Jacinto™ 7 家族 HS 芯片中的 JTAG 加解锁控制

王力（Neo Wang）李彪（Biao Li） ***Central FAE***

摘要

Jacinto™ 7 家族处理器是 TI 公司基于 Keystone 架构推出的最新一代汽车处理器，主要致力于辅助驾驶系统（ADAS），自动驾驶（AD）以及网关（Gateway）等领域的芯片解决方案。

其中 TDA4x 系列汽车处理器基于片上多核异构芯片架构，16nm 系统工艺不仅为芯片带来了高系统集成度，而且大幅降低了多功能高级汽车平台设计所需的外设复杂度以及成本。性能上，在提供高达 ASIL-D 的系统功能安全等级支持的同时，以领先于市场的性能/功耗比为深度学习/视觉加速提供了卓越的处理能力。

而 DRA82x 系列处理器，凭借其片内集成的高性能 ARM 处理器以及内置的 9 端口的以太网交换机，能够做到基于硬件实现的高效率，低时延的网关数据交换以及转发。

而在上述 Jacinto™ 7 家族处理器中，客户可使用对应的 HS 加密芯片进行项目量产，因为在 HS 芯片中，加入了 OTP 标识 eFuse 以及多个硬件安全加速器，增加了芯片在启动过程中对系统镜像的加解密以及签名验签验证，能够避免系统遭受外围的恶意篡改，删除以及复制。其中 JTAG 作为开发过程以及未来售后都必要的外部接口，一般在所有产品架构中都会存在整个生命周期。所以 JTAG 作为唯一一个必要的对外接口，其安全性具有至关重要的地位，本文将详细介绍在 Jacinto™ 7 处理器中如何对 JTAG 接口进行加解锁控制。

目录

1. Jacinto™ 7 处理器中的 JTAG 权限介绍 3
2. JTAG Debug Unlock 的两种方法及适用场景 3
3. JTAG Debug 离线加解锁方案 3
4. JTAG Debug 的实时加解锁方案 4
4.1 JTAG Debug 的实时解锁流程 5
4.2 创建 x509 调试证书配置文件 5
4.3 通过 x509 配置文件生成 x509 证书并签名 7
4.4 CCS 环境及 dbgauth 工具搭建 8
4.5 利用证书进行实时 JTAG 解锁 8
5. 总结 8
6. 参考文献 8

图

图 1 JTAG Debug 的在线实时解锁流程 5

表

表 1 不同芯片类型中的 JTAG 权限及状态 3
表 2 板级配置文件中的 security 参数配置及说明 4
表 3 debug level 取值及其含义 7
表 4 DRA821 芯片内部处理器核心对应 ID 取值 7

1. Jacinto™ 7 处理器中的 JTAG 权限介绍

在 TI 最新一代 Jacinto™ 7 处理器芯片中，为了保证客户整体系统的安全以及功能隐私，确保应用镜像不被恶意篡改、复制以及删除，TI 为每一颗 Jacinto™ 7 家族的 SoC 芯片都提供了 HS（high security）的芯片类型，其中 HS 芯片的详细开发流程可参考应用手册 Jacinto™ 7 High Security Device Development。

而 JTAG 作为嵌入式开发过程中必不可少的调试接口，在应用开发以及产品发布阶段，推荐进行不同的处理，从而避免第三方通过 JTAG 接口对产品系统进行攻击从而造成损失。针对这种考虑，在 GP 和 HS 芯片中，JTAG 接口具有不同的权限，如下表 1 所示：

表 1 不同芯片类型中的 JTAG 权限及状态

芯片类型	芯片状态	M3 JTAG 状态	其它核心 JTAG 状态
General Purpose (GP)	GP	Open	Open
High Security (HS)	HS-FS	Closed	Open
High Security (HS)	HS-SE	Closed	Controlled

其中本文所介绍的方法，可以适用于 TDA4x 以及 DRA82x 系列所有 HS 芯片，为方便阐述以及演示，本文将基于 DRA821 EVM 开发板以及 DRA821 RTOS SDK8.2 来进行介绍。旨在帮助开发人员在 HS-SE 芯片中实现对 JTAG 的控制，进而保证产品的安全以及隐私性。

2. JTAG Debug Unlock 的两种方法及适用场景

一般来说，Jacinto™ 7 家族 HS-SE 系列 SoC 中，如上表 1 所示，M3/M4F 核心的 JTAG 默认关闭且不能打开，从而确保 DMSC 核心的安全，进而保证芯片内部的时钟电源安全。而对于其它核心，例如 R5F/A72/DSP 等等，则可以通过 JTAG Debug 的离线加解锁以及在线实时加解锁两种方案来实现对 JTAG 的控制。本文将介绍两种方法对 JTAG 端口进行 Lock/Unlock：

JTAG Debug 的离线加解锁方案通过直接对各个核心镜像的 x509 证书进行授权，使其在运行时在芯片内部直接进行对 JTAG 的加锁/解锁，从而实现对 JTAG 的控制。其适用于后期对产品调试需求较少且能够对镜像进行实时更新的开发环境。

JTAG Debug 的实时加解锁方案则是首先设置系统默认将 JTAG 进行锁死，然后通过外部 JTAG 口或者 TISCI 等方式向芯片发送特定的解锁证书，从而实现对指定核心的 JTAG 进行解锁；芯片下电后，会继续保持 JTAG 死锁。其适用于后期有调试需求且对镜像文件更新受限的开发环境。

总的来说，相对于离线加解锁方案，实时解锁方案的安全性更高，部署成功后操作更加便捷且高效；相对的，离线加解锁方案在前期部署工作上会更加简单。

3. JTAG Debug 离线加解锁方案

所有在 HS-SE 芯片中执行的 binary，都需要对镜像进行签名/加密才可正常运行，具体的流程可以参考应用手册“SPRAD04”中的第二章节。而 JTAG Debug 离线加解锁则是在对核心镜像 binary 进行签名加密生成 x509 证书时，通过配置证书中的 debug extension 来设置对应核心 JTAG Debug 的权限。以 SBL 为例，在 /packages/ti/build/makerules/common.mk 文件中，当开发人员执行 make sbl_mmcsd_img_hs 编译 SBL 时，会调用下述指令对 SBL 进行签名并设置 debug extension 的权限。

```
$(SBL_CERT_GEN) -b $(SBL_BIN_PATH) -o $(SBL_TIIMAGE_PATH) -c R5 -l $(SBL_RUN_ADDRESS) -k $($(APP_NAME)_SBL_CERT_KEY) -d DEBUG -j DBG_FULL_ENABLE -m $(SBL_MCU_STARTUP_MODE)
```

其中“DBG_FULL_ENABLE”代表默认将 SBL 运行的核心，即 MCU1_0 的 JTAG Debug 设置为 full，即打开对应核心的所有 debug 权限，即预解锁。若删除此编译选项，即默认 JTAG Debug 设置为关闭，即预加锁。开发人员在镜像文件的编译以及签名加密过程中，可通过这个方式来对 JTAG 进行默认的离线加锁/解锁。

4. JTAG Debug 的实时加解锁方案

出于安全性考虑，在实时加解锁方案中，可以首先默认配置 JTAG 是锁死状态。以 DRA821 SBL 为例，在 SDK8.2 中，在 SBL 的 Makefile 中默认会配置 x509 证书的 debug extension 并设置其为 FULL 权限，即 MCU1_0 JTAG enable 状态，所以需要将其先进行关闭并删除 debug extension，改动如下 patch 所示。

```
diff --git a/packages/ti/build/makerules/common.mk b/packages/ti/build/makerules/common.mk
index 7265095..c40b090 100644
--- a/packages/ti/build/makerules/common.mk
+++ b/packages/ti/build/makerules/common.mk
@@ -676,7 +676,7 @@ else ifeq ($(SOC),$(filter $(SOC), am65xx am64x j721e j7200 j721s2))
 ifneq ($(OS),Windows_NT)
     $(CHMOD) a+x $(SBL_CERT_GEN)
 endif
-    $(SBL_CERT_GEN) -b $(SBL_BIN_PATH) -o $(SBL_TIIMAGE_PATH) -c R5 -l $(SBL_RUN_ADDRESS) -k $($(APP_NAME)_SBL_CERT_KEY) -d DEBUG -j
DBG_FULL_ENABLE -m $(SBL_MCU_STARTUP_MODE)
+    $(SBL_CERT_GEN) -b $(SBL_BIN_PATH) -o $(SBL_TIIMAGE_PATH) -c R5 -l $(SBL_RUN_ADDRESS) -k $($(APP_NAME)_SBL_CERT_KEY) -m
$(SBL_MCU_STARTUP_MODE)
 else ifeq ($(SOC),$(filter $(SOC), tpr12 awr294x))
 ifneq ($(OS),Windows_NT)
     $(CHMOD) a+x $(PDK_INSTALL_PATH)/ti/build/makerules/tpr12rom_sign_non_secure.sh
```

其次，需要在板级配置文件中，设置其为支持外部实时解锁 JTAG Debug，这样后续的外部实时对 JTAG 的控制才能生效，其改动如下 patch 所示：

```
diff --git a/packages/ti/drv/sciclient/soc/V2/sciclient_defaultBoardcfg_security.c b/packages/ti/drv/sciclient/soc/V2/sciclient_defaultBoardcfg_security.c
index 7b887d6..764ada3 100755
--- a/packages/ti/drv/sciclient/soc/V2/sciclient_defaultBoardcfg_security.c
+++ b/packages/ti/drv/sciclient/soc/V2/sciclient_defaultBoardcfg_security.c
@@ -112,7 +112,7 @@ __attribute__(( aligned(128), section(".boardcfg_data") )) =
         .magic = TISCI_BOARDCFG_SEC_DBG_CTRL_MAGIC_NUM,
         .size = sizeof(struct tisci_boardcfg_secure_debug_config),
     },
-    .allow_jtag_unlock = 0U,
+    .allow_jtag_unlock = 0x5AU,
     .allow_wildcard_unlock = 0x0,
     .min_cert_rev = 0x0,
     .jtag_unlock_hosts = {0, 0, 0, 0},
```

其中变量 allow_jtag_unlock 等于 0x5A 代表支持外部进行实时的 JTAG unlock，如需配置为默认不支持外部实时的 JTAG unlock，应置为 0。同样的，JTAG Debug 中其它变量的取值以及对应含义如下表 2 所示。

表 2 板级配置文件中的 security 参数配置及说明

Element	Type	Description
subhdr	u32	Magic and size for integrity check
allow_jtag_unlock	u8	Set to 0x5A if runtime jtag unlock is allowed. Set to 0 otherwise.
allow_wildcard_unlock	u8	If this field is set to 0x5A, the same debug unlock certificate will work across all devices where it passes the remaining checks. Set to 0 otherwise to enforce UID match before jtag unlock The X509 certificate must contain the UID of the device being unlocked in the designated field
allow_debug_level_rsvd	u8	Reserved field. Currently unused. Set to 0.

		This field can be used in the future to control the level of debug allowed with a certificate.
rsvd	u8	Reserved field. Currently unused. Set to 0.
min_cert_rev	u32	Minimum revision value that must be contained in the debug unlock certificate for it to be accepted. Use this field to enforce rollback protection for debug unlock certificates. Set to 0 if you do not wish to use this field.
jtag_unlock_hosts	u8(4)	Hosts allowed send jtag unlock message via TISCI Set one of the entries to 128 if jtag unlock must be allowed for all hosts.

在本文所介绍的 JTAG 加解锁中，只需要将对应的 unlock JTAG 选项打开即可，也就意味着加载了此配置文件的 SOC 将会支持外部 JTAG 操作。

4.1 JTAG Debug 的实时解锁流程

外部设备通过 JTAG 来解锁对应核心 JTAG 调试权限时，首先需要将设计过的且符合 ANS.1 规则的 x509 证书传输给 DMSC，DMSC 会解析此 x509 证书，并经过校验 UID，revision 等信息并通过之后，才会对 JTAG Debug 进行对应的权限解锁。其大致流程如下图 1 所示：

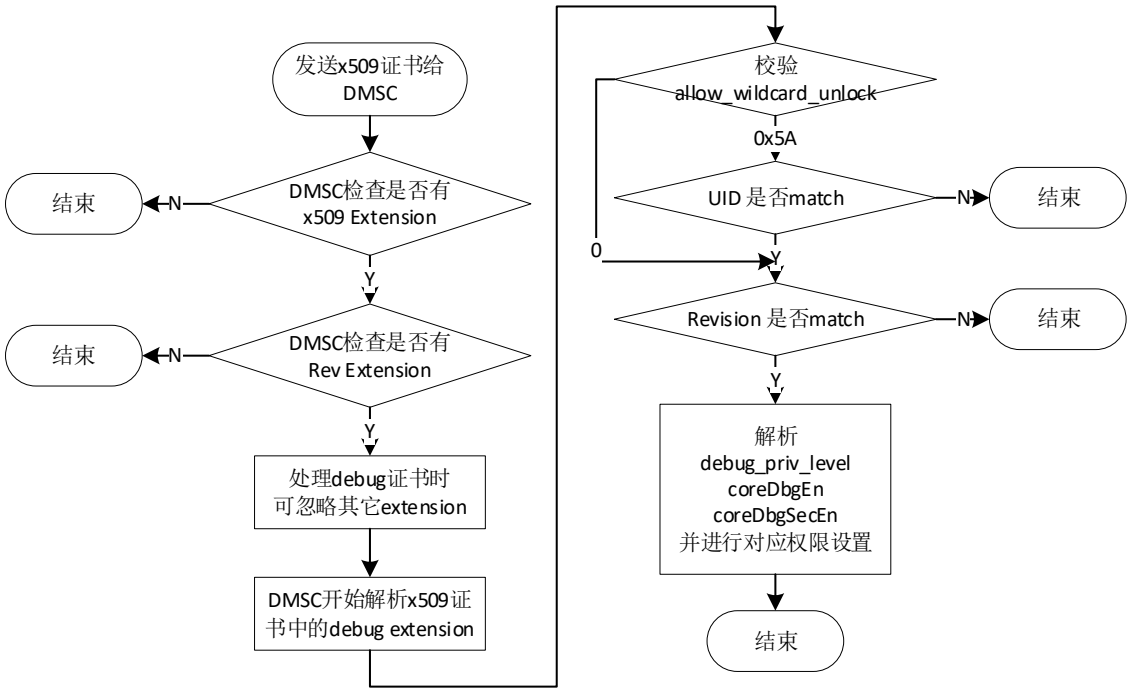

图 1 JTAG Debug 的在线实时解锁流程

4.2 创建 x509 调试证书配置文件

如 4.1 中图 1 所示的，首先要编写一个能够通过 DMSC 检查，并通过各项 UID，revision 等校验的 x509 证书，而在生成证书之前，首先需要按照 ANS.1 的规则编写一个配置文件。这样才能保证生成的调试证书能够被 DMSC 正确识别并进行配置。如下所示，为在 DRA821 中的 x509 调试证书配置文件模板。

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

 dirstring_type = nobmp

 [ req_distinguished_name ]
 C                      = US
 ST                     = SC
 L                      = SZ
 O                      = Texas Instruments., Inc.
 OU                     = DSP
 CN                     = Neo
 emailAddress           = li-wang@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:1

 [ debug ]
 debugUID = FORMAT:HEX,OCT:9190a27f5a03480cad48edd288f4efa3b6462294588a6e610c3a5df1c1a3e651
 debugType = INTEGER:0x00000004
 coreDbgEn = INTEGER:0x202101020607
 coreDbgSecEn = INTEGER:0x202101020607
```

值得注意的是，在不同芯片平台（TDA4x 或 DRA8x），亦或同一芯片平台的不同 UID 芯片中，进行不同核心（R5F/A72/DSP）的 JTAG Unlock 权限配置，开发人员需对【debug】下的内容进行更改。下面将对【debug】中的四个配置参数含义以及设置原则进行逐一介绍：

a. UID 获取

UID 是对于每颗芯片的独一无二的标志，在 Jacinto™ 7 家族中，其为 256bit 的 64 个 16 进制数构成。一般来说，获取 UID 的方法有三种：

- 在 UART boot 模式下通过 MCU UART 获取，并经过脚本解析得到。
- 利用 dbgauth 工具通过命令行获取。
- 利用 TISCI 在代码中获取 TISCI_MSG_GET_SOC_UID

其中第一种方法可参考应用手册“SPRAD04“中的第四章节获取，其需要板子实现支持 UART boot 模式并可以获取 MCU R5F 的串口 log。

其中第二中方法，将在本文 5.4 小节中介绍，对板级硬件无额外要求。

第三种方法参照 TISCI 手册，需要在代码中利用 TISCI Message 来获取 UID，此方法灵活度低，一般用于软件内部获取 UID 的接口。

开发人员只需根据自身开发环境选择其一即可。在获取到 UID 之后，只需将其替换到 【debug】规则下的 deviceUID 中即可。此处获取的 UID 是用来匹配 4.1 章节图 1 中的 UID 校验，只有 UID 校验通过之后，才会进行下一步的权限设置。

b. DebugType 配置原则

DebugType 为一个 32bit 数，其中低 16bit 数为 debug level 权限控制，高 16bit 目前为保留位，默认取全 0 即可。其中 debug level 的取值以及对应的含义如表 3 所示：

表 3 debug level 取值及其含义

Enumeration name	Value	Meaning
DEBUG_DISABLE	0	Disable debug
DEBUG_PRESERVE	1	Preserve current setting by locking registers
DEBUG_PUBLIC	2	Enable debug at public (non-secure) user and privileged level
DEBUG_PUBLIC_USER	3	Enable debug at public (non-secure) user level only
DEBUG_FULL	4	Enable full debug (both secure and non-secure privileged and user levels)
DEBUG_SECURE_USER	5	Enable debug for both secure and non-secure at user level only

在本文应用中，需要将其类型设置为 FULL，意味着一旦此 x509 脚本生效，将会将 JTAG 在 secure zone 以及 non secure 的 zone 都打开。

c. coreDbgEn 配置原则

coreDbgEn 控制哪些核心的 non-secure debug 将被打开，以 Processor ID 的形式列出，以 DRA821 为例，其内部所有处理器核心的 processor ID 如下表 4 所示：

表 4 DRA821 芯片内部处理器核心对应 ID 取值

Processor ID	Processor Name	Location in SoC
0x20	A72SS0_CORE0_0	COMPUTE_CLUSTER_J7VCL_TB_VDC_MAIN_0_MSMC_EN: (Cluster 0 Processor 0)
0x21	A72SS0_CORE0_1	COMPUTE_CLUSTER_J7VCL_TB_VDC_MAIN_0_MSMC_EN: (Cluster 0 Processor 1)
0x01	MCU_R5FSS0_CORE0	J7VCL_MCU_SEC_MMR_MCU_0: (Cluster 0 Processor 0)
0x02	MCU_R5FSS0_CORE1	J7VCL_MCU_SEC_MMR_MCU_0: (Cluster 0 Processor 1)
0x06	R5FSS0_CORE0	J7VCL_MAIN_SEC_MMR_MAIN_0: (Cluster 0 Processor 0)
0x07	R5FSS0_CORE1	J7VCL_MAIN_SEC_MMR_MAIN_0: (Cluster 0 Processor 1)

在 coreDbgEn 中，将所有的需要使能的 core 顺序列出即可，以 4.2 章节中的配置模板为例，coreDbgEn 值设置为 0x202101020607 意味着一旦 JTAG 配置文件生效，将会将 A72_0, A72_1, MCU_R5F0, MCU_R5F1, MAIN_R5F0, MAIN_R5F1 六个核心的 JTAG 权限全部按照配置生效。

d. coreDbgSecEn 配置原则

coreDbgSecEn 控制哪些核心的 secure debug 将被打开，以 Processor ID 的形式列出，以 DRA821 为例，其内部所有处理器核心的 processor ID 同样可参照表 4 所示。

4.3 通过 x509 配置文件生成 x509 证书并签名

开发人员根据需求设计完配置文件后，需要使用 openssl 来生成对应的 x509 证书，并使用自己的密钥对此证书进行签名，才能保证此证书能被对应的 HS-SE 板子验签并解析。

```
openssl req -new -x509 -key k3_dev_mpk.pem -nodes -outform der -out debug_unlock_cert.der -config x509template_Neo_UnlockJTAG.txt -sha512
```

其中“-key”后需要保持和开发人员 HS-SE 板子中烧写的密钥保持一致；“-out”后的文件即为将生成的 x509 证书文件；“-config”中即为在本文 4.2 小节中设计得到的配置文件。

此处生成的 debug_unlock_cert.der 文件即为后续用来实时加载到芯片中用于加锁/解锁的证书文件。

4.4 CCS 环境及 dbgauth 工具搭建

在生成完加锁/解锁所需要的证书文件之后，需要通过外部 JTAG 对 JTAG Debug 选项进行实时加锁/解锁，其中 PC 端的工具为 CCS 中的组件工具 dbgauth。开发人员需要安装 CCS，建议安装 CCS 9.3 及其之后版本：https://www.ti.com/tool/CCSTUDIO

安装完毕后，按照下列步骤来配置环境：

- 针对芯片类型新建 CCS configuration 并进行 launch，具体步骤可参照 SDK Guide，完毕后会在 ~/.ti/<ccs_version>/0/0/BrdDat/目录下有对应的配置文件 ccBoard0.dat。
- 同样的，CCS 安装完成后 dbgauth 工具会在<ccs install>/ccs/ccs_base/common/uscif/ 目录下。
- 需要确认当前版本的 dbgauth 是支持 Jacinto™ 7 系列 SoC 的 K3 架构的，可通过下述指令查询。

```
$ ./dbgauth --help
...
  --mode 3 = SUPPORTED TARGETS: All Keystone 3 devices
...
```

至此，PC 端的证书加载环境搭建完毕。

4.5 利用证书进行实时 JTAG 解锁

基于 4.4 小节建立的环境，可通过 dbgauth 工具来获取设备 UID，以及实时解锁 JTAG 等操作。首先需要将 4.1 小节中编译得到的 SBL 以及 TIFS 放到 boot media 中，确认 MCU1_0 以及 DMSC 已经正常运行起来，例如获取 UID 的命令以及打印为：

```
wangli@Ubuntu18:~/ti/ccs1011/ccs/ccs_base/common/uscif$ ./dbgauth -c ~/.ti/ccs1011/0/0/BrdDat/ccBoard0.dat -x xds110 -s cs_dap_0 -o getuid -m 3
Using board config file: /home/wangli/.ti/ccs1011/0/0/BrdDat/ccBoard0.dat
UID = 9190a27f5a03480cad48edd288f4efa3b6462294588a6e610c3a5df1c1a3e651
```

此时如果通过 CCS 直接链接 MCU1_0 的核心是无法连接的，因为默认的 JTAG 配置为 lock 状态。

通过 5.3 小节中生成的证书进行 JTAG 解锁的命令及打印为：

```
wangli@Ubuntu18:~/ti/ccs1011/ccs/ccs_base/common/uscif$ ./dbgauth -c ~/.ti/ccs1011/0/0/BrdDat/ccBoard0.dat -x xds110 -s cs_dap_0 -o unlock -m 3 -f
/home/wangli/Documents/J7/J7_SDK/J7200/unlock_JTAG/debug_unlock_cert.der
Using board config file: /home/wangli/.ti/ccs1011/0/0/BrdDat/ccBoard0.dat
Successfully opened certificate file /home/wangli/Documents/J7/J7_SDK/J7200/unlock_JTAG/debug_unlock_cert.der.
Read 1544 bytes from certificate file /home/wangli/Documents/J7/J7_SDK/J7200/unlock_JTAG/debug_unlock_cert.der.
The target device is unlocked.
```

在线执行完上述指令后，JTAG Debug 的权限将被打开，可以通过 CCS 对 MCU1_0，MCU2_0 等核心进行连接。一旦板子重启之后，板子又会回到 JTAG 加密锁死的状态，这样可以确保产品在默认状态下的安全性，仅在产品需要 debug 时才解锁对应核心。

5. 总结

Jacinto™ 7 家族芯片作为 TI 最新一代的汽车处理器，其出于安全性以及私密性考虑，所有的处理器都提供了 GP 以及 HS 两种版本，用户可在使用 GP 芯片开发完成后，使用 HS 芯片烧录自己的密钥并进行产品量产，这样能够最大程度上保证客户自身系统以及镜像文件的安全。而 JTAG 作为调试接口，是系统从始至终都会保留的外部接口，其安全性的保证即至关重要，本文通过介绍离线加解锁以及在线加解锁两种方案能够实现在不同环境下的 JTAG 控制，实现了 Jacinto™ 7 家族 HS 芯片中的 JTAG 加解锁控制。

6. 参考文献

1. TDA4VM Jacinto™ Processors for ADAS and Autonomous Vehicles Silicon Revisions 1.0 and 1.1 datasheet (Rev. J)
2. DRA829/TDA4VM/AM752x Technical Reference Manual (Rev. B)
3. TDA4 PSDKRA User Guide.
4. TDA4 PSDKLA User Guide.

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