KeyStone Architecture
SOC Security

User Guide

Literature Number: <SPRUHC3>
October 2011

Release History

Release	Date	Chapter/Topic	Description/Comments
Initial Release	October 2011	All	First release

Contents

List of Tables

List of Figures

Preface

About This Manual

This user guide provides an overview of the security concepts implemented on secure KeyStone architecture devices.

IMPORTANT NOTE—The information in this document should be used in conjunction with information in the device-specific Keystone Architecture data manual that applies to the part number of your device.

Notational Conventions

This document uses the following conventions:

- Commands and keywords are in **boldface** font.
- Arguments for which you supply values are in *italic* font.
- Terminal sessions and information the system displays are in `screen font`.
- Information you must enter is in **`boldface screen font`**.
- Elements in square brackets ([ ]) are optional.

Notes use the following conventions:

Note—Means reader take note. Notes contain helpful suggestions or references to material not covered in the publication.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

CAUTION—Indicates the possibility of service interruption if precautions are not taken.

WARNING—Indicates the possibility of damage to equipment if precautions are not taken.

Related Documentation from Texas Instruments

C66x CorePac User Guide	SPRUGW0
Bootloader User Guide	SPRUGY5

Related Standards

1. National Institute of Standards and Technology (NIST). *Specification for the Advanced Encryption Standard (AES)*. FIPS Publication 197, November 2001.
2. National Institute of Standards and Technology (NIST). *Specifications for Secure Hash Standard (SHA-1)*. FIPS Publication 180-2, August 2002.
3. Network Working Group. *US Secure Hash Algorithm 1 (SHA1)*. RFC 3174, September 2001.
4. Network Working Group. *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*. RFC 3447, February 2003.
5. RSA Laboratories. PKCS #1 v2.1: RSA Cryptography Standard. June 2002.

Glossary

AES	Advanced Encryption Standard – An encryption process that employs symmetric encryption, using a single key to both encrypt and decrypt a message.
BIST	Built-In Self Test
DRM	Digital Rights ManaDSPent – A system for protecting the copyrights of data circulated in electronic form by enabling secure distribution and/or disabling unauthorized distribution of the data.
DSP	Digital Signal Processor – A specialized processor designed specifically for digital signal processing.
DSP/BIOS	A scalable real-time kernel, designed specifically for the TMS320C5000 and TMS320C6000 DSP platforms.
FIPS	Federal Information Processing Standards – United States Government technical standards published by the National Institute of Standards and Technology (NIST).
GP	General purpose device
HS	High security device
ICEPick	In-Circuit Emulation TAP Selection Module.
ISTP	Internal Service Table Pointer – A register in the C66x DSP architecture that is used to locate interrupt service routines.
JTAG	Joint Test Action Group – A standard for boundary scan controllers, specifying how to control and monitor the pins of compliant devices on a board. Also referred to as *IEEE Standard 1149.1*.
KEK	Key encryption key.
LBIST	Logic Built-In Self Test.
LED	Load/Execute/Dump – A production test sequence
MBIST	Memory Built-In Self Test
MMR	Memory Mapped Register
NAND flash	A type of nonvolatile memory in which the cells are linked serially to each other and the grid of interconnects. NAND is best suited for sequential data reads and writes.
NIST	National Institute of Standards and Technology – A United States governmental body that helps develop standards including FIPS.
NOR flash	A fast type of nonvolatile memory in which the cells are linked in parallel to the grid. NOR is suited for random-access usage.
OS	Operating System
OTP	One time programmable memory.
PBIST	Power-on Build-In Self Test

PKCS	Public Key Cryptography Standard – A group of standards created and published by RSA Laboratories, the research center of RSA Security Inc.
POR	Power On Reset
RAM	Random Access Memory
REP	Restricted Entry Point – A register in the C66x+ DSP architecture that is used to designate the entry point of the internal exception handler used to request Supervisor / Secure mode services.
ROM	Read Only Memory – A type of data storage device which is manufactured with fixed contents.
RPC	Remote Procedure Call – A protocol which allows a program running on one processor to cause code to be executed on another processor without the programmer needing to explicitly code for this.
RSA	A public-key cryptosystem for both encryption and authentication, invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. Its name comes from the initials of their surnames.
SMEK	Secondary master encryption key.
SMPK	Secondary master publik key hash.
SHA-1	Secure Hash Algorithm – A one-way hash function developed by NIST.
TAP	J#TAG Test Access Port

Trademarks

Chapter 1

Introduction

IMPORTANT NOTE—The information in this document should be used in conjunction with information in the device-specific Keystone Architecture data manual that applies to the part number of your device.

This chapter provides an overview of the security concepts implemented on secure KeyStone architecture devices.

1.1 Introduction

Some KeyStone devices contains hardware features to support security within a device. This allows critical code to be executed on the DSP in a secure environment, hiding sensitive information from the outside world with the help of the following hardware features:

- Secure thread support within the DSP - hardware support that allows trusted code to be executed from secure ROM or secure RAM
- Protected transitions within the DSP between the secure and non-secure world via secure kernel software provided in the secure ROM
- Secure RAM
- Secure ROM, containing secure boot, cryptographic, and secure kernel support

This allows the customers to make use of the following feature sets to be implemented.

- Customer software authentication by means of secure boot.
- Customer software protection by means of hardware firewalls, encryption and runtime security.

This user guide will discuss and provide hardware and software details of secure boot, encryption and runtime security in detail.

Note—There is currently no support for ARM Trustzone within the KeyStone security architecture.

1.2 Device Types

After power on reset, a KeyStone secure device can be identified as one of three distinct device types: high security (secure), general purpose (non-secure), or bad (not usable). The configuration of devices is defined by 8 EFUSE bits that are programmed in the factory.

1.2.1 High Security (HS) Device

All security features are enabled in this device. All the security goals are full enforced in the secure device (i.e. debug features disabled, test features disables, etc.)

1.2.2 General Purpose (GP) Device

Device is not used for secure operation. Security is transparent and will not affect functionality, debug or test. Secure ROM code and other secure peripherals are not accessible (secure keys will not be programmed on a General Purpose device).

1.2.3 BAD Device

Only way to have a bad device in the factory is due to mis-blow of the EFUSE bits. BAD devices will be discarded by the factory.

1.3 ROM Boot Loader

The DSP ROM boot loader resides in the ROM of the device beginning at DSP address 0x00100000. The ROM boot loader (RBL) implements methods for booting in the listed modes. It reads the contents of the DEVSTAT(BOOT_REG0) register to determine boot mode and performs appropriate commands to effect boot of device. More details on bootloader and the different types of boots can be found in the *Bootloader User Guide* in "Related Documentation from Texas Instruments" on page ø-viii.

1.4 Secure Boot

Secure DSP boot is the foundation of security on an HS device. Secure boot allows the user bootloader to be authenticated before use, ensuring that the customer software is not modified from the original customer signed image. Secure devices always boot from the DSP internal secure ROM, enforcing a secure boot mechanism where code loaded on the platform is authenticated before it is executed. Only valid boot images, with the proper size, format, encryption and signature, will be accepted.

GP devices will not perform secure boot and will not be able to use any of the run-time security features within the DSP. However, GP devices will be able to reconfigure the HW firewalls (MPUs) and use the HW cryptography resources.

Note that theuser bootloader (UBL) can be signed only, or signed and encrypted. The UBL can be encrypted using just the master encryption key, a combination of the master encryption key and a statistically unique device specific key, or a customer delegated device specific key. By encrypting with a device specific key, this will bind the flash image to a specific device, further enhancing the security. More details about encryption and keys will be provided later in this guide.

1.5 Secure Kernel

The CorePac secure kernel resides in the secure ROM. It provides a protected transition between the secure and the non-secure execution levels within the DSP. The secure kernel uses a combination of security and privilege levels, hardware features, and ROM-based software to provide security to both code and data.

- **Hardware Management:** The CorePac hardware features present security challenges. Cache modes, EDC hardware and so on all present interesting security complications. The CorePac addresses this by restricting problematic functionality to secure supervisor access only, implying that the secure kernel in the ROM must handle or help load new secure modules to handle these functions.
- **Software Interface:** The secure kernel provides a mechanism for invoking its APIs and allowing a hosted operating system to load and execute applications, as well as secure programs.

1.5.1 Hardware Management

The secure kernel provides the following functionality:

- **Hardware Initialization:** Sets the Corepac hardware in a known state after a reset and transfers control to the secure boot.
- **Interrupt and Exception Servicing:** The only way to enter secure supervisor mode on a secure CorePac is through a reset, interrupt, or exception. All these entrypoints are controlled by the secure supervisor ROM. Note that software exceptions are used to communicate between the non-secure application and the secure kernel and secure algorithms.

- **Proxy Access to Hardware Features:** The secure kernel provides access to the secure keys in the system for encryption and decryption.

1.5.2 Software Interface

All secure kernel API calls are invoked from non-secure code via the software exception (SWE) instruction. The secure kernel also provides a means to invoke host operating system calls via the downcall API.

1.6 Run-time Security

Through the secure kernel running on the DSP, the HS devices provide security services to OS applications. While some services can be accessed directly via the secure ROM interface code, the user can develop their own trusted software, encrypted and signed for protected applications. Run-time security uses the CorePac memory protection unit (MPU) features and the system input output memory protection unit (IOMPU) features to protect memory access and MMR access for specific masters using privilege levels. Hence the trusted software can be run from a secure memory location for protected applications.

1.7 Security vs. Privilege

On HS devices, all C66x+ security features are enabled upon power up. Upon reset, the C66x+ CorePac starts executing from Secure ROM in secure supervisor mode. All HS devices based on the C66x+ architecture, support two security levels and two privilege levels, with security being orthogonal to privilege. Available privilege levels are *supervisor* and *user*; available security levels are secure and non-secure.

Privilege includes the roles within the system:

- Protects relatively fixed OS from more dynamic application-level code
- Insulates stable/proven code from non-trusted newer/unproven code
- Allows finer-grain recovery from transient application failures

In contrast, security includes the following roles:

- Protects vendors from customers, competing vendors and pirates
- Protects consumers from information and identity theft
- Enforces usage restrictions set by vendors, despite allowing great programmability for the device

Table 1-1 Orthogonality of Security and Privilege

Privilege	Security	
	Secure	**Non-secure**
Supervisor	Secure Kernel and Secure Boot Loader	DSP/BIOS or other OS Kernels
User	Licensed Algorithms, other software	Non-secure applications on other OS kernels

Chapter 2

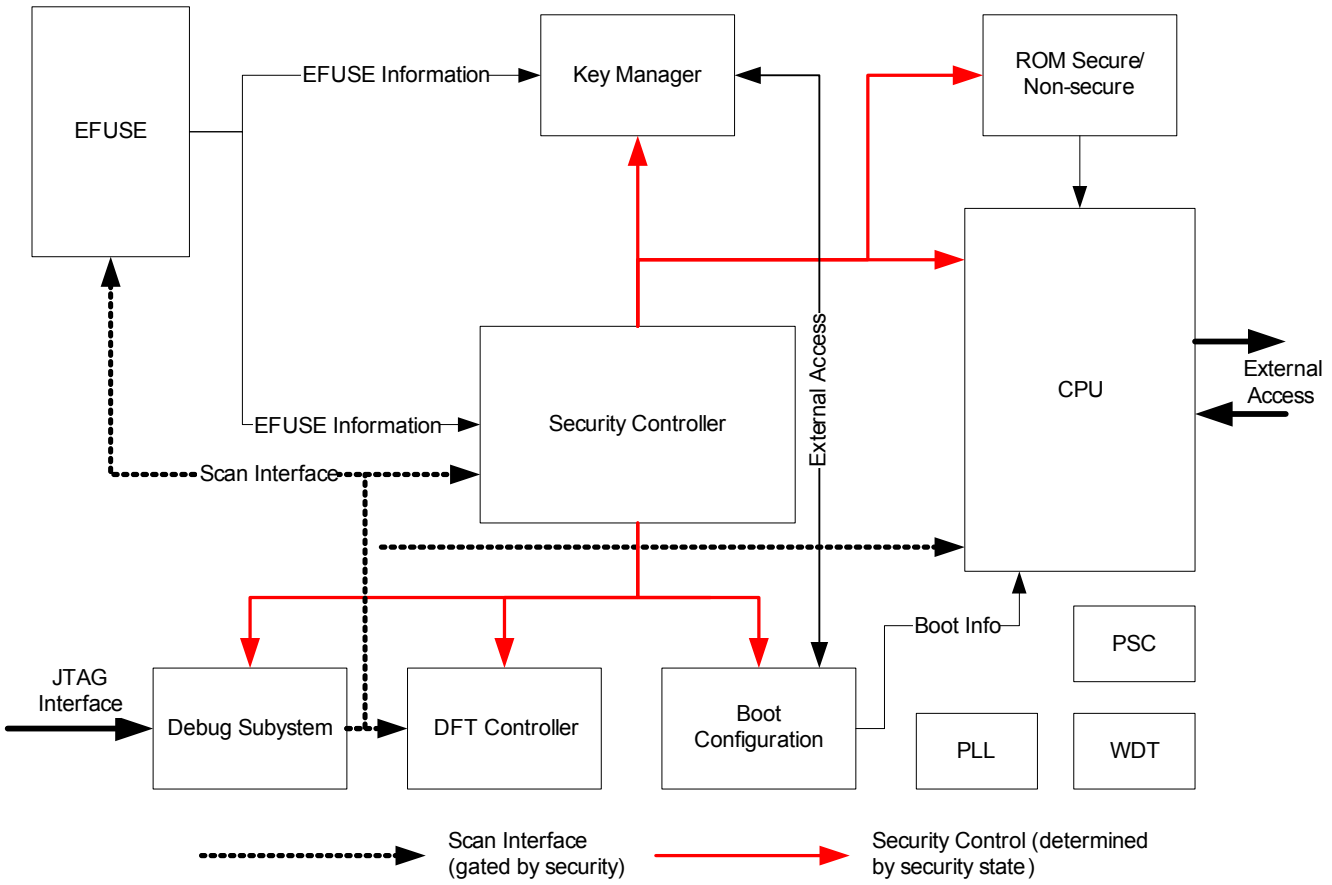
Hardware Security Features

This chapter provides an overview of the hardware features of secure KeyStone devices that help in implementing device security, and it describes how this hardware can be used in implementing security in the system.

2.1 Secure Hardware Overview

Security features on the secure KeyStone devices are enabled by a number of different hardware components. Hence security spans the entire system within the KeyStone architecture. The key components in the security architecture are shown in Figure 2-1.

Figure 2-1 Overview of Core Security Hardware Components



The CPU block in the figure is the C66x+ DSP and associated memory system logic. The Cortex A8 and associated memory system logic are not pictured and are not involved with the security system.

In the keystone security architecture, there are two types of devices: a general purpose (GP) device in which the security features are not enabled and there is no secure boot; and a high security (HS) device in which all the security features are enabled and secure boot is mandatory. The device type is set within the non-volatile EFUSE block within the system and understood by the security controller and related blocks.

These hardware components along with a secure kernel and secure bootloader form the integral components of a secure device.

2.2 Security Keys

2.2.1 Security Keys supported

A high security (HS) device contains three defined security keys. Apart from these keys, the device may support one-time-programmable (OTP) memory (see the device device data manual for details of the amount of OTP supported). The security keys are the foundation for the HW based root of trust in the system that enables secure boot. The KEK and SMEK keys can only be accessed by a secure supervisor on an HS device.

- **Key Encryption Key** (KEK) – Is a random 128-bit key used for symmetrical encryption, that is statistically unique. During manufacturing, TI will use a NIST-800-22 certified random number generator to create this key and program it in to the device. The key can be used to encrypt data specific to a device, for example a device specific key store.
- **Secondary Master Encryption Key** (SMEK) – Is a customer specific 128-bit key used for symmetrical encryption. The SMEK is a shared secret between the device and the customer and is unique to a given customer. The key is programmed by the customer and supports error correction (see specific device for details on error correction). The key can be used to encrypt / decrypt data specific to a customer.
- **Secondary Master Public Key Hash** (SMPK_hash) – Is a 256-bit value containing the result of a SHA2-256 hash of the customer public key. It is used to establish a root public key which in turn is used to authenticate the software running on the device. The value is programmed by the customer and supports error correction (see the device device data manual for details on error correction).
- **One-Time-Programmable Memory (OTP)** – Some HS devices will support one-time-programmable (OTP) memory. The amount of OTP memory is device dependent, so please refer to the appropriate device for more information. The OTP memory is used for customer specific information and should employ appropriate error detection and correction mechanisms. Depending on the device, the OTP memory when read by the CPU will be organized and protected in different manners. For example in some KeyStone devices, there are 4096-bits of OTP memory which will be divided into 32, 128-bit *keys* that can be individually protected when read by the CPU. However, within the OTP memory array, the memory is always organized into 32-bit rows with each row containing a read protect and a write protect bit. It is recommended that when programming the OTP memory, the minimum amount of information programmed is one row (i.e. 32-bits). This way, once the information is programmed, the read and/or write protect bits can be programmed at the same time and thus guarantee the integrity and secrecy of the information.

All keys and OTP memory are protected by two mechanisms: one inside the storage array itself and the other inside the hardware that allows the value to be read via a CPU memory mapped register (MMR) read. The first protection mechanism consists of the read and write protect bits inside the storage array. The storage is organized into 32-bit rows. Each row has a read protect and write protect bit associated with that row. The row can be read or programmed through the programming interface as long as the read protect or write protect bits are not programmed. If any sensitive data is being stored, it is recommended that the data is stored in 32-bit chunks and that the read and write protect bits are programmed immediately after it has been validated the row was programmed correctly. The second protection mechanism is in the MMR interface when the values of the keys or OTP are being read by the CPU. For example, in some KeyStone devices, the key manager provides the interface to access the values stored in

the OTP Memory. The key manager divides the 4096 bits of storage into 32 128-bit *keys*. Each key has a corresponding public / private enable and this can be used to limit the visibility of a key. By default, all keys are public, i.e. can be read by any master that can access the key manager. Therefore, if sensitive data is being stored in the OTP memory, it is recommended that the corresponding public/private enable bit is programmed to make the 128-bit key private. Please note that while you need to program the public/private enable for a 128-bit key, you do not necessarily need to program all 128-bits at one time. As stated above, it is recommended that you program at least 32 bits at a time so that the read/write protect bits can be programmed at the same time. When you program a value in the OTP memory or in one of the customer keys in order for the CPU to read the value, you need to issue a power-on-reset to the device so that the updated value becomes visible.

The SMEK, KEK and the OTP keys marked as private are only visible in secure supervisor mode (CorePac). These memory mapped registers (MMR for private keys) will become invisible if the device is switched to a non-secure operating mode. The public keys, i.e. the SMPK and the OTP keys marked as public, can be accessed by any master that is programmed to access the MMRs in the system. Table 2-1 shows access permissions of the security keys in various types of devices.

Table 2-1 Access permission for security keys

Device type	HS	GP	Bad
Masters that can access private (aka secure) information	Any secure supervisor (Core Pac)	None (keys are not programmed)	None
Masters that can access public information	All masters that can access the key manager		

2.2.2 Key Programming

The keys SMEK, SMPK, and the OTP memory can be programmed by the customer during manufacturing. Secure devices will support a VPP power pin in the package for programming. TI will provide the programming, validation and error correction software and the methodology.

2.2.3 Error Correction and Detection of Keys

In order to protect against programming errors, both defined keys and OTP memory use error detection and correction (ECC) algorithms. See the device device data manual for more information on the error detection and correction algorithm used.

For example, in some keystone devices, the information programmed by the customer includes 16 bits of ECC data per 128 bits. The ECC data is generated by performing a polynomial divide of the data (after multiplying by 2^{16}) by the generator polynomial (g(x) = 10110111101100011) and using the remainder as the ECC data.

```
EccData(x) = (key(x) * 2^16) / g(x)
```

This provides for two bits of error correction and three bits of error detection. The ECC data is generated from the keys using the following sequence:

- Arrange the 128 bits as a byte stream, for example: 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10. Because the secure mode only supports little endian, these values will appear like this when viewed as 32 bit values in the boot config space: 0x04030201 0x08070605 0x0c0b0a09 0x100f0e0d.
- Convert the data string into a bit stream. Note that bit 0 of the last byte is the least significant byte.

- Perform the division. The remainder will be a 16-bit value that must be converted to two bytes.

2.2.4 Security Information in the Device

Within each device, security information will be programmed. These values will be available via memory mapped registers, the details of which are given later in this document. Table 2-2 shows the bits that will be programmed in EFUSE for each secure device:

Table 2-2 Standard EFUSE Bits for Security

Bits	Description
8	Device Type
128	KEK (Key Encryption Key – 128 bit random number)
128	MEK (Master Encryption Key - Programmed to the TI MEK value)
256	MPK_hash (Master Public Key Hash – Programmed to the TI MPK_hash value)

Table 2-3 shows the bits that can be programmed by the customer in EFUSE for each secure device:

Table 2-3 Customer EFUSE Bits for Security (Part 1 of 2)

Bits	Description
128	Customer Control Fields [127:36] - Reserved [35 - 32] - JTAG disable [31] - Controls public / private enable for Key Mgr 1 – Key 15 [30] - Controls public / private enable for Key Mgr 1 – Key 14 ... [1] - Controls public / private enable for Key Mgr 0 – Key 1 [0] - Controls public / private enable for Key Mgr 0 – Key 0
128	Unique ID (UID) Maps to MMR in boot config module.
160	SMEK + ECC (Secondary Master Encryption Key – Programmed by the Customer to their 128 bit SMEK value; Also includes associated error detection / correction information)

Table 2-3 Customer EFUSE Bits for Security (Part 2 of 2)

Bits	Description
320	SMPK_hash + ECC (Secondary Master Public Key Hash - Programmed by the Customer to their 256 bit SMPK_hash value; Also includes associated error detection / correction information)
4096	One-Time Programmable bits OTP[3968:4095] - Key Mgr 1 - Key 15 OTP[3840:3967] - Key Mgr 1 - Key 14 OTP[3712:3839] - Key Mgr 1 - Key 13 OTP[3584:3711] - Key Mgr 1 - Key 12 OTP[3456:3583] - Key Mgr 1 - Key 11 OTP[3328:3455] - Key Mgr 1 - Key 10 OTP[3200:3327] - Key Mgr 1 - Key 9 OTP[3072:3199] - Key Mgr 1 - Key 8 OTP[2944:3071] - Key Mgr 1 - Key 7 OTP[2816:2943] - Key Mgr 1 - Key 6 OTP[2688:2815] - Key Mgr 1 - Key 5 OTP[2560:2687] - Key Mgr 1 - Key 4 OTP[2432:2559] - Key Mgr 1 - Key 3 OTP[2304:2431] - Key Mgr 1 - Key 2 OTP[2176:2303] - Key Mgr 1 - Key 1 OTP[2048:2175] - Key Mgr 1 - Key 0 OTP[1920:2047] - Key Mgr 0 - Key 15 OTP[1792:1919] - Key Mgr 0 - Key 14 OTP[1664:1791] - Key Mgr 0 - Key 13 OTP[1536:1663] - Key Mgr 0 - Key 12 OTP[1408:1535] - Key Mgr 0 - Key 11 OTP[1280:1407] - Key Mgr 0 - Key 10 OTP[1152:1279] - Key Mgr 0 - Key 9 OTP[1024:1151] - Key Mgr 0 - Key 8 OTP[896:1023] - Key Mgr 0 - Key 7 OTP[768:895] - Key Mgr 0 - Key 6 OTP[640:767] - Key Mgr 0 - Key 5 OTP[512:639] - Key Mgr 0 - Key 4 OTP[384:511] - Key Mgr 0 - Key 3 OTP[256:383] - Key Mgr 0 - Key 2 OTP[128:255] - Key Mgr 0 - Key 1 OTP[0:127] - Key Mgr 0 - Key 0
End of Table 2-3	

2.3 CPU (CorePac and ARM)

The C66x CorePac can execute code with two levels of security: secure and non-secure. For secure devices that contain a CorePac and an ARM, the CorePac is always the security master while either the CorePac or the ARM can be the boot master. In case of multiple CorePacs in the system, CorePac0 is the security master and will authenticate and decrypt the boot image for the ARM. The boot image for other CorePacs in a multiple DSP system can be both authenticated and decrypted by any of the CorePacs. This forms the basis of secure boot in a multiple CPU keystone device.

After the secure boot, the CorePac supports security outside the boundary of the module because the CorePac can issue both secure and non-secure supervisor and user transactions into the system. The fact that the CorePac can issue secure supervisor transactions that are understood by various device components, forms the basis of run-time security in a secure KeyStone device.

2.3.1 Corepac Memory Protection Architecture

In a CorePac, the memory protection unit (MPU) allows memories and peripherals to be protected. The architecture divides the internal memory map into pages. Each page has an associated set of permissions. Access to each page is restricted based on memory requester, requester privilege, access type, and security settings (see Figure 2-2).

Figure 2-2 Memory Protection Register

31 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	AID5	AID4	AID3	AID2	AID1	AID0	AIDX	LOCAL	NS	EMU	SR	SW	SX	UR	UW	UX
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 2-4 Memory Protection Register Description

Bits	Field	Description
31-16	Reserved	Reserved
15-10	AID5	1=Allow accesses from ID 5-0
9	AIDX	1=Allow accesses from ID > = 6
8	LOCAL	1=Allow access from DSP to its local memories (L1/L2 only)
7	NS	1=Non-secure access. Refer Table 2-5
6	EMU	1=Emulation access. Refer Table 2-5
5	SR	1=Supervisor may read
4	SW	1=Supervisor may write
3	SX	1=Supervisor may execute
2	UR	1=User may read
1	UW	1=User may write
0	UX	1=User may execute
End of Table 2-4		

Each memory protection page has two bits marked NS and EMU, indicating the security configuration for that page. The encoding of these bits is shown in Table 2-5. For more details refer to *C66x CorePac User Guide* in "Related Documentation from Texas Instruments" on page ø-viii

Table 2-5 CorePac Memory Protection Security Fields

NS	EMU	CPU Effects	Emulation Effects
0	0	Page is secure. Only Secure mode may access this page. Secure code executing within this page retains its secure status.	Emulation reads/writes to this page are not permitted. Emulation halts and trace are not permitted when executing from this page.
0	1	Page is secure. Only Secure mode may access this page. Secure code executing within this page retains its secure status.	Emulation reads/writes to this page are permitted. Emulation halts and trace are permitted when executing from this page.
1	x	Page is not secure. Both Secure and Non-secure code may access this page. Secure code may branch to this page, but upon doing so it will lose its secure status.	Emulation reads/writes to this page are permitted. Emulation halts and trace are permitted when executing from this page.

2.3.2 Secure vs. Non-secure Execution Modes

Secure execution mode is entered on any interrupt or exception. Potential security holes here can be managed by having all interrupt handling managed in the secure ROM. After secure boot, the secure ROM actually maps the vector table to secure RAM, although after boot all ISRs are in ROM. This supports user installable ISRs. Secure execution mode is exited when the CPU branches to non-secure code. This scheme allows for a kernel in a secure boot ROM to be in complete control of code execution, including boot sequence and interrupt/exception. When the CPU is in secure execution mode, it issues secure sideband signals along with each access that are used by the memory protection logic within the CorePac to determine whether or not the data/program access is allowed to that particular page in CorePac memory. A non-secure access to secure memory results in no access (zero-data read, no write) and an exception. CorePac will propagate the security signals and allow secure communication outside of CorePac via the MDMA path through the MSMC as well as CFG interface. So the CorePac can issue secure transactions into the entire system and non-secure transactions results in no access and an exception.

The CPUs have complete access to all L1 and L2 memories except for the secure ROM and part of L2 memory reserved by secure kernel (Lowest 64kb) which will be completely disabled for GP devices.

2.4 Hardware Security Controller

In order to implement all the functionality needed by the security system, KeyStone secure devices will have a number of registers to implement security control and provide status of the security system.

2.4.1 Chip-level-registers Memory Map

Table 2-6 shows the memory map for the unique ID bits, TI programmed and the customer programmed keys and error correction bits for the keys. Refer to individual device manual for base address of these registers in the chip-level-registers memory map section.

Table 2-6 Memory Map - Chip Config Registers for Security (Part 1 of 2)

Address Offset	Register Name	Description
0x000	UID0[31:0]	Unique ID [31:0]
0x004	UID1[63:32]	Unique ID [63:32]
0x008	UID2[95:64]	Unique ID [95:64]
0x00C	UID3[127:96]	Unique ID [127:96]
0x010	Customer Control Field 0[31:0]	Customer Control Field [31:0]
0x014	Customer Control Field 1[63:32]	Customer Control Field [63:32]
0x018	Customer Control Field 2[95:64]	Customer Control Field [95:64]
0x01C	Customer Control Field 3[127:96]	Customer Control Field [127:96]
0x020 – 0x06C	Reserved	
0x070	CONTROL_KEK_0	Key Encryption Key [31:0]
0x074	CONTROL_KEK_1	Key Encryption Key [63:32]
0x078	CONTROL_KEK_2	Key Encryption Key [95:64]
0x07C	CONTROL_KEK_3	Key Encryption Key [127:96]
0x080	CONTROL_MEK_0	Master Encryption Key [31:0]
0x084	CONTROL_MEK_1	Master Encryption Key [63:32]
0x088	CONTROL_MEK_2	Master Encryption Key [95:64]
0x08C	CONTROL_MEK_3	Master Encryption Key [127:96]
0x090	CONTROL_MPK_hash_0	Master Public Key Hash [31:0]
0x094	CONTROL_MPK_hash_1	Master Public Key Hash [63:32]
0x098	CONTROL_MPK_hash_2	Master Public Key Hash [95:64]
0x09C	CONTROL_MPK_hash_3	Master Public Key Hash [127:96]
0x0A0	CONTROL_MPK_hash_4	Master Public Key Hash [159:128]
0x0A4	CONTROL_MPK_hash_5	Master Public Key Hash [191:160]
0x0A8	CONTROL_MPK_hash_6	Master Public Key Hash [223:192]
0x0AC	CONTROL_MPK_hash_7	Master Public Key Hash [255:224]
0x0B0	CONTROL_SMEK_BCH_0	Secondary Master Encryption Key + BCH Error Correction[31:0]
0x0B4	CONTROL_SMEK_BCH_1	Secondary Master Encryption Key + BCH Error Correction [63:32]
0x0B8	CONTROL_SMEK_BCH_2	Secondary Master Encryption Key +BCH Error Correction [95:64]
0x0BC	CONTROL_SMEK_BCH_3	Secondary Master Encryption Key + BCH Error Correction [127:96]
0x0C0	CONTROL_SMEK_BCH_4	Secondary Master Encryption Key + BCH Error Correction [159:128]
0x0C4 - 0x0CC	Reserved	
0x0D0	CONTROL_SMPK_hash_BCH_0	Secondary Master Public Key Hash + BCH Error Correction [31:0]
0x0D4	CONTROL_SMPK_hash_BCH_1	Secondary Master Public Key Hash + BCH Error Correction [63:32]
0x0D8	CONTROL_SMPK_hash_BCH_2	Secondary Master Public Key Hash + BCH Error Correction [95:64]

Table 2-6 Memory Map - Chip Config Registers for Security (Part 2 of 2)

Address Offset	Register Name	Description
0x0DC	CONTROL_SMPK_hash_BCH_3	Secondary Master Public Key Hash + BCH Error Correction [127:96]
0x0E0	CONTROL_SMPK_hash_BCH_4	Secondary Master Public Key Hash + BCH Error Correction [159:128]
0x0E4	CONTROL_SMPK_hash_BCH_5	Secondary Master Public Key Hash + BCH Error Correction [191:160]
0x0E8	CONTROL_SMPK_hash_BCH_6	Secondary Master Public Key Hash + BCH Error Correction [223:192]
0x0EC	CONTROL_SMPK_hash_BCH_7	Secondary Master Public Key Hash + BCH Error Correction [255:224]
0x0F0	CONTROL_SMPK_hash_BCH_8	Secondary Master Public Key Hash + BCH Error Correction [287:256]
0x0F4	CONTROL_SMPK_hash_BCH_9	Secondary Master Public Key Hash + BCH Error Correction [319:288]
0x0F8 – 0x1FC	Reserved	
End of Table 2-6		

2.4.1.1 Unique ID (UID)

All UID registers are readable by any master. The fields in this register are shown in Figure 2-3:

Figure 2-3 UID0 - 3

127 96	95 64	63 32	31 0
UID3	UID2	UID1	UID0
R-e	R-e	R-e	R-e

Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE — see the device-specific data manual

Table 2-7 UID 0/1/2/3 register field description

Bits	Field	Description
127-96	UID3	Value of the UID programmed in to the customer programmable EFUSE
95-64	UID2	
63-32	UID1	
31-0	UID0	

2.4.1.2 Customer-control Registers

All customer control field registers are readable by any master. The bits are used to control the privacy of the OTP keys in the key manager and to disable/enable JTAG in a secure device. The fields in this register are shown in Figure 2-4:

Figure 2-4 Customer Control Register 0

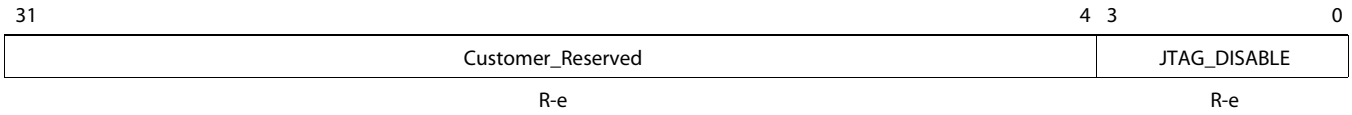
31 0
PUB_PRI_EN
R-e

Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE — see the device-specific data manual

Table 2-8 Customer Control Register 0 description

Bits	Field	Description
31-0	PUB_PRI_EN	Public / Private enable for OTP keys in the key manager. 0=Public; Any master can access the keys 1=Private; Only secure supervisor can access the keys

Figure 2-5 Customer Control Register 1

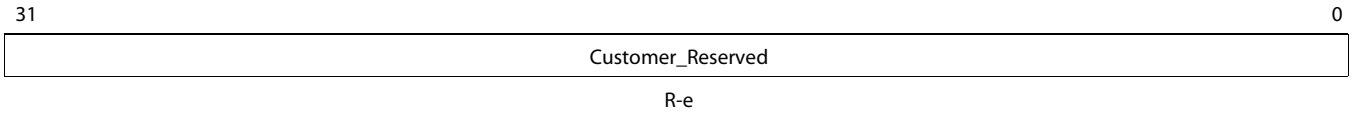


Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE — see the device-specific data manual

Table 2-9 Customer Control Register 1 description

Bits	Field	Description
31-4	Customer_Reserved	Value of the customer EFUSE reserved programmed into the customer programmable EFUSE
3-0	JTAG_DISABLE	Value of JTAG disable. Refer Table 2-40 for more details.

Figure 2-6 Customer Control Register 2

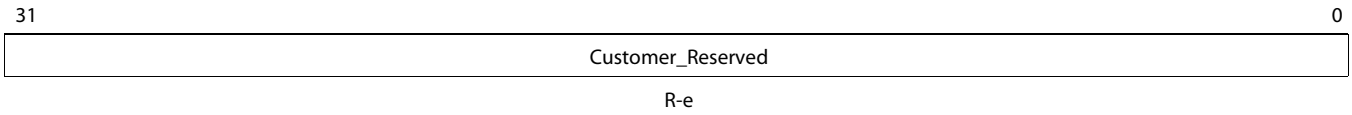


Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE — see the device-specific data manual

Table 2-10 Customer Control Register 2 description

Bits	Field	Description
31-0	Customer_Reserved	Value of the Customer EFUSE reserved programmed into the customer programmable EFUSE

Figure 2-7 Customer Control Register 3



Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE — see the device-specific data manual

Table 2-11 Customer Control Register 3 description

Bits	Field	Description
31-0	Customer_Reserved	Value of the customer EFUSE reserved programmed into the customer programmable EFUSE

2.4.1.3 CONTROL_KEK_0 - 3 Registers

These four registers together contain the 128-bit value of the KEK. The reset value will come from the standard EFUSE. These registers will only be accessible by secure supervisor from CorePac and ARM for both reads and writes. The registers are write once per POR so that the EFUSE value of the KEK can be replaced once with a derivative key value during boot. The fields in this register are shown in Figure 2-8:

Figure 2-8 CONTROL_KEK_0 - 3

127 96	95 64	63 32	31 0
CONTROL_KEK_3	CONTROL_KEK_2	CONTROL_KEK_1	CONTROL_KEK_0
R/WPOR-es	R/WPOR-es	R/WPOR-es	R/WPOR-es

Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

Table 2-12 CONTROL_KEK_0 - 3 Register Field Description

Bits	Field	Description
127-96	CONTROL_KEK_3	Contains 32 bits of 128-bit KEK (Key Encryption Key) value
95-64	CONTROL_KEK_2	
63-32	CONTROL_KEK_1	
31-0	CONTROL_KEK_0	

2.4.1.4 CONTROL_MEK_0 - 3 Registers

These four registers together contain the 128-bit value of the MEK. The reset value will come from the standard EFUSE. These registers will only be accessible by secure supervisor from the CorePacs for both reads and writes. The registers are write once per POR so that the EFUSE value of the MEK can be replaced once with the error corrected value of the SMEK during boot. The fields in this register are shown in Figure 2-9:

Figure 2-9 CONTROL_MEK_0 -3

127 96	95 64	63 32	31 0
CONTROL_MEK_3	CONTROL_MEK_2	CONTROL_MEK_1	CONTROL_MEK_0
R/WPOR-es	R/WPOR-es	R/WPOR-es	R/WPOR-es

Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

Table 2-13 CONTROL_MEK_0 - 3 Register Field Description

Bits	Field	Description
127-96	CONTROL_MEK_3	Contains 32 bits of 128-bit MEK (Master Encryption Key) value
95-64	CONTROL_MEK_2	
63-32	CONTROL_MEK_1	
31-0	CONTROL_MEK_0	

2.4.1.5 CONTROL_MPK_hash_0 - 7 Registers

These eight registers together contain the 256-bit value of the MPK hash. The reset value will come from the standard EFUSE. These registers will be readable by any master at any privilege level since its a public key hash. The registers are write once per POR and restricted to secure supervisor from the CorePac so that the EFUSE value of the MPK hash can be replaced once with the error corrected version of the SMPK hash during boot. The fields in this register are shown in Figure 2-10:

Figure 2-10 CONTROL_MPK_hash_0 - 7

255 224	223 192	191 160	159 128
CONTROL_MPK_hash_7	CONTROL_MPK_hash_6	CONTROL_MPK_hash_5	CONTROL_MPK_hash_4
R/WPOR-es	R/WPOR-es	R/WPOR-es	R/WPOR-es

127 96	95 64	63 32	31 0
CONTROL_MPK_hash_3	CONTROL_MPK_hash_2	CONTROL_MPK_hash_1	CONTROL_MPK_hash_0
R/WPOR-es	R/WPOR-es	R/WPOR-es	R/WPOR-es

Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

Table 2-14 CONTROL_MPK_hash_0 - 7 Register Field Description

Bits	Field	Description
255-224	CONTROL_MPK_hash_7 [255:224]	Contains the appropriate 32-bit value of the 256-bit MPK (Master Public Key) Hash Value
223-192	CONTROL_MPK_hash_6 [223:192]	
191-160	CONTROL_MPK_hash_5 [191:160]	
159-128	CONTROL_MPK_hash_4 [159:128]	
96-127	CONTROL_MPK_hash_3 [127:96]	
64-95	CONTROL_MPK_hash_2 [95:64]	
32-63	CONTROL_MPK_hash_1 [63:32]	
31-0	CONTROL_MPK_hash_0 [31:0]	

2.4.1.6 CONTROL_SMEK_BCH_0 - 4 Registers

These five registers together contain the 160-bit value of the SMEK_BCH. The reset value will come from the Customer EFUSE. These registers will only be accessible by secure supervisor from the DSPs for reads. These registers are read only. The fields in this register are shown in Figure 2-11:

Figure 2-11 CONTROL_SMEK_BCH_0 - 4

159 128	127 96	95 64	63 32	31 0
CONTROL_SKEK_4	CONTROL_SKEK_3	CONTROL_SKEK_2	CONTROL_SKEK_1	CONTROL_SKEK_0
R-e	R-e	R-e	R-e	R-e

Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

Table 2-15 CONTROL_SMEK_BCH_0 - 4 Register Field Description

Bits	Field	Description
159-128	CONTROL_SMEK_BCH_4	Contains the appropriate 32-bit value of the 160-bit SMEK_BCH (BCH encoded Secondary Master Encryption Key) value.
127-96	CONTROL_SMEK_BCH_3	
95-64	CONTROL_SMEK_BCH_2	
63-32	CONTROL_SMEK_BCH_1	
31-0	CONTROL_SMEK_BCH_0	

2.4.1.7 CONTROL_SMPK_hash_BCH_0 - 9 Registers

These ten registers together contain the 320-bit value of the MPK_BCH. The reset value will come from the customer EFUSE. These registers will be readable by any master at any privilege level. These registers are read only. The fields in this register are shown in Figure 2-12:

Figure 2-12 CONTROL_SMPK_hash_BCH_0 - 9

319 288	287 256	255 224	223 192
CONTROL_SMPK_hash_BCH_9	CONTROL_SMPK_hash_BCH_8	CONTROL_SMPK_hash_BCH_7	CONTROL_SMPK_hash_BCH_6
R-e	R-e	R-e	R-e

191 160	159 128	127 96
CONTROL_SMPK_hash_BCH_5	CONTROL_SMPK_hash_BCH_4	CONTROL_SMPK_hash_BCH_3
R-e	R-e	R-e

95 64	63 32	31 0
CONTROL_SMPK_hash_BCH_2	CONTROL_SMPK_hash_BCH_1	CONTROL_SMPK_hash_BCH_0
R-e	R-e	R-e

Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

Table 2-16 CONTROL_SMPK_hash_BCH_0 - 9 Register Field Description

Bits	Field	Description
319-288	CONTROL_SMPK_hash_BCH_9	Contains the appropriate 32-bit value of the 320-bit SMPK_hash_BCH (BCH encoded Secondary Master Public Key Hash) value.
287-256	CONTROL_SMPK_hash_BCH_8	
255-224	CONTROL_SMPK_hash_BCH_7	
223-192	CONTROL_SMPK_hash_BCH_6	
191-160	CONTROL_SMPK_hash_BCH_5	
159-128	CONTROL_SMPK_hash_BCH_4	
96-127	CONTROL_SMPK_hash_BCH_3	
64-95	CONTROL_SMPK_hash_BCH_2	
32-63	CONTROL_SMPK_hash_BCH_1	
31-0	CONTROL_SMPK_hash_BCH_0	
End of Table 2-16		

2.4.2 Key Manager Memory map

The memory map in Table 2-17 accommodates 2048 bits of key information. A KeyStone device may have multiple key managers to accommodate the keys in the device. For example some KeyStone devices have two key managers which memory maps to customer programmable OTP's in the device. See the device device data manual for base address (SEC_KEY_MGR).

Table 2-17 Key Manager Memory Map

Address Offset	Register Description
0x000 – 0x00C	Reserved
0x010 – 0x01C	Key 1
0x020 – 0x02C	Key 2
…	…
0x0F0 – 0x0FC	Key 15
0x100 – 0x10C	Key 16
0x110 – 0x3FF	Reserved

2.4.2.1 Key Information

The Key Information registers holds the actual key information from the OTP EFUSE. These registers are grouped in sets of 4 registers each corresponding to one 128-bit key. These registers contain no storage and are a direct reflection of the value in the OTP EFUSE. They are read-only with an access level based on the corresponding public / private enable.

Table 2-18 Key Information Register 1

Bits	Field	Description
31-0	KEY_INFORMATION	Bits 0 to 31 of Key 1. This is only valid when key_ready = 1

Table 2-19 Key Information Register 2

Bits	Field	Description
31-0	KEY_INFORMATION	Bits 32 to 63 of Key 1. This is only valid when key_ready = 1

Table 2-20 Key Information Register 3

Bits	Field	Description
31-0	KEY_INFORMATION	Bits 64 to 95 of Key 1. This is only valid when key_ready = 1

Table 2-21 Key Information Register 4

Bits	Field	Description
31-0	KEY_INFORMATION	Bits 96 to 127 of Key 1. This is only valid when key_ready = 1

This group of 4 registers is then replicated 16 times within the memory map.

2.4.3 Controller Memory Map

Table 2-22 shows the security controller memory map. It provides the security status to the system and is used in debug during secure boot. JTAG registers are used by the CPU for bidirectional communication between the JTAG and the external world.

Table 2-22 Security Controller Memory Map

Address Offset	Register
0x00 – 0x0C	Reserved
0x10	System Status Register
0x14	System Write-once Register
0x18 – 0x1C	Reserved
0x20	System Control Register
0x24	Reserved
0x28	System TAP Enable Register
0x2C	Reserved
0x30	JTAG Status Register
0x34	JTAG Read Debug Register
0x38	JTAG Write Debug Register
0x3C – 0x7C	Reserved

2.4.3.1 System Status Register

The system status register contains status information on the state of the security system. This is the only register that must be read to determine the security status of the entire system. It is a read-only register and some bits will be replicated in other registers. The fields in this register are shown in Figure 2-13:

Figure 2-13 System Status Register

31 – 20	19	18	17 – 16
Reserved	TAP_EN_REG_WRITTEN	Reserved	SYSTEM_REG_WRITTEN
R-0	R/WPOR-0	R-0	R/WPOR-0

15 – 12	11	10	9 – 6	5	4 – 3	2 – 0
Reserved	SEC_ROM_PASS	SEC_ROM_DONE	Reserved	SEC_OVERRIDE	Reserved	DEVICE_TYPE
R-0	R-0	R-0	R-0	R-0	R-0	R-es

Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

Table 2-23 System Status Register Description (Part 1 of 2)

Bits	Field	Description
31-20	Reserved	
19	TAP_EN_REG_WRITTEN	System TAP Enable Register Written. Indicates whether the System TAP Enable Register has been written. If written no changes can be made till next POR. 0 = Register not yet written 1 = Register written; no further writes allowed
18 - 17	Reserved	
16	SYSTEM_REG_WRITTEN	System Write-once Register Written. Indicates whether the System Write-once Register has been written. If written no changes can be made till next POR. 0 = Register not yet written 1 = Register written; no further writes allowed
15-12	Reserved	
11	SEC_ROM_PASS	Secure ROM Pass
10	SEC_ROM_DONE	Secure ROM Done
9-6	Reserved	

Table 2-23 System Status Register Description (Part 2 of 2)

Bits	Field	Description
5	SEC_OVERRIDE	Security Override
4-3	Reserved	
2-0	DEVICE_TYPE	Device Type. Indicates the device security type value. Value scanned from EFUSE during POR 2 = HIGH SECURITY (HS) 3 = GENERAL PURPOSE (GP) Other = BAD
End of Table 2-23		

2.4.3.2 System Write-once Register

The system write-once register provides control of the security system bits that should only be changed once during each POR. These bits are only allowed to be written by an allowed secure supervisor. The fields in this register are shown in Figure 2-14:

Figure 2-14 System Write-once Register

31	10	9	8	7	1	0
Reserved		SEC_ROM_PASS	SEC_ROM_DONE	Reserved		SEC_OVERRIDE
R-0		R/WPOR	R/WPOR	R-0		R/WPOR

Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

Table 2-24 System Write-once Register Description

Bits	Field	Description
31-10	Reserved	Always read as 0. Writes have no effect.
9	SEC_ROM_PASS	Secure ROM Pass. This bit indicates whether the secure ROM passed all integrity checks during the override sequence (this includes checks like PBIST). This bit is only valid when *sec_rom_done* = 1. 0 = Secure ROM failed checks 1 = Secure ROM passed checks
8	SEC_ROM_DONE	Secure ROM Done. This bit indicates whether the secure ROM contents have finished being validated as part of the override sequence. 0 = Secure ROM has not been validated 1 = All checks have been performed on Secure ROM
7-1	Reserved	
0	SEC_OVERRIDE	Security Override. This bit indicates whether a security override should take place. 0 = No security override 1 = Security override granted

2.4.3.3 System Control Register

The system control register provides control of the security system but is not as stringent as other bits within the security system. These bits may be read or written by any master. The fields in this register are shown in Figure 2-15:

Figure 2-15 System Control Register

31	1	0
Reserved		SEC_UPDATE
R-0		R/W-0

Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 2-25 System Control Register Description

Bits	Field	Description
31-1	Reserved	
0	SEC_UPDATE	Security Controller Update. This bit will cause the security controller to update its security outputs. A write of '1' to this field will cause the security controller to update its outputs. A write of '0' will have no effect. This field will always be read as '0'

2.4.3.4 System Test Access Port (TAP) Enable Register

The system test access port (TAP) enable register will enable or disable access to various modules that can be included on the JTAG scan chain. The fields in this register are shown in Figure 2-16:

Figure 2-16 System Tap Enable Register

31	16	15	0
TEST_TAP_EN		DBG_TAP_EN	
R/WPOR		R/WPOR	

Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

Table 2-26 System Tap Enable Register

Bits	Field	Description
31-16	TEST_TAP_EN	Test TAP Enable 1 = Test TAP can be selected by Debug Subsystem 0 = Test TAP not selectable by Debug Subsystem
15-0	DBG_TAP_EN	Debug TAP Enable 1 = Debug TAP can be selected by Debug Subsystem 0 = Debug TAP not selectable by Debug Subsystem

2.4.3.5 JTAG Status Register

The JTAG status register mirrors the values in the JTAG control data register of the JTAG interface. This register provides a path so that security override requests can be passed to the CPU. The fields in this register are shown in Figure 2-17:

Figure 2-17 JTAG Status Register

31	7	6	5	4	3	2	0
Reserved		SEC_ROM_PASS	SEC_ROM_DONE	Reserved		DEVICE_TYPE	
R-0		R-0	R-0	R-0		R-es	

Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

Table 2-27 JTAG Status Register Description (Part 1 of 2)

Bits	Field	Description
31-7	Reserved	
6	SEC_ROM_PASS	Secure ROM Pass. Value of '1' indicates that the secure ROM contents match the corresponding checksum value

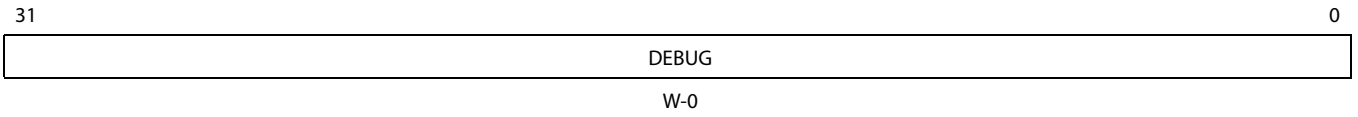
Table 2-27 JTAG Status Register Description (Part 2 of 2)

Bits	Field	Description
5	SEC_ROM_DONE	Secure ROM Done. Value of '1' indicates that the secure ROM contents have been checked against the corresponding checksum value and that the sec_rom_pass bit is valid.
4-3	Reserved	
2-0	DEVICE_TYPE	Device Type. Indicates the device security type value. 2 = HIGH SECURITY 3 = GENERAL PURPOSE Other = BAD

2.4.3.6 JTAG Read Debug Register

The JTAG read debug register mirrors the values in the JTAG read debug register of the JTAG interface. This register provides a bridge between the JTAG interface and the internal bus for debug purposes. This register is writable by the CPU so that the value can be read out on the JTAG port. Reads of this register by the CPU return zero. The fields in this register are shown in Figure 2-18:

Figure 2-18 JTAG Read Debug Data Register



Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

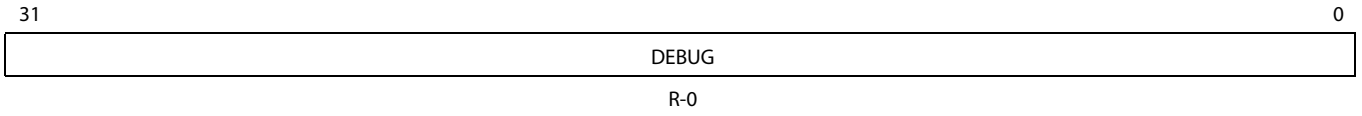
Table 2-28 JTAG Read Debug Register

Bits	Field	Description
31-0	DEBUG	Read Debug Data Register. The bits of this register are all software defined.

2.4.3.7 JTAG Write Debug Register

The JTAG write debug register mirrors the values in the JTAG write debug register of the JTAG interface. This register provides a bridge between the JTAG interface and the internal bus for debug purposes. This register is read-only by the CPU and will reflect the value shifted in on the JTAG port. The fields in this register are shown in Figure 2-19:

Figure 2-19 JTAG Write Debug Register



Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

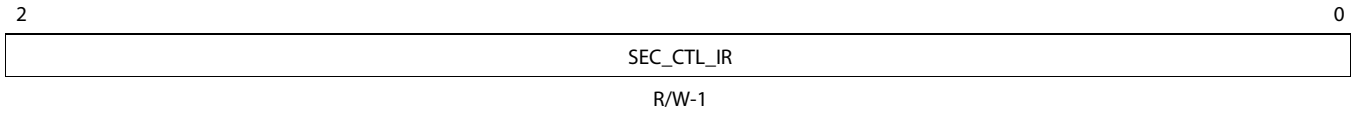
Table 2-29 JTAG Write Debug Register

Bits	Field	Description
31-0	DEBUG	Write Debug Data Register. The bits of this register are all software defined.

2.4.4 Security Controller JTAG Interface

The security controller provides JTAG interface to the control data registers and the debug and status registers. The table below shows the details of the IR register and the corresponding IR values. The security controller module has a 3-bit Instruction Register (IR) as defined in Figure 2-20:

Figure 2-20 Security Controller JTAG Interface Register



Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 2-30 Security Controller JTAG Interface Register Description

Bits	Field	Description
2-0	SEC_CTL_IR	Security Controller Instruction Register 0 = Reserved (32 bits in DR scan path) 1 = JTAG Control Data Register selected in scan path 2 = JTAG Read Debug Data Register selected in the DR scan path 3 = JTAG Write Debug Data Register selected in the DR scan path 4 to 6 = Reserved. 1 bit JTAG bypass bit will be selected in DR scan path 7 = Bypass. 1 bit JTAG bypass bit will be selected in the DR scan path.

2.4.5 JTAG Memory Map

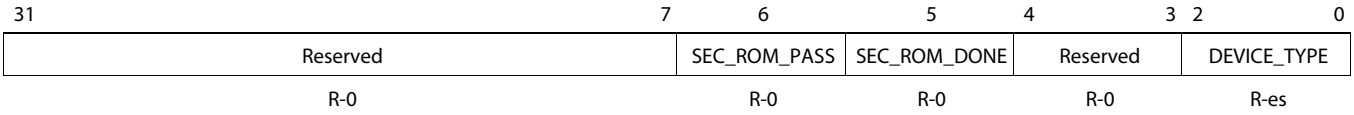
Table 2-31 JTAG Memory Map

Address Offset	Register
0x0	Reserved
0x1	JTAG Control Data Register (DR)
0x2	JTAG Read Debug Data Register (DR)
0x3	JTAG Write Debug Data Register (DR)
0x4 – 0x6	Reserved
0x7	Bypass

2.4.5.1 JTAG Control Data Register

This register contains status information about the security system as well as requests to move between security states. The fields in this register are shown in Figure 2-21:

Figure 2-21 JTAG Status Register



Legend: R = Read only; W = Write only; R/W = Read/Write; R/WPOR = Read, Write once Per POR; -n = value after reset; -x, value is indeterminate; -e = reset value from EFUSE; -es = reset value from std EFUSE; — see the device-specific data manual

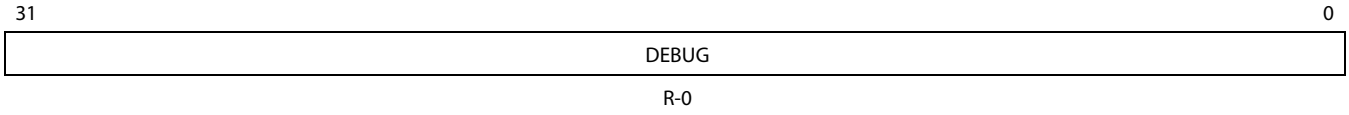
Table 2-32 JTAG Control Data Register Description

Bits	Field	Description
31-7	Reserved	
6	SEC_ROM_PASS	Secure ROM Pass. Value of '1' indicates that the secure ROM contents match the corresponding checksum value
5	SEC_ROM_DONE	Secure ROM Done. Value of 1 indicates that the secure ROM contents have been checked against the corresponding checksum value and that the sec_rom_pass bit is valid.
4-3	Reserved	
2-0	DEVICE_TYPE	Device Type. Indicates the device security type value. Value scanned from EFUSE during each hard chip reset. 2 = HIGH SECURITY 3 = GENERAL PURPOSE Other = BAD

2.4.5.2 JTAG Read Debug Data Register

In order to facilitate debug during the override sequences, this register will reflect the value written to the JTAG read debug register (0x34) via the CPU MMR interface. This value can then be shifted out of the JTAG interface by scanning out the 32 bits of this register. The fields in this register are shown in Figure 2-22:

Figure 2-22 JTAG Read Debug Data Register



Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

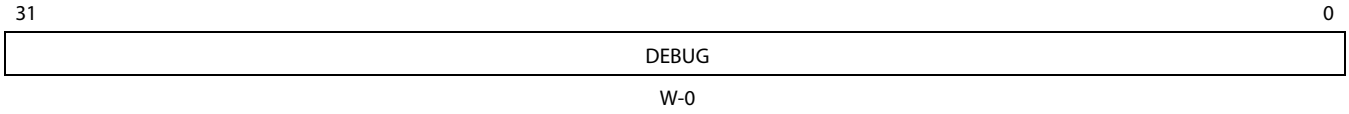
Table 2-33 JTAG Read Debug Data Register Description

Bits	Field	Description
31-0	DEBUG	Read Debug Data Register. The bits of this register are all software defined.

2.4.5.3 JTAG Write Debug Data Register

In order to facilitate debug during the override sequences, this register can have any value shifted in to it from the JTAG interface and the value will be reflected on the JTAG write debug Register (0x38) in the security controller memory map. This value can then be read by the CPU via the MMR interface. The fields in this register are shown in Figure 2-23:

Figure 2-23 JTAG Write Debug Data Register



Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 2-34 JTAG Write Debug Data Register Description

Bits	Field	Description
31-0	DEBUG	Write Debug Data Register. The bits of this register are all software defined.

2.5 Types of boot

A KeyStone device can support a maximum of up to four types of boot depending on the device type.

For GP devices, the KeyStone device supports:

- Public ROM boot when DSP is the boot master
- Public ROM boot when ARM is the boot master

For a HS device, the KeyStone device supports:

- Secure ROM boot when DSP is the boot master
- Secure ROM boot when ARM is the bootmaster

The chip-level boot ROM is used by DSP. ARM includes its own internal boot ROM just for ARM boot. Both DSPs and ARM need to read the DEVSTAT (BOOT_REG0) register to determine how to proceed with the boot. ARM does not support no-boot mode. ARM only supports boot through its internal ROM. The four types of boots are described in Table 2-35.

Table 2-35 Types of boots supported on different devices

Device type	HS	GP	Bad
Boot restrictions	Secure ROM based boot flow	Public ROM based boot flow	N/A

2.5.1 Public ROM Boot When DSP is the Boot Master

On a GP device, all DSPs and ARM are released from reset at the same time and begin execution from the respective ROM base address. Both ARM and DSP need to read the bootmode register inside the bootCFG module to determine who the boot master is. After the bootROM for ARM reads the bootmode to determine that DSP is the boot master, ARM clears a predefined memory value. When the value becomes non-zero it branches to that value. The chip bootROM reads the bootmode register to determine that DSP is the boot master, then DSP0 performs the boot process and secondary DSP cores execute an IDLE instruction. The boot writes a predefined memory location in corePac0 local L2 memory. After the boot process is completed, DSP0 interrupts the secondary DSP cores and ARM through the IPC register. Then the secondary DSP cores and the ARM complete boot operations and begin executing from the respective predefined locations in memory.

2.5.2 Public ROM Boot When ARM is the Boot Master

The only difference from the public ROM Boot when DSP is the boot master is that while ARM performs the boot process while both DSP0 and secondary DSP cores execute idle instructions. When ARM finishes the boot process, ARM sends interrupts to DSP0 and secondary DSP cores through IPC registers, and DSPs complete the boot operations and begin executing from the predefined memory locations. DSP cores branch to the address that resides in the last address in local L2.

2.5.3 Secure ROM Boot When DSP is the Boot Master

On HS devices, DSP0 is always the secure master. All DSPs and ARMs are released from reset at the same time and begin execution from the secure ROM address. Each DSP core configures the secure kernel for that core. Each DSP core then exits the secure state and enters the non-secure (public boot ROM). The image is received exactly as it is on non-secure devices, but it must be authenticated before it can be run. Only DSP0 will do the initial image authentication. This image can contain code for any and all DSP

and ARM cores. When operating in secure mode the DSPs do not execute an IDLE. The ARM does not execute a WFI. They both poll their *boot magic* addresses waiting for a non-zero value. The external memory is not configured by default, but it can be configured during the boot process. The MSMC is marked as securable, but with all memory initially non-secure. The initial ARM code is usually run from MSMC.

2.5.4 Secure ROM Boot When ARM is the Boot Master

On HS devices, DSP0 is always the secure master. All DSPs and ARMs are released from reset at the same time and begin execution from secure ROM address. The DSPs configure portions of their L2 RAMs to be public to enable bootload. DSP0 authenticates and decrypts the boot image for the ARMs, while the other DSPs and ARMs are in a poll loop. DSP0 sends interrupt to ARMs after the boot image is authenticated and decrypted.

When ARM is the boot master it performs the normal non-secure boot operation. Once the boot image has been received, however, it will not execute the image unless it has been authenticated and optionally decrypted. To do this it invokes DSP core 0 by writing a message into DSP memory that is non-secure, but marked as non-accessible by any master peripherals. The DSP runs the authentication and returns the result. If the image is authenticated, the ARM will execute this code. This code is now free to bring the DSPs out of their polling loop. The code that the DSPs run is not re-authenticated. The code that wakes them was re-authenticated.

When the ARM is the boot master, ARM0 is responsible for the boot process at the chip level. ARM0 executes the bootloader code necessary to support various device bootmodes. After it is released from reset, ARM0 immediately begins fetching from 0x4000_0000 for secure ROM boot or 0x4002_0000 for public ROM boot. The bootcfg module drives a ready output for each DSP and ARM that controls whether to boot the DSP or not. The bootcfg module will be parameterized such that all DSPs and ARM will boot immediately. Hence all DSPs and ARMs will have their ready go active when the boot config module is ready, i.e. after all the boot fields are latched.

The boot control interface consists of a block of three 32-bit registers, as shown in Table 2-36. See the device device data manual for the memory map.

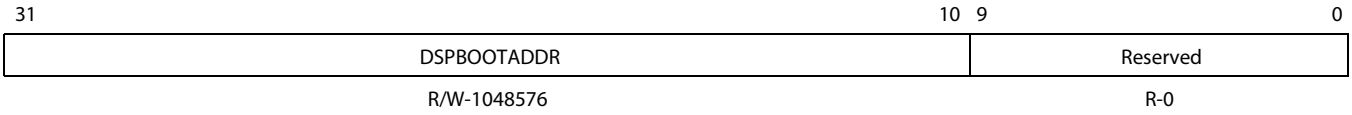
Table 2-36 Boot Control Registers

Register Name	Description
BOOTADDR_GEMx_REG	Boot Address of DSP, decoded by boot loader software for host boots
BOOTADDR_ARM_REG	Boot Address of ARM, decoded by boot loader software for host boots
BOOTCOMPLETE	Boot Complete register

2.5.5 Boot Address Register

The BOOTADDR_GEMx_REG register is a 32-bit read/write register. The most significant 22 bits of this register are used as the starting address for the DSP. For secure devices, the BOOTADDR_GEMx_REG register is reset by POR to the start address of the boot ROM (0x00100000). The fields in this register are shown in Figure 2-24:

Figure 2-24 Boot Address Register (BOOTADDR_GEMx_REG)



Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

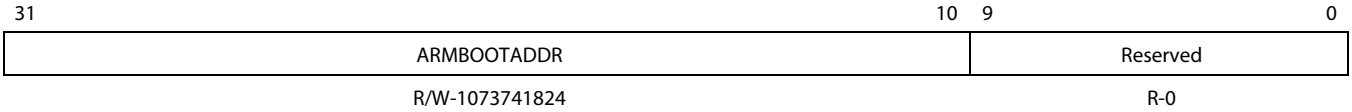
Table 2-37 Boot Address Register Description (BOOTADDR_GEMx_REG)

Bits	Field	Description
0-9	Reserved	
31-10	DSPBOOTADDR	DSP Boot Address. Only the 22 MSBs of this field are passed to DSP. The lower 10 bits are ignored

2.5.6 Boot Address Register ARM

There is a separate register for storing the BOOTADDR_ARM_REG. The ARM should be able to read this value. The BOOTADDR_ARM_REG register access should be permitted to any master or emulator when the device is nonsecure. The emulator cannot access this register in secure mode. The ARM boot address register uses the same format as the BOOTADDR_GEMx_REG register. The reset value for this register is 0x4002_0000 for non-secure device and 0x4000_0000 for secure device. The fields in this register are shown in Figure 2-25:

Figure 2-25 Boot Address Register (BOOTADDR_ARM_REG)



Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 2-38 Boot Address Register Description ARM

Bits	Field	Description
0-9	Reserved	
31-10	ARMBOOTADDR	ARM Boot Address

2.5.7 Boot Completion Register

The BOOTCOMPLETE register provides a communications interface between the boot host and the DSP for host boot modes. The register layout is as shown in Figure 2-26 "Boot Completion Register" on page 2-25. The BOOTCOMPLETE register is simply a 32-bit R/W register. The BOOTCOMPLETE register controls the BOOTCOMPLETE pin status. The purpose is to indicate the completion of the ROM booting process. The BCx bit indicates the boot complete status of DSPx. The BC_ARM field indicates the boot complete status for ARM. All the BCx bits and the BC_ARM bit will be sticky bits, that is they can be set only once by the software after device reset and they will be cleared to 0 on all device resets. Boot ROM code will be implemented such that each core will set its corresponding BCx bit immediately before

branching to the predefined location in memory. Some devices have a dedicated device output pin BOOTCOMPLETE to indicate the external system that the Device ROM booting process is complete. The AND result of the boot complete bits (BCx in BOOTCOMPLETE register) of all cores is the BOOTCOMPLETE pin status. So this pin status will be high when the ROM code marks ALL BCx bits to 1. It will be low after the device reset. The fields in this register are shown in Figure 2-26.

Figure 2-26 Boot Completion Register

31 9	8	7 4	3 0
Reserved	BC_ARM	Reserved	BC_GEM
R-0	R/W-0	R-0	R/W-0

Legend: R = Read only; W = Write only; R/W = Read/Write; -n = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 2-39 Boot Completion Register Description

Bits	Field	Description
31-9	Reserved	
8	BC_ARM	Boot Complete flag from ARM 0 = ARM boot is not yet complete 1 = ARM boot is complete
7-4	Reserved	
3-0	BC_GEM	Boot Complete for Corepac x (x = number of CorePacs in a device) 0 = Corepac x boot is not yet complete 1 = Corepac x boot is complete

2.6 Security Override Sequences

All HS devices can be run in NONSECURE mode after initially booting in SECURE mode using this sequence.

2.6.1 SECURE to NONSECURE after boot.

A secure device can used in non secure mode after initial boot to prevent run-time overhead introduced due to security. As part of the over-ride sequence, all internal memories that could contain sensitive customer code information (see the device for which memories this includes) are purged. For example, the SECURE to NONSECURE override sequence in a KeyStone device will purge all CorePac and MSMC RAM information. This sequence will be a mixture of hardware and software and imposes some requirements on the bootROM contents. Refer to the *Bootloader User Guide* in "Related Documentation from Texas Instruments" on page ø-viii for more details.

2.6.2 Secure to Non-Secure Override Sequence

If running in secure mode, with the secure kernel active, you can request a override to non-secure mode by using the SK_switchNonSec() API call. See the *Secure Kernel Users Guide* for more information on the secure kernel APIs.

2.7 JTAG Disable

Some KeyStone devices will support a customer programmable JTAG disable feature. This feature may be enabled on some parts (see the device device data manual for details). If the feature is enabled, there are four bits within the customer control field that will determine the state of the JTAG interface. The JTAG disable field uses double bit redundancy. The value of the TAP enables will be determined by the Table 2-40.

Table 2-40 JTAG Disable values

Device Type	JTAG Disable [3:0] Value	Description
[BAD]	[xxxx]	Bad Value Debug Tap Enable = [0xFFFF] Test Tap Enable = [0x0000]
[GP, HS]	[0000]	JTAG Open Debug TAP Enables = [0xFFFF] Test TAP Enables = [0xFFFF]
[GP, HS]	[0000, 0010, 0001]	JTAG Protected Debug TAP Enables = [sec_dbg_tapenable_po] Test TAP Enables = [sec_test_tapenable_po]
[GP, HS]	[11XX,10XX,01XX]	JTAG Disabled Debug TAP Enables = [0x0000] Test TAP Enables = [0x000C]

2.7.1 JTAG Open

All debug and test TAPs are forced open. This allows complete accessibility to on-chip resources. This is the default state of the JTAG_DISABLE field within the customer programmable EFUSE.

2.7.2 JTAG Protected

All debug and test TAPs receive the value exported from the security controller. In this state, the TAPs will either be disabled by default (HS device) or enabled by default (GP device). For HS devices, the ROM will implement a certificate based protection scheme to allow the JTAG interface to be unlocked with a properly signed certificate.

2.7.3 JTAG Disabled

All TAPs are closed and cannot be opened. This allows for complete protection against JTAG based attacks. If JTAG is disabled, there is no way to perform Failure Analysis on the device.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265