【注意事項】

RL78 ファミリ用 C コンパイラパッケージ (注意事項 CCRL#035-#037)

R20TS1145JJ0100
Rev.1.00
2025.07.05

概要

RL78 ファミリ用 C コンパイラパッケージ CC-RL の使用上の注意事項を連絡します。

1. -dbl_size=8 オプションを使用する場合の注意事項 (CCRL#035)
2. セクション・アドレス演算子を複数使用する場合の注意事項 (CCRL#036)
3. far 属性の const 変数の extern 宣言を使用している場合の注意事項 (CCRL#037)

注: 注意事項の後ろの番号は、注意事項の識別番号です。

1. -dbl_size=8 オプションを使用する場合の注意事項 (CCRL#035)

1.1 該当製品

CC-RL V1.13.00 ～ V1.15.00

1.2 内容

倍精度浮動小数点型(double 型)の乗算用のランタイムライブラリの実行中に割り込みが発生すると、乗算の結果が不正になる場合があります。

1.3 発生条件

次の(1) (2)の条件を共に満たす場合に発生する可能性があります。

(1) -dbl_size=8 オプションを指定している。
※ -dbl_size=8 は、-cpu=S3 オプション指定時のみ有効なオプションです。

(2) double 型乗算用のランタイムライブラリである_COM_dmul が呼び出されるプログラムを記述している。
※ double 型の乗算・除算を記述している場合や、以下の標準ライブラリを使っている場合に呼び出されることがあります。

該当関数
acos, asin, atan, atan2, cos, sin, tan, cosh, sinh, tanh, exp, frexp, ldexp, log, log10, pow, sqrt, fmod, acosh, asinh, atanh, log1p, scalbn, scalbln, acosl, asinl, atanl, atan2l, cosl, sinl, tanl, acoshl, asinhl, atanhl, coshl, sinhl, tanhl, expl, frexpl, ldexpl, logl, log10l, log1pl, scalbnl, scalblnl, powl, sqrtl, fmodl, printf, scanf, snprintf, sprintf, sscanf, vprintf, vscanf, vsnprintf, vsprintf, vsscanf, atof, _COM_atof_f, strtod, _COM_strtod_ff, strtold, _COM_strtold_ff

リンク時に-list オプションおよび-show=symbol オプションを指定してリンク・マップ・ファイル(拡張子.map)を生成し、次のように__COM_dmul がシンボル名として出力されていれば、該当します。

リンク・マップ出力例
FILE=_COM_dmul 000020a1　000022dc　　23c __COM_dmul

[発生例]

以下に発生例を記します。

ccrl -cpu=S3 -dbl_size=8 tp.c　// (1)

```
/* tp.c */
double d;
void main(void) {
  d = d * 7.0;   // (2)
}
```

1.4　回避策

ランタイムライブラリ__COM_dmul の呼び出しの前に、組み込み関数__DI()等により、割り込み禁止にすることで回避可能です。

また、-dbl_size=4 を指定し、double 型を単精度浮動小数点型と扱うことでも回避可能です。

1.5　恒久対策

次版以降で修正する予定です。リリース時期は未定です。

2. セクション・アドレス演算子を複数使用する場合の注意事項 (CCRL#036)

2.1 該当製品

CC-RL V1.15.00

2.2 内容

セクション・アドレス演算子(__sectop または__secend)を複数記述している場合に、その結果が不正になることがあります。

2.3 発生条件

次の(1) (2)の条件を共に満たす場合に、正しいアドレスにアクセスすることができないことがあります。

(1) 同じ関数内にセクション・アドレス演算子(__sectop または__secend)で取得したアドレスへのアクセスが複数個所ある。
(2) (1)の複数アドレスの中には、上位 16 ビットが異なるものが含まれている。
※ リンク時に-list オプションを指定して出力するリンク・マップ・ファイル(拡張子.map)で確認することができます

__sectop は START で示されている開始アドレスを取得します。__secend は END で示されている終了アドレスに+1 した値を取得します。

例えば次のリンク・マップ・ファイルにおいて、__sectop(“ram_text_f”)は 0x00003000、__secend(“ram_text_f”)は 0x0000302a(0x00003029+1)の値になります。また、これらの上位 16 ビットは 0x0000 になります。同様に__sectop(“ram_text_fR”)は 0x000ff000、__secend(“ram_text_fR”)は 0x000ff02a(0x000ff029+1)の値になります。また、これらの上位 16 ビットは 0x000f になります。

リンク・マップ出力例

```
*** Mapping List ***

SECTION                 START       END         SIZE    ALIGN
ram_text_f
                        00003000    00003029    2a      1
ram_text_fR
                        000ff000    000ff029    2a      1
```

[発生例]

以下に発生例を記します。

ccrl -cpu=S3 tp2.c

```
/* tp2.c */
#define SIZE  42

void test(void){
  int i;
  unsigned char __far *dst, *src;
  src = __sectop("ram_text_f");
  dst = __sectop("ram_text_fR");
  for(i = 0; i < SIZE; i++) {
    *dst++ = *src++;  // (1) __sectop で取得したアドレスにアクセスしている
  }
}
```

セクション ram_text_f、ram_text_fR は__sectop で取得したアドレスの上位 16 ビットが異なるとします。(2)

2.4 回避策

セクション・アドレス演算子で取得したアドレスを volatile 修飾した大域変数の初期値にし、その大域変数経由でアクセスすることで回避できます。

回避策

```
#define SIZE 42

static unsigned char __far* const volatile secTopRamTextF =
__sectop("ram_text_f");
static unsigned char __far* const volatile secTopRamTextFR =
__sectop("ram_text_fR");
void test(void){
  int i;
  unsigned char __far *dst, *src;
  src = secTopRamTextF;
  dst = secTopRamTextFR;
  for(i = 0; i < SIZE; i++) {
    *dst++ = *src++;
  }
}
```

2.5 恒久対策

次版以降で修正する予定です。リリース時期は未定です。

3. far 属性の const 変数の extern 宣言を使用している場合の注意事項 (CCRL#037)

3.1 該当製品

CC-RL V1.15.00

3.2 内容

far 属性の const 変数の extern 宣言を使用している場合に、変数の値を正しく参照できないことがあります。

3.3 発生条件

次の(1)～(4)の条件を全て満たす場合に、変数の値を正しく参照できないことがあります。

(1) 複数の far 属性の const 修飾された大域変数が存在する。
※ far 修飾子だけでなく、-far_rom オプションによって far 属性になる場合も対象です。
(2) それらのうち複数を、同一の関数内で使用している。
(3) 使用している変数のうち、少なくとも 1 つは、extern 宣言されたものである。
(4) 使用している変数の中に、変数が配置されるセクションが異なり、かつ、アドレスの上位 16 ビットが異なるものが含まれている。
※ ただし、以下のいずれかの機能を使わない限り、この条件を満たすことはありません。

- #pragma section
- 以下のリンカ・オプションのいずれか
  - -rom
  - -split_section
  - -stride_dsp_memory
  - -stride_self_area
  - -stride_ocdtr_area
  - -stride_ocdhpi_area

変数のアドレスは、リンク時に-list オプションと-show=symbol オプションを指定して出力するリンク・マップ・ファイル(拡張子 .map)で確認することができます。

例えば次のリンク・マップ・ファイルにおいて、変数 gv1 のアドレスは 0x0000001a であり、その上位 16 ビットは 0x0000 になります。同様に変数 gv2 のアドレスは 0x00010000 であり、その上位 16 ビットは 0x0001 になります。

リンク・マップ出力例

```
*** Symbol List ***

SECTION=
FILE=                                START        END      SIZE
  SYMBOL                             ADDR         SIZE     INFO       COUNTS   OPT

SECTION=.constf
FILE=C:¥test1.obj
                                     0000001a   0000001b          2
  _gv1
                                     0000001a          2    data ,g         *

SECTION=$.constf_part1
FILE=C:¥test2.obj
                                     00010000   00014001       4002
  _gv2
                                     00010000          2    data ,g         *
```

[発生例]

以下に発生例を記します。

ccrl -cpu=S3 tp3.c

```
/* tp3.c */
const int __far gv1 = 0x100;  // (1)
extern const int __far gv2;    // (1) (3)
int test1(void){
  return gv1 + gv2;   // (2)
}
```

変数 gv1、gv2 のアドレスの上位 16 ビットが異なるものとします。(4)

この例の場合、変数 gv1 の値を正しく参照できません。

3.4 回避策

以下のいずれかの方法により、同一関数内で使われている変数のアドレスの上位 16 ビットが同じになるように調整することで回避できます。

- リンカの-start オプションで変数が配置されているセクションの開始アドレスを変更する
- 変数を同じセクションに配置する

3.5 恒久対策

次版以降で修正する予定です。リリース時期は未定です。

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jul.05.25	-	新規発行

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/