Applications Processor Security Reference Manual for i.MX 6SoloLite

Document Number: IMX6SLSRM
Rev. 0, 03/2013

freescale™

Contents

Section number **Title** **Page**

Chapter 1
Security Overview

Chapter 2
Security System Integration

Chapter 3
Central Security Unit (CSU)

Chapter 4
Data Co-Processor (DCP)

Chapter 5
Random Number Generator (RNGB)

Chapter 6
Secure Non-Volatile Storage (SNVS)

Chapter 7
System Boot

Chapter 1
Security Overview

1.1 Chapter overview

This chapter provides an overview of the following chip security components, explaining the purpose and features of each of them.

- TrustZone (TZ) architecture including: security extensions in the Cortex-A9 processors, Generic Interrupt Controller (GIC), TrustZone Watchdog (TZ WDOG), on-chip RAM (OCRAM) and Trust Zone Address Space Controller (TZASC)
- High Assurance Boot (HAB) feature in the system boot
- Secure Non-volatile Storage (SNVS) with security monitor, key storage and real-time clock
- Data Coprocessor with Cryptographic Acceleration
- On-chip One-time Programmable Element Controller (OCOTP_CTRL) with on-chip electrical fuse arrays
- Central Security Unit (CSU)
- Random Number Generator (RNGB)
- System JTAG Controller (SJC) with secure debug
- Locked mode in Smart Direct Memory Access Controller (SDMA)

Detailed descriptions of each component are found either in the remaining chapters of this security reference manual or in *i.MX 6SoloLite Multimedia Applications Processor Reference Manual* (IMX6SLRM).

1.2 Feature summary

The following figure shows a simplified diagram of the security subsystem.

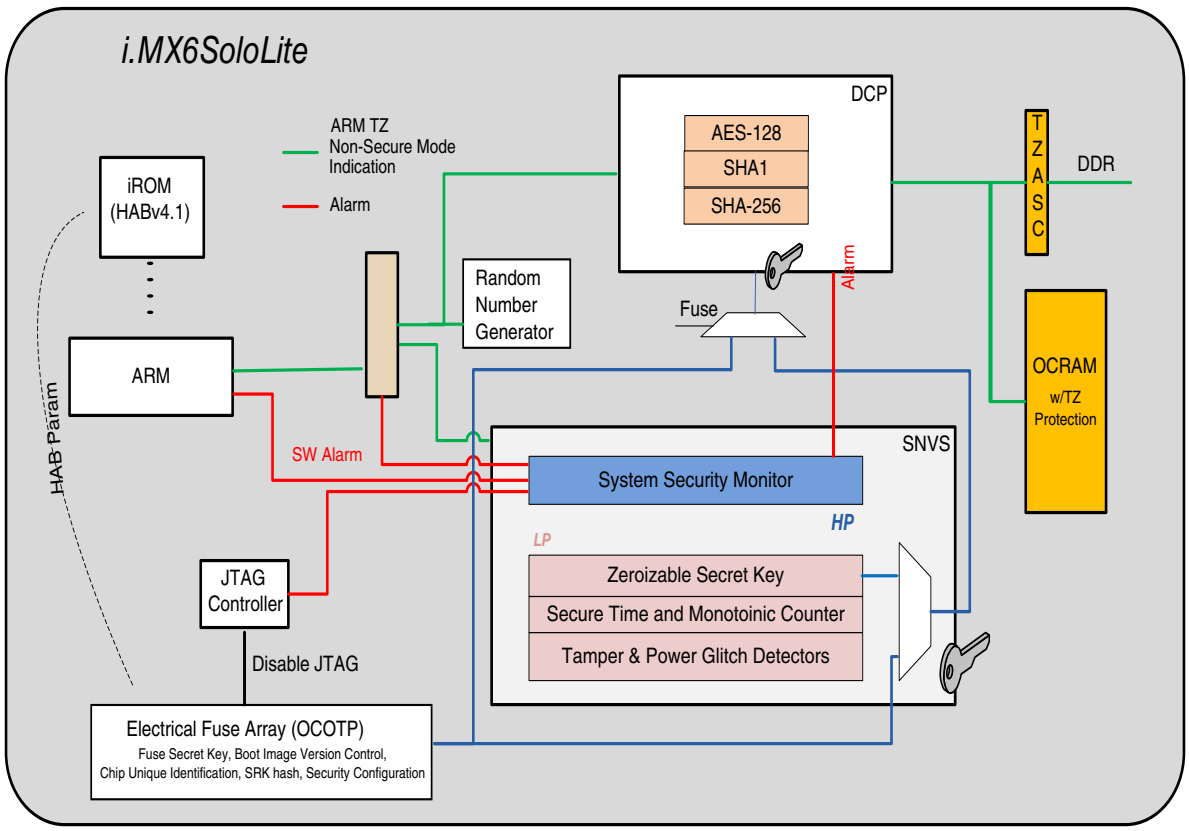


Figure 1-1. Security subsystem (simplified)

This diagram represents one example of the use of CSU.

All platforms built using this chip share a general need for security, though specific security requirements vary greatly from platform to platform. For example, portable consumer devices need to protect a different type and cost of assets than do automotive or industrial platforms. Likewise, each market must protect against different kinds of attacks. Platform designers need an appropriate set of counter measures to meet their specific platform's security needs.

To help platform designers meet the requirements for each market, the chip incorporates a range of security features. Most of these features provide protection against specific kinds of attack, and can be configured for different levels according to the required degree of protection. These features are designed to work together or independently. They can also be integrated with appropriate software to create defensive layers. In addition, the chip includes a general-purpose accelerator that enhances the performance of selected industry-standard cryptographic algorithms.

The security features include:

- ARM TrustZone Architecture, a trusted execution environment for security-critical software

    - Hardware-assisted virtualization-two virtual machines: Secure and Normal Worlds (processor modes)
    - Hardware firewalls
        - Control access from CPU and DMA peripherals to on-chip peripherals and to both on-chip and off-chip memory
        - Interrupt separation
        - Secure storage separation
        - Cryptographic separation
- Secure High Assurance Boot
    - Security library embedded in tamper-proof on-chip ROM
    - Authenticated boot, which protects against unauthorized software

NOTE

i.MX 6SoloLite does not support encrypted boot.

        - Verification of the code signature during boot
        - RSA-1024/2048/3072/4096 keys anchored to OTP fingerprint (SHA-256)
    - Runs every time chip is reset
    - Image version control/image revocation (on-chip OTP-based)
- Secure storage
    - Off-chip storage protection using AES-128 and the chip's unique hardware-only key
- Hardware cryptographic accelerators
    - Symmetric: AES-128,
    - Hash message digest: SHA-1, SHA-256,
- True and pseudorandom number generator
- On-chip secure real-time clock with autonomous power domain
- Secure debugging
    - Configurable protection against unauthorized JTAG manipulation
    - Three security levels plus complete JTAG disable
    - Support for JTAG port secure reopening for field return debugging
- Universal unique ID
- Electrical fuses (OTP Memory)
- Physical tamper detection
    - Tamper input signal available for cover seal and power glitch detection
    - Hardware and software tamper response

1.3 TrustZone architecture

The TrustZone architecture provides a trusted execution environment for security-critical software. Software running in this environment is protected against attacks from potentially compromised platform software, including applications, services, drivers, and even the operating system itself. The TrustZone hardware protects the confidentiality and integrity of both security services and sensitive data. Furthermore, security services cannot be starved of access to processor resources or hijacked by uncontrolled interrupts. TrustZone allows security-critical software to coexist with a rich platform software environment.

The following features work together to create the TrustZone architecture.

- TrustZone security extensions in the ARM core (see Figure 1-2) duplicate the user, supervisor and other privileged modes of the processor in a Secure World and a Normal World. Security services execute in Secure World under the control of a security kernel, while normal services and applications run in Normal World with a rich operating system. TrustZone also provides a monitor mode, to which Normal World operating system traps when security services are required.
- TrustZone extensions to the MMU and memory caches separate Secure World and Normal World memory spaces (including memory-mapped peripherals). All read, write and instruction fetch operations from the ARM core indicate the current world, and the page tables and cache lines can be isolated from each other. With this, the security kernel controls access to Secure World memory and peripherals from Normal World software (even the operating system).
- TrustZone Access Controller (TZASC) separates Secure World and Normal World external memory spaces for other bus masters such as DMA-equipped peripherals. More details on TZASC are found in TrustZone Address Space Controller (TZASC).
- The on-chip RAM (OCRAM) controller separates secure world and normal world internal memory spaces for other bus masters. More details on OCRAM are found in the OCRAM chapter of the *i.MX 6SoloLite Multimedia Applications Processor Reference Manual (IMX6SLRM)*.
- Central Security Unit (CSU) and the peripheral bridge (AIPS-TZ) separates Secure World and Normal World peripheral address spaces for other bus masters such as DMA-equipped peripherals. More details on CSU are found in Central Security Unit (CSU). Some information on the AIPS-TZ is in this document. Details on AIPS-TZ are found in *i.MX 6SoloLite Multimedia Applications Processor Reference Manual (IMX6SLRM)*.
- Generic Interrupt Controller (GIC) collects interrupt requests from all sources and provides the interrupt interface to the core. Each interrupt source can be configured dynamically as a normal or a secure interrupt by Secure World software. The context

switch to handle a normal or secure interrupt when executing in Normal World or Secure World is configured in the ARM core.

- TrustZone Watchdog (TZ WDOG) protects against Normal World software preventing a switch back to Secure World, thereby starving security services of access to the core. Once TZ WDOG is activated, Secure World software must service it on a periodic basis. If servicing does not take place before the configured timeout, TZ WDOG asserts a secure interrupt that forces a switch to Secure World. If it is still not served, TZ WDOG asserts a security violation alarm. TZ WDOG cannot be programmed or deactivated from Normal World. More details on TZ WDOG are found in TrustZone Watchdog (TZ WDOG)

The MMU and the TrustZone architecture are capable of distinguishing between four different code execution modes:

- Code executing in Normal World mode:
  - Code running in kernel mode (also called supervisor mode or privileged mode)
  - Code running in user mode
- Code executing in TrustZone Secure World mode:
  - Code running in TrustZone kernel mode (also called supervisor mode or privileged mode)
  - Code running in TrustZone user mode

The following figure shows the four execution modes.

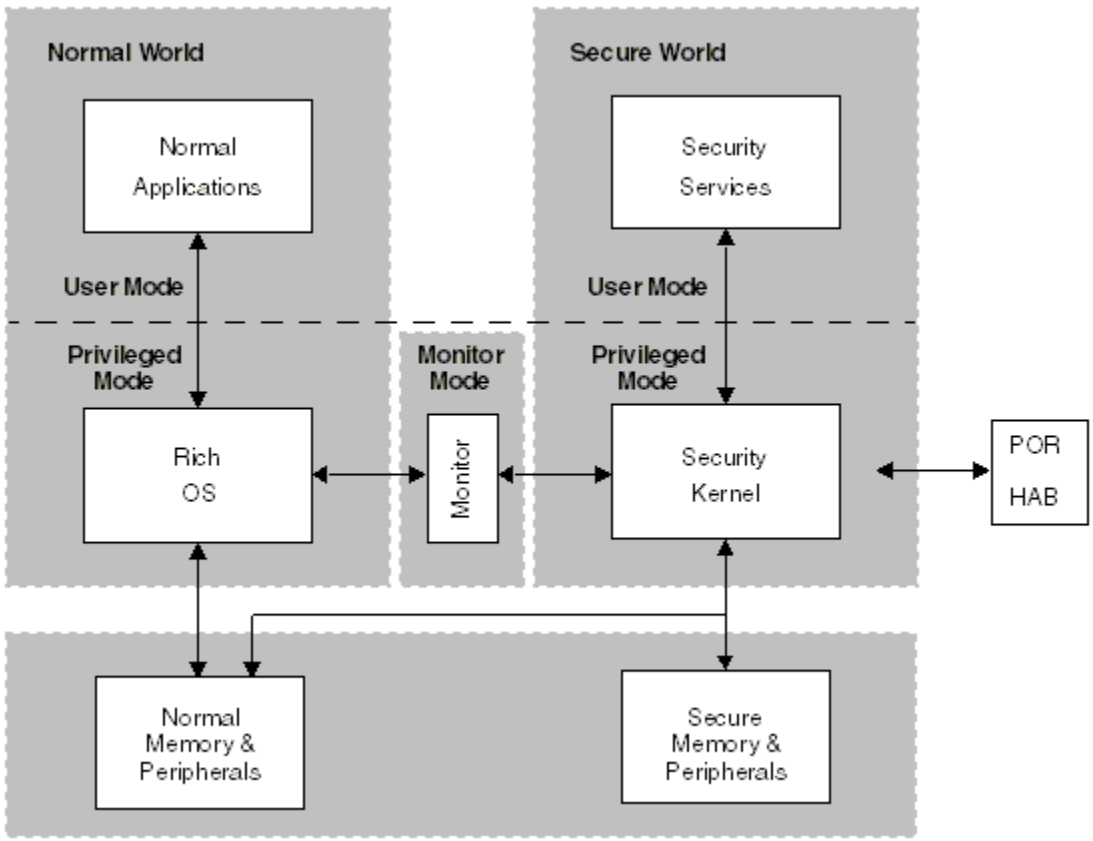


Figure 1-2. TrustZone security extensions

The TrustZone architecture also provides hardware support for a limited virtualization of the ARM core. In the limited virtualization, there are two guest virtual machines: one secure and the other non-secure.

The TrustZone architecture is integrated with other security features for trusted execution support, as follows:

- After power on reset (POR), the ARM core is in Secure World, all interrupts are secure interrupts, all bus masters are configured as Secure World masters, and all bus slaves can be accessed by Secure World bus masters. This has two implications:
  - If a trusted execution environment is not required, there is no need to switch to Normal World. In this case, the system is backwards compatible with a non-TrustZone system. All platform software runs in Secure World without modification for TrustZone.
  - If enabled, HAB executes in Secure World to authenticate either the security kernel (on a platform using TrustZone) or the normal operating system bootloader.
- The CSU, AIPS, TZASC and OCRAM controller enforce configurable core access rights to peripherals and memory from Secure and Normal Worlds.

- The CSU configures other bus masters to make either Secure or Normal World accesses.
- Secure storage separation (DCP/SNVS)
- If not serviced, the TZ WDOG security violation alarm goes to the SNVS.

For more details on the components of the chip TrustZone architecture, see the descriptions of the Cortex A9 Platform in the *i.MX 6SoloLite Multimedia Applications Processor Reference Manual (IMX6SLRM)*.

1.4 High Assurance Boot (HAB)

HAB, which is the high assurance boot feature in the system boot ROM, protects the platform from executing unauthorized software (malware) during the boot sequence. When unauthorized software is permitted to gain control of the boot sequence, it can be exploited for a variety of goals, such as exposing stored secrets; circumventing access controls to sensitive data, services, or networks; or repurposing the platform. Unauthorized software can enter the platform during upgrades or reprovisioning, or when booting from USB connections or removable devices.

HAB protects against unauthorized software by:

- Using digital signatures to recognize authentic software. This allows the user to boot the device to a known initial state, running software signed by the device manufacturer.

1.4.1 HAB process flow

Figure below shows the flow for creating and verifying digital signatures. The top half of this figure shows the signing process, which is performed once off-chip. The bottom half shows the verification process performed on-chip during every system boot.

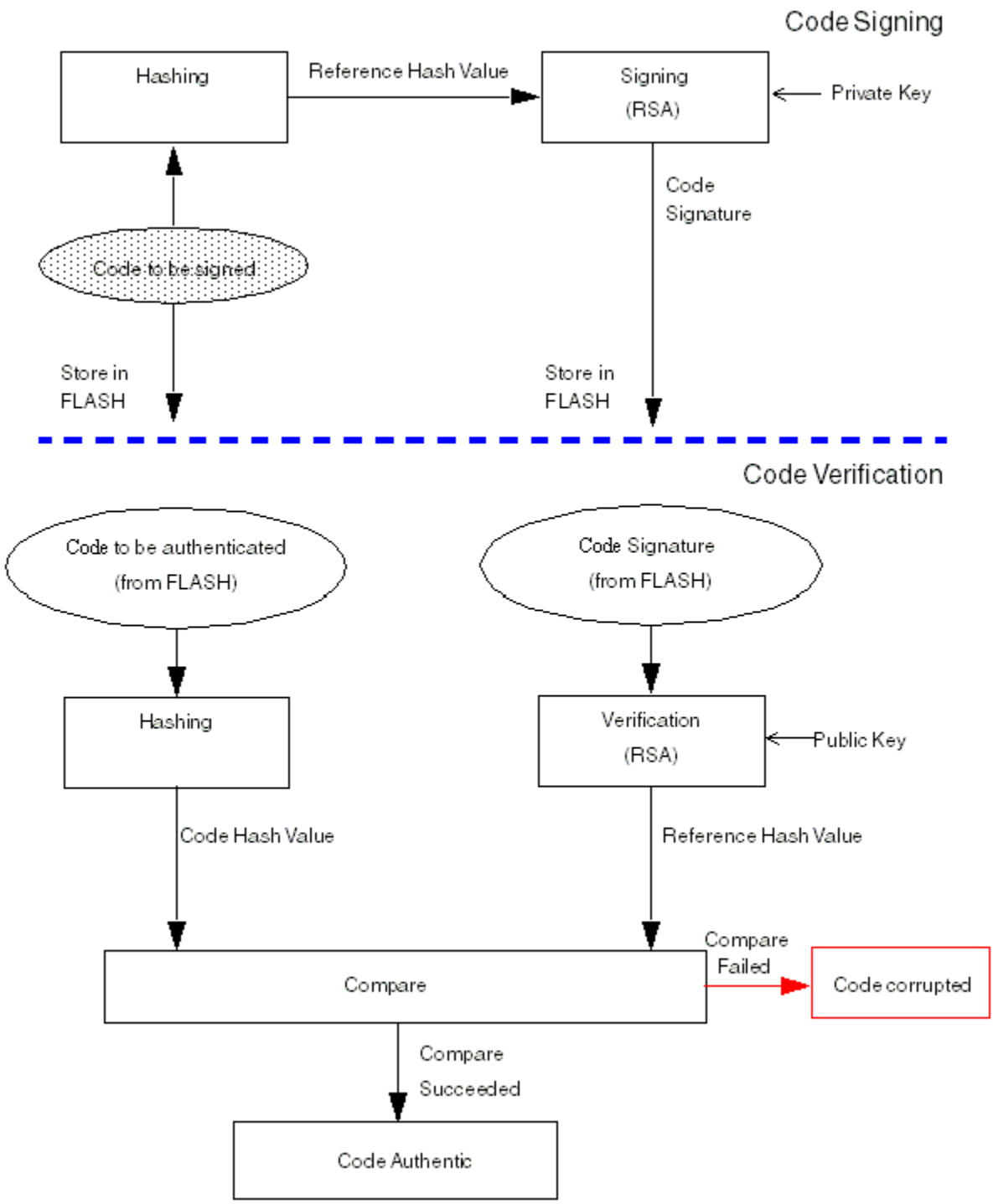


Figure 1-3. Code signing and authentication processes

The original software is programmed in flash memory (or any other boot device) along with the signature. HAB uses a public key to recover the reference hash value from the signature; it then compares the reference hash value to a current hash value produced

from the software in flash. If the contents of the flash have been modified either intentionally or unintentionally, the two hash values do not match and the verification fails.

1.4.2 HAB feature summary

HAB incorporates:

- Enforced internal boot via on-chip masked ROM
- Authentication of software loaded from any boot device (including USB download)
- CMS PKCS#1 signature verification using RSA public keys (1024 bit to 4096 bit) and the SHA-256 hash algorithm
- Public key infrastructure (PKI) support using X.509v3 certificates
- Root public key fingerprint in manufacturer-programmable on-chip fuses
- Multiple root public keys with revocation by fuses
- Initialization of other security components
- Authenticated USB download fallover on any security failure
- Open configuration for development purposes and non-secure platforms
- Closed configuration for shipping secure platforms

On the chip, HAB is integrated with other security features as follows:

- HAB executes in the TrustZone Secure World.
- HAB initializes the SNVS security monitor state machine. Successful secure boot with HAB is required for platform software to gain use of the DCP master secret key selected by SNVS.
- HAB reads the root public key fingerprint, revocation mask, and security configuration from the OCOTP_CTRL.
- HAB initializes the CSU.
- HAB can use DCP to accelerate hash calculations.

See the *i.MX 6SoloLite Multimedia Applications Processor Reference Manual (IMX6SLRM)* for more information on HAB.

1.5 Secure Non-volatile Storage Module (SNVS)

- Provides a non-volatile real-time clock maintained by a coin cell during system power down for use in both secure and non-secure platforms
- Protects the real-time clock against rollback attacks in time-sensitive protocols such as DRM and PKI

- Deters replay attacks in time-independent protocols such as certificate or firmware revocation
- Handles tamper detection and tamper reaction to defend sensitive data and operations against compromise, both at run-time and during system power-down
- Controls the access to the OTP master secret key used by DCP to protect confidential data in off-chip storage
- Provides non-volatile highly protected storage for an alternative master secret key

1.5.1 SNVS architecture

SNVS is partitioned into two sections: a low power part (SNVS_LP) and a high power part (SNVS_HP)

The SNVS_LP block is in the always powered up domain. It is isolated from the rest of the logic by means of isolation cells, which are library-instantiated cells that insure that the powered up logic is not corrupted when power goes down in the rest of the chip.

SNVS_LP has the following functional units:

- Zeroizable Master Key
- Secure non-rollover real time counter with alarm
- Non-rollover monotonic counter
- Power glitch detector
- Tamper detection monitor
- General-purpose register
- Control and status registers

SNVS_HP is in the chip power supply domain. SNVS_HP provides an interface between SNVS_LP and the rest of the system. Access to SNVS_LP registers can only be gained through the SNVS_HP when it is powered up according to access permission policy.

SNVS_HP has the following functional units:

- IP bus interface
- SNVS_LP interface
- System Security Monitor (SSM)
- Zeroizable Master Key programming mechanism
- Master Key control block
- Non-secure real time counter with alarm
- Control and status registers

1.6 Data Co-Processor (DCP)

For security purposes, the Data Co-Processor (DCP) provides a hardware acceleration for cryptographic algorithms. The features of DCP are:

- Encryption Algorithms: AES-128 (ECB and CBC modes)
- Hashing Algorithms: SHA-1 and SHA-256
- Key selection from SNVS, DCP internal key storage or general memory
- Internal Memory for storage of up to four AES-128 keys. Once a key is written to a key slot it may only be read by the DCP AES-128 engine.
- IP slave interface
- DMA

1.7 OCOTP_CTRL

OCOTP_CTRL provides the primary user-visible mechanism for interfacing with on-chip fuses. These fuses' uses include the following:

- Unique chip identifiers
- Mask revision numbers
- Cryptographic keys
- Security configuration
- Boot characteristics
- Various control signals requiring permanent non-volatility.

For security purposes, the fuses protect the confidentiality or integrity of critical security data against both software attacks and board-level hardware attacks.

OCOTP_CTRL provides:

- Shadow cache of fuse values loaded at reset prior to system boot.
- Ability to read and override fuse values in shadow cache (doesn't affect fuse element).
- Ability to read fuses directly (ignoring shadow cache).
- Ability to write (program) fuses by software or JTAG.
- Fuses and shadow cache bits enforcing read-protect, override-protect, and write-protect.
- Lock fuses for selected fuse fields
- Scan protection
- Volatile software-accessible signals which can be used for software control of hardware elements not requiring non-volatility.

1.8 Central Security Unit (CSU)

Central Security Unit (CSU) sets access control policies between bus masters and bus slaves, allowing peripherals to be separated into distinct security domains. This protects against indirect unauthorized access to data such as occurs when software programs a DMA bus master to access addresses that the software itself is prohibited from accessing directly. Configuring DMA bus master privileges in CSU consistently with software privileges defends against such indirect unauthorized access.

CSU provides:

- Configuration of peripheral access permissions for those peripherals unable to control their own access permissions
- Configuration of bus master privileges for those bus masters unable to control their own privileges
- TrustZone support to enhance non-TrustZone aware bus masters
- Optional locking of individual CSU settings until the next power-on reset

On the chip, CSU interfaces with:

- ARM TrustZone architecture to assign peripherals to Secure World and Normal World domains

1.9 AHB to IP Peripheral Bridge (AIPSTZ)

The AIPSTZ bridge provides programmable access protections for both masters and peripherals. It allows the privilege level of a master to be overridden, forcing it to user-mode privilege, and allows masters to be designated as trusted or untrusted.

Peripherals may require supervisor privilege level for access, may restrict access to a trusted master only, and may be write-protected. IP bus peripherals are subject to access control policies set in both CSU registers and AIPSTZ registers. An access is blocked if it is denied by either policy.

1.10 System JTAG Controller (SJC)

The JTAG port provides debug access to hardware blocks, including the ARM processor and the system bus. This allows program control and manipulation as well as visibility to the chip peripherals and memory. The JTAG port must be accessible during initial

platform development, manufacturing tests, and general troubleshooting. Given its capabilities, JTAG manipulation is a known attack vector for accessing sensitive data and gaining control over software execution. System JTAG Controller (SJC) protects against the whole range of attacks based on unauthorized JTAG manipulation. It also provides a JTAG port that conforms to IEEE 1149.1 and IEEE 1149.6 (AC) standards for BSR (boundary scan) testing.

SJC provides the following security levels:

- JTAG Disabled-JTAG use is permanently blocked.
- No-Debug-All security sensitive JTAG features are permanently blocked.
- Secure JTAG-JTAG use is restricted (as in the No-Debug level) unless a secret-key challenge/response protocol is successfully executed.
- JTAG Enabled-JTAG use is unrestricted.

Security levels are selected via e-fuse configuration.

1.10.1 Scan protection

The chip includes further scan protection logic for SJC modes where JTAG use is allowed. This ensures that access to critical security values is protected as follows.

- The chip is reset upon entering scan mode.
- All modules are reset two clock cycles before receiving a scan enable indication.
- The chip cannot exit scan mode without reset.
- Security modules, including SNVS, CSU, and OCOTP_CTRL, have additional scan protection logic to protect sensitive internal data and functionality.

See the "System JTAG Controller (SJC)" chapter in *i.MX 6SoloLite Multimedia Applications Processor Reference Manual (IMX6SLRM)* for more information on the SJC.

1.11 TrustZone Address Space Controller (TZASC)

TrustZone Address Space Controller (TZASC) protects security-sensitive software and data in a trusted execution environment against potentially compromised software running on the platform.

The TZASC:

- Supports 2, 4, 8, or 16 independent address regions.

- Uses access controls that are independently programmable for each address region and permits data transfers between master and slave only if the security status of the system bus transaction matches the security settings of the memory region it addresses.
- Allows locking of sensitive registers.
- Host interrupt may be programmed to signal attempted access control violations

For a detailed specification of TZASC see the ARM® Infocenter documentation center, Revision: r0p1 *CoreLink™ TrustZone Address Space Controller TZC-380 Technical Reference Manual*.

1.12 Smart Direct Memory Access Controller (SDMA)

Smart Direct Memory Access Controller (SDMA) enables data transfers between peripheral I/O devices and internal/external memories, which maximizes system performance by offloading the CPU in dynamic data routing. Because it is software-programmable, if left unprotected, SDMA could be abused by malicious software to gain indirect access to addresses that the software itself is prohibited from accessing directly. However, SDMA can be configured to protect against this abuse.

SDMA supports two security levels, which are configurable until the next reset by a write-once lock bit:

- In open mode, the processor has full control to load scripts and context into SDMA RAM and modify SDMA registers. This is the default mode.
- In locked mode, selected SDMA registers become read-only to prevent modification of software reset, exception and debug handling. Scripts and their context cannot be loaded into the SDMA RAM anymore.

On the chip, SDMA privileges are configured in the CSU as a further precaution against software abuse.

For more details on the SDMA, see the "System DMA (SDMA)" chapter in the *i.MX 6SoloLite Multimedia Applications Processor Reference Manual (IMX6SLRM)*.

1.13 TrustZone Watchdog (TZ WDOG)

The TrustZone Watchdog (TZ WDOG) timer module protects against denial of service attacks on Secure World software by Normal World software. TZ starvation is a situation where the normal OS prevents switching to the TZ mode. This situation is undesirable because it can compromise the system's security.

Once the TZ WDOG module is activated, it must be serviced by Secure World software on a periodic basis. If servicing does not take place, the timer expires and the TZ WDOG asserts a secure interrupt that forces a processor switch to Secure World. If this interrupt is not serviced, the TZ WDOG asserts a security violation alarm to the SNVS. The TZ WDOG module cannot be programmed or deactivated by Normal World software.

TZ WDOG is another instantiation of the system WDOG. It enables the following features:

- Time-out periods from 0.5 seconds up to 128 seconds
- Time resolution of 0.5 seconds
- Configurable counters to run or stop during low power modes
- Configurable counters to run or stop during debug mode
- Two event time points: one for TrustZone interrupt assertion and one for security alarm assertion

Chapter 2
Security System Integration

2.1 Master ID allocation

The following table summarizes the master IDs for all system master modules.

Table 2-1. Master IDs

Module	Master ID
Cortex A9	01b
SDMA	11b
All others	00b

2.2 System-level SNVS connections

2.2.1 SNVS clock tamper input

The system provides automatic detection of external SRTC clock state and provides an alternative internal clock source when a failure is detected. When the external clock detection fails, the system switches automatically to an internally generated clock from a ring oscillator on the i.MX6 processor. Once the clock on the RTC_xtali appears again, the system switches back to the external clock. Thus the clock tamper detection function is reduced to detection of power failure in SNVS power domain or SNVS reset.

2.2.2 System security violation alarm signals monitored by SNVS

The SNVS supports six system security violation alarm inputs, as shown in the following table. This allocation is related to HPSVCR and LPSVCR SNVS registers.

Table 2-2. SNVS system security violation alarm input signals

SNVS registers			Source	Description
HPSVSR register bit field	HPSVCR register bit field	LPSVCR register bit field		
SEC_VIO1	SV_CFG1	SV_EN1	SJC	JTAG active
SEC_VIO2	SV_CFG2	SV_EN2	WDOG2	WatchDOG 2 reset
SEC_VIO3	SV_CFG3	SV_EN3	(Reserved)	-
SEC_VIO4	SV_CFG4	SV_EN4	SRC	Internal Boot
SEC_VIO5	SV_CFG5	SV_EN5	IOMUX	External Tamper Detection pad

2.3 Security access error

The system slave modules can be configured to return a bus access error when a security-violating access is detected, using the SEC_ERR_RESP bit (GPR10[11] register):

- When set, the slave modules return a bus error indication on a non-proper security level access.
- When cleared, the operation does not proceed on a non-proper security level access, but the slave modules do not indicate an error.

The bit is set by default, enabling the error indications. SEC_RRR_RESP itself can be locked, preventing further modifications by LOCK_SEC_ERR_RESP bit (GPR10[27]) to assure the system security integrity.

For more information, see the "General-Purpose Registers 10" section *i.MX 6SoloLite Multimedia Applications Processor Reference Manual (IMX6SLRM)*.

2.4 OCRAM TrustZone support

OCRAM supports TrustZone and non-TrustZone accesses to internal, on-chip RAM. There is an option to configure TrustZone only access region.

When OCRAM_TZSECURE_REGION[SECURE_ENBL] bit in the OCRAM module is set, the STARTADDR and ENDADDR bit-fields in this register establish the region of OCRAM that can only be accessed (both read and write) according to the execution mode policy defined in the "Execution Mode Access Policy" section of the CSU chapter. If this bit is cleared, the entire OCRAM can be accessed in either secure or non-secure mode.

The TrustZone bits are described in the "Programmable Registers" section *i.MX 6SoloLite Multimedia Applications Processor Reference Manual (IMX6SLRM)*.

NOTE

The ENDADDR is not configurable. Its value is the last address of the OCRAM space. The STARTADDR granularity is of 4 Kbytes.

The following figure shows OCRAM schematic connectivity.

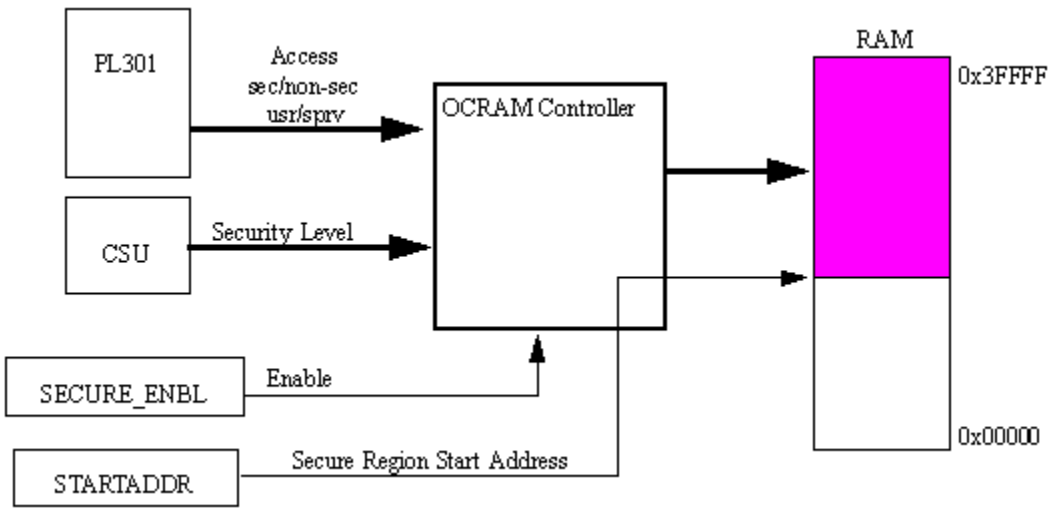


Figure 2-1. OCRAM schematic connectivity

2.5 WatchDOG mechanism

The chip has two WDOG modules: WDOG1 and WDOG2 (TZ). Both modules are disabled by default after reset. WDOG1 is configured during the boot whereas WDOG2 is dedicated for secure world purposes and is only activated by TrustZone software if required.

The WDOG module operates as follow:

- If servicing does not take place, the timer times out and asserts the internal system reset signal (wdog_rst_B), which goes to SRC, the system reset controller.
- Interrupt can be generated before the counter actually times out.
- wdog_rst_B signal can be activated by software.
- There is a power down counter that is enabled out of any reset. This counter has a fixed time out period of 16 seconds after which it asserts the WDOG_B signal.

2.6 Security configuration

Following figure illustrates the typical device security configuration lifecycle, starting from IC fabrication and continuing with OEM development and assembly, through to the final product in the end user's hands. It also shows the option for field return debugging and re-testing at either the OEM facilities or Freescale. Note that Field Return configuration is required for Freescale to run test patterns even on non-secure products (Open configuration).

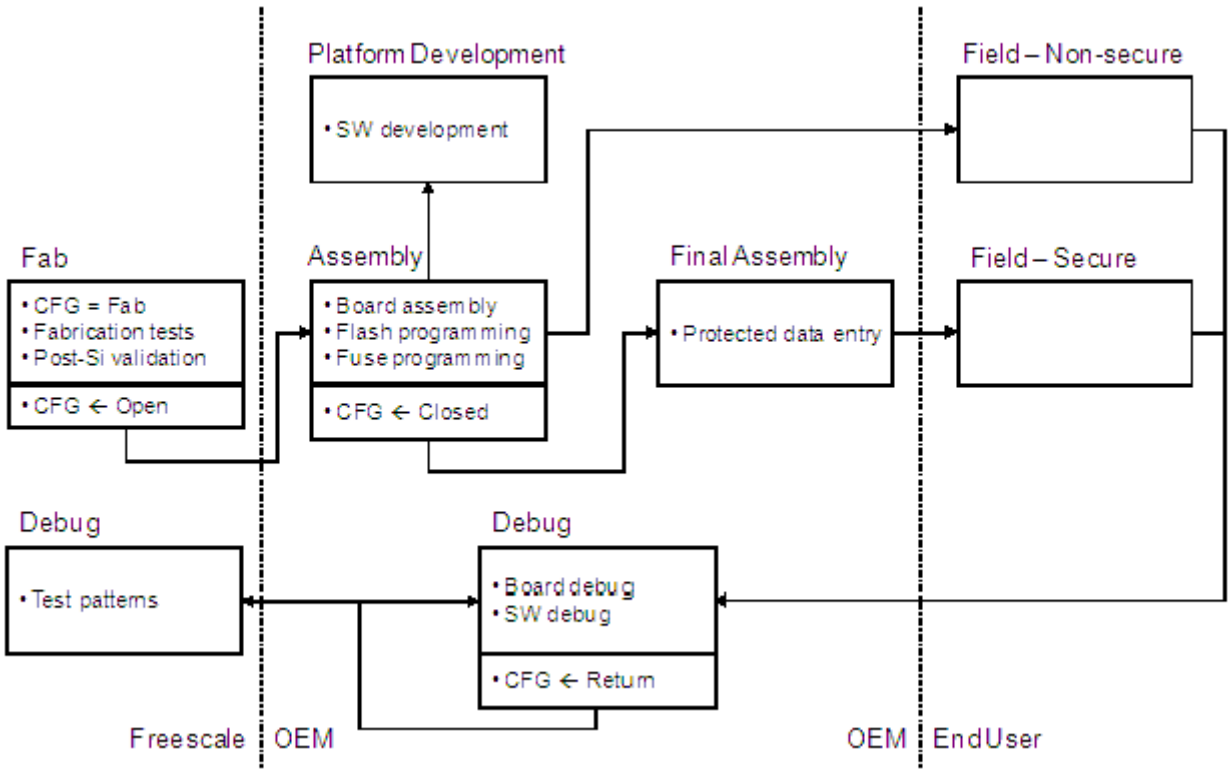


Figure 2-2. Device security configuration life cycle

2.7 Field return for retest procedure

On devices shipped to end users, manufacturers can enable debugging restrictions designed to protect device keys and other sensitive data. These debugging restrictions include the following measures.

- Disable Freescale test modes (using the DIR_BT_DIS fuse).

- Disable JTAG (using JTAG_SMODE or SJC_DISABLE fuses).
- Prevent execution of unauthorized bootloader software (using the SEC_CONFIG[1] fuse).

Naturally, these debugging restrictions also constrain legitimate debugging of field return devices with suspected faults. The chip includes a Field Return configuration to allow legitimate debugging, including the possibility of Freescale running test modes, on returned parts. This Field Return configuration:

- Enables Freescale test modes (overriding the DIR_BT_DIS fuse),
- Enables JTAG (overriding JTAG_SMODE and SJC_DISABLE fuses),
- Enables execution of unsigned bootloader software as in Open configuration (overriding the SEC_CONFIG[1] fuse).

To protect sensitive data already provisioned, the Field Return configuration permanently disables access to device keys (including access from DCP modules).

The entry to the Field Return configuration is strictly controlled in order to deter inadvertent, unauthorized or widespread use.

- The FIELD_RETURN fuse is protected by the FIELD_RETURN_LOCK sticky bit in the OCOTP_CTRL fuse controller.
- Before leaving the boot ROM, the FIELD_RETURN_LOCK bit is set by default so the FIELD_RETURN fuse cannot be burned.
- Setting the FIELD_RETURN_LOCK bit can be avoided by including an Unlock command in the CSF or DCD (Open configuration only) providing:
  - the CSF and bootloader software pass signature verification (Closed configuration), and
  - the Unlock command arguments match the value in the UNIQUE_ID fuses on the device.

The typical mass production bootloader on shipping devices has no such Unlock command, so entry to Field Return configuration requires booting to a special bootloader which is customized for a single device. For Closed devices, the special bootloader must be signed for that single device, so it cannot be used to unlock other devices even if it is leaked to end users.

The boot flow to activate Field Return configuration is shown in next figure.

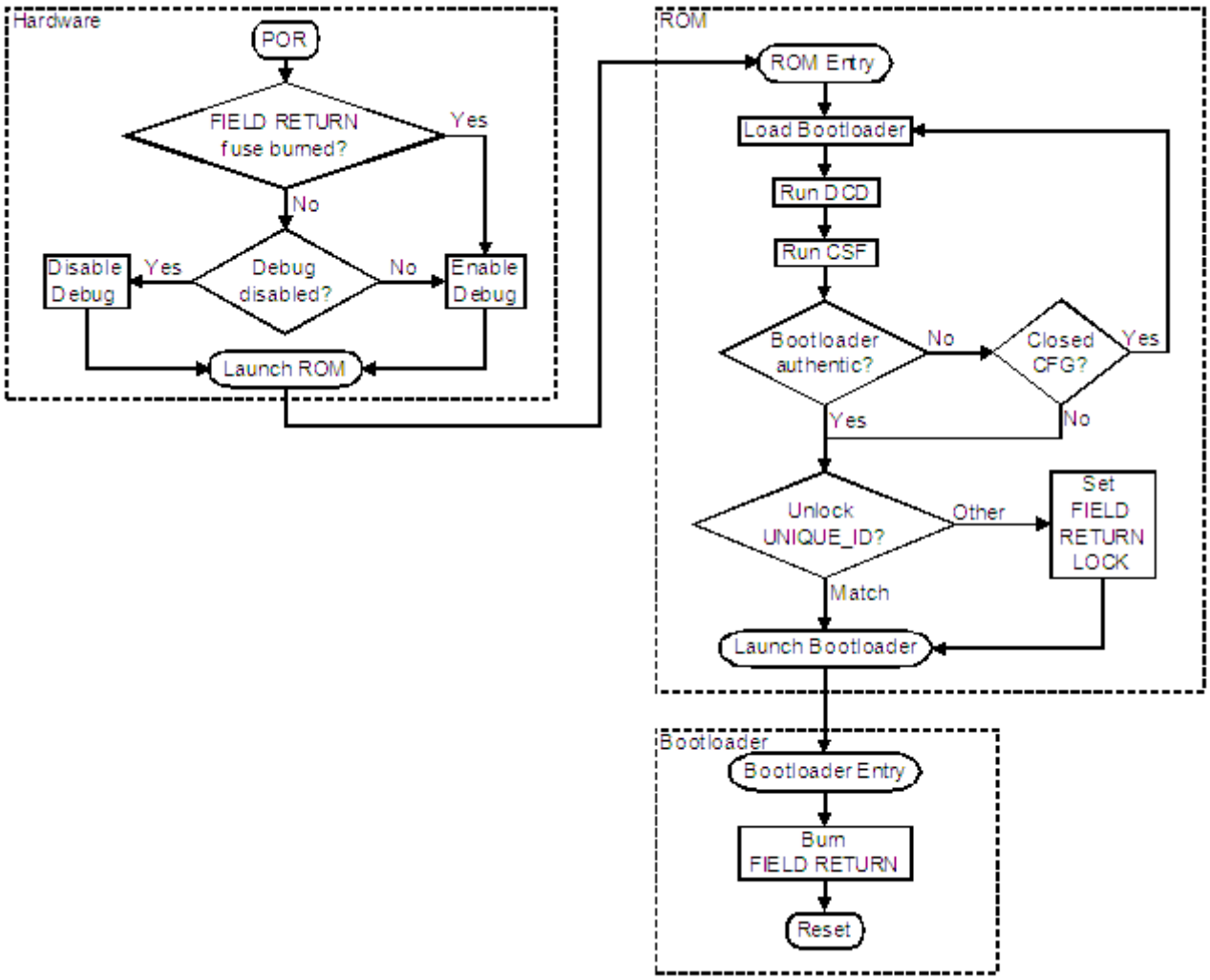


Figure 2-3. HAB FIELD_RETURN flow chart

Once FIELD_RETURN fuse is burned, the part is nearly returned to its OPEN state after the next POR reset, including:

- Revert back value of DIR_BT_DIS fuse, to allow Freescale test mode run
- Enable JTAG (overriding blocking by other means)
- Allow unsigned images to execute, as in OPEN configuration
- Block access to sensitive keys provided by OCOTP_CTRL or SNVS to DCP
- Note that Field Return configuration is required for Freescale to run test patterns even on non-secure products (Open configuration).

Chapter 3
Central Security Unit (CSU)

3.1 Overview

3.1.1 Features

The Central Security Unit (CSU) sets access control policies between bus masters and bus slaves, allowing peripherals to be separated into distinct security domains. This protects against unauthorized access to data e.g. when software programs a DMA bus master to access addresses that the software is prohibited from accessing directly. Configuring DMA bus master privileges in the CSU consistent with software privileges defends against such attempted accesses. Additionally, the CSU manages system security alarms. These alarms are signals routed from various SoC peripherals and I/O that indicate security violation conditions.

CSU has the following security related features:

- Peripheral access policy - Appropriate bus master privilege and identity are required to access each peripheral.
- Masters privilege policy - CSU overrides bus master privilege signals, i.e. user/supervisor secure/non-secure, according to access control policy.

3.2 Functional description

The CSU enables secure software to set bus privilege security policy within the platform.

Security policies may be set, and optionally locked in the CSU registers. Examples of secure software include the command sequence file (CSF) processed by the High Assurance Boot (HAB) or a HAB-authenticated image which executes after the boot ROM.

3.2.1 Peripheral access policy

According to its programmed policy, the CSU determines the bus master privileges and the masters that are allowed to access each of the slave peripherals.

There are four security modes of operation (i.e. bus privileges) in the system distinguished by security (TrustZone/non-TrustZone) and privilege (Supervisor/User) setting of the module. Below is the list of these security modes from the highest security level to the lowest:

- TrustZone (Secure) Privilege (Supervisor) Mode - Highest Security Level
- TrustZone (Secure) non-Privilege (User) Mode - Medium Security Level
- non-TrustZone (Regular) Privilege (Supervisor) Mode - Medium Security Level
- non-TrustZone (Regular) non-Privilege (User) Mode - Lowest Security Level

This functionality is implemented as follows:

The Configure Slave Level (CSL) Register value for a specified peripheral resource defines the output signal -- csu_sec_level for that peripheral. The value of this signal determines by what master privileges a peripheral is accessible. The relationship between the value of the csu_sec_level signal and security operation mode is shown in the table below.

Table 3-1. Permission Access Table

CSU_SEC_LEVEL[2:0]	Non-Secure User Mode	Non-Secure Spvr Mode	Secure (TZ) User Mode	Secure (TZ) Spvr Mode	CSL register value
(0) 000	RD+WR	RD+WR	RD+WR	RD+WR	8'b1111_1111
(1) 001	None	RD+WR	RD+WR	RD+WR	8'b1011_1011
(2) 010	RD	RD	RD+WR	RD+WR	8'b0011_1111
(3) 011	None	RD	RD+WR	RD+WR	8'b0011_1011
(4) 100	None	None	RD+WR	RD+WR	8'b0011_0011
(5) 101	None	None	None	RD+WR	8'b0010_0010
(6) 110	None	None	RD	RD	8'b0000_0011
(7) 111	None	None	None	None	Any other value

For more detailed information on the CSU, see the *Multimedia Applications Processor Security Reference Manual*.

3.2.2 Initialization policy

The recommended initialization procedure is as follows:

1. Write the CSU_CSL Register Field value to indicate each peripheral privilege mode.
2. Write the HP Register Field value to override a Master's privilege mode.

NOTE

After programming, to prevent further modification to a register value the register Lock bit should be set.

3.3 Programmable Registers

The following sections provide detailed descriptions of the CSU registers and their respective bit and field assignments. Assume that the base address is 021C.

- The CSU registers: CSU_CSL, CSU_HP, CSU_SA and CSU_HPCONTROL can only be written in Secure Supervisor Mode. (Note: These registers are also referred to as Security Control Registers or SCRs in this document)
- The previous cycle's Lock bit is checked while writing to a register. If the Lock bit was cleared in the previous cycle and is being set during the current cycle, then the register fields covered by that Lock bit may be written during the current cycle.

CSU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Config security level register (CSU_CSL0)	32	R/W	0033_0033h	3.3.1/38
4	Config security level register (CSU_CSL1)	32	R/W	0033_0033h	3.3.1/38
8	Config security level register (CSU_CSL2)	32	R/W	0033_0033h	3.3.1/38
C	Config security level register (CSU_CSL3)	32	R/W	0033_0033h	3.3.1/38
10	Config security level register (CSU_CSL4)	32	R/W	0033_0033h	3.3.1/38
14	Config security level register (CSU_CSL5)	32	R/W	0033_0033h	3.3.1/38
18	Config security level register (CSU_CSL6)	32	R/W	0033_0033h	3.3.1/38
1C	Config security level register (CSU_CSL7)	32	R/W	0033_0033h	3.3.1/38
20	Config security level register (CSU_CSL8)	32	R/W	0033_0033h	3.3.1/38
24	Config security level register (CSU_CSL9)	32	R/W	0033_0033h	3.3.1/38
28	Config security level register (CSU_CSL10)	32	R/W	0033_0033h	3.3.1/38
2C	Config security level register (CSU_CSL11)	32	R/W	0033_0033h	3.3.1/38
30	Config security level register (CSU_CSL12)	32	R/W	0033_0033h	3.3.1/38
34	Config security level register (CSU_CSL13)	32	R/W	0033_0033h	3.3.1/38
38	Config security level register (CSU_CSL14)	32	R/W	0033_0033h	3.3.1/38
3C	Config security level register (CSU_CSL15)	32	R/W	0033_0033h	3.3.1/38
40	Config security level register (CSU_CSL16)	32	R/W	0033_0033h	3.3.1/38

Table continues on the next page...

CSU memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
44	Config security level register (CSU_CSL17)	32	R/W	0033_0033h	3.3.1/38
48	Config security level register (CSU_CSL18)	32	R/W	0033_0033h	3.3.1/38
4C	Config security level register (CSU_CSL19)	32	R/W	0033_0033h	3.3.1/38
50	Config security level register (CSU_CSL20)	32	R/W	0033_0033h	3.3.1/38
54	Config security level register (CSU_CSL21)	32	R/W	0033_0033h	3.3.1/38
58	Config security level register (CSU_CSL22)	32	R/W	0033_0033h	3.3.1/38
5C	Config security level register (CSU_CSL23)	32	R/W	0033_0033h	3.3.1/38
60	Config security level register (CSU_CSL24)	32	R/W	0033_0033h	3.3.1/38
64	Config security level register (CSU_CSL25)	32	R/W	0033_0033h	3.3.1/38
68	Config security level register (CSU_CSL26)	32	R/W	0033_0033h	3.3.1/38
6C	Config security level register (CSU_CSL27)	32	R/W	0033_0033h	3.3.1/38
70	Config security level register (CSU_CSL28)	32	R/W	0033_0033h	3.3.1/38
74	Config security level register (CSU_CSL29)	32	R/W	0033_0033h	3.3.1/38
78	Config security level register (CSU_CSL30)	32	R/W	0033_0033h	3.3.1/38
7C	Config security level register (CSU_CSL31)	32	R/W	0033_0033h	3.3.1/38
80	Config security level register (CSU_CSL32)	32	R/W	0033_0033h	3.3.1/38
84	Config security level register (CSU_CSL33)	32	R/W	0033_0033h	3.3.1/38
88	Config security level register (CSU_CSL34)	32	R/W	0033_0033h	3.3.1/38
8C	Config security level register (CSU_CSL35)	32	R/W	0033_0033h	3.3.1/38
90	Config security level register (CSU_CSL36)	32	R/W	0033_0033h	3.3.1/38
94	Config security level register (CSU_CSL37)	32	R/W	0033_0033h	3.3.1/38
98	Config security level register (CSU_CSL38)	32	R/W	0033_0033h	3.3.1/38
9C	Config security level register (CSU_CSL39)	32	R/W	0033_0033h	3.3.1/38
200	HP register (CSU_HP0)	32	R/W	0000_0000h	3.3.2/43
204	HP register (CSU_HP1)	32	R/W	0000_0000h	3.3.2/43
218	Secure access register (CSU_SA)	32	R/W	0000_0000h	3.3.3/47

3.3.1 Config security level register (CSU_CSL*n*)

There are 40 Config Security Level (CSU_CSL0-CSU_CSL39) registers. Each CSU_CSL is comprised of two fields, each field used to determine the read and write access permissions for a slave peripheral. These 8-bit fields for the first and second slaves are in the locations b23-b16 and bits b7-b0, respectively.

Permission Access Table Table 3-1 shows security levels and csu_sec_level signal levels corresponding to different values of the 8-bit CSU_CSL field for a given slave.

Most slaves have unique CSL registers. Some slaves are grouped together. The following table shows allocation of CSL register per slave or group of slave modules.

Table 3-6. CSL Slave Modules Mapping

Corresponding CSL register and bit field	Slave Module	Comments
CSL0 [7:0]	PWM1 PWM2 PWM3 PWM4	
CSL0 [23:16]	DBGMON	
CSL1 [7:0]	QOS	
CSL1 [23:16]	GPT EPIT1 EPIT2	
CSL2 [7:0]	GPIO1 GPIO2	
CSL2 [23:16]	GPIO3 GPIO4	
CSL3 [7:0]	GPIO5	
CSL3 [23:16]	----	
CSL4 [7:0]	KPP	
CSL4 [23:16]	WDOG1	
CSL5 [7:0]	WDOG2	
CSL5 [23:16]	CCM SNVS_HP SRC GPC	
CSL6 [7:0]	ANATOP	
CSL6 [23:16]	IOMUXC	
CSL7 [7:0]	CSI TCON	
CSL7 [23:16]	SDMA	
CSL8 [7:0]	USB	
CSL8 [23:16]	FEC	
CSL9 [7:0]	MSHC	
CSL9 [23:16]	USDHC1	
CSL10 [7:0]	USDHC2	
CSL10 [23:16]	USDHC3	
CSL11 [7:0]	USDHC4	
CSL11 [23:16]	I2C1	
CSL12 [7:0]	I2C2	
CSL12 [23:16]	I2C3	
CSL13 [7:0]	ROMCP	

Table continues on the next page...

Table 3-6. CSL Slave Modules Mapping (continued)

Corresponding CSL register and bit field	Slave Module	Comments
CSL13 [23:16]	DCP MMDC	
CSL14 [7:0]	EIM	Do not modify the EIM peripherals' CSL register bits while they are being accessed through the AHB/AXI slave bus.
CSL14 [23:16]	OCOTP_CTRL	
CSL15 [7:0]	----	
CSL15 [23:16]	PERFMON1 PERFMON2	
CSL16 [7:0]	TZASC1	
CSL16 [23:16]	RNGB	
CSL17 [7:0]	AUDMUX	
CSL17 [23:16]	----	
CSL18 [7:0]	SPDIF	
CSL18 [23:16]	eCSPI1	
CSL19 [7:0]	eCSPI2	
CSL19 [23:16]	eCSPI3	
CSL20 [7:0]	eCSPI4	
CSL20 [23:16]	UART5	
CSL21 [7:0]	UART1	
CSL21 [23:16]	UART2	
CSL22 [7:0]	SSI1	
CSL22 [23:16]	SSI2	
CSL23 [7:0]	SSI3	
CSL23 [23:16]	UART3	
CSL24 [7:0]	----	
CSL24 [23:16]	ROMCP	
CSL25 [7:0]	----	
CSL25 [23:16]	----	
CSL26 [7:0]	OCRAM	
CSL26 [23:16]	----	
CSL27 [7:0]	----	
CSL27 [23:16]	----	
CSL28 [7:0]	----	
CSL28 [23:16]	PXP	
CSL29 [7:0]	OPENVG	
CSL29 [23:16]	ARM	
CSL30 [7:0]	EPDC	

Table continues on the next page...

Table 3-6. CSL Slave Modules Mapping (continued)

Corresponding CSL register and bit field	Slave Module	Comments
CSL30 [23:16]	----	
CSL31 [7:0]	LCDIF	
CSL31 [23:16]	EIM	
CSL32 [7:0]	----	
CSL32 [23:16]	GPU2D	
CSL33 [7:0]	----	
CSL33 [23:16]	----	
CSL34 [7:0]	----	
CSL34 [23:16]	----	
CSL35 [7:0]	----	
CSL35 [23:16]	----	
CSL36 [7:0]	----	
CSL36 [23:16]	----	
CSL37 [7:0]	----	
CSL37 [23:16]	----	
CSL38 [7:0]	----	
CSL38 [23:16]	UART4	
CSL39 [7:0]	SPBA	
CSL39 [23:16]	----	

NOTE

Do not modify the EIM peripherals' CSL register bits while they are being accessed through the AHB/AXI slave bus.

Address: 0h base + 0h offset + (4d × i), where i=0d to 39d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							LOCK_S1	NSW_S1	NUW_S1	SSW_S1	SUW_S1	NSR_S1	NUR_S1	SSR_S1	SUR_S1
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							LOCK_S2	NSW_S2	NUW_S2	SSW_S2	SUW_S2	NSR_S2	NUR_S2	SSR_S2	SUR_S2
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1

CSU_CSL*n* field descriptions

Field	Description
31–25 -	This field is reserved. Reserved
24 LOCK_S1	Lock bit corresponding to the first slave. Written by secure software. 0 Not locked. Bits 16-23 may be written by software 1 Bits 16-23 locked and cannot be written by software
23 NSW_S1	Non-secure supervisor write access control for the first slave 0 Non-secure supervisor write access disabled for the first slave. 1 Non-secure supervisor write access enabled for the first slave
22 NUW_S1	Non-secure user write access control for the first slave 0 Non-secure user write access disabled for the first slave. 1 Non-secure user write access enabled for the first slave.
21 SSW_S1	Secure supervisor write access control for the first slave 0 Secure supervisor write access disabled for the first slave. 1 Secure supervisor write access enabled for the first slave.
20 SUW_S1	Secure user write access control for the first slave 0 Secure user write access disabled for the first slave. 1 Secure user write access enabled for the first slave.
19 NSR_S1	Non-secure supervisor read access control for the first slave 0 Non-secure supervisor read access disabled for the first slave. 1 Non-secure supervisor read access enabled for the first slave.
18 NUR_S1	Non-secure user read access control for the first slave 0 Non-secure user read access disabled for the first slave. 1 Non-secure user read access enabled for the first slave.
17 SSR_S1	Secure supervisor read access control for the first slave 0 Secure supervisor read access disabled for the first slave. 1 Secure supervisor read access enabled for the first slave.
16 SUR_S1	Secure user read access control for the first slave 0 Secure user read access disabled for the first slave. 1 Secure user read access enabled for the first slave
15–9 -	This field is reserved. Reserved
8 LOCK_S2	Lock bit corresponding to the second slave. Written by secure software. 0 Not locked. Bits 7-0 may be written by software 1 Bits 7-0 locked and cannot be written by software
7 NSW_S2	Non-secure supervisor write access control for the second slave 0 Non-secure supervisor write access disabled for the second slave. 1 Non-secure supervisor write access enabled for the second slave

Table continues on the next page...

CSU_CSL*n* field descriptions (continued)

Field	Description
6 NUW_S2	Non-secure user write access control for the second slave 0 Non-secure user write access disabled for the second slave. 1 Non-secure user write access enabled for the second slave.
5 SSW_S2	Secure supervisor write access control for the second slave 0 Secure supervisor write access disabled for the second slave. 1 Secure supervisor write access enabled for the second slave.
4 SUW_S2	Secure user write access control for the second slave 0 Secure user write access disabled for the second slave. 1 Secure user write access enabled for the second slave.
3 NSR_S2	Non-secure supervisor read access control for the second slave 0 Non-secure supervisor read access disabled for the second slave. 1 Non-secure supervisor read access enabled for the second slave.
2 NUR_S2	Non-secure user read access control for the second slave 0 Non-secure user read access disabled for the second slave. 1 Non-secure user read access enabled for the second slave.
1 SSR_S2	Secure supervisor read access control for the second slave 0 Secure supervisor read access disabled for the second slave. 1 Secure supervisor read access enabled for the second slave.
0 SUR_S2	Secure user read access control for the second slave 0 Secure user read access disabled for the second slave. 1 Secure user read access enabled for the second slave

3.3.2 HP register (CSU_HP*n*)

The HP register may be programmed to determine the privilege (either User Mode or Supervisor Mode) for up to sixteen different masters. The privilege of a particular master may be overridden by muxing it with the corresponding bit in this register.

The sixteen even bit positions (CSU_HP[30,28,...0]) in the register hold the privilege indicator bits; while the odd bit positions (CSU_HP[31,29,...,1])) contain lock bits which enable/disable writing to the corresponding privilege indicator bits.

Table 3-88. HP Slave Modules Mapping

Corresponding HP register and bit field	Master Module	Comments
HP[1:0]	----	
HP[3:2]	DCP	

Table continues on the next page...

Table 3-88. HP Slave Modules Mapping (continued)

Corresponding HP register and bit field	Master Module	Comments
HP[5:4]	SDMA	
HP[7:6]	EPDC TCON LCDIF PXP GPU2D	
HP[9:8]	USB	
HP[11:10]	Test Port	
HP[13:12]	CSI	
HP[15:14]	MSHC	
HP[17:16]	----	
HP[19:18]	----	
HP[21:20]	FEC	
HP[23:22]	DAP/ARM	
HP[25:24]	USDHC1	
HP[27:26]	USDHC2	
HP[29:28]	USDHC3	
HP[31:30]	USDHC4	

Address: 0h base + 200h offset + (4d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	L	HP	L	HP	L	HP	L	HP	L	HP	L	HP	L	HP	L	HP
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	L	HP	L	HP	L	HP	L	HP	L	HP	L	HP	L	HP	L	HP
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSU_HP*n* field descriptions

Field	Description
31 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
30 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
29 L	Lock bit set by secure software

Table continues on the next page...

CSU_HP*n* field descriptions (continued)

Field	Description
	0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
28 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
27 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
26 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
25 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
24 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
23 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
22 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
21 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
20 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
19 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
18 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
17 L	Lock bit set by secure software

Table continues on the next page...

CSU_HP*n* field descriptions (continued)

Field	Description
	0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
16 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
15 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
14 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
13 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
12 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
11 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
10 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
9 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
8 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
7 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
6 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
5 L	Lock bit set by secure software

Table continues on the next page...

CSU_HP*n* field descriptions (continued)

Field	Description
	0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
4 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
3 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
2 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master
1 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
0 HP	Privilege indicator bits 0 User Mode for the corresponding master 1 Supervisor Mode for the corresponding master

3.3.3 Secure access register (CSU_SA)

The Secure Access register may be programmed to specify the access policy (either Secure or Non-secure) for up to sixteen different masters. This register is used to set the access policy for Type 1 masters which are incapable of setting the policy by themselves.

The sixteen even bit positions (CSU_SA[30,28,...,0]) in the register hold the policy indicator bits; while the odd bit positions (CSU_SA[31,29,...,1]) contain lock bits which enable/disable writing to the corresponding policy indicator bits.

Table 3-92. SA Slave Modules Mapping

Corresponding SA register and bit field	Master Module	Comments
SA[1:0]	---	
SA[3:2]	DCP	
SA[5:4]	SDMA	

Table continues on the next page...

Table 3-92. SA Slave Modules Mapping (continued)

Corresponding SA register and bit field	Master Module	Comments
SA[7:6]	EPDC TCON LCDIF PXP GPU2D	
SA[9:8]	USB	
SA[11:10]	Test Port	
SA[13:12]	CSI	
SA[15:14]	MSHC	
SA[17:16]	FEC	
SA[19:18]	DAP	
SA[21:20]	USDHC1	
SA[23:22]	USDHC2	
SA[25:24]	USDHC3	
SA[27:26]	USDHC4	
SA[29:28]	----	
SA[31:30]	----	

Address: 0h base + 218h offset = 218h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	L	NSA	L	NSA	L	NSA	L	NSA	L	NSA	L	NSA	L	NSA	L	NSA
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	L	NSA	L	NSA	L	NSA	L	NSA	L	NSA	L	NSA	L	NSA	L	NSA
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSU_SA field descriptions

Field	Description
31 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
30 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
29 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software

Table continues on the next page...

CSU_SA field descriptions (continued)

Field	Description
28 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
27 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
26 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
25 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
24 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
23 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
22 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
21 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
20 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
19 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
18 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
17 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software

Table continues on the next page...

CSU_SA field descriptions (continued)

Field	Description
16 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
15 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
14 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
13 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
12 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
11 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
10 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
9 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
8 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
7 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
6 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
5 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software

Table continues on the next page...

CSU_SA field descriptions (continued)

Field	Description
4 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
3 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
2 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master
1 L	Lock bit set by secure software 0 No lock -- adjacent (next lower) bit can be written by software 1 Lock -- adjacent (next lower) bit cannot be written by software
0 NSA	Non-Secure Access Policy indicator bits 0 Secure access for the corresponding Type 1 master 1 Non-secure access for the corresponding Type 1 master

Chapter 4
Data Co-Processor (DCP)

4.1 Overview

This chapter describes the data co-processor (DCP) block included on the i.MX 6SoloLite and how to use it.

It includes sections on memory copy functionality, Advanced Encryption Standard (AES), hashing, and arbitration. Sections on programming channel operations and example code are also provided. Programmable registers are described in DCP Register Reference Index ().

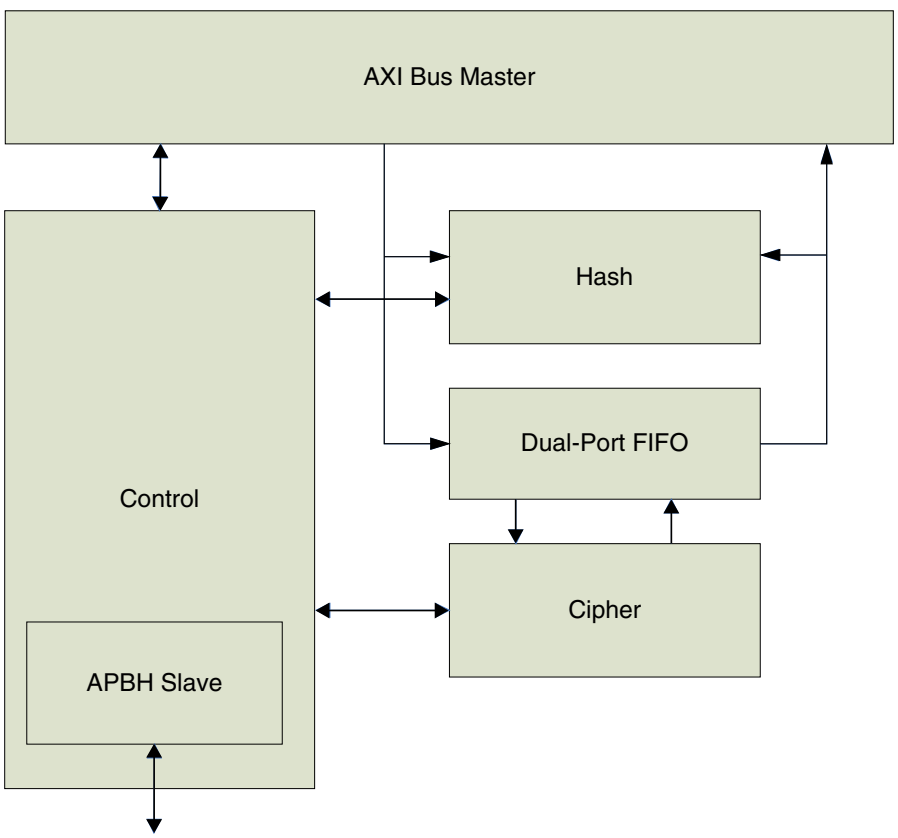


Figure 4-1. Data Co-Processor (DCP) Block Diagram

The DCP module provides support for general encryption and hashing functions typically used for security functions. Because its basic job is moving data from memory to memory, it also incorporates memory-copy (memcopy) function for both debugging and as a more efficient method of copying data between memory blocks than the DMA-based approach. The memcopy function also has a "blit" mode of operation where it can transfer a rectangular block of data to a video frame buffer.

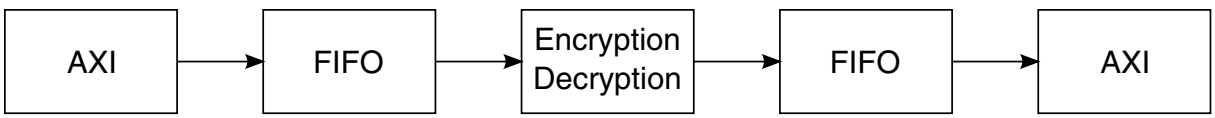
The DCP has been designed to support a wide variety of encryption and hashing algorithms, and the control structures have been designed to allow flexibility in adding additional algorithms and modes in the future. It supports up to 16 encryption algorithms with up to 16 different modes of operation as well as up to 16 hashing algorithms . While the DCP module has been designed to support numerous algorithms, only a subset may be implemented in any given implementation (see the Capability Register).

The DCP module processes data based on chained command structures written to system memory by software (in a manner similar to the DMA engine). The control block maintains registers to support four independent and concurrent chains, allowing software to virtualize access to the DCP block. Each command in a chain represents a work unit that the module will process to completion. At the end of each work unit, the control logic arbitrates among chains with outstanding commands and processes a command from that chain. Arbitration among the channels is round-robin, allowing all active channels equal access to the data engine. Each channel also supports a "high priority" mode that allows it to have priority over the remaining channels. If multiple channels are selected as high-priority, the channel arbiter selects among the high priority channels in round-robin fashion.

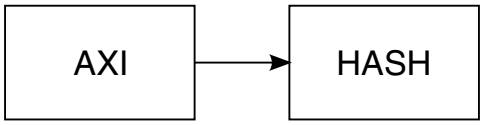
The data flow through the DCP module can be configured in one of five fashions, depending on the functionality activated by the control packet:

- **Memcopy/Blit Mode-**Data is moved unchanged from one memory buffer to another.

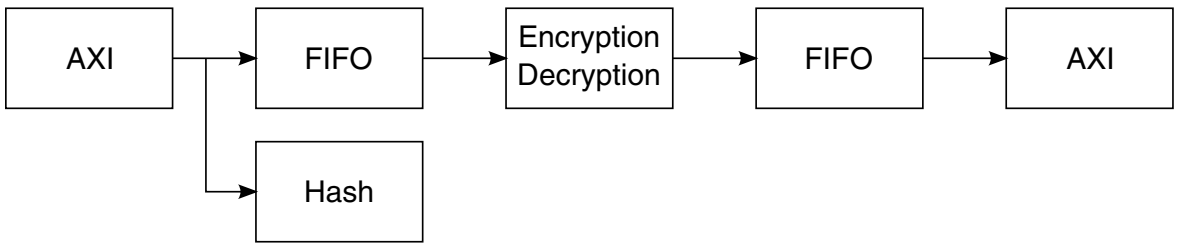
- **Encryption Only-**Data from source buffer is encrypted/decrypted into the destination buffer



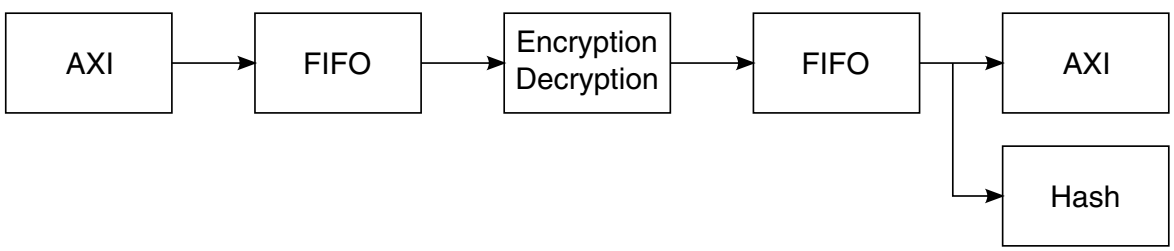
- **Hashing Only-**Data from source buffer is read, and a hash is generated.



- **Encryption and Input Hashing-**Data from source buffer is encrypted/decrypted into destination, and source buffer is hashed.



- **Encryption and Output Hashing-**Data from source buffer is encrypted/decrypted into destination, and output data is hashed.



4.1.1 DCP Limitations for Software

While the DCP module has been designed to be as flexible as possible, there are a few limitations to which software must adhere:

- Buffer sizes for all operations MUST be aligned to the natural size of the transfer algorithm used. Memcopy operations can transfer any number of bytes (one-byte granularity) and AES operations must be multiples of 16 bytes (four-word granularity). For all operations, if the byte count is not a word granularity, the hardware rounds up to the next word. Hashing is supported at a byte granularity.
- The DCP module supports buffer operations to any byte alignment, but performance will be improved if buffers are aligned to a four-byte boundary, since fetch/store operations can be performed without having to do byte operations to accommodate the misaligned addresses.
- Hash operations are limited to a 512 Mbyte buffer size. The hardware only implements a 32-bit hash length counter instead of the 64-bit counter supported by

the SHA-1/SHA-256 algorithm (counter counts bits, not bytes, therefore a total of 512 Mbytes).

- For chained hashing operations (operations involving multiple descriptors), every descriptor except the last must have a byte count that is a 16-word multiple (granularity of the hash algorithm).
- Key values cannot be written while the AES block is active. This limitation exists because the key RAM is in use while AES is operational. Any writes from the peripheral bus cannot be held in wait states; therefore, the RAM must be accessible during key writes.
- The byte-swap controls can only be used with modulo-4 length buffers. For non-modulo-4 lengths, the final partial word will contain incorrect data. Any address alignment can be used with byte swapping, however.
- The word-swap controls are only useful with cipher operations, because the logic forms the 128-bit cipher data from four words from system memory. The word-swap controls are ignored for memcopy or hashing operations.
- DCP only supports writes to word boundaries to OCRAM.

4.2 Operation

The top-level DCP module contains the AXI master, peripheral slave bus interface units, the main control block and FIFO, and any encryption or hashing blocks included with the design.

The controller manages the fetching of work blocks, the fetching/storing of context information when switching between chain pointers, and the managing of the data flow through the FIFO, SHA, and AES blocks. Data entering the block from the AXI master is placed in the FIFO for consumption by the cipher block. After the cipher module has finished its operation, data is placed back into the FIFO and stored back to memory via the AXI master. When hashing is enabled, the SHA block takes its inputs from the bus side of the FIFO to allow it to operate without waiting for the cipher block to complete. The peripheral slave provides all register controls and interfaces mainly with the control block.

4.2.1 Memory Copy, Blit, and Fill Functionality

In its most basic operation, the DCP supports moving unmodified data from one place in system memory to another.

This functionality is referred to as "memcopy", because it operates only on memory and it copies data from one place to another. Typical uses for memcopy might be for fast virtual memory page moves or repositioning data blocks in memory. Memcopy buffers can be aligned to any memory address and can be of any length (byte granularity). For best performance, buffers should be word-aligned, although the DCP includes enhancements to improve performance for unaligned cases.

The DCP also has the ability to do basic "blit" operations that are typical in graphics operations. To specify a blit, the control packet must have the ENABLE_BLIT bit set in the packet control register. Blit source buffers must be contiguous. The output destination buffer for a blit operation is defined as a "M runs of N bytes" that define a rectangular region in a frame buffer. For blit operations, each line of the blit may consist of any number of bytes. After performing a "run", the DCP updates the destination pointer such that the next destination address falls on the pixel below the start of the previous run operation. This is done by incrementing the starting pointer by the frame buffer width, which is specified in the Control field.

In addition to being able to copy data within memory, the DCP also provides a "fill" operation, where source data comes not from another memory location, but from an internal register (the source buffer address in the control packet). This is done whenever the DCP_Control0[CONSTANT_FILL] flag is set in the packet control register (see Control1 Field). This feature may be used with memcopy to prefill memory with a specified value or during a blit operation to fill a rectangular region with a constant color.

4.2.2 Advanced Encryption Standard (AES)

The AES block implements a 128-bit key/data encryption/decryption block as defined by the National Institute of Standards and Technology (NIST) as US FIPS PUB 197, dated November 2001 (see references for specifications and toolkits).

4.2.2.1 Key Storage

The DCP implements four SRAM-based keys that may be used by software to securely store keys on a semi-permanent basis. The keys may be written via the programming interface by specifying a key index to specify which key to load and a subword pointer that indicates which word to write within the key. After a subword is written, the logic automatically increments the subword pointer so that software can program the higher-order words of the key without rewriting the key index. Keys written into the key storage are not readable.

To use a key in the key storage, the cipher descriptor packet should select the key by setting the DCP_Control1[KEY_SELECT] field in the Control1 descriptor field without setting the OTP_KEY or PAYLOAD_KEY fields in the Control0 register. See Control0 Field and Control1 Field.

4.2.2.2 AES OTP Key

After a system reset, the OTP controller reads the e-fuse devices and provides OTP key information to the DCP. The DCP receives a 64-bit UNIQUE KEY and a 128-bit CRYPTO KEY. Depending on a key path control fuse the DCP receives the CRYTPO KEY either directly or indirectly through the SNVS trust controller module. The CRYPTO KEY in fuses is actually 256 bits and a mux is used to select the high or low 128 bits of the key. The DCP internally generates the control logic signals to capture this key into the key RAM. The system reset controller coordinates the deassertion of reset such that the OTP key is stable before the DCP reads it. If the SNVS key path is chosen then a security violation removes the CRYPTO KEY, resets the key in the DCP, and makes the key unavailable to the DCP until the next successful secure boot.

To use the OTP key, the descriptor packet should set the OTP_KEY field in the Control1 register (see Control1 Field).

4.2.2.3 Encryption Modes

The most basic form of encryption is the Electronic Code Book (ECB) mode. In this mode, the encryption output is a function only of the key and the plaintext, thus it can be visualized as a giant lookup table. While this provides a great deal of security, there are a few limitations. For instance, if the same plaintext appears more than once in a block of data, the same ciphertext will also appear. This can be very evident if the plaintext contains large blocks of constant data (0s for example) and can be used to formulate attacks against a key.

In order to make ciphers stronger, several modes of operation can be implemented around the basic ECB cipher to provide additional security. One such mode is CBC mode (Cipher Block Chaining), which takes the previous encrypted data and logically XORs it with the next incoming plaintext before performing the encryption. During decryption, the process is reversed and the previous encrypted data is XORed with the decrypted ECB data to provide the plaintext again.

The AES engine supports handling the various modes of operation. The core AES block supports ECB mode, and other algorithms are handled in the wrapper around the encryption blocks. The DCP module supports Cipher Block Chaining (CBC), which chains the data blocks by XORing the previously encrypted data with the plaintext before encryption. Cipher block chaining encryption is illustrated in Figure 4-2.

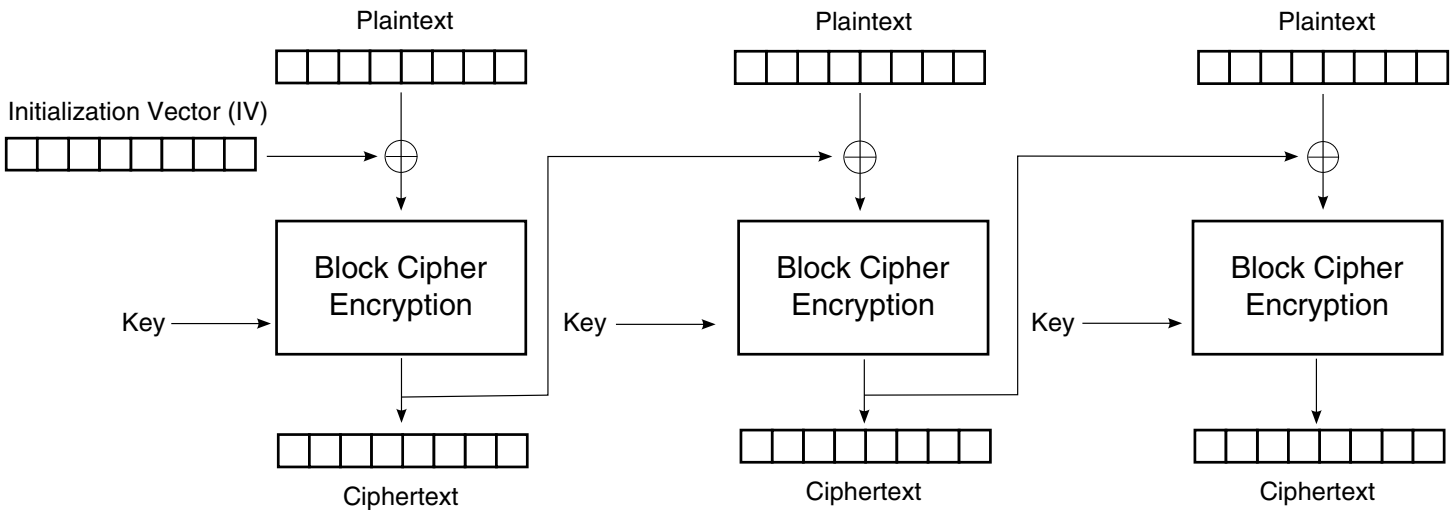


Figure 4-2. Cipher Block Chaining (CBC) Mode Encryption

Decryption (shown in Figure 4-3) works in a similar manner, where the cipher text is first decrypted and then XORed with the previous ciphertext. For the first encryption/ decryption operation, an initialization vector (IV) is used to seed the operation. The IV must be the same for both the encryption and decryption steps; otherwise, decrypted data will not match the original plaintext.

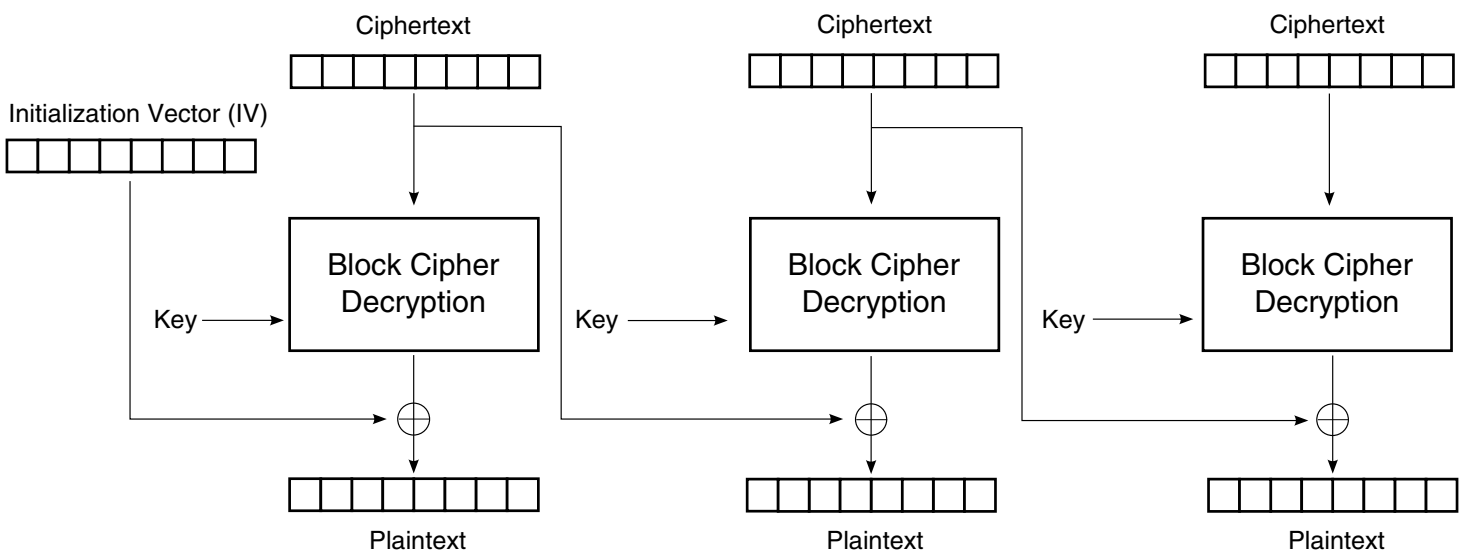


Figure 4-3. Cipher Block Chaining (CBC) Mode Decryption

4.2.3 Hashing

The hashing module implements the SHA-1 and SHA-256 hashing algorithms and a modified CRC-32 checksum algorithm.

These algorithms produce a signature for a block of data that can be used to determine whether the data is intact.

The CRC-32 algorithm implements a 32-bit CRC algorithm similar to the one used by Ethernet and many other protocols. The CRC differs from the Unix cksum() function in three ways:

- The CRC is initialized as 0xFFFFFFFF instead of 0x00000000.
- Logic pads zeros to a 32-bit boundary for trailing bytes.
- Logic does not post-pend the file length.

The SHA-1 block implements a 160-bit hashing algorithm that operates on 512-bit (64-byte) blocks as defined by US FIPS PUB 180-1 in 1995. The SHA-256 mode implements a 256-bit hashing algorithm that operates on 512-bit (64-byte) blocks as defined by US FIPS PUB 180-2 in 2002. The purpose of the hashing module is to generate a unique signature for a block of data that can be used to validate the integrity of the data by comparing the resulting "digest" with the original digest.

Results from hash operations are written to the beginning of the payload for the descriptor. The DCP also has the ability to check the resulting hash against a value in the payload and issue an interrupt if a mismatch occurs.

4.2.4 One Time Programmable (OTP) Key

After a system reset, the One Time Programmable (OTP) controller will read the e-fuse devices and provide OTP key information to the DCP through the 128-bit wide otp_crypto_key_data input data bus and the 64 bit wide otp_unique_key_data unique key input that come directly from the OTP controller.

These bus input values are held constant and originate from the eFuse shadow registers after reset. (Note that the shadow registers for these bits can be changed in the OTP block if they are unlocked.) They will only be loaded by the DCP once after the deassertion of system reset. After reset, the DCP automatically loads these wide keys 32 bits at the time into the internal key RAM.

In i.MX 6SoloLite, DCP can also get a 128-bit OCOTP key through SNVS as another option of 128-bit wide key directly from OTP controller. Setting of fuse bit "OTP_KEY_TO_DCP_DISABLE" will select the SNVS key mode, in this mode DCP will load the SNVS key with each software reset.

While OTP key is normally not available to read operations, the module will allow the key to be read if the otp_crypto_key_ren (read-enable) input is active (high). This allows verification of the key after it has been programmed into the eFuse device. After programming has been validated, another fuse will be set to disable the read capability.

The OTP key (crypto key) may be selected using the DCP_Control1[OTP_KEY] bit in the control field of the packet descriptor or by using key select 0xFF in the CTRL1 field of the descriptor. DCP also supports a second hardware key called the UNIQUE_KEY which is generated from the OTP KEY and key modifier bits from other OTP fuse fields. This key is unique to the device and may be used for encrypting private data stored on the NAND. This key may be selected by writing 0xFE to the KEY_SELECT field in the CTRL1 packet data.

4.2.5 Managing DCP Channel Arbitration and Performance

The DCP can have four channels all competition for DCP resources to complete their operations.

Depending on the situation, critical operations may need to be prioritized above less important operations to ensure smooth system operation. To help software achieve this goal, the DCP implements a programmable arbiter for internal DCP operations and provides "recovery" timers on each channel to throttle channel activity.

4.2.5.1 DCP Arbitration

The DCP implements a multi-tiered arbitration policy that allows software a lot of flexibility in scheduling DCP operations. Figure 4-4 illustrates the arbitration levels and where each channel fit into the arbitration scheme.

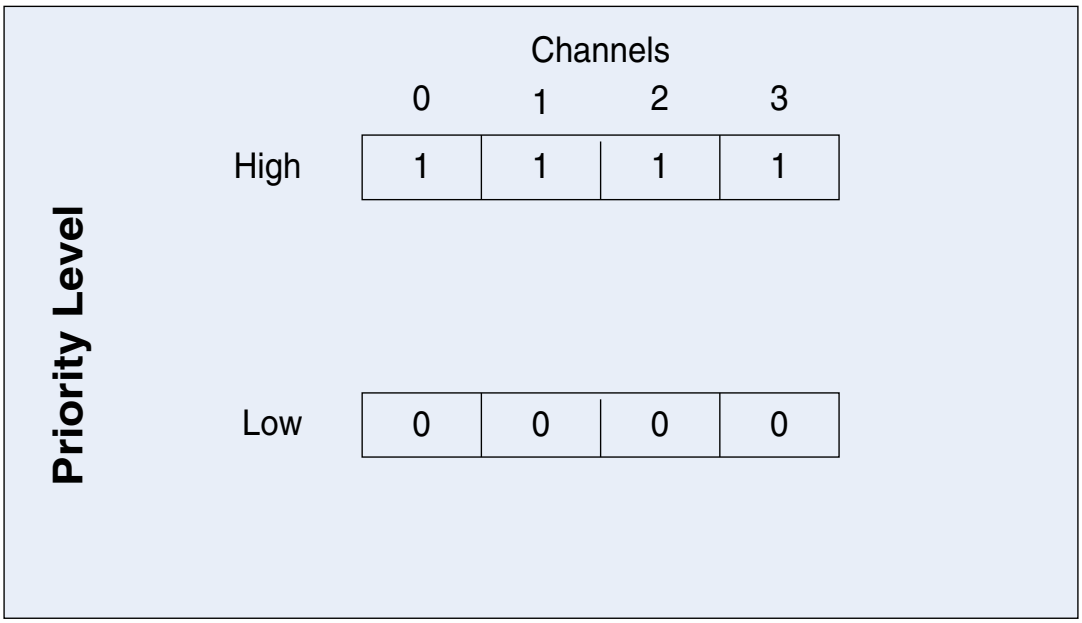


Figure 4-4. DCP Arbitration

Each channel can be programmed as being in either the high-priority or low-priority arbitration pool, depending on the setting in the HIGH_PRIORITY_CHANNEL field of the DCP_CHANNELCTRL register. When the corresponding bit is programmed as a 1, the channel arbitrates in the high-priority pool; otherwise it arbitrates in the low-priority pool. Once a channel has been selected, it completes one packet and then the arbiter re-arbitrates. The channel that just completed participates in the new arbitration round.

4.2.5.2 Channel Recovery Timers

Each channel also contains a channel recovery timer in its DCP_CHnOPTS register. The purpose of the recovery timer is to keep the channel inactive for a period of time after it completes an operation. This capability could be used for a high-priority channel to ensure that at least some lower-priority requests get serviced between packets or to simply allow more timeslices for other channels to perform operations. The value programmed into the recovery timers register delays the channel from operations until 16 times the programmed value. Any non-zero value should prevent the channel from participating in the next arbitration cycle.

4.2.6 Programming Channel Operations

The control logic block maintains the channel pointers and manages arbitration and context switching between the different channels.

It also manages the fetching of work packets and data fetch/store operations from the AXI master interface and coordinates the actions of the hashing and encryption blocks.

The control logic maintains four channels that allow software to effectively create four independent work sets for the DCP module. Software can construct chained control packets in memory that describe encryption/hashing/memcopy operations to the hardware unit. The address for this first control packet can be written to one of the four virtual channels and enabled. When one or more of the channels is enabled, the controller fetches the control packet pointed to by that channel and initiate data fetches from the source buffer. Data is then processed as described in the control packet and stored back to system memory.

Once the processing for a control packet is complete, the controller writes completion status information back to the control packet, and optionally stores relevant state information to the context buffer. If the control packet specifies a subsequent control packet, the channel's pointer is updated to the address for the next packet and an optional interrupt can be issued to the processor.

At this point, the DCP module arbitrates among the virtual channels for the next operation and processes its control packet. If a subsequent operation is continued from a previous operation, the controller automatically loads the context from the previous session into the working registers before resuming operation for that channel.

4.2.6.1 Virtual Channels

The DCP module processes data via work packets stored in memory. Each of the channels contains a pointer to the current work packet and enough control logic to determine whether the channel is active and to provide status to the processor. Each channel also provides a recovery timer to help throttle processing by a particular channel. After processing a packet, the channel enables its 16-bit recovery timer (if the recovery time is set to a non-zero value). The channel will not become active again until its recovery timer has expired. The recovery timer timebase is 16 HCLK cycles, so the timer acts as a 20-bit timer with the bottom four bits implicitly tied to 0. This provides an effective range of zero to 2^{20-1} clocks or 0 ns to 7.8 ms at 133 MHz.

A channel is activated any time its semaphore is non-zero and its recovery timer is cleared. The semaphore can be incremented by software to indicate that the chain pointer has been loaded with a valid pointer. As the hardware completes the work packets, it decrements the semaphore if the Decrement Semaphore flag in the Control 0 field set. Work packets may be chained together using the CHAIN or CHAIN_CONTIGUOUS

bits in the Control0 field, which causes the channel to automatically update the work packet pointer to the value in the NEXT_COMMAND_ADDRESS field at the end of the current work packet.

4.2.6.2 Context Switching

The control logic maintains four virtual channels that allow the DCP block to time-multiplex encryption, hashing, and memcopy operations, it must also retain state information when changing channels so that when a channel is resumed, it can resume the operation from where it left off. This process is called context-switching.

To minimize the number of registers used in the design, the controller saves context information from each channel into a private context area in system memory. When initializing the DCP module, software must allocate memory for the context buffer and write the address into the Context Buffer Pointer register. As the DCP module processes packets, it saves the context information for each channel to the buffer after completing each control packet. When the channel is subsequently activated, the DCP module's internal registers are then reloaded with the proper context before resuming the operation.

Each channel reserves one-fourth of the context buffer area for its context storage. The context buffer consumes 208 bytes of system memory and is formatted as shown in Table 4-1.

Table 4-1. DCP Context Buffer Layout

Range	Channel	Data	Size
0x00-0x0C	3	Cipher Context	16 bytes
0x10-0x30	-	Hash Context	36 bytes
0x34-0x40	2	Cipher Context	16 bytes
0x44-0x64	-	Hash Context	36 bytes
0x68-0x74	1	Cipher Context	16 bytes
0x78-0x98	-	Hash Context	36 bytes
0x9C-0xA8	0	Cipher Context	16 bytes
0xAC-0xCC	-	Hash Context	36 bytes

The control logic writes to the context buffer only if the function is being used. This effectively means that the cipher context is stored for CBC encryption/decryption operations only, and the hash context is written only if SHA-1/SHA-256 is utilized. If neither of these modes are used for a given channel, the memory for the context buffer need not be allocated by software.

Since channel 0 is likely to be used for VMI in an SDRAM-based system-to-page data from the SDRAM to on-chip SRAM, the buffer allocation table has been organized so that the *highest* numbered channel uses the *lowest* area in the context buffer. For this reason, software should allocate the higher numbered channels for encryption/hashing operations and the lower numbered channels for memcopy operations to reduce the size of the context buffer.

If only a single channel is used for CBC mode operations or hashing operations, the controller provides a bit in the control register to disable context switching. In this scenario, context switching is not required, because other channels will not corrupt the state of the hashing or cipher modes. Thus, when the channel resumes, a context load is not required.

Additionally, the control logic monitors the use of CBC/hashing, so that a context reload is not done if the previous channel resumes an operation without an intermediate operation from another channel.

4.2.6.3 Working with Semaphores

Each channel has a semaphore register to indicate that control packets are ready to be processed. Several techniques can be used when programming the semaphores to control the execution of packets within a channel. The channel will continue to execute packets as long as the semaphore is non-zero, a chain bit is set in the descriptor, and no error has occurred.

- Software can write the number of pending packets into the semaphore register with the Decrement Semaphore bit in each control packet set. In this scenario, the channel simply decrements the semaphore for each packet set and terminates at the end of the packet chain. The benefit of this method is that software can easily determine how many packets have executed by reading the semaphore status register.
- Software can create a packet chain with the Decrement Semaphore bit set only in the last packet. In this case, software would write a 1 to the semaphore register to start the chains and the DCP will terminate after executing the last packet.
- Software can create a packet chain with the Decrement Semaphore bit set for each packet and write either a 1 or a number less than the number of packets to the semaphore register. This can be useful when debugging, because it allows the channel to execute only a portion of the packets and software can inspect intermediate values before restarting the channel again.

If an error occurs, the channel issues an interrupt and clears the semaphore register. The channel does not perform any further operations until the error bits in the status register have been cleared. Software can manually clear a non-zero semaphore by writing 0xFF to the CLR (clear) address of the semaphore register.

4.2.6.4 Work Packet Structure

Work packets for the DCP module are created in memory by the processor. Each work packet includes all information required for the hardware to process the data. The general structure of the packet is shown in Figure 4-5.

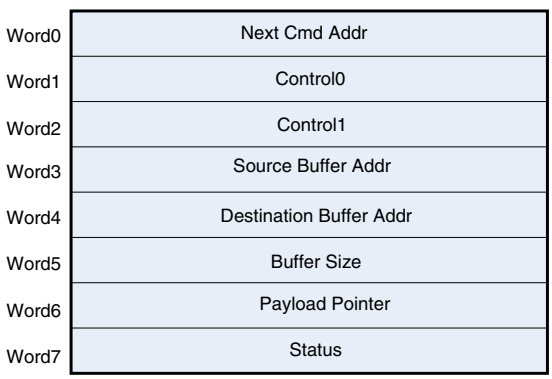


Figure 4-5. DCP Work Packet Structure

For some operations, additional information is required to process the data in the packet. This additional information is provided in the variable-sized payload buffer, which can be found at the address specified in the payload pointer field. When encryption is specified, the payload may include the encryption key (if the key selected resides in the packet), an initialization vector (for modes of operation such as CBC), and expected hash values when data hashing is indicated. The hardware can automatically compare the expected hash with the actual hash and interrupt the processor only on a mismatch. The payload area is used by software to store the calculated hash value at the end of hashing operations (when the HASH_TERM control bit in DCP_Control0 is set).

4.2.6.4.1 Next Command Address Field

The NEXT_COMMAND_ADDRESS field (as shown in Table 4-2) is used to point to the next work packet in the chain. This field is loaded into the channel's command pointer after the current packet has completed processing if the CHAIN bit in the CONTROL0

field (see Table 4-3) is set. The Next Command Address field should be programmed to a non-zero value when the CHAIN bit is set; otherwise, the channel will flag an invalid setup error.

Table 4-2. DCP Next Command Address Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															

4.2.6.4.2 Control0 Field

>The main functions of the DCP module are enabled with the ENABLE_MEMCOPY, ENABLE_BLIT, ENABLE_CIPHER, and ENABLE_HASH bits from the Control0 field in the work packet.

The combinations of these bits determine the function performed by the DCP. Table 4-4 summarizes the function performed for each combination.

Table 4-3. DCP Control0 Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Table continues on the next page...

Operation

Table 4-3. DCP Control0 Field (continued)

TAG	OUTPUT_WORDSWAP	OUTPUT_BYTESWAP	INPUT_WORDSWAP	INPUT_BYTESWAP	KEY_WORDSWAP	KEY_BYTESWAP	TEST_SEMA_IRQ	CONSTANT_FILL	HASH_OUTPUT	HASH_CHECK	HASH_TERM	HASH_INIT	PAYLOAD_KEY	OTP_KEY	CIPHER_INIT	CIPHER_ENCRYP	ENABLE_BLIT	ENABLE_HASH	ENABLE_CIPHE	ENABLE_MEMCOPY	CHAIN_CONTINUOUS	CHAIN	DECR_SEMAPHOR	INTERRUPT_ENABL

Table 4-4. DCP Function Enable Bits

Hash	Cipher	Blit	Memcopy	Description
0	0	0	X	Simple memcopy operation.
0	0	1	0	Blit operation.
0	1	0	0	Simple encrypt/decrypt operation.
1	0	0	0	Simple hash. Only reads performed.
1	0	0	1	Memcopy and hash operation.
1	1	0	0	Hash with encryption/decryption.
All Others				Invalid setup.

The CHAIN bit should be set if the NEXT_COMMAND_ADDRESS field has a valid pointer to the next work packet. When set, this bit causes the channel to update its pointer to the next packet. The CHAIN_CONTINUOUS bit is similar, but it indicates that the next packet follows immediately after the current packet. This allows software to generate templates of operations without regard to the actual physical addresses used, which makes programming easier, especially in a virtual memory environment.

If the INTERRUPT_ENABLE bit is set, the channel generates an interrupt to the processor after completing the work for this packet. When the interrupt is issued, the packet will have been completely processed, including the update of the status/payload fields in the work packet.

Each channel contains an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore in non-zero, the channel is ready to run and process commands. Whenever a command finishes its operation, it checks the DECR_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the idle state and remains there until the semaphore is incremented by software. When the semaphore goes to non-zero and the channel is in its idle state, it then uses the value in the NEXT_COMMAND_ADDRESS field to begin processing.

If the ENABLE_CIPHER bit is set, the data is encrypted or decrypted based on the CIPHER_ENCRYPT bit using the key/encryption settings from the Control1 field. The OTP_KEY and PAYLOAD_KEY indicate the source of the keys for the operation. If the OTP_KEY value is set, the KEY_SELECT field from the Control1 register indicates which OTP key is to be used. If the PAYLOAD_KEY bit is set, the first entry in the payload is the key to be used for the operation. If neither bit is set, the KEY_SELECT field indicates an index in the key RAM that contains the key to be used. (For cases when both the OTP_KEY and PAYLOAD_KEY bits are set, the PAYLOAD_KEY takes precedence.)

The HASH_ENABLE enables hashing of the data with the HASH_OUTPUT bit controlling whether the input data (HASH_OUTPUT=0) or output data (HASH_OUTPUT=1) is hashed. In addition, the hardware can be programmed to automatically check the hashed data if the HASH_CHECK bit is set. If the hash does not match, an interrupt is generated and the channel terminates operation on the current chain. The hashing algorithms also require two additional fields in order to operate properly. The HASH_INIT bit should be set for the first block in a hashing pass to properly initialize the hashing function. The HASH_TERM bit must be set on the final block of a hash to notify the hardware that the hashing operation is complete so that it can properly pad the tail of the buffer according to the hashing algorithm. This flag also triggers the write-back of the hash output to the work packet's payload area.

The CONSTANT_FILL flag may be used for both memcopy and blit operations. When this is set, the source address field is used instead as a constant that is written to all locations in the output buffer.

The WORDSWAP and BYTESWAP bits enable muxes around the FIFO to swap data to handle little-endian and big-endian storage of data in system memory. When these bits are cleared, data is assumed to be in little-endian format. When all bits are set, data is assumed to be in big-endian format.

The TAG field is used to identify the work packet and the associated completion status. As each packet is completed, the channel provides the status and tag information for the last packet processed.

4.2.6.4.3 Control1 Field

The Control1 field (shown in Table 4-5) provides additional values used when hashing or encrypting/decrypting data.

For blit operations, this field indicates the number of bytes in a frame buffer line, which is used to calculate the address for successive lines in each blit operation.

Table 4-5. DCP Control1 Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
CIPHER_CONFIG												HASH_ SELECT				KEY_SELECT								CIPHER_ MODE				CIPHER_ SELECT			
																FRAMEBUFFER_LENGTH (in bytes, blit mode)															

The CIPHER_SELECT field selects from one of sixteen possible encryption algorithms (0=AES128). The CIPHER_MODE selects the mode of operation for that cipher (0=ECB, 1=CBC).

The KEY_SELECT field allows key selection for one of several sources. Keys can be included in the packet payload or may come from OTP or write-only registers.

The HASH_SELECT field selects a hashing algorithm for the operation (0=SHA-1, 1=CRC32, 2=SHA-256).

The CIPHER_CONFIG field provides optional configuration information for the selected cipher. An example would be a key length for the RC4 algorithm.

4.2.6.4.4 Source Buffer

The SOURCE_BUFFER pointer (shown in Table 4-6) specifies the location of the source buffer in memory. The buffer may reside at any byte alignment and should refer to an on-chip SRAM of off-chip SDRAM location. For optimal performance, buffers should be word aligned. When the CONSTANT_FILL flag is set in the Control0 field, the value in this field is used as the data written to all destination buffer locations.

Table 4-6. DCP Source Buffer Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SOURCE_BUFFER																															
CONSTANT (CONSTANT_FILL mode)																															

4.2.6.4.5 Destination Buffer

The DESTINATION_BUFFER (shown in Table 4-7) specifies the location of the destination buffer in on-chip SRAM or off-chip SDRAM memory and may be set to any byte alignment. For in-place encryption, the destination buffer and source buffer should be the same value. For optimal performance, the buffer location should be word-aligned.

Table 4-7. DCP Destination Buffer Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DESTINATION_BUFFER																															

4.2.6.4.6 Buffer Size Field

The BUFFER_SIZE field (shown in Table 4-8) indicates the size of the buffers for memcopy, encryption, and hashing modes. For memcopy and hashing operations, the value may be any number of bytes (byte granularity), and for encryption modes, the length must be a multiple of the selected encryption algorithm's natural data size (16 bytes for AES).

Table 4-8. DCP Buffer Size Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NUMBER_LINES (blit mode)																BLIT_WIDTH (in bytes, blit mode)															
BUFFER_SIZE (in bytes, memcopy, encryption, hashing modes)																															

For blit operations, the buffer size field is split into two portions, a BLIT_WIDTH value, which specifies the number of bytes in each "line" of the blit operation, and a NUMBER_LINES value, which specifies how many vertical rows of pixels are in the image buffer.

4.2.6.4.7 Payload Pointer

Some operations require additional control values that are stored in a Payload Buffer (shown in Table 4-9), which is pointed to by this field. After the DCP reads the control packet, it examines the Control0 register and determines whether any payload information is required. If so, the DCP loads the payload from the address specified in this field. (See Payload for more details.)

Table 4-9. DCP Payload Buffer Pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PAYLOAD_POINTER																															

The payload area is also written to by the DCP at the completion of a hash operation (when the HASH_TERM) bit is set. Software must allocate the appropriate amount of storage (20 bytes for SHA-1 and 4 bytes for CRC32) in the payload or risk having the DCP write to an unallocated address.

4.2.6.4.8 Status

After the DCP engine has completed processing of a descriptor, it writes the packet status (shown in Table 4-10) back to the descriptor In the Status field. The packet status is the value of the channel status register at the time the packet completed processing.

Table 4-10. DCP Status Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Table continues on the next page...

Table 4-10. DCP Status Field (continued)

TAG	ERROR_CODE										ERROR_DS	ERROR_SR	ERROR_PACKET	ERROR_SETU	HASH_MISMATCH	COMPLETE

For various error conditions, the DCP encodes additional error information in the ERROR_CODE field. See DCP Register Reference Index () for more details on assignments of error codes. For any completion codes besides the COMPLETE flag, the channel suspends processing of the command chain until the error status values are cleared in the channel status register.

The format for this field is the same as for the channel status register, with the exception that the COMPLETE bit is written to the payload status but is not present in the channel status register.

4.2.6.4.9 Payload

>The payload is a variable-length field that is used to provide key, initialization values, and expected results from hashing operations.

The payload may consist of the data fields listed in Table 4-11.

Table 4-11. DCP Payload Field

Field Name	Size	Description	Condition
Cipher Key	16 bytes	AES key	Cipher enable with PAYLOAD_KEY
Cipher IV	16 bytes	CBC initialization vector	Cipher enable with CBC mode.
Hash Check	20 bytes	Hash completion value	Hash enabled with HASH_TERM fields set.

If fields are not used, they do not appear in the payload and the other payload values move upwards in the payload area. For instance, if only hashing is used, then the HASH_CHECK value would appear at offset 0 in the payload area. Table 4-12 should be used by software to determine the amount of payload to allocate and initialize.

Table 4-12. DCP Payload Allocation by Software

Control Bits Present			Payload Size
Hash_Check	Cipher_Init	Payload_Key	
0	0	0	0 words / 0 bytes

Table continues on the next page...

Table 4-12. DCP Payload Allocation by Software (continued)

Control Bits Present			Payload Size
Hash_Check	Cipher_Init	Payload_Key	
0	0	1	4 words / 16 bytes
0	1	0	4 words / 16 bytes
0	1	1	8 words / 32 bytes
1	0	0	SHA-1: 5 words / 20 bytes SHA-2: 8 words / 32 bytes
1	0	1	SHA-1: 9 words / 36 bytes SHA-2:12 words / 48 bytes
1	1	0	SHA-1: 9 words / 36 bytes SHA-2: 12 words / 48 bytes
1	1	1	SHA-1: 13 words / 52 bytes SHA-2: 16 words / 64 bytes

For hashing operations, the DCP module writes the final hash value back to the start of the payload area for descriptors with the HASH_TERM bit set in the control packet. It is important that software allocate the required payload space, even though it is not required to set up the payload for control purposes.

4.2.7 Programming DCP Functions

4.2.7.1 Basic Memory Copy Programming Example

To perform a basic memcopy operation, only a single descriptor is required. The DCP simply copies data from the source to the destination buffer. This process is illustrated in Figure 4-6.

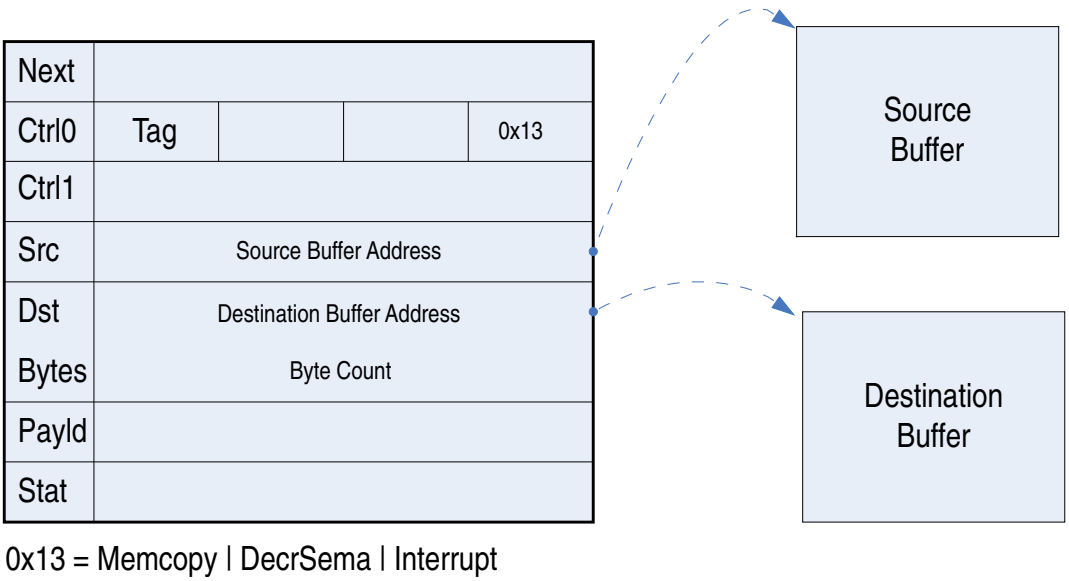


Figure 4-6. Basic Memory Copy Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32              *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32              *src,
                     *dst,
                      buf_size,
                     *payload,
                      stat;
}  DCP_DESCRIPTOR;
  DCP_DESCRIPTOR dcp1;
  u32 *srcbuffer, *dstbuffer;
  // set up control packet
  dcp1.next = 0;                    // single packet in chain
  dcp1.ctrl0.U = 0;                 // clear ctrl0 field
  dcp1.ctrl0.B.ENABLE_MEMCOPY = 1; // enable memcopy
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.INTERRUPT = 1;       // interrupt
  dcp1.ctrl1.U = 0;                 // clear ctrl1
  dcp1.src = srcbuffer;             // source buffer
  dcp1.dst = dstbuffer;             // destination buffer
  dcp1.buf_size = 512;              // 512 bytes
  dcp1.payload = NULL;              // not required
  dcp1.status = 0;                  // clear status
  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);     // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 1);          // increment semaphore by 1
  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);
</code>
```

4.2.7.2 Basic Hash Operation Programming Example

To perform a basic hash operation, only a single descriptor is required. The DCP simply reads data from the source buffer and computes the hash value on the contents. This process is illustrated in Figure 4-7.

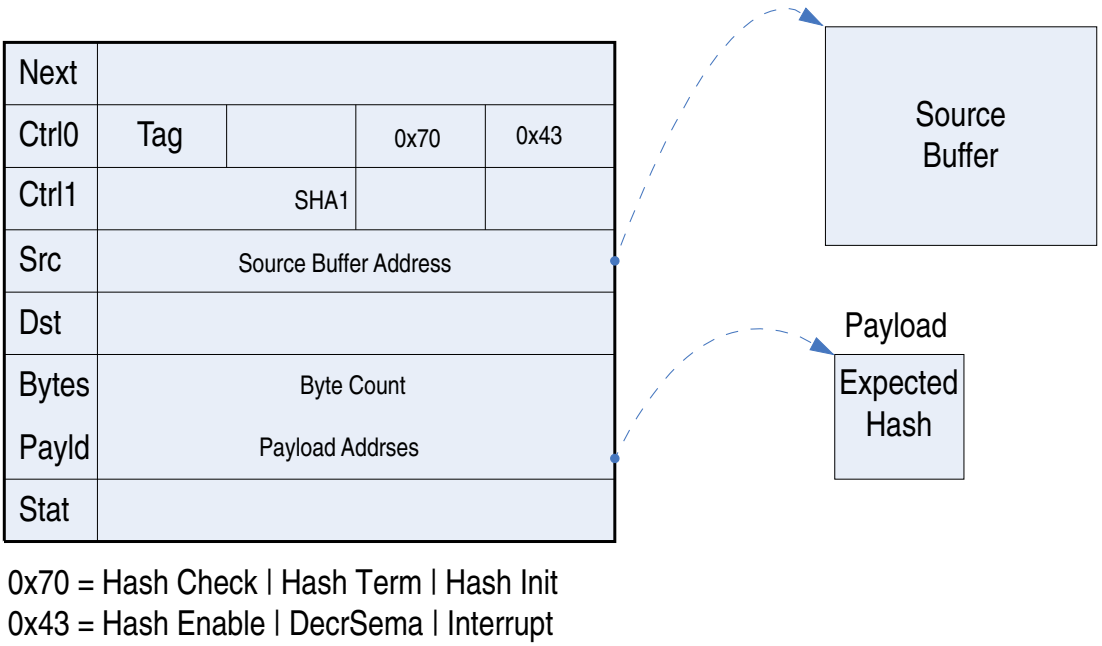


Figure 4-7. Basic Hash Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32               *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32               *src,
                      *dst,
                       buf_size,
                      *payload,
                       stat;
}  DCP_DESCRIPTOR;
  DCP_DESCRIPTOR dcp1;
  u32 *srcbuffer;
  u32 payload[5];
  // set up expected hash check value
  payload[0]=0x01234567;
  payload[1]=0x89ABCDEF;
  payload[2]=0x00112233;
  payload[3]=0x44556677;
  payload[4]=0x8899aabb;
  // set up control packet
  dcp1.next = 0;                    // single packet in chain
  dcp1.ctrl0.U = 0;                 // clear ctrl0 field
  dcp1.ctrl0.B.HASH_CHECK = 1;      // check hash when complete
  dcp1.ctrl0.B.HASH_INIT = 1;       // initialize hash with this block
  dcp1.ctrl0.B.HASH_TERM = 1;       // terminate hash with this block
  dcp1.ctrl0.B.ENABLE_HASH = 1;     // enable hash
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.INTERRUPT = 1;       // interrupt
  dcp1.ctrl1.U = 0;                 // clear ctrl1
  dcp1.src = srcbuffer;             // source buffer
```

```
  dcp1.dst = 0;                     // no destination buffer
  dcp1.buf_size = 512;              // 512 bytes
  dcp1.payload = payload;           // holds expected hash value
  dcp1.status = 0;                  // clear status
  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);     // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 1);          // increment semaphore by 1
  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);
</code>
```

4.2.7.3 Basic Cipher Operation Programming Example

To perform a basic cipher operation, only a single descriptor is required. The DCP reads data from the source buffer, encrypts it, and writes it to the destination buffer. For this example, the key is provided in the payload and the algorithm uses the AES CBC mode of operation. This requires a payload with both the key and CBC initialization value. This process is illustrated in Figure 4-8.

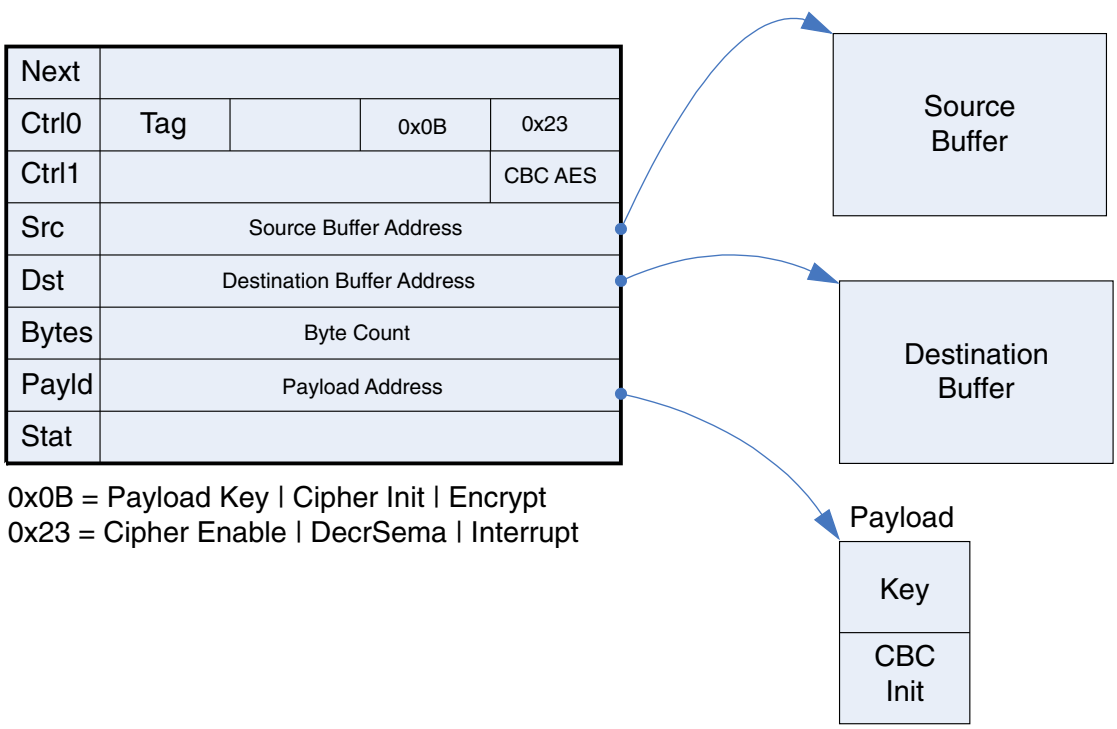


Figure 4-8. Basic Cipher Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32             *next;
```

```
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32              *src,
                     *dst,
                      buf_size,
                     *payload,
                      stat;
}  DCP_DESCRIPTOR;
  DCP_DESCRIPTOR dcp1;
  u32 *srcbuffer;
  u32 *dstbuffer;
  u32 payload[8];
  // set up key/CBC init in the payload
  payload[0]=0x01234567;           // key
  payload[1]=0x89ABCDEF;
  payload[2]=0xfedcba98;
  payload[3]=0x76543210;
  payload[4]=0x00112233;           // CBC initialization
  payload[5]=0x44556677;
  payload[6]=0x8899aabb;
  payload[7]=0xccddeeff;
  // set up control packet
  dcp1.next = 0;                   // single packet in chain
  dcp1.ctrl0.U = 0;                // clear ctrl0 field
  dcp1.ctrl0.B.PAYLOAD_KEY = 1;    // key is located in payload
  dcp1.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
  dcp1.ctrl0.B.CIPHER_INIT = 1;    // init CBC for this block (from payload)
  dcp1.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.INTERRUPT = 1;      // interrupt
  dcp1.ctrl1.U = 0;                // clear ctrl1
  dcp1.ctrl1.B.CIPHER_MODE = 1;    // select CBC mode of operation
  dcp1.src = srcbuffer;            // source buffer
  dcp1.dst = dstbuffer;            // destination buffer
  dcp1.buf_size = 512;             // 512 bytes
  dcp1.payload = payload;          // holds key/CBC init
  dcp1.status = 0;                 // clear status
  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);    // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 1);         // increment semaphore by 1
  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);
</code>
```

4.2.7.4 Multi-Buffer Scatter/Gather Cipher and Hash Operation Programming Example

For this example, three separate buffers are encrypted and hashed with the results being directed to a unified buffer (gather operation). Three descriptors are used for the operation because there are three separate source buffer pointers. The DCP reads data from the source buffer and computes a hash on the unencrypted data. It then encrypts the data and writes it to the destination buffer. For this example, the key is located in the key RAM, and the algorithm uses the AES CBC mode of operation with a SHA-1 hash. The

payload for the first operation contains the CBC initialization value, and the payload for the last packet contains the expected hash value. The middle packet requires no payload. This process is illustrated in Figure 4-9.

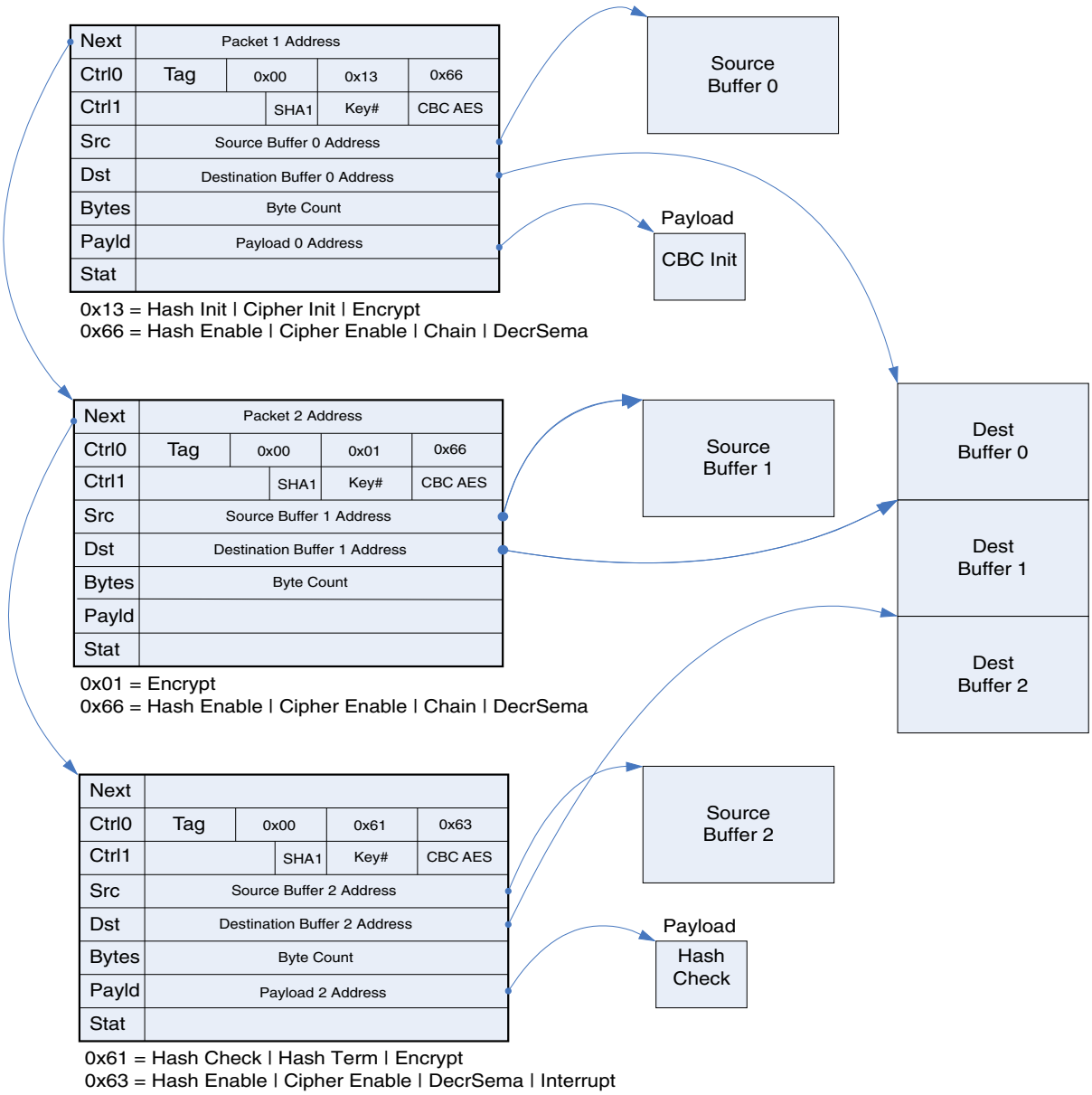


Figure 4-9. Multi-Buffer Scatter/Gather Cipher and Hash Operation

```
<code>
typedef struct _dcp_descriptor
{
    u32              *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32              *src,
                     *dst,
                      buf_size,
                     *payload,
                      stat;
}  DCP_DESCRIPTOR;
  DCP_DESCRIPTOR dcp1, dcp2, dcp3;
```

```
  u32 *srcbuffer0, *srcbuffer2, *srcbuffer3;
  u32 *dstbuffer;
  u32 payload0[4], payload2[5];
  // set up CBC init in the payload
  payload0[0]=0x01234567;           // key
  payload0[1]=0x89ABCDEF;
  payload0[2]=0xfedcba98;
  payload0[3]=0x76543210;
  payload2[0]=0x00112233;           // CBC initialization
  payload2[1]=0x44556677;
  payload2[2]=0x8899aabb;
  payload2[3]=0xccddeeff;
  payload2[3]=0xaabbccdd;
  // set up control packet
  dcp1.next = dcp2;                 // point to packet 2
  dcp1.ctrl0.U = 0;                 // clear ctrl0 field
  dcp1.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
  dcp1.ctrl0.B.CIPHER_INIT = 1;     // init CBC for this block (from payload)
  dcp1.ctrl0.B.HASH_INIT = 1;       // init hash this block
  dcp1.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
  dcp1.ctrl0.B.ENABLE_HASH = 1;     // enable cipher
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.CHAIN = 1;           // chain to next packet
  dcp1.ctrl1.U = 0;                 // clear ctrl1
  dcp1.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE__CBC; //select CBC mode of
operation
  dcp1.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT__SHA1;    // select SHA1 hash
  dcp1.ctrl1.B.KEY_SELECT  = 2;     // select key #2
  dcp1.src = srcbuffer0;             // source buffer
  dcp1.dst = dstbuffer;             // destination buffer
  dcp1.buf_size = 512;              // 512 bytes
  dcp1.payload = payload0;           // holds key/CBC init
  dcp1.status = 0;                  // clear status
  // set up control packet
  dcp2.next = dcp3;                 // point to packet 2
  dcp2.ctrl0.U = 0;                 // clear ctrl0 field
  dcp2.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
  dcp2.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
  dcp2.ctrl0.B.ENABLE_HASH = 1;     // enable cipher
  dcp2.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp2.ctrl0.B.CHAIN = 1;           // chain to next packet
  dcp2.ctrl1.U = 0;                 // clear ctrl1
  dcp2.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE__CBC; //select CBC mode of
operation
  dcp2.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT__SHA1;    // select SHA1 hash
  dcp2.ctrl1.B.KEY_SELECT  = 2;     // select key #2
  dcp2.src = srcbuffer1;            // source buffer
  dcp2.dst = dstbuffer+512;         // destination buffer
  dcp2.buf_size = 512;              // 512 bytes
  dcp2.payload = NULL;              // no payload required
  dcp2.status = 0;                  // clear status
  // set up control packet
  dcp3.next = dcp3;                 // point to packet 2
  dcp3.ctrl0.U = 0;                 // clear ctrl0 field
  dcp3.ctrl0.B.HASH_TERM      = 1; // terminate hash block
  dcp3.ctrl0.B.HASH_CHECK     = 1; // check hash against payload value
  dcp3.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
  dcp3.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
  dcp3.ctrl0.B.ENABLE_HASH = 1;     // enable cipher
  dcp3.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp3.ctrl0.B.INTERRUPT = 1;       // interrupt on completion
  dcp3.ctrl1.U = 0;                 // clear ctrl1
  dcp3.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE__CBC; //select CBC mode of
operation
  dcp3.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT__SHA1;    // select SHA1 hash
  dcp3.ctrl1.B.KEY_SELECT  = 2;     // select key #2
  dcp3.src = srcbuffer1;            // source buffer
  dcp3.dst = dstbuffer+1024;        // destination buffer
  dcp3.buf_size = 512;              // 512 bytes
  dcp3.payload = payload2;          // payload is hash check value
```

```
  dcp3.status = 0;                  // clear status
  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);     // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 3);          // increment semaphore by 3 (for 3 packets)
  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);
</code>
```

4.3 Programmable Registers

DCP Hardware Register Format Summary

DCP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	DCP Control Register 0 (DCP_CTRL)	32	R/W	F080_0000h	4.3.1/82
10	DCP Status Register (DCP_STAT)	32	R/W	1000_0000h	4.3.2/84
20	DCP Channel Control Register (DCP_CHANNELCTRL)	32	R/W	0000_0000h	4.3.3/87
30	DCP Capability 0 Register (DCP_CAPABILITY0)	32	R/W	0000_0404h	4.3.4/88
40	DCP Capability 1 Register (DCP_CAPABILITY1)	32	R	0007_0001h	4.3.5/89
50	DCP Context Buffer Pointer (DCP_CONTEXT)	32	R/W	0000_0000h	4.3.6/90
60	DCP Key Index (DCP_KEY)	32	R/W	0000_0000h	4.3.7/90
70	DCP Key Data (DCP_KEYDATA)	32	R/W	0000_0000h	4.3.8/91
80	DCP Work Packet 0 Status Register (DCP_PACKET0)	32	R	0000_0000h	4.3.9/92
90	DCP Work Packet 1 Status Register (DCP_PACKET1)	32	R	0000_0000h	4.3.10/92
A0	DCP Work Packet 2 Status Register (DCP_PACKET2)	32	R	0000_0000h	4.3.11/96
B0	DCP Work Packet 3 Status Register (DCP_PACKET3)	32	R	0000_0000h	4.3.12/97
C0	DCP Work Packet 4 Status Register (DCP_PACKET4)	32	R	0000_0000h	4.3.13/97
D0	DCP Work Packet 5 Status Register (DCP_PACKET5)	32	R	0000_0000h	4.3.14/98
E0	DCP Work Packet 6 Status Register (DCP_PACKET6)	32	R	0000_0000h	4.3.15/98
100	DCP Channel 0 Command Pointer Address Register (DCP_CH0CMDPTR)	32	R/W	0000_0000h	4.3.16/99
110	DCP Channel 0 Semaphore Register (DCP_CH0SEMA)	32	R/W	0000_0000h	4.3.17/100
120	DCP Channel 0 Status Register (DCP_CH0STAT)	32	R/W	0000_0000h	4.3.18/101
130	DCP Channel 0 Options Register (DCP_CH0OPTS)	32	R/W	0000_0000h	4.3.19/103
140	DCP Channel 1 Command Pointer Address Register (DCP_CH1CMDPTR)	32	R/W	0000_0000h	4.3.20/104

Table continues on the next page...

DCP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
150	DCP Channel 1 Semaphore Register (DCP_CH1SEMA)	32	R/W	0000_0000h	4.3.21/105
160	DCP Channel 1 Status Register (DCP_CH1STAT)	32	R/W	0000_0000h	4.3.22/106
170	DCP Channel 1 Options Register (DCP_CH1OPTS)	32	R/W	0000_0000h	4.3.23/108
180	DCP Channel 2 Command Pointer Address Register (DCP_CH2CMDPTR)	32	R/W	0000_0000h	4.3.24/109
190	DCP Channel 2 Semaphore Register (DCP_CH2SEMA)	32	R/W	0000_0000h	4.3.25/110
1A0	DCP Channel 2 Status Register (DCP_CH2STAT)	32	R/W	0000_0000h	4.3.26/111
1B0	DCP Channel 2 Options Register (DCP_CH2OPTS)	32	R/W	0000_0000h	4.3.27/113
1C0	DCP Channel 3 Command Pointer Address Register (DCP_CH3CMDPTR)	32	R/W	0000_0000h	4.3.28/114
1D0	DCP Channel 3 Semaphore Register (DCP_CH3SEMA)	32	R/W	0000_0000h	4.3.29/115
1E0	DCP Channel 3 Status Register (DCP_CH3STAT)	32	R/W	0000_0000h	4.3.30/116
1F0	DCP Channel 3 Options Register (DCP_CH3OPTS)	32	R/W	0000_0000h	4.3.31/118
400	DCP Debug Select Register (DCP_DBGSELECT)	32	R/W	0000_0000h	4.3.32/118
410	DCP Debug Data Register (DCP_DBGDATA)	32	R	0000_0000h	4.3.33/119
420	DCP Page Table Register (DCP_PAGETABLE)	32	R/W	0000_0000h	4.3.34/119
430	DCP Version Register (DCP_VERSION)	32	R	0201_0000h	4.3.35/120

4.3.1 DCP Control Register 0 (DCP_CTRL)

The CTRL register contains controls for the DCP module.

CTRL: 0x000

CTRL_SET: 0x004

CTRL_CLR: 0x008

CTRL_TOG: 0x00C

The Control register contains the primary controls for the DCP block. The present bits indicate which of the sub-features of the block are present in the hardware. The context control bits control how the DCP utilizes it's context buffer and the gather residual writes bit controls how the master handles writing misaligned data to the bus. Each channel and the color-space converter contains an independent interrupt enable.

EXAMPLE

```
DCP_CTRL_SET(BM_DCP_CTRL_SFTRST);
DCP_CTRL_CLR(BM_DCP_CTRL_SFTRST | BM_DCP_CTRL_CLKGATE);
```

Address: 0h base + 0h offset = 0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	PRESENT_CRYPTO	PRESENT_SHA	Reserved				GATHER_RESIDUAL_WRITES	ENABLE_CONTEXT_CACHING	ENABLE_CONTEXT_SWITCHING	Reserved				
W																
Reset	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							Reserved	CHANNEL_INTERRUPT_ENABLE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CTRL field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal DCP operation. Set this bit to one (default) to disable clocking with the DCP and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the DCP block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 PRESENT_ CRYPTO	Indicates whether the crypto (Cipher/Hash) functions are present. 0x1 **Present** — 0x0 **Absent** —
28 PRESENT_SHA	Indicates whether the SHA1/SHA2 functions are present. 0x1 **Present** — 0x0 **Absent** —
27–24 -	This field is reserved. Reserved, always set to zero.
23 GATHER_ RESIDUAL_ WRITES	Software should set this bit to enable ragged writes to unaligned buffers to be gathered between multiple write operations. This improves performance by removing several byte operations between write bursts. Trailing byte writes are held in a residual write data buffer and combined with a subsequent write to the buffer to form a word write.
22 ENABLE_ CONTEXT_ CACHING	Software should set this bit to enable caching of contexts between operations. If only a single channel is used for encryption/hashing, enabling caching causes the context to not be reloaded if the channel was the last to be used.
21 ENABLE_ CONTEXT_ SWITCHING	Enable automatic context switching for the channels. Software should set this bit if more than one channel is doing hashing or cipher operations that require context to be saved (for instance, when CBC mode is enabled). By disabling context switching, software can save the 208 bytes used for the context buffer.
20–9 -	This field is reserved. Reserved, always set to zero.
8 RSVD_CSC_ INTERRUPT_ ENABLE	This field is reserved. Reserved, always set to zero.
7–0 CHANNEL_ INTERRUPT_ ENABLE	Per-channel interrupt enable bit. When set, the channel's interrupt will get routed to the interrupt controller. Channel 0 is routed to the dcp_vmi_irq signal and the other channels are combined (along with the CRC interrupt) into the dcp_irq signal. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —

4.3.2 DCP Status Register (DCP_STAT)

The DCP Interrupt Status register provides channel interrupt status information.

STAT: 0x010

STAT_SET: 0x014

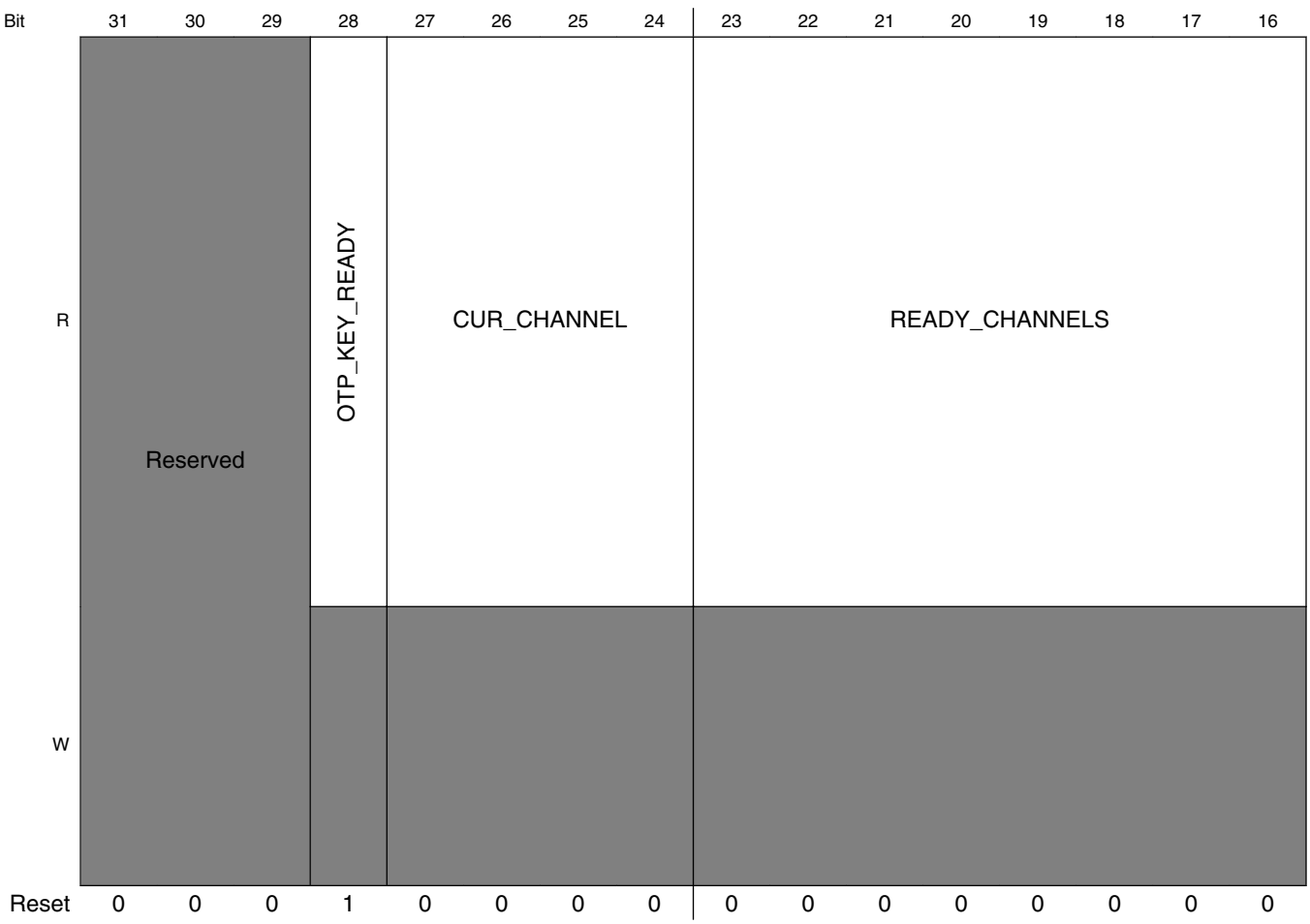
STAT_CLR: 0x018

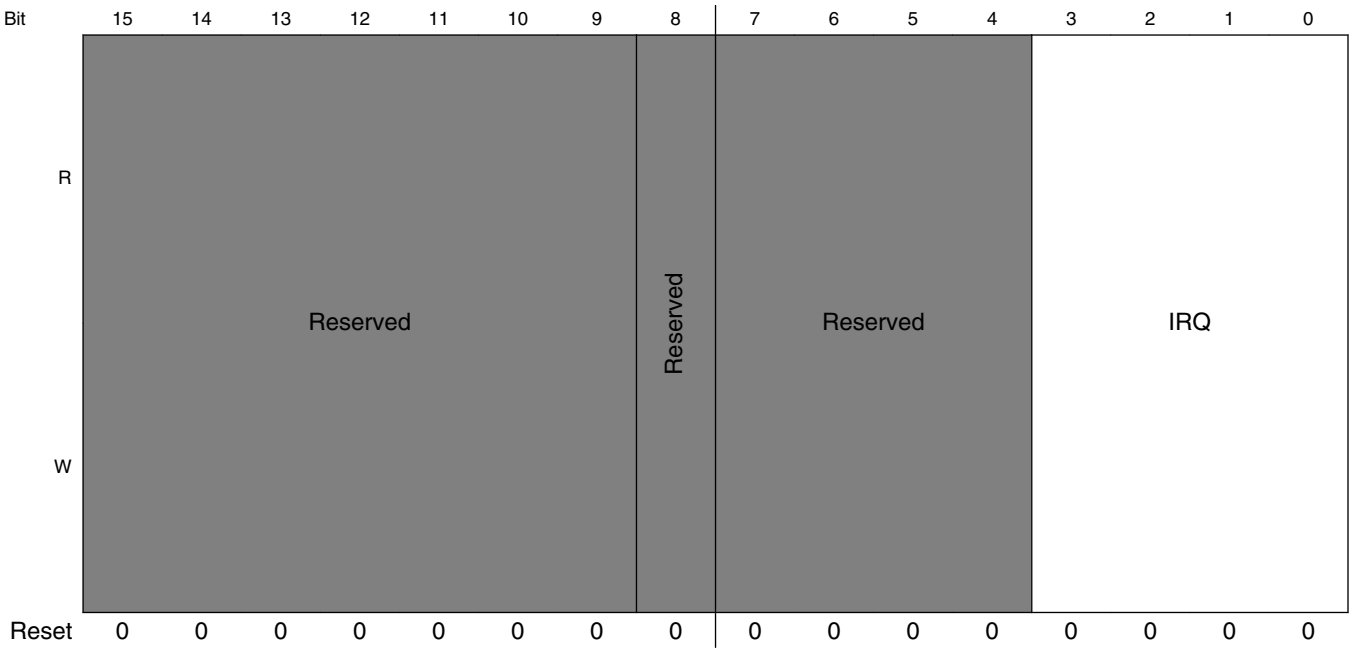
STAT_TOG: 0x01C

This register provides status feedback indicating the channel currently performing an operation and which channels have pending operations.

EXAMPLE

Address: 0h base + 10h offset = 10h





DCP_STAT field descriptions

Field	Description
31–29 -	This field is reserved. Reserved, always set to zero.
28 OTP_KEY_READY	When set, indicates that the OTP key has been shifted from the fuse block and is ready for use.
27–24 CUR_CHANNEL	Current (active) channel (encoded). 0x0 **None** — 0x1 **CH0** — 0x2 **CH1** — 0x3 **CH2** — 0x4 **CH3** —
23–16 READY_CHANNELS	Indicates which channels are ready to proceed with a transfer (active channel also included). Each bit is a one-hot indicating the request status for the associated channel. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —
15–9 -	This field is reserved. Reserved, always set to zero.
8 RSVD_IRQ	This field is reserved. Reserved, always set to zero.
7–4 -	This field is reserved. Reserved, always set to zero.
3–0 IRQ	Indicates which channels have pending interrupt requests. Channel 0's interrupt is routed through the dcp_vmi_irq and the other interrupt bits are routed through the dcp_irq.

4.3.3 DCP Channel Control Register (DCP_CHANNELCTRL)

The DCP Channel Control register provides controls for channel arbitration and channel enables.

CHANNELCTRL: 0x020

CHANNELCTRL_SET: 0x024

CHANNELCTRL_CLR: 0x028

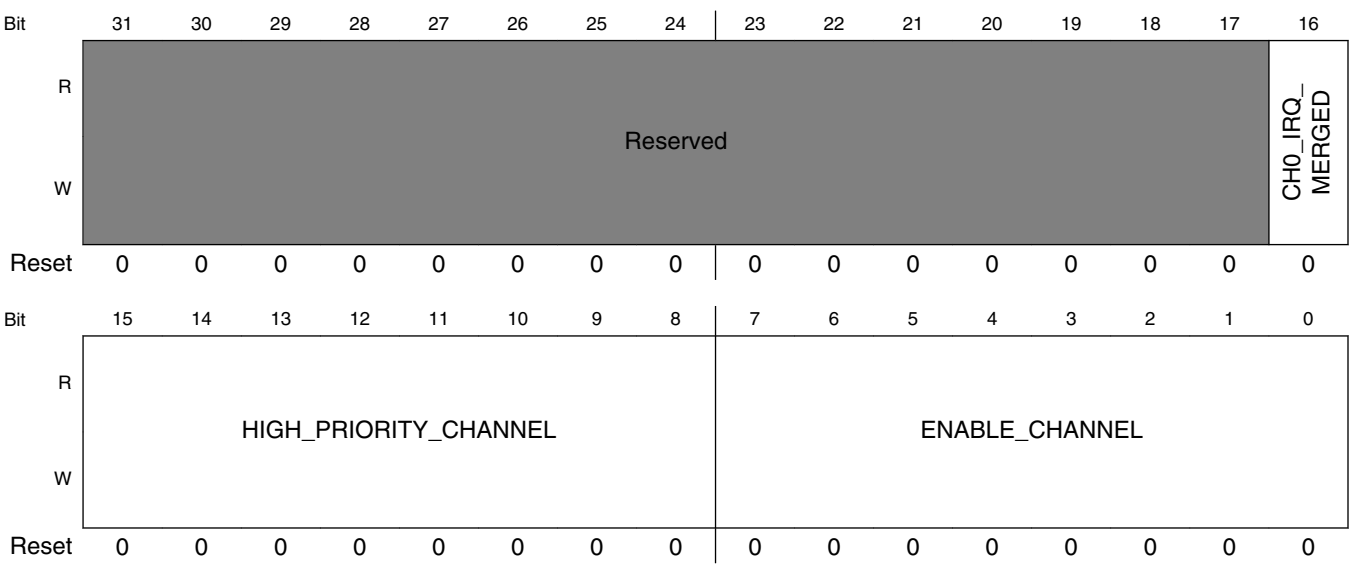
CHANNELCTRL_TOG: 0x02C

This register provides status feedback indicating the channel currently performing an operation and which channels have pending operations.

EXAMPLE

```
        BW_DCP_CHANNELCTRL_ENABLE_CHANNEL(BV_DCP_CHANNELCTRL_ENABLE_CHANNEL__CH0);  //
enable channel 0
```

Address: 0h base + 20h offset = 20h



DCP_CHANNELCTRL field descriptions

Field	Description
31–17 RSVD	This field is reserved. Reserved, always set to zero.
16 CH0_IRQ_MERGED	Indicates that the interrupt for channel 0 should be merged with the other interrupts on the shared dcp_irq interrupt. When set to 0, channel 0's interrupt will be routed to the separate dcp_vmi_irq. When set to 1, the interrupt will be routed to the shared DCP interrupt.

Table continues on the next page...

DCP_CHANNELCTRL field descriptions (continued)

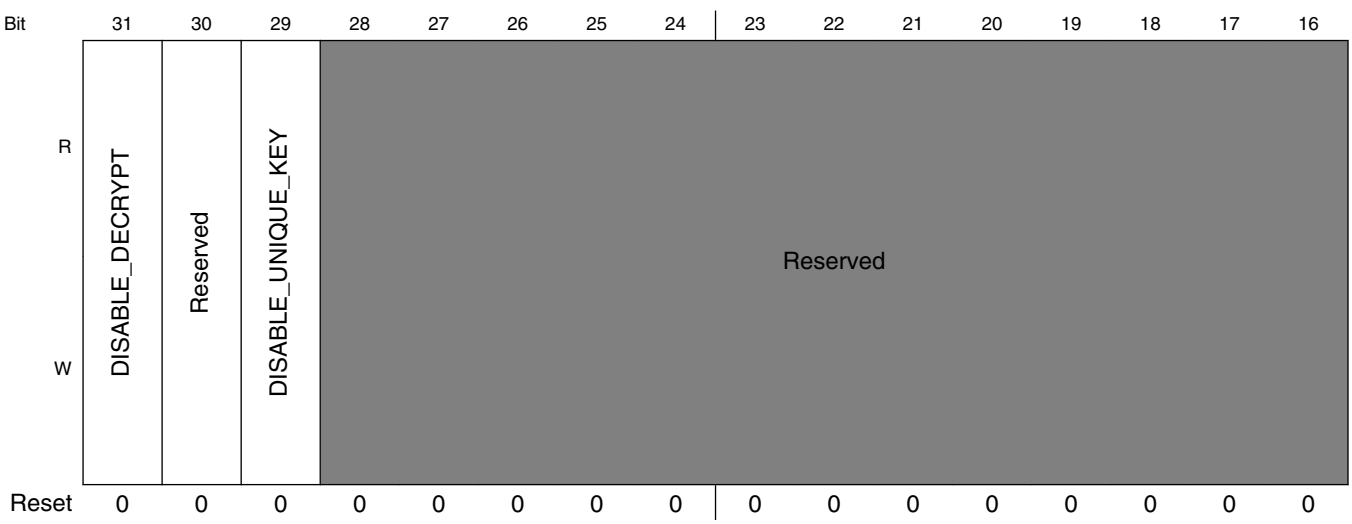
Field	Description
15–8 HIGH_ PRIORITY_ CHANNEL	Setting a bit in this field causes the corresponding channel to have high-priority arbitration. High priority channels will be arbitrated round-robin and will take precedence over other channels that are not marked as high priority. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —
7–0 ENABLE_ CHANNEL	Setting a bit in this field will enabled the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When not enabled, the channel is denied access to the central DMA resources. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —

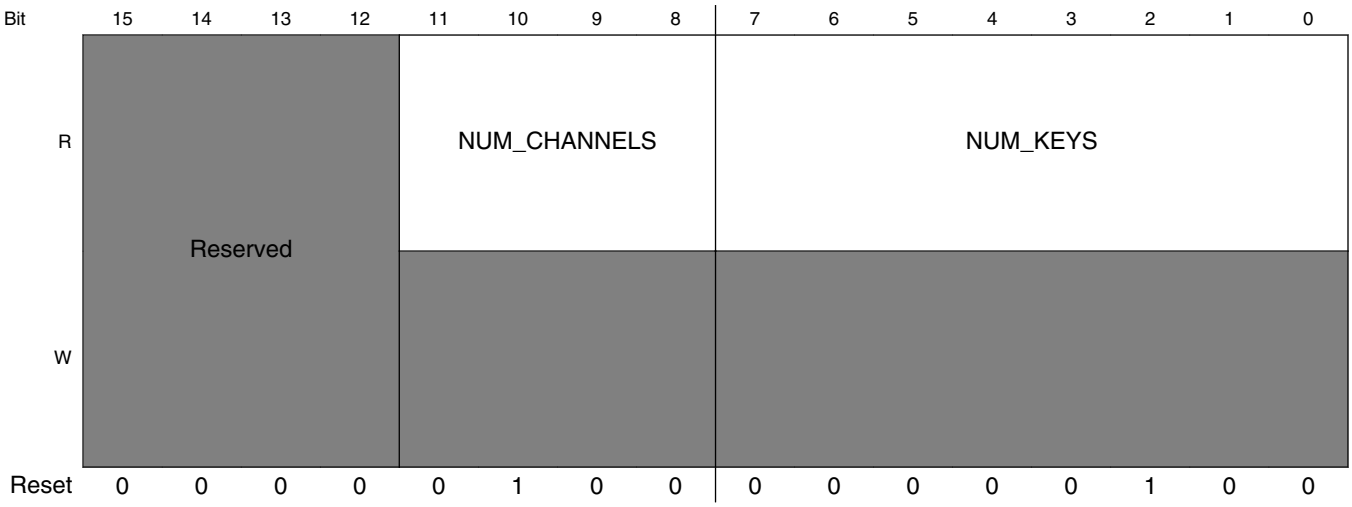
4.3.4 DCP Capability 0 Register (DCP_CAPABILITY0)

This register contains additional information about the DCP module implementation parameters.

This register provides capability information for the DCP block. It indicates the number of channels implemented as well as the number of key storage locations available for software use.

Address: 0h base + 30h offset = 30h





DCP_CAPABILITY0 field descriptions

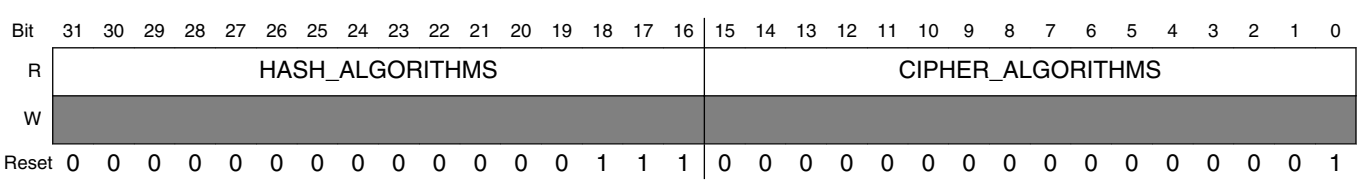
Field	Description
31 DISABLE_ DECRYPT	Write to a 1 to disable decryption. This bit can only be written by secure software and the value can only be cleared by a reset.
30 Reserved	This field is reserved. -
29 DISABLE_ UNIQUE_KEY	Write to a 1 disable the per-device unique key. The device-specific hardware key may be selected by using a value of 0xFE in the key select field.
28–12 RSVD	This field is reserved. Reserved, always set to zero.
11–8 NUM_ CHANNELS	Encoded value indicating the number of channels implemented in the design.
7–0 NUM_KEYS	Encoded value indicating the number of key storage locations implemented in the design.

4.3.5 DCP Capability 1 Register (DCP_CAPABILITY1)

This register contains information about the algorithms available on the implementation.

This register provides capability information for the DCP block. It contains two fields indicating which encryption and hashing algorithms are present in the design. Each bit set indicates that support for the associated function is present.

Address: 0h base + 40h offset = 40h



DCP_CAPABILITY1 field descriptions

Field	Description
31–16 HASH_ ALGORITHMS	One-hot field indicating which hashing features are implemented in HW. 0x0001 **SHA1** — 0x0002 **CRC32** — 0x0004 **SHA256** —
15–0 CIPHER_ ALGORITHMS	One-hot field indicating which cipher algorithms are available. 0x0001 **AES128** —

4.3.6 DCP Context Buffer Pointer (DCP_CONTEXT)

This register contains a pointer to the memory region to be used for DCP context swap operations.

This register contains a pointer to the start of the context pointer memory in on-chip SRAM or off-chip SDRAM. This buffer will be used to store state information when the DCP module changes from one channel to another.

Address: 0h base + 50h offset = 50h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CONTEXT field descriptions

Field	Description
31–0 ADDR	Context pointer address. Address should be located in system RAM and should be word-aligned for optimal performance.

4.3.7 DCP Key Index (DCP_KEY)

This register contains a pointer to the key location to be written.

The DCP module maintains a set of write-only keys that may be used by software. To write a key, software must first write the desired key index/subword to this register and then write the key values to the key registers (below). After each write to the key data register, the SUBWORD field will increment to allow software to write the subsequent key to be written without having to rewrite the key index.

EXAMPLE

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeff
DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0));  // set key index to key 0, subword
0
DCP_KEYDATA_WR(0xccddeeff);  // write key values (subword 0)
DCP_KEYDATA_WR(0x8899aabb);  // write key values (subword 1)
DCP_KEYDATA_WR(0x44556677);  // write key values (subword 2)
DCP_KEYDATA_WR(0x00112233);  // write key values (subword 3)
```

Address: 0h base + 60h offset = 60h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								Reserved		INDEX		Reserved		SUBWORD	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_KEY field descriptions

Field	Description
31–8 RSVD	This field is reserved. Reserved, always set to zero.
7–6 RSVD_INDEX	This field is reserved. Reserved, always set to zero.
5–4 INDEX	Key index pointer. Valid indices are 0-[number_keys].
3–2 RSVD_ SUBWORD	This field is reserved. Reserved, always set to zero.
1–0 SUBWORD	Key subword pointer. Valid indices are 0-3. After each write to the key data register, this field will increment.

4.3.8 DCP Key Data (DCP_KEYDATA)

This register provides write access to the key/key subword specified by the Key Index Register.

Writing this location updates the selected subword for the key located at the index specified by the Key Index Register. A write also triggers the SUBWORD field of the KEY register to increment to the next higher word in the key.

EXAMPLE

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeff
DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0));  // set key index to key 0, subword
0
DCP_KEYDATA_WR(0xccddeeff);  // write key values (subword 0)
DCP_KEYDATA_WR(0x8899aabb);  // write key values (subword 1)
DCP_KEYDATA_WR(0x44556677);  // write key values (subword 2)
```

```
DCP_KEYDATA_WR(0x00112233);  // write key values (subword 3)
```

Address: 0h base + 70h offset = 70h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_KEYDATA field descriptions

Field	Description
31–0 DATA	Word 0 data for key. This is the least-significant word.

4.3.9 DCP Work Packet 0 Status Register (DCP_PACKET0)

This register displays the values for the current work packet offset 0x00 (Next Command) field.

The Work Packet Status Registers show the contents of the currently executing packet. When the channels are inactive, the packet status register return 0. The register bits are fully documented here to document the packet structure in memory.

Address: 0h base + 80h offset = 80h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_PACKET0 field descriptions

Field	Description
31–0 ADDR	Next Pointer Register,

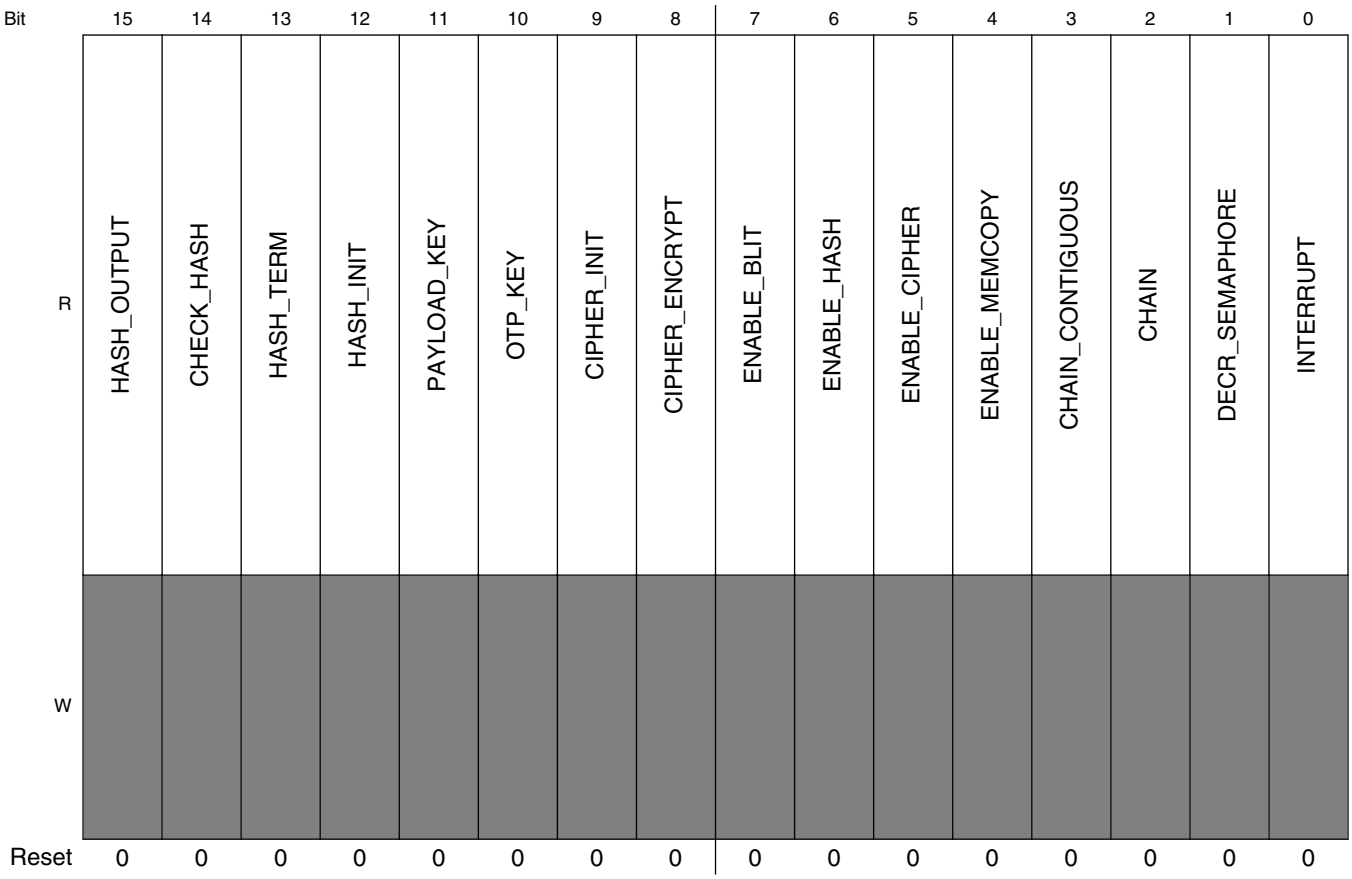
4.3.10 DCP Work Packet 1 Status Register (DCP_PACKET1)

This register displays the values for the current work packet offset 0x04 (control) field.

This register shows the contents of the Control0 register from the packet being processed.

Address: 0h base + 90h offset = 90h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TAG								OUTPUT_WORDSWAP	OUTPUT_BYTESWAP	INPUT_WORDSWAP	INPUT_BYTESWAP	KEY_WORDSWAP	KEY_BYTESWAP	TEST_SEMA_IRQ	CONSTANT_FILL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DCP_PACKET1 field descriptions

Field	Description
31–24 TAG	Packet Tag
23 OUTPUT_ WORDSWAP	Reflects whether the DCP engine will wordswap output data (big-endian data).
22 OUTPUT_ BYTESWAP	Reflects whether the DCP engine will byteswap output data (big-endian data).
21 INPUT_ WORDSWAP	Reflects whether the DCP engine will wordswap input data (big-endian data).
20 INPUT_ BYTESWAP	Reflects whether the DCP engine will byteswap input data (big-endian data).
19 KEY_ WORDSWAP	Reflects whether the DCP engine will swap key words (big-endian key).
18 KEY_ BYTESWAP	Reflects whether the DCP engine will swap key bytes (big-endian key).

Table continues on the next page...

DCP_PACKET1 field descriptions (continued)

Field	Description
17 TEST_SEMA_ IRQ	This bit is used to test the channel semaphore transition to 0. FOR TEST USE ONLY!
16 CONSTANT_ FILL	When this bit is set (MEMCOPY and BLIT modes only), the DCP will simply fill the destination buffer with the value found in the Source Address field.
15 HASH_OUTPUT	When hashing is enabled, this bit controls whether the input or output data is hashed. 0 **INPUT** — 1 **OUTPUT** —
14 CHECK_HASH	Reflects whether the calculated hash value should be compared against the hash provided in the payload.
13 HASH_TERM	Reflects whether the current hashing block is the final block in the hashing operation, so the hash padding should be applied by hardware.
12 HASH_INIT	Reflects whether the current hashing block is the initial block in the hashing operation, so the hash registers should be initialized before the operation.
11 PAYLOAD_KEY	When set, indicates the payload contains the key. This bit takes precedence over the OTP_KEY control
10 OTP_KEY	Reflects whether a hardware-based key should be used. The KEY_SELECT field from the Control1 field is used to select from multiple hardware keys. The PAYLOAD_KEY bit takes precedence over the OTP_KEY bit.
9 CIPHER_INIT	Reflects whether the cipher block should load the initialization vector from the payload for this operation.
8 CIPHER_ ENCRYPT	When the cipher block is enabled, this bit indicates whether the operation is encryption or decryption. 1 **ENCRYPT** — 0 **DECRYPT** —
7 ENABLE_BLIT	Reflects whether the DCP should perform a blit operation. Source data is always continuous and the destination buffer is written in run/stride format. When set, the BUFFER_SIZE field is treated as two 16-bit values for the X-Y extents of the blit operation.
6 ENABLE_HASH	Reflects whether the selected hashing function should be enabled for this operation.
5 ENABLE_ CIPHER	Reflects whether the selected cipher function should be enabled for this operation.
4 ENABLE_ MEMCOPY	Reflects whether the selected hashing function should be enabled for this operation.
3 CHAIN_ CONTIGUOUS	Reflects whether the next packet's address is located following this packet's payload.
2 CHAIN	Reflects whether the next command pointer register should be loaded into the channel's current descriptor pointer.
1 DECR_ SEMAPHORE	Reflects whether the channel's semaphore should be decremented at the end of the current operation. When the semaphore reaches a value of zero, no more operations will be issued from the channel.

Table continues on the next page...

DCP_PACKET1 field descriptions (continued)

Field	Description
0 INTERRUPT	Reflects whether the channel should issue an interrupt upon completion of the packet.

4.3.11 DCP Work Packet 2 Status Register (DCP_PACKET2)

This register displays the values for the current work packet offset 0x08 (Control1) field.

This register shows the contents of the Control0 register from the packet being processed.

Address: 0h base + A0h offset = A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CIPHER_CFG								Reserved				HASH_SELECT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	KEY_SELECT								CIPHER_MODE				CIPHER_SELECT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_PACKET2 field descriptions

Field	Description
31–24 CIPHER_CFG	Cipher configuration bits. Optional configuration bits required for ciphers
23–20 RSVD	This field is reserved. Reserved, always set to zero.
19–16 HASH_SELECT	Hash Selection Field 0x00 **SHA1** — 0x01 **CRC32** — 0x02 **SHA256** —
15–8 KEY_SELECT	Key Selection Field. The value here reflects the key index for the cipher operation. Values 0-3 refer to the software keys that can be written to the key RAM. The OTP key or the unique device-specific key may also be selected with a value of 0xFF (OTP key) or 0xFE (unique key). 0x00 **KEY0** — 0x01 **KEY1** — 0x02 **KEY2** — 0x03 **KEY3** — 0xFE **UNIQUE_KEY** — 0xFF **OTP_KEY** —
7–4 CIPHER_MODE	Cipher Mode Selection Field. Reflects the mode of operation for cipher operations.

Table continues on the next page...

DCP_PACKET2 field descriptions (continued)

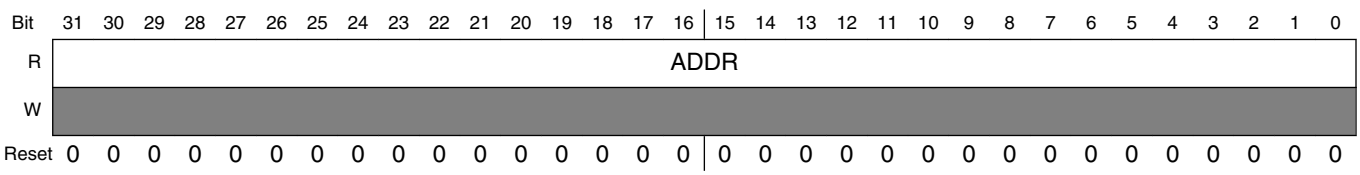
Field	Description
	0x00 **ECB** — 0x01 **CBC** —
3–0 CIPHER_ SELECT	Cipher Selection Field 0x00 **AES128** —

4.3.12 DCP Work Packet 3 Status Register (DCP_PACKET3)

This register displays the values for the current work packet offset 0x0C (Source Address) field.

This register shows the contents of the Source Address register from the packet being processed. When the CONSTANT_FILL bit in the Control 0 field is set, this field contains the data written to the destination buffer.

Address: 0h base + B0h offset = B0h



DCP_PACKET3 field descriptions

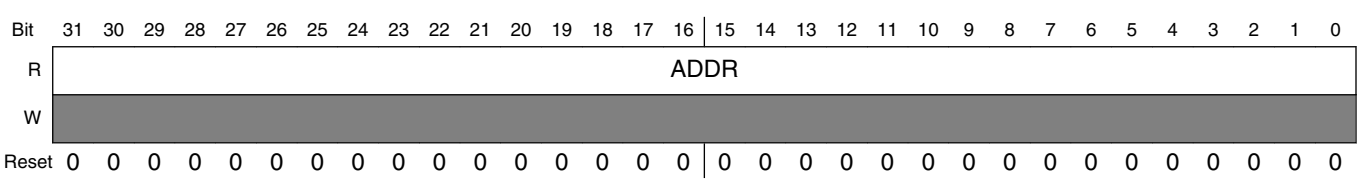
Field	Description
31–0 ADDR	Source Buffer Address Pointer. This value is the working value and will update as the operation proceeds.

4.3.13 DCP Work Packet 4 Status Register (DCP_PACKET4)

This register displays the values for the current work packet offset 0x10 (Destination Address) field.

This register shows the contents of the Destination Address register from the packet being processed.

Address: 0h base + C0h offset = C0h



DCP_PACKET4 field descriptions

Field	Description
31–0 ADDR	Destination Buffer Address Pointer. This value is the working value and will update as the operation proceeds.

4.3.14 DCP Work Packet 5 Status Register (DCP_PACKET5)

This register displays the values for the current work packet offset 0x14 (Buffer Size) field.

This register shows the contents of the bytecount register from the packet being processed. The field can be considered either a byte count or a buffer size. The logic treats this as a decrmenting count of bytes from the buffer size programmed into the field. As the transaction proceeds, the logic will decrement the bytecount as data is written to the destination buffer. For blit operations, the top 16-bits of this field represents the number of lines (y size) in the blit and the lower 16-bits represent the number of bytes in a line (x size).

Address: 0h base + D0h offset = D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_PACKET5 field descriptions

Field	Description
31–0 COUNT	Byte Count register. This value is the working value and will update as the operation proceeds.

4.3.15 DCP Work Packet 6 Status Register (DCP_PACKET6)

This register displays the values for the current work packet offset 0x1C (Payload Pointer) field.

This register shows the contents of the payload pointer fieldr from the packet being processed.

Address: 0h base + E0h offset = E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_PACKET6 field descriptions

Field	Description
31–0 ADDR	This regiser reflects the payload pointer for the current control packet.

4.3.16 DCP Channel 0 Command Pointer Address Register (DCP_CH0CMDPTR)

The DCP channel 0 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 0 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
            DCP_CHnCMDPTR_WR(0, v);    // Write channel 0 command pointer
            pCurptr = (DCP_chncmdptr_t *) DCP_CHnCMDPTR_RD(0);  // Read current command
pointer
```

Address: 0h base + 100h offset = 100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH0CMDPTR field descriptions

Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 0.

4.3.17 DCP Channel 0 Semaphore Register (DCP_CH0SEMA)

The DCP Channel 0 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjuction with the "decrement_semaphore" bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. After processing each control packet, the DCP decrements the semaphore if it is non-zero. The channel will continue processing packets as long as the semaphore contains a non-zero value and the CHAIN or CHAIN_CONTIGOUS control bits in the Control0 field are set.

Address: 0h base + 110h offset = 110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								VALUE								Reserved								INCREMENT							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH0SEMA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the DCP_CHnSEMA_CLR register.

4.3.18 DCP Channel 0 Status Register (DCP_CH0STAT)

The DCP Channel 0 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

DCP_CH0STAT: 0x120

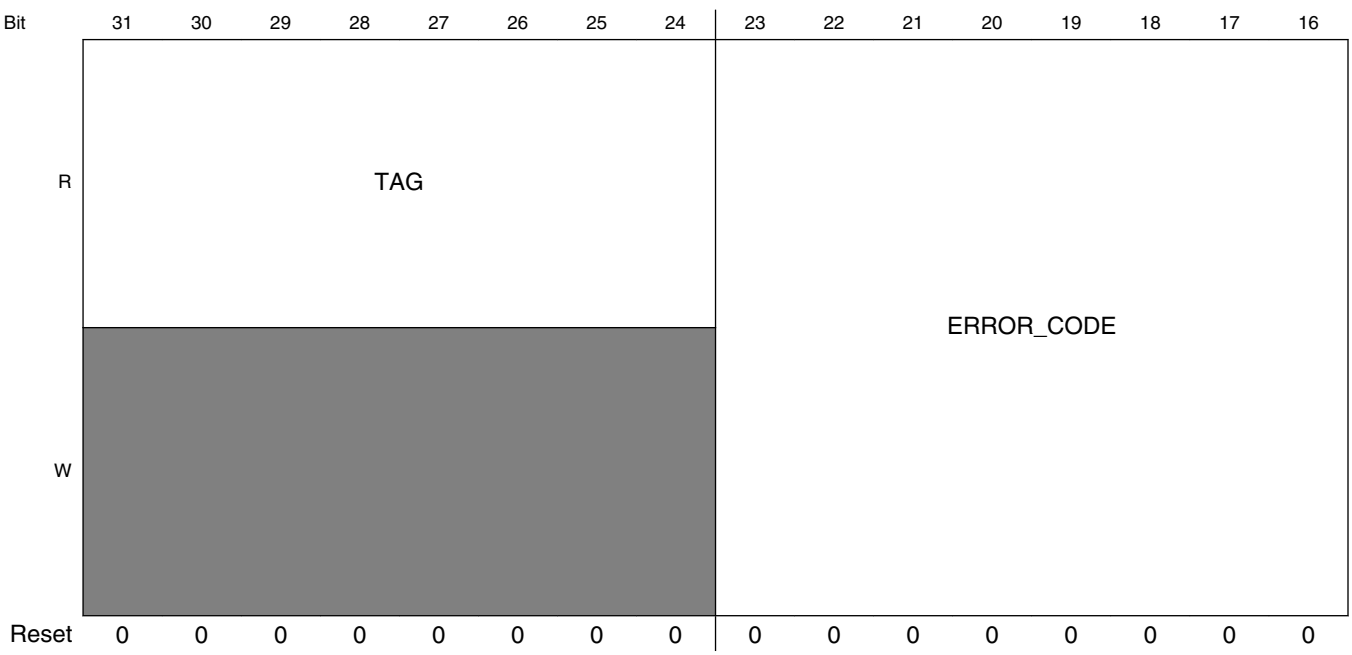
CH0STAT_SET: 0x124

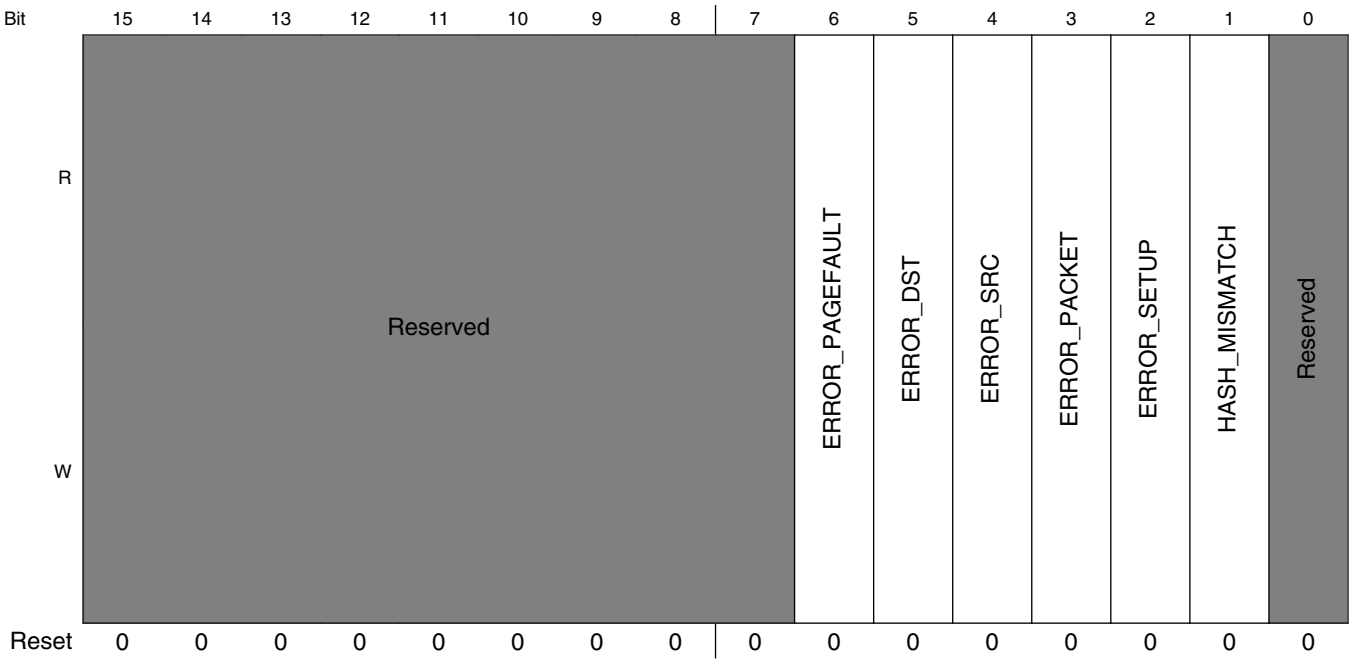
CH0STAT_CLR: 0x128

CH0STAT_TOG: 0x12C

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: 0h base + 120h offset = 120h





DCP_CH0STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 -	This field is reserved. Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a a bus error occurred when reading the packet or payload or when writing status back to the packet payload. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware has detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.

Table continues on the next page...

DCP_CH0STAT field descriptions (continued)

Field	Description
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This field is reserved. This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

4.3.19 DCP Channel 0 Options Register (DCP_CH0OPTS)

The DCP Channel 0 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

DCP_CH0OPTS: 0x130

CH0OPTS_SET: 0x134

CH0OPTS_CLR: 0x138

CH0OPTS_TOG: 0x13C

The options register can be used to control optional features of the channels.

Address: 0h base + 130h offset = 130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																RECOVERY_TIMER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH0OPTS field descriptions

Field	Description
31–16 RSVD	This field is reserved. Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate another operation for the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

4.3.20 DCP Channel 1 Command Pointer Address Register (DCP_CH1CMDPTR)

The DCP channel 1 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 1 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
            DCP_CHn_CMDPTR_WR(1, v);    // Write channel 1 command pointer
            pCurptr = (DCP_chn_cmdptr_t *) DCP_CHn_CMDPTR_RD(1);  // Read current command
pointer
```

Address: 0h base + 140h offset = 140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH1CMDPTR field descriptions

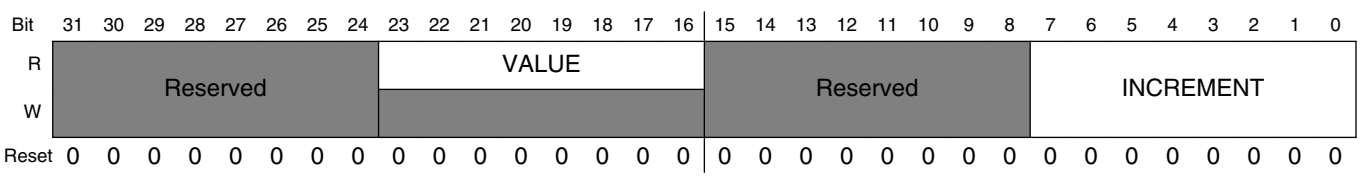
Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 1.

4.3.21 DCP Channel 1 Semaphore Register (DCP_CH1SEMA)

The DCP Channel 1 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjuction with the "decrement_semaphore" bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

Address: 0h base + 150h offset = 150h



DCP_CH1SEMA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the DCP_CHnSEMA_CLR register.

4.3.22 DCP Channel 1 Status Register (DCP_CH1STAT)

The DCP Channel 1 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

CH1STAT: 0x160

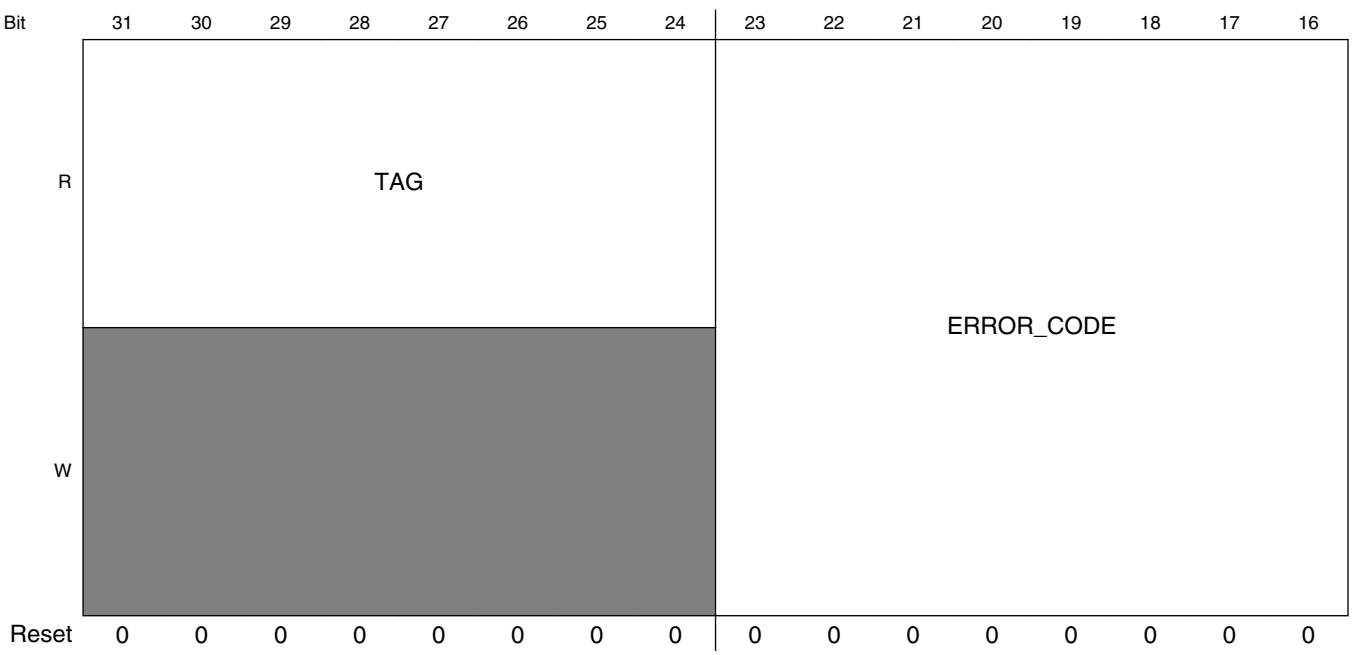
CH1STAT_SET: 0x164

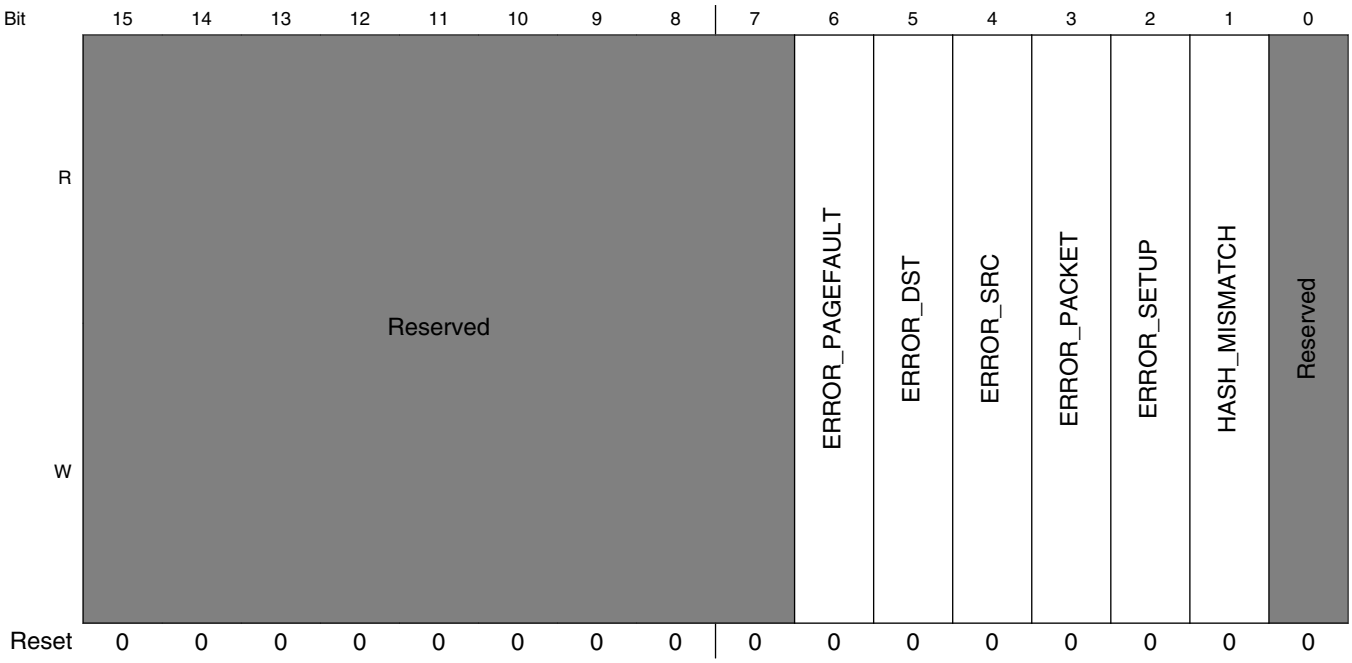
CH1STAT_CLR: 0x168

CH1STAT_TOG: 0x16C

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: 0h base + 160h offset = 160h





DCP_CH1STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 -	This field is reserved. Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a bus error occurs when reading the packet or payload or when writing status back to the packet paylaod. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.

Table continues on the next page...

DCP_CH1STAT field descriptions (continued)

Field	Description
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This field is reserved. This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

4.3.23 DCP Channel 1 Options Register (DCP_CH1OPTS)

The DCP Channel 1 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

DCP_CH1OPTS: 0x170

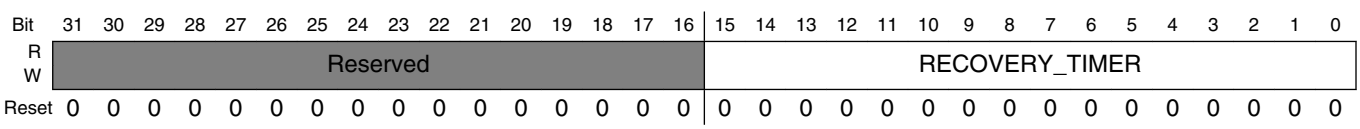
DCP_CH1OPTS_SET: 0x174

CH1OPTS_CLR: 0x178

CH1OPTS_TOG: 0x17C

The options register can be used to control optional features of the channels.

Address: 0h base + 170h offset = 170h



DCP_CH1OPTS field descriptions

Field	Description
31–16 RSVD	This field is reserved. Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

4.3.24 DCP Channel 2 Command Pointer Address Register (DCP_CH2CMDPTR)

The DCP channel 2 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 2 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
            DCP_CHn_CMDPTR_WR(2, v);    // Write channel 2 command pointer
            pCurptr = (DCP_chn_cmdptr_t *) DCP_CHn_CMDPTR_RD(2);  // Read current command
pointer
```

Address: 0h base + 180h offset = 180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH2CMDPTR field descriptions

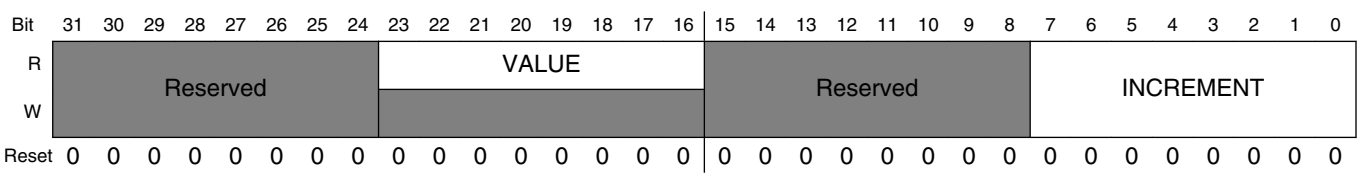
Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 2.

4.3.25 DCP Channel 2 Semaphore Register (DCP_CH2SEMA)

The DCP Channel 2 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjuction with the "decrement_semaphore" bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

Address: 0h base + 190h offset = 190h



DCP_CH2SEMA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the DCP_CHnSEMA_CLR register.

4.3.26 DCP Channel 2 Status Register (DCP_CH2STAT)

The DCP Channel 2 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

CH2STAT: 0x1A0

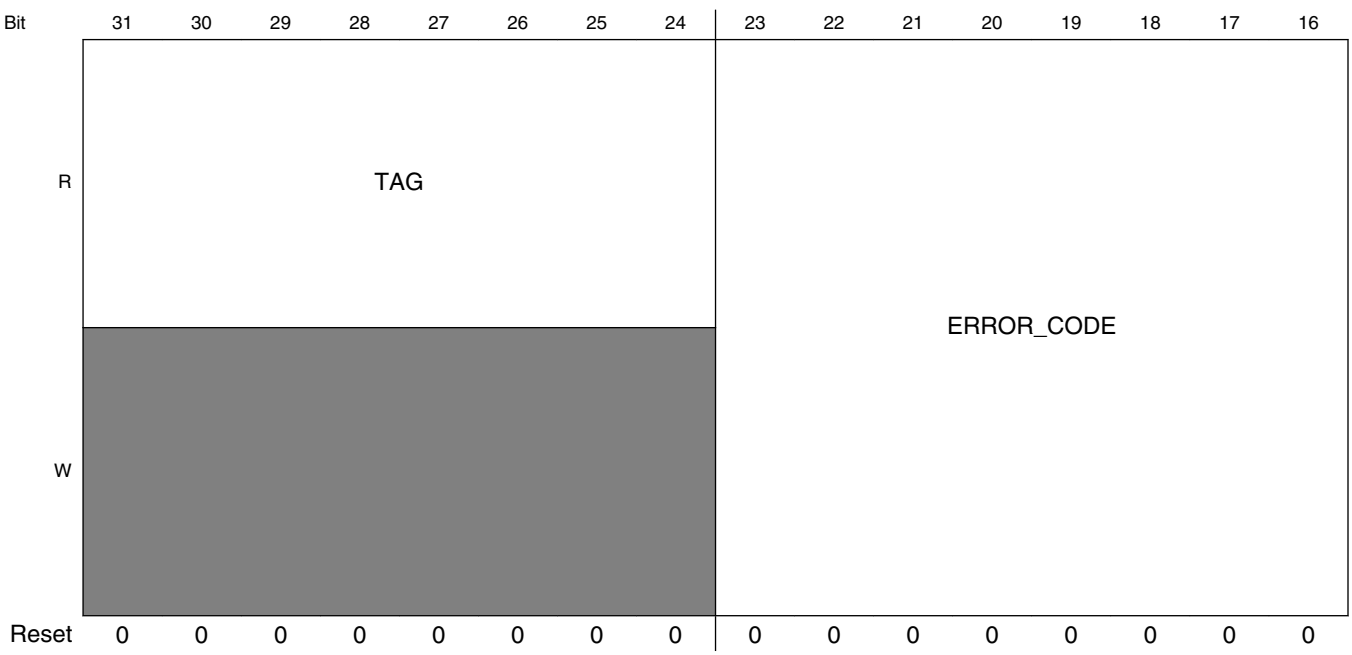
CH2STAT_SET: 0x1A4

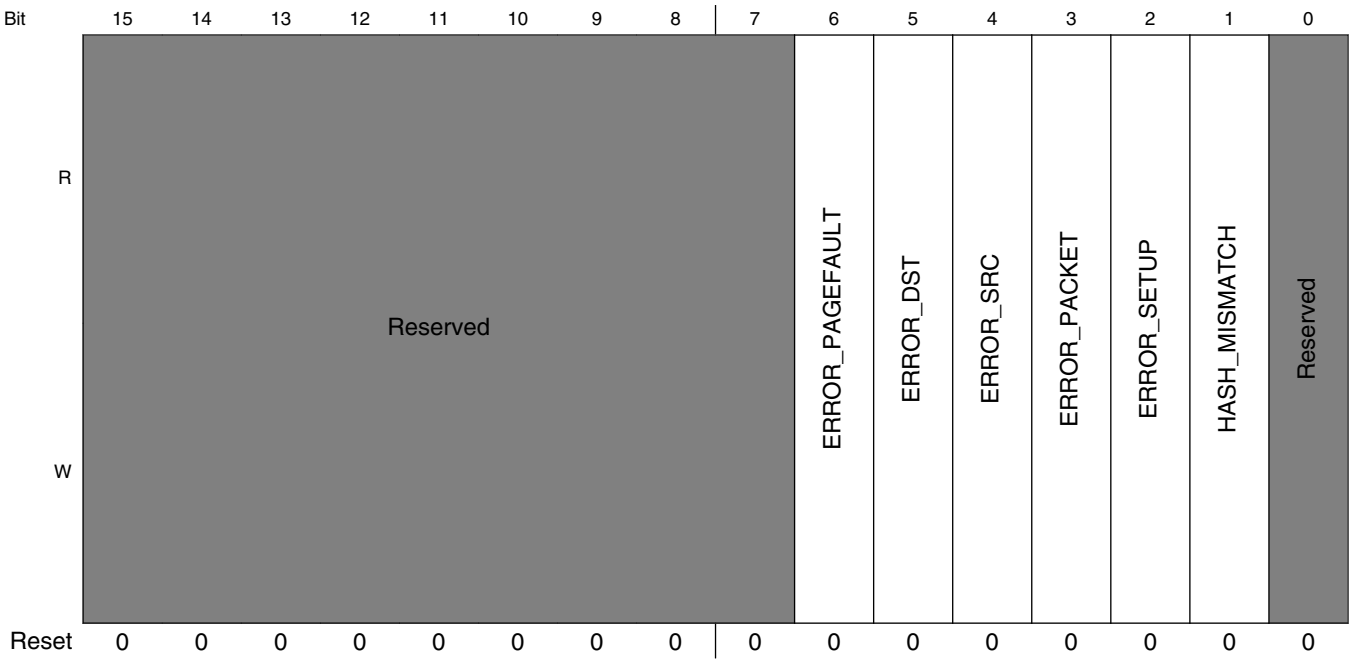
CH2STAT_CLR: 0x1A8

CH2STAT_TOG: 0x1AC

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: 0h base + 1A0h offset = 1A0h





DCP_CH2STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 -	This field is reserved. Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a bus error occurred when reading the packet or payload or when writing status back to the packet paylaod. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.

Table continues on the next page...

DCP_CH2STAT field descriptions (continued)

Field	Description
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This field is reserved. This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

4.3.27 DCP Channel 2 Options Register (DCP_CH2OPTS)

The DCP Channel 2 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

CH2OPTS: 0x1B0

CH2OPTS_SET: 0x1B4

CH2OPTS_CLR: 0x1B8

CH2OPTS_TOG: 0x1BC

The options register can be used to control optional features of the channels.

Address: 0h base + 1B0h offset = 1B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																RECOVERY_TIMER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH2OPTS field descriptions

Field	Description
31–16 RSVD	This field is reserved. Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

4.3.28 DCP Channel 3 Command Pointer Address Register (DCP_CH3CMDPTR)

The DCP channel 3 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the "next_ptr" field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 3 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
            DCP_CHn_CMDPTR_WR(3, v);    // Write channel 3 command pointer
            pCurptr = (DCP_chn_cmdptr_t *) DCP_CHn_CMDPTR_RD(3);  // Read current command
pointer
```

Address: 0h base + 1C0h offset = 1C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH3CMDPTR field descriptions

Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 3.

4.3.29 DCP Channel 3 Semaphore Register (DCP_CH3SEMA)

The DCP Channel 3 semaphore register is used to synchronize the ARM platform instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a "decrement semaphore" bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjuction with the "decrement_semaphore" bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

Address: 0h base + 1D0h offset = 1D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								VALUE								Reserved								INCREMENT							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH3SEMA field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 -	This field is reserved. Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the DCP_CHnSEMA_CLR register.

4.3.30 DCP Channel 3 Status Register (DCP_CH3STAT)

The DCP Channel 3 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

DCP_CH3STAT: 0x1E0

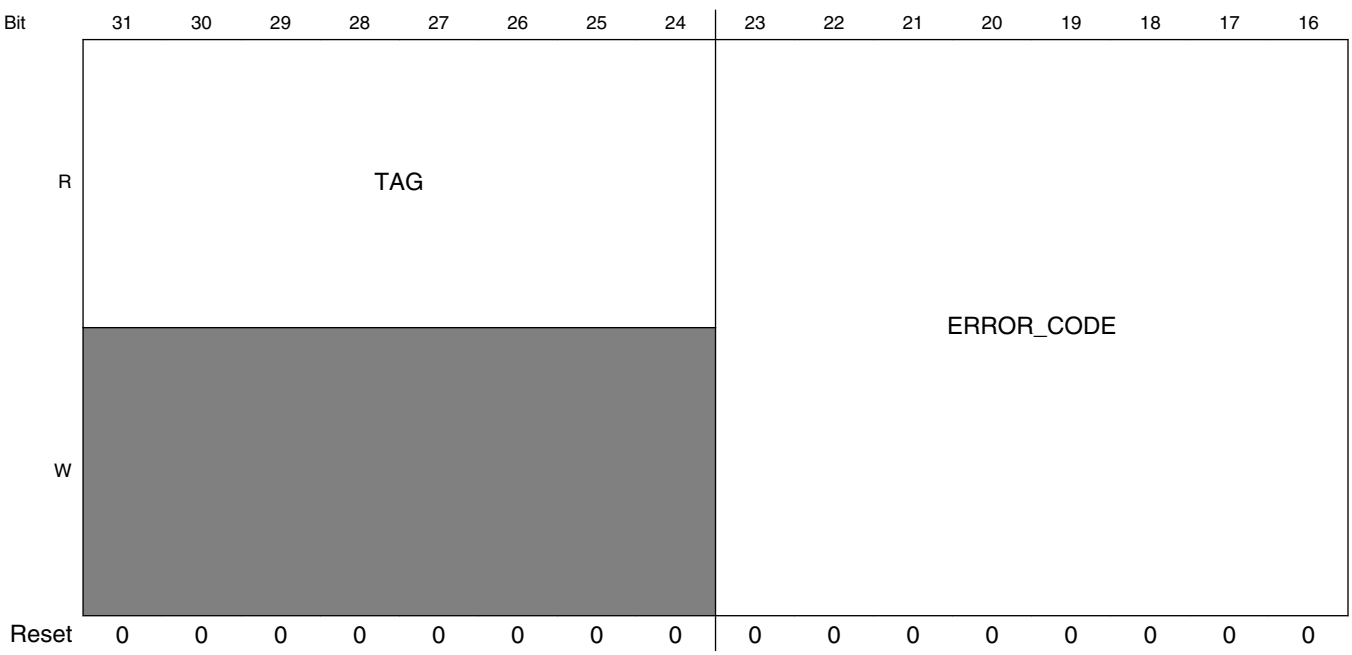
CH3STAT_SET: 0x1E4

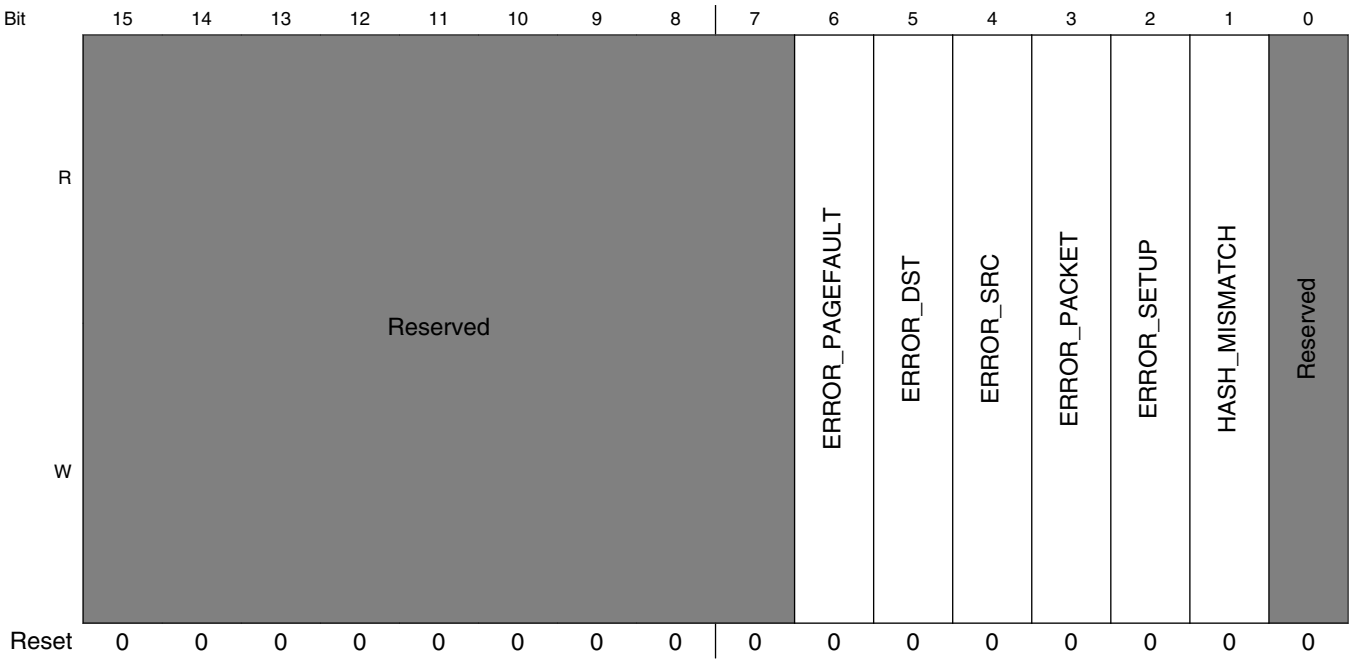
CH3STAT_CLR: 0x1E8

CH3STAT_TOG: 0x1EC

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: 0h base + 1E0h offset = 1E0h





DCP_CH3STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 -	This field is reserved. Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a bus error occurred when reading the packet or payload or when writing status back to the packet paylaod. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.

Table continues on the next page...

DCP_CH3STAT field descriptions (continued)

Field	Description
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This field is reserved. This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

4.3.31 DCP Channel 3 Options Register (DCP_CH3OPTS)

The DCP Channel 3 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

DCP_CH3OPTS: 0x1F0

CH3OPTS_SET: 0x1F4

CH3OPTS_CLR: 0x1F8

CH3OPTS_TOG: 0x1FC

The options register can be used to control optional features of the channels.

Address: 0h base + 1F0h offset = 1F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																RECOVERY_TIMER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_CH3OPTS field descriptions

Field	Description
31–16 RSVD	This field is reserved. Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

4.3.32 DCP Debug Select Register (DCP_DBGSELECT)

This register selects a debug register to view.

This register selects debug information to return in the debug data register.

Address: 0h base + 400h offset = 400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																								INDEX							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_DBGSELECT field descriptions

Field	Description
31–8 RSVD	This field is reserved. Reserved, always set to zero.
7–0 INDEX	Selects a value to read via the debug data register. 0x01 **CONTROL** — 0x10 **OTPKEY0** — 0x11 **OTPKEY1** — 0x12 **OTPKEY2** — 0x13 **OTPKEY3** —

4.3.33 DCP Debug Data Register (DCP_DBGDATA)

Reading this register returns the debug data value from the selected index.

This register returns the debug data from the selected debug index source.

Address: 0h base + 410h offset = 410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_DBGDATA field descriptions

Field	Description
31–0 DATA	Debug Data

4.3.34 DCP Page Table Register (DCP_PAGETABLE)

The DCP Page Table register controls the virtual memory functionality of the DCP. It provides a base address for the page table as well as an enable/disable bit and the ability to flush the cached page table entries.

This register returns the debug data from the selected debug index source.

Address: 0h base + 420h offset = 420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BASE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BASE														FLUSH	ENABLE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_PAGETABLE field descriptions

Field	Description
31–2 BASE	Page Table Base Address. The page table must be word aligned and the pointer should reference a page table in the standard ARM format.
1 FLUSH	Page Table Flush control. To flush the TLB, write this bit to a 1 then back to a 0.
0 ENABLE	Page Table Enable control. Virtual addressing will only be used when this bit is set to a 1. Disabling the page table will not flush any cached entries, so software should write the FLUSH high and enable LOW when updating page tables.

4.3.35 DCP Version Register (DCP_VERSION)

Read-only register indicating implemented version of the DCP.

This register returns the debug data from the selected debug index source.

Address: 0h base + 430h offset = 430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-onlyl value reflecting the MAJOR version of the design implementation.
23–16 MINOR	Fixed read-onlyl value reflecting the MINOR version of the design implementation.
15–0 STEP	Fixed read-onlyl value reflecting the stepping of version of the design implementation.

DCP_VERSION field descriptions (continued)

Field	Description

Chapter 5
Random Number Generator (RNGB)

5.1 Introduction

The purpose of the RNGB is to generate cryptographically strong random data.

It uses a true random number generator (TRNG) and a pseudo-random number generator (PRNG) to achieve true randomness and cryptographic strength. The RNGB generates random numbers for secret keys, per message secrets, random challenges, and other similar quantities used in cryptographic algorithms.

This chapter describes the random number generator (RNGB), including a programming model, functional description, and application information.

5.1.1 Block Diagram

The below figure shows the RNGB's three main blocks: PRNG, TRNG, and XSEED generator.

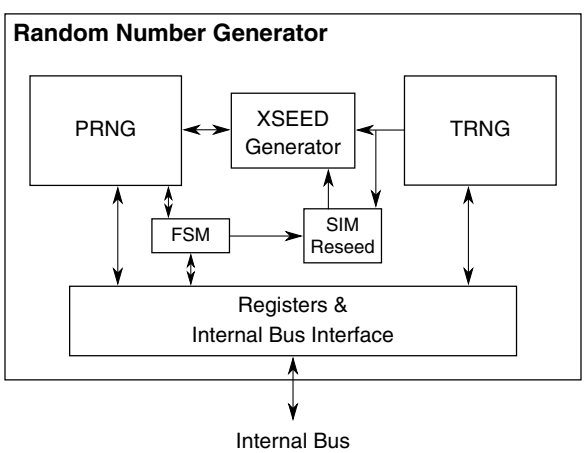


Figure 5-1. RNG Block Diagram

5.1.2 Features

The RNG includes these distinctive features:

- National Institute of Standards and Technology (NIST)-approved pseudo-random number generator
  - http://csrc.nist.gov
- Supports the key generation algorithm defined in the Digital Signature Standard
  - http://www.itl.nist.gov/fipspubs/fip186.htm
- Integrated entropy sources capable of providing the PRNG with entropy for its seed.

5.2 Modes of Operation

Operational modes of the RNGB can be found here.

5.2.1 Self Test Mode

In this mode the RNGB performs a self test of the statistical counters and the PRNG algorithm to verify that the hardware is functioning properly. The self test takes ~29,000 cycles to complete. When self test completes an interrupt may be generated, if there are no outstanding commands in the command register. This mode is entered by setting the RNG_CMD[ST] bit. When self test mode completes, the RNGB remains idle until seed mode is requested or the RNGB transitions to seed mode if automatic seeding is enabled.

5.2.2 Seed Generation Mode

During seed generation, the RNGB adds entropy generated in the TRNG to the 256-bit XKEY register. The PRNG algorithm executes 20,000 times sampling the entropy from the TRNG to create an initial seed for random number generation. At the same time, the TRNG runs simple statistical tests on its output.

When seed generation is complete, the TRNG reports the pass/fail result of the tests through RNG_ESR. If the new seed passes the statistical tests, RNG_SR[SDN] is set, signalling that the RNG is ready to compute secure pseudo-random data. The RNG then transitions to random number generation mode.

5.2.3 Random Number Generation Mode

When seed generation mode completes and the output FIFO is empty, the RNG enters this mode automatically. Random number generation mode quickly creates computationally random data that is derived by the initial seed produced in seed generation mode.

During random number generation, a new 160-bit random number is generated whenever the five word output FIFO is empty. When the output FIFO contains data, the RNGB automatically enters sleep mode, waiting for the data to be read. When the data is read, the RNGB generates a new 160-bit word and goes back to sleep.

After generating 2^{20} words of random data, the RNGB lets the user know that it requires reseeding through RNG_SR and continues to generate random data until it is directed to reseed. However, if auto-seeding is selected, the RNGB automatically completes seeding whenever it is needed.

5.3 Memory Map/Register Definition

The following table shows the address map for the RNGB module. Detailed register descriptions are found in the following sections.

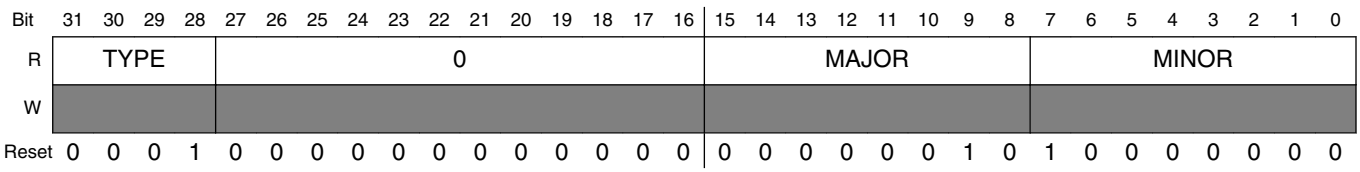
RNG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	RNGB Version ID Register (RNG_VER)	32	R	1000_0280h	5.3.1/126
4	RNGB Command Register (RNG_CMD)	32	R/W	0000_0000h	5.3.2/126
8	RNGB Control Register (RNG_CR)	32	R/W	0000_0000h	5.3.3/128
C	RNGB Status Register (RNG_SR)	32	R	0000_500Dh	5.3.4/130
10	RNGB Error Status Register (RNG_ESR)	32	R	0000_0000h	5.3.5/132
14	RNGB Output FIFO (RNG_OUT)	32	R	0000_0000h	5.3.6/134

5.3.1 RNGB Version ID Register (RNG_VER)

The read-only RNG_VER register contains the current version of the RNGB. It consists of the RNG type and major and minor revision numbers.

Address: 0h base + 0h offset = 0h



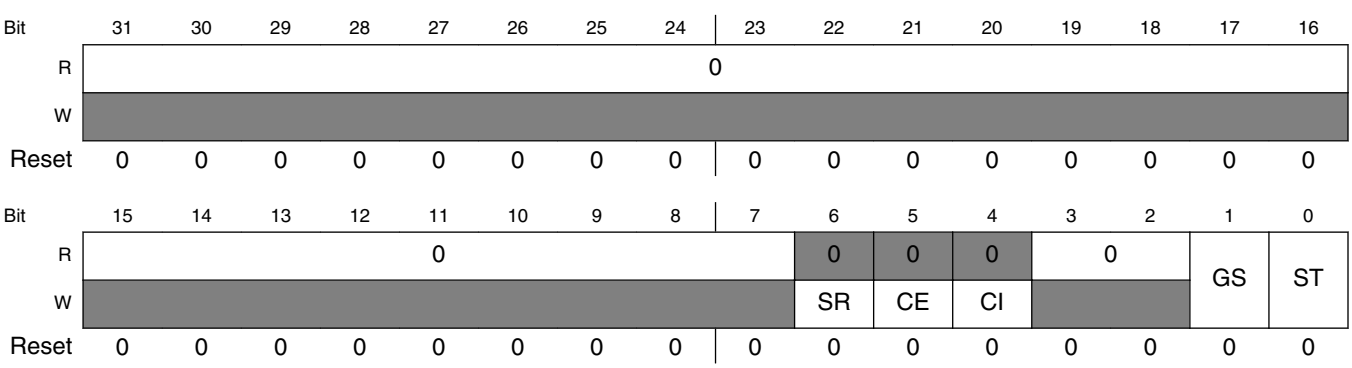
RNG_VER field descriptions

Field	Description
31–28 TYPE	Random number generator type 0000 RNGA 0001 RNGB (This is the type used in this module) 0010 RNGC Else Reserved
27–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15–8 MAJOR	Major version number. This field is always set to 0x02.
7–0 MINOR	Minor version number. Subjiect to change.

5.3.2 RNGB Command Register (RNG_CMD)

RNG_CMD controls the RNG's operating modes and interrupt status.

Address: 0h base + 4h offset = 4h



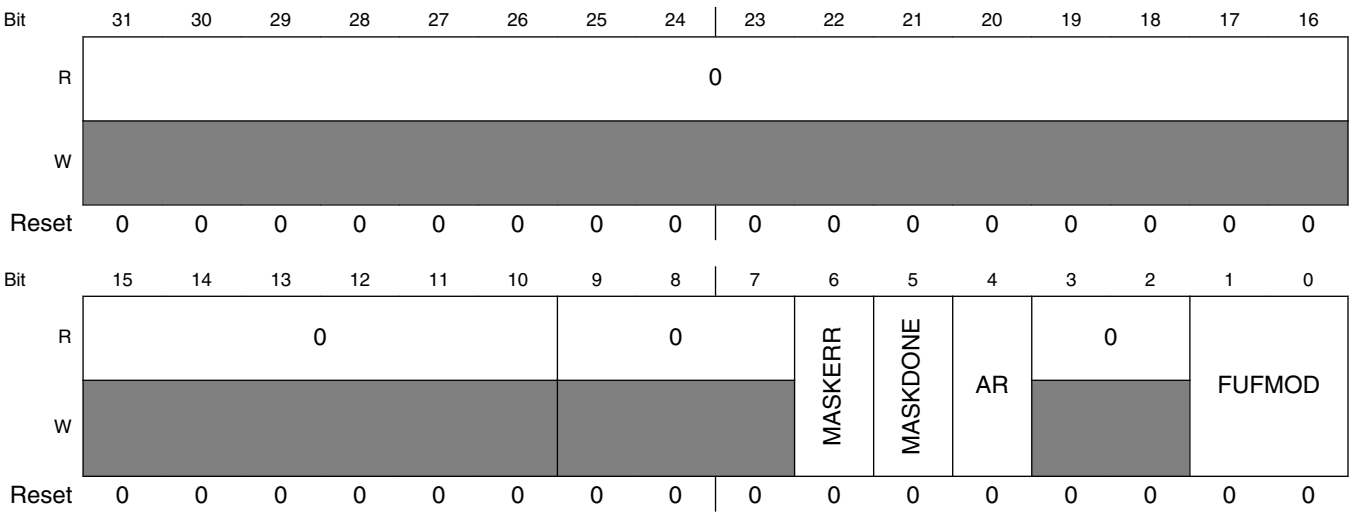
RNG_CMD field descriptions

Field	Description
31–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SR	Software reset. Performs a software reset of the RNGB. This bit is self-clearing. 0 Do not perform a software reset. 1 Software reset.
5 CE	Clear error. Clears the errors in the RNG_ESR register and the RNGB interrupt.This bit is self-clearing. 0 Do not clear errors and interrupt. 1 Clear errors and interrupt.
4 CI	Clear interrupt. Clears the RNGB interrupt if an error is not present. This bit is self-clearing. 0 Do not clear interrupt. 1 Clear interrupt.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 GS	Generate seed. Initiates the seed generation process. Seed generation starts • When RNG_SR[BUSY] is cleared • If set simultaneously with ST, after self-test When the seed generation process completes, this bit automatically clears and an interrupt may be generated if all requested operations are complete. 0 Not in seed generation mode. 1 Generate seed mode.
0 ST	Self test. Initiates a self test of the RNGB's internal logic. The self-test starts • When RNG_SR[BUSY] is cleared, or • If set simultaneously with GS, self test takes precedence and is completed first. When self test completes, this bit automatically clears and an interrupt may be generated if all requested operations are complete. 0 Not in self test mode. 1 Self test mode.

5.3.3 RNGB Control Register (RNG_CR)

Through use of this register, the RNGB can be programmed to provide slightly different functionality based on its desired use.

Address: 0h base + 8h offset = 8h



RNG_CR field descriptions

Field	Description
31–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 MASKERR	Mask error interrupt. Masks interrupts generated by errors in the RNGB. These errors can still be viewed in RNG_ESR. **NOTE:** Since masked errors do not interrupt the operation of the RNGB and thus hide potentially fatal errors or conditions that could result in corrupted results, it is strongly recommended that errors only be masked while debugging. All errors are considered fatal, requiring the RNGB to be reset. Until the a reset occurs, the RNGB does not service any random data. 0 No mask applied. 1 Mask applied to the error interrupt.
5 MASKDONE	Mask done interrupt. Masks interrupts generated upon completion of seed and self test modes. The status of these jobs can be viewed by: • Reading RNG_SR and viewing the seed done and self test done bits (RNG_SR[SDN, STDN]) • Viewing RNG_CMD for generate seed or self test bits (RNG_CMD[GS,ST]) being set, indicating that the operation is still taking place.

Table continues on the next page...

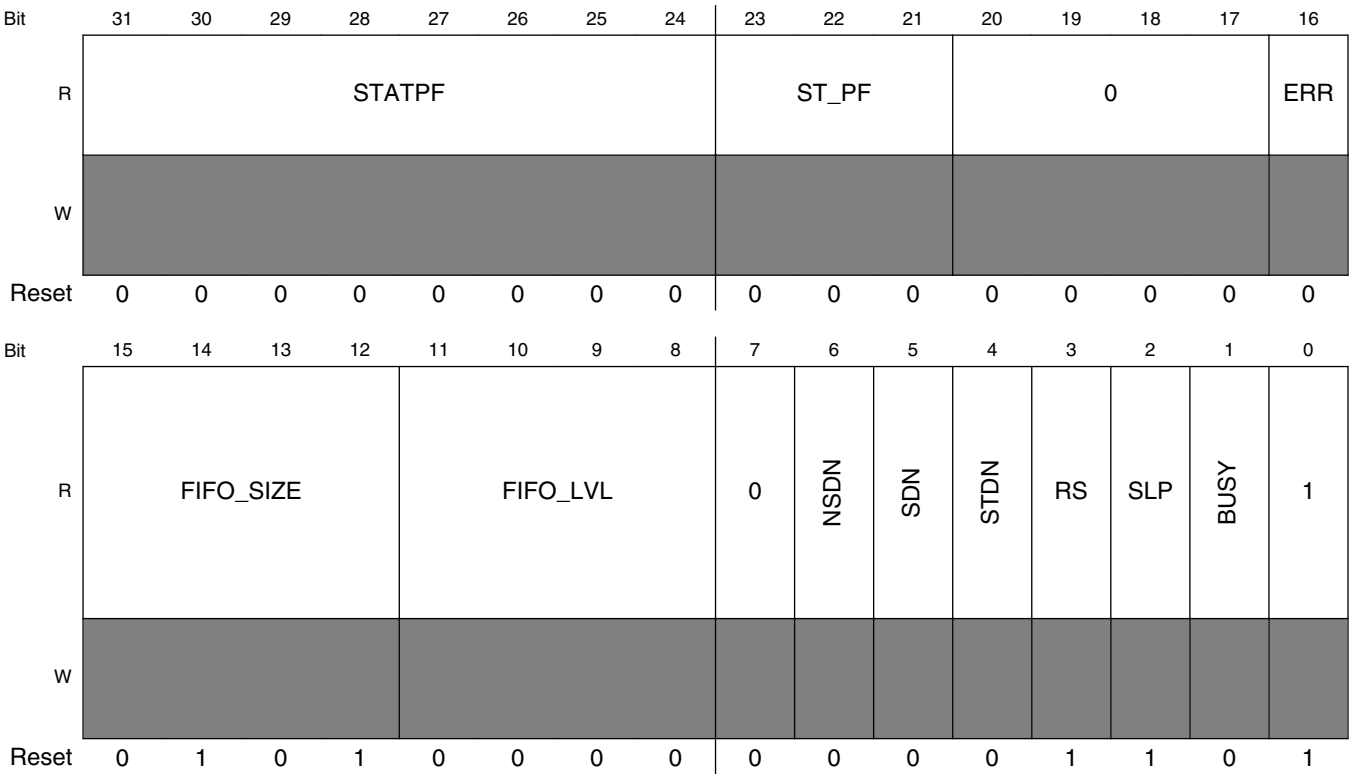
RNG_CR field descriptions (continued)

Field	Description
	0 No mask applied. 1 Mask applied.
4 AR	Auto-reseed. Setting this bit allows the RNGB to automatically generate a new seed whenever one is needed. This allows software to never use the RNG_CMD[GS], although it is still possible. A new seed is needed whenever the RNG_SR[RS] is set. 0 Do not enable automatic reseeding. 1 Enable automatic reseeding.
3–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–0 FUFMOD	FIFO underflow response mode. Controls the RNGB's response to a FIFO underflow condition. 00 Return all zeros and set RNG_ESR[FUFE] 01 Return all zeros and set RNG_ESR[FUFE] 10 Generate bus transfer error 11 Generate interrupt and return all zeros (Overrides RNG_CR[MASKERR])

5.3.4 RNGB Status Register (RNG_SR)

The RNGBSR is a read-only register which reflects the internal status of the RNGB.

Address: 0h base + Ch offset = Ch



RNG_SR field descriptions

Field	Description
31–24 STATPF	Statistics test pass fail. Indicates pass or fail status of the various statistics tests on the last seed generated. • Bit 31 - Long run test (>34) • Bit 30 - Length 6+ run test • Bit 29 - Length 5 run test • Bit 28 - Length 4 run test • Bit 27 - Length 3 run test • Bit 26 - Length 2 run test • Bit 25 - Length 1 run test • Bit 24 - Monobit test 0 Pass. 1 Fail.
23–21 ST_PF	Self Test Pass Fail. Indicates Pass or Fail status of the TRNG, PRNG, and RESEED self tests, • Bit 23 - TRNG self test pass/fail

Table continues on the next page...

RNG_SR field descriptions (continued)

Field	Description
	• Bit 22 - PRNG self test pass/fail • Bit 21 - RESEED self test pass/fail 0 Pass. 1 Fail.
20–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 ERR	Error. Indicates an error was detected in the RNGB. Read the RNG_ESR register for details. 0 No error. 1 Error detected.
15–12 FIFO_SIZE	FIFO size. Size of the FIFO, and maximum possible FIFO level. The bits should be interpreted as an integer. This value is set to five on the default version of RNGB.
11–8 FIFO_LVL	FIFO level. Indicates the number of random words currently in the output FIFO. The bits should be interpreted as an integer.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 NSDN	New seed done. Indicates that a new seed is ready for use during the next seed generation process.
5 SDN	Seed done. Indicates the RNG has generated the first seed. 0 Seed generation process not complete. 1 Completed seed generation since the last reset.
4 STDN	Self test done. Indicates the self test is complete. This bit is cleared by hardware reset or a new self test is initiated by setting RNG_CMD[ST]. 0 Self test not complete. 1 Completed a self test since the last reset.
3 RS	Reseed needed. Indicates the RNGB needs to be reseeded. This is done by setting RNG_CMD[GS], or automatically if RNG_CR[AR] is set. 0 RNGB does not need to be reseeded. 1 RNGB needs to be reseeded.
2 SLP	Sleep. Indicates if the RNGB is in sleep mode. When set, the RNGB is in sleep mode and all internal clocks are disabled. While in this mode, access to the FIFO is allowed. Once the FIFO is empty, the RNGB fills the FIFO and then enters sleep mode again.

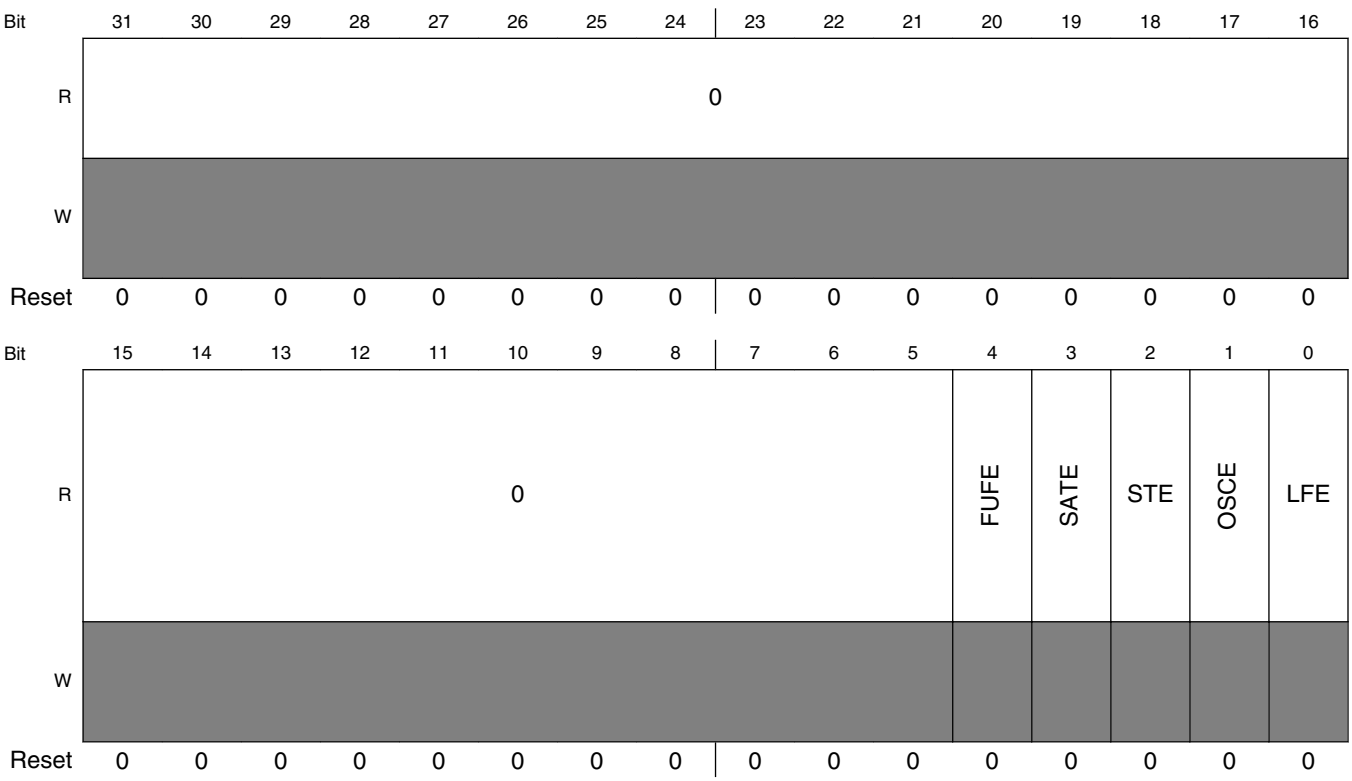
Table continues on the next page...

RNG_SR field descriptions (continued)

Field	Description
	0 RNGB is not in sleep mode. 1 RNGB is in sleep mode.
1 BUSY	Busy. Reflects the current state of RNGB. If RNGB is currently seeding, generating the next seed, creating a new random number, or performing a self test, this bit is set. 0 Not busy. 1 Busy.
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

5.3.5 RNGB Error Status Register (RNG_ESR)

Address: 0h base + 10h offset = 10h



RNG_ESR field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 FUFE	FIFO underflow error

Table continues on the next page...

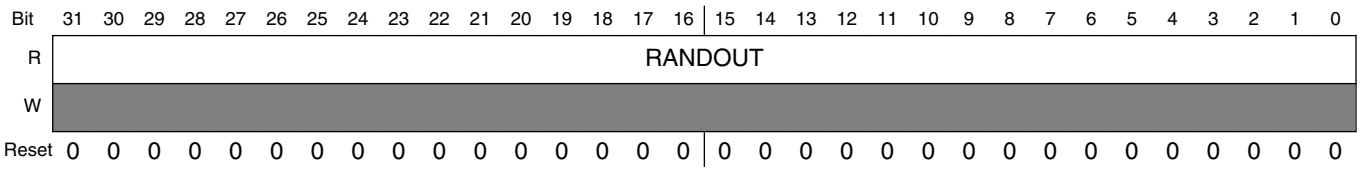
RNG_ESR field descriptions (continued)

Field	Description
	Indicates the RNGB has experienced a FIFO underflow condition resulting in the last random data read being unreliable. This bit can be masked by RNG_CR[FUFMOD] and is cleared by hardware or software reset or by writing one to RNG_CMD[CE]. 0 FIFO underflow has not occurred. 1 FIFO underflow has occurred
3 SATE	Statistical test error. Indicates if RNGB has failed the statistical tests for the last generated seed. This bit is sticky and is cleared by a hardware or software reset or by writing one to RNG_CMD[CE]. 0 RNGB has not failed the statistical tests. 1 RNGB has failed the statistical tests during initialization.
2 STE	Self test error. Indicates the RNGB has failed the most recent self test. This bit is sticky and can only be reset by a hardware reset or by writing one to RNG_CMD[CE]. 0 RNGB has not failed self test. 1 RNGB has failed self test.
1 OSCE	Oscillator error. Indicates the oscillator in the RNG may be broken. This bit is sticky and can only be cleared by a software or hardware reset. 0 RNG oscillator is working properly. 1 Problem detected with the RNG oscillator.
0 LFE	Linear feedback shift register (LFSR) error. When this bit is set, the interrupt generated was caused by a failure of one of the LFSRs in one of the RNGB's three entropy sources. This bit is sticky and can only be cleared by a software or hardware reset. 0 LFSRs are working properly. 1 LFSR failure has occurred.

5.3.6 RNGB Output FIFO (RNG_OUT)

The RNGBOUT provides temporary storage for random data generated by the RNGB. This allows the user to read multiple random longwords back-to-back. A read of this address when the FIFO is not empty, returns 32 bits of random data. If the FIFO is read when empty, a FIFO underrun response is returned according to RNG_CR[FUFMOD]. For optimal system performance, poll RNG_SR[FIFO_LVL] to ensure random values are present before reading the FIFO.

Address: 0h base + 14h offset = 14h



RNG_OUT field descriptions

Field	Description
31–0 RANDOUT	Random Output

5.4 Functional Description

The RNGB performs two functional operations, as described in Modes of Operation seed generation and random number generation.

These operations are performed with cooperation from the major functional blocks in the RNGB described below.

5.4.1 Pseudorandom Number Generator (PRNG)

The PRNG implements the NIST-approved PRNG described in the *Digital Signature Standard*. The 160-bit output of the SHA-1 block is the next five words of random data. The PRNG is designed to generate 2^{20} words of random data before requiring reseeding, using the TRNG only during the seeding/initialization process. The initial seed takes approximately two million clock cycles. After this the RNGB can generate five 32-bit words every 112 clock cycles. Reseeding takes place transparently through use of the

simultaneous reseed LFSRs. The entropy stored in this 128-bit LFSR and 128-bit shift register is added directly into the XKEY structure via the RNGB XSEED generator whenever reseeding is required.

5.4.2 True Random Number Generator (TRNG)

The TRNG is comprised of two entropy sources each providing a single bit of output. Concatenated together, these two output bits are expected to provide one bit of entropy every 100 clock cycles. In addition to generating entropy, the TRNG also performs several statistical tests on its output. The pass/fail status of these tests are reflected in RNG_ESR.

5.4.3 Resets

There are two ways to reset the RNGB: power-on/hardware reset and software reset. The software reset is functionally equivalent to the power-on/hardware reset. The power-on/hardware reset is asynchronous. Software reset is performed by setting the RNG_CMD[SR] bit. These are summarized in the table below.

Table 5-8. Reset Summary

Reset	Source	Characteristics	Internally resets:	Affect on External Signal:
Hardware	ipg_hard_async_reset_b	Active-low, asynchronous, minimum 1-cycle	All interface registers and puts RNGB into the IDLE state	—
Software	RNG_CMD[SR]	Active-high	All interface registers and puts RNGB into the IDLE state	—

5.4.3.1 Power-on/Hardware Reset

Asserting the ipg_hard_async_reset_b signal sets all interface registers to their default state and puts the state machine into the IDLE mode.

5.4.3.2 Software Reset

The software reset is functionally equivalent to the hardware reset, but allows the RNGB to be fully reset by writing to the SW_RST bit (bit-6) in the RNGB Command Register. This bit is self-resetting. A software reset may be performed at any time.

5.4.4 RNG Interrupts

There is a single RNG interrupt generated to the processor's interrupt controller. The source of the interrupt is determined by reading the RNG status register. If an error is the cause of the interrupt, further information is available by reading the RNG error status register. The interrupts can be masked by the RNG_CR[MASKDONE or MASKERR] bits

It is strongly recommended that the error interrupt is only masked while debugging, since masking the error interrupt could hide potentially fatal errors or conditions that could result in corrupted results. All errors are considered fatal, requiring the RNG to be reset. The RNG does not service any random data until a reset occurs.

The available interrupt sources are described in the following table.

Table 5-9. RNG Interrupt Sources

Sources	Status Bit Field	RNG_CR Mask Bit Field	Description
Seed generation done	RNG_SR[SDN]	MASKDONE	First seed was generated
Self test done	RNG_SR[STDN]	MASKDONE	Self test finished
Error	RNG_SR[ERR]	MASKERR	Error detected. See RNG_ESR for details.
Linear feedback shift register (LFSR)	RNG_ESR[LSFRE]	MASKERR	Fault in one of the TRNG's LFSRs
Oscillator	RNG_ESR[OSCE]	MASKERR	TRNG ring oscillator may be malfunctioning
Self test	RNG_ESR[STE]	MASKERR	Self test failed
Statistical test	RNG_ESR[SATE]	MASKERR	Statistics test for last seed generation failed
FIFO Underflow	RNG_ESR[FUFE]	MASKERR	FIFO read while empty

5.5 Initialization/Application Information

Information found here describes the module initialization.

5.5.1 Manual Seeding

The intended general operation of the RNGB is as follows:

1. Reset/initialize.

2. Write to the RNG_CR to setup the RNGB for the desired functionality.
3. Write to RNG_CMD to run self-test or seed generation.
4. Wait for interrupt to indicate completion of the requested operation(s).
5. Repeat steps 3–4 if seed generation is not complete.
6. Poll RNG_SR for FIFO level.
7. Read available random data from output FIFO.
8. Repeat steps 6 and 7 as needed, until 2^{20} words have been generated.
9. Write to RNG_CMD to run seed mode.
10. Repeat steps 4–9.

5.5.2 Automatic Seeding

The intended general operation of the RNGB with automatic seeding enabled is as follows:

1. Reset/initialize.
2. Write to the RNG_CR to setup the RNGB for automatic seeding and the desired functionality.
3. Wait for interrupt to indicate completion of first seed
4. Poll RNG_SR for FIFO level.
5. Read available random data from output FIFO.
6. Repeat steps 4 and 5 as needed. Automatic seeding occurs when necessary and is transparent to operation.

Chapter 6
Secure Non-Volatile Storage (SNVS)

6.1 SNVS structure

The block is divided into two major submodules based on power supply: the high power part (SNVS_HP) and the low power part (SNVS_LP). They are powered as follows:

- SNVS_LP- dedicated always-powered-on domain
- SNVS_HP - system (chip) power domain

The following figure illustrates the low power and chip power domains.

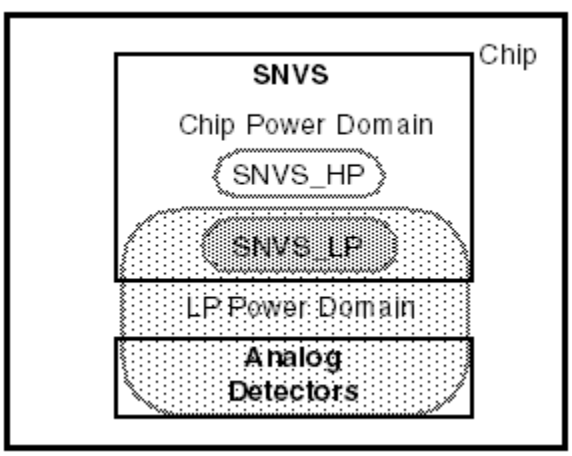


Figure 6-1. SNVS power supply

SNVS _HP implements all features that enable system communication and provisioning of the SNVS _LP module. It also incorporates the system security monitor, which controls the system security state, and the master key control block, which is responsible for checking and selecting the master key value.

SNVS _LP provides hardware that enables secure storage and protection of sensitive data. It detects and responds to the real time security threats even when system is in power-down mode.

6.2 SNVS _HP (high power part)

SNVS _HP is partitioned into the following functional units:

- IP bus interface
- SNVS _LP interface
- System security monitor
- Master key control block
- Zeroizable master key programming mechanism
- Real time counter with alarm
- Control and status registers

SNVS _HP is in the chip's power supply domain and thus receives power along with the rest of the chip. SNVS _HP provides an interface between SNVS _LP and the rest of the system; there is no way to access the SNVS _LP registers except through the SNVS _HP. For access to the SNVS _LP registers, SNVS _HP must be powered up. It uses a register access permission policy to determine whether access to particular registers is permitted.

6.2.1 System security monitor (SSM)

The system security monitor (SSM) monitors system security violations and controls the security state of the system. In essence, the SSM is a state machine. It classifies security violation sources as disabled, fatal, or non-fatal, and each classification results in different state transitions. A security violation source that is classified as a fatal security violation always results in the SSM transition to the soft fail state. A non-fatal security violation results in a transition from check to non-secure or from trusted/secure to soft fail states. The SSM does not react to a disabled violation. See HP security violation policy for a detailed description of the SNVS security violation policy.

6.2.1.1 Transitioning among system security monitor states

The following figure and tables explain the SSM's states and transitions.

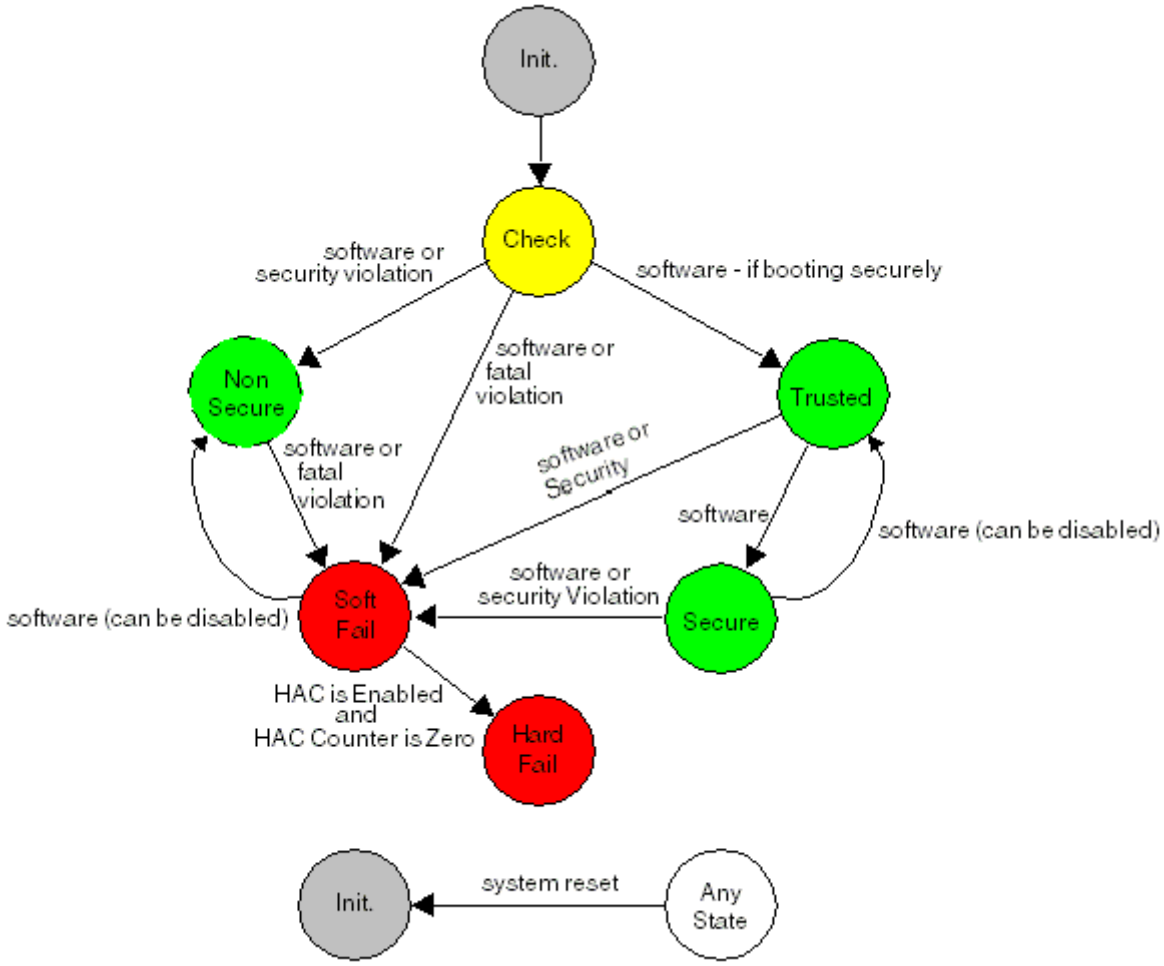


Figure 6-2. SSM state machine

Table 6-1. State definitions

State	What happens	State transition
Init	System enters this state on the system POR. In this state, the SSM ignores all security violations sources; security violations are not recorded and not responded to in the SNVS _HP domain. SNVS registers cannot be programmed in this state.	SSM transitions to the check state .

Table continues on the next page...

Table 6-1. State definitions (continued)

State	What happens	State transition
Check	System performs the check sequence. SNVS registers cannot be programmed in this state, with the exception of several bits in the HP Command Register.	• Software sets the control bit in the HP Command Register to initiate the transition from check to Trusted State. The transition is allowed if the system is booting internally or if System Security Configuration is Fab (000) . Upon transition from check state to trusted state, SNVS _LP is reset if it was previously provisioned in the non-secure state. • If the SSM detects a non-fatal security violation condition while in this state, it immediately transitions to the non-secure state. • If the SSM detects a fatal violation, it transitions to the soft fail state if fatal violation is detected.
Non-Secure	In this state, the secure and trusted mode indication signals are not set. In non-secure state any write accesses to the SNVS _LP registers are denied if SNVS _LP was provisioned in the trusted/secure state.	SSM transitions to the soft fail state if a fatal violation is detected.
Trusted	In this state, the trusted and secure mode indication signals are asserted.	Privileged software triggers the transition from secure state to trusted state or vice versa by writing to the SSM_ST bit in the HP Command Register. The secure state to trusted state transition can be disabled by setting the SSM_ST_DIS bit in the HP Command Register. If the SSM detects a security violation condition, it immediately (without clock) transitions to the soft fail state.
Secure	In this state, the secure mode indication signal is asserted and trusted mode indication signal is de-asserted.	Privileged software triggers the transition from trusted state to secure state or vice versa by writing to the SSM_ST bit in the HP Command Register. The secure state to trusted state transition can be disabled by setting the SSM_ST_DIS bit in the HP Command Register. If the SSM detects a security violation condition, it immediately (without clock) transitions to the soft fail state.

Table continues on the next page...

Table 6-1. State definitions (continued)

State	What happens	State transition
Soft fail	In this state SSM generates a security interrupt and the fail state indication output, which is treated as a security violation by other security modules. SNVS registers cannot be programmed in this state, except for several bits in the HP Command Register.	If high assurance configuration has been enabled by setting the bit in the HP Command Register, the SSM transitions to the hard fail state after the HAC counter counts down to zero. Software can stop this counter to prevent the transition to the hard fail state. Software triggers the transition to the non-secure state only if the HAC counter is deactivated (disabled or stopped) and the transition to the non-secure state is not disabled. Note that in cases of active fatal security violation, the SSM stays in the soft fail state.
Hard Fail	Entering this state triggers hard reset request output, which should be used in the system to perform a hardware reset without the aid of software. SNVS registers cannot be programmed in this state.	The SSM remains in the hard fail state until the system is reset.

Table 6-2. SSM state transition table

	Next State (transition to next state determined by conditions listed in the table cells below)						
Current State	**Init**	**Check**	**NonSecure**	**Soft Fail**	**Hard Fail**	**Trusted**	**Secure**
Init	Sll other conditions	LP Section is Powered On	—	—	—	—	—
Check	System Reset	All other conditions	Non-Fatal Security Violation[1] or write 1 to the SSM_ST bit of the HPCOMR when non-secure boot	Fatal Security Violation [1]	—	Write 1 to the SSM_ST bit of the HPCOMR when secure boot	—
NonSecure	System Reset	—	All other conditions	Fatal Security Violation [1]	—	—	—
Soft Fail	System Reset	—	Write 1 to the SSM_ST bit of the HPCOMR (if not disabled by SSM_SFNS_DIS bit of the HPCOMR and HAC Counter4 is either disabled or stopped)	All other conditions	HAC is Enabled[2] & HAC Counter[3] is Zero	—	—
Hard Fail	System Reset	—	—	—	All other conditions	—	—
Trusted	System Reset	—	—	Non-Fatal or Fatal Security Violation [1]	—	All other conditions	Write 1 to the SSM_ST bit of the HPCOMR

Table continues on the next page...

Table 6-2. SSM state transition table (continued)

	Next State (transition to next state determined by conditions listed in the table cells below)						
Current State	**Init**	**Check**	**NonSecure**	**Soft Fail**	**Hard Fail**	**Trusted**	**Secure**
Secure	System Reset	—	—	Non-Fatal or Fatal Security Violation [1]	—	Write 1 to the SSM_ST bit of HPCOMR (if not disabled by SSM_ST_DIS bit of the HPCOMR)	All other conditions

1. See Table 1 for a list of non-fatal and fatal security violations
2. See HAC_EN field of HPCOMR
3. See HAC Counter

6.2.1.2 HP security violation policy

All HP security violation sources are classified into three categories: disabled, non-fatal security, and fatal security.

- Disabled violation-SSM does not react on this violation.
- Non-fatal security violation-Results in the following transitions:
  - Check state → non-secure state
  - Trusted state → soft fail state
  - Secure state → soft fail state
- Fatal security violation-SSM transitions to the soft fail state from check, non-secure, trusted, or secure states

Regardless of the category all security violation events are recorded in the corresponding status registers.

6.2.1.3 SNVS _HP violation sources

The following table explains all SNVS _HP security violation sources.

Table 6-3. SNVS _HP security violation sources

Security violation source	Default behavior	Configuration options[1]	Comments
Scan exit violation	Fatal	-	Asserted upon scan exit and stays active till system reset
Software fatal violation	Fatal	-	Asserted by software
Software non-fatal violation	Non-Fatal	-	Asserted by software

Table continues on the next page...

Table 6-3. SNVS _HP security violation sources (continued)

Security violation source	Default behavior	Configuration options[1]	Comments
Bad master key violation	Non-Fatal	-	Asserted when: - OTPMK does not pass validity check - ZMK ECC Check failure (if ZMK ECC Check is enabled) - ZMK is zero when it is selected for use
Security violation Inputs 0-4	Non-Fatal	• Non-fatal • Fatal	Asserted on the security violation input ports 0-4
Security violation input 5	Disabled	• Disabled • Non-fatal • Fatal	Asserted on the security violation input port 5
From LP section			
LP power-on-reset	Fatal	-	Asserted upon LP POR detection. This signal is asynchronous to any clock in the module.
LP security violation	Disabled	• Disabled • Non-Fatal • Fatal	Asserted in the LP section. This signal can be asserted asynchronously to any clock in the module.

1. The configuration is set in the HP Security Violation Control Register (HPSVCR). Once set, the configuration can be locked to prevent further changes

6.2.2 Master key control

The master key control block functionality depends on the particular security conditions and configuration. The master key control block selects among the following master key sources:

- One time programmable master key (OTPMK)

  The OTPMK is a random value stored in the non-volatile memory (typically fuses) outside the SNVS module.

- Zeroizable master key (ZMK)

  The ZMK is a random value stored in the ZMK registers of the SNVS _LP section.

- Combined master key (CMK)

  This key is created by a bitwise XOR operation of OTPMK and ZMK values.

The following figure shows the master key selection scheme.

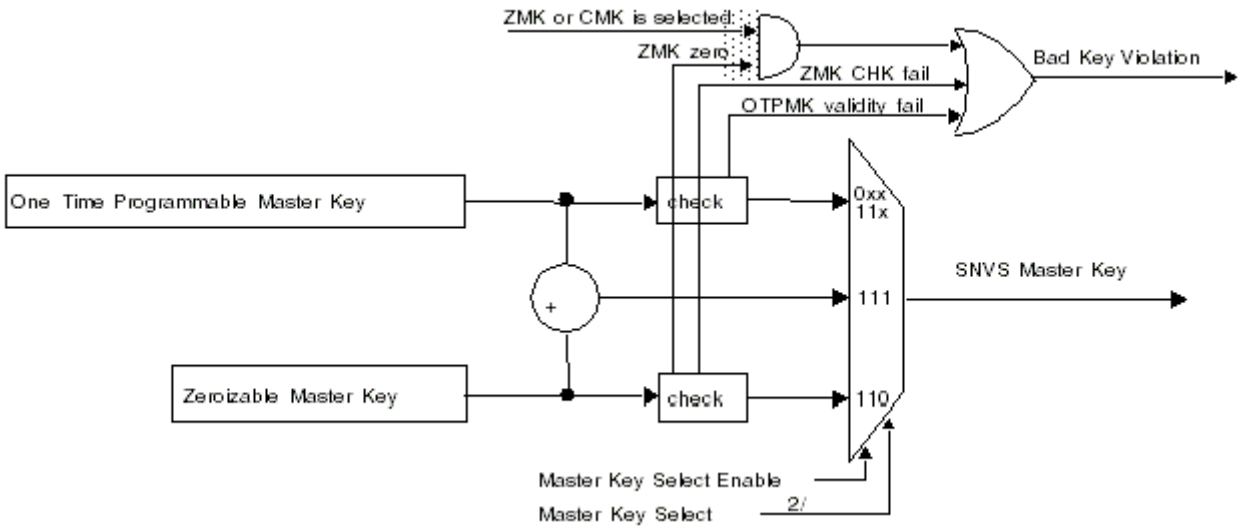


Figure 6-3. Master key control scheme

The master key control block incorporates hardware checking logic for OTPMK and ZMK values to assure that the keys are programmed with their intended valid values and have not been corrupted.

The OTPMK's value is checked for whether it is all zeros. The OTPMK can be optionally checked for whether it has generated a valid nine-bit Hamming codeword (see Error Code for the OTPMK, for more information). If the OTPMK is found to be all zeros or to be an invalid codeword, the validity check fails.

If ZMK is selected for use and marked as valid, it is also checked for whether it different from the default all-zero value. In addition, the ZMK can be optionally checked for whether it has generated a valid nine-bit Hamming codeword.

If the validity check fails, the master key control block reports the bad key violation to the system security monitor, which treats it as a non-fatal violation. The check results can always be found in the HP status registers.

6.2.2.1 Error Code for the OTPMK

The Hamming code used for the 256-bit OTPMK has nine code bits. It can detect all 1, 2, and 3 bit errors in the codeword. It also detects most, but not all, errors of more than 3 bits. Therefore, the OTPMK actually contains 247 random bits.

Numbering the bits in the OTPMK from zero to 255, the code bits are bits 0, 1, 2, 4, 8, 16, 32, 64, and 128. All remaining bits are data bits. Each of these code bits is the XOR of a subset of the bits in the entire code word.

To determine which bits are used to form each code bit, look at the binary representation of the bit position of each code bit (ignoring bit zero for the time being) in the following table.

Table 6-4. Error codes

Code bit	Binary representation
0	00000000
1	00000001
2	00000010
4	00000100
8	00001000
16	00010000
32	00100000
64	01000000
128	10000000

For code bit number 1, there is a single one in its binary representation. This code bit is the XOR of all bit positions that also have a one in this same point in their binary representation (i.e., all odd bit positions). The same is true for the other code bits, excepting code bit 0. For example, code bit 2 is the XOR of bits 3, 6, 7, 10, 11, 14, 15,..., 254, 255, and code bit 128 is the XOR of all bits from 129 through 255 inclusive. After all of the other code bits have been calculated, code bit zero is simply the XOR of all of the other bits, including the code bits.

Another way of looking at this is to ask which code bits does a data bit affect. If we look at the binary representation of the bit position of a data bit, the 1s represent the code bits that this data bit affects. For example, data bit 99 (01100011) is XORed into code bits 1, 2, 32, 64, and bit 0 (the overall parity bit).

6.2.2.2 Generating the code bits

To generate a code word to be programmed into the one time programmable elements for the OTPMK, it is simplest to start with an array of 256 random bits (the array numbered 0 through 255). The following pseudocode then replaces the nine code bits with their proper values. The starting random bits at those locations are discarded.

```
// Generate the Hamming code bits for the 256 bits
// stored in Number. The values at the locations of the code bits
// are ignored and is overwritten with the generated values.
Generate_code_bits(bool Number[256])
  {
  int i, j;

  // Calculate each code bit in turn
  for(i = 1; i <= 128; i = (i << 1)) {
```

```
    // Examine each data bit
    // Only bits greater than i need to be checked as no
    // bit less than i will ever be XOR'ed into i
    // J starts at i so that Number[i] is initialized to 0
    for(j = i; j <= 255; j = j + 1) {
      if ( (i & j) != 0) {
        Number[i] = Number[i] ^ Number[j];
      }
    }
  }
  // Calculate the overall parity
  // J starts at 0 so that Number[0] is initialized to 0
  // Number[0] contains the even parity of all of the bits
  for(j = 0; j <= 255; j = j + 1) {
    Number[0] = Number[0] ^ Number[j];
  }
```

6.2.2.3 Checking the code bits

The logic within the SNVS uses an equivalent of the following pseudocode to check whether the 256-bit OTPMK value is a legal codeword or if any of the bits were not programmed properly.

```
// Check the Hamming code bits for the 256 bits stored in Number.
// If an error is detected, return BAD.
// If no errors are detected, return GOOD.
bool Check_code_bits(bool Number[256])
{
  int i, j, n;
  bool Check[9];
  // Calculate each code bit in turn
  // i steps through Number
  // n steps through Check; n = log(i)
  for(i = 1, n=0; i <= 128; i = (i << 1), n = n + 1) {
    // Examine each data bit
    // Only bits greater than i need to be checked as no
    // bit less than i will ever be XOR'ed into Check[n]
    Check[n]= 0;
    for(j = i; j <= 255; j = j + 1) {
      if ( (i & j) != 0) {
        Check[n] = Check[n] ^ Number[j];
      }
    }
  }
  // Calculate the overall parity
  // Check[8] is the even parity of all of the bits
  Check[8]= 0;
  for(j = 0; j <= 255; j = j + 1) {
    Check[8] = Check[8] ^ Number[j];
  }
  // Any non-zero Check bit means an error has been detected
  for(n = 0; n <=8; n = n + 1) {
    if(Check[n] != 0) {
      return BAD;
    }
  }
  return GOOD;
}
```

6.2.2.4 Error code for the ZMK

The Hamming code used for the 256-bit ZMK has nine code bits. It can detect all 1, 2 and 3 bit errors in the codeword. It also detects most, but not all, errors in more than 3 bits. Unlike OTPMK, where the check code bits are part of the OTPMK value, the ZMK code bits are stored separately from the ZMK value. Therefore, the presence of code bits does not reduce the randomness of the ZMK value.

SNVS logic automatically calculates and stores the ZMK ECC value when software enables the ZMK ECC feature. Software is not allowed to write this value. If read access to the ZMK registers is not allowed, software cannot read the ZMK ECC value.

Once the ZMK ECC checking mechanism is enabled and the ZMK ECC value is captured, the SNVS logic continuously checks the stored ECC value against the current ECC value. If the current ECC value is found to be different than the one stored in the register, the ZMK ECC failure violation is generated and the ZMK syndrome value is captured.

The following figure shows the flow:

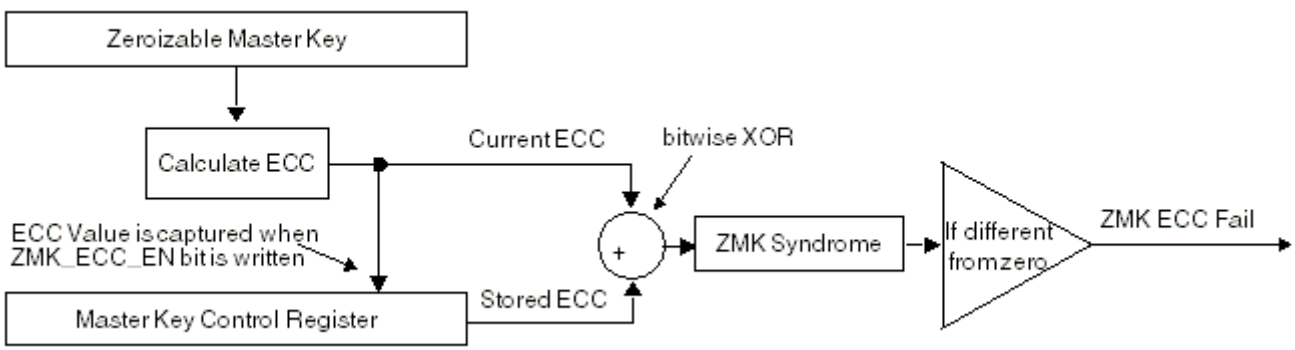


Figure 6-4. ZMK error code programming and checking flow

The steps are as follows:

1. Use either software or the hardware programming mechanism to program the ZMK value.
2. Write the ZMK_ECC_EN bit of the SNVS _LP Master Key Control Register (LPMKCR) to enable the ZMK ECC mechanism.
3. Hardware calculates ZMK_ECC_VALUE's value and sets it automatically.
4. SNVS logic checks whether ZMK_ECC_VALUE is different from the current ZMK ECC value.
5. If it is different, the ZMK_ECC_FAIL and ZMK_SYNDROME fields of the SNVS _HP Security Violation Status Register (HPSVSR) are set and a security violation is generated.

6.2.3 ZMK hardware programming mechanism

The ZMK_HWP bit controls the ZMK programming flow. When ZMK_HWP is set, only hardware can program the ZMK. Otherwise, only software can program the ZMK. In either case, the ZMK cannot be programmed if it is locked for writes.

To initiate hardware programming of the ZMK, software sets the PROG_ZMK bit of the HP Command Register. In hardware programming mode, the ZMK_VAL bit is set automatically after the ZMK's last word is programmed. This bit (ZMK_VAL) indicates to the software whether the programming is still in progress or has already completed.

6.2.4 Non-secure real time counter

SNVS _HP has an autonomous non-secure real time counter. The counter is not active and is reset when the system is powered down. The HP RTC can be used by any application; it has no privileged software access restrictions. The counter can be synchronized with the SNVS_LP SRTC by writing to a specific bit in the SNVS_HP Control Register.

6.2.4.1 Calibrating the time counter

The RTC accuracy may suffer from a drift in the clock, which is used to increment the RTC register. To compensate for this drift, a clock calibration mechanism can adjust the RTC value. It is up to the system processor to decide whether calibration is required or not. If RTC correction is required, enable the mechanism and set the calibration value in the control register. The calibration value is a 5 bit value including the sign bit, which is implemented in 2's complement.

If the calibration mechanism is enabled, the calibration value is added or subtracted from the RTC on a periodic basis, once per 32768 cycles of the RTC clock. The following table shows the available correction range.

Table 6-5. Time counter calibration settings

Calibration value setting	Correction in counts per 32768 cycles of the counter clock
01111	+15
:	:
00010	+2
00001	+1
00000	0

Table continues on the next page...

Table 6-5. Time counter calibration settings (continued)

Calibration value setting	Correction in counts per 32768 cycles of the counter clock
11111	-1
11110	-2
:	:
10001	-15
10000	-16

6.2.4.2 Time counter alarm

The SNVS _HP non-secure RTC has its own time alarm register. Any application can update this register. The SNVS _HP time alarm can generate interrupts to alert the host processor and can wake up the host processor from one of its low-power modes (wait, doze, stop). Note that this alarm cannot wake up the entire system if it is powered off because this alarm would also be powered off.

6.2.4.3 Periodic interrupt

The SNVS _HP non-secure RTC incorporates a periodic interrupt. The periodic interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the RTC. The periodic interrupt source is chosen from 16 bits of the HP RTC according to the PI_FREQ field setting in the HP Control Register. This bit selection also defines the frequency of the periodic interrupt.

The following figure shows the SNVS _HP RTC and its interrupts.

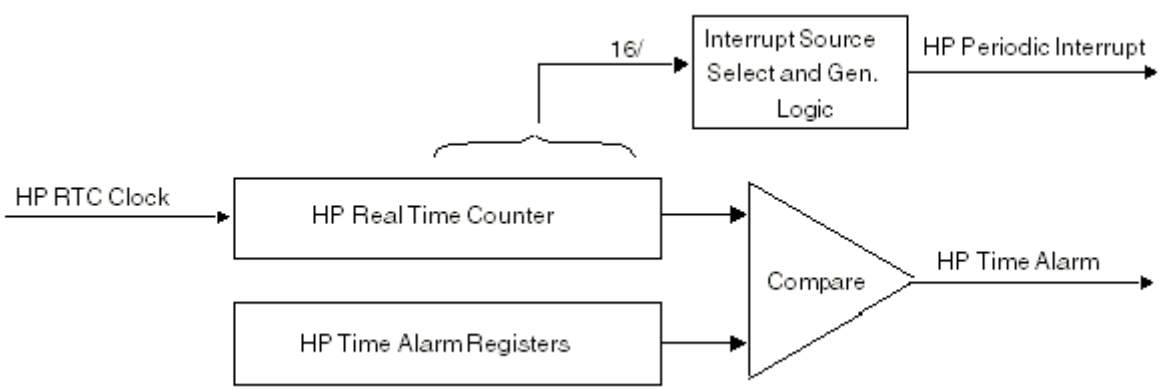


Figure 6-5. SNVS _HP RTC, alarm, and interrupts

6.3 Low power part (SNVS _LP)

SNVS _LP has the following functional units:

- Zeroizable master key
- Secure non-rollover real time counter with alarm
- Non-rollover monotonic counter
- Power glitch detector
- General purpose register
- Control and status registers

It also incorporates security violation and tamper detection logic.

The SNVS _LP is a data storage subsystem with enhanced security capabilities. Its purpose is to store and protect system secure data, regardless of the main system power state.

SNVS _LP is in the always-powered-up domain, which is a separate power domain with its own power supply.

6.3.1 Behavior during system power down

When the chip power supply domain loses power, SNVS _LP continues to operate normally, and it ignores all inputs from SNVS _HP. The tamper input signal remains functional

6.3.2 Zeroizable master key (ZMK)

SNVS _LP incorporates logic for storage of a 256-bit ZMK value. This value (or the ZMK XORed with the OTPMK) can be selected as the master key input to DCP .

Either hardware or software can program the ZMK value. In hardware programming mode, software cannot read the ZMK value. In software programming mode, software can read the ZMK value before it is locked.

When there is a security violation, the ZMK is asynchronously zeroized and invalidated.

6.3.3 Secure real time counter (SRTC)

The SNVS _LP incorporates an autonomous SRTC. This is a non-rollover counter, which means that the SRTC does not rollover when it reaches the maximum value of all ones. Instead, a time rollover indication is generated to the SNVS _LP tamper monitor, which generates a security violation and interrupt.

6.3.3.1 Calibrating the SRTC time counter

To compensate for possible drift in the SRTC clock source, use the clock calibration mechanism implemented with the SRTC. This mechanism works the same way as the non-secure RTC clock calibration mechanism. Refer to the Calibrating the time counter for the detailed description of its functionality.

6.3.3.2 Time counter alarm (zmk)

The following figure shows the SNVS _LP Secure Real Time Counter, time alarms, and rollover security violation.

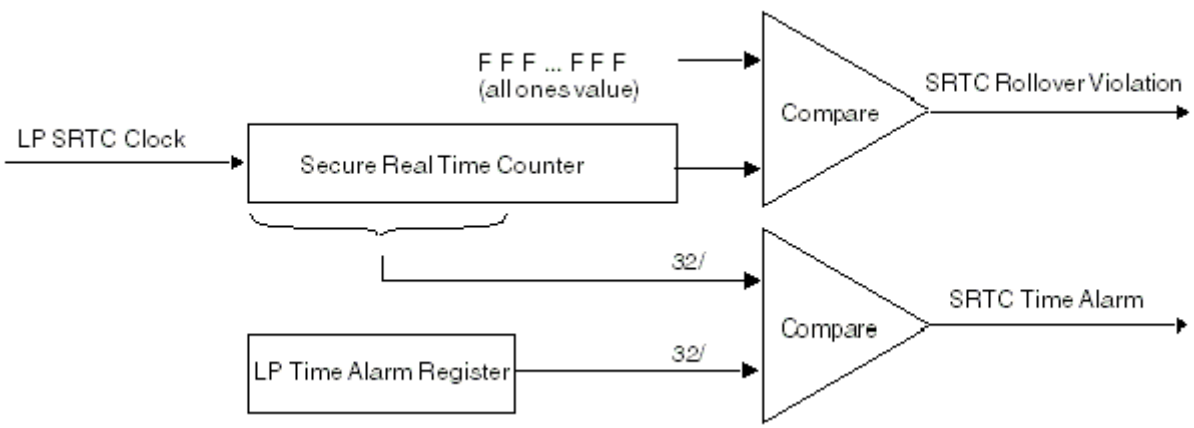


Figure 6-6. SNVS _LP secure real time counter

SNVS _LP has its own 32-bit LP Time Alarm register. As illustrated in Figure 6-6, this register generates an SRTC time alarm once the secure real time clock's 32 most significant bits match with the LP Time Alarm register. The time alarm can generate an interrupt to alert the host processor and can wake the host processor from one of its low-power modes (wait, doze, stop). This alarm can also wake up the entire system in the power-down mode by asserting the wake-up external output signal.

6.3.4 Monotonic counter (MC)

The following figure shows the MC and its rollover security violation.

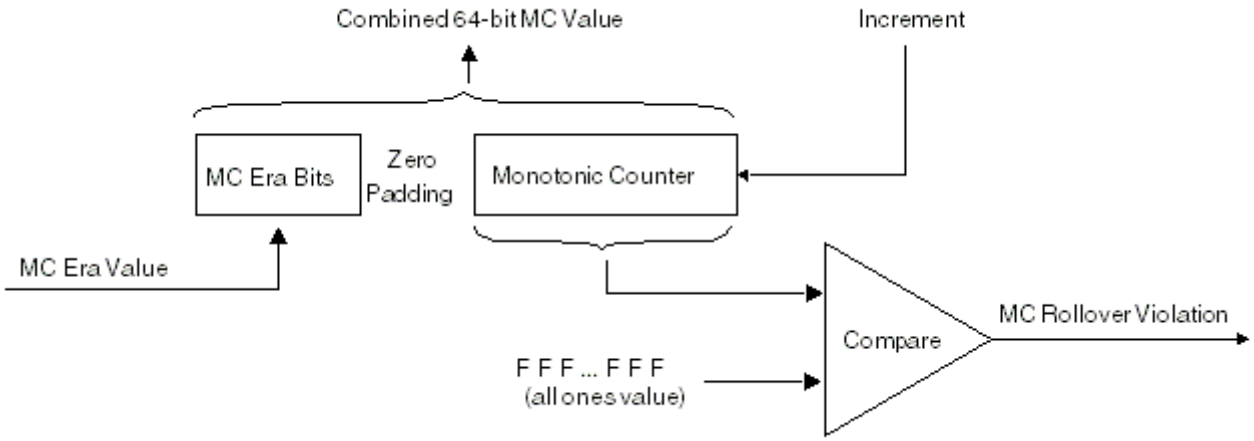


Figure 6-7. SNVS _LP monotonic counter

Some security applications require a monotonic counter (MC) that cannot be exhausted or returned to any previous value during the product's lifetime. Because the MC can never repeat a number, it cannot be reset or cycled back to its starting count. If it reaches its maximum value, it does not rollover. Instead, a monotonic counter rollover indication is generated to the SNVS _LP tamper monitor. This generates a security violation to the SSM and the host processor, which prevents a hacker from virtually running the MC backwards.

The SNVS uses an ERA value derived from the OTP elements as a mechanism for recovery from an MC failure (for example, due to a failure of LP power) where the MC value was compromised or cleared. The ERA value is prepended to the MC to form its most significant bits. Once any of the ERA value bits are set, the MC can count up from any value, including zero. This guarantees that any future value of the combined monotonic counter will be greater than any of its past values.

6.3.5 Power glitch detector (PGD)

The following figure shows the PGD mechanism.

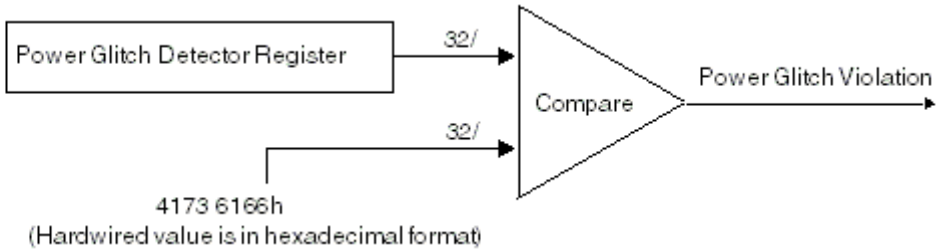


Figure 6-8. Power glitch detector

SNVS _LP incorporates a PGD mechanism to detect an attack in which someone power gates the SNVS _LP power supply for short periods to cause the LP control, status, and secure counter values to change. This mechanism detects and alerts the system whenever power supply glitches endanger SNVS _LP register values. The mechanism works as follows:

1. The PGD register (LPPGDR) is loaded to the known specific value 4173 6166h as part of the SNVS initialization process.
2. This register's value is compared to a hardwired value to detect a glitch in the LP power supply.

Power glitch detection is always enabled and cannot be disabled. Immediately after LP POR, this register is cleared and a power glitch violation is asserted. Therefore, before programming any feature in the SNVS, users should set a proper value into LPPGDR (see SNVS_LP Power Glitch Detector Register (SNVS_LPPGDR)) and should clear the power glitch record in the LP status register (see SNVS_LP Status Register (SNVS_LPSR)).

6.3.6 General-Purpose Register

The 32-bit general-purpose register allows data storage during system power-down mode as long as the SNVS _LP remains powered.

6.3.7 LP security violation/tamper policy

The following table describes SNVS _LP security violation sources.

Table 6-6. SNVS _LP security violation sources

Security violation/tamper source	Default behavior[1]	Configuration options[2]	Comments
Internal to LP Sources[3]			

Table continues on the next page...

Table 6-6. SNVS _LP security violation sources (continued)

Security violation/tamper source	Default behavior[1]	Configuration options[2]	Comments
SRTC Rollover Violation	Disable	• Enable • Disable	Asserted when SRTC is enabled and it reaches maximal value (all ones)
MC Rollover Violation	Disable	• Enable • Disable	Asserted when MC is enabled and it reaches maximal value (all ones)
Power Glitch Violation	Enable	-	Asserted when value in the Power Glitch Detector Register is different from the predefined hardwired value
External Tamper	Disable	• Enable • Disable	Asserted on the external tamper pin
From HP Section			
LP software violation	Enable	• Enable • Disable	Asserted by software
HP Security Violation Inputs 0-5	Disable	• Enable • Disable	Asserted on the Security Violation Port

1. Default behavior refers to the setting after LP POR, before the LP Security Violation policy is configured.
2. The configuration is set in the LPTDCR and LPSVCR registers. Once set, the configuration can be locked to prevent further changes.
3. These violation/tamper sources can be asserted and detected also in the system power-down mode.

Once generated, an LP security violation is reported to SNVS _HP. The source of the violation is recorded in the LP Status Register. An LP security violation clears the ZMK registers.

6.4 Modes of operation

The SNVS operates in either the system power-down or system power-up mode of operation.

During system power-down, SNVS _HP is powered-down. SNVS _LP is powered from the backup power supply and is electrically isolated from the rest of the chip. In this mode, SNVS _LP keeps its registers' values and monitors the SNVS _LP tamper detection inputs.

During system power-up, SNVS _HP and SNVS _LP are both powered-up and all SNVS functions are operational.

6.4.1 Operational states

The SNVS incorporates a system security monitor (SSM) module, which is responsible for monitoring system security violations and controlling security state of the system. The SSM states are defined as follows:

- Init.—System is powered up after system POR
- Check—System performs security checks
- Non-Secure (Functional)—System operates in non-secure state
- Trusted (Functional)—System operates in trusted state
- Secure (Functional)—System operates in secure state
- Soft Fail—Security violation/tamper was detected. Command to clear sensitive data is generated
- Hard Fail—System hard reset is requested

Three of the SSM states-trusted, secure, and non-secure-are considered functional states, meaning SSM states during which DCP and the rest of the chip's functional blocks are expected to operate normally.

6.5 SNVS clock sources

Table 6-7. SNVS clock sources

Clock	Description	Synchronization requirements
System peripheral clock	• Used by the SNVS internal logic, such as the system security monitor. • Can be gated outside of the module when SNVS indicates that it is not in use.	This is the main clock with respect to which the System IP bus access clock is synchronized.
System IP bus access clock	• Used by the SNVS for clocking its registers during read/write accesses. • Active only during the IP Bus access cycle.	Synchronized with the system peripheral clock.
LP SRTC clock	• Used by the Secure Real Time Counter. • Clock input should be driven directly from an external clock source (through the chip pin) without any intervening logic.	Does not have to be synchronous with the other clocks.
HP RTC clock	• Used by SNVS _HP real time counter.	Does not have to be synchronous with the other clocks.

6.6 SNVS reset and system power up

Table 6-8. Reset summary

Reset	Source	Characteristics	Internally resets
HP Hard	ipg_hard_async_reset_b	active-low, asynchronous	All SNVS _HP SNVS _LP registers and flops.
LP Power On Reset (POR)	lp_por_b	active-low, asynchronous	All SNVS _LP registers and flops
Scan Enter	T-Secure	active-high, synchronous, 2 cycles	The following registers/flops are cleared: • ZMK Registers and ZMK_VAL bit • SRTC_ENV bit • MC_ENV bit
Scan Exit	T-Secure	active-high, synchronous	All SNVS _LP registers except one Scan Exit status bit (SED) of the SNVS _LP Status Register (LPSR)
Field Return System Configuration	System Security Configuration Input	active-high	ZMK Registers and ZMK_VAL bit. This reset is permanently active in Field Return Configuration
LP software Reset	software	active-high, synchronous, 1 cycle	All SNVS _LP registers and flops. LP software Reset can be asserted if not disabled.
LP Security Violation	LP Tamper Monitor	active-high, asynchronous	The following registers/flops are cleared: • ZMK Registers and ZMK_VAL bit • SRTC_ENV bit • MC_ENV bit
ZMK Programming Mode Change	LP Control Register	active-high, synchronous, 1 cycle	The ZMK Registers and ZMK_VAL bit. This reset is asserted when a zero-to-one or one-to-zero transition occurs on the ZMK_HWP bit.

6.7 SNVS interrupts, alarms, and security violations

SNVS provides the following interrupt and alarm lines:

- Functional interrupt (active low)
- Security interrupt (active low)
- Wake-up alarm (active high)
- Security violation (active high)
- Hard failure violation (active high)

The security violation is triggered immediately upon an SSM transition to the soft fail state. Note that its assertion can be asynchronous as a result of an asynchronous LP tamper event. The SNVS hard failure indication is triggered immediately upon an SSM transition to the hard fail state.

The following table summarizes all SNVS interrupts and alarm sources.

Table 6-9. Interrupts and alarms summary

Interrupt/violation	Source	Default configuration[1]	Configuration options
SNVS functional interrupt	SRTC time alarm	Disable	Enable/Disable
	RTC time alarm	Disable	Enable/Disable
	RTC periodic interrupt	Disable	Enable/Disable
SNVS security interrupt	Security violation inputs	Disable	Enable/Disable
	SSM transition to the soft fail state	Enable	-
	LP security violation	Disable	Enable/Disable
SNVS wake-up alarm[2]	SRTC time alarm	Disable	Enable/Disable
	LP security violation	Disable	Enable/Disable
SNVS security violation	SSM transition to the soft fail state	Enable	-
SNVS hard failure violation	SSM transition to the hard fail state	Enable	-

1. Default behavior refers to the setting after LP/HP reset.
2. This alarm is functional only in the system power-down mode when the LP section is isolated.

6.8 Programming Guidelines

This section provides initialization and application information for the SNVS module.

6.8.1 RTC/SRTC control bits setting

All SNVS registers are programmed from the register bus. Therefore, any changes are synchronized with the IP clock. Several registers can also change synchronously with the RTC/SRTC clock after they are programmed. To avoid IP clock and RTC/SRTC clock synchronization issues, these values can only be programmed when the corresponding function is disabled. The following table presents the list of these values with the control bit setting required for programming.

Table 6-10. RTC/SRTC synchronized values list

Function	Value/register	Control bit setting
HP section		

Table continues on the next page...

Table 6-10. RTC/SRTC synchronized values list (continued)

Function	Value/register	Control bit setting
HP Real Time Counter	HPRTCMR and HPRTCLR Registers	RTC_EN = 0 - HPRTCMR/HPRTCLR can be programmed RTC_EN = 1 - HPRTCMR/HPRTCLR cannot be programmed
HP Time Alarm	HPTAMR and HPTALR Registers	HPTA_EN = 0 - HPTAMR/HPTALR can be programmed HPTA_EN = 1 - HPTAMR/HPTALR cannot be programmed
HP Time Calibration Value	HPCALB_VAL Value	HPCALB_EN = 0 - HPCALB_VAL can be programmed HPCALB_EN = 1 - HPCALB_VAL cannot be programmed
LP section		
LP Secure Real Time Counter	LPRTCMR and LPRTCLR Registers	SRTC_ENV = 0 - LPRTCMR/LPRTCLR can be programmed SRTC_ENV = 1 - LPRTCMR/LPRTCLR cannot be programmed
LP Time Alarm	LPTAR Register	LPTA_EN = 0 - LPTAR can be programmed LPTA_EN = 1 - LPTAR cannot be programmed
LP Time Calibration Value	LPCALB_VAL Value	LPCALB_EN = 0 - LPCALB_VAL can be programmed LPCALB_EN = 1 - LPCALB_VAL cannot be programmed

Use the following step to program synchronized values:

1. Check the enable bit value. If set, clear it.
2. Verify that the enable bit is cleared.

   There are two reasons to verify the enable bit's setting:

   - Enable bit clearing does not happen immediately; it takes three IPclock cycles and two RTC/SRTC clock cycles to change the enable bit's value.
   - If the enable bit is locked for programming, it cannot be cleared.
3. Program the desired value.
4. Set the enable bit; it takes three IP clock cycles and two RTC/SRTC clock cycles for the bit to set.

NOTE

Incrementing the value programmed into RTC/SRTC registers by two compensates for the two RTC/SRTC clock cycle delay that is required to enable the counter.

6.8.2 RTC/SRTC value read

There are two scenarios when software can read corrupted values from the RTC (HPRTCMR and HPRTCLR) and SRTC (LPSRTCMR and LPSRTCLR)registers:

- The RTC and SRTC counters are incremented by the slow 32 kHz clock, which is asynchronous to the system clock. The counter value is synchronized to the system clock before software reads that. The synchronization register may capture the counter value in the middle of the counter update. In this case, it is not guaranteed that all bits are properly sampled by the synchronization register; the value read by software can be wrong.
- The RTC and SRTC value is longer than the single bus read transaction of 32-bits. Therefore, software reads two registers, each holding a portion of the counter value. After reading one of these registers but before reading the second register, both registers may update their values. In this case, the value combined by software will be incorrect.

To avoid these issues, it is strongly recommended that software perform two consecutive reads of the RTC/SRTC value:

- If two consecutive reads are similar, the value is correct.
- If two consecutive reads are different, perform two more reads.

The worst case scenario may require three sessions of two consecutive reads.

6.8.3 ZMK programming guidelines

Either software or hardware can program the ZMK.

To program ZMK by software, do the following:

1. Verify that ZMK_HWP bit is not set.
2. Verify that ZMK is not locked for write.
3. Write key value to the ZMK registers.
4. Verify that the correct key value is written.
5. Set ZMK_VAL bit if the ZMK (or the ZMK XORed with the OTPMK) will be used by DCP as the master key. There is no need to set this bit if the ZMK registers are only read by software.
6. (optional) Set ZMK_ECC_EN bit to enable ZMK error correction code verification. Software can verify that the correct nine bit codeword is generated by reading ZMK_ECC_VALUE field.
7. (optional) Block software read accesses to the ZMK registers and ZMK_ECC_VALUE field by setting ZMK Read lock bit.
8. (optional) Block software write accesses to the ZMK registers by setting ZMK Write Lock bit.
9. Set MASTER_KEY_SEL and MKS_EN bits to select combination of OTPMK and ZMK to be provided to the hardware cryptographic module.

10. (optional) Block software write accesses to the MASTER_KEY_SEL field by setting MKS lock bit.

To program ZMK by hardware, do the following:

1. Set ZMK_HWP bit.
2. Verify that ZMK is not locked for write.
3. Write one to the PROG_ZMK bit.
4. Check when ZMK_VAL bit is set. Hardware sets this bit at the end of the ZMK programming cycle.
5. (optional) Set ZMK_ECC_EN bit to enable ZMK error correction code verification.

   Note that software cannot read the ZMK registers and ZMK_ECC_VALUE field in the hardware programming mode except in Fab configuration.

6. (optional) Block hardware programming option of the ZMK registers by setting ZMK Write Lock bit.
7. Set MASTER_KEY_SEL and MKS_EN bits to select the combination of OTPMK and ZMK to be provided to the hardware cryptographic module.
8. (optional) Block software write accesses to the MASTER_KEY_SEL field by setting MKS lock bit.

6.8.4 General initialization guidelines

Complete the following steps in order to properly initialize the module:

1. Transition SSM from check state to functional state (trusted/secure/non-secure).
2. Set the correct value in the Power Glitch Detector Register.
3. Clear the power glitch record in the LP Status Register.
4. User Specific: Enable security violations and interrupts in SNVS control and configuration registers.
5. User Specific: Program SNVS general functions/configurations.
6. User Specific: Set lock bits.

NOTE

Even if ZMK is not used, set the ZMK Write Lock bit. Otherwise, software can program a value to the ZMK register and select this value as a master key output.

6.9 SNVS Memory Map/Register Definition

This section contains detailed register descriptions for the SNVS registers. Each description includes a standard register diagram and register table. The register table provides detailed descriptions of the register bit and field function, in bit order.

SNVS registers consist of two types:

- Privileged read/write accessible
- Non-privileged read/write accessible

Privileged read/write accessible registers can only be accessed for read/write by privileged software. Unauthorized write accesses are ignored, and unauthorized read accesses return zero. Non-privileged software can access privileged access registers when the non-privileged software access enable bit is set in the SNVS _HP Command Register.

In addition, all privileged access registers (except HPSVSR and LPSR) can be written to only if the system security monitor is in one of the three functional states:

- Non-Secure
- Trusted
- Secure

Certain fields in the SNVS _HP Command Register can be written in non-functional system security monitor states (init, check, soft fail, and hard fail) as follows:

- SW_LPSV, SSM_ST, SW_SV, and SW_FSV can be written in check or soft fail state.
- The HAC_STOP bit can only be set in soft fail state but can be cleared in either soft fail or a functional state.
- HAC_LOAD, HAC_CLEAR bits and HPSVSR, LPSR Registers can be accessed in soft fail or a functional state.

The system security monitor state does not restrict read access to SNVS registers.

Non-privileged read/write accessible registers are read/write accessible by any software.

The following table shows the SNVS memory map. The LP register values are set only on LP POR and are unaffected by System (HP) POR. The HP registers are set only on System POR and are unaffected by LP POR.

SNVS memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SNVS_HP Lock Register (SNVS_HPLR)	32	R/W	0000_0000h	6.9.1/165
4	SNVS_HP Command Register (SNVS_HPCOMR)	32	R/W	0000_0000h	6.9.2/168
8	SNVS_HP Control Register (SNVS_HPCR)	32	R/W	0000_0000h	6.9.3/172

Table continues on the next page...

SNVS memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C	SNVS_HP Security Interrupt Control Register (SNVS_HPSICR)	32	R/W	0000_0000h	6.9.4/174
10	SNVS_HP Security Violation Control Register (SNVS_HPSVCR)	32	R/W	0000_0000h	6.9.5/176
14	SNVS_HP Status Register (SNVS_HPSR)	32	R/W	See section	6.9.6/178
18	SNVS_HP Security Violation Status Register (SNVS_HPSVSR)	32	R/W	0000_0000h	6.9.7/180
1C	SNVS_HP High Assurance Counter IV Register (SNVS_HPHACIVR)	32	R/W	0000_0000h	6.9.8/181
20	SNVS_HP High Assurance Counter Register (SNVS_HPHACR)	32	R/W	0000_0000h	6.9.9/182
24	SNVS_HP Real Time Counter MSB Register (SNVS_HPRTCMR)	32	R/W	0000_0000h	6.9.10/182
28	SNVS_HP Real Time Counter LSB Register (SNVS_HPRTCLR)	32	R/W	0000_0000h	6.9.11/183
2C	SNVS_HP Time Alarm MSB Register (SNVS_HPTAMR)	32	R/W	0000_0000h	6.9.12/183
30	SNVS_HP Time Alarm LSB Register (SNVS_HPTALR)	32	R/W	0000_0000h	6.9.13/184
34	SNVS_LP Lock Register (SNVS_LPLR)	32	R/W	0000_0000h	6.9.14/185
38	SNVS_LP Control Register (SNVS_LPCR)	32	R/W	0000_0000h	6.9.15/187
3C	SNVS_LP Master Key Control Register (SNVS_LPMKCR)	32	R/W	0000_0000h	6.9.16/189
40	SNVS_LP Security Violation Control Register (SNVS_LPSVCR)	32	R/W	0000_0000h	6.9.17/191
44	SNVS_LP Tamper Glitch Filters Configuration Register (SNVS_LPTGFCR)	32	R/W	0000_0000h	6.9.18/192
48	SNVS_LP Tamper Detectors Configuration Register (SNVS_LPTDCR)	32	R/W	0000_0000h	6.9.19/194
4C	SNVS_LP Status Register (SNVS_LPSR)	32	R/W	0000_0008h	6.9.20/196
50	SNVS_LP Secure Real Time Counter MSB Register (SNVS_LPSRTCMR)	32	R/W	0000_0000h	6.9.21/199
54	SNVS_LP Secure Real Time Counter LSB Register (SNVS_LPSRTCLR)	32	R/W	0000_0000h	6.9.22/199
58	SNVS_LP Time Alarm Register (SNVS_LPTAR)	32	R/W	0000_0000h	6.9.23/200
5C	SNVS_LP Secure Monotonic Counter MSB Register (SNVS_LPSMCMR)	32	R/W	0000_0000h	6.9.24/200
60	SNVS_LP Secure Monotonic Counter LSB Register (SNVS_LPSMCLR)	32	R/W	0000_0000h	6.9.25/201
64	SNVS_LP Power Glitch Detector Register (SNVS_LPPGDR)	32	R/W	0000_0000h	6.9.26/201
68	SNVS_LP General Purpose Register (SNVS_LPGPR)	32	R/W	0000_0000h	6.9.27/202
6C	SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR0)	32	R/W	0000_0000h	6.9.28/202
70	SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR1)	32	R/W	0000_0000h	6.9.28/202

Table continues on the next page...

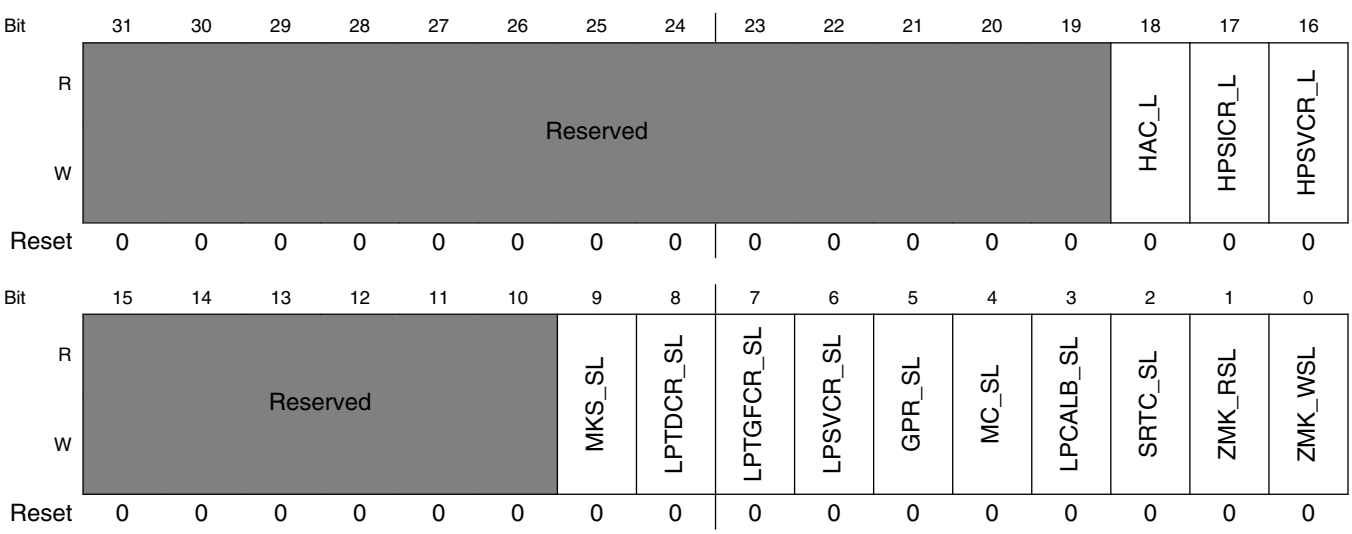
SNVS memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
74	SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR2)	32	R/W	0000_0000h	6.9.28/202
78	SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR3)	32	R/W	0000_0000h	6.9.28/202
7C	SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR4)	32	R/W	0000_0000h	6.9.28/202
80	SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR5)	32	R/W	0000_0000h	6.9.28/202
84	SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR6)	32	R/W	0000_0000h	6.9.28/202
88	SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR7)	32	R/W	0000_0000h	6.9.28/202
BF8	SNVS_HP Version ID Register 1 (SNVS_HPVIDR1)	32	R	See section	6.9.29/203
BFC	SNVS_HP Version ID Register 2 (SNVS_HPVIDR2)	32	R	0000_0000h	6.9.30/204

6.9.1 SNVS_HP Lock Register (SNVS_HPLR)

The SNVS _HP Lock Register contains lock bits for the SNVS registers. This is a privileged write register.

Address: 0h base + 0h offset = 0h



SNVS_HPLR field descriptions

Field	Description
31–19 -	This field is reserved. Reserved
18 HAC_L	High Assurance Configuration Lock When set, prevents any writes to HPHACIVR, HPHACR, and HAC_EN bit of HPCOMR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
17 HPSICR_L	HP Security Interrupt Control Register Lock When set, prevents any writes to the HPSICR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
16 HPSVCR_L	HP Security Violation Control Register Lock When set, prevents any writes to the HPSVCR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
15–10 -	This field is reserved. Reserved
9 MKS_SL	Master Key Select Soft Lock When set, prevents any writes to the MASTER_KEY_SEL field of the LPMKCR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
8 LPTDCR_SL	LP Tamper Detectors Configuration Register Soft Lock When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
7 LPTGFCR_SL	LP Tamper Glitch Filter Configuration Register Soft Lock When set, prevents any writes to the LPTGFCR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
6 LPSVCR_SL	LP Security Violation Control Register Soft Lock When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
5 GPR_SL	General Purpose Register Soft Lock When set, prevents any writes to the GPR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed

Table continues on the next page...

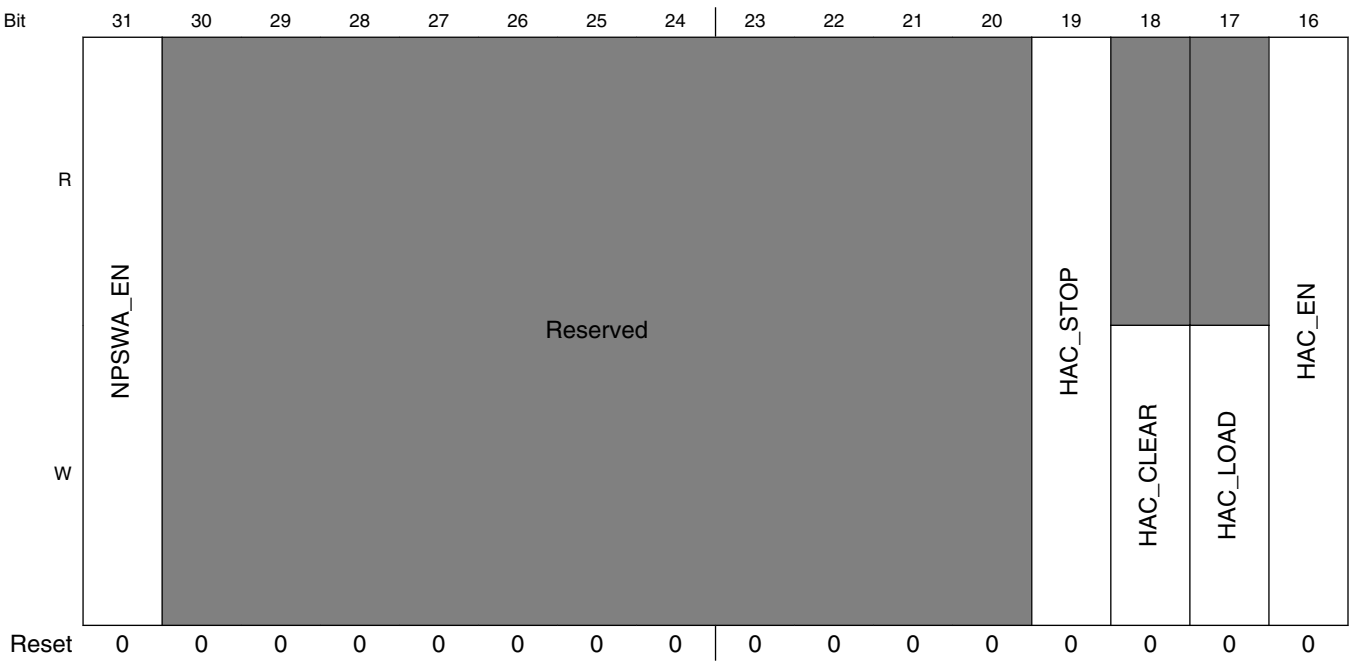
SNVS_HPLR field descriptions (continued)

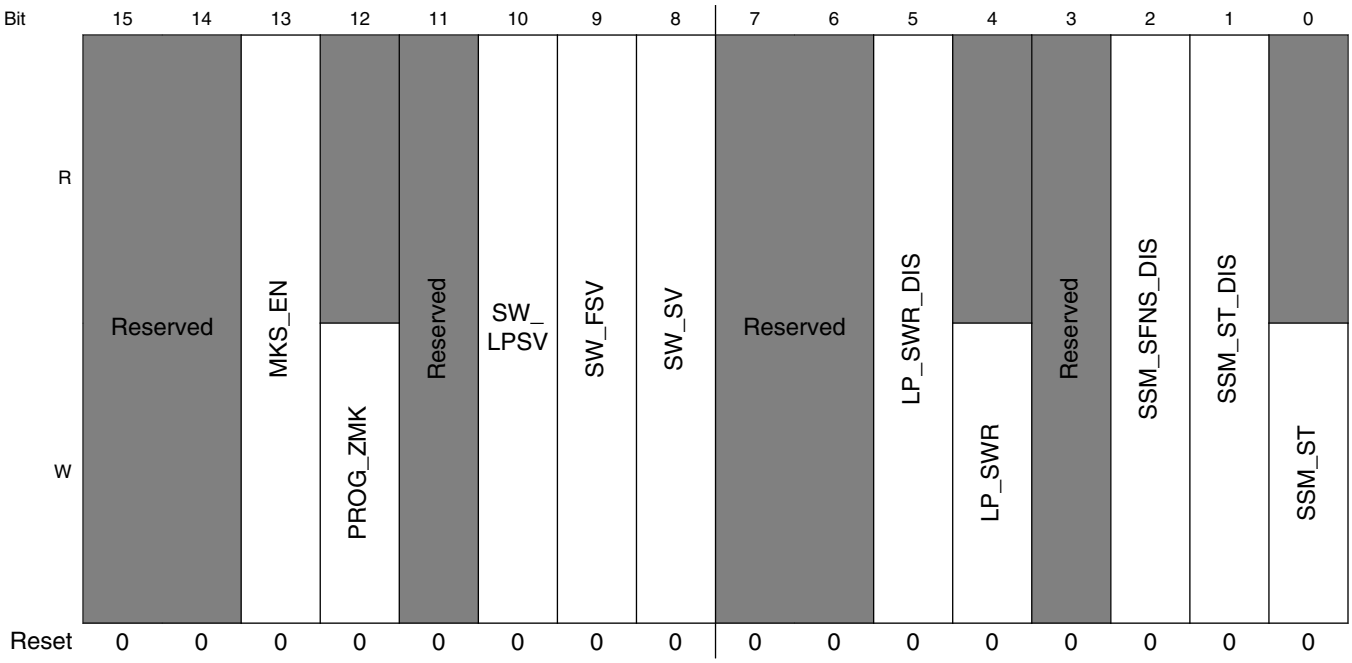
Field	Description
4 MC_SL	Monotonic Counter Soft Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the system reset. 0 Write access (increment) is allowed 1 Write access (increment) is not allowed
3 LPCALB_SL	LP Calibration Soft Lock When set, prevents any writes to the LP Calibration Value (LPCALB_VAL) and LP Calibration Enable (LPCALB_EN). Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
2 SRTC_SL	Secure Real Time Counter Soft Lock When set, prevents any writes to the SRTC Registers, SRTC_ENV, and SRTC_INV_EN bits. Once set, this bit can only be reset by the system reset. 0 Write access is allowed 1 Write access is not allowed
1 ZMK_RSL	Zeroizable Master Key Read Soft Lock When set, prevents any software reads to the ZMK Registers and ZMK_ECC_VALUE field of the LPMKCR. In ZMK hardware programming mode (ZMK_HWP is set), the ZMK and ZMK_ECC_VALUE cannot be read by software. Regardless of the bit setting, hardware can use the ZMK value when ZMK is selected. Once set, this bit can only be reset by the system reset. 0 Read access is allowed (only in software Programming mode) 1 Read access is not allowed
0 ZMK_WSL	Zeroizable Master Key Write Soft Lock When set, prevents any writes (software and hardware) to the ZMK registers and MASTER_KEY_SEL, ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be cleared by system reset. 0 Write access is allowed 1 Write access is not allowed

6.9.2 SNVS_HP Command Register (SNVS_HPCOMR)

The SNVS _HP Command Register contains the command, configuration, and control bits for the SNVS block. Some fields of this register can be written to in check and soft fail states in addition to the standard write access in functional states. This is a privileged write register.

Address: 0h base + 4h offset = 4h





SNVS_HPCOMR field descriptions

Field	Description
31 NPSWA_EN	Non-Privileged Software Access Enable When set, allows non-privileged software to access all SNVS registers, including those that are privileged software read/write access only. 0 Only privileged software can access privileged registers 1 Any software can access privileged registers
30–20 -	This field is reserved. Reserved
19 HAC_STOP	High Assurance Counter Stop This bit can be set only when SSM is in soft fail state. When set, it stops the high assurance counter and prevents transition to the hard fail state. This bit can be cleared in a functional or soft fail state. If the bit is cleared in the soft fail state, the high assurance counter counts down from the place where it was stopped. 0 HAC counter can count down 1 HAC counter is stopped
18 HAC_CLEAR	High Assurance Counter Clear When set, it clears the High Assurance Counter Register. It can be cleared in a functional or soft fail state. If the HAC counter is cleared in the soft fail state, the SSM transitions to the hard fail state if high assurance configuration is enabled (HAC_EN is set). This self-clearing bit is always read as zero. 0 No Action 1 Clear the HAC
17 HAC_LOAD	High Assurance Counter Load When set, it loads the High Assurance Counter Register with the value of the High Assurance Counter Load Register. It can be done in a functional or soft fail state. This self-clearing bit is always read as zero.

Table continues on the next page...

SNVS_HPCOMR field descriptions (continued)

Field	Description
	0 No Action 1 Load the HAC
16 HAC_EN	High Assurance Configuration Enable This bit controls the SSM transition from the soft fail to the hard fail state. When this bit is set and software fails to stop the HAC before it expires, the SSM transitions to the hard fail state. This bit cannot be changed once HAC_L bit is set. 0 High Assurance Configuration is disabled 1 High Assurance Configuration is enabled
15–14 -	This field is reserved. Reserved
13 MKS_EN	Master Key Select Enable When not set, the one time programmable (OTP) master key is selected by default. When set, the master key is selected according to the setting of the master key select field (MASTER_KEY_SEL) of LPMKCR. Once set, this bit can only be reset by the system reset. 0 OTP master key is selected as an SNVS master key 1 SNVS master key is selected according to the setting of the MASTER_KEY_SEL field of LPMKCR
12 PROG_ZMK	Program Zeroizable Master Key This bit activates ZMK hardware programming mechanism. This mechanism is activated only if the ZMK is configured to the hardware programming mode and ZMK in not locked for writes. This self-clearing bit is always read as zero. 0 No Action 1 Activate hardware key programming mechanism
11 -	This field is reserved. Reserved
10 SW_LPSV	LP Software Security Violation When set, SNVS _LP treats this bit as a security violation. The LP secure data is zeroized or invalidated according to the configuration. This security violation may result in a system security monitor transition if the LP Security Violation is enabled in the SNVS _HP Security Violation Control Register.
9 SW_FSV	Software Fatal Security Violation When set, the system security monitor treats this bit as a fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM: Check State → Soft Fail Non-Secure State → Soft Fail Trusted State → Soft Fail Secure State → Soft Fail
8 SW_SV	Software Security Violation When set, the system security monitor treats this bit as a non-fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM: Check → Non-Secure Trusted → Soft Fail Secure → Soft Fail

Table continues on the next page...

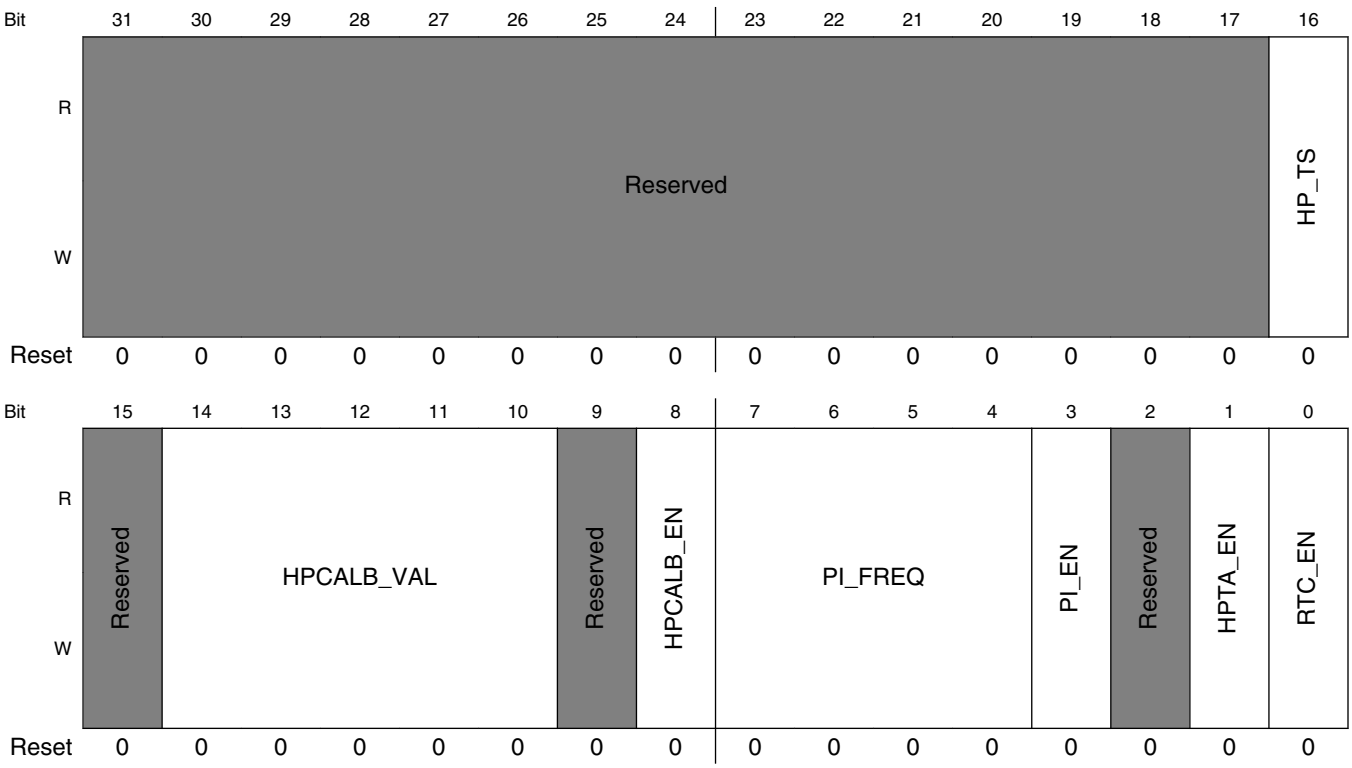
SNVS_HPCOMR field descriptions (continued)

Field	Description
7–6 -	This field is reserved. Reserved
5 LP_SWR_DIS	LP Software Reset Disable When set, disables the LP software reset. Once set, this bit can only be reset by the system reset. 0 LP software reset is enabled 1 LP software reset is disabled
4 LP_SWR	LP Software Reset When set, it resets the SNVS _LP section. This bit cannot be set when the LP_SWR_DIS bit is set. This self-clearing bit is always read as zero. 0 No Action 1 Reset LP section
3 -	This field is reserved. Reserved
2 SSM_SFNS_DIS	SSM Soft Fail to Non-Secure State Transition Disable When set, it disables the SSM transition from soft fail to non-secure state. Once set after the reset this bit cannot be changed 0 Soft Fail to Non-Secure State transition is enabled 1 Soft Fail to Non-Secure State transition is disabled
1 SSM_ST_DIS	SSM Secure to Trusted State Transition Disable When set, disables the SSM transition from secure to trusted state. Once set after the reset, this bit cannot be changed. 0 Secure to Trusted State transition is enabled 1 Secure to Trusted State transition is disabled
0 SSM_ST	SSM State Transition Transition state of the system security monitor. This self-clearing bit is always read as zero. This command results only in the following transitions of the SSM: Check State → Non-Secure (when Non-Secure Boot and not in Fab Configuration) Check State → Trusted (when Secure Boot or in Fab Configuration) Trusted State → Secure Secure State → Trusted (if not disabled by SSM_ST_DIS bit) Soft Fail → Non-Secure (if not disabled by SSM_SFNS_DIS bit)

6.9.3 SNVS_HP Control Register (SNVS_HPCR)

The SNVS _HP Control Register contains various control bits of the HP section of SNVS .

Address: 0h base + 8h offset = 8h



SNVS_HPCR field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HP_TS	HP Time Synchronize When set, this updates the HP Time Counter with the LP Time Counter value. This self-clearing bit is always read as zero. 0 No Action 1 Synchronize the HP Time Counter to the LP Time Counter
15 -	This field is reserved. Reserved
14–10 HPCALB_VAL	HP Calibration Value

Table continues on the next page...

SNVS_HPCR field descriptions (continued)

Field	Description
	Defines signed calibration value for the HP Real Time Counter. This field can be programmed only when RTC Calibration is disabled (HPCALB_EN is not set). This is a 5-bit 2's complement value. Hence, the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter. 00000 +0 counts per each 32768 ticks of the counter 00001 +1 counts per each 32768 ticks of the counter 00010 +2 counts per each 32768 ticks of the counter 01111 +15 counts per each 32768 ticks of the counter 10000 -16 counts per each 32768 ticks of the counter 10001 -15 counts per each 32768 ticks of the counter 11110 -2 counts per each 32768 ticks of the counter 11111 -1 counts per each 32768 ticks of the counter
9 -	This field is reserved. Reserved
8 HPCALB_EN	HP Real Time Counter Calibration Enabled Indicates that the time calibration mechanism is enabled. 0 HP Timer calibration disabled 1 HP Timer calibration enabled
7–4 PI_FREQ	Periodic Interrupt Frequency Defines frequency of the periodic interrupt. The interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the HP Real Time Counter and Real Time Counter and Periodic Interrupt are both enabled (RTC_EN and PI_EN are set). It is recommended to program this field when Periodic Interrupt is disabled (PI_EN is not set). The possible frequencies are: 0000 - bit 0 of the RTC is selected as a source of the periodic interrupt 0001 - bit 1 of the RTC is selected as a source of the periodic interrupt 0010 - bit 2 of the RTC is selected as a source of the periodic interrupt 0011 - bit 3 of the RTC is selected as a source of the periodic interrupt 0100 - bit 4 of the RTC is selected as a source of the periodic interrupt 0101 - bit 5 of the RTC is selected as a source of the periodic interrupt 0110 - bit 6 of the RTC is selected as a source of the periodic interrupt 0111 - bit 7 of the RTC is selected as a source of the periodic interrupt 1000 - bit 8 of the RTC is selected as a source of the periodic interrupt 1001 - bit 9 of the RTC is selected as a source of the periodic interrupt 1010 - bit 10 of the RTC is selected as a source of the periodic interrupt 1011 - bit 11 of the RTC is selected as a source of the periodic interrupt 1100 - bit 12 of the RTC is selected as a source of the periodic interrupt 1101 - bit 13 of the RTC is selected as a source of the periodic interrupt 1110 - bit 14 of the RTC is selected as a source of the periodic interrupt 1111 - bit 15 of the RTC is selected as a source of the periodic interrupt
3 PI_EN	HP Periodic Interrupt Enable The periodic interrupt can be generated only if the HP Real Time Counter is enabled. 0 HP Periodic Interrupt is disabled 1 HP Periodic Interrupt is enabled
2 -	This field is reserved. Reserved

Table continues on the next page...

SNVS_HPCR field descriptions (continued)

Field	Description
1 HPTA_EN	HP Time Alarm Enable When set, the time alarm interrupt is generated if the value in the HP Time Alarm Registers is equal to the value of the HP Real Time Counter. 0 HP Time Alarm Interrupt is disabled 1 HP Time Alarm Interrupt is enabled
0 RTC_EN	HP Real Time Counter Enable 0 RTC is disabled 1 RTC is enabled

6.9.4 SNVS_HP Security Interrupt Control Register (SNVS_HPSICR)

The HP Security Interrupt Control Register defines the SNVS security interrupt generation policy. This is a privileged write register.

Address: 0h base + Ch offset = Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LPSVI_EN	Reserved														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										SVI_EN5	SVI_EN4	SVI_EN3	SVI_EN2	SVI_EN1	SVI_EN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_HPSICR field descriptions

Field	Description
31 LPSVI_EN	LP Security Violation Interrupt Enable This bit enables generating of the security interrupt to the host processor upon security violation signal from the LP section. 0 LP Security Violation Interrupt is Disabled 1 LP Security Violation Interrupt is Enabled
30–6 -	This field is reserved. Reserved

Table continues on the next page...

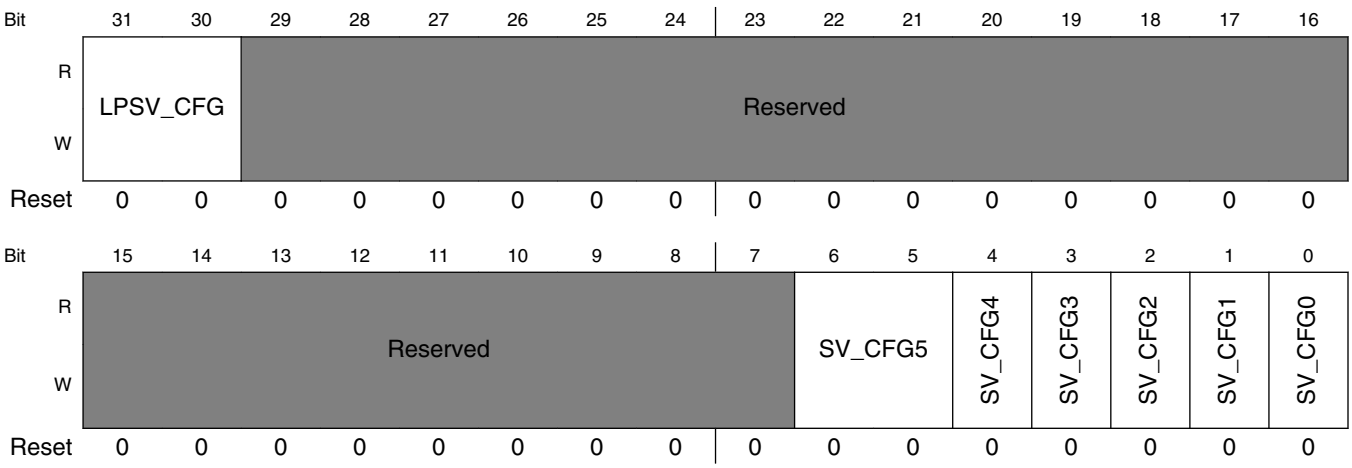
SNVS_HPSICR field descriptions (continued)

Field	Description
5 SVI_EN5	Security Violation Interrupt 5 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 5. 0 Security Violation Interrupt 5 is Disabled 1 Security Violation Interrupt 5 is Enabled
4 SVI_EN4	Security Violation Interrupt 4 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 4. 0 Security Violation Interrupt 4 is Disabled 1 Security Violation Interrupt 4 is Enabled
3 SVI_EN3	Security Violation Interrupt 3 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 3. 0 Security Violation Interrupt 3 is Disabled 1 Security Violation Interrupt 3 is Enabled
2 SVI_EN2	Security Violation Interrupt 2 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 2. 0 Security Violation Interrupt 2 is Disabled 1 Security Violation Interrupt 2 is Enabled
1 SVI_EN1	Security Violation Interrupt 1 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 1. 0 Security Violation Interrupt 1 is Disabled 1 Security Violation Interrupt 1 is Enabled
0 SVI_EN0	Security Violation Interrupt 0 Enable This bit enables generation of the security interrupt to the host processor upon security violation detection on input port 0. 0 Security Violation Interrupt 0 is Disabled 1 Security Violation Interrupt 0 is Enabled

6.9.5 SNVS_HP Security Violation Control Register (SNVS_HPSVCR)

The HP Security Violation Control Register defines types for each security violation input. This is a privileged write register.

Address: 0h base + 10h offset = 10h



SNVS_HPSVCR field descriptions

Field	Description
31–30 LPSV_CFG	LP Security Violation Configuration This field configures the LP security violation source. 00 LP security violation is disabled 01 LP security violation is a non-fatal violation 1x LP security violation is a fatal violation
29–7 -	This field is reserved. Reserved
6–5 SV_CFG5	Security Violation Input 5 Configuration This field configures the Security Violation Input 5. This setting instructs the SSM how to respond when a security violation on port 5 is detected. 00 Security Violation 5 is disabled 01 Security Violation 5 is a non-fatal violation 1x Security Violation 5 is a fatal violation
4 SV_CFG4	Security Violation Input 4 Configuration This field configures the security violation Input 4. This setting instructs the SSM how to respond when a security violation on port 4 is detected. 0 Security Violation 4 is a non-fatal violation 1 Security Violation 4 is a fatal violation

Table continues on the next page...

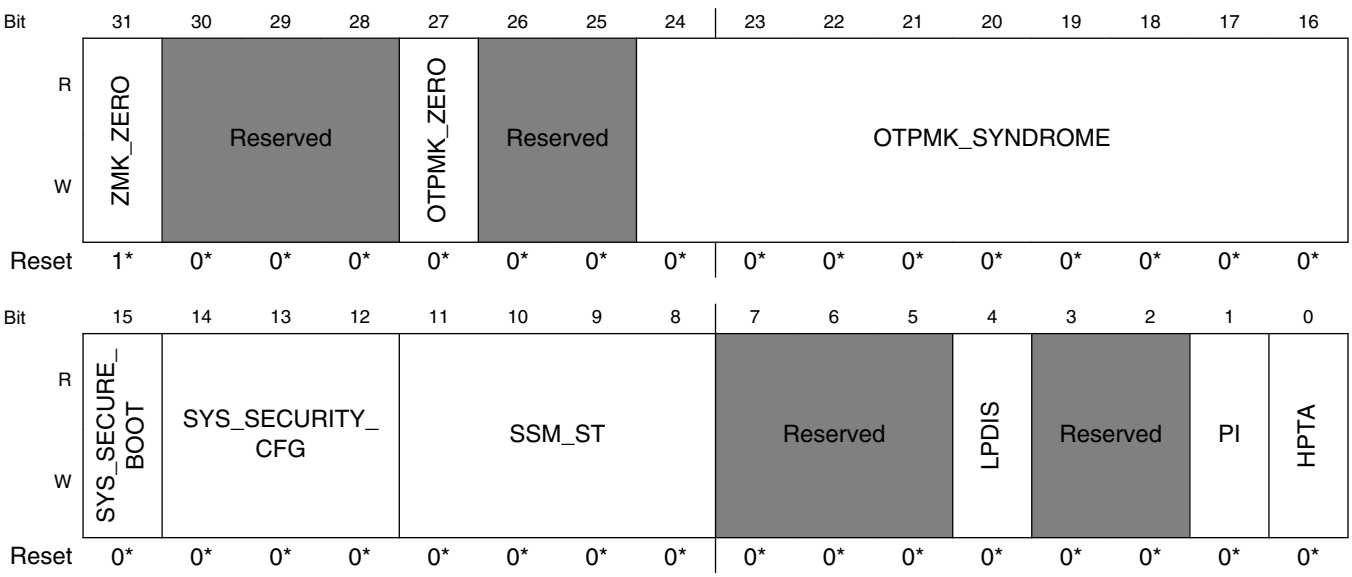
SNVS_HPSVCR field descriptions (continued)

Field	Description
3 SV_CFG3	Security Violation Input 3 Configuration This field configures the security violation input 3. This setting instructs the SSM how to respond when a security violation on port 3 is detected. 0 Security Violation 3 is a non-fatal violation 1 Security Violation 3 is a fatal violation
2 SV_CFG2	Security Violation Input 2 Configuration This field configures the security violation input 2. This setting instructs the SSM how to respond when a security violation on port 2 is detected. 0 Security Violation 2 is a non-fatal violation 1 Security Violation 2 is a fatal violation
1 SV_CFG1	Security Violation Input 1 Configuration This field configures the Security Violation Input 1. This setting instructs the SSM how to respond when a security violation on port 1 is detected. 0 Security Violation 1 is a non-fatal violation 1 Security Violation 1 is a fatal violation
0 SV_CFG0	Security Violation Input 0 Configuration This field configures the security violation input 0. This setting instructs the SSM how to respond when a security violation on port 0 is detected. 0 Security Violation 0 is a non-fatal violation 1 Security Violation 0 is a fatal violation

6.9.6 SNVS_HP Status Register (SNVS_HPSR)

The HP Status Register reflects the internal state of the SNVS .

Address: 0h base + 14h offset = 14h



* Notes:

- The reset value depends on the value of the LPZMKR registers and the value of the OTPMK programmed in fuses

SNVS_HPSR field descriptions

Field	Description
31 ZMK_ZERO	Zeroizable Master Key is Equal to Zero. When set, this bit triggers "bad key" violation if the ZMK is selected for use 0 The ZMK is not zero. 1 The ZMK is zero.
30–28 -	This field is reserved. Reserved
27 OTPMK_ZERO	One Time Programmable Master Key is Equal to Zero. When set, this bit always triggers "bad key" violation 0 The OTPMK is not zero. 1 The OTPMK is zero.
26–25 -	This field is reserved. Reserved
24–16 OTPMK_SYNDROME	One Time Programmable Master Key Syndrome Value The eight lower bits of this value indicate error location in case of a single-bit error. For example, syndrome word 10010110 indicates that key bit 150 has an error.

Table continues on the next page...

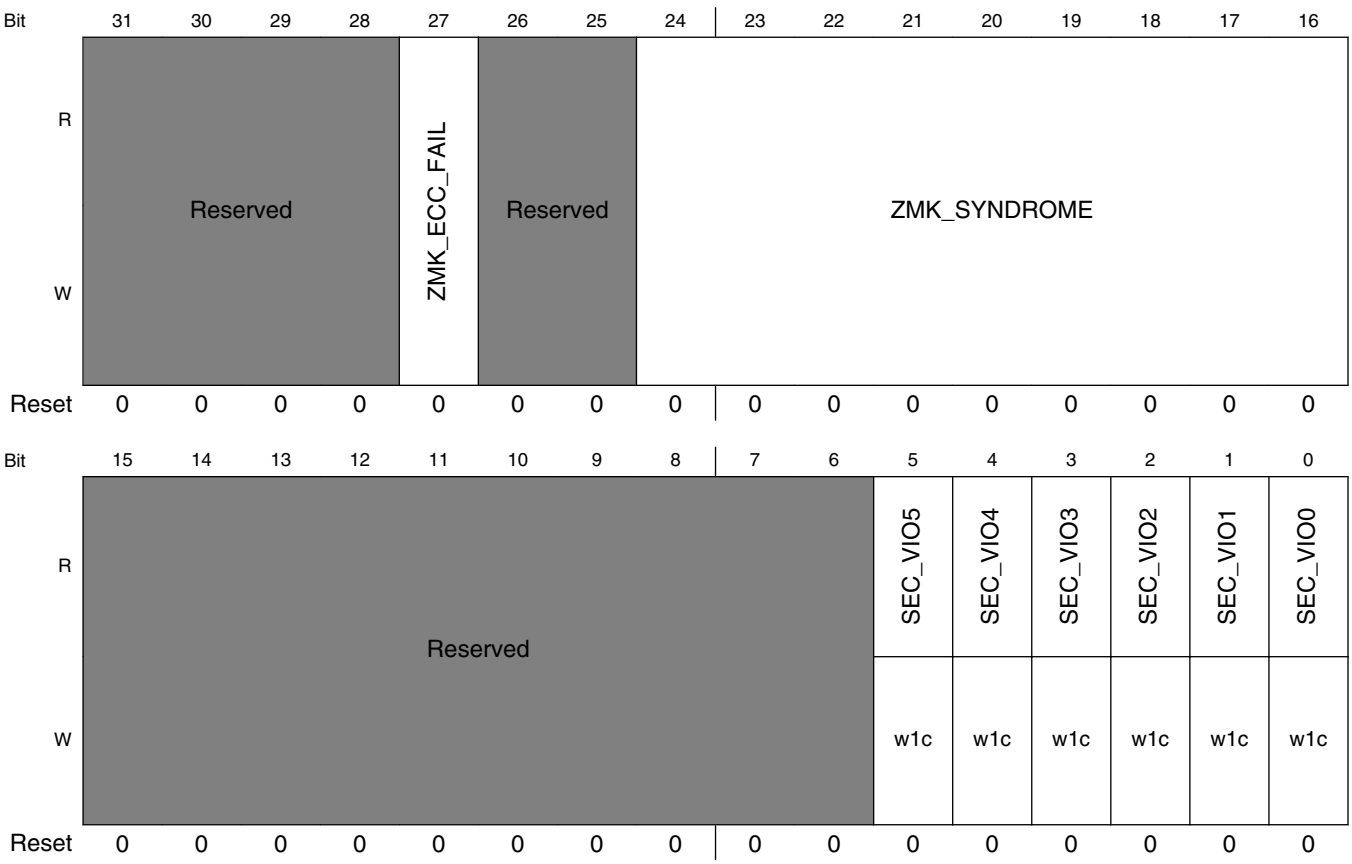
SNVS_HPSR field descriptions (continued)

Field	Description
	The ninth bit of the syndrome word checks parity of the whole key value. This bit is 1 when the odd number of errors are occurred and it is 0 when the number of errors is even. For example, if one of the eight bits indicates a failure and the ninth bit is zero then the number of errors in the one time programmable master key is at least 2 and it cannot be corrected. when one of the syndrome bits is set, the bad key violation is always generated
15 SYS_SECURE_BOOT	This bit reflects the value of the sys_secure_boot input signal to SNVS . If this bit is 1, the chip boots from internal ROM.
14–12 SYS_SECURITY_CFG	This field reflects the value of the sys_security_cfg input signal, which is defined as follows: 000 Fab Configuration - the default configuration of newly fabricated chips 001 Open Configuration - the configuration after Freescale- programmable fuses have been blown 01x Closed Configuration - the configuration after OEM-programmable fuses have been blown 1xx Field Return Configuration - the configuration of chips that are returned to Freescale for analysis
11–8 SSM_ST	System Security Monitor State This field contains the encoded state of the SSM's state machine. The encoding of the possible states are: 0000 Init 1000 Init Intermediate (transition state between Init and Check - SSM stays in this state only one clock cycle) 1001 Check 1011 Non-Secure 1101 Trusted 1111 Secure 0011 Soft Fail 0001 Hard Fail
7–5 -	This field is reserved. Reserved
4 LPDIS	Low Power Disable If 1, the SNVS low power section has been disabled by means of an input signal to the SNVS .
3–2 -	This field is reserved. Reserved
1 PI	Periodic Interrupt Indicates that periodic interrupt has occurred since this bit was last cleared. 0 No periodic interrupt occurred. 1 A periodic interrupt occurred.
0 HPTA	HP Time Alarm Indicates that the HP Time Alarm has occurred since this bit was last cleared. 0 No time alarm interrupt occurred. 1 A time alarm interrupt occurred.

6.9.7 SNVS_HP Security Violation Status Register (SNVS_HPSVSR)

The HP Security Violation Status Register reflects the HP domain security violation records. Write a 1 to SEC_VIO5-0 to clear the corresponding security violation detection flag. Note that this does not automatically clear the security violation signal that is connected to the input, so the security violation may immediately be detected again

Address: 0h base + 18h offset = 18h



SNVS_HPSVSR field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27 ZMK_ECC_FAIL	Zeroizable Master Key Error Correcting Code Check Failure When set, this bit triggers a bad key violation to the SSM and a security violation to the SNVS _LP section, which clears security sensitive data. Writing a one to this bit clears the record of this failure. It also clears this register's ZMK_SYNDROME field. 0 ZMK ECC Failure was not detected. 1 ZMK ECC Failure was detected.

Table continues on the next page...

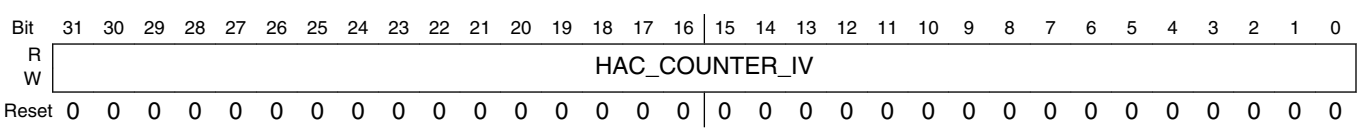
SNVS_HPSVSR field descriptions (continued)

Field	Description
26–25 -	This field is reserved. Reserved
24–16 ZMK_ SYNDROME	Zeroizable Master Key Syndrome Value The ZMK syndrome indicates error location and parity similar to the OTPMK syndrome . This value is set and locked when a ZMK ECC failure is detected. It is cleared by writing one into ZMK_ECC_FAIL bit.
15–6 -	This field is reserved. Reserved
5 SEC_VIO5	Security volation on input 5 is detected. 0 No security violation occurred on port 5. 1 Security violation occurred on port 5.
4 SEC_VIO4	Security violation on input 4 is detected. 0 No security violation occurred on port 4. 1 Security violation occurred on port 4.
3 SEC_VIO3	Security violation on input 3 is detected. 0 No security violation occurred on port 3. 1 Security violation occurred on port 3.
2 SEC_VIO2	Security violation on input 2 is detected. 0 No security violation occurred on port 2. 1 Security violation occurred on port 2.
1 SEC_VIO1	Security violation on input 1 is detected. 0 No security violation occurred on port 1. 1 Security violation occurred on port 1.
0 SEC_VIO0	Security violation on input 0 is detected. 0 No security violation occurred on port 0. 1 Security violation occurred on port 0.

6.9.8 SNVS_HP High Assurance Counter IV Register (SNVS_HPHACIVR)

The SNVS _HP High Assurance Counter IV Register contains the initial value for the high assurance counter.

Address: 0h base + 1Ch offset = 1Ch



SNVS_HPHACIVR field descriptions

Field	Description
31–0 HAC_ COUNTER_IV	High Assurance Counter Initial Value This register is used to set the starting count value to the high assurance counter. This register cannot be programmed when HAC_L bit is set.

6.9.9 SNVS_HP High Assurance Counter Register (SNVS_HPHACR)

The SNVS _HP High Assurance Counter Register contains the value of the high assurance counter. The high assurance counter is a delay introduced before the system security monitor transitions from soft fail to hard fail state if this transition is enabled.

Address: 0h base + 20h offset = 20h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HAC_COUNTER																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_HPHACR field descriptions

Field	Description
31–0 HAC_COUNTER	High Assurance Counter When the HAC_EN bit is set and the SSM is in the soft fail state, this counter starts to count down with the system clock. When the counter reaches zero, the SSM transitions to the Hard Fail State. • When HAC_STOP bit is set, the HAC Counter is stopped. • When HAC_CLEAR bit is set, the HAC Counter is cleared. • When HAC_LOAD bit is set, the HAC Counter is loaded with the value of the HPHACIVR.

6.9.10 SNVS_HP Real Time Counter MSB Register (SNVS_HPRTCMR)

The SNVS _HP Real Time Counter MSB register contains the most significant bits of the HP Real Time Counter.

Address: 0h base + 24h offset = 24h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																RTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_HPRTCMR field descriptions

Field	Description
31–15 -	This field is reserved. Reserved
14–0 RTC	HP Real Time Counter Most significant 15 bits. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

6.9.11 SNVS_HP Real Time Counter LSB Register (SNVS_HPRTCLR)

The SNVS _HP Real Time Counter LSB register contains the 32 least significant bits of the HP real time counter.

Address: 0h base + 28h offset = 28h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RTC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_HPRTCLR field descriptions

Field	Description
31–0 RTC	HP Real Time Counter Least significant 32 bits. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

6.9.12 SNVS_HP Time Alarm MSB Register (SNVS_HPTAMR)

The SNVS _HP Time Alarm MSB register contains the most significant bits of the SNVS _HP Time Alarm value.

Address: 0h base + 2Ch offset = 2Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																HPTA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_HPTAMR field descriptions

Field	Description
31–15 -	This field is reserved. Reserved
14–0 HPTA	HP Time Alarm Most significant 15 bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

6.9.13 SNVS_HP Time Alarm LSB Register (SNVS_HPTALR)

The SNVS _HP Time Alarm LSB register contains the 32 least significant bits of the SNVS _HP Time Alarm value.

Address: 0h base + 30h offset = 30h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HPTA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

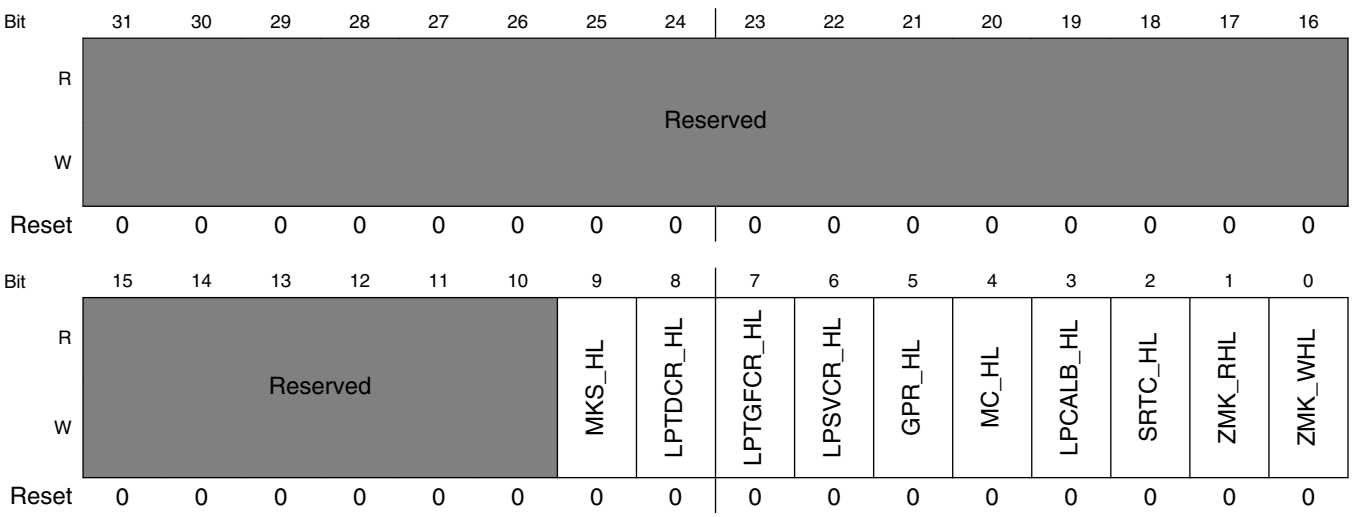
SNVS_HPTALR field descriptions

Field	Description
31–0 HPTA	HP Time Alarm Least significant bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

6.9.14 SNVS_LP Lock Register (SNVS_LPLR)

The SNVS _LP Lock Register contains lock bits for the SNVS _LP registers.

Address: 0h base + 34h offset = 34h



SNVS_LPLR field descriptions

Field	Description
31–10 -	This field is reserved. Reserved
9 MKS_HL	Master Key Select Hard Lock When set, prevents any writes to the MASTER_KEY_SEL field of the LP Master Key Control Register. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed. 1 Write access is not allowed.
8 LPTDCR_HL	LP Tamper Detectors Configuration Register Hard Lock When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed. 1 Write access is not allowed.
7 LPTGFCR_HL	LP Tamper Glitch Filter Configuration Register Hard Lock When set, prevents any writes to the LPTGFCR. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed. 1 Write access is not allowed.
6 LPSVCR_HL	LP Security Violation Control Register Hard Lock When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the LP POR.

Table continues on the next page...

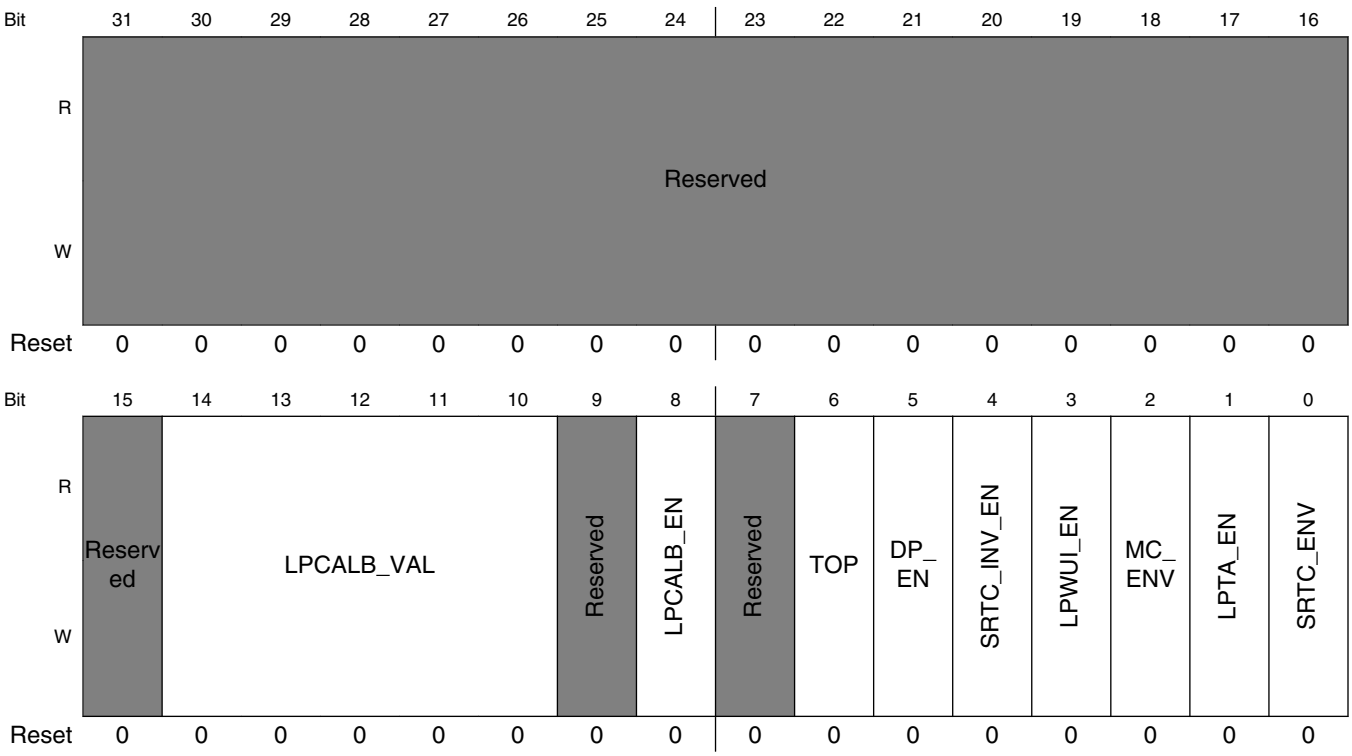
SNVS_LPLR field descriptions (continued)

Field	Description
	0 Write access is allowed. 1 Write access is not allowed.
5 GPR_HL	General Purpose Register Hard Lock When set, prevents any writes to the GPR. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed. 1 Write access is not allowed.
4 MC_HL	Monotonic Counter Hard Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the LP POR. 0 Write access (increment) is allowed. 1 Write access (increment) is not allowed.
3 LPCALB_HL	LP Calibration Hard Lock When set, prevents any writes to the LP Calibration Value (LPCALB_VAL) and LP Calibration Enable (LPCALB_EN). Once set, this bit can only be reset by the LP POR. 0 Write access is allowed. 1 Write access is not allowed.
2 SRTC_HL	Secure Real Time Counter Hard Lock When set, prevents any writes to the SRTC registers, SRTC_ENV, and SRTC_INV_EN bits. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed. 1 Write access is not allowed.
1 ZMK_RHL	Zeroizable Master Key Read Hard Lock When set, prevents any software reads to the ZMK registers and ZMK_ECC_VALUE field of the LPMKCR. In ZMK hardware programming mode (ZMK_HWP is set), software cannot read the ZMK or ZMK_ECC_VALUE. Regardless of the setting of this bit, hardware can use the ZMK value when ZMK is selected. Once set, this bit can only be reset by the LP POR. 0 Read access is allowed (only in software programming mode). 1 Read access is not allowed.
0 ZMK_WHL	Zeroizable Master Key Write Hard Lock When set, prevents any writes (software and hardware) to the ZMK registers and ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed. 1 Write access is not allowed.

6.9.15 SNVS_LP Control Register (SNVS_LPCR)

The SNVS _LP Control Register contains various control bits of the LP section of SNVS .

Address: 0h base + 38h offset = 38h



SNVS_LPCR field descriptions

Field	Description
31–15 -	This field is reserved. Reserved
14–10 LPCALB_VAL	LP Calibration Value Defines signed calibration value for SRTC. This field can be programmed only when SRTC calibration is disabled and not locked, i.e. when LPCALB_EN, LPCALB_SL, and LPCALB_HL bits are not set. This is a 5-bit 2's complement value. Hence, the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter clock 00000 +0 counts per each 32768 ticks of the counter clock 00001 +1 counts per each 32768 ticks of the counter clock 00010 +2 counts per each 32768 ticks of the counter clock 01111 +15 counts per each 32768 ticks of the counter clock 10000 -16 counts per each 32768 ticks of the counter clock 10001 -15 counts per each 32768 ticks of the counter clock

Table continues on the next page...

SNVS_LPCR field descriptions (continued)

Field	Description
	11110 -2 counts per each 32768 ticks of the counter clock 11111 -1 counts per each 32768 ticks of the counter clock
9 -	This field is reserved. Reserved
8 LPCALB_EN	LP Calibration Enable When set, enables the SRTC calibration mechanism. This bit cannot be changed once LPCALB_SL or LPCALB_HL bit is set. 0 SRTC Time calibration is disabled. 1 SRTC Time calibration is enabled.
7 -	This field is reserved. Reserved
6 TOP	Turn off System Power Asserting this bit causes a signal to be sent to the Power Management IC to turn off the system power. This bit will clear once power is off. This bit is only valid when the Dumb PMIC is enabled. 0 Leave system power on. 1 Turn off system power.
5 DP_EN	Dumb PMIC Enabled When set, software can control the system power. When cleared, the system requires a Smart PMIC to automatically turn power off. 0 Smart PMIC enabled. 1 Dumb PMIC enabled.
4 SRTC_INV_EN	Secure Real Time Counter Invalidation Enable When set, the SRTC is invalidated (SRTC_ENV bit is cleared) in the case of security violation. This field cannot be changed once SRTC_SL or SRTC_HL bit is set. 0 SRTC stays valid in the case of security violation. 1 SRTC is invalidated in the case of security violation.
3 LPWUI_EN	LP Wake-Up Interrupt Enable This interrupt line should be connected to the external pin and is intended to inform the external chip about an SNVS _LP event (tamper event, MC rollover,SRTC rollover, or time alarm). This wake-up signal can be asserted only when the chip (HP section) is powered down, and the LP section is isolated. 0 LP wake-up interrupt is disabled. 1 LP wake-up interrupt is enabled.
2 MC_ENV	Monotonic Counter Enable and Valid When set, the MC can be incremented (by write transaction to the LPSMCMR or LPSMCLR). This bit cannot be changed once MC_SL or MC_HL bit is set. 0 MC is disabled or invalid. 1 MC is enabled and valid.
1 LPTA_EN	LP Time Alarm Enable When set, the SNVS functional interrupt is asserted if the LP Time Alarm Register is equal to the 32 MSBs of the secure real time counter.

Table continues on the next page...

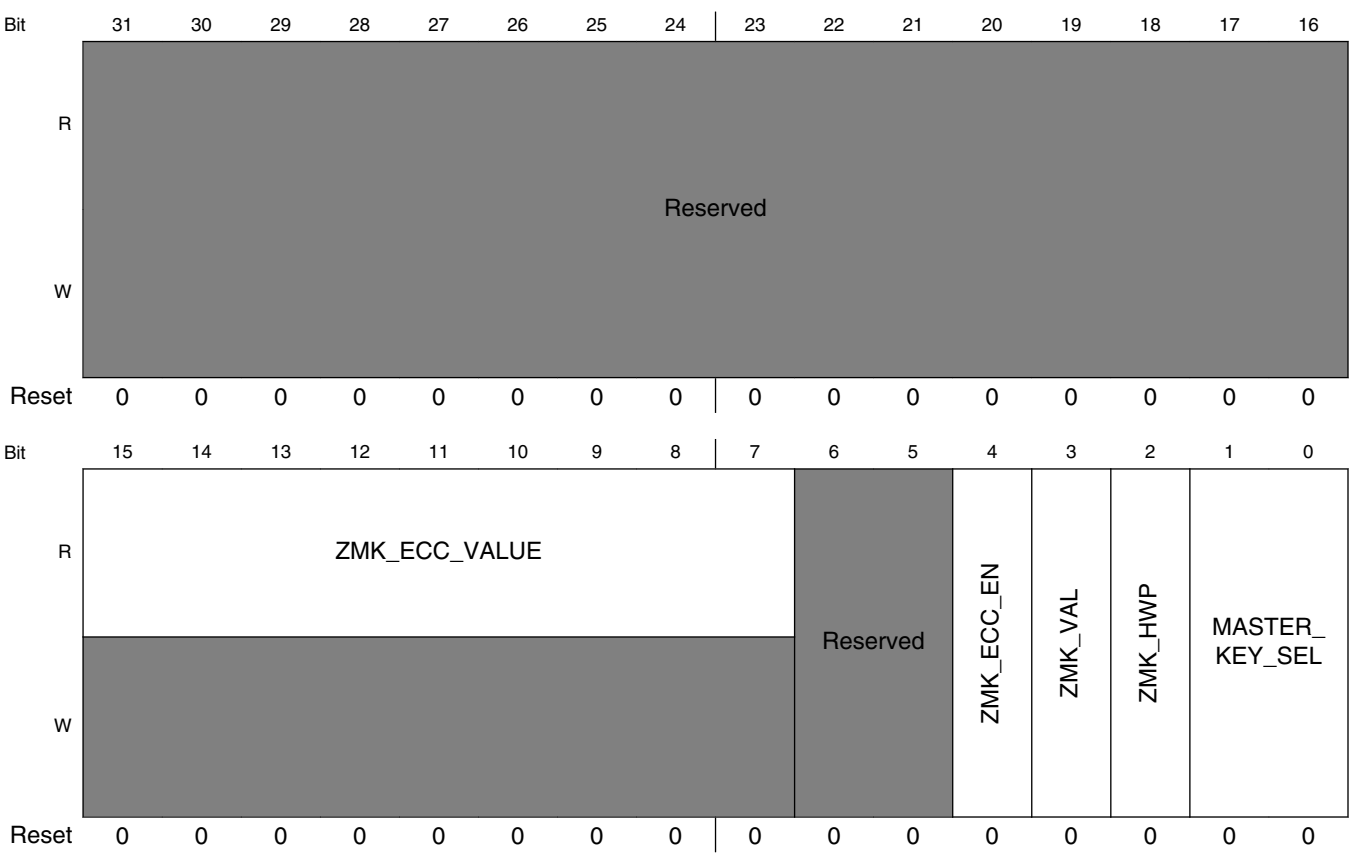
SNVS_LPCR field descriptions (continued)

Field	Description
	0 LP time alarm interrupt is disabled. 1 LP time alarm interrupt is enabled.
0 SRTC_ENV	Secure Real Time Counter Enable and Valid When set, the SRTC becomes operational. This bit cannot be changed once SRTC_SL or SRTC_HL bit is set. 0 SRTC is disabled or invalid. 1 SRTC is enabled and valid.

6.9.16 SNVS_LP Master Key Control Register (SNVS_LPMKCR)

The SNVS _LP Master Key Control Register contains the master keys configuration.

Address: 0h base + 3Ch offset = 3Ch



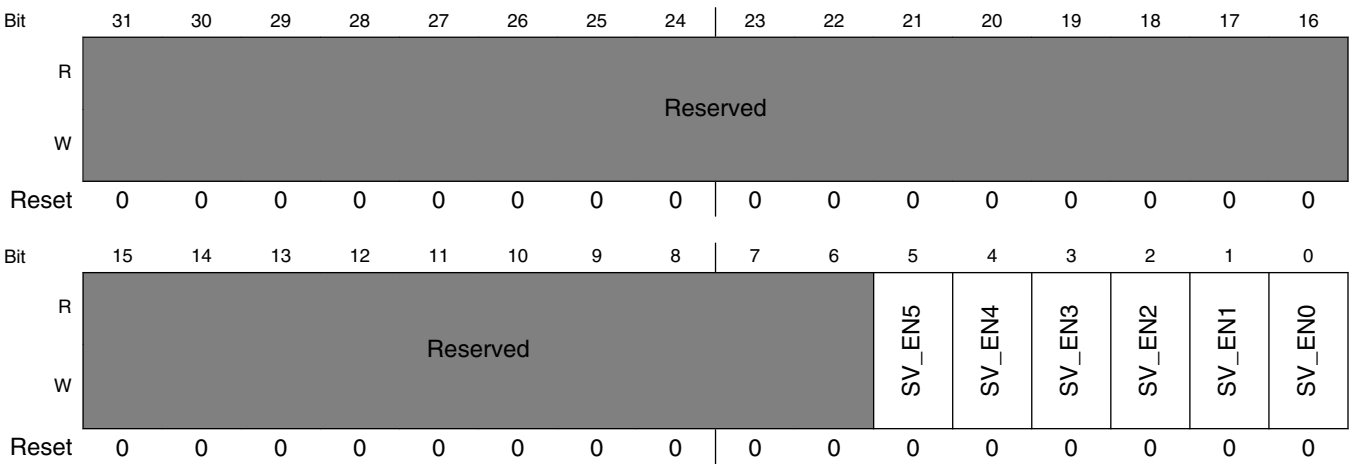
SNVS_LPMKCR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–7 ZMK_ECC_VALUE	Zeroizable Master Key Error Correcting Code Value This field is automatically calculated and set when one is written into ZMK_ECC_EN bit of this register. This field cannot be programmed by software. It keeps the ECC value of the zeroizable master key, which allows checking that ZMK has not been corrupted/altered with time. Note that this ZMK ECC code is equivalent to the ECC bits encoded into the OTPMK value but for the ZMK, the ECC value is kept separate from the ZMK value. See Error code for the ZMK for details. Read restrictions similar to the ZMK Registers are applied to this field (see SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR*n*)).
6–5 -	This field is reserved. Reserved
4 ZMK_ECC_EN	Zeroizable Master Key Error Correcting Code Check Enable Writing one to this field automatically calculates and sets the ZMK ECC value in the ZMK_ECC_VALUE field of this register. When both ZMK value is valid (ZMK_VAL is set) and ZMK ECC check is enabled (ZMK_ECC_EN is set), the ZMK value is continuously checked for the valid ECC word. If the ZMK ECC word calculated every clock cycle does not match the one recorded in this register, the ZMK ECC Check Fail Violation is generated. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0 ZMK ECC check is disabled. 1 ZMK ECC check is enabled.
3 ZMK_VAL	Zeroizable Master Key Valid When set, the ZMK value can be selected by the master key control block for use by cryptographic modules. In hardware programming mode, hardware sets this bit when the ZMK provisioning is complete. In software programming mode, software should set this bit. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0 ZMK is not valid. 1 ZMK is valid.
2 ZMK_HWP	Zeroizable Master Key hardware Programming mode When set, only the hardware key programming mechanism can set the ZMK and software cannot read it. When not set, the ZMK can be programmed only by software. See ZMK hardware programming mechanism for details. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0 ZMK is in the software programming mode. 1 ZMK is in the hardware programming mode.
1–0 MASTER_KEY_SEL	Master Key Select These bits select the SNVS Master Key output when Master Key Select bits are enabled by MKS_EN bit in the HPCOMR . When MKS_EN bit is not set, the one time programmable master key is selected by default. This field cannot be programmed when MKS_SL(or the hard lock) bit is set. 0x Select one time programmable master key. 10 Select zeroizable master key when MKS_EN bit is set . 11 Select combined master key when MKS_EN bit is set .

6.9.17 SNVS_LP Security Violation Control Register (SNVS_LPSVCR)

The LP Security Violation Control Register configures security violation inputs. This register cannot be programmed when the LPSVCR Lock bit is set. Note that configurations of the security violation inputs in the HP section (HPSVCR) and LP section (LPSVCR Register) are independent and have different functionality.

Address: 0h base + 40h offset = 40h



SNVS_LPSVCR field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5 SV_EN5	Security Violation 5 Enable This bit enables security violation input 5. When set, a security violation 5 causes an LP security violation, which clears LP sensitive data. 0 Security Violation 5 is disabled in the LP domain. 1 Security Violation 5 is enabled in the LP domain.
4 SV_EN4	Security Violation 4 Enable This bit enables security violation input 4. When set, a security violation 4 causes an LP security violation, which clears LP sensitive data. 0 Security Violation 4 is disabled in the LP domain. 1 Security Violation 4 is enabled in the LP domain.
3 SV_EN3	Security Violation 3 Enable This bit enables security violation input 3. When set, a security violation 3 causes an LP security violation, which clears LP sensitive data. 0 Security Violation 3 is disabled in the LP domain. 1 Security Violation 3 is enabled in the LP domain.

Table continues on the next page...

SNVS_LPSVCR field descriptions (continued)

Field	Description
2 SV_EN2	Security Violation 2 Enable This bit enables security violation input 2. When set, a security violation 2 causes an LP security violation, which clears LP sensitive data. 0 Security Violation 2 is disabled in the LP domain. 1 Security Violation 2 is enabled in the LP domain.
1 SV_EN1	Security Violation 1 Enable This bit enables security violation input 1. When set, a security violation 1 causes an LP security violation, which clears LP sensitive data. 0 Security Violation 1 is disabled in the LP domain. 1 Security Violation 1 is enabled in the LP domain.
0 SV_EN0	Security Violation 0 Enable This bit enables security violation input 0. When set, a security violation 0 causes an LP security violation, which clears LP sensitive data. 0 Security Violation 0 is disabled in the LP domain. 1 Security Violation 0 is enabled in the LP domain.

6.9.18 SNVS_LP Tamper Glitch Filters Configuration Register (SNVS_LPTGFCR)

The SNVS _LP Tamper Glitch Filters Configuration Register is used to configure the glitch filters for the SNVS _LP tamper inputs. This register cannot be programmed when the LPTGFCR_SL or LPTGFCR_HL bit is set.

Address: 0h base + 44h offset = 44h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								ETGF1_EN	Reserved		ETGF1				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

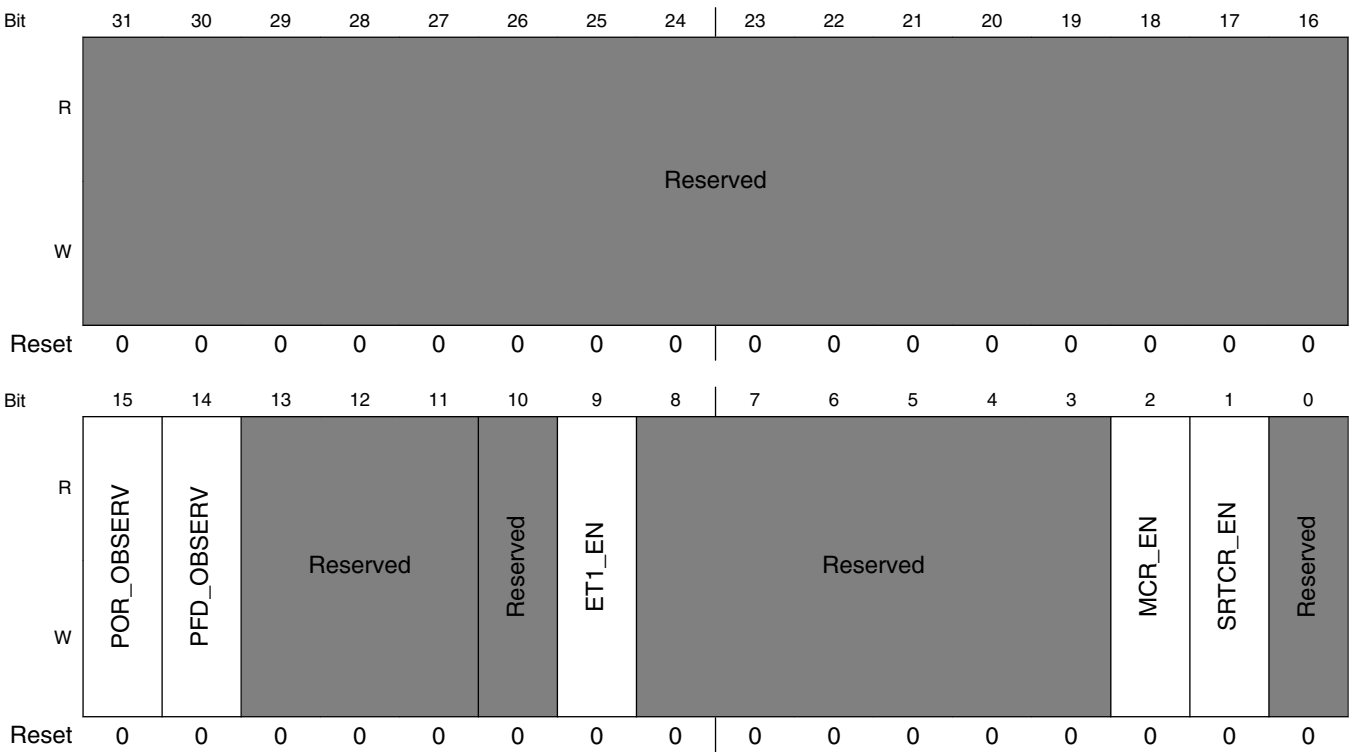
SNVS_LPTGFCR field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 ETGF1_EN	External Tamper Glitch Filter 1 Enable When set, enables the external tamper glitch filter 1. 0 External tamper glitch filter 1 is bypassed. 1 External tamper glitch filter 1 is enabled.
22–21 -	This field is reserved. Reserved
20–16 ETGF1	External Tamper Glitch Filter 1 Configures the length of the digital glitch filter for the external tamper 1 pin between 128 and 8064 SRTC clock cycles. Any assertion on external tamper 1 that is equal to or less than the value of the digital glitch filter is ignored. The length of the glitches filtered out is: 128 + (ETGF1 x 256), where ETGF1 = 0, ... , 31
15–0 -	This field is reserved. Reserved

6.9.19 SNVS_LP Tamper Detectors Configuration Register (SNVS_LPTDCR)

The SNVS _LP Tamper Detectors Configuration Register is used to configure analog and digital tamper detector sources. This register cannot be programmed when LPTDCR is locked for write.

Address: 0h base + 48h offset = 48h



SNVS_LPTDCR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 POR_OBSERV	Power On Reset (POR) Observability Flop The asynchronous reset input of this flop is connected directly to the output of the POR analog circuitry (external to the SNVS block). This flop can be used to detect brown-out voltage of the POR circuitry.
14 PFD_OBSERV	System Power Fail Detector (PFD) Observability Flop The asynchronous reset input of this flop is connected directly to the inverted output of the PFD analog circuitry (external to the SNVS block). This flop can be used to detect brown-out voltage of the PFD circuitry.
13–11 -	This field is reserved. Reserved

Table continues on the next page...

SNVS_LPTDCR field descriptions (continued)

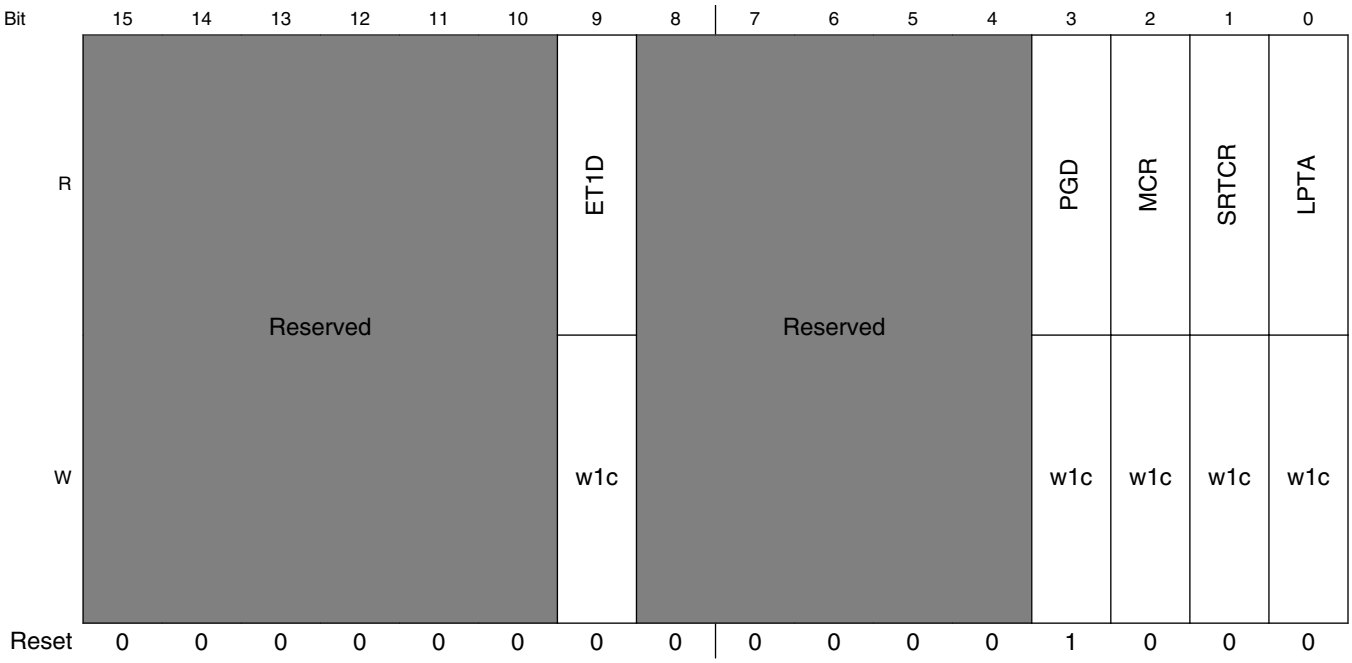
Field	Description
10 -	This field is reserved. Reserved
9 ET1_EN	External Tampering 1 Enable When set, external tampering 1 detection generates an LP security violation. 0 External tamper 1 is disabled. 1 External tamper 1 is enabled.
8–3 -	This field is reserved. Reserved
2 MCR_EN	MC Rollover Enable When set, an MC Rollover event generates an LP security violation. 0 MC rollover is disabled. 1 MC rollover is enabled.
1 SRTCR_EN	SRTC Rollover Enable When set, an SRTC rollover event generates an LP security violation. 0 SRTC rollover is disabled. 1 SRTC rollover is enabled.
0 -	This field is reserved. Reserved

6.9.20 SNVS_LP Status Register (SNVS_LPSR)

The SNVS _LP Status Register reflects the internal state and behavior of the SNVS _LP.

Address: 0h base + 4Ch offset = 4Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LPS	LPNS	Reserved									SED	Reserved	SPO	EO	ESVD
W												w1c		w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SNVS_LPSR field descriptions

Field	Description
31 LPS	LP Section is Secured Indicates that the LP section is provisioned/programmed in the secure or trusted state. The first write to the LP registers in secure or trusted state sets this bit. This bit can never be set together with the LPNS bit. When set the SNVS _LP section cannot be programmed , and ZMK cannot be read in the non-secure state of the SSM . 0 LP section was not programmed in secure or trusted state. 1 LP section was programmed in secure or trusted state.
30 LPNS	LP Section is Non-Secured Indicates that LP section was provisioned/programmed in the non-secure state. The first successful write to the LP Registers in non-secure state sets this bit. This bit can never be set together with the LPS bit. When set, the entire SNVS _LP section (all LP registers) are cleared upon an SSM transition from check to trusted state. 0 LP section was not programmed in the non-secure state. 1 LP section was programmed in the non-secure state.
29–21 -	This field is reserved. Reserved
20 SED	Scan Exit Detected 0 Scan exit was not detected. 1 Scan exit was detected.
19 -	This field is reserved. Reserved
18 SPO	Set Power Off This bit is set when power off was requested by a button press, but the button was not pressed for 5 seconds.

Table continues on the next page...

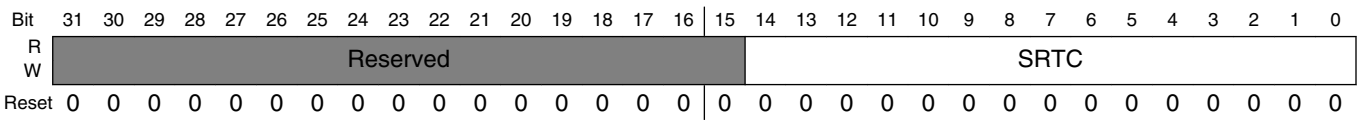
SNVS_LPSR field descriptions (continued)

Field	Description
	0 Emergency Off was not detected. 1 Emergency Off was detected..
17 EO	Emergency Off This bit is set when a power off is requested. 0 Emergency off was not detected. 1 Emergency off was detected.
16 ESVD	External Security Violation Detected Indicates that a security violation is detected on one of the HP security violation ports. The record of the port on which the violation has occurred can be found in the HP Security Violation Status Register. 0 No external security violation. 1 External security violation is detected.
15–10 -	This field is reserved. Reserved
9 ET1D	External Tampering 1 Detected. 0 External tampering 1 not detected. 1 External tampering 1 detected.
8–4 -	This field is reserved. Reserved
3 PGD	Power Supply Glitch Detected. 0 No power supply glitch. 1 Power supply glitch is detected.
2 MCR	Monotonic Counter Rollover. 0 MC has not reached its maximum value. 1 MC has reached its maximum value.
1 SRTCR	Secure Real Time Counter Rollover. 0 SRTC has not reached its maximum value. 1 SRTC has reached its maximum value.
0 LPTA	LP Time Alarm. 0 No time alarm interrupt occurred. 1 A time alarm interrupt occurred.

6.9.21 SNVS_LP Secure Real Time Counter MSB Register (SNVS_LPSRTCMR)

The SNVS _LP Secure Real Time Counter MSB register contains the most-significant bits of the secure real time counter.

Address: 0h base + 50h offset = 50h



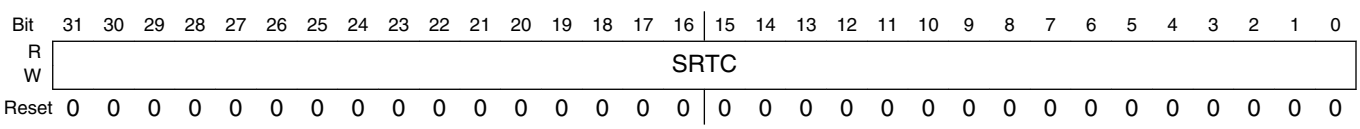
SNVS_LPSRTCMR field descriptions

Field	Description
31–15 -	This field is reserved. Reserved
14–0 SRTC	LP Secure Real Time Counter most significant 15 bits This register can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set.

6.9.22 SNVS_LP Secure Real Time Counter LSB Register (SNVS_LPSRTCLR)

The SNVS _LP Secure Real Time Counter LSB register contains the 32 least-significant bits of the secure real time counter.

Address: 0h base + 54h offset = 54h



SNVS_LPSRTCLR field descriptions

Field	Description
31–0 SRTC	LP Secure Real Time Counter least significant 32 bits This register can be programmed only when SRTC is not active and not locked, meaning the SRTC_ENV, SRTC_SL, and SRTC_HL bits are not set.

6.9.23 SNVS_LP Time Alarm Register (SNVS_LPTAR)

The SNVS _LP Time Alarm register contains the 32-bit LP Time Alarm value.

Address: 0h base + 58h offset = 58h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LPTA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_LPTAR field descriptions

Field	Description
31–0 LPTA	LP Time Alarm This register can be programmed only when the LP time alarm is disabled (LPTA_EN bit is not set).

6.9.24 SNVS_LP Secure Monotonic Counter MSB Register (SNVS_LPSMCMR)

The SNVS _LP Secure Monotonic Counter MSB Register contains the monotonic counter era bits and the most significant 16 bits of the monotonic counter. The monotonic counter is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register.

Address: 0h base + 5Ch offset = 5Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MC_ERA_BITS																MON_COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_LPSMCMR field descriptions

Field	Description
31–16 MC_ERA_BITS	Monotonic Counter Era Bits These bits are inputs to the module and typically connect to fuses.
15–0 MON_COUNTER	Monotonic Counter Most Significant 16 Bits The MC is incremented by one when: • A write transaction to the LPSMCMR or LPSMCLR register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set.

6.9.25 SNVS_LP Secure Monotonic Counter LSB Register (SNVS_LPSMCLR)

The SNVS _LP Secure Monotonic Counter LSB Register contains the 32 least significant bits of the monotonic counter. The MC is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register.

Address: 0h base + 60h offset = 60h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MON_COUNTER																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_LPSMCLR field descriptions

Field	Description
31–0 MON_COUNTER	Monotonic Counter bits The MC is incremented by one when: • A write transaction to the LPSMCMR or LPSMCLR Register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set.

6.9.26 SNVS_LP Power Glitch Detector Register (SNVS_LPPGDR)

The SNVS _LP Power Glitch Detector Register provides a 32 bit read write register, which is used for storing power glitch detector value as described in Power glitch detector (PGD).

Address: 0h base + 64h offset = 64h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PGD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_LPPGDR field descriptions

Field	Description
31–0 PGD	Power Glitch Detector Value

6.9.27 SNVS_LP General Purpose Register (SNVS_LPGPR)

The SNVS _LP General Purpose Register provides a 32 bit read write register, which can be used by any application for retaining 32 bit data during a power-down mode.

Address: 0h base + 68h offset = 68h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_LPGPR field descriptions

Field	Description
31–0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

6.9.28 SNVS_LP Zeroizable Master Key Register n (SNVS_LPZMKR*n*)

The SNVS _LP Zeroizable Master Key Registers contain the 256-bit zeroizable master key value. These registers are programmable as follows:

- When ZMK write lock bit is set, they cannot be programmed.
- When ZMK_HWP is not set, they are in software programming mode and can be programmed only by software.
- When ZMK_HWP is set, they are in hardware programming mode and can be programmed only by hardware.

These registers cannot be read by software when the ZMK_HWP or ZMK read lock bit is set.

Address: 0h base + 6Ch offset + (4d × i), where i=0d to 7d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ZMK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SNVS_LPZMKR*n* field descriptions

Field	Description
31–0 ZMK	Zeroizable Master Key Each of these registers contains part of the 256-bit ZMK value: LPZMKR0 - ZMK[31-0]

SNVS_LPZMKR*n* field descriptions (continued)

Field	Description
	LPZMKR1 - ZMK[63-32] LPZMKR2 - ZMK[95-64] LPZMKR3 - ZMK[127-96] LPZMKR4 - ZMK[159-128] LPZMKR5 - ZMK[191-160] LPZMKR6 - ZMK[223-192] LPZMKR7 - ZMK[255-224]

6.9.29 SNVS_HP Version ID Register 1 (SNVS_HPVIDR1)

The SNVS _HP Version ID Register 1 is a read-only register that contains the current version of the SNVS . The version consists of a module ID, a major version number, and a minor version number.

Address: 0h base + BF8h offset = BF8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IP_ID																MAJOR_REV								MINOR_REV							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

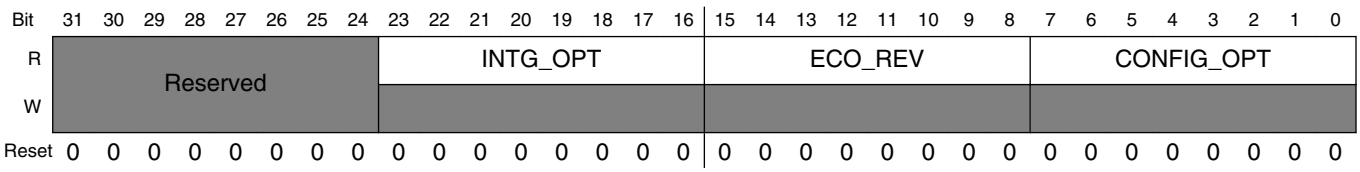
SNVS_HPVIDR1 field descriptions

Field	Description
31–16 IP_ID	SNVS block ID
15–8 MAJOR_REV	SNVS block major version number
7–0 MINOR_REV	SNVS block minor version number

6.9.30 SNVS_HP Version ID Register 2 (SNVS_HPVIDR2)

The SNVS _HP Version ID Register 2 is a read-only register that indicates the current version of the SNVS . The version consists of the following fields: integration options, ECO revision, and configuration options.

Address: 0h base + BFCh offset = BFCh



SNVS_HPVIDR2 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 INTG_OPT	SNVS Integration Option
15–8 ECO_REV	SNVS ECO Revision
7–0 CONFIG_OPT	SNVS Configuration Option

Chapter 7
System Boot

7.1 Overview

The boot process begins at Power On Reset (POR) where the hardware reset logic forces the ARM core to begin execution starting from the on-chip boot ROM.

Boot ROM code uses the state of the internal register BOOT_MODE[1:0] as well as the state of various eFUSEs and/or GPIO settings to determine the boot flow behavior of the device.

The main features of the ROM include:

- Support for booting from various boot devices
- Serial downloader support (USB OTG)
- Device configuration data (DCD)
- Digital signature High Assurance Boot (HAB)
- Wake-up from low power modes
- Plugin image

The boot ROM supports the following boot devices:

- NOR Flash
- OneNAND Flash
- SD/MMC
- Serial (I2C/SPI) NOR Flash and EEPROM

In normal operation, the Boot ROM uses the state of the BOOT_MODE register and eFUSEs to determine the boot device. For development purposes, eFUSEs used to determine the boot device may be overridden by using GPIO pin inputs.

Boot ROM code also allows the downloading of programs to be run on the device. An example is a provisioning program that can make further use of the serial connection to provision a boot device with a new image. Typically the provisioning program is

downloaded to internal RAM and allows the programming of boot devices, such as an SD/MMC Flash. The ROM Serial Downloader uses high speed USB in a non-stream mode connection.

The device configuration data (DCD) feature allows boot ROM code to obtain SOC configuration data from an external Program Image residing on the boot device. As an example, DCD can be used to program the DDR controller for optimal settings improving the boot performance. DCD is restricted to memory areas and peripheral addresses that are considered essential for boot purposes (see Table 7-29).

A key feature of the boot ROM is the ability to perform a secure boot or High Assurance Boot (HAB). This is supported by the HAB security library which is a subcomponent of the ROM code. HAB uses a combination of hardware and software together with a Public Key Infrastructure (PKI) protocol to protect the system from executing unauthorized programs. Before the HAB allows a user's image to execute, the image must be signed. The signing process is done during the image build process by the private key holder and the signatures are then included as part of the final Program Image. If configured to do so, the ROM verifies the signatures using the public keys included in the Program Image. A secure boot with HAB can be performed on all boot devices supported on the chip in addition to the Serial Downloader. The HAB library in the boot ROM also provides API functions, allowing additional boot chain components (bootloaders) to extend the secure boot chain. The out-of-fab setting for SEC_CONFIG is the Open configuration in which the ROM/HAB performs image authentication, but all authentication errors are ignored and the image is still allowed to execute.

7.2 Boot modes

During reset, the chip checks ARM core ID and Power Gating Controller status register.

On normal boot, the core's behavior is defined by the Boot Mode pins settings as described in Boot mode pin settings. On waking up from low power boot mode, the core skips clock settings. Boot ROM checks that PERSISTENT_ENTRY0 (see Persistent Bits) is a pointer to valid address space (OCRAM, DDR or EIM). If PERSISTENT_ENTRY0 is a pointer to valid range, it starts execution using entry point from PERSISTENT_ENTRY0 register. If PERSISTENT_ENTRY0 is a pointer to invalid range, the core performs system reset.

7.2.1 Boot mode pin settings

The device has four boot modes (one is reserved for Freescale use). Boot mode is selected based on the binary value stored in the internal BOOT_MODE register.

BOOT_MODE is initialized by sampling the BOOT_MODE0 and BOOT_MODE1 inputs on the rising edge of POR_B. After these inputs are sampled, their subsequent state does not affect the contents of the BOOT_MODE internal register. The state of the internal BOOT_MODE register may be read from the BMOD[1:0] field of the SRC Boot Mode Register (SRC_SBMR2). The available boot modes are: Boot From Fuses, serial boot via USB, and Internal Boot. See the table below for settings.

Table 7-1. Boot MODE Pin Settings

BOOT_MODE[1:0]	Boot Type
00	Boot From Fuses
01	Serial Downloader
10	Internal Boot
11	Reserved

7.2.2 High level boot sequence

The figure found here shows the high-level boot ROM code flow.

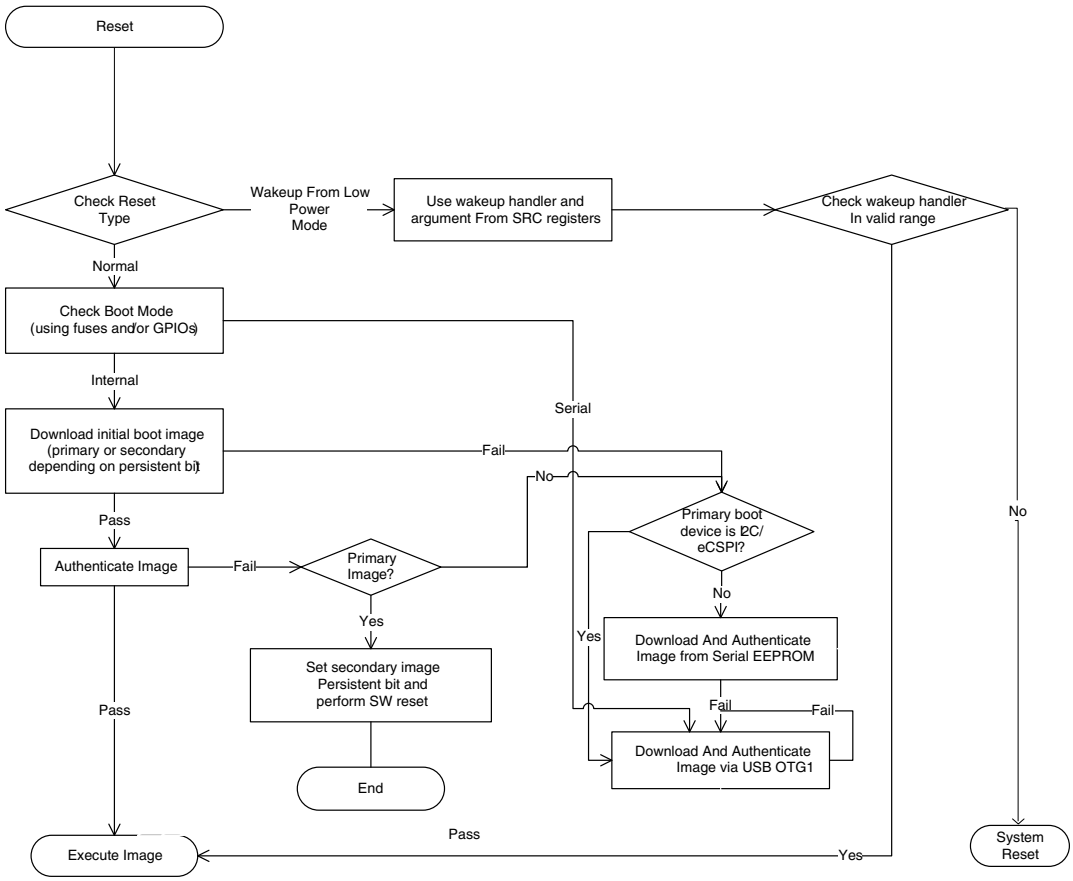


Figure 7-1. Boot Flow

NOTE

For External Interface Module (EIM) boot devices, downloading initial load region to OCRAM is skipped. IVT is read from EIM address space (see Image Vector Table and Boot Data). Copying initial load region and the rest of the program image is done only if the absolute start address of the image is not equal to EIM CS0 start address.

7.2.3 Boot From Fuses Mode (BOOT_MODE[1:0] = 00b)

A value of 00b in the BOOT_MODE[1:0] register selects the Boot From Fuses mode.

This mode is similar to the Internal Boot mode described in Internal Boot Mode (BOOT_MODE[1:0] = 0b10)with one difference. In this mode the GPIO boot override pins are ignored. The boot ROM code uses the boot eFUSE settings only. This mode also supports a secure boot using HAB.

If set to Boot From Fuses, the boot flow is controlled by the BT_FUSE_SEL eFUSE value. If BT_FUSE_SEL = 0, indicating that the boot device (for example, Flash, SD/MMC) has not yet been programmed, the boot flow jumps directly to the Serial Downloader. If BT_FUSE_SEL = 1, the normal boot flow is followed, where the ROM attempts to boot from the selected boot device.

The first time a board is used, the default eFUSEs may be configured incorrectly for the hardware on the platform. In such a case, the Boot ROM code may try to boot from a device that does not exist. This may cause an electrical/logic violation on some pads. Using Boot From Fuses mode addresses this problem.

The first time the BT_FUSE_SEL = 0 eFUSE is encountered, it is not blown (thus setting BT_FUSE_SEL). This forces the ROM code to jump directly to the Serial Downloader. This allows a bootloader to be downloaded which can then provision the boot device with a Program Image and blow the BT_FUSE_SEL and the other boot configuration eFUSEs. After reset, the Boot ROM code determines that BT_FUSE_SEL is blown (BT_FUSE_SEL = 1) and the ROM code performs internal boot according to the new eFUSE settings. This allows a user to set BOOT_MODE[1:0]=00b on a production device and burn fuses on the same device (by forcing entry to the Serial Downloader), without changing the value of BOOT_MODE[1:0] or pullups/pulldowns on the BOOT_MODE pins.

7.2.4 Serial Downloader

The Serial Downloader provides a means to download a Program Image to the chip over USB serial connection.

In this mode the ROM programs WDOG-1 for a 32-second time-out if WDOG_ENABLE eFuse is 1, and then continuously polls for USB connection. If no activity is found on USB OTG1 and the watchdog timer expires, the ARM core is reset.

NOTE

The downloaded image must continue to service the watchdog timer to avoid an undesired reset from occurring.

The USB boot flow is shown in the figure below.

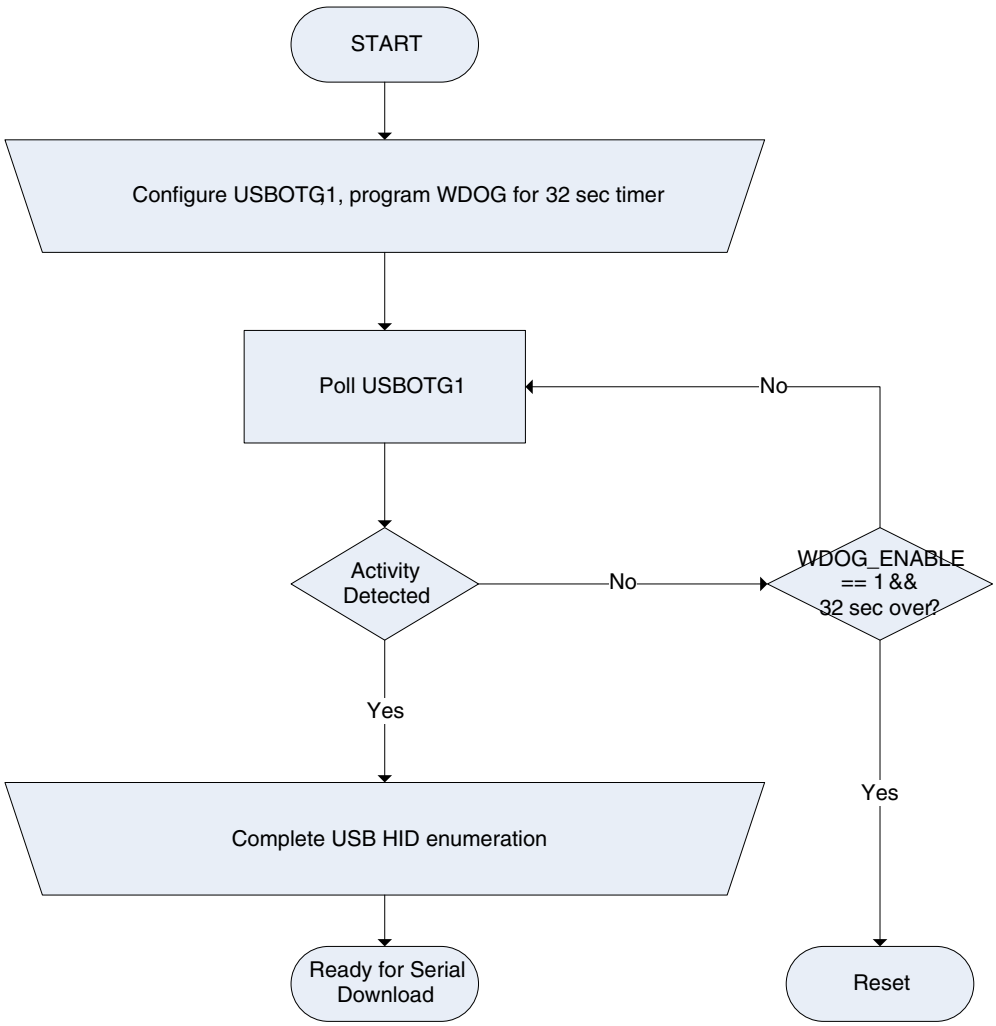


Figure 7-2. USB Boot Flow

7.2.5 Internal Boot Mode (BOOT_MODE[1:0] = 0b10)

A value of 0b10 in the BOOT_MODE[1:0] register selects the internal boot mode. In this mode, the processor continues to execute boot code from the internal boot ROM.

The boot code performs hardware initialization, loads the Program Image from the chosen boot device, performs image validation using the HAB library (see Boot security settings), and then jumps to an address derived from the Program Image. If any error occurs during internal boot, the boot code jumps to the Serial Downloader (see Serial Downloader). A secure boot using the HAB is possible in all the three boot modes.

When set to internal boot, the boot flow may be controlled by a combination of eFUSE settings with an option of overriding the fuse settings using General Purpose I/O (GPIO) pins. The GPIO Boot Select FUSE (BT_FUSE_SEL) determines whether the ROM uses GPIO pins for a select number of configuration parameters or eFUSEs in this mode. See Table 7-4 for more details.

- If BT_FUSE_SEL = 1, all boot options are controlled by the eFUSEs described in Table 7-2.
- If BT_FUSE_SEL = 0, specific boot configuration parameters may be set using GPIO pins rather than eFUSEs. The fuses that can be overridden when in this mode are indicated in the GPIO column of Table 7-2. Table 7-3 provides the details on the GPIO pins.

The use of GPIO overrides is intended for development since these pads are used for other purposes in deployed products. Freescale recommends controlling the boot configuration by eFUSEs in deployed products and reserving the use of the GPIO mode for development and testing purposes only.

7.2.6 Boot security settings

Internal boot modes use one of three security configurations:

- Closed: This level is intended for use with shipping secure products. All HAB functions are executed and security hardware is initialized (the Security Controller, or SNVS, enters Secure state), DCD is processed if present, and the program image is authenticated by HAB prior to its execution. All detected errors will be logged, and the boot flow aborted with control passing to the serial downloader. At this level, execution does not leave the internal ROM unless the target executable image has been authenticated.
- Open: This level is intended for use in non-secure products or during the development phases of a secure product. All HAB functions are executed as for a closed device. Security hardware is initialized (except the SNVS is left in Non-Secure state), DCD is processed if present, and the program image is authenticated by HAB prior to its execution. All detected errors will be logged, but have no influence on the boot flow, which continues as if the errors did not occur. This configuration is useful for secure product development, since the Program Image will run even if the authentication data is missing or incorrect, and the error log can be examined to determine the cause of authentication failure.
- Field Return: This level is intended for parts returned from shipped products.

NOTE

If the DIR_BT_DIS eFuse is not blown, authentication may be bypassed. In this case the system is not secure.

7.3 Device Configuration

This section describes the external inputs that control the behavior of the Boot ROM code.

This includes boot device selection (SPI, EIM, NOR, SD, MMC, etc.), boot device configuration (SD bus width, speed, etc), and so on. In general, the source for this configuration comes from eFUSEs embedded inside the chip. However, certain configuration parameters can be sourced from GPIO pins allowing further flexibility during the development process.

7.3.1 Boot eFUSE Descriptions

The table below is a comprehensive list of the configuration parameters that the ROM uses.

Table 7-2. Boot eFUSE Descriptions

Fuse	Configuration	Definition	GPIO[1]	Shipped Value	Settings[2]
DIR_BT_DIS	OEM	Disables Freescale reserved modes. Must be set for secure boot.	NA	0	0 Reserved Freescale modes enabled 1 Reserved Freescale modes disabled
BT_FUSE_SEL	OEM	In internal Boot mode BOOT_MODE[1:0] = 10, the BT_FUSE_SEL fuse determines whether the boot settings indicated by a Yes in the GPIO column are controlled by GPIO pins or eFUSE settings in the On-Chip OTP Controller (OCOTP). In Boot From Fuse mode BOOT_MODE[1:0] = 00, BT_FUSE_SEL fuse indicates whether bit configuration eFuses have been programmed.	NA	0	If BOOT_MODE[1:0] = 0b10 0 Bits of SBMR are overridden by GPIO pins. 1 Specific bits of SBMR are controlled by eFUSE settings. If BOOT_MODE[1:0] = 0b00 0 BOOT configuration eFuses are not yet programmed. Boot flow jumps to serial downloader. 1 BOOT configuration eFuses have been programmed. Regular boot flow is performed.

Table continues on the next page...

Table 7-2. Boot eFUSE Descriptions (continued)

Fuse	Configuration	Definition	GPIO[1]	Shipped Value	Settings[2]
SEC_CONFIG[1:0]	SEC_CONFIG[0] - Freescale SEC_CONFIG[1] - OEM	Security Configuration as defined in Boot security settings	NA	01	00 Reserved 01 Open (allows any program image, even if authentication fails) 1x Closed (Program image executes only if authenticated)
FIELD_RETURN	OEM	Enables Freescale reserved modes			0 - Freescale reserved modes are enabled/ disabled based on DIR_BT_DIS value 1 - Freescale reserved modes are enabled
SRK_HASH[255:0]	OEM	256-bit hash value of super root key (SRK_HASH)	NA	0	Settings vary - used by HAB
DIE-X-CORDINATE[7:0] DIE-Y-CORDINATE[7:0] WAFER_NO[4:0] LOT_NO_ENC[42:40] LOT_NO_ENC[39:32] LOT_NO_ENC[31:24] LOT_NO_ENC[23:16] LOT_NO_ENC[15:8] LOT_NO_ENC[7:0]	Freescale	Device Unique ID, 64-bit UID.	NA	Unique ID	Settings vary - used by HAB
BT_MMU_DISABLE	OEM	MMU/L1 D Cache/PL310 disable bit used by boot ROM for fast HAB processing	No	0	0 - MMU/L1 D Cache/PL310 is enabled by ROM during the boot 1 - MMU/L1 D Cache/PL310 is disabled by ROM during the boot
L1 I-Cache DISABLE	OEM	L1 I Cache disable bit used by boot during entire execution	No	0	0 - L1 I Cache is enabled by ROM during the boot 1 - L1 I Cache is disabled by ROM during the boot
BT_FREQ (BOOT_CFG2[2])	OEM	Frequency Selection	Yes	0	0 - ARM - 792MHz, DDR - 532MHz, AXI - 264MHz 1 - ARM - 396MHz, DDR - 352MHz, AXI - 176MHz
BOOT_CFG1[7:0]	OEM	Boot Configuration1	Yes	0	Specific to selected boot mode
BOOT_CFG2[7:0]	OEM	Boot Configuration2	Yes	0	Specific to selected boot mode
BOOT_CFG4[6:0]	OEM	Boot Configuration4		0	Specific to selected boot mode

Table continues on the next page...

Table 7-2. Boot eFUSE Descriptions (continued)

Fuse	Configuration	Definition	GPIO[1]	Shipped Value	Settings[2]
BOOT_CFG4[7]	OEM	Infinite Loop Enable at start of boot ROM. Used for debugging purposes. Ignored if DIR_BT_DIS is 1 and FIELD_RETURN is 0.	Yes	0	0 - Disabled 1 - Enabled
LPB_BOOT	OEM	USB Low Power Boot	No	0	00 - LPB Disable 01 - 1 GPIO (default frequencies) 10 - Divide by 2 11 - Divide by 4
BT_LPB_POLARITY	OEM	USB Low Power Boot GPIO polarity	No	0	0 - low on GPIO pad indicates low power condition 1 - high on GPIO pad indicates low power condition
WDOG_ENABLE	OEM	Watchdog reset counter enable	No	0	0 - watchdog reset counter is disabled during serial downloader 1 - watchdog reset counter is enabled during serial downloader
MMC_DLL_DLY[6:0]	OEM	USDHC Delay Line settings	No	0	USDHC Delay Line settings
SRK_REVOKE[2:0]	OEM	SRK revocation mask	No	0	SRK revocation mask
DISABLE_SDMMC_MFG	OEM	Disable SDMMC manufacture mode	No	0	0: enable SD/MMC MFG mode 1: disable SD/MMC MFG mode
PAD_SETTINGS	OEM	Override values for SD/MMC and NAND boot modes	No	0	Override the following IO PAD settings: PAD_SETTINGS[0] - Slew Rate PAD_SETTINGS[3:1] Drive Strength PAD_SETTINGS[5:4] - Speed Settings .
USE_L2_CACHE_AS_OCRAM (BOOT_CFG1[7])	OEM	L2 cache memory to be configured as OCRAM	Yes	0	0: L2 cache will be used as cache 1: L2 cache will be used as OCRAM[3]

1. Setting can be overridden by GPIO settings when GPIO_FUSE_SEL fuse is intact. See Table 1 for corresponding GPIO pin.
2. 0 = intact fuse and 1= blown fuse
3. L2 Cache will be locked after boot to prevent the leaking of cache contents.

7.3.2 GPIO Boot Overrides

The table below provides a list of GPIO boot overrides.

Table 7-3. GPIO Override Contact Assignments

Package Pin	Direction on reset	eFuse
BOOT_MODE1	Input	Boot Mode Selection
BOOT_MODE0	Input	
LCD_DAT0	Input	BOOT_CFG1[0]
LCD_DAT1	Input	BOOT_CFG1[1]
LCD_DAT2	Input	BOOT_CFG1[2]
LCD_DAT3	Input	BOOT_CFG1[3]
LCD_DAT4	Input	BOOT_CFG1[4]
LCD_DAT5	Input	BOOT_CFG1[5]
LCD_DAT6	Input	BOOT_CFG1[6]
LCD_DAT7	Input	BOOT_CFG1[7]
LCD_DAT8	Input	BOOT_CFG2[0]
LCD_DAT9	Input	BOOT_CFG2[1]
LCD_DAT10	Input	BOOT_CFG2[2]
LCD_DAT11	Input	BOOT_CFG2[3]
LCD_DAT12	Input	BOOT_CFG2[4]
LCD_DAT13	Input	BOOT_CFG2[5]
LCD_DAT14	Input	BOOT_CFG2[6]
LCD_DAT15	Input	BOOT_CFG2[7]
LCD_DAT16	Input	BOOT_CFG4[0]
LCD_DAT17	Input	BOOT_CFG4[1]
LCD_DAT18	Input	BOOT_CFG4[2]
LCD_DAT19	Input	BOOT_CFG4[3]
LCD_DAT20	Input	BOOT_CFG4[4]
LCD_DAT21	Input	BOOT_CFG4[5]
LCD_DAT22	Input	BOOT_CFG4[6]
LCD_DAT23	Input	BOOT_CFG4[7]

The input pins provided are sampled at boot, and can be used to override corresponding eFUSE values, depending on the setting of the BT_FUSE_SEL fuse. Table below describes boot options control sampling in different boot modes.

Table 7-4. Boot Options Control Selection

BOOT_MODE[1:0]	BT_FUSE_SEL Value	Boot Options Controlled By
00	0	eFUSEs
	1	

Table continues on the next page...

Table 7-4. Boot Options Control Selection (continued)

BOOT_MODE[1:0]	BT_FUSE_SEL Value	Boot Options Controlled By
01	0	GPIO pins
	1	eFUSEs
10	0	GPIO pins
	1	eFUSEs

7.3.3 Device Configuration Data

DCD is configuration information contained in a Program Image, external to the ROM, that the ROM interprets to configure various on-chip peripherals. See Device Configuration Data (DCD) for more details on Device Configuration Data.

7.4 Device Initialization

This section describes the details on the ROM and provides initialization details.

This includes details on:

- The ROM Memory Map
- The RAM Memory Map
- On-chip blocks that the ROM should make use of or change POR register default values
- Clock initialization
- Enabling the MMU/L2 cache when performing a secure boot (SEC_CONFIG = Closed)
- Exception handling and interrupt handling

7.4.1 Internal ROM /RAM memory map

Figure 7-3 shows the iROM memory map for the i.MX 6SoloLite.

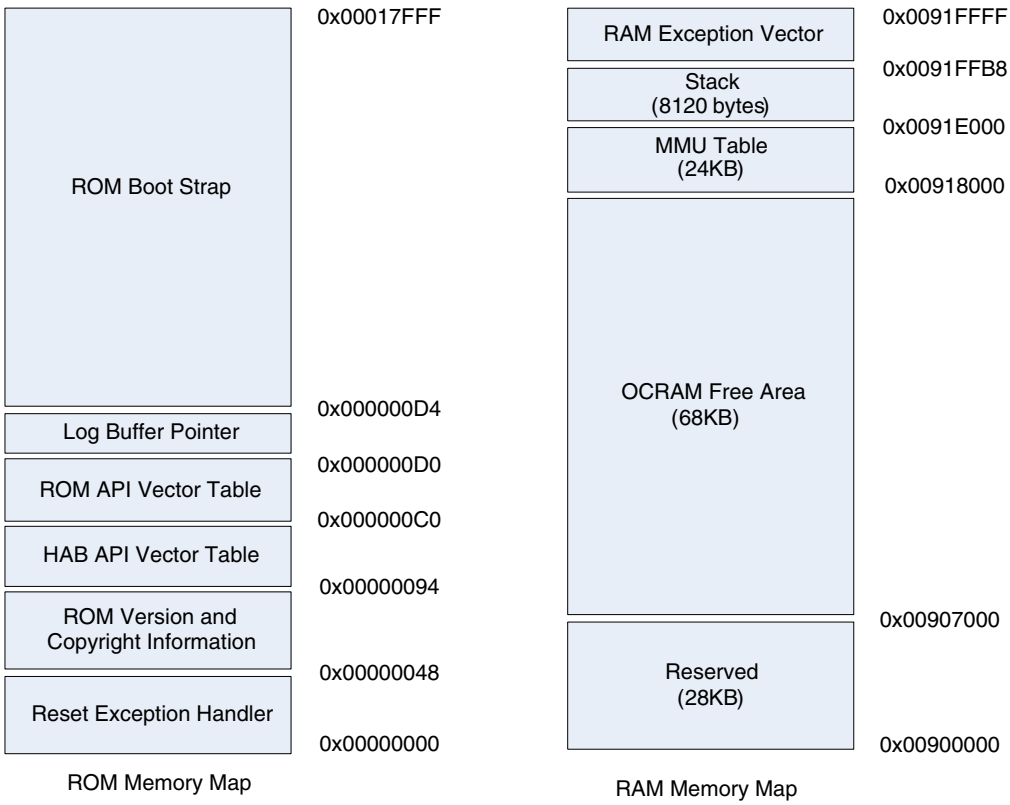


Figure 7-3. Internal Rom and Ram memory map for i.MX 6SoloLite

NOTE

The entire OCRAM region can be used freely post boot.

7.4.2 Boot Block Activation

The boot ROM affects a number of different hardware blocks which are activated and play a vital role in the boot flow.

The ROM configures and uses the following blocks (listed in alphabetical order) during the boot process. Note that the blocks actually used depend on the boot mode and boot device selection:

- CCM - Clock Control Module
- ECSPI - Enhanced Configurable Serial Peripheral Interface
- EIM - External Interface Module. Used for NOR and OneNAND devices
- I2C - I2C Controller
- OCOTP_CTRL - On-Chip OTP Controller. The OCOTP contains the eFUSEs.

- IOMUXC - I/O Multiplexer Control allows GPIO use to override eFUSE boot settings
- DCP - Data CoProcessor Module for Hash acceleration
- SNVS - Secure Non-Volatile Storage
- SRC - System Reset Controller
- USB - Used for serial download of a boot device provisioning program
- USDHC - Ultra Secure Digital Host Controller
- WDOG-1 - Watchdog Timer

7.4.3 Clocks at Boot Time

The table below shows the various clocks and their sources used by ROM.

Table 7-5. Normal Frequency Clocks Configuration

Clock	CCM signal	Source	Frequency(MHz) BT_FREQ=0	Frequency(MHz) BT_FREQ=1
System PLL	pll1_sw_clk		792	792
528MHz PLL	pll2_sw_clk		528	528
480MHz PLL	pll3_sw_clk		480	480
ARM core clock	arm_clk_root	System PLL	792	396
EIM	emi_slow_clk_root	528MHz PLL	132	132
AHB	ahb_clk_root	528MHz PLL/PFD352	132	76.6
IPG	ipg_clk_root	528MHz PLL/PFD352	66	38.3
AXI_A	axi_a	528MHz PLL/PFD352	396	306
AXI_B	axi_b	528MHz PLL/PFD352	264	153.3
USB	usboh3_clk_root	480MHz PLL	60	60
USDHC	usdhc1_clk_root usdhc2_clk_root usdhc3_clk_root usdhc4_clk_root	PFD400	198	198
ECSPI	ecspi_clk_root	480MHz PLL	60	60
I2C	per_clk_root	528MHz PLL	66	38.3

Following reset, each ARM core has access to all peripherals. The ROM code will disable the clocks listed in the following table, except for the boot devices listed in the second column.

Table 7-6. List Of Disabled Clocks

Clock Name	Enabled For Boot Device
CCGR0_APBHDMA	
CCGR1_ECSPI1	ECSPI1
CCGR1_ECSPI2	ECSPI2
CCGR1_ECSPI3	ECSPI3
CCGR1_ECSPI4	ECSPI4
CCGR1_FEC	
CCGR1_EPIT1	
CCGR1_EPIT2	
CCGR2_I2C1_SERIAL	I2C1
CCGR2_I2C2_SERIAL	I2C2
CCGR2_I2C3_SERIAL	I2C3
CCGR3_CSI_CORE	
CCGR3_PXP_AXI	
CCGR3_EPDC_AXI	
CCGR3_LCDIF_AXI	
CCGR3_LCDIF_PIX	
CCGR3_EPDC_PIX	
CCGR3_OPENVGAXICLK	
CCGR4_PWM1	
CCGR4_PWM2	
CCGR4_PWM3	
CCGR4_PWM4	
CCGR5_SDMA	
CCGR5_SPDIF	
CCGR5_SSI1	
CCGR5_SSI2	
CCGR5_SSI3	
CCGR5_UART	
CCGR5_UART_SERIAL	
CCGR6_USBOH3	USB
CCGR6_USDHC1	USDHC1
CCGR6_USDHC2	USDHC2
CCGR6_USDHC3	USDHC3
CCGR6_USDHC4	USDHC4
CCGR6_EMI_SLOW	NOR, OneNAND

7.4.4 Enabling MMU and Caches

The boot ROM includes a feature of enabling the Memory Management Unit (MMU) and caches to improve boot speed when performing a secure boot with SEC_CONFIG=Closed (High Assurance Boot (HAB)). L1 instruction cache is enabled at the start of image download. L1 data cache, L2 cache and MMU are enabled during image authentication. Once HAB authentication completes the ROM disables the L2 cache and MMU.

L1 Instruction cache is controlled by L1 I-Cache eFuse. By default the L1 I-Cache is enabled.

The MMU feature is controlled by the BT_MMU_DISABLE eFUSE. By default the BT_MMU_DISABLE eFUSE is not blown meaning the ROM uses MMU, L1 Data and L2 caches of the ARM core. ROM establishes a page table in the MMU Page Table area of OCRAM. The ROM also configures the L1 and L2 cache as write through. This improves the performance of the HAB signature verification software.

Enabling the MMU when booting non-securely with SEC_CONFIG=Open, and setting the CSF pointer in the Image Vector Table to NULL, has no impact on the boot performance. With this configuration it is recommended to blow BT_MMU_DISABLE fuse.

7.4.5 Exception Handling

The exception vectors located at the start of ROM are used to map all the ARM exceptions (except the reset exception) to a duplicate exception vector table in internal RAM.

During the boot phase of CPU0, the RAM vectors point to the serial downloader in ROM.

During the boot phase of a secondary CPU, the internal RAM vectors point to a function that sets the error status registers (see Persistent Bits), sends a wakeup error interrupt and performs the Wait For Interrupt instruction. The interrupt service routine of primary CPU must reconfigure the system and reset the secondary CPU.

After boot the program image can overwrite the vectors as required. The code shown below is used to map the ROM exception vector table to the duplicate one in RAM.

Mapping ROM Exception Vector Table

```
;; Define linker area for ROM exception vector table
AREA IROM_VECTORS, CODE, READONLY
LDR     PC, Reset_Addr
LDR     PC, Undefined_Addr
LDR     PC, SWI_Addr
```

```
LDR     PC, Prefetch_Addr
LDR     PC, Abort_Addr
NOP                         ; Reserved vector
LDR     PC, IRQ_Addr
LDR     PC, FIQ_Addr
LDR     PC, SW_monitor_addr
;; Define exception vector table
Reset_Addr      DCD     start_address
Undefined_Addr  DCD     iRAM_Undefined_Handler
SWI_Addr        DCD     iRAM_SWI_Handler
Prefetch_Addr   DCD     iRAM_Prefetch_Handler
Abort_Addr      DCD     iRAM_Abort_Handler
                DCD     0               ; Reserved vector
IRQ_Addr        DCD     iRAM_IRQ_Handler
FIQ_Addr        DCD     iRAM_FIQ_Handler
SW_monitor_add  DCD                 SW monitor exception
start_address DCD start ;reset handler vector
```

7.4.6 Interrupt Handling During Boot

No special interrupt handling routines are required during the boot process. Interrupts are disabled during boot ROM execution and may be enabled in a later boot stage.

7.4.7 Persistent Bits

Some modes of boot ROM require registers that keep their values after warm reset. SRC General Purpose registers are used for this purpose.

See the table below for persistent bits list and description.

Table 7-7. Persistent Bits

Bit Name	Bit Location	Description
PERSIST_SECONDARY_BOOT	SRC_GPR10[30]	This bit identifies which image must be used - primary and secondary. Used only for boot modes that support redundant boot.
PERSISTENT_ENTRY0[31:0]	SRC_GPR1[31:0]	Holds entry function for CPU0 for waking-up from low power mode.
PERSISTENT_ARG0[31:0]	SRC_GPR2[31:0]	Holds argument of entry function for CPU0 for waking-up from low power mode.

7.5 Boot Devices (Internal Boot)

The Chip supports the following boot Flash devices:

- NOR Flash with External Interface Module (EIM), located on CS0, 16-bits and 32-bit bus width
- OneNAND Flash with EIM interface, located on CS0, 16-bits bus width

- SD/MMC/eSD/SDXC/eMMC4.4 via USDHC interface, supporting high capacity cards
- EEPROM boot via SPI (serial flash) and I2C (via ECSPI and I2C blocks respectively)

The selection of external boot device type is controlled by BOOT_CFG1[7:4] eFUSEs. See the table below for more details.

Table 7-8. Boot Device Selection

BOOT_CFG1[7:4]	Boot Device
0000	NOR/OneNAND (EIM)
0001	Reserved
0011	Serial ROM (I2C/SPI)
010x	SD/eSD/SDXC
011x	MMC/eMMC

7.5.1 NOR Flash/OneNAND using EIM Interface

The External Interface Module (EIM) works in the asynchronous mode, and supports either muxed, Address/Data, or non-muxed schemes based on fuse settings.

Table 7-9. EIM Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG1[7:4]	OEM	Boot Device Selection	Yes	0000	0000 - Boot from EIM Interface
BOOT_CFG1[3]	OEM	NOR/OneNAND Selection	Yes	0	0 - NOR 1 - OneNAND
BOOT_CFG2[7:6]	OEM	Muxing Scheme	Yes	00	00 - Muxed, 16-bit data (low half) interface 01 - Reserved 10 - Not muxed, 16-bit data (low half) interface 11 - Reserved
BOOT_CFG2[5:4]	OEM	OneNAND Page Size	Yes	00	00 - 1K 01 - 2K 10 - 4K 11 - Reserved

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 1 for corresponding GPIO pin.

7.5.1.1 NOR Flash Boot Operation

Booting from the NOR Flash is supported via EIM interface. The ROM reads Image Vector Table and Boot Data structures to determine if the image can be executed directly from EIM address space or should be copied to other memory.

The start field of Boot Data Structure specifies the final location of the image (see Image Vector Table and Boot Data).

7.5.1.2 OneNAND Flash Boot Operation

At system power-up, the OneNAND device automatically copies an Initial Load Region of 1 Kbyte from the start of the flash array (sector 0 and sector 1, page 0, block 0) to its Boot RAM (OneNAND's internal RAM).

NOTE

The OneNAND boot RAM memory containing the Initial 1K Load Region must contain the IVT, DCD and the Boot Data structures.

Next, the ROM processes the DCD and then proceeds to copy the Program Image contents to the application destination pointer (located in the start entry of Boot Data (see Image Vector Table and Boot Data). The ROM determines the size of the Program Image by the length specified by size entry in Boot Data structure (see Image Vector Table and Boot Data). A failure loading data from the OneNAND device for any reason forces the Chip to enter the Serial Downloader, otherwise the booting from the OneNAND device continues.

The figure below illustrates the layout of the Program Image on a OneNAND boot device.

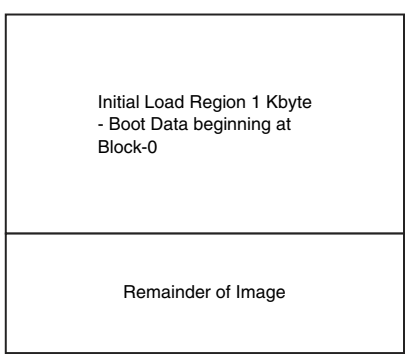


Figure 7-4. Program Image Layout on a OneNAND Flash Device

Prior to accessing the OneNAND device, the Chip waits approximately 500 µs after Power On Reset. This delay is required for the OneNAND device to become ready. After this initial 500 µs delay it can take an addition 70 µs for the OneNAND device to load the Initial Load Region of 1 Kbyte into its boot RAM. The Chip polls the OneNAND device Interrupt Status Register to confirm that the first 1 Kbytes has been loaded to the OneNAND boot RAM before continuing with the boot flow.

7.5.1.3 IOMUX Configuration for EIM Devices

The EIM interface uses dedicated contacts on the IC.

The contacts assigned to the data signals used by EIM are shown in the table below.

Table 7-10. EIM IOMUX Pin Configuration

Signal	Muxed	Not muxed
DATA0	KEY_COL0.alt3	LCD_DAT6.alt3
DATA1	KEY_ROW0.alt3	LCD_DAT7.alt3
DATA2	KEY_COL1.alt3	LCD_DAT8.alt3
DATA3	KEY_ROW1.alt3	LCD_DAT9.alt3
DATA4	KEY_COL2.alt3	LCD_DAT10.alt3
DATA5	KEY_ROW2.alt3	LCD_DAT11.alt3
DATA6	KEY_COL3.alt3	LCD_DAT12.alt3
DATA7	KEY_ROW3.alt3	LCD_DAT13.alt3
DATA8	KEY_COL4.alt3	LCD_DAT14.alt3
DATA9	KEY_ROW4.alt3	LCD_DAT15.alt3
DATA10	KEY_COL5.alt3	LCD_DAT16.alt3
DATA11	KEY_ROW5.alt3	LCD_DAT17.alt3
DATA12	KEY_COL6.alt3	LCD_DAT18.alt3
DATA13	KEY_ROW6.alt3	LCD_DAT19.alt3
DATA14	KEY_COL7.alt3	LCD_DAT20.alt3
DATA15	KEY_ROW7.alt3	LCD_DAT21.alt3
ADDR0	KEY_COL0.alt3	
ADDR1	KEY_ROW0.alt3	
ADDR2	KEY_COL1.alt3	
ADDR3	KEY_ROW1.alt3	
ADDR4	KEY_COL2.alt3	
ADDR5	KEY_ROW2.alt3	
ADDR6	KEY_COL3.alt3	
ADDR7	KEY_ROW3.alt3	
ADDR8	KEY_COL4.alt3	
ADDR9	KEY_ROW4.alt3	

Table continues on the next page...

Table 7-10. EIM IOMUX Pin Configuration (continued)

Signal	Muxed	Not muxed
ADDR10	KEY_COL5.alt3	
ADDR11	KEY_ROW5.alt3	
ADDR12	KEY_COL6.alt3	
ADDR13	KEY_ROW6.alt3	
ADDR14	KEY_COL7.alt3	
ADDR15	KEY_ROW7.alt3	
ADDR16	EPDC_D8.alt3	
ADDR17	EPDC_D9.alt3	
ADDR18	EPDC_D10.alt3	
ADDR19	EPDC_D11.alt3	
ADDR20	EPDC_D12.alt3	
ADDR21	EPDC_D13.alt3	
ADDR22	EPDC_D14.alt3	
ADDR23	EPDC_D15.alt3	
ADDR24	EPDC_VCOM0.alt3	
ADDR25	EPDC_VCOM1.alt3	
ADDR26	EPDC_BDR0.alt3	
CS0	EPDC_PWRCTRL2.alt3	
OE	EPDC_PWRCTRL1.alt3	
RW	EPDC_PWRCTRL0.alt3	
LBA	EPDC_SDCE1.alt3	

7.5.2 Expansion Device

The ROM supports booting from MMC/eMMC and SD/eSD compliant devices.

7.5.2.1 Expansion Device eFUSE Configuration

SD/MMC/eSD/eMMC/SDXC boot can be performed using either USDHC-1, USDHC-2, USDHC-3, or USDHC-4 ports, based on setting of the BOOT_CFG2[4:3] (Port Select) fuse or it's associated GPIO input value at boot. All USDHC ports support eMMC4.3 and eMMC4.4 fast boot.

See the table below for details.

Table 7-11. USDHC Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG1[7:6]	OEM	Boot Device Selection	Yes	00	01 - Boot from USDHC Interfaces
BOOT_CFG1[5]	OEM	SD/MMC Selection	Yes	0	0 - SD/eSD/SDXC 1 - MMC/eMMC
BOOT_CFG1[4]	OEM	Fast Boot Support	Yes	0	0 - Normal Boot 1 - Fast Boot
BOOT_CFG1[3:2]	OEM	SD/MMC Speed Mode	Yes	00	MMC 0x - High Speed Mode 1x - Normal Speed Mode SD 0x - High/Normal 10 - SDR50 11 - SDR104
BOOT_CFG1[1]	OEM	SD Power Cycle Enable/ eMMC Reset Enable	Yes	0	MMC 0 - eMMC reset disabled 1 - eMMC reset enabled via SD_RST pad (on USDHC3 and USDHC4 only) SD 0 - No power cycle 1 - Power cycle enabled via SD_RST pad (on USDHC3 and USDHC4 only)
BOOT_CFG1[0]	OEM	SD Loopback Clock Source Sel(for SDR50 and SDR104 only)	Yes	00	0 - through SD pad 1 - direct

Table continues on the next page...

Table 7-11. USDHC Boot eFUSE Descriptions (continued)

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG2[7:5]	OEM	Bus Width/SD Calibration Step	Yes	000	SD/eSD/SDXC (BOOT_CFG1[5]=0) Bus Width xx0 - 1-bit xx1 - 4-bit SD Calibration Step 00x - 1 delay cells 01x - 1 delay cells 10x - 2 delay cells 11x - 3 delay cells MMC/eMMC (BOOT_CFG1[5]=1) 000 - 1-bit 001 - 4-bit 010 - 8-bit 101 - 4-bit DDR (MMC 4.4) 110 - 8-bit DDR (MMC 4.4) Else - reserved.
BOOT_CFG2[4:3]	OEM	Port Select	Yes	00	00 - USDHC-1 01 - USDHC-2 10 - USDHC-3 11 - USDHC-4
BOOT_CFG2[2]	OEM	DLL Override (eMMC Only)	Yes	0	0 - Boot ROM default. 1 - Apply value per fuse field MMC_DLL_DLY[6:0]
BOOT_CFG2[1]	OEM	Boot Acknowledge Disable / Pull-Down During Power Cycle Enable	Yes	0	MMC 0 - Boot Acknowledge Enabled. 1 - Boot Acknowledge Disabled.
BOOT_CFG2[0]	OEM	Override Pad Settings	Yes	0	0 - Use default values 1 - Use PAD_SETTINGS values
MMC_DLL_DLY[6:0]	OEM	MMC DLL Value / UHSI Calibaration Start Value	No	0000000	MMC DLL Value / UHSI Calibaration Start Value

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 1 for corresponding GPIO pin.

Boot code supports following standards.

- MMCv4.4 or less
- eMMCv4.4 or less
- SDv2.0 or less

- eSDv2.10 rev-0.9, with or without FAST_BOOT.
- SDXCv3.0

MMC/SD/eSD/SDXC/eMMC can be connected to any of USDHC-1,2,3,4 blocks and can be booted by copying 4Kbyte of data from MMC/SD/eSD/eMMC device to internal RAM. After checking the Image Vector Table header value (0xD1) from Program Image, the ROM code performs a DCD check. After successful DCD extraction, the ROM code extracts from Boot Data Structure the destination pointer and length of image to be copied to RAM device from where code execution occurs.

The maximum image size to load in SD/MMC boot is 32MB. This is due to the limited number of uSDHC ADMA Buffer Descriptors allocated by ROM.

NOTE

The Initial 4Kbyte of Program Image must contain the IVT, DCD and the Boot Data structures.

Table 7-12. SD/MMC Frequencies

	SD	MMC	MMC (DDR Mode)
Identification (KHz)	347.22		
Normal Speed Mode (MHz)	25	20	25
High Speed Mode (MHz)	50	40	50
UHSI SDR50 (MHz)	100		
UHSI SDR104 (MHz)	200		

NOTE

The boot ROM code reads application image length and application destination pointer from image.

7.5.2.2 MMC and eMMC Boot

The following table provides MMC and eMMC boot details.

Table 7-13. MMC and eMMC Boot Details

Normal Boot Mode	During initialization (normal boot mode) the MMC frequency is set to 347.22 KHz. When the MMC card enters the identification portion of the initialization, voltage validation is performed and the ROM boot code checks high voltage settings and card capacity. The ROM boot code supports both high capacity and low capacity MMC/eMMC cards. After initialization phase is complete, the ROM boot code switches to a higher frequency (20 MHz in Normal boot mode or 40MHz in High Speed mode). eMMC is also interfaced via USDHC and follows the same flow as MMC. The boot partition can be selected for an MMC4.x card after the card initialization is complete. The ROM code reads the BOOT_PARTITION_ENABLE field in the Ext_CSD[179] to get the boot partition to be set. If there is no boot partition mentioned in BOOT_PARTITION_ENABLE field or the user partition has been mentioned, ROM boots from the user partition.
eMMC4.3 or eMMC4.4 Device Supporting Special Boot Mode	If using an eMMC4.3 or eMMC4.4 device supporting special boot mode, it can be initiated by pulling the CMD line low. If BOOT ACK is enabled, the eMMC4.3/eMMC4.4 device sends the BOOT ACK via DATA lines and ROM can read the BOOT ACK [S010E] to identify the eMMC4.3/eMMC4.4 device. eMMC4.3/eMMC4.4 device with "Boot mode" feature can only be supported via ESDHCV3-3 and with or without BOOT ACK. If BOOT ACK is enabled ROM waits 50 ms to get the BOOT ACK and if BOOT ACK is received by ROM. If BOOT ACK is disabled ROM waits 1 second for data. If BOOT ACK or data was received then eMMC4.3/eMMC4.4 is booted in "Boot mode", otherwise eMMC4.3/eMMC4.4 boots as a normal MMC card from the selected boot partition. This boot mode can be selected by BOOT_CFG1[4] (Fast Boot) fuse. BOOT ACK is selected by BOOT_CFG2[1].
eMMC4.4 Device	If using eMMC4.4 device, Double Data Rate (DDR) mode can be used. This mode can be selected by BOOT_CFG2[7:5] (Bus Width) fuse.

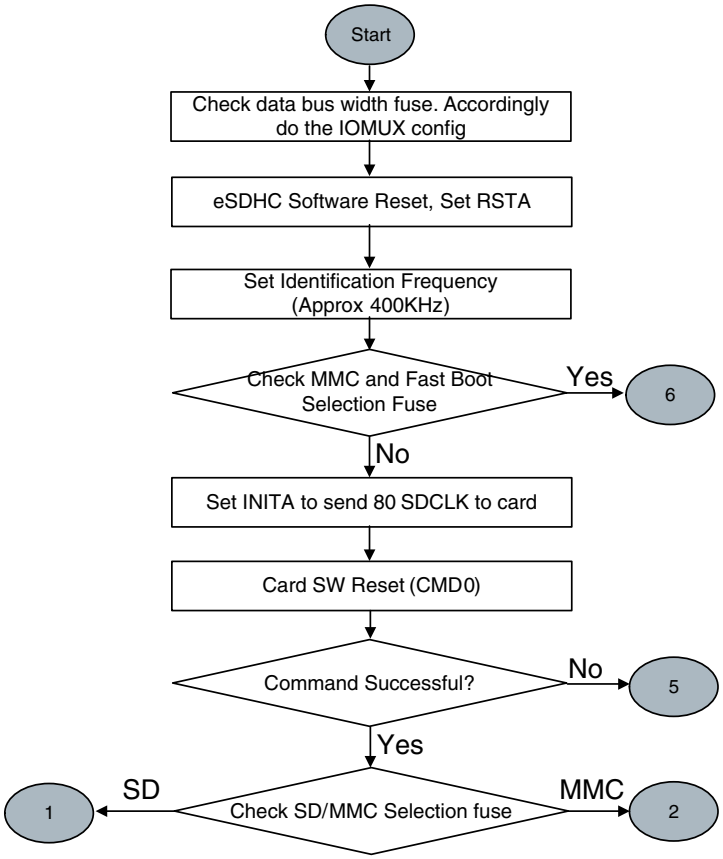


Figure 7-5. Expansion Device Boot Flow (1 of 6)

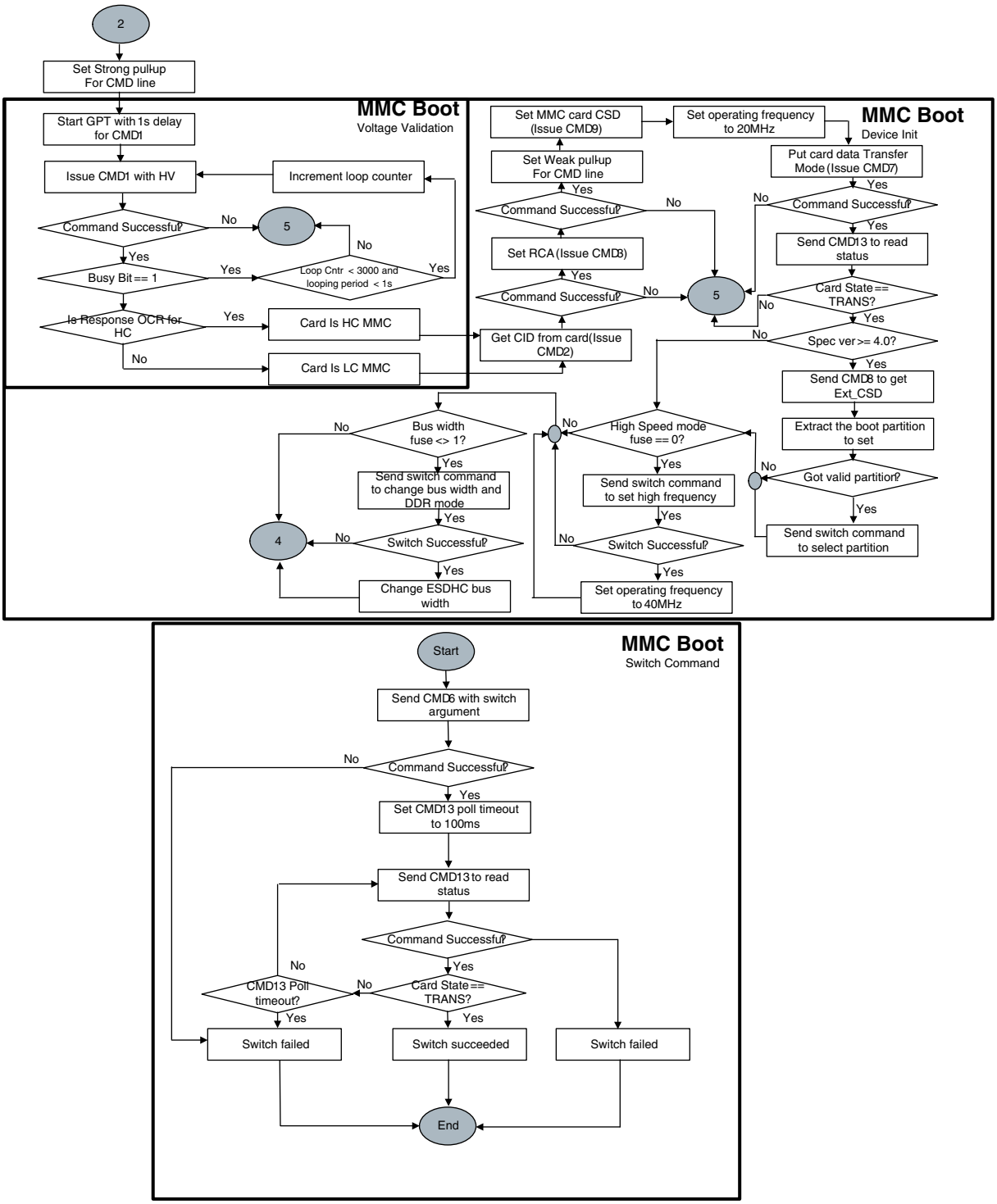


Figure 7-6. Expansion Device (MMC) Boot Flow (2 of 6)

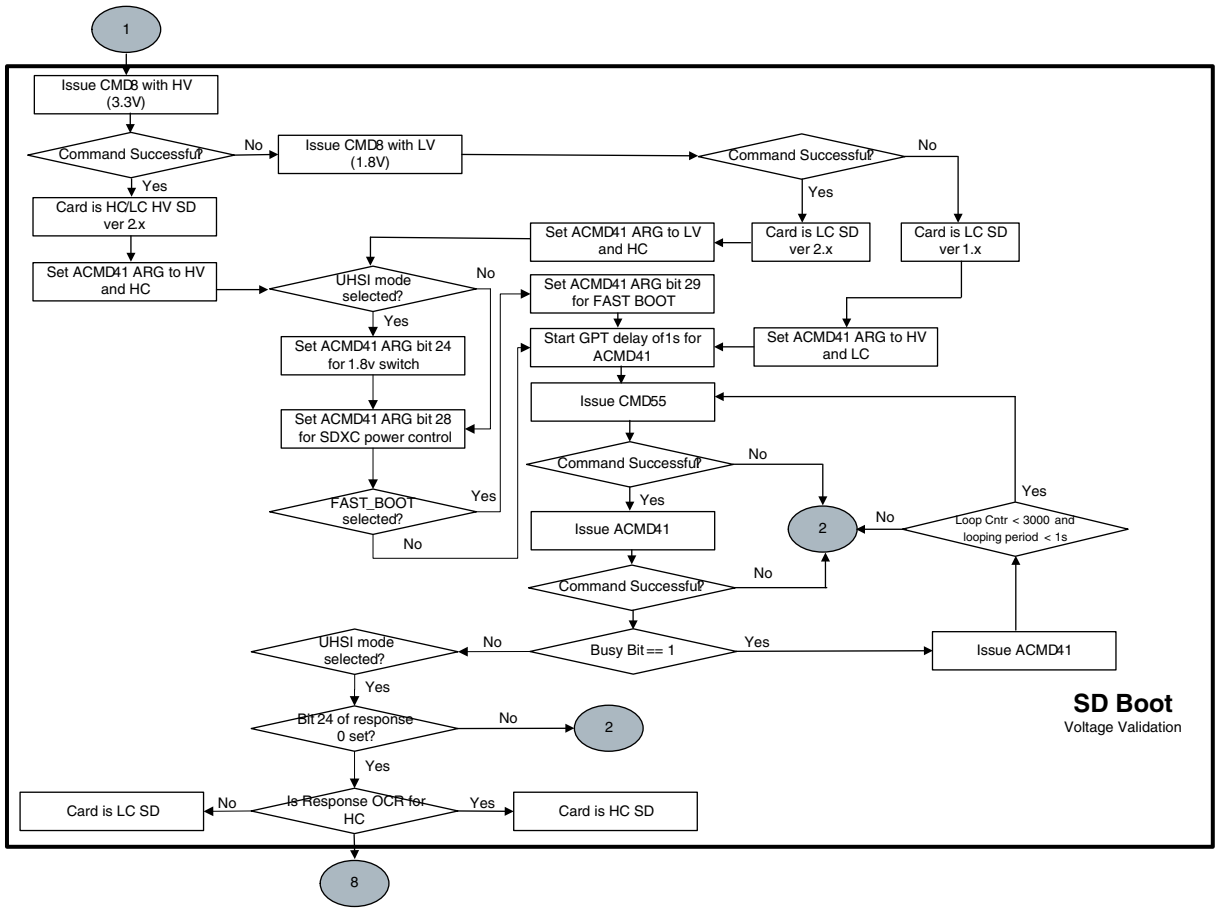


Figure 7-7. Expansion Device (SD/eSD/SDXC) Boot Flow (3 of 6) Part 1

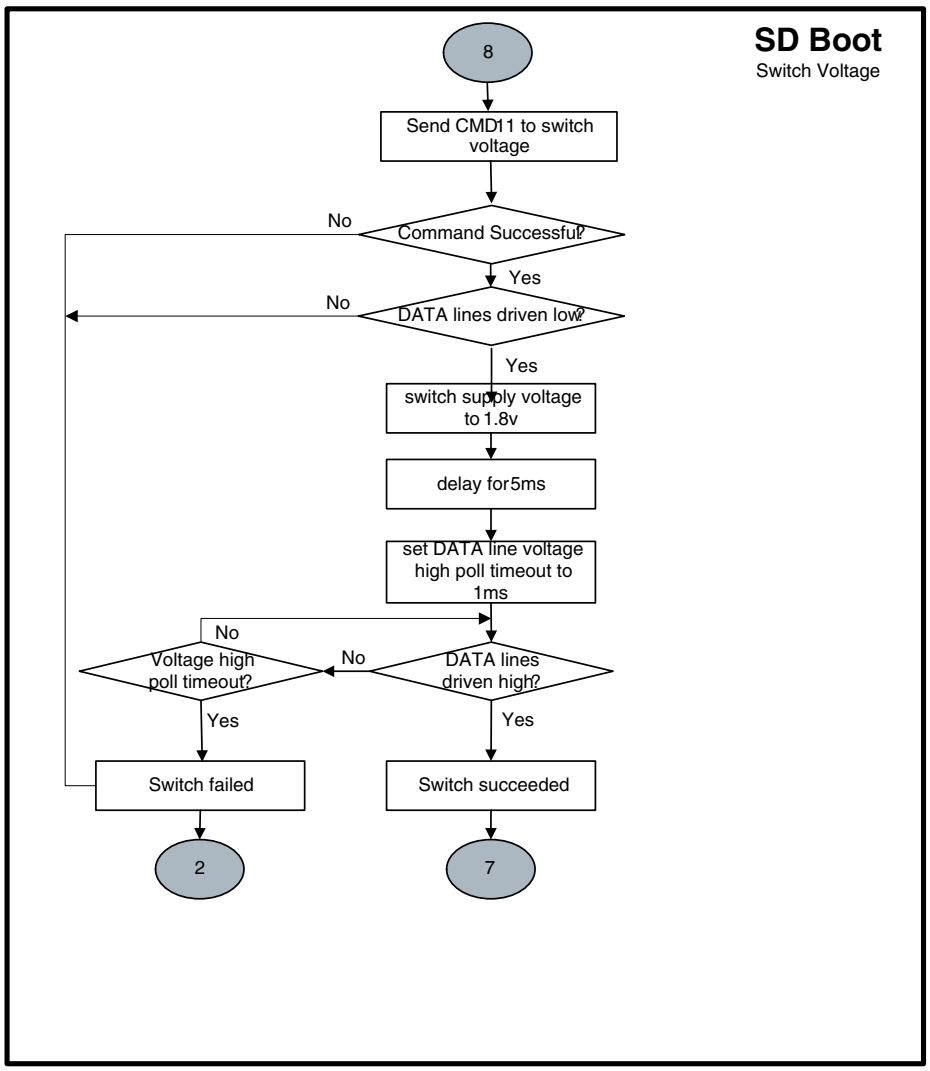
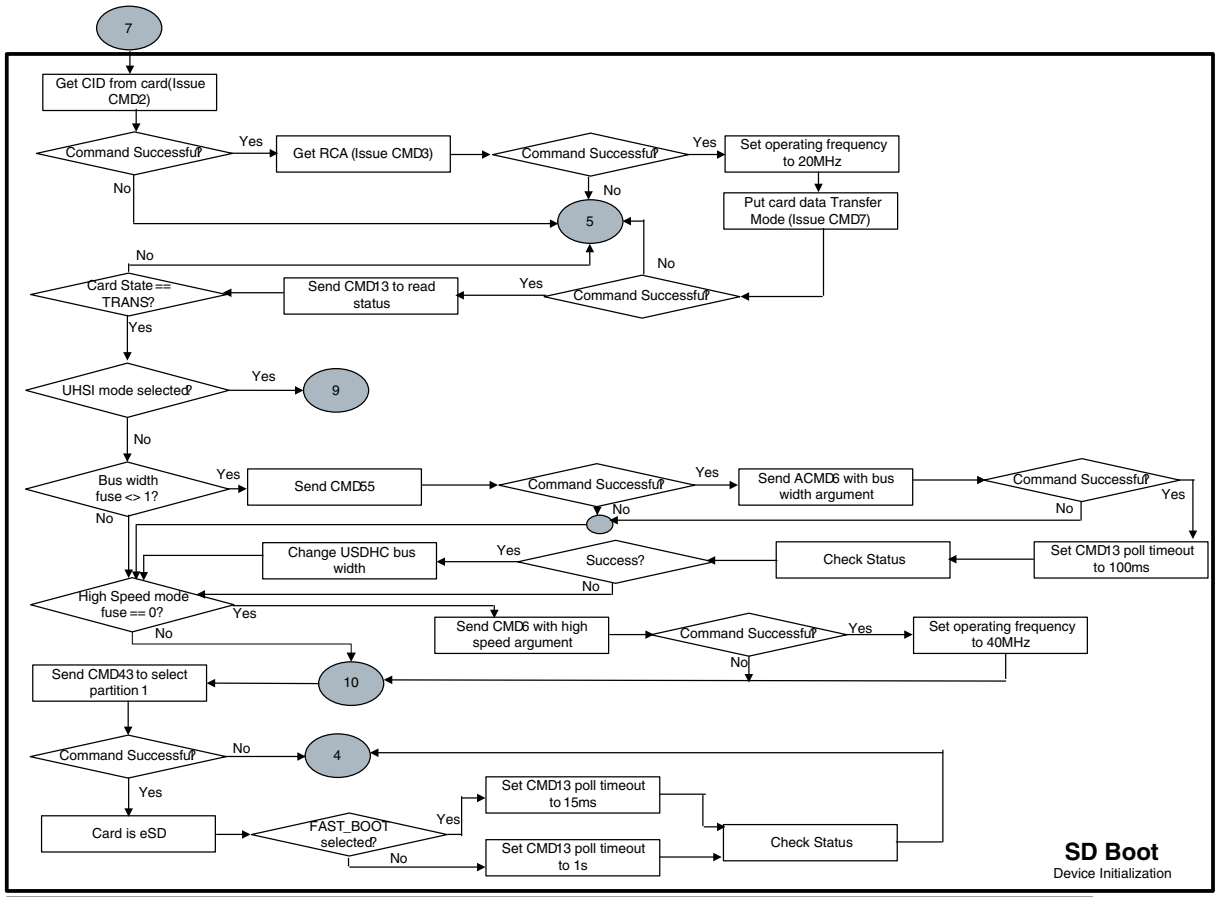


Figure 7-8. Expansion Device (SD/eSD/SDXC) Boot Flow (3 of 6) Part 2



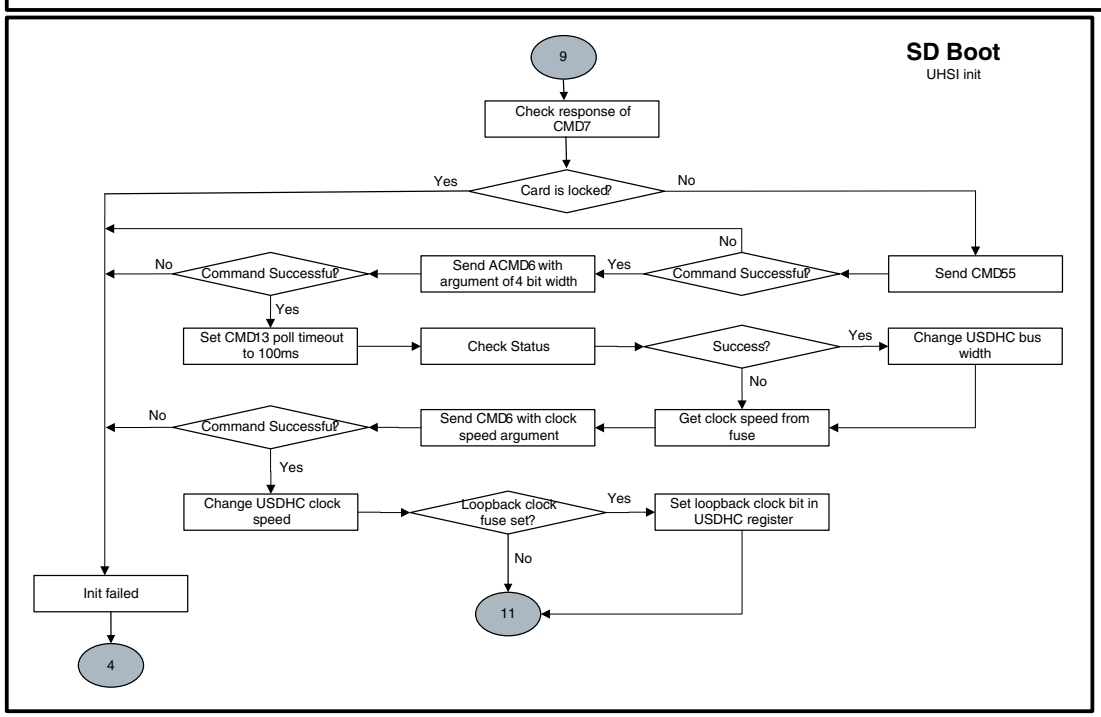


Figure 7-9. Expansion Device (MMCSD/eSD/SDXC) Boot Flow (4 of 6)

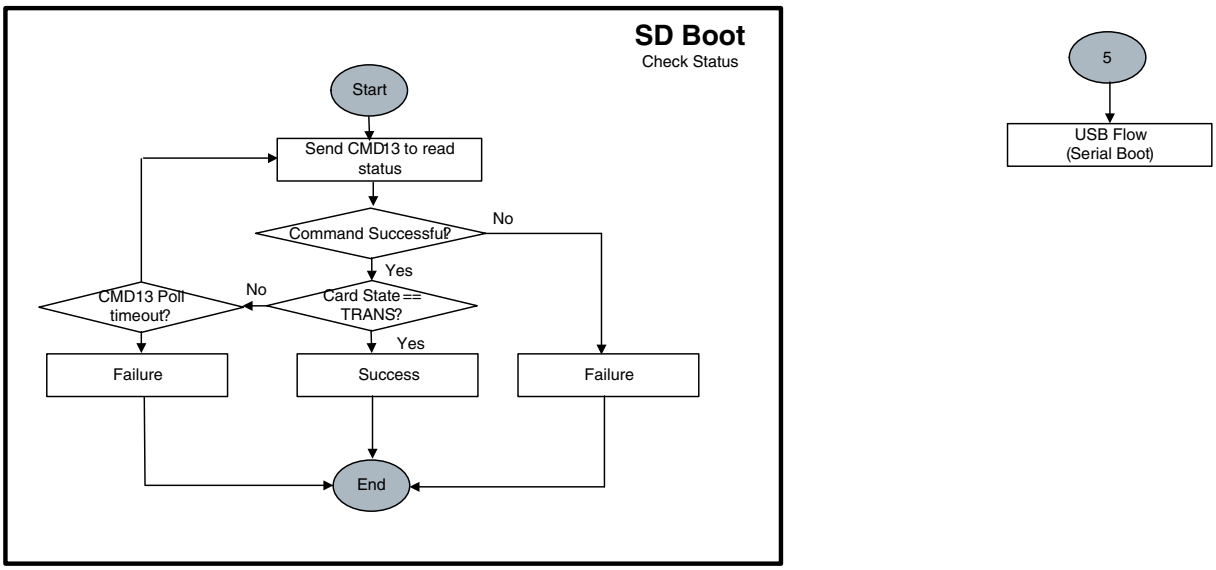
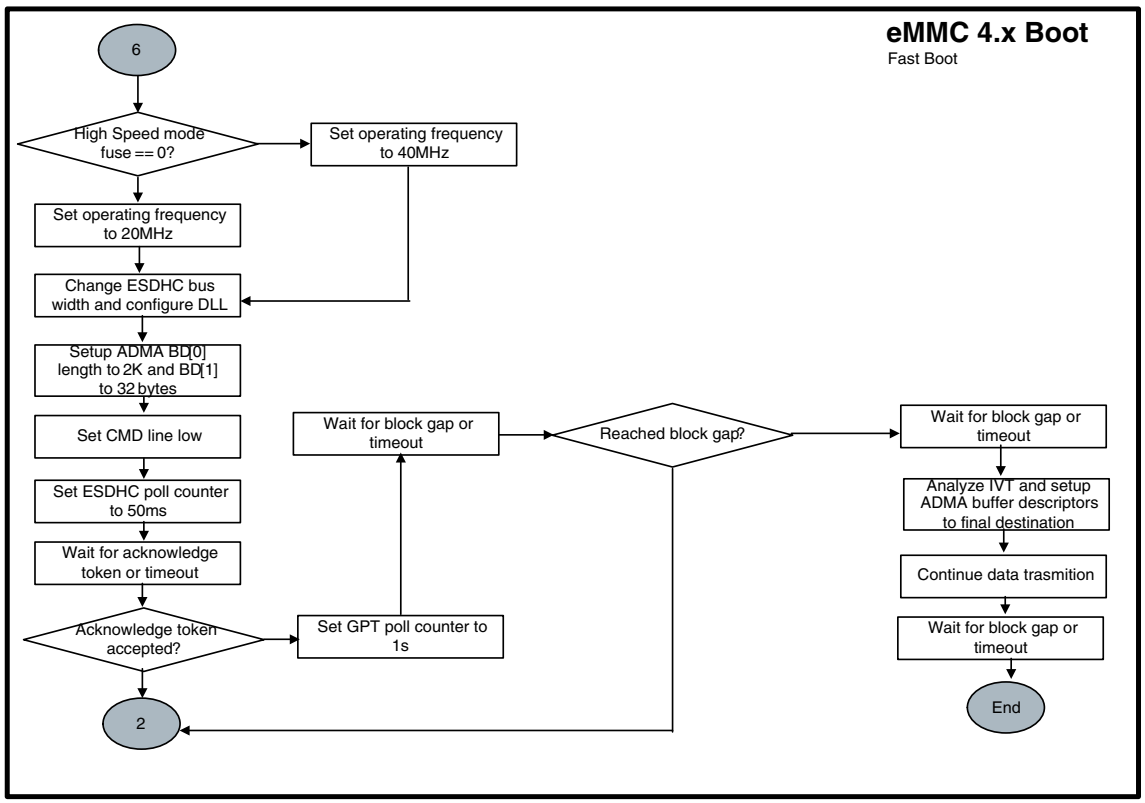


Figure 7-10. Expansion Device (SD/eSD) Boot Flow (5 of 6)



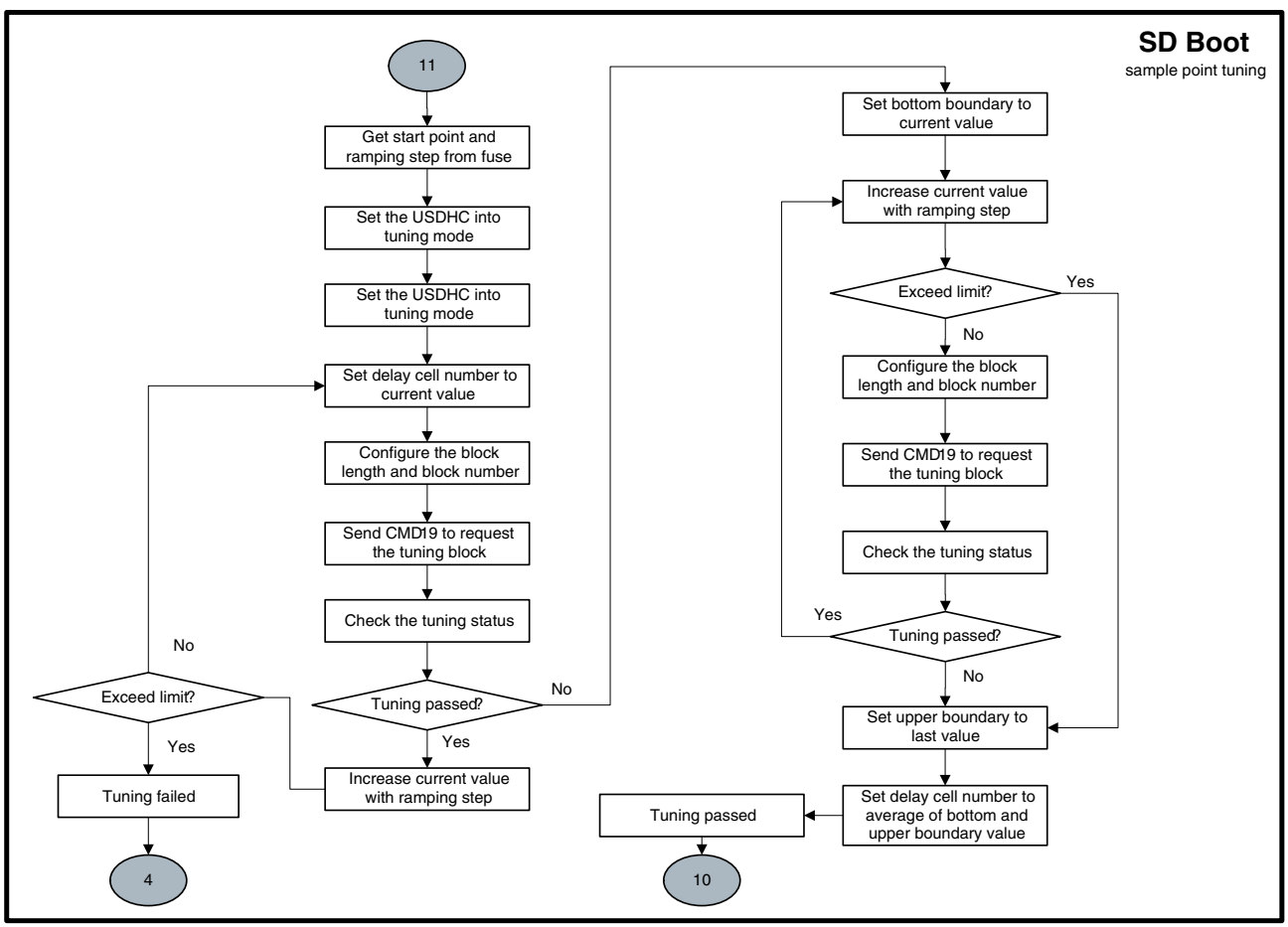


Figure 7-11. Expansion Device Boot Flow (6 of 6)

7.5.2.3 SD, eSD and SDXC

After the normal boot mode initialization begins, the SD/eSD/SDXC frequency is set to 347.22 kHz. During the identification phase, SD/eSD/SDXC card voltage validation is performed. During voltage validation, boot code first checks with high voltage settings; if that fails, it checks with low voltage settings.

The capacity of the card is also checked. Boot code supports high capacity and low capacity SD/eSD/SDXC cards after voltage validation card initialization is done.

During card initialization, the ROM boot code attempts to set the boot partition for all SD, eSD, and SDXC devices. If this fails, the boot code assumes the card is a normal SD card or SDXC card. If it does not fail, the boot code assumes it is an eSD card. After the initialization phase is over, boot code switches to a higher frequency (25 MHz in Normal Speed mode or 50 MHz in High Speed Mode). ROM also supports FAST_BOOT mode booting from eSD card. This mode can be selected by BOOT_CFG1[4] (Fast Boot) fuse described in Table 7-11.

For UHSI cards, clock speed fuses can be set to SDR50 or SDR104 on USDHC3 and USDHC4 ports. This will enable the voltage switch process to set the signaling voltage to 1.8V during voltage validation. The bus width is fixed at 4 bits wide and a sampling point tuning process is needed to calibrate the number of delay cells. If SD Loopback Clock eFuse is set, the feedback clock will come directly from the loopback SD clock, instead of the card clock (by default). The SD clock speed can be selected by BOOT_CFG1[3:2], and the SD Loopback Clock is selected by BOOT_CFG1[0].

UHSI calibration start value (MMC_DLL_DLY[6:0]) and step value (BOOT_CFG2[7:5]) can be set to optimize the sample point tuning process.

If SD Power Cycle Enable eFuse is 1, ROM will set SD_RST pad low, wait 5ms and then set SD_RST pad high. If SD_RST pad is connected to SD power supply enable logic on board, it enables power cycle of SD card. This may be crucial in case when SD logic is in 1.8V states and must be reset to 3.3V states.

7.5.2.4 IOMUX Configuration for SD/MMC

Table 7-14. SD/MMC IOMUX Pin Configuration

Signal	USDHC-1	USDHC-2	USDHC-3	USDHC-4
CLK	SD1_CLK.alt0	SD2_CLK.alt0	SD3_CLK.alt0	FEC_MDIO.alt1
CMD	SD1_CMD.alt0	SD2_CMD.alt0	SD3_CMD.alt0	FEC_TX_CLK.alt1
DAT0	SD1_DAT0.alt0	SD2_DAT0.alt0	SD3_DAT0.alt0	FEC_RX_ER.alt1

Table continues on the next page...

Table 7-14. SD/MMC IOMUX Pin Configuration (continued)

Signal	USDHC-1	USDHC-2	USDHC-3	USDHC-4
DAT1	SD1_DAT1.alt0	SD2_DAT1.alt0	SD3_DAT1.alt0	FEC_CRS_DV.alt1
DAT2	SD1_DAT2.alt0	SD2_DAT2.alt0	SD3_DAT2.alt0	FEC_RXD1.alt1
DAT3	SD1_DAT3.alt0	SD2_DAT3.alt0	SD3_DAT3.alt0	FEC_TXD0.alt1
DAT4	SD1_DAT4.alt0	SD2_DAT4.alt0	KEY_COL1.alt4	FEC_MDC.alt1
DAT5	SD1_DAT5.alt0	SD2_DAT5.alt0	KEY_ROW1.alt4	FEC_RXD0.alt1
DAT6	SD1_DAT6.alt0	SD2_DAT6.alt0	KEY_COL2.alt4	FEC_TX_EN.alt1
DAT7	SD1_DAT7.alt0	SD2_DAT7.alt0	KEY_ROW2.alt4	FEC_TXD1.alt1
VSELECT	ECSPI2_MOSI.alt4	ECSPI1_MOSI.alt4	KEY_ROW6.alt6	EPDC_PWCTRL1.alt6
RESET [1]	KEY_COL3.alt5	SD2_RESET.alt0	KEY_COL6.alt5	FEC_REFOUT.alt1
CD	KEY_ROW7.alt6			

1. Active low

7.5.2.5 Redundant Boot Support for Expansion Device

ROM supports redundant boot for expansion device. Primary or Secondary image is selected depending on PERSIST_SECONDARY_BOOT setting (see Table 7-7).

If PERSIST_SECONDARY_BOOT is 0, the boot ROM uses address 0x0 for primary image.

If PERSIST_SECONDARY_BOOT is 1, the boot ROM will read secondary image table from address 0x200 on boot media and will use address specified in the table.

Table 7-15. Secondary Image Table Format

Reserved (chipNum)
Reserved (driveType)
tag
firstSectorNumber
Reserved (sectorCount)

Where:

- tag: used as indication of valid secondary image table. Must be 0x00112233.
- firstSectorNumber is the first 512B sector number of the secondary image.

For secondary image support, the primary image must reserve space for secondary image table. See the figure below for typical structures layout on expansion device.

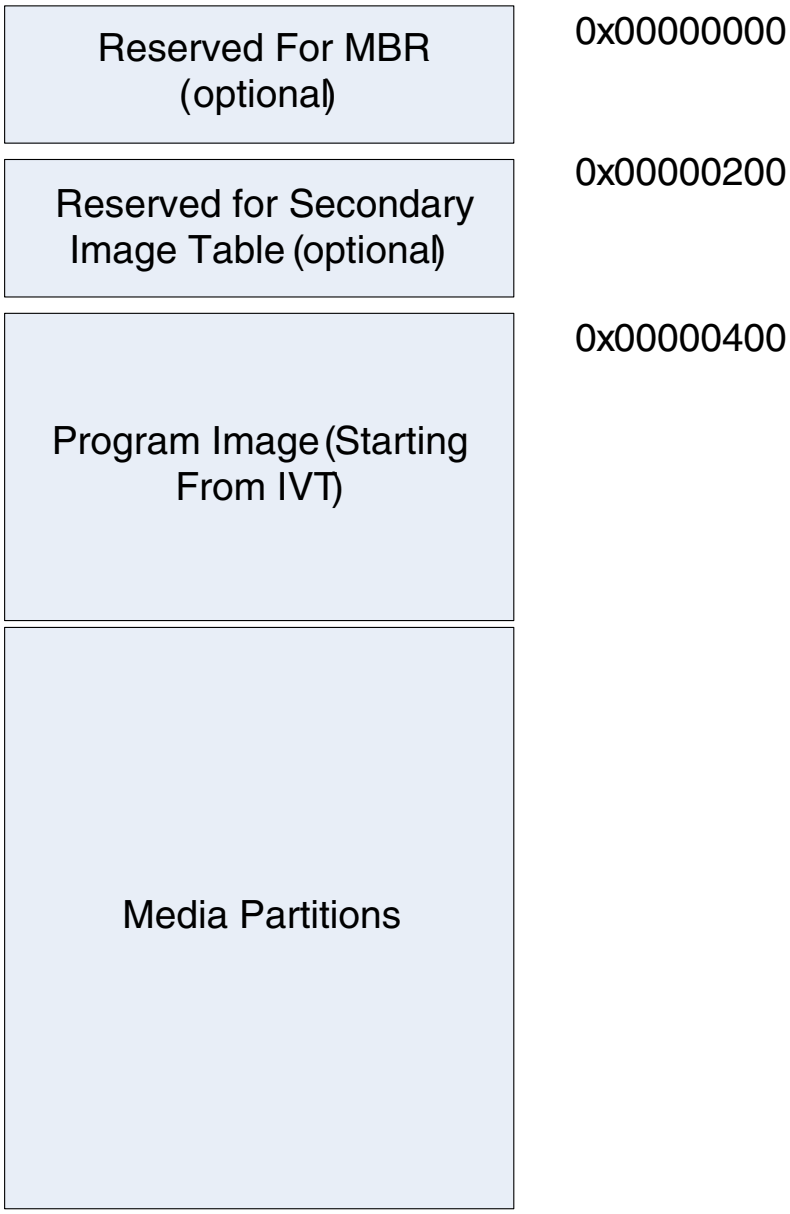


Figure 7-12. Expansion Device Structures Layout

For Closed mode, if there are failures during primary image authentication, the boot ROM will turn on PERSIST_SECONDARY_BOOT bit (see Table 7-7) and perform software reset. (After software reset, secondary image will be used.)

7.5.3 Serial ROM through SPI and I2C

The chip supports boot from serial memory devices, such as EEPROM and Serial Flash using the SPI.

The following ports are available for serial boot: ECSPI (ECSPI-1, ECSPI-2, ECSPI-3, ECSPI-4, ECSPI-5), and I2C Controller (I2C-1, I2C-2 and I2C-3) interfaces.

7.5.3.1 Serial ROM eFUSE Configuration

The boot ROM code determines the type of device using the following parameters, either provided by eFUSE settings or sampled on the I/O pins, during boot.

See the table below for details:

Table 7-16. Serial ROM Boot eFUSE Descriptions

Fuse	Config	Definition	GPIO[1]	Shipped Value	Settings
BOOT_CFG1[7:4]	OEM	Boot Device Selection	Yes	0000	0011 - Boot from Serial ROM
BOOT_CFG4[5:4]	OEM	CS select (SPI only)	Yes	00	00 - CS#0 01 - CS#1 10 - CS#2 11 - CS#3
BOOT_CFG4[3]	OEM	SPI Addressing (SPI only)	Yes	0	0 - 2-bytes (16-bit) 1 - 3-bytes (24-bit)
BOOT_CFG4[2:0]	OEM	Port Select	Yes	00	000 - ECSPI-1 001 - ECSPI-2 010 - ECSPI-3 011 - ECSPI-4 101- I2C-1 110- I2C-2 111- I2C-3

1. Setting can be overridden by GPIO settings when BT_FUSE_SEL fuse is intact. See Table 1 for corresponding GPIO pin.

The ECPSI-1/ECPSI-2/ECPSI-3/ECPSI-4 block can be used as boot device using ECSPI interface for serial ROM boot. The SPI interface is configured to operate at 15MHz for 3-byte addressing device and 3.75MHz for 2-byte addressing devices.

The I2C-1/I2C-2/I2C-3 block can be used as boot device using I2C interface, for serial ROM boot. The I2C interface is configured to operate at 343.75 Kbps.

The boot ROM will copy 4Kbyte of data from Serial ROM device to internal RAM. After checking the Image Vector Table header value (0xD1) from Program Image, the ROM code performs a DCD check. After successful DCD extraction, the ROM code extracts from Boot Data Structure the destination pointer and length of image to be copied to RAM device from where code execution occurs.

NOTE

The Initial 4K of Program Image must contain the IVT, DCD and the Boot Data structures.

7.5.3.2 I2C Boot

The boot flow when booting from an I2C device is shown in Figure 7-13.

The boot ROM code reads the fuses BOOT_CFG1[7:4] (Boot Device Selection) and BOOT_CFG1[7:4] (Port select) to detect EEPROM device type. The ROM program copies 4K data from the EEPROM device to internal RAM. The boot ROM code next copies the initial 4Kbyte of data as well as rest of image directly to application destination extracted from application image.

The chip uses the Device Select Code/Device Address in the table below to boot from an EEPROM.

Table 7-17. EEPROM via I2C Device Select Code

Bits	Device Type Identifier				Chip Enable Address[1]			R/W
	7	6	5	4	3	2	1	0
Device Select Code	1	0	1	0	0	0	0	R/W

1. These address bits, should be configured at the memory device, to match this '000' value.

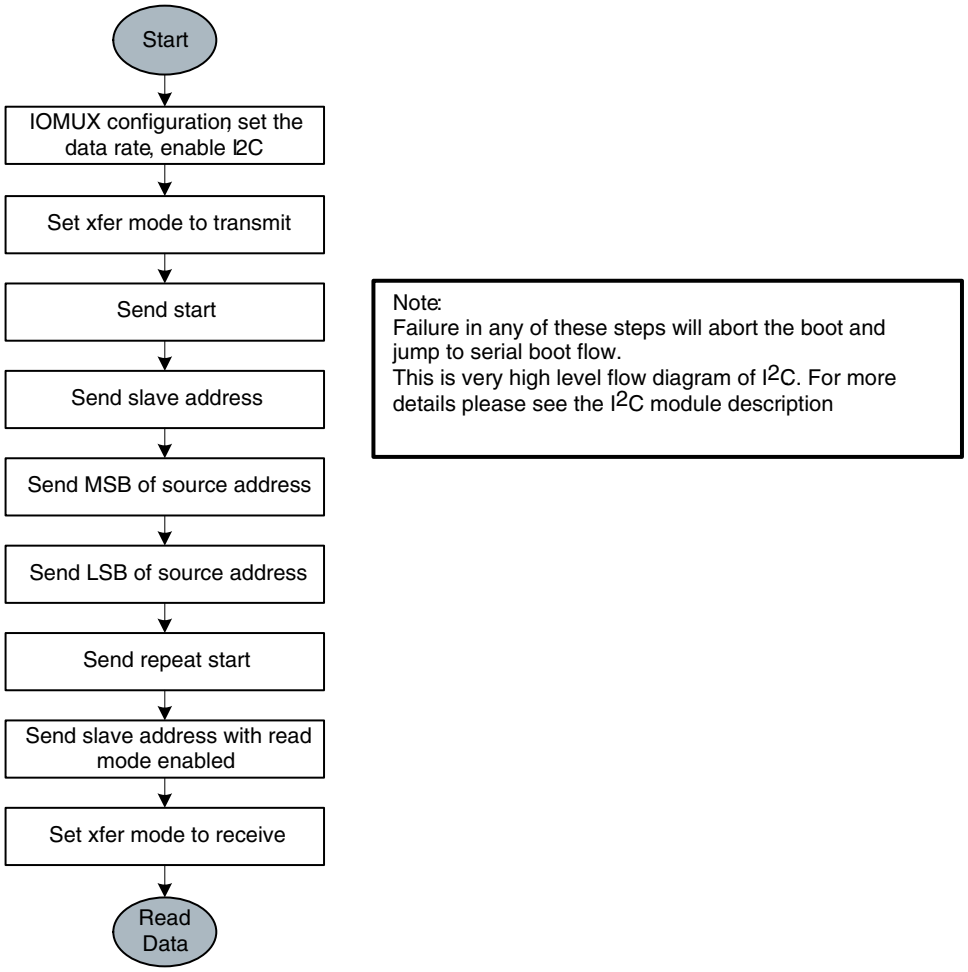


Figure 7-13. I2C Flow Chart

7.5.3.2.1 I2C IOMUX Pin Configuration

The contacts assigned to the signals used by the three I2C blocks is shown in the table below.

Table 7-18. I2C IOMUX Pin Configuration

Signal	I2C-1	I2C-2	I2c-3
SDA	I2C1_SCL.alt0	I2C2_SCL.alt0	AUD_RXFS.alt4
SCL	I2C1_SDA.alt0	I2C2_SDA.alt0	AUD_RXC.alt4

7.5.3.3 ECSPI Boot

The Enhanced Configurable SPI (ECSPI) interface is configured in master mode and the EEPROM device is connected to ECSPI interface as a slave.

The boot ROM code copies 4 KB data from EEPROM device to the internal RAM. If DCD verification is successful, the ROM code copies the initial 4 KB data, as well as the rest of the image extracted from application image, directly to the application destination. The ECSPI can read data from EEPROM using 2 or 3 byte addressing. Its burst length is 32 bytes.

NOTE

The Serial ROM Chip Select Number is determined by BOOT_CFG4[5:4] (Chip Select) fuse.

When using the SPI as boot device, the Chip supports booting from both Serial EEPROM and Serial Flash devices. The boot code determines which device is being used by reading the appropriate eFUSE/I/O values at boot (see Table 7-16 for details).

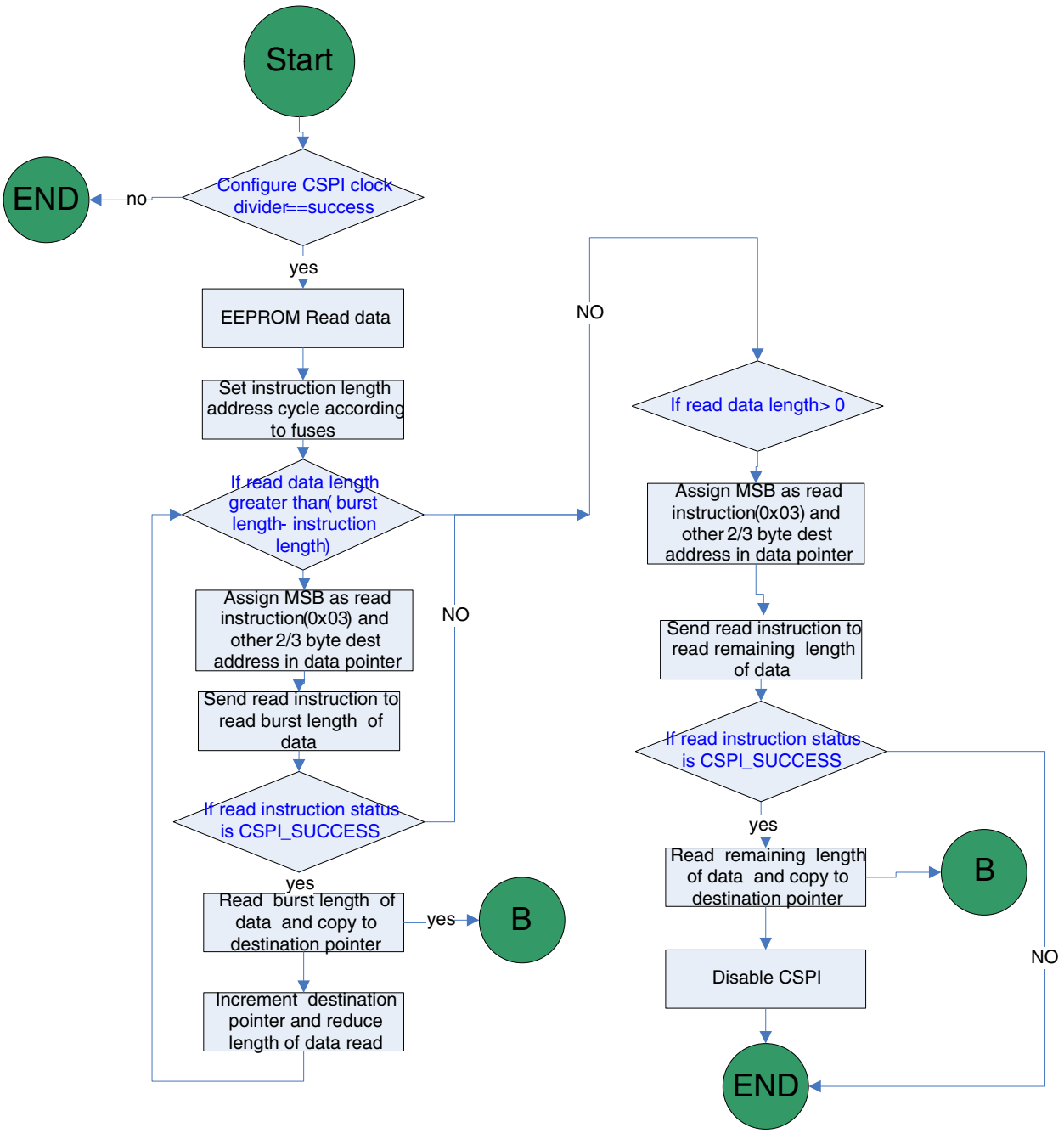


Figure 7-14. CSPI Flow chart

7.5.3.3.1 ECSPI IOMUX Pin Configuration

The contacts assigned to the signals used by the three CSPI blocks is shown in the table below.

Table 7-19. SPI IOMUX Pin Configuration

Signal	ECSPI-1	ECSPI-2	ECSPI-3	ECSPI4	ECSPI-5
MISO	ECSPI1_MISO.alt0	ECSPI2_MISO.alt0	EPDC_D9.alt1	EPDC_D1.alt1	N/A
MOSI	ECSPI1_MOSI.alt0	ECSPI2_MOSI.alt0	EPDC_D8.alt1	EPDC_D0.alt1	N/A
RDY	N/A[1]	N/A	N/A	N/A	N/A
SCLK	ECSPI1_SCLK.alt0	ECSPI2_SCLK.alt0	EPDC_D11.alt1	EPDC_D3.alt1	N/A
SS0	ECSPI1_SS0.alt0	ECSPI2_SS0.alt0	EPDC_D10.alt1	EPDC_D2.alt1	N/A
SS1	I2C1_SCL.alt6	EPDC_SDCE0.alt1	EPDC_D12.alt6	EPDC_D4.alt1	N/A
SS2	I2C1_SDA.alt6	EPDC_GDCLK.alt1	EPDC_D13.alt6	EPDC_D5.alt1	N/A
SS3	ECSPI2_SS0.alt1	EPDC_GDOE.alt1	EPDC_D14.alt6	EPDC_D6.alt1	N/A

1. N/A in the ROM code indicates the pins are not available or not used.

7.6 Program image

This section describes the data structures that are required to be included in a user's program image. A program image consists of:

- Image vector table—A list of pointers located at a fixed address that the ROM examines to determine where other components of the program image are located
- Boot data—A table indicating the program image location, program image size in bytes, and the plugin flag
- Device configuration data—IC configuration data
- Data used by HAB—CSF command data, signatures and certificates (optional for non-secure boot). CSF commands are the instructions HAB uses to determine which keys to install and which memory areas correspond to the digital signatures.
- User code and data

7.6.1 Image Vector Table and Boot Data

The Image Vector Table (IVT) is the data structure that the ROM reads from the boot device supplying the program image containing the required data components to perform a successful boot.

The IVT includes the program image entry point, a pointer to Device Configuration Data (DCD) and other pointers used by the ROM during the boot process.The ROM locates the IVT at a fixed address that is determined by the boot device connected to the Chip. The IVT offset from the base address and initial load region size for each boot device

type is defined in the table below. The location of the IVT is the only fixed requirement by the ROM. The remainder or the image memory map is flexible and is determined by the contents of the IVT.

Table 7-20. Image Vector Table Offset and Initial Load Region Size

Boot Device Type	Image Vector Table Offset	Initial Load Region Size
NOR	4 Kbyte = 0x1000 bytes	Entire Image Size
OneNAND	256 bytes = 0x100 bytes	1 Kbyte
SD/MMC/eSD/eMMC/SDXC	1 Kbyte = 0x400 bytes	4 Kbyte
I2C/SPI EEPROM	1 Kbyte = 0x400 bytes	4 Kbyte

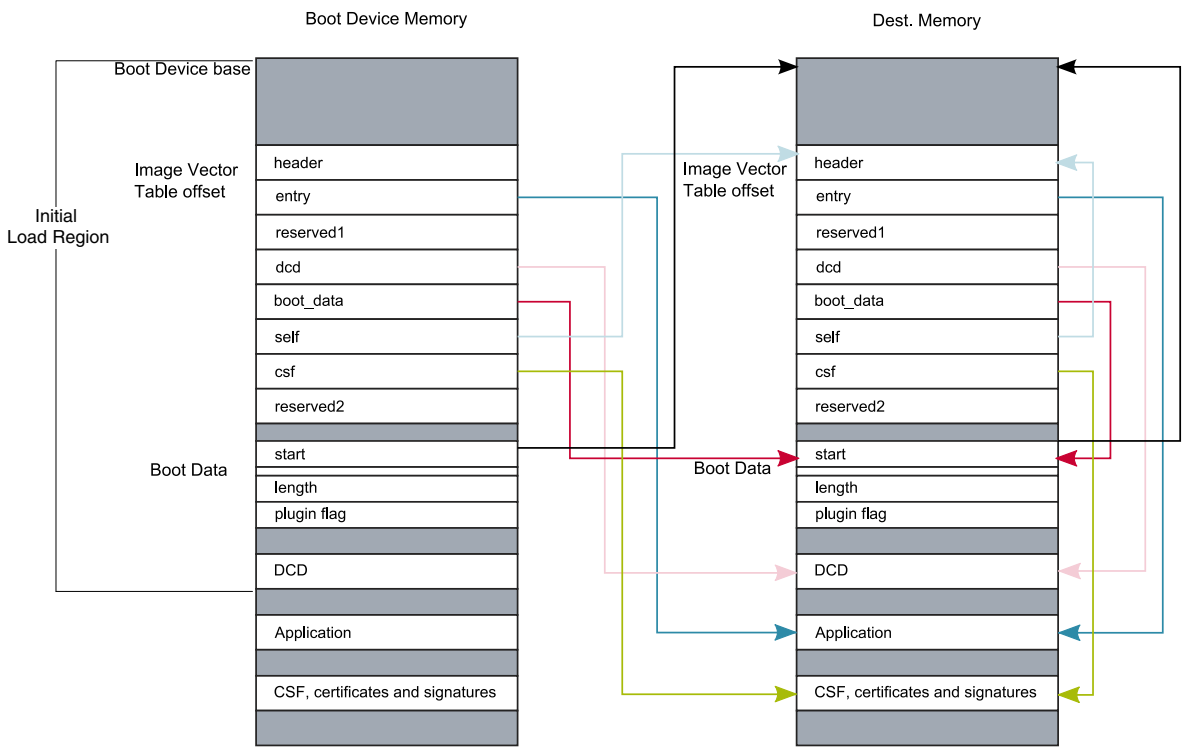


Figure 7-15. Image Vector Table

7.6.1.1 Image Vector Table Structure

The IVT has the following format where each entry is a 32 bit word:

Table 7-21. IVT Format

header

Table continues on the next page...

Table 7-21. IVT Format (continued)

entry: Absolute address of the first instruction to execute from the image
reserved1: Reserved and should be zero
dcd: Absolute address of the image DCD. The DCD is optional so this field may be set to NULL if no DCD is required. See Device Configuration Data (DCD) for further details on DCD.
boot data: Absolute address of the Boot Data
self: Absolute address of the IVT. Used internally by the ROM
csf: Absolute address of Command Sequence File (CSF) used by the HAB library. See High Assurance Boot (HAB) for details on secure boot using HAB. This field must be set to NULL when not performing a secure boot
reserved2: Reserved and should be zero

The IVT header has the following format:

Table 7-22. IVT Header Format

Tag	Length	version

where:

```
Tag: A single byte field set to 0xD1
Length: a two byte field in big endian format containing the overall length of the IVT,
in bytes, including the header. (the length is fixed and must have a value of
32 bytes)
Version: A single byte field set to 0x40 or 0x41
```

7.6.1.2 Boot Data Structure

The Boot Data must follow the format defined in the table found here, each entry is a 32-bit word.

Table 7-23. Boot Data Format

start	Absolute address of the image
length	Size of the program image
plugin	Plugin flag (see Plugin Image)

7.6.2 Device Configuration Data (DCD)

Upon reset, the Chip uses the default register values for all peripherals in the system. However, these settings typically are not ideal for achieving optimal system performance and there are even some peripherals that must be configured before they can be used.

The DCD is configuration information contained in a Program Image, external to the ROM, that the ROM interprets to configure various peripherals on the Chip.

For example, the EIM default settings allow the core to interface to a NOR flash device immediately out of reset. This allows the Chip to interface with any NOR flash device, but has the cost of slow performance. Additionally, some components such as DDR require some sequence of register programming as part of configuration before it is ready to be used. The DCD feature can be used to program the EIM registers and MMDC registers to the optimal settings.

The ROM determines the location of the DCD table based on information located in the Image Vector Table (IVT). See Image Vector Table and Boot Data for more details. The DCD table shown below is a big endian byte array of the allowable DCD commands. The maximum size of the DCD limited to 1768 bytes.

Table 7-24. DCD Data format

Header
[CMD]
[CMD]
...

The DCD header is 4 bytes with the following format:

Table 7-25. DCD Header

Tag	Length	Version

where:

```
Tag: A single byte field set to 0xD2
Length: a two byte field in big endian format containing the overall length of the DCD,
in bytes, including the header
Version: A single byte field set to 0x41
```

7.6.2.1 Write Data Command

The Write Data Command is used to write a list of given 1-, 2- or 4-byte values or bitmasks to a corresponding list of target addresses.

The format of Write Data Command, again a big endian byte array, is shown in the table below.

Table 7-26. Write Data Command Format

Tag	Length	Parameter

Table continues on the next page...

Table 7-26. Write Data Command Format (continued)

Address
Value/Mask
[Address]
[Value/Mask]
...
[Address]
[Value/Mask]

where:

```
Tag: A single byte field set to 0xCC
Length: A two byte field in big endian format containing the length of the Write Data
Command, in bytes, including the header
Address: target address to which data should be written
Value/Mask: data value or bitmask to be written to preceding address
```

The Parameter field is a single byte divided into bitfields as follows:

Table 7-27. Write Data Command Parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: width of target locations in bytes. Either 1, 2 or 4
flags: control flags for command behavior.
Data Mask = bit 3: if set, only specific bits may be overwritten at target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, bits at the target address overwritten with this flag
(otherwise it is ignored)
```

One or more target address and value/bitmask pairs can be specified. The same bytes and flags parameters apply to all locations in the command.

When successful, this command writes to each target address in accordance with the flags as follows:

Table 7-28. Interpretation of Write Data Command Flags

"Mask"	"Set"	Action	Interpretation
0	0	*address = val_msk	Write value
0	1	*address = val_msk	Write value
1	0	*address &= ~val_msk	Clear bitmask
1	1	*address |= val_msk	Set bitmask

NOTE

If any of the target addresses does not have the same alignment as the data width indicated in the parameter field, none of the values are written.

If any of the values is larger or any of the bitmasks is wider than permitted by the data width indicated in the parameter field, none of the values are written.

If any of the target addresses do not lie within an allowed region, none of the values are written. The list of allowable blocks and target addresses for the Chip are given below.

Table 7-29. Valid DCD Address Ranges

Address range	Start address	Last Address
IOMUX Control (IOMUXC) registers	0x020E0000	0x020E3FFF
CCM register set	0x020C4000	0x020C7FFF
MMDC register set	0x021B0000	0x021B7FFF
OCRAM Free Space	0x00907000	0x00937FF0
EIM	0x08000000	0x0FFEFFFF
DDR	0x10000000	0xFFFFFFFF

7.6.2.2 Check Data Command

The Check Data Command is used to test for a given -1, 2- or 4-byte bitmasks from a source address.

The Check Data Command is a big endian byte array with format shown in the table below.

Table 7-30. Check Data Command Format

Tag	Length	Parameter
Address		
Mask		
[Count]		

where:

```
Tag: A single byte field set to 0xCF
Length: A two byte field in big endian format containing the length of the Check Data
Command, in bytes, including the header
Address: source address to test
Mask: bit mask to test
```

```
Count: optional poll count. If count is not specified this command will poll indefinitely
until the exit condition is met. If count = 0, this command behaves as for NOP.
```

The Parameter field is a single byte divided into bitfields as follows:

Table 7-31. Check Data Command Parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: width of target locations in bytes. Either 1, 2 or 4
flags: control flags for command behavior.
Data Mask = bit 3: if set, only specific bits may be overwritten at target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, bits at the target address overwritten with this flag
(otherwise it is ignored)
```

This command polls the source address until either the exit condition is satisfied, or the poll count is reached. The exit condition is determined by the flags as follows:

Table 7-32. Interpretation of Check Data Command Flags

"Mask"	"Set"	Action	Interpretation
0	0	(*address & mask) == 0	All bits clear
0	1	(*address & mask) == mask	All bits set
1	0	(*address & mask)!= mask	Any bit clear
1	1	(*address & mask)!= 0	Any bit set

NOTE

If the source address does not have the same alignment as the data width indicated in the parameter field, the value is not read.

If the bitmask is wider than permitted by the data width indicated in the parameter field, the value is not read.

7.6.2.3 NOP Command

This command has no effect.

The format of NOP Command is a big endian four byte array as shown in the table below.

Table 7-33. NOP Command Format

Tag	Length	Undefined

where:

```
Tag: A single byte field set to 0xC0
Length: A two byte field in big endian containing the length of the NOP Command in bytes.
Fixed to a
value of 4.
Undefined: This byte is ignored and can be set to any value.
```

7.6.2.4 Unlock Command

The Unlock Command is used to prevent specific engine features being locked when exiting ROM.

The format of Unlock Command, again a big endian byte array, is shown in the table below.

Table 7-34. Unlock Command Format

Tag	Length	Eng
Value		
Value		
...		
Value		

where:

```
Tag: A single byte field set to 0xB2
Eng: Engine to be left unlocked, supported engines are SNVS and OCOTP.
Values: [optional] unlock values required by engine. Valid values are:

             for SNVS: 0x01 (HAB_SNVS_UNLOCK_LP_SWR) - leave LP SW reset unlocked (leave
LP_SWR bit cleared in register SNVS_HP command register) and
                               0x02 (HAB_SNVS_UNLOCK_ZMK_WRITE) - leave zeroisable master key
write unlocked (leave ZMK_WSL bit cleared in register SNVS_HP lock register).
            for OCOTP:
                               0x01 (HAB_OCOTP_UNLOCK_FIELD_RETURN) - leave field return
activation unlocked in OCOTP_SW_STICKY register,
                               0x02 (HAB_OCOTP_UNLOCK_SRK_REVOKE) - leave SRK revocation
unlocked in OCOTP_SW_STICKY register,
                               0x04 (HAB_OCOTP_UNLOCK_SCS) - leave lock bit cleared in
OCOTP_SCS register and
                               0x08 HAB_OCOTP_UNLOCK_JTAG) - leave HAB_JDE bit cleared in
OCOTP_SCS register to unlock JTAG.
```

NOTE

This command may not be used in DCD structure if the SEC_CONFIG is configured as closed.

7.7 Plugin Image

The ROM supports a limited number of boot devices. For using other devices as boot source (for example, Ethernet, CDROM, or USB), the supported boot device must be used (typically serial ROM) for firmware with the missing boot drivers.

Additionally plugin can customize supported boot drivers. It is more flexible when doing device initialization, such as condition judging, delay assertion, applying custom settings to boot device and memory system, although not recommended when building a secure system (i.e. SEC_CONFIG[1] efuse blown).

In addition to standard images, the chip also supports plugin images. Plugin images return execution to the ROM whereas a standard image does not. The boot ROM detects the image type using the plugin flag of the boot data structure (see Boot Data Structure). If the plugin flag is 1, then the ROM uses the image as a plugin function. The function must initialize the boot device and copy the program image to the final location. At the end the plugin function must return with the program image parameters. (See High level boot sequence for details about boot flow).

The boot ROM authenticates the plugin image prior to running the plugin function and then authenticates the program image.

The plugin function must follow the API described below:

typedef unsigned char (*) plugin_download_f(void **start, size_t *bytes, UINT32 *ivt_offset)

ARGUMENTS PASSED:

- start - Image load address on exit.
- bytes - Image size on exit.
- ivt_offset - Offset in bytes of the IVT from the image start address on exit.

RETURN VALUE:

- 1 - on success
- 0 - on failure

7.8 Serial Downloader

The Serial Downloader provides a means to download a Program Image to the chip over USB serial connection.

In this mode the ROM programs WDOG-1 for a 32-second time-out if WDOG_ENABLE eFuse is 1, and then continuously polls for USB connection. If no activity is found on USB OTG1 and the watchdog timer expires, the ARM core is reset.

NOTE

The downloaded image must continue to service the watchdog timer to avoid an undesired reset from occurring.

The USB boot flow is shown in the figure below.

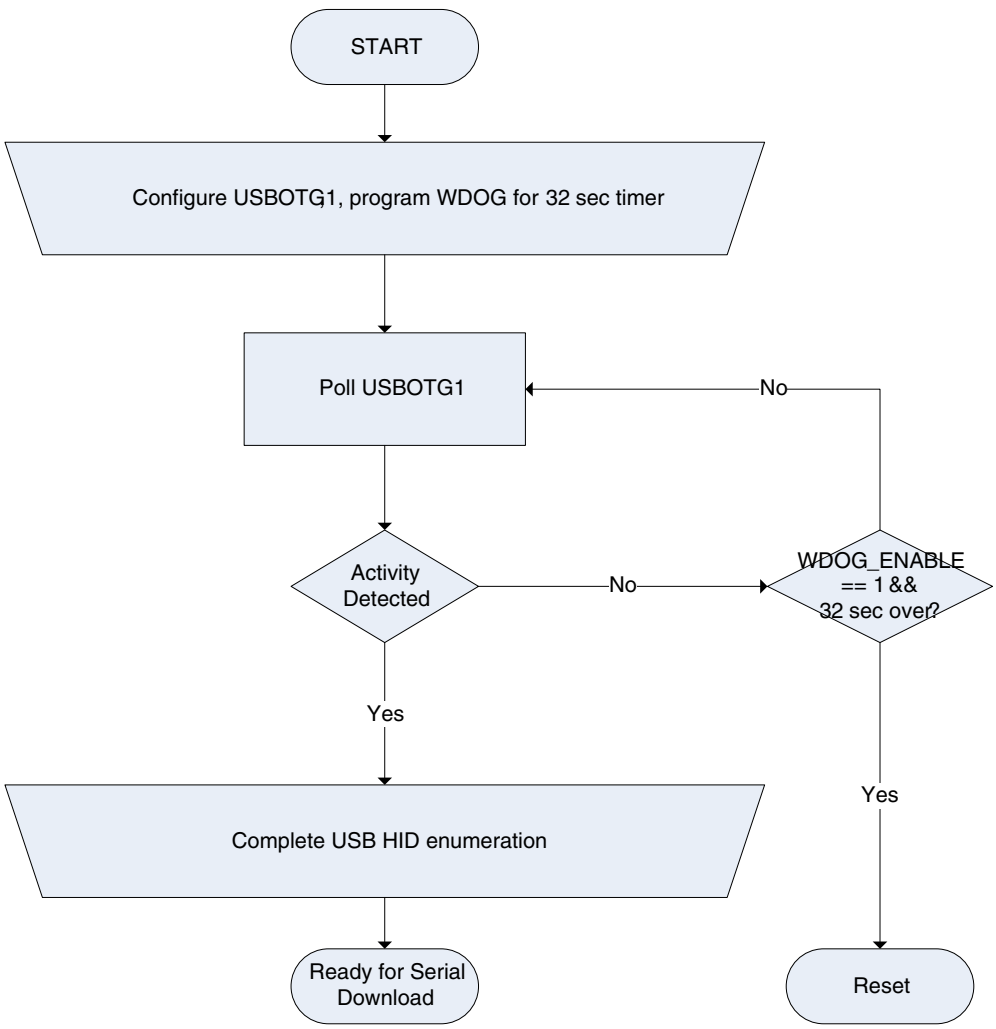


Figure 7-16. USB Boot Flow

7.8.1 USB

USB support is composed of the USBOH3 (USB OTG1 core controller, compliant with the USB 2.0 specification) and the USBPHY (HS USB transceiver).

The ROM supports the USB OTG port for boot purposes. The other USB ports on the chip are not supported for boot purposes.

The USB Driver is implemented as a USB HID class. A collection of 4 HID reports are used to implement SDP protocol for data transfers as described in Table 7-35.

Table 7-35. USB HID Reports

Report ID (first byte)	Transfer Endpoint	Direction	Length	Description
1	control OUT	Host to device	17 bytes	SDP command from host to device
2	control OUT	Host to device	Up to 1025 bytes	Data associated with report 1 SDP command
3	interrupt	Device to host	5 bytes	HAB security configuration. Device sends 0x12343412 in closed mode and 0x56787856 in open mode.
4	interrupt	Device to host	Up to 65 bytes	Data in response to SDP command in report 1

7.8.1.1 USB Configuration Details

The USB OTG function device driver supports a high speed (HS for UTMI) non-stream mode with a maximal packet size of 512 B and a low-level USB OTG function.

The VID/PID and strings for USB device driver are listed in the table below.

Table 7-36. VID/PID and Strings for USB Device Driver

Descriptor	Value
VID	0x15A2 (Freescale vendor ID)
PID[1]	
String Descriptor1 (manufacturer)	Freescale Semiconductor, Inc.
String Descriptor2 (product)	S Blank SE Blank NS Blank

Table continues on the next page...

Table 7-36. VID/PID and Strings for USB Device Driver (continued)

Descriptor	Value
String Descriptor4	Freescale Flash
String Descriptor5	Freescale Flash

1. Allocation based on BPN (Before Part Number)

7.8.1.2 IOMUX Configuration for USB

The interface signals of the UTMI PHY are not configured in the IOMUX. The UTMI PHY interface uses dedicated contacts on the IC. See the Chip data sheet for details.

7.8.2 Serial Download protocol

The 16 byte SDP command from host to device is sent using HID report 1.

The table below describes 16 byte SDP command data structure:

Table 7-37. 16 Byte SDP Command Data Structure

BYTE Offset	Size	Name	Description
0	2	COMMAND TYPE	The following commands are supported for ROM: • 0x0101 READ_REGISTER • 0x0202 WRITE_REGISTER • 0x0404 WRITE_FILE • 0x0505 ERROR_STATUS • 0x0A0A DCD_WRITE • 0x0B0B JUMP_ADDRESS
2	4	ADDRESS	Only relevant for following commands: READ_REGISTER, WRITE_REGISTER, WRITE_FILE, DCD_WRITE, and JUMP_ADDRESS. For READ_REGISTER and WRITE_REGISTER commands, this field is address to a register. For WRITE_FILE and JUMP_ADDRESS commands, this field is an address to internal or external memory address.
6	1	FORMAT	Format of access, 0x8 for 8-bit access, 0x10 for 16-bit and 0x20 for 32-bit access. Only relevant for READ_REGISTER and WRITE_REGISTER commands.
7	4	DATA COUNT	Size of data to read or write. Only relevant for WRITE_FILE, READ_REGISTER, WRITE_REGISTER and DCD_WRITE commands. For WRITE_FILE and DCD_WRITE commands DATA COUNT is in byte units.

Table continues on the next page...

Table 7-37. 16 Byte SDP Command Data Structure (continued)

BYTE Offset	Size	Name	Description
11	4	DATA	Value to write. Only relevant for WRITE_REGISTER command.
15	1	RESERVED	Reserved

7.8.2.1 SDP Command

SDP commands are decribed in the following sections.

7.8.2.1.1 READ REGISTER

The transaction for command READ_REGISTER consists of following reports: Report1 for command, Report3 for security configuration and Report4 for response or register value.

The register to read is specified in ADDRESS field of SDP command. First device sends Report3 with security configuration followed by Report4 with bytes read at given address. If count is greater than 64 then multiple reports with report id 4 are sent until entire data requested by host is sent. The STATUS is either 0x12343412 for closed parts and 0x56787856 for open or field return parts.

Report1, Command, Host to Device:

1	Valid values for READ_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host: first response report

4	Register Value

ID 4 bytes of data containing register value. If number of bytes requested is less than 4 then remaining bytes should be ignored by host.

Multiple reports of report id 4 are sent until entire data requested is sent

Report4, Response, Device to Host: Last response report

4	Register Value

ID 64 bytes of data containing register value. If number of bytes requested is less than 64 then remaining bytes should be ignored by host.

7.8.2.1.2 WRITE REGISTER

The transaction for command WRITE_REGISTER consists of the following reports: Report1 for command, Report3 for security configuration and Report4 for write status.

Host sends Report1 with WRITE_REGISTER command. The register to write is specified in ADDRESS field of SDP command of Report1, with FORMAT field set to data type (number of bits to write 8, 16 or 32) and value to write in DATA field of SDP command. Device writes the DATA to register address and returns WRITE_COMPLETE code using Report4 and security configuration using Report3 to complete the transaction.

Report1, Command, Host to Device:

1	Valid values for WRITE_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT and DATA

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes data with first 4 bytes to indicate write is completed with code 0x128A8A12. On failure device will report HAB error status.

7.8.2.1.3 WRITE_FILE

The transaction for command WRITE_FILE consists of following reports: Report1 for command-phase, Report2 for data-phase, Report3 for hab mode and Report4 to indicate data received in full.

The size of each Report2 is limited to 1024 bytes (limitation of USB HID protocol) hence multiple Report2 packets will be sent by host in data phase until entire data is transferred to device. Once entire data (DATA_COUNT bytes) is received then device sends report 3 with hab mode and report 4 with 0x88888888, indicating file download completed.

Report1, Host to Device:

1	Valid values for WRITE_FILE COMMAND, ADDRESS, DATA_COUNT

ID 16 byte SDP Command

==========Optional Begin==========

Host sends ERROR_STATUS command to query if HAB rejected the address

========== Optional End==========

Report2, Host to Device:

2	File data

ID Max 1024 bytes data per report

Report2, Host to Device:

2	File data

ID Max 1024 bytes data per report

Report3, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	COMPLETE (0x88888888) status

ID 64 bytes data with first 4 bytes to indicate file download has completed with code 0x88888888. On failure device will report HAB error status.

7.8.2.1.4 ERROR_STATUS

The transaction for SDP command ERROR_STATUS consists of three reports.

Report1 is used by host to send the command; device sends global error status in 4 bytes of Report4 after returning security configuration in Report3. When device receives ERROR_STATUS command it will return global error status that is updated for each command. This command is useful to find out if last command resulted in device error or succeeded.

Report1, Command, Host to Device:

1	ERROR_STATUS COMMAND

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	4 bytes Error status

ID first 4 bytes status in 64 bytes report 4

7.8.2.1.5 DCD WRITE

The SDP command DCD_WRITE is used by host to send multiple register writes in one shot. This command is provided to speed up the process of programming register writes such as to configure external RAM device.

The command goes with Report1 from host with COMMAND TYPE set to DCD_WRITE, ADDRESS which is used for temporary location of DCD data and DATA_COUNT to number of bytes sent in data out phase. In data phase host sends data

for number of registers using Report2. Device completes the transaction with Report3 indicating security configuration and report 4 with WRITE_COMPLETE code 0x12828212.

Report1, Command, Host to Device:

1	DCD_WRITE COMMAND, ADDRESS, DATA_COUNT

ID 16 byte SDP Command

Report2, Data, Host to Device:

2	DCD binary data

ID Max 1024 bytes per report

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes report with first 4 bytes to indicate write is completed with code 0x128A8A12. On failure device will report HAB error status.

See Device Configuration Data (DCD) for DCD format description.

7.8.2.1.6 JUMP ADDRESS

The SDP command JUMP_ADDRESS will be the last command host can send to the device, after this command device will jump to the address specified in the ADDRESS field of SDP command and start executing.

This command should typically follow after WRITE_FILE command. The command is sent by host in command-phase of transaction using Report1, there is no data phase for this command but device send status report3 to complete the transaction. And if HAB authentication fails then it will also send report 4 with HAB error status.

Report1, Command, Host to Device:

1	JUMP_ADDRESS COMMAND, ADDRESS

ID 16 byte SDP Command

Report3, Response, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

This report is sent by device only in case of an error jumping to the given address, device reports error in Report4, Response, Device to Host:

4	4 bytes HAB error status

ID 4 bytes status, 64 bytes report length

7.9 Recovery Devices

The Chip supports recovery devices. If primary boot device fails, boot ROM will try to boot from recovery device using one of I2C or ECSPI ports.

For enabling recovery device BOOT_CFG4[6] fuse must be set. Additionally Serial EEPROM fuses must be set as described in Serial ROM through SPI and I2C.

7.10 USB Low Power Boot

ROM supports USB Low Power Boot. This feature enables a device with dead or weak battery to power up and boot if the device is connected to a USB upstream port, no matter the upstream port is a USB charger or USB host/hub.

If a USB dedicated charger or host/hub charger are connected, as soon as the device is connected to the upstream port, a stable current (Max.1.5A) can be supplied by charger. If USB host/hub are connected, the maximal 1 00mA current is supplied to the device, the device should be able to power up to boot the image with less than 100mA.

If LPB_BOOT fuses are blown, the Chip will check if there is low power condition via GPIO_3 pad. If there is low power boot condition USB charger detection will be activated. If there is no USB charger, ROM will initialize USB as device and apply

division factors on ARM, DDR, AXI and AHB root clocks based on LPB_BOOT fuses value (see the table below). Polarity of low power boot condition on GPIO_3 pad is set by BT _LPB_POLARITY fuse (see the figure below).

Table 7-60. USB Low Power Boot Frequencies

LPB_BOOT	Boot Frequencies=0	Boot Frequencies=1
00	ARM_CLK_ROOT=792MHz MMDC_CH0_AXI_CLK_ROOT=528MHz MMDC_CH1_AXI_CLK_ROOT=528MHz AXI_CLK_ROOT=264MHz AHB_CLK_ROOT=132MHz	ARM_CLK_ROOT=396MHz MMDC_CH0_AXI_CLK_ROOT=352MHz MMDC_CH1_AXI_CLK_ROOT=352MHz AXI_CLK_ROOT=176MHz AHB_CLK_ROOT=88MHz
01	ARM_CLK_ROOT=792MHz MMDC_CH0_AXI_CLK_ROOT=528MHz MMDC_CH1_AXI_CLK_ROOT=528MHz AXI_CLK_ROOT=264MHz AHB_CLK_ROOT=132MHz	ARM_CLK_ROOT=396MHz MMDC_CH0_AXI_CLK_ROOT=352MHz MMDC_CH1_AXI_CLK_ROOT=352MHz AXI_CLK_ROOT=176MHz AHB_CLK_ROOT=88MHz
10	ARM_CLK_ROOT=396MHz MMDC_CH0_AXI_CLK_ROOT=264MHz MMDC_CH1_AXI_CLK_ROOT=264MHz AXI_CLK_ROOT=132MHz AHB_CLK_ROOT=66MHz	ARM_CLK_ROOT=264MHz MMDC_CH0_AXI_CLK_ROOT=176MHz MMDC_CH1_AXI_CLK_ROOT=176MHz AXI_CLK_ROOT=88MHz AHB_CLK_ROOT=44MHz
11	ARM_CLK_ROOT=264MHz MMDC_CH0_AXI_CLK_ROOT=132MHz MMDC_CH1_AXI_CLK_ROOT=132MHz AXI_CLK_ROOT=66MHz AHB_CLK_ROOT=66MHz	ARM_CLK_ROOT=132MHz MMDC_CH0_AXI_CLK_ROOT=88MHz MMDC_CH1_AXI_CLK_ROOT=88MHz AXI_CLK_ROOT=44MHz AHB_CLK_ROOT=44MHz

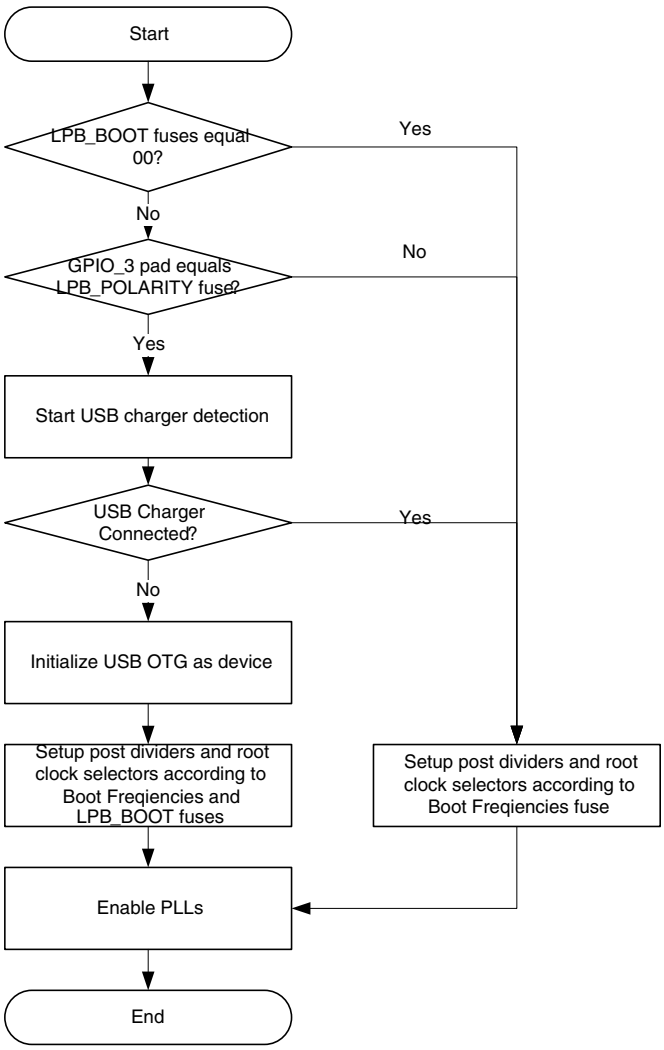


Figure 7-17. USB Low Power Boot Flow

7.11 SD/MMC Manufacture Mode

When internal boot and recover boot (if enabled) failed, boot will go to SD/MMC manufacture mode before serial download mode. In manufacture mode, one bit bus width is used despite of the fuse setting.

In manufacture mode, SD or MMC card will be scanned on uSDHC1 and uSDHC2. If card is detected and valid boot image is found in card, then boot image will be loaded then executed. Pad of SD1_CD and SD2_CD are used to detect whether card is inserted.

By default, SD/MMC manufacture mode is enabled, blow the fuse of DISABLE_SDMMC_MFG to disable it.

NOTE

Secondary boot is not supported on SD/MMC manufacture mode.

NOTE

SD/MMC Manufacture mode is not available on this chip's ROM.

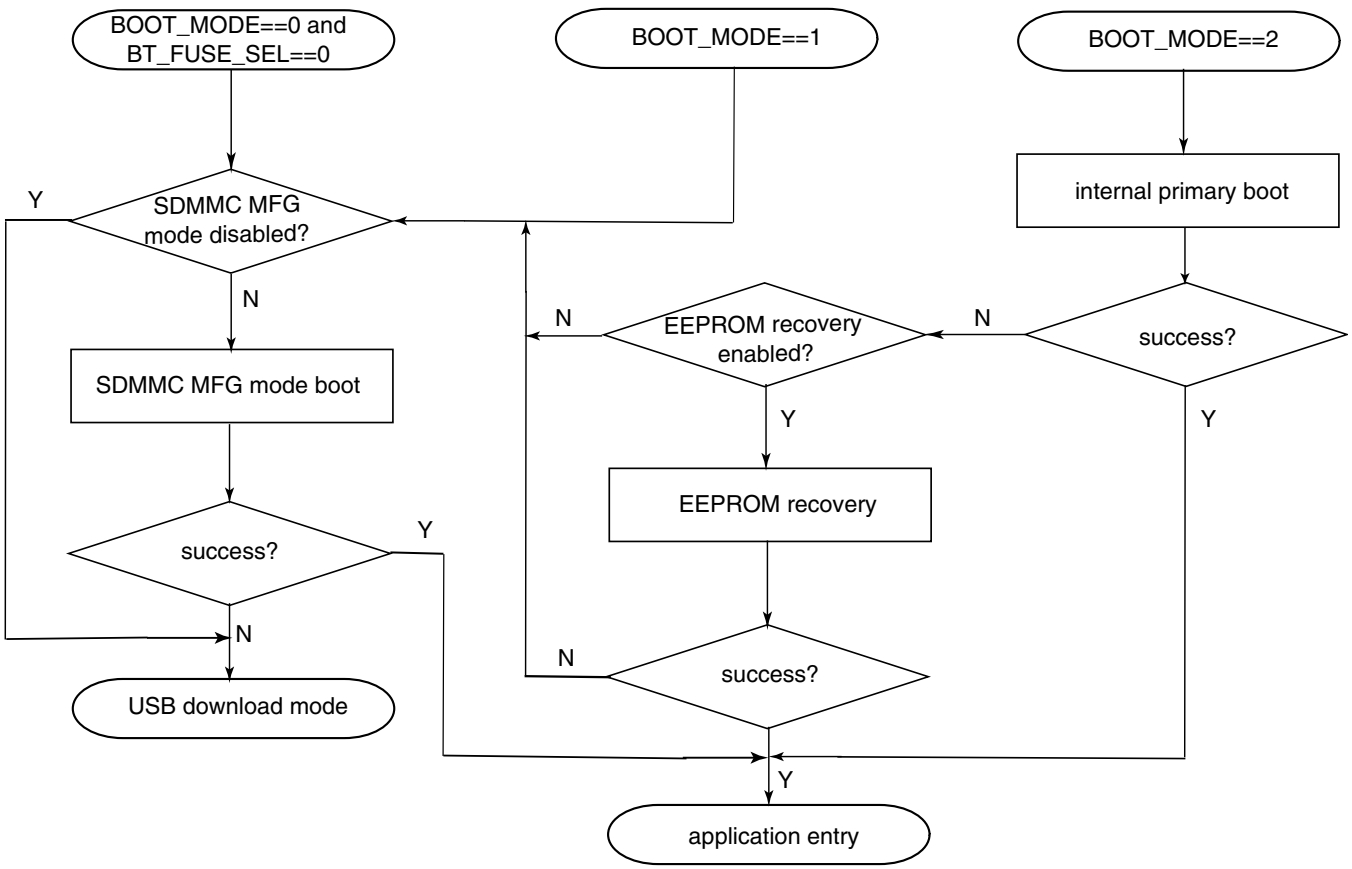


Figure 7-18. SD/MMC Manufacture boot flow

7.12 High Assurance Boot (HAB)

The High Assurance Boot (HAB) component of the ROM protects against the potential threat of attackers modifying areas of code or data in programmable memory to make it behave in an incorrect manner. The HAB also prevents attempts to gain access to features which should not be available.

The integration of the HAB feature with the ROM code ensures that Chip does not enter an operational state if the existing hardware security blocks have detected a condition that may be a security threat or areas of memory deemed to be important have been modified. The HAB uses RSA digital signatures to enforce these policies.

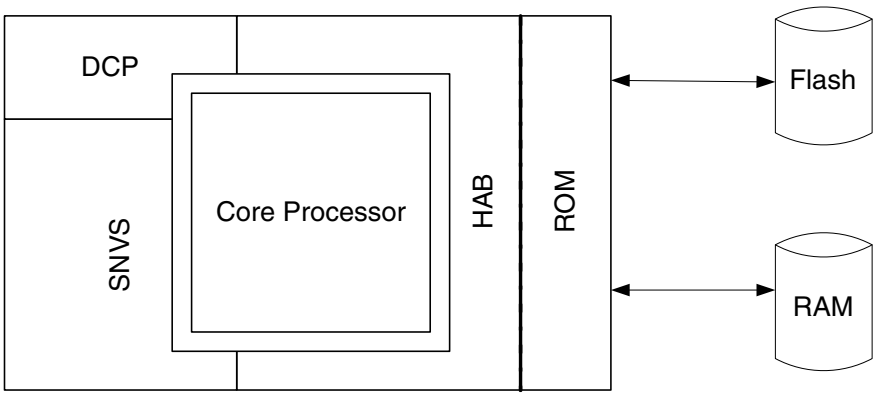


Figure 7-19. Secure Boot Components

The figure above illustrates the components used during a secure boot using HAB. The HAB interfaces with the SNVS to ensure the system security state is as expected. The HAB also makes use of DCP hardware block to accelerate SHA-256 message digest operations performed during signature verifications. The HAB also includes a software implementation of SHA-256 for cases where a hardware accelerator cannot be used. The RSA key sizes supported are 1024, 2048 and 3072 bits. The RSA signature verification operations are performed by a software implementation contained in the HAB library. The main features supported by HAB are:

- X.509 Public key certificate support
- CMS signature format support

NOTE

Freescale provides a reference Code Signing Tool (CST) for key generation and code signing for use with the HAB library. The CST can be found by searching for "IMX_CST_TOOL" at: http://www.freescale.com.

NOTE

For further details on making use of the secure boot feature using HAB contact your local Freescale representative.

7.12.1 ROM Vector Table Addresses

For devices that perform a secure boot, the HAB library may be called by boot stages that execute after ROM code. The RVT table contains the pointers to the HAB API functions and is located at 0x00000094.

NOTE

For additional information on secure boot including the HAB API, contact your local Freescale representative.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Document Number: IMX6SLSRM
Rev. 0
03/2013