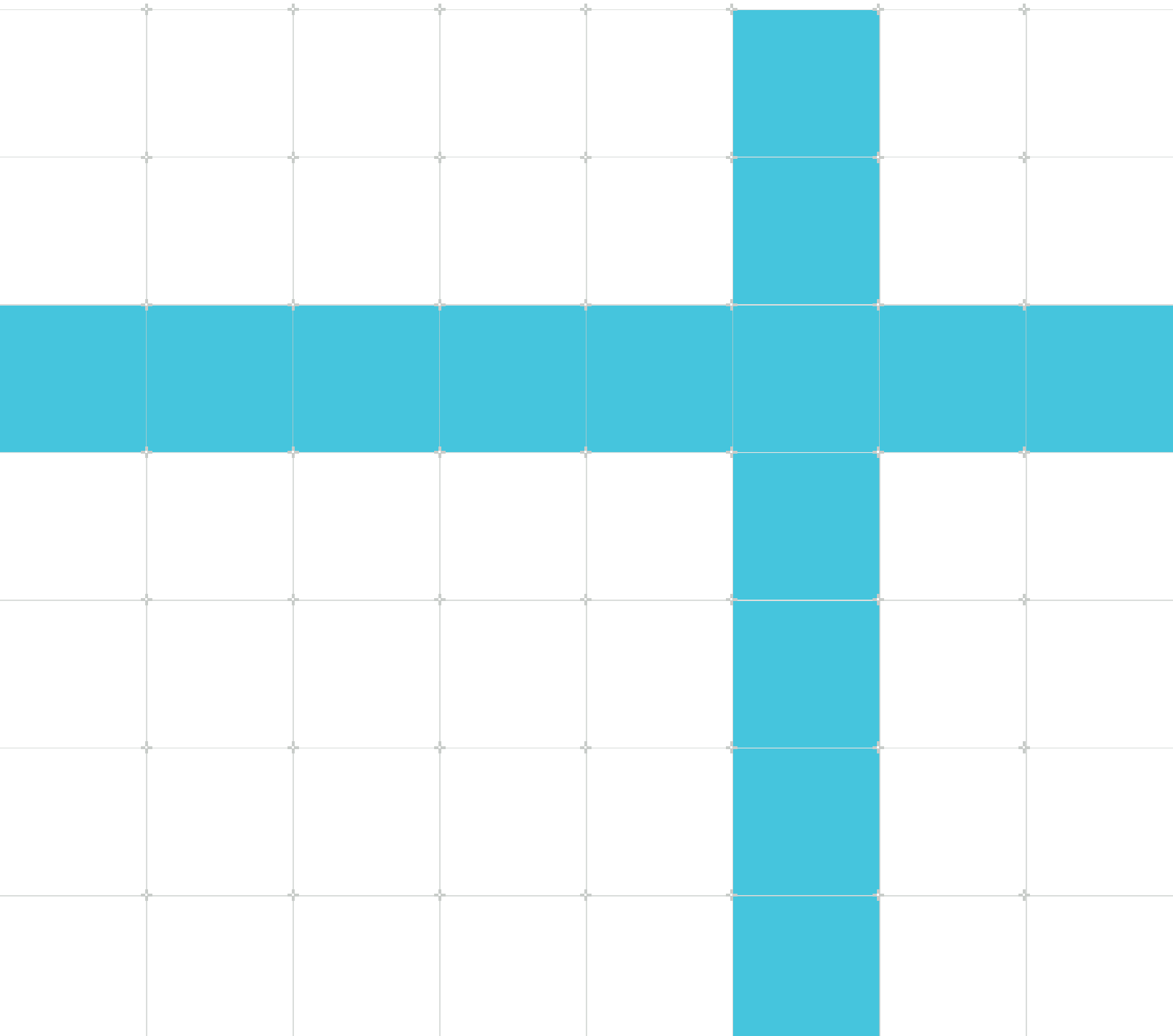
Arm® Key Management Unit

Version 1.0

Specification

Issue 01
107715_0000_01_en



Arm® Key Management Unit

Specification

Release information

Document history

Issue	Date	Confidentiality	Change
0000-01	31 May 2023	Non-Confidential	Initial release

Proprietary Notice

and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Introduction

1.1 Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.
Caution	Recommendations. Not following these recommendations might lead to system failure or damage.
Warning	Requirements for the system. Not following these requirements might result in system failure or damage.
Danger	Requirements for the system. Not following these requirements will result in system failure or damage.
Note	An important piece of information that needs your attention.

Convention	Use
 Tip	A useful tip that might make it easier, better or faster to perform a task.
 Remember	A reminder of something important that relates to the information you are reading.

1.2 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual	DDI 0571	Non-Confidential

Arm architecture and specifications	Document ID	Confidentiality
Arm® Lifecycle Manager Specification	107616	Non-Confidential
Arm® Security Alarm Manager Specification	107716	Non-Confidential

Non-Arm resources	Document ID	Organization
NIST SP 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications	NIST SP 800-22	NIST


Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at http://www.adobe.com

1.3 Other information

See the Arm website for other relevant information.

- Arm® Developer.
- Arm® Documentation.
- Technical Support.

- Arm® Glossary.

2. KMU overview

The Arm® *Key Management Unit* (KMU) is a centralized key management architecture that stores symmetric key material (assets). The keys stored in the KMU are not readable to software or other hardware components.

Software can only use an asset stored in the KMU to perform cryptographic operations by exporting a key to a cryptographic device. The key material stored in the KMU can be exported to the distributed hardware countermeasure components.

The KMU implements key slots that can be defined as either Hardware key slots or Software key slots.

The KMU is only accessible by secure Software, see 3.7.1 APB4 subordinate programming port on page 23. Non-secure software must call Secure software to set or use a key. After a key slot is loaded, Secure software can lock it to prevent accidental reload or malicious override. The locking scheme of a key slot may be permanent or temporary:

Permanent

The key slot is locked until the next reset using the KSIP bit of the KMUKSC<n> register.

Temporary

The key slot is locked until it is invalidated by a software action using the IKS bit of the KMUKSC<n> register. When a software key slot is invalidated, the KMU zeroizes its key and unlocks it to allow software to reload and lock it again. Hardware key slots are zeroized when invalidated but they cannot be reused by the software.

For more information about the KMUKSC<n> register, see 6.1.6 KMUKSC<n>, Key slot configuration register on page 38.

Key slots can only be used for loading assets to hardware devices. Software cannot read or access the KMU assets in any way when they are locked. The key registers of the destination devices, which the KMU loads, are assumed to be *write only* (WO).

To protect Secure assets in transit, moving the assets from the KMU to a destination cryptographic device can be protected by a *local write mask filter* (LWMF) inside the KMU and an optional *remote write mask filter* (RWMF). The optional RWMF unmasks the Secure data and verifies its parity. It is attached as close as possible to the destination cryptographic device to reduce the risk of an electromagnetic attack on the clear key (asset) data in transit. The KMU is responsible for altering the masks at both sides of every key transfer.

2.1 Topology

This topics provides information about the KMU's internal and external components.

The centralized KMU can provide an easier way for the system to keep its assets invisible and reduce the amount of hardware implemented to preserve keys through low-power states.

Figure 2-1: KMU block diagram

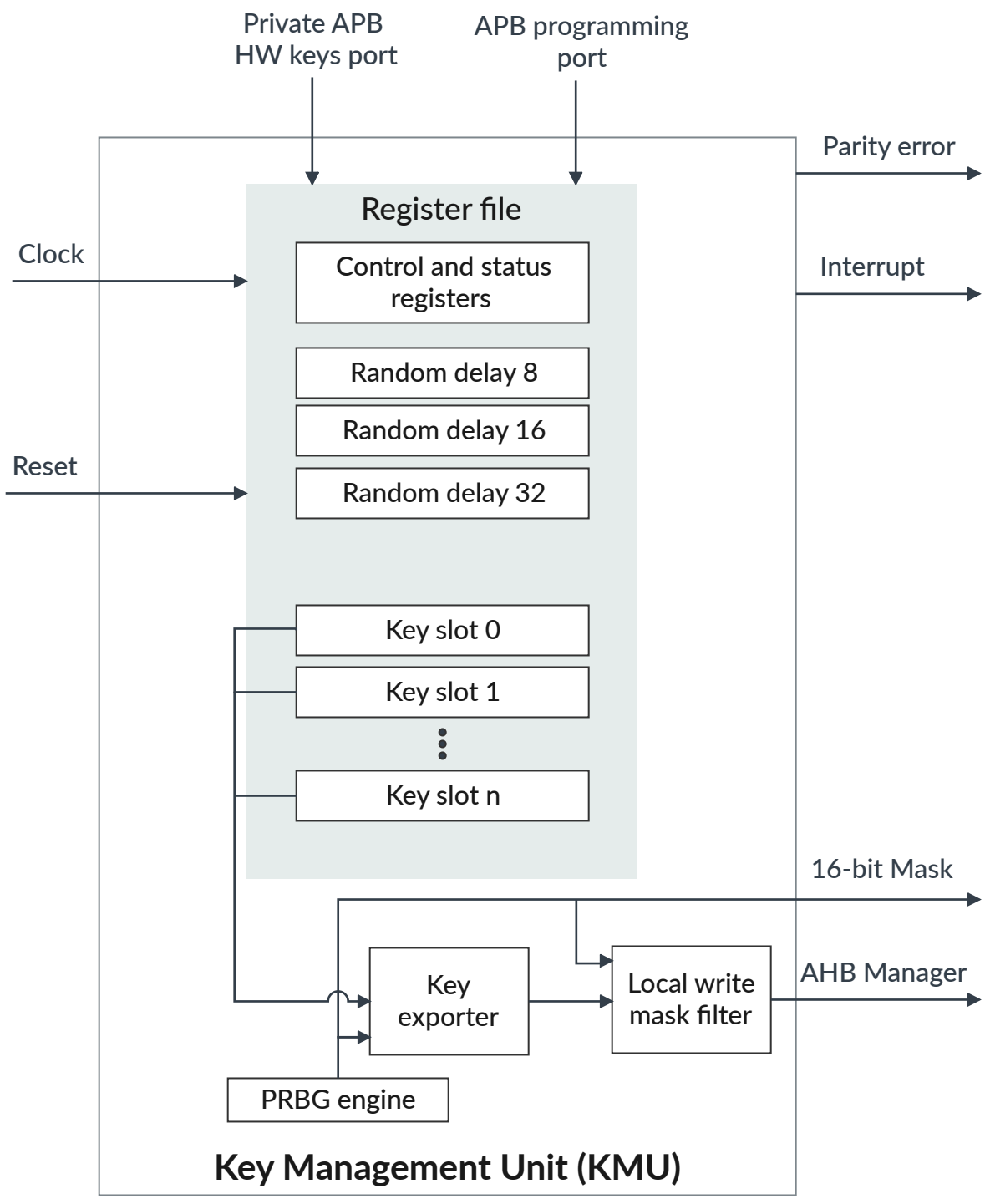


Figure 2-1: KMU block diagram on page 11 shows the KMU components. The internal and external hardware components of the KMU are:

Internal hardware components

These are components that are internal to the KMU.

3.1 KMU register block on page 16

Used by software to configure the KMU, initiate key export, and check its status. The KMU registers block includes:

- Control registers for Hardware key slots and Software key slots
- Random delay registers

3.2 Key exporter on page 16

Exports a key slot asset to its destination.

3.3 Local write mask filter on page 20

An AMBA AHB5 version of the *remote write mask filter* (RWMF). Responsible for masking the keys which are written to the destination devices. It is also responsible for generating parity on the exported masked keys.

3.4 Pseudorandom bit generator on page 20

The *pseudorandom bit generator* (PRBG) produces pseudorandom bits using algorithms which pass *NIST SP 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications* tests.

External hardware components

These are components that are external to the KMU but still part of the subsystem. They are optional. You can use them when integrating the KMU.

3.5 Remote write mask filter on page 21

Integrated as a filter between the Arm TrustZone® *Peripheral Protection Controller* (PPC) and the remote AMBA APB cryptographic device. For security reasons, the RWMF should be physically placed near the destination cryptographic device. RWMF is also responsible for checking the parity of the incoming masked key and providing a parity error alarm signal to the subsystem, if the subsystem supports parity.

3.6 Key store device on page 22

An APB device which supports the storing of a 64-bit key to be provided as a 64-bit output signal. The 64-bit output signal can be used by some special crypto devices, for example the CC315 with its PKA memory encryption key.

KMU register file

The KMU register file stores the KMU control state and the key-slots values.

The KMU register file supports a parity bit for every eight bits of its registers. When a register is used, its parity is verified. In case of a mismatch, an alarm interrupt and the parity error signal are set.

The KMU supports only key sizes of 128 bits and 256 bits. Other key sizes are rejected by the KMU.

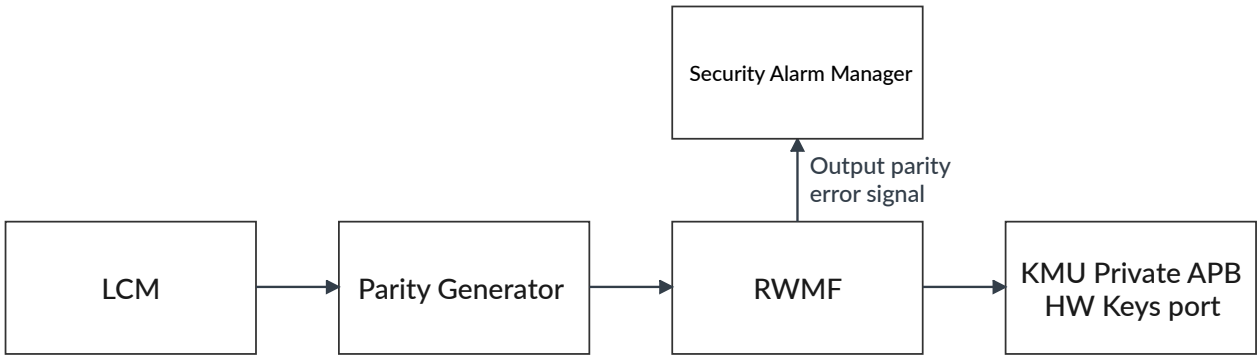
2.2 Hardware key slots

Hardware key slots are typically used to hold the *Root of Trust* (RoT) keys and the hardware unique keys of a specific device. RoT and hardware unique keys are set once and canot be modified by the software.

The configuration of the hardware key slots is hardcoded as hardware reset values for the key slot registers. This default configuration also locks the hardware key slots. The keys in the hardware key slots can only be set and loaded through a private APB3 subordinate hardware keys port (KMU_PAHKP), which connects through a private link from an external source.

Figure 2-2: Loading hardware keys into the KMU on page 13 shows an example of the process to load a hardware key from an external RoT unit to the KMU. In this example, the Arm Lifecycle Manager (LCM) is used as the external RoT unit.

Figure 2-2: Loading hardware keys into the KMU



The LCM is connected to the KMU through a private link. The LCM does not generate parity. Therefore, for additional protection of the LCM and the KMU, it is recommended to use a parity generator. The parity generator, located near the LCM, generates parity on the address and data signals between the LCM and the KMU private APB3 subordinate hardware keys port. If the key source supports write masking, a *remote write mask filter* (RWMF) can also be placed in front of the KMU private APB3 subordinate hardware keys port to check the transaction parity and unmask the transferred key. When the LCM loads the hardware key slots, software can use them as a locked software key slot.

For more information on LCM, see the *Arm® Lifecycle Manager Specification*.

The private APB3 subordinate hardware keys port should be accessible only to the external RoT unit, in a way that it is not accessible to software or other hardware components. Writes to the private APB3 subordinate hardware keys port are allowed only to the hardware key slots. Any write to other addresses in the KMU results in:

- A bus error at the private APB3 subordinate hardware keys port
- The write is ignored
- The AWBHKSKR bit is set in the KMUIS register

An invalidated hardware key slot cannot be reused or loaded again until Cold reset. For more information about invalidated hardware key slots, see 5.1 Key slots allocation example on page 28.

The hardware key slot control registers are set with default values on exit from reset which locks them. The hardware key slots key registers are written by the external RoT unit through a private APB3 subordinate hardware keys port, accessible only to the external RoT unit. The private APB3 subordinate hardware keys port can write to the hardware key slots, even though the hardware key slots are locked by default for access from the APB4 subordinate programming port.

2.3 Software key slots

Software key slots are programmed by Secure software through an APB4 subordinate programming port.

For each key slot, Secure software sets:

- The key slot key values
- They key slot sizes. The KMU supports only 128-bit and 256-bit keys
- The address of the destination register to which the device key is exported
- The export write technique, which includes the following options:
    - Incremental addresses, either with an increment size or a fixed non-incremental write address
    - The clock delay between writes
    - The data width per write: 8 bits, 16 bits, or 32 bits
    - The number of writes required to transfer the key: 8, 16, or 32
    - Write Mask Filter Control (enabled/disabled)
    - Key slot control registers lock

The KMU supports software key slot asset delivery through its AHB5 manager port. The KMU KMUTANG configuration parameters filter the used address. The AHB5 manager port can be integrated in a system to deliver the assets in the following ways:

- Export through a private AHB bus interconnect, either fully or partially, to an intended single-target peripheral. This ensures that the peripherals cannot see each other's assets.
- Export through an AHB bus interconnect, either in part or fully, to a 64-bit PKA key store. The 64-bit PKA key store converts the written assets onto a custom set of signals that a target cryptographic hardware can accept, if it does not support a compatible bus interface.

All key slots have a standard size of 256 bits. However, the export target for the key can require a smaller key than 256-bit. Export is customizable to the requirements of the export target using the Destination Port Data Width (DPDW) field and Destination Port Address Increment (DPAI) field of the appropriate KMUKSC register.

The Software loading of a key asset into the key slot of the KMUKSK<n><m> registers (m = 0..7 for a 256-bit key) might be performed in any order. Random write order is suggested as a countermeasure. When the software writes a key asset into the key slot of the KMUKSK<n><m> registers, it performs the following process:

1. The software writes to the intended key slot.
2. The software can read the KMUKSK<n><m> registers and compare them to the intended value and ensure the write was not attacked. If the value is mismatched and flagged as attacked, the writing can be tried again.
3. Software locks the key slot.

   The KMUKSC<n> register is read verified by software after it is locked. From this point on, the KMUKSK<n><m> registers are no longer readable and all other key slot related registers become read-only.
4. The software must verify key slot consistency by setting the KMUKSC<n>.VKS bit to 1 and verify that the KMUKSC<n>.KSR is set to 1. If not set, then the key slot cannot be used.

When the software wants to export a key asset written into the key slot of the KMUKSK<n><m> registers, it performs the following process:

1. The software sets the KMUKSC<n>.EK bit to 1.
2. After the key has been exported the KMUKSC<n>.EK bit returns to 0 and the KMUIS.KEC interrupt bit is set.

The KMU provides random delay read-only registers, that allow software to perform a sequence of Secure operations, with a random delay in between. The random delay acts as a countemeasure against fault injection.

3. Functional description

The KMU has several internal and external hardware components that provide different functionality.

3.1 KMU register block

The software uses the KMU registers block to configure the KMU and check its status. It is accessible through the APB4 subordinate programming port.

An external RoT unit, for example the LCM, that provides the root key can access the hardware key slots part of the KMU registers block through the private APB4 subordinate hardware keys port.

3.1.1 Random delay registers

When a read transaction occurs by the software from the APB4 subordinate programming port to a random delay register, the register consumes a pseudorandom initialization value.

The pseudorandom initialization value is a counter based on the size of the used KMURD register from the PRBG according to the declared maximal delay of the used KMURD register. Each random delay register is read-only, which means it ignores writes. The counter is decremented every clock cycle. When the counter reaches 0, the register completes the response to the reader. The reader is stalled until receiving this response.

After the counter pseudorandom initialization value from the PRBG is used, the PRBG generates a new pseudorandom value.

The response value for reads from these registers is always 0.

The random delay registers are:

[KMURD_8]
Provides its response after 0 to 7 clock cycles of delay.

[KMURD_16]
Provides its response after 0 to 15 clock cycles of delay.

[KMURD_32]
Provides its response after 0 to 31 clock cycles of delay.

For more information about the PRBG, see 3.4 Pseudorandom bit generator on page 20.

3.2 Key exporter

Setting the EK bit in the KMUKSC<n>.EK bit triggers the key exporter. The key exporter starts to export a key slot asset to its destination as defined in the KMUKSC<n> and KMUDKPA<n> registers of the key slot, where <n> represents the key slot number.

For more information about the KMUKSC<n> register, see 6.1.6 KMUKSC<n>, Key slot configuration register on page 38. For more information about the KMUDKPA<n> regsiter, see6.1.7 KMUDKPA<n>, destination key port address n register on page 42.

The key exporter uses the key slot configuration register values, KMUKSC<n>. Whenever a key slot register is used, its parity is checked. If a parity error occurs, a parity interrupt is signaled and a parity error signal, kmu_parerr, is set at the output of the KMU.

The properties of the key exporter are as follows:

- Key slot assets are transferred from the selected KMU key slot to a destination cryptographic device using AHB5 transactions of the required size.
- All key slot key registers, KMUKSK<n><m>, are 256-bit wide.
- A selected key slot asset can be exported using 8-bit, 16-bit, or 32-bit write transfers, according to the requirements of the destination cryptographic device. These requirements are set by the software in the DPDW field of the KMUKSC<n> register.
- When a key fragment, 8-bit or 16-bit, is exported, the unused MS bits of the transaction are zeroized. This is because of security concerns.
- The number of key fragment write transfers to the destination cryptographic device is according to the requirements of the destination cryptographic device. These requirements are set by software in the NDPW field of the KMUKSC<n> register.
- The actual key size that is delivered in bits to the destination cryptographic device: $2^{2+NDPW} * 2^{3+DPDW}$

Optionally, as a countermeasure:

1. The asset in transit from the KMU to its destination can be masked at the KMU by an LWMF.
2. The asset in transit is unmasked by the optional RWMF at the destination.
3. The key exporter reads a new random mask for every transfer that it makes from the PRBG.
4. It provides 16 mask bits as user signals (HWUSER[15:0]) to the LWMF.
5. It adds these 16 User signals to the AHB5 write that the key exporter initiates to the target cryptographic device with parity bits. All other bus managers that write to the destination cryptographic device must provide 16 user bits (HWUSER[15:0]) which are 0s, to not harm its write.

High-level description of the key exporter flow:

1. Key slot assets are written in sequential order to the pointed remote cryptographic device key register using AHB5 write transactions.
2. After every write, the pointer is advanced to the next remote cryptographic device key register, according to the export policy defined by the DPAI field of the KMUKSC<n> register.

3. As a countermeasure, the key data in transit is masked by the LWMF at the KMU, and is unmasked by the RWMF at the destination.

For a more detailed information about the key exporter flow, see 3.2.1 Key exporter flow on page 18.

3.2.1 Key exporter flow

The software sets the KMUKSC<n>.EK bit to 1, which triggers the key exporter.

If the software attempts to export from another key slot while the key exporter is busy, the KMU sets the AWBE error bit in the KMUIS register, which generates an interrupt, but completes the key export already in progress.

When the key export starts, the state of the key exporter becomes “busy” and it performs the following flow:

1. Copy the parameters of the initiating key slot, including the key from this slot, to the key exporter. The eight key slot registers are copied into a temporary 256-bit key slot register.
2. Internally set `MoreBlindingMasks = 1`
3. Internally set `BlindingMask = 0x0000_0000`
4. Set `LoopCount` based on the NDPW bit of the KMUKSC<n> register

   0x0

   `LoopCount` = 8

   0x1

   `LoopCount` = 16

   0x2

   `LoopCount` = 32

For each sub key element (8 bits, 16 bits, or 32 bits) of the key slot key, up to the key size, the following occurs while (`LoopCount` != 0):

1. Set the `BlindingMask` for the transaction.
2. If the destination requires a key blinding mask (the WMD bit in the KMUKSC<n> register is zero) and `MoreBlindingMasks`>0, the following occurs:
   a. Read from the PRBG a random `BlindingMask` value.
   b. Set the LS 16 bits of the `BlindingMask` value as the 16 user signals (HWUSER[15:0]).
3. Write the sub key element to the currently pointed remote cryptographic device key register, through the LWMF. The pointer to the remote cryptographic device key register starts at KMUDKPA.
   a. Read the 32 LS key bits from the temporary temporary 256-bit key slot register.
   b. Zeroize the unused key bits according to the destination port transaction width.

      This operation ANDs the current 32 bits key with one of the following values:

0xFFFF_FFFF
if DPDW value is 2 (32 bits)

0x0000_FFFF
if DPDW value is 1 (16 bits)

0x0000_00FF
if DPDW value is 0 (8 bits)

If the KMU does not support write delivery of less than 32 bits, then this AND operation is not implemented.

c. According to the destination port transaction width, add random bits to the zeroized key bits.

This operation ORs the ANDed key with one of the following values:

- `0x0000` OR with ANDed key [31:16] if DPDW value is 2 (32 bits)
- `BlindingMask` [31:16] OR with ANDed key [31:16] if DPDW value is 1 (16 bits)
- `BlindingMask` [31:16] OR with ANDed key [31:16], `BlindingMask` [31:24] OR with ANDed key [15:8] if DPDW value is 0 (8 bits)

If the KMU does not support write delivery of less than 32 bits, then this OR operation is not implemented.

d. Write the ANDed then ORed 32-bit key to the currently pointed remote cryptographic device key register.

e. Right shift the temporary 256-bit key slot register by the number of consumed bits per the destination port transaction width.

Number of right shifts per one of the following values:

32
if DPDW value is 2 (32 bits)

16
if DPDW value is 1 (16 bits)

8
if DPDW value is 0 (8 bits)

f. Advance the pointer to the remote cryptographic device key register according to the value in KMUKSC<n>.DPAI.

g. If the remote cryptographic device key register requires delay between consecutive writes (KMUKSC<n>.DPWD != 0), wait for the programmed number of clock cycles (KMUKSC<n>.DPWD).

h. Decrement the remaining number of writes (`LoopCount`).

i. If a new blinding mask for the next key writes is not required (KMUKSC<n>.NMNKW is 0), set `MoreBlindingMasks=0`

4. End while loop.

3.3 Local write mask filter

The KMU integrates a *Local write mask filter* (LWMF). The LWMF is responsible for masking the keys which are written to the destination devices.

The LWMF is positioned between the key exporter and the AHB5 manager port of the KMU. The key exporter provides to the LWMF:

- The key value at its HWDATA[31:0] input
- The mask value at its HWUSER[15:0] inputs. The key exporter also provides the mask value to the AHB5 manager port.

The LWMF provides the masked key at HWDATA[31:0] to the KMU AHB5 manager port by performing the following operations:

1. XOR the incoming HWDATA[n] with HWUSER[n] for n in the range of 0-15 to generate the outgoing HWDATA[n].
2. XOR the incoming HWDATA[n] with HWUSER[n-16] for n in the range of 16-31 to generate the outgoing HWDATA[n].

The LWMF also generates four parity bits on HWDATACHK signals for checking against the outgoing HWDATA[31:0] value.

3.4 Pseudorandom bit generator

The PRBG produces pseudorandom bits using algorithms which pass *NIST SP 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications* tests.

The PRBG does not provide cryptography worthy random values. It provides only masks to the key exporter as a countermeasure.

The following sections describe the operations of the PRBG:

- PRBG setup
- Read pseudorandom value from the PRBG

3.4.1 PRBG setup

The setup of the PRGB is performed by a secure software.

Use the following flow to set up the PRBG:

1. The PRBG seed is initialized by the writing of a non-repeating initial value to the PRBG seed input register. For example, this value can be based on the boot counter and a unique random value for each platform. The initialization of the PRBG seed causes:
    - The initialization of the PRBG internal state by the seed input register

- The automatic disabling of the PRBG engine to preserve power after the initialization of the seed.

2. The software writes a boot unique value in four writes to the KMUPRBGSI register.

   To lock the KMUPRBGSI register, the software sets the L_KMUPRBGSI bit in one of the KMUKSC<n> registers. This locked bit is then duplicated in all instances of the KMUKSC<n> registers. For more information about the KMUKSC<n> register, see 6.1.6 KMUKSC<n>, Key slot configuration register on page 38.

3. The PRBG is ready for use.

3.4.2 Read pseudorandom value from the PRBG

When the key exporter or the random delay registers require a 32-bit pseudorandom value, it performs the following steps:

1. Enables the PRBG
2. Reads a value from the PRBG
   - The PRBG generates a new pseudo random value as a result of the read
3. Disables the PRBG to save power

3.5 Remote write mask filter

The KMU architecture provides an optional *remote write mask filter* (RWMF). If write masking is supported, the RWMF should to be integrated as a filter on the path on the interconnect remote cryptographic device.

The RWMF is responsible for unmasking the keys which are written to the destination devices and uses HWUSER[15:0]) as a mask value. All managers that access this device and typically do not mask their data, provide 16 user bits (HWUSER[15:0]) of zeroes. Even if the RWMF is integrated, write masking can be turned off by disabling key masking in the key slot entry. When disabled, this key masking behaves as a bypass for writes because the KMU drives HWUSER[15:0] to zeroes. The RWMF is also responsible for checking the parity of the incoming masked data and of any other incoming data if the subsystem supports parity.

To use the RWMF as a good countermeasure peripheral, you must integrate it as close as possible to the programmer's interface of the target cryptographic device.

The following steps describe the flow for unmasking the key fragments at the remote side of the key transfer.

For all key fragment transfers, the RWMF:

1. XORs the following to get the unmasked PWDATA[31:0]:
   - input data PWDATA[15:0] with the APB Write User signal mask (PWUSER[15:0])
   - input data PWDATA[31:16] with the APB Write User signal mask (PWUSER[15:0])

2. Computes parity on its input PWDATA[31:0] and compares its result with the incoming parity signals. For a parity error, it sets the parity error signal. This signal is typically connected to alarm input event in the Security Alarm Manager (SAM).

The KMU_WMF_PRESENT parameter in Configuration options specifies if the RWMF is enabled.

3.6 Key store device

When a cryptographic engine does not provide a compatible subordinate bus interface, the system integrator must provide a key store device to allow the key to be delivered as a plain multibit signal.

The KMU exports its key by using write accesses through its AHB interface. However not all cryptographic engines are able to provide a compatible subordinate bus interface that allows the required keys to be deposited in the way the KMU expects.

For example, a 64-bit key store device can be used to support the delivery of a PKA memory encryption key through 64-bit signals to a cryptographic device that does not have an APB interface to perform PKA memory key encryption.

The minimal key size that the KMU supports is of 128 bits. To align with this KMU limitation, the 64-bit key store device receives four 32-bit write transactions, so that:

- The first two writes are latched
- The next two writes are ignored, therefore their data is not saved in a register

Therefore, the KMU writes the four 32-bit transactions to the key store device's internal byte address offsets `0x00` to `0x0F`. However, only address offsets `0x00` – `0x07` (64 bits) are latched in an internal key register.

The device can also write the key into a target cryptographic engine that uses a different address scheme or a signaling protocol that the KMU does not support.

3.7 Hardware interfaces

The KMU has a number of hardware interfaces.

APB4 subordinate programming port

The subsystem software uses the APB4 subordinate programming port to control the KMU operation.

Private APB3 subordinate hardware keys port

The Lifecycle Manager (LCM) uses the private APB3 subordinate hardware keys port to load hardware keys to the KMU.

AHB5 interface

The KMU uses the AHB5 interface to export keys to target cryptographic devices.

Parity error interface

The KMU uses the parity error interface to generate an alarm when the KMU detects an internal parity error.

Interrupt interface

The KMU uses the interrupt interface to detect programming error or key export error.

Scan interface

The KMU uses the scan interface to reset its registers when the kmu_dftscanmode signal transitions to 1 or 0.

3.7.1 APB4 subordinate programming port

The KMU includes an APB4 subordinate programming port used by the subsystem software to control the KMU operation and read its status.

The APB4 subordinate programming port does not filter transactions according to their security attributes (secure or non-secure) or their privilege level (privileged, non-privileged) attributes. Therefore, we recommend that the KMU APB4 subordinate programming port is accessible only to secure software. We also recommend that the KMU APB4 subordinate programming port is integrated with an Arm APB4 TrustZone® *Peripheral Protection Controller* (PPC) for that purpose. For more information about the APB4 TrustZone Peripheral Protection Controller, see the *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual*.

The Programmers model section details the memory map and registers exposed in the APB4 subordinate programming port.

3.7.2 Private APB3 subordinate hardware keys port

The KMU includes a private APB3 subordinate hardware keys port used by an external RoT unit, for example an LCM, to load hardware keys to the hardware key slots of the KMU.

The private APB3 subordinate hardware keys port connects to the external RoT unit in a point-to-point way through an optional RWMF and a Parity Generator. This ensures that the transferred keys are not accessible to software or any other hardware.

The address space that is exposed in this private APB3 subordinate hardware keys port, `0x0000_0000` to `0x0000_00FF`, is designed to match the external RoT unit hardware keys export implementation.

The only registers in the KMU which are accessible through this port, when KMUNHWKSLTS is not 0, are those belonging to the hardware key slots from KMUKSK<0><m> to KMUKSK<KMUNHWKSLTS-1><m>.

The following table lists the routing of the hardware keys written by the LCM at the Hardware keys port to the KMU key registers.

Table 3-1: Routing of writes into the Hardware keys port to key registers

Key ID	Hardware keys port address offset (in bytes)	Routed to KMU key register	Comment
0	`0x00..0x1F`	KMUKSK0<m>	If no hardware key slot is supported (KMUNHWKSLTS is 0), then a write to address offset `0x00..0x1F` results in a bus error.
1	`0x20..0x3F`	KMUKSK1<m>	If fewer than 2 hardware key slots are supported (KMUNHWKSLTS < 2), then a write to address offset `0x20..0x3F` results in a bus error.
2	`0x40..0x5F`	KMUKSK2<m>	If fewer than 3 hardware key slots are supported (KMUNHWKSLTS < 3), then a write to address offset `0x40..0x5F` results in a bus error.
3	`0x60..0x7F`	KMUKSK3<m>	If fewer than 4 hardware key slots are supported (KMUNHWKSLTS < 4), then a write to address offset `0x60..0x7F` results in a bus error.
4	`0x80..0x9F`	KMUKSK4<m>	If fewer than 5 hardware key slots are supported (KMUNHWKSLTS < 5), then a write to address offset `0x80..0x9F` results in a bus error.
5	`0xA0..0xBF`	KMUKSK5<m>	If fewer than 6 hardware key slots are supported (KMUNHWKSLTS < 6), then a write to address offset `0xA0..0xBF` results in a bus error.
6	`0xC0..0xDF`	KMUKSK6<m>	If fewer than 7 hardware key slots are supported (KMUNHWKSLTS < 7), then a write to address offset `0xC0..0xDF` results in a bus error.
7	`0xE0..0xFF`	KMUKSK7<m>	If fewer than 8 hardware key slots are supported (KMUNHWKSLTS < 8), then a write to address offset `0xE0..0xFF` results in a bus error.

3.7.3 AHB5 interface

The KMU includes an AHB5 interface, which exports keys to the target cryptographic devices.

The AHB5 interface optionally supports the use of 16 user bits (HWUSER) to send the mask value associated with the high and low 16 bits of each exported 32-bit key word. Four parity bits (HWDATACHK) for the masked key can also be generated. The transaction attributes that the KMU provides through this interface are Secure privileged.

3.7.4 Parity error interface

The KMU includes a parity error interface.

We recommend using the kmu_parerr signal to raise a security alarm or interrupt, for example by connecting this signal to an entity in the subsystem. The entity can be the IRQ of the subsystem, or an alarm handling unit, such as the SAM.

For more information about the SAM, see the *Arm® Security Alarm Manager Specification*.

The entity can:

- Raise an alarm to the Secure world
- Allow this alarm to be handled according to the security requirements of the subsystem

3.7.5 Interrupt interface

The KMU includes an interrupt interface with a single output signal, kmu_irq, which interrupts the software on successful completion of key export, on detection of a programming error, or an error detected during a key export.

We recommend connecting this interface signal to an IRQ of the subsystem.

3.7.6 Scan interface

The KMU includes a scan interface with a kmu_dftscanmode input signal.

The KMU uses the scan interface to reset its registers when the kmu_dftscanmode signal transitions to 1, or when it transitions to 0.

4. Configuration options

The KMU provides configuration options to configure its features and components.

The following table describes the configuration options.

Table 4-1: KMU configuration options

Configuration option name	Size in bits	Legal values	Description
KMUNKS	3	1,2,3,4,5	Defines the number of key slots that the KMU supports. This value is reflected in KMUBC.NKS bits. Supported values: **0x1** 2 key slots **0x2** 4 key slots **0x3** 8 key slots **0x4** 16 key slots **0x5** 32 key slots (default)
KMUTANG	16	4 sets of four-bit values	Specifies four possible values for the top address nibble groups of the target cryptographic devices that the KMU allows software to set in the KMUDKPA<n> registers. A value of zero means that this nibble value is not implemented. All key export regions must be located in regions with top nibble of the address defined by KMUTANG, except for the value zero. This configuration option supports up to four top address nibble groups: **[3:0]** Top nibble of the first target cryptographic device addresses group. **[7:4]** Top nibble of the second target cryptographic device addresses group. **[11:8]** Top nibble of the third target cryptographic device addresses group. **[15:12]** Top nibble of the fourth target cryptographic device addresses group. The top nibble writes to the KMUDKPA<n> registers are compared to all the enabled (nonzero) KMUTANG nibble parameters. In the case of no match to all groups, the write is ignored, and alarm signal is set.
KMUNHWKSLTS	4	0-8	Number of hardware key slots. Zero indicates that there are no hardware key slots. The first KMUNHWKSLTS key slots are filled by hardware through the private APB3 subordinate interface, and the other slots are software key slots. The KMUNHWKSLTS value is reflected in KMUBC.NHWKSLTS bits.

Configuration option name	Size in bits	Legal values	Description
KMU_WMF_PRESENT	1	0,1	Specifies whether the RWMF component is enabled. **0** Write masking is disabled. Forces WMD bits of all KMUKSC <n> registers to `0b1` and **RAO/WI**, overriding bit 21 of all KMUKSCRV<n> configuration options. **1** Write masking is enabled. The value of WMD bits for each KMUKSC<n> register is RW if not locked using LKS bit and is determined by the corresponding KMUKSCRV<n> configuration options.
KMUKSCRV0	32	Any	Reset value of the KMUKSC<0> register. If no hardware key slot is supported (KMUNHWKSLTS is 0), this configuration option must be set to `0x0000_0000`.
KMUKSCRV<n>	32	Any	Reset value of the KMUKSC<n> register. If fewer than n+1 hardware key slots are supported (KMUNHWKSLTS < n+1), this configuration option must be set to `0x0000_0000`. (n= 1..7)
KMUDKPARV0	32	Any	Reset value of the KMUDKPA<0> register. If no hardware key slot is supported (KMUNHWKSLTS is 0), this configuration option must be set to `0x0000_0000`.
KMUDKPARV<n>	32	Any	Reset value of the KMUDKPA<n> register. If fewer than n+1 hardware key slots are supported (KMUNHWKSLTS is < n+1), this configuration option must be set to `0x0000_0000`. (n= 1..7)

5. KMU integration

You can integrate the KMU through key-slot allocations, key export transactions, and different signals.

5.1 Key slots allocation example

The first key slots (from 0 to KMUNHWKSLTS-1) are hardware key slots. Software can only initiate exporting these key slots or invalidate them.

Invalidation of key slots occurs when higher-privilege software wants to prevent lower-privilege software from using them. Hardware key slot allocation is defined by the order an external entity writes the hardware keys to the KMU hardware key slots.

For external cryptographic devices that are integrated with the KMU, at least one software key slot for each cryptographic device should be used to set its keys. The keys can be set either by the KMU writing directly to the external cryptographic device, or through a key store device.

For example, you can use a key slot to set a PKA memory encryption key in the key store device, which is then connected to an external cryptographic device.

After the external cryptographic device returns from a low-power state, you must set the PKA memory encryption key again, by using one of the following options:

- Use the same key. In this case, the key slot is preserved in the KMU.
- Set to a new key. After the software exports the key to the key store device, this key slot can be invalidated and be used for any other purpose. When a new key initialization is required again for the same cryptographic device, another unused key slot can be used instead.

All other key slots for repeated use cases can be allocated by software and not be invalidated.

5.2 Security attributes of key-export transactions

The KMU exports the keys to its destination cryptographic devices.

To export the keys, the KMU uses Write AHB5 manager transactions with the following properties:

- HPROT[3:0] signals set to:
  - Data access
  - Privileged Access
  - Non-bufferable
  - Non-cacheable
- HNONSEC signal, set to 0 to denote a Secure transaction

- HPROT[6:4] set to `0b000`
- Parity signals and 16 mask bits (if configured) are transferred as part of User signals HWUSER

5.3 KMU signals

This section describes the signals of the KMU.

Signal definitions

Table 5-1: KMU signals

Signal	Name	Direction	Description
kmu_parerr	Parity error signal	Output	Connect the signal as an input to raise a security alarm. Connect the signal to the IRQ of the subsystem to raise an interrupt. It can also be routed to an alarm handling unit, such as the SAM. A parity error is indicated when any of the KMU registers has bad parity, or if the key exporter reads its setup from any of the registers with a parity error.
kmu_irq	Interrupt signal	Output	It allows software to know that the key export operation has completed. Connect signal to the IRQ of the subsystem.
kmu_dftscanmode	KMU DFT scan mode signal	Input	The integrator must connect this signal to the dftscanmode signal of the system to provide a notification to the KMU that the DFT scan has started. It allows the KMU to reset its registers and wipe keys at the start and at the end of the DFT scan. The flops generating the internal reset, which clears the KMU registers on kmu_dftscanmode input transitions, must be excluded from DFT scan chain.

6. Programmers model

This chapter provides general information about the KMU register properties.

The following information applies to the KMU registers:

- The base address is not fixed and can be different for any system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in unexpected behavior.
- Unless otherwise stated in the accompanying text:
  - Do not modify undefined register bits.
  - Ignore undefined register bits on reads.
  - All register bits are reset to the reset value specified in the register summary table of each block.

Access type is described as follows:

RW
: Read/write

RO
: Read-only

WO
: Write-only

WI
: Write-ignore

RAZ/WI
: Read as zero, write ignore

RAZW1C
: Read as zero, write 1 to clear the respective bit in the register

RW1S
: Read, write 1 to set this bit

The KMU must be accessible only to Secure software.

The KMU does not perform its own checks and does not filter access based on its security. Therefore, when integrating the KMU, the system must filter the access on behalf of the KMU. Arm provides a compatible *Peripheral Protection Controller* (PPC) that can perform this task. For more information about the PPC, see the *Arm® CoreLink™ SIE-200 System IP for Embedded Technical Reference Manual*.

We recommend that all KMU registers reside in the always-on power domain and are reset by the system Cold reset. In addition, depending on the Secure provisioning flow of the SoC, the KMU might also require that its reset is sensitive to Cold reset during Secure provisioning.

Just after the deassertion of KMU reset input, the KMU blocks access to itself until the reset process is complete and it is ready for access. The KMU blocks access to itself by pulling the pready signals LOW on all APB subordinate interfaces.

6.1 KMU register summary

The KMU register summary table lists the registers in the KMU register block, where n is 0 to (KMUBC.KSC - 1).

Table 6-1: KMU registers

Offset	Name	Type	Reset	Width	Description
0x000	KMUBC	RO	0x0000_0000	32-bit	KMU Build Configuration register. This register reflects the build configuration for software usage. Its reset value depends on the selected configuration option in Configuration options.
0x004	KMUIS	RO	0x0000_0000	32-bit	KMU Interrupt Status register. This register indicates the current interrupt events.
0x008	KMUIE	RW	0x0000_0000	32-bit	KMU Interrupt Enable register. This register is used to enable the corresponding interrupt events.
0x00C	KMUIC	RW	0x0000_0000	32-bit	KMU Interrupt Clear register. This register is used to clear the corresponding interrupt events.
0x010	KMUPRBGSI	RW	0x0000_0000	32-bit	PRBG Seed Input register. Software writes the PRBG initial seed through this register.
0x030	KMUKSC<n>	RW	0x0000_0000	32-bit	KMU key slot configuration n register. This register holds the configuration for slot n.
0x0B0	KMUDKPA<n>	RW	0x0000_0000	32-bit	KMU Destination Key Port Address n Register. This register holds the destination key register address to write the key from slot n.
0x130	KMUKSC<0><n>	Unlocked: RW Locked: **RAZ/WI**	0x0000_0000	32-bit	Key slot word 0 of key slot <0>.
0x150	KMUKSK<1><m>	Unlocked: RW Locked: **RAZ/WI**	0x0000_0000	32-bit	Key slot word 1 of key slot <1>.
0x170	KMUKSK<2><m>	Unlocked: RW Locked: **RAZ/WI**	0x0000_0000	32-bit	Key slot word 2 of key slot <2>.
0x190	KMUKSK<3><m>	Unlocked: RW Locked: **RAZ/WI**	0x0000_0000	32-bit	Key slot word 3 of key slot <3>.
0x1B0	KMUKSK<4><m>	Unlocked: RW Locked: **RAZ/WI**	0x0000_0000	32-bit	Key slot word 4 of key slot <4>.

Offset	Name	Type	Reset	Width	Description
0x1D0	KMUKSK<5><m>	Unlocked: RW Locked: **RAZ/WI**	0x0000_0000	32-bit	Key slot word 5 of key slot <5>.
0x1F0	KMUKSK<6><m>	Unlocked: RW Locked: **RAZ/WI**	0x0000_0000	32-bit	Key slot word 6 of key slot <6>.
0x210	KMUKSK<7><m>	Unlocked: RW Locked: **RAZ/WI**	0x0000_0000	32-bit	Key slot words 7 of key slot <7>.
0x230-0x52C	KMUKSK<8..31><m>	Unlocked: RW Locked: **RAZ/WI**	0x0000_0000	32-bit	Key slot word 8 of key slot <8..31>.
0x530	KMURD_8	RO	0x0000_0000	32-bit	When this register is read, it provides a random delay of 0-7 clocks.
0x534	KMURD_16	RO	0x0000_0000	32-bit	When this register is read, it provides a random delay of 0-15 clocks.
0x538	KMURD_32	RO	0x0000_0000	32-bit	When this register is read, it provides a random delay of 0-31 clocks.
0x53C – 0xFCC	Reserved	**RAZ/WI**	0x0000_0000	32-bit	Reserved
0xFD0	PIDR4	RO	0x0000_0004	32-bit	The Peripheral ID4 register. It returns byte[4] of the peripheral ID
0xFD4 – 0xFDC	Reserved	**RAZ/WI**	0x0000_0000	32-bit	Reserved
0xFE0	PIDR0	RO	0x0000_00F3	32-bit	The Peripheral ID0 register. It returns byte[0] of the peripheral ID.
0xFE4	PIDR1	RO	0x0000_00B0	32-bit	The Peripheral ID1 register. It returns byte[1] of the peripheral ID.
0xFE8	PIDR2	RO	0x0000_001B	32-bit	The Peripheral ID2 register. It returns byte[2] of the peripheral ID.
0xFEC	PIDR3	RO	0x0000_0000	32-bit	The Peripheral ID3 register. It returns byte[3] of the peripheral ID.
0xFF0	CIDR0	RO	0x0000_000D	32-bit	The Component ID0 register. It returns byte[0] of the component ID.
0xFF4	CIDR1	RO	0x0000_00F0	32-bit	The Component ID1 register. It returns byte[1] of the component ID.
0xFF8	CIDR2	RO	0x0000_0005	32-bit	The Component ID2 register. It returns byte[2] of the component ID.
0xFFC	CIDR3	RO	0x0000_00B1	32-bit	The Component ID3 register. It returns byte[3] of the component ID.

6.1.1 KMUBC, Build Configuration register

The KMUBC register reflects the build configuration of the software.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x000

Type

RO

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-2: KMUBC register bit description

Bits	Name	Description	Type	Reset
[31:23]	-	Reserved	RAZ/WI	0x0
[22:19]	NHWKSLTS	Number of existing hardware key slots. Supports values from 0-8. Zero indicates that there no hardware key slots. The first NHWKSLTS key slots are filled by hardware (typically by the LCM) and the rest are software key slots. This field reflects the KMUNHWKSLTS configuration option. See Configuration options.	RO	KMUNHWKSLTS
[18:16]	NKS	Specifies the total number of implemented key slots, including the hardware key slots. Supported values are defined by KMUNKS in Configuration options.	RO	KMUNKS
[15:0]	KMUTANG	Reflects the value of the KMUTANG configuration option. See Configuration options.	RO	KMUTANG

6.1.2 KMUIS, Interrupt Status register

The KMUIS register indicates the status of the current interrupt events.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x004

Type

RO

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-3: KMUIS register bit description

Bits	Name	Description	Type	Reset
[31:15]	-	Reserved	RAZ/WI	0x0
[14]	WDALSBDKPA	Wrong Destination Address LS Bits Detected. Indicates that a write to one of the KMUDKPA<n> registers used a destination address which is not word-aligned.	RO	0b0
[13]	AWBHKSKR	Indicates an attempt to write from the private APB3 subordinate hardware keys port beyond the hardware key slot key registers.	RO	0b0
[12]	AKSWPI	Indicates a write attempt to a key slot word register which is permanently invalidated. This bit is set if the external RoT unit (typically the LCM) attempts to write to a KMUKSK<n><m> register when it is permanently invalidated.	RO	0b0
[11]	MWKSW	Multiple write attempts to the key slot word register. This bit is set if software or hardware (typically the LCM) attempts to write more than one time to a KMUKSK<n><m> register.	RO	0b0
[10]	AIKSWE	Indicates an attempt to invalidate a key slot while exporting.	RO	0b0
[9]	KSDPANS	Indicates that the key slot destination port address register (KMUDKPA<n>) is not set.	RO	0b0
[8]	KSKRSM	Key slot key registers size mismatch. That is, there is a mismatch between the number of KMUKSK registers written versus the values of the KMUKSC<n>.NDPW and KMUKSC<n>.DPDW bits.	RO	0b0
[7]	KSNL	Key slot is not locked because either the KMUKSC<n>.LKSKR or the KMUKSC<n>.LKS bits are not set.	RO	0b0
[6]	AEWNR	Attempted to Export While Not Ready. This status bit is set when software attempts to export a key when it is not ready to be exported (KMUKSC<n>.KSR bit is not set).	RO	0b0
[5]	WTE	Write Transaction Error. Reported BusFault (HRESP) while the key exporter writes to the pointed cryptographic device.	RO	0b0
[4]	WDADDKPA	Wrong Destination Address Detected. Write to one of KMUDKPA<n> registers has used a destination address which is not supported by any of the KMUTANG configuration options.	RO	0b0
[3]	INPPE	Parity Error has been detected at the PRBG. When this bit is set, a kmu_parerr signal is set. This signal is connected to the SAM.	RO	0b0

Bits	Name	Description	Type	Reset
[2]	IPE	Parity Error has been detected in the register file. When this bit is set, a kmu_parerr signal is set. This signal is connected to the SAM.	RO	0b0
[1]	AWBE	Activation While Busy Error. Indicates an attempt to activate a key export while the key exporter is busy with current exporting task, for example, before the KEC status is set. This error does not stop or disturb the key exporter operation.	RO	0b0
[0]	KEC	The key export completed its operation.	RO	0b0

6.1.3 KMUIE, Interrupt Enable register

The KMUIE register enables or disables the corresponding interrupt events. When the interrupt enable field is set and the corresponding interrupt event in the status register is set, the coalesced interrupt output is asserted.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x008

Type

RW

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-4: KMUIE register bit description

Bits	Name	Description	Type	Reset
[31:15]	-	Reserved	RAZ/WI	0x0
[14]	WDALSBDKPA	Enable or disable the WDALSBDKPA interrupt	RW	0b0
[13]	AWBHKSKR	Enable or disable the AWBHKSKR interrupt	RW	0x0
[12]	AKSWPI	Enable or disable the AKSWPI interrupt	RW	0x0
[11]	MWKSW	Enable or disable the MWKSW interrupt	RW	0x0
[10]	AIKSWE	Enable or disable the AIKSWE interrupt	RW	0x0
[9]	KSDPANS	Enable or disable the KSDPANS interrupt	RW	0x0
[8]	KSKRSM	Enable or disable the KSKRSM interrupt	RW	0x0

Bits	Name	Description	Type	Reset
[7]	KSNL	Enable or disable the KSNL interrupt	RW	0x0
[6]	AEWNR	Enable or disable the AEWNR interrupt	RW	0x0
[5]	WTE	Enable or disable the WTE interrupt	RW	0x0
[4]	WDADDKPA	Enable or disable the WDADDKPA interrupt	RW	0b0
[3]	INPPE	Enable or disable the INPPE interrupt	RW	0b0
[2]	IPE	Enable or disable the IPE interrupt	RW	0b0
[1]	AWBE	Enable or disable the AWBE interrupt	RW	0b0
[0]	KEC	Enable or disable the KEC interrupt	RW	0b0

6.1.4 KMUIC, Interrupt Clear register

The KMUIC register clears the corresponding interrupt events.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x00C

Type

RW

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-5: KMUIC register bit description

Bits	Name	Description	Type	Reset
[31:15]	-	Reserved	**RAZ/WI**	0x0
[14]	WDALSBDKPA	When written as 1, clears the WDALSBDKPA interrupt status in the KMUIS register.	**RAZ**W1C	0b0
[13]	AWBHKSKR	When written as 1, clears the AWBHKSKR interrupt status in the KMUIS register.	**RAZ**W1C	0b0
[12]	AKSWPI	When written as 1, clears the AKSWPI interrupt status in the KMUIS register.	**RAZ**W1C	0b0
[11]	MWKSW	When written as 1, clears the MWKSW interrupt status in the KMUIS register.	**RAZ**W1C	0b0
[10]	AIKSWE	When written as 1, clears the AIKSWE interrupt status in the KMUIS register.	**RAZ**W1C	0b0
[9]	KSDPANS	When written as 1, clears the KSDPANS interrupt status in the KMUIS register.	**RAZ**W1C	0b0

Bits	Name	Description	Type	Reset
[8]	KSKRSM	When written as 1, clears the KSKRSM interrupt status in the KMUIS register.	RAZW1C	0b0
[7]	KSNL	When written as 1, clears the KSNL interrupt status in the KMUIS register.	RAZW1C	0b0
[6]	AEWNR	When written as 1, clears the AEWNR interrupt status in the KMUIS register.	RAZW1C	0b0
[5]	WTE	When written as 1, clears the WTE interrupt status in the KMUIS register.	RAZW1C	0b0
[4]	WDADDKPA	When written as 1, clears the WDADDKPA interrupt status in the KMUIS register.	RAZW1C	0b0
[3]	INPPE	When written as 1, clears the INPPE interrupt status in the KMUIS register.	RAZW1C	0b0
[2]	IPE	When written as 1, clears the IPE interrupt status in the KMUIS register.	RAZW1C	0b0
[1]	AWBE	When written as 1, clears the AWBE interrupt status in the KMUIS register.	RAZW1C	0b0
[0]	KEC	When written as 1, clears the KEC interrupt status in the KMUIS register.	RAZW1C	0b0

6.1.5 KMUPRBGSI, Seed Input register

The Secure software uses the KMUPRBGSI register to set the PRBG state machine with an initial seed.

The Secure software must write four times into this register to initialize the entire state of the PRBG engine.

Any write to this register is XORed with the current state of the PRBG and then added to the state so that entropy can be added continuously.

This register is read/write while it is not locked (by the KMUKSC<n>.L_KMUPRBGSI bit), otherwise this register is RAZ/WI.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x010

Type

RW

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-6: KMUPRBGSI register bit description

Bits	Name	Description	Type	Reset
[31:0]	KMUPRBGSI	Secure software initializes the state of the PRBG engine through this field.	RW	0x0000_0000

6.1.6 KMUKSC<n>, Key slot configuration register

The Secure software uses the KMUKSC<n> register to control key slot configuration <n> of the KMU.

Configurations

The number of actual supported key slots (KMUNKS) is a configuration option. See Configuration options.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x030

Type

RW

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-7: KMUKSC<n> register bit description

Bits	Name	Description	Type	Reset
[31]	L_KMUPRBGSI	Allows Secure software to lock the KMUPRBGSI register from further writes. This security capability prevents malicious software from harming the functionality of the PRBG in the KMU. This lock bit is duplicated in all instances of the KMUKSC<n> registers. Setting the bit in any of them locks the KMUPRBGSI register.	RW1S	0b0
[30:29]	-	Reserved	RAZ/WI	0x0

Bits	Name	Description	Type	Reset
[28]	EK	Export a key to the destination of its key-slot. This bit is not locked by the write of the LKS bit. Setting this bit to 1, triggers the key exporter to export the key using the export scheme defined in the KMUKSC<n> and KMUDKPA<n> registers. Setting this bit to 0 after the software has already initiated RW1S by setting this bit to 1 does not stop the key exporter. Reading this bit after the key exporter was triggered returns 1. When the export of the key completes, this bit is cleared by hardware. In other words, when the key exporter is exporting a key, this bit can be used as a busy status. The key is not exported and an AEWNR interrupt bit is set in the KMUIS register if: • Its key slot is not ready (KSR bit is 0) • Its key slot is invalidated permanently (KSIP bit is set)	RW1S	0b0
[27]	KSIP	Invalidated the key slot permanently. This status bit is set by the KMU when software invalidates the hardware key slot by setting the KMUKSC<n>.IKS bit. When this bit is set, the respective KMUKSK<n><m> registers are not writable through the private APB3 subordinate hardware keys port.	RO	0b0
[26]	IKS	Invalidate the key slot temporarily. When the software sets this bit: • KMUKSK<n><0..7> registers of the key slot are zeroized • The respective KSK_NE<n><m> flags are zeroized • The KMUDKPA<n> register is zeroized • The KSA_NE<n> flag is cleared • The KMUKSC<n>.LKS control bits are zeroized • The KMUKSC<n>.LKSKR control bits are zeroized • The KMUKSC<n>.KSR control bits are zeroized At the completion of the invalidation, This bit is also cleared. Setting this bit makes its key slot useless and ready to be reprogrammed to store another key. An attempt to invalidate a key slot when the KMU exports a key slot is ignored and an AIKSWE interrupt bit is set in the KMUIS register. When this bit is set in the hardware key slot, its KMUKSC<n>.KSIP bit is automatically set by the KMU. The KMUKSC<n>.LKS and KMUKSC<n>.LKSKR bits are not reset because their default hardware values remain set to 1. This action zeroizes the hardware key as any other key slot and prevents further use of this key. It is a security feature which allows early-stage boot trusted software to use a hardware key and then prevent a less trusted, loaded software from using it. For more information about the KSA_NE<n> flag, see KMUKSK<n><m>.	RW1S	0b0
[25]	KSR	Key slot ready. The software can check this bit to make sure the key slot is ready to use. When a key slot is invalidated, its KSR bit becomes 0. KSR is set only after setting the VKS bit completes successfully.	RO	0b0

Bits	Name	Description	Type	Reset
[24]	VKS	Verify key slot. The software sets this bit to trigger the KMU to verify that the key slot is set consistently and is ready to be used. When this bit is set, the KMU checks the following: • The key slot must be set, that is KMUKSC<n>.LKSKR and KMUKSC<n>.LKS bits are set. If one of the lock bits is not set, the KMU sets the KMUIS.KSNL bit. • The key size must be either 128 bits or 256 bits, according to the values in KMUKSC<n>.NDPW and in KMUKSC<n>.DPDW. If there is a key size mismatch, the KMU sets the KMUIS.KSKRSM bit. • All required key slot key registers must be written, according to the expected key size. The KMU checks this by verifying whether their respective KSK_NE<n><m> flags are set. If there is a key size mismatch, the KMU sets KMUIS.KSKRSM. • Its KSA_NE<n> flag must be set. If the KMUDKPA<n> is not set, the KMU sets the KMUIS.KSDPANS bit. If all checks pass, the KSR bit is set. When key slot verification completes, KMU clears VKS.	RW1S	0b0
[23]	LKSKR	Lock key slot key registers. When set to 1, it locks KMUKSK<n><0..7> registers for reads and writes. This is a security feature.	RW1S	0b0
[22]	LKS	Lock key slot controls for further writes. This security capability prevents malicious software from harming the setup of key slot <n>. It locks the KMUDKPA<n> register and the following KMUKSC<n> register bits for writes: • DPWD • DPAI • DPDW • NDPW • NMNKW • WMD	RW1S	0b0
[21]	WMD	Write mask disable. This control bit tells the key exporter if the RWMF is physically close to the destination cryptographic device. Secure software should only disable masking functionality if there is no RWMF in this destination. Supported values: **0** Write masking is enabled **1** Write masking is disabled When the KMU_WMF_PRESENT bit is 0, this bit is RA0WI and its default is 0.	RW	0b0

Bits	Name	Description	Type	Reset
[20]	NMNKW	New mask for next key writes. This control bit tells the key exporter if it should change the mask before each key write to the destination cryptographic device. This option is used when it is safe to use the mask of the first write for additional writes to the same destination. Using this option allows faster key export. **Note:** • Even if this control bit is not set, the key exporter gets a new mask for the first write it makes. • The priority to support this bit set is medium. • For security reasons it is highly recommended to set this bit to 0. Supported values: **0** New mask is required **1** New mask is not required	RW	0b0
[19:18]	NDPW	Number destination port writes code. This sets the requirements that define the number of key fragment write transfers to the destination cryptocgraphic device that are necessary. Key size (in bits) at the destination device is $2^{2+NDPW} * 2^{3+DPDW}$. Key size must be a multiple of the number of bytes and must be either 128 bits or 256 bits. Supported values per code: **0x0** 4 writes **0x1** 8 writes **0x2** 16 writes **0x3** 32 writes	RW	0b00
[17:16]	DPDW	Destination port data width code. It customizes the key exporter to the needs of the destination cryptographic device. Supported values per code: **0x0** 8 bits **0x1** 16 bits (low priority) **0x2** 32 bits **0x3** Reserved	RW	0b00

Bits	Name	Description	Type	Reset
[15:8]	DPAI	Destination port address increment. It customizes the key exporter to advance the destination address for exporting the next key value according to the needs of the destination cryptographic device. The values in this bit are in words. Supported values: • `0x00` – 0 bytes. This option is used when the key data must be injected through a single MMIO address with no increments. • `0x01` – 4 bytes. This option is used when the key data, according to its selected width, must be written to consecutive MMIO word addresses. • `0x02` – 8 bytes • `0x03` – 12 bytes • `0x04` – 16 bytes • ... • `0x80` – 512 bytes • `0x81` - `0xFF` – Reserved	RW	`0b00`
[7:0]	DPWD	Destination port write delay in clocks. It customizes the key exporter to the needs of the destination cryptographic device. Supported values: 0-255.	RW	`0x00`

6.1.7 KMUDKPA<n>, destination key port address n register

Software uses the KMUDKPA<n> register to set the destination key port address for writes from a key slot <n> of the KMU. If the destination key is stored in registers spanning contiguous destination addresses, the destination key port is the address of the first register.

The top nibble of the address written to this register is compared to one of the four enabled (nonzero) nibbles in the KMUTANG configuration option. If it does not match any of the supported options, the write is discarded and the WDADDKPA status bit is set in the KMUIS register.

The two LS bits of the address written to this register must be `2b00` as the destination address must be word-aligned. If the written address violates this rule, the write is discarded and the WDALSBDKPA status bit is set in the KMUIS register.

When KMUDKPA<n> is written successfully by software, the KMU sets a KSA_NE<n> flag to indicate that the Key Slot Address register <n> associated with the written register is not empty. This flag is set by default for the hardware key slots.

Configurations

The number of actual supported key slots (KMUNKS) is a configuration option, see Configuration options.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x0B0

Type

- RW when it is not locked by the KMUKSC<n>.LKS bit
- RWI when is locked by the KMUKSC<n>.LKS bit
- **RAZ/WI** when unimplemented key slot registers are implemented

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-8: KMUDKPA<n> register bit description

Bits	Name	Description	Type	Default
[31:2]	DKPA_31_2	Destination Key Port address bits [31:2]	RW	0x0000_0000
[1:0]	DKPA_1_0	LS bits of the Destination Key Port address. These two bits must be written as 0s.	RAZ/WI	2b00

6.1.8 KMUKSK<n><m>, KMU key slot key registers

Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

The key values for key slot <n> must be populated from KMUKSK<n><0> to KMUKSK<n><7> registers. 128-bits key value for key slot <n> must be populated from KMUKSK<n><0> to KMUKSK<n><3> registers. 256-bits key value for key slot <n> must be populated from KMUKSK<n><0> to KMUKSK<n><7> registers.

When software or hardware uses the private APB3 subordinate hardware keys port to write to the KMUKSK<n><m> registers, the KMU sets a KSK_NE<n><m> flag associated with the written register to indicate the KMU key slot key <n><m> register is not empty.

The KSK_NE<n><m> flag is an internal flag bit set by the system inside the KMU. It is used when software commands the KMU to verify the key slot by setting the KMUKSC<n>.VKS bit.

The reset value of the KSK_NE<n><m> flags is 0. The flags that belong to a key slot are cleared when software invalidates the key slot by setting the KMUKSC<n>.IKS bit. An attempt to write

again over a KMUKSK<n><m> register which has KSK_NE<n><m> flag set is ignored. It also results in a bus error response and the setting of the MWKSW bit in the KMUIS register.

An attempt to write to a KMUKSK<n><m> register when the KMUKSC<n>.KSIP bit is set is ignored. It also results in a bus error response and the setting of the AKSWPI bit in the KMUIS register.

An attempt to write from the private APB3 subordinate hardware keys port to a KMUKSK<n><m> register which is not part of the hardware key slot key registers is ignored. It also results in a bus error response and the setting of the AWBHKSKR bit in the KMUIS register.

A software write to the KMUKSK<n><m> register, which is part of the hardware key slot key registers, is prevented as its respective KMUKSC<n>.LKSKR bit is set as hardware default.

For reliability and security, the software writes to key slot key register and reads to verify. For a mismatch, if the software wants to rewrite this register, it must first invalidate the key slot and reprogram it entirely from start. Otherwise, a rewrite results in a bus error and the setting of the MWKSW bit in the KMUIS register.

6.1.8.1 KMUKSK<0><m>, KMU key slot key <0><m> register

Key slot n=0 of the KMUKSK<n><m>. Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

`0x130`

Type

- RW
- **RAZ/WI** when key slot register is unimplemented or locked

Reset value

`0x0000_0000`

Bit descriptions

The following table shows the register bit assignments.

Table 6-9: KMUKSK<2><m> bit description

Bits	Name	Description	Type	Erase
[31:0]	Key	Key word m=0..7 of key slot <0>	RW	0x0000_0000

6.1.8.2 KMUKSK<1><m>, KMU key slot key <1><m> register

Key slot n=1 of the KMUKSK<n><m>. Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x150

Type

- RW
- **RAZ/WI** when key slot register is unimplemented or locked

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-10: KMUKSK<1><m> bit description

Bits	Name	Description	Type	Erase
[31:0]	Key	Key word m=0..7 of key slot <1>	RW	0x0000_0000

6.1.8.3 KMUKSK<2><m>, KMU key slot key <2><m> register

Key slot n=2 of the KMUKSK<n><m>. Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x170

Type

- RW
- **RAZ/WI** when key slot register is unimplemented or locked

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-11: KMUKSK<2><m> bit description

Bits	Name	Description	Type	Erase
[31:0]	Key	Key word m=0..7 of key slot <2>	RW	0x0000_0000

6.1.8.4 KMUKSK<3><m>, KMU key slot key <3><m> register

Key slot n=3 of the KMUKSK<n><m>. Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x190

Type

- RW
- **RAZ/WI** when key slot register is unimplemented or locked

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-12: KMUKSK<3><m> bit description

Bits	Name	Description	Type	Erase
[31:0]	Key	Key word m=0..7 of key slot <3>	RW	0x0000_0000

6.1.8.5 KMUKSK<4><m>, KMU key slot key <4><m> register

Key slot n=4 of the KMUKSK<n><m>. Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x1B0

Type

- RW
- **RAZ/WI** when key slot register is unimplemented or locked

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-13: KMUKSK<4><m> bit description

Bits	Name	Description	Type	Erase
[31:0]	Key	Key word m=0..7 of key slot <4>	RW	0x0000_0000

6.1.8.6 KMUKSK<5><m>, KMU key slot key <5><m> register

Key slot n=5 of the KMUKSK<n><m>. Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x1D0

Type

- RW
- **RAZ/WI** when key slot register is unimplemented or locked

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-14: KMUKSK<5><m> bit description

Bits	Name	Description	Type	Erase
[31:0]	Key	Key word m=0..7 of key slot <5>	RW	0x0000_0000

6.1.8.7 KMUKSK<6><m>, KMU key slot key <6><m> register

Key slot n=6 of the KMUKSK<n><m>. Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x1F0

Type

- RW
- **RAZ/WI** when key slot register is unimplemented or locked

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-15: KMUKSK<6><m> bit description

Bits	Name	Description	Type	Erase
[31:0]	Key	Key word m=0..7 of key slot <6>	RW	0x0000_0000

6.1.8.8 KMUKSK<7><m>, KMU key slot key <7><m> register

Key slot n=7 of the KMUKSK<n><m>. Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x210

Type

- RW
- **RAZ/WI** when key slot register is unimplemented or locked

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-16: KMUKSK<7><m> bit description

Bits	Name	Description	Type	Erase
[31:0]	Key	Key word m=0..7 of key slot <7>	RW	0x0000_0000

6.1.8.9 KMUKSK<8..31><m>, KMU key slot key <8..31><m> register

Key slot n=8..31 of the KMUKSK<n><m>. Software uses the KMUKSK<n><m> registers to set the key words m=0..7 of key slot <n>. All eight key registers of key slot <n> are contiguous to each other.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x230-0x52C

Type

- RW
- **RAZ/WI** when key slot register is unimplemented or locked

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-17: KMUKSK<8..31><m> bit description

Bits	Name	Description	Type	Erase
[31:0]	Key	Key word m=0..7 of key slot <8..31>	RW	0x0000_0000

6.1.9 KMURD_8, random delay 8 clocks register

When the software reads the KMURD_8 register, it loads a temporary 3-bits counter from the PRBG and starts counting down. When the counter reaches zero, the hardware responds to the read operation with 32 bits of zeroes.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x530

Type

RAZ/WI

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-18: KMURD_8 register bit description

Bits	Name	Description	Type	Reset
[31:0]	KMURD_8	Always reads as zero	**RAZ/WI**	0x0

6.1.10 KMURD_16, KMU random delay 16 clocks register

When the software reads the KMURD_16 register, it loads a temporary 4-bits counter from the PRBG and starts counting down. When the counter reaches zero, the hardware responds to the read operation with 32 bits of zeroes.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x534

Type

RAZ/WI

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-19: KMURD_16 register bit description

Bits	Name	Description	Type	Reset
[31:0]	KMURD_16	Always reads as zero	RAZ/WI	0x0

6.1.11 KMURD_32, KMU random delay 32 clocks register

When the software reads the KMURD_32 register, it loads a temporary 5-bits counter from the PRBG and starts counting down. When the counter reaches zero, the hardware responds to the read operation with 32 bits of zeroes.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0x538

Type

RAZ/WI

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-20: KMURD_32 register bit description

Bits	Name	Description	Type	Reset
[31:0]	KMURD_32	Always reads as zero	RAZ/WI	0x0

6.1.12 PIDR4, Peripheral ID 4 register

The Peripheral ID4 register returns byte[4] of the peripheral ID.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

`0xFD0`

Type

RO

Reset value

`0x0000_0004`

Bit descriptions

The following table shows the register bit assignments.

Table 6-21: Peripheral ID 4 register bit description

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	-	-
[7:4]	SIZE	4KB Count, the number of 4K pages used. • `0x00`: 4K • `0x01`: 8K • `0x02`: 16K • `0x03`: 32K	RO	`0x0`
[3:0]	DES_2	JEP106 Continuation Code	RO	`0x4`

6.1.13 PIDR0, Peripheral ID 0 register

The Peripheral ID0 register returns byte[0] of the peripheral ID.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

`0xFE0`

Type

RO

Reset value

0x0000_00F3

Bit descriptions

The following table shows the register bit assignments.

Table 6-22: Peripheral ID 0 register bit description

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	-	-
[7:0]	PART_0	Identification register part number, bits [7:0]	RO	0xF3

6.1.14 PIDR1, Peripheral ID 1 register

The Peripheral ID1 register returns byte[1] of the peripheral ID.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0xFE4

Type

RO

Reset value

0x0000_00B0

Bit descriptions

The following table shows the register bit assignments.

Table 6-23: Peripheral ID 1 register bit assignments

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	-	-
[7:4]	DES_0	JEP106 identification code, bits [3:0]	RO	0xB
[3:0]	PART_1	Identification register part number, bits [11:8]	RO	0x0

6.1.15 PIDR2, Peripheral ID 2 register

The Peripheral ID2 register returns byte[2] of the peripheral ID.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

`0xFE8`

Type

RO

Reset value

`0x0000_001B`

Bit descriptions

The following table shows the register bit assignments.

Table 6-24: Peripheral ID 2 register bit assignments

Bits	Name	Description	Type	Reset
[31:8]	_	Reserved	_	_
[7:4]	REVISION	Revision Code	RO	`0x1`
[3]	JEDEC	JEDEC	RO	`0x1`
[2:0]	DES_1	JEP106 identification code, bits [6:4]	RO	`0x3`

6.1.16 PIDR3, Peripheral ID 3 register

The Peripheral ID3 register returns byte[3] of the peripheral ID.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0xFEC

Type

RO

Reset value

0x0000_0000

Bit descriptions

The following table shows the register bit assignments.

Table 6-25: Peripheral ID 3 register bit assignments

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	-	-
[7:4]	REVAND	Manufacturer revision number	R0	0x0
[3:0]	CMOD	Customer Modified	R0	0x0

6.1.17 CIDR0, Component ID 0 register

The Component ID0 register returns byte[0] of the component ID.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0xFF0

Type

RO

Reset value

0x0000_000D

Bit descriptions

The following table shows the register bit assignments.

Table 6-26: Component ID 0 register bit assignments

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	-	-
[7:0]	PRMBL_0	Preamble	RO	0xD

6.1.18 CIDR1, Component ID 1 register

The Component ID1 register returns byte[1] of the component ID.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0xFF4

Type

RO

Reset value

0x0000_00F0

Bit descriptions

The following table shows the register bit assignments.

Table 6-27: Component ID 1 register bit assignments

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	-	-
[7:4]	CLASS	Component class	RO	0xF
[3:0]	PRMBL_1	Preamble	RO	0x0

6.1.19 CIDR2, Component ID 2 register

The Component ID2 register returns byte[2] of the component ID.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0xFF8

Type

RO

Reset value

0x0000_0005

Bit descriptions

The following table shows the register bit assignments.

Table 6-28: Component ID 2 register bit assignments

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	-	-
[7:0]	PRMBL_2	Preamble	R0	0x5

6.1.20 CIDR3, Component ID 3 register

The Component ID3 register returns byte[3] of the component ID.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

KMU register summary

Address offset

0xFFC

Type

RO

Reset value

0x0000_00B1

Bit descriptions

The following table shows the register bit assignments.

Table 6-29: Component ID 3 register bit assignments

Bits	Name	Description	Type	Reset
[31:8]	-	Reserved	-	-
[7:0]	PRMBL_3	Preamble	RO	`0xB1`

Appendix A Revisions

The following table describes the technical changes between released issues of this document.

Table A-1: Issue 0000-01

Change	Location
Initial issue of the document	-