RL78ファミリ用Cコンパイラ CA78K0R SADDR領域とCALLT命令の使用

株式会社ルネサス ソリューションズ
ツールビジネス本部 ツール技術部

2014/6/20 Rev. 1.00

R20UT3048JJ0100

はじめに

- 本資料は、CA78K0R Cコンパイラを使用して、RL78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。

- RL78の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(-qx3)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。

- 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／関数情報ファイル生成ツールの利用
- プログラム解析機能（CubeSuite＋）の利用

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
    - __sreg宣言 ： SADDR領域
    - __callt宣言 ： CALLT命令
  - 変数／関数情報ファイル生成ツールを使用する
    - 静的に参照頻度を解析し、参照頻度順に__sreg宣言、__callt宣言を指定した変数／関数情報ファイルを作成する
    - 生成したファイルをコンパイル時にオプションで指定する

- 変数／関数の参照頻度はプログラム解析機能で参照することが可能です。

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、__sreg宣言をしてください。
- 特に、1ビットのビットフィールドは__sreg宣言の効果が大きくなる傾向にあります。
- __sreg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

```
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;
```

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;
__sreg BITF data0, data1;

                data0.b4 = data1.b1;
```

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sreg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

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

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		
@@CODEL CSEG _func:			@@CALT CSEG CALLT0 ?func_sub: DW _func_sub @@CODEL CSEG _func:		2
call	!!_func_sub	4	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(@@CALT)。
  - 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宣言の指定が可能です。

変数／関数情報ファイル生成ツールの利用（1/3）

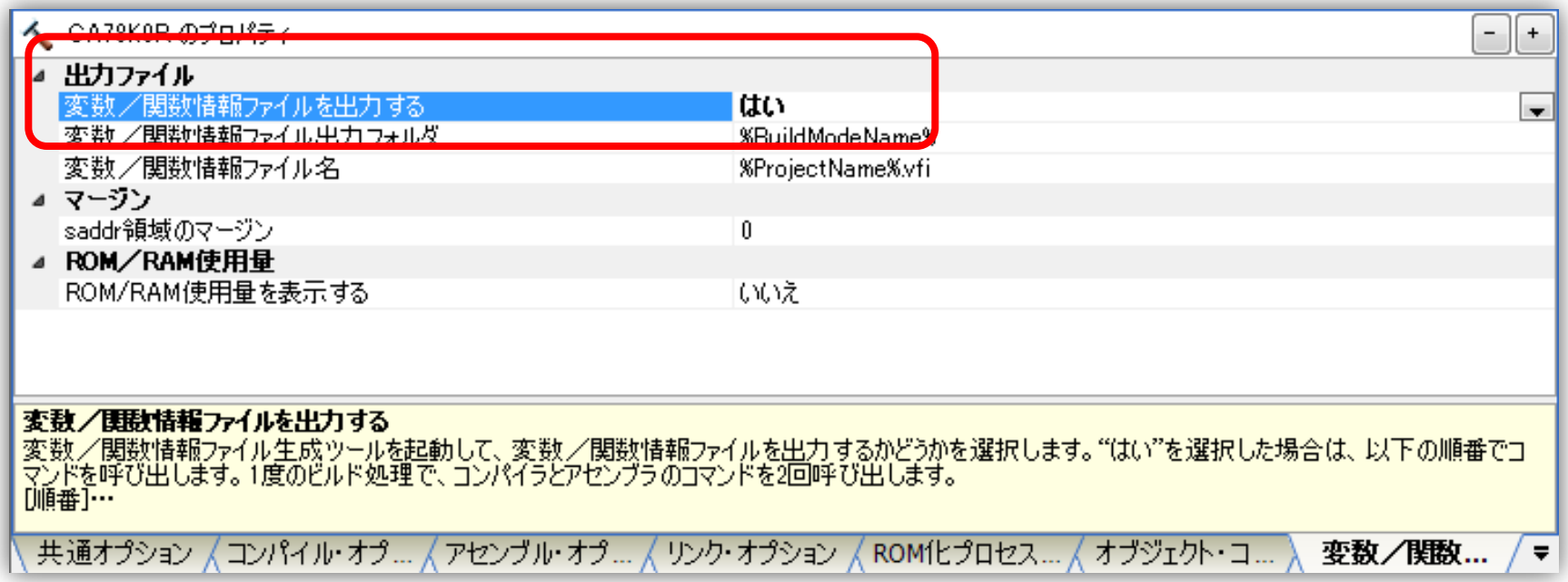
- **機能**
  - **参照頻度の高い変数をsaddr領域に割り当てます。**
  - **参照頻度の高い関数をcallt関数にします。**
  - **ソース上で指定した修飾子（__sreg、__callt）に加えて、変数／関数情報ファイルで指定した変数を saddr領域に割り当て、関数をcallt関数にします。**

- **使用方法**
  - CubeSuite+ **の** CA78K0R(**ビルド・ツール**)**のプロパティの「変数／関数配置オプション」タブで設定してください（次ページ参照）。**

- **注意**
  - **この機能をご使用になる前に、ビルドが正常に終了して、.lmfが生成されていることを確認してください。**

変数／関数情報ファイル生成ツールの利用（2/3）

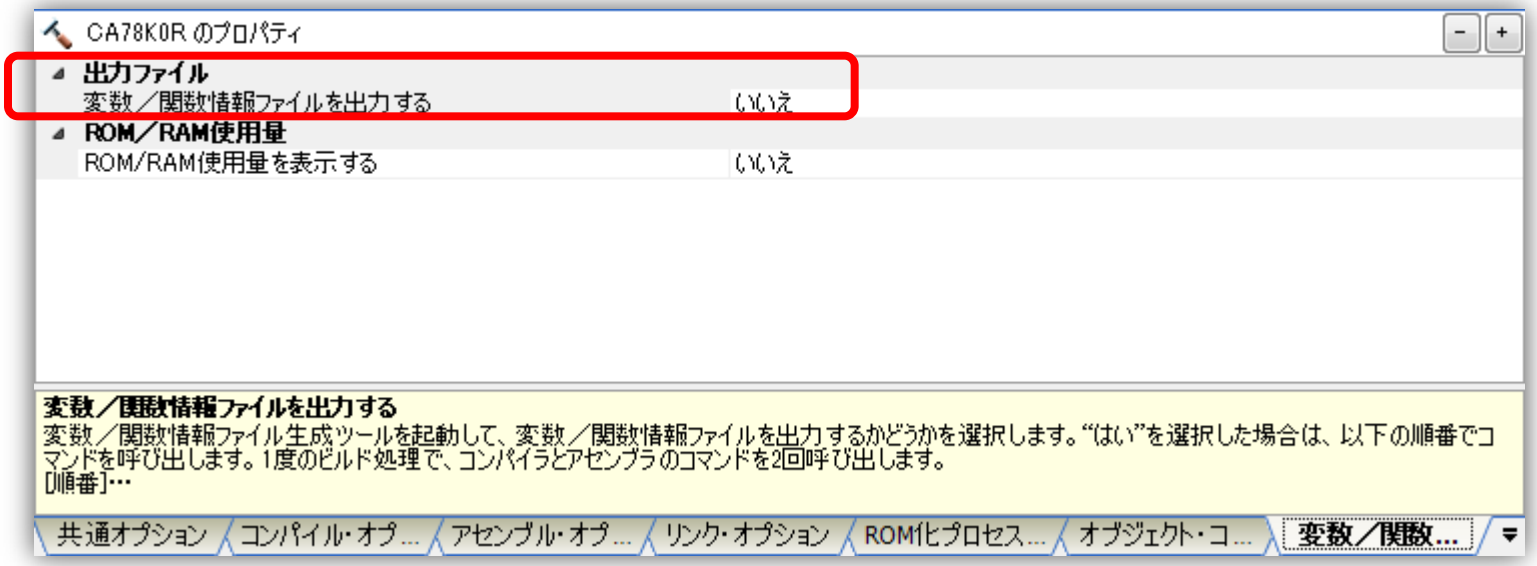
- **変数／関数情報ファイルを自動で生成する場合**
  - **「変数／関数配置オプション」タブ**
    - **→“変数／関数情報を出力する”を“はい”にしてください。**



  - **プロジェクトツリーパネルに「*プロジェクト名*.vfi」ファイルが登録されます。**

変数／関数情報ファイル生成ツールの利用（3/3）

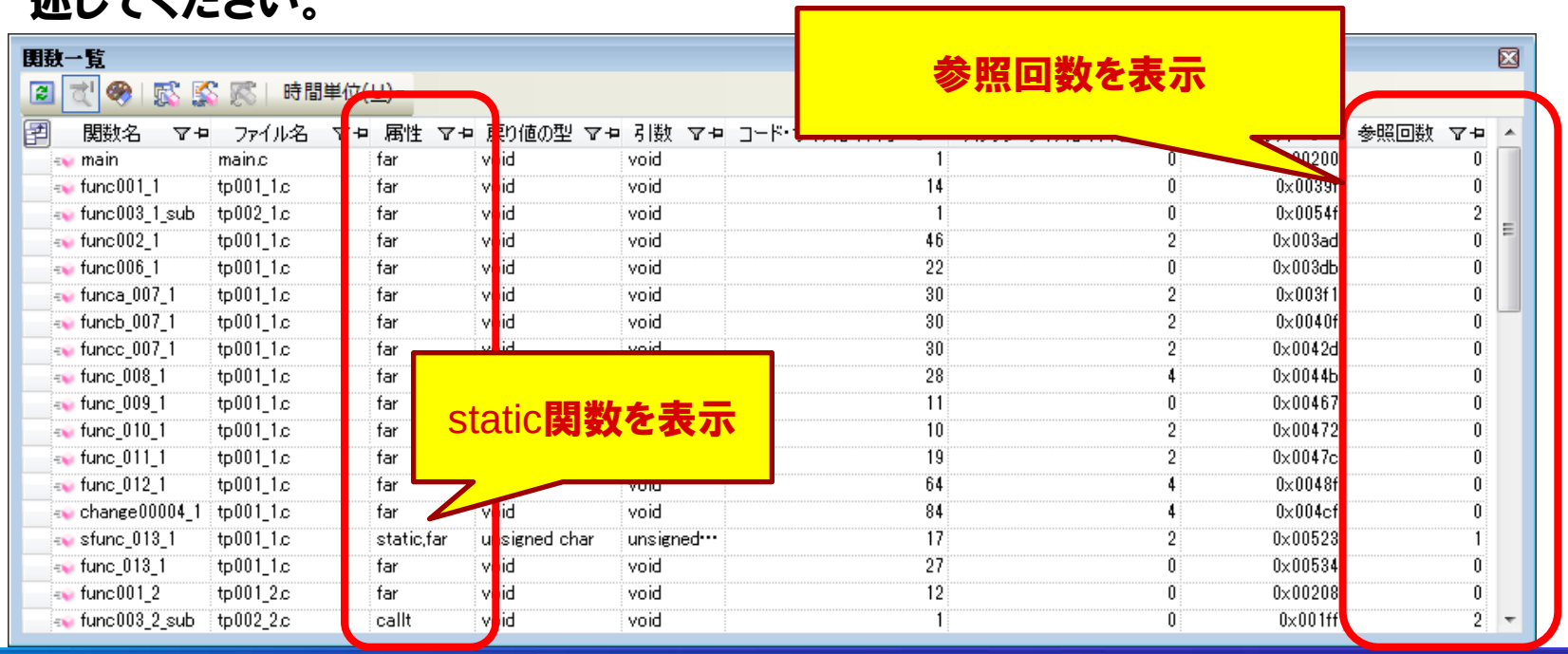
- 変数／関数情報ファイルを編集する場合
  - 「変数／関数配置オプション」タブ
    - →“変数／関数情報を出力する”を“いいえ”にしてください。



- プロジェクトツリーパネルの「*プロジェクト名*.vfi」ファイルをダブルクリックして、エディタで編集してください。
- 注意
  - “はい”のままだと、ビルド毎に変数／関数情報ファイルを生成し直します。

プログラム解析機能(CubeSuite+)の利用

- CubeSuite+のプログラム解析機能を使用して、次を実施することによりコードサイズが削減される可能性があります。
  - 関数やconst定数や変数で、使用されていないもの(参照回数が 0 のもの)を削除してください。
    - 注意:ビルドしてエラーにならないことを確認してください。
  - 1回しか参照されていないstatic関数を、関数呼び出しではなく、処理を直接記述してください。



ルネサス ソリューションズ株式会社