0 !LOWW(_data)**	3
and a, !LOWW(_data)	3		
sknz	2	sknz	2
ret	1	ret	1
inc !LOWW(_dummy)	3	inc !LOWW(_dummy)	3
ret	1	ret	1
12バイト		10バイト	

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

メモリモデル

© 2015 Renesas System Design Co., Ltd. All rights reserved.

メモリモデル（1/2）

RL78ファミリはデバイスの特徴として、

- プログラムのサイズが64Kバイト／それ以上、
- データ（ROMデータを含む)のサイズが64Kバイト／それ以上

で、各々関数呼出しのコードサイズ、データアクセスのコードサイズが異なります。

CC-RLでは、2つのメモリモデルがあります。

モデル	サイズ	関数	変数
スモールモデル	プログラム64Kバイト以下、データ64Kバイト以下	near	near
ミディアムモデル	プログラム64Kバイト以上、データ64Kバイト以下	far	near

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

メモリモデル（2/2）

プログラムが大きい場合は、ミディアムモデルを選択し、関数呼出し頻度の高い関数に__near修飾子を付加することで、コードサイズを削減できます。

ただし、__near修飾子、__far修飾子を付加した場合は、それらを取り扱うポインタ変数の型を合わせる必要があることもご注意ください。

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

変数／関数情報ファイルの利用

© 2015 Renesas System Design Co., Ltd. All rights reserved.
RENESAS

変数／関数情報ファイルの利用（1/3）

機能

- 参照頻度の高い変数をsaddr領域に割り当てます。
- 参照頻度の高い関数をcallt関数にします。
- ソース上で指定した修飾子（__saddr、__callt）や、#pragma宣言（saddr、callt）に加えて、変数／関数情報ファイルで指定した変数を saddr領域に割り当て、関数をcallt関数にします。

使用方法

- リンカの-vfinfoオプションで変数／関数情報ファイルを生成させる
- 変数／関数情報ファイルを、次のいずれかの方法でコンパイル時にインクルードする
  - コンパイラの-preincludeオプションで指定する
  - 各ソースファイルに、#includeでインクルードする

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

変数／関数情報ファイルの利用（2/3）

注意

- リンカの-vfinfoオプションで変数／関数情報ファイルを生成する際には、ビルドが正常に終了して、ロードモジュールファイルが生成されていることを確認してください。

リンカの-vfinfoオプション

- 変数のサイズ、および変数や関数の参照頻度を元にして、コードサイズの削減効果が高い変数や関数を選択し、それらの変数や関数に対して#pragma指令によるsaddr変数やcallt関数の宣言を追加したヘッダ・ファイル（変数/関数情報ファイル）を出力します。

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

変数／関数情報ファイルの利用（3/3）

（例）

```
/* RENESAS OPTIMIZING LINKER GENERATED FILE yyyy.mm.dd */
/*** variable information ***/
#pragma saddr data0 /* count:10,size:1,near,tp0.obj */
#pragma saddr data1 /* count:5,size:1,near,tp0.obj */
                  :
/* #pragma saddr datann */ /* count:1,size:1,near,tp1.obj */
                  :
/*** function information ***/
#pragma callt func_sub0 /* count:4,far,tp0.obj */
#pragma callt func_sub1 /* count:1,far,tp0.obj */
                  :
/* #pragma callt func0 */ /* count:1,far,tp1.obj */
                  :
```

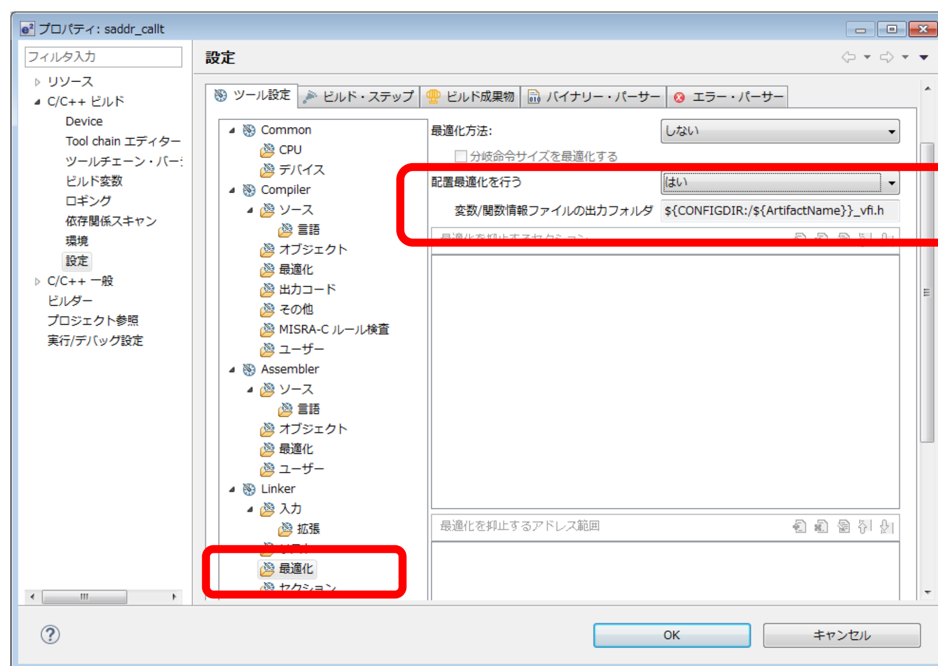
© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

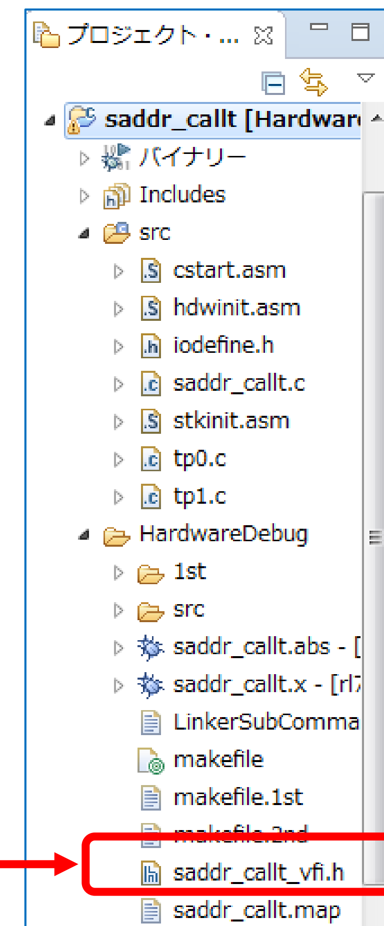
変数／関数情報ファイルの利用（e2 studio）（1/2）

変数／関数情報ファイルを自動で生成する場合

- リンカの配置最適化を有効にしてください。



- プロジェクトツリーに「プロジェクト名.h」ファイルが登録されます。



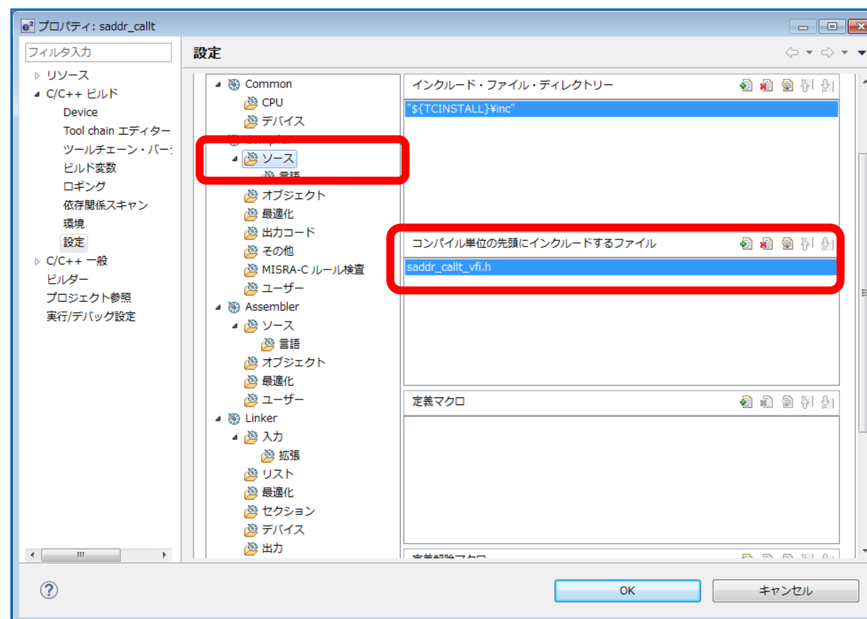
© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

変数／関数情報ファイルの利用（e2 studio）（2/2）

変数／関数情報ファイルを編集する場合（自動生成した後）

- 前ページで設定した、リンカの配置最適化を無効にしてください。
- 自動生成された「プロジェクト名.h」ファイルをsrcフォルダにインポートしてください。
- 「プロジェクト名.h」を「コンパイル単位の先頭にインクルードするファイル」として、登録してください。

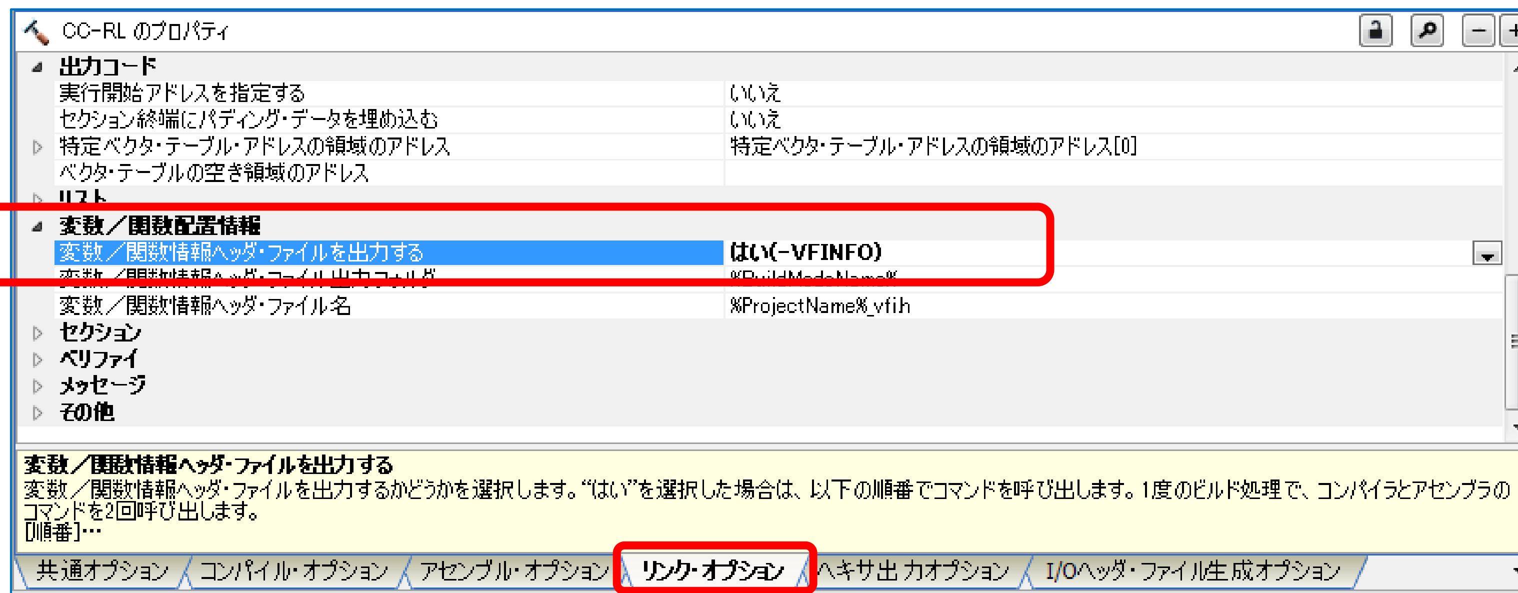


© 2015 Renesas System Design Co., Ltd. All rights reserved.

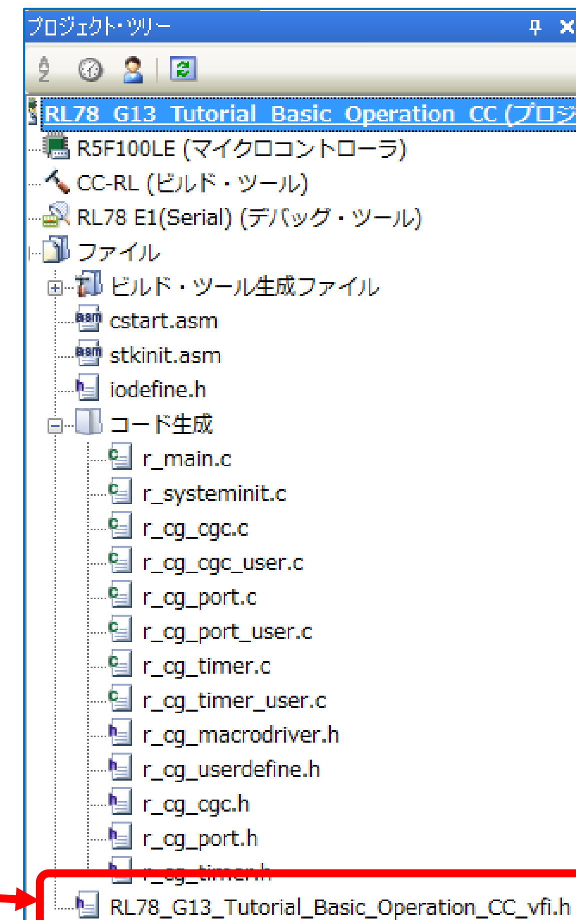
変数／関数情報ファイルの利用（CS+）（1/2）

変数／関数情報ファイルを自動で生成する場合

- 変数／関数情報ファイルの出力を有効にしてください。



- プロジェクトツリーに「プロジェクト名.h」ファイルが登録されます。



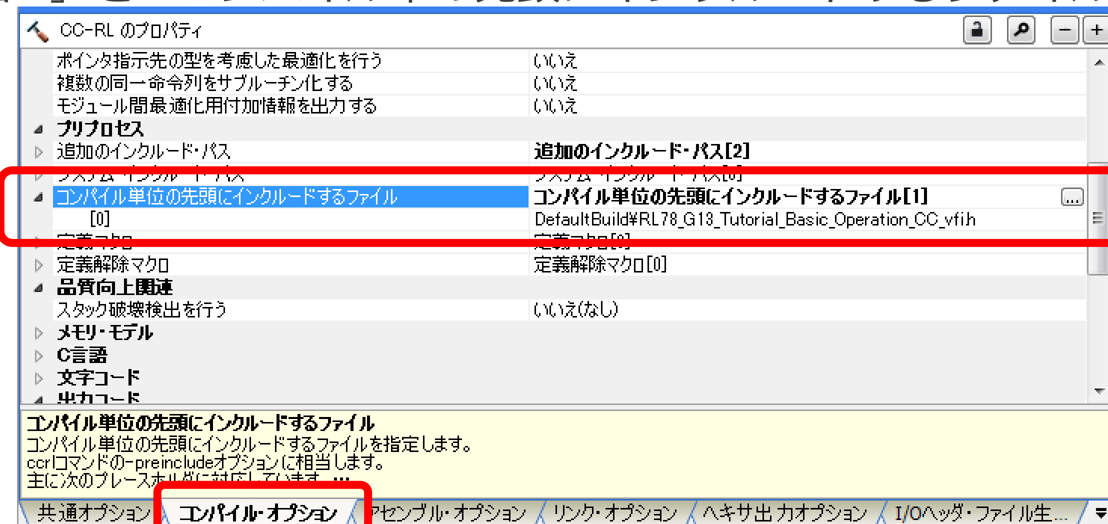
© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

変数／関数情報ファイルの利用（CS+）（2/2）

変数／関数情報ファイルを編集する場合（自動生成した後）

- 前ページで設定した、変数／関数情報ファイルの出力を無効にしてください。
- 「プロジェクト名.h」を別のフォルダ（ソースフォルダ等）にコピーしてください。
（コピーせずにそのまま使用することは可能ですが、変数／関数情報ファイルの出力を有効にした場合に、ツールにより上書き、削除されてしまいます）
- 「プロジェクト名.h」を「コンパイル単の先頭にインクルードするファイル」として、登録してください。



© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS

ルネサス システムデザイン株式会社

© 2015 Renesas System Design Co., Ltd. All rights reserved.

RENESAS