XAPP1278 (v1.0) 2016 年 3 月 23 日

# デバイス プログラマを使用した eFUSE のプログラム

著者 : Bryan Penner、Habib El-Khoury、Randal Kuramoto

## 概要

このアプリケーション ノートは、デバイス プログラマを使用して Zynq®-7000 All Programmable (AP) SoC デバイスの eFUSE をプログラムする際に必要なファイルの事前準備について説明しています。デバイス プログラマは、次の場合に効果的です。

- 量産製品のオフボード デバイス プログラミング。
- ボード製造サイトでアセンブリを行う前に eFUSE に機密データをプログラムして保護する。

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。デザイン ファイルの詳細は、「リファレンス デザイン」を参照してください。

Zynq-7000 AP SoC eFUSE をプログラムするためのインシステム ソリューションの詳細は、『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 1] を参照してください。

このアプリケーション ノートで説明するソリューションのサポートは、次のサイトを参照してください。

- デバイス プログラミング サービス プロバイダー :
  - Avnet 社 (http://www.avnet.com)
- デバイス プログラマ ベンダー :
  - BPM Microsystems 社 (http://www.bpmmicro.com)

**注記** : ご使用の Zynq-7000 AP SoC デバイスおよびパッケージのサポート状況に関しては、プログラミング サービス プロバイダーまたはデバイス プログラマ ベンダーへお問い合わせください。

## プログラム可能な eFUSE の概要

Zynq-7000 AP SoC デバイスの eFUSE は、ワンタイム プログラマブル (OTP) な不揮発性メモリであり、一度書き込みを実行すると消去できないように設定されます。これらの設定によって、ブート イメージの認証および暗号化などのデバイス セキュリティ機能やユーザー指定の 32 ビット値が有効になります。このように eFUSE で対応できる機能の詳細は、「参考資料」の資料リストから参照してください。

ユーザがプログラムできる eFUSE は、各デバイスともザイリンクスから出荷される時点で 0 に設定されています。これを 1 にプログラムすることで、設定を有効にしたり、ユーザー定義の値を設定することが可能になります。

Zynq-7000 AP SoC デバイスには、次に示す 2 種類の eFUSE が含まれています。

- プロセッサ システム (PS) eFUSE :
  - Rivest、Shamir、Adleman (RSA) 認証、および boot ROM CRC チェック用の設定が含まれます。
- プログラマブル ロジック (PL) eFUSE :
  - AES (Advance Encryption Standard) で暗号化されたブート イメージ、ユーザー定義の 32 ビット値、およびその他のセキュリティの設定が含まれます。

# eFUSE のプログラムにデバイス プログラマを使用

デバイス プログラマの用途は次のとおりです。

- デバイスの eFUSE ビットをプログラムする。
- トレーサビリティ用にプログラム済みデバイスの DNA 値を記録する。
- デバイス が予想された設定であらかじめプログラムされていることをスタンドアロンで検証する。

Zynq-7000 AP SoC デバイスの場合、デバイス プログラマは、『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 1] に記載されているザイリンクスの eFUSE プログラミング ソリューションを使用します。このソリューション向けに、ザイリンクスはデバイスの eFUSE をプログラムする PS ARM® プロセッサ用のコード ライブラリを提供しています。デバイス プログラマは、あらかじめ構築されているこのコードを各デバイスへロードし、eFUSE をプログラムするためのユーザーの設定を個別にデバイスに送信します。

スタンドアロン検証では、正しくプログラムされたデバイスの最終的なサインオフ検証を実行します。この検証動作は、プログラム済みの在庫製品または RMA で返品された製品を再検証する場合にも使用できます。

一部の保護された Zynq-7000 AP SoC eFUSE 設定は読み出し不可となるため、スタンドアロン検証では従来のリードバック検証方法を使用できません。その代わり、スタンドアロン検証用にセキュア ブート イメージがデバイス プログラマに与えられます。デバイス プログラマは、このセキュア ブート イメージを Zynq-7000 AP SoC デバイスへロードして、eFUSE のセキュリティ設定を間接的に検証します。デバイスに、このセキュア ブート イメージを認証および復号化できる正しい設定が含まれている場合は、スタンドアロン検証が無事完了します。デバイスに既知のセキュアな eFUSE 設定が含まれていない場合には、ブート セキュリティ エラーが生じてスタンドアロン検証がエラーになります。

通常、デバイス プログラマは、プログラムされたデバイスのログ (デバイス DNA 値、プログラム結果、およびスタンドアロン検証結果を含む) を作成します。

# プログラマおよびファイルの概要

図 1 に、Zynq-7000 AP SoC のデバイス プログラマ ソケット、ファイル、およびフローの一般的なブロック図を示します

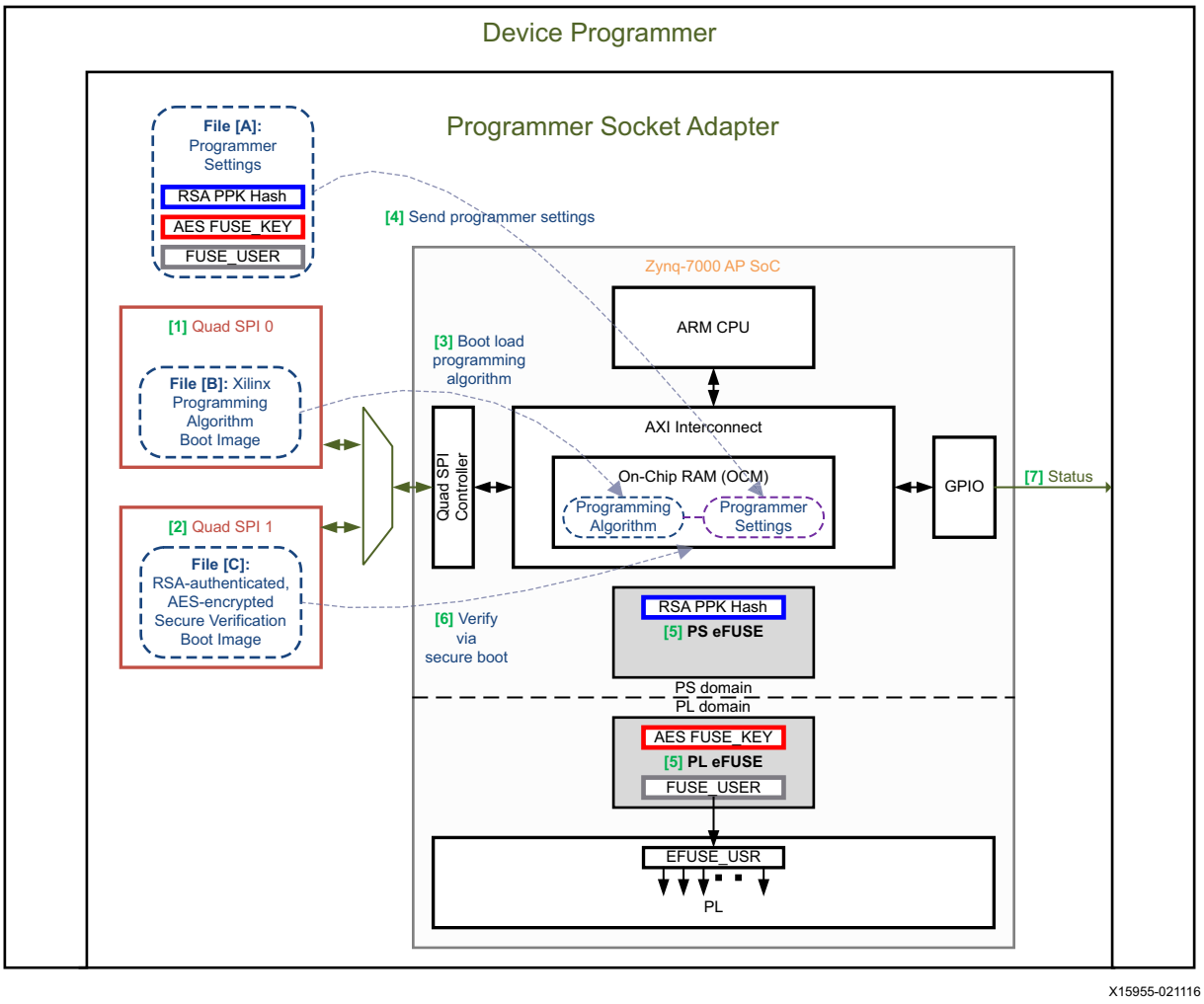



**図 1 : デバイス プログラマ ソケット アダプターとファイルのブロック図**

プログラム動作では、デバイス プログラマは次のシーケンスを実行します。

1. プログラムされていない場合、デバイス プログラマは、Quad Serial Peripheral Interface (SPI) 0 フラッシュ メモリにザイリンクスのプログラミング アルゴリズム ブート イメージ ファイル [B] をプログラムします。
2. プログラムされていない場合、デバイス プログラマは、Quad SPI 1 フラッシュ メモリに検証ブート イメージ ファイル [C] をプログラムします。
3. デバイス プログラマは、Quad SPI 0 フラッシュ メモリからオンチップ メモリ (OCM) にザイリンクス プログラミング アルゴリズムをブートロードします。
4. デバイス プログラマは、プログラマ設定ファイル [A] を OCM へ送信します。
5. デバイス プログラマは、ARM CPU がプログラミング アルゴリズム コードを実行して eFUSE 設定をプログラムすることを許可します。

プログラム後の検証動作またはスタンドアロン検証動作では、デバイス プログラマは次のシーケンスを実行します。

1. デバイス プログラマは、Quad SPI 1 フラッシュ メモリから安全な検証イメージをブートします。
2. ブート イメージが問題なく認証および復号化された場合、検証ブート イメージ アプリケーションはデバイス プログラマに完了ステータスを送信します。

# デバイス プログラマの準備

デバイス プログラマを使用してプログラムを開始する前に、次の準備が必要です。

1. デバイス プログラマのサイト/サービスを確立する。
2. セキュリティ キーと eFUSE 設定を定義する。
3. プログラマの検証ブート イメージ ファイルを作成する。
4. プログラマ設定ファイルを作成する。
5. ザイリンクスのあらかじめ構築されているアルゴリズム イメージ ファイルを指定する。
6. 最初のサンプル プログラミング用にデバイス プログラミング サイトにデバイス サンプルとファイルを提供する。

## ステップ 1 : デバイス プログラマのサイト/サービスを確立する

デバイス プログラミング サービスの詳細は、ザイリンクス販売代理店へお問い合わせください。また、社内プログラミング用デバイス プログラマを取得する場合は、デバイス プログラマ ベンダーへお問い合わせください。

**重要** : デバイス プログラマのサイトまたはサービスはできるだけ早期に確立することが重要です。プログラミング サイトで事前構築されていないデバイス/パッケージのデバイス プログラマ ソケットを開発および取得するには、8 ～ 12 週間 (またはそれ以上) かかる場合があります。

## ステップ 2 : セキュリティ キーと eFUSE 設定を定義する

セキュリティ キーおよび eFUSE に関するすべてのパラメーター値を表 1 のシートに記入します。セキュリティ キー生成の詳細は、『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 1] を参照してください。eFUSE 設定の詳細は、『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 1]、『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2]、および『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] を参照してください。

## セキュリティ キーおよび eFUSE 設定シート

Zynq-7000 AP SoC eFUSE プログラミング プロジェクトのすべての設定をシート (表 1) に定義します。

**注記** : RSA 認証を使用する場合は、プライマリおよびセカンダリの公開キーと非公開キーを定義します。RSA ハッシュ値は、次の 「ステップ 3 : プログラマの検証ブート イメージ ファイルを作成する」 で、RSA キーから生成可能です。

表 1 : セキュリティ キーおよび eFUSE 設定シート

| セキュリティ キー | | | |
|---|---|---|---|
| セキュリティ名 | 説明 | 形式 | 設定 (ここに定義) |
| RSA プライマリ非公開キー (Psk.pempsk.pem ファイル) | RSA 認証用。RSA プライマリ非公開キー。このキー /ファイルの生成方法の詳細は、『Zynq-7000 All Programmable SoC のセキュア ブート』 (XAPP1175) [参照 1] を参照。 | .pem ファイル | |
| RSA プライマリ公開キー (Ppk.pubppk.pub ファイル) | RSA 認証用。RSA プライマリ公開キー。このキー /ファイルの生成方法の詳細は、『Zynq-7000 All Programmable SoC のセキュア ブート』 (XAPP1175) [参照 1] を参照。 | .pub ファイル | |
| RSA セカンダリ非公開キー (ssk.pem ファイル) | RSA 認証用。RSA セカンダリ非公開キー。このキー /ファイルの生成方法の詳細は、『Zynq-7000 All Programmable SoC のセキュア ブート』 (XAPP1175) [参照 1] を参照。 | .pem ファイル | |
| RSA セカンダリ公開キー (spk.pubファイル) | RSA 認証用。RSA セカンダリ公開キー。このキー /ファイルの生成方法の詳細は、『Zynq-7000 All Programmable SoC のセキュア ブート』 (XAPP1175) [参照 1] を参照。 | .pub ファイル | |
| AES キー | AES 暗号化用。AES キー。画像暗号化用のユーザー定義値であり、復号用に eFUSE に格納。 | 64 桁の 16 進数 | |
| AES StartCBC | AES 暗号化用。AES ブロック暗号モード用の初期化ベクター。ユーザー指定のランダム値。 | 32 桁の 16 進数 | |
| HMAC | AES 暗号化用。AES 復号化における認証用 HMAC キー。ユーザー指定のランダム値。 | 64 桁の 16 進数 | |

表 1 : セキュリティ キーおよび eFUSE 設定シート (続き)

| PS eFUSE パラメーター | | | |
|---|---|---|---|
| PS eFUSE パラメーター名 | 説明 | 設定オプション (デフォルト) | 設定 (ここに定義) |
| XSK_EFUSEPS_ENABLE_WRITE_PROTECT | TRUE の場合は、PS eFUSE 書き込み保護ビットをプログラムして、その後の PS eFUSE の変更を禁止する。この設定は、デバイスのパワーサイクルまたは POR (パワー オン リセット) 後に作用する。<br>FALSE の場合は、書き込み保護ビットを変更できない。<br>『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「eFUSE 書き込み保護」を参照。 | TRUE/FALSE (FALSE) | |
| XSK_EFUSEPS_ENABLE_ROM_128K_CRC | TRUE の場合は、FSBL をロードする前に bootROM が 128K CRC を計算およびチェックできるように PS eFUSE をプログラムする。<br>FALSE の場合は、ROM 128K CRC ビットを変更できない。<br>『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「OCM ROM 128KB CRC 有効」を参照。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPS_DISABLE_DFT_JTAG | TRUE の場合は、デバイスの JTAG 機能を無効にする PS eFUSE ビットをプログラムする。これで RMA 解析を制限できる。<br>FALSE の場合は、DFT (Design-For-Test) JTAG を無効にするビットを変更できない。<br>『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「DFT JTAG 無効」を参照。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPS_DISABLE_DFT_MODE | TRUE の場合は、DFT モードを無効にする PS eFUSE ビットをプログラムする。これで RMA 解析を制限できる。<br>FALSE の場合は、DFT モードを無効にするビットを変更できない。<br>『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「DFT モード無効」を参照。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPS_ENABLE_RSA_AUTH | TRUE の場合は、PS RSA 認証イネーブル ビットをプログラムして、NAND フラッシュ、NOR フラッシュ、Quad SPI フラッシュ、および SD カードからの認証ブートを有効にする。このビットをプログラムする前には、XSK_EFUSEPS_ENABLE_RSA_KEY_HASH を TRUE に設定し、RSA HASH に有効な RSA PPK ハッシュ値を設定して、RSA HASH に有効な RSA PPK ハッシュ値が含まれていることを確認する必要がある。<br>FALSE の場合は、RSA イネーブル ビットを変更できない。<br>Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「RSA 認証有効」を参照。 | TRUE、FALSE (FALSE) | |

表 1 : セキュリティ キーおよび eFUSE 設定シート (続き)

| | | | |
|---|---|---|---|
| XSK_EFUSEPS_RSA_KEY_HASH | TRUE の場合は、RSA HASH パラメーターの変更を可能にし、特定の RSA HASH 値をプログラムできる。<br>FALSE の場合は、RSA HASH 値を無視し、RSA HASH eFUSE ビットは変更できない。<br>**注記** : この XSK_EFUSEPS_RSA_KEY_HASH パラメーターに対応する eFUSE ビットはありません。このパラメーターは、RSA HASH フィールドの変更を可能にし、ザイリンクスのプログラマ設定ファイル ジェネレーター ツールでの RSA HASH のプログラミング命令を制御します。 | | |
| RSA HASH | RSA プライマリ公開キー (PPK) の SHA-256 ハッシュ値。<br>• この値の変更およびプログラミングを有効にするには、XSK_EFUSEPS_ENABLE_RSA_KEY_HASH パラメーターを TRUE に設定する。<br>• RSA プライマリ/セカンダリの公開/非公開キーを定義または取得する。詳細は、「ステップ 2 : セキュリティ キーと eFUSE 設定を定義する」を参照。<br>• 検証ブート イメージを作成する場合、ザイリンクス Bootgen ツールを使用して、公開キーのハッシュを生成する。<br>• Bootgen -efuseppkbits オプションで生成された出力ファイル `efuse_ppk_hash.txt` のハッシュ値をコピーして、このフィールドにペーストする。これは 16 進数値で、長さは 64 文字となる。有効な文字は 0-9、a-f、および A-F となり、その他の文字は無効として認識されてプログラムできない。RSA 値を与えた後、XSK_EFUSEPS_ENABLE_RSA_AUTH パラメーターを TRUE に設定して、ブート イメージの RSA 認証を有効にする。<br>『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「RSA PPK ハッシュ」、および『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 1] の「PPK のハッシュ値の生成」を参照。 | 64 桁の 16 進数 (すべて 0) | |

表 1 : セキュリティ キーおよび eFUSE 設定シート (続き)

| PL eFUSE パラメーター | | | |
|---|---|---|---|
| PL eFUSE パラメーター名 | 説明 | 設定オプション (デフォルト) | 設定 (ここに定義) |
| XSK_EFUSEPL_FORCE_PCYCLE_RECONFIG | TRUE の場合は、PL のパーシャル リコンフィギュレーションを無効にする PL eFUSE ビットをプログラムする。これで RMA 解析を制限できる。eFUSE でパーシャル リコンフィギュレーションを無効にするもう 1 つの方法として、PL ビットストリーム セキュリティ プロパティを Level2 に設定する方法がある。<br>FALSE の場合は、この eFUSE の制御ビットを変更できない。<br>『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] および『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] の「AES_Exclusive」を参照。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPL_BBRAM_KEY_DISABLE | TRUE の場合は、BBRAM キー機能を無効にする PL eFUSE ビットをプログラムする。<br>FALSE の場合は、この eFUSE の制御ビットを変更できない。<br>『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「BBRAM Key Disable」を参照。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPL_DISABLE_JTAG_CHAIN | TRUE の場合は、Zynq-7000 デバイスの JTAG 機能を無効にする PL eFUSE ビットをプログラムする。これで RMA 解析を制限できる。この設定は、デバイスのパワーサイクルまたは POR (パワー オン リセット) 後に作用する。<br>FALSE の場合は、この eFUSE 制御ビットを変更できない。<br>『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「JTAG チェーン ディスエーブル」を参照。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPL_FORCE_USE_AES_ONLY | TRUE の場合は、eFUSE AES キーを使用したセキュア ブートを強制する PL eFUSE ビットをプログラムする。RMA 解析の実行は不可。<br>FALSE の場合は、この eFUSE 制御ビットを変更できない。<br>『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「eFUSE セキュア ブート」および『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] の「CFG_AES_Only」を参照。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPL_DISABLE_KEY_WRITE | TRUE の場合は、FUSE_KEY および FUSE_USER eFUSE 値への今後の書き込み (変更) を無効にする PL eFUSE ビットをプログラムする。この設定は、デバイスのパワーサイクルまたは POR (パワー オン リセット) 後に作用する。<br>FALSE の場合は、この eFUSE 制御ビットを変更できない。<br>『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] の「W_EN_B_Key_User」を参照。 | TRUE、FALSE (FALSE) | |

表 1 : セキュリティ キーおよび eFUSE 設定シート (続き)

| | | | |
|---|---|---|---|
| XSK_EFUSEPL_DISABLE_AES_KEY_READ | TRUE の場合は、FUSE_KEY 値への読み出しアクセスを無効にする PL eFUSE ビットをプログラムする。XSK_EFUSEPL_AES_KEY 値をプログラムする場合は、この値を TRUE に設定して、XSK_EFUSEPL_AES_KEY 値が不正に読み出されないように保護する必要がある。<br>**注記** : これは、FUSE_KEY および FUSE_USER 値への今後の書き込み (変更) も無効にします。<br>FALSE の場合は、この eFUSE 制御ビットを変更できない。<br>『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] の「R_EN_B_Key」を参照。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPL_DISABLE_USER_KEY_READ | TRUE の場合は、FUSE_USER 値への JTAG 読み出しアクセスを無効にする PL eFUSE ビットをプログラムする。<br>**注記** : これは、FUSE_KEY および FUSE_USER 値への今後の書き込み (変更) も無効にします。<br>FALSE の場合は、この eFUSE 制御ビットを変更できない。<br>この設定とは無関係に、FUSE_USER は EFUSE_USR プリミティブを介して常に PL へアクセス可能。<br>『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] の「R_EN_B_User」を参照。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPL_DISABLE_FUSE_CNTRL_WRITE | TRUE の場合は、次に示す PL FUSE_CTRL eFUSE ビットへの今後の書き込み (変更) を無効にする PL eFUSE ビットをプログラムする。XSK_EFUSEPL_FORCE_PCYCLE_RECONFIG、XSK_EFUSEPL_DISABLE_KEY_WRITE、XSK_EFUSEPL_DISABLE_AES_KEY_READ、XSK_EFUSEPL_DISABLE_USER_KEY_READ、XSK_EFUSEPL_FORCE_USE_AES_ONLY、XSK_EFUSEPL_DISABLE_JTAG_CHAIN、および XSK_EFUSEPL_BBRAM_KEY_DISABLE の設定。この設定は、デバイスのパワーサイクルまたは POR (パワー オン リセット) 後に作用する。<br>FALSE の場合は、この eFUSE 制御ビットを変更できない。<br>『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] の「W_EN_B_Cntl」を参照。 | TRUE、FALSE (FALSE) | |

表 1 : セキュリティ キーおよび eFUSE 設定シート (続き)

| | | | |
|---|---|---|---|
| XSK_EFUSEPL_PROGRAM_AES_AND_USER_LOW | TRUE の場合は、AES KEY および USER LOW パラメーターの変更を可能にして、指定値をプログラムできる。<br>FALSE の場合は、AES KEY および USER LOW 値を無視し、AES KEY および USER LOW eFUSE ビットを変更できない。<br>**注記** : AES KEY と USER LOW の値は、別々にプログラムできません。<br>**注記** : XSK_EFUSEPL_PROGRAM_AES_AND_USER_LOW パラメーターに対応する eFUSE ビットはありません。このパラメーターは、AES KEY および USER LOW パラメーター フィールドの変更を可能にし、ザイリンクスのプログラマ設定ファイル ジェネレーター ツールでの AES KEY および USER LOW のプログラミング命令を制御します。 | TRUE、FALSE (FALSE) | |
| XSK_EFUSEPL_PROGRAM_USER_HIGH_KEY | TRUE の場合は、USER HIGH パラメーターの変更を可能にし、特定の RSA HASH 値をプログラムできる。<br>FALSE の場合は、USER HIGH 値を無視し、USER HIGH eFUSE ビットを変更できない。<br>**注記** : この XSK_EFUSEPL_PROGRAM_USER_HIGH_KEY パラメーターに対応する eFUSE ビットはありません。このパラメーターは、USER HIGH パラメーターのフィールドの変更を可能にし、ザイリンクスのプログラマ設定ファイル ジェネレーター ツールでの USER HIGH プログラミング命令を制御します。 | TRUE、FALSE (FALSE) | |
| AES KEY | 暗号化されたブート イメージ用の AES キー (FUSE_KEY)。<br>XSK_EFUSEPL_PROGRAM_AES_AND_USER_LOW パラメーターを TRUE に設定して、この値の変更とプログラミングを可能にする。<br>`aes.nky` ファイルから上記の AES KEY 値または Key 0 値をコピーして、このフィールドにペーストする。<br>この値の長さは 64 文字で、有効な文字は 0-9、a-f、および A-F となり、<br>その他の文字は無効として認識されるため、プログラムされない。<br>AES キー値を指定した場合、XSK_EFUSEPL_DISABLE_AES_KEY_READ も TRUE に設定してキーの読み出しを禁止する必要がある。<br>**注記** : AES KEY と USER LOW の値は、別々にプログラムできません。<br>参考資料 : 『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 2] の「AES」、『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] の「FUSE_KEY」、および『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 1] の「AES キーの生成」 | FUSE_KEY 用の 64 桁の 16 進数 [0:255]<br>(すべて 0) | |

表 1 : セキュリティ キーおよび eFUSE 設定シート (続き)

| | | | |
|---|---|---|---|
| USER HIGH | FUSE_USER[31:0] 値のビット [31:8]。<br>XSK_EFUSEPL_PROGRAM_USER_HIGH_KEY パラメーターを TRUE に設定して、この値の変更とプログラミングを可能にする。<br>この値の長さは 6 文字で、有効な文字は 0-9、a-f、および A-F となり、その他の文字は無効として認識されてプログラムできない。<br>『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] の「FUSE_USER」を参照。 | 6 桁の 16 進数 (すべて 0) | |
| USER LOW | FUSE_USER[7:0] 値のビット [31:0]。<br>XSK_EFUSEPL_PROGRAM_USER_LOW_KEY パラメーターを TRUE に設定して、この値の変更とプログラミングを可能にする。<br>この値の長さは 6 文字で、有効な文字は 0-9、a-f、および A-F となり、その他の文字は無効として認識されてプログラムできない。<br>**注記** : AES_KEY と USER LOW の値は、別々にプログラムできません。<br>『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) [参照 3] の「FUSE_USER」を参照。 | 2 桁の 16 進数 (すべて 0) | |

## ステップ 3 : プログラマの検証ブート イメージ ファイルを作成する

プログラマの検証ブート イメージは、対応する eFUSE の設定をテストするために、RSA 認証 (オプション) または AES 暗号化 (オプション) されています。スタンドアロン検証動作では、デバイス プログラマはこの検証ブート イメージを使用してデバイスを起動します。Zynq-7000 AP SoC がこのブート イメージを認証および復号化できた場合には、検証ブート イメージ アプリケーションが MIO (多目的 I/O) を介してプログラマへ完了ステータスを送信します。一方、ブート エラーが生じた場合、デバイスはセキュア ロックダウン ステートへ遷移するため、プログラマはデバイスからブート完了ステータスを受信しません。

このセクションでは、リファレンス デザインの検証アプリケーションからデバイス プログラマ用のセキュアな検証ブート イメージを構築する際のガイドラインを示します。

**注記** : ユーザーの標準ブート イメージを用いる AES 暗号および RSA サイン フローをすでに開発している場合は、リファレンス デザインからサンプル `verify.elf` ファイルを使用して同じフローを使用できます。

**注意** : 完全なカスタマー アプリケーション (`.elf/.bit`) デザインは、「検証ブート イメージのカスタマイズにおける要件」で説明する制約により、一般的なデバイス プログラマでは検証ブート イメージとして使用できません。

## プログラマの検証ブート イメージ ファイルを作成する場合の要件

プログラマの検証ブート イメージ ファイルを作成する前に必要な手順を次に示します。

- リファレンス デザイン (zip) アーカイブを C:\ ドライブの直下に解凍します。
- ザイリンクスのソフトウェア開発キット (SDK) をインストールします。

  SDK Standalone WebInstall Client は、http://japan.xilinx.com/support/download/index.html の [エンベデッド開発] タブからダウンロード可能です。

- 「セキュリティ キーおよび eFUSE 設定シート」から、これらの設定またはファイルを取得します。
  - RSA 認証を使用する場合 :
    - RSA プライマリ非公開キー ファイル : `psk.pem`
    - RSA プライマリ公開キー ファイル : `ppk.pub`
    - RSA セカンダリ非公開キー ファイル : `ssk.pem`
    - RSA セカンダリ公開キー ファイル : `spk.pub`
  - AES 暗号化を使用する場合 :
    - AES キー値
    - AES StartCBC 値
    - HMAC キー値

## プログラマの検証ブート イメージ ファイルを作成する

ザイリンクス SDK に含まれる Bootgen ユーティリティを使用して、ブート イメージ ファイルを生成します。Bootgen ユーティリティは、`verify.bif` ファイルから検証ブート イメージ ファイルを作成するためにパラメーターを取得します。`verify.bif` ファイルは、リファレンス デザインで提供されており、内容は次のとおりです。

```
the_ROM_image:
{
        [aeskeyfile]aes.nky
        [ppkfile]ppk.pub
        [pskfile]psk.pem
        [spkfile]spk.pub
        [sskfile]ssk.pem
        [bootloader, encryption=aes, authentication=rsa]..\Debug\verify.elf
}
```

説明 :

`aes.nky` = ザイリンクス ファイル形式の AES/HMAC キーの入力ファイル名

`ppk.pub` = RSA プライマリ公開キーの入力ファイル名

`psk.pem` = RSA プライマリ非公開キーの入力ファイル名

`spk.pub` = RSA セカンダリ公開キーの入力ファイル名

`ssk.pem` = RSA セカンダリ非公開キーの入力ファイル名

ブートローダーの行 :

`encryption=aes` = AES 暗号のブートロードを有効化

`authentication=rsa` = RSA 認証のブートロードを有効化

`verify.elf` = リファレンス デザインのブート アプリケーション コードを検証

検証ブート イメージの作成手順 :

1. Windows のコマンド プロンプトを開きます。
2. 解凍したリファレンス デザインがある `C:\xapp1278\Zynq7000\verify\verify.sdk\verify\bootimage\` ディレクトリへ移動します。
3. ザイリンクスの Bootgen ユーティリティを実行可能であるかを確認します。
   a. 次を入力して、Windows が Bootgen を検索できるかテストします。
      ```
      bootgen
      ```
   b. Bootgen の検索にパス設定が必要な場合は、次を入力します。
      ```
      call C:\Xilinx\SDK\2015.4\settings64.bat
      ```
      つまり、`C:\Xilinx\SDK\2015.4\` は、ザイリンクス SDK がインストールされている場所を示しています。

      **注記** : SDK バージョンが 2015.4 以外の場合は、`C:\Xilinx\SDK\` ディレクトリで適切なバージョンを確認し、それに従って、このパスを変更してください。
4. RSA 認証を使用する場合は、これらのファイルをローカル ディレクトリにコピーします (ステップ 2 のパスを参照)。

   `ppk.pub` = ユーザーの RSA プライマリ公開キー ファイル

   `psk.pem` = ユーザーの RSA プライマリ非公開キー ファイル

   `spk.pub` = ユーザーの RSA セカンダリ公開キー ファイル

   `ssk.pem` = ユーザーの RSA セカンダリ非公開キー ファイル

   **注記** : RSA 認証を使用しない場合は、`verify.bif` ファイルでこれらの 4 つのキー ファイルについて記述されている行を削除し、またブートローダーの行から「authentication=rsa」を削除してください。
5. AES 暗号を使用する場合、`aes.nky` ファイルのパラメーターを、「セキュリティ キーおよび eFUSE 設定シート」のユーザーのキー値で置き換えます。リファレンス デザインのサンプル `aes.nky` ファイルの内容は次のとおりです。

   ```
   Device xc7z010;
   Key 0        0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF;
   Key StartCBC 00000000000000000000000000000000;
   Key HMAC     0000000000000000000000000000000000000000000000000000000000000000;
   ```

   説明 :

   Key 0 <64 桁の 16 進数値> = 256 ビット AES キー値

   Key StartCBC <32 桁の 16 進数値> = 128 ビット AES ブロック暗号の初期化ベクター

   Key HMAC <64 桁の 16 進数> = 256 ビット HMAC キー値

   **注記** : AES 認証を使用しない場合は、`verify.bif` ファイルで `aes.nky` ファイルについて記述されている行を削除し、またブートローダーの行から「encryption=aes」を削除してください。
6. これらのオプションを使用して Bootgen を実行し、RSA 認証 (オプション) および AES 暗号化 (オプション) された検証ブート イメージを作成します。

   ```
   bootgen -image verify.bif -efuseppkbits efuse_ppk_hash.txt -encrypt efuse -w on -o verify.mcs
   ```

   出力は次のとおりです。

   `verify.mcs` = デバイス プログラマ用の検証ブート イメージ

   `efuse_ppk_hash.txt` = RSA ハッシュ 64 桁の eFUSE 設定が含まれる

   **注記** : 状況に応じて、Bootgen コマンド ラインから次のオプションを削除します。

   - RSA 認証を使用しない場合 : `-efuseppkbits efuse_ppk_hash.txt` を削除
   - AES 暗号化を使用しない場合 : `-encrypt efuse` を削除
7. 必要に応じて、次のファイルを保存します。

`verify.mcs` = デバイス プログラマへ送信するために保存

`aes.nky` = 「ステップ 4 : プログラマ設定ファイルを作成する」のために保存

`efuse_ppk_hash.txt` = 「ステップ 4 : プログラマ設定ファイルを作成する」のために保存

# ステップ 4 : プログラマ設定ファイルを作成する

プログラマ設定ファイルには、デバイス プログラマでプログラムされる eFUSE 設定が含まれます。プログラマ設定ファイルを作成するために、リファレンス デザインには Tcl ベースのザイリンクスのプログラマ設定ファイル ジェネレーター ツールが提供されています。

このセクションでは、ザイリンクスのプログラマ設定ファイル ジェネレーター ツールを使用してプログラマ設定ファイルを作成する手順を説明します。Zynq-7000 AP SoC eFUSE 設定シートに定義した設定値を適用する準備をしてください。RSA 認証を使用する場合、RSA ハッシュ値は、「プログラマの検証ブート イメージ ファイルを作成する」の `efuse_ppk_hash.txt` ファイル内にあります。

## ザイリンクスのプログラマ設定ファイル ジェネレーター ツールの要件

プログラマ設定ファイルを作成する前に必要な手順を次に示します。

1. リファレンス デザイン (zip) アーカイブを C:\ ドライブの直下に解凍します。
2. Tcl/Tk 8.5 スクリプト インタープリターをインストールします。Tcl/Tk 8.5 スクリプト インタープリターは、http://www.activestate.com/activetcl/downloads からダウンロードできます。

**重要** : 8.5.*.* バージョンをダウンロードしてください。ザイリンクスのプログラマ設定ファイル ジェネレーター ツールは、Tcl 8.6 またはそれ以降のバージョンに対応していません。

3. Tcl 8.5 スクリプト インタープリターをインストールした後、tclsh85 を起動し、tclsh85 コマンド ラインに次のコマンドを入力して Iwidgets パッケージをインストールします。

```
teacup install Iwidgets
```

4. 「ステップ 3 : プログラマの検証ブート イメージ ファイルを作成する」から次に示すファイルを取得します。
   - RSA 認証を使用する場合は、RSA ハッシュ値用に `efuse_ppk_hash.txt` ファイルを取得します。
   - AES 暗号化を使用する場合は、AES キー値用に`aes.nky` ファイルを取得します。

## ザイリンクスのプログラマ設定ファイル ジェネレーター ツールを使用する

プログラマ設定ファイルの作成手順

1. Windows のコマンド プロンプトを開きます。
2. 解凍したリファレンス デザインがある `C:\xapp1278\Zynq7000_programmer_settings\` ディレクトリへ移動します。
3. 次のコマンドを入力して、ザイリンクスのプログラマ設定ファイル ジェネレーター ツールを起動します。

```
tclsh85 zynq7000_programmer_settings_mcs_file_generator.tcl
```

   プログラマ設定ファイル ジェネレーター ツールのインターフェイスは、図 2 を参照してください。

4. Zynq-7000 AP SoC eFUSE 設定シートで定義した値を使用して、プログラマ設定ファイル ジェネレーター ツールでパラメーターを設定します。確実な値を入力するには、次の方法も利用できます。
   - RSA 認証を使用する場合は、`efuse_ppk_hash.txt` ファイルの RSA ハッシュ値を、プログラマ設定ファイル ジェネレーター ツールの RSA HASH フィールドにコピーします。
   - AES 暗号化を使用する場合は、`aes.nky` ファイルの AES key 0 値を、プログラマ設定ファイル ジェネレーター ツールの AES KEY フィールドにコピーします。
5. [Generate File] をクリックして `programmer_settings.mcs` ファイルを生成します。

**注記** : programmer_settings.mcs ファイルは、バイナリ ファイルとして扱ってください。プログラマ設定ファイルは手動で変更しないでください。デバイス プログラマは、特定フォーマットのファイル データに対応します。

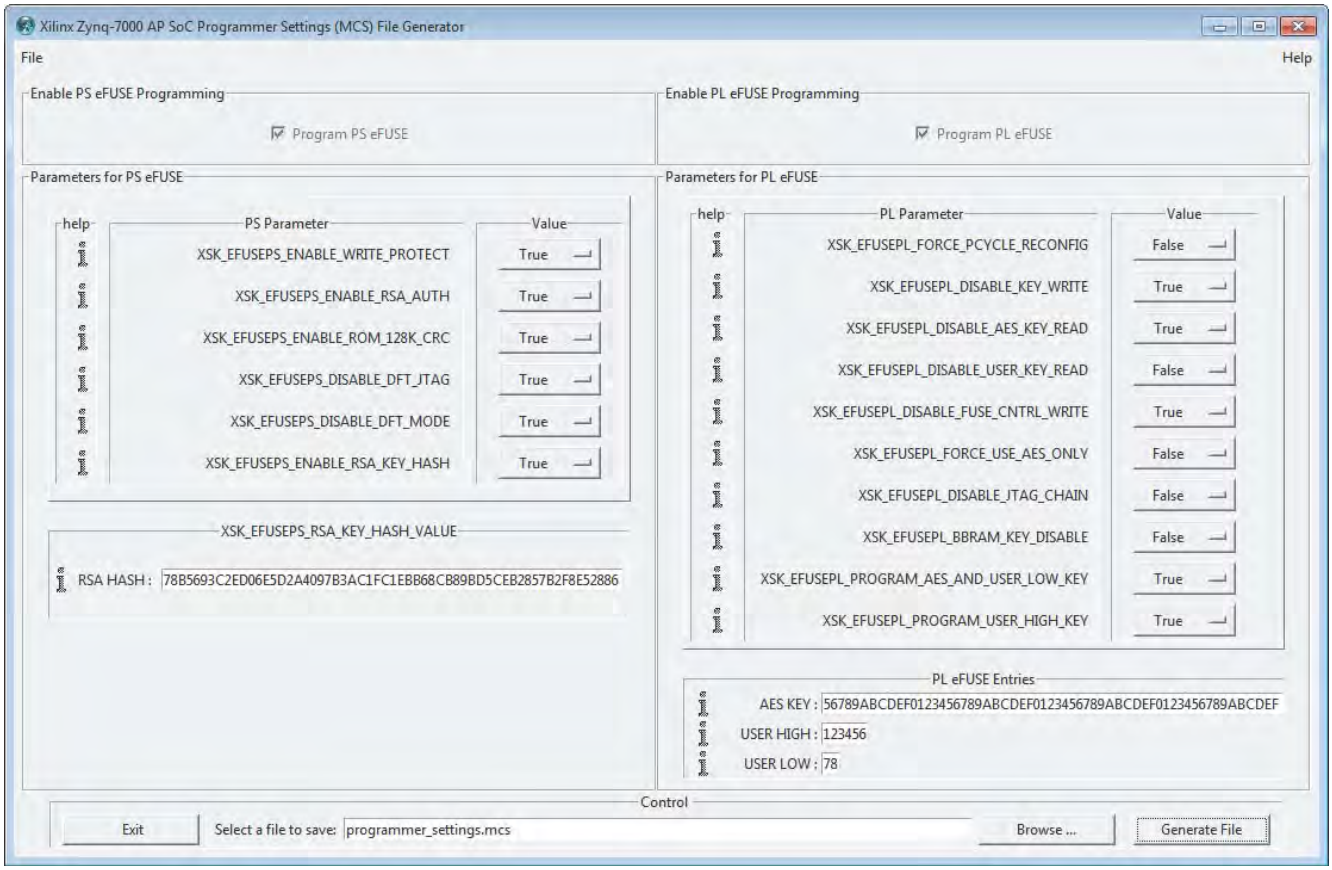



**図 2 : プログラマ設定ファイル ジェネレーター ツールの画面表示の例**

プログラマ設定ファイル ジェネレーター ツールを使用する際に役に立つヒントを次に示します。

- 「i」シンボルの上にマウスを配置すると、各パラメーターの説明が表示されます。
- Value ボタンをクリックすると、[False] または [TRUE] の切り替えが可能です。
- RSA HASH、AES KEY、USER HIGH、USER LOW 入力ボックスには、それぞれ 16 進数値を入力します。

  **注記** : これらのフィールドには対応するパラメーターがあり、各フィールドの編集を可能にするには TRUE に設定する必要があります。たとえば、RSA HASH 値の入力ボックスを編集可能にするには、XSK_EFUSE_PS_ENABLE_RSA_KEY_HASH パラメーターを TRUE に設定する必要があります。編集時に入力ボックスの背景が黄色表示されている場合は、16 進数の桁数が足りない、または 16 進数以外の文字が含まれていることを意味します。

**推奨** : 入力ミスを防ぐために、efuse_ppk_hash.txt ファイルから直接 RSA HASH 値をコピーして、入力ボックスにペーストしてください。

**推奨** : 入力ミスを防ぐために、aes.nky ファイルから直接 AES KEY 値をコピーして、入力ボックスにペーストしてください。

## ステップ 5 : ザイリンクスの構築済みプログラミング アルゴリズム イメージ ファイルを指定する

ザイリンクスの構築済みプログラミング アルゴリズム イメージ ファイルは、eFUSE をプログラムするための PS ARM プロセッサ コードを含むブート イメージです。このプログラミング アルゴリズム イメージ ファイルは、次のディレクトリにある解凍済みのリファレンス デザイン ファイル内に含まれています。

```
C:\xapp1278\Zynq7000_programming_algorithm_image_file\32MHz_QSPI0_program_01_08_16_slowcpu_Pl_ps.mcs
```

## ステップ 6 : デバイス サンプルとプログラミング ファイルを送信する

表 2 にリストされている 3 つのファイルを指定および作成した後、これらのファイルはプログラムされるサンプル デバイスと共にデバイス プログラミング サイトへ安全に送信される必要があります。

表 2 : デバイス プログラマ用ファイル

| ファイル | ソースおよびリファレンス ファイル名 (タイプ) | 説明 |
|---|---|---|
| ファイル [A] | ユーザーが生成したプログラマ設定ファイル<br>`programmer_settings.mcs`<br>(バイナリ形式の MCS ファイル) | このファイルには、PS eFUSE および PL eFUSE にプログラムされるユーザー定義の設定が含まれます (PS RSA PPK ハッシュおよび AES キー )。リファレンス デザインには、このファイルを作成するためにザイリンクスのプログラマ設定ファイル ジェネレーター ツールが提供されています。このファイルは、バイナリ ファイルとしてプログラマ バッファーにロードされる必要があります。プログラマは、Zynq-7000 AP SoC UART1 インターフェイスを介して、このファイルを OCM のプログラミング アルゴリズム コードへ送信します。 |
| ファイル [B] | ザイリンクスの構築済みプログラミング アルゴリズム ブート イメージ ファイル<br>`32MHz_QSPI0_program_01_08_16_slowcpu_Pl_ps.mcs`<br>(Intel Hex 形式の MCS ファイル) | このブート イメージには、PS eFUSe および PL eFUSE 用のプログラミング アルゴリズム コードが含まれます。ザイリンクスは、リファレンス デザインで、この構築済みブート イメージを提供しています。この Intel HEX 形式の MCS ファイルのデータ コンテンツは、プログラマ バッファーへロードされます。このコードは OCM にロードされて、Zynq-7000 AP SoC PS 内で実行されます。 |
| ファイル [C] | ユーザーが作成したセキュアな検証ブート イメージ ファイル<br>`verify.mcs`<br>(Intel Hex 形式の MCS ファイル) | ユーザーは、ザイリンクスが提供するリファレンス デザインをベースにして、RSA 認証 (オプション) および AES 暗号化 (オプション) のファイルを作成します。ブート ロード完了後、リファレンス デザインは指定された Zynq-7000 AP SoC GPIO status[3:0] ピンに Validation_image_passed (`0xA`) の値を駆動します。この Intel HEX 形式の MCS ファイルのデータ コンテンツは、プログラマ バッファーへロードされます。このセキュア ブート イメージは、一致する RSA PPK ハッシュまたは AES キーで Zynq-7000 AP SoC デバイスの eFUSE がプログラムされているかを検証するスタンドアロン検証で使用されます。 |

ここまでが、デバイス プログラマのファイル準備プロセスです。

# デバイス プログラマの設定

デバイス プログラマの詳しい設定方法は、各デバイス プログラマ ベンダーが提供するガイドラインを参照してください。デバイス プログラマの設定には、表 2 にリストしたファイルが必要です。このセクションでは、参照用として BPM Microsystems 社の 2800 プログラマ アプリケーションと XC7Z010-CLG400 デバイスを使用する場合の設定例を示します。BPM Microsystems 社のプログラマは、すべてのファイル データがリニア バッファー メモリ空間にロードされていることを要求します。表 3 に XC7Z010-CLG400 デバイスの場合でのバッファー メモリ マップを示します。

**表 3 : XC7Z010-CLG400 デバイスの場合での BPM Microsystems 社のバッファー メモリ マップ**

| 開始アドレス (16 進数) | 終了アドレス (16 進数) | 説明 |
|---|---|---|
| 00000 | 3FFFF | 32 MHz_QSPI0_program_01_08_16_slowcpu_Pl_ps.mcs : BPM Microsystems 社製プログラマ ソケット モジュールの QSPI0 用のザイリンクスの構築済みプログラミング アルゴリズム ファイルです。 |
| 40000 | 7FFFF | verify.mcs : BPM Microsystems 社製プログラマ ソケット モジュールの QSPI1 用のザイリンクスの構築済みプログラミング アルゴリズム ファイルです。 |
| 80000 | 827FF | programmer_settings.mcs : Zynq-7000 AP SoC デバイスにプログラムされるユーザー定義の eFUSE 設定ファイルです。 |

BPM Microsystems 社のプログラマ アプリケーションのバッファー メモリ マップにファイルをロードする方法は次のとおりです。

1. ザイリンクスの XC7Z010-CLG400 デバイスを選択します。
2. programmer_settings.mcs ファイル [A] をプログラマのバッファー メモリ (開始アドレス 00000h) にロードします。ファイルをロードする場合、バッファーをクリアしてからバイナリ ファイルとしてファイルをロードします。
3. バッファー メモリを編集し、アドレス 80000h にアドレス範囲 00000h-027FFh をコピーします。
4. verify.mcs ファイル [C] をプログラマのバッファー メモリ (開始アドレス 00000h) にロードします。ファイルをロードする際にはバッファーをクリアせずに、Intel Hex 形式でファイルをロードします。
5. バッファー メモリを編集し、アドレス 40000h にアドレス範囲 00000h-3FFFFh をコピーします。
6. 32MHz_QSPI0_program_01_08_16_slowcpu_Pl_ps.mcs ファイル [B] をプログラマのバッファー メモリ (開始アドレス 00000h) にロードします。ファイルをロードする際にはバッファーをクリアせずに、Intel Hex 形式でファイルをロードします。
7. 今後のデバイス プログラミング用にバッファー メモリをファイルに保存します。

プログラマの設定完了後、デバイス プログラマを使用してデバイスをプログラムおよびスタンドアロン検証できます。

このアプリケーション ノートで説明するソリューションの場合、プログラマにはソケット アダプターに 2 つの Quad SPI フラッシュ デバイスがあります。プログラマは、一方の Quad SPI フラッシュ (QSPI_0) デバイスをザイリンクスのプログラミング アルゴリズム イメージ ファイルでプログラムし、もう一方の Quad SPI フラッシュ (QSPI_1) デバイスを検証ブート イメージでプログラムします。

プログラム動作では、プログラマは次を実行します。

- QSPI_0 フラッシュからターゲット デバイスを起動して、プログラミング アルゴリズムをロードします。
- プログラミング設定ファイルをデバイスに送信して、各デバイスの eFUSE をプログラムします。
- プログラム済み eFUSE 設定の最終検証として、QSPI_1 からデバイスを起動します。

検証動作では、プログラマは次を実行します。

- QSPI_1 からターゲット デバイスを起動して、検証ブート イメージのロードを可能にする適切な eFUSE 設定がターゲット デバイスに含まれていることを検証します。

**重要** : BPM Microsystems 社のプログラマは、プログラム動作中にソケット アダプター上の Quad SPI フラッシュをプログラムします。したがって、新たにスタンドアロン検証動作を実行する前には必ずプログラム動作を実行する必要があります。そうしなければ、スタンドアロン検証動作は異なるプログラミング ジョブに属する Quad SPI フラッシュ イメージからターゲット デバイスを起動することになります。

# プログラムされたデバイスの確認

このセクションでは、プログラミング ファイル、プログラマの設定、およびプログラムされたデバイスの適正を確認する推奨方法を示します。

## デバイス プログラマのファイルおよび設定の確認

このセクションでは、デバイス プログラマのファイルおよび設定を確認するための推奨方法を示します。

デバイス プログラマの設定が完了した後、「完了」が予測される場合と「エラー」が予測される場合の両方を実行することによって、有効な検証ブート イメージであることを検証できます。

1. 1 つのサンプル デバイス (sample #1) をプログラムします。
2. この sample #1 に対してスタンドアロン検証を実行 - 問題なく完了することが予測される
3. あらかじめプログラムされていない 2 つ目のサンプル デバイス (sample #2) に対してスタンドアロン検証を実行 - エラーの発生が予測される

上記の 2 または 3 のいずれかの結果が予測と異なる場合は、ファイル提供者に潜在的な問題を確認してください。

# プログラムされたデバイス サンプルの確認

このセクションでは、サンプル デバイスが適切な設定でプログラムされていることを確認するための推奨方法を示します。

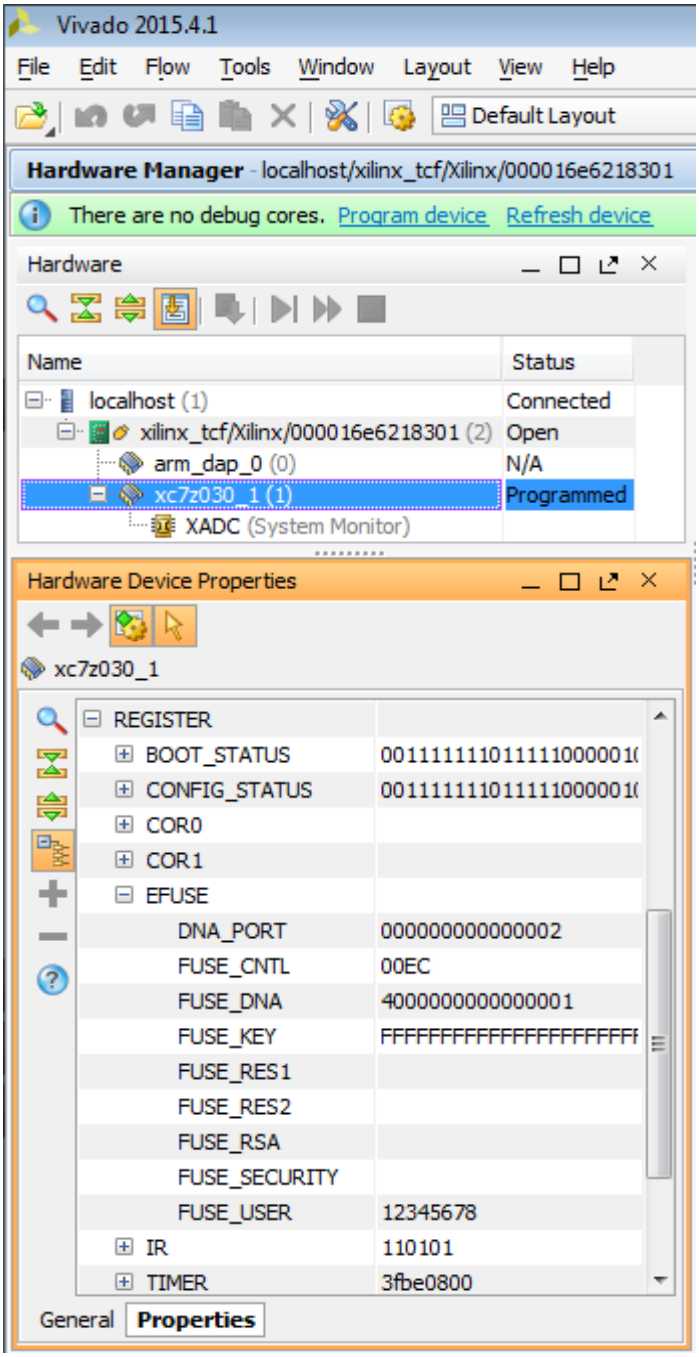
サンプル デバイスがプログラムされた後、実際にデバイスをボード上にアセンブルして確認します。

1. プログラムされたデバイスを含むボードに JTAG で接続された Vivado ハードウェア マネージャーを使用して、PL eFUSE の設定を確認します。Zynq-7000 AP SoC デバイスの場合は、[Hardware Device Properties] ウィンドウで [REGISTER] → [EFUSE] を表示して設定を確認します。図 3 に、ウィンドウ表示の例を示します。
    a. AES キーがプログラムされている場合は、FUSE_CNTL[3] ビットが 1 にプログラムされていることを確認します。FUSE_CNTL[3] は、AES キーの読み出し保護機能です。

       **注記** : FUSE_KEY が読み出し保護されている場合、この値は、非表示またはすべて「F」の 16 進数として表示されます。Vivado の [Hardware Device Properties] ウィンドウでは、FUSE_CNTL 値はビット順が [13:0] の 16 進数となります。表 4 に、PL eFUSE 設定に対応する FUSE_CNTL のビット位置を示します。

    b. 32 ビットのユーザー定義値がプログラムされている場合は、Vivado の [Hardware Device Properties] ウィンドウで FUSE_USER[31:0] の値を確認します。

       **注記** : Vivado の [Hardware Device Properties] ウィンドウの FUSE_RSA プロパティは無視してください。FUSE_RSA プロパティは、Zynq-7000 AP SoC の PS eFUSE RSA 値ではありません。




X15925-021116

図 3 : Vivado ハードウェア マネージャーの FUSE_CNTL 値

表 4 : PL eFUSE 設定の FUSE_CNTL[13:0] ビットの割り当て

| FUSE_CNTL (ビット インデックス) | PL eFUSE パラメーター名 |
|---|---|
| 1 | XSK_EFUSEPL_FORCE_PCYCLE_RECONFIG |
| 2 | XSK_EFUSEPL_DISABLE_KEY_WRITE |
| 3 | XSK_EFUSEPL_DISABLE_AES_KEY_READ |
| 4 | XSK_EFUSEPL_DISABLE_USER_KEY_READ |
| 5 | XSK_EFUSEPL_DISABLE_FUSE_CNTRL_WRITE |
| 8 | XSK_EFUSEPL_FORCE_USE_AES_ONLY |
| 9 | XSK_EFUSEPL_DISABLE_JTAG_CHAIN |
| 10 | XSK_EFUSEPL_BBRAM_KEY_DISABLE |

**注記** :

この表に含まれてないビット位置は、ザイリンクス専用として予約されているため、任意の値が可能です。

2. ザイリンクスの SDK 2015.4 (またはそれ以前のバージョン) に含まれる Xilinx Microprocessor Debugger (XMD) ツールを使用して、PS eFUSE の設定を確認します。XMD ツールの使用手順は次のとおりです。
   a. Windows のコマンド プロンプトを開きます。
   b. 解凍したリファレンス デザインの `C:\xapp1278\Zynq7000_ps_efuse_check_method\` ディレクトリへ移動します。
   c. 次のコマンドを入力して XMD ツールを起動します。

      ```
      xmd
      ```

      XMD が起動しない場合は、次のコマンドを入力して Windows が XMD を検出できるようにパスを設定します。

      ```
      call C:\Xilinx\SDK\2015.4\settings64.bat
      ```

      つまり、`C:\Xilinx\SDK\2015.4\` は、ザイリンクス SDK がインストールされている場所を示しています。

      **注記** : SDK バージョンが 2015.4 以外の場合は、`C:\Xilinx\SDK\` ディレクトリで適切なバージョンを確認し、それに従って、このパスを変更してください。

   d. XMD ツールがローカル ディレクトリの `xmd.ini` スクリプトを実行します。`xmd.ini` スクリプトの出力サンプルは次のとおりです。

      ```
      ===============================================================
      PS eFUSE Bit Function Name     =Value- Programmed Status
      ------------------------------ ----- ------------------------
      eFUSE Write Protection (2-bits) = 00 - NOT programmed
      OCM ROM 128KB CRC Enable        = 0  - NOT programmed
      RSA Authentication Enable       = 1  - **PROGRAMMED** <---
      DFT JTAG Disable                = 0  - NOT programmed
      DFT Mode Disable                = 0  - NOT programmed
      ===============================================================
      ```

3. 実際のアプリケーション イメージを起動して、問題なく起動することを確認します。

   eFUSE 設定が予想と異なる場合、またはアプリケーション イメージがブート エラーとなった場合は、プログラマの設定を確認してください。

# トラブルシューティング

プログラミング アルゴリズムは、プログラム動作中に表 5 に示すステータス値をレポートできます。

**表 5 : プログラミング アルゴリズムの STATUS[3:0] 値**

| STATUS[3:0] 値 (バイナリ) | プログラミング アルゴリズム ステータス名 | 説明 |
|---|---|---|
| 0000 | PS_eFUSE_driver_booted_error | イメージがブート エラーとなった場合の MIO ピンのデフォルト値です。この場合、MIO ピンはトライステートのままとなり、プログラマ ソケットのプルダウン終端によって STATUS[3:0] ピンが Low を維持します。<br>ソケットがプログラマに正しくインストールされていることを確認します。<br>デバイスがソケットに正しく固定されていることを確認します。<br>プログラマには最新のザイリンクス プログラミング アルゴリズム ブート イメージ ファイル [B] が含まれていることを確認します。<br>ザイリンクス プログラミング アルゴリズム ブート イメージ ファイルがプログラマに正しくロードされていることを確認します。 |
| 0001 | PS_eFUSE_driver_booted | ザイリンクス プログラミング アルゴリズム ブート イメージ ファイルによるブートが問題なく完了した場合に、この値がレポートされます。<br>このステータスでプログラム動作が終了した場合は、`programmer_settings.mcs` ファイルに関する次の事項を確認してください。<br>• このファイルが適切に生成されている<br>• このファイルの最後に「efuse program」テキストが含まれている<br>• このファイルがプログラマに適切にロードされている |
| 0010 | PS_eFUSE_driver_programming_started | PS eFUSE プログラミングが開始すると、この値がレポートされます (PL eFUSE プログラミングの後)。 |
| 0011 | PS_eFUSE_driver_programming_completed | PS eFUSE プログラミングが完了すると、この値がレポートされます。 |
| 0100 | PS_eFUSE_driver_validation_passed | PL および PS eFUSE が正常にプログラムされて検証された後、この値がプログラム動作の最後でレポートされます。 |
| 0101 | PS_eFUSE_driver_validation_failed | PS eFUSE のプログラム中にプログラミング アルゴリズムでエラーが検出されると、この値がプログラム動作の最後にレポートされます。 |
| 0110 | PL_eFUSE_driver_programming_started | プログラミング アルゴリズムがブートを開始し、「efuse program」コマンドを含む `programmer_settings.mcs` ファイル データをアルゴリズムが受信した後に、この値がレポートされます。この値は、PL eFUSE プログラミングの開始を示します。 |
| 0111 | PL_eFUSE_driver_programming_completed | PL eFUSE プログラミングが完了すると、この値がレポートされます。 |

表 5 : プログラミング アルゴリズムの STATUS[3:0] 値 (続き)

| STATUS[3:0] 値 (バイナリ) | プログラミング アルゴリズム ステータス名 | 説明 |
|---|---|---|
| 1000 | PL_eFUSE_driver_validation_passed | PL eFUSE プログラミングがエラーなしで完了した場合、この値がレポートされます。 |
| 1001 | PL_eFUSE_driver_validation_failed | PL eFUSE プログラミングでエラーが発生すると、この値がレポートされます。 |
| 1100 | INVALID_CMD | 無効なコマンドを含む programmer_settings.mcs ファイルをプログラミング アルゴリズムが受信した場合に、この値がレポートされます。<br>programmer_settings.mcs ファイルが適切に生成され、最初に生成された形式から変更されていないことを確認してください。 |
| 1101 | INVALID_CHECKSUM | プログラミング アルゴリズムが programmer_settings.mcs ファイル データを受信しているが、そのデータの特定ラインのチェックサムが不正な場合に、この値がレポートされます。<br>programmer_settings.mcs ファイルが適切に生成され、最初に生成された形式から変更されていないことを確認してください。 |

# アドバンス デバイス プログラマ ファイルの詳細

このセクションでは、アドバンス デバイス プログラマ アプリケーションの詳細を示します。

## 検証ブート イメージのカスタマイズにおける要件

検証ブート イメージは、必要に応じてカスタマイズできますが、カスタマイズしたブート イメージは、デバイス プログラマ ソケット ハードウェアの制約事項を満たす必要があります。

通常、デバイス プログラマ ハードウェアは、eFUSE プログラミングをサポートするのに必要最小限の機能を備えて構築されます。したがって、カスタマイズした検証ブート アプリケーションは、この最小限のシステム サポート機能のみで動作する必要があります。この後のセクションで、カスタマイズした検証ブート イメージのガイドラインを示します。

## 検証ブートの STATUS[3:0]

スタンドアロン検証動作が完了すると、デバイス プログラマは各デバイス ピンから特定の 4 ビットの STATUS[3:0] 値を受信します。表 6 に有効な STATUS[3:0] 値を示します。

表 6 : 検証ブートの STATUS[3:0] 値

| STATUS[3:0] 値 (バイナリ) | 検証結果名 (リファレンス デザインの ブート アプリケーション コード) | 説明 |
|---|---|---|
| 1010 | Validation_image_passed | リファレンス デザインのブート アプリケーション コードでこの値が出力されると、ブートが正常に完了したことを示します。その他の値は、スタンドアロン検証動作でエラーが生じたことを示します。 |
| 1011 | Validation_image_failed | コードの追加検証用に、エラーが生じているカスタム ブート アプリケーション コードでのみ使用されます。 |
| 0000 | PS_eFUSE_driver_booted_error | イメージがブート エラーとなる場合の MIO ピンのデフォルト値です。この場合、MIO ピンはトライステートのままとなり、プログラマ ソケットのプルダウン終端によって STATUS[3:0] ピンが Low を維持します。 |

ブート アプリケーションは、PS の汎用 I/O (GPIO) を介して 4 ビットの STATUS[3:0] 値を表 7 にリストされた MIO ピンに出力する必要があります。

表 7 : 検証ブートの STATUS[3:0] ビットと MIO ピンの割り当て

| STATUS ビット | MIO の割り当て | I/O タイプ |
|---|---|---|
| STATUS[3] | MIO37 | GPIO 出力、LVCMOS18 |
| STATUS[2] | MIO36 | GPIO 出力、LVCMOS18 |
| STATUS[1] | MIO35 | GPIO 出力、LVCMOS18 |
| STATUS[0] | MIO34 | GPIO 出力、LVCMOS18 |

選択したデバイス/パッケージにおける各 MIO ピンの位置は、Zynq-7000 AP SoC パッケージ ファイルを参照してください。

## システムの属性と制約

ブート イメージは、次のハードウェアおよび PS の属性と互換性がある必要があります。

- PS_CLK 周波数 = 32MHz
- APU クロック周波数 = 200MHz (電力の制約のため)
- I/O 電圧範囲 = 1.8V
- DDR メモリなし - ブート アプリケーションのメモリ フットプリントは PS OCM と一致する必要がある

**注記** : DDR、トランシーバー、および SelectIO バンクは、デバイス プログラマ上で使用されないため、電源を供給する必要はありません。

## ブート フラッシュの制約

Quad SPI ブート フラッシュのサイズは 128Mb です。検証ブート イメージは、このフラッシュ内に収まる必要があります。

## 電力の制約

デバイス プログラマは、それぞれの電源において、電源投入時の最小電流要件または最大スタティック電流引き込み要件のいずれか大きい方を満たすのに十分なだけの電流が供給されます。デバイス プログラマは、200MHz の最小クロック周波数で PS アプリケーション プロセッサ ユニット (APU) を実行するのに十分なだけの電流が供給されます。

### ランタイムの制約

ブート イメージは、電源を投入してから 2 秒以内に起動して結果をレポートする必要があります。レポートされない場合、デバイス プログラマがタイムアウトとなり、スタンドアロン検証のエラーをレポートします。

# リファレンス デザイン

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。

表 8 に、リファレンス デザインの詳細を示します。

**表 8 : リファレンス デザインの詳細**

| パラメーター | 説明 |
|---|---|
| **一般** | |
| 開発者 | Habib El-Khoury、Randal Kuramoto |
| ターゲット デバイス | Zynq-7000 AP SoC ファミリ |
| ソース コードの提供 | あり |
| ソース コードの形式 | C |
| 既存のザイリンクス アプリケーション ノート/リファレンス デザイン、またはサードパーティからデザインへのコード/IP の使用 | なし |
| **インプリメンテーション** | |
| 使用した合成ツール/バージョン | なし |
| 使用したインプリメンテーション ツール/バージョン | ザイリンクス SDK 2015.4 |
| スタティック タイミング解析の実施 | なし |
| **ハードウェア検証** | |
| ハードウェア検証の実施 | あり |
| 使用したハードウェア プラットフォーム | BPM Microsystems 社の 2800 プログラマ (XC7Z010-CLG400 デバイス用ソケット付き) |

# 参考資料

このアプリケーション ノートの参考資料は次のとおりです。


1. 『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175 : 英語版、日本語版)
2. 『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585 : 英語版、日本語版)
3. 『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470 : 英語版、日本語版)
4. 『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』(UG908 : 英語版、日本語版)

# 改訂履歴

次の表に、この文書の改訂履歴を示します。

| 日付 | バージョン | 内容 |
|---|---|---|
| 2016 年 3 月 23 日 | 1.0 | 初版 |

# お読みください : 重要な法的通知

本通知に基づいて貴殿または貴社 (本通知の被通知者が個人の場合には「貴殿」、法人その他の団体の場合には「貴社」。以下同じ) に開示される情報 (以下「本情報」といいます) は、ザイリンクスの製品を選択および使用することのためにのみ提供されます。適用される法律が許容する最大限の範囲で、(1) 本情報は「現状有姿」、およびすべて受領者の責任で (with all faults) という状態で提供され、ザイリンクスは、本通知をもって、明示、黙示、法定を問わず (商品性、非侵害、特定目的適合性の保証を含みますがこれらに限られません)、すべての保証および条件を負わない (否認する) ものとします。また、(2) ザイリンクスは、本情報 (貴殿または貴社による本情報の使用を含む) に関係し、起因し、関連する、いかなる種類・性質の損失または損害についても、責任を負わない (契約上、不法行為上 (過失の場合を含む)、その他のいかなる責任の法理によるかを問わない) ものとし、当該損失または損害には、直接、間接、特別、付随的、結果的な損失または損害 (第三者が起こした行為の結果被った、データ、利益、業務上の信用の損失、その他あらゆる種類の損失や損害を含みます) が含まれるものとし、それは、たとえ当該損害や損失が合理的に予見可能であったり、ザイリンクスがそれらの可能性について助言を受けていた場合であったとしても同様です。ザイリンクスは、本情報に含まれるいかなる誤りも訂正する義務を負わず、本情報または製品仕様のアップデートを貴殿または貴社に知らせる義務も負いません。事前の書面による同意のない限り、貴殿または貴社は本情報を再生産、変更、頒布、または公に展示してはなりません。一定の製品は、ザイリンクスの限定的保証の諸条件に従うこととなるので、http://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。IP コアは、ザイリンクスが貴殿または貴社に付与したライセンスに含まれる保証と補助的条件に従うことになります。ザイリンクスの製品は、フェイルセーフとして、または、フェイルセーフの動作を要求するアプリケーションに使用するために、設計されたり意図されたりしていません。そのような重大なアプリケーションにザイリンクスの製品を使用する場合のリスクと責任は、貴殿または貴社が単独で負うものです。http://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。

**自動車用のアプリケーションの免責条項**

ザイリンクスの製品は、フェイルセーフとして設計されたり意図されてはおらず、また、フェイルセーフの動作を要求するアプリケーション (具体的には、(I) エアバッグの展開、(II) 車のコントロール (フェイルセーフまたは余剰性の機能 (余剰性を実行するためのザイリンクスの装置にソフトウェアを使用することは含まれません) および操作者がミスをした際の警告信号がある場合を除きます)、(III) 死亡や身体傷害を導く使用、に関するアプリケーション) を使用するために設計されたり意図されたりもしていません。顧客は、そのようなアプリケーションにザイリンクスの製品を使用する場合のリスクと責任を単独で負います。



この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。フィードバックは日本語で入力可能です。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。