



XAPP1225 (v1.0) 2014 年 10 月 23 日

# Zynq-7000 AP SoC システム メモリのランタイム インテグリティと認証チェック

著者 : Lester Sanders

## 概要

このアプリケーション ノートでは、Zynq®-7000 All Programmable SoC 用のランタイム インテグリティ チェッカー (RTIC) について説明します。RTIC ソフトウェアは、Zynq-7000 AP SoC デバイス上で動作し、外部メモリが変更されているかをチェックします。RTIC を定期的に行うことによって、不揮発性メモリやダブルデータレート (DDR) メモリといった比較的影響を受けやすい2つのシステム コンポーネントを不正な変更から有効に保護することができます。

## 含まれるシステム

リファレンス デザイン (xapp1225-rtic) には、次のソフトウェア プロジェクトが含まれています。

- fsbl
- hello\_world
- hello
- rtic

これらのソフトウェア プロジェクトの ELF/BIN ファイル、暗号化キー、および Bootgen イメージフォーマット (BIF) ファイルは、xapp1225-rtic/completed ディレクトリにあります。ソース ファイルは、hello\_src および rtic\_src ディレクトリにあります。

## はじめに

ランタイム インテグリティ チェッカーは、エンベデッド システムの安全性全般に不可欠です。エンベデッド システムのメモリ変更は、ブート時や動作中に生じます。『Zynq-7000 All Programmable SoC のセキュア ブート』(XAPP1175) [参照 1] で説明されているセキュリティ技術を使用した場合、Zynq-7000 AP SoC デバイスは信頼された状態で起動します。Zynq-7000 AP SoC デバイスは、セキュリティ範囲内で高度な統合を提供します。Zynq-7000 AP SoC デバイスへ接続された外部メモリは、保護されていない状態のままでは攻撃の標的となります。RTIC は、動作中における外部メモリの安全性を保証し、完全なセキュア ブート機能をサポートします。

このアプリケーション ノートは、次のセクションで構成されます。

- 「Zynq-7000 AP SoC デバイスの RSA 認証」では、Zynq-7000 AP SoC デバイスで使用される RSA (Rivest, Shamir, Adleman) 認証について説明しています。Zynq-7000 AP SoC デバイスで RSA を使用する場合は、『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。
- 「RTIC アプリケーション」では、RTIC が使用されるタイミング、およびインテグリティ (完全性) のチェック対象となる情報の種類について説明しています。
- 「RTIC リファレンス デザインの開発」では、RTIC の作成方法を示しています。このセクションでは、ザイリンクスのソフトウェア開発キット (SDK) の使用経験があることを前提に説明しています。
- 「RTIC システム テスト」では、ZC702 評価プラットフォーム上で RTIC を実行し、結果を確認する方法を説明しています。
- 「RTIC を使用する際のセキュリティ考察」では、RTIC を安全に実行する方法を説明しています。

本資料は表記のバージョンの英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。資料によっては英語版の更新に対応していないものがあります。日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

## 必要なハードウェアおよびソフトウェア

Zynq-7000 AP SoC デバイスのアップデートにおけるハードウェア要件は次のとおりです。

- ZC702、ZC706、ZED、または MicroZed 評価ボード
- AC 電源アダプター (12VDC)
- USB ケーブル (Type-A to Micro-B)
- ザイリンクス プラットフォーム ケーブル USB II
- ザイリンクス Vivado Design Suite 2014.3
- ザイリンクス ソフトウェア開発キット 2014.3

## Zynq-7000 AP SoC デバイスの RSA 認証

RSA は、ソフトウェア、データ、またはハードウェア (ビットストリーム) が改ざんされないように、Zynq-7000 AP SoC デバイスで使用される公開キー暗号方式です。RSA は暗号化および認証に非公開キーと公開キーのペアを用います。エンベデッド システムの RSA 暗号化では、非公開キーは使用されません。セキュア ブートおよびランタイム インテグリティ チェック時には、ザイリンクスの RSA ライブラリが使用されます。

RSA 認証では、ファクトリーで非公開キーを使用して SDK イメージライター (Bootgen) がパーティションを符号化します。FSBL (ファースト ステージブートローダー) および U-Boot は、RSA ライブラリ、RSA 公開キー、および署名を使用してパーティションを認証します。セキュア ブート プロセスの詳細は、『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。また、このアプリケーション ノートでは、RSA 非公開キー/公開キーの生成と使用、およびインテグリティ チェックにおける 認証証明 (Authentication Certificate) の使用についても詳しく説明されています。セキュア ブートでの認証方法が RTIC で再び使用されます。

## RTIC アプリケーション

動作中における外部メモリの不正な変更を検出するために、起動後 RTIC が実行されます。これは、FSBL またはスケジューラーで開始でき、システム リセットまたは電源が切断されるまで中断することなく、定期的に行われます。この実行頻度はユーザーが指定できます。最適な形としては、TOCTTOU (Time of Check to Time of Use) 問題に対処するために、実行されるコードに対してインテグリティ テストを実行します。不正アクセスが素早く検出できれば、適切な措置によってダメージが最小限に抑えられます。

RTIC は、ソフトウェア、データ、またはハードウェア (ビットストリーム) のいずれか 1 つのパーティションをチェックします。パーティションは、プレーン テキストや暗号化となります。RTIC は、読み出し専用で静的データを含む外部メモリ (NVM または DDR) のデータをテストします。最も一般的なインテグリティ チェックは、ソフトウェア (コード) に対して行われます。NVM/DDR 内にあるコードが読み出し/書き込み可能で、内容およびその位置に変更がない場合にのみインテグリティ チェックを実行できます。

コンフィギュレーション メモリのビットストリームは、認証できません。これらが NVM または DDR 内にある場合は、認証可能です。このビットストリームはその後、コンフィギュレーション メモリへコピーできます。コンフィギュレーション メモリは、Zynq-7000 AP SoC デバイスのセキュリティ範囲内にあるため、外部メモリよりも外部からの攻撃を受けにくくなります。コンフィギュレーション メモリに対する SEU チェックは可能です。

## RTIC リファレンス デザインの開発

リファレンス デザインでは、rtic プロジェクトが hello パーティションのインテグリティをチェックします。hello パーティションは、DDR 内の 0x20000000 に配置されています。

図 1 に、RTIC リファレンス デザインの構築手順を示します。最初の 2 つの手順 (RSA キーの生成とプログラム) については、『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。次に、fsbl、hello\_world、hello、および rtic ソフトウェアプロジェクトを作成します。SDK では、全プロジェクトのアプリケーションプロジェクト テンプレートが提供されます。hello、hello\_world、rtic の各プロジェクトのテンプレートを変更して、インテグリティ チェック機能を有効にします。

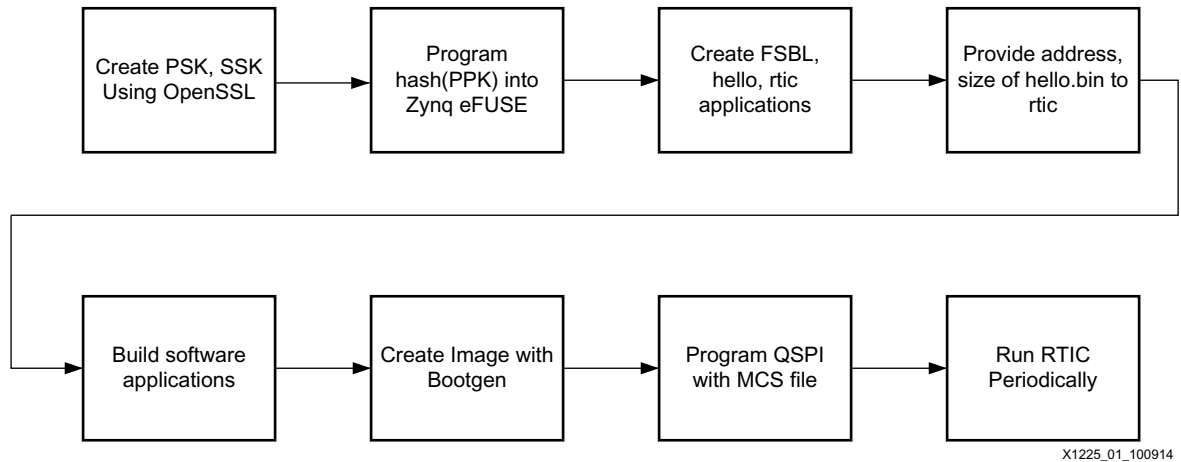


図 1: ランタイム インテグリティ チェックのフロー

リファレンス デザインでは、FSBL が hello\_world、hello、および rtic ソフトウェアプロジェクトをロードします。rtic が定期的に hello のコンテンツをチェックします。

hello のインテグリティ チェックする場合、rtic は hello パーティションのアドレスとサイズ情報を必要とします。Bootgen を使用し、認証証明付きでスタンドアロンの hello.bin パーティションを作成します。パーティションのアドレスとサイズ情報および認証証明は、hello.bin で判断されます。これらのパラメーターが rtic プロジェクトに入力されると、インテグリティ チェックが有効になります。

SDK を使用して、fsbl、hello\_world、hello、および rtic ソフトウェアプロジェクトのテンプレートを作成します。

1. xapp1225-rtic ディレクトリを作成します。
2. 次のコマンドを実行して FSBL ソフトウェアプロジェクトを作成します。
  - a. [File] → [New] → [Application Project] をクリックします。
  - b. [Project Name] に「fsbl」と入力します。
  - c. [Target Hardware] の下にある [Hardware Platform] で [ZC702\_hw\_platform(predefined)] を選択します。
  - d. [Next] をクリックします。
  - e. [Zynq FSBL] を選択します。
  - f. [Finish] をクリックします。
3. [fsbl] を右クリックして、[C/C++ Build] → [Settings] をクリックします。
4. [Symbols] をクリックします。[Defined Symbols] ビューで [+] をクリックします。
5. テキスト ボックスに「RSA Support」と入力します。[OK]、[Apply]、[OK] をクリックします。
6. hello および hello\_world ソフトウェアプロジェクトを作成するには、両プロジェクトの [Template] ダイアログ ボックスで [Hello World] を選択して、fsbl と同様の手順を繰り返します。
7. xapp1225-rtic/hello\_world\_src から src エリアへ helloworld.c および handoff.S をコピーして、hello\_world プロジェクトを変更します。この変更に基づいて、hello\_world 完了後に CPU が rtic ソフトウェアを実行します。
8. rtic ソフトウェア プロジェクトを作成するために、[Templates] ダイアログ ボックスで [RSA Authenticate App] を選択して、fsbl プロジェクトの作成で使った手順を繰り返します。
9. rtic プロジェクトを構築後、[Project Explorer] ペインで [rtic\_bsp] を右クリックして [Board Support Package Settings] をクリックし、[xilrsa] をオンにして [OK] をクリックします。

10. [Finish] をクリックします。

[Project Explorer] ペインで、[rtic] の下の `src` ディレクトリに `rsa_auth_app.c` と `rsa_auth_app.h` ファイルが配置されています。

図 2 の [Project Explorer] ペインには、`fsbl`、`hello_world`、`hello`、および `rtic` ソフトウェアプロジェクトが表示されています。

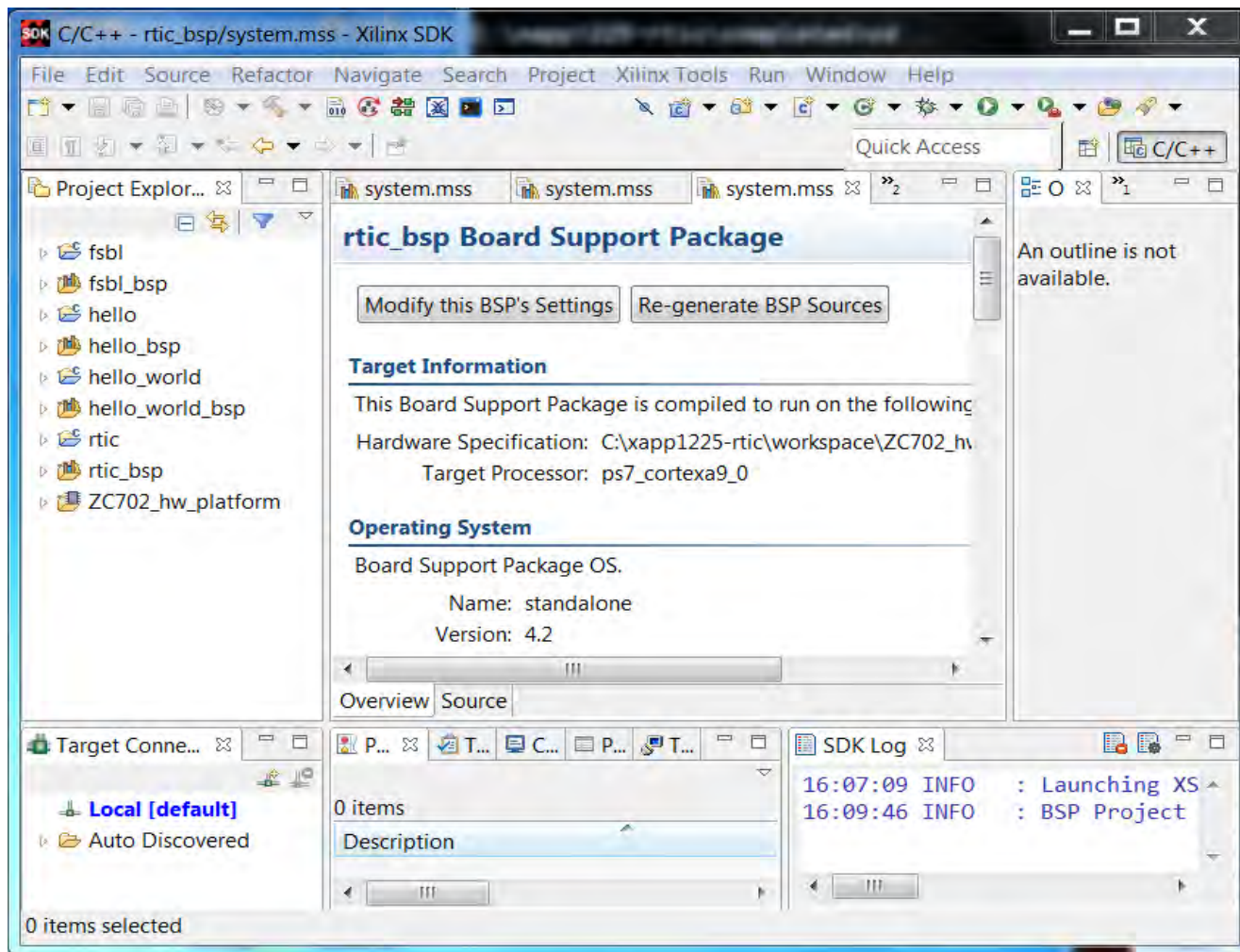


図 2 : FSBL、Hello、Hello\_World、および RTIC プロジェクトを含む SDK

次の手順に従って、`hello` および `rtic` プロジェクトを変更します。

このプロジェクトでは、SDK ツールの `Bootgen` を 2 回使用します。ブート ステージ用のイメージが作成される場合、`Bootgen` に入力される `Bootgen` イメージフォーマット (BIF) には、`FSBL` ビットストリームとすべてのソフトウェアパーティションが含まれます。`Bootgen` は、`BIN` 形式または `MCS` 形式で 1 つのイメージファイルを生成します。

`hello` パーティションの `RSA` 認証時に、再度 `Bootgen` を使用して、`hello` パーティションのみを含むイメージを生成します。前述のように、これは、`hello.bin` から位置およびサイズのパラメーターを抽出するためです。

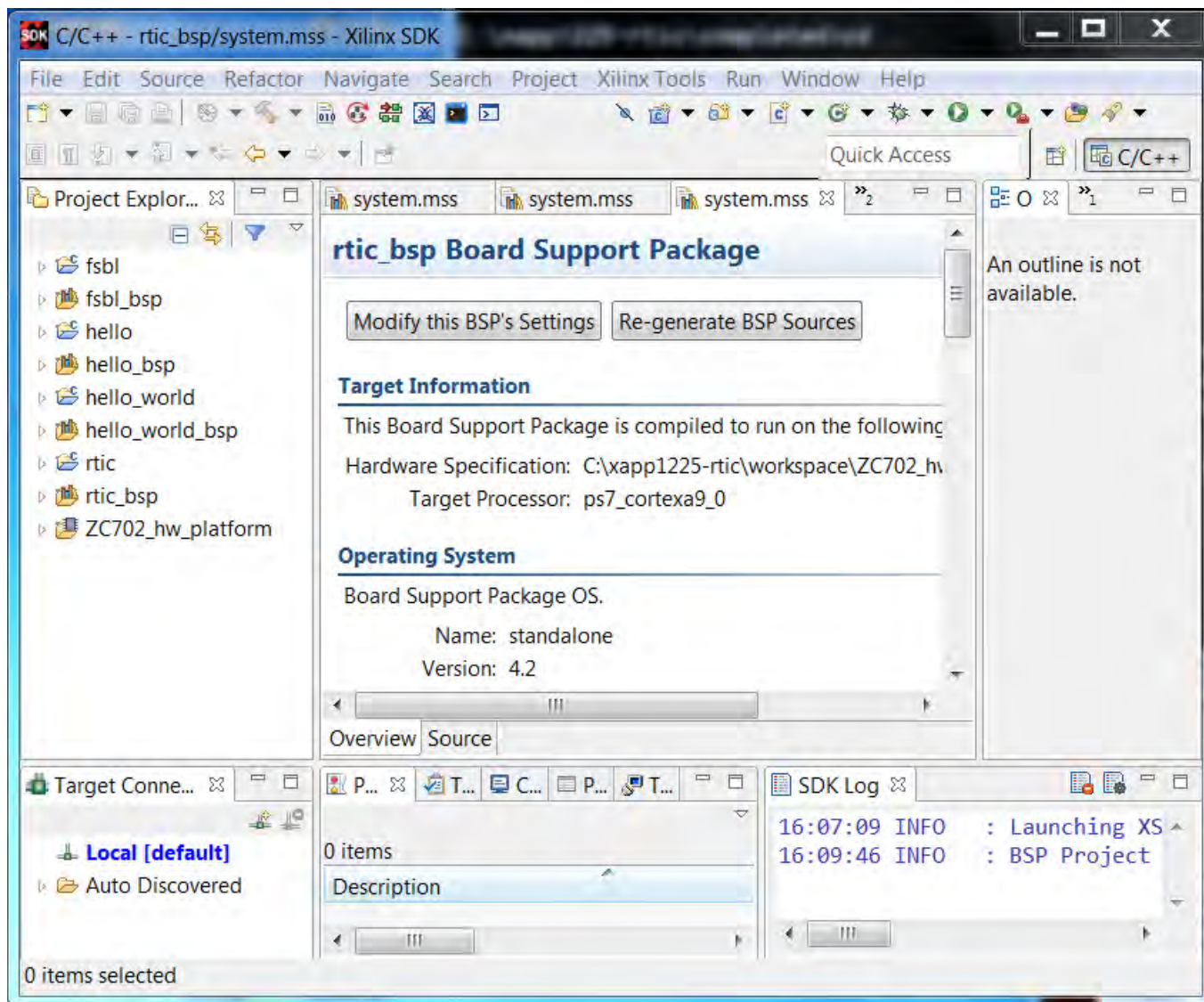
1. SDK で [Xilinx Tools] → [Create Zynq Boot Image] をクリックします。
2. [Create Zynq Boot Image] ダイアログ ボックスで、[BIF file path] の [Browse] をクリックして `xapp1225-rtic` の `hello.bif` を選択します。
3. [Use Authentication] をオンにします。

4. [Authentication Keys] セクションで、PSK には `psk.pem` を選択します。サンプルの PSK (Sample Primary Secret Key)、SSK (Secondary Secret Key)、および PPK (Primary Public Key) のハッシュは、`xapp1225-rtic/completed` ディレクトリに提供されています。SSK には `ssk.pem` を選択します。
5. [Add] をクリックします。
6. [Add partition] ダイアログ ボックスで、[File path] の [Browse] をクリックして `xapp1225-rtic/hello/Debug/hello.elf` を選択します。
7. [Partition type] に [datafile] を選択します。
8. [Authentication] に [rsa] を選択します。
9. [Output path] の [Browse] をクリックして、`xapp1225-rtic/hello` の `hello.bin` を選択します。
10. [Create Image] をクリックします。
11. 図 4 に示すように、TextPad などのテキスト エディターで `hello.bin` を開きます。
12. アドレス C80 から開始する 3 つの値を検索します。

Partition Start Address - C0 05 00 00 (C90 行の 2 つ目のワード)

Partition Size - 03 20 00 00 (C80 行の 2 つ目のワード)

Authentication Certificate (AC) Start Address - D0 25 00 00 (CA0 行の 3 つ目のワード)



X1225\_02\_102214

図 3: パーティションの開始アドレス、サイズ、および AC 開始アドレスを示す

13. 10 進乗算を使用してサイズの値を計算します。この動作にはバイト スワップが必要です。
- バイト スワップ後、値 `0xC0050000` が `0x5C0` に変換されます。1 ワードは 4 バイトとなるため、`0x1700` にするには 4 で乗算します。
  - 例として、Python を使用して開始アドレスを計算するために、Python を起動して次のコマンドを入力します。

```
int("5C0",16)
1472
1472 * 4
5888
hex(5888)
0x1700
```

計算された値は、次のとおりです。

```
PARTITION START ADDRESS - 0x1700
PARTITION SIZE - 0x8700
AUTHENTICATION CERTIFICATE START ADDRESS - 0x9740
```

14. 図 4 に示すように、rsa\_auth\_app.h を開いて、先ほど計算された値を #define に与えます。認証テストが実行される時に、このイメージが 0x2000000 のオフセットにロードされます。APPLICATION START ADDRESS および CERTIFICATE START ADDRESS にベース アドレス 0x20000000 を追加します。

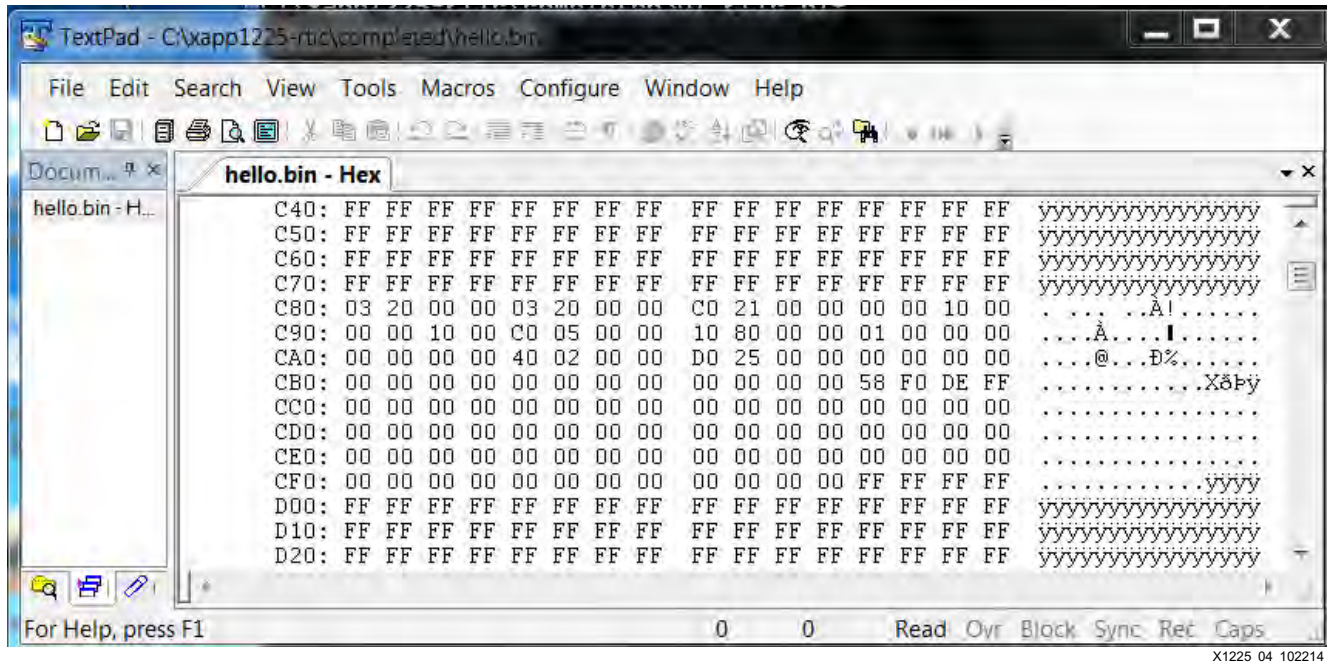
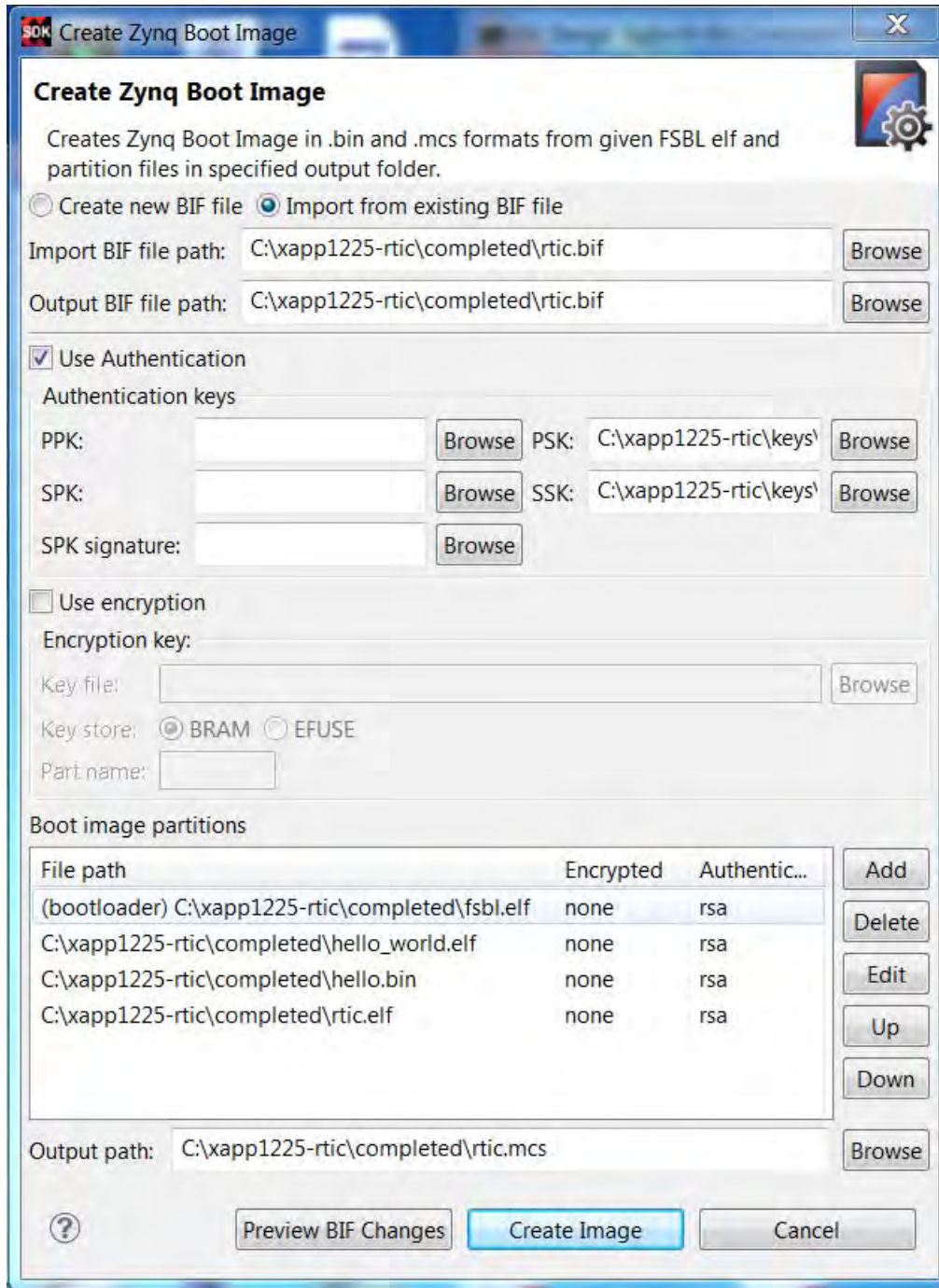


図 4 : rsa\_auth\_app.h のパラメーターをアップデート

15. rtic プロジェクトをハイライトして右クリックし、[Build Project] を選択します。
16. Bootgen を使用して xapp1225-rtic プロジェクトのイメージを作成するために、[Xilinx Tools] → [Create Zynq Boot Image] をクリックします。
17. 4つのパーティションを追加するために、hello.bin パーティションで使用した手順に従います。

または、[Import from existing BIF file] をオンにし、[Browse] をクリックして xapp1225-rtic/completed/rtic.bif を選択します。

18. 図 5 に示す Bootgen の詳細設定を使用して、[Create Image] をクリックして rtic.mcs を生成します。



X1225\_05\_102214

図 5 : Bootgen を使用して RTIC イメージを作成

## RTIC システム テスト

このセクションでは、xapp1225-rtic リファレンス デザインのシステム テストを実行する 2 つの方法を示します。最もシンプルな 1 つ目のテストでは、XMD コマンドを使用します。2 つ目のテストでは、Quad SPI フラッシュ メモリからリファレンス デザインを実行します。



1. システム テストの結果を表示するために、TeraTerm などの通信端末のボーレート 115,200 に設定します。
2. シンプルなテストを実行する場合は、XMD を起動して次のコマンドを実行します。

```
connect arm hw
dow fsbl.elf
run
stop
dow -data hello.bin 0x20000000
dow rtic.elf
con
```
3. Quad SPI フラッシュ メモリからリファレンス デザインを実行する場合は、ZC702 のモード選択ピンを JTAG モードに設定します。
4. SDK で [Xilinx Tools] → [Program Flash] をクリックします。
5. [Image File] の [Browse] をクリックして、xapp1225-rtic/completed/rtic.mcs を選択します。
6. [Offset] に「0x0」と入力します。
7. [Program] をクリックします。
8. ZC702 ボードのモード選択ピンを Quad SPI フラッシュ メモリ モードに変更します。
9. ボードの電源を一旦切って再度入れます。

図 6 に、システム テストの予想出力の詳細を示します。RTIC は定期的に行われます。システム テストでは、5 秒間隔で 3 回実行されます。

完全な rtic.log ファイルは、xapp1225-rtic/completed ディレクトリにあります。FSBL は、FSBL\_DEBUG\_INFO の Build 設定を使用してコンパイルされるため、パーティションのロードで詳細情報が提供されます。ログ ファイルには、各パーティションのアドレス ロケーション、サイズ、および認証情報が含まれます。この後、システム テストが 3 回実行されます。



```

COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help
PCAP:Clear done
PCAP register dump:
PCAP CTRL 0xF8007000: 0x0C00E07F
PCAP LOCK 0xF8007004: 0x0000001A
PCAP CONFIG 0xF8007008: 0x00000508
PCAP ISR 0xF800700C: 0x00033000
PCAP IMR 0xF8007010: 0xFFFFFFFF
PCAP STATUS 0xF8007014: 0x50000A30
PCAP DMA SRC ADDR 0xF8007018: 0xFC030301
PCAP DMA DEST ADDR 0xF800701C: 0x00300001
PCAP DMA SRC LEN 0xF8007020: 0x000031C0
PCAP DMA DEST LEN 0xF8007024: 0x000031C0
PCAP ROM SHADOW CTRL 0xF8007028: 0xFFFFFFFF
PCAP MBOOT 0xF800702C: 0x0000C000
PCAP SW ID 0xF8007030: 0x00000000
PCAP UNLOCK 0xF8007034: 0x757BDF0D
PCAP MCTRL 0xF8007080: 0x34800110

DMA Done !
Authentication Done
Handoff Address: 0x00100000
In FshlHookBeforeHandoff function
SUCCESSFUL_HANDOFF
FSBL Status = 0x1
Hello World

Start Runtime Integrity Check of hello.bin
RTIC of hello.bin: Passed

Delay 5 s and run RTIC of hello.bin
RTIC of hello.bin: Passed

Delay 5 s and run RTIC of hello.bin
RTIC of hello.bin: Passed

```

X1225\_06\_100914

図 6 : hello.bin の RTIC

## RTIC を使用する際のセキュリティ考察

RTIC を定期的に行う方法の基本原則を、リファレンス デザインを用いて説明しました。実際のシステムでは、この周期が永久的に続けられる必要があり、その他のセキュリティ対策を講じる必要もあります。対策を講じない場合は、RTIC をシステム メモリ内 (NVM または DDR) に置く必要があります。RTIC はその実行が促されるときに、認証される必要があります。

RTIC は、OCM から最も高い特権モードで実行する必要があります。Trustzone を使用する場合、RTIC はセキュア領域内に配置する必要があります。

## まとめ

このアプリケーション ノートでは、システム メモリのコードおよび静的データを含むセクションが改ざんされていないかをチェックする方法について説明しました。ランタイム インテグリティ チェッカーは、動作中に生じる可能性が高いシステム メモリへの攻撃を検出する効果的な手段を提供します。

# 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](#) 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 1 に、リファレンス デザインの詳細を示します。

表 1: リファレンス デザインの詳細

パラメーター	説明
<b>全般</b>	
開発者	Lester Sanders
ターゲット デバイス	Zynq-7000 All Programmable SoC
ソース コードの提供	あり
ソース コードの形式	VHDL、Verilog
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 デザインあるい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
<b>シミュレーション</b>	
論理シミュレーションの実施	なし
タイミングシミュレーションの実施	N/A
論理シミュレーションおよびタイミング シミュレーションでのテストベンチの利用	N/A
テストベンチの形式	Verilog
使用したシミュレータ/バージョン	Vivado シミュレータ
SPICE/IBIS シミュレーションの実施	N/A
<b>インプリメンテーション</b>	
使用した合成ツール/バージョン	Vivado 合成
使用したインプリメンテーション ツール/バージョン	Vivado インプリメンテーション
スタティック タイミング解析の実施	あり
<b>ハードウェア検証</b>	
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	ZC702、ZC706、または Zed 評価ボード

## 参考資料

- 『Zynq-7000 All Programmable SoC のセキュア ブート』([XAPP1175](#))
- 『Zynq-7000 XC7020 AP SoC 向け ZC802 評価ボード』([UG850](#))
- 『Zynq-7000 All Programmable SoC ソフトウェア開発者向けガイド』([UG821](#))
- 『Zynq-7000 All Programmable SoC システムのセキュア アップデート』([XAPP1224](#))

## 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
2014年10月23日	1.0	初版

## 法的通知

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.

© Copyright 2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

### Automotive Applications Disclaimer

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO: (I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY. CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.

© Copyright 2014 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

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](mailto: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。