RX ファミリ C/C++コンパイラパッケージ V.1.00 Release 01 注意事項およびユーザーズマニュアル修正

RX ファミリ C/C++コンパイラ V.1.00 Release 01 には、以下の注意事項および添付ユーザーズマニュアル(RJJ10J2570-0100)に以下に示す項目を追加いたしますので、マニュアルを修正してくださるようお願い申し上げます。

1. 注意事項

1.1 C1804 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 C1804 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。

1.2 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。

1.3 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。

1.4 パス名、ファイル名に関する注意事項

[最適化リンケージエディタ]

最適化リンケージエディタでは、括弧記号"("および")"をオプション指定で使用するので、パス名およびファイル名に括弧記号を含まないようにしてください。

1.5 I/O ライブラリを使用する場合の注意事項

[統合開発環境(プロジェクト生成時)]

統合開発環境でプロジェクトを生成するときに、以下の画面で［I/O ライブラリ使用］を有効にする場合には、同時に［ヒープメモリ使用］も有効にしてください。

[I/O ライブラリ使用］を有効にした状態で［ヒープメモリ使用］を無効にすると、以下のリンクエラーとなります。

```
L2310 (E) Undefined external symbol "_sbrk" referenced in "xgetmem"
```

もし上記のリンクエラーが出るプロジェクトを生成した場合には、以下の C 言語ソースファイルをプロジェクトに追加登録して回避してください。

```
#include <stddef.h>
#include <stdio.h>

#define HEAPSIZE 0x400

signed char  *sbrk(size_t size);

union HEAP_TYPE {
    signed long  dummy ;
    signed char  heap[HEAPSIZE];
};

static union HEAP_TYPE heap_area ;

/* End address allocated by sbrk     */
static signed char *brk=(signed char *)&heap_area;

signed char  *sbrk(size_t size)
{
    signed char  *p;

    if(brk+size > heap_area.heap+HEAPSIZE){
        p = (signed char *)-1;
    }
    else {
        p = brk;
        brk += size;
    }
    return p;
}
```

2. マニュアル修正事項

2.1 追加・修正

以下の項目を訂正いたします。

■(p.14) 2.1 ソースオプション / check オプション / check=nc

＜修正内容＞

[変更前]

check=nc を指定した場合、以下に対してメッセージを出力します。

[変更後]

check=nc 指定時にチェックされるオプションや型には、次のようなものがあります。

■(p.14) 2.1 ソースオプション / check オプション / check=ch38

＜修正内容＞

[変更前]

check=ch38 を指定した場合、以下に対してメッセージを出力します。

[変更後]

check=ch38 を指定時にチェックされるオプションや型には、次のようなものがあります。

■(p.26) 2.4 最適化オプション inline,noinline

＜追加内容＞

備考:

<数値>として使える値の範囲は 0～65535 です。

■(p.28) 2.4 最適化オプション case / 説明

＜修正内容＞

[誤] ジャンプテーブルは、定数領域のセクションに出力されます。

[正] ジャンプテーブルは、switch 文分岐テーブル領域のセクションに出力されます。

■(p.31) 2.4 最適化オプション map,smap,nomap / 説明

＜修正内容＞

[誤]

[output=obj または src の場合]

[正]

[output=obj の場合]

■(p.31) 2.4 最適化オプション map,smap,nomap / 説明

＜追加内容＞

備考:

C/C++ソースをコンパイルする場合のみ適用されます。コンパイル時に output=src を用いて作成、またはアセンブリ言語で記述されたプログラムには適用されません。

■(p.39) 2.5 マイコンオプション round

＜追加内容＞

備考:

本オプションでは、実行時の浮動小数点演算における丸め方式を変更することはできません。

■(p.39) 2.5 マイコンオプション denormalize

＜追加内容＞

備考:

本オプションでは、実行時の浮動小数点演算における非正規化数の扱いを変更することはできません。

■(p.54) 表 3.1 ライブラリジェネレータオプション一覧 No.5 lang

＜修正内容＞

[誤] リエントラントライブラリを生成

[正] 使用可能な C 言語標準ライブラリ関数構成の選択

■(p.54) 表 3.1 ライブラリジェネレータオプション一覧

＜追加内容＞

No.6

`-cpu=rx600` 内容:RX600 シリーズ向けのライブラリを生成

■(p.109) 5.2.7 その他オプション / DELete オプション

＜追加内容＞

備考

form=library のときに、モジュールを削除できます。

form={absolute|relocate|hexadecimal|stype|binary}のときに、外部シンボルを削除できます。

■(p.142) 表 8.1 メモリ領域の種類とその性質の概要 No.5

＜修正内容＞

[誤] swith 文のジャンプテーブルを格納

[正] switch 文のジャンプテーブルを格納

■(p.190) 9.1.2 データの内部表現 / 表 9.15 / No.22 bool

＜修正内容＞

[誤]

サイズ = 4

アライメント数 = 4

符号の有無 = 有

[正]

サイズ = 1

アライメント数 = 1

符号の有無 = －（設定されていません）

■(p.190) 9.1.2 データの内部表現 / 表 9.15 / No.22 bool

＜追加内容＞

C99 コンパイルにおいては、「_Bool」型を使用することができます。

_Bool 型は bool 型と同じ型としてコンパイルされます。

■(p.190) 9.1.2 データの内部表現 / 表 9.15 / No.22 bool / 注釈*5

＜修正内容＞

[誤]

*5 C++および C99 コンパイルのみで有効です。

[正]

*5 C++コンパイル および stdbool.h インクルードのある C99 コンパイルのみで有効です。

■(p.197) 9.1.2 データの内部表現 / 表 9.17 / No.1 / 注釈*1(bool)

＜修正内容＞

[誤]

*1 C++および C99 プログラムのみ bool を指定できます。

[正]

*1 C++コンパイル および stdbool.h インクルードのある C99 プログラムのみ bool を指定できます。"

■(p.205) 9.1.3 浮動小数点型の仕様 / (2)float 型 /【注】

＜修正内容＞

[誤]

【注】　仮数部の最上位ビットが 0 の非数を qNaN、仮数部の最上位ビットが 1 の非数を sNaN と呼びます。

[正]

【注】　仮数部の最上位ビットが 1 の非数を qNaN、仮数部の最上位ビットが 0 の非数を sNaN と呼びます。

■(p.206) 9.1.3 浮動小数点型の仕様 / (3)double 型と long double 型 /【注】

＜修正内容＞

[誤]

【注】　仮数部の最上位ビットが 0 の非数を qNaN、仮数部の最上位ビットが 1 の非数を sNaN と呼びます。

[正]

【注】　仮数部の最上位ビットが 1 の非数を qNaN、仮数部の最上位ビットが 0 の非数を sNaN と呼びます。

■(p.216) 9.2.1 #pragma interrupt / 備考

＜追加内容＞

割り込み仕様として vect を使った場合、指定の無い空きベクタのアドレスは 0 になります。

このアドレスは、最適化リンケージエディタで任意のアドレスやシンボルに変更することができます。詳しくは、5.2.2 出力オプションの VECT および VECTN オプションの項目を参照してください。

■(p.238 / (p.228)) 9.2.2 組み込み関数 / (表 9.24 組み込み関数の一覧 No.14)

＜修正内容＞

[誤] void int_exception(unsigned long num)

[正] void int_exception(signed long num)

■(p.240 / (p.229)) 9.2.2 組み込み関数 / (表 9.24 組み込み関数の一覧 No.17)

＜修正内容＞

[誤] void set_ipl(unsigned long level)

[正] void set_ipl(signed long level)

■(p.553) 表 10.1 名前の種類 / ラベル名,シンボル名

＜追加内容＞

(シンボル名にはラベル名を含みます。)

■(p.554) 10.1.4 ラベルの記述方法

＜追加内容＞

備考:

定義されているセクション名と同じ名前のシンボルを定義することはできません。セクションとシンボルを同じ名前で定義した場合、先に定義したものが有効となり、後で定義したものは A2118 エラーとなります。

■(p.595) 10.3.3 リンク制御命令 / .SECTION / 備考

＜追加内容＞

定義されているシンボルと同じ名前のセクション名を定義することはできません。セクションとシンボルを同じ名前で定義した場合、先に定義したものが有効となり、後で定義したものは A2118 エラーとなります。

"$iop"という名前のセクション名を定義することはできません。"$iop"セクションを定義した場合 A2049 エラーとなります。

```
.SECTION  $iop,code    ; A2049 エラー
```

■(p.601) 10.3.6 拡張機能制御命令 / 表 10.35 拡張機能制御命令

＜削除内容＞

【注】コンパイラが出力するアセンブリ言語ファイルには、.FILE が記述されます。.FILE はコンパイラ出力アセンブリ言語ファイルにおいてのみ有効です。ユーザ作成アセンブリ言語ファイルでは使用しないでください。

■(p.716) 12.2 メッセージ一覧 / A2040 (E) Include nesting over

＜修正内容＞

[誤]インクルードのネストレベルが 9 以下になるように記述し直してください。

[正]インクルードのネストレベルが 30 以下になるように記述し直してください。"

■(p.725) 12.2 メッセージ一覧 / A3202 (F) Can't find work dir

＜修正内容＞

[変更前]

ワークディレクトリが見つかりません。

[変更後]

ワークディレクトリが見つかりません。
環境変数 TMP_RX が正しく設定されているか確認してください。

■(p.733) 13.2 メッセージ一覧 / L1200 (W) Backed up file "ファイル 1" into "ファイル 2"
＜修正内容＞
[変更前]
"ファイル 1"を"ファイル 2"にバックアップしました。
[変更後]
入力ファイル"ファイル 1"は書き換えられました。書き換える前の"ファイル 1"の内容は"ファイル 2"にバックアップされています。

■(p.751) 15.1 コーディング上の注意事項 / (4)オーバフロー演算、ゼロ除算
＜削除内容＞
fa=3.5e+40f; /* (W) 浮動小数点演算のオーバフローを検出します */

＜補足＞ 現在のマニュアルの記述に対しては、C5030(E)のエラーとなります。
C5030 (E) Floating constant is out of range

■(p.753) 15.1 コーディング上の注意事項 / (8)C89 と C99 の動作の差異
＜修正内容＞
[変更前]
上記を-lang=c99 を指定してコンパイルすると、以下の解釈となります。

```
enum {a,b};
int g(void)
{
    if(sizeof(enum{b,a}))
    {
        return a;
    }
    return b;
}
```

[変更後]
上記を-lang=c99 を指定してコンパイルすると、以下の解釈となります。

```
enum {a,b};
int g(void)
{
    {
        if(sizeof(enum{b,a}))
            return a;
    }
    return b;
}
```

■(p.753) 15.1 コーディング上の注意事項

＜追加内容＞

(9) オーバーフローを伴う演算および型変換に関する注意事項

数値演算や型変換を行う場合、その型で取り扱える値の範囲外(オーバーフロー)とならないようにご注意ください。オーバーフローが発生すると、得られる結果がコンパイルオプションなどの条件によって変化する場合があります。

標準の C 言語では、オーバーフローを伴う演算処理の結果は未定義となっており、コンパイル条件などにより得られる結果が異なる場合があります。

演算処理を行うプログラムでは、オーバーフローが発生させないようにご注意ください。

実際に結果が異なる例を次に示します。

例) float 型→unsigned short 型への変換

```
float f = 2147483648.0f;
unsigned short ui2;
void ex1func(void)
{
    ui2 = f;    /* float → unsigned short への変換 */
}
```

ex1func 関数を実行して得られる ui2 の値は、FPU あり(-fpu)と FPU なし(-nofpu)とで次のように異なります。

FPU あり: ui2 = 65535

FPU なし: ui2 = 0

これは、float 型から unsigned short 型への変換の方法が FPU ありと FPU なしで異なるためです。

以上