CS+ RX コンパイラ CC-RX V3.01.00 リリースノート

R20UT4453JJ0100
Rev.1.00
2018.12.20

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-RX 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-RX コンパイラ ユーザーズマニュアル	R20UT3248JJ0107
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	R20UT3478JJ0106

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-RX V3.01.00」を選択してください。

第4章　変更点

本章では、CC-RX V3.00.00 から V3.01.00 への主な変更点について説明します。

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　倍精度浮動小数点処理命令への対応

RXv3 命令セットアーキテクチャが備える倍精度浮動小数点処理命令を使用したコード生成を行えるようになりました。これにより double、 long double 型を用いたプログラムの性能・コードサイズが改善されます。

利用時にはコンパイル・オプション、アセンブラオプションのそれぞれに新規オプション-dpfpu の指定が必要です。

4.2　レジスター括退避機能への対応

RXv3 命令セットアーキテクチャが備えるレジスター括退避機能を利用できるようになりました。

#pragma interrupt で bank=<バンク番号>を記述することで、割り込みハンドラ先頭でのレジスタ退避と、割り込みハンドラ末尾でのレジスタ復帰を一括して高速に行うことが可能です。

利用時にはアセンブラオプションに-bank の指定が必要です。

4.3　複数ファイルにまたがる MISRA-C:2012 チェックオプションの追加【professional】

複数ファイルにまたがる MISRA-C:2012 ルールのソース・チェックを行うオプション-misra_intermodule を追加しました。

従来は個々のファイル内でのチェックのみでしたが、本オプションを指定することにより、複数ファイルにまたがってチェックできるようになりました。

4.4 MISRA-C:2012 ルールによる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-misra2012 オプションの引数に、下記のルール番号を指定できるようにしました。

【必要ルール】 **8.5** **8.6**

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）	V3.00.00	**V3.01.00**
必須ルール（16）	7	**7**
必要ルール（108）	86	**88**
推奨ルール（32）	26	**26**
合計ルール（156）	119	**121**

4.5 コンパイル・オプション-truncated_address_initializer の追加

コンパイラのオプションに-truncated_address_initializer を追加しました。

本オプションを指定することで、C 言語における 1、2 バイト型の外部変数もしくは静的変数をアドレスで初期化する記述に対し、E0520069 エラーを出力しなくなります。

代わりに警告メッセージ W0520069 を出力します。

4.6 ライブラリファイル入力時のセクション名変更機能の追加

最適化リンケージエディタ(rlink)オプションに-lib_rename を追加しました。本オプションを使用することにより、リンク時に入力するライブラリファイル内の、セクション名やシンボル名を変更してリンクできます。

4.7 最適化の強化

CC-RX V3.01.00 では主に以下の最適化強化を実施しました。

(a) MAX/MIN/BFMOV 命令の活用

MAX/MIN/BFMOV 命令をより活用するようになりました。

<ソースコード例>

```
int func(int x) {
  if (x >= 100) {
    x =  100;
  }
  if (x <= -100) {
    x = -100;
  }
  if (x <= 0) {
    return -x;
  }
  return x;
}
```

<V3.00.00>

```
_func:
    MIN #64H, R1
    CMP #0FFFFFF9DH, R1
    BGE L12
L11:
    MOV.L #0FFFFFF9CH, R1
L12:
    ABS R1
    RTS
```

<V3.01.00>

```
_func:
    MIN #64H, R1
    MAX #0FFFFFF9CH, R1
L11:
    ABS R1
    RTS
```

(b) 命令スケジューリングの強化

パイプラインがより効率的に動作するようにメモリアクセス命令を並べ替える機能を強化しました。

<ソースコード例>

```
int a[1024], b[1024], c[1024];
void func(int n) {
  int i;
  for (i = 0; i < 4; ++i) {
    a[i] = b[i] + c[i];
  }
}
```

<V3.00.00>

```
_func:
    MOV.L #00000002H, R5
    MOV.L #_c, R14
    MOV.L #_b, R15
    MOV.L #_a, R1
L11:
    MOV.L [R14+], R4
    MOV.L [R15+], R3
    ADD R3, R4
    MOV.L R4, [R1]
    MOV.L [R14+], R2
    MOV.L [R15+], R4
    ADD R4, R2
    MOV.L R2, 04H[R1]
    ADD #08H, R1
    SUB #01H, R5
    BNE L11
L12:
    RTS
```

<V3.01.00>

```
_func:
    MOV.L #00000002H, R5
    MOV.L #_c, R14
    MOV.L #_b, R15
    MOV.L #_a, R1
L11:
    MOV.L [R14+], R4
    MOV.L [R15+], R3
    MOV.L [R14+], R2
    ADD R3, R4
    MOV.L [R15+], R3
    MOV.L R4, [R1]
    ADD R3, R2
    MOV.L R2, 04H[R1]
    ADD #08H, R1
    SUB #01H, R5
    BNE L11
L12:
    RTS
```

(c) 冗長な比較命令の削除処理の強化

冗長な比較命令の削除処理を強化しました。

<ソースコード例>

```
void func2(void);

void func1(unsigned int a, unsigned int b, unsigned int c, unsigned int d){
  if (((a != 0) && (a > b)) && (c == d)){
    func2();
  }
}
```

<V3.00.00>

```
_func1:
    CMP R2, R1
    BLEU L14
L11:
    CMP #00H, R1
    BEQ L14
L12:
    CMP R4, R3
    BNE L14
L13:
    BRA func2_
L14:
    RTS
```

<V3.01.00>

```
_func1:
    CMP R4, R3
    BNE L13
L11:
    CMP R2, R1
    BLEU L13
L12:
    BRA func2_
L13:
    RTS
```

4.8 注意事項の改修

以下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。

- -misra2012 オプション指定時の注意事項(CCRX#050)

4.9 その他の改善点

主に以下の改善を行いました。

- 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。

第5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。

5.1 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。

5.2 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。

5.3 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6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。

6.1 統合環境 CS+のヘルプ コンパイラ等のオプションの参照について

CS+添付ヘルプから、CC-RX に含まれる C/C++コンパイラ(ccrx)、アセンブラ(asrx)、最適化リンケージエディタ(rlink)およびライブラリジェネレータ(lbgrx)のコマンドラインオプションを参照いただく場合は、RX(CC-RX 環境)の、「コンパイラ編」以下の内容でご確認ください。

同様の内容は「ビルド編」にも記載されていますが、こちらは V2.02.00 のものですのでご注意ください。

6.2 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 = frxep(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);
}
```

6.3 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。

6.4 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。

6.5 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。

6.6 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
```

6.7 -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。

6.8 _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。

6.9 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。

6.10 -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。

6.11 -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。

6.12 -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7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。

7.1 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 7.1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。

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

表 7.1 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。

7.2 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。

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、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。
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、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。
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、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。
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。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。
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.4.0-1　2017.11)

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/