RX ファミリ用 C/C++コンパイラパッケージ

CC-RX V3.08.00

リリースノート

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

1.　マニュアルについて

本製品に関連した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	R20UT3248JJ0115
(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	R20UT3478JJ0113

2.　動作環境

推奨する動作環境は以下の通りです。

Windows 10 - 64bit

Windows 11 - 64bit

Ubuntu 22.04 LTS - 64bit

Ubuntu 24.04 LTS - 64bit

3.　変更点

本章では、CC-RX V3.07.00 から V3.08.00 のバージョンアップによる主な変更点について説明します。

3.1　アトリビュートの追加

以下の GCC 互換のアトリビュートを使用できるようになりました。

- aligned
- always_inline
- naked
- noinline
- section
- weak

ただし、C++ソースプログラムファイルの中での記述はサポートしていません。

本機能は CC-RX コンパイラユーザーズマニュアル(R20UT3248JJ0115)には掲載されていません。詳細仕様については下記をご参照ください。

3.1.1　アトリビュートの指定形式

アトリビュートは以下の(1)～(3)の指定形式で使用することができます。

```
(1)　<アトリビュート>[…] <{変数|関数|メンバ}宣言>;
(2)　<{変数|関数|メンバ}宣言> <アトリビュート>;
(3)　<宣言指定子> (<アトリビュート> <{変数|関数|メンバ}宣言>)[, …];

<アトリビュート>:
  __attribute__((<attribute list>))
<attribute list>:
  <アトリビュート名>[(<アトリビュート引数>[,…])][,…]
```

(1)　<アトリビュート>以降に宣言した全ての{変数|関数|メンバ}に有効です。

(2)　<アトリビュート>の直前に宣言した{変数|関数|メンバ}のみに有効です。

(3)　()内で宣言した{変数|関数|メンバ}のみに有効です。

次の例において__attribute__((section))は var1, var2, var4, var5 に適用されます。var3, var6 には適用されません。

```
__attribute__((section("MyBSS"))) int var1, var2;        // 指定形式(1)
int var3, var4 __attribute__((section("MyBSS")));        // 指定形式(2)
int (__attribute__((section("MyBSS"))) var5), var6;      // 指定形式(3)
```

また、矛盾しない限りアトリビュートを複数同時に指定することができます。

```
__attribute__((noinline)) __attribute__((weak)) void func1();
__attribute__((noinline,weak)) void func2();
__attribute__((noinline)) void func3();
__attribute__((weak)) void func3() { }
```

3.1.2　aligned

[指定形式]

```
__attribute__((aligned[(<整数>)]))
```

[詳細説明]

- 指定した変数またはメンバの先頭アドレスを<整数>の倍数のアドレスに整列します。
- 指定可能な対象および<整数>の値は次の通りです。

指定対象	指定可能な<整数>
大域変数	4096 以下の 2 のべき乗
static 修飾された変数	4096 以下の 2 のべき乗
static 修飾された構造体型または共用体型メンバ	4096 以下の 2 のべき乗
大域変数ではなく、static 修飾されていない構造体型または共用体型メンバ	1,2,4 のいずれかの値

- 同一の対象に複数同時に指定した場合、最大の<整数>の指定を有効にします。
- 指定した変数、メンバのデフォルトのアライメント数よりも小さな値を<整数>に指定することはできません。コンパイル・エラーになります。
- -stuff オプションが有効な場合、大域変数、static 修飾された変数とメンバは、サフィックス”_X”がついたセクションに配置されます。以下の 2 つの事項について注意する必要があります。
  - リンカの-start オプション、-rom オプションで、サフィックス”_X”がついたセクションの指定が必要になります。その他、セクション名を指定しているオプションがある場合にはご確認ください。
  - セクションによってはスタートアッププログラムでの初期化が必要になります。
- <整数>の指定を省略した場合、4 を<整数>に指定したものとして扱います。

3.1.3　always_inline

[指定形式]

```
__attribute__((always_inline))
```

[詳細説明]

- 指定した関数をインライン展開します。
- 以下のいずれかに該当する場合はインライン展開をしません。
  - 指定した関数が可変引数を持つ関数である。
  - 指定した関数がアドレスを介して呼び出されている。
- __attribute__((always_inline))は、インライン展開されることを保証するものではありません。
- #pragma noinline と同時に指定した場合、__attribute__((always_inline))を優先します。
- __attribute__((noinline))と同時に指定した場合、__attribute__((noinline))を優先します。
- 次の#pragma 指令と同時に指定することはできません。コンパイル・エラーになります。
  - interrupt
  - inline_asm
  - entry
  - stack_protector
- __attribute__((weak))と同時に指定することはできません。コンパイル・エラーになります。

3.1.4　naked

[指定形式]

```
__attribute__((naked))
```

[詳細説明]

- 指定した関数に対して、関数の入口、出口コードの出力を抑止します。
- 関数本体には asm 文を記述してください。asm 文以外を記述した時の動作は保証しません。
- 次の#pragma 指令と同時に指定することはできません。コンパイル・エラーになります。
  - inline_asm
  - stack_protector
- __attribute__((always_inline))と同時に指定することはできません。コンパイル・エラーになります。

3.1.5　noinline

[指定形式]

```
__attribute__((noinline))
```

[詳細説明]

- 指定した関数に対して、オプションや最適化レベルによらず、常にインライン展開を抑止します。
- #pragma inline と同時に指定した場合、__attribute__((noinline))を優先します。
- __attribute__((always_inline))と同時に指定した場合、__attribute__((noinline))を優先します。
- #pragma inline_asm と同時に指定することはできません。コンパイル・エラーになります。

3.1.6　section

[指定形式]

```
__attribute__((section("<セクション名>"))
```

[詳細説明]

- 指定した変数や関数を<セクション名>で指定した名前のセクションに配置します。
- 同じ変数や関数に複数同時に指定した場合、最後に指定した名前のセクションに配置します。
- <セクション名>には、アセンブリ言語ファイルの中でセクション名として使用可能な文字および文字数を指定できます。これに反した場合にはアセンブル・エラーになります。
- #pragma section と同時に指定した場合、__attribute__((section))を優先します。

3.1.7　weak

[指定形式]

```
__attribute__((weak))
```

[詳細説明]

- 変数定義や関数定義に指定することで、weak 定義を生成します。また、変数宣言や関数宣言に指定することで weak 参照とします。weak 定義および、weak 参照は、通常の定義・参照と次の点で異なります。
  - ➢ weak 定義：

    リンク時に、同名の通常定義が他のモジュールに存在することができます。このとき、リンカはエラーを出力せず、通常定義をリンクして weak 定義を無視します。
  - ➢ weak 参照：

    リンク時に、参照先の変数や関数の定義が存在しなくてもリンカはエラーを出力せず、参照先が NULL であるとみなしてリンクを行います。
- #pragma inline と同時に指定することはできません。コンパイル・エラーになります。
- __attribute__((always_inline))と同時に指定することはできません。コンパイル・エラーになります。

3.2　-linker_script オプションの追加

リンク時に、GNU リンカのスクリプト形式をベースとしたリンカスクリプトを使用できるようになりました。リンカスクリプトは、最適化リンケージエディタ(rlink)・オプション -linker_script で指定します。

本機能は CC-RX コンパイラユーザーズマニュアル(R20UT3248JJ0115)には掲載されていません。詳細仕様については下記をご参照ください。

[指定形式]

```
-linker_script=< ファイル >
```

[詳細説明]

- リンカスクリプト・ファイルを指定します。
- rlink がサポートするリンカスクリプトのキーワード・構文は以下のとおりです。 サポートしない記述を行った場合、予期しない動作を引き起こすことがあります。
  各項目の詳細は、GNU リンカのマニュアルをご参照ください。
  - MEMORY コマンド
    下記の形式の MEMORY コマンドをサポートします。

```
MEMORY
{
      name[(attr)]:ORIGIN=address,LENGTH=size
      ...
}
```

    name はリージョン（メモリ領域）の名前を、*address* は先頭アドレス、*size* はサイズを指定します。(*attr*)は省略可能です。*attr* には'r'(読み取り専用)、'w'(読み書き可能)、'x'(実行可能)が指定できます。

  - SECTIONS コマンド
    下記の形式の SECTIONS コマンドをサポートします。

```
SECTIONS
{
      sections-statement
      sections-statement
      ...
}
```

    sections-statement には以下を記述することができます。

    - ENTRY コマンド
    - シンボルの定義（シンボルへの代入）
      - *symbol* = *expression* ;の形式の単純なシンボル定義
      - キーワード HIDDEN, PROVIDE, PROVIDE_HIDDEN を用いたシンボル定義
    - データ記述（キーワードは BYTE、SHORT、LONG のみをサポート）
    - 以下の形式の出力セクション記述

```
section_name [address] [(type)] : [AT(load_address)] [ALIGN(section_align)]
{
    output-section-statement
    output-section-statement
    ...
} [>region] [AT>load_region] [=fill]
```

- *section_name* は出力セクションの名前を、*address* は実行時アドレス（仮想アドレス）、 *load_address* はロード先アドレス、*section_align* はアライメント数、 *region* は実行時のリージョン、*load_region* はロード先のリージョン、 *fill* は空き領域の充填値を指定します。
- []で囲まれた項目は省略可能です。
- *type* には、NOLOAD、READONLY、TYPE=SHT_PROGBITS, TYPE=SHT_NOBITS のいずれかを指定可能です。
- *output-section-statement* には以下を記述することができます。
  - *sections_statement* と同様の形式のシンボルの定義
  - 以下の形式の入力セクション記述

    ```
    input_file_name(input_section_name)
    ```

    ワイルドカードおよびキーワード SORT_BY_NAME、SORT (SORT_BY_NAME と同じ)、 EXCLUDE_FILE、KEEP を記述できます。

○ その他のコマンド

以下のコマンドをサポートします。

- ENTRY
- INCLUDE
- GROUP
- OUTPUT
- OUTPUT_FORMAT（elf32-rx-be または elf32-rx-le を指定可能）
- TARGET（OUTPUT_FORMAT と同じ）
- SEARCH_DIR
- STARTUP
- ASSERT
- PROVIDE
- HIDDEN
- PROVIDE_HIDDEN

○ 組み込み関数

以下の組み込み関数をサポートします。

- ABSOLUTE
- ADDR
- ALIGN
- DEFINED
- LENGTH
- LOADADDR
- MAX
- MIN
- ORIGIN
- SIZEOF

[備考]

- INCLUDE で再帰的にリンカスクリプト・ファイルを指定した場合、メッセージを出力して終了します。
- form={object | relocate | library}または strip 指定時、本オプションは無効です。
- start オプションと本オプションで指定したリンカスクリプトによるセクションの配置指定は、オプションで指定された順に解釈されます。
- 本オプションと library_directory オプションは指定順に解釈します。このため、リンカスクリプト・ファイルの配置先フォルダを library_directory で指定する場合は、本オプションより前に指定してください。
- BYTE、SHORT、LONG は、常にリトルエンディアンでセクションにデータを挿入します。

[注] GNU リンカとは、以下の点が異なります。

- rlink は、GROUP 指定の有無によらず、全てのライブラリにおいて GROUP が指定されたように動作します。
- OUTPUT コマンドを複数回指定すると、指定した分だけファイルが出力されます。
- OUTPUT_FORMAT および TARGET は受容しますが、効果はありません。
- 出力セクションの指定がない入力セクションは、最後に配置されます。
- ADDR の結果は、セクション相対ではなく絶対値になります。

[メッセージ]

以下の表はリンカスクリプト使用時に出力するメッセージです。

番号	メッセージ	説明
M0560800	Detailed error information	リンカスクリプトの構文解析における詳細エラー情報です。
W0561703	Duplicate definition of “*名前*”	リンカスクリプトで”*名前*”の定義が重複しています。最後に指定したものが有効になります。

番号	メッセージ	説明
W0561705	Invalid alignment value	リンカスクリプトで指定されたアライメントが不正です。
E0562023	Duplicate file specified "*ファイル*"	リンカスクリプトで"*ファイル*"の記述が重複しています。
E0562115	Linker script specified section cannot be specified in "*セクション*" option : "*オプション*"	リンカスクリプトで記述したセクション"*セクション*"は、オプション"*オプション*"で指定できません。
E0562312	Undefined section symbol "*セクション*" referenced in "*ファイル*"	ファイル"*ファイル*"で"セクション・シンボル"*シンボル*"が参照されていますが、対応するセクションが存在しません。
E0562327	Section "*セクション1*" overlaps section "*セクション2*" in load memory	リンカスクリプトで指定したセクション間で、ロードメモリ配置（LMA）が重複しています。
E0562700	ASSERT:"*文字列*"	リンカスクリプトに記述された ASSERT コマンドで指定した条件が満たされなかったため、"*文字列*"が表示されました。
E0562701	Syntax error	リンカスクリプトの記述に構文エラーがあります。
E0562702	Syntax error	リンカスクリプトの記述に構文エラーがあります。
E0562703	Nested Include file "*ファイル*"	リンカスクリプトの INCLUDE コマンドで、直接または間接的に同一ファイルを読み込んでいます。循環したインクルード関係を見直してください。
E0562705	Division or modulo by zero	リンカスクリプトに 0 での除算または剰余算があります。
E0562706	Undefined format "*フォーマット*"	リンカスクリプトの OUTPUT_FORMAT または TARGET の指定が正しくありません。以下のいずれかを指定してください。 - elf32-rx-be - elf32-rx-le
E0562707	Undefined symbol "*シンボル*"	リンカスクリプトで指定されたシンボル"*シンボル*"が定義されていません。
E0562708	Undefined section "セクション"	リンカスクリプトで指定されたセクション"*セクション*"が定義されていません。
E0562709	Undefined memory "*メモリ*"	リンカスクリプトで指定されたメモリ領域"*メモリ*"が定義されていません。
E0562711	Memory definition out of range "*メモリ*"	リンカスクリプトで指定されたメモリ領域"*メモリ*"への配置が、サイズを超過しています。
E0562712	Section "*セクション*" unfit in memory "*メモリ*" range	リンカスクリプトで指定されたセクション"*セクション*"がメモリ領域"*メモリ*"の範囲に適合しません。

番号	メッセージ	説明
E0562714	Cannot move location counter backwards	ロケーションカウンタを値が減る方向に移動することはできません。
E0562715	Section "*セクション*" conflicts with option "*オプション 1*" and option "オプション 2"	セクション"*セクション*"の指定が、オプション"*オプション 1*"およびオプション"*オプション 2*"と矛盾しています。
E0562717	Reserved section name in SECTIONS : "*セクション*"	リンカスクリプトの SECTIONS コマンドで指定されたセクション名"*セクション*"は予約されています。他の名前に変えてください。

3.3　-library_directory オプションの追加

ライブラリ・ファイルの配置先フォルダを指定する最適化リンケージエディタ(rlink) ・オプション -library_directory が追加されました。

本機能は CC-RX コンパイラユーザーズマニュアル(R20UT3248JJ0115)には掲載されていません。詳細仕様については下記をご参照ください。

[指定形式]

```
-library_director=<フォルダ>
```

[詳細説明]

- ライブラリ・ファイル、リロケータブル・ファイル、オブジェクト・ファイルとリンカスクリプト・ファイルの配置先フォルダを追加します。
- ファイルは、カレントフォルダ、本オプションで指定したフォルダ、環境変数 HLNK_DIR で指定したフォルダの順序で探索します。

[使用例]

¥libdir にある a.lib を入力します。

```
> rlink main.obj -library_directory=¥libdir -library=a.lib
```

3.4　プリプロセッサ指令__has_include の追加

ヘッダが存在するかどうかを調べる__has_include を追加しました。

指定したヘッダが存在する場合は 1 を返し、存在しない場合は 0 を返します。

3.5　__asm 文のサポート

__asm 文を用いて、GCC がサポートするインラインアセンブリ記述を利用できます。

基本構文のみが記述できます。拡張構文はサポートしていません。

修飾子 volatile および inline は GCC との互換性のために受理されます。特別な効果はありません。

関数内にのみ記述することができます。関数外に記述するとエラーになります。

3.6　<time.h>のサポート

標準ヘッダ<time.h>が提供する機能のうち、型およびマクロをサポートしました。

3.7　アセンブリ言語における名前の規則の変更

アセンブリ言語仕様における名前に、ピリオド(.)を使用できるように改善しました。

シンボル名やセクション名などにピリオド(.)を含めることができます。

3.8　アドレス制御疑似命令の改善

アセンブリ言語におけるアドレス制御疑似命令(.ALIGN, .ORG, .OFFSET)において、アドレス補正により発生する空き領域に書き込む値は NOP(03H)で固定していましたが、任意の値を指定できるようにパラメータを追加しました。

3.9　アライメント補正値の指定範囲拡張

アセンブリ言語におけるアドレス制御疑似命令(.ALIGN)および、リンク制御疑似命令(.SECTION)で指定可能なアライメント補正値の範囲を 2～65536 の範囲の 2 のべき乗に拡張しました。

3.10 依存関係ファイル出力オプション

ソースファイルとインクルードファイルの依存関係を、依存関係ファイルに出力するオプションを公開しました。

コンパイル・オプション：-output=dep、-MM、-MP、-MT

アセンブル・オプション：-MM、-MP、-MT、-MF

また、以下のオプションを追加し、依存関係ファイルを利用しやすくしました。

コンパイル・オプション：-MD、-MMD、-MF

アセンブル・オプション：-MMD

出力した依存関係ファイルは Makefile 内に記述して利用できます。オプションの詳細はユーザーズマニュアルを参照してください。

3.11 -whole_program オプションの機能改善

入力ファイルがプログラム全体であることを仮定した最適化を強化しました。

入力ファイル全てをマージした状態において以下のような外部変数に対する最適化が行われるようになります。

- const 修飾されていない外部変数が書き込みの対象にならない場合に、const 修飾されているものと同様に最適化します。
- static 修飾されていない外部変数を、static 修飾されているものと同様に最適化します。

ソースコード例 (test.c)

```
unsigned char globalIndex;
unsigned char neverReadArray[10];
void
autoVariableReplacementAndStoreEliminationExample(void){
   for(globalIndex = 0; globalIndex < 10; ++globalIndex){
      neverReadArray[globalIndex] = globalIndex;
   }
}

int neverOverwrittenVariable = 0;
int* writeOncePointer;
void constantPropagationExample0(void){
   writeOncePointer = &neverOverwrittenVariable;
}
int constantPropagationExample1(void){
   return (*writeOncePointer);
}
```

出力コード (ccrx -isa=rxv3 -whole_program -output=src test.c)

<V3.07.00>

```
_autoVariableReplacementAndStoreEliminationExample:
        MOV.L #00000000H, R14
        MOV.L #_neverReadArray, R15
        MOV.L R14, R5
L11:    ; bb6
        CMP #0AH, R5
        BEQ L13
L12:    ; bb
        MOVU.B R5, R4
```

<V3.08.00>

```
_autoVariableReplacementAndStoreEliminationExample:
        .STACK _autoVariableReplacementAndStoreEliminationExample=4
        MOV.L #0000000AH, R14
L11:    ; bb
        SUB #01H, R14
        BNE L11
L12:    ; return
        RTS
```

```
        ADD #01H, R5
        MOV.B R14, [R15,R4]
        ADD #01H, R14
        BRA L11
L13:    ; return
        MOV.L #_globalIndex, R14
        MOV.B #0AH, [R14]
        RTS
_constantPropagationExample0:
        .STACK
_constantPropagationExample0=4
        MOV.L #_writeOncePointer, R14
        MOV.L #_neverOverwrittenVariable, R15
        MOV.L R15, [R14]
        RTS
_constantPropagationExample1:
        .STACK
_constantPropagationExample1=4
        MOV.L #_writeOncePointer, R14
        MOV.L [R14], R14
        MOV.L [R14], R1
        RTS
```

```
_constantPropagationExample0:
        .STACK
_constantPropagationExample0=4
        RTS
_constantPropagationExample1:
        .STACK
_constantPropagationExample1=4
        MOV.L #00000000H, R1
        RTS
```

3.12 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

- 無名共用体または無名構造体と指示付き初期化子の使用に関する注意事項(No.69)
- -avoid_cross_boundary_prefetch オプションの使用に関する注意事項(No.70)

3.13 その他変更・改善

以下の変更・改善を行いました。

- リンカの不具合改善

  V3.07.00 で.rel ファイルをリンクすると C0564001 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。

- 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。

4.　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。

4.1　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 4.1 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
オプション</th><th colspan="3">マイコン・オプション *1</th></tr>
<tr><th rowspan="2">-endian</th><th>-cpu</th><th rowspan="2">その他 *2</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 4.1 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。

4.2　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
1.00	Jul.1.2026	-	新規発行

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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。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。