参考文献

User's Guide

如何借助 Lauterbach 在 Jacinto7 具有信息安全机制的器件上解锁 JTAG 并调试硬件安全模块 (HSM)

摘要

本用户指南介绍使用 Lauterbach 的 TRACE32™ 调试程序解锁 JTAG 并访问硬件安全模块 (HSM) 的过程。本指南中提供的说明适用于包含安全管理子系统第 2 代架构的所有 Jacinto 7 具有信息安全机制 (HS-SE) 的器件。

内容

插图清单

表格清单

商标

1 简介

Jacinto7 安全管理子系统 (SMS) 由两个 Arm® Cortex®-M4F 内核组成。主要内核称为 TI 基础安全模块 (TIFS)，它执行 TI 基础安全功能。此外，辅助内核称为硬件安全模块 (HSM)，用于运行客户或第三方安全功能。本用户指南介绍借助 Lauterbach 的 TRACE32 在 Jacinto7 具有信息安全机制的器件上解锁 JTAG 并访问 HSM（Arm Cortex - M4F 安全内核 1）的过程。

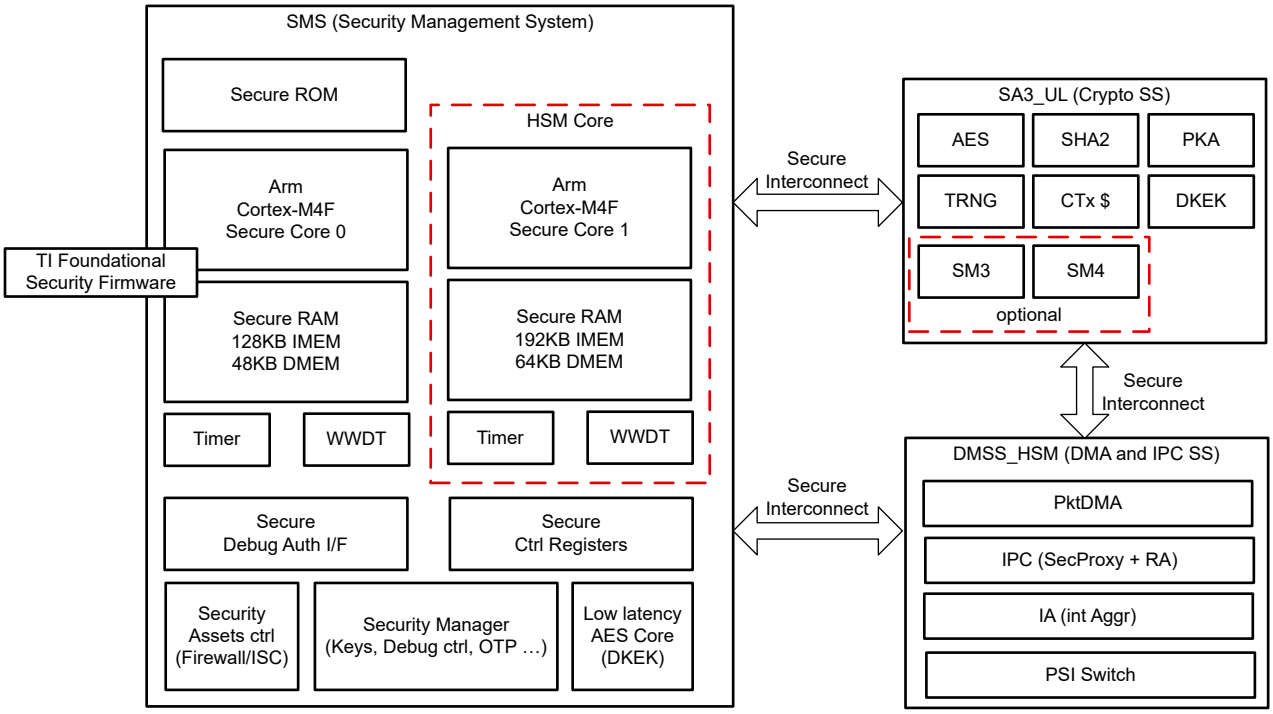

图 1-1. 安全管理子系统第 2 代架构

1.1 使用 Jacinto7 启用安全功能的器件解锁 JTAG

一旦 Jacinto7 器件转换为高安全性 - 强制安全型器件 (HS-SE)，就会强制执行所有安全策略，例如安全启动、启用防火墙和锁定 JTAG。当在 HS-SE 器件上锁定了 JTAG 访问时，有三种方法可以解锁 JTAG 以进行调试：

- 通过二级引导加载程序 (SBL) x509 证书的调试扩展进行 JTAG 解锁
  - TISCI 用户指南 — 安全 X509 文档 — 系统固件调试扩展
- 通过 TI 基础安全 (TIFS) 固件 TISCI API 进行 JTAG 解锁
  - TISCI 用户指南 — 运行时调试 TISCI 说明
- 借助 Lauterbach TRACE32 或 TI Code Composer Studio 通过 JTAG 证书进行 JTAG 解锁
  - TISCI 用户指南 — 基于 JTAG 的安全 AP 命令界面

备注

Jacinto7 ROM 加载程序不支持针对 HSM 内核解锁 JTAG 访问。这意味着不能按照上面列出第一项所述使用二级引导加载程序 x509 证书的调试扩展通过 HS-SE 器件解锁 JTAG。本用户指南明确介绍如何借助 Lauterbach TRACE32 通过 JTAG 证书解锁 JTAG（上面列出的第三项）。

2 使用 TRACE32 针对 HSM 内核解锁 JTAG 的步骤

2.1 修改 SCI 客户端默认安全电路板配置

2.1.1 PROCESSOR-SDK-RTOS

在 PROCESSOR-SDK-RTOS 目录 *<pdk_path/ti/drv/sciclient/soc/V4/* 中，有一个名为 *“sciclient_defaultBoardcfg_security.c”* 的文件。此文件包含两个结构元素，必须正确配置它们的值以允许运行时 JTAG 解锁发生。

表 2-1. 用于 JTAG 解锁的安全电路板配置元素

元素	类型	说明
allow_jtag_unlock	uint8_t	必须设置为 0x5A 以允许运行时 JTAG 解锁发生
allow_wildcard_unlock	uint8_t	设置为 0 以在 JTAG 解锁可以发生之前强制 UID 匹配。因此，x509 证书必须包含待解锁器件的 UID。 设置为 0x5A 可在 JTAG 解锁之前绕过 UID 匹配。

```
    /* Secure JTAG Unlock Configuration */
    .sec_dbg_config = {
        .subhdr = {
            .magic = TISCI_BOARDCFG_SEC_DBG_CTRL_MAGIC_NUM,
            .size = sizeof(struct tisci_boardcfg_secure_debug_config),
        },
        .allow_jtag_unlock = 0x5A,
        .allow_wildcard_unlock = 0x5A,
        .min_cert_rev = 0x0,
        .jtag_unlock_hosts = {0, 0, 0, 0},
    },
};
```

图 2-1. 安全调试结构 — 示例配置

在 图 2-1 中，设置 "*.allow_jtag_unlock = 0x5A*" 以允许运行时 JTAG 解锁发生，并设置 "*.allow_wildcard_unlock = 0x5A*" 以在 JTAG 解锁之前绕过任何 UID 检查。对于生产用例，建议设置 "*.allow_wildcard_unlock = 0x0*" 以强制执行 UID 检查。强制执行 UID 检查可以防止解锁具有相同调试证书的不同器件的 JTAG。

有关 SCI 客户端安全电路板配置中所有安全调试解锁元素的详细说明，请参阅 TISCI 用户指南 — 安全调试解锁。

2.1.2 PROCESSOR-SDK-LINUX

在 PROCESSOR-SDK-LINUX 版本 9.0 或更低版本中，可以在 *<linux sdk>/k3-image-gen-xxxx.xx/soc/<soc_name>/evm/sec-cfg.c* 下找到安全电路板配置文件。

图 2-2. (SPL < 9.0 SDK) 安全调试结构 — 示例配置

在 PROCESSOR-SDK-LINUX 版本 9.0 或更高版本中，可以在 <u-boot-xxxx.xx+x>/board/ti/<soc_name>/sec-cfg.yaml 下找到安全电路板配置文件。

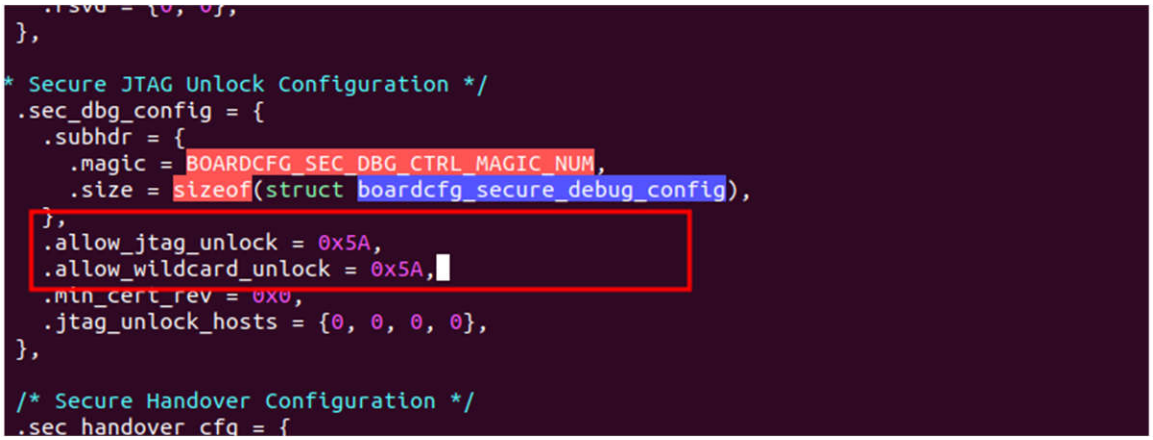

图 2-3. (SPL 9.0+ SDK) 安全调试结构 — 示例配置

2.2 构建 SCI 客户端安全电路板配置

修改安全电路板配置结构后，下一步是构建 SCI 客户端电路板配置。如果您选择使用二级引导加载程序 (SBL) 进行引导，请继续阅读介绍如何执行此构建的 *“PROCESSOR-SDK-RTOS”* 一节。另外，如果您选择使用二级程序加载程序 (SPL) 进行引导，请继续阅读 *“PROCESSOR-SDK-LINUX”* 一节。

2.2.1 PROCESSOR-SDK-RTOS

要在 PROCESSOR-SDK-RTOS 中构建 SCI 客户端电路板配置，请导航到 <pdk_path>/packages/ti/build 目录并打开命令提示符。接着，应执行以下 make 命令以独立构建 SCI 客户端电路板配置：

```
make sciclient_boardcfg_hs BOARD=j721s2_evm CORE=mcu1_0 OS=linux
```

上面的 make 命令将使用位于 PROCESSOR-SDK-RTOS 中的默认 MEK 和 MPK 开发密钥对 SCI 客户端安全板配置进行编译、加密和签名。该构建命令的最终输出将是一个位于 <pdk_path>/packages/ti/drv/sciclient/soc/V4/ 目录中的 C 数组。

2.2.2 PROCESSOR-SDK-LINUX

为了使用 SPL 构建 SCI 客户端电路板配置，请导航到 PROCESSOR-SDK-LINUX 中的 U-boot 树的根目录。U-Boot 树的根目录是顶层目录，可以通过查找 "MAINTAINTERS" 文件来识别。对于 *PROCESSOR-SDK-LINUX 版本 8.6 及更低版本*，应在命令提示符中执行以下命令，以重新构建 SPL 和 tiboot3.bin 映像：

```
export TI_SECURE_DEV_PKG=<path-to-board-support-directory>/core-secdev-k3

export PATH=$HOME/gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf/bin:$PATH

export PATH=$HOME/gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu/bin:$PATH

make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabihf- j721s2_hs_evm_r5_defconfig O=<output
directory>/r5

make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabihf- O=<output directory>/r5

cd ../k3-image-gen-<version>

make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabihf- SOC=j721s2 HS=1 SW_REV=1 SBL=<output
directory>/r5/spl/u-boot-spl.bin SYSFW_HS_PATH=<path to tisdk>/board-support/prebuilt-images/ti-fs-
firmware-j721s2-hs-enc.bin SYSFW_HS_INNER_CERT_PATH=<path to tisdk>/board-support/prebuilt-
images/ti-fs-firmware-j721s2-hs-cert.bin
```

从 *PROCESSOR-SDK-LINUX 版本 9.0* 及更高版本开始，引导加载程序映像的编译将不再需要 GP 和 HS 器件的不同 defconfig。因此，相同的构建命令将生成 GP、HS-SE 和 HS-FS 器件的映像。构建 SPL 和 tiboot3.bin 映像时，应在命令提示符下执行以下命令。

```
export PATH=$HOME/gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabihf/bin:$PATH

export PATH=$HOME/gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu/bin:$PATH

make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabihf- j721s2_evm_r5_defconfig O=<output
directory>/r5

make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabihf- O=<output directory>/r5 BINMAN_INDIRS=<path
to tisdk>/board-support/prebuilt-images
```

最后，可将 tiboot3-j721s2-hs-evm.bin 文件从 <output directory/r5> 目录复制到 SD 卡的引导分区中或编程到受支持的非易失性引导介质中。

```
cp <output directory>/r5/ tiboot3-j721s2-hs-evm.bin /media/<xyz>/boot/tiboot3.bin
```

备注

下面的链接包含安全电路板配置签名过程的详细说明：TISCI 用户指南 - 对 HS 器件上的电路板配置进行签名。

2.3 修改二级引导加载程序的 x509 证书

JTAG 访问通过位于 x509 证书中的调试扩展字段控制。默认情况下，为实现轻松的开箱即用用户体验，PROCESSOR-SDK-RTOS 可通过二级引导加载程序 (SBL) x509 证书在 HS-SE 器件上启用通过 JTAG 调试。为了在 HS-SE 器件上禁用或更改 JTAG 访问级别，用户必须在构建 SBL 时手动更改在 PROCESSOR-SDK-RTOS x509 签名脚本和模板中使用的调试扩展。ROM 加载程序不支持解锁 JTAG 以调试 HSM，因此用户需要删除 SBL x509 证书中的调试扩展。下面的步骤介绍了在 Windows 或 Ubuntu 环境中执行构建时，如何在 PROCESSOR-SDK-RTOS 中删除 SBL 的调试扩展。

2.3.1 Windows 构建环境

如果用户正在 Windows 环境中构建 SBL，则必须导航到位于 *<pdk_path>/packages/ti/build/makerules* 目录中的 “*x509template.txt*” 文件，然后手动删除调试扩展字段（请参阅图 2-4）。

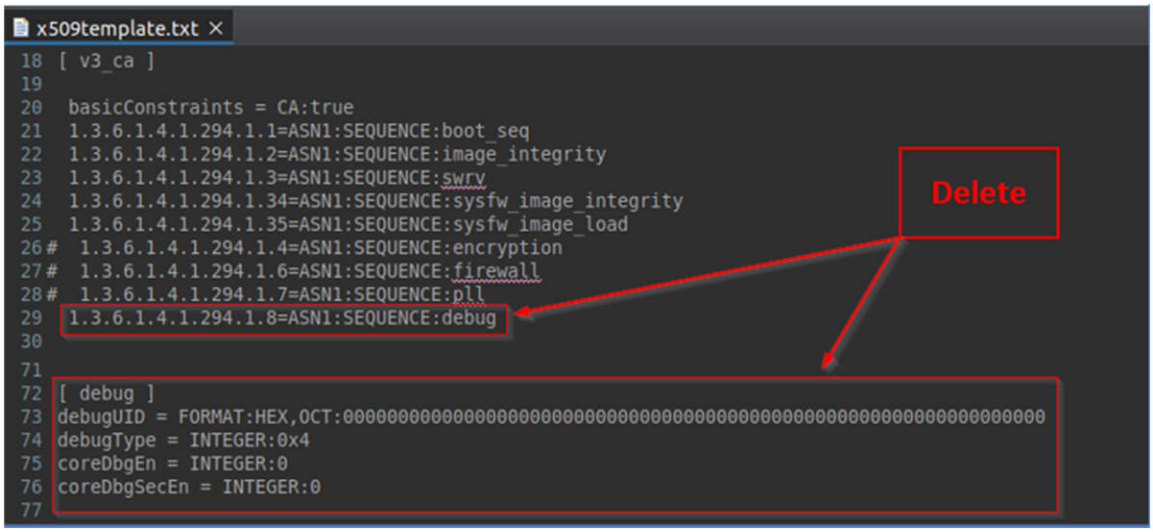

图 2-4. Windows 构建 — x509 模板

2.3.2 Ubuntu 构建环境

如果用户要在 Ubuntu 环境中构建 SBL，他们必须导航到 *<pdk_path>/packages/ti/build/makerules* 目录并手动编辑 “*x509CertificateGen.sh*” 脚本，从而避免将调试扩展字段插入 SBL 的 x509 证书中。"*x509CertificateGen.sh*" 脚本在 make 文件构建过程中自动调用，并包含插入到 SBL 的 x509 证书中的字段。为了查看并防止在构建过程结束后删除本地 x509 模板，请导航到 "*x509CertificateGen.sh*" 脚本的底部，并注释掉以下脚本删除语句（请参阅图 2-5）。这使您可以验证对 "*x509CertificateGen.sh*" 所做的修改是否正确删除了调试扩展。

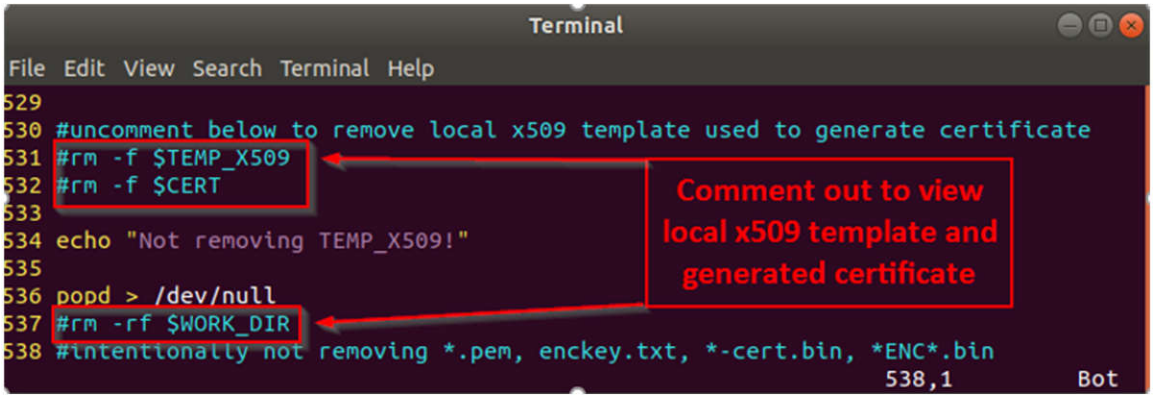

图 2-5. Ubuntu 构建 — x509 证书生成脚本

2.4 构建二级引导加载程序

确认 SBL 的 x509 证书不再包含用于 JTAG 解锁的调试扩展后，下一步是在 PROCESSOR-SDK-RTOS 中构建 SBL。为此，请导航到 *<pdk_path>/packages/ti/build* 目录并打开命令提示符。接着，应按顺序执行以下 make 命令且不出现任何构建错误：

```
make sciclient_boardcfg_hs BOARD=j721s2_evm CORE=mcu1_0 OS=linux

make pdk_libs BOARD=j721s2_evm CORE=mcu1_0 OS=linux

make sbl_mmcsd_img_hs BOARD=j721s2_evm CORE=mcu1_0 OS=linux

cp packages/ti/boot/sbl/binary/j721s2_evm_hs/mmcsd/bin/sbl_mmcsd_img_mcu1_0_release.tiimage /media/
<xyz>/boot/tiboot3.bin
```

备注

用于构建 sbl_*_img_hs 的 make 命令可能会发生变化，具体取决于您使用的引导介质。上面的 SBL make 命令示例是使用 MMC SD 卡作为引导介质，并使用 Ubuntu 作为构建环境。

2.5 验证二级引导加载程序和 TIFS 正在执行

由于 ROM 加载程序无法针对 HSM 解锁 JTAG，因此需要确认 SBL 成功引导并且 TI 基础安全 (TIFS) 固件正在运行。为了验证这些内容，请将新构建的 SBL 映像和 TIFS 二进制文件复制或刷写到非易失性引导介质中。接下来，连接到电路板上的 MCU UART 端口，让器件上电，并通过终端窗口验证 TIFS 正在成功运行（请参阅图 2-6）。

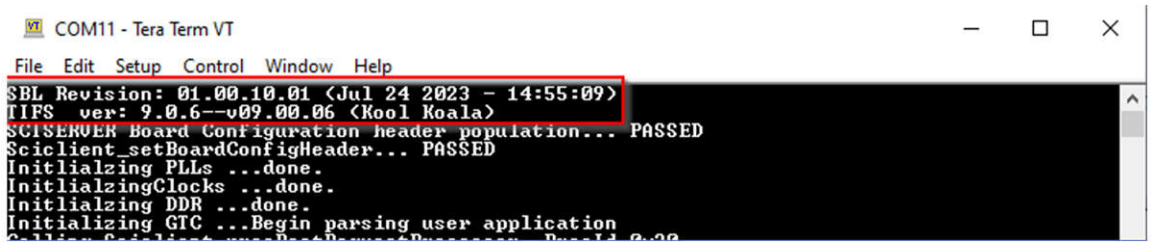

图 2-6. 验证 SBL 和 TIFS 引导

2.6 创建带调试扩展的可下载 x509 证书

现在，有必要创建一个包含已适当配置的调试扩展的可下载 x509 证书。x509 配置模板可在线下载，相应的超链接在本节末尾提供。x509 配置模板可通过下面的链接下载：TISCI 用户指南 - X509 配置模板。

为了便于参考，明确配置了 x509 证书的字段来解锁下面的 HSM：

```
[ req ]
distinguished_name    = req_distinguished_name
x509_extensions       = v3_ca
prompt                = no
dirstring_type        = nobmp

[ req_distinguished_name ]
C                     = US
ST                    = SC
L                     = Dallas
O                     = Texas Instruments., Inc.
OU                    = PBU
CN                    = Albert
emailAddress          = Albert@ti.com

[ v3_ca ]
basicConstraints = CA:true
1.3.6.1.4.1.294.1.3=ASN1:SEQUENCE:swrv
1.3.6.1.4.1.294.1.8=ASN1:SEQUENCE:debug

[ swrv ]
swrv= INTEGER:0
```

```
[ debug ]
debugUID  = FORMAT:HEX,OCT:0000
debugType = INTEGER:5
coreDbgEn = INTEGER:0x010206070809
coreDbgSecEn = INTEGER:0x202180
```

如上所述创建含有用于 JTAG 解锁的相应 x509 字段的 "*cert.txt*" 文件后，现在需要执行下面的 OpenSSL 签名脚本：

```
openssl req -new -x509 -key  k3_dev_mpk.pem -nodes -outform der -out cert.bin -config cert.txt
-sha512
```

备注

使用 OpenSSL 对 x509 证书进行签名时，必须使用适用于 J7 HS-DK 器件的“k3_dev_mpk.pem”密钥。“k3_dev_mpk.pem”密钥位于以下目录中：*<pdk_path>/packages/ti/build/makerules*。

2.7 执行 TRACE32 解锁脚本

成功创建 x509 证书二进制文件并使用 OpenSSL 签名后，即可使用 TRACE32 脚本执行 HSM JTAG 解锁过程。首先，请联系您当地的 TI FAE 以获取定制的 Jacinto7 TRACE32 脚本包。获取 TI 脚本包后，导航到以下文件夹位置 *~/t32/cmm-ti/cmm-dra/cmm-tda4v_j721s2/x_gel_to_cmm/* 并设置位于 *“J721S2_secure_unlock.cmm”* 脚本中的以下参数。

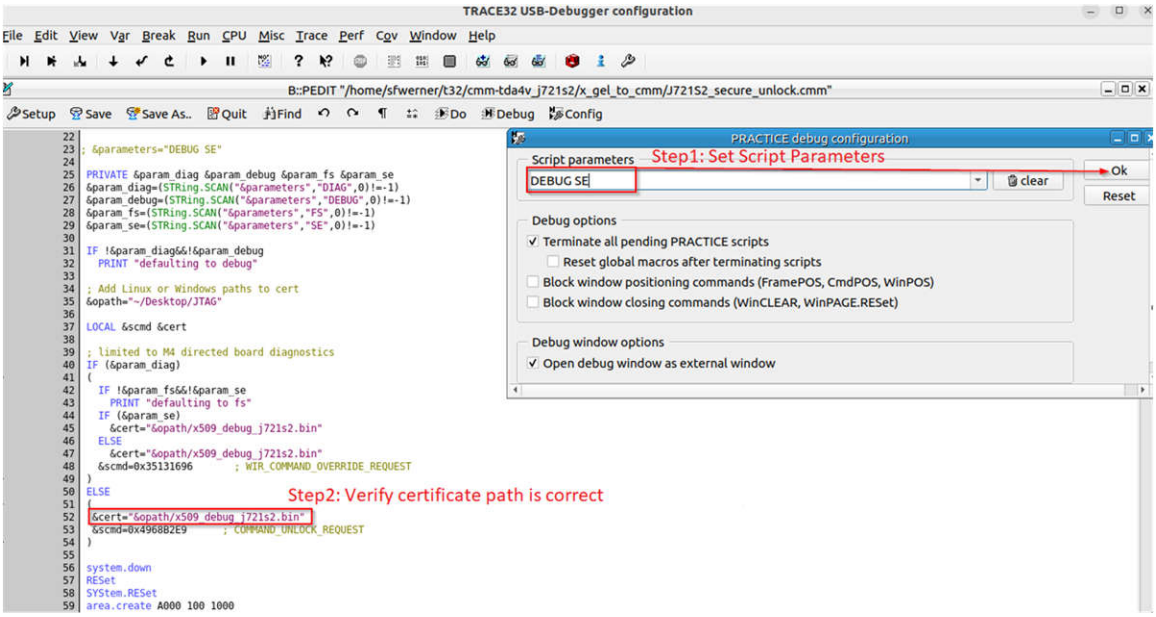

图 2-7. TRACE32 — 安全 JTAG 解锁脚本

修改 *“J721S2_secure_unlock.cmm”* 脚本后，保存更改，让 Jacinto7 器件上电，然后使用 "Do" 命令通过 TRACE32 执行该脚本。成功执行脚本和 JTAG 解锁序列后，您应该会在 TRACE32 区域窗口中看到以下字符串：***“unlock response was : 0xDEAD3A17”***（请参阅图 2-8）。

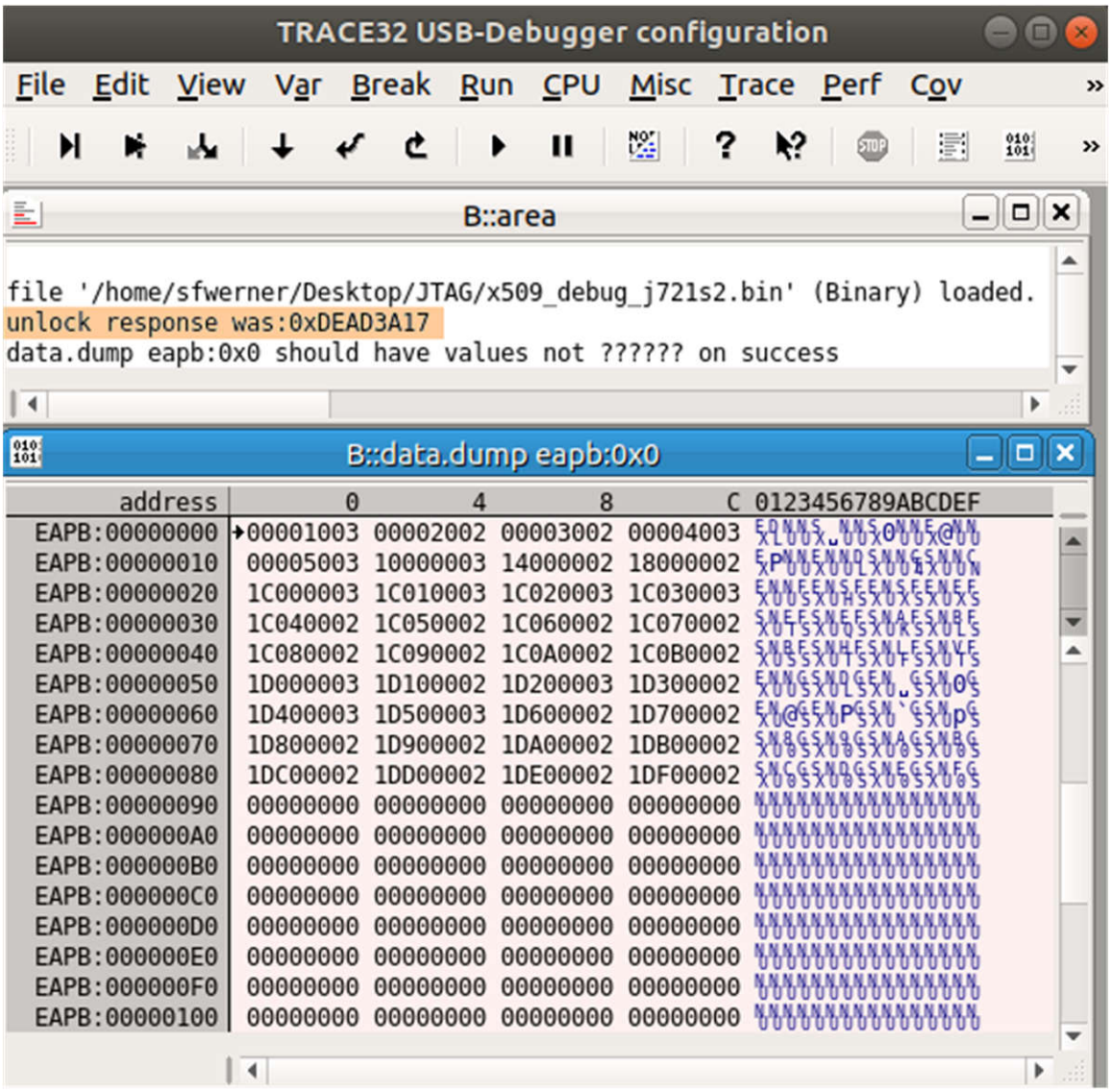

图 2-8. TRACE32 — 成功的安全 JTAG 解锁响应

2.8 使用 TRACE32 连接 HSM 内核

通过 TRACE32 脚本“*J721S2_secure_unlock.cmm*”针对 HSM 成功执行和解锁 JTAG 访问后，可以在 TRACE32 中连接并加载 HSM 符号信息以进行调试。请导航到以下文件夹位置 *~/t32/cmm-ti/cmm-dra/cmm-tda4v_j721s2/x_gel_to_cmm/* 并编辑位于“*_J721S2_m4.cmm*”脚本中的以下脚本参数，以便连接到 HSM (Cortex M4F)。

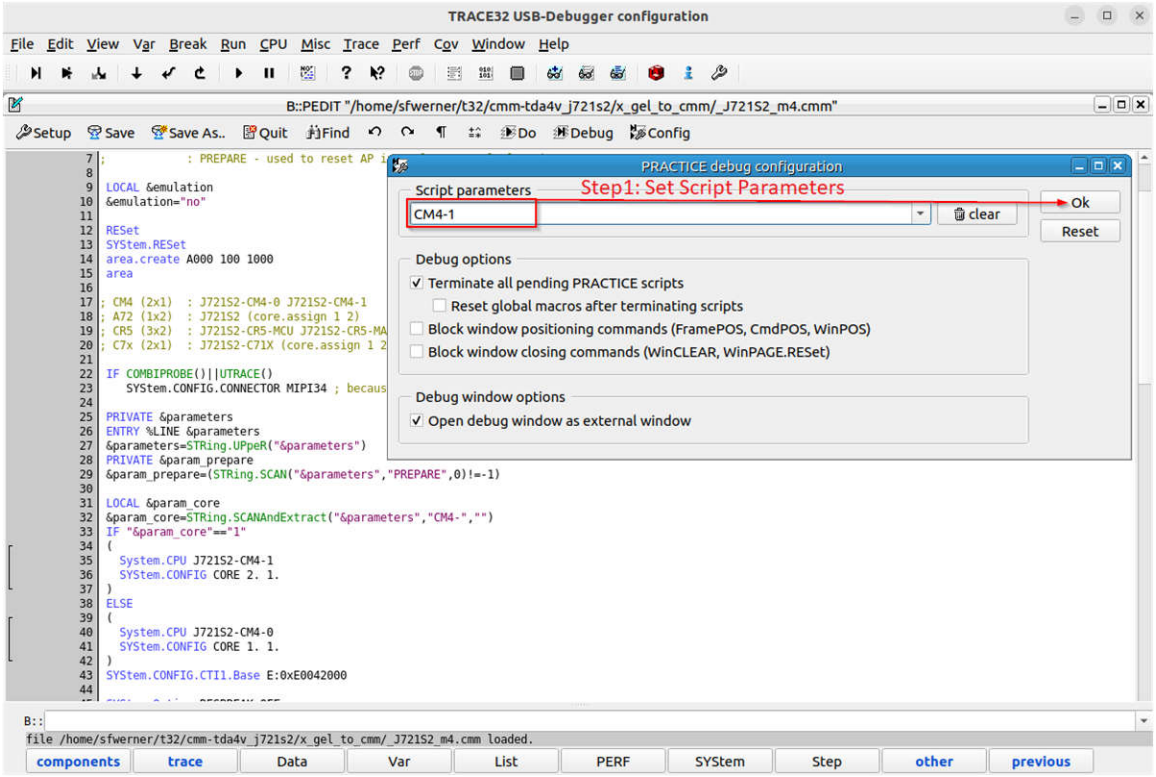

图 2-9. TRACE32 - HSM 连接脚本

修改“_J721S2_m4.cmm”脚本并保存该文件后，请再次点击“Do”按钮以在 TRACE32 中执行该脚本。最后，您此时应已连接到 HSM 并能够在 TRACE32 中看到调试窗口，如图 2-10 中所示。

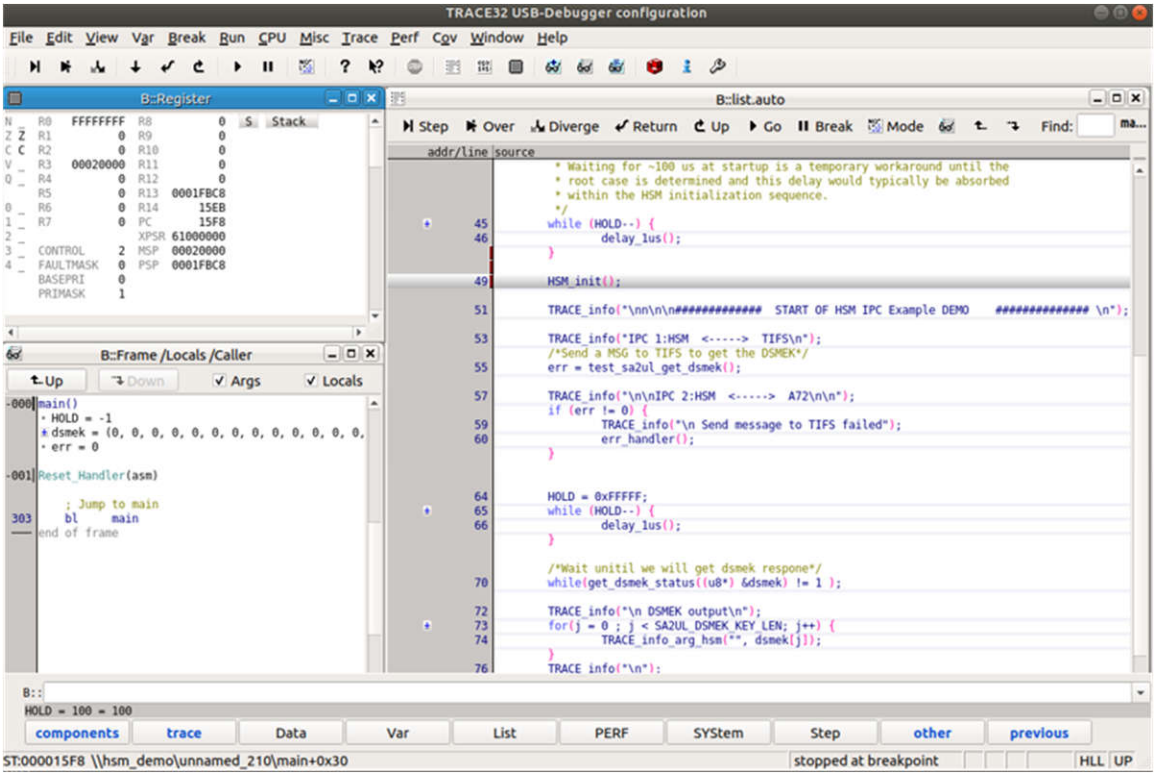

图 2-10. TRACE32 - HSM 调试

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265