RZ/A1LU グループ

SPI マルチ I/O バスコントローラを使用した シリアルフラッシュメモリへのライト例

R01AN3201JJ0100
Rev.1.00
2017.5.17

要旨

本アプリケーションノートは、RZ/A1LUの SPI マルチ I/O バスコントローラ（以下、SPIBSC とします）を使用して、シリアルフラッシュメモリにライトのアクセスを行う例について説明します。

対象デバイス

RZ/A1LU

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

目次

1. 仕様

ブートモード 1（シリアルフラッシュブート）で SPI マルチ I/O バス空間に配置されたシリアルフラッシュメモリからブート後、SPIBSC を SPI 動作モードに切り替え、シリアルフラッシュメモリに対してコマンドを発行し、リード/ライトアクセス等の処理を行います。

本サンプルコードでは、SPIBSC の設定とともに、メモリ管理ユニット、1 次キャッシュ（L1 キャッシュ）、2 次キャッシュ（L2 キャッシュ）のメンテナンス処理を行います。

本アプリケーションノートでは、FIFO 内蔵シリアルコミュニケーションインタフェースを SCIF、汎用入出力ポートを PORT、低消費電力モードを STB、メモリ管理ユニットを MMU とします。

表 1.1に使用する周辺機能と用途を、図 1.1にサンプルコード実行時の動作環境を示します。

表1.1　使用する周辺機能と用途

周辺機能	用途
SPI マルチ I/O バスコントローラ（SPIBSC）	外部アドレス空間リードモードと SPI 動作モードを切り替える制御を行い、シリアルフラッシュメモリのリードライトに使用
FIFO 内蔵シリアルコミュニケーション インタフェース（SCIF）	SCIF チャネル 0 を用いて、ホスト PC との通信用として使用
汎用入出力ポート（PORT）	SPIBSC、SCIF チャネル 0 の兼用端子の切り替えに使用
低消費電力モード（STB）	SPIBSC のモジュールスタンバイを解除するために使用（注）
メモリ管理ユニット（MMU）、 L1 キャッシュ、L2 キャッシュ	SPIBSC の動作モードを切り替え時に、MMU の変換テーブルの操作、L1 キャッシュおよび L2 キャッシュのメンテナンス処理に使用

【注】　シリアルフラッシュブート時に実行されるブート起動用内蔵 ROM プログラムにより、SPIBSC のモジュールスタンバイを解除しています。

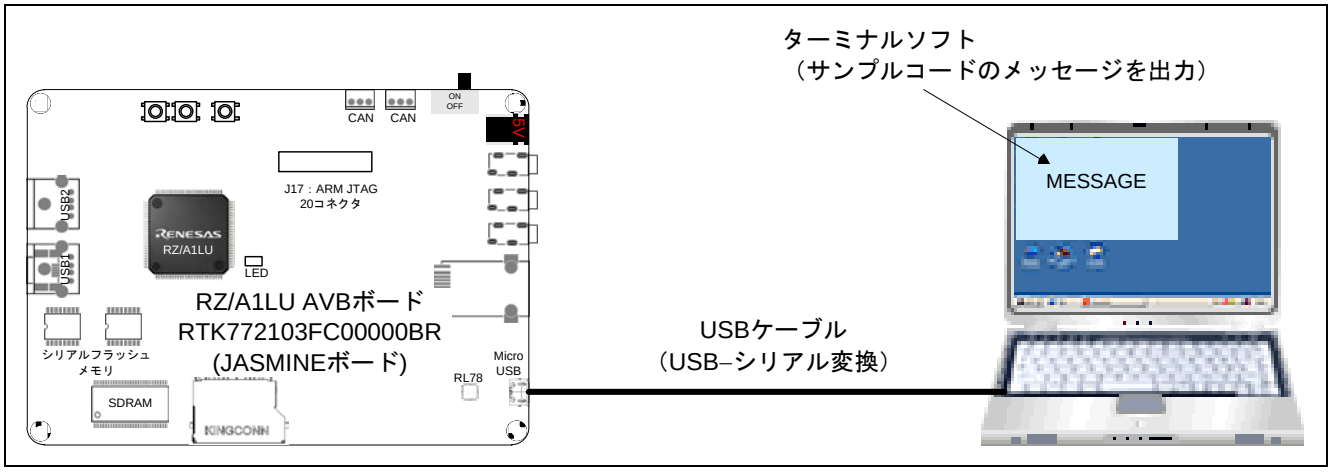


図1.1　動作環境

2. 動作確認条件

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表2.1 動作確認条件

項目	内容
使用マイコン	RZ/A1LU
動作周波数	CPU クロック（Iφ）：400MHz 内部バスクロック（Bφ）：133.33MHz 周辺クロック（P1φ）：66.67MHz 周辺クロック（P0φ）：33.33MHz
動作電圧	電源電圧（I/O）：3.3V 電源電圧（内部）：1.18V
統合開発環境	ARM®統合開発環境 ARM Development Studio 5（DS-5™）Version 5.24
C コンパイラ	ARM C/C++ Compiler/Linker/Assembler Ver.5.06 update 2 [Build 183]
	コンパイラオプション -O3 -Ospace --cpu=Cortex-A9 --littleend --arm --apcs=/interwork --no_unaligned_access --fpu=vfpv3_fp16 -g --asm
動作モード	ブートモード 1 （シリアルフラッシュブート）
使用ボード	RZ/A1LU AVB ボード RTK772103FC00000BR（以下、JASMINE ボードとします）
ターミナルソフトの通信設定	• 通信速度：115200bps • データ長：8 ビット • パリティ：なし • ストップビット長：1 ビット • フロー制御：なし
使用デバイス （ボード上で使用する機能）	• シリアルフラッシュメモリ（SPI マルチ I/O バス空間に接続） - メーカ：Macronix社 - 型名：MX25L51245G • RL78/G1C（USB 通信とシリアル通信を変換し、ホスト PC との通信に使用） • LED1

【注】 クロックモード 0（EXTAL 端子からの 13.33MHz のクロック入力）で使用時の動作周波数です。

3. 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。

- RZ/A1H グループ レジスタ定義ヘッダ・ファイル iodefine.h（R01AN1860JJ）
- RZ/A1H グループ 初期設定例（R01AN1864JJ）
- RZ/A1LU グループ シリアルフラッシュメモリからのブート例（R01AN3093JJ）

4. 周辺機能説明

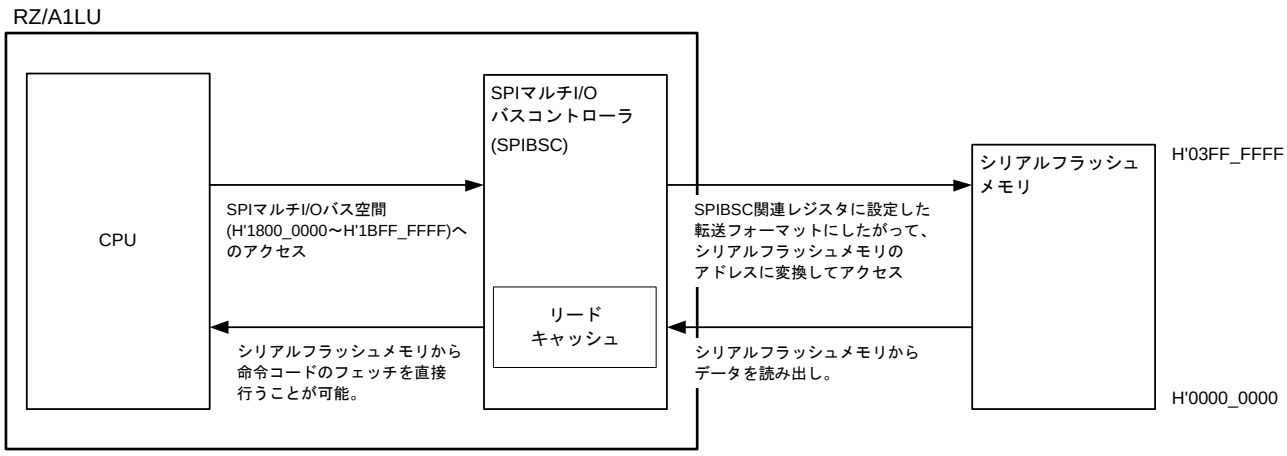
RZ/A1LUの SPIBSC の動作モードについて説明します。詳細については、「RZ/A1L グループ、RZ/A1LU グループ、RZ/A1LC グループ ユーザーズマニュアル ハードウェア編」を参照してください。

表 4.1に示すように、SPIBSC は外部アドレス空間リードモードと SPI 動作モードの 2 つのモードに対応しています。2 つのモードを切り替えてシリアルフラッシュメモリにアクセスすることで、リードアクセス、イレーズアクセス、ライトアクセス、およびシリアルフラッシュメモリのレジスタへのアクセスが可能です。

表4.1　SPIBSC を使用したシリアルフラッシュメモリへのアクセス

動作モード	概要
外部アドレス空間リードモード （CMNCR.MD = 0）	• SPI マルチ I/O バス空間（H'1800_0000～H'1BFF_FFFF）へのリードアクセスで使用。 • バスマスタがシリアルフラッシュメモリから直接データを参照可能（CPU から命令コードの直接フェッチが可能）。 • 直接リードアクセスが可能な範囲は SPI マルチ I/O バス空間の 64MB まで（64MB を超えるアクセスは SPIBSC のレジスタ制御が必要）。
SPI 動作モード （CMNCR.MD = 1）	• ソフトウェアにて SPIBSC 関連レジスタの設定を行うことにより、シリアルフラッシュメモリに任意のコマンド発行が可能。 • イレーズアクセス、ライトアクセス、およびシリアルフラッシュメモリのレジスタ（Status Register, Configuration Register など）をアクセスする場合に使用。 • シリアルフラッシュ 1 個接続時は 4GB、2 個接続時は 8GB の空間へのリードアクセスが可能。 • SPI 動作モードに設定時、SPI マルチ I/O バス空間へのアクセスは不可。

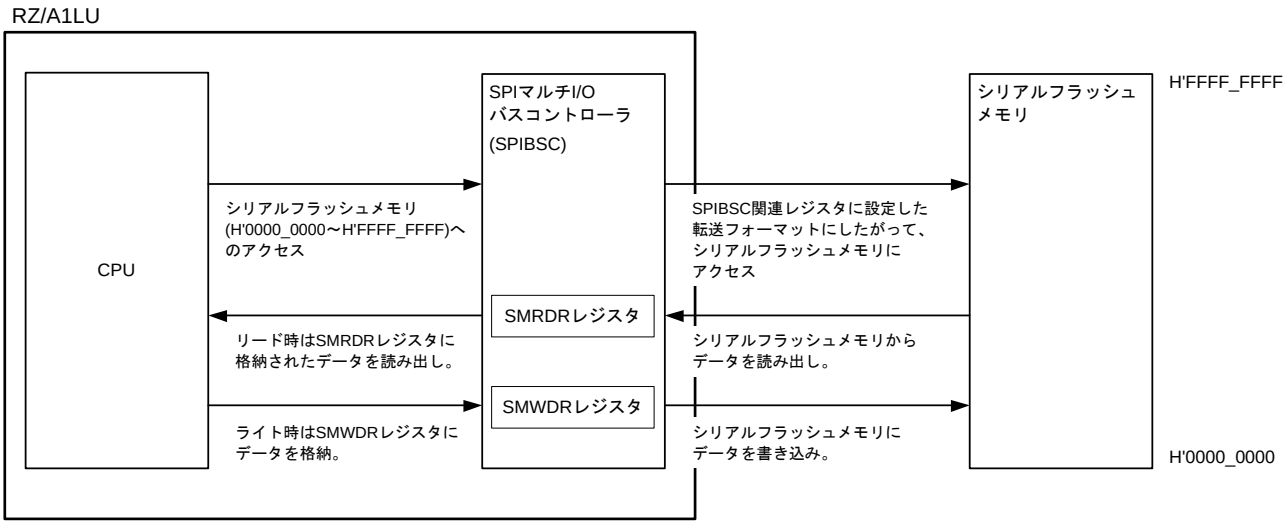
外部アドレス空間リードモードでは､シリアルフラッシュメモリが接続された SPI マルチ I/O バス空間へのリードアクセスを SPI 通信（リードコマンド発行、アドレス出力、ダミーサイクル出力）に変換しデータを受信することで、シリアルフラッシュメモリに配置された命令を直接実行することが可能で、RZ/A1LUではブートデバイスにシリアルフラッシュメモリを使用するシリアルフラッシュブート（ブートモード 1）で使用しています。



【注】 外部アドレス空間リードモードでは、データリードコントロールレジスタ（DRCR）の RBE ビットを１に設定することで、リードキャッシュにシリアルフラッシュメモリのデータをキャッシングすることが可能です。CPU が要求したアドレスのデータがリードキャッシュに格納されている場合は、外部のシリアルフラッシュメモリにはアクセスは行わず、リードキャッシュ内のデータを CPU に返します。

図4.1　外部アドレス空間リードモードでのシリアルフラッシュメモリへのアクセス

SPI 動作モードは、シリアルフラッシュメモリのレジスタ（ステータスレジスタ、コンフィグレーションレジスタなど）へのアクセスまたはデータ書き込み時のライトアクセス（ライトコマンド発行、アドレス出力、ライトデータ出力）に使用します。ストレージデバイスとしてシリアルフラッシュメモリを使用する場合は、SPI 動作モードでシリアルフラッシュメモリへのアクセスを行います。



【注】 SPI 動作モードではシリアルフラッシュメモリの 4G バイト（2 個接続時は 8G バイト）の領域にアクセス可能ですが、外部アドレス空間リードモードでは SPI マルチ I/O バス空間として割り当てられた 64MB の領域（H'1800_0000～H'1BFF_FFFF 番地）の直接アクセスが可能です。

図4.2　SPI 動作モードでのシリアルフラッシュメモリへのアクセス

5. ハードウェア説明

5.1 ハードウェア構成例

図 5.1にブートモード 1 にてシリアルフラッシュメモリ（MX25L51245G）からブートし、シリアルフラッシュメモリにリード/ライトアクセスする場合の接続例を示します。

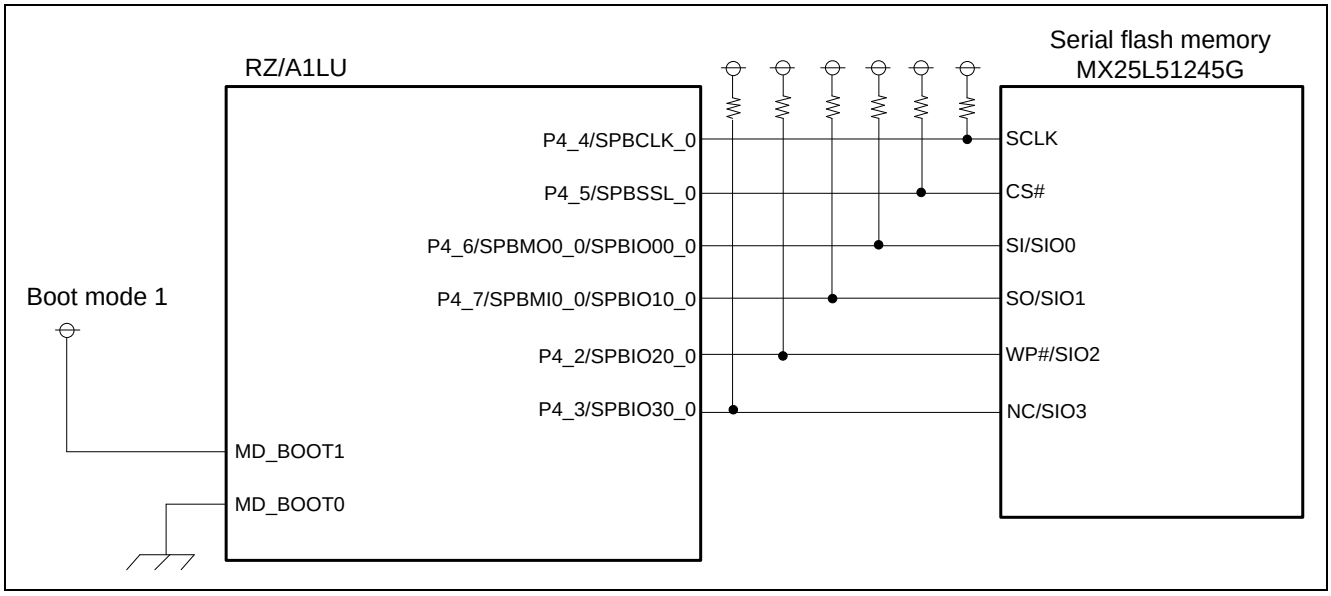


図5.1 接続例

5.2 使用端子一覧

表 5.1に使用端子と機能を示します。

表5.1 使用端子と機能

端子名	入出力	内容
SPBCLK_0	出力	クロック出力
SPBSSL_0	出力	スレーブセレクト
SPBMO0_0/SPBIO00_0	入出力	マスタ送出データ／データ 0
SPBMI0_0/SPBIO10_0	入出力	マスタ入力データ／データ 1
SPBIO20_0	入出力	データ 2
SPBIO30_0	入出力	データ 3
MD_BOOT1	入力	ブートモードの選択 MD_BOOT1："H"、MD_BOOT0："L" （ブートモード 1 に設定） ブート後に MD_BOOT0 は、RxD0 としてシリアル受信データ信号の入力機能に切り替えて使用（注）
MD_BOOT0/RxD0	入力	
P8_12	出力	LED1 の点灯および消灯
TxD0	出力	シリアル送信データ信号

【注】 P0_0 端子に、MD_BOOT0 と RxD0 の機能がマルチプレクスされており、パワーオンリセット解除時は MD_BOOT0 機能として動作し、ブートモードを決定する端子として使用します。サンプルコードでは、JASMINE ボード上のマルチプレクサ/デマルチプレクサ（SN74CB3Q3257）の入力端子の選択制御を行うことにより、パワーオンリセット解除時はボード上のスイッチからブート機能の選択信号を入力し、パワーオンリセット解除後に端子のプルアップ処理を行い RxD0 機能として動作するように設定しています。

6. ソフトウェア説明

6.1 動作概要

本サンプルコードは、SPIBSC の外部アドレス空間リードモードを使用して、シリアルフラッシュブートでプログラムを起動後、ターミナルから入力されたコマンド文字列にしたがって、シリアルフラッシュメモリに対してイレーズ、ライト、リードのアクセスの処理を実行します。サンプルコードで準備しているコマンドについては「6.6 サンプルコマンド」参照してください。

シリアルフラッシュブートで起動する場合には、SPIBSC を外部アドレス空間リードモードに設定することで、SPI マルチ I/O バス空間に接続されたシリアルフラッシュメモリに格納されたプログラムを直接実行できるようにしています。シリアルフラッシュメモリにイレーズまたはライトのアクセスを行う場合、SPIBSC を SPI 動作モードに切り替えて SPIBSC のレジスタを設定し、シリアルフラッシュメモリに対してコマンドを発行することにより処理を実現します。

6.2 SPIBSC を使用したシリアルフラッシュメモリへのアクセス

SPIBSC を SPI 動作モードに設定している場合は SPI マルチ I/O バス空間へのアクセスはできないため、SPI マルチ I/O バス空間に配置されているコードやデータにアクセスしないようにする必要があります。

SPI 動作モードへ設定後に実行が必要な処理は、SPI マルチ I/O バス空間ではなく、別のメモリ空間に転送して処理を行う必要があります。割り込み処理を使用する場合には、例外ベクタおよび使用する割り込みのハンドラ処理が SPI マルチ I/O バス空間以外のメモリ空間に配置するように注意してください。

また、ARM 社 Cortex-A9 プロセッサは、処理の高速化のために、分岐予測/投機実行をサポートしています。条件分岐命令がある場合に、分岐予測/投機実行により、条件が確定する前に分岐先の命令のフェッチやデータのアクセスが発生し、予測が外れた場合に発生しないアクセスが行われる可能性があります。外部アドレス空間リードモードでプログラムを実行後に、シリアルフラッシュメモリのイレーズまたはライトアクセスなどの処理を行うために、SPI 動作モードに切り替えた場合、分岐予測/投機実行により、SPI 動作モード中に SPI マルチ I/O バス空間へのアクセスが発生し、正常に動作しない可能性があります。本サンプルコードでは、SPIBSC の動作モードを切り替え時に、SPI マルチ I/O バス空間の MMU の変換テーブルで「アクセス不可/可能」、「実行不可/可能」の属性を設定し、SPI マルチ I/O バス空間への不正なアクセスが発生しないようにしています。この処理の詳細については、「6.3 外部アドレス空間リードモードと SPI 動作モードの切り替え」で説明します。

6.3 外部アドレス空間リードモードと SPI 動作モードの切り替え

以下の手順で、SPIBSC を外部アドレス空間リードモードと SPI 動作モードを切り替えます。

なお、SPIBSC の動作モードの切り替え処理は、SPI マルチ I/O バス空間に配置されたシリアルフラッシュメモリでは実行できないため、別のメモリ空間で実行する必要があります。サンプルコードでは、6.3.1および6.3.2の処理は大容量内蔵 RAM に転送し、大容量内蔵 RAM で処理を実行するようにしています。

MMU 変換テーブルを変更する手順については、ARM 社より提供される「ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C」の「TLB maintenance operations and the memory order model」の内容を合わせて参照してください。

6.3.1 外部アドレス空間リードモードから SPI 動作モードへの切り替え

以下に SPIBSC の SPI 動作モードへの切り替え手順を示します。

1. SPI マルチ I/O バス空間に対応する MMU 変換テーブルを変更
   SPI マルチ I/O バス空間の MMU の変換テーブルの設定を、アクセス不可、実行不可の属性に設定し、Cortex-A9 の投機実行が行われる場合でも SPI マルチ I/O バス空間へのアクセスが発生しないようにします。
2. L1 データキャッシュのクリーニング（書き戻し）とインバリデート（無効化）処理
   キャッシュに取り込まれた SPI マルチ I/O バス空間のデータと MMU の変換テーブルの内容を無効化するために、L1 データキャッシュのクリーニングとインバリデートを行います。
3. DSB 命令の実行
   上記2の L1 データキャッシュのクリーニングの完了を保証するために実行します。
4. L2 キャッシュのクリーニング（書き戻し）およびインバリデート（無効化）処理
   キャッシュに取り込まれた SPI マルチ I/O バス空間の命令およびデータを無効化するために、L2 キャッシュのクリーニングとインバリデートを行います。
5. Cache Sync の実行
   上記4の L2 キャッシュのメンテナンス処理の完了を保証するために実行します。
6. TLB のインバリデート（無効化）
   MMU の変換テーブルの設定を変更したため、変換ルックサイドバッファ（TLB）をインバリデートします。

7. L1 命令キャッシュのインバリデート（無効化）
   MMU の変換テーブルの設定を変更したため、L1 命令キャッシュをインバリデートします。
8. DSB 命令の実行
   上記7の L1 命令キャッシュのインバリデート処理の完了を保証するために実行します。
9. ISB 命令の実行
   これ以降の命令が書き換え後の MMU 変換テーブルを参照することを保証するために実行します。
10. SPIBSC を外部アドレス空間リードモードから SPI 動作モードに切り替え
    共通ステータスレジスタ（CMNSR）の TEND ビットをリードしてシリアルフラッシュメモリへのアクセスが発生していないことを確認します。共通コントロールレジスタ（CMNCR）の MD ビットに 1 をライトして、CMNCR レジスタをダミーリードし、SPI 動作モードに切り替えます。

【注】 外部アドレス空間リードモードから SPI 動作モードに切り替え後に SPI マルチ I/O バス空間へのアクセスが発生することがないように、割り込みが許可されている場合には、割り込み処理をシリアルフラッシュメモリから別のメモリ空間に転送して実行するように対応いただく必要があります。本サンプルコードでは、SPI 動作モードに切り替える前に割り込みを禁止し、外部アドレス空間リードモードに切り替える前に割り込みを許可するようにして、シリアルフラッシュメモリへのアクセスが発生しないようにしています。

6.3.2 SPI 動作モードから外部アドレス空間リードモードへの切り替え

以下に SPIBSC の外部アドレス空間リードモードへの切り替え手順を示します。

1. SPIBSC を SPI 動作モードから外部アドレス空間リードモードに切り替え
   共通ステータスレジスタ（CMNSR）の TEND ビットをリードしてシリアルフラッシュメモリへのアクセスが発生していないことを確認します。データリードコントロールレジスタ（DRCR）の RCF ビットに 1 をライトして、DRCR レジスタをダミーリードし、SPIBSC のリードキャッシュをクリアします。共通コントロールレジスタ（CMNCR）レジスタの MD ビットに 0 をライトして、CMNCR レジスタをダミーリードし、外部アドレス空間リードモードに切り替えます。
2. SPI マルチ I/O バス空間に対応する MMU 変換テーブルを変更
   SPI マルチ I/O バス空間の MMU の変換テーブルの設定を、アクセス可能、実行可能の属性に設定し、SPI マルチ I/O バス空間にアクセスできるようにします。
3. L1 データキャッシュのクリーニング（書き戻し）とインバリデート（無効化）処理
   キャッシュに取り込まれた SPI マルチ I/O バス空間のデータと MMU の変換テーブルの内容を無効化するために、L1 データキャッシュのクリーニングとインバリデートを行います。
4. DSB 命令の実行
   上記3の L1 データキャッシュのクリーニングの完了を保証するために実行します。
5. L2 キャッシュのクリーニングおよびインバリデート処理
   キャッシュに取り込まれた SPI マルチ I/O バス空間の命令およびデータを無効化するために、L2 キャッシュのクリーニングとインバリデートを行います。
6. Cache Sync の実行
   上記5の L2 キャッシュのメンテナンス処理の完了を保証するために実行します。
7. TLB のインバリデート（無効化）
   MMU の変換テーブルの設定を変更したため、変換ルックサイドバッファ（TLB）をインバリデートします。
8. L1 命令キャッシュのインバリデート（無効化）
   MMU の変換テーブルの設定を変更したため、L1 命令キャッシュをインバリデートします。
9. DSB 命令の実行
   上記8の L1 命令キャッシュのインバリデート処理の完了を保証するために実行します。
10. ISB 命令の実行
    これ以降の命令が書き換え後の MMU 変換テーブルを参照することを保証するために実行します。

【注】 1. **上記の5および6の処理は以下の両方の条件を満たす場合には実施する必要はありません。**
MMU の変換テーブルが外部メモリではなく大容量内蔵 RAM に配置されている場合。
外部アドレス空間リードモードから SPI 動作モードに切り替え時の処理で、5および6の処理を実施しており、キャッシュとシリアルフラッシュメモリのコヒーレンシが保証されている場合。
2. **6.3.1で割り込みを禁止にしている場合は、上記以降の処理でご使用の割り込みを許可にしてください。**

6.4 SPIBSC の動作モード切り替え処理の実装方法

ここでは、「6.3 外部アドレス空間リードモードと SPI 動作モードの切り替え」に記載している各処理の実装方法を示します。

6.4.1 外部アドレス空間リードモードから SPI 動作モードへの切り替え処理の実装方法

1. MMU 変換テーブルの変更処理の実装方法

外部アドレス空間リードモードから SPI 動作モードに切り替え時の MMU 変換テーブルの変更処理の実装例を示します。サンプルコードでは、Change_MMU_TTbl_SpibscSpi 関数に実装しており、図 6.1および図 6.2に処理の抜粋を示します。Change_MMU_TTbl_SpibscSpi 関数の引数として、SPI マルチ I/O バス空間の開始アドレスおよび終了アドレスを、start_addr および end_addr にそれぞれ指定します。

```
static int32_t Change_MMU_TTbl_SpibscSpi(uint32_t start_addr, uint32_t end_addr)
{
    uint32_t index_start;         /* Start address table index */
    uint32_t index_end;           /* End address table index */
    uint32_t index;               /* Loop variable: table index */
    mmu_ttbl_desc_section_t desc; /* Loop variable: descriptor */

    index_start = MMU_TTbl_GetIndex(start_addr); /* Get start address table index */
    index_end = MMU_TTbl_GetIndex(end_addr);     /* Get end address table index */

    for(index = index_start; index <= index_end; index++)
    {
        /* Get descriptor from translation table */
        MMU_TTbl_GetDesc(index, &desc);

        /* Modify memory attribute descriptor */
        desc.AP1_0 = 0x0u; /* AP[2:0] = b'000 (No access) */
        desc.AP2 = 0x0u;
        desc.XN = 0x1u;    /* XN = 1 (Execute never)  */

        /* Write descriptor back to translation table */
        MMU_TTbl_SetDesc(index, &desc);
    }

    return 0;
}
```

図6.1 SPI 動作モードに切り替え時の MMU 変換テーブルの変更処理

```
/*  Definition of the short-descriptor translation table first-level descriptor format (section) */
typedef struct {
    uint32_t b0        : 1 ;   /* bit 0       : -        (0)                                    */
    uint32_t b1        : 1 ;   /* bit 1       : -        (1)                                    */
    uint32_t B         : 1 ;   /* bit 2       : B        Memory region attribute bit            */
    uint32_t C         : 1 ;   /* bit 3       : C        Memory region attribute bit            */
    uint32_t XN        : 1 ;   /* bit 4       : XN       Execute-never bit                      */
    uint32_t Domain    : 4 ;   /* bit 8-5     : Domain   Domain field                           */
    uint32_t b9        : 1 ;   /* bit 9       : IMP      IMPLEMENTATION DEFINED                 */
    uint32_t AP1_0     : 2 ;   /* bit 11-10   : AP[1:0]  Access permissions bits:bit1-0         */
    uint32_t TEX       : 3 ;   /* bit 14-12   : TEX[2:0] Memory region attribute bits           */
    uint32_t AP2       : 1 ;   /* bit 15      : AP[2]    Access permissions bits:bit2           */
    uint32_t S         : 1 ;   /* bit 16      : S        Shareable bit                          */
    uint32_t nG        : 1 ;   /* bit 17      : nG       Not global bit                         */
    uint32_t b18       : 1 ;   /* bit 18      : -        (0)                                    */
    uint32_t NS        : 1 ;   /* bit 19      : NS       Non-secure bit                         */
    uint32_t base_addr : 12;   /* bit 31-20   : PA[31:20] PA(physical address) bits:bit31-20    */
} mmu_ttbl_desc_section_t;

/* Definition of the base address for the MMU translation table */
#define TTB    ((uint32_t)&Image$$TTB$$ZI$$Base)     /* using linker symbol */

uint32_t MMU_TTbl_GetIndex(uint32_t addr)
{
    uint32_t index;
    index = addr >> 20;
    return index;
}

int32_t MMU_TTbl_GetDesc(uint32_t index, mmu_ttbl_desc_section_t * pdesc)
{
    mmu_ttbl_desc_section_t * table;

    /* ==== Get descriptor value from translation table ==== */
    table = (mmu_ttbl_desc_section_t *)TTB;
    *pdesc = table[index];

    return 0;
}

int32_t MMU_TTbl_SetDesc(uint32_t index, mmu_ttbl_desc_section_t * pdesc)
{
    mmu_ttbl_desc_section_t * table;

    /* ==== Set descriptor value in translation table ==== */
    table = (mmu_ttbl_desc_section_t *)TTB;
    table[index] = *pdesc;

    return 0;
}
```

図6.2　MMU 変換テーブルの設定処理

2. L1 データキャッシュのクリーニングとインバリデート処理の実装方法

図 6.3および図 6.4にL1 データキャッシュのクリーニングとインバリデート処理の実装例を示します。サンプルコードでは、L1_D_CacheWritebackFlushAll 関数に実装しています。

```
void L1_D_CacheWritebackFlushAll(void)
{
    /* ==== Invalidate and clean all D cache by set/way ==== */
    L1_D_CacheOperationAsm(L1CACHE_WB_FLUSH);     /* L1CACHE_WB_FLUSH == 2 */
}


;*****************************************************************************
; Function Name: L1_D_CacheOperationAsm
; Description  : r0 = 0 : DCISW. Invalidate data or unified cache line by set/way.
;              : r0 = 1 : DCCSW. Clean data or unified cache line by set/way.
;              : r0 = 2 : DCCISW. Clean and Invalidate data or unified cache line by set/way.
;*****************************************************************************
L1_D_CacheOperationAsm FUNCTION

    PUSH {r4-r11}

    MRC  p15, 1, r6, c0, c0, 1     ;;; Read CLIDR
    ANDS r3, r6, #0x07000000       ;;; Extract coherency level
    MOV  r3, r3, LSR #23           ;;; Total cache levels << 1
    BEQ  Finished                  ;;; If 0, no need to clean

    MOV  r10, #0                   ;;; R10 holds current cache level << 1
Loop1
    ADD  r2, r10, r10, LSR #1      ;;; R2 holds cache "Set" position
    MOV  r1, r6, LSR r2            ;;; Bottom 3 bits are the Cache-type for this level
    AND  r1, r1, #7                ;;; Isolate those lower 3 bits
    CMP  r1, #2
    BLT  Skip                      ;;; No cache or only instruction cache at this level

    MCR  p15, 2, r10, c0, c0, 0    ;;; Write the Cache Size selection register (CSSELR)
    ISB                            ;;; ISB to sync the change to the CacheSizeID reg
    MRC  p15, 1, r1, c0, c0, 0     ;;; Reads current Cache Size ID register (CCSIDR)
    AND  r2, r1, #7                ;;; Extract the line length field
    ADD  r2, r2, #4                ;;; Add 4 for the line length offset (log2 16 bytes)
    LDR  r4, =0x3FF
    ANDS r4, r4, r1, LSR #3        ;;; R4 is the max number on the way size (right aligned)
    CLZ  r5, r4                    ;;; R5 is the bit position of the way size increment
    LDR  r7, =0x7FFF
    ANDS r7, r7, r1, LSR #13       ;;; R7 is the max number of the index size (right aligned)
Loop2
    MOV  r9, r4                    ;;; R9 working copy of the max way size (right aligned)
```

図6.3　L1 データキャッシュのクリーニングとインバリデート処理

```
Loop3
    ORR  r11, r10, r9, LSL r5      ;;; Factor in the Way number and cache number into R11
    ORR  r11, r11, r7, LSL r2      ;;; Factor in the Set number
    CMP  r0, #0
    BNE  Dccsw
    MCR  p15, 0, r11, c7, c6, 2    ;;; Invalidate by Set/Way (DCISW)
    B    Count
Dccsw
    CMP  r0, #1
    BNE  Dccisw
    MCR  p15, 0, r11, c7, c10, 2   ;;; Clean by set/way (DCCSW)
    B    Count
Dccisw
    MCR  p15, 0, r11, c7, c14, 2   ;;; Clean and Invalidate by set/way (DCCISW)
Count
    SUBS r9, r9, #1                ;;; Decrement the Way number
    BGE  Loop3
    SUBS r7, r7, #1                ;;; Decrement the Set number
    BGE  Loop2
Skip
    ADD  r10, r10, #2              ;;; increment the cache number
    CMP  r3, r10
    BGT  Loop1

Finished
    DSB
    POP  {r4-r11}
    BX   lr

    ENDFUNC
```

図6.4　L1 データキャッシュのクリーニングとインバリデート処理

3.　DSB 命令の実装方法

L1 データキャッシュのクリーニングとインバリデート処理の完了を保証するための DSB 命令は、L1_D_CacheOperationAsm 関数に実装しています。

4. L2 キャッシュのクリーリングおよびインバリデート処理の実装方法

図 6.5にL2 キャッシュのクリーニングとインバリデート処理の実装例を示します。

```
#define L2CACHE_8WAY   (0x000000FFuL)         /* All entries(8way) in the L2 cache */

void L2CacheWritebackFlushAll(void)
{
    /* ==== Clean and Invalidate all cache by Way ==== */
    L2C.REG7_CLEAN_INV_WAY = L2CACHE_8WAY;    /* Set 1 to Way bits[7:0] of the reg7_clean_inv_way */
    while ((L2C.REG7_CLEAN_INV_WAY &  L2CACHE_8WAY)
                        != 0x00000000uL)      /* Wait until Way bits[7:0] is cleared */
    {
    }

    /* ==== Cache Sync ==== */
    L2C.REG7_CACHE_SYNC = 0x00000000uL;       /* Ensures completion of the operation */
}
```

図6.5　L2 キャッシュのクリーニングとインバリデート処理

5. Cache Sync 処理の実装方法

L2 キャッシュのクリーニングとインバリデート処理の完了を保証するための Cache Sync の処理は、L2CacheWritebackFlushAll 関数に実装しています。

6. TLB のインバリデート処理の実装方法

図 6.6にTLB のインバリデート処理の実装例を示します。サンプルコードでは、TLB_FlushAll 関数に実装しています。

```
TLB_FlushAll    FUNCTION
    MOV  r0,#0
    MCR  p15, 0, r0, c8, c7, 0     ;;; Cortex-A9 I-TLB and D-TLB invalidation (TLBIALL)
    DSB
    ISB

    BX   lr

    ENDFUNC
```

図6.6　TLB のインバリデート処理

7. L1 命令キャッシュのインバリデート処理の実装方法

図 6.7にL1 命令キャッシュのインバリデート処理の実装例を示します。サンプルコードでは、L1_I_CacheFlushAllAsm 関数に実装しています。

```
L1_I_CacheFlushAllAsm FUNCTION

    MOV  r0, #0
    MCR  p15, 0, r0, c7, c5, 0     ;;; ICIALLU
    DSB
    ISB

    BX   lr

    ENDFUNC
```

図6.7　L1 命令キャッシュのインバリデート処理

8. DSB 命令の実装方法

L1 命令キャッシュのインバリデート処理の完了を保証するための DSB 命令は、L1_I_CacheFlushAllAsm 関数に実装しています。

9. ISB 命令の実装方法

これ以降の命令が書き換え後の MMU 変換テーブルを参照することを保証するための ISB 命令は、L1_I_CacheFlushAllAsm 関数に実装しています。

10. SPIBSC を外部アドレス空間リードモードから SPI 動作モードに切り替える実装方法

SPIBSC の共通コントロールレジスタ（CMNCR）レジスタの MD ビットに 1 をライトして、SPI 動作モードに切り替える処理を行います。サンプルコードでは、R_SFLASH_Spimode 関数に実装しています。

6.4.2 SPI 動作モードから外部アドレス空間リードモードへの切り替え処理の実装方法

1. SPIBSC を SPI 動作モードから外部アドレス空間リードモードに切り替える実装方法

   SPIBSC のデータリードコントロールレジスタ（DRCR）の RCF ビットに 1 をライトして、SPIBSC のリードキャッシュをクリアし、共通コントロールレジスタ（CMNCR）レジスタの MD ビットに 0 をライトして、外部アドレス空間リードモードに切り替える処理を行います。サンプルコードでは、R_SFLASH_Exmode 関数に実装しています。

2. MMU 変換テーブルの変更処理の実装方法

   SPI 動作モードから外部アドレス空間リードモードに切り替え時の MMU 変換テーブルの変更処理の実装例を示します。サンプルコードでは、Change_MMU_TTbl_SpibscXip 関数に実装しており、図 6.8に処理の抜粋を示します。Change_MMU_TTbl_SpibscXip 関数の引数として、SPI マルチ I/O バス空間の開始アドレスおよび終了アドレスを、start_addr および end_addr にそれぞれ指定します。MMU 変換テーブルを設定する処理は、MMU_TTbl_GetIndex 関数、MMU_TTbl_GetDesc 関数、および MMU_TTbl_SetDesc 関数で実装しており、図 6.2を参照してください。

```
static int32_t Change_MMU_TTbl_SpibscXip(uint32_t start_addr, uint32_t end_addr)
{
    uint32_t index_start;         /* Start address table index */
    uint32_t index_end;           /* End address table index */
    uint32_t index;               /* Loop variable: table index */
    mmu_ttbl_desc_section_t desc; /* Loop variable: descriptor */

    index_start = MMU_TTbl_GetIndex(start_addr);  /* Get start address table index */
    index_end = MMU_TTbl_GetIndex(end_addr);      /* Get end address table index */

    for(index = index_start; index <= index_end; index++)
    {
        /* Get descriptor from translation table */
        MMU_TTbl_GetDesc(index, &desc);

        /* Modify memory attribute descriptor */
        desc.AP1_0 = 0x3u; /* AP[2:0] = b'011 (Full access) */
        desc.AP2 = 0x0u;
        desc.XN = 0x0u;    /* XN = 0 (Executable)  */

        /* Write descriptor back to translation table */
        MMU_TTbl_SetDesc(index, &desc);
    }


    return 0;
}
```

図6.8　外部アドレス空間リードモードに切り替え時の MMU 変換テーブルの変更処理

3. L1 データキャッシュのクリーニングとインバリデート処理の実装方法

図 6.3および図 6.4にL1 データキャッシュのクリーニングとインバリデート処理の実装例を示しています。サンプルコードでは、L1_D_CacheWritebackFlushAll 関数に実装しています。

4. DSB 命令の実装方法

L1 データキャッシュのクリーニングとインバリデート処理に実装例を示しています。サンプルコードでは、L1_D_CacheWritebackFlushAll 関数に実装しています。

5. L2 キャッシュのクリーリングおよびインバリデート処理の実装方法

図 6.5にL2 キャッシュのクリーニングとインバリデート処理の実装例を示しています。

6. Cache Sync 処理の実装方法

L2 キャッシュのクリーニングとインバリデート処理に実装例を示しています。サンプルコードでは、L2CacheWritebackFlushAll 関数に実装しています。

7. TLB のインバリデート処理の実装方法

図 6.6にTLB のインバリデート処理の実装例を示しています。サンプルコードでは、TLB_FlushAll 関数に実装しています。

8. L1 命令キャッシュのインバリデート処理の実装方法

図 6.7にL1 命令キャッシュのインバリデート処理の実装例を示しています。サンプルコードでは、L1_I_CacheFlushAllAsm 関数に実装しています。

9. DSB 命令の実装方法

L1 命令キャッシュのインバリデート処理に実装例を示しています。サンプルコードでは、L1_I_CacheFlushAllAsm 関数に実装しています。

10. ISB 命令の実装方法

L1 命令キャッシュのインバリデート処理に実装例を示しています。サンプルコードでは、L1_I_CacheFlushAllAsm 関数に実装しています。

6.5 MMU の変換テーブルについて

本サンプルコードでは、MMU の変換テーブルの第 1 レベル記述子の記述子タイプを"セクション"に設定しており、1MB のメモリ領域の情報を 32 ビット幅の記述子を用いて指定します。また、変換テーブル制御レジスタ（TTBCR）および変換テーブルベースレジスタ 0（TTBR0）により、TTBR0 を使用して 16KB（4096 エントリ）の変換テーブルの領域を MMU で管理するように設定し、H'2003_4000 番地～H'2003_7FFF 番地（16KB）を MMU の変換テーブルの格納領域としています。H'0000_0000 番地～H'FFFF_FFFF 番地の 4GB のアドレス空間に対して、1MB 単位でキャッシュの有効/無効、メモリタイプ属性、実行属性、アクセス属性などを MMU の変換テーブルで指定します。

本サンプルコードでは、SPIBSC の動作モードの切り替え時に、SPI マルチ I/O バス空間の MMU の変換テーブルの情報を変更します。図 6.9にサンプルコードの MMU 変換テーブルの設定を示します。SPIBSC を SPI 動作モードに切り替える場合は、MMU 変換テーブルの AP[2:0]ビットを B'000（アクセス不可）に、XN ビットを 1（実行不可）に設定します。SPIBSC を外部アドレス空間リードモードに切り替える場合は、MMU 変換テーブルの AP[2:0]ビットを B'011（アクセス可能）に、XN ビットを 0（実行可能）に設定します。なお、マネージャとして設定されたドメインに対しては、XN ビットの属性チェックや AP[2:0]ビットの設定によるアクセスのチェックが行われません。MMU 変換テーブルのエントリの設定によってアクセスを保護するためには、ドメインはクライアントに設定する必要があります。

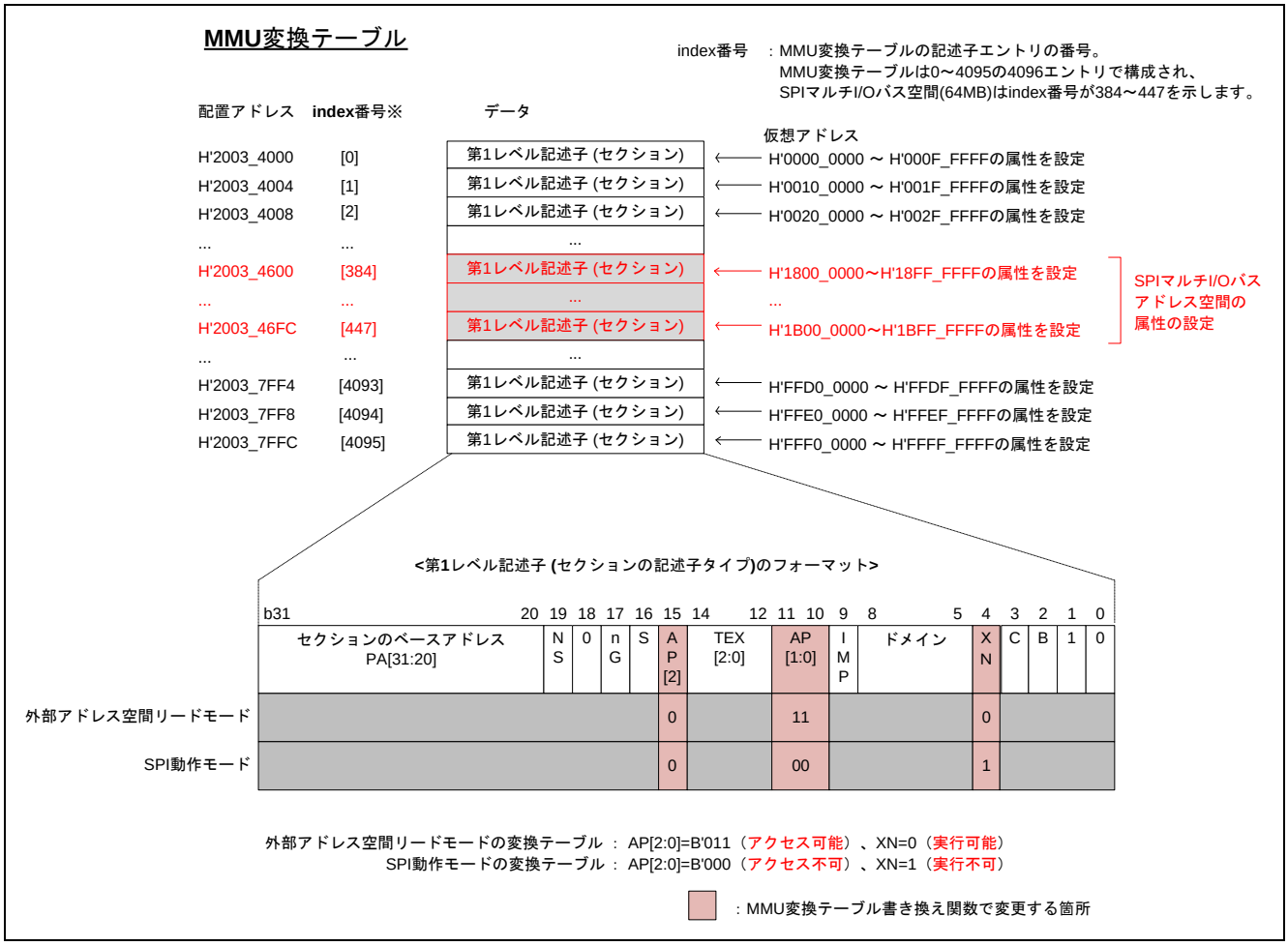


図6.9　サンプルコードの MMU 変換テーブルの設定内容

6.6 サンプルコマンド

本サンプルコードでは、シリアルフラッシュメモリのイレーズ、ライト、およびリードの処理を行うコマンドを準備しており、ターミナルからコマンドを入力して実行します。

リセット解除後、シリアルフラッシュブートで起動時にブート起動用内蔵 ROM プログラムが SPIBSC を外部アドレス空間リードモードに設定して、シリアルフラッシュメモリへの直接リードアクセスを行うことでサンプルコードが実行されます。本コマンドにより、シリアルフラッシュメモリへのイレーズまたはライト処理を行う場合には、SPI 動作モードでシリアルフラッシュメモリにアクセスする必要があるため、これらのコマンドの実行時には「6.3　外部アドレス空間リードモードと SPI 動作モードの切り替え」に記載の処理手順が実行されるようにしています。シリアルフラッシュメモリへのリード処理は、SPI 動作モードでリードする処理と、外部アドレス空間リードモードで直接リードする処理の 2 種類のコマンドを準備しています。各コマンド処理の実行後は、SPIBSC の動作モードは外部アドレス空間リードモードで動作するようにしています。表 6.1にサンプルコマンドの処理の一覧を示します。

サンプルコードを動作させるための基本的なコマンド入力手順を以下に示します。

1. ターミナルより、"SPIBSC"＋"Enter"を入力し、サンプルコードの処理に遷移します。
2. ターミナルより、"XREAD"＋"Enter"を入力し、SPI マルチ I/O バス空間に接続されたシリアルフラッシュメモリを CPU が直接リードします。
3. ターミナルより、"ERASE"＋"Enter"を入力し、シリアルフラッシュメモリのデータをイレーズします。
4. ターミナルより、"WRITE"＋"Enter"を入力し、シリアルフラッシュメモリにデータをライトします。
5. ターミナルより、"SREAD"＋"Enter"を入力し、シリアルフラッシュメモリをリードします。

表6.1　サンプルコマンドの処理の一覧

サンプル処理	概要	コマンド
SPI マルチ I/O バス空間の直接リード（XIP）処理	外部アドレス空間リードモードにて、SPI マルチ I/O バス空間に接続されているシリアルフラッシュメモリのデータを直接リードし、リードしたデータをターミナルに表示。	XREAD
イレーズ処理	SPI 動作モードに切り替え後、R_SFLASH_EraseSector 関数を使用してシリアルフラッシュメモリに対してイレーズコマンドを発行し、シリアルフラッシュメモリをイレーズする。 イレーズ処理完了後、外部アドレス空間リードモードで動作。	ERASE
ライト処理	SPI 動作モードに切り替え後、R_SLASH_ByteProgram 関数を使用して、ソフトウェア処理より、シリアルフラッシュメモリ対してプログラムコマンドを発行してテストデータをライトする。ライト処理完了後、外部アドレス空間リードモードで動作。	WRITE
SPI 動作モードによるリード処理	SPI 動作モードに切り替え後、R_SLASH_ByteRead 関数を使用して、ソフトウェア処理より、シリアルフラッシュメモリにリードコマンドを発行して、シリアルフラッシュメモリのデータをリードして、リードしたデータをターミナルに表示する。リード処理完了後、外部アドレス空間リードモードで動作。	SREAD

6.6.1 XREAD コマンド

XREAD コマンドは、ターミナルから指定された SPI マルチ I/O バス空間上のアドレス（H'1800_0000～H'1BFF_FFFF）から指定されたリードするバイト数分のデータを、CPU が直接リードし、リードしたデータをターミナルに表示します。

図 6.10にXREAD コマンド実行時のターミナル表示例を示します。

```
SPIBSC> XREAD                                                    --- ①
Please input start address (hex) ?
18000000                                                         --- ②
Please input size ?
256                                                              --- ③



address     +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F     --- ④
-------------------------------------------------------------
0x18000000 : 18 f0 9f e5 18 f0 9f e5 18 f0 9f e5 18 f0 9f e5
0x18000010 : 18 f0 9f e5 18 f0 9f e5 18 f0 9f e5 18 f0 9f e5
0x18000020 : 00 40 00 18 60 40 00 18 64 40 00 18 68 40 00 18

(中略)

0x180000D0 : ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
0x180000E0 : ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
0x180000F0 : ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

SPIBSC>
```

図6.10　XREAD コマンド実行時のターミナル表示例

① "XREAD"+"Enter"を入力すると、XREAD コマンドを起動します。

② "Please input start address (hex) ?"に続いて、"リードする先頭アドレス（16 進数）"＋"Enter"を入力します。
（指定可能なアドレスの範囲は、H'1800_0000～H'1BFF_FFFF。）

③ "Please input size ?"に続いて、"リードするバイト数（10 進数）"＋"Enter"を入力します。
（指定可能なバイト数の範囲は、1～4096。また、リードするアドレスの範囲が H'1800_0000～H'1BFF_FFFF の範囲を超える場合には、エラー終了します。）

④ 指定したアドレス、バイト数の領域を、外部アドレス空間リードモードにて CPU が直接リードした結果を表示します。

6.6.2 ERASE コマンド

ERASE コマンドは、ターミナルから指定されたシリアルフラッシュメモリのアドレス（H'0010_0000～H'03FF_FFFF）と、指定されたイレーズするバイト数（1～1048576）からイレーズするブロック数（ブロックサイズは 4K バイト）を計算して、R_SFLASH_EraseSector 関数をコールし、シリアルフラッシュメモリのデータをイレーズします。

図 6.11にERASE コマンド実行時のターミナル表示例を示します。

```
SPIBSC> ERASE                                                    --- ①
Please input start address (hex) ?
100000                                                           --- ②
Please input size ?
256                                                              --- ③

 ERASE command: success                                          --- ④
 Erased 4096 bytes data from 0x00100000 to 0x00100FFF.

SPIBSC>
```

図6.11　ERASE コマンド実行時のターミナル表示例

① "ERASE"＋"Enter"を入力すると、ERASE コマンドを起動します。

② "Please input start address (hex) ?"に続いて、"イレーズする先頭アドレス（16 進数）"＋"Enter"を入力します。
（指定可能なアドレスの範囲は、H'0010_0000～H'03FF_FFFF。シリアルフラッシュメモリの H'0000_0000～H'000F_FFFF の 1MB の領域は、本サンプルコードが格納されており、この領域をイレーズすることはできません。）

③ "Please input size ?"に続いて、"イレーズするバイト数（10 進数）"＋"Enter"を入力します。
（指定可能なバイト数の範囲は、1～1048576。また、先頭アドレスにバイト数を加算して、H'0010_0000～H'03FF_FFFF の範囲を超える場合には、エラー終了します。）

④ 指定したアドレス、バイト数の領域をイレーズします。コマンドの実行結果を表示した後、ERASE コマンドを終了します。

6.6.3 WRITE コマンド

WRITE コマンドは、ターミナルから指定されたシリアルフラッシュメモリのアドレス（H'0010_0000～H'03FF_FFFF）から、指定されたライトするバイト数（1～1048576）分のデータをライトします。ライトするデータは初期値（0～255）をターミナルから入力します。ターミナルより入力されたライトデータの初期値を使用して、ライトするバイト数分の連番データを生成します。R_SFLASH_ByteProgram 関数をコールして、シリアルフラッシュメモリにデータをライトします。ライト後、R_SLLASH_ByteRead 関数をコールしてシリアルフラッシュメモリからデータをリードし、ライトデータとベリファイ処理を行います。

なお、WRITE コマンドでライトするシリアルフラッシュメモリの領域については、WRITE コマンドの実行前に ERASE コマンドを実行してデータをイレーズしてください。イレーズを実行しない場合、意図したデータをライトできません。

図 6.12にWRITE コマンド実行時のターミナル表示例を示します。

```
SPIBSC> WRITE                                                    --- ①
Please input start address (hex) ?
100000                                                           --- ②
Please input size ?
256                                                              --- ③
Please input start value to write ?
1                                                                --- ④

Are you sure data is erased before write operation (Y/N)?
Y                                                                --- ⑤

 WRITE command: success                                          --- ⑥
 Wrote 256 bytes data from 0x00100000 to 0x001000FF.
 Verify OK.

SPIBSC>
```

図6.12　WRITE コマンド実行時のターミナル表示例

① "WRITE"＋"Enter"を入力すると、WRITE コマンドを起動します。

② "Please input start address (hex) ?"に続いて、"ライトする先頭アドレス（16 進数）"＋"Enter"を入力します。
（指定可能なアドレスの範囲は、H'0010_0000～H'03FF_FFFF で、256 バイト（ページサイズ）の倍数となる値。シリアルフラッシュメモリの H'0000_0000～H'000F_FFFF の 1MB の領域は、本サンプルコードが格納されており、この領域をライトすることはできません。）

③ "Please input size ?"に続いて、"ライトするバイト数（10 進数）"＋"Enter"を入力します。
（指定可能なバイト数の範囲は、1～1048576。先頭アドレスにバイト数を加算して、H'0010_0000～H'03FF_FFFF の範囲を超える場合には、エラー終了します。）

④ "Please input start value to write ?"に続いて、"ライトする連番データの開始値（10 進数）"＋"Enter"を入力します。
（指定可能な値の範囲は、0～255。連番データは 0～255 の範囲で生成します。）

⑤ "Are you sure data is erased before write operation (Y/N)?"に続いて、"Y"＋"Enter"または"N"＋"Enter"を入力します。本コマンドの実行前に ERASE コマンドを実行している場合は"Y"を入力します。"N"を入力すると、ライト処理を行わずにコマンドを終了します。

⑥ 指定したアドレス、バイト数のライトをします。コマンドの実行結果を表示した後、WRITE コマンドを終了します。コマンド実行後にベリファイの結果を表示します。

ベリファイ処理でエラーが発生した場合には、ライトするバイト数分のデータの書き込み終わるまで処理を継続し、"Verify NG: "の行に"ベリファイエラーのバイト数 / ライトしたバイト数"を表示します。アドレス昇順にベリファイ処理を行い、ベリファイエラーが発生した場合には、最初にエラーとなったアドレスの情報（ライトしたアドレス、ライトしたデータ、リードしたデータ）を表示します。

図 6.13にWRITE コマンド実行時のターミナル表示例（ベリファイエラー発生時）を示します。

```
SPIBSC> WRITE
Please input start address (hex) ?
100000
Please input size ?
256
Please input start value to write ?
1

Are you sure data is erased before write operation (Y/N)?
Y

 WRITE command: success
 Wrote 256 bytes data from 0x00100000 to 0x001000FF.
 Verify NG            : 7 / 256
   Verify error data : address=0x00100000, write=0x11, read=0x01

SPIBSC>
```

図6.13　WRITE コマンド実行時のターミナル表示例（ベリファイエラー発生時）

6.6.4 SREAD コマンド

SREAD コマンドは、ターミナルから指定されたシリアルフラッシュメモリのアドレス（H'0000_0000～H'03FF_FFFF）、指定されたリードするバイト数（1～4096 バイト）をターミナルから入力します。SPIBSC を SPI 動作モード切り替えた後、R_SLASH_ByteRead 関数をコールして、シリアルフラッシュメモリをリードして、リードしたデータをターミナルに表示します。

図 6.14にSREAD コマンド実行時のターミナル表示例を示します。

```
SPIBSC> SREAD                                                    --- ①
Please input start address (hex) ?
100000                                                           --- ②
Please input size ?
256                                                              --- ③



address     +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F     --- ④
-------------------------------------------------------------
0x00100000 : 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10
0x00100010 : 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20
0x00100020 : 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30

(中略)

0x001000D0 : d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df e0
0x001000E0 : e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef f0
0x001000F0 : f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff 00

SPIBSC>
```

図6.14　SREAD コマンド実行時のターミナル表示例

① "SREAD"＋"Enter"を入力すると、SREAD コマンドを起動します。

② "Please input start address (hex) ?"に続いて、"リードする先頭アドレス（16 進数）"＋"Enter"を入力します。（指定可能なアドレスの範囲は、H'0000_0000～H'03FF_FFFF。）

③ "Please input size ?"に続いて、"リードするバイト数（10 進数）"＋"Enter"を入力します。（指定可能なバイト数は 1～4096。先頭アドレスにバイト数を加算して、H'0000_0000～H'03FF_FFFF の範囲を超えた場合には、エラー終了します。）

④ 指定したアドレス、バイト数の領域を、SPI 動作モードでリードした結果を表示します。

6.7 サンプルコード実行時の周辺機能の設定およびメモリ配置

6.7.1 周辺機能の設定

表 6.2にサンプルコード実行時の周辺機能の設定内容を示します。

表6.2 周辺機能の設定内容

モジュール	設定内容
CPG	CPU クロック（Iφ）：400MHz 内部バスクロック（Bφ）：133.33MHz 周辺クロック（P1φ）：66.67MHz 周辺クロック（P0φ）：33.33MHz
SPIBSC	外部アドレス空間リードモードに設定し、CPU が SPI マルチ I/O バス空間に接続されたシリアルフラッシュメモリから、直接リードするための信号を生成するための設定
PORT	PORT4 のマルチプレクス端子機能を設定 P4_4：SPBCLK_0 P4_5：SPBSSL_0 P4_6：SPBMO0_0/SPBIO00_0 P4_7：SPBMI0_0/SPBIO10_0 P4_2：SPBIO20_0 P4_3：SPBIO30_0
STB	保持用内蔵 RAM へのライト許可および周辺機能へのクロック供給 STBCR2～STBCR12 でクロックの供給および停止制御が可能なすべての周辺機能のクロックを供給
SCIF	チャネル 0 を調歩同期式モードに設定 ・データ長：8 ビット ・ストップビット長：1 ビット ・パリティ：なし P1φ=66.67MHz の時に、クロックソースを分周なし、ビットレート値に 17 を設定し、ビットレートが 115200bps となるように設定 誤差は 0.46%

6.7.2 メモリマップ

図 6.15にRZ/A1LUグループのアドレス空間とサンプルコードが動作するJASMINE ボードのメモリマップを示します。

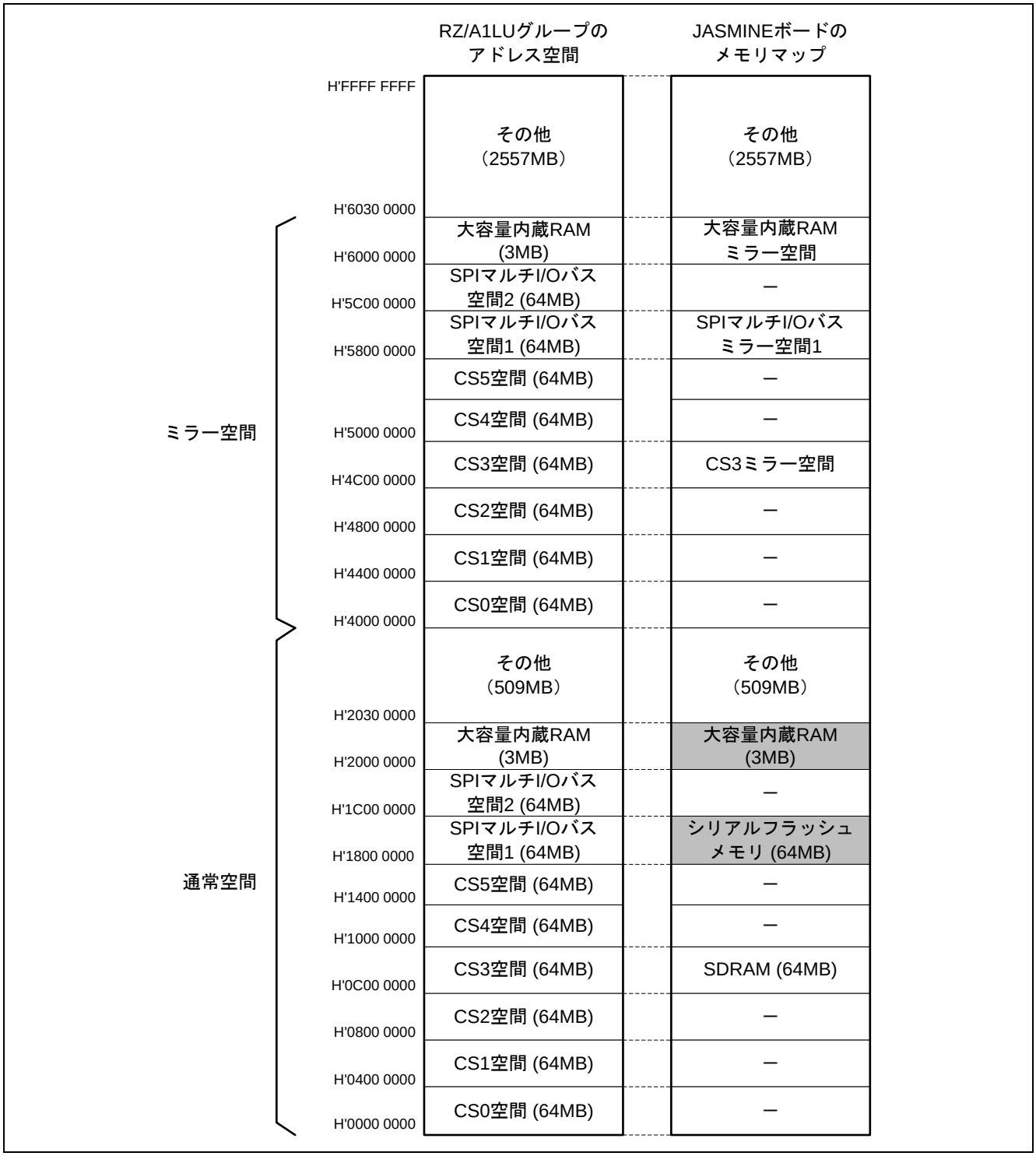


図6.15　RZ/A1LUグループのアドレス空間とJASMINE ボードメモリマップ

6.7.3 サンプルコードのセクション配置

表 6.3にSPIBSC 初期設定プログラムで使用するセクションを、表 6.4～表 6.6にアプリケーションプログラムで使用するセクションを示します。

表6.3 SPIBSC 初期設定プログラムで使用するセクション

領域の名前	内容	タイプ	ロード領域	実行領域
VECTOR_TABLE	例外処理ベクタテーブル	Code	S-FLASH	S-FLASH
CODE_SPIBSC_INIT1	SPIBSC 初期設定プログラム 1 用プログラムコード領域	Code	S-FLASH	LRAM
CODE_IO_REGRW	IO レジスタのリード/ライト関数のプログラムコード領域	Code	S-FLASH	LRAM
CODE_SPIBSC_INIT2	SPIBSC 初期設定プログラム 2 用プログラムコード領域	Code	S-FLASH	LRAM
DATA_SPIBSC_INIT2	SPIBSC 初期設定プログラム 2 用初期値ありデータ領域	RW Data	S-FLASH	LRAM
BSS_SPIBSC_INIT2	SPIBSC 初期設定プログラム 2 用初期値なし領域	ZI Data	―	LRAM
RESET_HANDLER	リセットハンドラ処理のプログラムコード領域	Code	S-FLASH	S-FLASH
CODE	デフォルトのプログラムコード領域 C ソースでセクション名を定義しない Code タイプのセクションは、すべてこの領域に配置されます	Code	S-FLASH	S-FLASH
SVC_STACK	スタック領域	ZI Data	―	LRAM

【注】 表中のロード領域および実行領域において、S-FLASH はシリアルフラッシュメモリの領域を、LRAM は大容量内蔵 RAM の領域を表します。

表6.4　アプリケーションプログラムで使用するセクション（1/3）

領域の名前	内容	タイプ	ロード領域	実行領域
VECTOR_TABLE	例外処理ベクタテーブル	Code	S-FLASH	S-FLASH
RESET_HANDLER	リセットハンドラ処理のプログラムコード領域 この領域は以下のセクションから構成されています ・INITCA9CACHE（L1 キャッシュ設定） ・INIT_TTB（MMU 設定） ・RESET_HANDLER（リセットハンドラ）	Code	S-FLASH	S-FLASH
CODE_BASIC_SETUP	保持用内蔵 RAM のライト許可のためのプログラムコード領域	Code	S-FLASH	S-FLASH
InRoot	この領域は C 標準ライブラリなどのルート領域に配置するセクションから構成されています	Code および RO Data	S-FLASH	S-FLASH
CODE_FPU_INIT	NEON および VFP 初期設定のプログラムコード領域 この領域は以下のセクションから構成されています ・CODE_FPU_INIT ・FPU_INIT	Code	S-FLASH	S-FLASH
CODE_RESET	ハードウェア初期設定のプログラムコード領域 この領域は以下のセクションから構成されています ・CODE_RESET（スタートアップ処理） ・INIT_VBAR（ベクタベース設定）	Code	S-FLASH	S-FLASH
CODE	デフォルトのプログラムコード領域 C ソースでセクション名を定義しない Code タイプのセクションは、すべてこの領域に配置されます	Code	S-FLASH	S-FLASH
CONST	デフォルトの定数データ領域 C ソースでセクション名を定義しない RO Data タイプのセクションは、すべてこの領域に配置されます	RO Data	S-FLASH	S-FLASH

表6.5　アプリケーションプログラムで使用するセクション（2/3）

領域の名前	内容	タイプ	ロード領域	実行領域
VECTOR_MIRROR_TABLE	例外処理ベクタテーブル （大容量内蔵 RAM に転送して実行するためのセクション）	Code	S-FLASH	LRAM
CODE_HANDLER_JMPTBL	IRQ 割り込みハンドラのユーザ定義関数のプログラムコード領域	Code	S-FLASH	LRAM
CODE_HANDLER	IRQ 割り込みハンドラのプログラムコード領域 この領域は以下のセクションから構成されています ・CODE_HANDLER ・IRQ_FIQ_HANDLER	Code	S-FLASH	LRAM
CODE_IO_REGRW	IO レジスタのリード/ライト関数のプログラムコード領域	Code	S-FLASH	LRAM
CODE_CACHE_OPERATION	L1 および L2 キャッシュ設定処理のプログラムコード領域（注 3）	Code	S-FLASH	LRAM
DATA_HANDLER_JMPTBL	IRQ 割り込みハンドラのユーザ定義関数の登録テーブルデータ領域	RW Data	S-FLASH	LRAM
ARM_LIB_STACK	アプリケーションスタック領域	ZI Data	―	LRAM
IRQ_STACK	IRQ モードのスタック領域	ZI Data	―	LRAM
FIQ_STACK	FIQ モードのスタック領域	ZI Data	―	LRAM
SVC_STACK	スーパバイザ（SVC）モードのスタック領域	ZI Data	―	LRAM
ABT_STACK	アボート（ABT）モードのスタック領域	ZI Data	―	LRAM
TTB	MMU 変換テーブル領域	ZI Data	―	LRAM
ARM_LIB_HEAP	アプリケーションヒープ領域	ZI Data	―	LRAM
DATA	デフォルトの初期値ありデータ領域 C ソースでセクション名を定義しない RW Data タイプのセクションは、すべてこの領域に配置されます	RW Data	S-FLASH	LRAM
BSS	デフォルトの初期値なしデータ領域 C ソースでセクション名を定義しない ZI Data タイプのセクションは、すべてこの領域に配置されます	ZI Data	―	LRAM

表6.6　アプリケーションプログラムで使用するセクション（3/3）

領域の名前	内容	タイプ	ロード領域	実行領域
CODE_SPIBSC_ WRITE_OPERATION	SPIBSC のモード切り替え処理用のプログラムコード領域	Code	S-FLASH	LRAM
CONST_SPIBSC_ WRITE_OPERATION	SPIBSC のモード切り替え処理用の定数データ領域	RO Data	S-FLASH	LRAM
DATA_SPIBSC_ WRITE_OPERATION	SPIBSC のモード切り替え処理用の初期値ありデータ領域	RW Data	S-FLASH	LRAM
BSS_SPIBSC_ WRITE_OPERATION	SPIBSC のモード切り替え処理用の初期値なしデータ領域	ZI Data	ー	LRAM
CODE_MMU_ OPERATION	MMU 変換テーブル変更処理用のプログラムコード領域	Code	S-FLASH	LRAM
CONST_MMU_ OPERATION	MMU 変換テーブル変更処理用の定数データ領域	RO Data	S-FLASH	LRAM
DATA_MMU_ OPERATION	MMU 変換テーブル変更処理用の初期値ありデータ領域	RW Data	S-FLASH	LRAM
BSS_MMU_ OPERATION	MMU 変換テーブル変更処理用の初期値なしデータ領域	ZI Data	ー	LRAM

【注】 1. 表中のロード領域および実行領域において、S-FLASH はシリアルフラッシュメモリの領域を、LRAM は大容量内蔵 RAM の領域を表します。
2. セクションの名前は基本的に領域と同じ名前にしていますが、RESET_HANDLER、InRoot、CODE_FPU_INIT、CODE_RESET、CODE、CONST、CODE_HANDLER、DATA、BSS の各領域は複数のセクションから構成されています。領域とセクションについては、ARM コンパイラツールチェーンのマニュアルを参照してください。
3. このセクションは、キャッシュ無効領域に配置する必要があります。

6.8 使用割り込み一覧

表 6.7にサンプルコードで使用する割り込みを示します。

表6.7 サンプルコードで使用する割り込み

割り込み要因（要因 ID）	優先度	処理概要
OSTM0（134）	5	500ms ごとに割り込みを発生

6.9 定数一覧

表 6.8～表 6.10に使用する定数を示します。

表6.8 サンプルコードで使用する定数（1/3）

定数名	設定値	内容
SPIBSC_1BIT	0	コマンド、オプショナルコマンド、アドレス、オプションデータ、転送データのビット幅を 1 ビットに設定
SPIBSC_4BIT	2	コマンド、オプショナルコマンド、アドレス、オプションデータ、転送データのビット幅を 4 ビットに設定
SPIBSC_CMNCR_BSZ_SINGLE	0	チャネルに接続しているシリアルフラッシュの個数を 1 個に設定
SPIBSC_CMNCR_BSZ_DUAL	1	チャネルに接続しているシリアルフラッシュの個数を 2 個に設定
SPIBSC_OUTPUT_DISABLE	0	コマンド、オプショナルコマンド、アドレス、オプションデータを出力しない設定
SPIBSC_OUTPUT_ENABLE	1	コマンド、オプショナルコマンド、アドレス、オプションデータを出力する設定
SPIBSC_OUTPUT_ADDR_24	0x07	24 ビットのアドレスを出力
SPIBSC_OUTPUT_ADDR_32	0x0f	32 ビットのアドレスを出力
SPIBSC_OUTPUT_OPD_3	0x08	オプションデータイネーブル OPD3 を出力
SPIBSC_OUTPUT_OPD_32	0x0c	オプションデータイネーブル OPD3,OPD2 を出力
SPIBSC_OUTPUT_OPD_321	0x0e	オプションデータイネーブル OPD3,OPD2,OPD1 を出力
SPIBSC_OUTPUT_OPD_3210	0x0f	オプションデータイネーブル OPD3,OPD2,OPD1,OPD0 を出力
SPIBSC_OUTPUT_SPID_8	0x08	SPI 動作モード時に転送データイネーブルを 8（または 16）ビット転送に設定
SPIBSC_OUTPUT_SPID_16	0x0c	SPI 動作モード時に転送データイネーブルを 16（または 32）ビット転送に設定
SPIBSC_OUTPUT_SPID_32	0x0f	SPI 動作モード時に転送データイネーブルを 32（または 64）ビット転送に設定
SPIBSC_SPISSL_NEGATE	0	SPI 動作モード時に転送終了後の SPBSSL 信号状態をネゲートに設定
SPIBSC_SPISSL_KEEP	1	SPI 動作モード時に転送終了後から次アクセス開始まで SPBSSL 信号レベルを保持する設定
SPIBSC_SPIDATA_DISABLE	0	SPI 動作モード時にデータをリード/ライトしない設定
SPIBSC_SPIDATA_ENABLE	1	SPI 動作モード時にデータリード/ライトする設定

表6.9　サンプルコードで使用する定数（2/3）

定数名	設定値	内容
SPIBSC_DUMMY_CYC_DISABLE	0	ダミーサイクルを挿入しない設定
SPIBSC_DUMMY_CYC_ENABLE	1	ダミーサイクルを挿入する設定
SPIBSC_DUMMY_1CYC	0	外部アドレス空間リードモードまたはSPI動作モード時にシリアルフラッシュメモリに出力する ダミーサイクル数を 1 に設定
SPIBSC_DUMMY_2CYC	1	外部アドレス空間リードモードまたはSPI動作モード時にシリアルフラッシュメモリに出力する ダミーサイクル数を 2 に設定
SPIBSC_DUMMY_3CYC	2	外部アドレス空間リードモードまたはSPI動作モード時にシリアルフラッシュメモリに出力する ダミーサイクル数を 3 に設定
SPIBSC_DUMMY_4CYC	3	外部アドレス空間リードモードまたはSPI動作モード時にシリアルフラッシュメモリに出力する ダミーサイクル数を 4 に設定
SPIBSC_DUMMY_5CYC	4	外部アドレス空間リードモードまたはSPI動作モード時にシリアルフラッシュメモリに出力する ダミーサイクル数を 5 に設定
SPIBSC_DUMMY_6CYC	5	外部アドレス空間リードモードまたはSPI動作モード時にシリアルフラッシュメモリに出力する ダミーサイクル数を 6 に設定
SPIBSC_DUMMY_7CYC	6	外部アドレス空間リードモードまたはSPI動作モード時にシリアルフラッシュメモリに出力する ダミーサイクル数を 7 に設定
SPIBSC_DUMMY_8CYC	7	外部アドレス空間リードモードまたはSPI動作モード時にシリアルフラッシュメモリに出力する ダミーサイクル数を 8 に設定
SPIBSC_SDR_TRANS	0	SPI 動作モード時の転送モードを SDR 転送に設定
SPIBSC_DDR_TRANS	1	SPI 動作モード時の転送モードを DDR 転送に設定
SF_REQ_PROTECT	0	シリアルフラッシュメモリ内のレジスタ設定情報を「プロテクト」に設定
SF_REQ_UNPROTECT	1	シリアルフラッシュメモリ内のレジスタ設定情報を「プロテクト解除」に設定

表6.10　サンプルコードで使用する定数（3/3）

定数名	設定値	内容
SF_PAGE_SIZE	256	シリアルフラッシュメモリのページサイズ
SF_SECTOR_SIZE	4 * 1024	シリアルフラッシュメモリのセクタサイズ
SF_NUM_OF_SECTOR	16384	シリアルフラッシュメモリのセクタ数
SPIBSC_ADDR_START	0x18000000	SPI マルチ I/O バス空間の開始アドレス
SPIBSC_ADDR_END	0x1BFFFFFF	SPI マルチ I/O バス空間の終了アドレス
FLASH_ADDR_START	0x00000000	シリアルフラッシュメモリの開始アドレス
FLASH_ADDR_END	0x03FFFFFF	シリアルフラッシュメモリの終了アドレス
FLASH_PROTECT_SIZE	0x00100000	SPIBSC のサンプルコマンドで使用するシリアルフラッシュメモリのイレーズおよびライト保護領域のサイズ （シリアルフラッシュメモリの H'0000_0000～H'000F_FFFF の領域がイレーズおよびライトされないように、1MB の領域を保護するようにしています。本サンプルコードの格納領域の内容が、変更されることのないように保護します。）
FLASH_SECTOR_BASE_MASK	~(SF_SECTOR_SIZE-1)	シリアルフラッシュメモリのセクタ先頭アドレスを計算するためのマスク値
FLASH_PAGE_OFFSET_MASK	SF_PAGE_SIZE - 1	シリアルフラッシュメモリのページ先頭アドレスからのオフセットを計算するためのマスク値
SPIBSC_WRITE_PAGE_BUF_SIZE	SF_SECTOR_SIZE	SPIBSC のサンプルコマンドで使用するライトバッファのサイズ
SPIBSC_READ_BUF_SIZE	SF_SECTOR_SIZE << 4	SPIBSC のサンプルコマンドで使用するリードバッファのサイズ
ARGUMENT_BUF_SIZE	128	SPIBSC のサンプルコマンドでコンソールからの入力文字を格納するバッファのサイズ

6.10 構造体/共用体一覧

表 6.11～表 6.16に使用する構造体を示します。

表6.11 SPIBSC SPI 動作モード設定構造体（st_spibsc_spimd_reg_t）（1/5）

メンバ名	内容
uint32_t cdb	コマンドビット幅 • SPI 動作モード時のコマンドビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅 • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の CDB[1:0]に設定します。
uint32_t ocdb	オプショナルコマンドビット幅 • SPI 動作モード時のオプショナルコマンドビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅 • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の OCDB[1:0]に設定します。
uint32_t adb	アドレスビット幅 • SPI 動作モード時のアドレスビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅 • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の ADB[1:0]に設定します。
uint32_t opdb	オプションデータビット幅 • SPI 動作モード時のオプションデータビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅 • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の OPDB[1:0]に設定します。
uint32_t spidb	転送データビット幅 • SPI 動作モード時の転送データビット幅を指定します。 • 設定可能な値： SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅 • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の SPIDB[1:0]に設定します。
uint32_t cde	SPI 動作モード時にコマンドを出力するかを設定します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_ENABLE：出力する • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の CDE に設定します。

表6.12　SPIBSC SPI 動作モード設定構造体（st_spibsc_spimd_reg_t）（2/5）

メンバ名	内容
uint32_t　ocde	オプショナルコマンドイネーブル • SPI 動作モード時にオプショナルコマンドを出力するかを設定します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_ENABLE：出力する • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の OCDE に設定します。
uint32_t　ade	アドレスイネーブル • SPI 動作モード時にアドレスを出力するかを設定します。 • 設定可能な値： • SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_ADDR_24：ADR[23:0]を出力 SPIBSC_OUTPUT_ADDR_32：ADR[31:0]を出力 • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の ADE[3:0]に設定します。
uint32_t　opde	オプションデータイネーブル • SPI 動作モード時にオプションデータを出力するかを設定します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_OPD_3：OPD3 を出力 SPIBSC_OUTPUT_OPD_32：OPD3,OPD2 を出力 SPIBSC_OUTPUT_OPD_321：OPD3,OPD2,OPD1 を出力 SPIBSC_OUTPUT_OPD_3210：OPD3,OPD2,OPD1,OPD0 を出力 • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の OPDE[3:0]に設定します。
uint32_t　spide	転送データイネーブル • SPI 動作モード時にデータ転送を行うかを設定します。 • 設定可能な値： SPIBSC_OUTPUT_DISABLE：出力しない SPIBSC_OUTPUT_SPID_8：8（または 16）ビット転送 SPIBSC_OUTPUT_SPID_16：16（または 32）ビット転送 SPIBSC_OUTPUT_SPID_32：32（または 64）ビット転送 • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の SPIDE[3:0]に設定します。
uint32_t　sslkp	SPBSSL 信号レベル保持 • SPI 動作モード時に転送終了後の SPBSSL 信号状態を設定します。 • 設定可能な値： SPIBSC_SPISSL_NEGATE：転送終了時にネゲート SPIBSC_SPISSL_KEEP：転送終了後から次アクセス開始まで SPBSSL 信号レベルを保持 • 本メンバに設定した値を SPI モードコントロールレジスタ（SMCR）の SSLKP に設定します。

表6.13　SPIBSC SPI 動作モード設定構造体（st_spibsc_spimd_reg_t）（3/5）

メンバ名	内容
uint32_t　spire	データリードイネーブル • SPI 動作モード時にデータリードするかを設定します。 • 設定可能な値： SPIBSC_SPIDATA_DISABLE：データリードしない SPIBSC_SPIDATA_ENABLE：データリードする • 本メンバに設定した値を SPI モードコントロールレジスタ（SMCR）の SPIRE に設定します。
uint32_t　spiwe	データライトイネーブル • SPI 動作モード時にデータライトするかを設定します。 • 設定可能な値： SPIBSC_SPIDATA_DISABLE：データライトしない SPIBSC_SPIDATA_ENABLE：データライトする • 本メンバに設定した値を SPI モードコントロールレジスタ（SMCR）の SPIWE に設定します。
uint32_t　dme	ダミーサイクルイネーブル • SPI 動作モード時にダミーサイクル挿入するかどうかを設定します。 • 設定可能な値： SPIBSC_DUMMY_CYC_DISABLE：挿入しない SPIBSC_DUMMY_CYC_ENABLE：挿入する • 本メンバに設定した値を SPI モードイネーブル設定レジスタ（SMENR）の DME に設定します。
uint32_t　addre	アドレス DDR イネーブル • SPI 動作モード時に出力するアドレスの SDR/DDR 転送を選択します。 • 設定可能な値： SPIBSC_SDR_TRANS：SDR 転送 SPIBSC_DDR_TRANS：DDR 転送 • 本メンバに設定した値を SPI モード DDR イネーブルレジスタ（SMDRENR）の ADDRE に設定します。
uint32_t　opdre	オプションデータ DDR イネーブル • SPI 動作モード時に出力するオプションデータの SDR/DDR 転送を選択します。 • 設定可能な値： SPIBSC_SDR_TRANS：SDR 転送 SPIBSC_DDR_TRANS：DDR 転送 • 本メンバに設定した値を SPI モード DDR イネーブルレジスタ（SMDRENR）の OPDRE に設定します。
uint32_t　spidre	転送データ DDR イネーブル • SPI 動作モード時に転送するデータの SDR/DDR 転送を選択します。 • 設定可能な値： SPIBSC_SDR_TRANS：SDR 転送 SPIBSC_DDR_TRANS：DDR 転送 • 本メンバに設定した値を SPI モード DDR イネーブルレジスタ（SMDRENR）の SPIDRE に設定します。

表6.14　SPIBSC SPI 動作モード設定構造体（st_spibsc_spimd_reg_t）（4/5）

メンバ名	内容
uint8_t　dmdb	ダミーサイクルビット幅 • SPI 動作モード時のダミーサイクルのビット幅を設定します。 • 設定可能な値： SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅 • 本メンバに設定した値を SPI モードダミーサイクル設定レジスタ（SMDMCR）の DMDB[1:0]に設定します。
uint8_t　dmcyc	ダミーサイクル数 • SPI 動作モード時のダミーサイクル数を設定します。 • 設定可能な値： SPIBSC_DUMMY_1CYC：1 サイクル SPIBSC_DUMMY_2CYC：2 サイクル SPIBSC_DUMMY_3CYC：3 サイクル SPIBSC_DUMMY_4CYC：4 サイクル SPIBSC_DUMMY_5CYC：5 サイクル SPIBSC_DUMMY_6CYC：6 サイクル SPIBSC_DUMMY_7CYC：7 サイクル SPIBSC_DUMMY_8CYC：8 サイクル • 本メンバに設定した値を SPI モードダミーサイクル設定レジスタ（SMDMCR）の DMCYC[2:0]に設定します。
uint8_t　cmd	コマンド • SPI 動作モード時に出力するコマンドを設定します。 • 本メンバに設定した値を SPI モードコマンド設定レジスタ（SMCMR）の CMD[7:0]に設定します。
uint8_t　ocmd	オプショナルコマンド • SPI 動作モード時に出力するオプショナルコマンドを設定します。 • 本メンバに設定した値を SPI モードコマンド設定レジスタ（SMCMR）の OCMD[7:0]に設定します。
uint32_t　addr	アドレス • SPI 動作モード時に出力するアドレスを設定します。 • 本メンバに設定した値を SPI モードアドレス設定レジスタ（SMADR）の ADR[31:0]に設定します。
uint8_t　opd[4]	オプションデータ • SPI 動作モード時に出力するオプションデータを設定します。 • 本メンバに設定した値を SPI モードオプション設定レジスタ（SMOPR）の OPDn[7:0]に以下のように設定します。 OPD3[7:0] ← opd[0] OPD2[7:0] ← opd[1] OPD1[7:0] ← opd[2] OPD0[7:0] ← opd[3]

表6.15　SPIBSC SPI 動作モード設定構造体（st_spibsc_spimd_reg_t）（5/5）

メンバ名	内容
uint32_t　smrdr[2]	リードデータ格納バッファ • SPI 動作モード時にリードしたデータ（SPI モードリードデータレジスタ n（SMRDRn））を以下のように格納します。 SMRDR0 → smrdr[0] SMRDR1 → smrdr[1]
uint32_t　smwdr[2]	ライトデータ格納バッファ • SPI 動作モード時にライトするデータ（SPI モードライトデータレジスタ n（SMWDRn））を以下のように格納します。 SMWDR0 ← smwdr[0] SMWDR1 ← smwdr[1]

表6.16　MMU 変換テーブルの第 1 レベル記述子（セクション）のディスクリプタ構造体
（mmu_ttbl_desc_section_t）

メンバ名	内容
uint32_t b0 : 1	第 1 レベル記述子（セクション）の bit0
uint32_t b1 : 1	第 1 レベル記述子（セクション）の bit1
uint32_t B : 1	第 1 レベル記述子（セクション）の B ビット
uint32_t C : 1	第 1 レベル記述子（セクション）の C ビット
uint32_t XN: 1	第 1 レベル記述子（セクション）の XN ビット
uint32_t Domain : 4	第 1 レベル記述子（セクション）の Domain ビット
uint32_t b9 : 1	第 1 レベル記述子（セクション）の bit9
uint32_t AP1_0 : 2	第 1 レベル記述子（セクション）の AP[1:0]ビット
uint32_t TEX : 3	第 1 レベル記述子（セクション）の TEX[2:0]ビット
uint32_t AP2 : 1	第 1 レベル記述子（セクション）の AP[2]ビット
uint32_t S : 1	第 1 レベル記述子（セクション）の S ビット
uint32_t nG : 1	第 1 レベル記述子（セクション）の nG ビット
uint32_t b18: 1	第 1 レベル記述子（セクション）の bit18
uint32_t NS : 1	第 1 レベル記述子（セクション）の NS ビット
uint32_t base_addr : 12	第 1 レベル記述子（セクション）の PA[31:20]ビット

6.11 変数一覧

表 6.17にグローバル変数を示します。

表6.17 グローバル変数

型	変数名	内容
st_spibsc_spimd_reg_t	g_spibsc_spimd_reg	SPIBSC SPI モード動作設定内容格納変数 • SPIBSC SPI モードで使用する場合に、SPIBSC 設定内容を格納します。 サンプルコードでは、API 関数およびユーザ定義関数内でシリアルフラッシュ制御関数を実行する際の引数として共用で使用しています。
uint8_t	write_page_buf[]	SPIBSC のサンプルコマンドで使用するライトバッファ
uint8_t	read_buf[]	SPIBSC のサンプルコマンドで使用するリードバッファ
char_t	arg_buf[]	SPIBSC のサンプルコマンドでコンソールからの入力文字列を格納するバッファ

6.12 関数一覧

サンプルコードは、周辺機能を使用するためのインタフェース関数（API 関数）、ユーザシステムの用途に合わせてユーザで準備が必要なユーザ定義関数（API 関数からコールされる関数）、サンプルコードを動作させるために必要なサンプル関数から構成されています。

表 6.18にサンプル関数一覧を、表 6.19にAPI 関数一覧を、表 6.20にユーザ定義関数を示します。

なお、シリアルフラッシュメモリのイレーズ、ライト、およびリードの処理を行うサンプルコマンド実行時にコールされる関数は、シリアルフラッシュメモリでは実行できないため、大容量内蔵 RAM に配置して大容量内蔵 RAM で処理を実行するようにしています。キャッシュ操作関数や MMU 変換テーブルの記述子の取得および設定関数はスキャッタローディングの処理で、SPIBSC の動作モード切り替え関数、サンプルコマンドの処理関数、およびユーザ定義関数を含む SPIBSC の制御関数は Sample_SPIBSC_WriteSectionInit 関数で、大容量内蔵 RAM に転送しています。

表6.18　サンプル関数一覧

関数名	概要
Sample_SPIBSC_Main	SPIBSC サンプルコードのメイン処理
Sample_SPIBSC_Xread	XREAD コマンド処理 （外部アドレス空間リードモードでリード）
Sample_SPIBSC_Erase	ERASE コマンド処理
Sample_SPIBSC_Write	WRITE コマンド処理
Sample_SPIBSC_Sread	SREAD コマンド処理 （SPI 動作モードでリード）
Sample_SPIBSC_WriteSectionInit	SPIBSC サンプルコマンドのセクション初期化処理
Sample_SPIBSC_ChangeModeSpi	SPI 動作モードへの切り替え処理
Sample_SPIBSC_ChangeModeXip	外部アドレス空間リードモードへの切り替え処理
Change_MMU_TTbl_SpibscSpi	SPI 動作モードで使用する SPI マルチ I/O バス空間の MMU 変換テーブルの変更処理
Change_MMU_TTbl_SpibscXip	外部アドレス空間リードモードで使用する SPI マルチ I/O バス空間の MMU 変換テーブルの変更処理
MMU_TTbl_GetIndex	MMU 変換テーブルの記述子のインデックス番号の取得処理
MMU_TTbl_GetDesc	MMU 変換テーブルの記述子の値の取得処理
MMU_TTbl_SetDesc	MMU 変換テーブルの記述子の値の設定処理
L1_I_CacheFlushAll	すべての L1 命令キャッシュのフラッシュ
L1_D_CacheWritebackFlushAll	すべての L1 データキャッシュのライトバックおよびフラッシュ
L2CacheWritebackFlushAll	すべての L2 キャッシュのライトバックおよびフラッシュ
TLB_FlushAll	すべての TLB のフラッシュ

表6.19　API 関数一覧

関数名	概要
R_SFLASH_SpibscStop	SPIBSC 停止関数 SSL をネゲートして、シリアルフラッシュメモリへのアクセスを停止します。
R_SFLASH_Exmode	SPIBSC 外部アドレス空間リードモード切り替え関数 SPIBSC を SPI 動作モードから外部アドレス空間リードモードに切り替えます。
R_SFLASH_Spimode	SPIBSC SPI 動作モード切り替え関数 SPIBSC を外部アドレス空間リードモードから SPI 動作モードに切り替えます。
R_SFLASH_Spimode_Init	SPIBSC SPI 動作モード初期化関数 SPIBSC を SPI 動作モードで使用するために必要な初期設定を行います。初期設定後、SPI 動作モードに設定します。
R_SFLASH_EraseSector	シリアルフラッシュメモリのセクタイレーズ関数 シリアルフラッシュメモリにブロックイレーズコマンドを発行し、イレーズ処理を行います。 SPIBSC を SPI 動作モードにして実行してください。
R_SFLASH_ByteProgram	シリアルフラッシュメモリのライト関数 シリアルフラッシュメモリにバイトプログラムコマンドを発行し、ライト処理を行います。 SPIBSC を SPI 動作モードにして実行してください。
R_SFLASH_ByteRead	シリアルフラッシュメモリのリード関数 シリアルフラッシュメモリにリードコマンドを発行し、リード処理を行います。 SPIBSC を SPI 動作モードにして実行してください。
R_SFLASH_Ctrl_Protect	シリアルフラッシュメモリプロテクト制御関数 シリアルフラッシュメモリのレジスタを設定し、プロテクトの設定または解除を行います。 SPIBSC を SPI 動作モードにして実行してください。
R_SFLASH_Spibsc_Transfer	シリアルフラッシュメモリへのコマンド発行関数 引数の内容にしたがってシリアルフラッシュメモリにコマンドを発行します。 SPIBSC を SPI 動作モードにして実行してください。

表6.20　ユーザ定義関数

関数名	概要
Userdef_SFLASH_Write_Enable	シリアルフラッシュメモリライト許可関数 使用するシリアルフラッシュメモリに合わせて、シリアルフラッシュメモリのレジスタへのライトを許可する処理を実装してください。 サンプルコードでは、Macronix社製シリアルフラッシュメモリ（MX25L51245G）に"Write Status Register(WRSR)"コマンドを発行する処理を行っています。
Userdef_SFLASH_Busy_Wait	シリアルフラッシュメモリライト完了待ち関数 使用するシリアルフラッシュメモリに合わせて、シリアルフラッシュメモリのレジスタを読み出し、シリアルフラッシュメモリのライト完了を待つ処理を実装してください。 サンプルコードでは、Macronix社製シリアルフラッシュメモリ（MX25L51245G）に"Read Status Register(RDSR)"コマンドを発行し、Status Register の内容を参照することでライト完了を待つ処理を行っています。
Userdef_SFLASH_Ctrl_Protect	シリアルフラッシュメモリプロテクト解除関数 使用するフラッシュメモリに合わせて、シリアルフラッシュメモリのレジスタにプロテクトを解除する処理を実装してください。 サンプルコードでは、Macronix社製シリアルフラッシュメモリ（MX25L51245G）のプロテクトを解除しています。

6.13 関数仕様

サンプルコードの関数仕様を示します。

Sample_SPIBSC_Main		
概　要	SPIBSC サンプルコードのメイン処理	
宣　言	int32_t Sample_SPIBSC_Main(int32_t argc, char_t **argv)	
説　明	JASMINE ボードとシリアルインタフェースで接続されたホスト PC 上のターミナルに SPIBSC サンプルコードの情報を表示します。サンプルコマンドの入力を待つ処理の前に、Sample_SPIBSC_WriteSectionInit 関数をコールしてサンプルコマンドの処理を大容量内蔵 RAM に転送します。 以下のコマンドが入力された場合、それぞれのサンプルコマンドの処理を行います。 "XREAD"＋"Enter"キーの入力　：XREAD コマンドを実行 "ERASE"＋"Enter"キーの入力　：ERASE コマンドを実行 "WRITE"＋"Enter"キーの入力　：WRITE コマンドを実行 "SREAD"＋"Enter"キーの入力　：SREAD コマンドを実行	
引　数	int32_t argc	ターミナルから入力されたコマンド引数の数 本関数内では使用しません。
	char_t **argv	ターミナルから入力されたコマンドへのポインタ 本関数内では使用しません。
リターン値	COMMAND_EXIT	：SPIBSC サンプルコード処理の終了

Sample_SPIBSC_Xread		
概　要	XREAD コマンドの処理	
宣　言	int32_t Sample_SPIBSC_Xread(int32_t argc, char_t **argv)	
説　明	本関数は、ターミナルより入力した SPI マルチ I/O バス空間のアドレス、バイト数の領域を CPU が直接リードし、リードしたデータをターミナルに表示します。 H'1800_0000～H'1BFF_FFFF の範囲で、アドレスおよびバイト数を指定することができます。バイト数には最大 4K バイトを指定することができます。 本関数実行時は、SPIBSC は外部アドレス空間リードモードで動作します。本関数は、SPIBSC サンプルコードの起動後に、ターミナルより"XREAD"+"Enter"キーの入力時にコールされます。	
引　数	int32_t argc	ターミナルから入力されたコマンド引数の数 本関数内では使用しません。
	char_t **argv	ターミナルから入力されたコマンドへのポインタ 本関数内では使用しません。
リターン値	0　：正常終了	
	-1　：エラー終了	

Sample_SPIBSC_Erase		
概　要	ERASE コマンドの処理	
宣　言	int32_t Sample_SPIBSC_Erase(int32_t argc, char_t **argv)	
説　明	SPIBSC を SPI モード動作に切り替え後、ターミナルより入力したアドレス、バイト数からシリアルフラッシュメモリをイレーズする処理を行います。H'0010_0000～H'03FF_FFFF の範囲で、アドレスおよびバイト数を指定することができます。バイト数には最大 1M バイトを指定することができます。 サンプルコードでは、Macronix社製シリアルフラッシュメモリ（MX25L51245G）のセクタサイズ（4K バイト）をもとにイレーズするブロックを計算し、セクタサイズごとに R_SFLASH_EraseSector 関数をコールしてイレーズ処理を行っています。 イレーズ処理の後、SPIBSC を外部アドレス空間リードモードに切り替え、本関数を終了します。 SPIBSC サンプルコードの起動後に、ターミナルより"ERASE"+"Enter"キーの入力時にコールされます。	
引　数	int32_t argc	ターミナルから入力されたコマンド引数の数 本関数内では使用しません。
	char_t **argv	ターミナルから入力されたコマンドへのポインタ 本関数内では使用しません。
リターン値	0　：正常終了	
	-1　：エラー終了	

Sample_SPIBSC_Write		
概　要	WRITE コマンドの処理	
宣　言	int32_t Sample_SPIBSC_Write(int32_t argc, char_t **argv)	
説　明	SPIBSC を SPI モード動作に切り替え後、ターミナルより入力したアドレス、バイト数の領域に、入力したデータをシリアルフラッシュメモリにライトする処理を行います。H'0010_0000～H'03FF_FFFF の範囲で、アドレスおよびバイト数を指定することができます。アドレスにはシリアルフラッシュメモリのページの先頭アドレス（256 バイトの倍数）を、バイト数には最大 1M バイトを、データにはライトする連番データの開始値を 0～255 の範囲の値を指定し、指定したデータから 0～255 の範囲でインクリメントした連番データを生成します。 サンプルコードでは、Macronix社製シリアルフラッシュメモリ（MX25L51245G）のページサイズ（256 バイト）ごとに R_SFLASH_ByteProgram 関数をコールしてライト処理を行っています。 ライト処理の後、SPIBSC を外部アドレス空間リードモードに切り替え、本関数を終了します。 SPIBSC サンプルコードの起動後に、ターミナルより"WRITE"+"Enter"キーの入力時にコールされます。本関数をコールする前に、シリアルフラッシュメモリのライトする領域のイレーズ処理が行われている必要があります。	
引　数	int32_t argc	ターミナルから入力されたコマンド引数の数 本関数内では使用しません。
	char_t **argv	ターミナルから入力されたコマンドへのポインタ 本関数内では使用しません。
リターン値	0	：正常終了
	-1	：エラー終了

Sample_SPIBSC_Sread		
概　要	SREAD コマンドの処理	
宣　言	int32_t Sample_SPIBSC_Sread(int32_t argc, char_t **argv)	
説　明	SPIBSC を SPI モード動作に切り替え後、ターミナルより入力したアドレス、バイト数の領域を、シリアルフラッシュメモリからリードし、リードしたデータをターミナルに表示します。H'0000_0000～H'03FF_FFFF の範囲で、アドレスおよびバイト数を指定することができます。バイト数には最大 4K バイトを指定することができます。 リード処理の後、SPIBSC を外部アドレス空間リードモードに切り替え、本関数を終了します。 SPIBSC サンプルコードの起動後に、ターミナルより"SREAD"+"Enter"キーの入力時にコールされます。	
引　数	int32_t argc	ターミナルから入力されたコマンド引数の数 本関数内では使用しません。
	char_t **argv	ターミナルから入力されたコマンドへのポインタ 本関数内では使用しません。
リターン値	0	：正常終了
	-1	：エラー終了

Sample_SPIBSC_WriteSectionInit	
概　要	SPIBSC サンプルコマンドのセクション初期化処理
宣　言	void Sample_SPIBSC_WriteSectionInit(void)
説　明	サンプルコマンドで実行する処理を大容量内蔵 RAM に転送します。SPIBSC を SPI 動作モードに切り替え後に、大容量内蔵 RAM 領域でサンプルコマンドの処理が行われるようにしています。
引　数	なし
リターン値	なし

Sample_SPIBSC_ChangeModeSpi	
概　要	SPI 動作モードへの切り替え処理
宣　言	int32_t Sample_SPIBSC_ChangeModeSpi(void)
説　明	SPIBSC を外部アドレス空間リードモードから SPI 動作モードに切り替えるための処理を行います。Change_MMU_TTbl_SpibscSpi 関数をコールして、SPI マルチ I/O バス空間の MMU の変換テーブルを、アクセス不可、実行不可の属性に設定し、SPI マルチ I/O バス空間へのアクセスが発生しないようにします。また、MMU 変換テーブルの設定後、キャッシュ操作を行うことにより、キャッシュの内容を更新します。その後、R_SFLASH_Spimode 関数をコールして、SPIBSC の CMNCR レジスタの MD ビットを 1 に設定し、CMNCR レジスタをダミーリードし、SPI 動作モードに切り替えます。
引　数	なし
リターン値	0　：正常終了 -1　：エラー終了

Sample_SPIBSC_ChangeModeXip	
概　要	外部アドレス空間リードモードへの切り替え処理
宣　言	int32_t Sample_SPIBSC_ChangeModeXip(void)
説　明	SPIBSC を SPI 動作モードから外部アドレス空間リードモードに切り替えるための処理を行います。R_SFLASH_Exmode 関数をコールして、TEND ビットをリードすることでシリアルフラッシュメモリへのアクセスが発生していないことを確認します。DRCR レジスタの RCF ビットに 1 をライトして DRCR レジスタをダミーリードしリードキャッシュをクリアします。CMNCR レジスタの MD ビットを 0 に設定し、CMNCR レジスタをダミーリードし、外部アドレス空間リードモードに切り替えます。その後、Change_MMU_TTbl_SpibscXip 関数をコールして、SPI マルチ I/O バス空間の MMU の変換テーブルを、アクセス可能、実行可能の属性に設定し、SPI マルチ I/O バス空間にアクセスができるようにします。また、MMU 変換テーブルの設定後、キャッシュ操作を行うことにより、キャッシュの内容を更新します。
引　数	なし
リターン値	0　：正常終了 -1　：エラー終了

Change_MMU_TTbl_SpibscSpi		
概　要	SPI 動作モードで使用する SPI マルチ I/O バス空間の MMU 変換テーブルの変更処理	
宣　言	static int32_t Change_MMU_TTbl_SpibscSpi(uint32_t start_addr, uint32_t end_addr)	
	SPI 動作モードで使用する SPI マルチ I/O バス空間（H'1800_0000～H'1BFF_FFFF）の MMU 変換テーブルをアクセス不可、実行不可の属性に変更するために、AP[2:0]ビットおよび XN ビットを以下のように設定します。 AP[2:0]ビット　：B'000（アクセス不可） XN ビット　：1（実行不可）	
引　数	uint32_t start_addr	MMU 変換テーブル変更処理の対象となるアドレス範囲の開始アドレス
	uint32_t end_addr	MMU 変換テーブル変更処理の対象となるアドレス範囲の終了アドレス
リターン値	0　：正常終了	
	-1　：エラー終了	

Change_MMU_TTbl_SpibscXip		
概　要	外部アドレス空間リードモードで使用する SPI マルチ I/O バス空間の MMU 変換テーブルの変更処理	
宣　言	static int32_t Change_MMU_TTbl_SpibscXip(uint32_t start_addr, uint32_t end_addr)	
	外部アドレス空間リードモードで使用する SPI マルチ I/O バス空間（H'1800_0000～H'1BFF_FFFF）の MMU 変換テーブルをアクセス可能、実行可能の属性に変更するために、AP[2:0]ビットおよび XN ビットを、以下のように設定します。 AP[2:0]ビット　：B'011（アクセス可能） XN ビット　：0（実行可能）	
引　数	uint32_t start_addr	MMU 変換テーブル変更処理の対象となるアドレス範囲の開始アドレス
	uint32_t end_addr	MMU 変換テーブル変更処理の対象となるアドレス範囲の終了アドレス
リターン値	0　：正常終了	
	-1　：エラー終了	

MMU_TTbl_GetIndex		
概　要	MMU 変換テーブルの記述子のインデックス番号の取得処理	
宣　言	uint32_t MMU_TTbl_GetIndex(uint32_t addr) 引数 addr に対応する MMU 変換テーブルのセクションタイプの記述子のインデックス番号を取得します。	
引　数	uint32_t addr	記述子の内容を変更する仮想アドレス （H'0000_0000～H'FFFF_FFFF）
リターン値	引数 addr に対応する記述子のインデックス番号（0～4095）	

MMU_TTbl_GetDesc		
概　要	MMU 変換テーブルの記述子の値の取得処理	
宣　言	int32_t MMU_TTbl_GetDesc(uint32_t index, mmu_ttbl_desc_section_t *pdesc) 引数 index に対応する MMU 変換テーブルのセクションタイプの記述子の値を取得し、引数*pdesc に格納します。	
引　数	uint32_t index	MMU 変換テーブルのインデックス番号（0～4095）
	mmu_ttbl_desc_section_t *pdesc	取得した記述子を格納する変数へのポインタ
リターン値	0	：正常終了
	-1	：エラー終了

MMU_TTbl_SetDesc		
概　要	MMU 変換テーブルの記述子の値の設定	
宣　言	int32_t MMU_TTbl_SetDesc(uint32_t index, mmu_ttbl_desc_section_t *pdesc) 引数 index に対応する MMU 変換テーブルのセクションタイプの記述子に、引数*pdesc に格納された値を設定します。	
引　数	uint32_t index	MMU 変換テーブルのインデックス番号（0～4095）
	mmu_ttbl_desc_section_t *pdesc	記述子に設定する値を格納された変数へのポインタ
リターン値	0	：正常終了
	-1	：エラー終了

L1_I_CacheFlushAll	
概　要	すべての L1 命令キャッシュのフラッシュ
宣　言	void L1_I_CacheFlushAll(void)
説　明	すべての L1 命令キャッシュのフラッシュを行います。
引　数	なし
リターン値	なし

L1_D_CacheWritebackFlushAll	
概　要	すべての L1 データキャッシュのライトバックおよびフラッシュ
宣　言	void L1_D_CacheWritebackFlushAll(void)
説　明	すべての L1 データキャッシュのライトバックおよびフラッシュを行います。 サンプルコードでは、セットおよびウェイにより、すべてのデータキャッシュラインのクリーニングおよび無効化を行っています。
引　数	なし
リターン値	なし

L2CacheWritebackFlushAll	
概　要	すべての L2 キャッシュのライトバックおよびフラッシュ
宣　言	void L2CacheWritebackFlushAll(void)
説　明	すべての L2 キャッシュのライトバックおよびフラッシュを行います。 サンプルコードでは、ウェイにより、すべてのキャッシュラインのクリーニングおよび無効化を行っています。
引　数	なし
リターン値	なし

TLB_FlushAll	
概　要	すべての TLB のフラッシュ
宣　言	void TLB_FlushAll(void)
	CP15 の TLBIALL を実行して、すべての TLB のフラッシュを行います。
引　数	なし
リターン値	なし

R_SFLASH_SpibscStop		
概　要	SPIBSC 停止関数	
宣　言	int32_t R_SFLASH_SpibscStop(uint32_t ch_no)	
説　明	SPIBSC を外部アドレス空間リードモード設定時に本関数をコールした場合、データリードコントロールレジスタ（DRCR）の SSLN ビットに 1 を設定し、SSL をネゲートしてシリアルフラッシュメモリへのアクセスを停止します。	
引　数	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
リターン値	0	：正常終了
	-1	：エラー終了

R_SFLASH_Exmode		
概　要	SPIBSC 外部アドレス空間リードモード切り替え関数	
宣　言	int32_t R_SFLASH_Exmode(uint32_t ch_no)	
説　明	SPIBSC を外部アドレス空間リードモードに切り替えます。外部アドレス空間リードモードに切り替え後、SPI マルチ I/O バス空間をリードする前に、リードキャッシュの全エントリをクリアします。	
引　数	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
リターン値	0	：正常終了
	-1	：エラー終了

R_SFLASH_Spimode		
概　要	SPIBSC SPI 動作モード切り替え関数	
宣　言	int32_t R_SFLASH_Spimode(uint32_t ch_no)	
説　明	SPIBSC を SPI 動作モードに切り替えます。SPI 動作モードに切り替え後、リードキャッシュの全エントリをクリアします。	
引　数	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
リターン値	0	：正常終了
	-1	：エラー終了

R_SFLASH_Spimode_Init

概　要	SPIBSC SPI 動作モード初期化関数	
宣　言	int32_t R_SFLASH_Spimode_Init(uint32_t ch_no, uint32_t dual, uint8_t data_width, uint8_t spbr, uint8_t brdv, uint8_t addr_mode)	
説　明	SPIBSC を SPI 動作モードとして動作するように初期化処理を行います。	
引　数	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
	uint32_t dual	チャネルに接続しているシリアルフラッシュの個数 SPIBSC_CMNCR_BSZ_SINGLE：1 個 SPIBSC_CMNCR_BSZ_DUAL：2 個
	uint8_t data_width	SPIBSC とシリアルフラッシュメモリのデータ転送のバス幅 SPIBSC_1BIT：1 ビット SPIBSC_4BIT：4 ビット
	uint8_t spbr	ビットレート設定 ビットレート設定レジスタ（SPBCR）のSPBR[7:0]ビットへの設定値を指定してください。 ビットレート分周設定（brdv）と合わせてビットレートを設定します。
	uint8_t brdv	ビットレート分周設定 ビットレート設定レジスタ（SPBCR）のBRDV[1:0]の設定値を指定してください。 ビットレート設定（spbr）と合わせてシリアルクロック（SPBCLK）のビットレートを設定します。
	uint8_t addr_mode	シリアルフラッシュメモリに発行するアドレスのビット幅 コマンド発行時に出力するアドレスのビット幅を指定します。 SPIBSC_OUTPUT_ADDR_24：24 ビットアドレス出力 SPIBSC_OUTPUT_ADDR_32：32 ビットアドレス出力
リターン値	0　：正常終了	
	-1　：エラー終了	

表6.21　引数 spbr と brdv の設定例

spbr の設定値(n)	brdv の設定値(N)	分周比	ビットレート		
			Bφ=100MHz	Bφ=128MHz	Bφ=133.33MHz
0	0	1	設定禁止		
1	0	2	50Mbps	64Mbps	66.67Mbps
2	0	4	25Mbps	32Mbps	33.33Mbps
3	0	6	16.67Mbps	21.33Mbps	22.22Mbps
4	0	8	12.5Mpbs	16Mpbs	16.67Mpbs

R_SFLASH_EraseSector

概　要	シリアルフラッシュメモリのセクタイレーズ関数	
宣　言	int32_t R_SFLASH_EraseSector(uint32_t addr, uint32_t ch_no, uint32_t dual uint8_t data_width, uint8_t addr_mode)	
説　明	引数で指定されたシリアルフラッシュメモリのアドレスが属するセクタ単位でデータをイレーズします。シリアルフラッシュメモリのセクタサイズの単位で本関数をコールしてください。 SPIBSC を SPI 動作モードに切り替えた後に実行してください。	
引　数	uint32_t addr	イレーズする先頭アドレス（H'0000_0000～H'03FF_FFFF）
	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
	uint32_t dual	チャネルに接続しているシリアルフラッシュの個数 SPIBSC_CMNCR_BSZ_SINGLE：1 個 SPIBSC_CMNCR_BSZ_DUAL：2 個
	uint8_t data_width	SPIBSC とシリアルフラッシュメモリのデータ転送のバス幅 SPIBSC_1BIT：1 ビット SPIBSC_4BIT：4 ビット
	uint8_t addr_mode	シリアルフラッシュメモリに発行するアドレスのビット幅 コマンド発行時に出力するアドレスのビット幅を指定します。 SPIBSC_OUTPUT_ADDR_24：24 ビットアドレス出力 SPIBSC_OUTPUT_ADDR_32：32 ビットアドレス出力
リターン値	0　：正常終了	
	-1　：エラー終了	

R_SFLASH_ByteProgram		
概　要	シリアルフラッシュメモリのライト関数	
宣　言	int32_t R_SFLASH_ByteProgram(uint32_t addr, uint8_t *buf, int32_t size, uint32_t ch_no, uint32_t dual, uint8_t data_width, uint8_t addr_mode)	
説　明	引数で指定されたパラメータに合わせてシリアルフラッシュメモリにライトします。シリアルフラッシュメモリにページ単位でライト処理を行います。引数 size に設定する値がシリアルフラッシュメモリのページサイズ以下となるように本関数をコールしてください。 SPIBSC を SPI 動作モードに切り替えた後に実行してください。	
引　数	uint32_t addr	ライトする先頭アドレス（H'0000_0000～H'03FF_FFFF）
	uint8_t *buf	ライトデータが格納されたバッファ
	int32_t size	ライトするバイト数
	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
	uint32_t dual	チャネルに接続しているシリアルフラッシュの個数 SPIBSC_CMNCR_BSZ_SINGLE：1 個 SPIBSC_CMNCR_BSZ_DUAL：2 個
	uint8_t data_width	SPIBSC とシリアルフラッシュメモリのデータ転送のバス幅 SPIBSC_1BIT：1 ビット SPIBSC_4BIT：4 ビット
	uint8_t addr_mode	シリアルフラッシュメモリに発行するアドレスのビット幅 コマンド発行時に出力するアドレスのビット幅を指定します。 SPIBSC_OUTPUT_ADDR_24：24 ビットアドレス出力 SPIBSC_OUTPUT_ADDR_32：32 ビットアドレス出力
リターン値	0　：正常終了	
	-1　：エラー終了	

R_SFLASH_ByteRead		
概　要	シリアルフラッシュメモリのリード関数	
宣　言	int32_t R_SFLASH_ByteRead(uint32_t addr, uint8_t *buf, int32_t size, uint32_t ch_no, uint32_t dual, uint8_t data_width, uint8_t addr_mode)	
説　明	引数で指定されたパラメータに合わせてシリアルフラッシュメモリからリードします。 SPIBSC を SPI 動作モードに切り替えた後に実行してください。	
引　数	uint32_t addr	リードする先頭アドレス（H'0000_0000～H'03FF_FFFF）
	uint8_t *buf	リードデータが格納されたバッファ
	int32_t size	リードするバイト数
	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
	uint32_t dual	チャネルに接続しているシリアルフラッシュの個数 SPIBSC_CMNCR_BSZ_SINGLE：1 個 SPIBSC_CMNCR_BSZ_DUAL：2 個
	uint8_t data_width	SPIBSC とシリアルフラッシュメモリのデータ転送のバス幅 SPIBSC_1BIT：1 ビット SPIBSC_4BIT：4 ビット
	uint8_t addr_mode	シリアルフラッシュメモリに発行するアドレスのビット幅 コマンド発行時に出力するアドレスのビット幅を指定します。 SPIBSC_OUTPUT_ADDR_24：24 ビットアドレス出力 SPIBSC_OUTPUT_ADDR_32：32 ビットアドレス出力
リターン値	0　：正常終了	
	-1　：エラー終了	

R_SFLASH_Ctrl_Protect

概　要	シリアルフラッシュメモリのプロテクト制御関数	
宣　言	int32_t R_SFLASH_Ctrl_Protect(en_sf_req_t req, uint32_t ch_no, uint32_t dual, uint8_t data_width)	
説　明	引数で指定されたパラメータに合わせてシリアルフラッシュメモリのステータスレジスタを設定し、プロテクトの設定または解除を行います。 SPIBSC を SPI 動作モードに切り替えた後に実行してください。	
引　数	en_sf_req_t req	レジスタ設定情報 SF_REQ_PROTECT：プロテクトに設定 SF_REQ_UNPROTECT：プロテクト解除に設定
	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
	uint32_t dual	チャネルに接続しているシリアルフラッシュの個数 SPIBSC_CMNCR_BSZ_SINGLE：1 個 SPIBSC_CMNCR_BSZ_DUAL：2 個
	uint8_t data_width	SPIBSC とシリアルフラッシュメモリのデータ転送のバス幅 SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅
リターン値	0	：正常終了
	-1	：エラー終了

R_SFLASH_Spibsc_Transfer

概　要	シリアルフラッシュメモリへのコマンド発行関数	
宣　言	int32_t R_SFLASH_Spibsc_Transfer(uint32_t ch_no, st_spibsc_spimd_reg_t * regset)	
説　明	SPI 動作モードで、引数 regset に指定された内容にしたがってシリアルフラッシュメモリにコマンドを発行します。	
引　数	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
	st_spibsc_spimd_reg_t * regset	SPIBSC SPI モード設定 設定内容は、表 6.11から表 6.15を参照してください。
リターン値	0	：正常終了
	-1	：エラー終了
備　考	外部アドレス空間リードモードで本関数をコールした場合は、SPI 動作モードに切り替えて、シリアルフラッシュメモリにコマンドを発行します。	

Userdef_SFLASH_Write_Enable		
概　要	シリアルフラッシュメモリライト許可関数	
宣　言	int32_t Userdef_SFLASH_Write_Enable(uint32_t ch_no)	
説　明	使用するシリアルフラッシュメモリに合わせて、シリアルフラッシュメモリのレジスタにライトを許可する処理を実装してください。 サンプルコードでは、Macronix社製シリアルフラッシュメモリ（MX25L51245G）に"Write Status Register(WRSR)"コマンドを発行する処理を行っています。	
引　数	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
リターン値	0	：正常終了
	-1	：エラー終了

Userdef_SFLASH_Busy_Wait		
概　要	シリアルフラッシュメモリライト完了待ち関数	
宣　言	int32_t Userdef_SFLASH_Busy_Wait(uint32_t ch_no, uint32_t dual, uint8_t data_width)	
説　明	使用するシリアルフラッシュメモリに合わせて、シリアルフラッシュメモリのレジスタを読み出し、シリアルフラッシュメモリのライト完了を待つ処理を実装してください。 サンプルコードでは、Macronix社製シリアルフラッシュメモリ（MX25L51245G）に"Read Status Register(RDSR)"コマンドを発行し、Status Register の内容を参照することでライト完了を待つ処理を行っています。	
引　数	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
	uint32_t dual	チャネルに接続しているシリアルフラッシュの個数 SPIBSC_CMNCR_BSZ_SINGLE：1 個 SPIBSC_CMNCR_BSZ_DUAL：2 個
	uint8_t data_width	SPIBSC とシリアルフラッシュメモリのデータ転送のバス幅 SPIBSC_1BIT：1 ビット SPIBSC_4BIT：4 ビット
リターン値	0	：正常終了

Userdef_SFLASH_Ctrl_Protect		
概　要	シリアルフラッシュメモリプロテクト解除関数	
宣　言	int32_t Userdef_SFLASH_Ctrl_Protect(en_sf_req_t req, uint32_t ch_no, uint32_t dual, uint8_t data_width)	
説　明	使用するシリアルフラッシュメモリに合わせて、シリアルフラッシュメモリのレジスタにプロテクトを解除する処理を実装してください。 サンプルコードでは、Macronix社製シリアルフラッシュメモリ（MX25L51245G）に"Write Status Register(WRSR)"コマンドを発行し、プロテクトを解除する処理を行っています。	
引　数	en_sf_req_t req	レジスタ設定情報 SF_REQ_PROTECT：プロテクトに設定 SF_REQ_UNPROTECT：プロテクト解除に設定
	uint32_t ch_no	SPIBSC のチャネル番号（0 のみ指定可能）
	uint32_t dual	チャネルに接続しているシリアルフラッシュの個数 SPIBSC_CMNCR_BSZ_SINGLE：1 個 SPIBSC_CMNCR_BSZ_DUAL：2 個
	uint8_t data_width	SPIBSC とシリアルフラッシュメモリのデータ転送のバス幅 SPIBSC_1BIT：1 ビット幅 SPIBSC_4BIT：4 ビット幅
リターン値	0	：正常終了
	-1	：エラー終了

6.14 フローチャート

6.14.1 SPIBSC サンプルコードのメイン処理

図 6.16にSPIBSC サンプルコードのメイン処理を示します。ホスト PC 上のターミナルからの文字入力を待ち、入力されたコマンドにしたがって、SPIBSC サンプルコードのサンプルコマンドの処理に分岐します。

シリアルフラッシュメモリのイレーズ、ライト、およびリードの処理を行うサンプルコマンド実行時にコールされる関数は、シリアルフラッシュメモリでは実行できないため、大容量内蔵 RAM に配置して大容量内蔵 RAM で処理を実行するようにしています。Sample_SPIBSC_Main 関数からコールされる Sample_SPIBSC_WriteSectionInit 関数では、SPIBSC の動作モード切り替え関数、サンプルコマンドの処理関数、およびユーザ定義関数を含む SPIBSC の制御関数を大容量内蔵 RAM に転送しています。

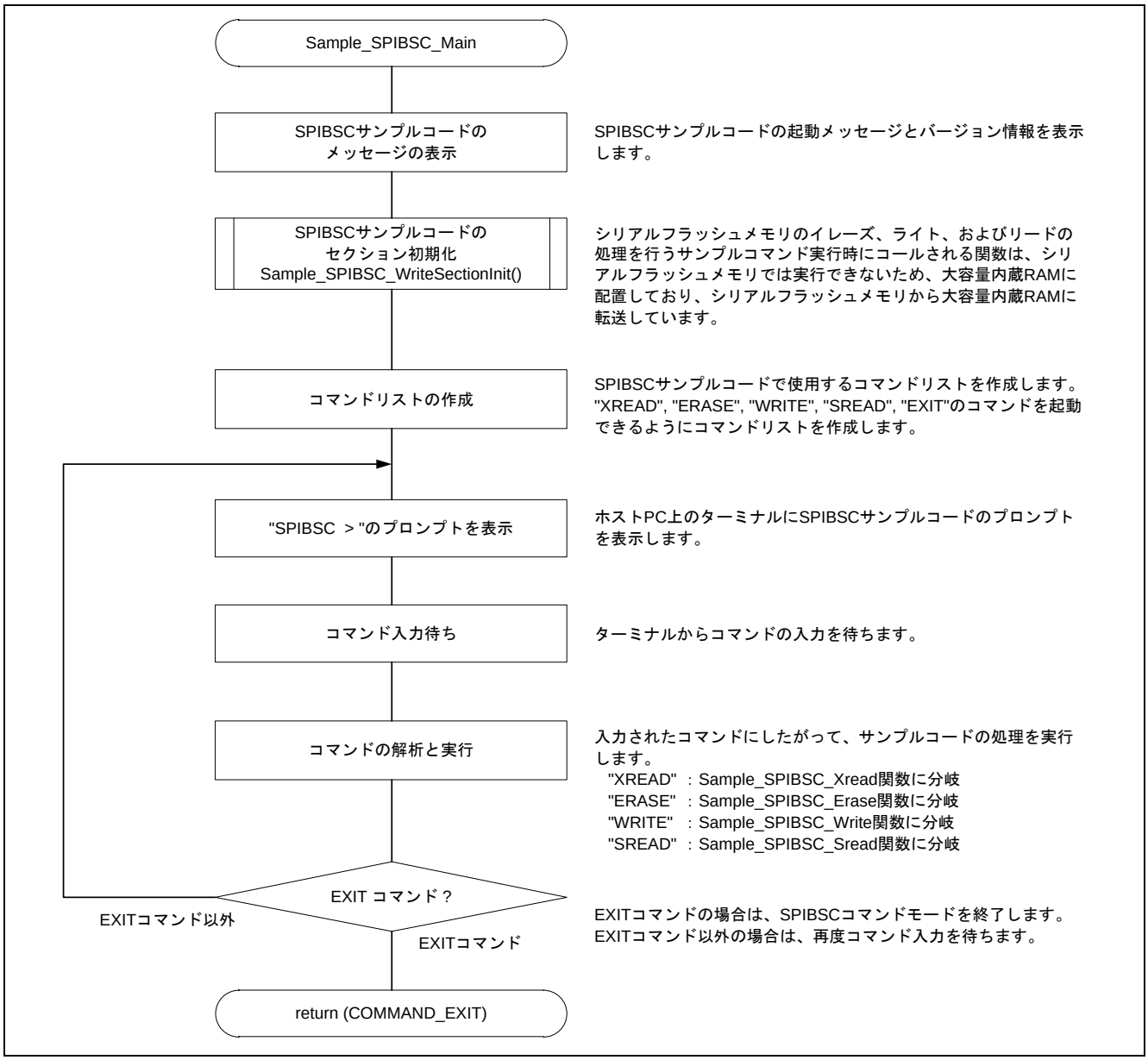


図6.16　SPIBSC サンプルコードのメイン処理

6.14.2 SPI 動作モードへの切り替え処理

図 6.17にSPI 動作モードへの切り替え処理を示します。「6.3.1 外部アドレス空間リードモードから SPI 動作モードへの切り替え」に記載している SPIBSC のモード切り替え処理を行います。

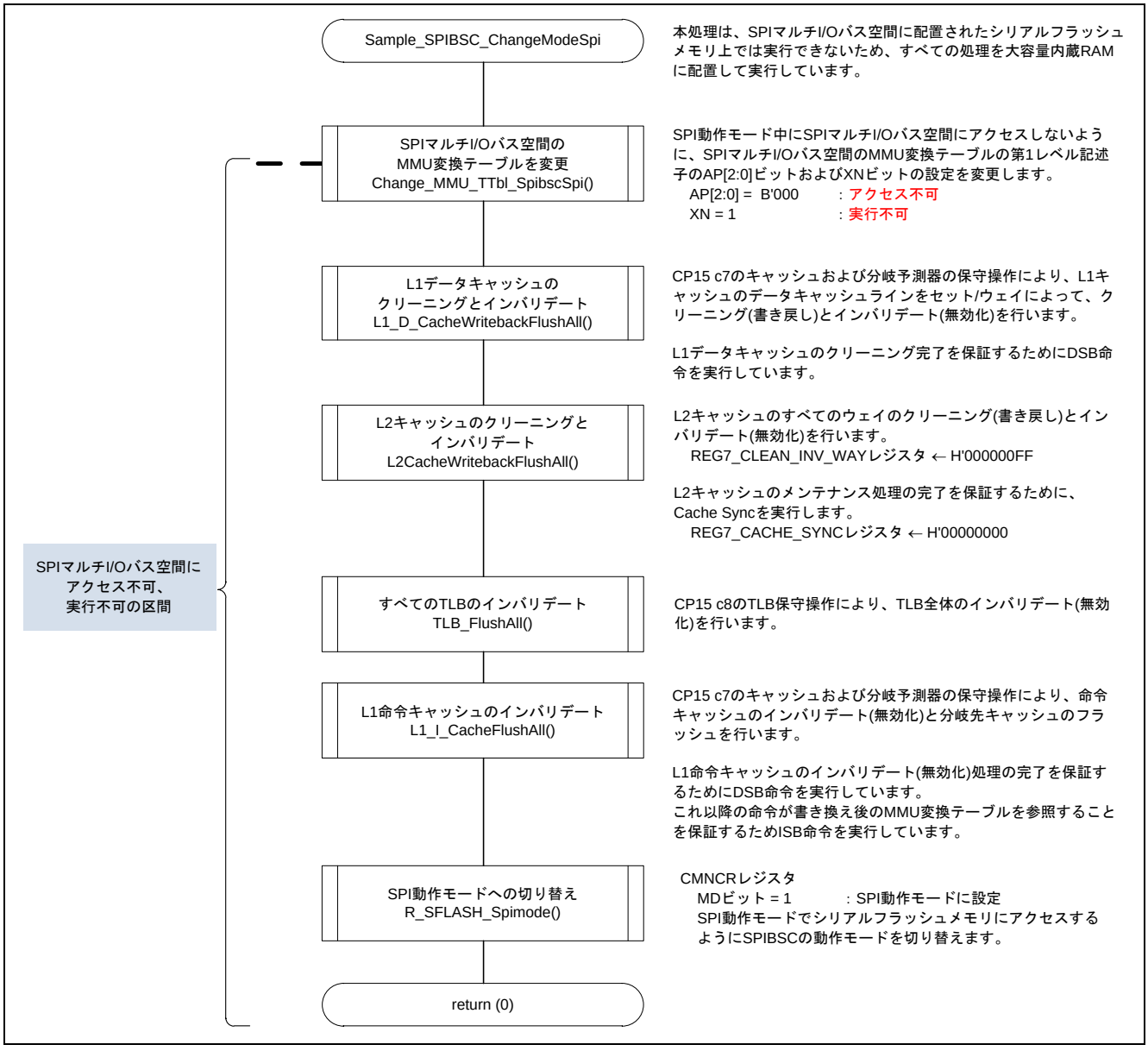


図6.17 SPI 動作モードへの切り替え処理

6.14.3 外部アドレス空間リードモードへの切り替え処理

図 6.18に外部アドレス空間リードモードへの切り替え処理を示します。「6.3.2 SPI 動作モードから外部アドレス空間リードモードへの切り替え」に記載している SPIBSC のモード切り替え処理を行います。

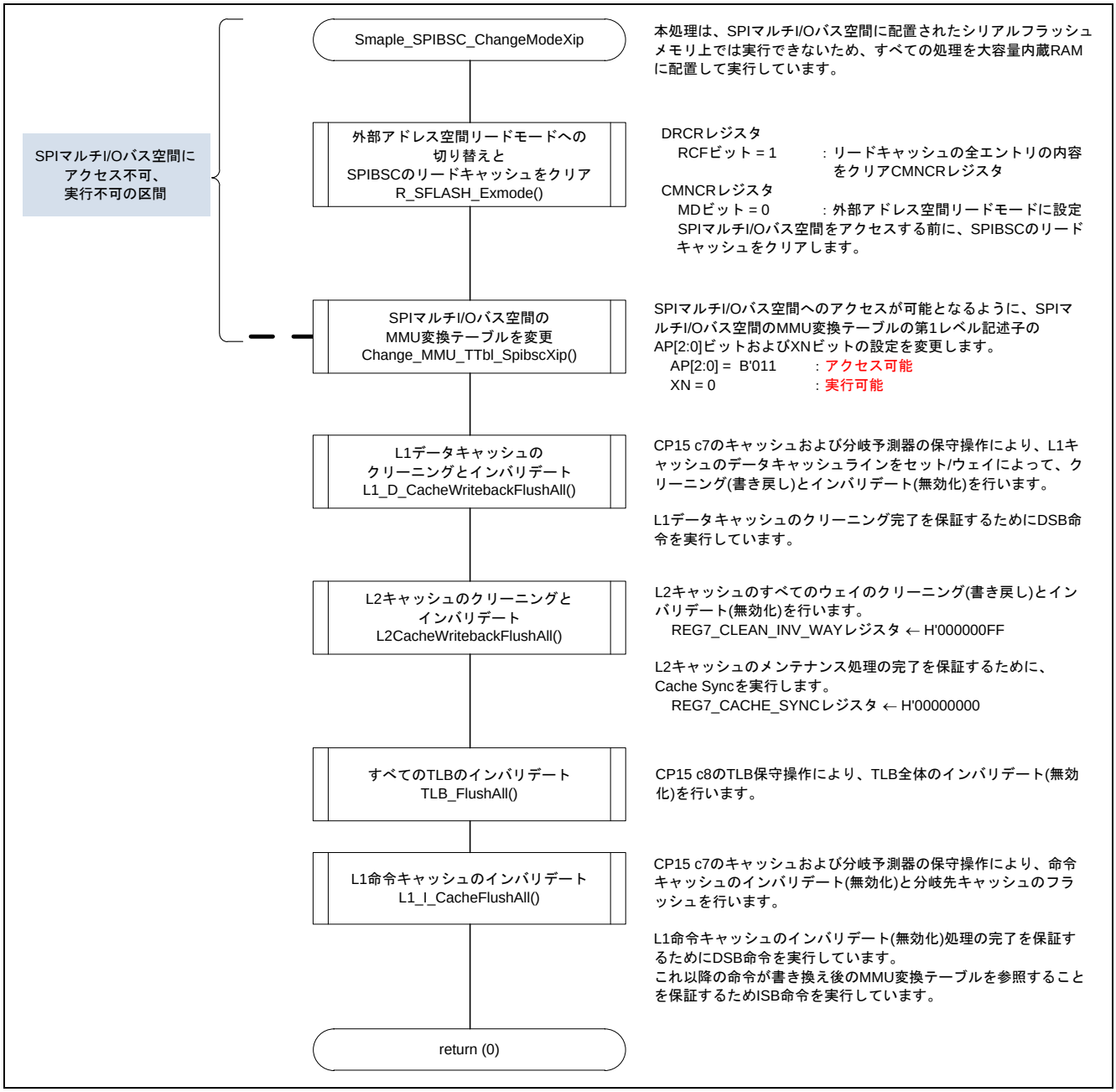


図6.18 外部アドレス空間リードモードへの切り替え処理

6.14.4 XREAD コマンド処理

図 6.19にXREAD コマンド処理を示します。

XREAD コマンドでは SPIBSC のモード切り替えは行わず、外部アドレス空間リードモードのまま動作します。XREAD コマンドは、CPU が SPI マルチ I/O バス空間をリードして、シリアルフラッシュメモリのデータを取得し、取得したデータをターミナルに表示します。

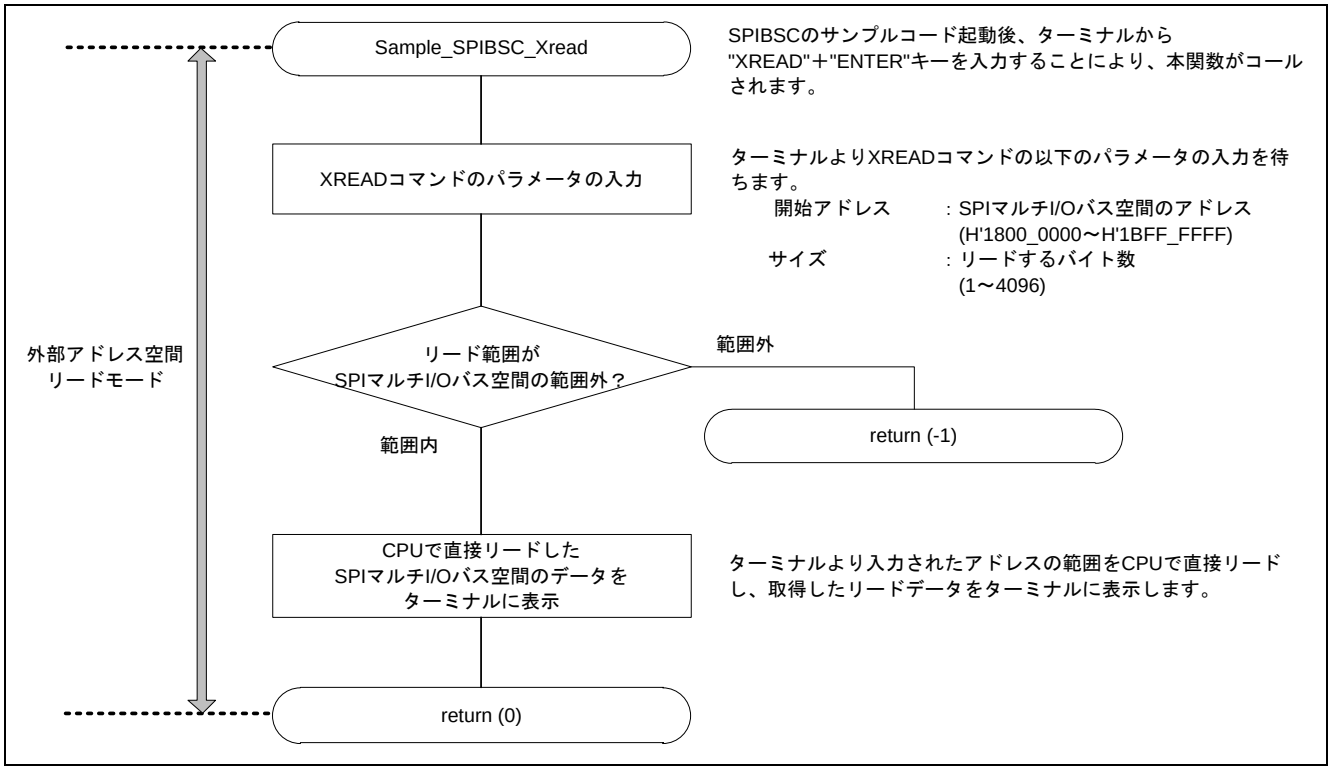


図6.19　XREAD コマンド処理

6.14.5 ERASE コマンド処理

図 6.20および図 6.21にERASE コマンド処理を示します。

ERASE コマンドでは、Sample_SPIBSC_ChangeModeSpi 関数をコールして SPIBSC を SPI 動作モードに切り替え後、R_SFLASH_EraseSector 関数をコールして、シリアルフラッシュメモリのデータを消去します。イレーズ処理完了後、Sample_SPIBSC_ChangeModeXip 関数をコールして SPIBSC を外部アドレス空間リードモードに切り替え処理を実行し、SPI マルチ I/O バス空間（シリアルフラッシュメモリ）に配置された命令やデータにアクセス可能となるように設定します。

イレーズ処理では、SPIBSC を SPI 動作モードに設定しており、シリアルフラッシュメモリに配置されたプログラムを直接実行できないため、ERASE コマンド処理は大容量内蔵 RAM に配置して実行しています。SPI 動作モードでは、SPI マルチ I/O バス空間へのアクセスはできないため、シリアルフラッシュメモリに配置されたプログラムに分岐することがないように、イレーズ処理中は IRQ 割り込みを禁止状態に設定しています。

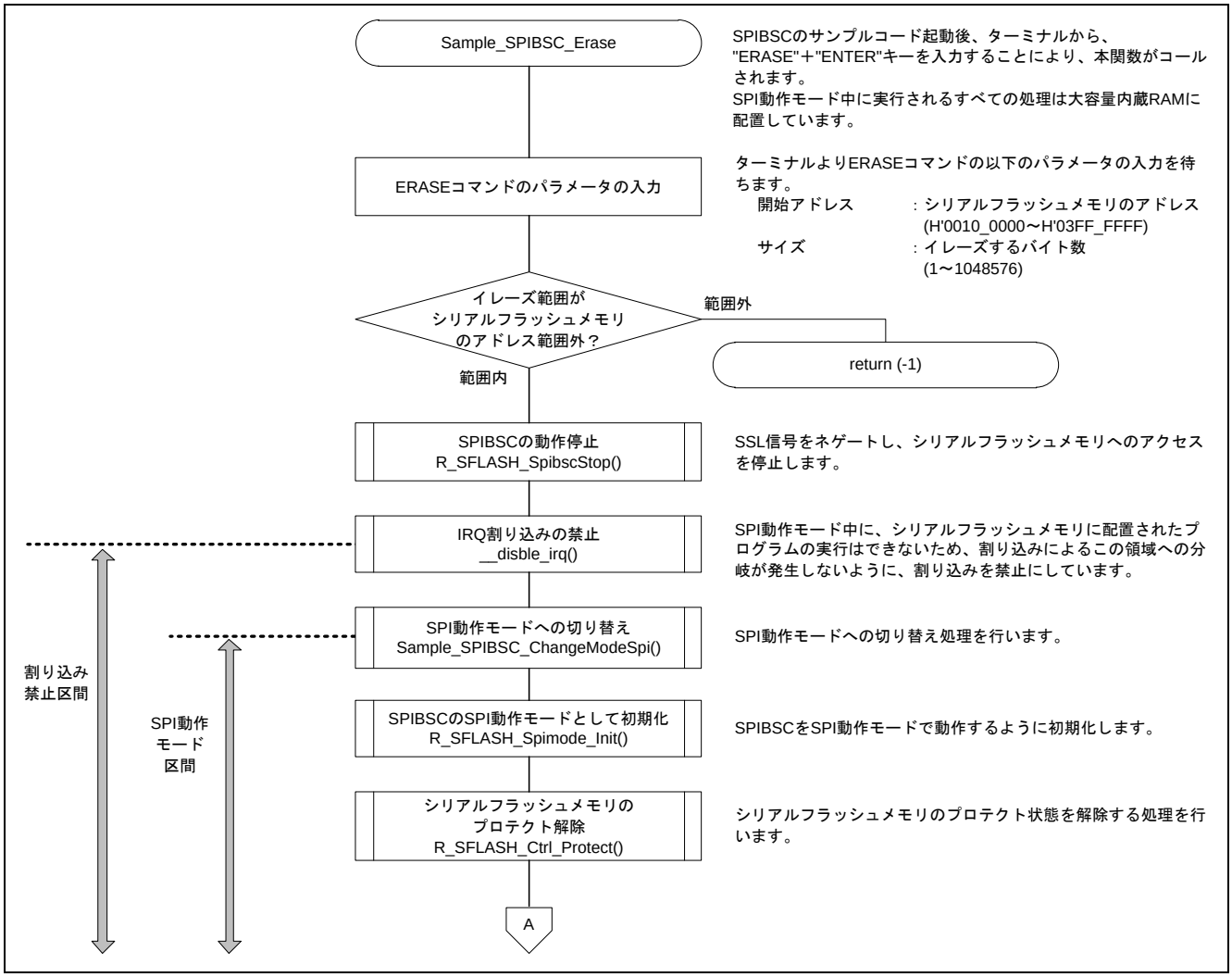


図6.20 ERASE コマンド処理（1/2）

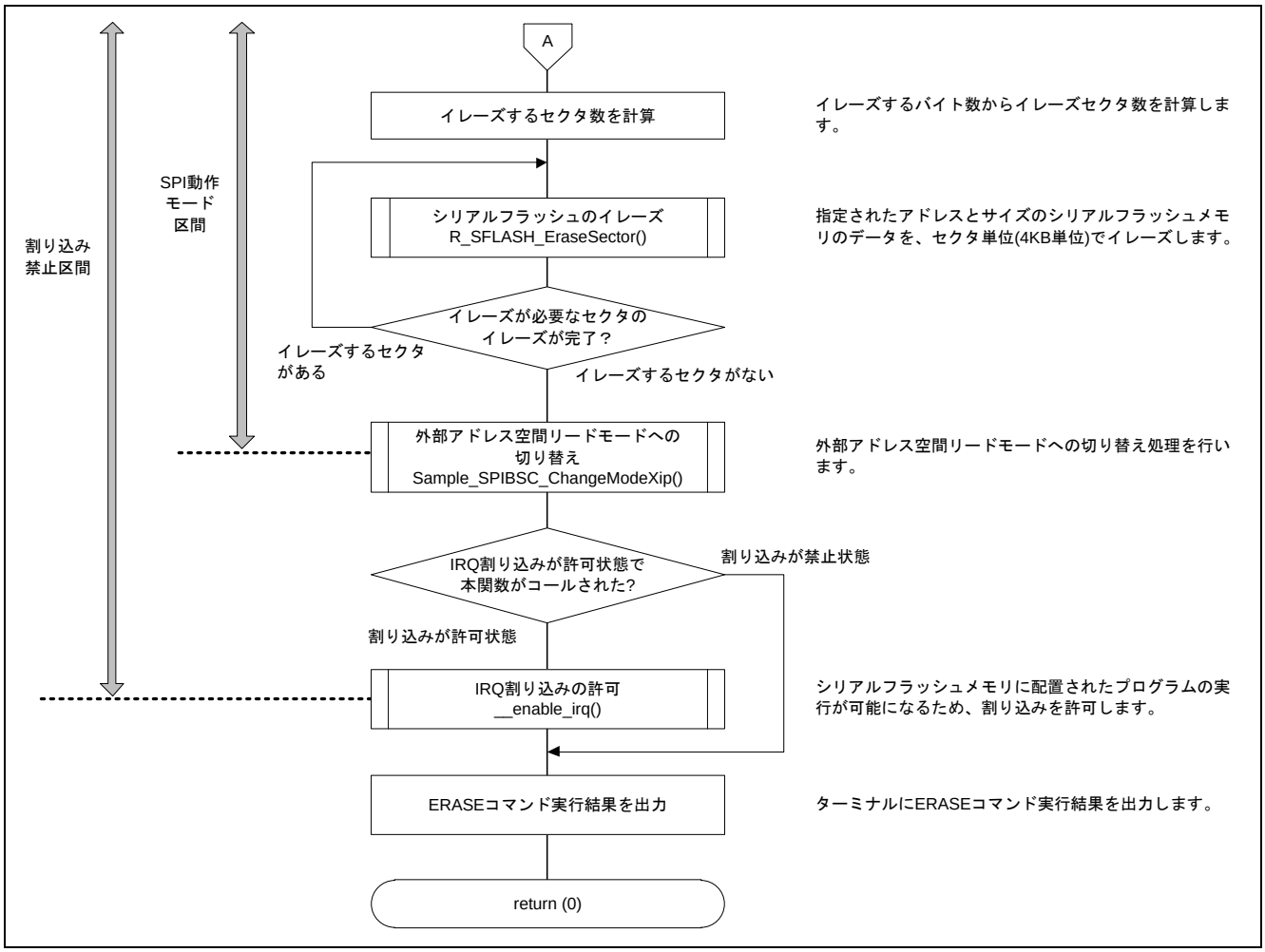


図6.21　ERASE コマンド処理（2/2）

6.14.6 WRITE コマンド処理

図 6.22および図 6.23にWRITE コマンド処理を示します。

WRITE コマンドでは、Sample_SPIBSC_ChangeModeSpi 関数をコールして SPIBSC を SPI 動作モードに切り替え後、R_SFLASH_ByteProgram 関数をコールして、シリアルフラッシュメモリのデータをライトします。ライト処理完了後、Sample_SPIBSC_ChangeModeXip 関数をコールして SPIBSC を外部アドレス空間リードモードに切り替え処理を実行し、SPI マルチ I/O バス空間（シリアルフラッシュメモリ）に配置された命令やデータにアクセス可能となるように設定します。

ライト処理では、イレーズ処理と同様に、大容量内蔵 RAM に配置して実行し、ライト処理中は IRQ 割り込みを禁止状態に設定しています。

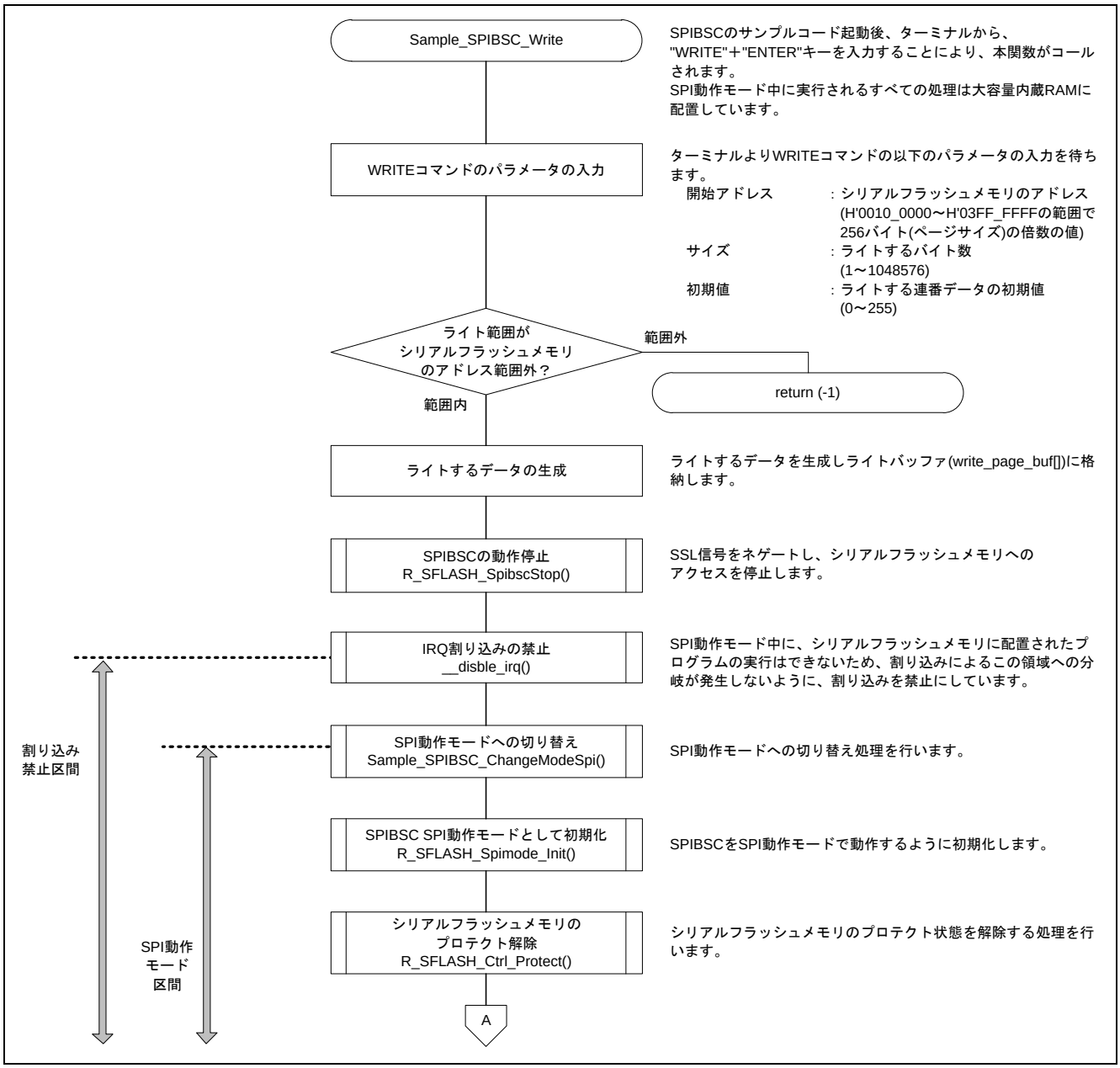


図6.22　WRITE コマンド処理（1/2）

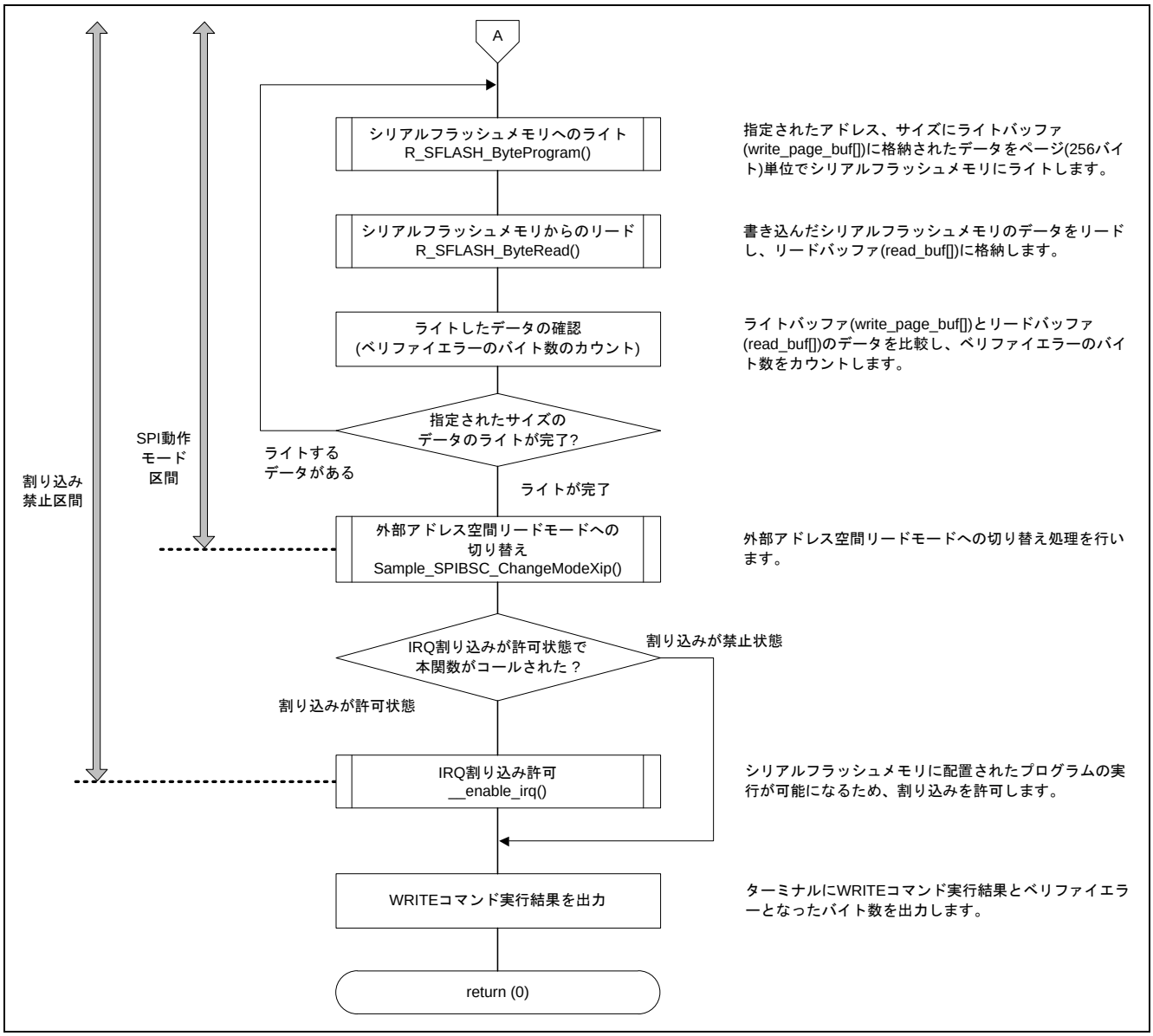


図6.23　WRITE コマンド処理（2/2）

6.14.7 SREAD コマンド処理

図 6.24および図 6.25にSREAD コマンド処理を示します。

SREAD コマンドでは、Sample_SPIBSC_ChangeModeSpi 関数をコールして SPIBSC を SPI 動作モードに切り替え後、R_SFLASH_ByteRead 関数をコールして、シリアルフラッシュメモリのデータをリードします。リード処理完了後、Sample_SPIBSC_ChangeModeXip 関数をコールして SPIBSC を外部アドレス空間リードモードに切り替え処理を実行し、SPI マルチ I/O バス空間（シリアルフラッシュメモリ）に配置された命令やデータにアクセス可能となるように設定します。

SPI 動作モードを使用したリード処理では、イレーズ処理と同様に、大容量内蔵 RAM に配置して実行し、リード処理中は IRQ 割り込みを禁止状態に設定しています。

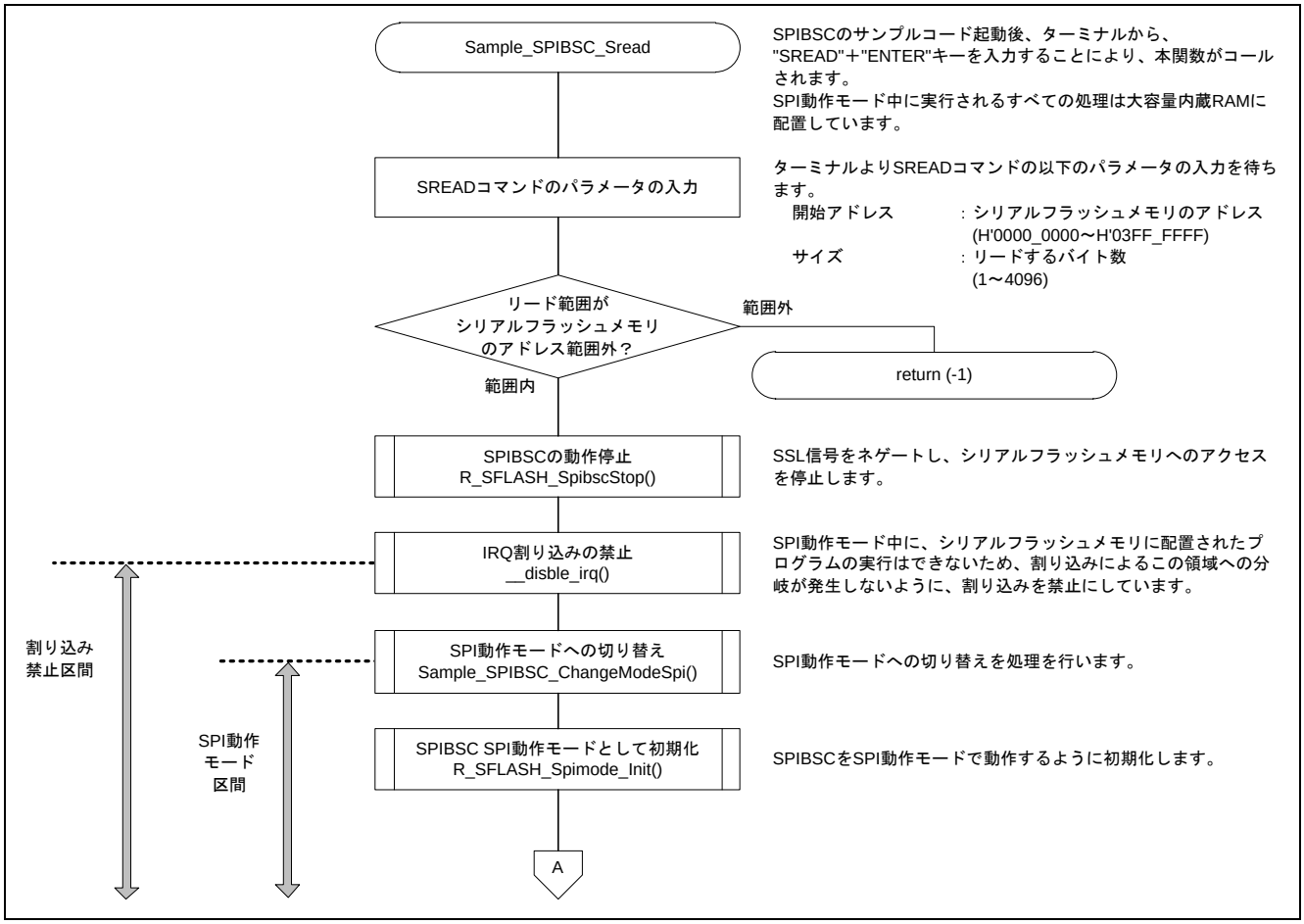


図6.24　SREAD コマンド処理（1/2）

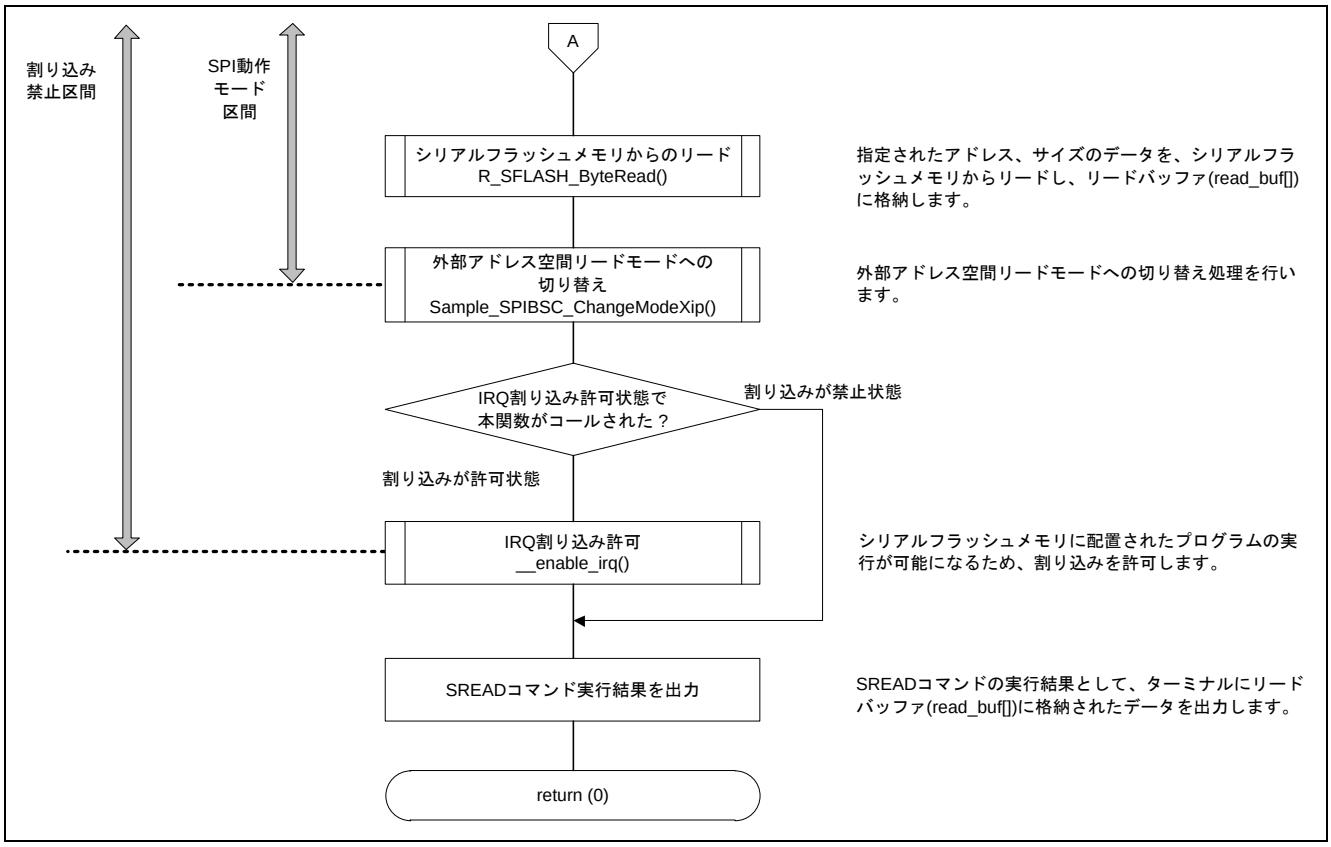


図6.25　SREAD コマンド処理（2/2）

6.14.8 SPI 動作モードで使用する MMU 変換テーブル変更処理

図 6.26にSPI 動作モードで使用する MMU 変換テーブル変更処理を示します。本処理では、SPI マルチ I/O バス空間（H'1800_0000～H'1BFF_FFFF）の MMU 変換テーブルについて、AP[2:0]ビット、XN ビットの値を書き換え、SPI 動作モード中に SPI マルチ I/O バス空間へのアクセスができないように設定します。

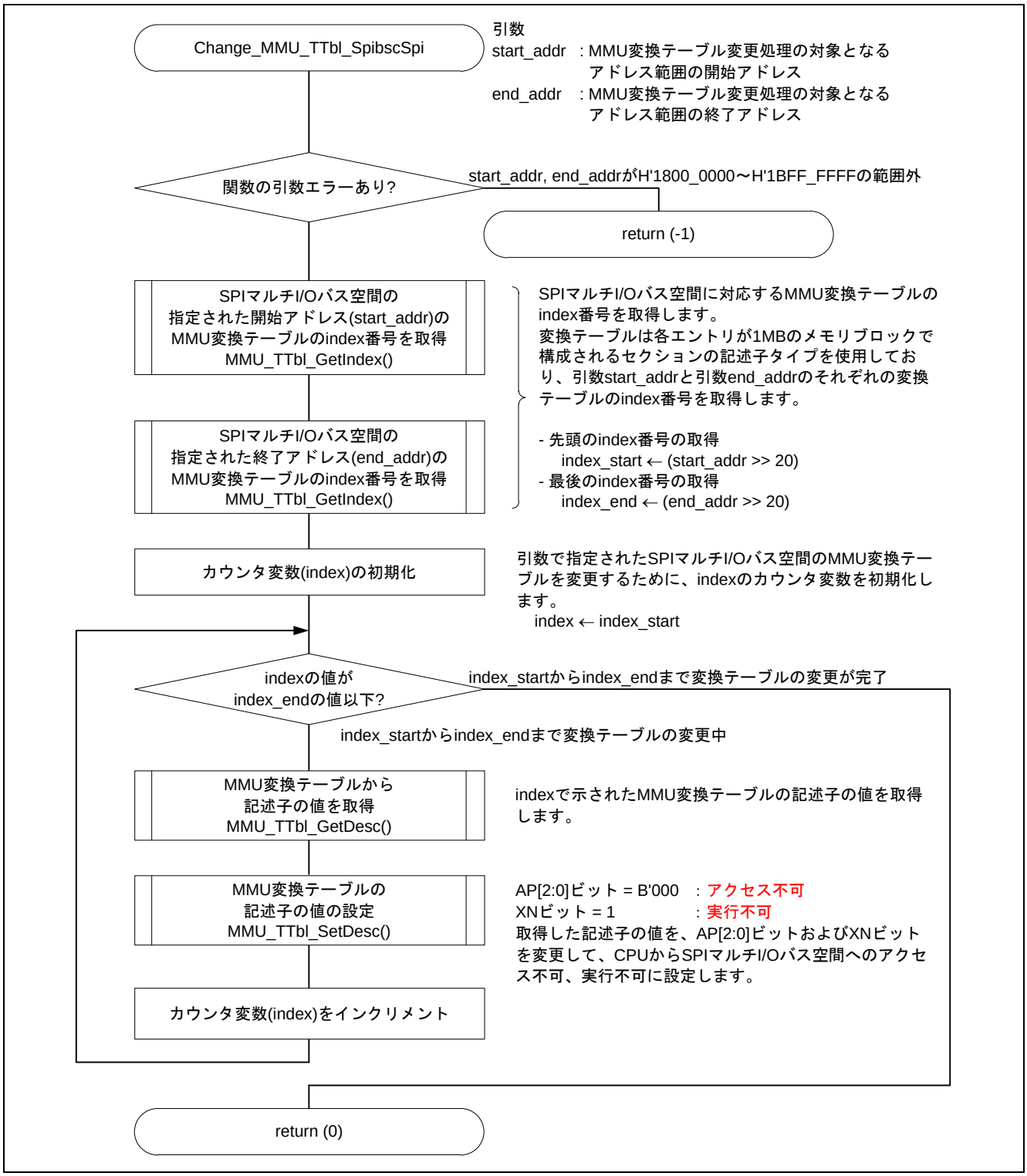


図6.26　SPI 動作モードで使用する MMU 変換テーブル変更処理

6.14.9　外部アドレス空間リードモードで使用する MMU 変換テーブル変更処理

図 6.27に外部アドレス空間リードモードで使用する MMU 変換テーブル変更処理を示します。本処理では、SPI マルチ I/O バス空間（H'1800_0000～H'1BFF_FFFF）の MMU 変換テーブルについて、AP[2:0]ビット、XN ビットの値を書き換え、外部アドレス空間リードモード設定時に SPI マルチ I/O バス空間へのアクセスができるように設定します。

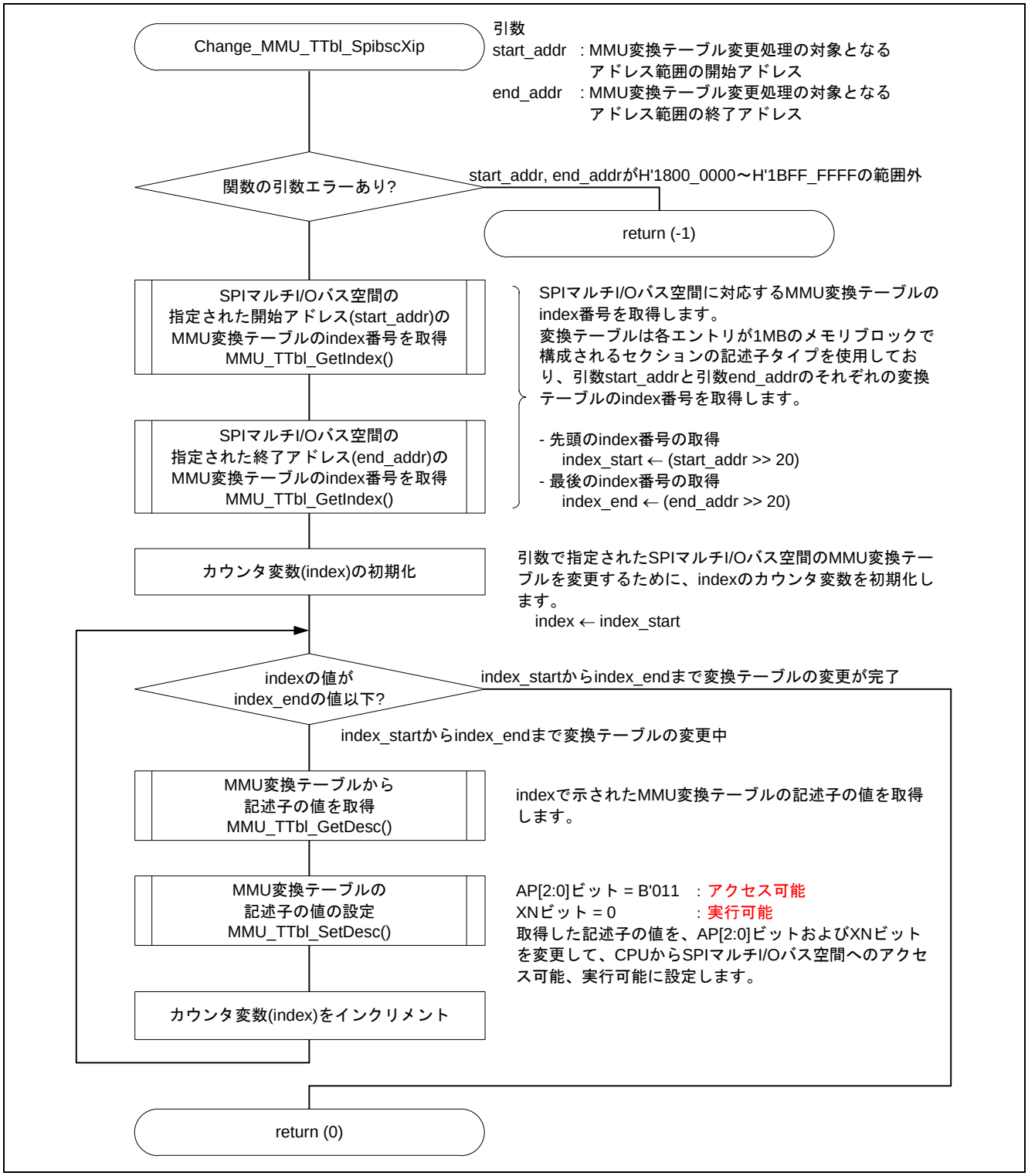


図6.27　外部アドレス空間リードモードで使用する MMU 変換テーブル変更処理

6.14.10　シリアルフラッシュメモリライト許可

シリアルフラッシュメモリのレジスタ（Status Register および Configuration Register）にライトするためには、事前にシリアルフラッシュメモリのライトを許可にしておく必要があります。

ご使用のシリアルフラッシュメモリの仕様に合わせて、シリアルフラッシュメモリのライト許可が行えるように、Userdef_SFLASH_Write_Enable 関数を実装してください。

サンプルコードでは、シリアルフラッシュメモリへのコマンド発行関数（R_SFLASH_Spibsc_Transfer）を使用して、Write Enable コマンド（H'06）を発行することで、ライト許可（Status Register の WEL ビットを"1"に設定）に変更する処理を行います。

図 6.28に、サンプルコードのUserdef_SFLASH_Write_Enable 関数のフローを示します。

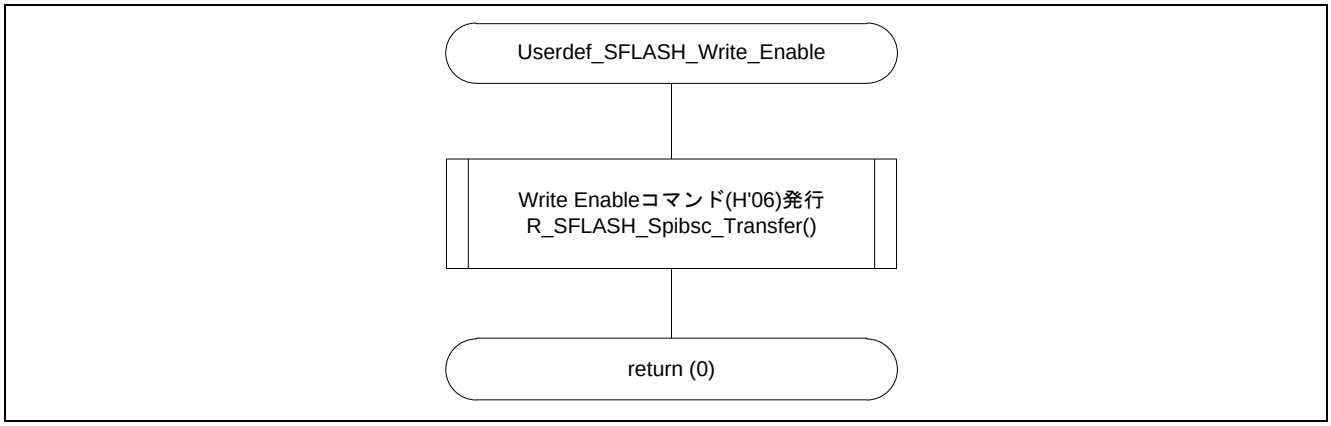


図6.28　Userdef_SFLASH_Write_Enable 関数のフロー

6.14.11　シリアルフラッシュメモリライト完了待ち

シリアルフラッシュメモリのレジスタ（Status Register および Configuration Register）にライトした場合、シリアルフラッシュメモリはビジー状態に遷移します。ビジー状態からレジスタにライトしたデータが反映されるまでウェイトする必要があります。

ご使用のシリアルフラッシュメモリの仕様に合わせて、シリアルフラッシュメモリのライト完了までウェイトするように、Userdef_SFLASH_Busy_Wait 関数を実装してください。

サンプルコードでは、Status Register の WIP ビットをリードしライト完了をウェイトする処理を行います。

図 6.29に、サンプルコードのUserdef_SFLASH_Busy_Wait 関数のフローを示します。

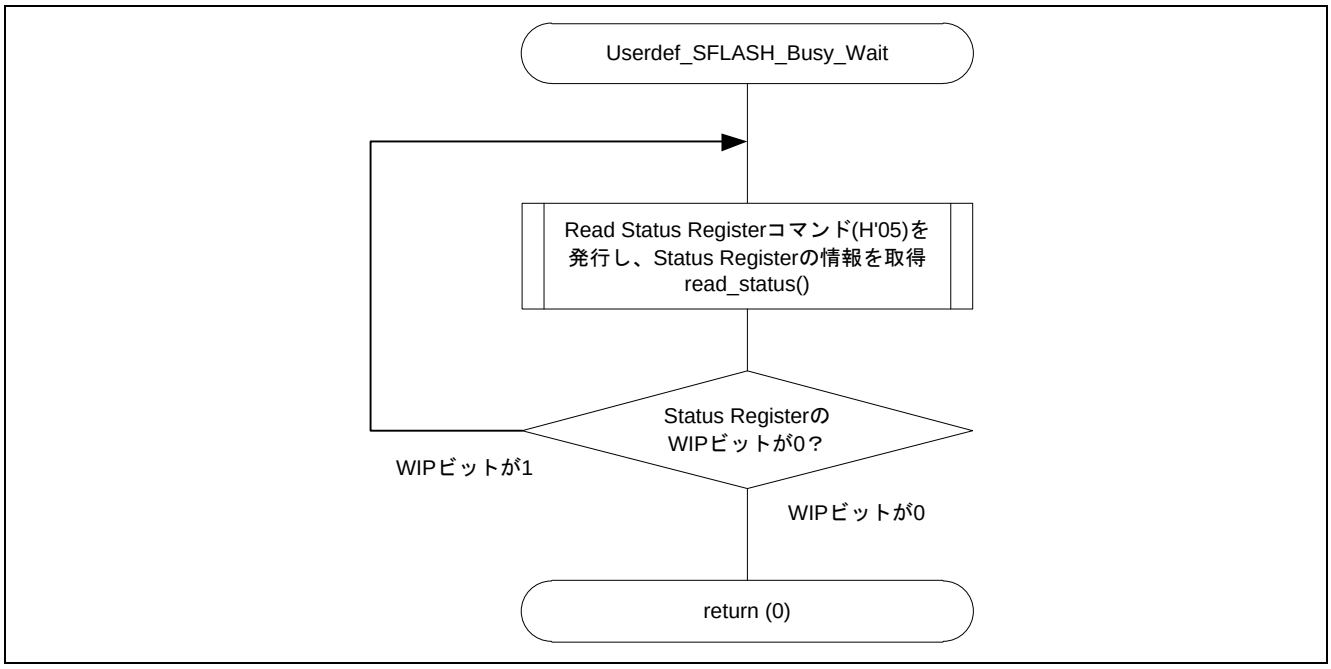


図6.29　Userdef_SFLASH_Busy_Wait 関数のフロー

6.14.12　シリアルフラッシュメモリプロテクト解除

シリアルフラッシュメモリにイレーズやライトを行う場合、シリアルフラッシュメモリのプロテクトを解除する必要があります。

ご使用のシリアルフラッシュメモリの仕様に合わせて、シリアルフラッシュメモリのプロテクトを解除するように、Userdef_SFLASH_Ctrl_Protect 関数を実装してください。

サンプルコードでは、write_status 関数からコールされるシリアルフラッシュメモリへのコマンド発行関数（R_SFLASH_Spibsc_Transfer）を使用して、Status Register の Block Protect ビット（BP3, BP2, BP1, BP0）に 0 を設定してすべてのブロックのプロテクトを解除しています。

図 6.30にUserdef_SFLASH_Ctrl_Protect 関数のフローを示します。

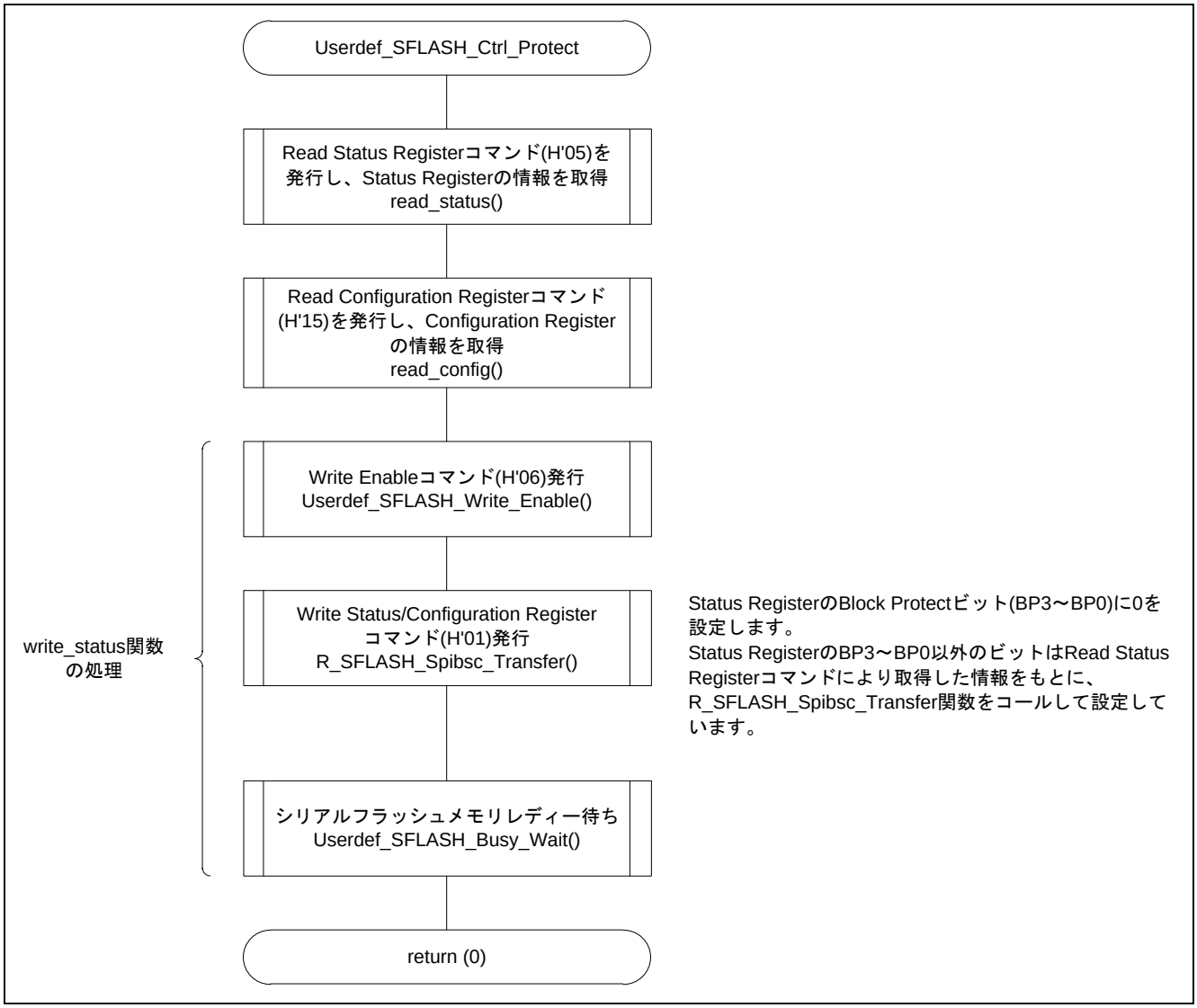


図6.30　Userdef_SFLASH_Ctrl_Protect 関数のフロー

7. サンプルコードの使用方法

7.1 サンプルコードの起動方法について

サンプルコードは、JASMINE ボードとシリアルインタフェースで接続したホスト PC 上のターミナルからコマンド入力を行うことで、動作します。

JASMINE ボードに電源を投入後、図 7.1の①に示すメッセージを出力します。SPIBSC のサンプルコードが起動した場合は、"SAMPLE>のプロンプトの後に、"SPIBSC"＋"Enter"キーを入力すると、図 7.1の②に示すメッセージを出力します。"SPIBSC >"のプロンプトの後に、以下のように入力することで SPIBSC サンプルコードを起動します。

1. "XREAD"＋"Enter"を入力すると、SPI マルチ I/O バス空間のダンプ処理を行うサンプルコードが起動
2. "ERASE"＋"Enter"を入力すると、シリアルフラッシュメモリのイレーズ処理を行うサンプルコードが起動
3. "WRITE"＋"Enter"を入力すると、シリアルフラッシュメモリのライト処理を行うサンプルコードが起動
4. "SREAD"＋"Enter"を入力すると、シリアルフラッシュメモリのリード処理を行うサンプルコードが起動

表示メッセージ

```
RZ/A1LU AVB Board S-Flash Boot Sample Program. Ver.X.XX
Copyright (C) 2017 Renesas Electronics Corporation. All rights reserved.   ①

select sample program.

SAMPLE>
```

```
RZ/A1LU SPIBSC Sample Program. Ver.Y.YY
Copyright (C) 2017 Renesas Electronics Corporation. All rights reserved.   ②

select sample program.

SPIBSC >
```

```
SPIBSC > help
                                                                           ③
  XREAD: Read data from the serial flash memory
         (by the external address space read mode)
  ERASE: Erase sector in the serial flash memory
  WRITE: Write data to the serial flash memory
  SREAD: Read data from the serial flash memory
         (by the SPI operating mode)
  EXIT:  Exit from SPIBSC Sample Program

SPIBSC >
```

図7.1　SPIBSC サンプルコード起動時のターミナル表示例

"HELP"＋"Enter"キーを入力することで、図 7.1の③に示すようなサンプルコードの情報を表示します。"EXIT"＋"Enter"キーを入力することで、SPIBSC サンプルコードの動作を終了します。

図 7.1の Ver.X.XX はサンプルコードのメイン処理のバージョンを、Ver.Y.YY は SPIBSC サンプルコードのバージョンを示します。

8. 応用例

8.1 シリアルフラッシュメモリを変更する場合のサンプルコード変更方法

シリアルフラッシュメモリを変更する場合、使用するシリアルフラッシュメモリの仕様に合わせてサンプルコードの処理を変更する必要があります。ここでは、シリアルフラッシュメモリのコマンド仕様に合わせて、サンプルコードを変更する場合の内容について説明します。

8.1.1 セクタイレーズ関数 R_SFLASH_EraseSector の変更点

使用するシリアルフラッシュメモリのセクタイレーズコマンドの仕様に合わせて、表 8.1に記載の項目について、R_SFLASH_EraseSector 関数の内容を変更してください。

表8.1 R_SFLASH_EraseSector 関数の変更が必要な項目

項目	内容
コマンド	g_spibsc_spimd_reg.cmd = イレーズコマンド
送信するアドレス長	• 24 ビット長の場合 g_spibsc_spimd_reg.ade = SPIBSC_OUTPUT_24 • 32 ビット長の場合 g_spibsc_spimd_reg.ade = SPIBSC_OUTPUT_32
アドレス送信時のバス幅	• 1 ビット場合 g_spibsc_spimd_reg.adb = SPIBSC_1BIT • 4 ビット場合 g_spibsc_spimd_reg.adb = SPIBSC_4BIT
アドレス送信時のデータ転送方式	• SDR 転送の場合 g_spibsc_spimd_reg.addre = SPIBSC_SDR_TRANS • DDR 転送の場合 g_spibsc_spimd_reg.addre = SPIBSC_DDR_TRANS
セクタサイズ（ブロックサイズ）	マクロ定義"SF_SECTOR_SIZE"を使用するシリアルフラッシュメモリのセクタサイズに合わせて変更 （"sflash.h"で定義）
総セクタ数	マクロ定義"SF_NUM_OF_SECTOR"を使用するシリアルフラッシュメモリの総セクタ数に合わせて変更 （"sflash.h"で定義）

8.1.2 ライト関数 R_SFLASH_ByteProgram の変更点

使用するシリアルフラッシュメモリのページプログラム（ライト）コマンドの仕様に合わせて、表 8.2に記載の項目について、R_SFLASH_ByteProgram 関数の内容を変更してください。

表8.2 R_SFLASH_ByteProgram 関数の変更が必要な項目

項目	内容
コマンド	g_spibsc_spimd_reg.cmd = ページプログラムコマンド
送信するアドレス長	• 24 ビット長の場合 g_spibsc_spimd_reg.ade = SPIBSC_OUTPUT_24 • 32 ビット長の場合 g_spibsc_spimd_reg.ade = SPIBSC_OUTPUT_32
アドレス送信時のバス幅	• 1 ビット場合 g_spibsc_spimd_reg.adb = SPIBSC_1BIT • 4 ビット場合 g_spibsc_spimd_reg.adb = SPIBSC_4BIT
アドレス送信時のデータ転送方式	• SDR 転送の場合 g_spibsc_spimd_reg.addre = SPIBSC_SDR_TRANS • DDR 転送の場合 g_spibsc_spimd_reg.addre = SPIBSC_DDR_TRANS
データ送信時のバス幅	• 1 ビット場合 g_spibsc_spimd_reg.spidb = SPIBSC_1BIT • 4 ビット場合 g_spibsc_spimd_reg.spidb = SPIBSC_4BIT
データ送信時のデータ転送方式	• SDR 転送の場合 g_spibsc_spimd_reg.spidre= SPIBSC_SDR_TRANS • DDR 転送の場合 g_spibsc_spimd_reg.spidre= SPIBSC_DDR_TRANS
ページサイズ	マクロ定義"SF_PAGE_SIZE"を使用するシリアルフラッシュメモリのページサイズに合わせて変更 （"sflash.h"で定義）

8.1.3 リード関数 R_SFLASH_ByteRead の変更点

使用するシリアルフラッシュメモリのリードコマンドの仕様に合わせて、表 8.3および表 8.4に記載の項目について、R_SFLASH_ByteRead 関数の内容を変更してください。

表8.3 R_SFLASH_ByteRead 関数の変更が必要な項目（1/2）

項目	内容
コマンド	g_spibsc_spimd_reg.cmd = リードコマンド
送信するアドレス長	• 24 ビット長の場合 g_spibsc_spimd_reg.ade = SPIBSC_OUTPUT_24 • 32 ビット長の場合 g_spibsc_spimd_reg.ade = SPIBSC_OUTPUT_32
アドレス送信時のバス幅	• 1 ビット場合 g_spibsc_spimd_reg.adb = SPIBSC_1BIT • 4 ビット場合 g_spibsc_spimd_reg.adb = SPIBSC_4BIT
アドレス送信時のデータ転送方式	• SDR 転送の場合 g_spibsc_spimd_reg.addre = SPIBSC_SDR_TRANS • DDR 転送の場合 g_spibsc_spimd_reg.addre = SPIBSC_DDR_TRANS
オプションデータの出力有無	• 出力しない場合 g_spibsc_spimd_reg.opde = SPIBSC_OUTPUT_DISABLE • OPD3 のデータを出力する場合 g_spibsc_spimd_reg.opde = SPIBSC_OUTPUT_OPD_3 • OPD3、OPD2 のデータを出力する場合 g_spibsc_spimd_reg.opde = SPIBSC_OUTPUT_OPD_32 • OPD3、OPD2、OPD1 のデータを出力する場合 g_spibsc_spimd_reg.opde = SPIBSC_OUTPUT_OPD_321 • OPD3、OPD2、OPD1、OPD0 のデータを出力する場合 g_spibsc_spimd_reg.opde = SPIBSC_OUTPUT_OPD_3210
オプションデータの出力時のバス幅	• SDR 転送の場合 g_spibsc_spimd_reg.opdb = SPIBSC_1BIT • DDR 転送の場合 g_spibsc_spimd_reg.opdb = SPIBSC_4BIT
オプションデータ出力時の データ転送方式	• SDR 転送の場合 g_spibsc_spimd_reg.opdre = SPIBSC_SDR_TRANS • DDR 転送の場合 g_spibsc_spimd_reg.opdre = SPIBSC_DDR_TRANS

表8.4　R_SFLASH_ByteRead 関数の変更が必要な項目（2/2）

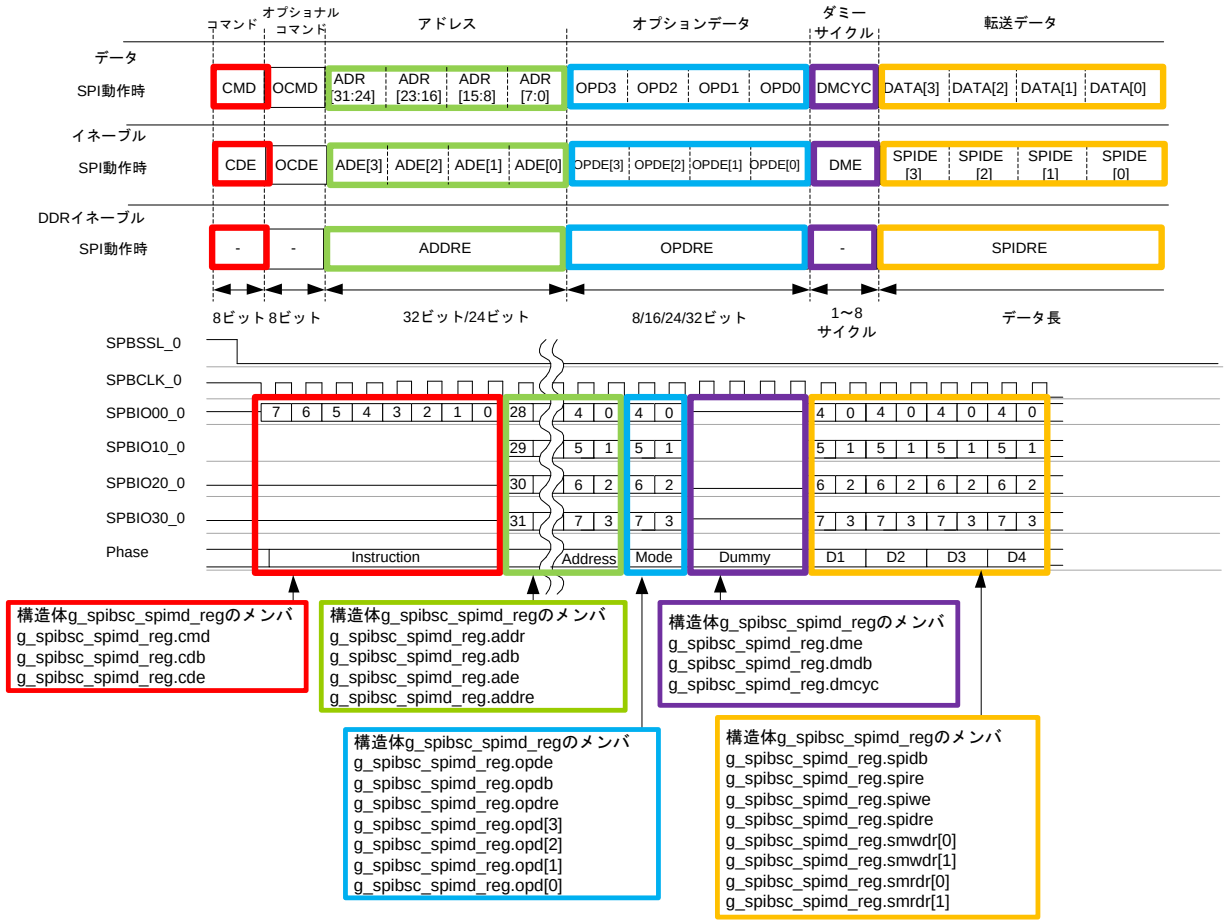
項目	内容
オプションデータの指定	• OPD3 の出力値 g_spibsc_spimd_reg.opd[0] = 出力値 • OPD2 の出力値 g_spibsc_spimd_reg.opd[1] = 出力値 • OPD1 の出力値 g_spibsc_spimd_reg.opd[2] = 出力値 • OPD0 の出力値 g_spibsc_spimd_reg.opd[3] = 出力値
ダミーサイクルの出力有無	• 出力しない場合 g_spibsc_spimd_reg.dme = SPIBSC_DUMMY_CYC_DISABLE • 出力する場合 g_spibsc_spimd_reg.dme = SPIBSC_DUMMY_CYC_ENABLE
ダミーサイクル出力時のバス幅	• 1 ビット場合 g_spibsc_spimd_reg.dmdb = SPIBSC_1BIT • 4 ビット場合 g_spibsc_spimd_reg.dmdb = SPIBSC_4BIT
ダミーサイクル出力時のサイクル数	g_spibsc_spimd_reg.dmcyc = SPIBSC_DUMMY_nCYC（注）
データ受信時のバス幅	• 1 ビット場合 g_spibsc_spimd_reg.spidb = SPIBSC_1BIT • 4 ビット場合 g_spibsc_spimd_reg.spidb = SPIBSC_4BIT
データ受信時の転送方式	• SDR 転送の場合 g_spibsc_spimd_reg.spidre = SPIBSC_SDR_TRANS • DDR 転送の場合 g_spibsc_spimd_reg.spidre = SPIBSC_DDR_TRANS

【注】 n は挿入するダミーサイクル数（1～8）を示します。

8.2 シリアルフラッシュメモリにコマンド発行時の出力信号

SPI 動作モードでは、コマンドの発行時に SPIBSC 関連レジスタの設定を行うことにより、リード、イレーズ、およびライトコマンドなどのシリアルフラッシュメモリの仕様に合わせた信号を出力することができます。

サンプルコードでは、SPIBSC のレジスタに設定する値を、表 8.1～表 8.4で示すセクタイレーズ、ページプログラム、およびリードの各関数で使用する構造体のメンバに設定する値を変更することで、シリアルフラッシュメモリに出力する信号を変更することが可能です。図 8.1にSPIBSC のレジスタ設定とコマンド発行時にシリアルフラッシュメモリに出力される波形の関係を示します。使用するシリアルフラッシュメモリのコマンドに合わせて、各関数の構造体のメンバに値を設定してください。



Macronix社製シリアルフラッシュメモリ(MX25L51245G)のプログラムコマンド(PP4B)を発行する場合、R_SFLASH_ByteProgram関数で構造体g_spibsc_spimd_regの各メンバを以下のように設定し、R_SFLASH_Spibsc_Transfer関数をコールすることでコマンド発行を実現しています。

コマンド	g_spibsc_spimd_reg.cmd	(コマンド)	: H'12 (PP4B)
	g_spibsc_spimd_reg.cdb	(コマンド送信時のビット幅)	: SPIBSC_1BIT
	g_spibsc_spimd_reg.cde	(コマンドの出力有無)	: SPIBSC_OUTPUT_ENABLE
オプショナルコマンド	g_spibsc_spimd_reg.ocmd	(オプションコマンド)	: H'00
	g_spibsc_spimd_reg.ocdb	(オプションコマンドビット時のビット幅)	: SPIBSC_1BIT
	g_spibsc_spimd_reg.ocde	(オプションコマンドの出力有無)	: SPIBSC_OUTPUT_DISABLE
アドレス	g_spibsc_spimd_reg.addr	(アドレス)	: シリアルフラッシュメモリのアドレス
	g_spibsc_spimd_reg.adb	(アドレス送信時のビット幅)	: SPIBSC_4BIT
	g_spibsc_spimd_reg.ade	(送信するアドレス長)	: SPIBSC_OUTPUT_ADDR_32
	g_spibsc_spimd_reg.addre	(アドレス転送方式)	: SPIBSC_SDR_TRANS
オプションデータ	g_spibsc_spimd_reg.opde	(オプションデータの出力有無)	: SPIBSC_OUTPUT_DISABLE
	g_spibsc_spimd_reg.opdb	(オプションデータ送信時のビット幅)	: SPIBSC_1BIT
	g_spibsc_spimd_reg.opdre	(オプションデータ転送方式)	: SPIBSC_SDR_TRANS
	g_spibsc_spimd_reg.opd[3]	(オプションデータOPD0)	: H'00
	g_spibsc_spimd_reg.opd[2]	(オプションデータOPD1)	: H'00
	g_spibsc_spimd_reg.opd[1]	(オプションデータOPD2)	: H'00
	g_spibsc_spimd_reg.opd[0]	(オプションデータOPD3)	: H'00
ダミーサイクル	g_spibsc_spimd_reg.dme	(ダミーサイクルの出力有無)	: SPIBSC_DUMMY_CYC_DISABLE
	g_spibsc_spimd_reg.dmdb	(ダミーサイクル送信時のビット幅)	: SPIBSC_1BIT
	g_spibsc_spimd_reg.dmcyc	(ダミーサイクル数)	: SPIBSC_DUMMY_1CYC
データ	g_spibsc_spimd_reg.spidb	(転送データのビット幅)	: SPIBSC_4BIT
	g_spibsc_spimd_reg.spire	(データリードイネーブル)	: SPIBSC_SPIDATA_DISABLE（リード無効)
	g_spibsc_spimd_reg.spiwe	(データライトイネーブル)	: SPIBSC_SPIDATA_ENABLE（ライト有効)
	g_spibsc_spimd_reg.spidre	(転送データの転送方式)	: SPIBSC_SDR_TRANS
	g_spibsc_spimd_reg.smwdr[0]	(SMWDR0へのライトデータ)	: ライトデータ
	g_spibsc_spimd_reg.smwdr[1]	(SMWDR1へのライトデータ)	: シリアルフラッシュメモリ1個接続時は未使用
	g_spibsc_spimd_reg.smrdr[0]	(SMRDR0からのリードデータ)	: リードデータ
	g_spibsc_spimd_reg.smrdr[1]	(SMRDR1からのリードデータ)	: シリアルフラッシュメモリ1個接続時は未使用
SSL設定	g_spibsc_spimd_reg.sslkp	(SSL信号キープ設定)	: SPIBSC_SPISSL_NEGATE

図8.1　SPIBSC のレジスタ設定とコマンド発行時にシリアルフラッシュメモリに出力される波形の関係

9. 注意事項

9.1 サンプルコマンド実行中に発生した割り込みについて

サンプルコマンド実行中は IRQ 割り込みを禁止の状態に設定しているため、サンプルコマンド実行中に発生した割り込み処理は実行されません。サンプルコマンドの処理が完了し割り込みを許可の状態に設定した後、保留された割り込み処理を実行します。

サンプルコマンド実行中に割り込み処理を実行する場合、IRQ 割り込みを許可の状態に設定することで、サンプルコマンド実行中でも割り込み処理を実行することができます。ただし、サンプルコマンド実行中は、SPIBSC が SPI 動作モードとなっている可能性があります。SPI 動作モード中は、SPI マルチ I/O バス空間（シリアルフラッシュメモリ）に配置されている命令は実行できません。このため、サンプルコマンド実行中（SPI 動作モード中）に割り込み処理や例外処理を実行する場合は、IRQ 例外処理ベクタを含むすべての IRQ 割り込み処理を大容量内蔵 RAM に配置して実行してください。

なお、FIQ 割り込みを許可の状態に設定した場合は禁止にすることはできませんので、FIQ 割り込みが発生する可能性がある場合は、FIQ 例外処理ベクタを含むすべての FIQ 割り込み処理を大容量内蔵 RAM に配置してください。

10. サンプルコード

サンプルコードは、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。

11. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RZ/A1L グループ、RZ/A1LUグループ、RZ/A1LC グループ ユーザーズマニュアル ハードウェア編
（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。）

RZ/A1LU AVB ボード RTK772103FC00000BR（JASMINE）ユーザーズマニュアル
（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。）

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
（最新版を ARM ホームページから入手してください。）

ARM Generic Interrupt Controller Architecture Specification Architecture version 1.0
（最新版を ARM ホームページから入手してください。）

ARM CortexTM-A9（Revision: r3p0）Technical Reference Manual
（最新版を ARM ホームページから入手してください。）

ARM CoreLinkTM Level 2 Cache Controller L2C-310（Revision: r3p2）Technical Reference Manual
（最新版を ARM 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／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境

ARM ソフトウェア開発ツール（ARM Compiler toolchain、ARM DS-5 等）に関しては、ARM ホームページから入手してください。
（最新版を ARM ホームページから入手してください。）

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/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
Rev.0.80	2016.12.27	ー	新規作成
Rev.1.00	2017.5.17	ー	アプリケーションノートの英語版の発行に合わせて、Rev.1.00に更新。

製品ご使用上の注意事項

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

1. 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（予約領域）のアクセス禁止

【注意】リザーブアドレス（予約領域）のアクセスを禁止します。

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

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、製品型名ごとにシステム評価試験を実施してください。

同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することはできません。たとえ、意図しない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

注1.　本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が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。

(Rev.3.0-1　2016.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口　http://www.renesas.com

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

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

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