

SADDR領域とCALLT命令の活用

RL78ファミリ用Cコンパイラ CC-RL

ルネサス システムデザイン株式会社
ツールビジネス本部 ツール技術部

2015/06/30 Rev. 1.00

R20UT3476JJ0100

はじめに

- **本資料は、RL78ファミリ用Cコンパイラ CC-RLを使用して、RL78ファミリのSADDR領域とCALLT命令を使用したコードを出力する方法について説明します。**
- **SADDR領域とCALLT命令を使用することにより、コードサイズの削減をすることが可能です。**
- **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、バージョンで説明をしています。**
 - **RL78ファミリ用Cコンパイラ CC-RL V1.01.00**
 - **統合開発環境 e² studio V4.0.0.26**
 - **統合開発環境 CS+ V3.01.00**

- SADDR領域とCALLT命令を使用するには
- Cソースファイル上でのSADDR領域の使用
- Cソースファイル上でのCALLT命令の使用
- 変数／関数情報ファイルのリンカでの生成
- 変数／関数情報ファイルの利用(e² studio)
- 変数／関数情報ファイルの利用(CS+)

SADDR領域とCALLT命令を使用するには

■ SADDR領域とCALLT命令の使用するには次の方法があります。

- Cソースファイル上で宣言する
- `__saddr`宣言 : SADDR領域
- `#pragma saddr`宣言 : SADDR領域
- `__callt`宣言 : CALLT命令
- `#pragma callt`宣言 : CALLT命令

■ 変数／関数情報ファイルを使用する

- リンカの`-vfinfo`オプションの機能で、静的に参照頻度を解析し、参照頻度順に`#pragma saddr`、`#pragma callt` 宣言を指定した変数／関数情報ファイルを生成する
- 生成したファイルをコンパイル時に`-preinclude`オプションオプションで指定する

Cソースファイル上でのSADDR領域の使用(1/2)

- 使用頻度の高い外部変数および関数内static変数は、__saddr宣言等をしてください。
- 特に、1ビットのビットフィールドは__saddr宣言の効果が大きくなる傾向にあります。
- __saddr宣言をすることにより、SADDR領域に割り当てられ、直接操作する命令やコードサイズの短い命令でアクセスします。
- (例)
 - Cソースプログラム

変更前	変更後
<pre>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;</pre>	<pre>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;</pre>

Cソースファイル上でのSADDR領域の使用(2/2)

■ (例)

● 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		

■ 注意

- 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。

Cソースファイル上でのCALLT命令の使用(1/2)

- 関数呼出し頻度の高い関数は、`__callt` 宣言をしてください。
- `__callt` 宣言をすることにより、`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
<pre>void func_sub(void) { ; } void func() { func_sub(); ; func_sub(); }</pre>	<pre>__callt void func_sub(void) { ; } void func() { func_sub(); ; func_sub(); }</pre>

Cソースファイル上でのCALLT命令の使用(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	.SECTION .textf,TEXTF	
_func:	call !!_func_sub	_func:	callt [@_func_sub]
	4		2
	call !!_func_sub		callt [@_func_sub]
	4		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。

変数／関数情報ファイルのリンクでの生成

■ リンカの-vfinfoオプション

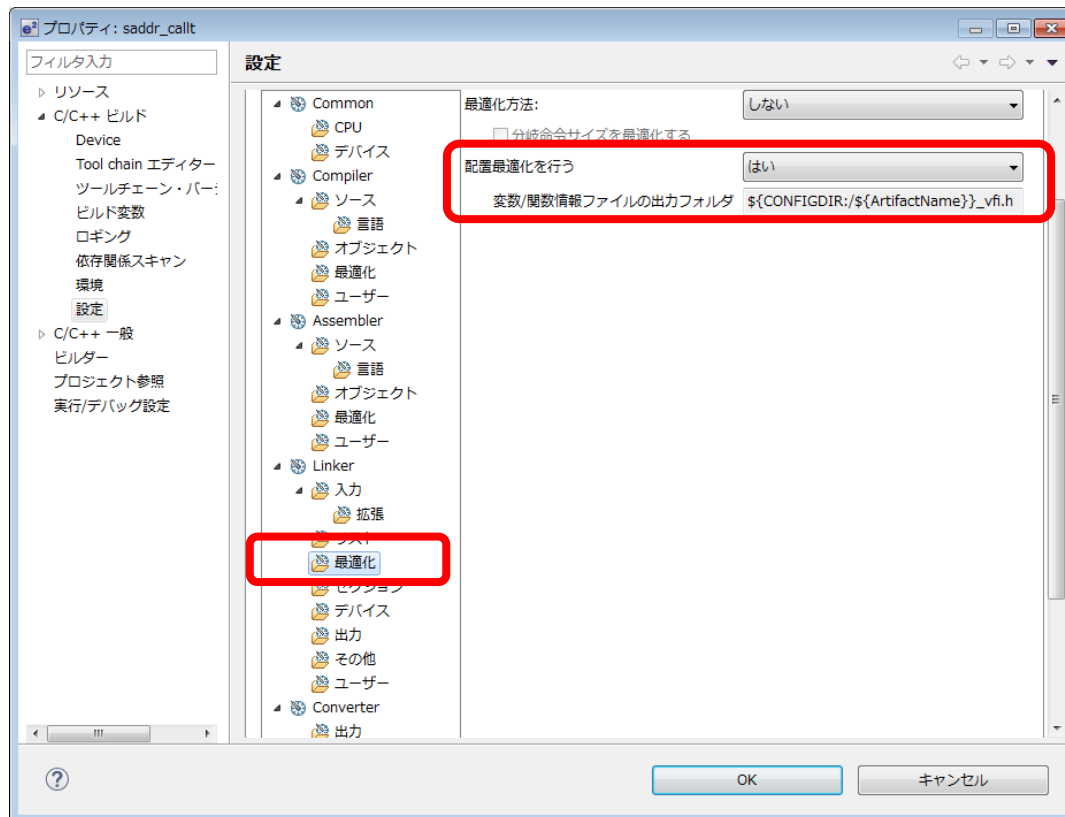
- 変数のサイズ、および変数や関数の参照頻度を元にして、コードサイズの削減効果が高い変数や関数を選択し、それらの変数や関数に対して#pragma指令によるsaddr変数やcallt関数の宣言を追加したヘッダ・ファイル(変数/関数情報ファイル)を出力します。
- (例)

```
/* 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 */  
:
```

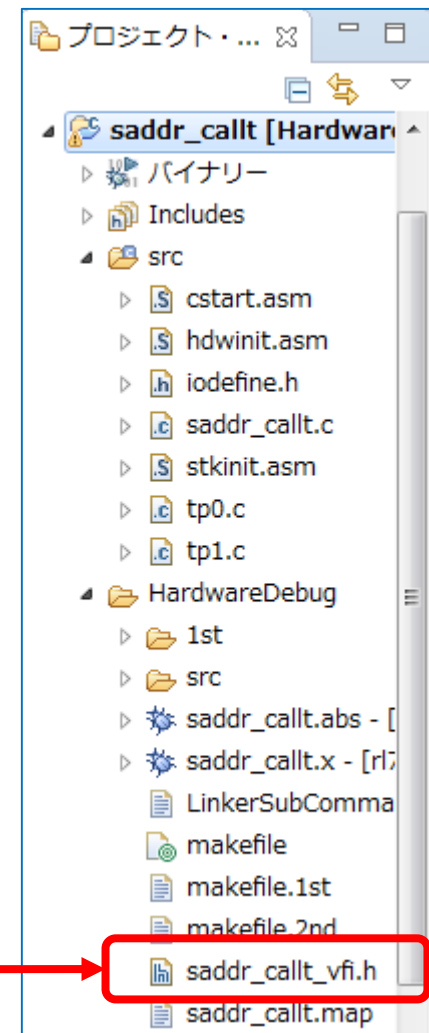
変数／関数情報ファイルの利用(e² studio)(1/2)

■ 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。



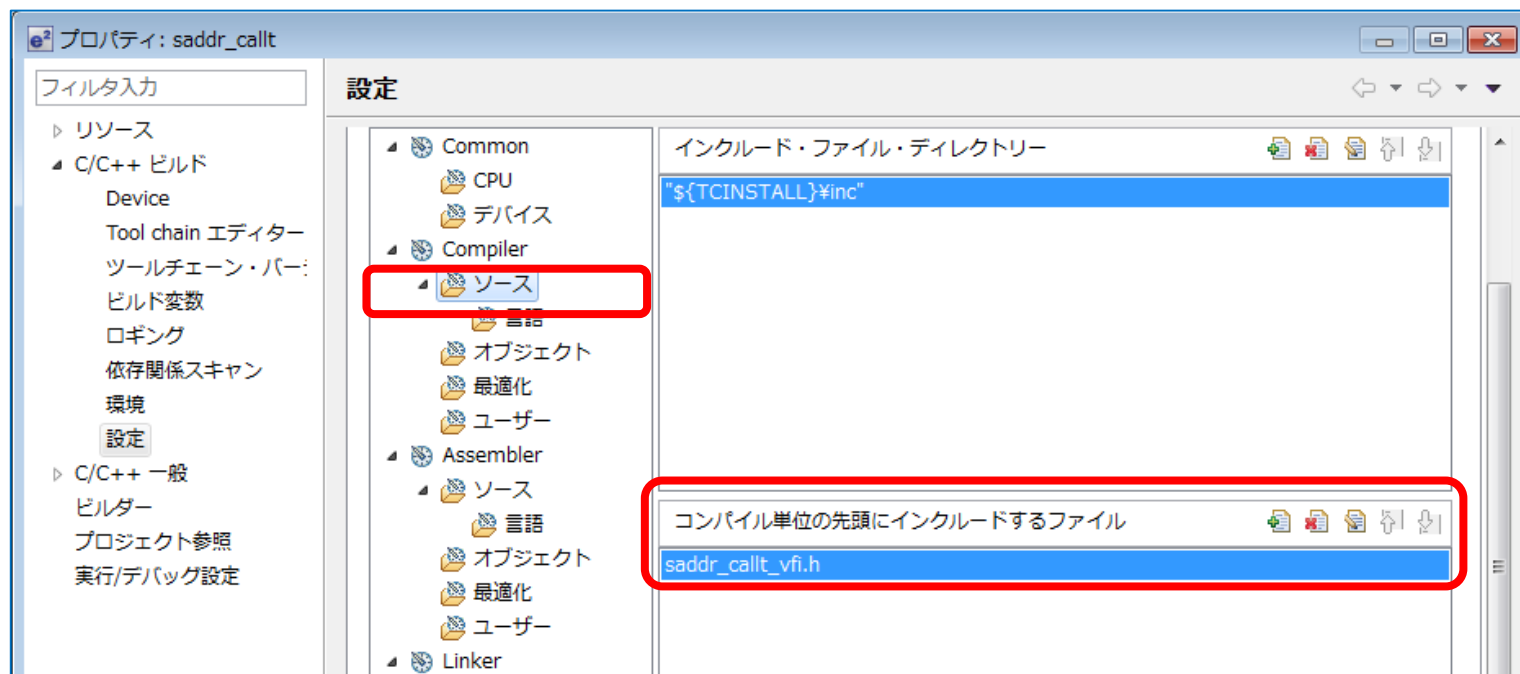
- プロジェクトツリーに、「プロジェクト名.h」ファイルが登録されます。



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。



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。

CC-RL のプロパティ

出力コード	
実行開始アドレスを指定する	いいえ
セクション終端にパディングデータを埋め込む	いいえ
▶ 特定ベクタ・テーブル・アドレスの領域のアドレス	特定ベクタ・テーブル・アドレスの領域の
ベクタ・テーブルの空き領域のアドレス	
▶ リスト	
■ 変数／関数配置情報	
変数／関数情報ヘッダ・ファイルを出力する	はい(-VFINFO)
変数／関数情報ヘッダ・ファイル出力フォルダ	%BuildModeName%
変数／関数情報ヘッダ・ファイル名	%ProjectName%_vfi.h
▶ セクション	
▶ ベリファイ	
▶ メッセージ	
▶ その他	

変数／関数情報ヘッダ・ファイルを出力する
変数／関数情報ヘッダ・ファイルを出力するかどうかを選択します。“はい”を選択した場合は、以下の順番でコマンドを呼び出すコマンドを2回呼び出します。
[順番]...

共通オプション / コンパイル・オプション / アセンブル・オプション / **リンカ・オプション** / ヘキサ出力オプション

プロジェクト・ツリー

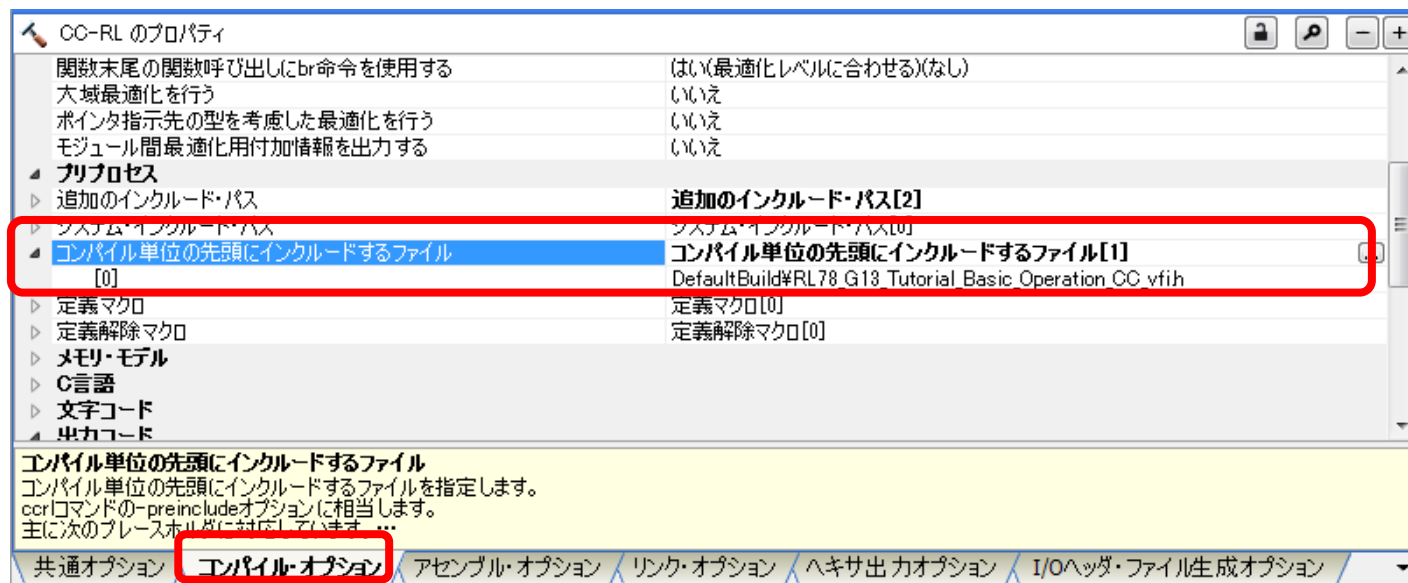
- RL78_G13_Tutorial_Basic_Operation_CC (プロジェクト)
- R5F100LE (マイクロコントローラ)
- CC-RL (ビルド・ツール)
- RL78 E1(Serial) (デバッグ・ツール)
- ファイル
 - ビルド・ツール生成ファイル
 - cstart.asm
 - stkinit.asm
 - iodefine.h
 - コード生成
 - r_main.c
 - r_systeminit.c
 - r_cg_cg.c
 - r_cg_cg_user.c
 - r_cg_port.c
 - r_cg_port_user.c
 - r_cg_timer.c
 - r_cg_timer_user.c
 - r_cg_macrodriver.h
 - r_cg_userdefine.h
 - r_cg_cg.h
 - r_cg_port.h
 - r_cg_timer.h
 - RL78_G13_Tutorial_Basic_Operation_CC_vfi.h**

- 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。

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。





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.