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

シリアルコミュニケーションインタフェース
クロック同期モードによるシリアル EEPROM（SPI EEPROM）のアクセス

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 2
2. 構成 2
3. サンプルプログラム 3
4. 参考文献 39

1. 概要

H8/3687 のシリアルコミュニケーションインタフェースのクロック同期式モードを使用して、SPI EEPROM の読み書きを行います。

2. 構成

図 2.1 に、SPI EEPROM との接続図を示します。

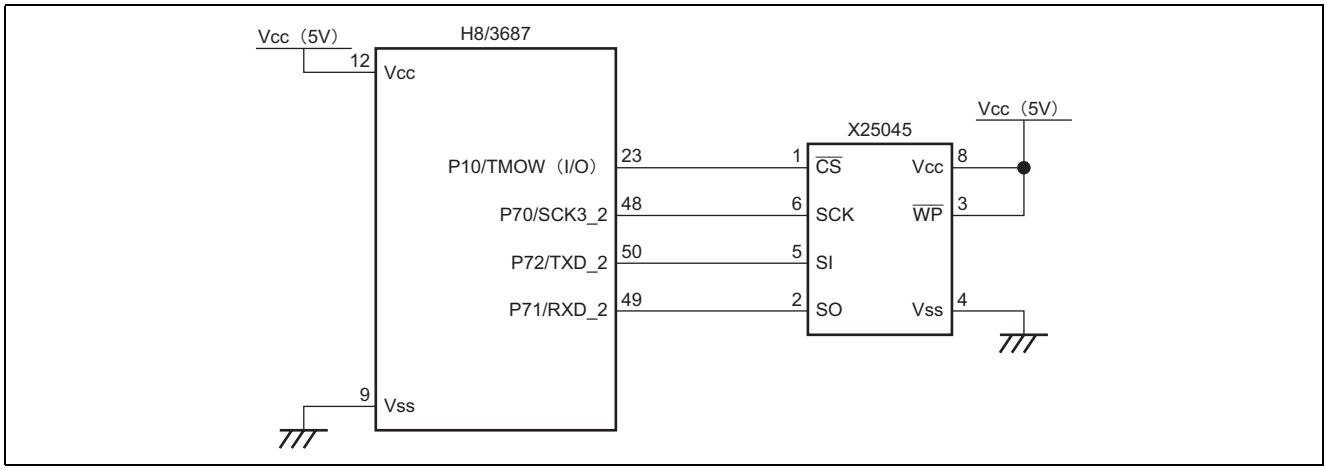


図 2.1　SPI EEPROM との接続図

仕様

- H8/3687 の動作周波数：16MHz
- SPI EEPROM（X25045）のピン仕様を表 2.1 に示します。
- SPI EEPROM 仕様：4Kbit（512×8 ビット）

表 2.1　SPI EEPROM ピン仕様

記号	機　　能
$\overline{CS}$	Chip Select Input
SO	Serial Output
SI	Serial Input
SCK	Serial Clock Input
$\overline{WP}$	Write Protect Input
Vss	Ground
Vcc	Supply Voltage
$\overline{RESET}$/RESET	Reset Output

3. サンプルプログラム

3.1 機能

1. SPI EEPROM に 1 バイトのデータを書き込みます。（Byte Write）
2. SPI EEPROM から 1 バイトのデータを読み出します。（Random Read）
3. SPI EEPROM に連続してデータを書き込みます。（Page Write）
4. SPI EEPROM から連続してデータを読み出します。（Sequential Read）

3.2 組み込み方法

1. サンプルプログラム 3-A　define 定義を組み込んでください。
2. サンプルプログラム 3-B　プロトタイプ宣言を組み込んでください。
3. サンプルプログラム 3-C　ソースプログラムを組み込んでください。
   — 初期設定処理を組み込んでください。
   — SPI EEPROM アクセス処理を組み込んでください。

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.renease.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、シリアルコミュニケーション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 5 秒でタイムアウト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 3-D
      - 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。
      - タイマ Z 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. ターゲットデバイスのスペックとマイコンの動作周波数に合せて、シリアルコミュニケーションインタフェースの転送レートを BRR レジスタで設定して下さい。
   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、転送レートは 100kbps に設定してあります。

3.4 使用方法

1. SPI EEPROM に 1 バイトのデータを書き込みます。

```
unsigned int com_spi_eeprom_write
    (unsigned int rom_addr , unsigned char rom_data )
```

引数	説　　明
rom_addr	書き込みを行う ROM アドレスを指定します
rom_data	書き込みデータを指定します

戻り値	説　　明
0	正常終了
1	異常終了（転送準備完了タイムアウト）
2	異常終了（転送完了待ちタイムアウト）
3	異常終了（受信完了待ちタイムアウト）
4	異常終了（書き込み待ちタイムアウト）

使用例

```
int ret ;
unsigned char rom_data ;
unsigned int rom_addr ;

ret = com_spi_eeprom_write ( rom_addr , rom_data )
```

2. SPI EEPROM から 1 バイトのデータを読み出します。

```
unsigned int com_spi_eeprom_read
    (unsigned int rom_addr , unsigned char *rom_data )
```

引数	説　　明
rom_addr	読み出しを行う ROM アドレスを指定します
*rom_data	読み出したデータを格納するアドレスを指定します

戻り値	説　　明
0	正常終了
1	異常終了（転送準備完了タイムアウト）
2	異常終了（転送完了待ちタイムアウト）
3	異常終了（受信完了待ちタイムアウト）
4	異常終了（書き込み待ちタイムアウト）

使用例

```
int ret ;
unsigned char *rom_data ;
unsigned int rom_addr ;

ret = com_spi_eeprom_read ( rom_addr , rom_length , *rom_data)
```

3. SPI EEPROM に連続してデータを書き込みます。

```
unsigned int com_spi_eeprom_page_write
    (unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data )
```

引数	説　　明
rom_addr	書き込みを行う ROM アドレスを指定します
rom_length	書き込みデータの長さを指定します 本サンプルプログラムで使用しているデバイスは、最大 4 バイトです。
*rom_data	書き込みデータを格納してある先頭アドレスを指定します

戻り値	説　　明
0	正常終了
1	異常終了（転送準備完了タイムアウト）
2	異常終了（転送完了待ちタイムアウト）
3	異常終了（受信完了待ちタイムアウト）
4	異常終了（書き込み待ちタイムアウト）

使用例

```
int ret ;
unsigned char *rom_data ;
unsigned int rom_length , rom_addr ;

ret = com_spi_eeprom_page_write ( rom_addr , rom_length , *rom_data)
```

4. SPI EEPROM から連続してデータを読み出します。

```
unsigned int com_spi_eeprom_seq_read
    (unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data )
```

引数	説　　明
rom_addr	読み出しを行う ROM アドレスを指定します
rom_length	読み出しデータの長さを指定します
*rom_data	読み出したデータを格納する先頭アドレスを指定します

戻り値	説　　明
0	正常終了
1	異常終了（転送準備完了タイムアウト）
2	異常終了（転送完了待ちタイムアウト）
3	異常終了（受信完了待ちタイムアウト）

使用例

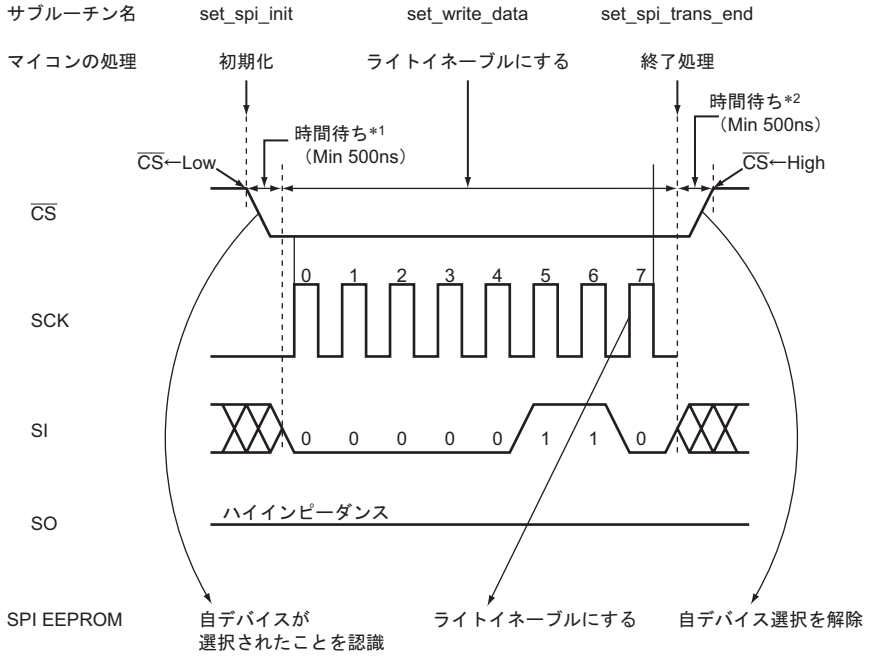
```
int ret ;
unsigned char *rom_data ;
unsigned int rom_length , rom_addr ;

ret = com_spi_eeprom_seq_read ( rom_addr , rom_length , *rom_data)
```

3.5 動作説明

以下に、動作説明を示します。

インタフェース信号の状態に対する H8 マイコンと EEEPROM の動作を下図に示します。

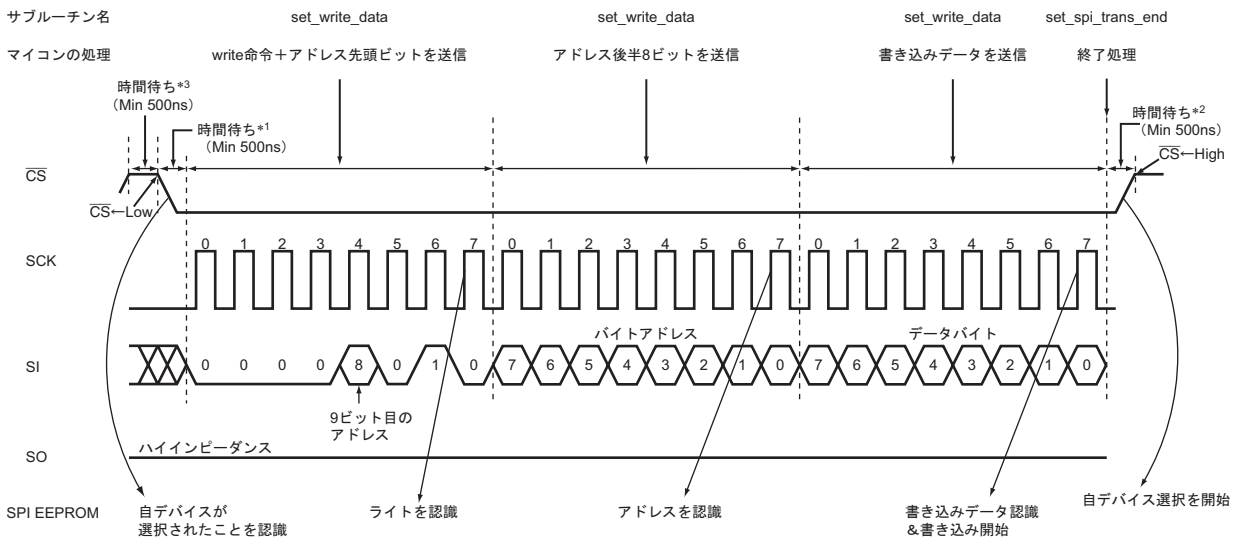
1. SPI EEPROM に 1 バイトのデータを書き込みます（Byte Write）。
   手順 a）SPI EEPROM の書き込み禁止状態を解除します。



【注】 *1 CSをONにした後、データ転送を開始するまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

*2 データ転送を終了した後、CSをOFFにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

手順 b）データを書き込みます。

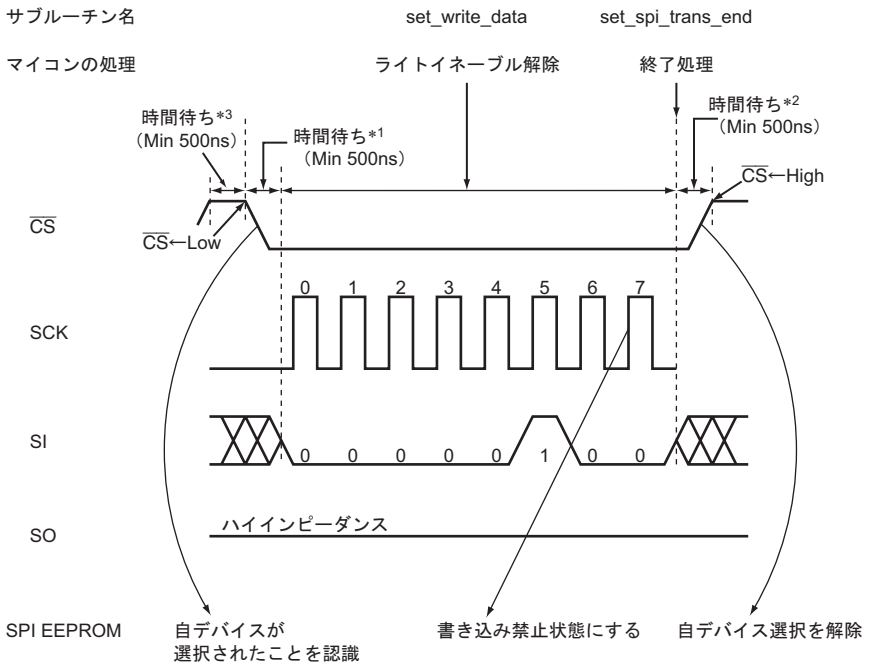


【注】 *1 CSをONにした後、データ転送を開始するまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

*2 データ転送を終了した後、CSをOFFにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

*3 CSをOFFにして再びONにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

手順 c）SPI EEPROM の書き込み禁止状態に戻します。

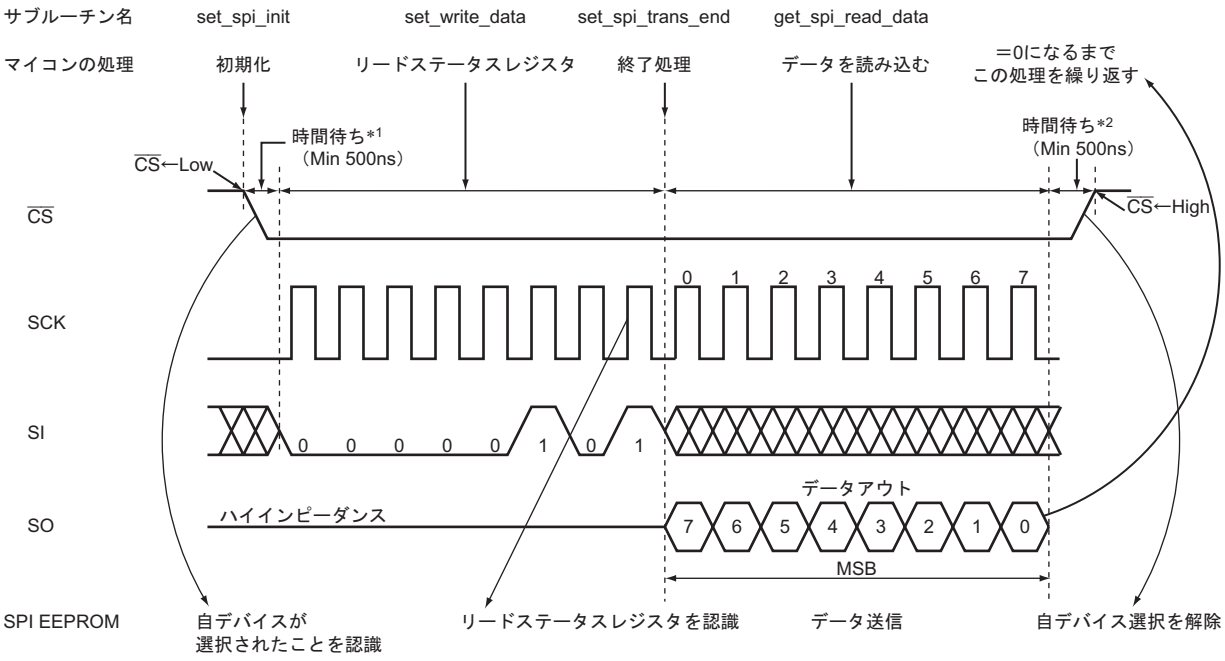


【注】 *1 CSをONにした後、データ転送を開始するまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

*2 データ転送を終了した後、CSをOFFにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

*3 CSをOFFにして再びONにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること

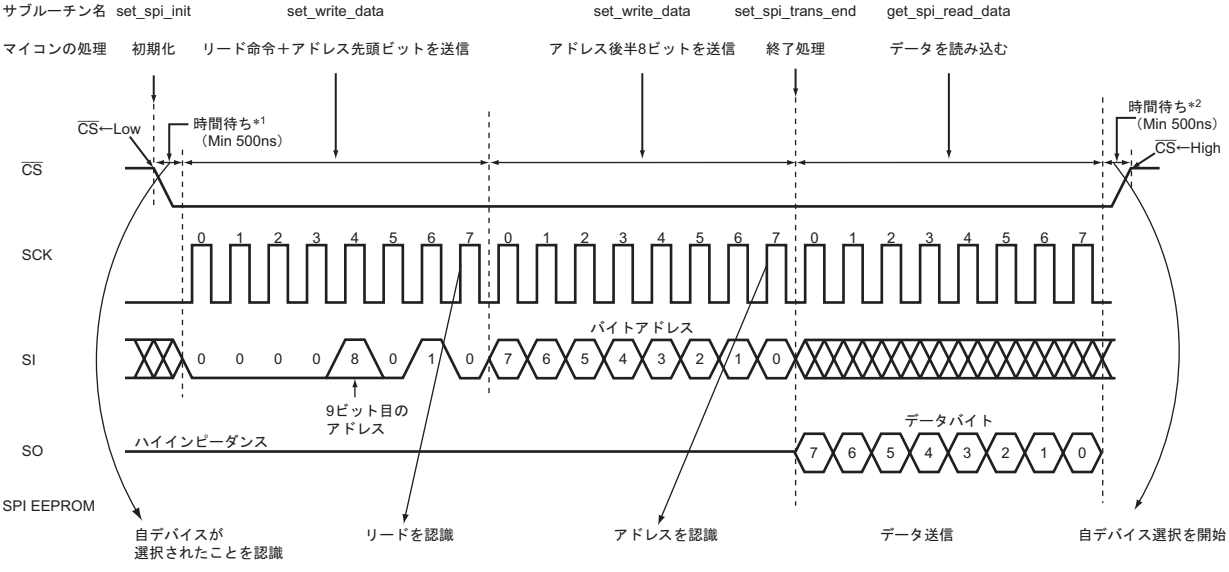
手順 d）書き込み終了をチェックする。



【注】 *1 CSをONにした後、データ転送を開始するまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

*2 データ転送を終了した後、CSをOFFにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

2. SPI EEPROM から 1 バイトのデータを読み出します（Random Read）。

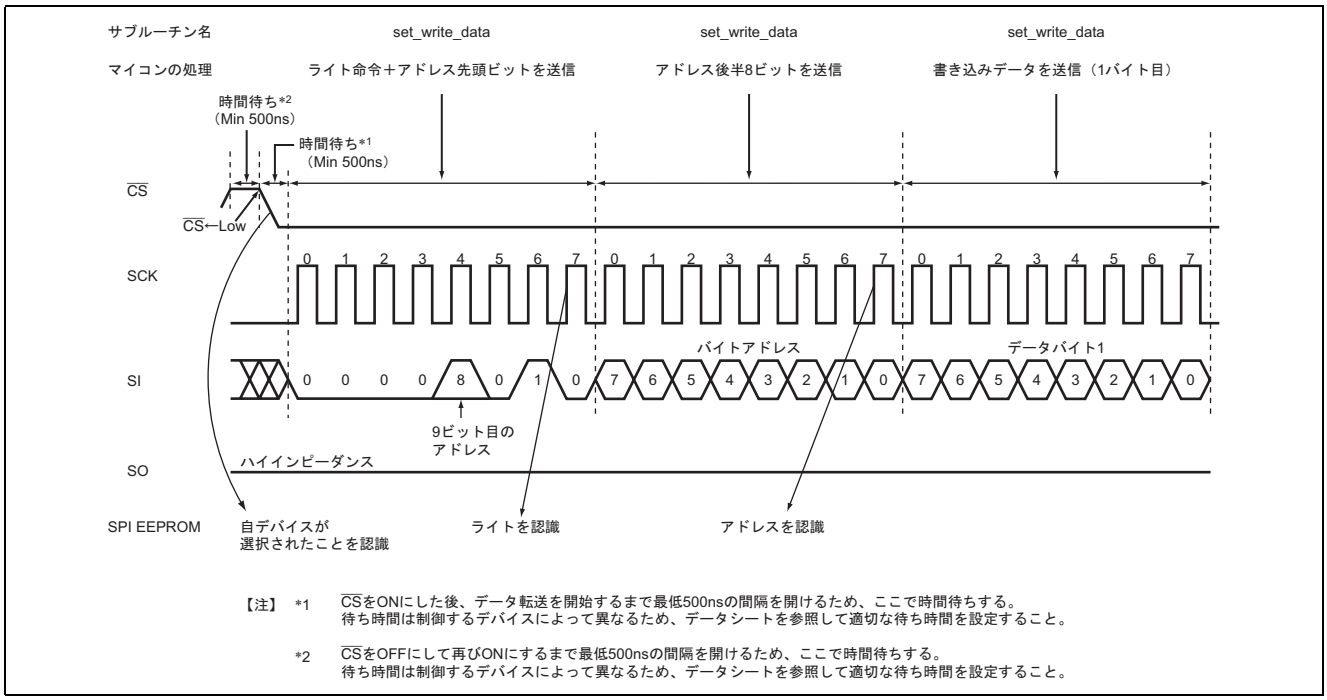


【注】 *1 CSをONにした後、データ転送を開始するまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

*2 データ転送を終了した後、CSをOFFにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

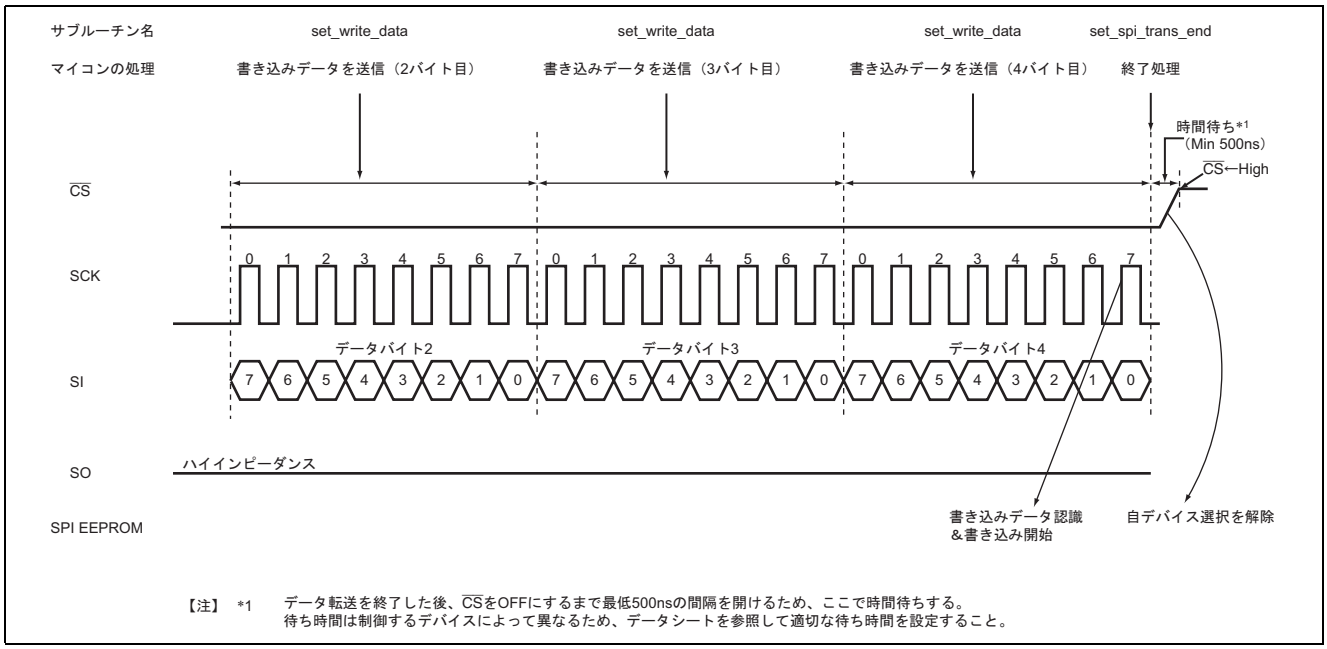
3. SPI EEPROM に連続してデータを書き込みます（Page Write）。
   動作例は、4 バイトの書き込みの場合を示します。
   手順 a）SPI EEPROM の書き込み禁止状態を解除します。3.5　1.手順 a）と同じです。

   手順 b）データを書き込みます。



【注】 *1　CSをONにした後、データ転送を開始するまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

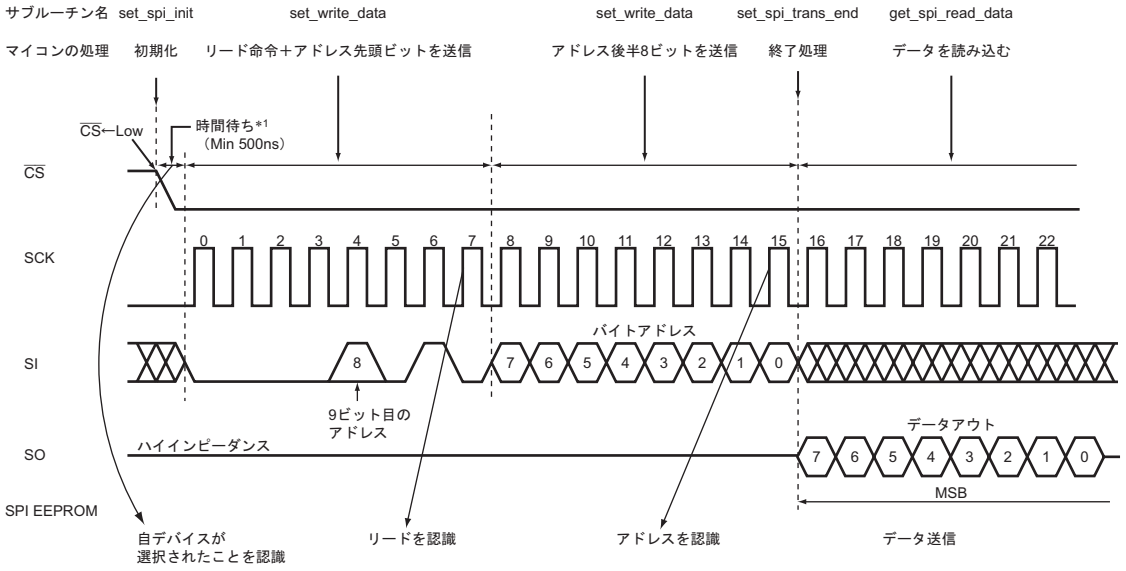
*2　CSをOFFにして再びONにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。



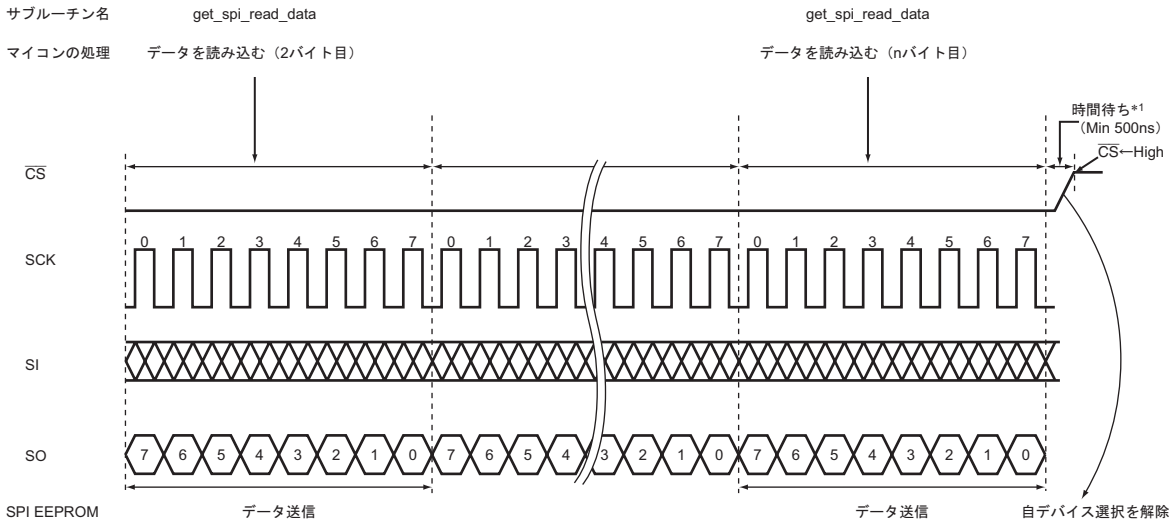
【注】 *1　データ転送を終了した後、CSをOFFにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

   手順 c）SPI EEPROM の書き込み禁止状態に戻します。3.5　1.手順 c）と同じです。
   手順 d）書き込み終了をチェックする。3.5　1.手順 d）と同じです。

4. SPI EEPROM から連続してデータを読み出します（Sequential Read）。



【注】 *1 CSをONにした後、データ転送を開始するまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。



【注】 *1 データ転送を終了した後、CSをOFFにするまで最低500nsの間隔を開けるため、ここで時間待ちする。
待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。

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_2 関連レジスタ

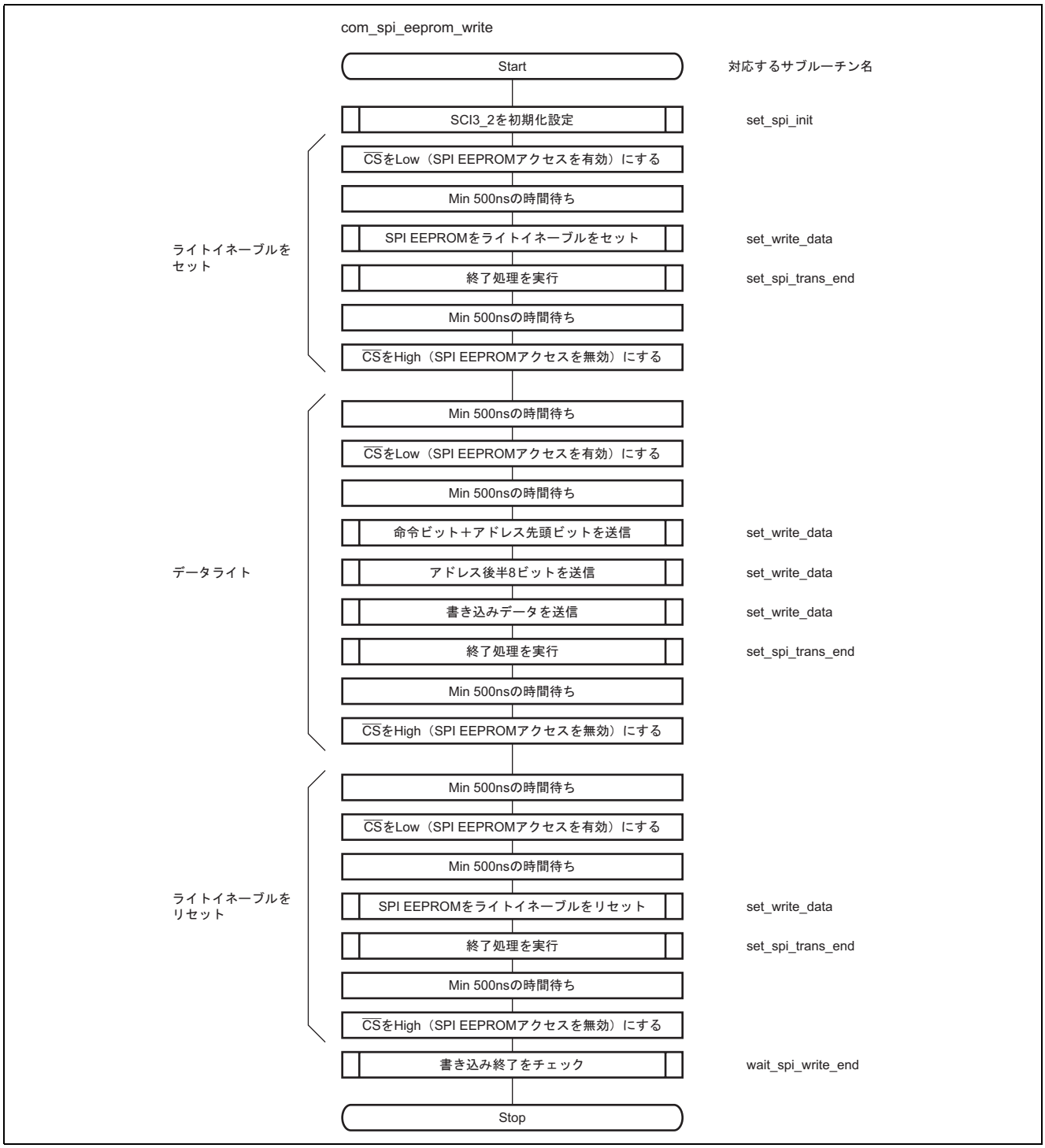
名称	概　　要
レシーブデータレジスタ（RDR）	受信データを格納するための 8 ビットのレジスタ
トランスミットデータレジスタ（TDR）	送信データを格納するための 8 ビットのレジスタ
シリアルモードレジスタ（SMR）	シリアルデータ通信フォーマットと内蔵ボーレートジェネレータのクロックソースを選択するためのレジスタ
シリアルコントロールレジスタ 3（SCR3）	送受信動作と割り込み制御、送受信クロックソースの選択を行うためのレジスタ
シリアルステータスレジスタ（SSR）	SCI3 のステータスフラグと送受信マルチプロセッサビット
ビットレートレジスタ（BRR）	ビットレートを設定する 8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

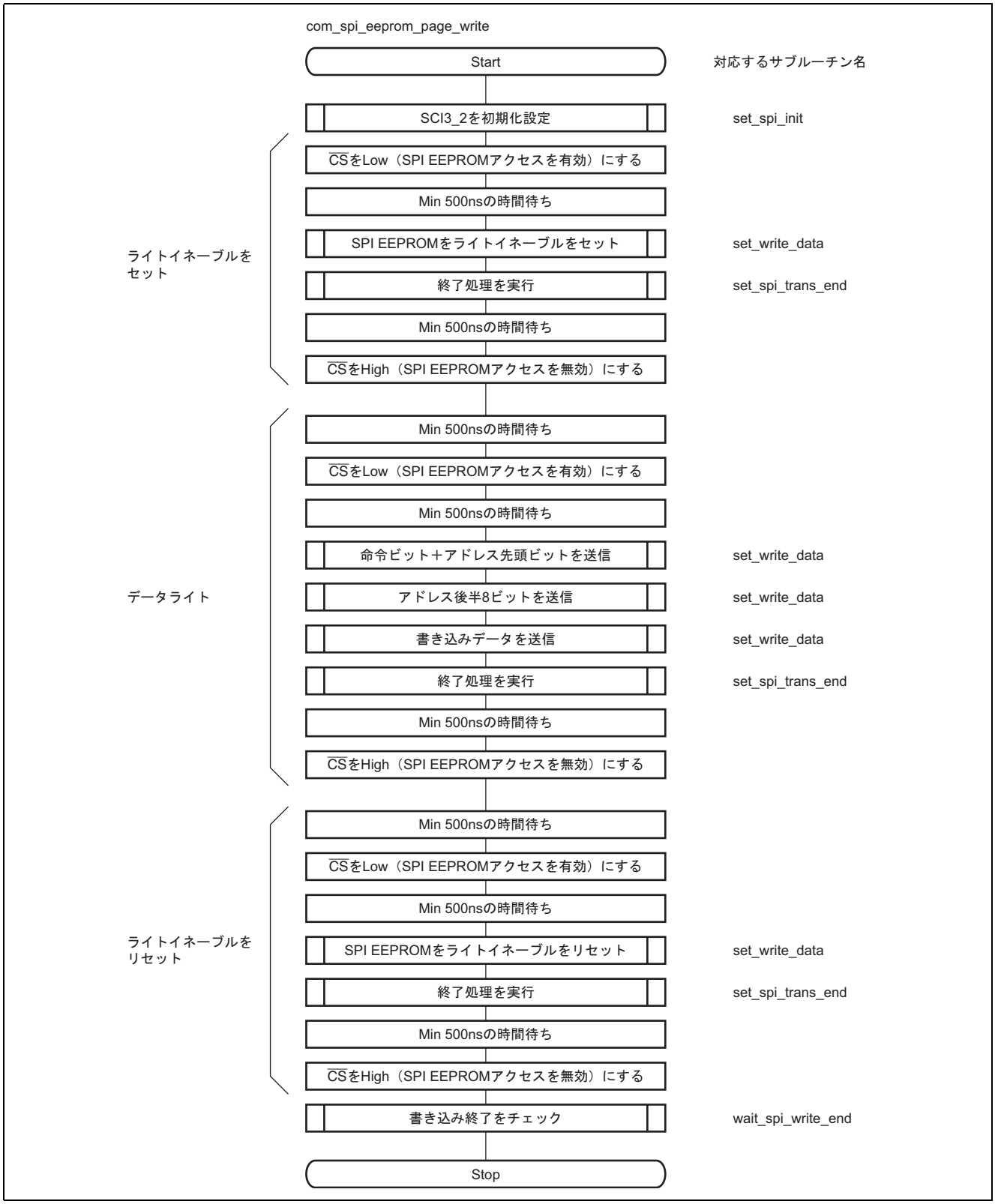
2. タイマ Z 関連レジスタ
タイマ Z は多様な機能を持ちますが、本サンプルプログラムでは、GRA レジスタのコンペアマッチ機能で 10ms ごとの割り込みを発生させるようにしています。

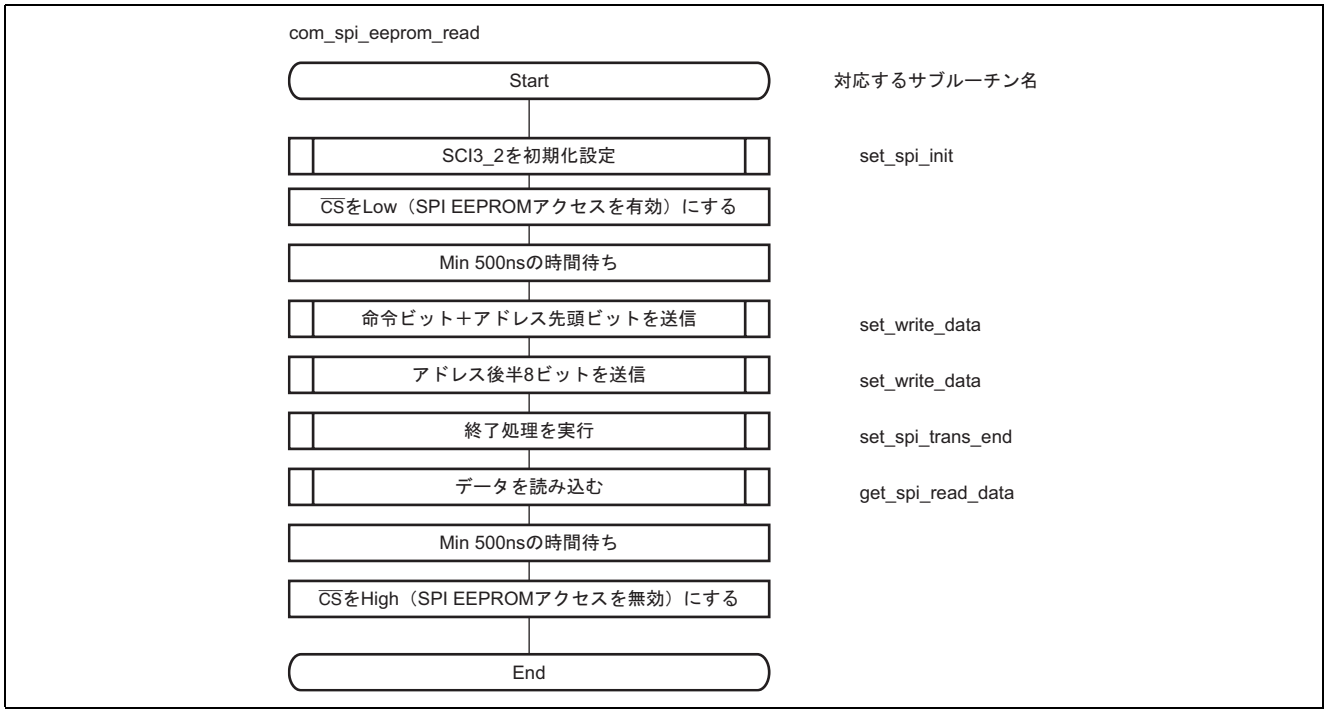
名称	概　　要
タイマスタートレジスタ（TSTR）	TCNT の動作／停止を選択する
タイマモードレジスタ（TMDR）	バッファ動作の設定、同期動作を選択する
タイマ PWM モードレジスタ（TPMR）	端子を PWM モードに設定する。 本サンプルプログラムでは使用しません。
タイマファンクションコントロールレジスタ（TFCR）	各動作モードの設定や出力レベルを選択する。 本サンプルプログラムでは使用しません。
タイマアウトプットマスタイネーブルレジスタ（TOER）	チャネル 0、1 の出力を許可／禁止を設定する。
タイマアウトプットコントロールレジスタ（TOCR）	コンペアマッチが最初に起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。

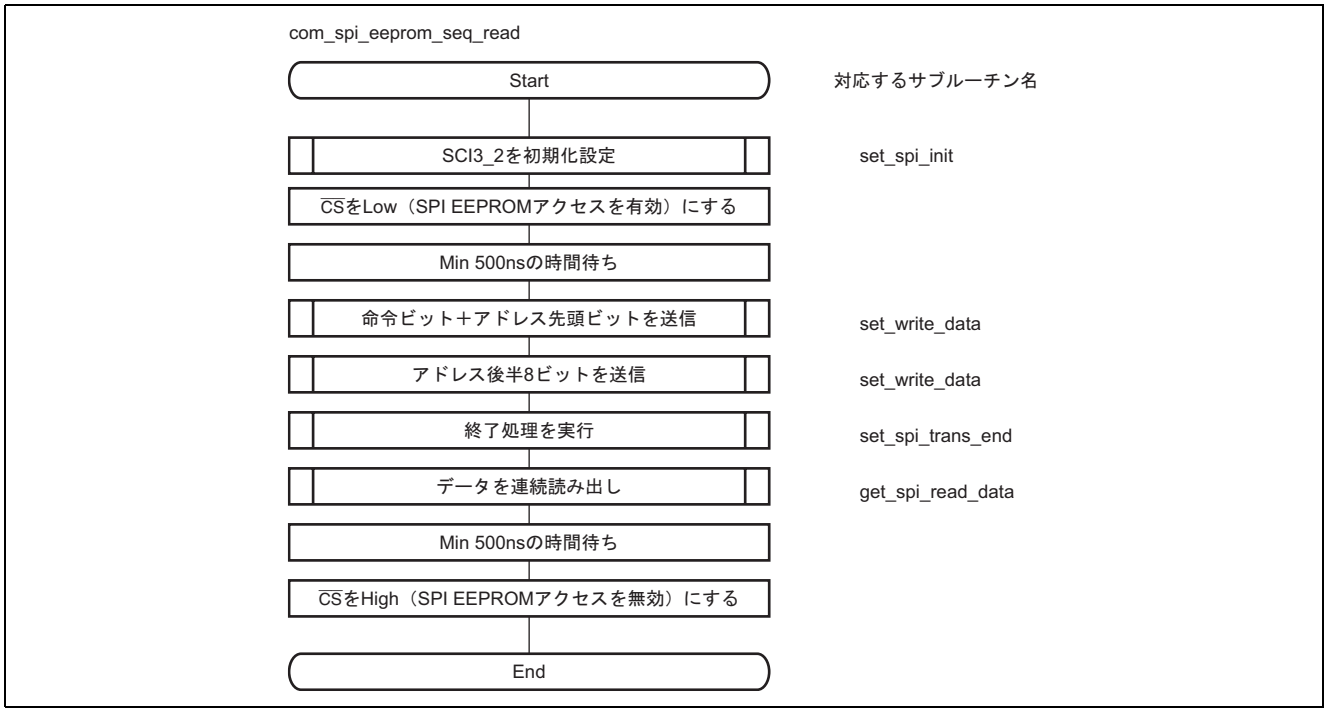
3.7 フローチャート

プログラムの処理フローを以下に示します。



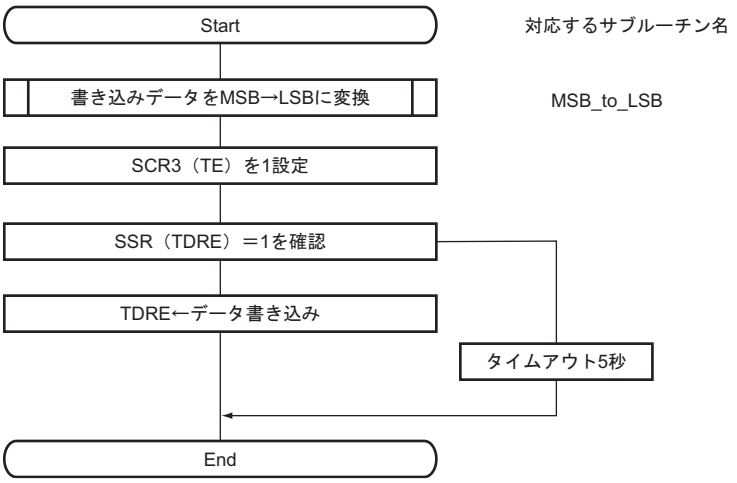

com_spi_eeprom_page_write
Start
対応するサブルーチン名
SCI3_2を初期化設定
set_spi_init
CSをLow（SPI EEPROMアクセスを有効）にする
Min 500nsの時間待ち
SPI EEPROMをライトイネーブルをセット
set_write_data
ライトイネーブルをセット
終了処理を実行
set_spi_trans_end
Min 500nsの時間待ち
CSをHigh（SPI EEPROMアクセスを無効）にする
Min 500nsの時間待ち
CSをLow（SPI EEPROMアクセスを有効）にする
Min 500nsの時間待ち
命令ビット＋アドレス先頭ビットを送信
set_write_data
アドレス後半8ビットを送信
set_write_data
データライト
書き込みデータを送信
set_write_data
終了処理を実行
set_spi_trans_end
Min 500nsの時間待ち
CSをHigh（SPI EEPROMアクセスを無効）にする
Min 500nsの時間待ち
CSをLow（SPI EEPROMアクセスを有効）にする
Min 500nsの時間待ち
SPI EEPROMをライトイネーブルをリセット
set_write_data
ライトイネーブルをリセット
終了処理を実行
set_spi_trans_end
Min 500nsの時間待ち
CSをHigh（SPI EEPROMアクセスを無効）にする
書き込み終了をチェック
wait_spi_write_end
Stop


com_spi_eeprom_read
Start
SCI3_2を初期化設定
CSをLow（SPI EEPROMアクセスを有効）にする
Min 500nsの時間待ち
命令ビット＋アドレス先頭ビットを送信
アドレス後半8ビットを送信
終了処理を実行
データを読み込む
Min 500nsの時間待ち
CSをHigh（SPI EEPROMアクセスを無効）にする
End
対応するサブルーチン名
set_spi_init
set_write_data
set_write_data
set_spi_trans_end
get_spi_read_data

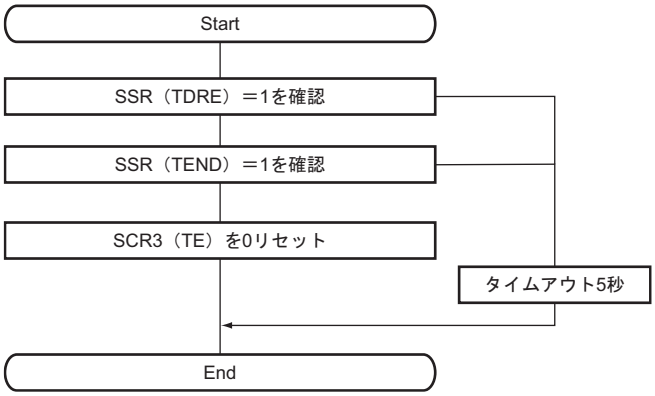

com_spi_eeprom_seq_read
Start
SCI3_2を初期化設定
CSをLow（SPI EEPROMアクセスを有効）にする
Min 500nsの時間待ち
命令ビット＋アドレス先頭ビットを送信
アドレス後半8ビットを送信
終了処理を実行
データを連続読み出し
Min 500nsの時間待ち
CSをHigh（SPI EEPROMアクセスを無効）にする
End
対応するサブルーチン名
set_spi_init
set_write_data
set_write_data
set_spi_trans_end
get_spi_read_data

set_spi_inite
：spiアクセス前の初期化設定を行う

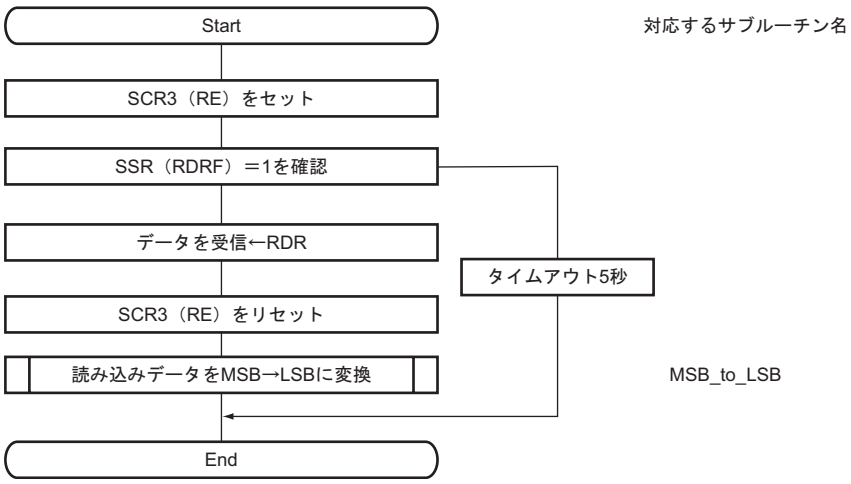
set_write_data（unsigned char write_data）
：データを送信する
write_data：書き込みデータ



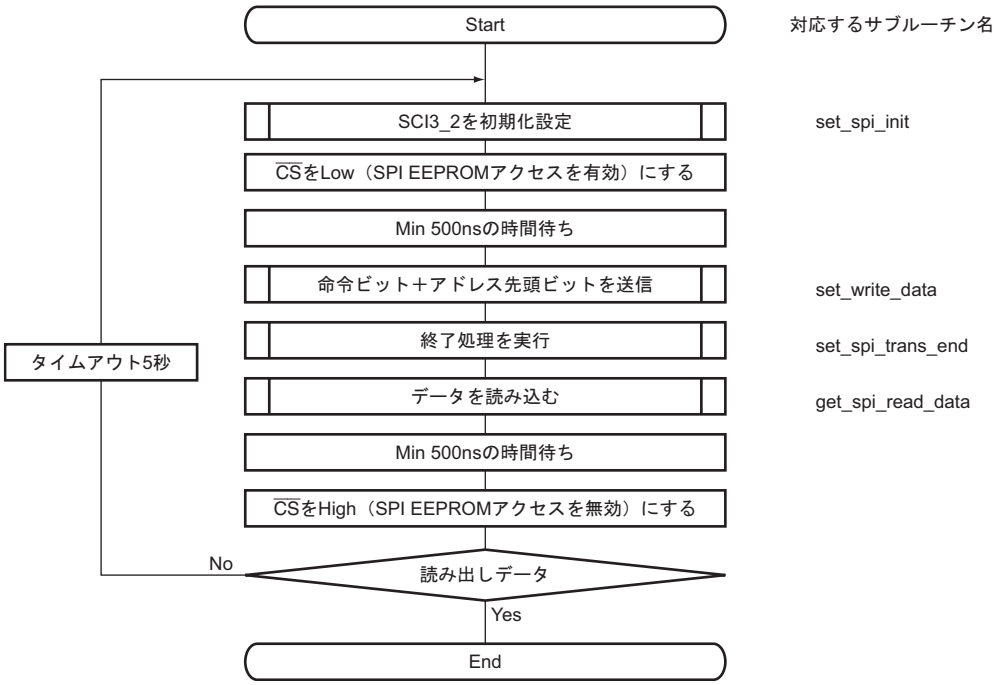
set_spi_trans_end
：データを送信の終了処理を実行する



get_spi_read_data（unsigned char *read_data）
：データを受信する
*read_data：読み出しデータ格納アドレス



wait_spi_write_end
：SPI書き込み終了チェックする
com_spi_eeprom_read



com_delay（int delaytime）
：任意時間のディレイ
delaytime：ディレイ時間（1で約0.5μs）

MSB_to_LSB（unsigned char in_data）
：ビットの並びを逆にする
in_data：MSBデータ

3.8 プログラムリスト

```
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/* 1 . サンプルプログラム 3-A define 定義---------------------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/*****************************************************************************************************************************************************/
/*   SPIEEPROM アクセス用                                                                                                                              */
/*****************************************************************************************************************************************************/
#define   SET_WRITE_MODE      0x02
#define   SET_READ_MODE       0x03
#define   RESET_WRITE_ENABLE  0x04
#define   READ_STATUS         0x05
#define   SET_WRITE_ENABLE    0x06

/*****************************************************************************************************************************************************/
/*   SPIEEPROM アクセスエラーコード(0 以外)                                                                                                            */
/*****************************************************************************************************************************************************/
#define   SPI_TDRE_TOUT       1
#define   SPI_TEND_TOUT       2
#define   SPI_RDRF_TOUT       3
#define   WRITE_TOUT          4

/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/* 2 . サンプルプログラム 3-B プロトタイプ宣言----------------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/*****************************************************************************************************************************************************/
/*****************************************************************************************************************************************************/
/*   SPI BUS アクセス処理                                                                                                                              */
/*****************************************************************************************************************************************************/
/*****************************************************************************************************************************************************/
void com_delay( int delaytime ) ;
void set_spi_init ( ) ;
unsigned char MSB_to_LSB (unsigned char in_data) ;
unsigned int set_write_data (unsigned char write_data);
unsigned int set_spi_trans_end ();
unsigned int wait_spi_write_end () ;
unsigned int get_spi_read_data (unsigned char *read_data);
unsigned int com_spi_eeprom_read ( unsigned int rom_addr , unsigned char *rom_data );
unsigned int com_spi_eeprom_write (unsigned int rom_addr , unsigned char rom_data );
unsigned int com_spi_eeprom_seq_read ( unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data ) ;
unsigned int com_spi_eeprom_page_write ( unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data ) ;
```

```
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* 3 . サンプルプログラム 3-C ソースコード------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */

/* ------------------------------------------------------------------------------------------------------------------------ */
/* 3.1 初期設定処理----------------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
/*   H8 立ち上げ処理の初期設定に下記を追加してください。                                                                    */

    /************************************************************************************************************************/
    /*   PCR1      IO ポート 1 の入出力を規定                                                                               */
    /*        PCR17     = 1  未使用(出力端子として定義)                                                                     */
    /*        PCR16     = 1  未使用(出力端子として定義)                                                                     */
    /*        PCR15     = 1  未使用(出力端子として定義)                                                                     */
    /*        PCR14     = 1  未使用(出力端子として定義)                                                                     */
    /*        PCR12     = 1  未使用(出力端子として定義)                                                                     */
    /*        PCR11     = 1  未使用(出力端子として定義)                                                                     */
    /*        PCR10     = 1  SPIEEPROM の CS として使用                                                                     */
    /************************************************************************************************************************/
    IO.PCR1             = 0xFF ;

    /************************************************************************************************************************/
    /*   PDR1      IO ポート 1 の出力データを設定                                                                           */
    /*        PDR17     = 0  未使用(出力端子として定義)                                                                     */
    /*        PDR16     = 0  未使用(出力端子として定義)                                                                     */
    /*        PDR15     = 0  未使用(出力端子として定義)                                                                     */
    /*        PDR14     = 0  未使用(出力端子として定義)                                                                     */
    /*        PDR12     = 0  未使用(出力端子として定義)                                                                     */
    /*        PDR11     = 0  未使用(出力端子として定義)                                                                     */
    /*        PDR10     = 1  CS = high (SPIEEPRON not active)を設定                                                         */
    /************************************************************************************************************************/
    IO.PDR1.BYTE            = 0x01 ;
        /* ##(program note)################################################################################################ */
        /* ## SPIEEPROMk の CS は、Low Active のため初期設定は High(Not active)を設定する。                              ## */
        /* ## この設定前は、CS のレベルは保証されないが、SPIEEPROM は自身で write protect しているため                   ## */
        /* ## SPIEEPROM の内容が不当に書き込まれることはない                                                              ## */
        /* ################################################################################################################ */
```

```
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */
/* 3.2 SPIEEPROM アクセス処理---------------------------------------------------------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------------------------------------ */

/*****************************************************************************************************************************************************/
/*****************************************************************************************************************************************************/
/*****************************************************************************************************************************************************/
/*                                                                                                                                                   */
/*                                    SPI  EEPROM 制御                                                                                               */
/*                                                                                                                                                   */
/*****************************************************************************************************************************************************/
/*****************************************************************************************************************************************************/
/*****************************************************************************************************************************************************/
/*****************************************************************************************************************************************************/
/*    1.モジュール名称:com_delay                                                                                                                     */
/*    2.機能概要:任意時間のディレイ                                                                                                                  */
/*    3.来歴:REV  作成/改訂日付      作成/改訂者      改訂内容                                                                                       */
/*           000   2002.03.25         上田           新規作成                                                                                       */
/*****************************************************************************************************************************************************/
void com_delay( int delaytime )
{
     register int i,a;

     for(i=0;i<delaytime;i++)
          a++;
}

/*****************************************************************************************************************************************************/
/*    1.モジュール名称:set_spi_init                                                                                                                  */
/*    2.機能概要:spi アクセス前の初期設定を行う                                                                                                      */
/*****************************************************************************************************************************************************/
void set_spi_init( )
{
     /*************************************************************************************************************************************************/
     /*    SCR3_2          SCI3_2 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 SCK3_2 を出力端子とする                                                                                               */
     /*************************************************************************************************************************************************/
     SCI3_2.SCR3.BYTE    = 0x00 ;

     /*************************************************************************************************************************************************/
     /*    SMR_2                SCI3_2 のモード設定                                                                                                   */
     /*         COM        = 1   クロック同期モードで動作                                                                                           */
     /*         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_2.SMR.BYTE     = 0x80 ;
```

```
    /************************************************************************************************************************/
    /*   BRR            100kbps 設定                                                                                        */
    /************************************************************************************************************************/
    SCI3_2.BRR = 0x27;
        /* ##(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                                                        ## */
        /* ################################################################################################################## */

    /************************************************************************************************************************/
    /*   SSR(OER,FER,PER)をリセットする                                                                                      */
    /************************************************************************************************************************/
    SCI3_2.SSR.BIT.OER = 0 ;                                        /* オーバーランエラーリセット                          */
    SCI3_2.SSR.BIT.FER = 0 ;                                        /* フレーミングエラーリセット                          */
    SCI3_2.SSR.BIT.PER = 0 ;                                        /* パリティエラーリセット                              */

}

/************************************************************************************************************************/
/*   1.モジュール名称:MSB_to_LSB                                                                                        */
/*   2.機能概要:bit の並びを逆にする                                                                                    */
/************************************************************************************************************************/
unsigned char MSB_to_LSB (unsigned char in_data)
{
    int i ;
    unsigned char out_data ;

    out_data = 0 ;
    for (i=0; i<8; i++){                                            /* 受信バッファクリア                                  */

        switch (i){
            case 0 :
                out_data = out_data | ((in_data & 0x01) << 7) ;
                break ;
            case 1 :
                out_data = out_data | ((in_data & 0x02) << 5) ;
                break ;
            case 2 :
                out_data = out_data | ((in_data & 0x04) << 3) ;
                break ;
            case 3 :
                out_data = out_data | ((in_data & 0x08) << 1) ;
                break ;
            case 4 :
                out_data = out_data | ((in_data & 0x10) >> 1) ;
                break ;
            case 5 :
                out_data = out_data | ((in_data & 0x20) >> 3) ;
                break ;
            case 6 :
                out_data = out_data | ((in_data & 0x40) >> 5) ;
                break ;
            case 7 :
                out_data = out_data | ((in_data & 0x80) >> 7) ;
                break ;
        }
    }

    return (out_data) ;
}
```

```
/*****************************************************************************************************************************************************/
/*    1.モジュール名称:set_write_data                                                                                                                  */
/*    2.機能概要:データを送信する                                                                                                                      */
/*****************************************************************************************************************************************************/
unsigned int set_write_data (unsigned char write_data)
{
     int ret , Timer_wk , i ;
     unsigned char buf ;

     ret = NORMAL_END ;

     /*****************************************************************************************************************************************************/
     /*    書き込みデータを MSB->LSB に変換する                                                                                                          */
     /*****************************************************************************************************************************************************/
     buf = MSB_to_LSB(write_data) ;
          /* ##(program note)################################################################################################################ */
          /* ## H8 マイコンの SCI インターフェースは LSB のため(bit0 からデータが入出力されるのに対し                                          ## */
          /* ## 本サンプルプログラムの EEPROM は MSB のため(bit7 からデータが入出力される)、ここで送信データの bit の並び替えを行う            ## */
          /* ################################################################################################################################ */

     /*****************************************************************************************************************************************************/
     /*    SCR3(TE)をセット                                                                                                                              */
     /*****************************************************************************************************************************************************/
     SCI3_2.SCR3.BIT.TE =     1    ;                                        /* 送信動作を有効にする                                                  */

     /*****************************************************************************************************************************************************/
     /*    SSR(TDRE) = 1 を確認                                                                                                                          */
     /*****************************************************************************************************************************************************/
     com_timer.wait_100ms_spi = 50 ;
     while(SCI3_2.SSR.BIT.TDRE == 0){                                       /* データ転送可能になるまで待つ                                          */
          Timer_wk = com_timer.wait_100ms_spi ;
          if (Timer_wk == 0){                                               /* 5s 間で Timeout する                                                  */
               ret = SPI_TDRE_TOUT ;                                        /* エラーでも何もしない                                                  */
               goto exit ;
          }
     #ifdef UT
          SCI3_2.SSR.BIT.TDRE = 1 ;
     #endif
     }

     /*****************************************************************************************************************************************************/
     /*    データ書き込み                                                                                                                                */
     /*****************************************************************************************************************************************************/
     SCI3_2.TDR = buf ;                                                     /* データ送信 => この動作で SSR(TDRE)はリセットされる                    */

exit :
     return (ret) ;

}
/*****************************************************************************************************************************************************/
/*    1.モジュール名称:set_spi_trans_end                                                                                                               */
/*    2.機能概要:データを送信の終了処理を実行する                                                                                                      */
/*****************************************************************************************************************************************************/
unsigned int set_spi_trans_end ()
{
     int ret , Timer_wk;

     ret = NORMAL_END ;
```

```
	/*****************************************************************************************************************************************/
	/*   SSR(TDRE) = 1 を確認                                                                                                                */
	/*****************************************************************************************************************************************/
	com_timer.wait_100ms_spi = 50 ;
	while(SCI3_2.SSR.BIT.TDRE == 0){                                    /* データ転送可能になるまで待つ                                    */
		Timer_wk = com_timer.wait_100ms_spi ;
		if (Timer_wk == 0){                                             /* 5s 間で Timeout する                                            */
			ret = SPI_TDRE_TOUT ;                                   /* エラーでも何もしない                                            */
			goto exit ;
		}
	#ifdef UT
		SCI3_2.SSR.BIT.TDRE = 1 ;
	#endif
	}

	/*****************************************************************************************************************************************/
	/*   SSR(TEND) = 1 を確認                                                                                                                */
	/*****************************************************************************************************************************************/
	com_timer.wait_100ms_spi = 50 ;
	while(SCI3_2.SSR.BIT.TEND == 0){                                    /* データ転送終了を待つ                                            */
		Timer_wk = com_timer.wait_100ms_spi ;
		if (Timer_wk == 0){                                             /* 5s 間で Timeout する                                            */
			ret = SPI_TEND_TOUT ;                                   /* エラーでも何もしない                                            */
			goto exit ;
		}
	#ifdef UT
		SCI3_2.SSR.BIT.TEND = 1 ;
	#endif
	}

exit :
	com_delay(10) ;

	/*****************************************************************************************************************************************/
	/*   SCR3(TE)をリセット                                                                                                                  */
	/*****************************************************************************************************************************************/
	SCI3_2.SCR3.BIT.TE =     0    ;                                     /* 送信動作を disable にする                                       */

	return (ret) ;

}
/*****************************************************************************************************************************************/
/*   1.モジュール名称:get_spi_read_data                                                                                                    */
/*   2.機能概要:データを受信する                                                                                                           */
/*****************************************************************************************************************************************/
unsigned int get_spi_read_data (unsigned char *read_data)
{
	int ret , Timer_wk ;
	unsigned char buf ;

	ret = NORMAL_END ;

	/*****************************************************************************************************************************************/
	/*   SCR3(RE)をセット                                                                                                                    */
	/*****************************************************************************************************************************************/
	SCI3_2.SCR3.BIT.RE =     1    ;                                     /* 受信動作を有効にする                                            */
```

```
      /*****************************************************************************************************************************************/
      /*    SSR(RDRF)=1 を確認                                                                                                                 */
      /*****************************************************************************************************************************************/
      com_timer.wait_100ms_spi = 50 ;

      while (SCI3_2.SSR.BIT.RDRF == 0){                                    /* Recive データが届くまで待つ(Max5 秒)          */
           Timer_wk = com_timer.wait_100ms_spi ;
           if (Timer_wk == 0){                                             /* 5s 間で Timeout する                          */
                ret =SPI_RDRF_TOUT ;                                       /* エラーでも何もしない                          */
                goto exit ;
           }
           #ifdef UT
                SCI3_2.SSR.BIT.RDRF = 1 ;
           #endif
      }

exit :

      buf = SCI3_2.RDR ;                                                   /* データ受信 => この動作で SSR(RDRF) はリセットされる  */

      /*****************************************************************************************************************************************/
      /*    SCR3(RE)をリセット                                                                                                                 */
      /*****************************************************************************************************************************************/
      SCI3_2.SCR3.BIT.RE =     0    ;                                      /* 受信動作を disable にする                     */

      /*****************************************************************************************************************************************/
      /* 読み出しデータを MSB->LSB に変換する                                                                                                    */
      /*****************************************************************************************************************************************/
      *read_data = MSB_to_LSB(buf) ;
           /* ##(program note)################################################################################################################ */
           /* ## H8 マイコンの SCI インターフェースは LSB のため(bit0 からデータが入出力されるのに対し                                    ## */
           /* ## 本サンプルプログラムの EEPROM は MSB のため(bit7 からデータが入出力される)、ここで受信データの bit の並び替えを行う      ## */
           /* ################################################################################################################################ */

      return (ret) ;

}

/*****************************************************************************************************************************************/
/*    1.モジュール名称:wait_spi_write_end                                                                                                    */
/*    2.機能概要:SPI 書き込み終了をチェックする                                                                                              */
/*****************************************************************************************************************************************/
unsigned int wait_spi_write_end ()
{

      int ret , i;
       unsigned char      status;
      union {
            unsigned int  d_int ;
            unsigned char      d_byte[2];
      } buf;


      ret = NORMAL_END ;

      com_timer.wait_100ms = 50 ;
      do{
```

```
        /*****************************************************************************************************************************/
        /*    SCI3_2 を初期化する                                                                                                      */
        /*****************************************************************************************************************************/
        set_spi_init( ) ;

        /*****************************************************************************************************************************/
        /*    CS を Low (SPIEEPROM アクセスを有効) にする                                                                               */
        /*****************************************************************************************************************************/
        IO.PDR1.BYTE = 0x00 ;
        com_delay(10) ;
             /* ##(program note)################################################################################################## */
             /* ## CS を ON にした後、データ転送を開始するまで最低 500ns の間隔を開けるためここで時間待ちする。                    ## */
             /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。              ## */
             /* ################################################################################################################## */

        /*****************************************************************************************************************************/
        /*****************************************************************************************************************************/
        /*    データを read する                                                                                                       */
        /*****************************************************************************************************************************/
        /*****************************************************************************************************************************/

        /*****************************************************************************************************************************/
        /*    命令データ (Read Status) を送信する                                                                                      */
        /*****************************************************************************************************************************/
        ret = set_write_data (READ_STATUS) ;
        if (ret !=0) { goto exit ;}

        /*****************************************************************************************************************************/
        /*    終了処理を実行する                                                                                                       */
        /*****************************************************************************************************************************/
        ret = set_spi_trans_end () ;
        if (ret !=0) { goto exit ;}

        /*****************************************************************************************************************************/
        /*    データを読み込む                                                                                                         */
        /*****************************************************************************************************************************/
        ret = get_spi_read_data (&status) ;
        if (ret !=0) { goto exit ;}

        /*****************************************************************************************************************************/
        /*    CS を High (SPIEEPROM アクセスを無効) にする                                                                              */
        /*****************************************************************************************************************************/
        com_delay(10) ;
             /* ##(program note)################################################################################################## */
             /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                    ## */
             /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。              ## */
             /* ################################################################################################################## */
        IO.PDR1.BYTE = 0x01 ;

        #ifdef UT
             status = 0x01 ;
        #endif

        if (com_timer.wait_100ms == 0){                          /* 5s 間で Timeout する                                    */
             ret = WRITE_TOUT;                                   /* 異常終了 (timeout)                                      */
             goto exit ;
        }

    } while ((status & 0x01) == 1) ;                             /* 書き込み中の間                                          */

exit :                                                           /* エラー処理                                              */
```

```
    /*****************************************************************************************************************************************/
    /*    CS を High(SPIEEPROM アクセスを無効)にする                                                                                          */
    /*****************************************************************************************************************************************/
    com_delay(10) ;
        /* ##(program note)################################################################################################################ */
        /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                                  ## */
        /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること                            ## */
        /* ################################################################################################################################ */
    IO.PDR1.BYTE = 0x01 ;

    return (ret) ;

}

/*****************************************************************************************************************************************/
/*    1.モジュール名称:com_spi_eeprom_read                                                                                                */
/*    2.機能概要:SPIEEPROM から 1byte データを読み出す                                                                                    */
/*****************************************************************************************************************************************/
unsigned int com_spi_eeprom_read ( unsigned int rom_addr , unsigned char *rom_data )
{
    int ret ;
    union {
        unsigned int  d_int ;
        unsigned char      d_byte[2];
    } buf;


    ret = NORMAL_END ;

    /*****************************************************************************************************************************************/
    /*    SCI3_2 を初期化する                                                                                                              */
    /*****************************************************************************************************************************************/
    set_spi_init( ) ;

    /*****************************************************************************************************************************************/
    /*    CS を Low(SPIEEPROM アクセスを有効)にする                                                                                        */
    /*****************************************************************************************************************************************/
    IO.PDR1.BYTE = 0x00 ;
    com_delay(10) ;
        /* ##(program note)################################################################################################################ */
        /* ## CS を ON にした後、データ転送を開始するまで最低 500ns の間隔を開けるためここで時間待ちする。                               ## */
        /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                        ## */
        /* ################################################################################################################################ */

    /*****************************************************************************************************************************************/
    /*****************************************************************************************************************************************/
    /*    データを read する                                                                                                               */
    /*****************************************************************************************************************************************/
    /*****************************************************************************************************************************************/

    /*****************************************************************************************************************************************/
    /*    命令データ+アドレス先頭 bit を送信する                                                                                          */
    /*****************************************************************************************************************************************/
    buf.d_int = rom_addr ;
    buf.d_byte[0] = (buf.d_byte[0] && 0x01) << 3 ;
    buf.d_byte[0] |= SET_READ_MODE ;

    ret = set_write_data (buf.d_byte[0]) ;
    if (ret !=0) { goto exit ;}
```

```
	/*****************************************************************************************************************************************************/
	/*	アドレス後半 8bit を送信する                                                                                                                         */
	/*****************************************************************************************************************************************************/
	ret = set_write_data (buf.d_byte[1]) ;
	if (ret !=0) { goto exit ;}

	/*****************************************************************************************************************************************************/
	/*	終了処理を実行する                                                                                                                               */
	/*****************************************************************************************************************************************************/
	ret = set_spi_trans_end () ;
	if (ret !=0) { goto exit ;}

	/*****************************************************************************************************************************************************/
	/*	データを読み込む                                                                                                                                 */
	/*****************************************************************************************************************************************************/
	ret = get_spi_read_data (&buf.d_byte[0]) ;
	if (ret !=0) { goto exit ;}


	*rom_data = buf.d_byte[0] ;

exit :
	/*****************************************************************************************************************************************************/
	/*	CS を High(SPIEEPROM アクセスを無効)にする                                                                                                       */
	/*****************************************************************************************************************************************************/
	com_delay(10) ;
		/* ##(program note)################################################################################################################ */
		/* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                                          ## */
		/* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                                    ## */
		/* ################################################################################################################################ */

	IO.PDR1.BYTE = 0x01 ;

	return (ret);

}
/*****************************************************************************************************************************************************/
/*	1.モジュール名称:com_spi_eeprom_seq_read                                                                                                         */
/*	2.機能概要:SPIEEPROM から 1byte データを読み出す                                                                                                 */
/*****************************************************************************************************************************************************/
unsigned int com_spi_eeprom_seq_read ( unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data )
{
	int ret , i;
	union {
		unsigned int  d_int ;
		unsigned char	d_byte[2];
	} buf;


	ret = NORMAL_END ;

	/*****************************************************************************************************************************************************/
	/*	SCI3_2 を初期化する                                                                                                                              */
	/*****************************************************************************************************************************************************/
	set_spi_init( ) ;
```

```
    /*****************************************************************************************************************************************/
    /*    CS を Low(SPIEEPROM アクセスを有効)にする                                                                                          */
    /*****************************************************************************************************************************************/
    IO.PDR1.BYTE = 0x00 ;
    com_delay(10) ;
        /* ##(program note)################################################################################################################ */
        /* ## CS を ON にした後、データ転送を開始するまで最低 500ns の間隔を開けるためここで時間待ちする。                                ## */
        /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
        /* ################################################################################################################################ */

    /*****************************************************************************************************************************************/
    /*****************************************************************************************************************************************/
    /*    データを read する                                                                                                                 */
    /*****************************************************************************************************************************************/
    /*****************************************************************************************************************************************/

    /*****************************************************************************************************************************************/
    /*    命令データ+アドレス先頭 bit を送信する                                                                                            */
    /*****************************************************************************************************************************************/
    buf.d_int = rom_addr ;
    buf.d_byte[0] = (buf.d_byte[0] && 0x01) << 3 ;
    buf.d_byte[0] |= SET_READ_MODE ;

    ret = set_write_data (buf.d_byte[0]) ;
    if (ret !=0) { goto exit ;}

    /*****************************************************************************************************************************************/
    /*    アドレス後半 8bit を送信する                                                                                                      */
    /*****************************************************************************************************************************************/
    ret = set_write_data (buf.d_byte[1]) ;
    if (ret !=0) { goto exit ;}

    /*****************************************************************************************************************************************/
    /*    終了処理を実行する                                                                                                                */
    /*****************************************************************************************************************************************/
    ret = set_spi_trans_end () ;
    if (ret !=0) { goto exit ;}


    /*****************************************************************************************************************************************/
    /*    データを連続読み出しする                                                                                                          */
    /*****************************************************************************************************************************************/
    for (i=0; i< (rom_length) ; i++){
        ret = get_spi_read_data (&buf.d_byte[0]) ;
        if (ret !=0) { goto exit ;}

        *rom_data = buf.d_byte[0] ;
        *rom_data ++ ;
    }

exit :
```

```
      /*****************************************************************************************************************************************/
      /*    CS を High(SPIEEPROM アクセスを無効)にする                                                                                          */
      /*****************************************************************************************************************************************/
      com_delay(10) ;
           /* ##(program note)################################################################################################################ */
           /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                                  ## */
           /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
           /* ################################################################################################################################ */

      IO.PDR1.BYTE = 0x01 ;

      return (ret);

}
/*****************************************************************************************************************************************/
/*    1.モジュール名称:com_spi_eeprom_write                                                                                                */
/*    2.機能概要:SPIEEPROM に 1byte データを書きこむ                                                                                       */
/*****************************************************************************************************************************************/
unsigned int com_spi_eeprom_write (unsigned int rom_addr , unsigned char rom_data )
{
      int ret ;
      union {
            unsigned int  d_int ;
            unsigned char      d_byte[2];
      } buf;


      ret = NORMAL_END ;

      /*****************************************************************************************************************************************/
      /*    SCI3_2 を初期化する                                                                                                               */
      /*****************************************************************************************************************************************/
      set_spi_init( ) ;

      /*****************************************************************************************************************************************/
      /*****************************************************************************************************************************************/
      /*    SPI EEPROM の write enable を解除する                                                                                             */
      /*****************************************************************************************************************************************/
      /*****************************************************************************************************************************************/
      /*****************************************************************************************************************************************/
      /*    CS を Low(SPIEEPROM アクセスを有効)にする                                                                                         */
      /*****************************************************************************************************************************************/
      IO.PDR1.BYTE = 0x00 ;
      com_delay(10) ;
           /* ##(program note)################################################################################################################ */
           /* ## CS を ON にした後、データ転送を開始するまで最低 500ns の間隔を開けるためここで時間待ちする。                              ## */
           /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                        ## */
           /* ################################################################################################################################ */

      /*****************************************************************************************************************************************/
      /*    SPI EEPROM の write enable にする                                                                                                 */
      /*****************************************************************************************************************************************/
      ret = set_write_data (SET_WRITE_ENABLE) ;
      if (ret !=0) { goto exit ;}
```

```
/*****************************************************************************************************************************************/
/*    終了処理を実行する                                                                                                                 */
/*****************************************************************************************************************************************/
ret = set_spi_trans_end () ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************************/
/*    CS を High(SPIEEPROM アクセスを無効)にする                                                                                          */
/*****************************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                                  ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                           ## */
     /* ################################################################################################################################ */

IO.PDR1.BYTE = 0x01 ;


/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*    データを write する                                                                                                                */
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*    CS を Low(SPIEEPROM アクセスを有効)にする                                                                                          */
/*****************************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## CS を OFF にして再び ON にするまで最低 500ns の間隔を開けるためここで時間待ちする。                                        ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                         ## */
     /* ################################################################################################################################ */

IO.PDR1.BYTE = 0x00 ;
     /* ##(program note)################################################################################################################ */
     /* ## CS を ON にした後、データ転送を開始するまで最低 500ns の間隔を開けるためここで時間待ちする。                               ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                         ## */
     /* ################################################################################################################################ */
com_delay(10) ;

/*****************************************************************************************************************************************/
/*    命令データ+アドレス先頭 bit を送信する                                                                                             */
/*****************************************************************************************************************************************/
buf.d_int = rom_addr ;
buf.d_byte[0] = (buf.d_byte[0] && 0x01) << 3 ;
buf.d_byte[0] |= SET_WRITE_MODE ;

ret = set_write_data (buf.d_byte[0]) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************************/
/*    アドレス後半 8bit を送信する                                                                                                       */
/*****************************************************************************************************************************************/
ret = set_write_data (buf.d_byte[1]) ;
if (ret !=0) { goto exit ;}
```

```
/*****************************************************************************************************************************************/
/*    書き込みデータを送信する                                                                                                           */
/*****************************************************************************************************************************************/
ret = set_write_data (rom_data) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************************/
/*    終了処理を実行する                                                                                                                 */
/*****************************************************************************************************************************************/
ret = set_spi_trans_end () ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************************/
/*    CS を High(SPIEEPROM アクセスを無効)にする                                                                                          */
/*****************************************************************************************************************************************/
com_delay(10) ;
IO.PDR1.BYTE = 0x01 ;

/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*    SPI EEPROM の write enable を解除する                                                                                              */
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*    CS を Low(SPIEEPROM アクセスを有効)にする                                                                                          */
/*****************************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## CS を OFF にして再び ON にするまで最低 500ns の間隔を開けるためここで時間待ちする。                                        ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                        ## */
     /* ################################################################################################################################ */

IO.PDR1.BYTE = 0x00 ;

com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## CS を ON にした後、データ転送を開始するまで最低 500ns の間隔を開けるためここで時間待ちする。                              ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                        ## */
     /* ################################################################################################################################ */

/*****************************************************************************************************************************************/
/*    SPI EEPROM の write enable を解除する                                                                                              */
/*****************************************************************************************************************************************/
ret = set_write_data (RESET_WRITE_ENABLE) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************************/
/*    終了処理を実行する                                                                                                                 */
/*****************************************************************************************************************************************/
ret = set_spi_trans_end () ;
if (ret !=0) { goto exit ;}
```

```
      /*****************************************************************************************************************************************/
      /*    CS を High(SPIEEPROM アクセスを無効)にする                                                                                          */
      /*****************************************************************************************************************************************/
      com_delay(10) ;
            /* ##(program note)################################################################################################################ */
            /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                                  ## */
            /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
            /* ################################################################################################################################ */

      IO.PDR1.BYTE = 0x01 ;

      /*****************************************************************************************************************************************/
      /*    書き込み終了をチェックする                                                                                                         */
      /*****************************************************************************************************************************************/
      ret = wait_spi_write_end () ;
            if (ret !=0) { goto exit ;}
            /* ##(program note)################################################################################################################ */
            /* ## SPIEEPROM は CS=high で書き込みを開始する。書き込み動作には時間がかかるため                                              ## */
            /* ## SPIEEPROM 内の status レジスタをチェックすることで書き込み終了をチェックする                                             ## */
            /* ################################################################################################################################ */

      return (ret);

exit :                                                                  /* エラー処理                                                      */
      /*****************************************************************************************************************************************/
      /*    CS を High(SPIEEPROM アクセスを無効)にする                                                                                          */
      /*****************************************************************************************************************************************/
      com_delay(10) ;
            /* ##(program note)################################################################################################################ */
            /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                                  ## */
            /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
            /* ################################################################################################################################ */

      IO.PDR1.BYTE = 0x01 ;

      return (ret);

}

/*****************************************************************************************************************************************/
/*    1.モジュール名称:com_spi_eeprom_page_write                                                                                             */
/*    2.機能概要:SPIEEPROM に 4byte データを書きこむ                                                                                         */
/*****************************************************************************************************************************************/
unsigned int com_spi_eeprom_page_write ( unsigned int rom_addr , unsigned int rom_length , unsigned char *rom_data )
{
      int ret , i ;
      union {
            unsigned int   d_int ;
            unsigned char      d_byte[2];
      } buf;

      union {
            unsigned long      d_long ;
            unsigned char      d_byte[4];
      } write_data;


      ret = NORMAL_END ;
```

```
/*****************************************************************************************************************************************/
/*    SCI3_2 を初期化する                                                                                                                */
/*****************************************************************************************************************************************/
set_spi_init( ) ;

/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*    SPI EEPROM の write enable を解除する                                                                                              */
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*    CS を Low(SPIEEPROM アクセスを有効)にする                                                                                          */
/*****************************************************************************************************************************************/
IO.PDR1.BYTE = 0x00 ;
com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## CS を ON にした後、データ転送を開始するまで最低 500ns の間隔を開けるためここで時間待ちする。                                ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
     /* ################################################################################################################################ */


/*****************************************************************************************************************************************/
/*    SPI EEPROM の write enable を解除する                                                                                              */
/*****************************************************************************************************************************************/
ret = set_write_data (SET_WRITE_ENABLE) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************************/
/*    終了処理を実行する                                                                                                                 */
/*****************************************************************************************************************************************/
ret = set_spi_trans_end () ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************************/
/*    CS を High(SPIEEPROM アクセスを無効)にする                                                                                          */
/*****************************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                                ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
     /* ################################################################################################################################ */

IO.PDR1.BYTE = 0x01 ;


/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*    データを write する                                                                                                                */
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*    CS を Low(SPIEEPROM アクセスを有効)にする                                                                                          */
/*****************************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## CS を OFF にして再び ON にするまで最低 500ns の間隔を開けるためここで時間待ちする。                                        ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
     /* ################################################################################################################################ */
```

```
IO.PDR1.BYTE = 0x00 ;
com_delay(10) ;
     /* ##(program note)################################################################################################ */
     /* ## CS を ON にした後、データ転送を開始するまで最低 500ns の間隔を開けるためここで時間待ちする。                    ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。            ## */
     /* ################################################################################################################ */


/*****************************************************************************************************************************/
/*    命令データ+アドレス先頭 bit を送信する                                                                                 */
/*****************************************************************************************************************************/
buf.d_int = rom_addr ;
buf.d_byte[0] = (buf.d_byte[0] && 0x01) << 3 ;
buf.d_byte[0] |= SET_WRITE_MODE ;

ret = set_write_data (buf.d_byte[0]) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*    アドレス後半 8bit を送信する                                                                                           */
/*****************************************************************************************************************************/
ret = set_write_data (buf.d_byte[1]) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*    書き込みデータを送信する                                                                                               */
/*****************************************************************************************************************************/
for (i=0; i< rom_length ; i++){
     buf.d_byte[0] = *rom_data  ;
     ret = set_write_data (buf.d_byte[0]) ;
          if (ret !=0) { goto exit ;}
     *rom_data ++ ;
}

/*****************************************************************************************************************************/
/*    終了処理を実行する                                                                                                     */
/*****************************************************************************************************************************/
ret = set_spi_trans_end () ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************/
/*    CS を High(SPIEEPROM アクセスを無効)にする                                                                              */
/*****************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################ */
     /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                    ## */
     /* ## ちする。待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。    ## */
     /* ################################################################################################################ */

IO.PDR1.BYTE = 0x01 ;
```

```
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*   SPI EEPROM の write enable を設定する                                                                                               */
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*****************************************************************************************************************************************/
/*   CS を Low(SPIEEPROM アクセスを有効)にする                                                                                           */
/*****************************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## CS を OFF にして再び ON にするまで最低 500ns の間隔を開けるためここで時間待ちする。                                          ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
     /* ################################################################################################################################ */
IO.PDR1.BYTE = 0x00 ;
com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## CS を ON にした後、データ転送を開始するまで最低 500ns の間隔を開けるためここで時間待ちする。                               ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
     /* ################################################################################################################################ */


/*****************************************************************************************************************************************/
/*   SPI EEPROM の write enable を解除する                                                                                               */
/*****************************************************************************************************************************************/
ret = set_write_data (RESET_WRITE_ENABLE) ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************************/
/*   終了処理を実行する                                                                                                                  */
/*****************************************************************************************************************************************/
ret = set_spi_trans_end () ;
if (ret !=0) { goto exit ;}

/*****************************************************************************************************************************************/
/*   CS を High(SPIEEPROM アクセスを無効)にする                                                                                          */
/*****************************************************************************************************************************************/
com_delay(10) ;
     /* ##(program note)################################################################################################################ */
     /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                                ## */
     /* ## 待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                          ## */
     /* ################################################################################################################################ */

IO.PDR1.BYTE = 0x01 ;

/*****************************************************************************************************************************************/
/*   書き込み終了をチェックする                                                                                                          */
/*****************************************************************************************************************************************/
ret = wait_spi_write_end () ;
     if (ret !=0) { goto exit ;}
     /* ##(program note)################################################################################################################ */
     /* ## SPIEEPROM は CS=high で書き込みを開始する。書き込み動作には時間がかかるため                                               ## */
     /* ## SPIEEPROM 内の status レジスタをチェックすることで書き込み終了をチェックする                                              ## */
     /* ################################################################################################################################ */

     return (ret);

exit :                                                    /* エラー処理                                                                */
```

```
    /*****************************************************************************************************************************************/
    /*   CS を High(SPIEEPROM アクセスを無効)にする                                                                                          */
    /*****************************************************************************************************************************************/
    com_delay(10) ;
        /* ##(program note)################################################################################################################ */
        /* ## データ転送を終了した後、CS を OFF するまで最低 500ns の間隔を開けるためここで時間待ちする。                                  ## */
        /* ## ちする。待ち時間は制御するデバイスによって異なるため、データシートを参照して適切な待ち時間を設定すること。                  ## */
        /* ################################################################################################################################ */

    IO.PDR1.BYTE = 0x01 ;

    return (ret);

}
```

```
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* 4 . サンプルプログラム 3-D TimerZ 処理---------------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------------------------------------------ */

/* ------------------------------------------------------------------------------------------------------------------------ */
/* 4.1 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                                                                                 */

/* ------------------------------------------------------------------------------------------------------------------------ */
/* 4.2 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;

/* ------------------------------------------------------------------------------------------------------------------------ */
/* 4.3 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 RA はアウトプットコンペアレジスタ   */
    TZ0.TIER.BYTE  = 0x01 ;
                                                               /* IMIEA = 1 MFA 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                                   */


/* ------------------------------------------------------------------------------------------------------------------------------------- */
/* 4.4 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）
- HN58X2464FPIAG　APPLICATION NOTES（Xicor, Inc.）

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。