お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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。
5. 輸出に際しては、「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。本資料に記載されている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の目的で使用しないでください。また、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に使用することができません。
6. 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。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されてい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生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   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、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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。

H8/3687

ターミナルソフトとの送受信

要旨

H8/3687 グループは、高速 H8/300H CPU を核に、システム構成に必要な周辺機能を集積したシングルチップマイクロコンピュータです。H8/300H CPU は、H8/300 CPU と互換性のある命令体系を備えています。

H8/3687 グループは、システム構成に必要な周辺機能として、4 種類のタイマ、I^2C バスインタフェース、シリアルコミュニケーションインタフェース、10 ビット A/D 変換器を内蔵しており、高度な制御システムの組み込み用マイコンとして活用できます。

H8/300H シリーズ-H8/3687-アプリケーションノートは､H8/3687 グループの内蔵周辺機能を単独で使用した場合の動作例を示した"基礎編"により構成されており､ユーザにてソフトウェア設計およびハードウェア設計の際、ご参考として役立てていただけるようにまとめたものです。

なお、本アプリケーションノートに掲載されているプログラム、回路等の動作は確認しておりますが、実際にご使用になる場合は、必ず動作確認の上ご使用くださいますようにお願い致します。

動作確認デバイス

H8/3687

目次

1. 概要

H8/3687 の SCI3 インタフェースにより、PC のターミナルソフトとの送受信処理を行います。このサンプルプログラムを利用すればターミナルソフトへのデバッグメッセージの出力やコマンド受信を容易に行うことができます。

2. 構成

図 2.1 に、ターミナルソフトの接続図を示します。

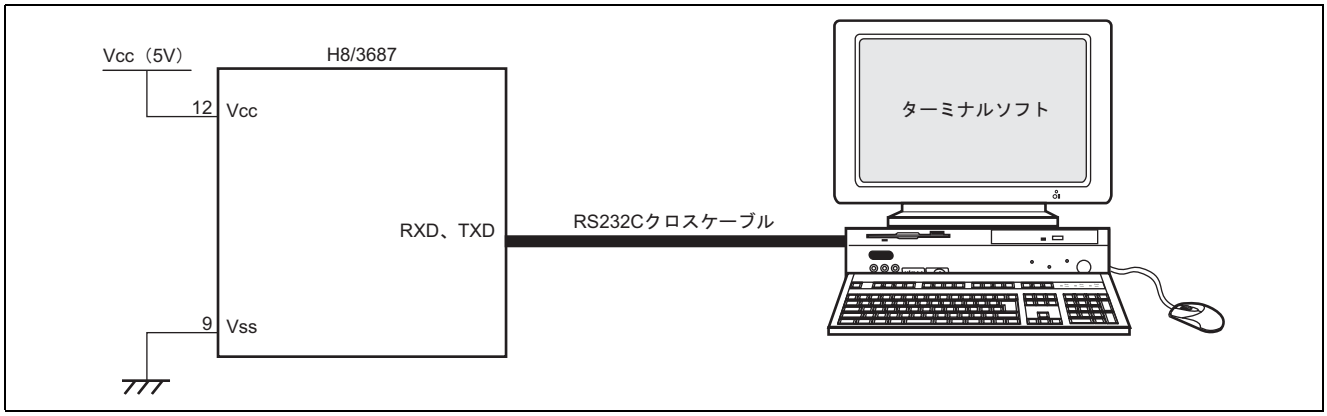
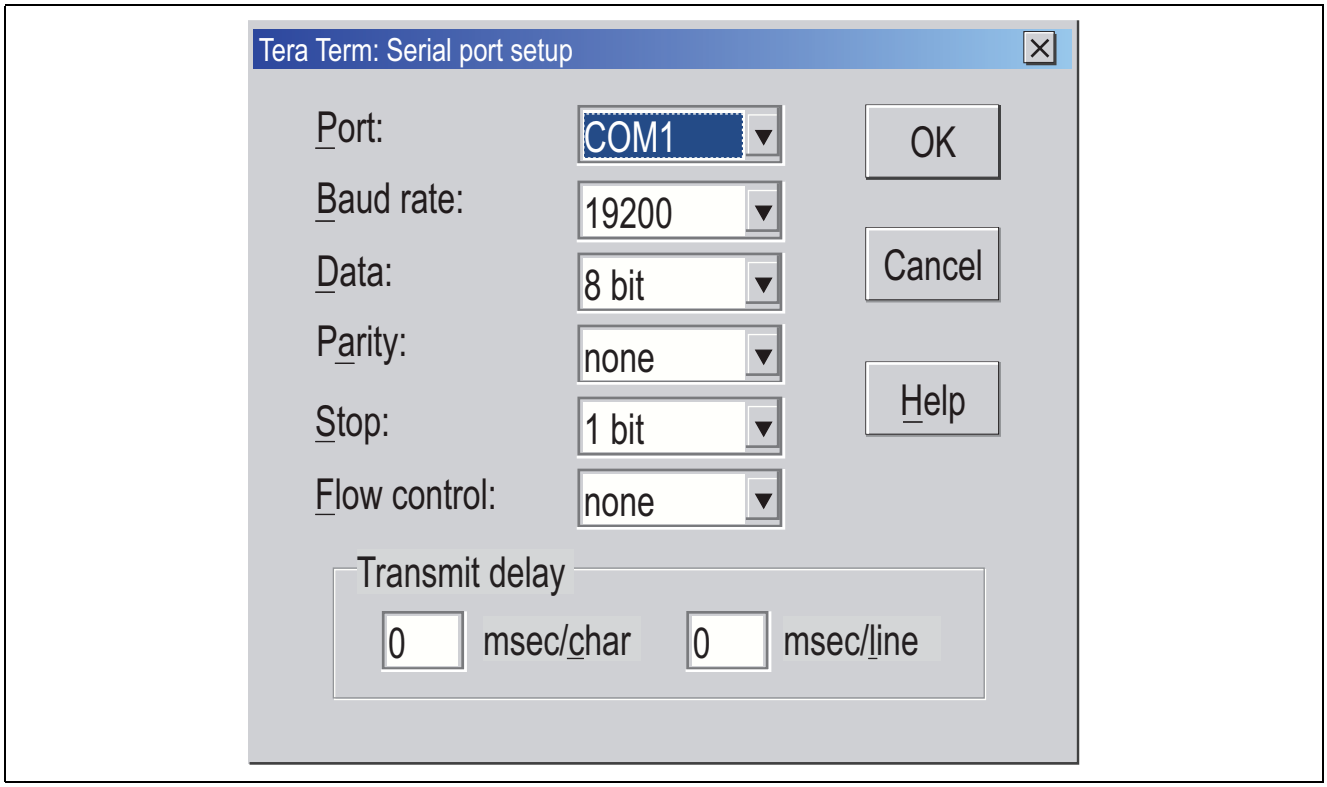


図 2.1　ターミナルソフト I²C EEPROM との接続図

仕様

- H8/3687 の動作周波数：16MHz
- 使用したターミナルソフト：Tera term
- 通信仕様



3. サンプルプログラム

3.1 機能

1. ターミナルソフトにメッセージを出力します。
2. ターミナルソフトからコマンドを受信し、H8 マイコン内レジスタの読み書きを行います。

3.2 組み込み方法

1. サンプルプログラム 10-A　define 定義を組み込んでください。
2. サンプルプログラム 10-B　共通変数宣言を組み込んでください。
3. サンプルプログラム 10-C　プロトタイプ宣言を組み込んでください。
4. サンプルプログラム 10-D　プロトタイプ宣言を組み込んでください。
5. サンプルプログラム 10-B　プロトタイプ宣言を組み込んでください。
   - 4.1　h8_sci3 のリセットベクタを追加してください。
   - 4.2　SCI3 の初期設定処理を追加してください。
   - 4.3　共通ルーチンを追加してください。
   - 4.4　SCI3 の割り込み処理を追加して下さい。

3.3 サンプルプログラムの変更

サンプルプログラムそのままでは、システムが動作しないことがあります。お客様のプログラムやシステム環境に合わせて修正を行う必要があります。

1. IO レジスタの構造体定義は、ルネサス Web
   http://www.renesas.com/jpn/products/mpumcu/tool/crosstool/iodef/index.html　で無償入手できる定義ファイルをご利用になるとサンプルプログラムをそのまま使用することができます。独自に作成される場合は、サンプルプログラム中に使用している IO レジスタの構造体を適宜変更して下さい。

2. サンプルプログラム中、SCI3 インタフェースの状態監視のためにタイマ Z を 10ms ごとに起動しタイムアウト監視するよう設計してあります。タイマ処理は、お客様の都合に合わせて変更して構いません。もちろんそのまま使用することも可能です。サンプルプログラムのタイマ処理をそのまま使用する場合は、次の変更を行ってください。
   A. サンプルプログラム 10-E
      - 5.1　タイマ Z のリセットベクタを追加してください。
      - 共通変数として、com_timer を追加して下さい。
      - TimerZ の初期設定処理を追加してください。
        （タイマ Z が 10ms に割り込むようご使用になるマイコンの動作周波数に合わせて GRA の設定値を変更してください。設定値は、H8/3687 のハードウェアマニュアルを、変更箇所はサンプルプログラム中の program note を参照して下さい。）
      - タイマ Z の割り込み処理を追加して下さい。

3. 使用する回線スピードとマイコンの動作周波数に合せて SCI3 インタフェースの転送レート BRR を設定して下さい。設定値は、H8/3687 のハードウェアマニュアルを、変更箇所はサンプルプログラム中の program note を参照して下さい。本サンプルプログラムでは、回線スピードは 19200bps に設定してあります。

3.4 使用方法

1. ターミナルソフトにメッセージを出力します。

```
void com_cnsl_msg( char *fmt)
```

引数	説　　明
*fmt	コンソールに出力したいメッセージを直接記述します。 書式は、ライブラリ stdio.h の sprintf を参照して下さい。 メッセージ文字数は、80 文字以内です。

戻り値	なし
0	正常終了
0x104	データ送信エラー

使用例

```
MSC_VER = 0x02
MSC_MAJOR_REV = 0x00
MSC_MINOR_REV = 0x01
ret = com_cnsl_msg("Ver-Rev-Mrev = %02hX-%02hX-%02hX¥n¥r" , MSC_VER , MSC_MAJOR_REV
                   , MSC_MINOR_REV ) ;
if (ret  !=0){ /* データ送信エラー    */}
   else{ /* データ送信正常終了 */}
```

【コンソール出力結果】

```
Ver-Rev-Mrev = 02-00-01
```

2. ターミナルソフトからコマンドを受信し、H8 マイコン内レジスタの読み書きを行います。

- H8 マイコンレジスタの read（全レジスタの read）

【コンソール入力】 hr

【出力例】

```
 Key in >hr
         0        4        8        C
addr:F700 = 238888E0 E1F8FFFF FFFFFFFF FFFFFFFF
addr:F710 = 008888C0 E0F8FFFF FFFFFFFF FFFFFFFF
addr:F720 = FD0E8880 FF00FFFF FFFFFFFF FFFFFFFF
addr:F730 = 70FCFFFF FFFFFFFF FFFFFFFF FFFFFFFF
                    ,
                    ,
                    ,
```

- H8 マイコンレジスタの read（レジスタ指定）

【コンソール入力】 hr△(**アドレス**:2byte)　　(**△はスペース**)

【出力例】

```
Key in >hr f700
  addr : F700 = 23
```

- H8 マイコンレジスタの write（レジスタ指定）

【コンソール入力】 hw△(**アドレス**:2byte) (**データ**:1byte)　(**△はスペース**)

【出力例】

```
Key in >hw f700 34
  (H8 REG WRITE F700 <- 34)
  addr : F700 = 34
```

3.5 動作説明

ここでは、各処理のプログラムリストを説明しながら、お客様のプログラムに応用する場合の留意点を示します。

1. ターミナルソフトにメッセージを出力します。
   プログラムソースリストの com_cnsl_msg の program note を参照して下さい。

2. ターミナルソフトからコマンドを受信し、H8 マイコン内レジスタの読み書きを行います。
   プログラムソースリストの h8_sci3（SCI 割り込み処理）の program note を参照して下さい。

3.6 使用レジスター覧

本サンプルプログラムで使用する H8 マイコンの内部レジスタの一覧を示します。内容の詳細は、H8/3687 グループハードウェアマニュアルを参照してください。

1. SCI3 関連レジスタ

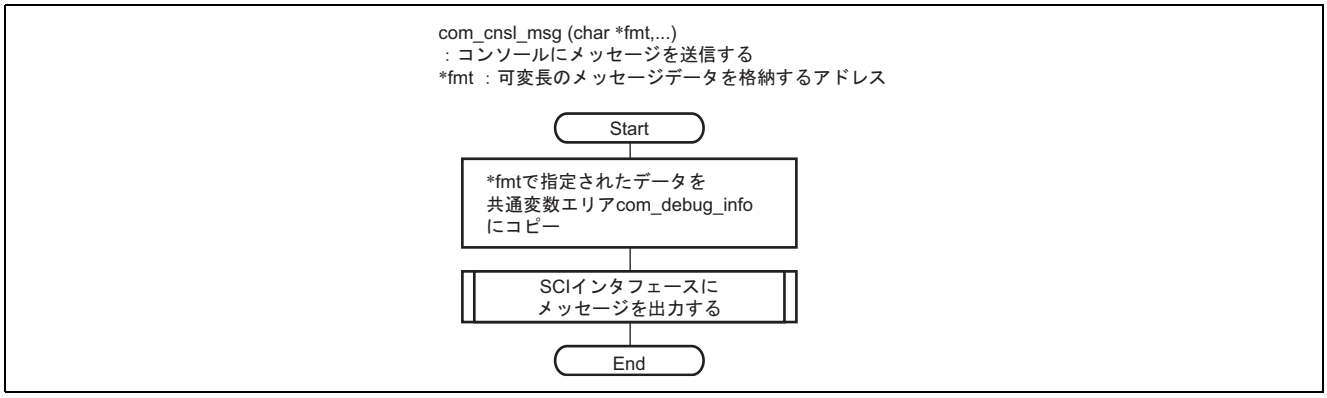
名称	概　　要
レシーブデータレジスタ（RDR）	受信データを格納するための 8 ビットのレジスタ
トランスミットデータレジスタ（TDR）	送信データを格納するための 8 ビットのレジスタ
シリアルモードレジスタ（SMR）	シリアルデータ通信フォーマットと内蔵ボーレートジェネレータのクロックソースを選択するためのレジスタ
シリアルコントロールレジスタ 3（SCR3）	送受信動作と割り込み制御、送受信クロックソースの選択を行うためのレジスタ
シリアルステータスレジスタ（SSR）	SCI3 のステータスフラグと送受信マルチプロセッサビット
ビットレートレジスタ（BRR）	ビットレートを設定する 8 ビットのレジスタ

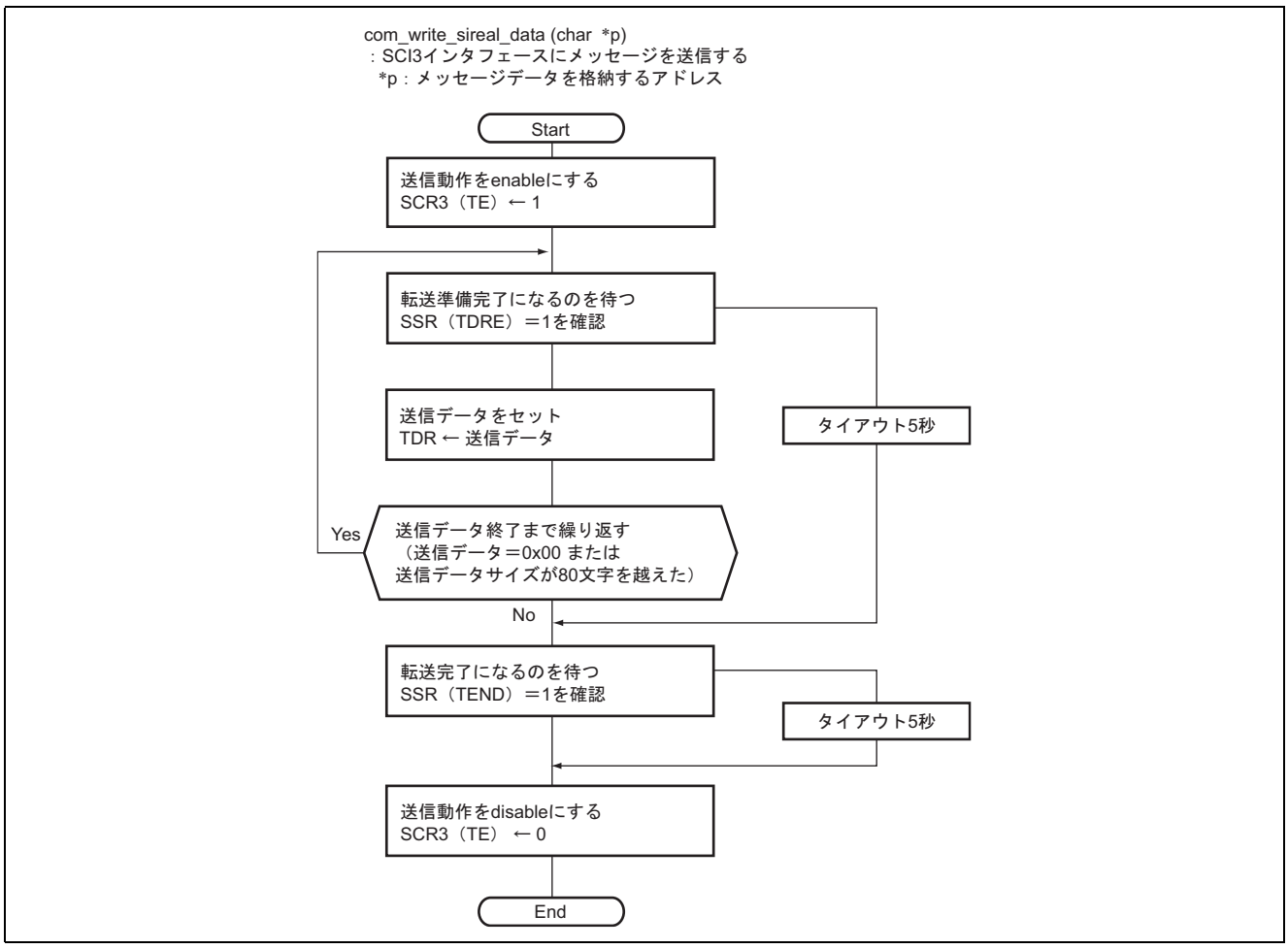
2. タイマ Z 関連レジスタ
   タイマ Z は多様な機能を持ちますが、本サンプルプログラムでは、GRA レジスタのコンペアマッチ機能で 10ms ごとの割り込みを発生させるようにしています。

名称	概　　要
タイマスタートレジスタ（TSTR）	TCNT の動作／停止を選択する
タイマモードレジスタ（TMDR）	バッファ動作の設定、同期動作を選択する
タイマ PWM モードレジスタ（TPMR）	端子を PWM モードに設定する。 本サンプルプログラムでは使用しません。
タイマファンクションコントロールレジスタ（TFCR）	各動作モードの設定や出力レベルを選択する。 本サンプルプログラムでは使用しません。
タイマアウトプットマスタイネーブルレジスタ（TOER）	チャネル 0、1 の出力を許可／禁止を設定する。
タイマアウトプットコントロールレジスタ（TOCR）	コンペアマッチが最初に起こるまでの初期出力を設定する。
タイマカウンタ（TCNT）	16 ビットのリード／ライト可能なレジスタで、入力したクロックによりカウント動作を行う。
ジェネラルレジスタ A、B、C、D（GRA、GRB、GRC、GRD）	GR は 16 ビットのリード／ライト可能なレジスタで、各チャネルに 4 本、計 8 本あります。アウトプットコンペアレジスタとインプットキャプチャレジスタの機能の切り換えを TIORA、TIORC により行う。
タイマコントロールレジスタ（TCR）	TCNT のカウンタクロック選択、外部クロック選択時のエッジ選択、およびカウンタクリア要因を選択する。
タイマ I/O コントロールレジスタ（TIORA）	GRA、GRB をアウトプットコンペアレジスタとして使用するか、インプットキャプチャレジスタとして使用するかを選択する。
タイマステータスレジスタ（TSR）	TCNT のオーバフロー／アンダフローの発生、および GRA、GRB、GRC、GRD のコンペアマッチ／インプットキャプチャの発生を示す
タイマインタラプトイネーブルレジスタ（TIER）	オーバフロー割り込み要求、GR のコンペアマッチ／インプットキャプチャ割り込み要求の許可／禁止を制御する。
PWM モードアウトプットレベルコントロールレジスタ（POCR）	PWM モード時のアクティブレベルを制御する。 本サンプルプログラムでは使用しません。

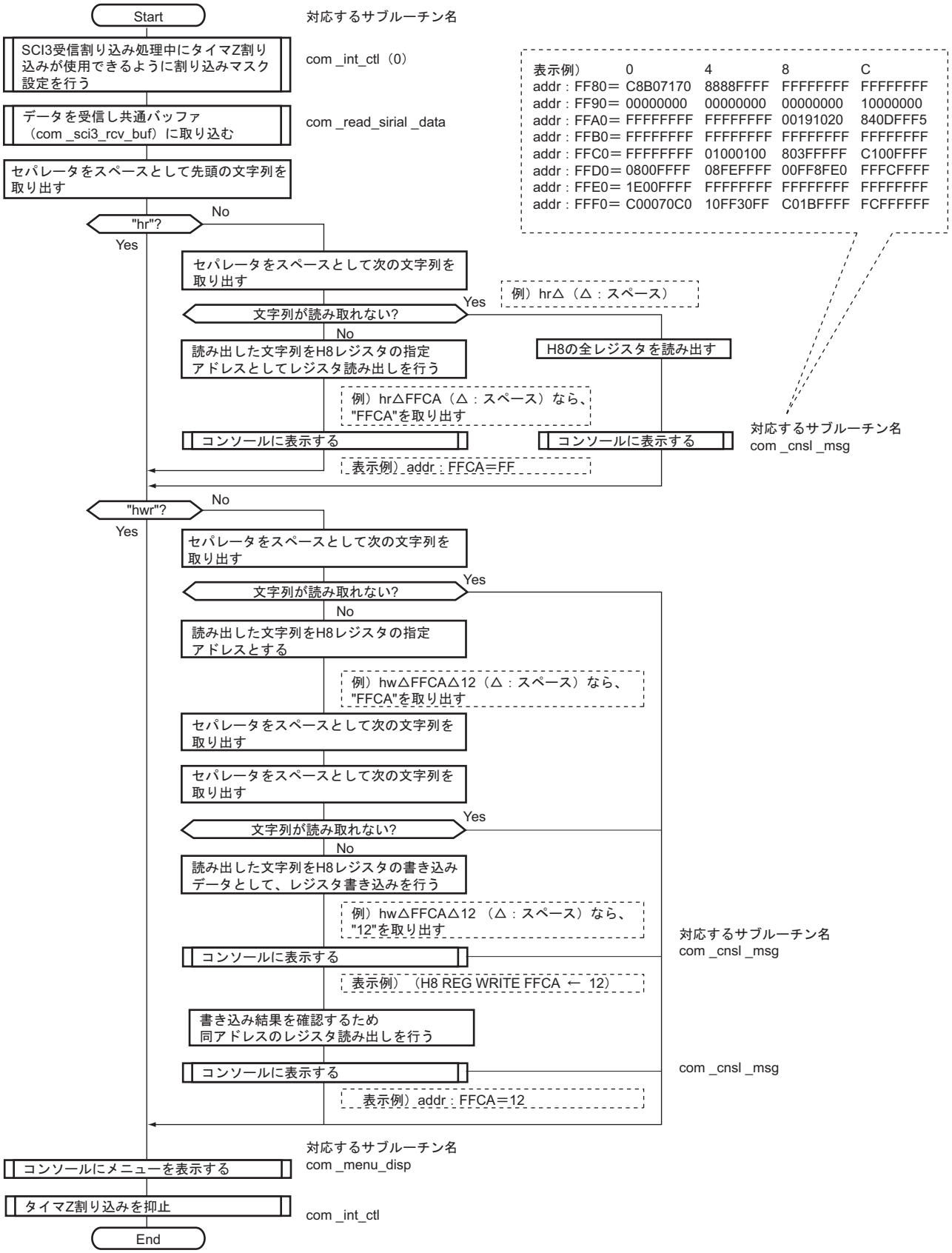
3.7 フローチャート

プログラムの処理フローを以下に示します。

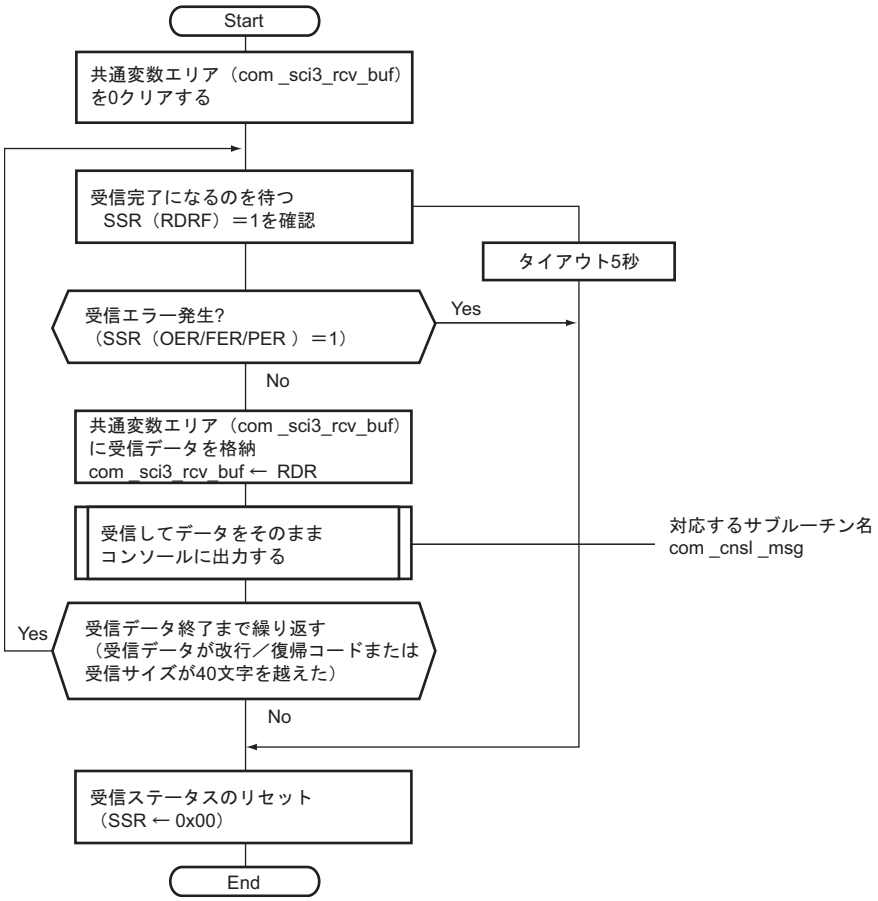




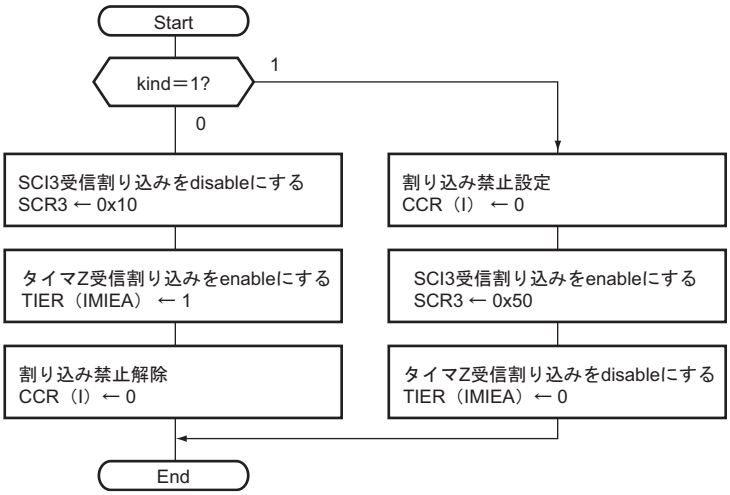
h8_sci 3
：SCI3受信割り込み処理



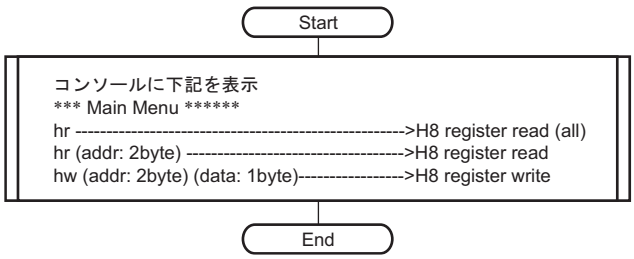
com _read_sirial _data
：SCI3インタフェースからメッセージを受信する
受信データは、共通変数エリア（com _sci3_rcv_buf）に最大40文字まで受信する。



com_int_ctl (unsigned char kind)
：割りマスクを制御する
kind：制御内容の種別
0：タイマZ割り込みを有効にする
1：タイマZ割り込みを抑止する



com _menu_disp
：コンソールに操作メニューを表示する



対応するサブルーチン名
com _cnsl _msg

3.8 プログラムリスト

```
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* 1 . サンプルプログラム 10-A define 定義---------------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/************************************************************************************************************************************************/
/*   RS232C アクセスエラー              コード                                                                                                    */
/************************************************************************************************************************************************/
#define   SCI3_RCV_OVERRUN          0x0101
#define   SCI3_RCV_FRAME_ERR        0x0102
#define   SCI3_RCV_PARITY_ERR       0x0103
#define   SCI3_SND_ENBL_TOUT        0x0104
#define   SCI3_RCV_FULLL_TOUT       0x0105
#define   SCI3_INVALID_PARM         0x0106

/************************************************************************************************************************************************/
/*   値の定義                                                                                                                                   */
/************************************************************************************************************************************************/
#define   DBG_INFO_SIZE             80

/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* 2 . サンプルプログラム 10-B プロトタイプ宣言----------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* コンソール処理                                                                                                                               */
void com_menu_disp(void) ;
unsigned int com_read_sireal_data ();
unsigned int com_write_sireal_data (char *p);
void com_int_ctl (unsigned char kind) ;
void com_cnsl_msg(char *fmt, ...)  ;

/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* 3 . サンプルプログラム 10-C 共通変数宣言---------------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------ */

    /************************************************************************************************************************************************/
    /*   共通変数                                                                                                                                   */
    /************************************************************************************************************************************************/
    #ifndef        _DEFINE_COMMON_TABLE
        extern
    #endif                                                          /* _DEFINE_COMMON_TABLE                          */
        struct    {
            unsigned char cw_dipsw ;                                /* cw_dipsw                                      */
            unsigned char batt_off ;                                /* battery off                                   */
            unsigned char dummy ;                                   /* dummy                                         */
        }com_global;
```

```
    /************************************************************************************************************************/
    /*    コンソールインターフェース                                                                                        */
    /************************************************************************************************************************/
    #ifndef        _DEFINE_COMMON_TABLE
         extern
    #endif                                                          /* _DEFINE_COMMON_TABLE                                 */
         char com_sci3_rcv_buf[40] ;                                /* SCI3 受信バッファ f                                  */

    #ifndef        _DEFINE_COMMON_TABLE
         extern
    #endif                                                          /* _DEFINE_COMMON_TABLE                                 */
         char com_debug_info[DBG_INFO_SIZE] ;                       /* デバッグメッセージ情報                               */

/* ---------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------- */
/* 4 . サンプルプログラム 10-D 共通処理ソースコード---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------- */

/* ---------------------------------------------------------------------------------------------------------------------- */
/* 4.1 リセットベクターの追加-------------------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------- */

/*    飛び先を h8_sci3 に設定してください                                                                                  */

/* ---------------------------------------------------------------------------------------------------------------------- */
/* 4.2 初期設定処理                    ---------------------------------------------------------------------------------- */
/* ---------------------------------------------------------------------------------------------------------------------- */

    /* ###################################################################################################################### */
    /* ###################################################################################################################### */
    /*                                                                                                                        */
    /*         シリアルインターフェース(SCI3)の設定                                                                           */
    /*                                                                                                                        */
    /* ###################################################################################################################### */
    /* ###################################################################################################################### */
    /************************************************************************************************************************/
    /*    PMR1       IO ポート 1 の使用方法を設定                                                                           */
    /*    TXD2       = 0   TXD_2(SCI3_2 の送信端子)                                                                         */
    /*    TXD        = 1   TXD(SCI3 の送信端子)として使用                                                                   */
    /************************************************************************************************************************/
    IO.PMR1.BIT.TXD     = 1 ;
         /* ##(program note)################################################################################################ */
         /* ##  P22/TXD ポートを TXD 端子として使用するように定義します                                                    ## */
         /* ###################################################################################################################### */

    /************************************************************************************************************************/
    /*    SCR3       SCI3 のコントロールレジスタ設定                                                                        */
    /*         TIE        = 0   送信完了割込みを抑止                                                                         */
    /*         RIE        = 0   受信割込み(ここではまだ抑止=>sleep 前に設定)                                                 */
    /*         TE         = 0   送信イネーブル(実際に使用時に設定、ここでは 0 に設定)                                        */
    /*         RE         = 0   受信イネーブル(実際に使用時に設定、ここでは 0 に設定)                                        */
    /*         MPIE       = 0   (未使用)                                                                                     */
    /*         TEIR       = 0   (未使用)                                                                                     */
    /*         CKE1:0     = 00 クロックソースを内蔵ボートレートジェネレータとする                                            */
    /************************************************************************************************************************/
    SCI3.SCR3.BYTE = 0x00 ;
```

```
    /*****************************************************************************************************************************************/
    /*   SMR            SCI3 のモード設定                                                                                                    */
    /*        COM       = 0  調歩同期で動作                                                                                                  */
    /*        CHR       = 0  データ長 8bit                                                                                                   */
    /*        PE        = 0  パリティビットなし                                                                                              */
    /*        PM        = 0  （未使用）                                                                                                      */
    /*        STOP      = 0  1 ストップビット                                                                                                */
    /*        MP        = 0  （未使用）                                                                                                      */
    /*        CKS1:0    = 00 内蔵ボージェネレートのクロックソース φクロック                                                                  */
    /*****************************************************************************************************************************************/
    SCI3.SMR.BYTE  = 0x00 ;

    /*****************************************************************************************************************************************/
    /*   BRR            19200bps 設定                                                                                                        */
    /*****************************************************************************************************************************************/
    SCI3.BRR = 25;
        /* ##(program note)################################################################################################################ */
        /* ## BRR は、必要とする転送レートに従って設定値を変更すること                                                                    ## */
        /* ## 詳細は、H8/3687 ハードウェアマニュアルを参照すること                                                                         ## */
        /* ################################################################################################################################ */

/* ------------------------------------------------------------------------------------------------------------------------------------- */
/* 4.3 共通処理処理                  ------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------------------- */

/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*                                                                                                                                       */
/*                                  シリアルインタフェース                                                                               */
/*                                                                                                                                       */
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*   1.モジュール名称:com_cnsl_msg                                                                                                       */
/*   2.機能概要:デバック用コンソールにメッセージを表示する                                                                               */
/*****************************************************************************************************************************************/
void com_cnsl_msg(char *fmt, ...) {

    va_list   argp ;
    va_start(argp,fmt) ;
    vsprintf(&(com_debug_info[0]),fmt,argp) ;
        /* ##(program note)################################################################################################################ */
        // ##  fmt で指定された可変長の文字列を com_debug_info にコピーします。                                                            ## */
        // ##  com_debug_info は 80 文字分のエリアを定義してあるため、fmt で指定できる文字列は                                             ## */
        // ##  80 文字以内ですが、com_debug_info を定義しなおせば、もっと長い文字列も表示可能です。                                        ## */
        /* ################################################################################################################################ */

    com_write_sireal_data(com_debug_info) ;
        /* ##(program note)################################################################################################################ */
        // ##  com_debug_info の内容をコンソールに出力します。                                                                             ## */
        /* ################################################################################################################################ */

    va_end(argp) ;
}
```

```
/*****************************************************************************************************************************************************/
/*    1.モジュール名称:com_menu_disp                                                                                                                   */
/*    2.機能概要:console command の Main menu 表示                                                                                                     */
/*****************************************************************************************************************************************************/
void com_menu_disp (void)
{

     com_cnsl_msg("¥n¥r") ; //改行
     com_cnsl_msg("*** Main Menu ***** ¥n¥r") ;
     com_cnsl_msg(" hr --------------------------> H8 register read(all)¥n¥r") ;
     com_cnsl_msg(" hr (addr:2byte) --------------> H8 register read ¥n¥r") ;
     com_cnsl_msg(" hw (addr:2byte) (data:1byte) -> H8 register write ¥n¥r") ;
     com_cnsl_msg("¥n¥r") ; //改行
     com_cnsl_msg("Key in >") ;

}

/*****************************************************************************************************************************************************/
/*    1.モジュール名称:com_write_sireal_data                                                                                                           */
/*    2.機能概要:シリアル if にデータを書き込む                                                                                                        */
/*    return code                                                                                                                                      */
/*    NORMAL_END          :    正常終了                                                                                                                */
/*    SCI3_SND_ENBL_TOUT  :    送信処理が enable にならない                                                                                            */
/*****************************************************************************************************************************************************/
unsigned int com_write_sireal_data ( char *p )
{
     unsigned char   send_data;
     int   i;
     unsigned int   ret;

     i = 0 ;
     ret = NORMAL_END ;

     SCI3.SCR3.BIT.TE = 1     ;                                  /* 送信動作を enable にする                                  */
     do {
          if((*p == 0x00) || (i>DBG_INFO_SIZE)) {               /* 送信データの最後を検出                                    */
               break ;
          }

          com_timer.wait_100ms_sci3 = 50 ;
          while(SCI3.SSR.BIT.TDRE == 0){                        /* 次のデータ転送可能になるまで待つ                          */
               if (com_timer.wait_100ms_sci3 == 0){             /* 5s 間 データ転送不可なら抜ける                            */
                    ret = SCI3_SND_ENBL_TOUT ;                  /* エラーでも何もしない                                      */
                    goto exit ;
               }
          }
          SCI3.TDR = *p++  ;                                    /* データ送信 => この動作で SCI3.SSR.BIT.TDRE はリセットされる */
          i++ ;
     } while(1);

exit :
     com_timer.wait_100ms_sci3 = 50 ;
     while(SCI3.SSR.BIT.TEND == 0){                             /* 送信完了まで待つ                                          */
          if (com_timer.wait_100ms_sci3 == 0){                  /* 5s 間 データ転送不可なら抜ける                            */
               ret = SCI3_SND_ENBL_TOUT ;                       /* エラーでも何もしない                                      */
          }
     }

     SCI3.SCR3.BIT.TE =  0    ;                                 /* 送信動作を disable にする                                 */

     return (ret) ;

}
```

```
/*****************************************************************************************************************************************************/
/*   1.モジュール名称:com_read_sireal_data                                                                                                           */
/*   2.機能概要:シリアル if からデータを取り込む                                                                                                     */
/*   return code                                                                                                                                     */
/*   NORMAL_END          :    正常終了                                                                                                               */
/*   SCI3_RCV_OVERRUN    :    over run                                                                                                               */
/*   SCI3_RCV_FRAME_ERR  :    frame err                                                                                                              */
/*   SCI3_RCV_PARITY_ERR :    parity error                                                                                                           */
/*****************************************************************************************************************************************************/
unsigned int com_read_sireal_data (void)
{
     unsigned char   rcv_data;
     int i ;

     unsigned int   ret;

     ret = NORMAL_END ;

     for (i=0; i<40; i++){                                          /* 受信バッファクリア                                                          */
          com_sci3_rcv_buf[0] = 0;
     }

     i = 0 ;
     do {
          com_timer.wait_100ms_sci3 = 300 ;
          while(SCI3.SSR.BIT.RDRF == 0){                            /* Recive データが届くまで待つ(Max30 秒)                                       */
               if (com_timer.wait_100ms_sci3 == 0){                 /* 1s 間  データ転送付加なら抜ける                                             */
                    ret =SCI3_RCV_FULLL_TOUT ;                      /* エラーでも何もしない                                                        */
                    goto exit ;
               }

               if ((SCI3.SSR.BYTE & 0x38) !=0) {                    /* エラー発生                                                                  */
                    if (SCI3.SSR.BIT.OER == 1){                     /* over run                                                                    */
                         SCI3.SSR.BIT.OER = 0;                      /* 要因をリセットする                                                          */
                         SCI3.SSR.BIT.RDRF = 0;                     /* recive data full bit リセット                                               */
                         ret = SCI3_RCV_OVERRUN ;
                         goto exit ;
                    }
                    if (SCI3.SSR.BIT.FER == 1){                     /* framing err                                                                 */
                         SCI3.SSR.BIT.FER = 0;                      /* 要因をリセットする                                                          */
                         SCI3.SSR.BIT.RDRF = 0;                     /* recive data full bit リセット                                               */
                         ret = SCI3_RCV_FRAME_ERR ;
                         goto exit ;
                    }
                    if (SCI3.SSR.BIT.PER == 1){                     /* parity err                                                                  */
                         SCI3.SSR.BIT.PER = 0;                      /* 要因をリセットする                                                          */
                         SCI3.SSR.BIT.RDRF = 0;                     /* recive data full bit リセット                                               */
                         ret = SCI3_RCV_PARITY_ERR ;
                         goto exit ;
                    }
               }
          }
```

```
          rcv_data = SCI3.RDR ;                                   /* データ受信 => この動作で SCI3.SSR.BIT.RDRF はリセットされる */

                                                                  /* 受信したデータをそのままコンソールに返す                    */
                                                                  /* -->入力データを画面に表示するため                           */
          com_cnsl_msg("%c",rcv_data) ;

          if (i<40){                                              /* バッファをオーバーするデータを受け取ったら読み捨てる        */
               com_sci3_rcv_buf[i] = rcv_data ;                   /* 受信バッファにデータを格納する                              */
          }

          i ++ ;
     } while((rcv_data != 0x0a) && (rcv_data != 0x0d));           /* 改行なら受信を終わる                                        */

exit :
     SCI3.SSR.BYTE = 00 ;                                         /* recive data full bit リセット                               */

     return (ret) ;

}

/*****************************************************************************************************************************/
/*    1.モジュール名称:com_int_ctl                                                                                             */
/*    2.機能概要:set_imask_ccr <- 0 とし、TimerZ 割込みのみ有効にする                                                          */
/*****************************************************************************************************************************/
void com_int_ctl (unsigned char kind)
{

     if (kind == 0){
          /*************************************************************************************************************/
          /*    SCI3 受信割込みを disable にする。                                                                      */
          /*************************************************************************************************************/
          SCI3.SCR3.BYTE      = 0x10;                             /* RCV int disable                                   */

          /*************************************************************************************************************/
          /*    TimerZ 割込みを有効にする                                                                              */
          /*************************************************************************************************************/
          TZ0.TIER.BIT.IMIEA  = 1 ;                               /* timerZ IMFA  enable                               */

          /*************************************************************************************************************/
          /*    割込み禁止解除                                                                                         */
          /*************************************************************************************************************/
          set_imask_ccr(0);                                       /* 割込み許可                                        */
     }
     else{
          /*************************************************************************************************************/
          /*    割込み禁止設定(理由;割込み処理中に割込みが入るのを防ぐため)                                            */
          /*************************************************************************************************************/
          set_imask_ccr(1);                                       /* 割込み禁止                                        */
          /*************************************************************************************************************/
          /*    SCI3 受信割込みを enable にする                                                                        */
          /*************************************************************************************************************/
          SCI3.SCR3.BYTE      = 0x50;                             /* RCV int のみ enable                               */

          /*************************************************************************************************************/
          /*    TimerZ 割込みを無効にする                                                                              */
          /*************************************************************************************************************/
          TZ0.TIER.BIT.IMIEA  = 0 ;                               /* timerz IMFA  disable                              */
     }
}
```

```
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/* 4.4 SCI 受信割込み処理    Z---------------------------------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/*****************************************************************************************************************************************************/
/*    1.モジュール名称:h8_sci3                                                                                                                           */
/*    2.機能概要:RS232C からの割込み                                                                                                                     */
/*****************************************************************************************************************************************************/
#pragma interrupt( h8_sci3 )
void h8_sci3( void )
{
          /* ##(program note)################################################################################################################### */
          // ##  コンソールにキー入力されると SCI 割込みが発生し、ここに飛んで来ます。                                                          ## */
          // ##################################################################################################################################### */
     int i , j ;
     unsigned int ret ;
     char *cmd_ptr, *addr_ptr , *data_ptr ;

     unsigned int    h8_addr;

     union {
      unsigned long  d_long ;
      unsigned int  d_int[2] ;
      unsigned char      d_byte[4];
     } buf;

     ret = NORMAL_END ;
     /*****************************************************************************************************************************************************/
     /*    set_imask_ccr <- 0 とし、IREQ0-3、SCI  rcvint のマスクを閉じる                                                                                 */
     /*    TimerZ 割込みは有効とする                                                                                                                      */
     /*****************************************************************************************************************************************************/
     com_int_ctl(0) ;                                                  /* ccr <- 0 とし、TimerZ 割込みのみ有効にする                                */
          /* ##(program note)################################################################################################################### */
          /* ##  H8/3687 は、割込み中は、ハードウェアにより CCR(I)←1 となり、NMI とアドレスブレーク                                              ## */
          /* ##  以外の割込みが全て抑止されます。このルーチンは、割込み処理中に Timer 割込みを使用可能とするためにマスクレジスタを操作します      ## */
          /* ##################################################################################################################################### */

     /*****************************************************************************************************************************************************/
     /*    シリアルインターフェースよりデータを受信する                                                                                                   */
     /*****************************************************************************************************************************************************/
     ret = com_read_sireal_data () ;                                   /* データを受信し、共通バッファに取り込む                                    */
          /* ##(program note)################################################################################################################### */
          // ##  先頭の 1byte だけ割込みで受信し、残りの文字列は、改行または復帰コードを受信するまで、                                          ## */
          // ##  com_read_sireal_data の中でポーリングで受信します。                                                                            ## */
          // ##  受信したデータは、com_sci3_rcv_buf に 40 文字まで記憶します。                                                                  ## */
          // ##################################################################################################################################### */

     if (ret != 0) {goto exit ;}

     /*****************************************************************************************************************************************************/
     /*    コマンド内容を読み込む                                                                                                                         */
     /*****************************************************************************************************************************************************/
     cmd_ptr = strtok(com_sci3_rcv_buf , " ") ;
```

```
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*   H8 レジスタ処理                                                                                                                     */
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*   H8 レジスタ read                                                                                                                    */
/*****************************************************************************************************************************************/
if (strncmp(cmd_ptr,"hr",2) == 0){                                      /* コマンド="hr"                                                */
        // ##(program note)################################################################################################ */
        // ##  先頭の 2 文字を評価します。"hr"なら H8 レジスタの Read です。                                                          ## */
        // ##  "hr"を"abc"としたいならば、3 文字を評価するため if (strncmp(cmd_ptr,"abc",3) == 0)と書きます。                        ## */
        // ################################################################################################################ */

    com_cnsl_msg("¥n¥r");
        // ##(program note)################################################################################################ */
        // ##  コンソールの表示を一行改行します。                                                                                    ## */
        // ################################################################################################################ */

    addr_ptr = strtok(NULL , " ") ;                                     /* ADDRESS を読み込む                                           */
        // ##(program note)################################################################################################ */
        // ##  文字列のセパレータを" "(スペース)として次の文字列を調べます。                                                          ## */
        // ##  セパレーターを","(コンマ)にしたいならば、addr_ptr = strtok(NULL , ",") ;と書きます。                                  ## */
        // ################################################################################################################ */

    if (addr_ptr == NULL) {                                             /* addr スペースの時                                            */
        // ##(program note)################################################################################################ */
        // ##  addr_ptr に何も読み取れない(=NULL)ならば、アドレス指定なしで全レジスタの読み出し処理を行います。                        ## */
        // ################################################################################################################ */

        /* 全レジスタを表示                                                                                                             */

        h8_addr = 0xF700 ;                                              /* F700 - F77F の表示                                           */
            com_cnsl_msg("           0       4        8      C¥n¥r");
            //        addr:YYYY = xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
        for (i=0; i<8; i++){
            // ##(program note)############################################################################################ */
            // ##  F700 - F7FF のレジスタの内容を、1 行 16byte ずつ表示します                                                    ## */
            // ############################################################################################################ */
            com_cnsl_msg("  addr:%04X = ", h8_addr + i*16);

            for (j=0; j<4; j++){
                com_cnsl_msg("%02X%02X%02X%02X "
                    , *((unsigned char *)h8_addr+i*16 + j*4)
                    , *((unsigned char *)h8_addr+i*16 + j*4 +1)
                    , *((unsigned char *)h8_addr+i*16 + j*4 +2)
                    , *((unsigned char *)h8_addr+i*16 + j*4 +3)) ;
            }
            com_cnsl_msg("¥n¥r") ;                                      /* 改行                                                         */
        }


        h8_addr = 0xFF80 ;                                              /* FF80 - FFFF の表示                                           */
            com_cnsl_msg("           0       4        8      C¥n¥r");
            //        addr:YYYY = xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
        for (i=0; i<8; i++){
```

```
                // ##(program note)################################################################################################ */
                // ##  FF80 - FFFF のレジスタの内容を、1 行 16byte ずつ表示します                                                  ## */
                // ################################################################################################################ */
                com_cnsl_msg("  addr:%04X = ", h8_addr + i*16);

                for (j=0; j<4; j++){
                     com_cnsl_msg("%02X%02X%02X%02X "
                          , *((unsigned char *)h8_addr+i*16 + j*4)
                          , *((unsigned char *)h8_addr+i*16 + j*4 +1)
                          , *((unsigned char *)h8_addr+i*16 + j*4 +2)
                          , *((unsigned char *)h8_addr+i*16 + j*4 +3)) ;
                }
                com_cnsl_msg("¥n¥r") ;                                  /* 改行                                                 */
           }
      }
      else{
           sscanf(addr_ptr , "%4hX" , &h8_addr) ;                     /* 16 進の文字列を整数値に変換                          */
                // ##(program note)################################################################################################ */
                // ##  addr_ptr に次の文字列が読み取れたら、16 進のデータにして h8_addr に格納し、H8 のレジスタアドレスとします。    ## */
                // ################################################################################################################ */

           goto h8_read_op ;
      }
}

/*****************************************************************************************************************************/
/*   H8 レジスタ write                                                                                                       */
/*****************************************************************************************************************************/
if (strncmp(cmd_ptr,"hw",2) == 0){                                      /* コマンド="hw"                                        */
           // ##(program note)################################################################################################ */
           // ##  先頭の 2 文字を評価します。"hw"なら H8 レジスタの write です。                                               ## */
           // ##  "hr"を"abc"としたいならば、3 文字を評価するため if (strncmp(cmd_ptr,"abc",3) == 0)と書きます。              ## */
           // ################################################################################################################ */

      com_cnsl_msg("¥n¥r");
           // ##(program note)################################################################################################ */
           // ##  コンソールの表示を一行改行します。                                                                           ## */
           // ################################################################################################################ */

      addr_ptr = strtok(NULL , " ") ;                                   /* ADDRESS を読み込む                                   */
           // ##(program note)################################################################################################ */
           // ##  文字列のセパレータを" "(スペース)として次の文字列(レジスタのアドレス)を調べます。                            ## */
           // ##  セパレーターを","(コンマ)にしたいならば、addr_ptr = strtok(NULL , ",") ;と書きます。                         ## */
           // ################################################################################################################ */
      if (addr_ptr == NULL) {
                // ##(program note)################################################################################################ */
                // ##  addr_ptr に何も読み取れない(=NULL)ならば、エラーで処理を終わります                                       ## */
                // ################################################################################################################ */

           ret = SCI3_INVALID_PARM ;
           goto exit ;
      }
      else{
           sscanf(addr_ptr , "%4hX" , &h8_addr) ; /* 16 進の文字列を整数値に変換 */
                // ##(program note)################################################################################################ */
                // ##  addr_ptr に次の文字列が読み取れたら、16 進のデータにして h8_addr に格納し、H8 のレジスタアドレスとします。    ## */
                // ################################################################################################################ */
      }
```

```
        data_ptr = strtok(NULL , " ") ;                                  /* data を読み込む                                    */
             // ##(program note)################################################################################################ */
             // ##  文字列のセパレータを" "(スペース)として次の文字列(書き込みデータ)を調べます。                                  ## */
             // ################################################################################################################ */
        if (addr_ptr == NULL) {
                // ##(program note)############################################################################################ */
                // ##  addr_ptr に何も読み取れない(=NULL)ならば、エラーで処理を終わります                                       ## */
                // ############################################################################################################ */

             ret = SCI3_INVALID_PARM ;
             goto exit ;
        }
        else{
             sscanf(data_ptr , "%2X" , &buf.d_int[0]) ;                  /* 16 進の文字列を整数値に変換                        */
                // ##(program note)############################################################################################ */
                // ##  data_ptr に次の文字列が読み取れたら、16 進のデータにして buf.d_int[0]に格納し H8 のレジスタへ書き込むデータとします。  ## */
                // ############################################################################################################ */
        }

        *((unsigned char *)h8_addr) = buf.d_byte[1] ;
             // ##(program note)############################################################################################ */
             // ##  h8_addr で指定された H8 レジスタアドレスに指定データを write します。                                    ## */
             // ##  H8 のレジスタに書き込むデータは 1byte ですが、sscanf には、char 型は指定できないため                    ## */
             // ##  int 型で buf.d_int[0]を指定し、書き込み時、buf.d_byte[1]を指定しています。                               ## */
             // ################################################################################################################ */

        com_cnsl_msg("  (H8 REG WRITE %04X <- %02X) ¥n¥r",h8_addr,buf.d_byte[1]);
             // ##(program note)############################################################################################ */
             // ##  書き込み内容をコンソールに表示します。                                                                   ## */
             // ################################################################################################################ */

h8_read_op :
        buf.d_byte[0] = *((unsigned char *)h8_addr) ;

                                                                         /* 読み出したデータをコンソールに表示する             */
        com_cnsl_msg("  addr : %04X = %02X ¥n¥r",h8_addr,buf.d_byte[0]);
             // ##(program note)############################################################################################ */
             // ##  指定のレジスタを読み出してコンソールに表示します。                                                       ## */
             // ##  レジスタ write の場合は、書き込み結果の確認のため指定のレジスタを読み出してコンソールに表示します。     ## */
             // ################################################################################################################ */
    }

exit :
    /*****************************************************************************************************************************/
    /*   console command menu を表示                                                                                             */
    /*****************************************************************************************************************************/
    com_menu_disp() ;

    /*****************************************************************************************************************************/
    /*   SCI rcvint のマスクを開く                                                                                               */
    /*****************************************************************************************************************************/
    com_int_ctl(1) ;

}
```

```
/* ------------------------------------------------------------------------------------------------------------------------ */
/* 4.5 メッセージ出力例ソースコード          ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
     /*##################################################################################################################*/
     /*    デバッグメッセージを表示                                                                                          */
     /*##################################################################################################################*/
     com_cnsl_msg("H8 application sample program start!! ¥n¥r") ;

     com_cnsl_msg("Ver Rev MRev : %02hX %02hX %02hX ¥n¥r",MSC_VER,MSC_MAJOR_REV,MSC_MINOR_REV) ;

/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* 5 . サンプルプログラム 10-E TimerZ 処理---------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */

/* ------------------------------------------------------------------------------------------------------------------------ */
/* 5.1 リセットベクターの追加----------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
/*   飛び先を h8_timerz に設定してください                                                                                  */

/* ------------------------------------------------------------------------------------------------------------------------ */
/* 5.2 TimerZ 用共通変数定義          ------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
     struct    {
          int counter;                                        /* 100ms カウンタ                                          */
          int wait_10ms;                                      /* 10mswait 用                                             */
          int wait_100ms;                                     /* 100ms 単位 wait 用(共通)                                */
          int wait_100ms_scan;                                /* 100ms 単位 wait 用(for I2C)                             */
     }com_timer;

/* ------------------------------------------------------------------------------------------------------------------------ */
/* 5.3 TimerZ の初期設定              ------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
     /* ############################################################################################################### */
     /* ############################################################################################################### */
     /*                                                                                                                  */
     /*        TimerZ の設定                                                                                             */
     /*                                                                                                                  */
     /* ############################################################################################################### */
     /* ############################################################################################################### */
     /*****************************************************************************************************************/
     /*   timerz を初期設定                                                                                              */
     /*****************************************************************************************************************/
     TZ.TSTR.BYTE = 0x00 ;
     TZ.TMDR.BYTE = 0x00 ;
     TZ.TPMR.BYTE = 0x00 ;
     TZ.TFCR.BYTE = 0x00 ;
     TZ.TOER.BYTE = 0xFF ;
     TZ.TOCR.BYTE = 0x00 ;

     TZ0.TCR.BYTE   = 0x23 ;
                                                         /* CCLR[2:0] = 001 GRA コンペアマッチでカウンタクリア          */
                                                         /* CKEG[1:0] = 00 立ち上がりエッジカウント                     */
                                                         /* TPSC[2:0] = 011 内部クロックΦ/8 でカウント                  */
     TZ0.TIORA.BYTE = 0x00 ;
                                                         /* IOA[2:0] = 000 GRA はアウトプットコンペアレジスタ           */
     TZ0.TIER.BYTE  = 0x01 ;
                                                         /* IMIEA = 1 IMFA イネーブル                                   */
```

```
    TZ0.GRA        = 20000 ;                                        /* 10msec 毎に割り込み                                    */
        /* ##(program note)################################################################################################ */
        /* ## マイコンの動作周波数により設定値は異なる。設定内容は、H8/3687 のハードウェアマニュアルを参照のこと。          ## */
        /* ################################################################################################################ */

    TZ0.TCNT       = 0 ;                                            /* タイマカウンタクリア                                   */

    /****************************************************************************************************************/
    /*   timerz を start させる                                                                                      */
    /****************************************************************************************************************/
    TZ.TSTR.BYTE   = 0x01 ;                                         /* timer start                                            */
                                                                    /* STR0 = 1 TCNT_0 の start                               */

/* ------------------------------------------------------------------------------------------------------------------- */
/* 5.4 TimerZ 割込み処理              ------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------- */
/****************************************************************************************************************/
/*   1.モジュール名称:h8_TimerZ                                                                                    */
/*   2.機能概要:10msec 毎のインターバルタイマ処理                                                                  */
/*   3.来歴:REV  作成/改訂日付      作成/改訂者      改訂内容                                                      */
/*          000  2002.02.11        上田             新規作成                                                      */
/****************************************************************************************************************/
#pragma interrupt( h8_timerz )
void h8_timerz( void )
{

    /****************************************************************************************************************/
    /*   要因クリア                                                                                                 */
    /****************************************************************************************************************/
    com_global.dummy = TZ0.TSR.BYTE;                                /* dummy read                                             */

    TZ0.TSR.BIT.IMFA = 0;                                           /* IMFA clear                                             */

    /****************************************************************************************************************/
    /*   10msec 単位で -1                                                                                           */
    /****************************************************************************************************************/
    if( com_timer.wait_10ms>0 )
        com_timer.wait_10ms --;

    /****************************************************************************************************************/
    /*   カウントアップ                                                                                             */
    /****************************************************************************************************************/
    com_timer.counter++;
    if( com_timer.counter >= 10 ){
        /****************************************************************************************************************/
        /*   100msec 単位で -1                                                                                        */
        /****************************************************************************************************************/
        if( com_timer.wait_100ms>0 )
            com_timer.wait_100ms --;
        if( com_timer.wait_100ms_scan>0 )
            com_timer.wait_100ms_scan --;

        com_timer.counter = 0;
    }

}
```

4. 参考文献

- H8/3687 グループ　ハードウェアマニュアル（ルネサス テクノロジ発行）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2003.09.24	—	初版発行

安全設計に関するお願い

1. 弊社は品質、信頼性の向上に努めておりますが、半導体製品は故障が発生したり、誤動作する場合があります。弊社の半導体製品の故障又は誤動作によって結果として、人身事故、火災事故、社会的損害などを生じさせないような安全性を考慮した冗長設計、延焼対策設計、誤動作防止設計などの安全設計に十分ご留意ください。

本資料ご利用に際しての留意事項

1. 本資料は、お客様が用途に応じた適切なルネサス テクノロジ製品をご購入いただくための参考資料であり、本資料中に記載の技術情報についてルネサス テクノロジが所有する知的財産権その他の権利の実施、使用を許諾するものではありません。
2. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例の使用に起因する損害、第三者所有の権利に対する侵害に関し、ルネサス テクノロジは責任を負いません。
3. 本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他全ての情報は本資料発行時点のものであり、ルネサス テクノロジは、予告なしに、本資料に記載した製品または仕様を変更することがあります。ルネサス テクノロジ半導体製品のご購入に当たりましては、事前にルネサス テクノロジ、ルネサス販売または特約店へ最新の情報をご確認頂きますとともに、ルネサス テクノロジホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
4. 本資料に記載した情報は、正確を期すため、慎重に制作したものですが万一本資料の記述誤りに起因する損害がお客様に生じた場合には、ルネサス テクノロジはその責任を負いません。
5. 本資料に記載の製品データ、図、表に示す技術的な内容、プログラム及びアルゴリズムを流用する場合は、技術内容、プログラム、アルゴリズム単位で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。ルネサス テクノロジは、適用可否に対する責任は負いません。
6. 本資料に記載された製品は、人命にかかわるような状況の下で使用される機器あるいはシステムに用いられることを目的として設計、製造されたものではありません。本資料に記載の製品を運輸、移動体用、医療用、航空宇宙用、原子力制御用、海底中継用機器あるいはシステムなど、特殊用途へのご利用をご検討の際には、ルネサス テクノロジ、ルネサス販売または特約店へご照会ください。
7. 本資料の転載、複製については、文書によるルネサス テクノロジの事前の承諾が必要です。
8. 本資料に関し詳細についてのお問い合わせ、その他お気付きの点がございましたらルネサス テクノロジ、ルネサス販売または特約店までご照会ください。