AN204438

FM0+、FM3、および FM4 MCU のフラッシュセキュリティ設定方法

このアプリケーションノートでは、FM0+、FM3、および FM4 デバイスのフラッシュセキュリティを有効にする方法について説明します。

1 はじめに

フラッシュセキュリティを有効にして、外部ツールがフラッシュメモリに保存されているコードまたはデータを読み出さないようにします。このアプリケーションノートでは、FM0+、FM3、および FM4 デバイスのフラッシュセキュリティを有効にする方法について説明します。フラッシュセキュリティは、ワークフラッシュまたはデバイスに存在する場合はデュアルフラッシュを含む、フラッシュメモリ全体に適用されます。

このドキュメントでは、フラッシュセキュリティを有効にする 2 つの方法について説明し、任意の FM デバイスでフラッシュセキュリティを有効にするために必要なフラッシュセクターアドレスとデータパターンを提供します。

2 Flash セキュリティの仕組み

フラッシュセキュリティは通常、FM デバイスではデフォルトで無効になっています。特定のデータパターン (0x0001 など) が特別なフラッシュセクターアドレスに書き込まれると、フラッシュセキュリティが有効になります。7 ページのフラッシュセキュリティセクターの概要には、各 FM デバイスのデータパターンとフラッシュセクターアドレスがリストされています。

フラッシュセキュリティが無効になっている場合、ユーザーはフラッシュプログラミングツールと JTAG デバッガを使用してフラッシュメモリに完全にアクセスできます。

フラッシュセキュリティが有効になっている場合、外部のツールからフラッシュコンテンツにアクセスできません。外部ツールによる読み取り操作では、意味のない値が読み取られます。ただし、ユーザーコードは常にフラッシュメモリアレイにアクセスできます。監視モードまたはメモリ保護ユニット (MPU) によって保護されていません。

フラッシュセキュリティは、Boot-ROM またはソフトウェアによって制御されません。これは、フラッシュメモリ自体のハードウェアマクロの一部です。フラッシュセキュリティのロックを解除する唯一の方法は、イベントの順序を処理するフラッシュプログラミングツールを使用してフラッシュチップ消去シーケンスを実行することです。シーケンスは次のとおりです。デバイスにワークフラッシュがある場合、メインフラッシュを消去する前に、まずワークフラッシュを消去する必要があります。デュアルフラッシュのデバイスでは、マクロ#1 をマクロ#0 の前に消去します。

フラッシュセキュリティを有効にする方法は 2 つあります。

- アセンブリファイルの使用とリンカーコマンドファイルの変更
- S レコードまたは HEX レコードファイルのパッチ

どちらの場合も、フラッシュプログラミングツール (シリアルまたは USB) を使用して、デバイスをプログラミングすることによりフラッシュセキュリティを有効にします。フラッシュセキュリティがすでに有効になっている場合は、JTAG インターフェースを使用できません。

次のセクションでは、各アプローチについて詳しく説明します。

3 アセンブリファイルを使用してフラッシュセキュリティを有効にする

3.1 手順

1. フラッシュセキュリティセクターのメモリセクションを含む小さなアセンブリファイルを作成してください。このメモリセクションでフラッシュセキュリティパターンを定義してください。正確な詳細は IDE によって異なります。
2. デバッグが終了して、プロジェクトの最終ビルドの直前に、このアセンブリファイルをプロジェクトに追加してください。
3. このアセンブリファイルをプロジェクトに含めるには、IDE のリンカーコマンドファイルを変更してください。
4. フラッシュプログラミングツールを使用して、フラッシュセキュリティを有効にしてフラッシュメモリをプログラムしてください。

3.1.1 IAR EWARM のアセンブリおよびリンカーファイル

以下のアセンブリコードは、フラッシュセキュリティを有効にする方法を示します。この例では、フラッシュセキュリティセクターアドレスは 0x0010 0000 で、パターンは 0x0001 です。

```
        MODULE FLASH_SECURITY
        SECTION .flashsec: CONST (2)

        DC16 0x0001

        END
```

ファイル名は無関係です。アセンブリモジュール名と同じ名前を付ける必要はありません。

対応するリンカーファイル (* .icf) を編集し、次の行を追加してください。

```
Define region FLASH_SECURITY_region = mem: [from 0x00100000 to
0x00100003];
Define symbol FLASH_SECURITY_ADDRESS = 0x00100000;
Place at address mem: FLASH_SECURITY_ADDRESS {read only
section .flashsec};
```

ファイル名が flash_security であると仮定すると、このセクションは、ビルド後にリンカーマップファイルで発生します。

```
"A2":                                   0x4
FLASHSEC const    0x00100000    0x2 flash_security.o [1]
                   - 0x00100002    0x2
```

3.1.2 KEIL µVISION のアセンブリファイル

以下のアセンブリコードは、フラッシュセキュリティを有効にする方法を示します。この例では、フラッシュセキュリティセクターアドレスは 0x0010 0000 で、パターンは 0x0001 です。

```
            AREA    |.ARM.__at_0x00100000|, DATA, READONLY
            DCB 0x01
            DCB 0x00

            END
```

ファイル名は無関係です。

ファイル名が *flash_security* であると仮定すると、このセクションは、ビルド後にリンカーマップファイルで発生します。

```
      Load Region LR$$.ARM.__at_0x00100000 (Base: 0x00100000, Size:
      0x00000004, Max: 0x00000004, ABSOLUTE)

          Execution Region ER$$.ARM.__at_0x00100000 (Base: 0x00100000, Size:
      0x00000004, Max: 0x00000004, ABSOLUTE, UNINIT)

          Base Addr    Size         Type   AttrIdx    E Section Name
      Object

      0x00100000   0x00000002   Data   RO
      83    .ARM.__at_0x00100000flash_security.o
```

3.1.3 AtollicTrueStudio のアセンブリおよびリンカーファイル

以下のアセンブリコードは、フラッシュセキュリティを有効にする方法を示します。この例では、フラッシュセキュリティセクターアドレスは 0x0010 0000 で、パターンは 0x0001 です。

```
      .section .flash security, "a"
      .global _flash security
      _flash security:    .long 0x00000001
```

ファイル名は無関係です。

対応するリンカーファイル (* .ld) を編集し、強調表示された行を MEMORY 定義に追加してください。

```
      /* specify the memory areas */
      MEMORY
      {
        FLASH (rx)      : ORIGIN = 0x00000000, LENGTH = 512K
        RAM (xrw)       : ORIGIN = 0x1FFF8000, LENGTH = 64K
        FLASHSEC (r)    : ORIGIN = 0x00100000, LENGTH = 4
        MEMORY_B1 (rx): ORIGIN = 0x60000000, LENGTH = 0K
      }
```

強調表示された行を、RAM セクション定義の直前の SECTION 定義に追加してください。

```
    . = ALIGN (4);
_edata=.;        /* define a global symbol at data end */
  } >RAM AT> FLASH

.flash security:
  {
    . = ALIGN(4);
    _flash security=.;
    KEEP(*(.flash security))
    . = ALIGN (4);
  } >FLASHSEC

  /* uninitialized data section */
  . = ALIGN (4);
```

ファイル名が *flashsec.s* であると仮定すると、このセクションはビルド後にリンカーマップファイルで発生します。

```
.flash security 0x00100000        0x4
                0x00100000                . = ALIGN (0x4)
                0x00100000                _flash security= .
 *(.flash security)
 .flash security
                0x00100000        0x4 source/flashsec.o
                0x00100004                . = ALIGN (0x4)
```

3.1.4 GNU コンパイラー環境用のアセンブリーおよびリンカーファイル

AtollicTrueStudio のアセンブリおよびリンカーファイルを参照してください。

4 S レコードまたは HEX ファイルにパッチを適用してフラッシュセキュリティを有効にする

4.1 S レコードファイル形式

S レコードファイル形式の行は、通常、文字「S」、行のタイプ、この行のデータペイロードのバイトカウント、アドレス、データペイロード、およびチェックサムバイトで構成されます。

S レコード行のタイプコードは次のとおりです。

コード	変更内容	アドレスバイト数	データフィールド
S0	ブロックヘッダー	2	有り
S1	データペイロード	2	有り
S2	データペイロード	3	有り
S3	データペイロード	4	有り
S5	レコード数	2	無し
S7	ブロックの終わり	4	無し
S8	ブロックの終わり	3	無し
S9	ブロックの終わり	2	無し

フラッシュセキュリティデータを最後の S レコード行（S7-9）の前に置くことをお勧めします。

アドレス 0x00100000 を値 0x0001 に設定してフラッシュセキュリティが有効になっているとすると、追加の S-Record 行は次のようになります。

```
S20810000001000000E6
```

この行の詳細な説明:

	3 バイトアドレスのデータペイロード	バイト数	3 バイトアドレス	データペイロード（ビッグエンディアン）	チェックサム
S	2	08	100000	01000000	E6

チェックサムを計算するには、Sn タイプの後、チェックサムバイト自体の前にすべてのバイトを追加します。値が 0x00 のバイトは計算でスキップされます。次に、合計の下位バイトのみを取り、合計を反転します。

```
0x08 + 0x10 + 0x01 = 0x19
0x19 = 0xE6
```

4.2 HEX ファイル形式

32 ビットアーキテクチャ用の Intel HEX ファイル形式の行は、通常、コロン、この行のデータペイロードのバイト数、アドレス、行のタイプ、データペイロード、およびチェックサムバイトで構成されます。

HEX ラインのタイプコードは次のとおりです。

コード	変更内容
0x00	データペイロード
0x01	ファイルの終わり
0x02	拡張セグメントアドレス

コード	変更内容
0x03	開始セグメントアドレス
0x04	拡張線形アドレス
0x05	線形アドレスの開始

フラッシュセキュリティデータを最後の HEX ラインの前に置くことをお勧めします。通常は次のとおりです。

```
: 00000001FF
```

アドレス 0x00100000 を値 0x0001 に設定してフラッシュセキュリティが有効になっているとすると、追加の HEX 行は次のようになります。

```
: 020000040010EA
: 04000000010000000FB
```

これらの行の詳細な説明：

	データ数	ダミーアドレス	拡張線形アドレス	セキュリティアドレスの上位 16 ビット	チェックサム
:	02	0000	04	0010	EA

	データ数	セキュリティアドレスの下位 16 ビット	データライン	セキュリティコードデータ（ビッグエンディアン）	チェックサム
:	04	0000	00	01000000	FB

チェックサムを計算するには、「：」の後、チェックサムバイト自体の前にすべてのバイトを追加します。値が 0x00 のバイトは計算でスキップされます。合計の下位バイトのみを取得します。合計を反転して 1 を加えます。

最初の行の例:

```
0x02 + 0x04 + 0x10 = 0x16
0x16 = 0xE9
0xE9 + 0x01 = 0xEA
```

5 フラッシュセキュリティセクターの概要

次の表に、フラッシュセキュリティセクターアドレスとそのアクティブ化パターンの概要を示します。デバイスタイプと部品番号の相互参照については、周辺機器のマニュアルを参照してください。

5.1 FM0+デバイス

シリーズ	フラッシュセキュリティアドレス	アクティベーションパターン
S6E1A	0x0010_0000	0x0001
S6E1B	0x0010_0000	0x0001
S6E1C	0x0010_0000	0x0001

5.2 FM3 デバイス

シリーズ	フラッシュセキュリティアドレス	アクティベーションパターン
CY9Ax3x	0x0010_0000	0x0001
CY9Ax4x	0x0010_0000	0x0001
CY9Ax5x	0x0010_0000	0x0001
CY9Ax10x	0x0010_0000	0x0001
CY9AxAx	0x0010_0000	0x0001
CY9Ax10A	0x0010_0000	0x0001
CY9Ax10K	0x0010_0000	0x0001
CY9AX20L	0x0010_0000	0x0001
CY9Bx10R	0x0010_0000	0x0001
CY9Bx10T	0x0010_0000	0x0001
CY9Bx20J	0x0010_0000	0x0001
CY9Bx20M	0x0010_0000	0x0001
CY9Bx20T	0x0040_0000	0x0001

5.3 FM4 デバイス

シリーズ	フラッシュセキュリティアドレス	アクティベーションパターン
CY9BFx6xK/L	0x0040_0000	0x0001
CY9BFx6x M/N/R	0x0040_0000	0x0001
S6E2C	0x0040_0000	0x0001
S6E2D	0x0040_0000	0x0001
S6E2G	0x0040_0000	0x0001
S6E2H	0x0040_0000	0x0001

改訂履歴

文書名: AN204438 – FM0+、FM3、および FM4 MCU のフラッシュセキュリティ設定方法

文書番号:

版数	変更内容
**	本版は英語版 002-04438 Rev. *Dについて、CYPRESS DEVELOPER COMMUNITYの参画者によって日本語に翻訳されたドキュメントです。

セールス, ソリューションおよび法律情報

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューションセンター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、サイプレスのロケーションページをご覧ください。

製品

Arm® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチセンシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス	cypress.com/wireless

PSoC®ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

サイプレス開発者コミュニティ

コミュニティ | サンプルコード | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

本書で言及するその他すべての商標または登録商標は、それぞれの所有者に帰属します。

Cypress Semiconductor
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709