CS+ RH850 コンパイラ CC-RH V1.06.00 リリースノート

R20UT4068JJ0100
Rev.1.00
2017.6.20

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-RH 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：

https://www.renesas.com/cs+

第2章　ユーザーズ･マニュアルについて

本製品に関連したユーザーズ･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-RH コンパイラ ユーザーズマニュアル	R20UT3516JJ0103
CS+ 統合開発環境 ユーザーズマニュアル CC-RH ビルド・ツール操作編	R20UT3283JJ0104

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-RH V1.06.00」を選択してください。

第4章　変更点

本章では、CC-RH の変更点について説明します。

4.1 CC-RHの変更点

CC-RH V1.05.00 から V1.06.00 への主な変更点を説明します。

なお、professional 版のライセンス登録時のみ使用できる機能は【professional】と明記します。

4.1.1 MISRA-C:2012ルールによるソース・チェック機能の拡充【professional】

MISRA-C:2012 ルールによりソース・チェックを行う-Xmisra2012 オプションの引数に、以下の番号を指定できるようにしました。
また、V1.06.00 より MISRA C:2012 Amendment 1 をサポートしました。

【必須ルール】　**12.5,　21.13**

【必要ルール】　**13.2,　13.5,　21.15,　21.16**

【推奨ルール】　**17.5,　17.8**

各リビジョンでチェック可能な MISRA-C:2012 ルール数は下記の通りです。

ルール分類（ルール数）	*V1.03.00*	*V1.04.00*	*V1.05.00*	*V1.06.00*
必須ルール（16）	3	3	4	**6**
必要ルール（108）	31	58	76	**80**
推奨ルール（32）	7	21	23	**25**
合計ルール（156）	41	82	103	**111**

4.1.2 制御レジスタ更新時の同期化機能【professional】

RH850 のストア命令によって複数の制御レジスタを連続して更新する際、制御レジスタの更新順序をソース・ファイルの記述順と一致させるには、制御レジスタの更新間に同期化処理を挿入しなければなりません。ただし、同一グループの制御レジスタを連続して更新する場合は、同期化処理は不要です。

CC-RH の professional 版では、制御レジスタの更新箇所の検出、及び、複数の制御レジスタを連続して更新する箇所に同期化処理を挿入する機能をサポートしました。

下記の手順により本機能を有効にします。

1. 制御レジスタのアドレス範囲とグループ情報を拡張言語で指定

#pragma register_group *開始アドレス*, *終端アドレス*[, id="*グループID*"]

2. コンパイラ・オプションを指定

-store_reg[=*mode*]

コンパイラ・オプションの引数"*mode*"には次のいずれかを指定します

- ***list***

  #pragma register_group で指定した制御レジスタの更新箇所を検出し、ソース上の記述位置を標準エラー出力に表示します。なお、後続命令が同一グループの制御レジスタの更新であると判断できる場合は表示しません。

- ***list_all***

  #pragma register_group で指定した制御レジスタの更新箇所を検出し、ソース上の記述位置を標準エラー出力に表示します。後続命令が同一グループの制御レジスタの更新であるかどうかに関わらず表示します。

- ***sync***

  #pragma register_group で指定した制御レジスタの更新箇所を検出し、更新後に同期化処理を挿入します。なお、後続命令が同一グループの制御レジスタの更新であると判断できる場合は挿入しません。

- ***ignore***

  #pragma register_group に対して警告を出力せずに無視します。

本機能の使用例を示します。

【C ソースファイル”scr.c”の記述例】

```
1:  #pragma register_group 0xfedf0000, 0xfedfffff, id=”CPU”
2:  #pragma register_group 0xfee00000, 0xfee0ffff, id=”0”
3:
4:  #define REG1 (*(volatile unsigned char*)0xfedf0000)      /* CPUグループの制御レジスタ  */
5:  #define REG2 (*(volatile unsigned char*)0xfedf0001)      /* CPUグループの制御レジスタ  */
6:  #define REG3 (*(volatile unsigned short*)0xfee00000)    /* 0グループの制御レジスタ  */
7:
8:  void func(void) {
9:      REG1 = 0;
10:     REG2 = 1;
11:     REG3 = 2;
12: }
```

関数 func 内で REG1（CPU グループ）→REG2（CPU グループ）→REG3（0 グループ）の順番で連続して制御レジスタを更新しています。

このとき、コンパイラは次のように判定し、コンパイラ・オプションの引数”*mode*”に応じて下記のような処理を行います。

✓ REG1 と REG2 は**同じグループのため**、REG1 の更新後に同期化処理は**不要**

✓ REG2 と REG3 は**異なるグループのため**、REG2 の更新後に同期化処理が**必要**

✓ REG3 の更新後の処理が不明のため、REG3 の更新後に同期化処理が**必要**

- -store_reg=list オプションを指定時

  標準エラー出力に以下のメッセージを表示します。

```
src.c(10):M0536001:制御レジスタを更新します。(id=CPU, 0xfedf0001)
src.c(11):M0536001:制御レジスタを更新します。(id=0, 0xfee00000)
```

- -store_reg=list_all オプションを指定時

  標準エラー出力に以下のメッセージを表示します。

```
src.c(9):M0536001:制御レジスタを更新します。(id=CPU, 0xfedf0000)
src.c(10):M0536001:制御レジスタを更新します。(id=CPU, 0xfedf0001)
src.c(11):M0536001:制御レジスタを更新します。(id=0, 0xfee00000)
```

- -store_reg=syncp オプションを指定時

  出力コード中に同期化処理を挿入します。

  同期化処理として、同じ制御レジスタからのロード命令と、syncp 命令を組み合わせて出力します。

```
_func:
        .stack _func = 0
        movhi 0x0000FEDF, r0, r2
        st.b r0, 0x00000000[r2]          ; REG0 の更新
        ; 後続が同じグループの制御レジスタの更新のため同期化処理を挿入しない
        movhi 0x0000FEDF, r0, r2
        mov 0x00000001, r5
        st.b r5, 0x00000001[r2]          : REG1 の更新
        ld.bu 0x00000001[r2], r10        ; 同じ制御レジスタのロード
        syncp                            ; 同期化命令
        movhi 0x0000FEE0, r0, r2
        mov 0x00000002, r5               ; REG2 の更新
        st.h r5, 0x00000000[r2]
        ld.hu 0x00000002[r2], r10        ; 同じ制御レジスタのロード
        syncp ;                          ; 同期化命令
        jmp [r31]
```

4.1.3 最適化強化

主に以下のような最適化を実装することにより、生成コードの性能を改善しました。

① switch 文の改善

<ソースコード例>

```
void sub(int);
void func(int key) {
    switch(key & 0x3){
    case 0:
        sub(0); break;
    case 1:
        sub(1); break;
    case 2:
        sub(2); break;
    case 3:
        sub(3); break;
    default:
        sub(4); break;
    }
}
```

3 行目の(key & 0x3) の結果は 0, 1, 2, 3 のいずれかのため、V1.06.00 では、default のブロックには到達しないことを考慮した最適化を行います。

<V1.05.00 の生成コード>

```
_func:
    .stack _func = 4
    prepare 0x00000001, 0x00000000
    andi 0x00000003, r6, r2
    cmp 0x00000003, r2
    bh9 .BB.LABEL.1_6
.BB.LABEL.1_1:   ; entry
    shl 0x00000001, r2
    jmp #.SWITCH.LABEL.1_7[r2]
.SWITCH.LABEL.1_7:
    br9 .BB.LABEL.1_2
    br9 .BB.LABEL.1_3
    br9 .BB.LABEL.1_4
    br9 .BB.LABEL.1_5
.SWITCH.LABEL.1_7.END:
.BB.LABEL.1_2:   ; bb
    mov 0x00000000, r6
    jarl _sub, r31
    dispose 0x00000000, 0x00000001, [r31]
.BB.LABEL.1_3:   ; bb2
    mov 0x00000001, r6
    jarl _sub, r31
    dispose 0x00000000, 0x00000001, [r31]
.BB.LABEL.1_4:   ; bb3
    mov 0x00000002, r6
    jarl _sub, r31
    dispose 0x00000000, 0x00000001, [r31]
.BB.LABEL.1_5:   ; bb4
    mov 0x00000003, r6
    jarl _sub, r31
    dispose 0x00000000, 0x00000001, [r31]
.BB.LABEL.1_6:   ; bb5
    mov 0x00000004, r6
    jarl _sub, r31
    dispose 0x00000000, 0x00000001, [r31]
```

<V1.06.00 の生成コード>

```
_func:
    .stack _func = 4
    prepare 0x00000001, 0x00000000
    andi 0x00000003, r6, r2
.BB.LABEL.1_1:   ; entry
    shl 0x00000001, r2
    jmp #.SWITCH.LABEL.1_6[r2]
.SWITCH.LABEL.1_6:
    br9 .BB.LABEL.1_2
    br9 .BB.LABEL.1_3
    br9 .BB.LABEL.1_4
    br9 .BB.LABEL.1_5
.SWITCH.LABEL.1_6.END:
.BB.LABEL.1_2:   ; bb
    mov 0x00000000, r6
    jarl _sub, r31
    dispose 0x00000000, 0x00000001, [r31]
.BB.LABEL.1_3:   ; bb2
    mov 0x00000001, r6
    jarl _sub, r31
    dispose 0x00000000, 0x00000001, [r31]
.BB.LABEL.1_4:   ; bb3
    mov 0x00000002, r6
    jarl _sub, r31
    dispose 0x00000000, 0x00000001, [r31]
.BB.LABEL.1_5:   ; bb4
    mov 0x00000003, r6
    jarl _sub, r31
    dispose 0x00000000, 0x00000001, [r31]
```

② 別名解析

<ソースコード例>

```
struct tag1 {
    char member1;
    int  member2;
    long long member3;
} StructArray[2];

struct tag2 {
    short index0;
    short index1;
    short index2;
};

void func(struct tag2 *p) {
    StructArray[p->index1].member1 = 1;
    StructArray[p->index1].member2 = 2;
    StructArray[p->index1].member3 = 3;
}
```

StructArray[p->index1]のアドレス計算を 3 回実施していたものを V1.06.00 では 1 回のみ実施します。

-Xalias=ansi オプション指定時のみ有効となります。

<V1.05.00 の生成コード>

```
_func:
        .stack _func = 0
        ld.h 0x00000002[r6], r2
        shl 0x00000004, r2
        mov #_StructArray, r5
        add r5, r2
        mov 0x00000001, r7
        st.b r7, 0x00000000[r2]
        ld.h 0x00000002[r6], r2
        shl 0x00000004, r2
        add r5, r2
        mov 0x00000002, r7
        st.w r7, 0x00000004[r2]
        ld.h 0x00000002[r6], r2
        shl 0x00000004, r2
        add r2, r5
        mov 0x00000003, r2
        st.w r2, 0x00000008[r5]
        st.w r0, 0x0000000C[r5]
        jmp [r31]
```

<V1.06.00 の生成コード>

```
_func:
        .stack _func = 0
        ld.h 0x00000002[r6], r2
        shl 0x00000004, r2
        mov #_StructArray, r5
        add r2, r5
        mov 0x00000001, r2
        st.b r2, 0x00000000[r5]
        mov 0x00000002, r2
        st.w r2, 0x00000004[r5]
        mov 0x00000003, r2
        st.w r2, 0x00000008[r5]
        st.w r0, 0x0000000C[r5]
        jmp [r31]
```

4.1.4 CAN通信時間計測ソリューション対応

CS+で E2 エミュレータを使用した CAN 通信時間計測ソリューション機能を使用するため、**-insert_dbtag_with_label** オプションを追加しました。本オプションを指定すると、CS+のエディタ上で指定した行にソフトウェアトレース命令の 1 つである DBTAG 命令を生成します。

本オプションは CAN 通信時間計測ソリューション機能を使用する際に CS+が使用するオプションであり、ユーザーがビルド時に指定するものでありません。

4.1.5 配列初期値データの出力方法変更

配列型の初期値データをアセンブリ・ソース上で出力する際、1 行にまとめて出力するように変更しました。この変更により、ビルド時の解析処理が削減され、ビルド時間を短縮することができます。

<ソースコード例>

```
float flt[4] = {1,2,3,4};
```

<V1.05.00 の生成コード>

```
_flt:
    .dw 0x3F800000   ; float value: 1
    .dw 0x40000000   ; float value: 2
    .dw 0x40400000   ; float value: 3
    .dw 0x40800000   ; float value: 4
```

<V1.06.00 の生成コード>

```
_flt:
    .dw 0x3F800000,0x40000000,0x40400000,0x40800000 ; float 1,2,3,4
```

4.1.6 特殊シンボルの追加

セクションの先頭、及び、セクションの終端にリンカが生成する特殊シンボルに、C ソースで参照できる形式を追加しました。

- セクションの先頭アドレス値を値として持つ予約シンボル：
  セクション名の「.」あるいは「@」の文字を「_」に置き換え、先頭に「__S」を付加したシンボル
- セクションの終端を越える最初のアドレス値を値として持つ予約シンボル：
  セクション名の「.」あるいは「@」の文字を「_」に置き換え、先頭に「__E」を付加したシンボル

.text セクションの先頭アドレスを値として持つ予約シンボルは、「__S_text」になります。C ソース上で参照する場合は"_（アンダースコア）"を 1 つ削除した「_S_text」になります。

<ソースコード例>

```
extern unsigned long _S_text;

unsigned long* func(void) {
  return &_S_text;   /* .text セクションの先頭アドレスを取得 */
}
```

4.1.7 リンク・マップ・ファイルの仕様変更

リンク・マップ・ファイルの”Mapping List” に再配置属性の ATTRIBUTE 欄を追加しました。

-show=relocation_attribute を指定した場合に、セクションに対応している再配置属性を TEXT, CONST, DATA, BSS の 4 種類に分類して表示します。再配置属性の出力例を以下に示します。

```
*** Mapping List ***

SECTION                 START       END         SIZE    ALIGN   ATTRIBUTE

.text
                        00000100    0000013b    3c      2       TEXT
.data
                        000f0400    000f0403    4       4       DATA
.bss
                        000f0404    000f040b    8       4       BSS
```

また、リンク・エラー発生時にリンク・マップ・ファイルの”Error information”に出力するエラーの種類を増やしました。

4.1.8 モトローラ・Sタイプ・ファイルのエンドレコード指定

モトローラ・S タイプ・ファイルのエンドレコードを指定するリンカ・オプション**-end_record** を追加しました。V1.06.00 未満ではエントリ・ポイント・アドレスに合わせてエンドレコードを出力していましたが、指定したエンドレコードでモトローラ・S タイプ・ファイルを生成できます。

```
-end_record=record
```

引数”*record*”には ***S7***, ***S8***, ***S9*** のいずれかを指定可能です。

4.1.9 無償評価版使用時メッセージの番号追加

無償評価版を使用してビルドする際に出力していたメッセージに対し、番号 W0561016, W0561017 を付与しました。これにより、無償評価版使用時にエラーとして処理する等、-change_message による制御ができるようになりました。

W0561016:The evaluation version is valid for the remaining *** days

W0561017:The evaluation period has expired

4.1.10 リンク・エラーのメッセージ変更

リンク・エラー”F0563102” のメッセージに、ファイル名を出力するようにしました。

- V1.06.00 未満

  F0563102:Section contents overlap in absolute section "*セクション名*"

- V1.06.00 以上

  F0563102:Section contents overlap in absolute section "*セクション名*" in "*ファイル名*"

4.1.11 ライセンスの認証方式の改善

ライセンスの認証方式を改善し、ビルド時間を短縮しました。
本改善に伴い、professional 版のライセンスが登録されていない場合に professional 版の#pragma 拡張言語を記述した際のコンパイラの動作が以下のように異なります。

- V1.06.00 未満
  - 警告を出力して無視
- V1.06.00 以上
  - #pragma 拡張言語の構文が正しい場合、警告を出力して無視
  - #pragma 拡張言語の構文に誤りがある場合、エラーを出力

4.1.12 注意事項の改修

以下 1 件の注意事項を改修しました。注意事項の詳細につきましてはツールニュースをご確認ください。

- #pragma pmodule 拡張言語の注意事項（No.15）

4.1.13 その他変更・改善

主に以下のような変更・改善を行いました。

(a) コンパイル・エラーの改善

-Xmerge_filesオプション指定時にコンパイル・エラー”F0530800”が発生することがありましたが、これを改善しました。

(b) 内部エラーの改善

ビルド時に内部エラーが発生する場合がありましたが、これを改善しました。

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生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。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）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   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（宇宙、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に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損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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。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. 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、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）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、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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。
12. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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。

(Rev.3.0-1　2016.11)

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　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■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：https://www.renesas.com/contact/