

# SH7280グループ

RJJ06B1043-0100

Rev.1.00

## ユーザプログラムモード使用例

2010.09.10

### 要旨

本アプリケーションノートは、SH7286 ユーザプログラムモードでのフラッシュメモリ書き換えプログラム動作例について説明しています。フラッシュメモリに書き込むデータは、SH7286 に接続した外部デバイスが保持しており、シリアルコミュニケーションインタフェースを使用して通信を行います。

本アプリケーションノートで紹介するフラッシュメモリ書き換えプログラムは、SH7286 のユーザマット上にあるものとしします。また、フラッシュメモリの書き換え処理部に関しては、ルネサス エレクトロニクスが提供している SH-2, SH-2A 用シンプルフラッシュ API (以下、標準 API と表記します) を使用しています。

### 動作確認デバイス

SH7286

### 目次

1. はじめに.....	2
2. 概要および使用機能説明.....	4
3. 参考プログラムの外部仕様.....	11
4. 参考プログラムの内部仕様.....	18
5. 参考プログラムリスト.....	24
6. 参考ドキュメント.....	34

## 1. はじめに

### 1.1 仕様

本応用例では、ユーザプログラムモードを使用して、フラッシュメモリへの書き込み/消去処理、および読み出し処理を行います。ユーザプログラムモードでは、任意のインターフェースを使用して、これらの処理を制御することができます。本応用例では、ホスト PC とのシリアル通信を使用してこれらの処理を制御します。

ユーザアプリケーション実行中に、ホスト PC からユーザ制御コマンドであるフラッシュ書き込み/消去処理コマンドを受信すると、フラッシュへの書き込み/消去処理を実行し、フラッシュ読み出し処理コマンドを受信すると、フラッシュ読み出し処理を実行します。

図 1 に本応用例のシステム構成を示します。

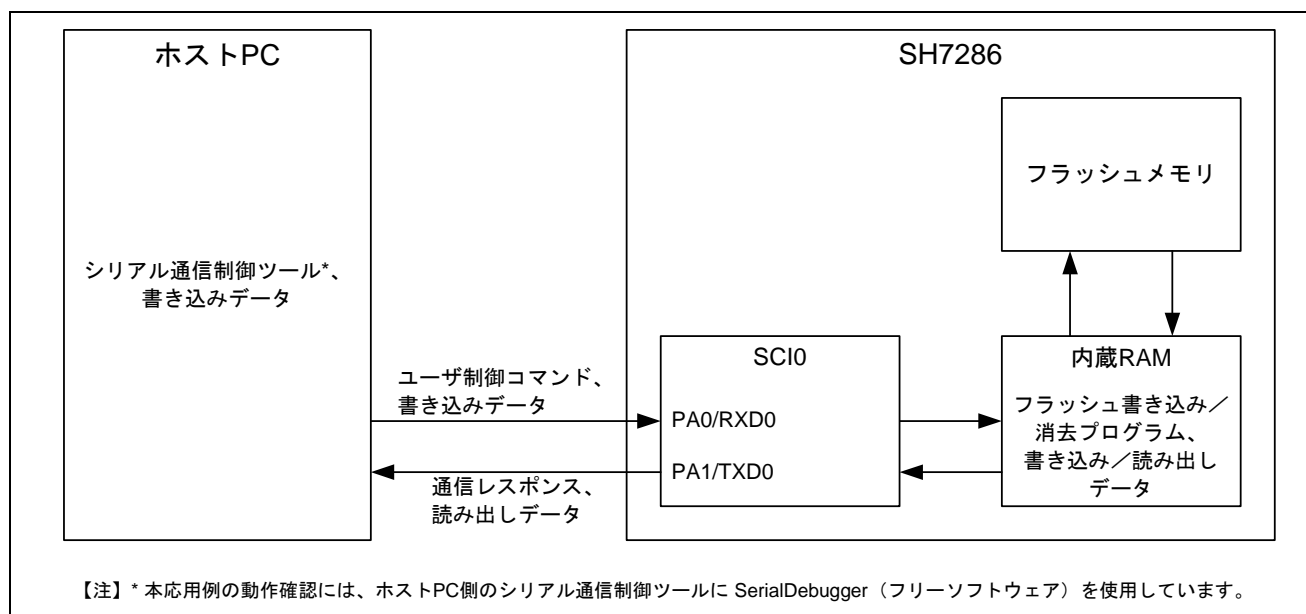


図1 本応用例のシステム構成

## 1.2 使用機能

- シリアルコミュニケーションインタフェース (SCI)
- フラッシュメモリ (ROM)

## 1.3 適用条件

マイコン	SH7286 (フラッシュ 1 M バイト品)
動作周波数* <sup>1</sup>	内部クロック : 25 MHz バスクロック : 25 MHz 周辺クロック : 25 MHz
統合開発環境* <sup>2</sup>	ルネサス エレクトロニクス製 High-performance Embedded Workshop Ver.4.04.01
C コンパイラ	ルネサス エレクトロニクス製 SuperH RISC engine ファミリ C/C++コンパイラパッケージ Ver.9.01 Release 01
コンパイルオプション	High-performance Embedded Workshop でのデフォルト設定 (-cpu=sh2a -debug -gbr=auto -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1)

【注】 \*<sup>1</sup> フラッシュ書き込み/消去処理用の内蔵プログラムをダウンロードする際、周波数制御レジスタ (FRQCR) による内部クロック (I $\phi$ )、バスクロック (B $\phi$ )、および周辺クロック (P $\phi$ ) の分周率の設定はすべて 1/4 : 1/4 : 1/4 としてください。

\*<sup>2</sup> E10A-USB エミュレータは、ブートモード、ユーザブートモード、およびユーザプログラムモードをサポートしていません。そのため、E10A-USB エミュレータを使用して、フラッシュメモリ書き換えプログラムをデバッグすることはできません。

## 1.4 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。合わせて参照してください。

- SH ファミリ SH-2, SH-2A 用シンプルフラッシュ API

## 2. 概要および使用機能説明

本応用例では、SH7286 と外部デバイスの接続にシリアルコミュニケーションインタフェース (SCI) を使用しています。

### 2.1 使用機能説明

#### 2.1.1 シリアルコミュニケーションインタフェース (SCI) 機能説明

SCI は、調歩同期式とクロック同期式の 2 方式のシリアル通信が可能なモジュールです。また、全二重通信が可能で、送信部および受信部ともにダブルバッファ構造となっていますので、シリアルデータの連続送信、連続受信が実現できます。

本応用例では、SH7286 と外部デバイス間でのハンドシェイク通信及び、フラッシュメモリの書き換えデータ通信に SCI を使用しています。

図 2 に SCI のブロック図を示します。

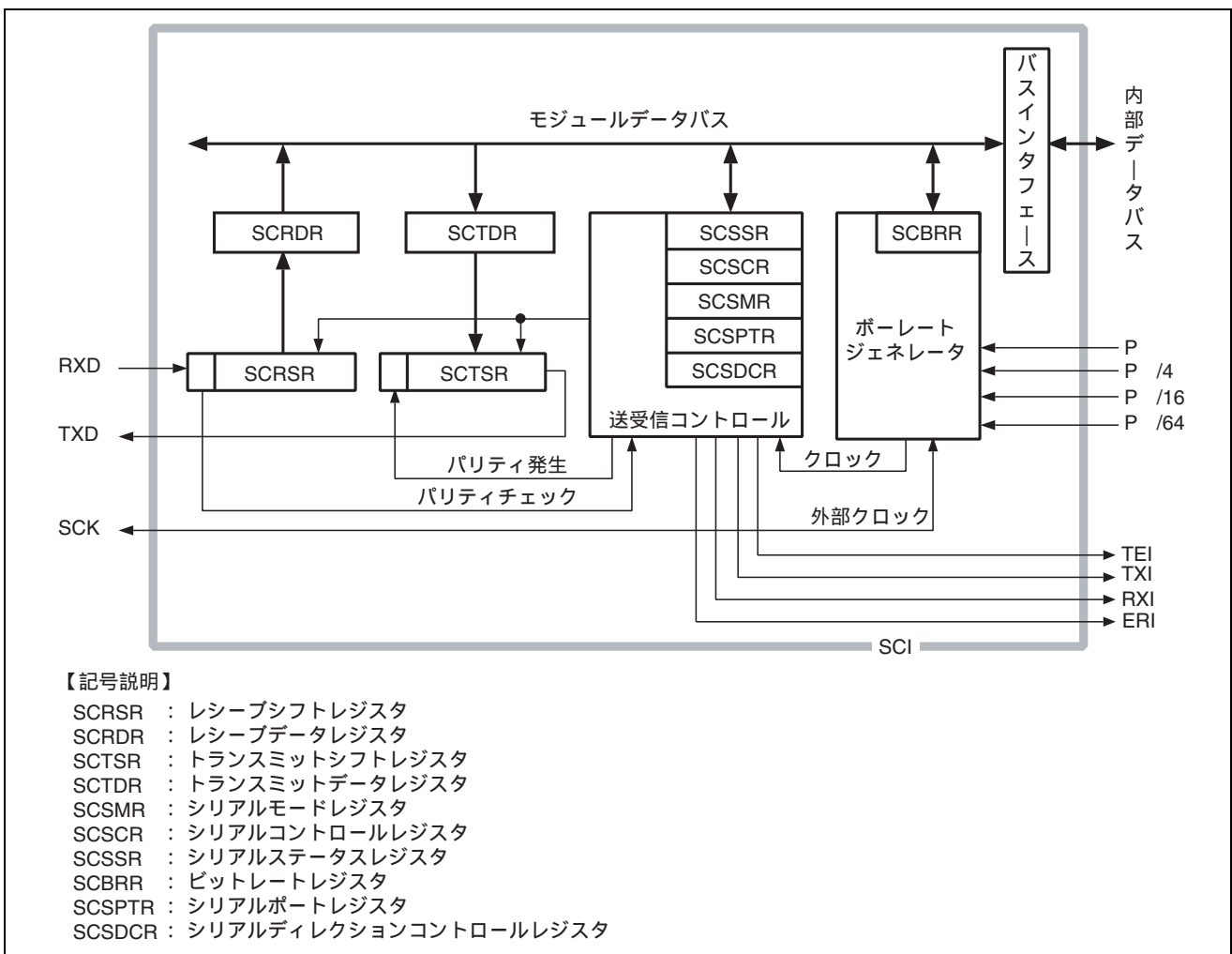


図2 SCI のブロック図

## 2.1.2 フラッシュメモリ (ROM)

SH7280グループは、あらかじめマイコンに内蔵されているプログラムを使用して、フラッシュメモリの書き込み/消去を行います。

図3にフラッシュメモリのブロック図を示します。

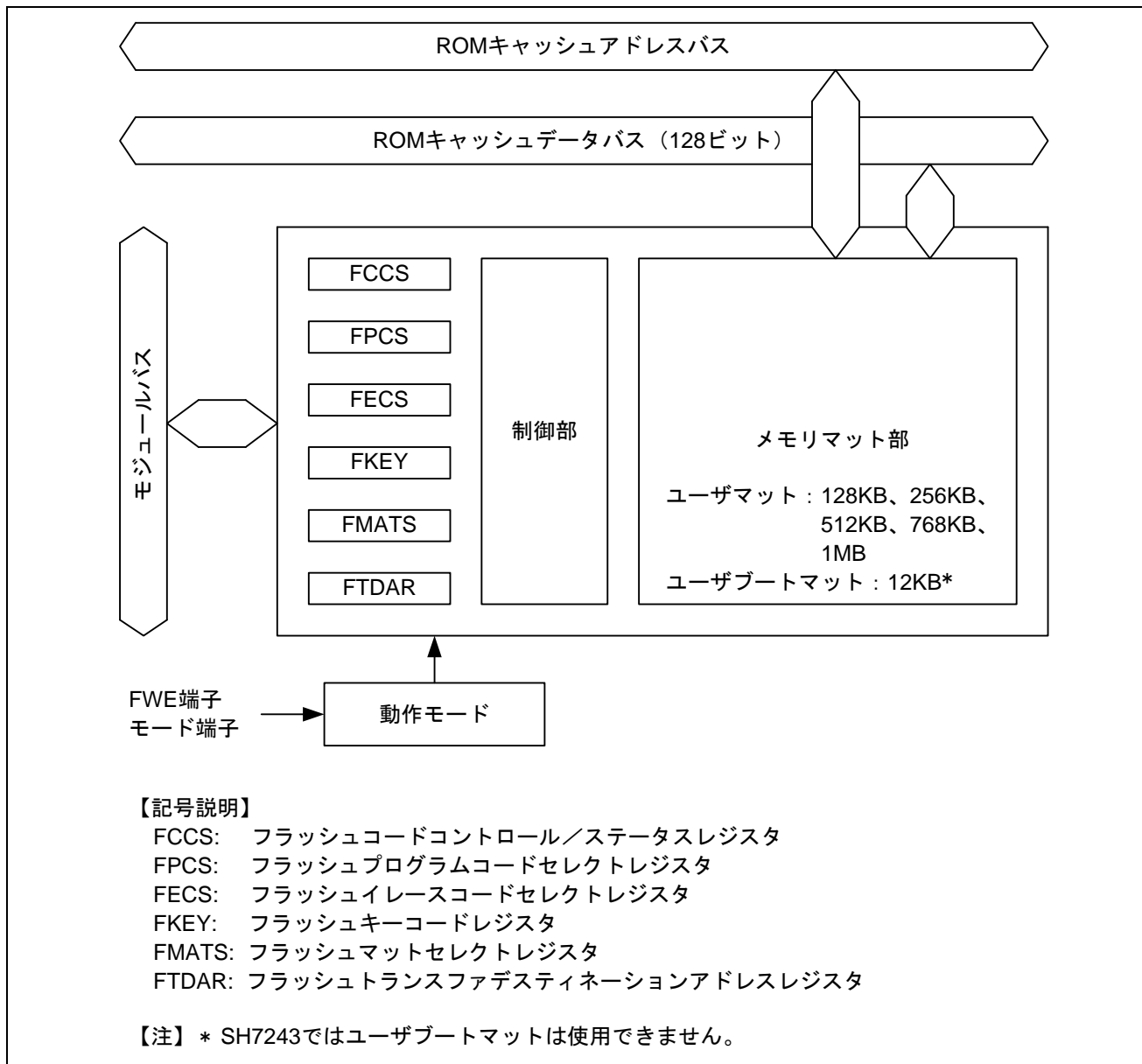


図3 フラッシュメモリのブロック図

## 2.2 フラッシュ書き込み／消去動作説明

SH7280 グループでは、あらかじめマイコンに内蔵されているプログラムを使用して、フラッシュメモリの書き込み／消去を行います。ここでは、フラッシュメモリの書き換え手順を説明しています。より詳細な内容は、SH7280 グループ ハードウェアマニュアルを参照してください。また、本応用例では、フラッシュメモリの書き込み／消去処理に標準 API を使用しています。標準 API の詳細は、関連アプリケーションノートを参照してください。

### 2.2.1 フラッシュ書き込み／消去準備

マイコンに内蔵されているプログラムを使用するためには、そのプログラムを内蔵 RAM 上にダウンロードする必要があります。ダウンロード完了後、書き込みアドレス／データ、消去ブロックなどを書き込み／消去インタフェースレジスタ／パラメータに指定することで、ダウンロードされたプログラムがフラッシュメモリの書き込み／消去を行います。

ダウンロードの要求、書き込み／消去の手順、結果の判定などの、ユーザ側で準備するプログラムの内、FCCS レジスタの SCO ビットを 1 に設定する部分は、内蔵 RAM 上で実行する必要があります。また、ダウンロードされる内蔵プログラムはすべて内蔵 RAM に存在します。これらが重複することのないように、内蔵 RAM 上の領域管理には注意してください。

図 4 にダウンロードされるプログラムの領域を示します。

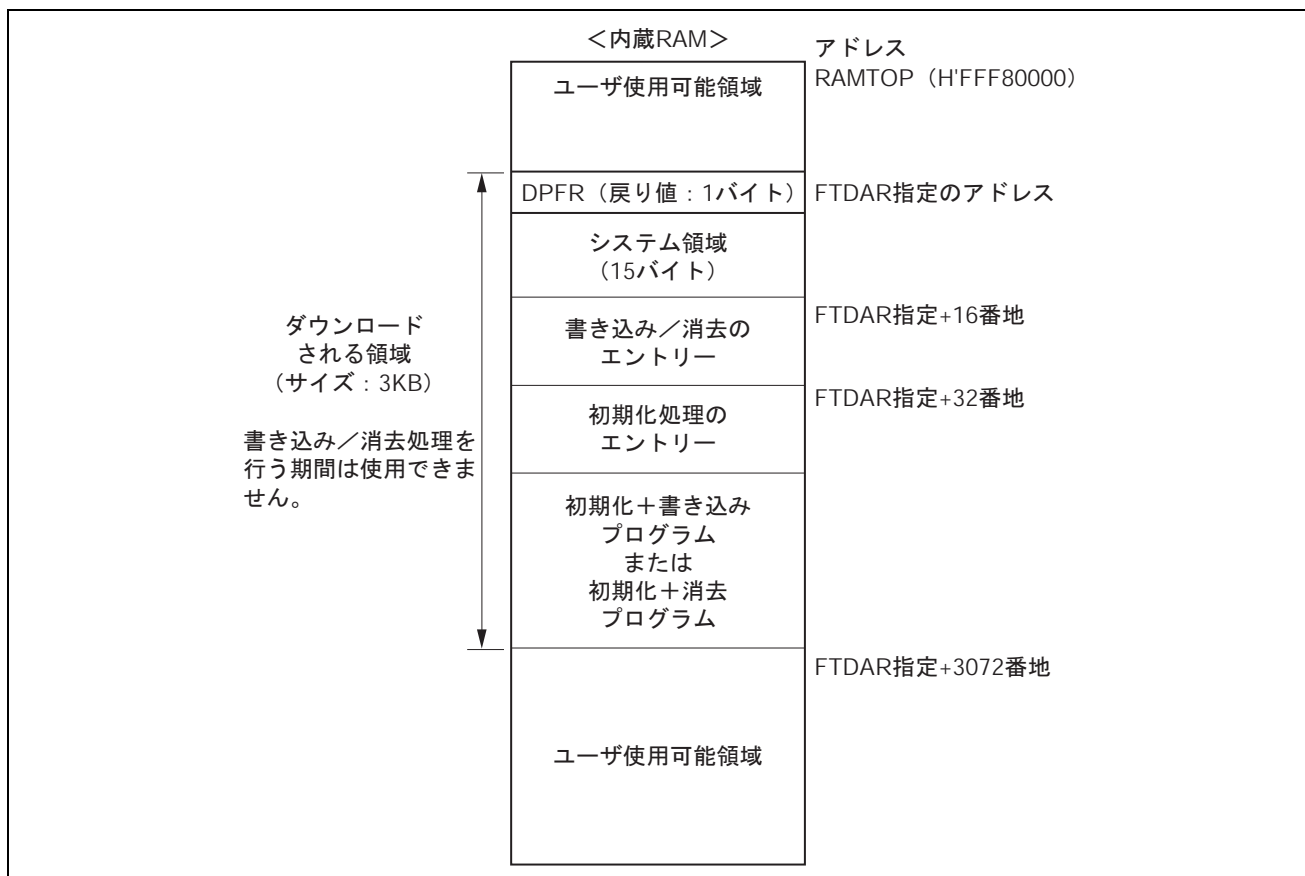


図4 ダウンロード後の内蔵 RAM マップ

## 2.2.2 フラッシュ消去

FTDAR レジスタで、ダウンロード先の内蔵 RAM アドレスを変更することで、消去プログラムと書き込みプログラムを別々の内蔵 RAM 領域にダウンロードすることができます。

図 5 に消去、書き込みを繰り返し実行する際の使用例を示します。

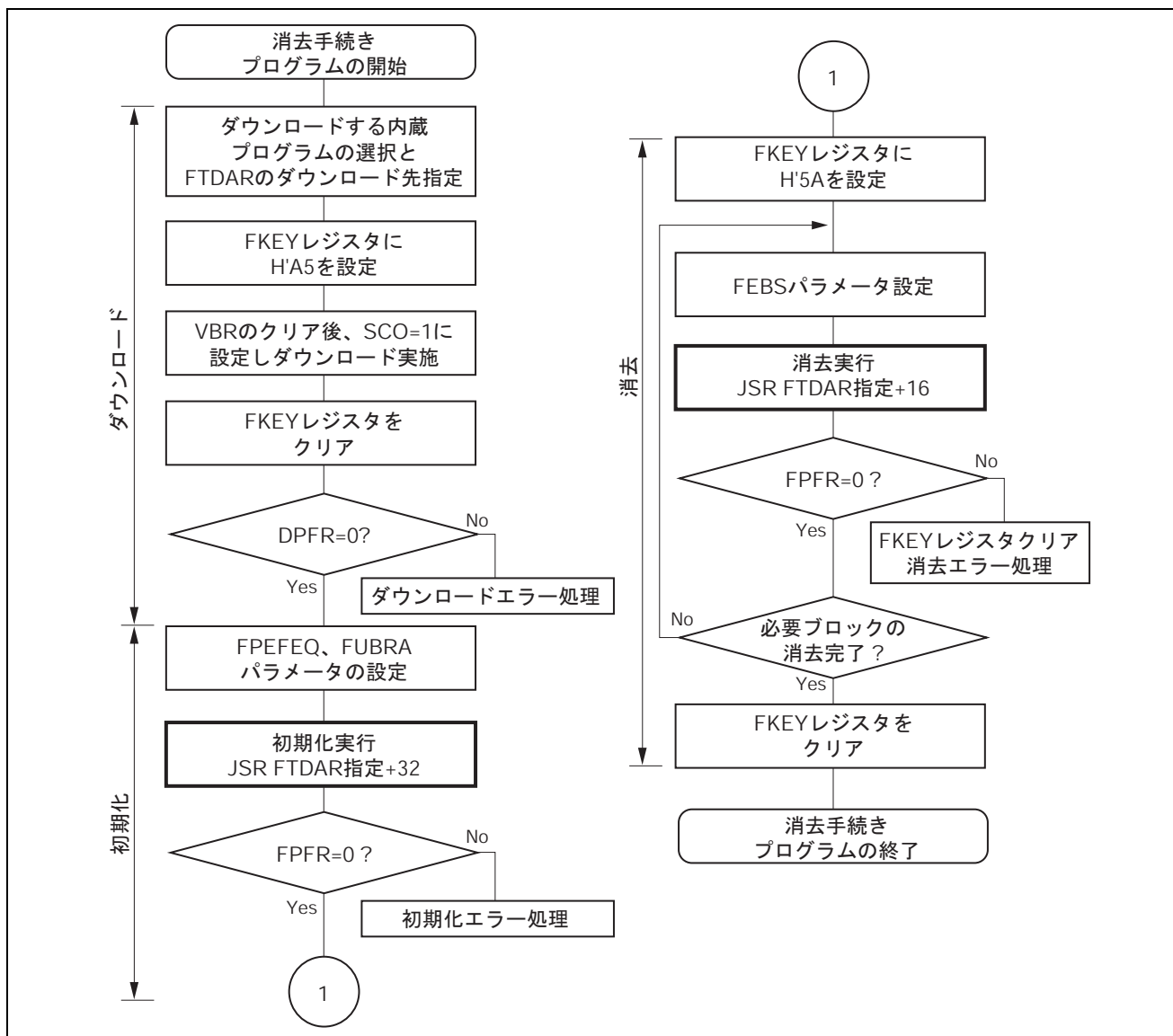


図5 フラッシュメモリの消去手順

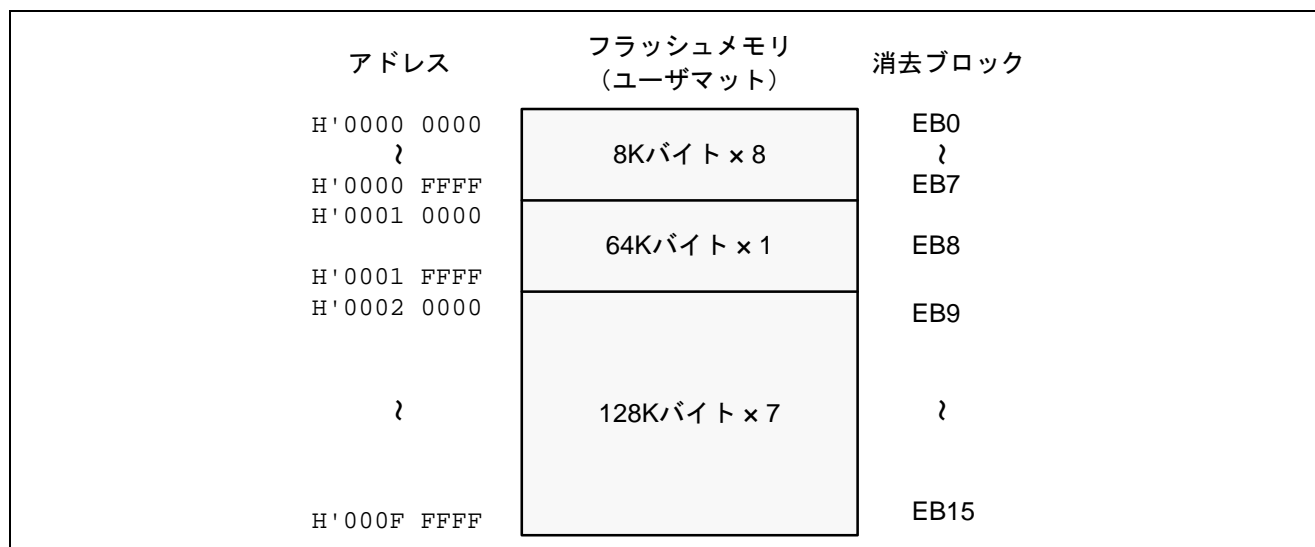


図6 フラッシュメモリの消去ブロック割

表1 消去ブロックとアドレス

消去ブロック	アドレス	単位容量
EB0	H'0000_0000~H'0000_1FFF	8K バイト
EB1	H'0000_2000~H'0000_3FFF	
EB2	H'0000_4000~H'0000_5FFF	
EB3	H'0000_6000~H'0000_7FFF	
EB4	H'0000_8000~H'0000_9FFF	
EB5	H'0000_A000~H'0000_BFFF	
EB6	H'0000_C000~H'0000_DFFF	
EB7	H'0000_E000~H'0000_FFFF	
EB8	H'0001_0000~H'0001_FFFF	64K バイト
EB9	H'0002_0000~H'0003_FFFF	128K バイト
EB10	H'0004_0000~H'0005_FFFF	
EB11	H'0006_0000~H'0007_FFFF	
EB12	H'0008_0000~H'0009_FFFF	
EB13	H'000A_0000~H'000B_FFFF	
EB14	H'000C_0000~H'000D_FFFF	
EB15	H'000E_0000~H'000F_FFFF	



## 2.2.3 フラッシュ書き込み

FTDAR レジスタで、ダウンロード先の内蔵 RAM アドレスを変更することで、消去プログラムと書き込みプログラムを別々の内蔵 RAM 領域にダウンロードすることができます。

図 5 に消去、書き込みを繰り返し実行する際の使用例を示します。

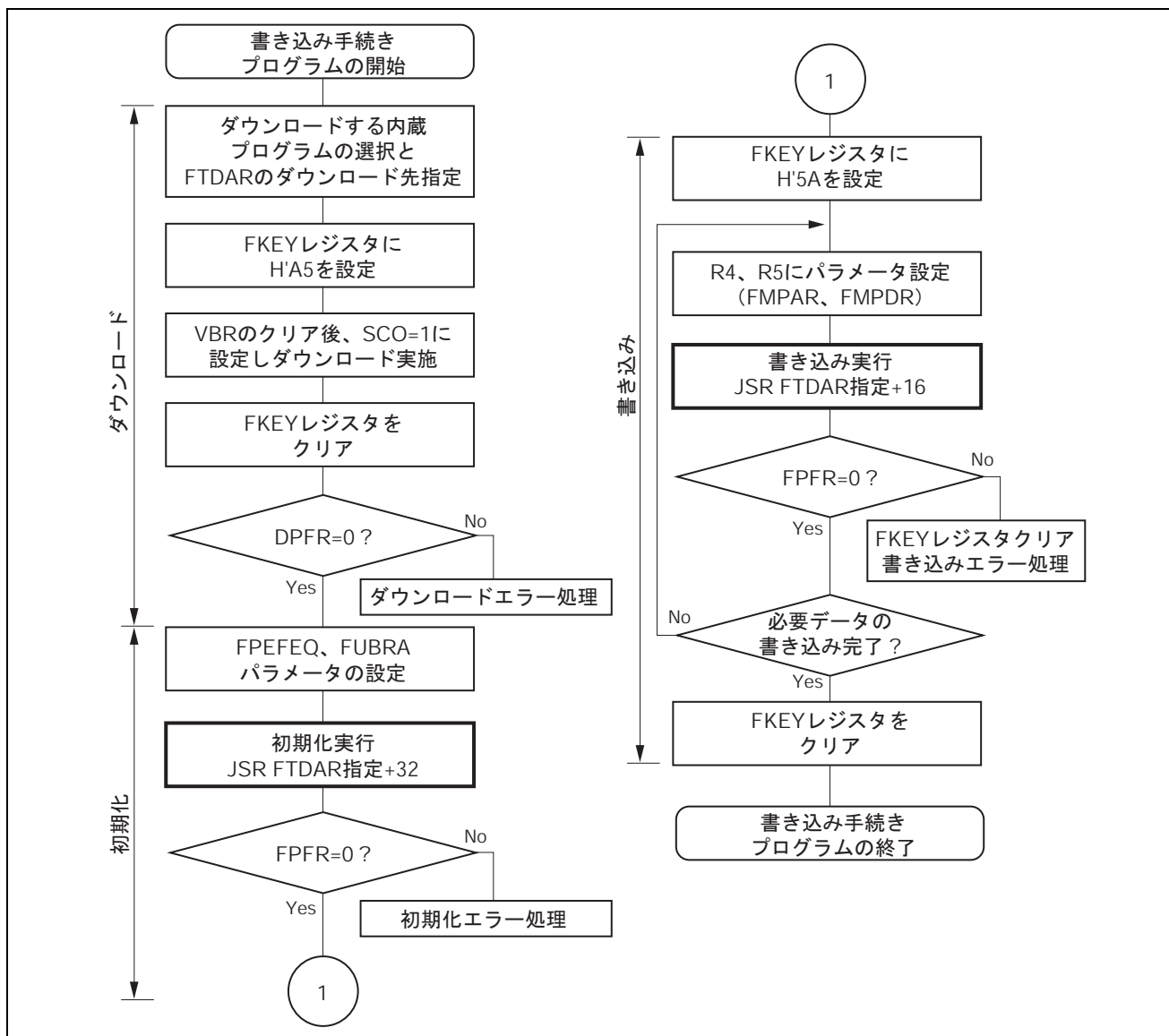


図7 フラッシュメモリの書き込み手順

## 2.3 フラッシュ書き換え用データバッファ

本応用例では、SH7286の内蔵RAM上に書き込みデータを保持しておくバッファエリアを確保します。バッファエリアの容量は、1回のフラッシュ書き込みに相当する256バイトとします。

図8に動作イメージを、表2にデータバッファエリアのアドレスを示します\*。

**【注】※** データバッファエリアはセクションを分けて管理しています。セクションの割付アドレスを変更することで、バッファエリアのアドレスを任意に設定可能です。ただし、内蔵プログラムのダウンロード先と重複することのないように、内蔵RAM上の領域管理には注意してください。

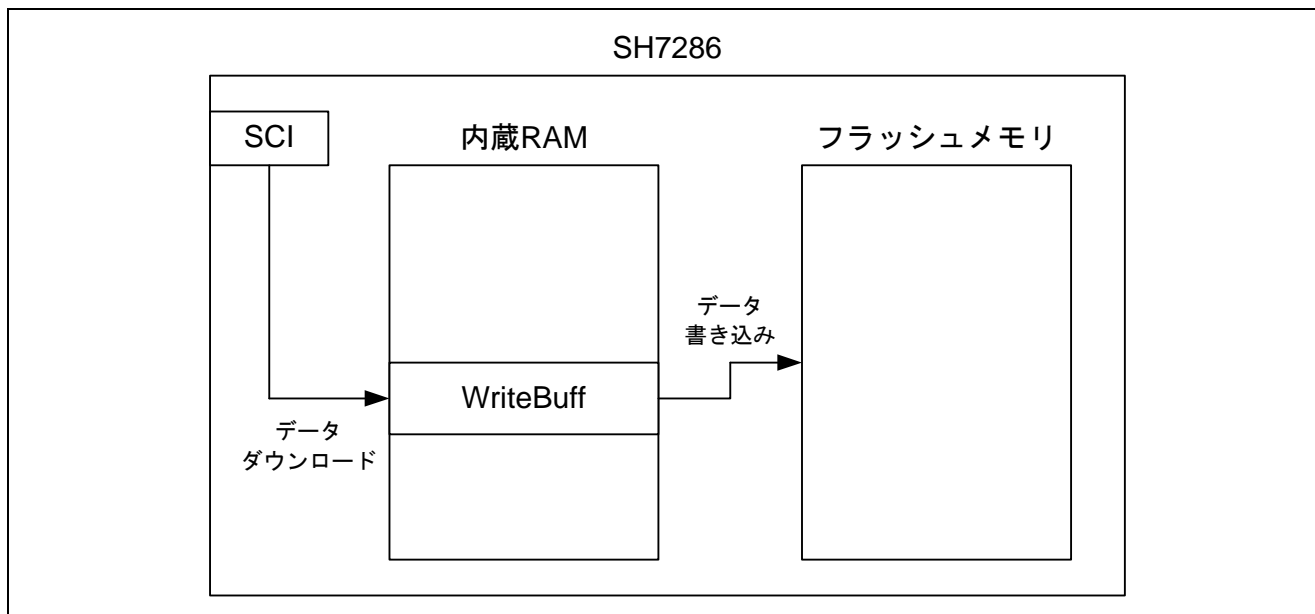


図8 バッファ動作イメージ

表2 データバッファエリア説明

バッファ名	アドレス	サイズ
WriteBuff	H'FFF8_5000~H'FFF8_50FF	256 バイト

### 3. 参考プログラムの外部仕様

本応用例では、main 関数を含むフラッシュ書き換え用サンプルプログラム（以下、参考プログラム）をユーザマットの EB0 ブロック（アドレス番地：H'0000 0000 ～ H'0000 1FFF）に配置しています。参考プログラムは、ユーザアプリケーション（main 関数）、シリアル通信制御プログラム、およびフラッシュ書き換え制御プログラムおよび標準 API で構成されます。

フラッシュメモリの書き込み/消去対象領域は、参考プログラムを配置している EB0 ブロックを除くユーザマット領域（EB1～EB15 ブロックのアドレス番地：H'0000 2000 ～ H'000F FFFF）です。

図 9 に参考プログラムのフラッシュ書き込み/消去処理イメージを示します。

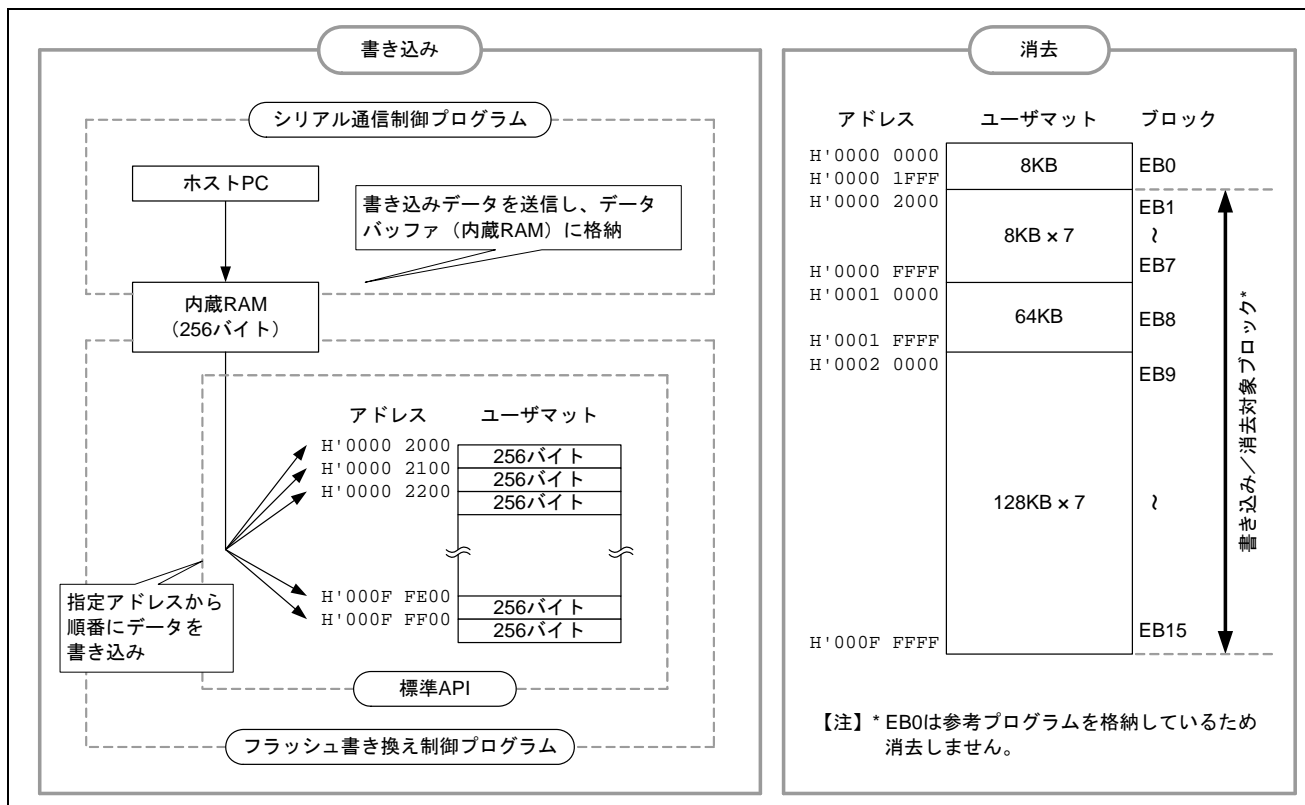


図9 フラッシュ書き込み/消去イメージ

### 3.1 参考プログラムの動作内容

本応用例では、ホスト PC\*とのシリアル通信によって、通信用ユーザ制御コマンドやデータの受け渡しをすることで、フラッシュへの書き込み/消去処理、および読み出し処理を行います。シリアル通信には、SCI チャンネル 0 (SCI0) を使用します。参考プログラムでは、これらのフラッシュ制御処理をすべて内蔵 RAM 上で実行します。

**【注】\*** 本応用例の動作確認には、ホスト PC 側のシリアル通信制御ツールに SerialDebugger (フリーソフトウェア) を使用しています。

参考プログラムでは、フラッシュ書き込み/消去の許可状態を確認し、フラッシュ書き込み/消去が許可状態に設定されている場合、参考プログラムはホスト PC に対し通信用ユーザ制御コマンドの発行を要求します。フラッシュ書き込み/消去が禁止状態に設定されている場合、参考プログラムは書き込み/消去が許可状態になるまで FWE ビットをポーリングします。

図 10にメイン処理の概略フローを示します。

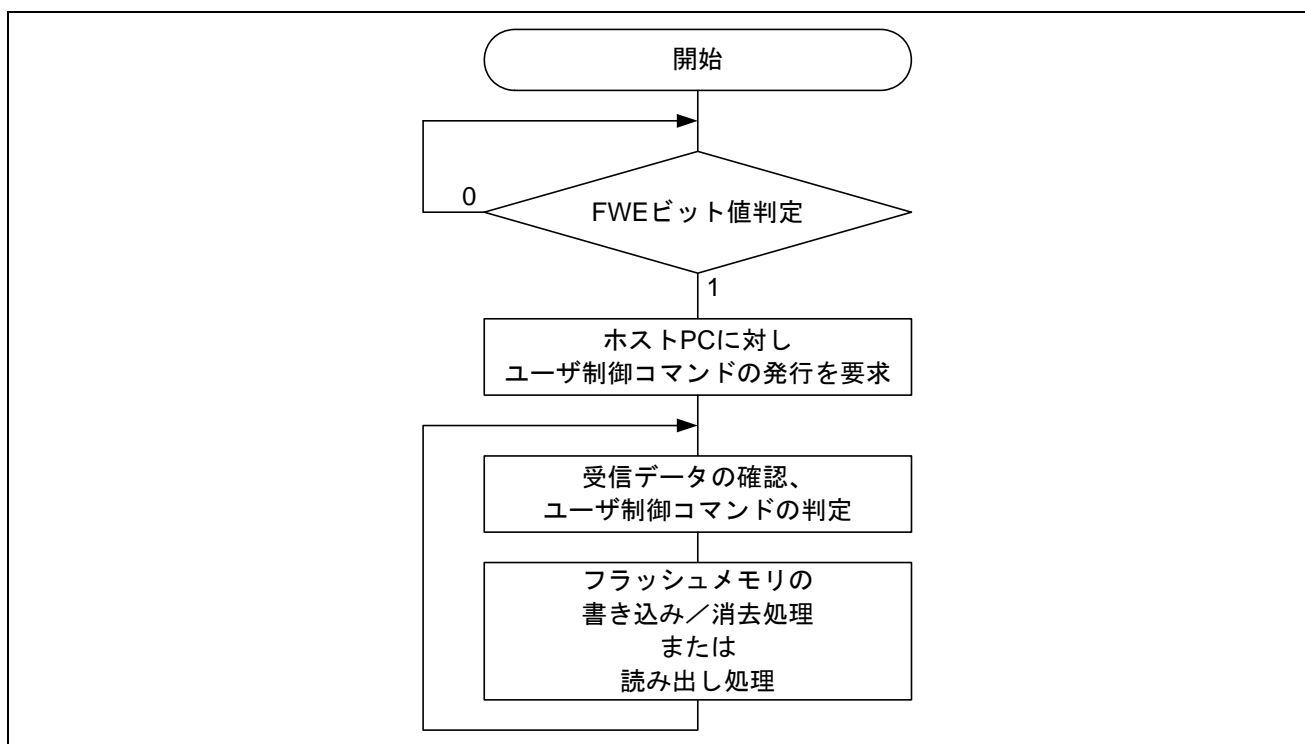


図10 メイン処理の概略フロー

表 3にホストPCからの通信用ユーザ制御コマンド一覧を、表 4にSH7286 からの通知一覧を示します。

フラッシュ書き込み／消去処理中に異常が発生した場合、参考プログラムの処理として、ホスト PC に対し異常終了 (RET\_NG) を通知し、無限ループとしています。必要に応じてエラー処理を追加してください。

表3 ホスト PC からの通信用ユーザ制御コマンド一覧 (ホスト PC → SH7286)

通信コマンド名	コマンド値	機能
CMD_GO	H'55	フラッシュ書き込み／消去処理開始コマンド
CMD_READ	H'AA	フラッシュ読み出し処理コマンド
CMD_ERASE	H'77	フラッシュ消去処理コマンド
CMD_WRITE	H'88	フラッシュ書き込み処理コマンド
CMD_WEND	H'99	フラッシュ書き込み／消去処理終了コマンド

表4 SH7286 からの通知一覧 (SH7286 → ホスト PC)

通知名	応答値	機能
正常終了 (RET_OK)	H'00	コマンド処理が正常に終了したことをホスト PC に通知
異常終了 (RET_NG)	H'01	コマンド処理中に異常が発生したことをホスト PC に通知
送信要求 (RET_REQ)	H'11	ユーザ制御コマンドまたは書き込みデータの送信要求状態であることをホスト PC に通知

### 3.1.1 フラッシュメモリの書き込み／消去

ホスト PC からフラッシュ書き込み／消去処理開始コマンド (CMD\_GO) を送信すると、参考プログラムはフラッシュメモリの書き込み／消去動作状態に遷移し、ホスト PC に対し送信要求 (RET\_REQ) を通知します。

次に、ホスト PC からフラッシュ消去処理コマンド (CMD\_ERASE) を送信し、続けて書き込み／消去先のブロック番号 (ただし、EB0 以外) を 2 バイト形式で指定します。この 2 バイト形式のデータは、指定ブロックの番号に対応するビット (ただし、ビット 0 以外) を 1 に設定した値にしてください (例えば EB1 に書き込みたい場合は H'0002 を指定し、EB11 に書き込みたい場合は H'0800 を指定します)。指定ブロックがない場合 (H'0000 を指定)、または EB0 に対応するビット 0 を 1 に設定した場合、参考プログラムはホスト PC に対し異常終了 (RET\_NG) を通知し無限ループとなります。指定ブロックにおけるフラッシュ消去処理が完了すると、参考プログラムはホスト PC に対し正常終了 (RET\_OK) を通知します。

次に、ホスト PC からフラッシュ書き込み処理コマンド (CMD\_WRITE) を送信し、続けて書き込み先の先頭アドレスおよび書き込みデータサイズ (4 バイト形式) を送信します。このとき、書き込み先の先頭アドレスにはフラッシュ消去時に指定したブロック (EB1～EB15) 内のアドレス (H'0000 2000 ～ H'000F FFFF) を 256 バイト境界で指定してください。上記以外のアドレスを送信して書き込み処理を行った場合の動作は保証されません。

書き込み先の先頭アドレスおよび書き込みデータサイズを送信すると、参考プログラムはホスト PC に対し書き込みデータの送信要求 (RET\_REQ) を通知するので、ホスト PC から書き込みデータサイズ分のデータを送信します。ここで、ユーザマットへの書き込み単位は 256 バイトになっているため、参考プログラムは 256 バイトのデータ受信ごとにフラッシュ書き込みを行います。(指定した書き込みデータサイズが 256 バイト未満の場合、参考プログラムは残り 256 バイトまでの端数分のデータを H'FF に設定します。)

フラッシュ書き込み総数が書き込みデータサイズに満たない場合、参考プログラムはホスト PC に対し書き込みデータの送信要求 (RET\_REQ) を通知するので、書き込みデータサイズに達するまでホスト PC からデータ送信を続けてください。フラッシュ書き込み総数が書き込みデータサイズに達すると、参考プログラムはホスト PC に対し正常終了 (RET\_OK) を通知します。

この後、ホスト PC からフラッシュ書き込み／消去処理終了コマンド (CMD\_WEND) を送信すると、参考プログラムはフラッシュ書き込み／消去処理を終了します。

図 11に、参考プログラムにおけるフラッシュ書き込み／消去処理時の通信コマンド制御シーケンスを、また図 12にフラッシュ書き込み／消去処理の通信コマンド制御例 (SerialDebugger使用) を示します。

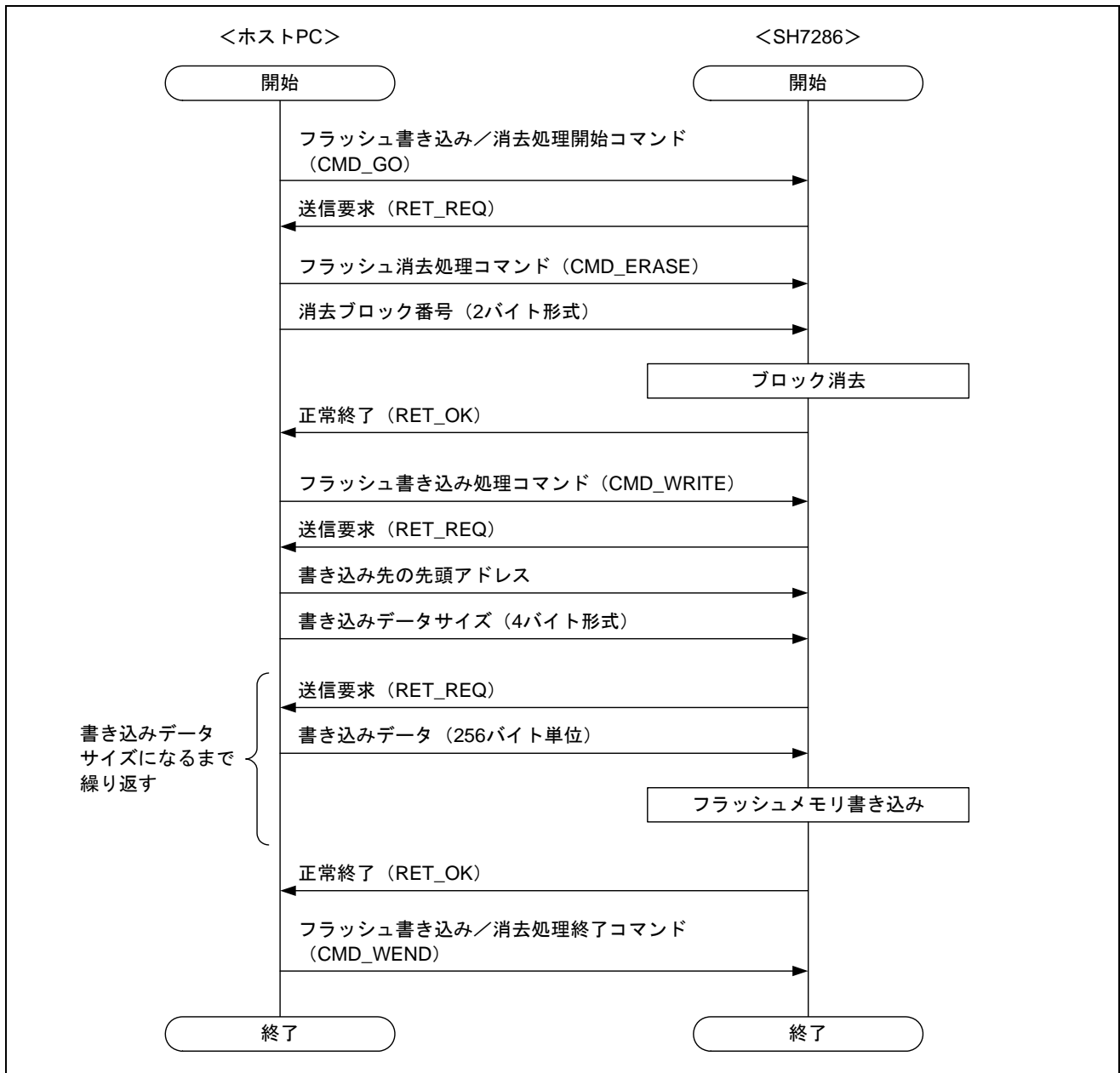


図11 フラッシュ書き込み/消去処理の通信コマンド制御シーケンス

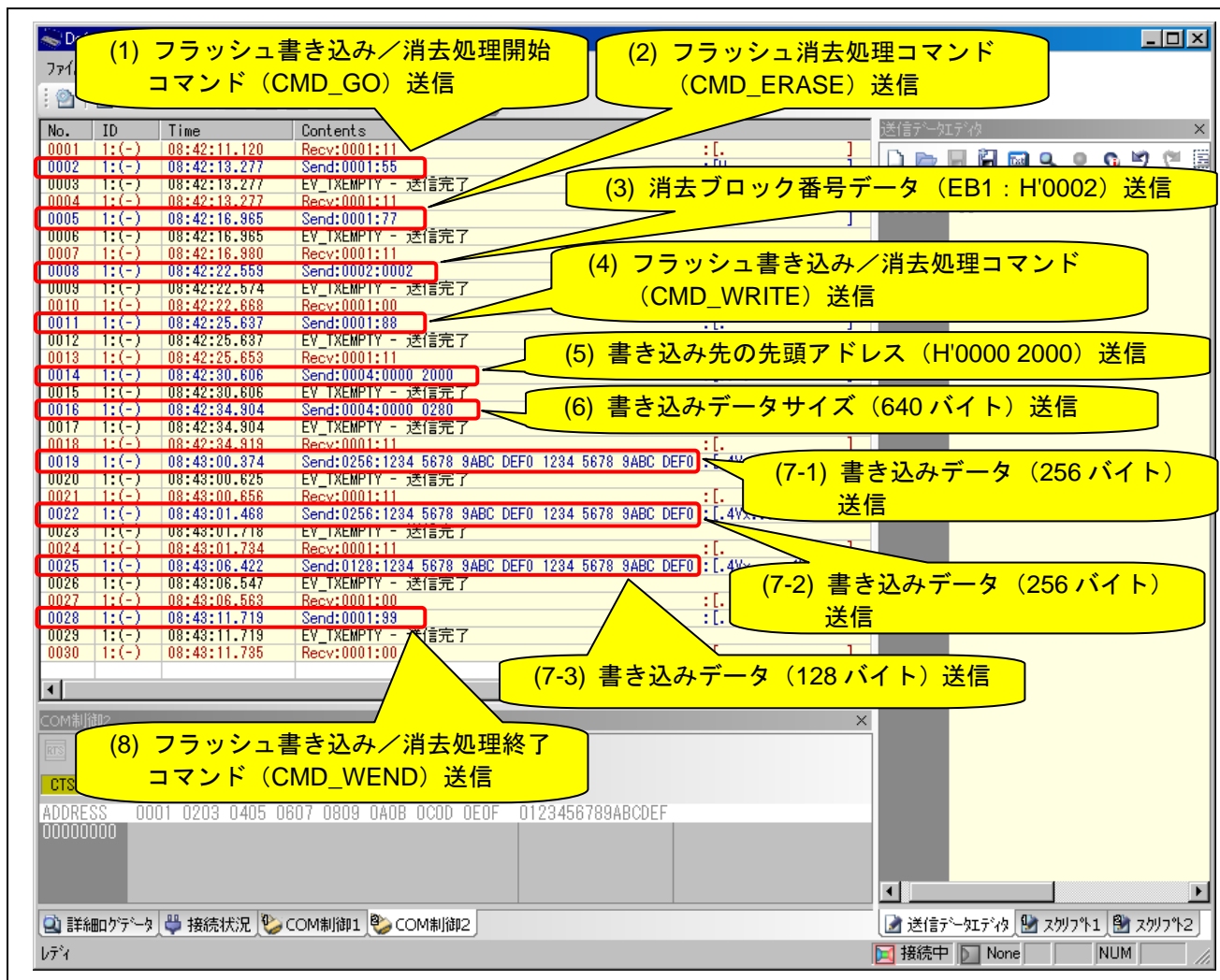


図12 フラッシュ書き込み/消去処理の通信コマンド制御例 (SerialDebugger 使用)

### 3.1.2 フラッシュメモリの読み出し

フラッシュ読み出し処理コマンド (CMD\_READ) により、参考プログラムは読み出し先の先頭アドレスから指定サイズのデータを読み出してホスト PC へ送信します。

参考プログラムは、フラッシュ読み出し処理コマンド (CMD\_READ) を受信すると、ホスト PC に対し送信要求 (RET\_REQ) を通知します。次に、ホスト PC から読み出し先の先頭アドレス (4 バイト形式) と読み出しデータサイズ (4 バイト形式) の計 8 バイトを受信すると、読み出し先のアドレスから指定サイズのデータを読み出してホスト PC へ送信します。

読み出し先の先頭アドレスには EB0~EB15 ブロック (ユーザマット) 内のアドレス (H'0000 0000 ~ H'000F FFFF) を指定してください。ユーザマット以外のアドレスが指定された場合、参考プログラムは読み出し処理を行わず、ホスト PC に対し異常終了 (RET\_NG) を通知し無限ループとなります。また、ユーザマット以外のアドレスが指定された場合のエラーチェックは行っていませんので、範囲外のアドレスは指定しないでください。

図 13 に参考プログラムにおけるフラッシュ読み出し処理の通信コマンド制御シーケンスを、また 図 14 にフラッシュ読み出し処理の通信コマンド制御例 (SerialDebugger 使用) を示します。



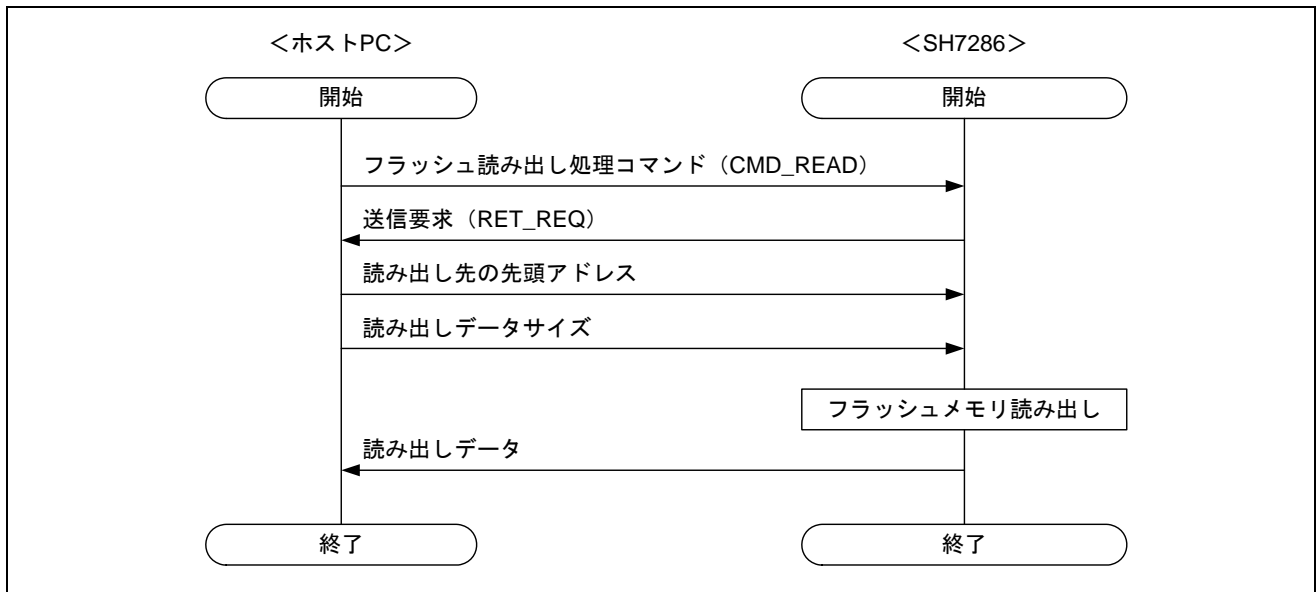


図13 フラッシュ読み出し処理の通信コマンド制御シーケンス

The screenshot shows the SerialDebugger interface with a log of communication events. Three specific events are highlighted with yellow callouts:

- (1) フラッシュ読み出し処理コマンド (CMD\_READ) 送信: This corresponds to the 'Send:0001:11' entry in the log.
- (2) 読み出し先の先頭アドレス (H'0000 2000) 送信: This corresponds to the 'Send:0004:0000 2000' entry in the log.
- (3) 読み出しデータサイズ (768 バイト) 送信: This corresponds to the 'Send:0004:0000 0300' entry in the log.

The log table below shows the sequence of events:

No.	ID	Time	Contents
0001	1:(-)	08:45:00.150	Recv:0001:11
0002	1:(-)	08:45:02.619	Send:0001:AA
0003	1:(-)	08:45:02.619	EV_TXEMPTY - 送信完了
0004	1:(-)	08:45:02.619	Recv:0001:11
0005	1:(-)	08:45:23.011	Send:0004:0000 2000
0006	1:(-)	08:45:23.011	EV_TXEMPTY - 送信完了
0007	1:(-)	08:45:27.856	Send:0004:0000 0300
0008	1:(-)	08:45:27.856	EV_TXEMPTY - 送信完了
0009	1:(-)	08:45:27.871	Recv:0014:1234 5678 9ABC DEFO
0010	1:(-)	08:45:27.887	Recv:0014:DEFO 1234 5678 9ABC DEFO 1234 5678 : [...4Vx....4Vx...
0011	1:(-)	08:45:27.902	Recv:0014:9ABC DEFO 1234 5678 9ABC DEFO 1234 : [...4Vx....4...
0012	1:(-)	08:45:27.918	Recv:0014:5678 9ABC DEFO 1234 5678 9ABC DEFO : [Vx....4Vx....
0013	1:(-)	08:45:27.934	Recv:0014:1234 5678 9ABC DEFO 1234 5678 9ABC : [.4Vx....4Vx..
0014	1:(-)	08:45:27.949	Recv:0014:DEFO 1234 5678 9ABC DEFO 1234 5678 : [...4Vx....4Vx...
0015	1:(-)	08:45:27.965	Recv:0014:9ABC DEFO 1234 5678 9ABC DEFO 1234 : [...4Vx....4...
0016	1:(-)	08:45:27.981	Recv:0014:5678 9ABC DEFO 1234 5678 9ABC DEFO : [Vx....4Vx....
0017	1:(-)	08:45:27.996	Recv:0014:1234 5678 9ABC DEFO 1234 5678 9ABC : [.4Vx....4Vx..
0018	1:(-)	08:45:27.996	Recv:0014:DEFO 1234 5678 9ABC DEFO 1234 5678 : [...4Vx....4Vx...
0019	1:(-)	08:45:28.012	Recv:0014:9ABC DEFO 1234 5678 9ABC DEFO 1234 : [...4Vx....4...
0020	1:(-)	08:45:28.027	Recv:0014:5678 9ABC DEFO 1234 5678 9ABC DEFO : [Vx....4Vx....
0021	1:(-)	08:45:28.043	Recv:0014:1234 5678 9ABC DEFO 1234 5678 9ABC : [.4Vx....4Vx..
0022	1:(-)	08:45:28.059	Recv:0014:DEFO 1234 5678 9ABC DEFO 1234 5678 : [...4Vx....4Vx...
0023	1:(-)	08:45:28.074	Recv:0014:9ABC DEFO 1234 5678 9ABC DEFO 1234 : [...4Vx....4...
0024	1:(-)	08:45:28.090	Recv:0014:5678 9ABC DEFO 1234 5678 9ABC DEFO : [Vx....4Vx....
0025	1:(-)	08:45:28.106	Recv:0014:1234 5678 9ABC DEFO 1234 5678 9ABC : [.4Vx....4Vx..
0026	1:(-)	08:45:28.121	Recv:0014:DEFO 1234 5678 9ABC DEFO 1234 5678 : [...4Vx....4Vx...
0027	1:(-)	08:45:28.137	Recv:0014:9ABC DEFO 1234 5678 9ABC DEFO 1234 : [...4Vx....4...
0028	1:(-)	08:45:28.152	Recv:0014:5678 9ABC DEFO 1234 5678 9ABC DEFO : [Vx....4Vx....
0029	1:(-)	08:45:28.168	Recv:0014:1234 5678 9ABC DEFO 1234 5678 9ABC : [.4Vx....4Vx..
0030	1:(-)	08:45:28.184	Recv:0014:DEFO 1234 5678 9ABC DEFO 1234 5678 : [...4Vx....4Vx...
0031	1:(-)	08:45:28.199	Recv:0014:9ABC DEFO 1234 5678 9ABC DEFO 1234 : [...4Vx....4...

図14 フラッシュ読み出し処理の通信コマンド制御例 (SerialDebugger 使用)

## 4. 参考プログラムの内部仕様

### 4.1 モジュール仕様

表 5に参考プログラムのモジュール仕様を示します。

表5 参考プログラムのモジュール仕様

分類	モジュール名	関数名	機能	処理フロー
ユーザアプリケーション	メイン処理	main	ユーザアプリケーションプログラムの実行	図 15参照
フラッシュ書き換え制御	フラッシュ書き込み／消去処理	ocf_write	フラッシュ書き込み／消去処理の実行	図 16, 図 17参照
	フラッシュ読み出し処理	ocf_read	フラッシュ読み出し処理の実行	図 18参照
	フラッシュ書き込み／消去許可状態チェック処理	ocf_pe_chk	フラッシュ書き込み／消去許可状態の確認	図 19参照
シリアル通信制御	SCI 初期化処理	io_sci_init	SCI (チャンネル 0) の初期設定	—
	SCI 受信データ有無チェック処理	io_sci_chk_rcv	受信データが SCRDR レジスタに格納されているか確認	—
	SCI 送信処理	io_sci_snd	1 バイトデータ送信	—
	SCI 受信処理	io_sci_rcv	指定されたバイト数分のデータ受信	—
	SCI モジュール動作停止	io_sci_stop	SCI (チャンネル 0) へのクロック供給を停止	—
標準 API	ブロック消去	R_FlashErase	指定ブロック内のデータ消去	—
	フラッシュ書き込み	R_FlashWrite	指定アドレス領域へのデータ書き込み	—

### 4.2 使用変数

表 6に参考プログラムにて使用する変数を示します。

表6 参考プログラムにおける使用変数

変数ラベル名	機能	使用モジュール
unsigned char WriteBuff[256]	書き込みデータ格納領域	ocf_write

## 4.3 レジスタ設定

表7に参考プログラムにて使用する周辺機能のレジスタ設定内容を示します。

表7 参考プログラムにおけるレジスタ設定内容

レジスタ	アドレス	設定値	設定内容
周波数制御レジスタ (FRQCR)	H'FFFE 0010	H'0333	<ul style="list-style-type: none"> <li>STC[2:0] = "B'011" : Bφの分周率=1/4</li> <li>IFC[2:0] = "B'011" : Iφの分周率=1/4</li> <li>PFC[2:0] = "B'011" : Pφの分周率=1/4</li> </ul>
スタンバイコントロール レジスタ 5 (STBCR5)	H'FFFE 0418	H'7F	MSTP57 = "0" : SCI0 は動作
シリアルモードレジスタ_0 (SCSMR_0)	H'FFFF 8000	H'00	<ul style="list-style-type: none"> <li>C/A# = "0" : 調歩同期式モード</li> <li>CHR = "0" : 8ビットデータ</li> <li>PE = "0" : パリティビットの付加、およびチェックを 禁止</li> <li>STOP = "0" : 1ストップビット</li> <li>MP = "0" : マルチプロセッサモードを禁止</li> <li>CKS[1:0] = "B'00" : Pφクロック</li> </ul>
ビットレートレジスタ_0 (SCBRR_0)	H'FFFF 8002	D'80	ビットレート=9600 bps (Pφ=25 MHz)
シリアルコントロール レジスタ_0 (SCSCR_0)	H'FFFF 8004	H'30	<ul style="list-style-type: none"> <li>TE = "1" : 送信動作を許可</li> <li>RE = "1" : 受信動作を許可</li> </ul>
ポート A コントロール レジスタ L1 (PACRL1)	H'FFFE 3816	H'0055	<ul style="list-style-type: none"> <li>PA1MD[2:0] = "B'101" : TXD0 出力 (SCI0)</li> <li>PA0MD[2:0] = "B'101" : RXD0 入力 (SCI0)</li> </ul>

## 4.4 フローチャート

本節では参考プログラムの各モジュール処理フローを示します。

## 4.4.1 メイン処理フロー

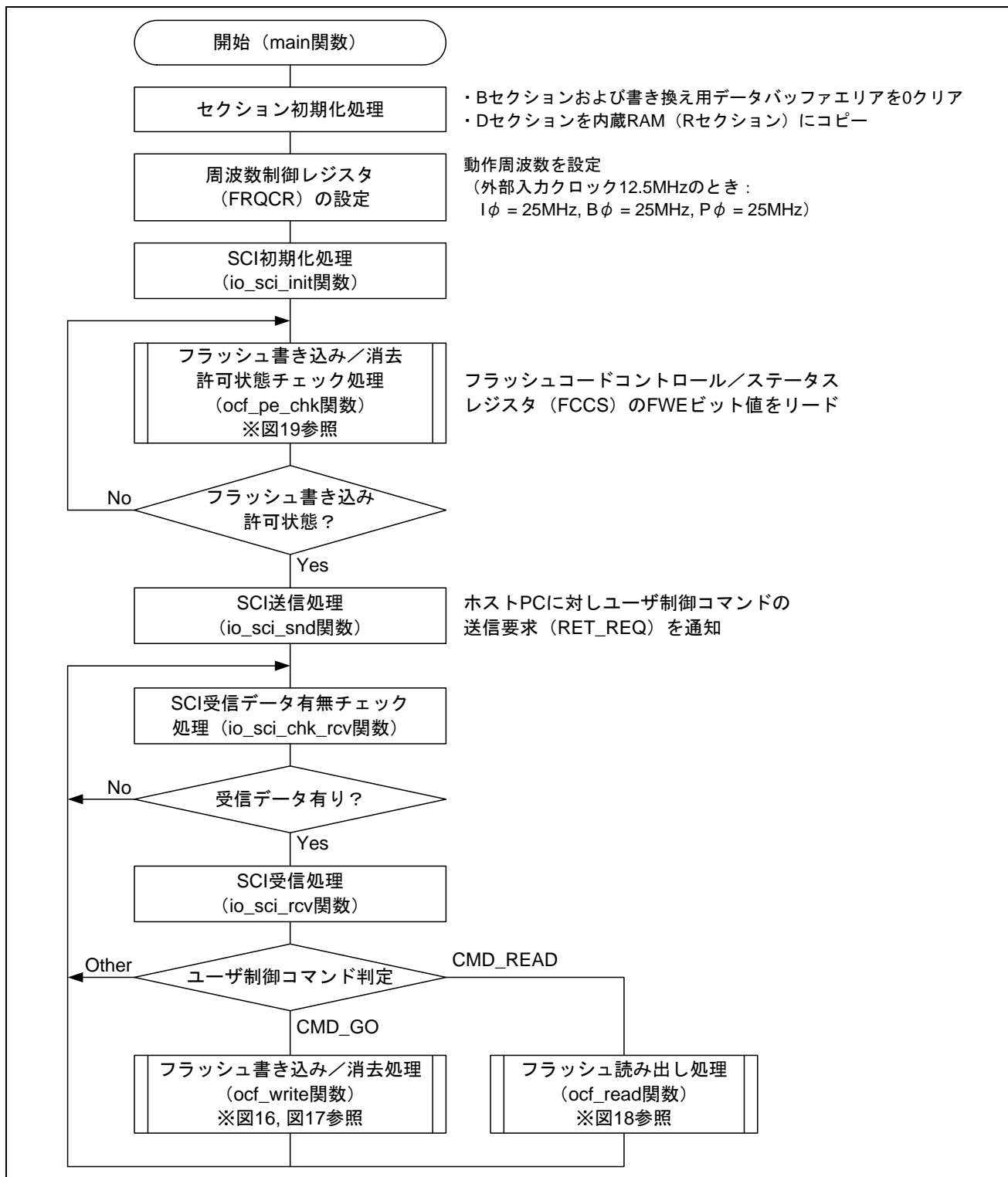


図15 メイン処理フロー

4.4.2 フラッシュ書き込み/消去処理フロー

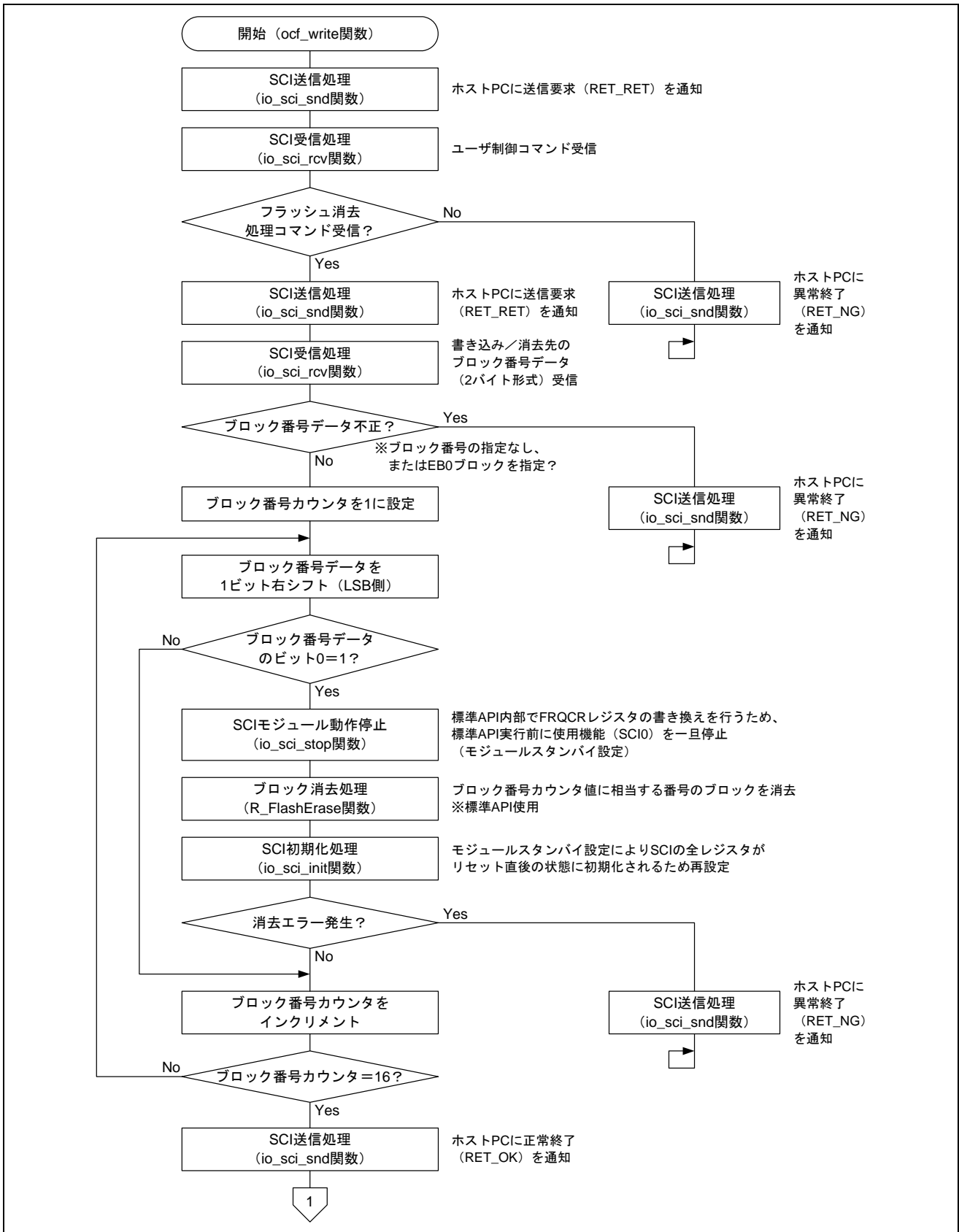


図16 フラッシュ書き込み/消去処理フロー (1)

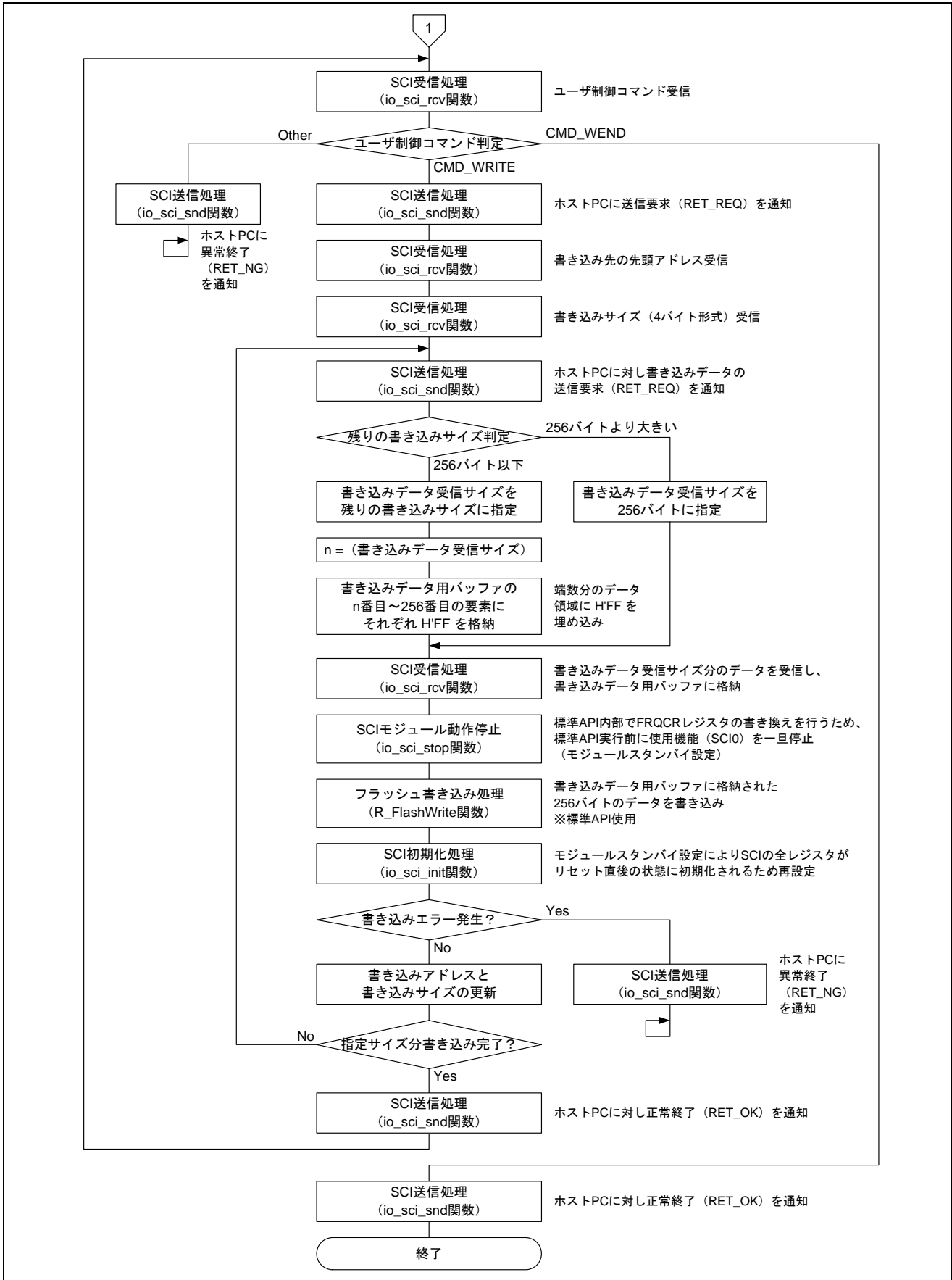


図17 フラッシュ書き込み/消去処理フロー (2)

## 4.4.3 フラッシュ読み出し処理フロー

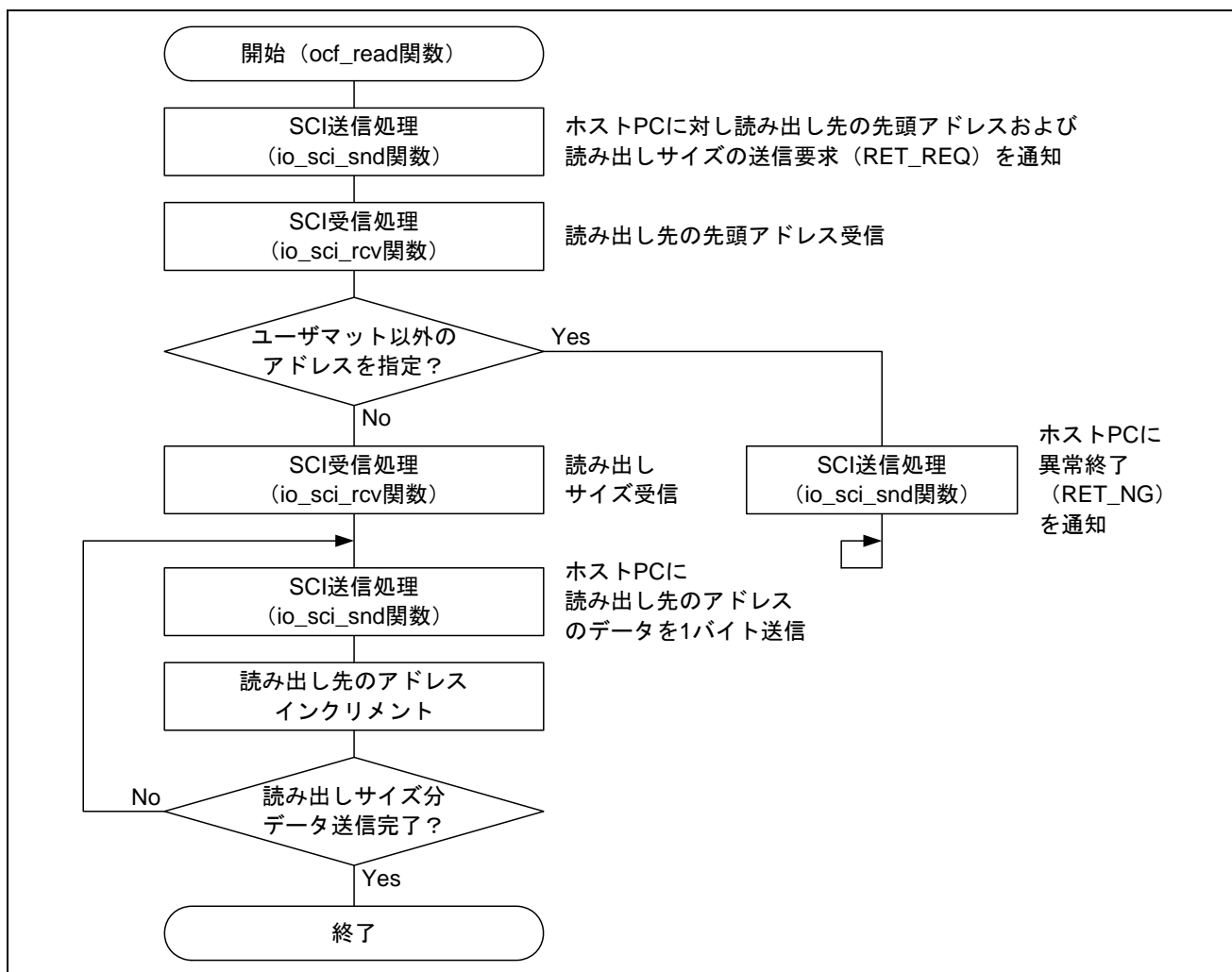


図18 フラッシュ読み出し処理フロー

## 4.4.4 フラッシュ書き込み/消去許可状態チェック処理フロー

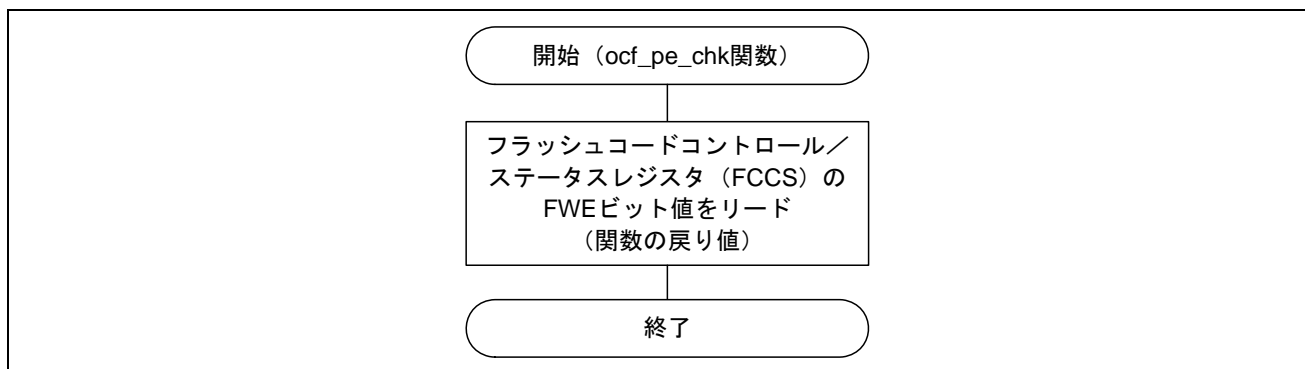


図19 フラッシュ書き込み/消去許可状態チェック処理フロー

## 5. 参考プログラムリスト

### 5.1 サンプルプログラムリスト "main.c" (1)

```
1  /*****
2  *   DISCLAIMER
3  *
4  *   This software is supplied by Renesas Electronics Corp. and is only
5  *   intended for use with Renesas products. No other uses are authorized.
6  *
7  *   This software is owned by Renesas Electronics Corp. and is protected under
8  *   all applicable laws, including copyright laws.
9  *
10 *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11 *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12 *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13 *   PARTICULAR PURPOSE AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY
14 *   DISCLAIMED.
15 *
16 *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17 *   ELECTRONICS CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18 *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19 *   FOR ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS
20 *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21 *
22 *   Renesas reserves the right, without notice, to make changes to this
23 *   software and to discontinue the availability of this software.
24 *   By using this software, you agree to the additional terms and
25 *   conditions found by accessing the following link:
26 *   http://www.renesas.com/disclaimer
27 *****/
28 *   Copyright (C) 2010 Renesas Electronics Corporation. All rights reserved.
29 *****/
30 /*"FILE COMMENT"***** Technical reference data *****
31 *   System Name : SH7286 Sample Program
32 *   File Name   : main.c
33 *   Abstract    : Example of use for the Flash user-program mode
34 *   Version     : 1.00.00
35 *   Device      : SH7286
36 *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
37 *               : C/C++ compiler package for the SuperH RISC engine family
38 *               :                               (Ver.9.01 Release01).
39 *   OS          : None
40 *   H/W Platform: M3A-HS87 (CPU board)
41 *   Description :
42 *****/
43 *   History     : Sep.10,2010 Ver.1.00.00
44 *"FILE COMMENT END"*****
45 #include <machine.h>
46 #include "iodefine.h"
47 #include "Flash_API_SH7280.h"
48
```



## 5.2 サンプルプログラムリスト "main.c" (2)

```
49  /* ==== Macro definition ==== */
50  #define FLASH_PE_ENABLE    1 /* Flash program/erase enabled    */
51  #define FLASH_PE_DISABLE  0 /* Flash program/erase disabled  */
52  #define PROGRAM_SIZE      256 /* Flash programming size unit   */
53
54  #define CMD_GO             0x55 /* フラッシュ書き込み/消去開始処理コマンド */
55  #define CMD_READ          0xaa /* フラッシュ読み出し処理コマンド */
56  #define CMD_ERASE         0x77 /* フラッシュ消去処理コマンド */
57  #define CMD_WRITE         0x88 /* フラッシュ書き込み処理コマンド */
58  #define CMD_WEND          0x99 /* フラッシュ書き込み/消去終了コマンド */
59  #define RET_OK            0x00 /* 正常終了 */
60  #define RET_NG            0x01 /* 異常終了 */
61  #define RET_REQ           0x11 /* 送信要求 */
62
63  /* ==== Prototype declaration ==== */
64  void main(void); /* main function */
65  void _section_init(void); /* section initialization function */
66  void io_cpg_init(void); /* FRQCR setting function */
67  int ocf_pe_chk(void); /* Flash P/E check function */
68  void ocf_write(void); /* Flash program/erase processing function */
69  void ocf_read(void); /* Flash reading function */
70  void dummy_f(void); /* Dummy interrupt function */
71  /* ---- External reference ---- */
72  extern void io_sci_init(void);
73  extern int io_sci_chk_rcv(void);
74  extern void io_sci_snd(unsigned char data);
75  extern void io_sci_rcv(unsigned char *data, unsigned long num);
76  extern void io_sci_stop(void);
77
78  /* ==== Global variable ==== */
79  #pragma section WriteDATA /* 書き込みデータバッファエリア */
80  unsigned char WriteBuff[PROGRAM_SIZE];
81  #pragma section
82
```

## 5.3 サンプルプログラムリスト "main.c" (3)

```

83  /*"FUNC COMMENT"*****
84  * ID      :
85  * Outline  : サンプルプログラムメイン
86  *-----
87  * Include  :
88  *-----
89  * Declaration : void main(void);
90  *-----
91  * Description :
92  *-----
93  * Argument   : void
94  *-----
95  * Return Value : void
96  *-----
97  * Note       : None
98  *"FUNC COMMENT END"*****/
99  void main(void)
100 {
101     unsigned char RcvData;
102     int pe_ok;
103
104     /* ==== セクション初期化 ==== */
105     _section_init();
106
107     /* ==== FRQCR 初期化 ==== */
108     io_cpg_init();
109
110     /* ==== SCI 初期化 ==== */
111     io_sci_init();
112
113     /* ==== フラッシュ書き込み/消去許可チェック ==== */
114     do{
115         pe_ok = ocf_pe_chk();      /* FWE pin = High ? */
116     }while(pe_ok != FLASH_PE_ENABLE);
117
118     /* ==== ホスト PC に対し送信要求を通知 ==== */
119     io_sci_snd(RET_REQ);
120
121     /* ==== フラッシュ書き込み/消去処理 or フラッシュ読み出し処理 ==== */
122     while(1){
123         /* ---- ユーザ制御コマンドチェック ---- */
124         if(io_sci_chk_rcv() != 0){
125             io_sci_rcv(&RcvData, 1);
126             if(RcvData == CMD_GO){
127                 ocf_write();      /* フラッシュ書き込み/消去処理 */
128             }
129             else if(RcvData == CMD_READ){
130                 ocf_read();      /* フラッシュ読み出し処理 */
131             }
132         }
133     }
134 }

```

## 5.4 サンプルプログラムリスト "main.c" (4)

```
135
136 /*"FUNC COMMENT"*****
137 * ID      :
138 * Outline  : セクション初期化処理
139 *-----
140 * Include   : <machine.h>
141 *-----
142 * Declaration : void _section_init(void);
143 *-----
144 * Description : Bセクションおよび書き込みデータバッファエリアを0クリアした後、
145 *              : Dセクションを内蔵RAM(Rセクション)にコピーします。
146 *-----
147 * Argument   : void
148 *-----
149 * Return Value : void
150 *-----
151 * Note       : None
152 *"FUNC COMMENT END"*****/
153 void _section_init(void)
154 {
155     unsigned char *src, *dst, *end;
156
157     /* Zero our all un-initialized (BSS) RAM data as specified by ANSI. */
158     src = (unsigned char *)(__sectop("B"));
159     end = (unsigned char *)(__secend("B"));
160     while(src < end){
161         *src++ = 0x00;
162     }
163
164     src = (unsigned char *)(__sectop("BwriteDATA"));
165     end = (unsigned char *)(__secend("BwriteDATA"));
166     while(src < end){
167         *src++ = 0x00;
168     }
169
170     /* Copy in our all initialized (DATA) RAM data as specified by ANSI. */
171     src = (unsigned char *)(__sectop("D"));
172     dst = (unsigned char *)(__sectop("R"));
173     end = (unsigned char *)(__secend("D"));
174     while(src < end){
175         *dst++ = *src++;
176     }
177 }
178
```

## 5.5 サンプルプログラムリスト "main.c" (5)

```
179  /*"FUNC COMMENT"*****
180  * ID      :
181  * Outline  : FRQCR 初期化処理
182  *-----
183  * Include  : <machine.h> and "iodefine.h"
184  *-----
185  * Declaration : void io_cpg_init(void);
186  *-----
187  * Description : FRQCR レジスタの初期設定を行います。
188  *-----
189  * Argument   : void
190  *-----
191  * Return Value : void
192  *-----
193  * Note       : None
194  *"FUNC COMMENT END"*****/
195  void io_cpg_init(void)
196  {
197     volatile unsigned short dummy;
198
199     WDT.WRITE.WTCSR = 0xa51e; /* WDT stop, WDT count clock: 1/4096 x P-clock */
200                               /* (Overflow cycle = 41.94ms at P-clock 25MHz) */
201     WDT.WRITE.WTCNT = 0x5ac2; /* WDT counter for 10mS */
202     CPG.FRQCR.WORD = 0x0333; /* Clock-in = 12.5MHz */
203                               /* I-clock = 25MHz */
204                               /* B-clock = 25MHz */
205                               /* P-clock = 25MHz */
206                               /* -> (I : B : P) = (1/4 : 1/4 : 1/4) */
207     dummy = CPG.FRQCR.WORD; /* FRQCR readout */
208     /* ---- 64 NOPs for 32 x P-clock (I:P = 1:1) ---- */
209     nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
210     nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
211     nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
212     nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
213     nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
214     nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
215     nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
216     nop(); nop(); nop(); nop(); nop(); nop(); nop(); nop();
217 }
218
```

## 5.6 サンプルプログラムリスト "main.c" (6)

```
219  /*"FUNC COMMENT"*****
220  * ID      :
221  * Outline  : フラッシュメモリの書き込み/消去許可状態チェック
222  *-----
223  * Include  : "iodefine.h"
224  *-----
225  * Declaration : int ocf_pe_chk(void);
226  *-----
227  * Description : フラッシュコードコントロール/ステータスレジスタ (FCCS)の
228  *              : FWE ビット値をリードし、戻り値として返却します。
229  *-----
230  * Argument   : void
231  *-----
232  * Return Value : ・0 ; フラッシュ書き込み/消去禁止状態
233  *              : ・1 ; フラッシュ書き込み/消去許可状態
234  *-----
235  * Note       : None
236  *"FUNC COMMENT END"*****/
237 int ocf_pe_chk(void)
238 {
239     return FLASH.FCCS.BIT.FWE;
240 }
241
242 /*"FUNC COMMENT"*****
243 * ID      :
244 * Outline  : フラッシュ書き込み/消去処理
245 *-----
246 * Include  : "Flash_API_SH7280.h"
247 *-----
248 * Declaration : void ocf_write(void);
249 *-----
250 * Description : ホスト PC より指定された書き込み/消去先のブロック (EB0 以外)を
251 *              : 消去した後、書き込み先の先頭アドレスから指定サイズ分のデータを
252 *              : フラッシュに書き込みます。
253 *-----
254 * Argument   : void
255 *-----
256 * Return Value : void
257 *-----
258 * Note       : None
259 *"FUNC COMMENT END"*****/
260 void ocf_write(void)
261 {
262     unsigned char error;          /* Function return value */
263     unsigned char RcvData;        /* Receive data */
264     unsigned char EraseBlkNum;    /* Erase block number */
265     unsigned short EraseBlkSelect; /* Specified erase block number by bit field */
266     unsigned long WriteAddr;      /* Start address to be programmed */
267     unsigned long WriteSize;      /* Data size to be programmed */
268     unsigned long RcvSize;        /* Receiving size for data to be programmed */
269     unsigned long i;              /* Loop counter */
270 }
```

## 5.7 サンプルプログラムリスト "main.c" (7)

```
271     /* ==== 送信要求 ==== */
272     io_sci_snd(RET_REQ);
273
274     /* ==== フラッシュ消去処理コマンド受信 ==== */
275     io_sci_rcv(&RcvData, 1);
276     if(RcvData != CMD_ERASE){ /* CMD_ERASE 以外のコマンド受信? */
277         io_sci_snd(RET_NG); /* エラー送信 */
278         while(1){
279             }
280     }
281
282     /* ==== 送信要求 ==== */
283     io_sci_snd(RET_REQ);
284
285     /* ==== 消去ブロック番号データ受信 ==== */
286     io_sci_rcv((unsigned char *)&EraseBlkSelect, 2);
287     if( (EraseBlkSelect == 0x0000) ||
288         ((EraseBlkSelect & 0x0001) != 0) ){
289         /* ブロック番号指定なし or EBO を指定? */
290         io_sci_snd(RET_NG); /* エラー送信 */
291         while(1){
292             }
293     }
294
295     /* ==== フラッシュ消去処理 ==== */
296     EraseBlkNum = BLOCK_1;
297     do{
298         EraseBlkSelect >>= 1;
299         if((EraseBlkSelect & 0x0001) != 0){
300             /* ---- SCI モジュールスタンバイ設定 ---- */
301             io_sci_stop();
302             /* ---- ブロック消去 ---- */
303             error = R_FlashErase((uint8_t)EraseBlkNum);
304             /* ---- SCI 初期化 ---- */
305             io_sci_init();
306             if(error != 0){ /* 消去エラー発生? */
307                 io_sci_snd(RET_NG); /* エラー送信 */
308                 while(1){
309                     }
310             }
311         }
312     } while(EraseBlkNum++ <= BLOCK_15);
313
314     io_sci_snd(RET_OK); /* 正常終了 */
315
316
```

## 5.8 サンプルプログラムリスト "main.c" (8)

```
317     /* ==== フラッシュ書き込み処理 ==== */
318     while(1){
319         io_sci_rcv(&RcvData, 1);          /* ユーザ制御コマンド受信 */
320         if(RcvData == CMD_WRITE){       /* CMD_WRITE 受信? */
321             io_sci_snd(RET_REQ);        /* 送信要求 */
322         }
323         else if(RcvData == CMD_WEND){   /* CMD_WEND 受信? */
324             io_sci_snd(RET_OK);         /* 正常終了 */
325             break;
326         }
327         else{
328             io_sci_snd(RET_NG);         /* エラー送信 */
329             while(1){
330                 }
331         }
332
333         /* ---- 書き込み先の先頭アドレス受信 ---- */
334         io_sci_rcv((unsigned char *)&WriteAddr, 4);
335
336         /* ---- 書き込みデータサイズ受信 ---- */
337         io_sci_rcv((unsigned char *)&WriteSize, 4);
338
339         while(WriteSize > 0){
340             io_sci_snd(RET_REQ);        /* 送信要求 */
341
342             if(WriteSize > PROGRAM_SIZE){
343                 RcvSize = PROGRAM_SIZE;
344             }
345             else{
346                 RcvSize = WriteSize;
347                 for(i = RcvSize; i < PROGRAM_SIZE; i++){
348                     WriteBuff[i] = 0xff;
349                 }
350             }
351
352             /* ---- 書き込みデータ受信 ---- */
353             io_sci_rcv(WriteBuff, RcvSize); /* 書き込みデータバッファに格納 */
354
355             /* ---- SCI モジュールスタンバイ設定 ---- */
356             io_sci_stop();
357             /* ---- フラッシュ書き込み ---- */
358             error=R_FlashWrite((uint32_t)WriteAddr, (uint32_t)WriteBuff, PROGRAM_SIZE);
359             /* ---- SCI 初期化 ---- */
360             io_sci_init();
361             if(error != 0){              /* 書き込みエラー発生? */
362                 io_sci_snd(error);      /* エラー送信 */
363                 while(1){
364                     }
365             }
366         }

```

## 5.9 サンプルプログラムリスト "main.c" (9)

```
367         WriteAddr += PROGRAM_SIZE;
368         WriteSize -= RcvSize;
369     }
370     io_sci_snd(RET_OK);          /* 正常終了 */
371 }
372
373 }
374
375 /*"FUNC COMMENT"*****
376 * ID          :
377 * Outline     : フラッシュ読み出し処理
378 *-----
379 * Include     :
380 *-----
381 * Declaration : void ocf_read(void);
382 *-----
383 * Description : 読み出し先の先頭アドレスから指定サイズのデータを読み出して
384 *               : ホストPCへ送信します。
385 *-----
386 * Argument    : void
387 *-----
388 * Return Value : void
389 *-----
390 * Note        : None
391 *"FUNC COMMENT END"*****/
392 void ocf_read(void)
393 {
394     unsigned char *ReadData;    /* Pointer for readout data */
395     unsigned long ReadAddr;     /* Start address to be read */
396     unsigned long ReadSize;     /* Reading size */
397     unsigned long i;           /* Loop counter */
398
399     /* ==== 送信要求 ==== */
400     io_sci_snd(RET_REQ);
401
402     /* ==== 読み出し開始アドレス受信 ==== */
403     io_sci_rcv((unsigned char *)&ReadAddr, 4);
404     if(ReadAddr >= 0x00100000){
405         /* ユーザマット以外のアドレスを指定? */
406         io_sci_snd(RET_NG);     /* エラー送信 */
407         while(1){
408             }
409     }
410
411     /* ==== 読み出しデータサイズ受信 ==== */
412     io_sci_rcv((unsigned char *)&ReadSize, 4);
413
```



## 5.10 サンプルプログラムリスト "main.c" (10)

```
414     /* ==== ROMから読み出したデータを送信 ==== */
415     ReadData = (unsigned char *)ReadAddr;
416     for(i = 0; i < ReadSize; i++){
417         io_sci_snd(*ReadData++);
418     }
419 }
420
421 /*"FUNC COMMENT"*****
422 * ID      :
423 * Outline : 割り込み処理(ダミー関数)
424 *-----
425 * Include :
426 *-----
427 * Declaration : void dummy_f(void);
428 *-----
429 * Description :
430 *-----
431 * Argument   : void
432 *-----
433 * Return Value : void
434 *-----
435 * Note       : None
436 *"FUNC COMMENT END"*****/
437 #pragma interrupt dummy_f
438 void dummy_f(void)
439 {
440     while(1){
441         /* Infinite loop */
442     }
443 }
444
445 /* End of File */
```

## 6. 参考ドキュメント

- ソフトウェアマニュアル  
SH-2A、SH2A-FPU ソフトウェアマニュアル Rev.3.00  
(最新版をルネサス エレクトロニクス ホームページから入手してください。)
- ハードウェアマニュアル  
SH7280 グループ ハードウェアマニュアル Rev.2.00  
(最新版をルネサス エレクトロニクス ホームページから入手してください。)
- テクニカルアップデート  
ユーザプログラムモードでの書き込み/消去処理中の割り込み禁止 (TN-SH7-A657A/J)

## ホームページとサポート窓口

ルネサス エレクトロニクス ホームページ

<http://japan.renesas.com/>

お問い合わせ先

<http://japan.renesas.com/inquiry>

すべての商標および登録商標は、それぞれの所有者に帰属します。

## 改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2010.09.10	—	初版発行

## 製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

### 1. 未使用端子の処理

【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。

CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

### 2. 電源投入時の処置

【注意】電源投入時は、製品の状態は不定です。

電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。

同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

### 3. リザーブアドレスのアクセス禁止

【注意】リザーブアドレスのアクセスを禁止します。

アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

### 4. クロックについて

【注意】リセット時は、クロックが安定した後、リセットを解除してください。

プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

### 5. 製品間の相違について

【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。

同じグループのマイコンでも型名が違っていると、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。

## 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て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、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とその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。



ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

<http://www.renesas.com>

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社 〒100-0004 千代田区大手町2-6-2 (日本ビル)

(03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。  
総合お問合せ窓口：<http://japan.renesas.com/inquiry>