RX ファミリ用 C/C++コンパイラパッケージ CC-RX V3.03.00 リリースノート

R20UT4877JJ0101
Rev.1.01
2021.01.16

この度は、弊社製品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 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。
本文章と合わせてお読みください。

マニュアル名	資料番号
CC-RX コンパイラ ユーザーズマニュアル	R20UT3248JJ0110
(統合開発環境 CS+と併用する場合) CS+ 統合開発環境 ユーザーズマニュアル CC-RX ビルド・ツール操作編	R20UT3478JJ0108

第2章 変更点

本章では、CC-RX V3.02.00 から V3.03.00 への主な変更点について説明します。

2.1 -branch_chaining,-nobranch_chaining オプションの追加

分岐命令のコードサイズを削減する最適化を行う-branch_chaining オプションを追加しました。

本オプションを指定した場合、分岐命令を最終的な分岐先まで直接分岐するのではなく、コードの小さな分岐命令を使用し、分岐先を行き先が同じ他の分岐命令にすることがあります。この時、実行速度は低下しますが、コードサイズが小さくなります。

-nobranch_chaining オプションを指定することで、この最適化を抑止することができます。

2.2 -VERBOSE オプションの追加

リンク時の詳細情報を表示する-VERBOSE オプションを追加しました。
パラメータに crc を指定することにより、CRC 演算の結果および出力位置アドレスを表示できるようになりました。

2.3 MULHI,MACHI,MULLH,MACLH 命令の生成

MULHI,MACHI,MULLH,MACLH 命令の生成をサポートしました。本機能を有効にするには次に示す 3 つのオプションを指定してください。

- -optimize=2 もしくは-optimize=max
- -speed
- -save_acc

MULLH または MACLH 命令の生成を有効にする場合には、上記オプションにあわせて次のオプションを指定してください。

- -isa=rxv2 もしくは-isa=rxv3

次のソースコードでは、コードサイズが小さくなり、実行速度が向上します。

<ソースコード例>

```
signed long mulhi(signed long lhs0, signed long rhs0,
                  signed long lhs1, signed long rhs1,
                  signed long lhs2, signed long rhs2) {
   lhs0 >>= 16;
   lhs1 >>= 16;
   lhs2 >>= 16;
   rhs0 >>= 16;
   rhs1 >>= 16;
   rhs2 >>= 16;
   return (lhs0 * rhs0 + lhs1 * rhs1 + lhs2 * rhs2);
}

signed long mac(signed long src, signed long lhs, signed long rhs) {
   signed short lhs0 = (signed short) lhs;
   signed short lhs1 = (signed short) (lhs >> 16);
   signed short rhs0 = (signed short) rhs;
   signed short rhs1 = (signed short) (rhs >> 16);
   src += lhs0 * rhs1;
   src += lhs1 * rhs0;
   src += lhs1 * rhs1;
   return (src);
}
```

<V3.02.00 (-isa=rxv2 -speed -save_acc)>

```
_mulhi:
        .STACK _mulhi = 4
        MOV.L 04H[R0], R5
        SHAR #10H,    R3
        MOV.L 08H[R0], R15
        SHAR #10H, R4
        MULLO R4, R3
        SHAR #10H, R1
        SHAR #10H, R2
        SHAR #10H, R5
        MACLO R2, R1
        SHAR #10H, R15
        MACLO R15, R5
        MVFACMI R1
        RTS

_mac:
        .STACK _mac = 4
        SHAR #10H, R2, R14
        MULLO R14, R3
        SHAR #10H, R3
        MACLO R3, R14
        MACLO R3, R2
        MVFACMI R14
        ADD R14, R1
        RTS
```

<V3.03.00 (-isa=rxv2 -speed -save_acc)>

```
_mulhi:
        .STACK _mulhi = 4
        MULHI R4, R3
        MOV.L 04H[R0], R5
        MOV.L 08H[R0], R3
        MACHI R2, R1
        MACHI R3, R5
        MVFACMI R1
        RTS

_mac:
        .STACK _mac = 4
        MULLH R3, R2, A0
        MACHI R3, R2
        MACLH R2, R3, A0
        MVFACMI R14
        ADD R14, R1
        RTS
```

2.4 別名解析の精度改善

組み込み関数呼び出しや集成体のコピーをまたいでメモリアクセス命令を移動するなどの、最適化を適用しやすくなるように改善しました。

次のソースコードでは、コードサイズが小さくなり、実行速度が向上します。

```
<ソースコード例>
unsigned short ShortArray[2];
signed long LongArray[2];
void test(void) {
    ShortArray[0] = 0;
    __xchg(&LongArray[0], &LongArray[1]);
    ShortArray[1] = 0;
}
```

<V3.02.00 (-isa=rxv1)>	<V3.03.00 (-isa=rxv1)>
_test:	_test:
.STACK __test = 4	.STACK _test = 4
MOV.L #_ShortArray, R1	MOV.L #_ShortArray, R14
MOV.W #0000H, [R1]	MOV.L #00000000H, [R14]
MOV.L #_LongArray, R14	MOV.L #_LongArray, R14
MOV.L [R14], R15	MOV.L [R14], R15
ADD #04H, R14, R5	ADD #04H, R14, R5
XCHG [R5].L, R15	XCHG [R5].L, R15
MOV.L R15, [R14]	MOV.L R15, [R14]
MOV.W #0000H, 02H[R1]	RTS
RTS	

2.5 注意事項の改修

以下 4 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。

- rmpab, rmpaw, rmpal または memchr を使用する場合の注意事項 (No.55)
- 末尾呼び出し最適化に関する注意事項 (No.56)
- -ip_optimize オプションの使用に関する注意事項 (No.57)
- 多次元配列を使用する場合の注意事項 (No.58)

第3章 注意事項

本章では、CC-RX の注意事項について説明します。

3.1 W0523041 メッセージが出力される場合の注意事項[C/C++コンパイラ]

C 標準ヘッダをインクルードしたファイルを C++または EC++コンパイルしたとき、-int_to_short オプションを指定すると W0523041 メッセージが出力されることがあります。この場合は動作には問題ありませんので無視してください。

【注意】
C++および EC++コンパイル時は、-int_to_short オプションの指定は無効になります。
C と C++(EC++)との間で共通にアクセスするデータは、int 型ではなく long 型または short 型で宣言してください。

3.2 MVTC、POPC 命令を使用する場合の注意事項 [アセンブラ]

アセンブリ言語において、MVTC、POPC 命令に対してプログラムカウンタ(PC)は指定できません。

3.3 -delete オプションをリンク時に指定する場合の注意事項[最適化リンケージエディタ]

-delete オプションで指定した関数シンボルが削除された場合、削除された関数定義の次の関数定義の関数名に対して、デバッグ時にエディタ上でブレークポイントを設定することができません。ラベルウィンドウからブレークポイントを設定するか、関数のプログラム行で指定してください。

3.4 パス名の指定に関する注意事項

入出力ファイル指定やフォルダ指定に使用できるパス名はドライブ名を含む絶対パス、もしくは相対パスです。また、指定可能なパス名の長さは 259 文字です。

第4章 制限事項

本章では、CC-RX の制限事項について説明します。

4.1 C++言語(EC++含む)で math.h の一部関数(frexp、ldexp、scalbn および remquo)を使用する場合の制限事項

C++/EC++コンパイル時に、math.h の一部の関数(frexp、ldexp、scalbn、remquo)の実引数を int 型にすると、実行時に無限ループとなるオブジェクトが生成されます。

発生条件:
次の条件(1)(2)を全て満たす場合が該当します。
(1) C++ソース(拡張子が.cpp)または、-lang=cpp オプションが有効である。
(2) math.h をインクルードして、以下の関数をそれぞれの条件で呼び出している。
(a) frexp(double, long *) の第 2 引数の値を (int *)型とする
ただし、第 1 引数が float 型で、-dbl_size=8 オプション指定時を除く
(b) ldexp(double, long) の第 2 引数の値を int 型とする
ただし、第 1 引数が float 型で、-dbl_size=8 オプション指定時を除く
(c) scalbn(double, long) の第 2 引数の値を int 型とする
ただし、第 1 引数が float 型で、-dbl_size=8 オプション指定時を除く
(d) remquo(double, double, long *) の第 3 引数の値を (int *)型とする
ただし、第 1 または第 2 引数が float 型で、-dbl_size=8 オプション指定時を除く

発生例:
[file.cpp]

```
// C++ソースとしてコンパイルした場合に無限ループになる例
#include <math.h>
double d1,d2;
int i;
void func(void)
{
  d2 = frexp(d1, &i);
}
```

[コマンドライン例]

```
ccrx -cpu=rx600 -output=src file.cpp
```

[file.src] ソース出力例

```
_func:
```

```
    ; . . . (中略)
    BSR __$frexp__tm__2_f__FZ1ZPi_Q2_21_Real_type__tm__4_Z1Z5_Type ; frexp の代替関数を呼ぶ
    ; . . . (中略)

__$frexp__tm__2_f__FZ1ZPi_Q2_21_Real_type__tm__4_Z1Z5_Type:
L11:
    BRA L11 ; 再帰呼び出しとなってしまう
```

回避策：

次のいずれかの方法で回避できます。

(1) -lang=c を指定し、C 言語としてコンパイルする。

(2) 引数の int および int*を long および long*に変更する。

(3) math.h の後に、使用する関数ごとに定義を追加する。

```
/* frexp 関数の場合 */
static inline double frexp(double x, int *y)
{ long v = *y; double d = frexp(x,&v); *y = v; return (d); }
/* ldexp 関数の場合 */
static inline double ldexp(double x, int y)
{ long v = y; double d = ldexp(x,v); return (d); }
/* scalbn 関数の場合 */
static inline double scalbn(double x, int y)
{ long v = y; double d = scalbn(x,v); return (d); }
/* remquo 関数の場合 */
static inline double remquo(double x, double y, int *z)
{ long v = *z; double d = remquo(x,y,&v); *z = v; return (d); }
```

回避策(2)の例

[file.cpp] の変更例

```
#include <math.h>
double d1,d2;
int i;
void func(void)
{
     long x = i; /* 一旦 long 型変数で受ける */
     d2 = frexp(d1, &x); /* long 型変数で呼び出し */
     i = x; /* i に値を設定 */
}
```

回避策(3)の例

[file.cpp] の変更例

```
#include <math.h>
/* 宣言を追加 */
static inline double frexp(double x, int *y)
{ long v = *y; double d = frexp(x,&v); *y = v; return (d); }
double d1,d2;
int i;
void func(void)
{
     d2 = frexp(d1, &i);
}
```

4.2 PIC/PID 機能ご利用に関する制限事項 (-pic および-pid オプション)

ライブラリジェネレータ lbgrx に-pic または-pid オプションを指定して、標準ライブラリを作成することができますが、その際に、次の警告が 1 回または複数回表示されます。

W0591301:"-pic" option ignored (-pic オプションを指定した場合)

W0591301:"-pid" option ignored (-pid オプションを指定した場合)

これらの警告は、EC++ライブラリに対して、-pic、-pid オプションが無効になるため出力されます。

4.3 以下のオプションを廃止しました【C/C++コンパイラ】

(a) -file_inline、 -file_inline_path

指定しても警告を表示し無効となります。代替手段としては、ソース中に#include を記述してください。C(C99 含む)の場合は、同様の機能を持つ-merge_files も使用できます。

(b) -enable_register

指定しても無視されます。指定の有無による生成コードの変化はありません。

4.4 C/C++ソースレベルデバッグに関する注意事項【C/C++コンパイラ】

(a) -debug オプションを指定しても、次の行に対しては、デバッガでブレークポイントの設定やステップ実行で停止ができない場合があります。

・グローバル変数の動的な初期化式(C++)

・次のような関数の先頭行

#pragma inline_asm が指定されている関数。

ループ文(do 文、while 文など)から始まり、かつ auto 変数を持たない関数。

・ループ(for 文、 while 文、do 文)の制御式と本体を 1 行で記述した場合の制御式および本体部分。

(b) 共用体型かつレジスタ渡しである仮引数のメンバが、ウォッチウィンドウ等で誤った値が表示される場合があります。

4.5 0xffffffff 番地を含むセクションを記述する場合の注意事項【アセンブラ】

アセンブリソース中に同じセクション名に対する.org 制御命令を含む.section 制御命令が複数あり、これらが互いに 0xffffffff 番地で重複している場合、アセンブル時に内部エラー(C0554098)が発生します。

例)

```
.section SS,ROMDATA
.org 0fffffffeh
.byte 1
.byte 2 ; 0xffffffff
.section SS,ROMDATA
.org 0ffffffffh
.byte 3; ; 0xffffffff
.end
```

4.6 -form と-output オプションを同時に使用する場合の注意事項【リンカ】

リンカ(rlink)に、-form=rel と-output=出力ファイル名を共に指定したとき、出力ファイル名の拡張子が無視され、常に.rel に置き換わります。

例) rlink -form=relocate -output=DefaultBuild¥lib_test.lib

出力ファイルを test.lib とすべきところが test.rel になります。

4.7 _builtin から始まる関数名を使用する場合の注意事項【C/C++コンパイラ】

include ディレクトリ内の machine.h に記述のある _builtin から始まる関数名をソースコードで宣言した場合、内部エラーになることがあります。アンダーバー(_)から始まる名前は予約されていますので、お客様のソースコードには記述しないでください。

4.8 save_acc が有効な#pragma interrupt 指定された関数が仮引数を持つ場合の注意事項【C/C++コンパイラ】

#pragma interrupt で指定された関数で、save_acc フラグが有効(-save_acc オプション指定時を含む)な場合、生成コードが R4 で渡される仮引数を正常に取得できない場合があります。

注意) #pragma interrupt で指定された関数に仮引数を定義することは、推奨しておりません。

4.9 -merge_files の制限事項

共用体型変数を記述した翻訳単位を -merge_files を指定してコンパイルすると F0530800 が発生する制限事項

[発生条件]
次の条件を全て満たす場合、F0530800 または W0530811 が発生する可能性があります。
(1) -merge_files または -whole_program を指定している。
(2) 2 つ以上のメンバを持つ共用体型外部変数を初期化しており、かつ初期化対象のメンバ以外に、アライメントとサイズの両方が他のいずれのメンバよりも大きいメンバが存在する。
(3) (2) に該当する変数を次のいずれかで extern 宣言し、参照している。
(3-1) (2) の外部変数定義が存在するソースファイルと別のソースファイル。
(3-2) (2) の外部変数定義が存在するソースファイルと別のソースファイルから、直接、または間接的にインクルードされているヘッダファイル。

[回避策]
次のいずれかの対応を実施してください。
(1) 発生条件 (1) のオプションをいずれも指定しない。
(2) 発生条件 (2) における "初期化" を関数内で実施する。
(3) 発生条件 (2) に該当する変数を、その外部変数定義を記述したソースファイル内でのみ参照する。

4.10 -cfi_ignore_module オプションの制限事項

C/C++ソースファイルを-output=abs を使用してコンパイルした時、生成されるオブジェクトファイルを -cfi_ignore_module で指定できません。
-output=obj を使用してオブジェクトファイルを生成し、その生成したオブジェクトファイルを -cfi_ignore_module で指定してください。

4.11 -dpfpu 指定時の fenv.h 利用に関する制限事項

fenv.h で提供される次の標準ライブラリ関数は、-dpfpu を指定してコンパイルした場合においても、FPSW レジスタの値のみが設定・参照され、DPSW レジスタは値が設定・参照されません。

feclearexcept
fegetexceptflag
feraiseexcept
fesetexceptflag
fetestexcept
fegetround
fesetround
fegetenv
feholdexcept

fesetenv

feupdateenv

DPSW レジスタの値を設定・参照したい場合は、組み込み関数__set_dpsw/__get_dpsw を使用してください。

第5章 添付標準ライブラリについて

本章では、RX ファミリ向け C/C++コンパイラ 添付標準ライブラリについて説明します。

本製品では、RX600 用に標準ライブラリファイル(*.lib)を 4 種類添付しています。
添付の標準ライブラリファイルを使用することにより、ビルドに要する時間を短縮することができます。

5.1 添付ライブラリー覧

本製品に添付される、標準ライブラリファイルの一覧を表 5.1に示します。

【ご注意】

ライブラリで選択している「マイコンオプション」は、ご使用のコンパイラオプションと一致させる必要があります。いずれとも一致しない場合は、これらの標準ライブラリは使用できませんので、ご利用のコンパイラオプション(アセンブラオプション)をライブラリジェネレータに指定して、作成されるライブラリをご使用ください。

ライブラリ名	用途	最適化*2 オプション	マイコンオプション *1 *2		
			-endian	-cpu -rtti -exception -noexception	その他 *3
rx600lq.lib	RX600、速度優先 リトルエンディアン	-speed -goptimize	-endian=little	-cpu=rx600 -rtti=on -exception	-round=nearest -denormalize=off -dbl_size=4 -unsigned_char -unsigned_bitfield -bit_order=right -unpack -fint_register=0 -branch=24
rx600ls.lib	RX600、サイズ優先 リトルエンディアン	-size -goptimize			
rx600bq.lib	RX600、速度優先 ビッグエンディアン	-speed -goptimize	-endian=big		
rx600bs.lib	RX600、サイズ優先 ビッグエンディアン	-size -goptimize			

表 5.1 ライブラリー覧

*1 マイコンオプションは、ユーザーズマニュアル RX ビルド編の「A.1.3 オプション」「(1)コンパイル・オプション」「 マイコンオプション」を参照ください。

*2 これらのオプション選択は省略時設定と同じです。

5.2 ライブラリ指定の方法

添付の標準ライブラリファイルを使用する場合は、製品に含まれているライブラリファイルを、lib ディレクトリから任意のディレクトリにコピーしてください。

次に、最適化リンケージエディタの-Library オプションにコピーしたライブラリファイルを指定して、リンクしてください。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
Rev.1.00	2020.12.01		新規作成
Rev.1.01	2021.01.16	4	2.1 -nobranch_chaining の説明追加
		5,6,7	サンプルコード修正

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。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、お客様の責任において、お客様の機器・システムを設計ください。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. 当社製品または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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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。
6. 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、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、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（宇宙機器と、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想定していません。たとえ、当社が想定してい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。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 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。
9. 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、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、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。
10. 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。
11. 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。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。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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.5.0-1　2020.10)

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/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。