CS+ RL78 コンパイラ CC-RL V1.01.00
リリースノート

R20UT3367JJ0100
Rev.1.00
2015.02.26

この度は，統合開発環境 CS+をご使用いただきまして，誠にありがとうございます。

この添付資料では，本製品をお使いいただく上での制限事項および注意事項等を記載しております。ご使用の前に，必ずお読みくださいますようお願い申し上げます。

目次

第1章　対象デバイスについて

CC-RL がサポートする対象デバイスに関しては，WEB サイトに掲載しています。

こちらをご覧ください。

CS+製品ページ：

http://japan.renesas.com/cs+

第2章　ユーザーズ･マニュアルについて

本製品に対応したユーザーズ･マニュアルは，次のようになります。本文書と合わせてお読みください。

マニュアル名	資料番号
CC-RL コンパイラ	R20UT3123JJ0101
CS+ CC-RL ビルド・ツール操作編	R20UT3284JJ0100

第3章　アンインストール時の選択キーワード

本製品をアンインストールする場合は，2 つの方法があります。

・統合アンインストーラを使用する(CS+自体をアンインストールする)

・個別にアンインストールする(本製品のみをアンインストールする)

個別にアンインストールを行なう場合，コントロールパネルの「プログラムと機能」から，「CS+ CC-RL V1.01.00」を選択してください。

第4章　変更点

本章では、CC-RL V1.01.00 の変更点について説明します。

4.1　CC-RL V1.01.00の変更点

CC-RL V1.00.00 から V1.01.00 への変更点について説明します。

4.1.1　最適化強化

生成コードの性能を改善しました。

4.1.2　再配置属性と疑似命令の追加【アセンブラ】

以下の再配置属性を追加しました。

アセンブラ再配置属性	概要
SBSS_BIT	saddr 領域内の初期値を持たないビットセクション向けの再配置属性です。
BSS_BIT	RAM 領域内の初期値を持たないビットセクション向けの再配置属性です。
BIT_AT	指定番地に配置する、初期値を持たないビットセクション向けの再配置属性です。

以下の疑似命令を追加しました。

アセンブラ疑似命令	概要
.BSEG	アセンブラにビットセクションの開始を指示します。
.DBIT	1 ビットのビット領域を確保します。

4.1.3　移行支援機能におけるサポートの追加【アセンブラ】

移行支援機能において、以下の記述をサポートしました。

記述内容 （CA78K0R アセンブラ言語仕様）	対応する記述内容 （RL78 アセンブラ言語仕様）
BSEG 再配置属性なし	.BSEG SBSS_BIT
BSEG UNIT	.BSEG SBSS_BIT
BSEG AT	.BSEG BIT_AT
DBIT	.DBIT

4.1.4　移行支援機能における領域確保命令の機能強化【アセンブラ】

移行支援を強化するため、移行支援機能の以下の疑似命令で複数オペランドの記述を可能にしました。

アセンブラ疑似命令
DB
DW
DG

4.1.5 移行支援機能におけるインクルードの機能強化【アセンブラ】

移行支援を強化するため、"$INCLUDE("直後に空白（以下、△とする）が書ける記述に対応しました。

"$INCLUDE(△a.inc)" は "$INCLUDE(a.inc)" と同じ意味となります。

4.1.6 マクロの追加【アセンブラ】

以下のマクロを追加しました。

マクロ名	定義
__RENESAS_VERSION__	バージョンが V.XX.YY.ZZ の場合，0xXXYYZZ00 とします。

但し、ccrl 起動時のみ有効です。asrl 起動時は追加されません。

4.1.7 標準ライブラリ及びランタイムライブラリコード用セクションの追加【ライブラリ】

以下のセクションを追加しました。

セクション名	再配置属性	概要
.SLIB	TEXTF	標準ライブラリ コード用セクションです。
.RLIB	TEXTF	ランタイムライブラリ コード用セクションです。

4.1.8 関数及びマクロの追加【ライブラリ】

以下の関数及びマクロを追加しました。

関数名/マクロ名	概要
printf_tiny	printf の簡易版です。
sprintf_tiny	sprintf の簡易版です。
__PRINTF_TINY__	printf/sprintf の呼び出しを printf_tiny/sprintf_tiny に置き換えます。

4.1.9 ライブラリ関数ldexp及びldexpfにおけるNaNに対する処理の変更【ライブラリ】

ライブラリ関数 ldexp 及び ldexpf における NaN に対する処理を以下のように変更しました。

・入力引数が NaN である場合、NaN を返し、グローバル変数 errno にマクロ EDOM を設定します。

4.1.10 数学関数の性能改善【ライブラリ】

以下の数学関数の処理時間とサイズを改善しました。

関数名	概要
sqrt	平方根関数（倍精度用）です。
sqrtf	平方根関数（単精度用）です。

4.1.11 マクロの追加と修正【コンパイラ】

以下のマクロを追加しました。

マクロ名	定義
__CNV_CA78K0R__	値は 10 進定数 1 に設定されます(-convert_cc オプションで ca78k0r を指定した場合に定義）。
__CNV_NC30__	値は 10 進定数 1 に設定されます(-convert_cc オプションで nc30 を指定した場合に定義）。
__CNV_IAR__	値は 10 進定数 1 に設定されます（-convert_cc オプションで iar を指定した場合に定義）。
__BASE_FILE__	C ソース・ファイル名が設定されます。

以下のマクロを-ansi オプション指定時のみ有効となるように修正しました。

__STDC_VERSION__	値は 10 進定数 199409L に設定されます。

4.1.12 可変個引数マクロに関する仕様の変更【コンパイラ】

可変個引数マクロを許可しました。

4.1.13 コンパイラのキーワードの追加【コンパイラ】

以下のキーワードを追加しました。

キーワード	概要
__callt	callt 命令で関数を呼び出します。

4.1.14 #pragma指令の追加【コンパイラ】

以下の#pragma 指令を追加しました。

#pragma 指令	概要
#pragma callt	callt 命令で関数を呼び出します。
#pragma saddr	saddr 領域に静的変数を割り当てます。

4.1.15 2進定数の記述可能箇所の追加【コンパイラ】

C 言語構文中、及び、以下の#pragma 指令で 2 進定数を記述可能としました。

#pragma 指令	2 進定数の記述可能箇所
#pragma interrupt	vect のパラメータ
#pragma rtos_interrupt	vect のパラメータ
#pragma address	アドレス

4.1.16 組み込み関数の追加【コンパイラ】

以下の組み込み関数を追加しました。

組み込み関数名	概要
__get_psw	PSW の内容を返します。
__set_psw	PSW に値を設定します。

4.1.17 #pragma rtos_interruptに関する記述仕様の変更【コンパイラ】

vect 指定を省略可能な記述方法に変更し、vect 未指定時の仕様を追加しました。

・#pragma△rtos_interrupt△[(]<関数名>[(vect=アドレス)][)]

4.1.18 #pragma interrupt/interrupt_brk/rtos_interrupt を指定した割込みハンドラのnear/far属性仕様の変更【コンパイラ】

#pragma interrupt 及び#pragma rtos_interrupt の仕様を変更しました。

・vect 指定時、割込みハンドラは強制的に__near 属性にします。

・vect 未指定時、割込みハンドラの__near/__far 属性は変更しません。

#pragma interrupt_brk の仕様を変更しました。

・割込みハンドラは強制的に__near 属性にします。

4.1.19 #pragma sectionにおける変更セクション名に関する仕様の変更【コンパイラ】

セクション種別を指定する場合のみ、“.”(ドット) を変更セクション名の先頭に使用できるように仕様を変更しました。

4.1.20 文字集合gb2312の変更【コンパイラ】

C 言語における多バイト文字及び-character_set オプションの指定形式における gb2312 を gbk に変更しました。

4.1.21 オプションの追加【コンパイラ】

以下のオプションを追加しました。

オプション名	概要
-convert_cc	他コンパイラ向けに書かれたプログラムの移行を支援します。
-pack	構造体メンバのアライメントを 1 にします。

4.1.22 Professional版で使用可能なオプションの設定【コンパイラ】

以下のオプションを Professional 版で使用可能なオプションとしました。

オプション名
-misra2004
-ignore_files_misra
-check_language_extension

4.1.23 奇数番地に対する2バイト以上の間接参照回避【コンパイラ】

奇数番地を２バイト以上の型で間接参照する場合、可能な範囲で警告を出力するようにしました。

4.1.24 チェック項目の追加【リンカ】

以下のチェック項目を追加しました。

・セクションが 64K バイト境界、及び、64K-1 バイト境界をまたいで配置されるかチェック

・メモリ配置、及び、セクションのメモリ配置に関する整合性のチェック

4.1.25 オプションの追加【リンカ】

以下のオプションを追加しました。

オプション名	概要
-VFINFO	変数/関数情報ファイルを出力します。
-AUTO_SECTION_LAYOUT	セクションを自動的に配置します。
-DEBUG_MONITOR	OCD モニタのメモリ領域を指定します。
-RRM	RRM/DMM 機能用ワーク領域を指定します。
-SELF	セルフ RAM 領域にセクションを配置しません。
-SELFW	セルフ RAM 領域にセクションを配置した場合に警告を出力します。
-OCDTR	トレース RAM とセルフ RAM 領域にセクションを配置しません。
-OCDTRW	トレース RAM とセルフ RAM 領域にセクションを配置した場合に警告を出力します。
-OCDHPI	ホット・プラグイン RAM, トレース RAM, およびセルフ RAM 領域にセクションを配置しません。
-OCDHPIW	ホット・プラグイン RAM, トレース RAM, およびセルフ RAM 領域にセクションを配置した場合に警告を出力します。
-CHECK_DEVICE	オブジェクト・ファイルで利用したデバイス・ファイルとリンカのオプションで指定したデバイス・ファイルの整合性をチェックします。
-CHECK_64K_ONLY	セクションが(64K-1)バイト境界をまたいで配置されるかをチェックしません。
-NO_CHECK_SECTION_LAYOUT	セクションの割り付けアドレスとデバイスが持つメモリのアドレスとの整合性をチェックしません。

4.1.26 -showオプションの出力情報の追加【リンカ】

-show=struct を指定した場合は、構造体、及び共用体メンバの情報をリンク・マップ・ファイルに出力するようにしました。

4.1.27 オプション指定により生成するシンボルの追加【リンカ】

オプション指定により生成する、以下のシンボルを追加しました。

シンボル名	概要
__STACK_ADDR_START	スタック領域の最上位アドレス+1 を指します。
__STACK_ADDR_END	スタック領域の最下位アドレス を指します。
__RAM_ADDR_START	RAM 領域の最下位アドレス を指します。
__RAM_ADDR_END	RAM 領域の最上位アドレス+1 を指します。
.monitor1	0xCE～0xD7 までの領域を指します。
.monitor2	OCD モニタ先頭アドレスから OCD モニタ終了アドレスの領域を指します。

4.1.28 スタートアップの変更

スタートアップにおけるスタック領域の設定仕様を変更しました。

4.1.29 注意事項の解除

以下の 4 件の注意事項を解除しました。

① 変数に一致と大小比較判定を両方行う場合の注意事項

同一関数内で、if 文 または ループの制御式として、"!=" または "==" による比較式と、大小比較判定式を組み合わせて記述している場合に、その if 文 またはループの判定を誤る場合があります。

【発生条件】

以下の全ての条件を満たす場合に発生する場合があります。

(1) -Odefault オプション、-Osize オプション または -Ospeed オプション の
いずれかが有効である。

(2) 定数(注 1)と変数(注 2)を "!=" または "==" により比較する式が存在する。
注 1: 静的に定数と分かる式を含む。
注 2: 配列変数、構造体メンバ、および共用体メンバを含む。

(3) (2)の比較式を含む関数内に、定数と(2)の変数を "<"、">"、"<="、
または ">=" により比較する式が存在する。

(4) (2)の変数は volatile 修飾されていない。

(5) (2)および(3)の比較式は以下のいずれかを満たす。
(5-1) (2)と(3)の比較式が "||" または "&&" で結合されている
(5-2) (2)と(3)の比較式がそれぞれ "if 文" または "?:式" の条件式にあり
かつそれぞれの"if 文" または "?:式" が連続して実行される

(6) (2)と(3)の比較式間に別の式および文がない。

(例) 発生条件の(1)を満たしているものとします。

```
int g(void);
int f(void)
 {
   int x = g();     // 発生条件(4)
   if (x == -2 ||   // 発生条件(2)(5-1)
       x > 1200) {  // 発生条件(3)(6)
     return 1;
   }
   return -1;
 }
```

【回避策】

以下のいずれかの方法で回避できます。

(1) -Onothing オプションを指定する。

(2) 発生条件(2)の変数を volatile 修飾する。
(3) 発生条件(2)および(3)の比較のうち後に実行される方の直前に、ダミーの volatile 変数を参照する。
(例)回避策(3)の例を、以下示します。

```
int g(void);
volatile int dummy;       // ダミーの volatile 変数の宣言
int f(void)
{
  int x = g();
  if (x == -2 ||
      (dummy, x > 1200)) { // 回避策(3)
    return 1;
  }
  return -1;
}
```

② ポインタをインデックスに使用した場合、正しくアクセスできないことがある注意事項
静的記憶域期間を持つシンボルのアドレスを整数型にキャストし、キャストの結果を他のポインタに加算し、加算した結果できたポインタの指示先を参照するコードを記述すると、該当の参照が不正なコードとなることがあります。
【発生条件】
以下の条件を全て満たす場合に発生する可能性があります。
インライン展開や畳み込みといった最適化の結果、以下の条件に適合した場合にも発生することがあります。
(1)静的記憶域期間を持つシンボルのアドレスを整数型にキャストする。
(2)(1)でキャストした結果を他のポインタに加算する。(注)
注: キャストの結果を配列のインデックスとして使う場合もここでいう「加算」に該当します。
例えば、以下のコードは不正なコードになることがあります。

```
array[i + (int)&globalVariable] = 0;
```

(3)(2)で加算した結果のポインタの指示先を参照する(ロードまたはストアをおこなう)コードを記述する。
(例)

```
extern char __near addressIsIndex;
extern char __near array[10];
char noteForAddressWithWordBase(char* base){
  return base[(signed)&addressIsIndex]; // 発生条件(1)(2)(3)
}
```

【回避策】
シンボルのアドレスを整数型にキャストした値を利用する前に、volatile で修飾した整数型の自動変数に格納し、当該の自動変数を参照する形で利用してください。

(例)発生条件の(注)に示す例では、以下の様に変更してください。

```
volatile int int_globalVariableAddress = (int)&globalVariable;
array[i + int_globalVariableAddress] = 0;
```

③ -cpu=S1 オプション指定、-memory_model オプション未指定時に、-D __RL78_MEDIUM__が指定される注意事項
-cpu=S1 オプションを指定し、-memory_model オプションは指定しない場合に、-D __RL78_SMALL__が指定されずに、誤って-D __RL78_MEDIUM__が指定されます。
【発生条件】
以下の条件を全て満たす場合に発生します。
(1)-cpu=S1 オプションを指定する。
(2)-memory_model オプションを指定しない。
【回避策】
以下のいずれかの方法で回避できます。
(1)-memory_model=small を指定する。
(2)-D __RL78_SMALL -U __RL78_MEDIUM__を指定してください。

④ CRC 計算範囲に padding 拡張した領域を含めた場合、CRC 計算結果が期待した結果とならない注意事項

CRC 計算範囲に padding 拡張した領域を含めた場合、CRC 計算結果は期待する値とならないことがあります。

【発生条件】

以下の条件を全て満たす場合に発生します。

(1)-padding オプションを指定してパディングする。

(2)-crc オプションを指定する。

(3)CRC 計算範囲に padding 拡張した領域が含まれる。

【回避策】

以下のいずれかの方法で回避できます。

(1)-padding オプションを指定しない。

(2)CRC 計算範囲に padding 拡張した領域を含めない。

第5章　注意事項

CC-RL V1.01.00 の注意事項についてはマニュアルを参照してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/