

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 変更点	2
1.1 MISRA-C:2012 ルールによるソースチェック機能【professional】	2
1.2 動的メモリ管理関数のセキュリティ強化【professional】	2
1.3 組み込み関数の追加	3
1.4 その他改善点	3
第 2 章 注意事項	9
2.1 W052304 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]	9
2.2 MVTC,POPC 命令を使用する場合の注意事項 [アセンブラ]	9
2.3 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]	9
第 3 章 制限事項	10
3.1 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	10
3.2 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)	12
3.3 以下のオプションを廃止しました【C/C++コンパイラ】	12
3.4 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】	12
3.5 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】	12
3.6 -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】	13
3.7 _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】	13
3.8 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】	13
第 4 章 添付標準ライブラリについて	14
4.1 添付ライブラリー覧	14
4.2 ライブラリ指定の方法	15

第1章 変更点

本章では、CC-RX V2.04.01からV2.05.00への主な変更点について説明します。

以降、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。

1.1 MISRA-C:2012ルールによるソースチェック機能【professional】

MISRA-C:2012ルールによりソースチェックを行う-misra2012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。

2.6 2.7 9.2 9.3 12.1 12.3 12.4 14.4 15.1 15.2 15.3 15.4 15.5 15.6 15.7
16.1 16.2 16.3 16.4 16.5 16.6 16.7 17.1 17.7 18.4 18.5 19.2 20.1 20.2
20.3 20.4 20.5 20.6 20.7 20.8 20.9 20.10 20.11 20.12 20.13 20.14

1.2 動的メモリ管理関数のセキュリティ強化【professional】

ヒープ領域の解放時に、不正操作を検出する機能を追加しました。本機能を有効にした標準ライブラリを作成するには、-secure_mallocオプションを指定してライブラリジェネレータを実行してください。本機能を有効にした標準ライブラリ関数は次のように動作します。

- (1) calloc, malloc, realloc関数内で、ヒープ領域内にユーザ用に割り当てる領域の他に、不正操作を検出するための領域として、その前後4バイトを余分に割り当てる。
- (2) free, realloc関数内で、引数が(1)で割り当てたポインタであるか、すでに開放済みのポインタではないか、また余分に確保した領域の値が書き換わっていないかを判定する。
- (3) いずれかの判定に該当した場合は不正な操作があったとして__heap_chk_fail関数を呼び出す。

__heap_chk_fail関数はユーザーが定義する必要があります。ヒープ領域の不正操作検出時に実行する処理を記述してください。例えば、以下のようなプログラムにおいて6行目で4文字分のバッファに対してstrから"ABCDEF"をコピーした場合、"EF"とヌル文字('¥0')分がオーバーフローしてヒープ領域を破壊してしまいます。この場合、8行目で当該ヒープ領域を解放する際に__heap_chk_fail関数が実行されることになります。

```
1: #include <string.h>
2: #include <stdlib.h>
3: void func(char *str) {
4:     char *buf;
5:     buf = (char*) malloc(4);
6:     strcpy(buf, str);    //str から"ABCDEF"をコピー
7:     ...
8:     free(buf);
9: }
```

本機能を使用することで、容易に安全性の向上（メモリ二重解放の問題やバッファ・オーバーフローの防止等）が可能となります。

1.3 組み込み関数の追加

1.3.1 新規組み込み関数の追加

ビット操作命令を生成する組み込み関数として以下を追加しました。

- __bclr()
BCLR命令を生成します。
- __bset()
BSET命令を生成します。
- __bnot()
BNOT命令を生成します。

1.3.2 既存の組み込み関数に対する別名の追加

V2.04.01 以前から存在する組み込み関数の別名を追加しました。

追加した関数は、対応する既存の組み込み関数の名前の先頭に"__" (アンダースコア2つ)を付加した名前を持ちます(※)。

例えば、max()関数に対して__max()関数が追加となります。

追加した組み込み関数の呼び出しインターフェースや展開結果は、対応する既存の組み込み関数と一致します。

例えば、max(a, b)と__max(a, b)の展開結果は一致します。

追加した組み込み関数は、ヘッダファイル(machine.h)をインクルードしなくても利用可能です。

(※) 先頭に"__"を付加した識別子はコンパイラによって予約されています。ユーザプログラムでは使用しないでください。

1.4 注意事項の改修

最適化の適用範囲に関する注意事項(RXC#038)

1.5 その他改善点

主に以下のような改善を行いました。

(a) 一時ファイルの読み込み・書き込みエラーの改善

一時ファイルに対する読み書きが失敗し、コンパイルエラーになる場合がある問題を改善しました。

(b) pow(), powf()の精度改善

pow(), powf()の演算で生じる誤差を改善しました。

(c) 内部エラーの改善

コンパイル時に内部エラーが発生する問題を改善しました。

1.6 最適化の強化

V2.05.00では、以降(a)～(e)の最適化強化を実施しました。

- (a) 異なるローカルスコープ内auto配列の領域を統合する最適化（スタックサイズの削減）
生存期間が重ならないブロック（{}）に属するauto配列のスタック領域を統合しました。

<ソースコード例>

```
int *g;
void func01(void){
    {
        int array01[10];
        g = array01;
        func();
    }
    {
        int array02[10]; // array01[10]と array02[10]
                        // は生存期間が重ならない
        g = array02;
    }
}
```

<V2.04.01 の生成コード>

```
_func01:
    .STACK _func01=88      ; 確保するスタック
                          ; サイズ=88byte

    PUSH.L R6
    MOV.L #_g, R6
    ADD #0FFFFFFB0H, R0
    MOV.L R0, R14
    MOV.L R14, [R6]
    BSR _foo
    ADD #28H, R0, R14
    MOV.L R14, [R6]
    RTSD #54H, R6-R6
```

<V2.05.00 の生成コード>

```
_func01:
    .STACK _func01=52      ; 確保するスタック
                          ; サイズ=52byte

    PUSHM R6-R7
    ADD #0FFFFFFD8H, R0
    MOV.L #_g, R6
    MOV.L R0, R7
    MOV.L R7, [R6]
    BSR _foo
    MOV.L R7, [R6]
    RTSD #30H, R6-R7
```

(b) 定数伝播最適化の改善

ループ内で定数だとわかる計算を省略しました。

<ソースコード例>

```
j = 1;
k = 2;
l = 3;

for (ix=0; ix<xtra; ix++) {
    for (i=0; i<n4; i++) {
        j = j*(k-j)*(l-k);    // j は常に 1
        k = l*k-(l-j)*k;      // k は常に 2
        l = (l-k)*(k+j);      // l は常に 3
        e1[l-2] = j+k+l;      // l-2 は常に 1, j+k+l は常に 6
        e1[k-2] = j*k*l;      // k-2 は常に 0, j*k*l は常に 6
    }
}

x = e1[0]+e1[1];
```

<V2.04.01 の生成コード>

```
:
L13:  ; bb47
      CMP R2, R6
      BGE L15
L14:  ; bb3
      SUB R5, R15, R7
           ; j*(k-j)*(l-k),
           ; l*k-(l-j)*k, (l-k)*(k+j), ...を毎回計算
      SUB R14, R5, R8
      MUL R7, R14
      MUL R8, R14
      MUL R5, R15, R8
      SUB R14, R15, R7
      MUL R5, R7
      ADD #01H, R6
      SUB R7, R8
      SUB R8, R15
      ADD R14, R8, R5
      MUL R5, R15
      ADD R15, R5
      SHLL #02H, R15, R7
:
```

<V2.05.00 の生成コード>

```
:
L13:  ; bb46
      CMP R2, R15
      BGE L15
L14:  ; bb2
      ADD #01H, R15
      MOV.L #00000006H, 04H[R3]
           ; e1[1]に常に 6 を代入
      MOV.L #00000006H, [R3]
           ; e1[0]に常に 6 を代入
      BRA L13
:
```

(c) 帰納変数最適化の改善

冗長なループ帰納変数の更新コードを生成しないようにしました。

<ソースコード例>

```
void callee(unsigned i);
void caller(void){
    unsigned i;
    for(i=128; i != 0; --i){
        callee(i);
    }
}
```

<V2.04.01 の生成コード>

```
      :
      MOV.L #00000080H, R6
      MOV.L R6, R7          ; 冗長なループ帰納
                           ; 変数の初期化
L11:   ; bb
      MOV.L R7, R1
      BSR _callee
      SUB #01H, R7          ; 冗長なループ帰納
                           ; 変数の更新
      SUB #01H, R6
      BNE L11
      :
```

<V2.05.00 の生成コード>

```
      :
      MOV.L #00000080H, R6
L11:   ; bb
      MOV.L R6, R1
      BSR _callee
      SUB #01H, R6
      BNE L11
      :
```

(d) min/max/abs命令利用最適化の改善

min/max/abs命令をより使うようにしました。

<ソースコード例>

```
int min_test(int a) {  
    if (a >= 17) {  
        a = 17;  
    }  
    return a;  
}
```

<V2.04.01 での生成コード>

```
_min_test:  
    .STACK _min_test=4  
    CMP #10H, R1  
    MOV.L #00000011H, R14  
    BGT L12  
L11: ; entry  
    MOV.L R1, R14  
L12: ; entry  
    MOV.L R14, R1  
    RTS
```

<2.05.00 での生成コード>

```
_min_test:  
    .STACK _min_test=4  
    MIN #11H, R1      ; min 命令を利用  
    RTS
```

(e) 不要コード削除最適化の改善

使われることのないコードの削除処理を強化しました。

<ソースコード例>

```
unsigned long test(unsigned long long variable, int var){  
    if (var){  
        variable &= 0x012345678abcdefULL;  
    }  
    return (variable >> 32);  
}
```

<V2.04.01 での生成コード>

```
_test:  
    .STACK _test=4  
    CMP #00H, R3  
    BEQ L12  
L11: ; if_then_bb  
    AND #78ABCDEFH, R1 ; R1 はこの後使用されない  
    AND #00123456H, R2  
L12: ; if_break_bb  
    MOV.L R2, R1  
    RTS
```

<V2.05.00 での生成コード>

```
_test:  
    .STACK _test=4  
    CMP #00H, R3  
    BEQ L12  
L11: ; if_then_bb  
    AND #00123456H, R2  
L12: ; if_break_bb  
    MOV.L R2, R1  
    RTS
```

第2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。

2.1 W052304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 W052304 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。

2.2 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。

アセンブラでの割り込み関数を記述する場合には、r1 レジスタの退避／復帰を行ってください。

2.3 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。

第3章 制限事項

本章では、CC-RX V2.05.00 の制限事項について説明します。

3.1 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 ; 再帰呼び出しになってしまう
```

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);
}
```

3.2 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。

3.3 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。

3.4 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】

(1)-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。

(2)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。

3.5 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 0fffffffh
.byte 3; ; 0xffffffff
.end
```

3.6 -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。

3.7 _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。

3.8 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。

第4章 添付標準ライブラリについて

本章では、RXファミリ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。

4.1 添付ライブラリー一覧

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	その他 *3
				-rtti -exception -noexception	
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1 ライブラリー一覧

*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。

4.2 ライブラリ指定の方法

コンパイラパッケージに含むライブラリファイルは、セクション 4.2.1 および 4.2.2 で示した方法でリンクしてください。

4.2.1 ライブラリ指定の方法

e² studio を C:\¥Renesas¥e2_studio へインストールした時、ライブラリファイルは、以下のフォルダに存在します。

C:\¥Program Files¥Renesas¥RX¥V2_4_0¥lib

(「V2.04.00」は、コンパイラパッケージのバージョンおよび改訂番号を示します。)

4.2.2 最適化リンケージエディタの中でライブラリファイルのフォルダを指定

コンパイラパッケージに含まれたライブラリファイル(セクション4.2.1で示した位置に格納されたファイル)を希望のディレクトリにコピーしてください。

次に、コピーしたライブラリファイルのうちの1つを指定してリンケージ処理を始めてください。

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。

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、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 〒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：<http://japan.renesas.com/contact/>