RH850 I/O ヘッダ・ファイル ガイド

目次

1. 概要

本書は RH850 の I/O ヘッダ・ファイルのガイドです。

RH850 の I/O ヘッダ・ファイルは、RH850 の I/O レジスタが定義された C 言語向けのヘッダ・ファイルです。I/O ヘッダ・ファイルは、CS+上で RH850 のプロジェクトを作成した際に、iodefine.h という名前で、サンプルとして生成されます。また、CS+のプロジェクト･ツリーから I/O ヘッダ・ファイルを生成させることも可能です。この I/O ヘッダ・ファイルを使用することで、RH850 の I/O レジスタにアクセスするコードを記述することができます。

本書では以降、I/O ヘッダ・ファイルと記述した場合、RH850 の I/O ヘッダ・ファイルを指します。

2. RH850のI/Oレジスタについて

本章では RH850 の I/O レジスタについて記述します。

2.1 I/Oレジスタとは

RH850 の I/O レジスタ（以降、I/O レジスタと記述します）は、周辺機能を使用する際にアクセスするレジスタです。I/O レジスタごとに、割り当てられたアドレスと、アクセスサイズが設定されています。アドレスは I/O レジスタごとに異なりますが、同じアドレスに複数の I/O レジスタが配置されることもあります。アクセスサイズには、4 バイト、2 バイト、1 バイト、1 ビット、N ビット(例：3 ビット目～7 ビット目の 5 ビット)などがあり、用途により異なります。

I/O レジスタの名前は、通常ではモジュール名とレジスタ名を組み合わせています。例として、 RSCAN0 モジュールの C0CFG レジスタの場合、I/O レジスタ名は RSCAN0 と C0CFG をドット（.）で繋げた RSCAN0.C0CFG となります。

2.2 I/Oレジスタの構成と命名規則

RH850 では、同じモジュール内で複数の I/O レジスタが同じアドレスに配置されることがあります。その中で、アクセスサイズが異なる構成の場合、通常ではアクセスサイズが小さい I/O レジスタの名前に文字を付加する命名規則が適用されています（一部例外もあります）。

例として、RSCAN0.C0CFG はアクセスサイズが 4 バイトの I/O レジスタですが、4 バイトの範囲内で別のアクセスサイズでアクセス可能な I/O レジスタが配置されています。

表 1. RSCAN0.C0CFG の構成

I/O レジスタ名	アドレス	アクセスサイズ
RSCAN0.C0CFG	0xFF000000	4 バイト
RSCAN0.C0CFGL	0xFF000000	2 バイト
RSCAN0.C0CFGH	0xFF000002	2 バイト
RSCAN0.C0CFGLL	0xFF000000	1 バイト
RSCAN0.C0CFGLH	0xFF000001	1 バイト
RSCAN0.C0CFGHL	0xFF000002	1 バイト
RSCAN0.C0CFGHH	0xFF000003	1 バイト

表 2. RSCAN0.C0CFG とアドレスの対応関係

	0xFF000003	0xFF000002	0xFF000001	0xFF000000
4 バイト	RSCAN0.C0CFG			
2 バイト	RSCAN0.C0CFGH		RSCAN0.C0CFGL	
1 バイト	RSCAN0.C0CFGHH	RSCAN0.C0CFGHL	RSCAN0.C0CFGLH	RSCAN0.C0CFGLL

このような最大アクセスサイズが 4 バイトの I/O レジスタの構成の場合、通常では最大のアクセスサイズを持つ I/O レジスタの名前を基準として、2 バイトの I/O レジスタの名前は、下位 2 バイトの I/O レジスタには L、上位 2 バイトの I/O レジスタには H を付加した命名が行われます。また、1 バイトの I/O レジスタの名前は、下位から順に LL、LH、HL、HH を付加した命名が行われます。

また、最大アクセスサイズが 2 バイトの I/O レジスタの構成の場合、通常では最大のアクセスサイズを持つ I/O レジスタの名前を基準として、1 バイトの I/O レジスタの名前は、下位 1 バイトの I/O レジスタには L、上位 1 バイトの I/O レジスタには H を付加した命名が行われます。

2.3 ビットレジスタと連続ビット

RH850 の I/O レジスタには、ビット単位で定義されるものがあります。特定の 1 ビットに対応するものをビットレジスタ、3～6 ビットのような連続するビットに対応するものを連続ビットと呼びます。ビットレジスタと連続ビットには、特に命名規則はありません。

例として、RSCAN0.C0CFG と同じアドレスには 4 つの連続ビットが用意されています。

表 3. RSCAN0.C0CFG と同じアドレスのビットレジスタ、連続ビットの構成

I/O レジスタ名	ビット位置	ビット長
RSCAN0.BRP	0 ビット目	10 ビット（0～9 ビット）
RSCAN0.TSEG1	16 ビット目	4 ビット（16～19 ビット）
RSCAN0.TSEG2	20 ビット目	3 ビット（20～22 ビット）
RSCAN0.SJW	24 ビット目	2 ビット（24～25 ビット）

表 4. RSCAN0.C0CFG とビットレジスタ、連続ビットの対応関係

ビット位置	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
I/O レジスタ名							SJW			TSEG2			TSEG1			
ビット位置	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I/O レジスタ名							BRP									

3. I/0ヘッダ・ファイルについて

本章では I/O ヘッダ・ファイルについて説明します。

3.1 I/Oヘッダ・ファイルの構成

I/O ヘッダ・ファイルは、主にモジュールを定義した構造体、構造体の先頭アドレスの定義、コメントの 3 つで構成されます。

モジュールを定義した構造体は、I/O レジスタをメンバに持つ構造体が出力されます。この構造体をモジュール、メンバを I/O レジスタとした、”モジュール名.I/O レジスタ名”を記述することができます。

構造体の先頭アドレスの定義は、構造体にモジュールの先頭アドレスを定義します。これにより、C ソース中に”モジュール名.I/O レジスタ名”へアクセスするコードを記述すると、コンパイル時に I/O レジスタのアドレスへのアクセスに変換されます。

コメントは、I/O ヘッダ・ファイルで定義した I/O レジスタ名の一覧が出力されます。

3.2 RH850のI/OレジスタとI/Oヘッダ・ファイルの対応関係

同じモジュール内であるアドレスに配置される I/O レジスタが 1 つの場合と、複数ある場合に分けて、I/O ヘッダ・ファイルと I/O レジスタの対応関係を説明します。

3.2.1 あるアドレスに配置されるI/Oレジスタが1つの場合

下記の I/O ヘッダ・ファイルを例に使用します。なお、__tag の後ろの数字は生成した I/O ヘッダ・ファイルによって異なります。また、I/O レジスタ名やアドレスは品種により異なります。以降の I/O ヘッダ・ファイルについても同様です。

```
struct __tag4637
{                                                       /* Module          */
    union __tag2402 ISR;                                /* ISR             */
    unsigned char  dummy14[2];                          /* Reserved        */
    unsigned short MBR;                                 /* MBR             */
    unsigned char  dummy15[2];                          /* Reserved        */
    unsigned short MBRC;                                /* MBRC            */
};

#define AUD       (*(volatile struct __tag4637 *)0xFA000000) /* AUD */
```

あるアドレスに割り当てられている I/O レジスタが 1 つの場合の例として AUD.MBR という名前の I/O レジスタを使用します。この I/O レジスタのアクセスサイズは 2byte です。

まず、モジュール名 AUD に対応する構造体を考えます。#define の行で__tag4637 の構造体は AUD として定義されており、0xFA000000 番地に割り当てられています。

次にモジュールの構造体である__tag4637 を見ると、メンバに MBR があり、型は unsigned short 形です。よって、AUD.MBR は I/O ヘッダ・ファイルでも AUD.MBR と定義されます。

AUD.MBR にアクセスするコードは、以下のように記述します。

```
#include  “iodefine.h”

void func()
{
    AUD.MBR = 0x3;
};
```

3.2.2 あるアドレスに配置されるI/Oレジスタが複数ある場合（命名規則あり）

下記の I/O ヘッダ・ファイルを例に使用します。

```
struct __tag572
{                                                   /* Bit Access       */
    unsigned short BRP:10;                          /* BRP[9:0]         */
    unsigned char  :6;                              /* Reserved Bits    */
    unsigned char  TSEG1:4;                         /* TSEG1[19:16]     */
    unsigned char  TSEG2:3;                         /* TSEG2[22:20]     */
    unsigned char  :1;                              /* Reserved Bits    */
    unsigned char  SJW:2;                           /* SJW[25:24]       */
    unsigned char  :6;                              /* Reserved Bits    */
};

union __tag2879
{                                                   /* IOR              */
    unsigned long  UINT32;                          /* 32-bit Access    */
    unsigned short UINT16[2];                       /* 16-bit Access    */
    unsigned char  UINT8[4];                        /* 8-bit Access     */
    struct __tag572 BIT;                            /* Bit Access       */
};

struct __tag4662
{                                                   /* Module           */
    union __tag2879 C0CFG;                          /* C0CFG            */
    union __tag2880 C0CTR;                          /* C0CTR            */
    union __tag2881 C0STS;                          /* C0STS            */
    union __tag2882 C0ERFL;                         /* C0ERFL           */
};

#define RSCAN0    (*(volatile struct __tag4662 *)0xFF000000) /* RSCAN0 */
```

あるアドレスに配置されるI/Oレジスタが複数ある場合の例として、2.2 で挙げたRSCAN0.C0CFGを使用します。

まず、モジュール名 RSCAN0 に対応する構造体を考えます。#define の行で__tag4662 の構造体は RSCAN0 として定義されており、0xFF000000 番地に割り当てられています。

次にモジュールの構造体である__tag4662 を見ると、メンバに C0CFG があり、型は__tag2879 の共用体であることがわかります。同じアドレスに複数の I/O レジスタが配置されている場合、共用体で表現されます。

__tag2879 の共用体を見ると、unsigned long 型の UINT32、unsigned short 型の配列 UINT16、unsigned char 型の配列 UINT8 と、BIT という名前の__tag572 型の構造体が定義されています。UINT32、UINT16、UINT8 はそれぞれ 4 バイト、2 バイト、1 バイトのレジスタに対応します。UINT16 と UINT8 は配列で定義されており、I/O ヘッダ・ファイルで定義しているマクロ L,H,LL,LH,HL,HH を配列の添え字に使用します。

__tag572 型の構造体には、BRP、TSEG1、TSEG2、SJW という名前のビットフィールドが定義されています。このビットフィールドは、連続ビットに対応します。

まとめますと、RSCAN0.C0CFG と同じアドレスに配置された I/O レジスタは、I/O ヘッダ上では表 5 のように定義されます。

表 5. I/O レジスタ名と I/O ヘッダ・ファイルでの定義の対応関係

I/O レジスタ名	アドレス	アクセスサイズ	I/O ヘッダ･ファイルでの定義
RSCAN0.COCFG	0xFF000000	4 バイト	RSCAN0.COCFG.UINT32
RSCAN0.COCFGL	0xFF000000	2 バイト	RSCAN0.COCFG.UINT16[L]
RSCAN0.COCFGH	0xFF000002	2 バイト	RSCAN0.COCFG.UINT16[H]
RSCAN0.COCFGLL	0xFF000000	1 バイト	RSCAN0.COCFG.UINT8[LL]
RSCAN0.COCFGLH	0xFF000001	1 バイト	RSCAN0.COCFG.UINT8[LH]
RSCAN0.COCFGHL	0xFF000002	1 バイト	RSCAN0.COCFG.UINT8[HL]
RSCAN0.COCFGHH	0xFF000003	1 バイト	RSCAN0.COCFG.UINT8[HH]
RSCAN0.BRP	0xFF000000	10 ビット	RSCAN0.COCFG.BIT.BRP
RSCAN0.TSEG1	0xFF000000	4 ビット	RSCAN0.COCFG.BIT.TSEG1
RSCAN0.TSEG2	0xFF000000	3 ビット	RSCAN0.COCFG.BIT.TSEG2
RSCAN0.SWJ	0xFF000000	2 ビット	RSCAN0.COCFG.BIT.SWJ

3.2.3 あるアドレスに配置されるI/Oレジスタが複数ある場合（命名規則なし）

あるアドレスに配置される I/O レジスタが複数あり、かつ 2.2 で挙げたような命名規則がない場合、I/O ヘッダ・ファイルには次のように出力されます。

a) アクセスサイズが同じ場合

同じアドレスに配置された I/O レジスタ名を定義した#define マクロを出力します。

```
struct __tag4665
{                                                       /* Module          */
    union __tag2965 SMR;                                /* SMR             */
    unsigned char  dummy686[3];                         /* Reserved        */
    unsigned char  BRR;                                 /* BRR             */
    unsigned char  dummy687[3];                         /* Reserved        */
    union __tag2966 SCR;                                /* SCR             */
    unsigned char  dummy688[3];                         /* Reserved        */
    unsigned char  TDR;                                 /* TDR             */
    unsigned char  dummy689[3];                         /* Reserved        */
    union __tag2967 SSR;                                /* SSR             */
    unsigned char  dummy690[3];                         /* Reserved        */
    unsigned char  RDR;                                 /* RDR             */
    unsigned char  dummy691[3];                         /* Reserved        */
    union __tag2968 SCMR;                               /* SCMR            */
    unsigned char  dummy692[3];                         /* Reserved        */
    union __tag2969 SEMR;                               /* SEMR            */
};

#define MDDR BRR
#define SCI30     (*(volatile struct __tag4665 *)0xFF000000) /* SCI30 */
```

表 6.I/O レジスタ名と I/O ヘッダ・ファイルでの定義の対応関係

I/O レジスタ名	アドレス	アクセスサイズ	I/O ヘッダ・ファイルでの定義
SCI30.BRR	0xFF000004	1バイト	SCI30.BRR
SCI30.MDDR	0xFF000004	1バイト	SCI30.MDDR

SCI30.MDDR にアクセスするコードは、以下のように記述します。

```
#include  “iodefine.h”

void func()
{
    SCI30.MDDR = 0x34;
};
```

b) アクセスサイズが異なる場合

REGS8、REGS16 という名前の共用体を出力します。共用体の名前は I/O レジスタのアクセスサイズに依存します。

```
union __tag2827
{                                                   /* IOR              */
    unsigned long  UINT32;                          /* 32-bit Access    */
    unsigned char  UINT8[4];                        /* 8-bit Access     */
    struct
    {
        union
        {                                           /* IOR              */
            unsigned char  UINT8;                   /* 8-bit Access     */
        } ECERDB;
        union
        {                                           /* IOR              */
            unsigned char  UINT8;                   /* 8-bit Access     */
        } ECECRD;
        union
        {                                           /* IOR              */
            unsigned char  UINT8;                   /* 8-bit Access     */
        } ECHORD;
        union
        {                                           /* IOR              */
            unsigned char  UINT8;                   /* 8-bit Access     */
        } ECSYND;
    } REGS8;
    struct __tag520 BIT;                            /* Bit Access       */
};
struct __tag4656
{                                                   /* Module           */
    union __tag2825 E710CTL;                        /* E710CTL          */
    unsigned char  dummy627[2];                     /* Reserved         */
    union __tag2826 E710TMC;                        /* E710TMC          */
    unsigned char  dummy628[2];                     /* Reserved         */
    union __tag2827 E710TRC;                        /* E710TRC          */
    union __tag2828 E710TED;                        /* E710TED          */
    union __tag2829 E710EAD;                        /* E710EAD          */
};

#define E7RCOM      (*(volatile struct __tag4656 *)0xFF000000) /* E7RCOM */
```

表 7. I/O レジスタ名と I/O ヘッダ・ファイルでの定義の対応関係

I/O レジスタ名	アドレス	アクセスサイズ	I/O ヘッダ・ファイルでの定義
E7RC0C.E710TRC	0xFF000008	4 バイト	E7RC0C.E710TRC.UINT32
E7RC0C.ECERDB	0xFF000008	1バイト	E7RC0C.E710TRC.REGS8.ECERDB.UINT8
E7RC0C.ECECRD	0xFF000009	1バイト	E7RC0C.E710TRC.REGS8.ECECRD.UINT8
E7RC0C.ECHORD	0xFF00000A	1バイト	E7RC0C.E710TRC.REGS8.ECHORD.UINT8
E7RC0C.ECSYND	0xFF00000B	1バイト	E7RC0C.E710TRC.REGS8.ECSYND.UINT8

共用体が I/O ヘッダ・ファイルにどのように出力にされるかは I/O レジスタの組み合わせにより異なりますので、ご利用になる I/O ヘッダ・ファイルをご確認ください。

E7RC0C.E710TRC、E7RC0C.ECERDB にアクセスするコードは、以下のように記述します。

```
#include “iodefine.h”

void func()
{
    E7RC0C.E710TRC.UINT32 = 0x750C;
    E7RC0C.E710TRC.REGS8.ECERDB.UINT8 = 0x14;
};
```

4. CS+V3.02の新機能について

本章では CS+V3.02 の新機能を使用して生成した I/O ヘッダ・ファイルについて説明します。

4.1 I/Oヘッダ・ファイルをカスタマイズする機能について

I/O ヘッダ・ファイルをカスタマイズする機能として、I/O ヘッダ・ファイルに出力するモジュールを選択する機能、I/O ヘッダ・ファイルを分割する機能が追加されました。これらの機能を使用して生成した I/O ヘッダ・ファイルについて説明します。

4.1.1 I/Oヘッダ・ファイルに出力するモジュールを選択する機能

I/O ヘッダ・ファイルに出力するモジュールを選択して生成した場合、I/O ヘッダ・ファイルで定義される構造体、共用体の名前が通常と異なります。通常は__tag の後ろに数字が付加されますが、この機能を使用した場合、__tag の後ろにモジュール名と_(アンダーバー)が挿入され、その後に数字が付加されます。

```
// 通常のタグ
struct __tag0
struct __tag1

// モジュールを選択した場合のタグ(タグと数字の間にモジュール名と_が挿入される)
struct __tagFLASH_0
struct __tagEINT_1
```

4.1.2 I/Oヘッダ・ファイルを分割する機能

I/O ヘッダ・ファイルを分割して生成した場合、I/O ヘッダ・ファイル中の多重 include 防止用のマクロが通常と異なります。通常は__デバイス名 IODEFINE_HEADER__が生成されますが、この機能を使用した場合、デバイス名の後ろに、生成する I/O ヘッダ・ファイル名から生成した文字列が挿入されます。この文字列は生成する I/O ヘッダ・ファイル名を大文字に、.(ドット)を_(アンダーバー)に変換したものです。

なお、この機能を使用して生成した複数の I/O ヘッダ・ファイルは、同時に include することができます。

```
// 通常のマクロ
#ifndef __R7F701270IODEFINE_HEADER__
#define __R7F701270IODEFINE_HEADER__

// ファイルを分割した場合のマクロ(マクロにファイル名から生成した文字列が挿入される)
#ifndef __R7F701270_IODEFINE2_H_IODEFINE_HEADER__
#define __R7F701270_IODEFINE2_H_IODEFINE_HEADER__
```

4.2 MISRA-C:2004ルールに適合したI/Oヘッダ・ファイルを出力する機能について

この機能を使用した場合、以下の MISRA-C:2004 ルールに適合した I/O ヘッダ・ファイルを出力します。

- Rule 6.3: typedefs that indicate size and signedness should be used in place of the basic types.
- Rule 10.6: A "U" suffix shall be applied to all constants of unsigned type.
- Rule 18.4: Unions shall not be used.

ルールに適合により変化する内容について記述します。

4.2.1 Rule 6.3: typedefs that indicate size and signedness should be used in place of the basic types.

Rule 6.3 に適合させるため、通常使用している unsigned char, unsigned short, unsigned long ではなく、I/O ヘッダ・ファイル内で uint8, uint16, uint32 を定義し、使用します。

```
// この機能を使用した場合の型定義
typedef unsigned char   uint8;
typedef unsigned short  uint16;
typedef unsigned long   uint32;
```

4.2.2 Rule 10.6: A "U" suffix shall be applied to all constants of unsigned type.

Rule 10.6 に適合させるため、構造体にアドレスを定義するマクロのアドレス表記の値に U を付加します。

```
// 通常の出力
#define AUD         (*(volatile struct __tag1126 *)0xFA005000) /* AUD */

// この機能を使用した場合の出力
#define AUD         (*(volatile struct __tag1126 *)0xFA005000U) /* AUD */
```

4.2.3 Rule 18.4: Unions shall not be used

Rule 18.4 に適合させるため、I/O ヘッダ・ファイル内で一切の共用体を使用しません。このため、同一アドレスに複数の I/O レジスタが配置されていた場合、アクセスサイズが最大の I/O レジスタのみを出力します。

```
// 通常の出力(同一アドレスのI/Oレジスタに共用体を使用する)
struct __tag1
{                                                   /* Bit Access       */
    unsigned char  IDST:1;                          /* IDST             */
    unsigned char  :7;                              /* Reserved Bits    */
    unsigned char  :8;                              /* Reserved Bits    */
    unsigned char  :8;                              /* Reserved Bits    */
    unsigned char  :8;                              /* Reserved Bits    */
};
union __tag1107
{                                                   /* IOR              */
    unsigned long  UINT32;                          /* 32-bit Access    */
    unsigned short UINT16[2];                       /* 16-bit Access    */
    unsigned char  UINT8[4];                        /* 8-bit Access     */
};
union __tag1108
{                                                   /* IOR              */
    unsigned long  UINT32;                          /* 32-bit Access    */
    unsigned short UINT16[2];                       /* 16-bit Access    */
    unsigned char  UINT8[4];                        /* 8-bit Access     */
    struct __tag1 BIT;                              /* Bit Access       */
};
struct __tag2233
{                                                   /* Module           */
    union __tag1107 SELFID0;                        /* SELFID0          */
    union __tag1107 SELFID1;                        /* SELFID1          */
    union __tag1107 SELFID2;                        /* SELFID2          */
    union __tag1107 SELFID3;                        /* SELFID3          */
    union __tag1108 SELFIDST;                       /* SELFIDST         */
    // 略
};


// この機能を使用した場合の出力(アクセスサイズが最大のI/Oレジスタのみを出力)
struct __tag1
{                                                   /* IOR              */
    uint32 UINT32;                                  /* 32-bit Access    */
};
struct __tag2
{                                                   /* IOR              */
    uint32 UINT32;                                  /* 32-bit Access    */
};
struct __tag1127
{                                                   /* Module           */
    struct __tag1 SELFID0;                          /* SELFID0          */
    struct __tag1 SELFID1;                          /* SELFID1          */
    struct __tag1 SELFID2;                          /* SELFID2          */
    struct __tag1 SELFID3;                          /* SELFID3          */
    struct __tag2 SELFIDST;                         /* SELFIDST         */
    // 略
};
```

以上

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2015.03.27	-	初版
2.00	2015.09.30	12 – 14	CS+V3.02 の新機能の説明として 4 章を追加
		全体	“IO”を”I/O”に変更

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ものですが、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も、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/