15 Security Packet Engine (PKTE)

The PKTE is a security packet engine designed to off-load the host processor to improve the speed of applications requiring cryptographic processing.

PKTE Features

The PKTE has the following features.

- Hardware assisted processing for the cryptographic ciphers, hashes, and pseudo-random number generation
- Header and trailer processing for Internet security protocols
- DMA capability to move data in and out of the engine efficiently and to allow the engine to run autonomously while moving data
- Interrupt controller to signal module status and errors
- Clock manager for enabling or disabling different features to save power

NOTE: Not all algorithms, decrypt, and hash functions and extra features are available on all product models. For complete information on included features, see the product-specific data sheet.

NOTE: This packet engine provides support for various network security protocols by processing headers and trailers as well as accelerating cryptographic functions. Not all processors have direct support for Ethernet. As such, the packet engine can still be used if Ethernet is indirectly used.

PKTE Functional Description

The packet engine contains a set of modules for encryption and decryption, hashing, and pseudo-random number generation. The following sections describe these functional blocks.

ADSP-BF70x PKTE Register List

The Security Packet Engine (PKTE) provides security-related features. A set of registers governs PKTE operations. For more information on PKTE functionality, see the PKTE register descriptions.

Table 15-1: ADSP-BF70x PKTE Register List

Name	Description
PKTE_BUF_PTR	Packet Engine Buffer Pointer Register
PKTE_BUF_THRESH	Packet Engine Buffer Threshold Register
PKTE_CDRBASE_ADDR	Packet Engine Command Descriptor Ring Base Address
PKTE_CDSC_CNT	Packet Engine Command Descriptor Count Register
PKTE_CDSC_INCR	Packet Engine Command Descriptor Count Increment Register
PKTE_CFG	Packet Engine Configuration Register
PKTE_CLK_CTL	PE Clock Control Register
PKTE_CONT	PKTE Continue Register
PKTE_CTL_STAT	Packet Engine Control Register
PKTE_DATAIO_BUF	Starting Entry of 256-byte Data Input/Output Buffer
PKTE_DEST_ADDR	Packet Engine Destination Address
PKTE_DMA_CFG	Packet Engine DMA Configuration Register
PKTE_ENDIAN_CFG	Packet Engine Endian Configuration Register
PKTE_HLT_CTL	Packet Engine Halt Control Register
PKTE_HLT_STAT	Packet Engine Halt Status Register
PKTE_IMSK_DIS	Interrupt Mask Disable Register
PKTE_IMSK_EN	Interrupt Mask Enable Register
PKTE_IMSK_STAT	Interrupt Masked Status Register
PKTE_INBUF_CNT	Packet Engine Input Buffer Count Register
PKTE_INBUF_INCR	Packet Engine Input Buffer Count Increment Register
PKTE_INT_CFG	Interrupt Configuration Register
PKTE_INT_CLR	Interrupt Clear Register
PKTE_INT_EN	Interrupt Enable Register
PKTE_IUMSK_STAT	Interrupt Unmasked Status Register
PKTE_LEN	Packet Engine Length Register
PKTE_OUTBUF_CNT	Packet Engine Output Buffer Count Register
PKTE_OUTBUF_DECR	Packet Engine Output Buffer Count Decrement Register
PKTE_RDRBASE_ADDR	Packet Engine Result Descriptor Ring Base Address
PKTE_RDSC_CNT	Packet Engine Result Descriptor Count Registers
PKTE_RDSC_DECR	Packet Engine Result Descriptor Count Decrement Registers
PKTE_RING_CFG	Packet Engine Ring Configuration

Table 15-1: ADSP-BF70x PKTE Register List (Continued)

Name	Description
PKTE_RING_PTR	Packet Engine Ring Pointer Status
PKTE_RING_STAT	Packet Engine Ring Status
PKTE_RING_THRESH	Packet Engine Ring Threshold Registers
PKTE_SA_ADDR	Packet Engine SA Address
PKTE_SA_CMD0	SA Command 0
PKTE_SA_CMD1	SA Command 1
PKTE_SA_IDIGEST[n]	SA Inner Hash Digest Registers
PKTE_SA_KEY[n]	SA Key Registers
PKTE_SA_NONCE	SA Initialization Vector Register
PKTE_SA_ODIGEST[n]	SA Outer Hash Digest Registers
PKTE_SA_RDY	SA Ready Indicator
PKTE_SA_SEQNUM[n]	SA Sequence Number Register
PKTE_SA_SEQNUM_MSK[n]	SA Sequence Number Mask Registers
PKTE_SA_SPI	SA SPI Register
PKTE_SRC_ADDR	Packet Engine Source Address
PKTE_STAT	Packet Engine Status Register
PKTE_STATE_ADDR	Packet Engine State Record Address
PKTE_STATE_BYTE_CNT[n]	State Hash Byte Count Registers
PKTE_STATE_IDIGEST[n]	State Inner Digest Registers
PKTE_STATE_IV[n]	State Initialization Vector Registers
PKTE_USERID	Packet Engine User ID

ADSP-BF70x PKTE Interrupt List

Table 15-2: ADSP-BF70x PKTE Interrupt List

Interrupt ID	Name	Description	Sensitivity	DMA Channel
67	PKTE0_IRQ	PKTE0 Security Packet Engine Interrupt	Level	

PKTE Definitions

Command Descriptor

An 8-word structure that is either written directly into the packet command MMR set or is placed in a Command Descriptor Ring (CDR) in the processor memory. The packet engine sequentially processes the structure. The

command descriptor contains the information that varies for every packet. This information includes pointers to the SA record, the state information, the source packet, and the destination packet.

Command Descriptor Ring (CDR)

A circular contiguous portion of memory which is used to manage one or more command descriptions for the packet engine.

Result Descriptor

When the packet engine completes the processing of a packet, it writes a result descriptor with the state information. The result descriptor can be read directly from the result register set or from the Result Descriptor Ring (RDR) in the processor memory.

Result Descriptor Ring (RDR)

A circular contiguous portion of memory which holds the mirror or copy of the CDR but contains the result descriptors. The RDR and CDR can be overlaid on top of each other.

Security Association (SA) Record

A structure that contains the remainder of the information the packet engine requires to process a packet. Most of the information fields in the SA record such as the key and encryption mode are static for the lifetime of the association. The fields do not require frequent manipulation by the processor core. The SA record non-static fields are the sequence number and sequence number mask. The SA record can have a corresponding state record for saving results from the current operations that are useful for future operations. The state record can hold the IV, the hash byte count, and the intermediate hash digest.

Hash

A cryptographic hash is a function that takes an arbitrary block of data and returns a fixed-size bit string. Four main properties define the function:

- It is easy to compute a hash value for any given input
- It is infeasible to generate the original input from a given hash
- It is infeasible to modify the input without changing the resulting hash
- It is infeasible to find two different inputs that result in the same hash

Autonomous Ring Mode (ARM)

Mode of operation in which most of the parameters as well as the data are set up in memory and moved to the engine for configuration and processing through DMA.

Target Command Mode (TCM)

Mode of operation where some parameters are set up in memory and moved into the packet engine through DMA while the other parameters are directly written to the registers. DMA moves the input and output data in and out of the engine.

Direct Host Mode (DHM)

Mode of operation that does not use DMA. All parameters are directly written to and read from the MMRs. The input and output are written to and read from the FIFO buffers.

Cipher Module

The cipher module does the symmetric encrypt or decrypt operations for:

- Data Encryption Standard (DES)
- Triple-DES
- Advanced Encryption Standard (AES) algorithms

The cipher module supports standard modes for DES and AES that include Electronic Code Book (ECB) and Cipher Block Chaining (CBC). The key size for DES is 56 bits, for Triple-DES is 168 bits. The AES module also provides support for AES counter-modes for IPsec and SRTP. All AES modes can use key sizes of 128 bits and 192/256 bits. Key scheduling is automatic and done in parallel with the encrypt or decrypt operation.

Hash Module

The hash module is tightly coupled with the encrypt or decrypt module and provides hardware accelerated one-way hash functions. Operations that combine both hash and encrypt or decrypt functions are provided to reduce processing time for data that needs both applied. For hash-then-decrypt operations, the packet engine performs parallel execution of both functions from the input buffer. For encrypt-then-hash operations, the processing is pipelined from the input buffer to provide minimum latency. An offset can be specified between the start of hashing and the start of encryption to support protocols such as IPsec or SRTP. The HMAC keyed hashing mechanism is supported for MD5, SHA-1, SHA-2-224 and SHA-2-256. The SSL-MAC is supported for MD5 and SHA-1. A second AES-CBC module for the hash function enables parallel processing for AES-CCM, a combined hash and encrypt algorithm.

Pseudo-Random Number Generator

Cipher algorithms that operate in CBC mode or counter-mode, require an IV. This IV must not be secret; however the IV must be unpredictable and unique for each execution of the encryption process. Pseudo-random number generators are deterministic algorithms that output statistically independent and unbiased numbers. True random number generators are non-deterministic and use the randomness that occurs in a physical process. The packet engine incorporates an ANSI X9.31 compliant Pseudo Random Number Generator (PRNG) that it can use to generate unique IVs using strong encryption. The ANSI X9.31 PRNG is defined as part of the ANSI X9 standards that

are used to secure financial transactions. The function can also be used for pseudo-random number generation as part of an implementation of the digital signature standard described in NIST FIPS PUB 186-2.

The PRNG function, as defined by ANSI X9.31, is based on the AES cipher. This section describes the function to promote understanding of the different inputs and outputs of the PRNG function itself.

NOTE: The PRNG in the packet engine is only based on the AES cipher with 128-bit keys. Other ciphers and key lengths are not supported for the PRNG based on ANSI X9.31.

Let $e \times K(Y)$ represent the AES encryption of Y under the key K.

The PRNG function uses three inputs:

- K, a 128-bit key
- V, a 128-bit seed value
- DT, a 128-bit date/ time vector which is updated on each iteration

The intermediate value I is the result of an AES encryption of the data and time vector under key K.

$I = e \times K(DT)$

That value I is then XOR-ed with the seed V and AES encrypted under key K. The result R is the output of the PRNG function.

$R = e \times K(I \; XOR \; V)$

A new seed value V is generated from the AES encryption of the result R XOR-ed with the intermediate value I under the key K.

$V = e \times K(R \; XOR \; I)$

The PRNG function is deeply integrated inside the datapath of the packet engine. The function is controlled indirectly through the PRNG mode bits in the `PKTE_CTL_STAT.PRNGMD` bit field of the command descriptor and the IV source selection bits in the `PKTE_SA_CMD0.IVSRC` bit field of the SA record.

The PKTE module supports four different modes.

1. Load IV from PRNG for the current operation: `PKTE_CTL_STAT.PRNGMD` = 0b00 and `PKTE_SA_CMD0.IVSRC` = 0b11.
2. PRNG init mode initializes the PRNG with a key, seed, and date/time value: `PKTE_CTL_STAT.PRNGMD` = 0b01.

   Before the PRNG function can be used, it must be initialized with a key, seed, and date/time value. At initialization, the key, seed, and date/time values are programmed. Other PRNG operations do not change the key, however the seed, and date/time values are updated (not re-programmed). The date/time is updated every 128 system clock cycles.

   The date/time is a 128-bit value with randomly distributed number of ones and zeros. It must not be all zeros.

The ***SA Record for PRNG Init Operation*** table shows how the key, seed and date/time values are loaded into the PKTE registers for initialization.

Table 15-3: SA Record for PRNG Init Operation

Parameter	SA Field	Description
K	SA_KEY0	PRNG key [127:96]
	SA_KEY1	PRNG key [95:64]
	SA_KEY2	PRNG key [63:32]
	SA_KEY3	PRNG key [31:0]
V	SA_IDIGEST0	PRNG seed [127:96]
	SA_IDIGEST1	PRNG seed [95:64]
	SA_IDIGEST2	PRNG seed [63:32]
	SA_IDIGEST3	PRNG seed [31:0]
DT	SA_ODIGEST0	PRNG date/time [127:96]
	SA_ODIGEST1	PRNG date/time [95:64]
	SA_ODIGEST2	PRNG date/time [63:32]
	SA_ODIGEST3	PRNG date/time [31:0]

3. PRNG generate mode generates pseudo-random data on initialized key, seed, and date/time value: `PKTE_CTL_STAT.PRNGMD` = 0b10.

   The PRNG function can be used to generated pseudo-random data for other purposes than IVs. For this mode, the PRNG must have been initialized once with the PRNG init mode.

   The PRNG generate mode uses the initialized key and unique changing date/time value as inputs. The pseudo-random data is output to the output buffer of the packet engine.

   The `LEN` field in the command descriptor indicates the amount of pseudo random data that is generated in multiples of 16 bytes. The maximum is 255 × 16 = 4080 bytes.

   In autonomous ring mode, the output data is copied to the host memory at the destination address in the command descriptor. In direct host mode, the host must read the data directly from the output buffer. No SA record is used for this function.

   Directly after the PRNG generate mode, a new pseudo-random number is generated and available for the next operation that uses the option `PKTE_SA_CMD0.IVSRC = PRNG`.

4. PRNG test mode generates pseudo-random data on initialized key, seed, and input (test) data: `PKTE_CTL_STAT.PRNGMD` = 0b11.

   The PRNG test mode can be used to test the correctness of the PRNG function. This mode is similar to the PRNG generate mode, except that the data is read from the input buffer of the packet engine, instead of the date/time value.

For this mode, the PRNG must have been initialized once with the PRNG Init mode.

The LEN field in the command descriptor indicates the amount of pseudo-random data to be generated in multiples of 16 bytes. The maximum is limited to the LEN field in bytes.

In autonomous ring mode, the output data is copied to the host memory at the destination address in the command descriptor. In the direct host mode, the host must read the data directly from the output buffer. No SA record is used for this function.

Directly after the PRNG test mode, a new pseudo-random number is generated and available for the next operation that uses the option PKTE_SA_CMD0.IVSRC = PRNG.

Packet Engine Processing Details

This section describes data processing through the packet engine. It describes padding and supported algorithms for each protocol.

A valid Security Association (SA) must be created before packet processing can start. A formatted SA record must reside in memory and be accessible to the packet engine. The host processor application is responsible for these tasks.

Crypto Padding

Padding is the process of adding data to fill-out a fixed-size plain text data structure. Three factors determine when to use a pad field:

1. If a block cipher encryption algorithm is used, a pad field is used to expand the plain text to a multiple of the block size.
2. Padding can be used to ensure that the cipher text terminates on an n-byte boundary.
3. Padding can conceal the actual length of the payload.

To facilitate peak encrypt or decrypt performance, the packet engine supports the following most commonly used padding functions in hardware:

1. Pad generation and insertion of pad bytes to the end of plain text prior to encryption, for outbound operations.
2. Pad verification to check for correct padding after decrypting a packet for inbound operations.
3. Pad consumption to strip the pad bytes from the plain text data after decrypting a packet, for inbound operations.

Pad Generation and Insertion

The pad type and how many bytes the packet engine inserts depends on the plain text length and the value of the following fields.

- PKTE_SA_CMD0.PADTYPE and PKTE_SA_CMD0.EXTPAD defines the type of padding.
- PKTE_CTL_STAT.PADVAL defines a value that is inserted in the pad.

- PKTE_SA_CMD0.CIPHER enforces a certain pad alignment.
- PKTE_SA_CMD1.CIPHERMD enforces a certain pad alignment.
- PKTE_SA_CMD0.SCPAD allows stream ciphers to be padded.
- PKTE_CTL_STAT.PADCTLSTAT controls the pad alignment.

The PKTE_CTL_STAT.PADCTLSTAT bit field of the result descriptor returns the total number of inserted pad bytes.

Pad Types

The pad type bit field (PKTE_SA_CMD0.PADTYPE) and the extended pad bit (PKTE_SA_CMD0.EXTPAD) select the pad type for the extended protocol group. The packet engine can generate different pad types in hardware as described in the ***Pad Examples*** table.

The PKTE_CTL_STAT.PADVAL bit field, together with the number of pad bytes, defines the value that is inserted in the pad. The format of the pad and the use of this field is best explained in an example (see the ***Pad Examples*** table).

For the IPsec pad type, this field holds the value that is inserted into the next header field (in the ESP trailer) of the innermost operation's header. For the Constant pad type or the Constant SSL pad type, this field holds the inserted fixed constant pad value. For all other pad types, this field is not used and must be zero.

Table 15-4: Pad Types

Pad Type	Value	Description
IPSec	0b000	Append 0 to 255 pad bytes, followed by a pad length byte and a next header byte. The first pad byte appended to the plain text is numbered 1, with subsequent pad bytes making up a monotonically increasing sequence: 1, 2, 3 and up. Append the pad length field that indicates the number of pad bytes (0–255), where a value of zero indicates no pad bytes present. Append the next header byte as specified in the PKTE_CTL_STAT.PADVAL field of the command descriptor. A minimum of 2 bytes are appended; zero pad bytes plus the pad length byte plus the next header byte, in which case the PKTE_CTL_STAT.PADCTLSTAT field in the result descriptor returns 0x02. A maximum of 257 bytes can be appended, in which case the PKTE_CTL_STAT.PADCTLSTAT field in the descriptor returns 0x01.
PKCS#7	0b001	Appends 1–128 pad bytes with a pad byte value equal to the pad length (1–128).
Constant	0b010	Appends 0–255 pad bytes of a user-specified character to the plain text data. This character is specified in the PKTE_CTL_STAT.PADVAL field of the command descriptor.
Zero	0b011	Appends 0–255 pad bytes of 0x00 to the plain text data.
Constant SSL	0b110	Appends 0–255 pad bytes of a user-specified character to the plain text data, followed by a ‘pad length’ byte (0–255). This character is specified in the PKTE_CTL_STAT.PADVAL field of the command descriptor. A total of 256 bytes can be appended, in which case the pad field returns 0x00.

For example, the ***Pad Examples*** table shows the appended pad for any of the pad types for an outbound (encrypt) operation. The table shows a plain text input of 2 bytes using the 8-byte block cipher crypto-algorithm DES-ECB and a PKTE_CTL_STAT.PADVAL field value of 0xAA.

Table 15-5: Pad Examples

Pad Type	Pad field (extended to Plain Text)	PKTE_CTL_STAT
IPSec	0x01, 0x02, 0x03, 0x04, 0x04, 0xAA	0x06
PKCS#7	0x06, 0x06, 0x06, 0x06, 0x06, 0x06	0x06
Constant	0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA	0x06
Zero	0x00, 0x00, 0x00, 0x00, 0x00, 0x00	0x06
Constant SSL	0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0x05	0x06

Pad Length

The ***Pad Alignment*** table lists the alignment (boundaries) to which the packet engine will pad, based on:

- The selected crypto-algorithm
- Crypto mode
- The value of the pad stream cipher bit
- The value of pad control

The minimum number of inserted pad bytes depends on the cipher algorithm, selected using the PKTE_SA_CMD0.CIPHER bit field, and the cipher mode, selected using the PKTE_SA_CMD1.CIPHERMD bit field.

For block ciphers, the plain text data is always (as a minimum) padded to the next block boundary. More pad bytes beyond the algorithm or protocol alignment requirements can be inserted using the pad control (PKTE_CTL_STAT.PADCTLSTAT) in the command descriptor. This feature can be used for *traffic flow security* to conceal the number of plain text bytes in an encrypted packet.

Encrypt operations that use block ciphers have minimum pad requirements based on their block size. The packet engine enforces a minimum pad alignment for block ciphers according to the ***Pad Alignment*** table. For ESP outbound operations, the minimum pad alignment is forced to 4 bytes.

For stream ciphers and null crypto the data is never padded when the stream cipher pad bit (PKTE_SA_CMD0.SCPAD) =0. When PKTE_SA_CMD0.SCPAD =1 the plain text data is padded to the length as defined by the PKTE_CTL_STAT.PADCTLSTAT bits in the command descriptor.

NOTE: The SSL protocol does not allow padding to exceed the ciphers block length. This length is 8 bytes for DES/Triple-DES and 16 bytes for AES. For SSL, the packet engine does not enforce this pad alignment value. The host processor must ensure that the PKTE_CTL_STAT.PADCTLSTAT bit field is configured correctly.

Table 15-6: Pad Alignment

Pad Control PADCTLSTAT = PKTE_CTL_STAT[31:24]			0x00	0x01*1	0x02	0x04	0x08	0x10	0x20	0x40	0x80*2
Crypto Algorithm	**Crypto Mode**	**Pad Stream Ciphers**	**%8**	**%1**	**%4**	**%8**	**%16**	**%32**	**%64**	**%128**	**%256**
DES	ECB, CBC	N/A	8	8	8	8	16	32	64	128	256
AES	ECB, CBC	N/A	16	16	16	16	16	32	64	128	256
	CTR, ICM with ESP	N/A	8	4	4	8	16	32	64	128	256
	CTR, ICM no ESP	no	0	0	0	0	0	0	0	0	0
		yes	8	0	4	8	16	32	64	128	256
NULL	with ESP	N/A	8	4	4	8	16	32	64	128	256
	no ESP	no	8	0	0	0	0	0	0	0	0
		yes	8	0	4	8	16	32	64	128	256

*1 If `PKTE_CTL_STAT.PADCTLSTAT` is configured for no padding (0x01), it does not mean that no padding bytes are inserted. When PKCS#7 padding is selected, a pad length field with a value =1 is inserted. When SSL or TLS padding is selected, a pad length field with a value =0 is inserted. When IPsec padding is selected, a pad length field is forced to (`PKTE_CTL_STAT.PADCTLSTAT`=0x20). When zero pad and constant pad are selected, no pad bytes are inserted.

*2 Pad type PKCS#7 supports a maximum length of 128 pad bytes, so the packet engine overrules a 256-byte alignment (`PKTE_CTL_STAT.PADCTLSTAT`=0x80) to a 128-byte boundary (`PKTE_CTL_STAT.PADCTLSTAT`=0x40).

The packet engine does not constrain the pad type that is used for an operation; any pad type can be used for each operation. The user must be aware that some protocol specifications only allow specific pad types. The SRTP specification does not have padding defined as padding performed by RTP. The host must pad the RTP packet.

The host software can implement padding that is not supported in hardware. In this case, the host must select the *zero pad* type and set the `PKTE_CTL_STAT.PADCTLSTAT` bit field in the packet descriptor to zero (no padding). The hardware padding engine does not add any bytes. When using a block cipher, the host must insert pad bytes. Then, the data to be encrypted (plain text and pad bytes) are a multiple of the block ciphers boundary. For stream ciphers, any number of pad bytes can be added.

Pad Verification and Consumption

The packet engine can validate a pad type against the expected values. The value of the following bits controls the pad verify function.

- `PKTE_SA_CMD0.PADTYPE` and `PKTE_SA_CMD0.EXTPAD` define the type of padding.
- `PKTE_SA_CMD0.CIPHER` enforces a certain pad alignment.
- `PKTE_SA_CMD1.CIPHERMD` enforces a certain pad alignment.
- `PKTE_SA_CMD0.SCPAD` allows stream ciphers to be padded.

When packet processing is complete, the STATUS byte in the first word of the result descriptor reports the pad verification status. Refer to the Table 15-24 Extended Error Codes - Status Encoding table.

The PKTE_CTL_STAT.PADCTLSTAT bits in the first word of the result descriptor return the total number of detected pad bytes and returns zero for a pad verify error.

When the IPsec pad type is selected, the PKTE_CTL_STAT.PADVAL bit field of the result descriptor returns the next header field. For IPsec ESP outbound operations, this field returns the decimal value 50. For IPsec inbound operations and basic inbound operations that use the IPsec pad mode, the packet engine returns the next header field it detects on the header of the innermost operation. This value is typical for the payload protocol, such as TCP or UDP. However, in bundling scenarios or in IPv6 with destination option headers, another header value could be seen. For all other inbound operations, the returned pad value is zero.

Pad verification is performed for inbound (decrypt) operations that use IPsec, TLS/DTLS or PKCS#7 pad type:

- In combination with a block cipher algorithm: DES-ECB, DES-CBC, AES-ECB, AES-CBC,
- In combination with a stream cipher and stream cipher padding PKTE_SA_CMD0.SCPAD enabled: AES-CTR, AES-ICM
- In combination with null-crypto (no encryption).

Pad Types

The PKTE_SA_CMD0.PADTYPE bit field and the extended pad PKTE_SA_CMD0.EXTPAD bit selects the pad type for the extended protocol group pad type. The packet engine can verify the different pad types in hardware as described in the ***Pad Types*** table.

The constant and zero pad types are not verified since they do not include a pad length field. The SSL pad type is not verified since it does not have a defined pattern.

Table 15-7: Pad Types

Pad Type	SA_CMD0[18, 7:6]	Description
IPSec	0b000	Verify that the pad field includes 0–255 pad bytes, followed by a correct pad length and a next header byte. Verify that pad bytes appended to the plain text are an incremental count, starting at one. Verify that the pad length field is the number of pad bytes (0–255), where a value of zero indicates no pad bytes present. Verify that a next header byte is present as the last byte of the packet, the value is not verified. This is after removal of the ICV. The total number of detected pad bytes is returned in the PKTE_CTL_STAT.PADCTLSTAT field in the result descriptor. **NOTE**: A minimum of 2 bytes must be present, zero pad bytes plus the pad length byte plus the next header byte, in which case the PKTE_CTL_STAT.PADCTLSTAT field in the descriptor returns 0x02. A maximum of 257 bytes can be present, in which case the PKTE_CTL_STAT.PADCTLSTAT field in the result descriptor returns 0x01. The value of the next header byte is returned in the PKTE_CTL_STAT.PADVAL field in the result descriptor.

Table 15-7: Pad Types (Continued)

Pad Type	SA_CMD0[18, 7:6]	Description
PKCS#7	0b001	Verify that the pad field includes 1–128 pad bytes, with a pad byte value equal to the pad length (1–128).

When a block cipher is used and the payload is not padded to the nearest block size boundary, as required by the protocol, a *block size error* is generated for all pad types.

Pad Consumption

The packet engine can optionally consume the decrypted pad bytes for an inbound operation that uses the IPsec, SSL, TLS/DTLS, or PKCS#7 pad types. The pad types constant and zero are not consumed since they do not include a pad length field.

Pad consumption (or stripping) is selected on a flow-by-flow basis in the SA-record with the `PKTE_SA_CMD1.CPYPAD` bit. When this bit is set, the length returned in the LEN field of the result descriptor is the total length of the plain text including the pad. When the `PKTE_SA_CMD1.CPYPAD` is disabled, the detected pad length, as returned in the `PKTE_CTL_STAT.PADCTLSTAT` bits, is subtracted from the total length and then returned in the LEN field of the result descriptor.

The pad is always written to the result packet buffer in memory. When the `PKTE_SA_CMD1.CPYPAD` bit is disabled, only the result length is corrected.

Crypto and Hash Algorithms

The packet engine supports a wide range of crypto and hash algorithms to accelerate basic operations and protocol operations. These algorithms are:

- Basic Encrypt and Basic Decrypt Operations
- Basic Hash Operations
- Basic Encrypt-Hash and Basic Hash-Decrypt Operations
- IPSec ESP Operations
- SRTP Operations

The following tables provide allowed algorithm combinations. Those algorithms not listed in the tables are invalid and can give unexpected results.

NOTE: Not all crypto and hash algorithms are available on all product models. For information on algorithm availability, see the product-specific data sheet.

Table 15-8: Algorithms for Basic Encrypt and Basic Decrypt Operations

Crypto Algorithm	Crypto Mode
DES, Triple-DES	ECB, CBC
AES	ECB, CBC, CRT, ICM
NULL	—

Table 15-9: Algorithms for Basic Encrypt and Basic Decrypt Operations

Crypto Algorithm	Crypto Mode
SHA-1	Basic hash, HMAC, SSL-MAC
SHA-224	Basic hash, HMAC
SHA-256	Basic hash, HMAC
NULL	—

Table 15-10: Algorithms for Basic Encrypt-Hash and Basic Hash-Decrypt Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	ECB, CBC	SHA-1	Basic hash, HMAC, SSL-MAC
		SHA-224	Basic hash, HMAC
		SHA-256	Basic hash, HMAC
		MD5	Basic hash, HMAC, SSL-MAC
		NULL	—
AES	ECB, CBC, CRT, ICM	SHA-1	Basic hash, HMAC, SSL-MAC
		SHA-224	Basic hash, HMAC
		SHA-256	Basic hash, HMAC
		MD5	Basic hash, HMAC, SSL-MAC
		NULL	—
NULL	—	SHA-1	Basic hash, HMAC, SSL-MAC
		SHA-224	Basic hash, HMAC
		SHA-256	Basic hash, HMAC
		MD5	Basic hash, HMAC, SSL-MAC

Table 15-11: Algorithms for IPsec ESP Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	CBC	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
AES	CBC, CTR	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
NULL	CBC	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	—

Table 15-12: Algorithms for Basic SSL and Extended SSL Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	CBC	SHA-1	SSL-MAC
		MD5	SSL-MAC
		NULL	—
AES	CBC	SHA-1	SSL-MAC
		MD5	SSL-MAC
		NULL	—
NULL	—	SHA-1	SSL-MAC
	—	MD5	SSL-MAC
	—	NULL	—

Table 15-13: Algorithms for Basic TLS, Extended TLS and DTLS Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
DES, Triple-DES	CBC	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
AES	CBC, CTR	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—
NULL	—	SHA-1	HMAC
		SHA-224	HMAC
		SHA-256	HMAC
		MD5	HMAC
		NULL	—

Table 15-14: Algorithms for SRTP Operations

Crypto Algorithm	Crypto Mode	Hash Algorithm	Hash Mode
AES	ICM	SHA-1	HMAC
NULL	—	SHA-1	HMAC

IV Processing

An initialization vector (IV) is necessary to start a cipher stream or a block cipher in any of the streaming modes of operation. The IV must be unique for each packet. The IV ensures that all cipher texts are unique even if produced by the same encryption key. This functionality prevents every packet from needing a unique encryption key.

Depending on the packet engine operation, the IV can be loaded from different sources. The IV format depends on the algorithm and the source of the IV. The ***Format of the IV*** table provides an overview of all IV formats.

Table 15-15: Format of the IV

Algorithm	IV Source (PKTE_SA_CMD0.IVSRC)	Format	IV Offset (PKTE_SA_CMD1.HSHCOFFST)
DES/Triple-DES (CBC)	Previous Result of IV	Internal IV register [63:0]	0x00
	Input Buffer	Input buffer [63:0]	0x02
	Saved IV	State Record Saved IV [63:0]	0x00
	Automatic	PRNG output [63:0]	0x00
AES (CBC)	Previous Result of IV	Internal IV register [127:0]	0x00
	Input Buffer	Input Buffer [127:0]	0x04
	Saved IV	State Record Saved IV [127:0]	0x00
	Automatic	PRNG output [127:0]	0x00
AES (ICM) for Basic and SRTP	Input Buffer	Input Buffer [127:0]	0x04
	Saved IV	State Record Saved IV [127:0]	0x00
AES (CTR) for Basic and IP-sec	Input Buffer	SA_NONCE || Input Buffer[63:0] || 0x00000001	0x02
	Saved IV	State Record Saved IV [127:0]	0x00
	Automatic	SA_NONCE || PRNG output [95:32] || 0x00000001	0x00

Notes:

1. The PKTE_SA_CMD1.HSHCOFFST bit field provides the IV offset. The offset is only applicable for basic hash or encrypt operations. For protocol operations, the offset is automatically enforced.
2. AES-CTR: The Nonce value as described in RFC 3686, is mapped to the PKTE_SA_NONCE register. This Nonce value remains constant for the lifetime of the security association.
3. The host processor controls the IV update using the save-IV bit (PKTE_SA_CMD0.SVIV). When part of a packet is processed using a stream cipher and the encrypt or decrypt data is not an integer multiple of the block size, the saved IV is invalid. It must not be used to resume processing the packet.
4. The packet engine supports automatic IV generation for outbound operations. A new IV is generated for every packet with the internal PRNG. This automatic IV generation can be used for all DES, Triple-DES, and AES modes that use an IV, except for AES-ICM mode.
5. For outbound operations, automatic IV generation is the most efficient. No additional I/O is required, and the host processor does not need to provide the IV. When the saved IV option supplies the IV, the IV must be changed for each packet sent. This activity happens when the packet engine writes back the IV to the state record after processing.
6. Outbound IPsec operations put the IV explicitly at the front of the packet. For an inbound IPsec operation, loading from the input buffer is most efficient and always used.

7. CBC processing must not use a predictable IV. Do not use the saved IV and previous result IV options for CBC processing. Refer to RFC 3602 for more details.

Hash State Loading

The hash state can be loaded from various sources, depending on the selected protocol and hash algorithm. The ***Different Sources for Loading the Hash State*** table provides a list of all the options.

Table 15-16: Different Sources for Loading the Hash State

Hash Algorithm `PKTE_SA_CMD0.HASHSRC =`	**From SA** **0b00**	**RESERVED** **0b01**	**From State** **0b10**	**No Load** **0b11**
SHA-1	yes	-	yes	Yes
SHA-224	Yes	-	Yes	Yes
SHA-256	Yes	-	Yes	Yes
Null hash	x	x	x	x

Sequence Number Processing

The packet engine supports sequence number generation and verification for IPsec and extended SSL, TLS, and DTLS protocol operations.

A Sequence Number (SN) is an unsigned number that a sender must implement and a receiver can use to support anti-replay service (replay attacks) for a specific SA. This processing includes detection of the same packet that arrives more than once and detection of packets that arrive in an incorrect sequence and is outside an accepted level of order relative to the last received packet.

The packet engine supports the following options.

- IPsec ESP with 32-bit SN generation, verification, and overflow protection (see RFC 4303).
- Sequence number loaded from SA for outbound operations and is retrieved from the input packet for inbound operations.

The sequence number and sequence number mask fields are part of the SA record.

Sequence Number Processing in IPsec

IPsec uses a 32-bit explicit sequence number that is sent in the packet. For outbound operations, with header processing (`PKTE_SA_CMD0.HDRPROC`) enabled, the packet engine first increments the sequence number from the `PKTE_SA_SEQNUM[n]` registers in the SA, then inserts the sequence number in the packet. Then the packet engine stores the incremented sequence number in the `PKTE_SA_SEQNUM[n]` registers in the SA. The sequence number mask `PKTE_SA_SEQNUM_MSK[n]` registers in the SA are not used.

For outbound operations, with the `PKTE_SA_CMD0.HDRPROC` bit =1 and the anti-replay service bit (`PKTE_SA_CMD1.ENSQNCHK`) =1, the packet engine generates a sequence number error

(PKTE_CTL_STAT.SQNMERR). The packet engine generates the error when the 32-bit sequence number counter overflows (counter is 2^{32}–1 and increments to 0). The host processor must not send the packet.

For outbound operations, when the PKTE_SA_CMD0.HDRPROC bit =0 or the PKTE_SA_CMD1.ENSQNCHK bit =0, the packet engine does not increment and verify a sequence number counter overflow. It, therefore, never generates a sequence number error. When the PKTE_SA_CMD0.HDRPROC bit =0, the packet engine does not update the sequence number and sequence number mask fields in the SA. It expects the host processor to provide the sequence number through the input buffer as part of the header.

For inbound operations, when the PKTE_SA_CMD0.HDRPROC bit =1 and the PKTE_SA_CMD1.ENSQNCHK bit =1, the packet engine verifies the PKTE_SA_SEQNUM[n] bit field against the sequence number in the packet. It uses the PKTE_SA_SEQNUM_MSK[n] value from the SA. Three situations can occur:

1. If the received sequence number falls outside and above the 64-bit sequence number mask, the mask is shifted. The packet engine updates the SA with the received sequence number and the shifted sequence number mask.
2. If the received sequence number falls inside the 64-bit sequence number mask and is not a duplicate sequence number, the corresponding bit in the mask is set. (A duplicate sequence number is the same as one previously received). The packet engine updates the SA with the received sequence number and the updated sequence number mask.
3. If the received sequence number falls outside and below the 64-bit sequence number mask or matches an earlier received number, a sequence number error is generated. The packet engine does not update the sequence number and sequence number mask fields in the SA. The host must discard the packet.

For inbound operations, when the PKTE_SA_CMD0.HDRPROC bit =0 or the PKTE_SA_CMD1.ENSQNCHK bit =0, the packet engine does not increment and verify the sequence number against the sequence number in the packet. It, therefore, never generates a sequence number error. When the PKTE_SA_CMD0.HDRPROC bit =0, the packet engine does not update the sequence number and sequence number mask fields in the SA. It expects the host to provide the sequence number through the input buffer as part of the header.

PKTE Block Diagram

The ***PKTE Block Diagram*** shows the functional blocks within the PKTE.

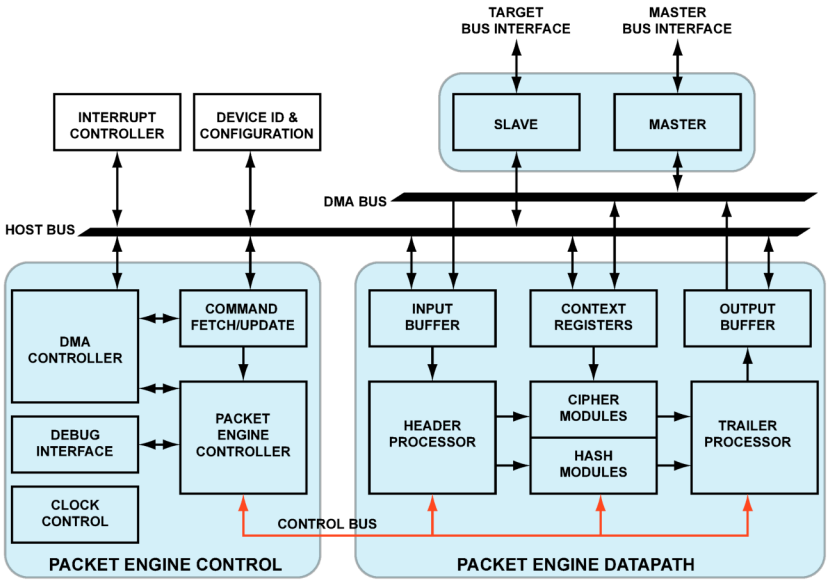

Figure 15-1: PKTE Block Diagram

PKTE Architectural Concepts

The following descriptions provide details on the functional blocks within the security packet engine.

Packet Engine

The packet engine contains symmetric cipher and hash engines. It is optimized to off-load intensive cryptographic operations from the host processor. It can perform parallel and pipelined encryption and hashing operations, reducing the latency, and processing time for packets that need both operations applied. The processor core provides the command information and packet data for the packet engine. The packet engine can run autonomously, using its local DMA controller to perform DMA transfers across the main bus to access the memory of the processor core. The DMA process incorporates flow-control to guarantee proper data flow. Two elements provide the command information that defines the processing for each packet:

- Command descriptor
- SA record

When the packet engine finishes the operation, it updates the SA record, when needed, and provides a result descriptor.

The packet engine has four different modes of operation. The modes give the processor core various levels of control over the command information and packet data transfers to and from the packet engine.

- Autonomous Ring Mode (ARM)
- Target Command Mode (with and without result descriptor ring) (TCM)
- Direct Host Mode (DHM)

Input/Output FIFO Buffers

The data for the packet engine is buffered at both input and output. These buffers decouple the DMA I/O process from the cipher and hash modules inside the packet engine. This functionality enables large DMA burst sizes and allows the crypto-engines to process data during I/O latency periods. Data moves automatically from the input buffer through the encryption and hash engines to the output buffer. If the output buffer is full, the process stops until the data is read and space is available in the output buffer. Each buffer is a 256-byte dual-port RAM.

Parallel Operations

The hash functionality and encrypt or decrypt functionality are tightly coupled. Operations that combine both hash and encrypt or decrypt functions are available to reduce processing time for data that must apply both. For hash-then-decrypt operations, the packet engine performs parallel execution of both functions from the input buffer. For encrypt-then-hash operations, the processing is pipelined from the input buffer to provide minimum latency. An offset can be specified between the start of the hashing and the start of the encryption to support protocols such as IPSec and SRTP.

DMA Controller

The packet engine uses a high-performance DMA controller for autonomous data transfers for:

- Command descriptor reads
- SA record and state record reads
- Packet data read
- Result packet writes
- SA record and state record writes
- Result descriptor writes

Interrupt Controller

The packet engine includes an interrupt controller that, under programmable configuration control, can generate an interrupt on completion of certain operations. Individual interrupts can be masked and cleared. The interrupt registers show both the raw and masked interrupt status of the internal interrupts. The processor core can use interrupts, together with their associated threshold settings, to optimize the overall packet processing in the system. One interrupt can inform the processor core that the input side of the packet engine is almost empty to avoid a stall-on-empty condition. One interrupt can inform the host that the output side is almost full to avoid a stall-on-full condition. The controller uses several interrupts to inform the processor core about errors inside the packet engine. All available interrupts are combined into a single output port as either a level- or edge-active programmable interrupt output.

Clock Controller

The packet engine includes a clock controller that generates clock enable signals. A clock manager external to the packet engine uses the clock enable signals to switch the clocks to modules in the packet engine, reducing power

consumption. The power saving can be significant, depending on the crypto-operation and the idle time of the packet engine. The clock controller generates the clock enable signals dynamically depending on the current crypto-operation. A clock control register provides the processor core the possibility to override this dynamic process.

PKTE Operating Modes

The packet engine can be configured in one of three command modes. For all modes the packet engine can generate an interrupt at completion of packet processing:

- Autonomous Ring Mode (ARM). The core prepares descriptors in the CDR and then initiates a descriptor fetch by triggering the packet engine. When a packet operation is complete, the packet engine writes the result descriptor out into a ring in host memory using the system master bus interface.
- Target Command Mode (TCM). The core directly writes the command descriptors to the packet command register set to initiate a packet operation. This process eliminates an extra DMA transfer to fetch a descriptor, but requires the core to synchronously initiate packet processing. This mode can be configured both without RDR or with RDR. In the latter case, the packet engine writes the result descriptor out into the RDR in host memory using the system master bus interface.
- Direct Host Mode (DHM). The core has full control over the packet engine and uses the system slave bus interface. The core provides all the command descriptors, SA record and state record, and packet input data. When the packet engine completes processing, the core must read the packet output data, the SA record, the state record, and the result descriptors.

Autonomous Ring Mode (ARM)

The *Autonomous Ring Mode* allows the packet engine and the host processor to operate asynchronously. A queue of multiple packets in the host processor memory can be processed continuously to provide the highest possible throughput. The packet engine autonomously fetches the command descriptor, the SA record, and optionally the state record and the input data from host processor memory. After the packet engine finishes processing, it autonomously writes the output data, updates the SA record and state record and writes the result descriptor in the host processor memory. It accesses the host processor memory through DMA read transfers across the system bus master.

This mode uses both command descriptor ring (CDR) and result descriptor ring (RDR).

Physically the CDR, RDR, SA record, source packet, and result packet can all be in different memories depending on the system memory architecture. The host processor writes command descriptors to the CDR in host processor memory. Then, it writes to the `PKTE_CDSC_INCR` register with the number of command descriptors that it prepared in the CDR. This write to the `PKTE_CDSC_INCR` register is the trigger for the packet engine to fetch the command descriptors sequentially from the CDR. When a command descriptor is fetched and written to the internal packet command register set, the descriptor is validated. If the ownership bits are set for the packet engine and the command is valid, processing starts. If not, that command is discarded and a result descriptor is written to the RDR with the error code *invalid command descriptor*. In this mode, the host processor can set a threshold on the CDR and enable an associated interrupt. The packet engine generates an interrupt when the number of command descriptors in the CDR is equal or below the threshold value.

The SA record and state record that contains the crypto context information are stored in a memory area. The packet engine autonomously accesses the memory area through DMA transfers across the system bus master. Also, the source packet and result packet are stored in a memory area that the packet engine autonomously accesses through the same bus master interface.

After decoding the command descriptor, the packet engine fetches the SA record and then, optionally, the state record.

Then, the source packet is fetched and stored in the input buffer. Packets less than the size of the input buffers are fetched entirely at once. Larger packets are fetched in parts that completely fill the input buffer. The packet engine initiates a new fetch each time the number of empty spaces in the input buffer reaches its threshold value. When the first packet data is available in the input buffer, the crypto engines start processing the data. After processing, the crypto engines write the result packet to the output buffer.

The packet engine writes the result packet from the output buffer to host processor memory when the number of bytes in the output buffer reaches its threshold value. Packets less than the threshold value are written entirely at once. Larger packets are written in parts that completely empty the output buffer

The source packet data fetching, data processing, and result packet data writing are parallel processes that continue until the last result packet is written to host processor memory. Then, the packet engine optionally writes the SA record and the state record to update the crypto context information. As a final step, the packet engine writes the result descriptor to the RDR. The host processor must either poll the RDR or wait for an interrupt from the packet engine to determine when packet processing is complete.

Target Command Mode (TCM)

This mode provides a synchronous interface between the processor core and the packet engine. The Command Descriptor Ring (CDR) is disabled and the processor core initiates packet processing by writing the command descriptor directly to the internal command descriptor MMRs of the packet engine. The Result Descriptor Ring (RDR) is optional.

- For `PKTE_CFG.MODE` =01, the RDR is disabled and the processor core reads the result descriptor directly from the internal result descriptor register set for the packet engine.
- For `PKTE_CFG.MODE` =10, the RDR is enabled and stored in a memory area that the packet engine can access through its master bus interface as in autonomous ring mode.

In target command mode, the packet engine autonomously fetches the SA record, state record, source packet data as in autonomous ring mode. Also, as in ARM, after processing, the packet engine updates state fields of the SA record and state record in the host processor memory.

Direct Host Mode (DHM)

This mode provides a synchronous interface between the processor core and the packet engine. The packet engine is under full control of the processor core. The host processor writes the command descriptors, SA record, and state record directly to the packet engine registers. Then, the processor core writes the source packet data into the input buffer. When processing is complete, the processor core reads back the result packet data from the output buffer.

Finally, it reads the result descriptor, the updated SA record, and state record directly from the packet engine registers.

PKTE Event Control

The following section provides information about interrupts in the PKTE module.

PKTE Interrupt Signals

The Packet Engine has an internal Interrupt Controller with 9 interrupt sources. There are 7 registers associated with the interrupt controller:

1. Interrupt Unmasked Status - PKTE_IUMSK_STAT
2. Interrupt Mask Status - PKTE_IMSK_STAT
3. Interrupt Clear Register - PKTE_INT_CLR
4. Interrupt Enable Register - PKTE_INT_EN
5. Interrupt Mask Disable - PKTE_IMSK_DIS
6. Interrupt Mask Enable - PKTE_IMSK_EN
7. Interrupt Configuration - PKTE_INT_CFG

Below shows a block diagram of the interrupt controller

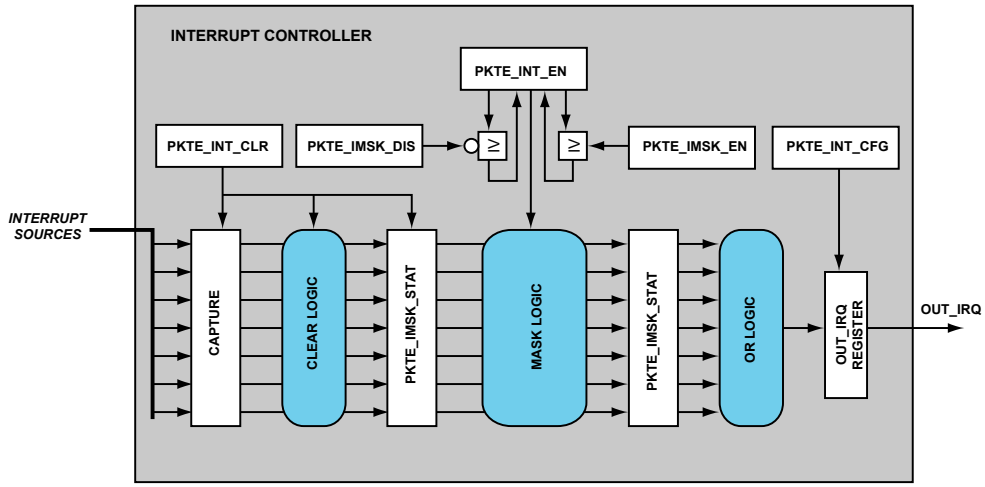

Figure 15-2: Block Diagram of Packet Engine Interrupt Controller

All of the interrupt sources are pulse or level events in their native form.

These interrupts are captured and stored at their unmasked and masked status in their respective PKTE_IUMSK_STAT and PKTE_IMSK_STAT registers. This allows the host processor to read the status of any interrupt source either before or after the mask is applied.

The PKTE_INT_EN register provides a mask to select what interrupt source are enabled to the output interrupt request. Writing a one to the PKTE_INT_CLR register resets both the masked and unmasked interrupt.

The PKTE_IMSK_EN and PKTE_IMSK_DIS registers can be set to enable and disable individual interrupts respectively in the PKTE_INT_EN register. This avoids the need for read-modify-write operations from the host processor.

Ring Interrupts

Two interrupts are provided for efficient ring management: The CDR threshold interrupt (*cdrthrs*h) and the RDR threshold interrupt (*rdrthrsh*).

Command Descriptor Ring

The CDR threshold interrupt (cdrthrsh) is a level-based interrupt and connects to the threshold value in the PKTE_RING_THRESH.CDRTHRSH bit field. It enables the host processor to efficiently fill the CDR. The host processor writes command descriptors to the CDR with this interrupt masked until the CDR is full. Then the host processor enables the CDR threshold interrupt. When the interrupt is activated, the host processor clears the interrupt and it is guaranteed that it can put CDR threshold number of descriptors in the CDR.

Example Configuration:

```
PKTE_RING_CFG.RINGSZ=256,
PKTE_RING_THRESH.CDRTHRSH=224,
PKTE_INT_EN.CDRTHRSH=0 /*(IRQ disabled)*/
```

1. The host writes 8 Command Descriptors at once, then writes the PKTE_CDSC_INCR register to 8.
2. This is repeated until there are less than 8 empty entries in the CDR.
3. The fill level is now equal to he threshold (224)
4. The host enables the CDR threshold IRQ (PKTE_INT_EN.CDRTHRSH=1)
5. The packet engine processes packets and the fill level (PKTE_CDSC_CNT register) decreases.
6. Then the CDR threshold IRQ is activated as the fill level equals 224.
7. The host handles the interrupt, clears it and continues with step 1.

Result Descriptor Ring

The RDR threshold interrupt (rdrthrsh) is a level-based interrupt and connects to the threshold value in the PKTE_RING_THRESH.RDRTHRSH bit field and the timeout value in the RD timeout (PKTE_RING_THRESH.RDTO) bit field. It enables the host processor to efficiently empty the RDR. The timeout reminds the host processor that when the threshold kicks in result descriptors stay long in the RDR and must be processed to reduce latency. The timeout counts when the ring is not empty, regardless of the fill level and restarts when the host processor writes the PKTE_RDSC_CNT register. Initially the host processor enables the RDR threshold interrupt. When the interrupt is activated the host processor reads result descriptors until the RDR is empty or contains less than the PKTE_RING_THRESH.RDRTHRSH number of descriptors.

Example Configuration:

```
PKTE_RING_CFG.RINGSZ=256,
PKTE_RING_THRESH.RDRTHRSH=32, timeout=1ms
PKTE_INT_EN.RDRTHRSH=1 /*(IRQ enabled)*/
```

1. The Packet Engine writes the Result Descriptor, timeout counter starts.
2. The Packet Engine writes 32 more Result Descriptors, fill level (PKTE_RDSC_CNT register) increases to 33.
3. The fill level exceeds threshold within 1ms, the RDR threshold IRQ is activated.
4. The host handles the interrupt, reads 8 Result Descriptors at once, then writes the PKTE_RDSC_DECR register with 8. The write to the PKTE_RDSC_DECR register restarts the timeout counter. The fill level is now under the threshold but there are still 25 descriptors left.
5. The Packet Engine writes 8 more Result Descriptors, fill level increases to 33.
6. The fill level exceeds threshold within 1 ms, the RDR threshold IRQ is activated.
7. The host handles the interrupt, reads 8 Result Descriptors at once, then writes the PKTE_RDSC_DECR register with 8. The write to the PKTE_RDSC_DECR register restarts the timeout counter. The fill level is now under the threshold but there are still 25 descriptors left.
8. After 1 ms, the timeout counter interrupt is activated.
9. The host handles the interrupt, reads 8 Result Descriptors at once, then writes the PKTE_RDSC_DECR register with 8. This is repeated until there are less than 8 full entries in the RDR. The fill level is now under the threshold and the RDR threshold IRQ interrupt is inactive. Each write to the PKTE_RDSC_DECR register restarts the timeout counter.

PKTE Programming Model

The host processor must always follow a pre-defined sequence of five phases required by the packet engine on a per packet basis when using direct host mode. The following sections describe the five phases.

Phase 1. Write the Command Descriptor

1. Write the first command descriptor word with status and control information to the PKTE_CTL_STAT register.
2. Optionally, write the user ID to the PKTE_USERID register.
3. Write the last descriptor word to the PKTE_LEN register.
4. Write the value 0x1 to the PKTE_CDSC_CNT register. This opertion triggers the packet engine to validate the command descriptor. If the command descriptor is invalid, an error is generated. (See the PKTE_CTL_STAT section in the Register Descriptions). If the command descriptor is valid, the packet engine waits for an PKTE_SA_RDY register write.

Phase 2. Write the State Registers and SA Registers

All required fields of the SA record and state record must be written. The fields required depend on the operation. The last field to be written is the PKTE_SA_RDY register. This register triggers the packet engine to start processing.

1. Write the required state record fields.
    - IV
    - Digest count
    - State digest
2. Write the required SA record data.
3. To complete the SA record and state record, write the PKTE_SA_RDY register.

Phase 3. Write the Source Packet Data and Read Result Packet Data

The packet engine has input and output buffers. If a source packet is smaller than the size of the input buffer, then the packet can be written in one part. Otherwise, it must be written in multiple parts. The same applies to the output data. If the result packet size is smaller than the size of the output buffer, then the packet can be read in one part. Otherwise, it must be read in multiple parts.

NOTE: An outbound packet that is smaller than the size of the input buffer can increase in size due to padding and does not always fit in the output buffer. Conversely, an inbound packet that is larger than the size of the input buffer can decrease in size, and due to de-padding, can fit in the output buffer. If the input buffer becomes empty or the output buffer becomes full, the engine stalls.

Two following steps describe different situations:

- Source packet smaller than the size of the input buffer, start at step 1.
- Source packet larger than the size of the input buffer, start at step 3.

The host processor must follow these steps:

1. Write the source packet data. Write the full source packet to the input buffer. Go to step 4.
2. Write the input buffer count register (PKTE_INBUF_CNT) with the number of valid bytes that are written to the input buffer. This value must correspond to the value in thePKTE_LEN.TOTLEN field of the command descriptor rounded up to the next multiple of 4 bytes. Go to step 5 to check the packet engine status.
3. Write part of the source packet data. The PKTE_STAT.IBUFEMPTYCNT field indicates the amount of free space in the input buffer. Programs write the number of bytes determined by the setting in the PKTE_BUF_THRESH register. Write the (partial) source packet to the input buffer. The host processor must resume where it ended the previous write operation. Do not write more than the buffer size at once. Go to step 4.
4. Write the PKTE_INBUF_CNT register with the number of valid bytes written to the input buffer. Go to step 5 to check the packet engine status.
5. Check packet engine status. Wait for an interrupt or poll the PKTE_STAT register for any of the following conditions:

- Condition 1 - An error interrupt or any of the bits [7:5] in the PKTE_STAT register becomes active to indicate a packet processing error. Depending on the type of error, the host processor must take appropriate action. Usually, the result packet is not valid after a processing error has occurred. Go to phase 5 to read the result descriptor.
- Condition 2 - An operation done interrupt or the PKTE_STAT.OPDN bit becomes active (the packet engine completed processing). Go to step 8 to read the remaining output data.
- Condition 3 - An output buffer threshold interrupt or the PKTE_STAT.OBUFREQ bit becomes active. Go to step 6 to read a block of output data.
- Condition 4 - An input buffer threshold interrupt or the PKTE_STAT.IBUFREQ bit becomes active. Go to step 3 to write a block of input data.

6. Read part of the output data. The PKTE_STAT.OBUFFULLCNT bit field indicates the number of bytes in the output buffer. This value is rounded up to full words. Programs read the number of bytes indicated in the PKTE_BUF_THRESH register. Read the (partial) output packet from the output buffer. The host processor must resume where it ended the previous read operation. Do not read more than the input buffer size at once. Go to step 7.
7. Write the PKTE_OUTBUF_CNT register with the number of valid bytes read from the output buffer. Go to step 5 to check the packet engine status.
8. Read the remaining output data. The PKTE_STAT.OBUFFULLCNT bit field indicates the number of bytes in the output buffer. This value is rounded up to words. Read the (partial) packet output data from the output buffer. The host processor must resume where it ended the previous read operation. Go to step 9.
9. Write the PKTE_OUTBUF_CNT register with the number of valid bytes read from the output buffer. Go to phase 4.

Phase 4. Read the Result Descriptor

1. Read the first result descriptor word from the PKTE_CTL_STAT.
2. Optionally read the user ID from the PKTE_USERID register.
3. Read the last result descriptor word from the PKTE_LEN register.
4. Write the value 0x1 to the PKTE_RDSC_DECR register. This operation allows the packet engine to accept new command descriptors. Go to phase 5.

Phase 5. Read the SA Record and State Record

Depending on the operation, the SA record or state record is updated. Check the bit fields [23:16] in the PKTE_CTL_STAT register for the following conditions:

- Condition 1 - At least one error bit in the bit fields [23:16] of the PKTE_CTL_STAT register is set. Do not update the local host processor maintained version of the SA record and state record but take any required action.

- Condition 2 - None of the error bits in the bit fields [23:16] of the PKTE_CTL_STAT register is set, the packet is processed normally (without errors). Update the host processor maintained version of the SA record and state record with the result read from the packet engine registers:
  - Sequence number
  - Sequence number mask
  - Result IV
  - Result digest count
  - Result digest

PKTE Mode Configuration

Before using the packet engine, it must be configured. The mode of the packet engine must be defined and the PRNG (if used) must be initialized.

Configure the packet engine in one of three command modes:

- Autonomous Ring Mode: PKTE_CFG.MODE =b'11
- Target Command Mode: PKTE_CFG.MODE =b'10
- Direct Host Mode: PKTE_CFG.MODE =b'00

PKTE Programming Concepts

The following sections provide conceptual information for programming the PKTE.

Packet Engine Descriptor

IMPORTANT: Depending on the mode, ARM, TCM, or DHM, the descriptor is either:

- in the memory of the host processor in the command descriptor ring, or
- written directly to the descriptor registers in the packet engine

References to descriptor registers are for either the register that is mirrored in the descriptor structure in memory or for the actual register itself.

Command descriptors are host-supplied commands that control the real-time operation of the packet engine. The packet engine returns result descriptors at the end of an operation that provide the status information to the host. The ***Command Descriptor Structure*** and the ***Result Descriptor Structure*** tables show these descriptors.

Table 15-17: Command Descriptor Structure

Word Offset	31:24	23:20	19:16	15:8	7:0	Address Offset
0	Pad Control	—		Next Header/ Pad Value	Control	0x000
1	Source Address					0x004
2	Destination Address					0x008
3	SA Address					0x00C
4	SA State Address					0x010
5	Reserved					0x014
6	User ID					0x018
7	Bypass (words)	Control	Reserved	Input Packet Length (bytes)		0x01C

Table 15-18: Result Descriptor Structure

Word Offset	31:24	23:20	19:16	15:8	7:0	Address Offset
0	Pad Status	Status		Next Header/ Pad Value	Control	0x000
1	Source Address					0x004
2	Destination Address					0x008
3	SA Address					0x00C
4	SA State Address					0x010
5	Reserved					0x014
6	User ID					0x018
7	Bypass (words)	Control	Reserved	Input Packet Length (bytes)		0x01C

When the packet engine is configured for autonomous ring mode, command descriptors and result descriptors reside in a ring in host memory. Command descriptors are automatically fetched from the Command Descriptor Ring (CDR) through DMA into the command descriptor registers. When an operation is complete, the result descriptors are automatically read from the packet engine and through DMA to the Result Descriptor Ring (RDR).

When the packet engine is configured for direct host mode, the host processor manually writes the command descriptor directly to the internal command descriptor MMR set. When an operation is complete, the host processor manually reads the result descriptor directly from the result descriptor MMR set.

The target command mode is a combination of the direct host mode and the autonomous ring mode. The host processor writes the command descriptor directly to the internal command descriptor register set. When an operation is complete there are two options.

1. The host processor can read the result descriptor directly from the result descriptor registers.

2. The result descriptors reside in a ring in host memory and the result descriptors are automatically DMA'd from the packet engine to the RDR.

When the host processor writes a command descriptor to the command descriptor registers, the packet engine is triggered when the host processor updates the PKTE_CDSC_CNT register. This functionality guarantees that all fields in the command descriptor are valid before the command is executed.

Descriptor Processing

This section describes the functional steps of the packet engine while processing the command descriptors.

Descriptor Ring Configuration

At initialization, the host processor specifies the size of the Command Descriptor Ring (CDR). The Result Descriptor Ring (RDR) has the same size.

NOTE: In some configurations, these two rings overlay each other with the results written on top of the command descriptors. This configuration is called overlaid ring mode.

When the packet engine is configured and enabled, it fetches the descriptors from the CDR using system bus master reads.

Descriptor Ring Processing

To validate the descriptor exchange between the host processor and the packet engine, the ownership bits PKTE_CTL_STAT.PERDY and PKTE_CTL_STAT.HOSTRDY are used. One pair of ownership bits is in the first word of the descriptor (PKTE_CTL_STAT), and one pair is in the last word (PKTE_LEN). The 'consumer' of a descriptor must verify that both ownership pairs match to ensure that a race condition did not occur between one party writing and the other party reading the descriptor. A race condition can occur when a memory locking scheme is not used.

Each pair [PKTE_CTL_STAT.PERDY, PKTE_CTL_STAT.HOSTRDY] of ownership bits provide 3 states:

- b'00 = idle or null descriptor
- b'01 = host processor has written a descriptor in the CDR and passed ownership to the packet engine
- b'10 = packet engine processing complete: packet engine has written the descriptor in the RDR and passed ownership back to the host.
- b'11 = Reserved

At initialization, the host sets the entire CDR memory area to zero, when the CDR is used.

1. The host processor writes one or more command descriptors to the CDR. The host processor must set the PKTE_CTL_STAT.HOSTRDY bit to 1 and the PKTE_CTL_STAT.PERDY bit to 0 to indicate that ownership has passed to the packet engine. These bits are mirrored in the PKTE_LEN descriptor word.
2. The host processor must write the PKTE_CDSC_INCR register with the number of new valid command descriptors in the CDR.

3. The packet engine reads and validates one command descriptor.
4. The packet engine reads the SA record and state record, processes the packet and updates the SA record and state record.
5. If the rings are not overlaid and the PKTE_CFG.ENCDRUPDT bit is 1, the packet engine writes the result descriptor to the CDR with the PKTE_CTL_STAT.HOSTRDY bit set to 0 and PKTE_CTL_STAT.PERDY bit set to 1. These bits are mirrored in the PKTE_LEN descriptor word.
6. The packet engine writes the result descriptor to the RDR. The packet engine sets the PKTE_CTL_STAT.PERDY bit to 1 and the PKTE_CTL_STAT.HOSTRDY bit to 0 to indicate that the ownership has passed to the host. These bits are mirrored in the PKTE_LEN descriptor word.
7. The packet engine decrements the value in the PKTE_CDSC_CNT register. If the value is not zero, the packet engine reads with the next command descriptor (step 3).
8. The packet engine increments the value in the PKTE_RDSC_CNT register.
9. The host processor reads one or more result descriptors from the RDR and processes the results.
10. The host processor must write the PKTE_RDSC_DECR register with the number of processed result descriptors in the RDR.

Ownership of the Descriptor

The ownership of the command descriptor and result descriptor is set by ownership bits in the first and last word of the respective descriptor, in the PKTE_CTL_STAT register and the PKTE_LEN register. For the command descriptor, it is the processor core that sets the ownership to the packet engine. For the result descriptor, it is the packet engine that sets the ownership to the host processor.

The packet engine reads the ownership bits before processing, irrespective of the mode of the packet engine. The ownership bits are used to validate and identify the descriptor. When two separate rings are used, the packet engine can be programmed to clear the ownership bits of the command descriptors in the CDR so the host processor knows which descriptors are processed.

NOTE: This update of the ownership bits can be disabled in the PKTE_CFG register when the host processor actively counts the number of descriptors in the CDR to prevent ring wrapping.

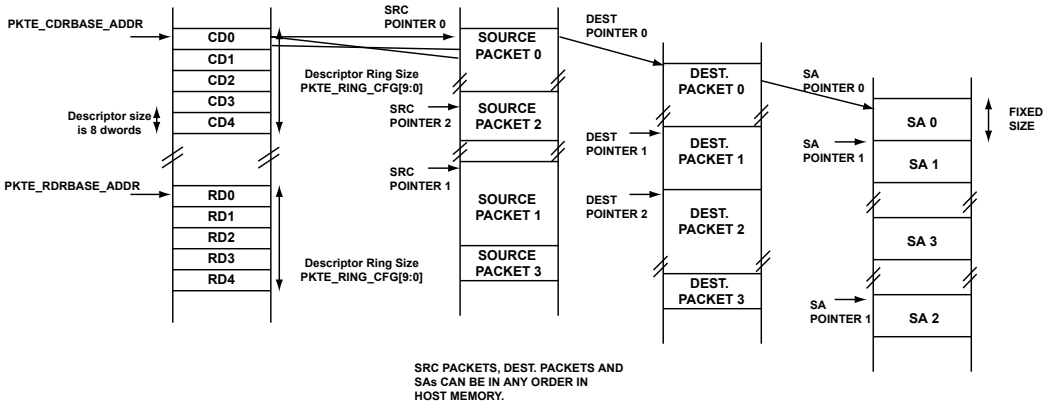

Figure 15-3: Descriptor Rings in Autonomous Ring Mode

Description and Use of the SA Record and State Record Structure

The SA record is a packed structure that contains the remainder of the information needed by the packet engine to process a packet. Most of the information fields in the SA record, such as the key and encryption mode, are static for the lifetime of the association. The fields do not require frequent manipulation by the host processor. The SA record non-static fields are the sequence number and sequence number mask.

The SA record can have a corresponding state record that is used to save results from the current operations that can be used for future operations. The state record can hold the IV, the hash byte count, and the intermediate hash digest.

There is no practical limit to how many SA records and corresponding state records the packet engine can support.

In the autonomous ring mode and target command mode, once the packet engine has validated a command descriptor, it automatically fetches the SA record and optional state record. After processing, the packet engine updates the stateful fields in the SA record and state record in the host processor memory.

In direct host mode, after the descriptor is validated, the host must write the SA record directly into the internal registers of the packet engine. After processing, the host reads the stateful fields from the SA registers of the packet engine and saves them back to the SA record in the host processor memory.

SA Record Structure

The *SA Record Structure* table shows the structure for an SA Record. When using direct host mode, the corresponding elements are accessed directly with the registers. When using autonomous ring mode or target command mode, the SA Record is defined, configured and accessed in host memory.

Table 15-19: SA Record Structure

Word Offset	Description (name)	Use
0	PKTE_SA_CMD0[31:0]	SA Control word 0 (all operations)
1	PKTE_SA_CMD1[31:0]	SA Control word 1 (all operations)

Table 15-19: SA Record Structure (Continued)

Word Offset	Description (name)	Use
2	PKTE_SA_KEY0[31:0]	Key word (DES, Triple-DES, AES-128/192/256,)
3	PKTE_SA_KEY1[63:32]	Key word (DES, Triple-DES, AES-128/192/256,)
4	PKTE_SA_KEY2[95:64]	Key word (Triple-DES, AES-128/192/256,)
5	PKTE_SA_KEY3[127:96]	Key word (Triple-DES, AES-128/192/256,)
6	PKTE_SA_KEY4[159:128]	Key word (Triple-DES, AES-192/256)
7	PKTE_SA_KEY5[191:160]	Key word (Triple-DES, AES-192/256)
8	PKTE_SA_KEY6[223:192]	Key word (AES-256)
9	PKTE_SA_KEY7[255:224]	Key word (AES-256)
10	PKTE_SA_IDIGEST0[31:0]	Inner Hash digest (Basic Hash and HMAC with ,SHA-1, SHA-224, SHA-256)
11	PKTE_SA_IDIGEST1[63:32]	Inner Hash digest (Basic Hash and HMAC with ,SHA-1, SHA-224, SHA-256)
12	PKTE_SA_IDIGEST2[95:64]	Inner Hash digest (Basic Hash and HMAC with ,SHA-1, SHA-224, SHA-256)
13	PKTE_SA_IDIGEST3[127:96]	Inner Hash digest (Basic Hash and HMAC with ,SHA-1, SHA-224, SHA-256)
14	PKTE_SA_IDIGEST4[159:128]	Inner Hash digest (Basic Hash and HMAC with SHA-1, SHA-224, SHA-256)
15	PKTE_SA_IDIGEST5[191:160]	Inner Hash digest (Basic Hash and HMAC with SHA-224, SHA-256)
16	PKTE_SA_IDIGEST6[223:192]	Inner Hash digest (Basic Hash and HMAC with SHA-224, SHA-256)
17	PKTE_SA_IDIGEST7[255:224]	Inner Hash digest (Basic Hash and HMAC with SHA-256)
18	PKTE_SA_ODIGEST0[31:0]	Outer Hash digest (HMAC with , SHA-1, SHA-224, SHA-256)
19	PKTE_SA_ODIGEST1[63:32]	Outer Hash digest (HMAC with , SHA-1, SHA-224, SHA-256)
20	PKTE_SA_ODIGEST2[95:64]	Outer Hash digest (HMAC with , SHA-1, SHA-224, SHA-256)
21	PKTE_SA_ODIGEST3[127:96]	Outer Hash digest (HMAC with , SHA-1, SHA-224, SHA-256)
22	PKTE_SA_ODIGEST4[159:128]	Outer Hash digest (HMAC with SHA-1, SHA-224, SHA-256)
23	PKTE_SA_ODIGEST5[191:160]	Outer Hash digest (HMAC with SHA-224, SHA-256)

Table 15-19: SA Record Structure (Continued)

Word Offset	Description (name)	Use
24	PKTE_SA_ODIGEST6[223:192]	Outer Hash digest (HMAC with SHA-224, SHA-256)
25	PKTE_SA_ODIGEST7[255:224]	Outer Hash digest (HMAC with SHA-256)
26	PKTE_SA_SPI[31:0]	SPI (IPsec), Type[23:16] / Version [15:0] (SSL, TLS, DTLS)
27	PKTE_SA_SEQNUM0[31:0]	Sequence Number (IPsec, SSL, TLS, DTLS with Header Processing)
28	PKTE_SA_SEQNUM1[63:32]	
29	PKTE_SA_SEQNUM_MSK0[31:0]	Sequence Number Mask (IPsec, DTLS inbound with Header Processing)
30	PKTE_SA_SEQNUM_MSK1[63:32]	
31	PKTE_SA_NONCE[31:0] / PKTE_SA_READY	Nonce value (AES-CTR, AES-ICM)/ i & j pointers ()/SA ready indicator (Direct Host Mode)

Some of these fields may be updated by the packet engine. These include:

- PKTE_SA_SEQNUM0
- PKTE_SA_SEQNUM1
- PKTE_SA_SEQNUM_MSK0
- PKTE_SA_SEQNUM_MSK1

All the other fields remain unchanged.

SA State Structure

The security association state structure contains information that may be updated after each packet, such as the IV and the intermediate hash result. The ***SA State Structure*** table shows the SA state structure and usage. In direct host mode, the elements are accessed directly using the PKTE registers. In target command mode and autonomous ring mode, this structure is defined and updated in host memory.

Table 15-20: SA State Structure

Word Offset	Description (name)	Use
0	PKTE_STATE_IV0[31:0]	Initialization Vector (DES, Triple DES, AES)
1	PKTE_STATE_IV1[63:32]	Initialization Vector (DES, Triple DES, AES)
2	PKTE_STATE_IV2[95:64]	Initialization Vector (AES)
3	PKTE_STATE_IV3[127:96]	Initialization Vector (AES)
4	PKTE_STATE_BYTE_CNT0[31:0]	Current hash byte count (SHA-1, SHA-224, SHA-256)
5	PKTE_STATE_BYTE_CNT1[63:32]	Current hash byte count (SHA-1, SHA-224, SHA-256)

Table 15-20: SA State Structure (Continued)

Word Offset	Description (name)	Use
6	PKTE_STATE_IDIGEST0[31:0]	Inner Hash digest (mirror of PKTE_SA_IDIGEST0)
7	PKTE_STATE_IDIGEST1[63:32]	Inner Hash digest (mirror of PKTE_SA_IDIGEST1)
8	PKTE_STATE_IDIGEST2[95:64]	Inner Hash digest (mirror of PKTE_SA_IDIGEST2)
9	PKTE_STATE_IDIGEST3[127:96]	Inner Hash digest (mirror of PKTE_SA_IDIGEST3)
10	PKTE_STATE_IDIGEST4[159:128]	Inner Hash digest (mirror of PKTE_SA_IDIGEST4)
11	PKTE_STATE_IDIGEST5[191:160]	Inner Hash digest (mirror of PKTE_SA_IDIGEST5)
12	PKTE_STATE_IDIGEST6[223:192]	Inner Hash digest (mirror of PKTE_SA_IDIGEST6)
13	PKTE_STATE_IDIGEST7[255:224]	Inner Hash digest (mirror of PKTE_SA_IDIGEST7)

Configuring Operations in the PKTE

The operation (cipher, hash function, and others) that the PKTE performs is configured primarily in the PKTE_SA_CMD0 register. The following sections include a series of tables to help configure the least significant 16 bits of the PKTE_SA_CMD0 register. These fields include:

- The operation code field (PKTE_SA_CMD0.OPCD)
- The direction field (PKTE_SA_CMD0.DIR)
- The operation group field (PKTE_SA_CMD0.OPGRP)
- The padding type (PKTE_SA_CMD0.PADTYPE)
- The cipher selection (PKTE_SA_CMD0.CIPHER)
- The hash selection (PKTE_SA_CMD0.HASH)

Basic Operations and Decoding

Table 15-21: Basic Operation Decoding

Outbound				Inbound			
OpGroup	Dir	OpCode	Operation	OpGroup	Dir	OpCode	Operation
0b00	0	0b000	Encrypt	0b00	1	0b000	Decrypt
0b00	0	0b001	Encrypt - Hash	0b00	1	0b001	Hash - Decrypt
0b00	0	0b010	Reserved	0b00	1	0b010	Reserved
0b00	0	0b011	Hash	0b00	1	0b011	Hash
0b00	0	0b100... 0b110	Reserved	0b00	1	0b100... 0b110	Reserved

Table 15-21: Basic Operation Decoding (Continued)

Outbound				Inbound			
OpGroup	**Dir**	**OpCode**	**Operation**	**OpGroup**	**Dir**	**OpCode**	**Operation**
0b00	0	0b111	PRNG	0b00	1	0b111	Reserved

Table 15-22: Protocol Operation Decoding

Outbound				Inbound			
OpGroup	**Dir**	**OpCode**	**Operation**	**OpGroup**	**Dir**	**OpCode**	**Operation**
0b01	0	0b000	ESP Outbound	0b01	1	0b000	ESP Inbound
0b01	0	0b001... 0b011	Reserved	0b01	1	0b001	Reserved
0b01	0	0b100	Basic SSL Outbound	0b01	1	0b010	Basic SSL Inbound
0b01	0	0b101	Basic TLS Outbound	0b01	1	0b011	Basic TLS Inbound
0b01	0	0b110	Reserved	0b01	1	0b100... 0b110	Reserved
0b01	0	0b111	SRTP Outbound	0b01	1	0b111	SRTP Inbound

NOTE: For SSL/TLS and SRTP, no header processing is performed in hardware.

Table 15-23: Extended Protocol Operation Decoding

Outbound				Inbound			
OpGroup	**Dir**	**OpCode**	**Operation**	**OpGroup**	**Dir**	**OpCode**	**Operation**
0b11 0	0	0b000	Reserved	0b11	1	0b000	Reserved
0b11 0	0	0b001	DTLS Outbound	0b11	1	0b001	DTLS Inbound
0b11 0	0	0b010... 0b011	Reserved	0b11	1	0b010... 0b011	Reserved
0b11 0	0	0b100	Ext. SSL Outbound	0b11	1	0b100	Ext. SSL Inbound
0b11 0	0	0b101	Ext. TLS v1.0 Outbound	0b11	1	0b101	Ext. TLS v1.0 Inbound

Table 15-23: Extended Protocol Operation Decoding (Continued)

Outbound				Inbound			
OpGroup	Dir	OpCode	Operation	OpGroup	Dir	OpCode	Operation
0b11 0	0	0b110	Ext. TLS v1.1 Outbound	0b11	1	0b110	Ext. TLS v1.1 Inbound
0b11 0	0	0b111	Reserved	0b11	1	0b111	Reserved

Error Code Description

The PKTE_CTL_STAT register is used to configure the packet engine for processing in Direct Host Mode (DHM) or Target Command Mode (TCM). The PKTE_CTL_STAT structure element in memory is used when the packet engine is configured for Autonomous Ring Mode (ARM). In both cases, when an operation is started, errors are reported in the status field (bits [23:16]) of this register or structure element. The ***Extended Error Codes - Status Encoding*** table provides a guide on how to decipher the meaning of the bits that are set when an error occurs.

Extended Error Codes

The following table provides information about the extended errors associated with the PKTE module.

Table 15-24: Extended Error Codes - Status Encoding

STATUS bits [23:16]	Hex Value	Priority	Description	Processing Result
0b0000_0000	0x00	NA	Successful completion. No errors occurred during processing of the packet.	Packet fully processed
0b----_---1	0x-1	NA	Authentication Error. For an inbound IPsec ESP operation, the Integrity Check Value (ICV) does not match the computed value. For an inbound SRTP operation, the authentication tag does not match the computed value. For a basic SSL/TLS, Extended SSL/TLS or DTLS operation the Message Authentication Code (MAC) does not match the computed value.	Packet fully processed
0b----_--1-	0x-2	NA	Pad Verify Error. For inbound operations that use pad type Constant TLS, IPsec or PKCS#7, the decrypted pad does not match the expected values for the selected pad type.	Packet fully processed
0b----_-1--	0x-4	NA	Sequence Number Error. For an inbound IPsec or DTLS operation, there was a fault in the Anti-Replay Sequence Number.	Packet fully processed

Table 15-24: Extended Error Codes - Status Encoding (Continued)

STATUS bits [23:16]	Hex Value	Priority	Description	Processing Result
			For an outbound IPsec packet, the sequence number overflows; count is 2^{32}– 1 and increments to 0. For an outbound DTLS operation, the sequence number overflows; count is 2^{48}– 1 and increments to 0. For an outbound SSL or TLS operation, the sequence number overflows; count is 2^{64}– 1 and increments to 0.	
0b0000_1---	0x08	1	System Bus error. The master bus interface generates an error due to ERROR response from system slave. The slave bus interface generates an error due to request for non-word (32-bit) access.	Packet is aborted. The host must reject the packet and apply a hardware reset to the system.
0b0001_1---	0x18	2	Invalid Command Descriptor Error. The ownership bits in the command descriptor are not set to the packet engine, after the PKTE_CDSC_CNT register is incremented.	Command descriptor is ignored, no packet is processed. The packet must be re-queued or discarded.
0b0010_1---	0x28	3	Invalid Crypto Operation Error. A reserved operation is selected.	The SA record is ignored, no packet is processed. The packet must be re-queued or discarded.
0b0011_1---	0x38	4	Invalid Crypto Algorithm Error. A reserved cipher is selected, refer to PKTE_SA_CMD0.CIPHER. A reserved hash is selected, refer to PKTE_SA_CMD0.HASH.	The SA record is ignored, no packet is processed. The packet must be re-queued or discarded.
0b0100_1---	0x48	5	SPI Error. On an inbound packet, the 32-bit SPI value in the packet does not match the value in the SA while header processing is enabled. Note: A failure caused by an SPI mismatch, in general should not occur because the host checks the SPI and does not send an incorrect SPI to the packet engine.	Packet is fully processed. The host must reject the packet.
0b0101_1---	0x58	3	Zero Length Error. The packet length defined in the command descriptor PKTE_LEN.TOTLEN is zero, which is illegal.	Packet command is ignored, no packet processed. The host must reject the packet.
0b0110_1xxx	0x68	6	Invalid Packet Length Error. For Basic Encrypt-Hash and Hash-Decrypt operations: PKTE_LEN.TOTLEN < Hash/Encrypt Offset For IPsec ESP inbound operations:	Packet processing is aborted. Result packet length is zero.

Table 15-24: Extended Error Codes - Status Encoding (Continued)

STATUS bits [23:16]	Hex Value	Priority	Description	Processing Result
			PKTE_LEN.TOTLEN < ICV length or PKTE_LEN.TOTLEN is non-4 byte aligned For SRTP inbound operations: PKTE_LEN.TOTLEN ≤ IV (opt.) + Bypass Offset + ROC For SSL inbound operations: PKTE_LEN.TOTLEN ≤ 1 or packet length > 65535 bytes (SSL packet-bypass length) For TLS and DTLS inbound operations: PKTE_LEN.TOTLEN ≤ 13 or payload length > 65535 bytes (data to be hashed) Note: For IPsec ESP the ICV is stripped before the length is checked.	The host must reject the packet and apply a software reset of the packet engine.
0b0111_1xxx	0x78	7	Block Size Error. The length of the inbound packet defined in the Command Descriptor PKTE_LEN.TOTLEN is not a multiple of the DES or AES block cipher length. For outbound packets the size is always automatically aligned (padded) to the correct block size. The hashed packet length is not a multiple of the hash block size for intermediate hash operation. For a final hash operation no error is generated. Note: For IPsec ESP operations the ICV is stripped before the block size is checked.	Packet is fully processed. The host must reject the packet.
0b1000_1xxx	0x88	8	Processing Error. The number of bytes in the input buffer is more than defined in the PKTE_LEN.TOTLEN field. The number of bytes written to the output buffer is less than processed in the datapath.	Packet processing aborted. Result packet length is zero. The host must reject the packet and apply a software reset.
0b1010_1xxx 0b1111_1xxx	Reserved			

Number Format

When dealing with cryptographic functions, data and keys are large vectors. For instance, AES supports keys of sizes 128, 192, and 256 bits. When a key needs to be loaded or read, multiple 32-bit key registers are used, namely PKTE_SA_KEY[n] registers. The first key register PKTE_SA_KEY[n]0 holds bits 31:0, and PKTE_SA_KEY[n]1 holds the next thirty two bits 63:32, and so on.

Generally, for large vectors defined as Byte0, Byte1, Byte2, Byte3 and so on, the values are stored in the PKTE registers as PKTE_REG0 = Byte3Byte2Byte1Byte0, followed by PKTE_REG1 = Byte7Byte6Byte5Byte4 and so on.

PKTE Programming Examples

Use these examples to extend your understanding of PKTE features, operating modes, event control, and programming modes.

Calculating SHA in Direct Host Mode

This section describes how to configure the packet engine to calculate a hash digest using one of supported SHA algorithms in direct host mode. This configuration follows the procedure outlined in the *PKTE Programming Model* section.

1. Configure the packet engine for direct host mode by setting the `PKTE_CFG.MODE` bit =0
2. Set the ownership back to the packet engine to process a command descriptor by setting the `PKTE_CTL_STAT.PERDY` bit =0 and the `PKTE_CTL_STAT.HOSTRDY` bit =1.
3. Set the `PKTE_CTL_STAT.HASHFINAL` bit to indicate this command descriptor handles all the input data for the hash calculation. This configuration is needed for the packet engine because the last block requires special handling (see FIPS 180-4 for details).
4. Set size of the input data in bytes in the `PKTE_LEN.TOTLEN` bit field.
5. Also set the `PKTE_LEN.PEDONE` bit =0 and the `PKTE_LEN.HSTRDY` bit =1. These bits must be the same as the `PKTE_CTL_STAT.PERDY` and `PKTE_CTL_STAT.HOSTRDY` bits to guarantee ownership.
6. Set the `PKTE_CDSC_CNT` register =1 to trigger the packet engine to start validating the command descriptor. In this case, the `PKTE_CTL_STAT`, `PKTE_LEN` and `PKTE_CDSC_CNT` registers are the only command descriptor registers modified.
7. Configure the `PKTE_SA_CMD0` and `PKTE_SA_CMD1` registers to define the operation. For an SHA, set the `PKTE_SA_CMD0.OPCD` bit field =0b011 for hash operation and the `PKTE_SA_CMD0.OPGRP` bit field =0b00 for basic operation.
8. Select the specific SHA function using the `PKTE_SA_CMD0.HASH` bit field as follows.
    - For SHA-1, `PKTE_SA_CMD0.HASH` =0b0001
    - SHA-224, `PKTE_SA_CMD0.HASH` =0b0010
    - for SHA-256, `PKTE_SA_CMD0.HASH` =0b0011
9. Depending on the SHA selected, the appropriate digest length must be chosen for the `PKTE_SA_CMD0.DIGESTLEN` bit field as follows.
    - For SHA-1, `PKTE_SA_CMD0.DIGESTLEN` =0b0101 (5 words)
    - For SHA-224, `PKTE_SA_CMD0.DIGESTLEN` =0b0111 (7 words)
    - For SHA-256, `PKTE_SA_CMD0.DIGESTLEN` =0b1000 (8 words)

10. The SHA specifies initial constants. These constants can be pre-loaded or read from memory. In this example, by setting the PKTE_SA_CMD0.HASHSRC bit field =0b11, the packet engine provides the correct initial constants depending on the SHA chosen.
11. Next, set the PKTE_SA_CMD1.CPYDGST bit =1 and PKTE_SA_CMD1.CPYPAD bit =1 to move the result to the output buffer of the packet engine at the PKTE_DATAIO_BUF location.
12. At this point, write to the PKTE_SA_RDY register with any value to trigger the operation.
13. Start writing the input to the data buffer of the packet engine starting at the PKTE_DATAIO_BUF location.
14. Write the PKTE_INBUF_CNT register with the length of the input rounded up to the next multiple of 4. For example, if the input length is 30 bytes, set this register to 32.
15. Poll the PKTE_STAT register to see if any errors occurred or if the operation completed without errors.

Once the operation is done, the digest is available in the packet engine data I/O buffer.

NOTE: The input data or message is input into the packet engine data buffer in big endian format while the result or digest is little endian format.

Performing AES Decryption in Direct Host Mode

This section describes how to configure the packet engine to decrypt using AES-128 in direct host mode. This configuration follows the procedure outlined in the *PKTE Programming Model* section.

1. Configure the packet engine for direct host mode by setting the PKTE_CFG.MODE bit =0
2. Start configuring the command descriptor registers. Set the ownership back to the packet engine to process a command descriptor by setting the PKTE_CTL_STAT.PERDY bit =0 and the PKTE_CTL_STAT.HOSTRDY bit =1.
3. Next, configure the PKTE_LEN.TOTLEN bit field with the size of the packet or message to decrypt. If the entire input message (cipher text) fits into the 256-byte data I/O buffer of the packet engine, the process can be done in one shot.
4. Set the PKTE_LEN.PEDONE bit =0 and the PKTE_LEN.HSTRDY bit =1. These bits must have the same setting as the PKTE_CTL_STAT.PERDY and PKTE_CTL_STAT.HOSTRDY bits to guarantee ownership.
5. Set the PKTE_CDSC_CNT register =1 to trigger the packet engine to start validating the command descriptor. In this case, the PKTE_CTL_STAT, PKTE_LEN, and PKTE_CDSC_CNT registers are the only command descriptor registers modified.
6. Next, configure the PKTE_SA_CMD0 and PKTE_SA_CMD1 registers to define the operation.
    - For a AES decrypt inbound cipher operation, set the PKTE_SA_CMD0.OPCD bit field =0b000 and the PKTE_SA_CMD0.DIR bit field =0b1.
    - Set thePKTE_SA_CMD0.OPGRP bit field =0b00 for basic operation.

- To choose the AES cipher, set the PKTE_SA_CMD0.CIPHER bit field =0b0011. Set the PKTE_SA_CMD0.HASH bit field to 0b1111 to choose the NULL function.

7. Next, set the PKTE_SA_CMD1.AESKEYLEN bit field to select the appropriate key length. In this case, setting it to 0b10 select 128 bits. Also, set PKTE_SA_CMD1.CIPHERMD bit field to select the mode. In this case, setting it to 0b01 select CBC mode.
8. Continue configuring the Security Association (SA) record by loading the key in the PKTE_SA_KEY[n] registers.
9. Next load the initialization vector in the SA state registers (PKTE_STATE_IV[n]).
10. Finally, write anything in to the PKTE_SA_RDY register to trigger the operation.
11. The input data can now be written into the data I/O buffer starting at PKTE_DATAIO_BUF.
12. After the data is written, write the length (or next multiple of 4) into PKTE_INBUF_CNT register.
13. Poll PKTE_STAT to see if any errors occurred or if the operation completed without errors.

Once the operation is done, the result can be found in the same data I/O buffer.

ADSP-BF70x PKTE Register Descriptions

Security Packet Engine (PKTE) contains the following registers.

Table 15-25: ADSP-BF70x PKTE Register List

Name	Description
PKTE_BUF_PTR	Packet Engine Buffer Pointer Register
PKTE_BUF_THRESH	Packet Engine Buffer Threshold Register
PKTE_CDRBASE_ADDR	Packet Engine Command Descriptor Ring Base Address
PKTE_CDSC_CNT	Packet Engine Command Descriptor Count Register
PKTE_CDSC_INCR	Packet Engine Command Descriptor Count Increment Register
PKTE_CFG	Packet Engine Configuration Register
PKTE_CLK_CTL	PE Clock Control Register
PKTE_CONT	PKTE Continue Register
PKTE_CTL_STAT	Packet Engine Control Register
PKTE_DATAIO_BUF	Starting Entry of 256-byte Data Input/Output Buffer
PKTE_DEST_ADDR	Packet Engine Destination Address
PKTE_DMA_CFG	Packet Engine DMA Configuration Register
PKTE_ENDIAN_CFG	Packet Engine Endian Configuration Register
PKTE_HLT_CTL	Packet Engine Halt Control Register

Table 15-25: ADSP-BF70x PKTE Register List (Continued)

Name	Description
PKTE_HLT_STAT	Packet Engine Halt Status Register
PKTE_IMSK_DIS	Interrupt Mask Disable Register
PKTE_IMSK_EN	Interrupt Mask Enable Register
PKTE_IMSK_STAT	Interrupt Masked Status Register
PKTE_INBUF_CNT	Packet Engine Input Buffer Count Register
PKTE_INBUF_INCR	Packet Engine Input Buffer Count Increment Register
PKTE_INT_CFG	Interrupt Configuration Register
PKTE_INT_CLR	Interrupt Clear Register
PKTE_INT_EN	Interrupt Enable Register
PKTE_IUMSK_STAT	Interrupt Unmasked Status Register
PKTE_LEN	Packet Engine Length Register
PKTE_OUTBUF_CNT	Packet Engine Output Buffer Count Register
PKTE_OUTBUF_DECR	Packet Engine Output Buffer Count Decrement Register
PKTE_RDRBASE_ADDR	Packet Engine Result Descriptor Ring Base Address
PKTE_RDSC_CNT	Packet Engine Result Descriptor Count Registers
PKTE_RDSC_DECR	Packet Engine Result Descriptor Count Decrement Registers
PKTE_RING_CFG	Packet Engine Ring Configuration
PKTE_RING_PTR	Packet Engine Ring Pointer Status
PKTE_RING_STAT	Packet Engine Ring Status
PKTE_RING_THRESH	Packet Engine Ring Threshold Registers
PKTE_SA_ADDR	Packet Engine SA Address
PKTE_SA_CMD0	SA Command 0
PKTE_SA_CMD1	SA Command 1
PKTE_SA_IDIGEST[n]	SA Inner Hash Digest Registers
PKTE_SA_KEY[n]	SA Key Registers
PKTE_SA_NONCE	SA Initialization Vector Register
PKTE_SA_ODIGEST[n]	SA Outer Hash Digest Registers
PKTE_SA_RDY	SA Ready Indicator
PKTE_SA_SEQNUM[n]	SA Sequence Number Register
PKTE_SA_SEQNUM_MSK[n]	SA Sequence Number Mask Registers
PKTE_SA_SPI	SA SPI Register

Table 15-25: ADSP-BF70x PKTE Register List (Continued)

Name	Description
PKTE_SRC_ADDR	Packet Engine Source Address
PKTE_STAT	Packet Engine Status Register
PKTE_STATE_ADDR	Packet Engine State Record Address
PKTE_STATE_BYTE_CNT[n]	State Hash Byte Count Registers
PKTE_STATE_IDIGEST[n]	State Inner Digest Registers
PKTE_STATE_IV[n]	State Initialization Vector Registers
PKTE_USERID	Packet Engine User ID

Packet Engine Buffer Pointer Register

The PKTE_BUF_PTR register contains the offset of the next buffer address (entry) to be read or written by the packet engine. This register is used in direct host mode only.

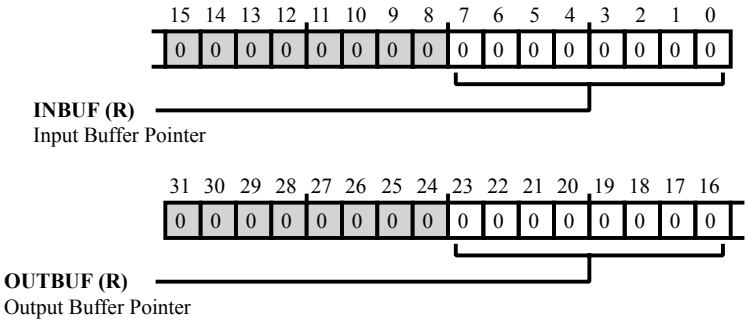

Figure 15-4: PKTE_BUF_PTR Register Diagram

Table 15-26: PKTE_BUF_PTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/NW)	OUTBUF	Output Buffer Pointer. The PKTE_BUF_PTR.OUTBUF bit field indicates the offset of the next address (entry) in the output buffer that will be written next by the packet engine. This bit field is reset to zero after starting up and decremented by 4 at every output buffer write operation. Pointers wrap around; the maximum value this field can have equals the output buffer size minus 4.
7:0 (R/NW)	INBUF	Input Buffer Pointer. The PKTE_BUF_PTR.INBUF bit field indicates the offset of the next address (entry) in the input buffer that will be read next by the packet engine. The bit field is reset to zero after starting up and incremented by 4 at every input buffer read operation. Pointers wrap around; the maximum value this field can have equals the input buffer size minus 4.

Packet Engine Buffer Threshold Register

When in autonomous ring mode or target command mode, the PKTE_BUF_THRESH register defines the high- and low-level value at which the packet engine starts to transfer packet data in or out of the internal packet buffers. These parameters can be used to control the DMA burst size for packet data input and output from the packet engine. In direct host mode, this register contains both threshold values to reduce the amount of packet engine interrupts.

The input buffer threshold (ibufthrsh) interrupt indicates that the input buffer counter is less than or equal to the input buffer threshold value set in this register - this interrupt can be used to wake up a process that stalled on a full input buffer.

The output buffer threshold (obufthrsh) interrupt indicates that the output buffer counter exceeds the output buffer threshold setting. The output buffer interrupt remains active until the output buffer counter is decremented to zero again.

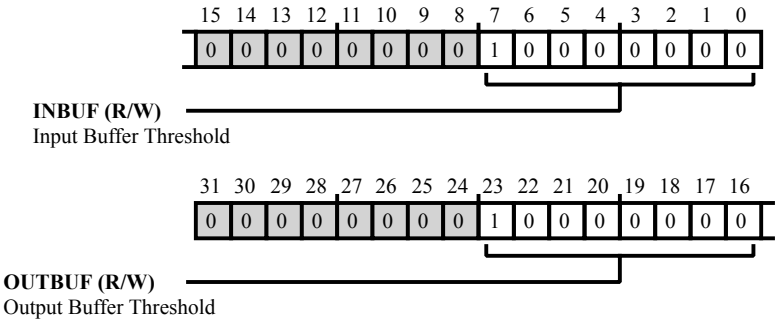

Figure 15-5: PKTE_BUF_THRESH Register Diagram

Table 15-27: PKTE_BUF_THRESH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	OUTBUF	Output Buffer Threshold. The PKTE_BUF_THRESH.OUTBUF bit field specifies how many bytes must be available in the packet engine output buffer before an output transfer starts. Valid values range from 0 to 252, in multiples of 4. In autonomous ring mode, a value of 128 generally gives a good performance, but the optimal value depends on the system and application. In direct host mode, the output buffer threshold (obufthrsh) interrupt activates when the output buffer counter for the output buffer exceeds the value set in this field. A value of 128 generally gives a good performance, but the optimal value depends on the system and application.
7:0 (R/W)	INBUF	Input Buffer Threshold.

Table 15-27: PKTE_BUF_THRESH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
		The PKTE_BUF_THRESH.INBUF bit field specifies how many bytes must be free in the packet engine input buffer before an input transfer starts. Valid values range from 0 to 252, in multiples of 4. In autonomous ring mode, a value of 128 generally gives a good performance, but the optimal value depends on the system and application. In direct host mode, the input buffer threshold (ibufthrsh) interrupt activates when the input buffer counter for the input buffer is below or equal the value set in this field. A value of 128 generally gives a good performance, but the optimal value depends on the system and application.

Packet Engine Command Descriptor Ring Base Address

The PKTE_CDRBASE_ADDR register holds the command descriptor ring base address in host memory. It is only applicable in autonomous ring mode. The PKTE_CDRBASE_ADDR register is ignored for all other modes when command descriptors are directly written into the descriptor registers.

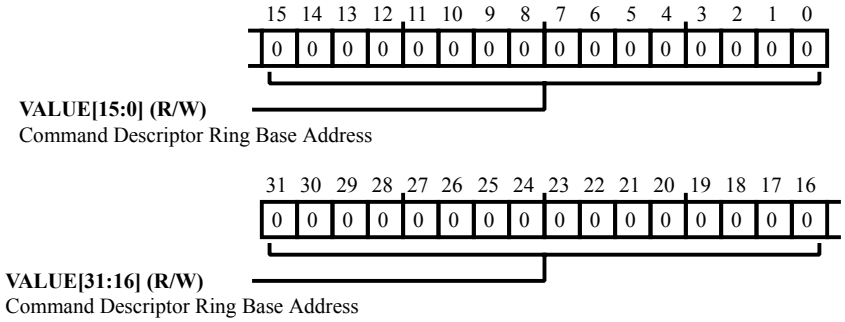

Figure 15-6: PKTE_CDRBASE_ADDR Register Diagram

Table 15-28: PKTE_CDRBASE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command Descriptor Ring Base Address. The PKTE_CDRBASE_ADDR.VALUE bit field specifies the base location of the command descriptor ring in the host memory space.

Packet Engine Command Descriptor Count Register

The PKTE_CDSC_CNT register holds the counter for the number of descriptors in the Command Descriptor Ring (CDR). It is decremented by the packet engine each time a valid descriptor is read from the CDR and processed.

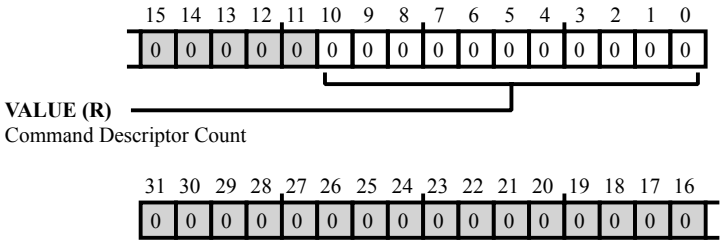

Figure 15-7: PKTE_CDSC_CNT Register Diagram

Table 15-29: PKTE_CDSC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	VALUE	Command Descriptor Count. The PKTE_CDSC_CNT.VALUE bit field provides the number of command descriptors in the command descriptor ring. The packet engine decrements the counter when a valid command descriptor is read from the CDR and processed.

Packet Engine Command Descriptor Count Increment Register

The PKTE_CDSC_INCR register is accessible by the host connected through the system slave bus. The host can increment the command descriptor counter by writing a value between 1 and 255 to the lowest byte of this register.

In autonomous ring mode, the host must prepare 1 to 255 valid command descriptors in the CDR and then write this register with a value between 1 and 255. The write triggers the packet engine to fetch the command descriptors from the CDR. In direct host mode or target command mode, the host must write one valid command descriptor to the internal descriptor registers and then write this register with the value 1, to indicate that one valid descriptor is available.

A CDR threshold interrupt is activated when the command descriptor counter is less than or equal to the threshold value set in the PKTE_RING_THRESH register. This interrupt can be used to wake up a process that stalled on a full CDR.

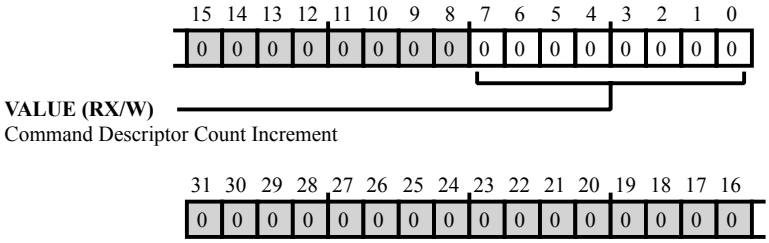

Figure 15-8: PKTE_CDSC_INCR Register Diagram

Table 15-30: PKTE_CDSC_INCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (RX/W)	VALUE	Command Descriptor Count Increment. The value written to the PKTE_CDSC_INCR.VALUE bit field is added to the command descriptor counter. The counter is protected against overflow (see the PKTE_RING_STAT register description). Note that bits[10:8] should be written with zeros.

Packet Engine Configuration Register

The PKTE_CFG register is used to select static settings that control the packet-processing path. This register is typically the last one to be written during the initialization sequence. These settings are typically set at initialization and not changed again.

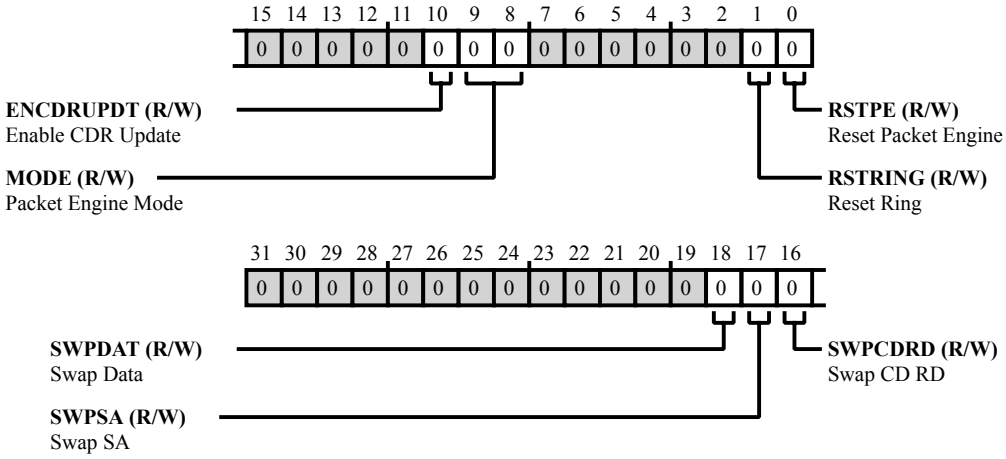

Figure 15-9: PKTE_CFG Register Diagram

Table 15-31: PKTE_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
18 (R/W)	SWPDAT	Swap Data. The PKTE_CFG.SWPDAT bit enables endian swap for packet data as configured in the PKTE_ENDIAN_CFG.MSTRBSWP bits for the packet data DMA read and write.	
		0	No Endian Swap
		1	Apply Endian Swap
17 (R/W)	SWPSA	Swap SA. The PKTE_CFG.SWPSA bit enables endian swap for a SA record as configured in the PKTE_ENDIAN_CFG.MSTRBSWP bits for the SA record and state record DMA read and write. If the PKTE_ENDIAN_CFG.MSTRBSWP bits specify no endian swap, this bit is ignored.	
		0	No Endian Swap
		1	Apply Endian Swap

Table 15-31: PKTE_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
16 (R/W)	SWPCDRD	Swap CD RD. The PKTE_CFG.SWPCDRD bit enables endian swap for descriptors as configured in the PKTE_ENDIAN_CFG.MSTRBSWP bits for the command descriptor DMA read and result descriptor DMA write. If the PKTE_ENDIAN_CFG.MSTRBSWP bits specify no endian swap, this bit is ignored.	
		0	No Endian Swap
		1	Apply Endian Swap
10 (R/W)	ENCDRUPDT	Enable CDR Update. The PKTE_CFG.ENCDRUPDT bit enables the packet engine to update, (clear the ownership bits) in the command descriptor in the CDR.	
		0	Do Not Clear Ownership Bits. The packet engine does not clear the ownership bits in the command descriptor when it completes an operation. The host application must clear the ownership bits in "old descriptors" before the packet engine is allowed to wrap around the CDR to re-encounter these "old descriptors". This setting has the advantage of eliminating a separate DMA write to the CDR.
		1	Clear Ownership Bits. The packet engine clears (set to zero) the ownership bits in the current command descriptor in the CDR. This prevents the packet engine from re-processing an "old descriptor" when it wraps around the CDR.
9:8 (R/W)	MODE	Packet Engine Mode. The PKTE_CFG.MODE bit field selects how the packet engine receives commands.	
		0	Direct Host Mode.
		1	Target Command Mode with Result Descriptor Ring Disabled.
		2	Target Command Mode with Result Descriptor Ring Enabled.
		3	Autonomous Ring Mode

Table 15-31: PKTE_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	RSTRING	Reset Ring. The PKTE_CFG.RSTRING bit resets the internal counters for the CDR and RDR, PKTE_CDSC_CNT and PKTE_RDSC_CNT registers) to zero. Resets the PKTE_RING_PTR register to the base address. After the reset the rings are empty. This bit must be written with a '1' to reset the descriptor ring manager and then re-written with a '0' to release the reset. Note that this bit can remain in the reset state if the CDR ring is disabled (PKTE_CFG.MODE is not 0b11). Note that this reset must be coordinated with the 'owner' of the descriptor ring to ensure that the pointers are in sync after the reset.	
		0	Release the Descriptor Ring Manager Reset
		1	Reset the Descriptor Ring Manager
0 (R/W)	RSTPE	Reset Packet Engine. The PKTE_CFG.RSTPE bit resets the packet engine and the state machine logic that drives header processing, DMA, and context management. The PKTE_CFG.RSTPE bit resets the PKTE_CTL_STAT and PKTE_LEN internal registers. This bit must be written with a 1 to reset the packet engine and then re-written with a 0 to release the reset. Note that this bit should not be used by a typical application. It is provided to use during development testing or to recover from critical errors. Note that the PKTE_CTL_STAT.PADVAL and PKTE_CTL_STAT.PADCTLSTAT bit fields are only reset when in autonomous ring mode, but the PKTE_CTL_STAT.PRNGMD bit is not reset. Halt mode is not affected by this reset as well. When exiting out of halt mode, a HW reset is required or a write to the PKTE_CONT register.	
		0	Release the Packet Engine Reset
		1	Reset the Packet Engine

PE Clock Control Register

The PKTE_CLK_CTL register controls the clock enable signals. This register can be used to enable the clock for read and write access to SA registers or to enable the required clock signals for certain crypto functions. The setting of this register overrides the packet engine dynamic clock enable.

In autonomous ring mode and target command modes, this register can be all zeros; the packet engine dynamically requests the external clock manager to activate the module clocks. This register can be used in combination with the debugging interface for internal register access.

In direct host mode, the clock enable bits for the packet engine (PKTE_CLK_CTL.ENPECLK) must be enabled to write and read the SA record and state record registers. All module clocks that are required for the current operation must be enabled during processing.

Note that all the clocks are enabled by default to reset all the registers within the packet engine. After a system reset the host can program this register to disable clocks for power reduction.

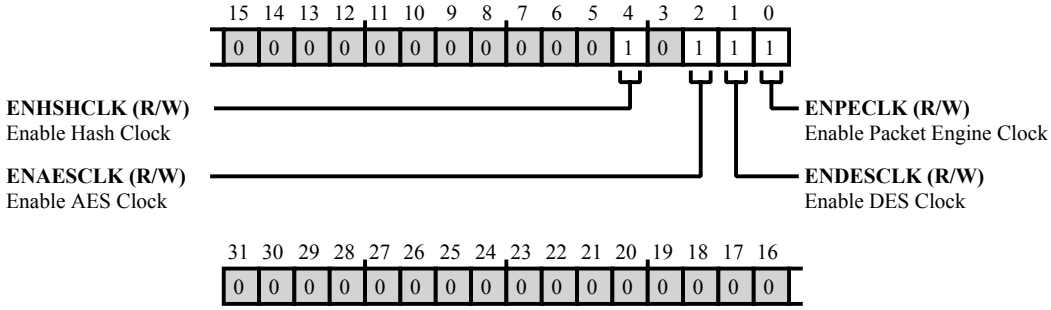

Figure 15-10: PKTE_CLK_CTL Register Diagram

Table 15-32: PKTE_CLK_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	ENHSHCLK	Enable Hash Clock. The PKTE_CLK_CTL.ENHSHCLK bit enables the clock to the hash functions.	
		0	Do not enable the hash clock
		1	Enable the hash clock
2 (R/W)	ENAESCLK	Enable AES Clock. The PKTE_CLK_CTL.ENAESCLK bit enables the clock to the AES encrypt/decrypt function.	
		0	Do not enable the AES clock
		1	Enable the AES clock

Table 15-32: PKTE_CLK_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	ENDESCLK	Enable DES Clock. The `PKTE_CLK_CTL.ENDESCLK` bit enables the clock to the DES function.	
		0	Do not enable the DES clock
		1	Enable the DES clock
0 (R/W)	ENPECLK	Enable Packet Engine Clock. The `PKTE_CLK_CTL.ENPECLK` bit enables the clock in the PKTE data path.	
		0	Do not enable the PKTE data path clock
		1	Enable the PKTE data path clock

PKTE Continue Register

A write to the PKTE_CONT register (with any value) releases the packet engine from a halt state when in halt mode.

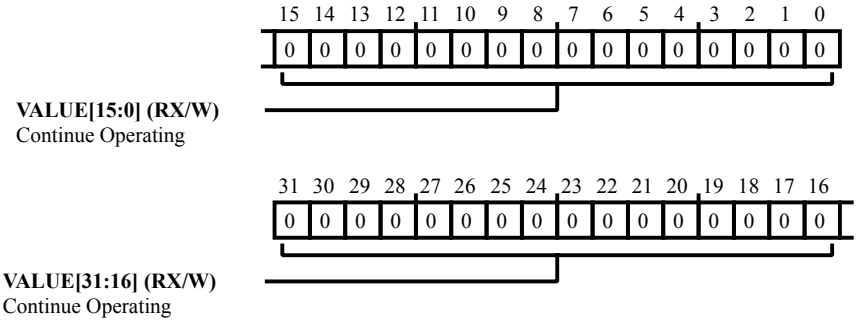

Figure 15-11: PKTE_CONT Register Diagram

Table 15-33: PKTE_CONT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Continue Operating. The PKTE_CONT.VALUE bit field releases the packet engine from a halt state when written with any value in halt mode.

Packet Engine Control Register

The PKTE_CTL_STAT register has a dual function. Together with the data in the SA, this register provides the basic command information for the packet engine to process a packet. When the packet engine successfully or unsuccessfully completes an operation, the packet engine control/status register provides the result status for the host.

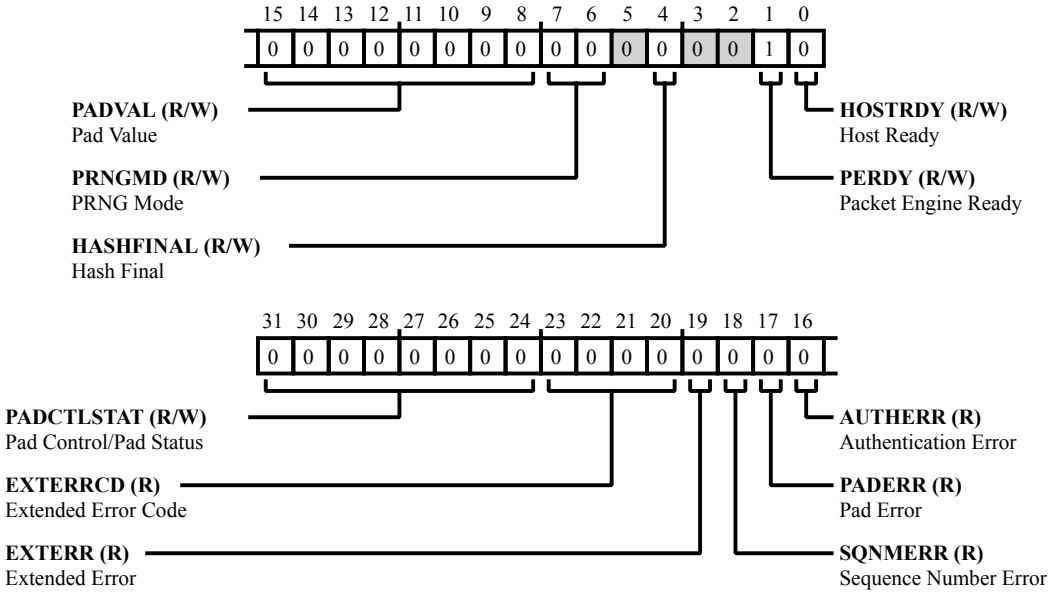

Figure 15-12: PKTE_CTL_STAT Register Diagram

Table 15-34: PKTE_CTL_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
31:24 (R/W)	PADCTLSTAT	Pad Control/Pad Status. The PKTE_CTL_STAT.PADCTLSTAT bit field is used to control the pad boundary for pad insertion (outbound) and after processing returns the number of inserted (outbound) or detected (inbound) pad bytes. For the command descriptor, the enumerations below provide the codes for the pad boundary for the outbound operations. This can be used for traffic flow security to conceal the number of payload bytes in an encrypted packet. For the result descriptor inbound operations that use pad types SSL, TLS, IPsec or PKCS#7, it returns the number of detected pad bytes. For all other inbound operations, it returns zero since the other pad modes do not allow implicit determination of pad count. If a pad verify failure occurs, it returns zero. For an outbound operation, it returns the number of inserted pad bytes for all pad types. The pad value includes added bytes such as the pad length and the next header field in an IPsec ESP pad type.	
		0	Align packet end to modulo 8-byte boundary
		1	Align packet end to modulo 1-byte boundary

Table 15-34: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		2	Align packet end to modulo 4-byte boundary
		4	Align packet end to modulo 8-byte boundary
		8	Align packet end to modulo 16-byte boundary
		16	Align packet end to modulo 32-byte boundary
		32	Align packet end to modulo 64-byte boundary
		64	Align packet end to modulo 128-byte boundary
		128	Align packet end to modulo 256-byte boundary
23:20 (R/NW)	EXTERRCD	Extended Error Code. The PKTE_CTL_STAT.EXTERRCD bit field represents an encoded error condition.	
19 (R/NW)	EXTERR	Extended Error. The PKTE_CTL_STAT.EXTERR bit field provides an extended error code.	
		0	No Extended Error
		1	Extended Error
18 (R/NW)	SQNMERR	Sequence Number Error. The PKTE_CTL_STAT.SQNMERR bit indicates that for an inbound operation, there was a fault in the anti-replay sequence number. For an outbound operation, there was a sequence number overflow condition.	
		0	No Sequence Number Error
		1	Sequence Number Error
17 (R/NW)	PADERR	Pad Error. The PKTE_CTL_STAT.PADERR bit indicates that for an inbound operation the decrypted pad does not match the expected values.	
		0	No Pad Error
		1	Pad Error
16 (R/NW)	AUTHERR	Authentication Error. The PKTE_CTL_STAT.AUTHERR bit indicates that for an inbound operation the authentication value in the packet does not match the computed value.	
		0	No Authentication Error
		1	Authentication Error
15:8 (R/W)	PADVAL	Pad Value. The PKTE_CTL_STAT.PADVAL bit field is used to pass the pad value between the host and the packet engine. Command Descriptor: (write-only)	

Table 15-34: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		For outbound operations that use pad type IPsec, the host must populate this field with the value that is to be inserted into the next header field. For the IPsec ESP operation, this next header is part of the ESP trailer of the innermost operation's header and the value must be 50 decimal. For outbound encrypt operations that use the pad type constant or constant SSL, the host must specify the fixed constant value in this field. For all other outbound and inbound operations, this field is not used. Result Descriptor: (read only) For inbound operations that use pad type IPsec, the packet engine returns the next header field that it detects. For IPsec ESP inbound operations, this is the next header field in the innermost operation's header, which will typically be the value for the payload protocol, such as TCP or UDP. However, in bundling scenarios or in IPv6 with destination option headers, another header value could be seen. For all other outbound operations, the packet engine will not update this field. For all other inbound operations, the returned pad value is zero.	
7:6 (R/W)	PRNGMD	PRNG Mode. The PKTE_CTL_STAT.PRNGMD bits select the pseudo-random number generator mode.	
		0	Operation does not use the PRNG function.
		1	PRNG Init. PRNG is initialized with a SEED, KEY and an LFSR value as defined in the SA.
		2	PRNG Generate. Pseudo-random data is generated with the LFSR as input value. Before this mode can be used, the PRNG must be initialized with a valid SEED, KEY and LFSR using PRNG Init (PKTE_CTL_STAT.PRNGMD=b'01).
		3	PRNG Test. It can be used to test the PRNG function with custom input data. Before this mode can be used, the PRNG must be initialized once with a valid SEED using PRNG Init (PKTE_CTL_STAT.PRNGMD=b'01).

Table 15-34: PKTE_CTL_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
4 (R/W)	HASHFINAL	Hash Final. When the PKTE_CTL_STAT.HASHFINAL bit is zero, the data to be hashed must be a multiple of the hash block size, 64 bytes for SHA-1, MD5, SHA-224, SHA-256. This bit is only applicable for Basic Hash, Basic Encrypt-Hash and Basic Hash-Decrypt operations that use the SHA-1, MD5, SHA-224, SHA-256 hash algorithm. The PKTE_CTL_STAT.HASHFINAL bit is overruled for HMAC operations that always completes the hash and always returns the last written value on a read by the host.	
		0	Perform Intermediate Hash Operation. The packet engine performs an intermediate hash operation by generating an intermediate hash digest on the data presented on the input. No hash pad is applied.
		1	Perform Final Hash Operation. The packet engine appends the required final hash pad and generates the final hash digest on the data presented on the input. This completes the hash operation.
1 (R/W)	PERDY	Packet Engine Ready. The PKTE_CTL_STAT.PERDY bit indicates that the packet engine has completed processing the command descriptor and returns the result descriptor with ownership set to the host. This bit can be reset to 0 by the host and the packet engine, but only the packet engine can set this bit. When the packet engine is idle (not processing), this bit always returns '1' on a read by the host.	
0 (R/W)	HOSTRDY	Host Ready. The PKTE_CTL_STAT.HOSTRDY bit indicates that the host has populated the command descriptor. This bit can be reset to 0 by the host and the packet engine, but only the host can set this bit. When the packet engine is idle (not processing), this bit always returns '0' on a read by the host.	

Starting Entry of 256-byte Data Input/Output Buffer

When in direct host mode, the source packet data is written here to be transferred to the packet engine. The host can monitor the available space in the input buffer through the PKTE_STAT register. This is also the location in the packet engine from where output data is read when in direct host mode. The host can monitor the available bytes in the output buffer through the PKTE_STAT register.

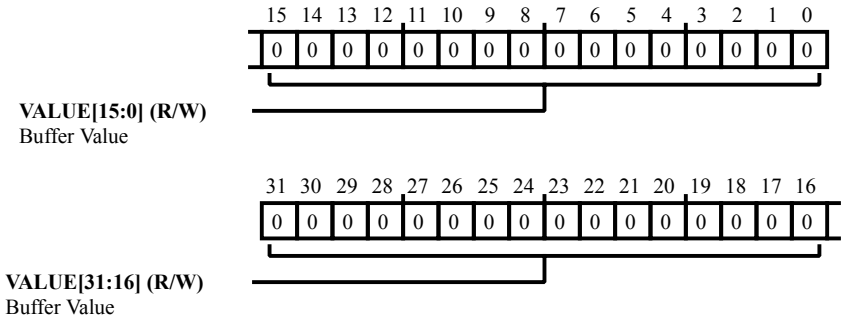

Figure 15-13: PKTE_DATAIO_BUF Register Diagram

Table 15-35: PKTE_DATAIO_BUF Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Buffer Value.

Packet Engine Destination Address

The PKTE_DEST_ADDR register holds the starting (byte) address to write the result packet from the requested operation.

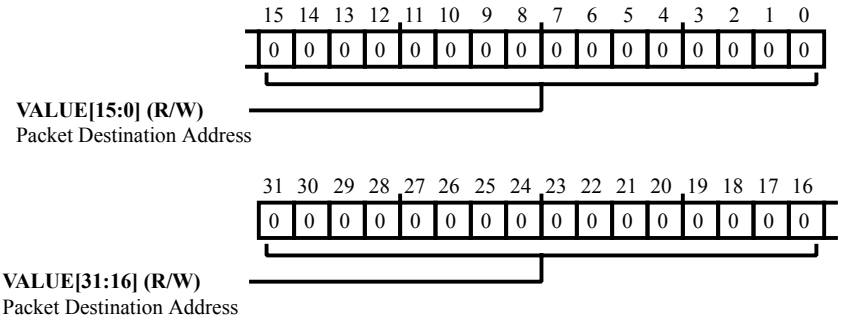

Figure 15-14: PKTE_DEST_ADDR Register Diagram

Table 15-36: PKTE_DEST_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Packet Destination Address.

Packet Engine DMA Configuration Register

The PKTE_DMA_CFG register configures the maximum burst transfer size, enables incremental transfers, and insertions IDLE cycles between two bus transfers.

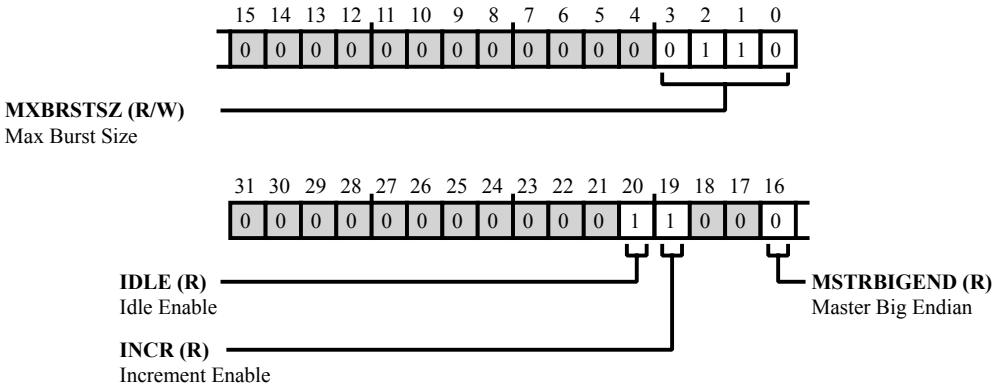

Figure 15-15: PKTE_DMA_CFG Register Diagram

Table 15-37: PKTE_DMA_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
20 (R/NW)	IDLE	Idle Enable. The PKTE_DMA_CFG.IDLE bit allows the peripheral bus master to insert one additional IDLE transfer between two successive peripheral bus master burst operations. This provides the arbiter one additional cycle to hand over the grant to another peripheral bus master.	
		0	The peripheral bus master inserts no IDLE cycle between two successive burst operations
		1	The peripheral bus master inserts one additional IDLE transfer between two successive burst operations

Table 15-37: PKTE_DMA_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
19 (R/NW)	INCR	Increment Enable. The PKTE_DMA_CFG.INCR bit lets the peripheral bus master generate INC4, INC8 and INC16 type of burst transfers. By default, the peripheral bus master generates the largest possible incremental burst of unspecified length (INCR) with a maximum length (in bytes) as configured by the PKTE_DMA_CFG.MXBRSTSZ bit field. In case there are less than 4 bytes of data available or the 1kB boundary will be crossed using a burst operation, then a single transfer of size byte is generated. When the PKTE_DMA_CFG.INCR bit is set, the peripheral bus master generates one or more incremental burst of specified length (INC4, INC8, INC16). In case there is less data available then the smallest possible burst (INC4) or the 1kB boundary will be crossed using a burst operation, then an unspecified length burst or a single transfer of size byte is generated.	
		0	The bus master will generate only INCR burst types
		1	The bus master will generate INC4, INC8 and INC16 burst types
16 (R/NW)	MSTRBIGEND	Master Big Endian. The PKTE_DMA_CFG.MSTRBIGEND bit determines whether the engine is used in a little or big endian system.	
		0	Little endian
		1	Big endian
3:0 (R/W)	MXBRSTSZ	Max Burst Size. The PKTE_DMA_CFG.MXBRSTSZ bit field configures the maximum size of an unspecified length burst (INC) at the bus in bytes. When there is less data available than the PKTE_DMA_CFG.MXBRSTSZ bit field setting or the 1kB boundary will be crossed using a burst operation, then the length of the burst can be less than PKTE_DMA_CFG.MXBRSTSZ. Any requested transfers larger than this size are broken up in to multiple burst transfers of this size or less.	

Packet Engine Endian Configuration Register

The packet engine incorporates a powerful interface specific endian handler. This endian handler allows byte lane swapping in each direction for data passing through the host interface.

The PKTE_ENDIAN_CFG register configures the byte order function for the peripheral bus master and peripheral bus slave interface. The bits for the peripheral bus master are combined in four sets of two bits; each group configures a byte swap function for a particular DMA transfer. The same applies for the peripheral bus slave interface.

The PKTE_ENDIAN_CFG register also defines the endian swapping that occurs for host-initiated target transfers and for packet engine master DMA read and write transfers. Individual endian swap enable bits in the configuration (PKTE_CFG) register can enable the endian swap for various transaction types: command descriptors and result descriptors, SA records and state records, packet data.

In direct host mode, only target operations are supported. Only the target endian configuration of this register is applicable.

Note: This register is typically programmed once during the initialization phase, although software is allowed to dynamically change the setting in this register just before initiating a data transfer. The developer will have to analyze the benefit of the cycles needed to write the endian register dynamically versus handling endian swapping for some data structures in the host system (most modern processors support a byte swap in zero cycles). Certainly the endian swap should be set correctly for the packet data, since this represents the majority of the data transferred.

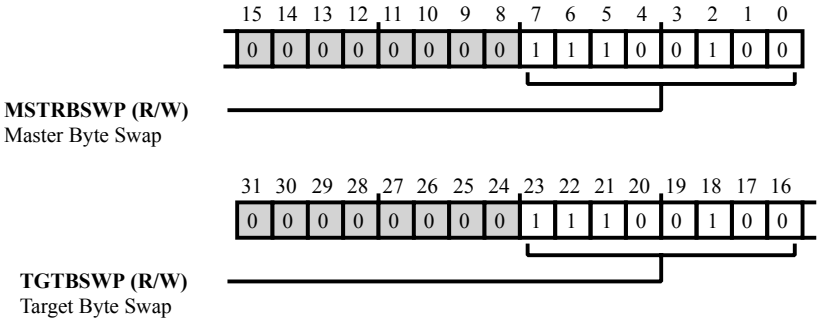

Figure 15-16: PKTE_ENDIAN_CFG Register Diagram

Table 15-38: PKTE_ENDIAN_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
23:16 (R/W)	TGTBSWP	Target Byte Swap. The PKTE_ENDIAN_CFG.TGTBSWP bit field configures the byte swap for peripheral bus target transfers. Note that only target word transfers are supported. Each double-bit field in this register specifies the source of the indicated byte lane. The field values are interpreted as follows: 00 = byte 0, 01 = byte 1, 10 = byte 2, 11 = byte 3. Note: Setting the value 0xE4 defines no swap (little endian) and setting value 0x1B defines a full byte swap within a 32-bit word (big endian).

Table 15-38: PKTE_ENDIAN_CFG Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (R/W)	MSTRBSWP	Master Byte Swap. The PKTE_ENDIAN_CFG.MSTRBSWP bit field configures the byte swap for peripheral bus master multi-byte transfers, including command descriptors, result descriptors, SA records, state records and packet data. Separate controls in the PKTE_CFG register can enable this swap individually for each of the 4 types of data. Each double-bit field in this register specifies the source of the indicated peripheral bus byte lane. The field values are interpreted as follows: 00=byte 0, 01=byte 1, 10=byte 2, 11=byte 3. Note: Setting the value 0xE4 defines no swap (little endian) and setting value 0x1B defines a full byte swap within a 32-bit word (big endian).

Packet Engine Halt Control Register

Th PKTE_HLT_CTL register controls the packet engine halt mode. This register can be used for debugging purposes while processing in autonomous ring mode or target command mode. During the halt mode, the host can read all internal registers for examination without side-effects. When halted, the host should not write to any registers. To continue packet engine operation, the host must write to the PKTE_CONT (PKTE continue) register.

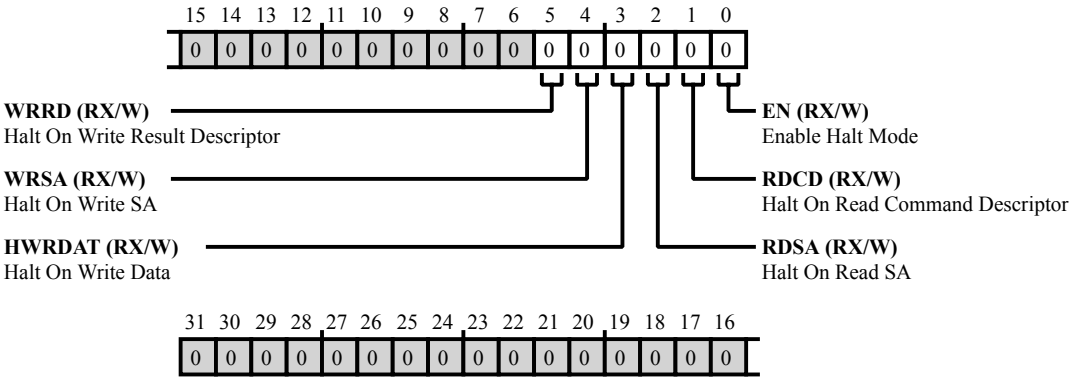

Figure 15-17: PKTE_HLT_CTL Register Diagram

Table 15-39: PKTE_HLT_CTL Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
5 (RX/W)	WRRD	Halt On Write Result Descriptor. The PKTE_HLT_CTL.WRRD bit halts the packet engine in the HALT_WRITE_STATUS state after it completes a result descriptor write operation to the result descriptor ring. The host can use this bit to examine the result descriptor that is currently in the host memory.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
4 (RX/W)	WRSA	Halt On Write SA. The PKTE_HLT_CTL.WRSA bit halts the packet engine in the HALT_WRITE_SA state after it completes an SA write operation to the host memory. The host can use this bit to examine the security context that is currently in the host memory.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation

Table 15-39: PKTE_HLT_CTL Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
3 (RX/W)	HWRDAT	Halt On Write Data. The PKTE_HLT_CTL.HWRDAT bit halts the packet engine in the HALT_DATA state after it completes writing the result packet data to the host memory. The host can use this bit to examine the result packet that is currently in the host memory.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
2 (RX/W)	RDSA	Halt On Read SA. The PKTE_HLT_CTL.RDSA bit halts the packet engine in the HALT_READ_SA state after it completes an SA read operation from the host memory. The host can use this bit to examine the security context that is currently in the SA registers.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
1 (RX/W)	RDCD	Halt On Read Command Descriptor. The PKTE_HLT_CTL.RDCD bit halts the packet engine in the HALT_READ_DESCR state after it completes a command descriptor read operation from the command descriptor ring. It will halt whether the descriptor is valid or invalid. The host can use this bit to examine the command descriptor that is currently in the internal command descriptor registers.	
		0	Do not halt the Packet Engine operation
		1	Halt the Packet Engine operation
0 (RX/W)	EN	Enable Halt Mode. The PKTE_HLT_CTL.EN bit enables halt mode where the packet engine can halt processing at any processing state as indicated by bits [5:1]. When halted, the packet engine continues operation on a write to the PKTE_CONT register.	
		0	Do not enable halt mode
		1	Enable halt mode

Packet Engine Halt Status Register

The PKTE_HLT_STAT register reflects the status of the packet engine in halt mode. This register can be used for debugging purposes while processing in autonomous ring mode or target command mode. When the packet engine is halted, the host can read all internal registers for examination without side effects. The host should not write to any registers.

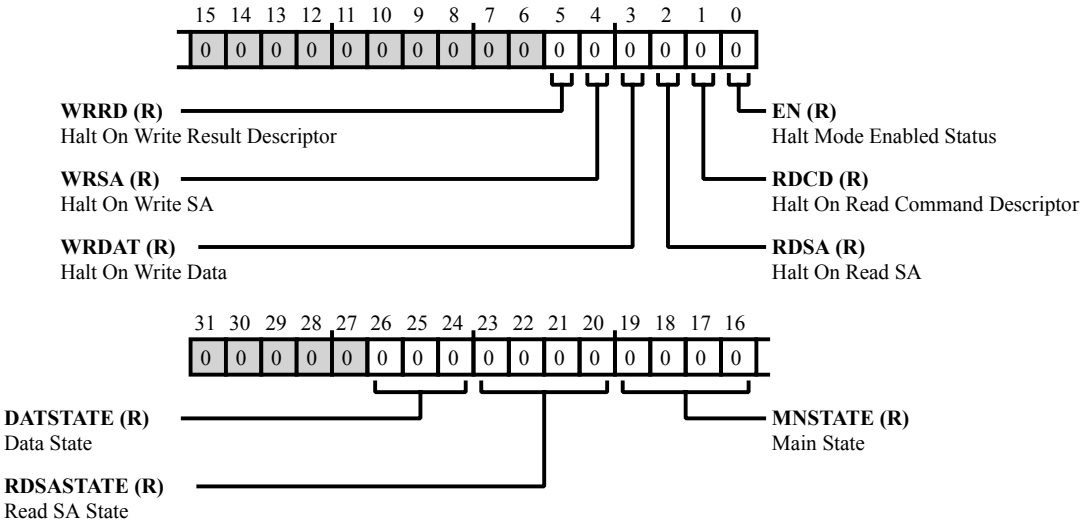

Figure 15-18: PKTE_HLT_STAT Register Diagram

Table 15-40: PKTE_HLT_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
26:24 (R/NW)	DATSTATE	Data State. The PKTE_HLT_STAT.DATSTATE bit field indicates the state of the packet engine read data FSM.	
		0	DATA_IDLE, no operation
		1	DATA_READ
		2	DATA_WRITE
		3	DATA_WAIT
		5	DATA_PAD_READ
		6	DATA_BYP_READ
		7	RESERVED
23:20 (R/NW)	RDSASTATE	Read SA State. The PKTE_HLT_STAT.RDSASTATE bit field indicates the state of the packet engine read SA FSM.	

Table 15-40: PKTE_HLT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		0	SA_IDLE, no operation
		1	SA_READ_CMD
		2	SA_READ_STATE_IV
		3-5	RESERVED
		6	RESERVED
		7	SA_READ_WAIT
		8	RESERVED
		9	SA_WRITE_PROT_HDR
		10	RESERVED
		11	SA_WRITE_IV
		12	SA_WRITE_DIGEST
		13	RESERVED
		14	RESERVED
		15	SA_WRITE_WAIT
19:16 (R/NW)	MNSTATE	Main State. The `PKTE_HLT_STAT.MNSTATE` bit field indicates the state of the packet engine main FSM.	
		0	MAIN_IDLE, no operation
		1	MAIN_READ_CD, reading command descriptor
		2	MAIN_READ_SA, reading SA
		3	MAIN_DATA, processing data
		4	MAIN_WRITE_SA, writing SA
		5	MAIN_WRITE_STATUS, writing status
		6	MAIN_WRITE_CD, updating command descriptor
		7	MAIN_WRITE_RD, updating result descriptor
		8	MAIN_INIT_WAIT, wait single clock
		9	MAIN_HALT_READ_CD, halt after read command descriptor
		10	MAIN_HALT_READ_SA, halt after read SA
		11	MAIN_HALT_DATA, halt after processing data
		12	MAIN_HALT_WRITE_SA, halt after write SA

Table 15-40: PKTE_HLT_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		13	MAIN_WAIT_FOR_CLOCK, wait for clocks to be active
		15	MAIN_HALT_WRITE_RD, halt after write result descriptor
5 (R/NW)	WRRD	Halt On Write Result Descriptor. The PKTE_HLT_STAT.WRRD bit reflects the value in the PKTE_HLT_CTL.WRRD bit.	
4 (R/NW)	WRSA	Halt On Write SA. The PKTE_HLT_STAT.WRSA bit reflects the value in the PKTE_HLT_CTL.WRSA bit.	
3 (R/NW)	WRDAT	Halt On Write Data. The PKTE_HLT_STAT.WRDAT bit reflects the value in the PKTE_HLT_CTL.HWRDAT bit.	
2 (R/NW)	RDSA	Halt On Read SA. The PKTE_HLT_STAT.RDSA bit reflects the value in the PKTE_HLT_CTL.RDSA bit.	
1 (R/NW)	RDCD	Halt On Read Command Descriptor. The PKTE_HLT_STAT.RDCD bit reflects the value in the PKTE_HLT_CTL.RDCD bit.	
0 (R/NW)	EN	Halt Mode Enabled Status.	
		0	Halt mode not enabled
		1	Halt mode enabled

Interrupt Mask Disable Register

The host can use the PKTE_IMSK_DIS register to clear individual bits in the PKTE_INT_EN register for the host interrupt. This register is a bitmap for each of the possible interrupt sources: A 1 clears the interrupt enable bit, a 0 does not affect the interrupt enable bit in the PKTE_INT_EN register. Clearing the enable bits through this register avoids the time-consuming read-modify-write operation on the host.

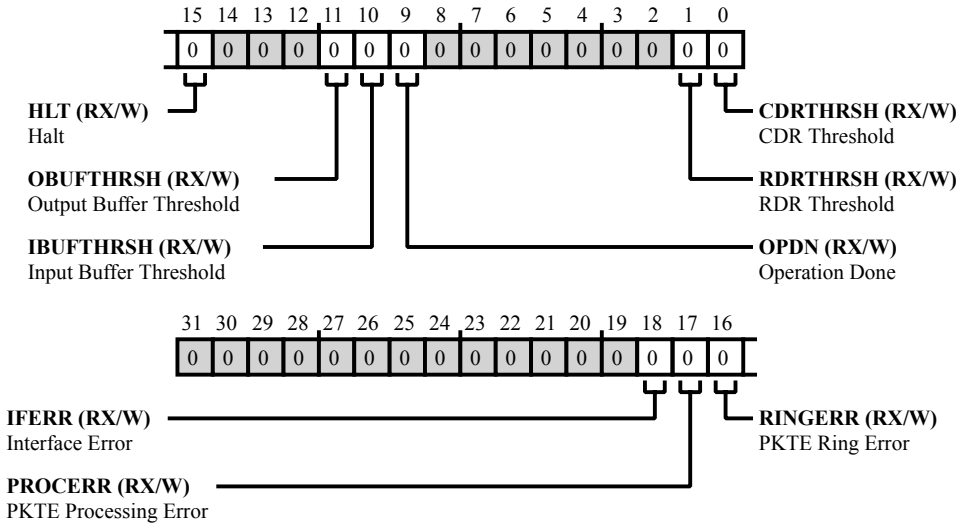

Figure 15-19: PKTE_IMSK_DIS Register Diagram

Table 15-41: PKTE_IMSK_DIS Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (RX/W)	IFERR	Interface Error. Write the PKTE_IMSK_DIS.IFERR bit to clear when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (RX/W)	PROCERR	PKTE Processing Error. Write the PKTE_IMSK_DIS.PROCERR bit to clear an extended error that occurred before, during or after processing the current packet in the packet engine.
16 (RX/W)	RINGERR	PKTE Ring Error. Write the PKTE_IMSK_DIS.RINGERR bit to clear a CDR overflow or an RDR underflow.
15 (RX/W)	HLT	Halt. Write the PKTE_IMSK_DIS.HLT bit to clear when the packet engine is in the halt state.
11	OBUFTHRSH	Output Buffer Threshold.

Table 15-41: PKTE_IMSK_DIS Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
(RX/W)		Write the PKTE_IMSK_DIS.OBUFTHRSH bit to clear the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.
10 (RX/W)	IBUFTHRSH	Input Buffer Threshold. Write the PKTE_IMSK_DIS.IBUFTHRSH bit to clear when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (RX/W)	OPDN	Operation Done.
1 (RX/W)	RDRTHRSH	RDR Threshold. Write the PKTE_IMSK_DIS.RDRTHRSH bit to clear when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in the PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (RX/W)	CDRTHRSH	CDR Threshold. Write the PKTE_IMSK_DIS.CDRTHRSH bit to clear when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Mask Enable Register

The host can use the PKTE_IMSK_EN register to set individual bits in the PKTE_INT_EN register for the host interrupt. This register is a bitmap for each of the possible interrupt sources: A 1 sets the interrupt enable bit, a 0 does not affect the interrupt enable bit in the PKTE_INT_EN register. Setting the enable bits through this register avoids the time-consuming read-modify-write operation on the host.

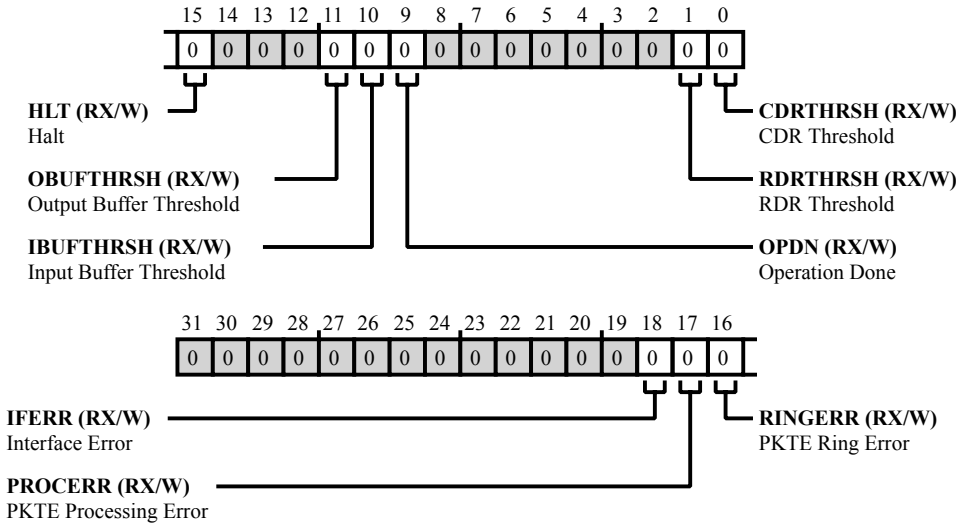

Figure 15-20: PKTE_IMSK_EN Register Diagram

Table 15-42: PKTE_IMSK_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (RX/W)	IFERR	Interface Error. Set the PKTE_IMSK_EN.IFERR bit to indicate a host request for a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (RX/W)	PROCERR	PKTE Processing Error. Set the PKTE_IMSK_EN.PROCERR bit to indicate an extended error occurred before, during or after processing the current packet in the packet engine.
16 (RX/W)	RINGERR	PKTE Ring Error.
15 (RX/W)	HLT	Halt. Set the PKTE_IMSK_EN.HLT bit to indicate when the packet engine is in the halt state.
11 (RX/W)	OBUFTHRSH	Output Buffer Threshold.

Table 15-42: PKTE_IMSK_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
		Set the PKTE_IMSK_EN.OBUFTHRSH bit to indicate that the output buffer counter exceeds the output buffer threshold value defined in the PKTE_BUF_THRESH.OUTBUF bit.
10 (RX/W)	IBUFTHRSH	Input Buffer Threshold. Set the PKTE_IMSK_EN.IBUFTHRSH bit to indicate the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (RX/W)	OPDN	Operation Done.
1 (RX/W)	RDRTHRSH	RDR Threshold. Set the PKTE_IMSK_EN.RDRTHRSH bit to indicate when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (RX/W)	CDRTHRSH	CDR Threshold. Set the PKTE_IMSK_EN.CDRTHRSH bit to indicate when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Masked Status Register

The PKTE_IMSK_STAT register provides interrupt status visibility to the host, after the interrupt mask is applied. This lets the host view the selected sources of interrupts that are directed to the interrupt output signal, that is connected to the system interrupt controller. As with the unmasked status register, all interrupt bits are latched and must be cleared using the PKTE_INT_CLR register in order to capture a subsequent event. A 1 indicates that the associated interrupt is present.

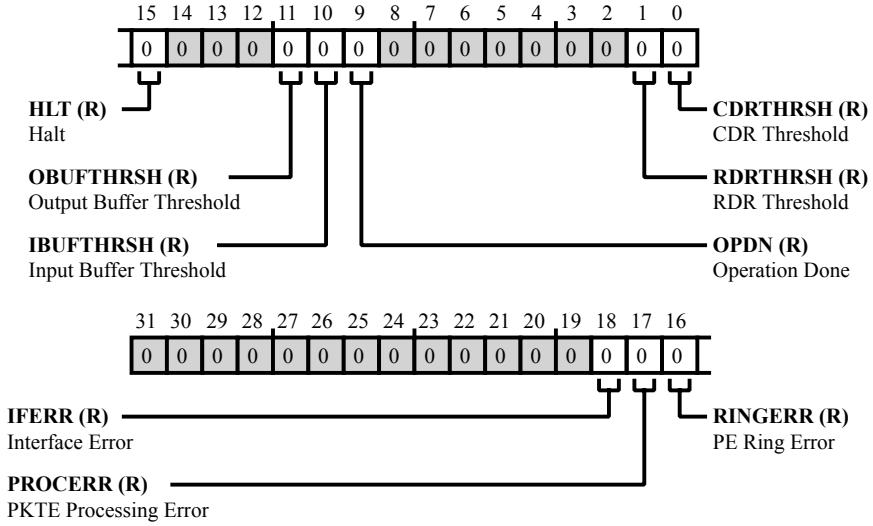

Figure 15-21: PKTE_IMSK_STAT Register Diagram

Table 15-43: PKTE_IMSK_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/NW)	IFERR	Interface Error. The PKTE_IMSK_STAT.IFERR bit is set when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (R/NW)	PROCERR	PKTE Processing Error. The PKTE_IMSK_STAT.PROCERR bit is when an extended error occurred before, during or after processing the current packet in the packet engine.
16 (R/NW)	RINGERR	PE Ring Error. The PKTE_IMSK_STAT.RINGERR bit is set on a CDR overflow or an RDR underflow.
15 (R/NW)	HLT	Halt. The PKTE_IMSK_STAT.HLT bit is set when the packet engine is in the HALT state.
11	OBUFTHRSH	Output Buffer Threshold.

Table 15-43: PKTE_IMSK_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
(R/NW)		The PKTE_IMSK_STAT.OBUFTHRSH bit is set when the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.
10 (R/NW)	IBUFTHRSH	Input Buffer Threshold. The PKTE_IMSK_STAT.IBUFTHRSH bit is set when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (R/NW)	OPDN	Operation Done.
1 (R/NW)	RDRTHRSH	RDR Threshold. The PKTE_IMSK_STAT.RDRTHRSH bit is set when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (R/NW)	CDRTHRSH	CDR Threshold. The PKTE_IMSK_STAT.CDRTHRSH bit is set when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Packet Engine Input Buffer Count Register

The PKTE_INBUF_CNT register provides the number of bytes available in the input buffer. The PKTE_INBUF_CNT register is used in direct host mode only.

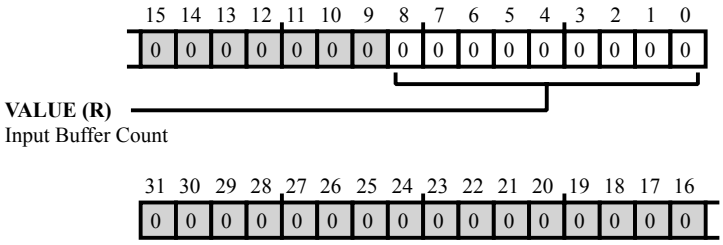

Figure 15-22: PKTE_INBUF_CNT Register Diagram

Table 15-44: PKTE_INBUF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/NW)	VALUE	Input Buffer Count. The PKTE_INBUF_CNT.VALUE bit field provides the number of bytes in the input buffer. The packet engine decrements the counter by 4 when a 32-bit word is read from the input buffer.

Packet Engine Input Buffer Count Increment Register

A host connected through the system slave bus can increment the input buffer counter by writing a value between 4 and 256, in multiples of 4, to the lowest bits of this register. The PKTE_INBUF_INCR register is used in direct host mode only.

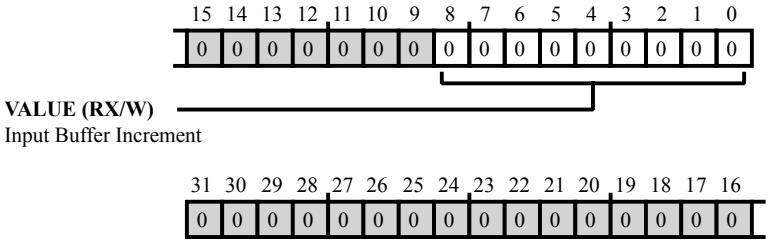

Figure 15-23: PKTE_INBUF_INCR Register Diagram

Table 15-45: PKTE_INBUF_INCR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (RX/W)	VALUE	Input Buffer Increment. The value written is added to the input buffer counter. Valid values range from 4 to 256, in multiples of 4.

Interrupt Configuration Register

The PKTE_INT_CFG register configures the interrupt type that is sent to the interrupt line connected to the system interrupt controller. (Note that this only effects the final output of the interrupt subsystem).

Configuring the interrupt output type for pulse causes the interrupt signal to pulse low for two clock cycles when activated. When set for level, the interrupt signal is set low until cleared by the host (it follows the bit in the masked status register). For the host, this is typically set to level.

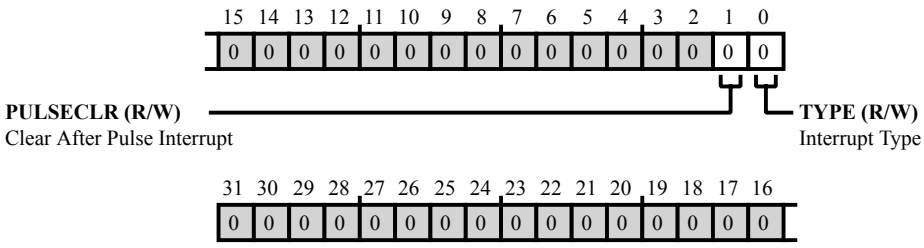

Figure 15-24: PKTE_INT_CFG Register Diagram

Table 15-46: PKTE_INT_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/W)	PULSECLR	Clear After Pulse Interrupt. The PKTE_INT_CFG.PULSECLR bit clears the latched interrupt source after the pulse interrupt.	
		0	Manually clear pulse interrupt source. Do not automatically clear the interrupt sources after pulsing the interrupt output. Clear the source by writing to the PKTE_INT_CLR register.
		1	Automatically clear pulse interrupt source. After pulsing the interrupt output, automatically clear the sources.
0 (R/W)	TYPE	Interrupt Type. The PKTE_INT_CFG.TYPE bit selects the type, pulse or level, for the interrupt output to the system.	
		0	Level. The interrupt output is a level signal that is set low when an enabled interrupt is active until the interrupt is cleared.
		1	Pulse. The interrupt output is a two clock cycle low-active pulse, activated when an enabled interrupt is active.

Interrupt Clear Register

The PKTE_INT_CLR register allows the host processor to clear pending interrupts. A 1 written to a given bit in this register clears the corresponding interrupt. A 0 leaves the interrupt latch unchanged for that position.

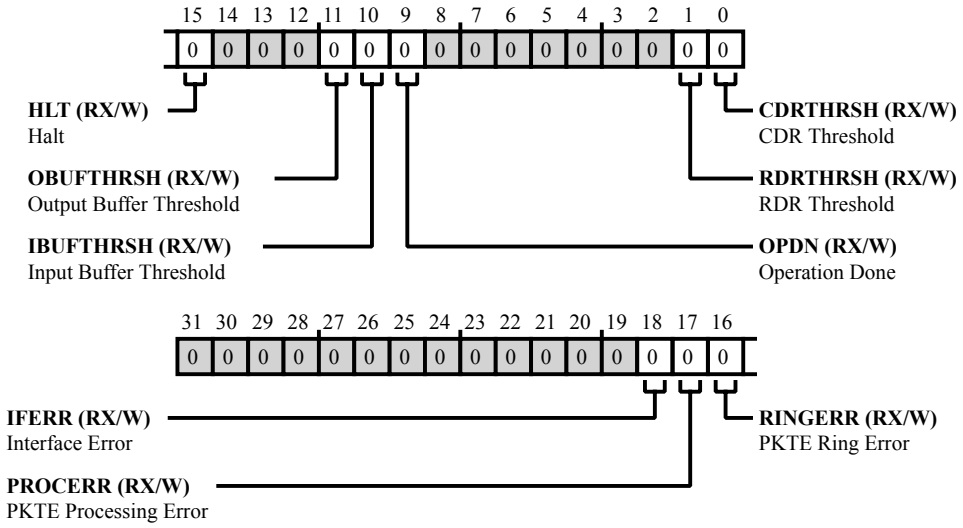

Figure 15-25: PKTE_INT_CLR Register Diagram

Table 15-47: PKTE_INT_CLR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (RX/W)	IFERR	Interface Error. The PKTE_INT_CLR.IFERR bit is set when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (RX/W)	PROCERR	PKTE Processing Error. The PKTE_INT_CLR.PROCERR bit is set when an extended error occurred before, during or after processing the current packet in the packet engine.
16 (RX/W)	RINGERR	PKTE Ring Error. The PKTE_INT_CLR.RINGERR bit is set on a CDR overflow or an RDR underflow.
15 (RX/W)	HLT	Halt. The PKTE_INT_CLR.HLT bit is set when the packet engine is in the HALT state.
11 (RX/W)	OBUFTHRSH	Output Buffer Threshold. The PKTE_INT_CLR.OBUFTHRSH bit is set when the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.
10 (RX/W)	IBUFTHRSH	Input Buffer Threshold.

Table 15-47: PKTE_INT_CLR Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
		The PKTE_INT_CLR.IBUFTHRSH bit is set when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (RX/W)	OPDN	Operation Done.
1 (RX/W)	RDRTHRSH	RDR Threshold. The PKTE_INT_CLR.RDRTHRSH bit is set when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (RX/W)	CDRTHRSH	CDR Threshold. The PKTE_INT_CLR.CDRTHRSH bit is set when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Enable Register

The PKTE_INT_EN register configures the interrupt mask for the host interrupt. This register is a bitmap for each of the possible interrupt sources. A 1 enables the interrupt source and a 0 disables the source. If an interrupt source is disabled, a cleared bit also clears the matching interrupt in the PKTE_IMSK_STAT register.

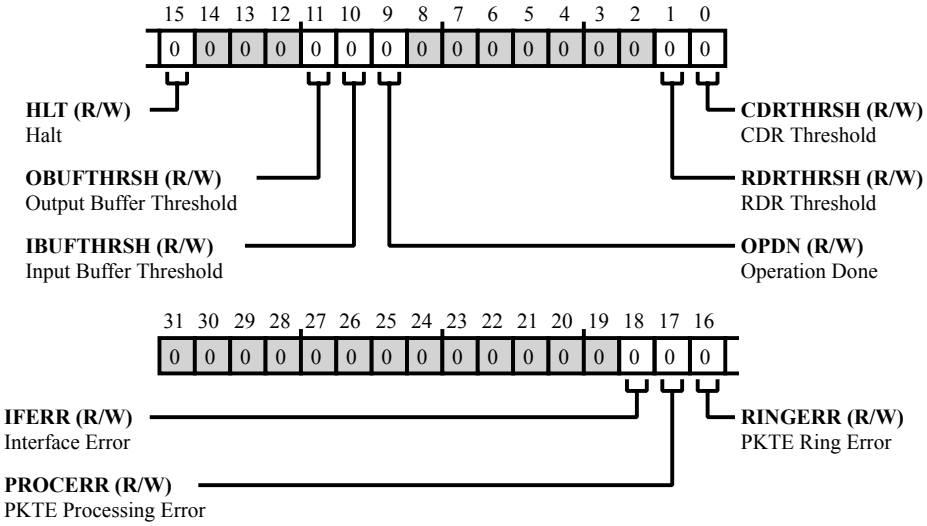

Figure 15-26: PKTE_INT_EN Register Diagram

Table 15-48: PKTE_INT_EN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/W)	IFERR	Interface Error. Set the PKTE_INT_EN.IFERR bit for host requests for a non 32-bit access to the packet engine interrupt or when the packet engine receives an error writing data back out to the host memory system.
17 (R/W)	PROCERR	PKTE Processing Error. Set the PKTE_INT_EN.PROCERR bit to enable the extended error occurred before, during or after processing the current packet in the packet engine interrupt.
16 (R/W)	RINGERR	PKTE Ring Error. Set the PKTE_INT_EN.RINGERR bit to enable the CDR overflow or RDR underflow interrupt.
15 (R/W)	HLT	Halt. Set the PKTE_INT_EN.HLT bit for when the packet engine is in the HALT state.
11 (R/W)	OBUFTHRSH	Output Buffer Threshold. Set the PKTE_INT_EN.OBUFTHRSH bit for to trigger an interrupt when the output buffer counter exceeds the output buffer threshold value defined in the PKTE_BUF_THRESH.OUTBUF bit.

Table 15-48: PKTE_INT_EN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
10 (R/W)	IBUFTHRSH	Input Buffer Threshold. Set the PKTE_INT_EN.IBUFTHRSH bit for to trigger an interrupt when the input buffer counter is less than or equal to the input buffer threshold value defined in the PKTE_BUF_THRESH.INBUF bit.
9 (R/W)	OPDN	Operation Done.
1 (R/W)	RDRTHRSH	RDR Threshold. Set the PKTE_INT_EN.RDRTHRSH bit for to trigger an interrupt when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH bit, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (R/W)	CDRTHRSH	CDR Threshold. Set the PKTE_INT_EN.CDRTHRSH bit for to trigger an interrupt when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Interrupt Unmasked Status Register

The PKTE_IUMSK_STAT register provides interrupt status visibility to the host, prior to the interrupt mask being applied. Using this register, the host can view all potential sources of incoming interrupts. All of these sources, whether masked in or out, are latched in this register and must be cleared using the PKTE_INT_CLR register in order to capture a subsequent event. A 1 indicates that the associated interrupt is present.

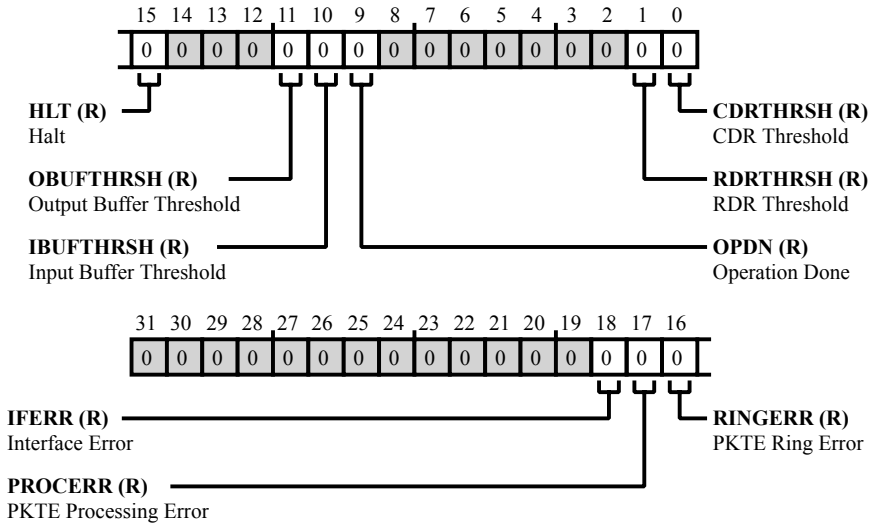

Figure 15-27: PKTE_IUMSK_STAT Register Diagram

Table 15-49: PKTE_IUMSK_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
18 (R/NW)	IFERR	Interface Error. The PKTE_IUMSK_STAT.IFERR bit is set when the host requests a non 32-bit access to the packet engine or when the packet engine receives an error writing data back out to the host memory system.
17 (R/NW)	PROCERR	PKTE Processing Error. The PKTE_IUMSK_STAT.PROCERR bit is set when an extended error occurred before, during or after processing the current packet in the packet engine.
16 (R/NW)	RINGERR	PKTE Ring Error. The PKTE_IUMSK_STAT.RINGERR bit is set on a CDR overflow or an RDR underflow.
15 (R/NW)	HLT	Halt. The PKTE_IUMSK_STAT.HLT bit is set when the packet engine is in the HALT state.
11 (R/NW)	OBUFTHRSH	Output Buffer Threshold.

Table 15-49: PKTE_IUMSK_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
		The PKTE_IUMSK_STAT.OBUFTHRSH interrupt is triggered when the output buffer counter exceeds the output buffer threshold value defined in PKTE_BUF_THRESH.OUTBUF bit.
10 (R/NW)	IBUFTHRSH	Input Buffer Threshold. The PKTE_IUMSK_STAT.IBUFTHRSH interrupt is triggered when the input buffer counter is less than or equal to the input buffer threshold value defined in PKTE_BUF_THRESH.INBUF bit.
9 (R/NW)	OPDN	Operation Done.
1 (R/NW)	RDRTHRSH	RDR Threshold. The PKTE_IUMSK_STAT.RDRTHRSH bit is set when the number of result descriptors for the host in the RDR exceeds the RD threshold value in the PKTE_RING_THRESH.RDRTHRSH, or the RD counter for the RDR in PKTE_RDSC_CNT register is non-zero for more than $2^{(N+10)}$ internal system clock cycles.
0 (R/NW)	CDRTHRSH	CDR Threshold. The PKTE_IUMSK_STAT.CDRTHRSH bit is set when the number of command descriptors for the packet engine in the CDR is less than or equal to the CD threshold value in the PKTE_RING_THRESH.CDRTHRSH bit.

Packet Engine Length Register

The PKTE_LEN register gives the length of the packet, the bypass data and a second set of ownership bits.

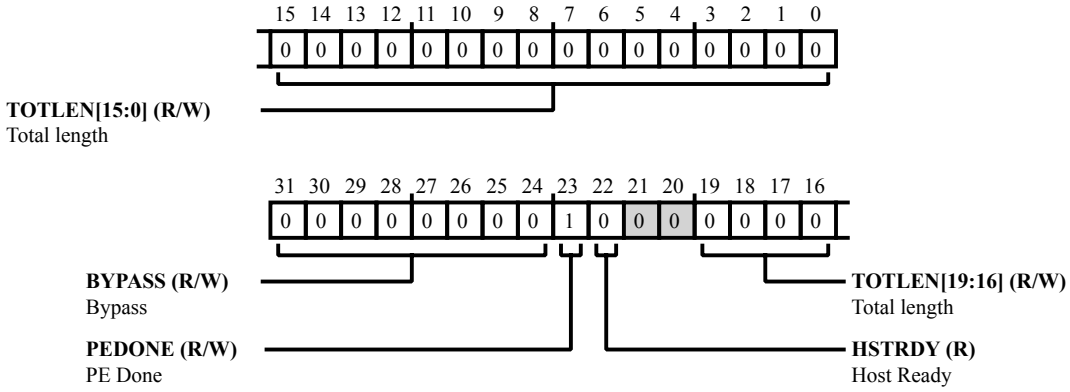

Figure 15-28: PKTE_LEN Register Diagram

Table 15-50: PKTE_LEN Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:24 (R/W)	BYPASS	Bypass. The PKTE_LEN.BYPASS bit field indicates the length of data in words that must bypass the packet engine and are directly copied from the source buffer to the destination buffer. The packet engine does not process this data. Valid bypass offsets range from 0 (0x00) to 255 (0xFF) words. For SRTP operations, this field specifies the offset in words between the hash and encrypt/decrypt data.
23 (R/W)	PEDONE	PE Done. The PKTE_LEN.PEDONE bit is a mirrored bit from the PKTE_CTL_STAT.PERDY bit. The bit is repeated here to guarantee ownership consistency between the first and last word. When the packet engine fetches a descriptor, these bits must match or the descriptor is discarded and fetched again.
22 (R/NW)	HSTRDY	Host Ready. The PKTE_LEN.HSTRDY bit is a mirrored bit of the PKTE_CTL_STAT.HOSTRDY bit. The bit is repeated here to guarantee ownership consistency between the first and last word. It should also be set along with the PKTE_CTL_STAT.HOSTRDY bit when the command descriptor is finished being populated. When the packet engine fetches a descriptor, these bits must match or the descriptor is discarded and fetched again.
19:0 (R/W)	TOTLEN	Total length. Command Descriptor: The PKTE_LEN.TOTLEN bit field indicates the total length (in bytes) of all data to be passed to the packet engines input buffer for an operation. Exceptions are the PRNG init and PRNG generate operations. The PRNG init operation does not require any input data; this field must be zero.

Table 15-50: PKTE_LEN Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
		For the PRNG generate operation, this field indicates the number of pseudo-random bytes to be generated. Valid lengths range from 16 (0x00010) to 255*16 = 4080 (0x00FF0) bytes in multiples of 16 bytes. Valid lengths for the basic operation range from 1 (0x00001) to 1,048,575 (0xFFFFF) bytes. This is the length of the data to be encrypted or hashed and includes the bypass data and padding bytes. Valid lengths for IPsec ESP range from 1 (0x00001) to 65535 (0x0FFFF) bytes. This is the length of the IP payload. Valid lengths for SSL v3.0, TLS v1.x and DTLS range from 1 (0x00001) to 16383 (0x03FFF). This is the length of the payload. Valid lengths for SRTP range from 1 (0x00001) to 65535 (0x0FFFF). This is the length of the payload. Note: A length of zero bytes is illegal and will result in an error status code in the result descriptor. Result Descriptor: Upon completion of an operation, the `PKTE_LEN.TOTLEN` field indicates the result length of the result packet. Valid lengths range from 1 (0x001) to 1,048,575 (0xFFFFF) bytes. This includes the bypass data and padding bytes. Note: When an extended error (`PKTE_CTL_STAT`[18]=1) is reported in the result descriptor and no packet data is processed, this field returns zero.

Packet Engine Output Buffer Count Register

The PKTE_OUTBUF_CNT register provides the number of data bytes there are in the output buffer. The PKTE_OUTBUF_CNT register is used in direct host mode only.

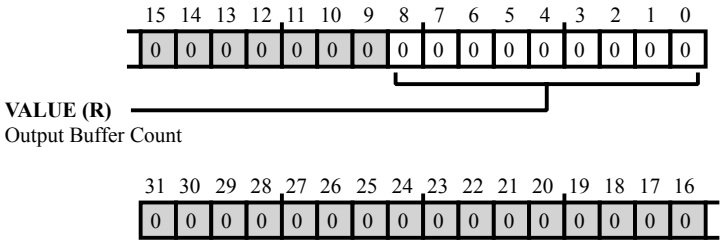

Figure 15-29: PKTE_OUTBUF_CNT Register Diagram

Table 15-51: PKTE_OUTBUF_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (R/NW)	VALUE	Output Buffer Count. The PKTE_OUTBUF_CNT.VALUE bit field provides the number of bytes in the output buffer. The packet engine increments the counter by 4 when a 32-bit word is written to the output buffer.

Packet Engine Output Buffer Count Decrement Register

A host connected via the system slave bus can decrement the output buffer counter by writing a value between 4 and 256, in multiples of 4, to the lowest bits of this register. The PKTE_OUTBUF_DECR register is used in direct host mode only.

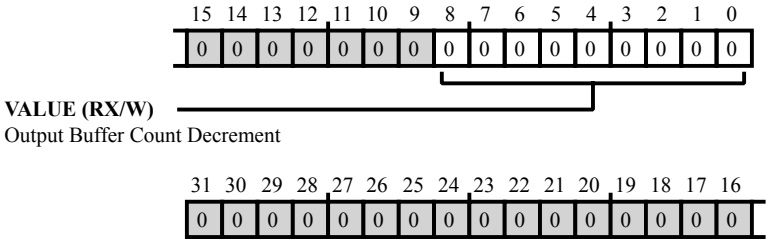

Figure 15-30: PKTE_OUTBUF_DECR Register Diagram

Table 15-52: PKTE_OUTBUF_DECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
8:0 (RX/W)	VALUE	Output Buffer Count Decrement. The PKTE_OUTBUF_DECR.VALUE bit field is the value written is subtracted to the output buffer counter. Valid values range from 4 to 256, in multiples of 4.

Packet Engine Result Descriptor Ring Base Address

The PKTE_RDRBASE_ADDR register holds the result descriptor ring base address in host memory. It is only applicable in autonomous ring mode and target command mode with RDR enabled. Note that in target command mode, the CDR is not used, but the RDR must be configured when enabled so that the packet engine knows where to write the result descriptors.

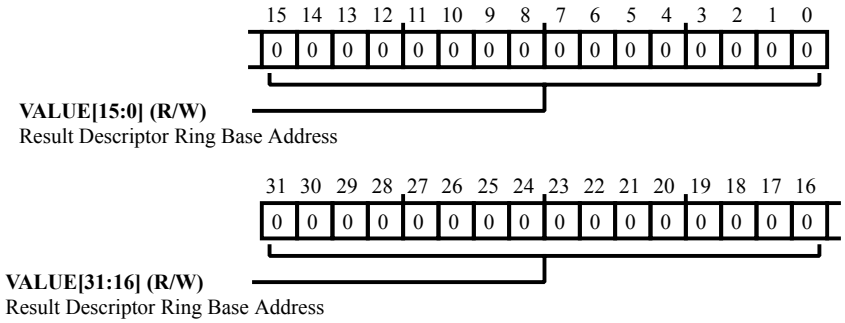

Figure 15-31: PKTE_RDRBASE_ADDR Register Diagram

Table 15-53: PKTE_RDRBASE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Result Descriptor Ring Base Address. The PKTE_RDRBASE_ADDR.VALUE bit field specifies the base location of the result descriptor ring in the host memory space.

Packet Engine Result Descriptor Count Registers

The PKTE_RDSC_CNT register holds the counter for the number of descriptors in the Result Descriptor Ring (RDR). It is incremented by the packet engine each time a valid result descriptor is written to the RDR.

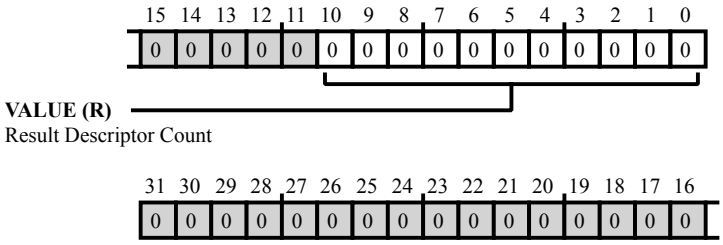

Figure 15-32: PKTE_RDSC_CNT Register Diagram

Table 15-54: PKTE_RDSC_CNT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
10:0 (R/NW)	VALUE	Result Descriptor Count. The PKTE_RDSC_CNT.VALUE bit field provides the number of result descriptors in the result descriptor ring. The packet engine increments the counter when a valid result descriptor is written to the RDR.

Packet Engine Result Descriptor Count Decrement Registers

The PKTE_RDSC_DECR register is accessible by the host connected through the system slave bus can decrement the result descriptor counter by writing a value between 1 and 255 to the lowest byte of this register.

With an RDR enabled, this is the number of result descriptors that have been read by the host. With an RDR disabled, this indicates that the host has read one valid result descriptor.

In autonomous ring mode or target command mode with the RDR enabled, the host must process 1 to 255 result descriptors from the RDR and then write this register with the number of result descriptors that have been processed by the host.

In direct host mode or target command mode with the RDR disabled, the host must read one result descriptor from the internal descriptor registers and then write this register with the value 1, to indicate that one valid descriptor is read. An RDR threshold interrupt is activated when the result descriptor counter exceeds the threshold value set in the PKTE_RING_THRESH register. This interrupt can be used to wake up a process that stalled on an empty RDR.

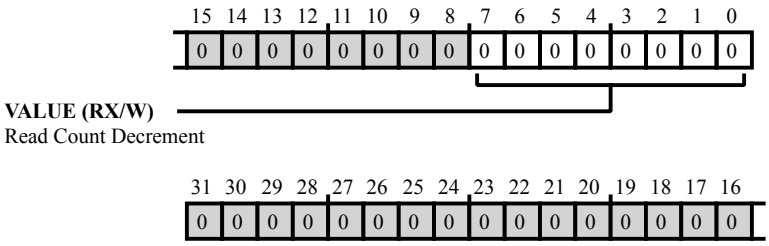

Figure 15-33: PKTE_RDSC_DECR Register Diagram

Table 15-55: PKTE_RDSC_DECR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
7:0 (RX/W)	VALUE	Read Count Decrement. The value written to the PKTE_RDSC_DECR.VALUE bit field is subtracted from the result descriptor counter. The counter is protected against underflow (See the PKTE_RING_STAT register). Note that bits [10:8] should be written with zeros.

Packet Engine Ring Configuration

The PKTE_RING_CFG register configures the size (in number of descriptor ring entries minus 1) for both the command descriptor ring and result descriptor ring in host memory. This register is only applicable for autonomous ring mode and target command mode with RDR enabled.

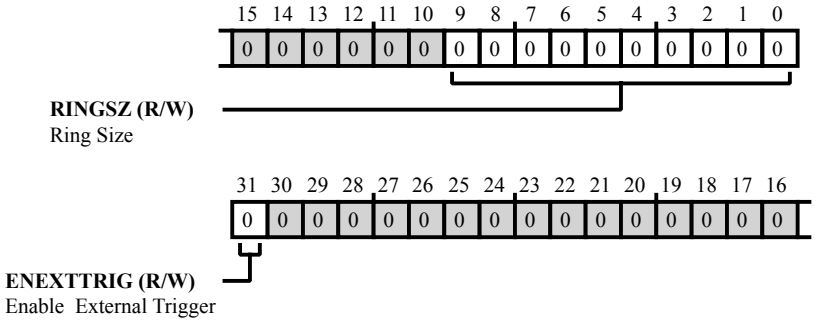

Figure 15-34: PKTE_RING_CFG Register Diagram

Table 15-56: PKTE_RING_CFG Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	ENEXTTRIG	Enable External Trigger. The PKTE_RING_CFG.ENEXTTRIG signal enables the increment of the PKTE_CDSC_CNT register through the external input pin ext_cd_cnt_incr and enables the decrement of the PKTE_RDSC_CNT fields through the external input pin ext_rd_cnt_decr.
9:0 (R/W)	RINGSZ	Ring Size. The PKTE_RING_CFG.RINGSZ bit field specifies the size of the command ring in number of descriptors, minus 1. Valid sizes range from 1 (for 2 descriptors) to 1023 (for 1024 descriptors). The accompanying result ring will have the same size.

Packet Engine Ring Pointer Status

The PKTE_RING_PTR register holds the pointers to the current entry of the Command Descriptor Ring (CDR) and Result Descriptor Ring (RDR).

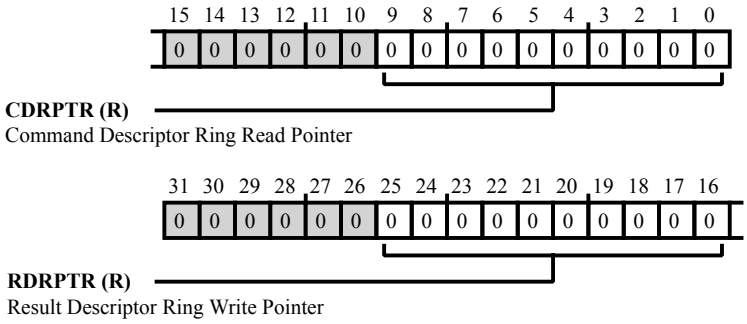

Figure 15-35: PKTE_RING_PTR Register Diagram

Table 15-57: PKTE_RING_PTR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
25:16 (R/NW)	RDRPTR	Result Descriptor Ring Write Pointer. The PKTE_RING_PTR.RDRPTR bit field indicates the entry number in the RDR that will be written next by the packet engine. The PKTE_RING_PTR.RDRPTR bit field is reset to zero after starting up and updated after every result descriptor write DMA operation. Pointers wrap around; the maximum value this field can have equals the contents of the ring size (PKTE_RING_CFG.RINGSZ) bit field.
9:0 (R/NW)	CDRPTR	Command Descriptor Ring Read Pointer. The PKTE_RING_PTR.CDRPTR bit field indicates the entry number in the CDR that will be read next by the packet engine. The PKTE_RING_PTR.CDRPTR bit field is reset to zero after starting up and updated after every command descriptor read DMA operation. Pointers wrap around; the maximum value this field can have equals the contents of the ring size (PKTE_RING_CFG.RINGSZ) field.

Packet Engine Ring Status

The PKTE_RING_STAT register gives indication of either a Command Descriptor Ring (CDR) overflow or a Result Descriptor Ring (RDR) underflow. A ring error (ringerr) interrupt in the interrupt controller is activated on a command descriptor ring overflow or a result descriptor ring underflow. This type of error can occur when the host and the packet engine get out-of-sync. The host can read this register to retrieve information on which ring is corrupted. The corrupted ring must be reset and reinitialized. See the PKTE_CFG.RSTRING bit.

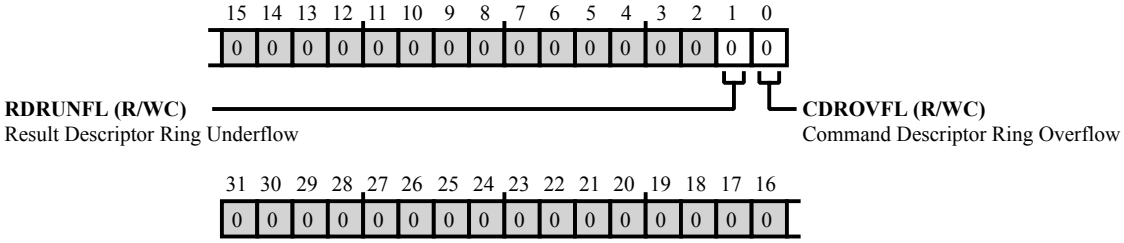

Figure 15-36: PKTE_RING_STAT Register Diagram

Table 15-58: PKTE_RING_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
1 (R/WC)	RDRUNFL	Result Descriptor Ring Underflow. The PKTE_RING_STAT.RDRUNFL bit is set when the command descriptor count (PKTE_RDSC_CNT) register is decremented below zero. This bit is reset with a write of any value.
0 (R/WC)	CDROVFL	Command Descriptor Ring Overflow. The PKTE_RING_STAT.CDROVFL bit is set when the command descriptor count (PKTE_CDSC_CNT) register is incremented above the ring size (PKTE_RING_CFG.RINGSZ) bits. This bit is reset with a write of any value.

Packet Engine Ring Threshold Registers

To reduce the amount of packet engine result interrupts, the PKTE_RING_THRESH register contains threshold and time-out values.

The CDR threshold (cdrthrsh) interrupt indicates that the command descriptor counter is less than or equal to the CDR threshold (cdrthrsh) value set in this register. This interrupt can be used to wake up a process that stalled on a full CDR.

The RDR threshold (rdrthrsh) interrupt indicates that the result descriptor counter exceeds the result descriptor threshold set here, or that the result descriptor counter is non-zero for a time longer than the result descriptor time-out setting. The RDR result interrupt remains active until the result descriptor counter is decremented below the RDR threshold (rdrthrsh) value. In case the interrupt is the result of a time-out and the result descriptor counter is below the threshold value, the result descriptor counter must be decremented once before the interrupt can be cleared in the interrupt controller.

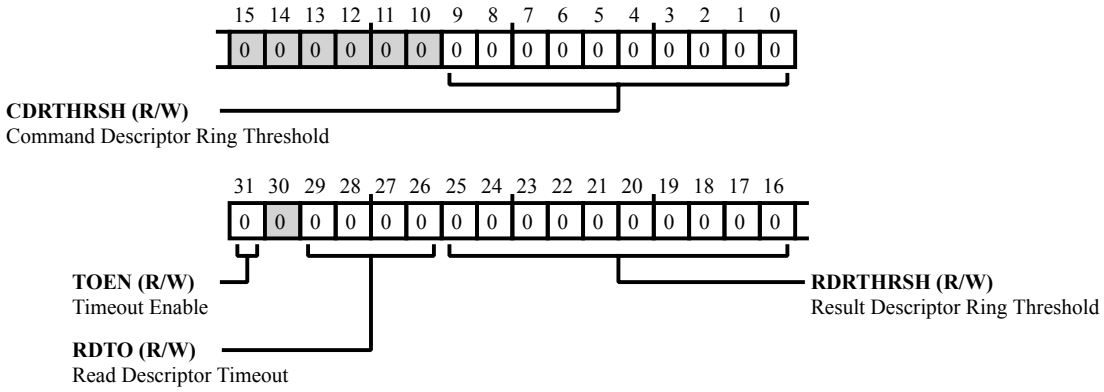

Figure 15-37: PKTE_RING_THRESH Register Diagram

Table 15-59: PKTE_RING_THRESH Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31 (R/W)	TOEN	Timeout Enable. A 1 in the PKTE_RING_THRESH.TOEN bit indicates the result descriptor timeout counter is enabled. This bit can be used to de-activate the timeout counter to save power.
29:26 (R/W)	RDTO	Read Descriptor Timeout. The timeout enable (PKTE_RING_THRESH.TOEN) bit in this register must be set to activate this PKTE_RING_THRESH.RDTO result descriptor timeout counter. The rdrthrsh interrupt activates when the RD counter for the RDR is non-zero for more than $2^{(N+10)}$ internal system clock cycles, where 'N' is the value set in this field. Valid settings range from 0 to 15. The minimum time-out value for N=0 is 1024 clock cycles and the maximum time-out value for N=15 is 33554432 clock cycles. At 100 MHz, this is 5.12 us for N=0 and ~335.55 ms for N=15.

Table 15-59: PKTE_RING_THRESH Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration
		Note: The time-out delay may not be exact - expect a variation on the order of 1024 system clock cycles (just more than one microsecond at 100 MHz system clock frequency).
25:16 (R/W)	RDRTHRSH	Result Descriptor Ring Threshold. The rdrthrsh interrupt activates when the RD counter for the RDR exceeds the value set in the `PKTE_RING_THRESH.RDRTHRSH` field. Valid settings range from 0 to 1023.
9:0 (R/W)	CDRTHRSH	Command Descriptor Ring Threshold. The cdrthrsh interrupt activates when CD counter for the CDR is below or equal the value set in the `PKTE_RING_THRESH.CDRTHRSH` field. Valid settings range from 0 to 1023.

Packet Engine SA Address

The PKTE_SA_ADDR register holds the start address of the SA record.

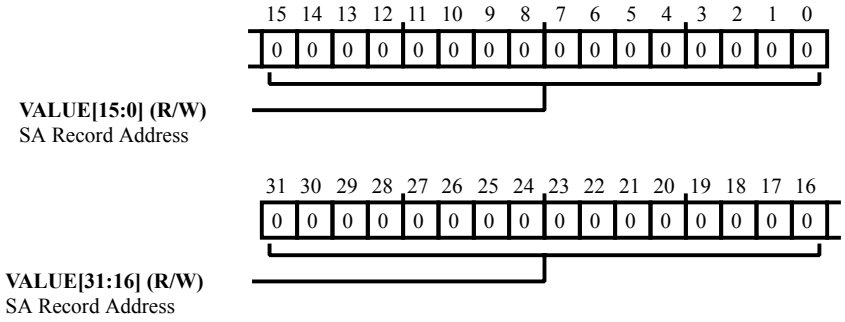

Figure 15-38: PKTE_SA_ADDR Register Diagram

Table 15-60: PKTE_SA_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Record Address. The PKTE_SA_ADDR.VALUE bit field holds the start address of the SA record.

SA Command 0

The two SA command registers, PKTE_SA_CMD0 and PKTE_SA_CMD1, are used to control the cryptographic operation of the packet engine. The PKTE_SA_CMD0 register contains the major control bits to define an operation while the PKTE_SA_CMD1 register contains the minor control bits. In direct host mode, this is a write-only register.

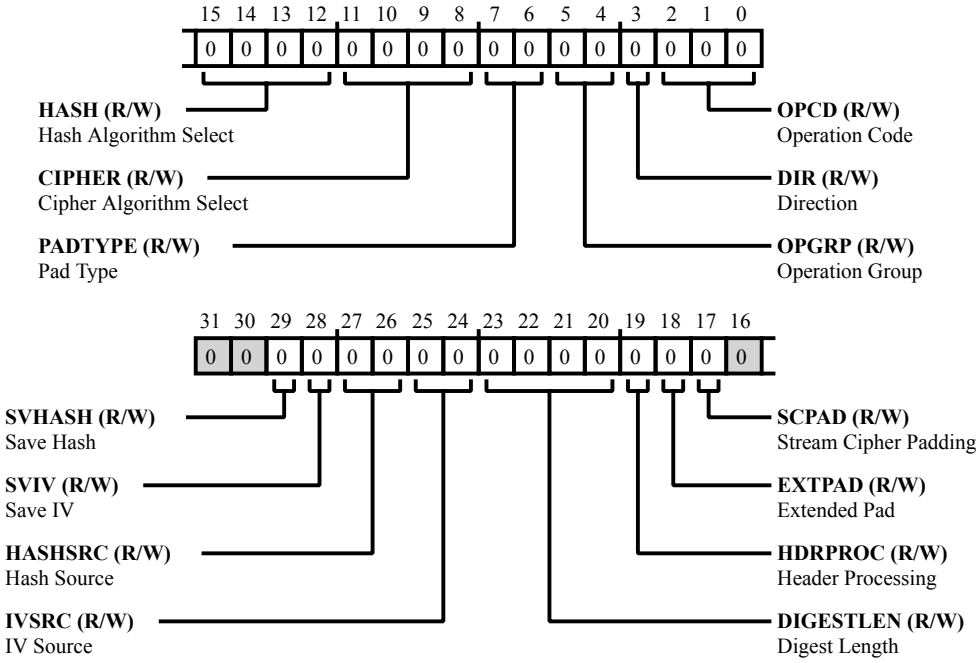

Figure 15-39: PKTE_SA_CMD0 Register Diagram

Table 15-61: PKTE_SA_CMD0 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	SVHASH	Save Hash. The PKTE_SA_CMD0.SVHASH bit indicates that the Hash State is saved to the STATE_BYTE_CNT_X and STATE_IDIGEST_X fields in the SA record in memory after completion of a crypto operation.	
		0	Hash state is not saved
		1	Hash state is saved
28 (R/W)	SVIV	Save IV. The PKTE_SA_CMD0.SVIV bit field indicates that for DES or the AES the Initialization Vector (IV) is saved to the STATE_IV_X fields in the state record after completion of the crypto operation.	
		0	IV
		1	IV

Table 15-61: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
27:26 (R/W)	HASHSRC	Hash Source. The `PKTE_SA_CMD0.HASHSRC` bit field selects the source of the hash digest used by the algorithm.	
		0	From SA. Digest only hash byte count is forced to 0x40.
		1	Reserved
		2	From State. Read saved inner hash digest and saved hash byte count.
		3	No Load. Use the hash algorithm defined constants for the initial hash. Hash byte count is 0x00.
25:24 (R/W)	IVSRC	IV Source. The `PKTE_SA_CMD0.IVSRC` bit field selects the source of the initialization vector used by the crypto algorithm.	
		0	No load. Use previous result IV, not applicable for inbound data. This option should never be used for operations with DES-CBC or AES-CBC, (see RFC3602) or any AES counter modes
		1	From input buffer. The IV is provided as part of the input data stream.
		2	From State. Read STATE_IV_X, from the SA structure. Refer to inner hash digest register structure. Useful for resume operations.
		3	From internal PRNG. Not applicable for inbound operations.
23:20 (R/W)	DIGESTLEN	Digest Length. The `PKTE_SA_CMD0.DIGESTLEN` bit field defines the length of the hash digest in words as put in the output buffer.	
		0	3 Words (96-bit output)
		1	1 Word
		2	2 Words
		3	3 Words (IPsec)
		4	4 Words (MD5 and AES-based hash)
		5	5 Words (SHA-1)
		6	6 Words
		7	7 Words (SHA-224)
		8	8 Words (SHA-256)

Table 15-61: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		9	Reserved
		10	10 bytes (SRTP and TLS)
		11-15	Reserved
19 (R/W)	HDRPROC	Header Processing. The PKTE_SA_CMD0.HDRPROC bit enables header processing for protocol operations. There is no header-processing support for basic SSL, basic TLS and SRTP protocol operations as defined in the protocol group (see the Crypto and Hash Algorithms section). This bit must be zero for these operations; however, the protocol header must be supplied to the packet engine since it is part of the hash calculation. Refer to the protocol specifications for more information about header-processing support for a protocol.	
		0	No header processing
		1	Header processing; insert the protocol header for outbound operations, verify the protocol header for inbound operations.
18 (R/W)	EXTPAD	Extended Pad. The PKTE_SA_CMD0.EXTPAD bit extends the number of padding types. Used in combination with PKTE_SA_CMD0.PADTYPE.	
17 (R/W)	SCPAD	Stream Cipher Padding. The PKTE_SA_CMD0.SCPAD bit enables padding for stream ciphers algorithms.	
15:12 (R/W)	HASH	Hash Algorithm Select. The PKTE_SA_CMD0.HASH bit field selects the hash algorithm.	
		0	MD5
		1	SHA-1
		2	SHA-224
		3	SHA-256
		4-14	Reserved
		15	Null
11:8 (R/W)	CIPHER	Cipher Algorithm Select. The PKTE_SA_CMD0.CIPHER bit field selects the cipher algorithm to be used for encryption and decryption. Note: Each type of protocol operation supports different sets of crypto algorithms. Refer to the Crypto and Hash Algorithms general processing section for details of the supported algorithms.	
		0	DES
		1	Triple-DES
		2	Reserved

Table 15-61: PKTE_SA_CMD0 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		3	AES
		4-14	Reserved
		15	Null
7:6 (R/W)	PADTYPE	Pad Type. The PKTE_SA_CMD0.PADTYPE bit field indicates the type of crypto that must be generated for outbound packets or checked for inbound packets.	
		0	Select IPSec operation (if Bit 18=0); Reserved (if Bit 18=1)
		1	PKCS#7 (if Bit 18=0); Select TLS/DTLS Pad, required for TLS/DTLS operation (if Bit 18=1)
		2	Constant pad (if Bit 18=0); Select Constant SSL Pad, required for SSL operation (if Bit 18=1)
		3	Zero pad (if Bit 18=0), Reserved (if Bit 18=1)
5:4 (R/W)	OPGRP	Operation Group. The PKTE_SA_CMD0.OPGRP bit field defines the operation groups. Refer to the Basic Operations and Decoding section for more information.	
		0	Basic operation group
		1	Protocol operation group
		2	Extended protocol operations group
		3	Reserved
3 (R/W)	DIR	Direction. The PKTE_SA_CMD0.DIR bit field selects the direction of operation.	
		0	Outbound operations
		1	Inbound operations
2:0 (R/W)	OPCD	Operation Code. The PKTE_SA_CMD0.OPCD bit field selects the operation within the operation group.	

SA Command 1

The PKTE_SA_CMD1 register contains the minor control bits that define an operation. In direct host mode, this is a write-only register.

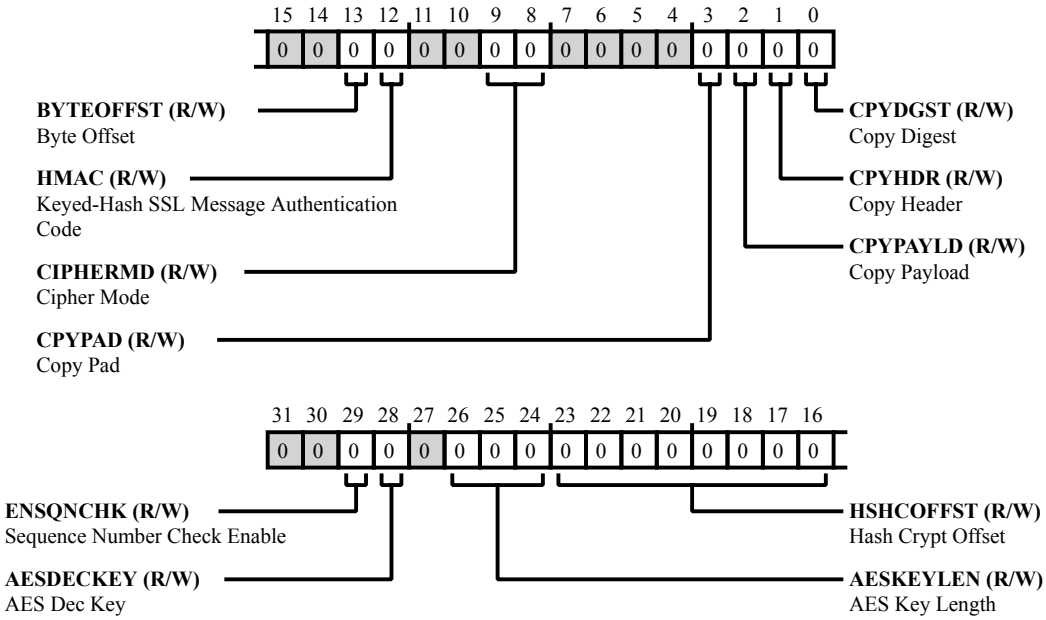

Figure 15-40: PKTE_SA_CMD1 Register Diagram

Table 15-62: PKTE_SA_CMD1 Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration	
29 (R/W)	ENSQNCHK	Sequence Number Check Enable. The PKTE_SA_CMD1.ENSQNCHK bit defines that the key in the SA key field is an AES encrypt key or an AES decrypt key.	
		0	Disable sequence number check
		1	Enable sequence number check
28 (R/W)	AESDECKEY	AES Dec Key. If the PKTE_SA_CMD1.AESDECKEY bit is set, the key in loaded in the PKTE_SA_KEY[n] registers are expected to be the key from the last round from key expansion. If not set, the key loaded in the PKTE_SA_KEY[n] registers are expected to be the same key used during the encryption process.	
		0	AES key is an encrypt key.
		1	AES key is a decrypt key.
26:24 (R/W)	AESKEYLEN	AES Key Length.	

Table 15-62: PKTE_SA_CMD1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		The PKTE_SA_CMD1.AESKEYLEN bit field select the size of the key used for the AES algorithm in increments of 64 bits.	
		0-1	Reserved
		2	128 Bits
		3	192 Bits
		4	256 Bits
		5-7	Reserved
23:16 (R/W)	HSHCOFFST	Hash Crypt Offset. For Basic Encrypt-Hash and Basic Hash-Decrypt operations, the PKTE_SA_CMD1.HSHCOFFST bit field specifies the offset between the hash data and the encrypt/decrypt data. The data to be hashed is assumed to come first, with an offset to the beginning of encrypt/decrypt data. When PKTE_SA_CMD1.BYTEOFFST, bit 13, is zero, then the offset is defined in 32-bit words. When an initialization vector is loaded through the input buffer, valid values range from IV size to 255. In all other cases, valid values range from 0 to 255. When PKTE_SA_CMD1.BYTEOFFST, bit 13, is one, then the offset is defined in 8-bit bytes. When an initialization vector is loaded through the input buffer, valid values range from IV size to 255. In all other cases, valid values range from 4 to 255. (The IV size is two words for DES, Triple-DES and AES-CTR and four words for AES-CBC and AES-ICM operations). Other operations do not use these bits (a default value is applied by the packet engine).	
13 (R/W)	BYTEOFFST	Byte Offset. The PKTE_SA_CMD1.BYTEOFFST bit defines how the PKTE_SA_CMD1.HSHCOFFST, bits of this register are used.	
		0	HASH_CRYPT_OFFSET is defined in 32-bit words
		1	HASH_CRYPT_OFFSET is defined in 8-bit bytes
12 (R/W)	HMAC	Keyed-Hash SSL Message Authentication Code. For basic operations that include hashing, the PKTE_SA_CMD1.HMAC bit enables the HMAC processing, which calls for an extra outer hash operation.	
		0	Standard Hash
		1	HMAC Processing

Table 15-62: PKTE_SA_CMD1 Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
9:8 (R/W)	CIPHERMD	Cipher Mode. The PKTE_SA_CMD1.CIPHERMD bit field selects the crypto mode to be used for the cipher algorithm.	
		0	Electronic Code Book (ECB) used for DES and AES
		1	Cipher Block Chaining (CBC) used for DES and AES
		2	AES Counter Mode (CTR) for IPsec using a 32-bit counter
		3	AES Integer Counter Mode (ICM) for SRTP using a 16-bit counter. Note: This is implemented as a 32-bit counter, the Host must check for an overflow from the value 0xFF to zero and then refresh the IV.
3 (R/W)	CPYPAD	Copy Pad. The PKTE_SA_CMD1.CPYPAD bit indicates that the padding data for an inbound operation is copied to the output buffer and saved in memory.	
		0	Do not copy the padding to output
		1	Copy padding to output
2 (R/W)	CPYPAYLD	Copy Payload. The PKTE_SA_CMD1.CPYPAYLD bit indicates that the payload data is copied to the output buffer and saved in memory.	
		0	Do not copy the payload to output
		1	Copy payload to output
1 (R/W)	CPYHDR	Copy Header. The PKTE_SA_CMD1.CPYHDR bit indicates that the protocol header is copied to the output buffer and saved in memory. For Basic Encrypt-Hash and Basic Hash-Decrypt operations, the header is defined as the Hash/Crypt Offset data (authenticated only).	
		0	Do not copy the header to output
		1	Copy header to output
0 (R/W)	CPYDGST	Copy Digest. The PKTE_SA_CMD1.CPYDGST bit copies the hash result is to the output buffer and saves in memory. The length of the hash result is defined by the PKTE_SA_CMD0.DIGESTLEN field.	
		0	Do not copy hash result to output
		1	Copy hash result to output, when the command descriptor PKTE_CTL_STAT.HASHFINAL bit is set.

SA Inner Hash Digest Registers

The PKTE_SA_IDIGEST[n] registers are a set of eight 32-bit read/write registers.

For MD5, SHA-1, SHA-224 and SHA-256, these read/write registers are used to enter a start hash state, and to read the interim or final hash digest.

For IPsec, TLS and DTLS operations that make use of MD5, SHA-1, SHA-224 or SHA-256 with basic hash or HMAC authentication with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the pre-computed inner hash digest. This is the hash of the hash-key padded with 0x36 hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-MD5 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the inner hash pre-compute; this is the hash of the MAC_WRITE_SECRET padded with 0x36 hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-SHA-1 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the MAC_WRITE_SECRET. Note that it is not possible to calculate a hash pre-compute for SHA-1 in combination with SSL-MAC (specification flaw). The packet engine appends the hash-key pad (0x36 hex) and sets the starting hash byte count automatically to 60 decimal / 0x3C hex (to indicate that 60 bytes have already been prepared for the hash).

The reset value for these registers is zero.

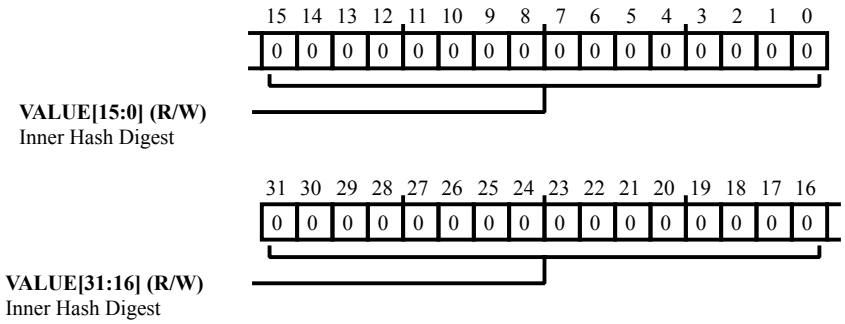

Figure 15-41: PKTE_SA_IDIGEST[n] Register Diagram

Table 15-63: PKTE_SA_IDIGEST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Inner Hash Digest.

SA Key Registers

These are the PKTE_SA_KEY[n] registers for DES, Triple-DES, and AES: A set of eight 32-bit write only registers. The reset value of these registers is zero.

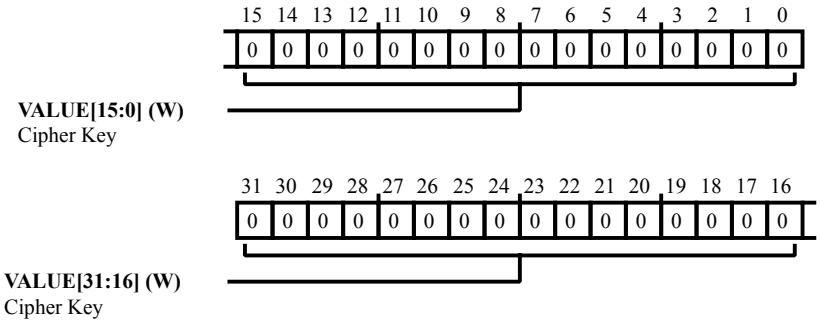

Figure 15-42: PKTE_SA_KEY[n] Register Diagram

Table 15-64: PKTE_SA_KEY[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Cipher Key.

SA Initialization Vector Register

The PKTE_SA_NONCE register is used for operations that make use of the IV value loaded from the SA record. This register is used both to enter a starting IV state, as well as for reading the interim or final IV. For IPsec outbound operations, it is recommended that the automatic IV insertion mode be used, this register is not needed. For IPsec inbound operations, the IV is extracted from the header of the packet.

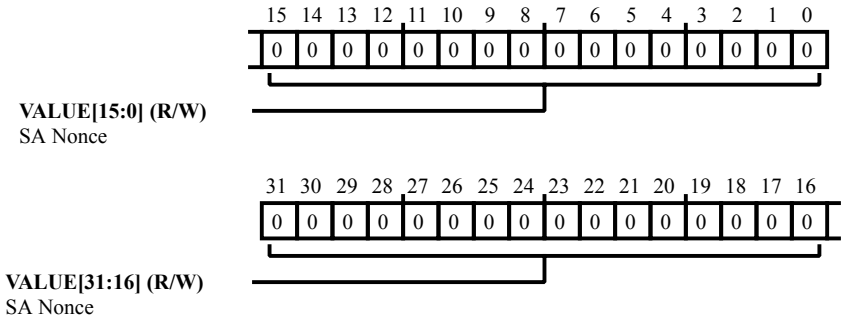

Figure 15-43: PKTE_SA_NONCE Register Diagram

Table 15-65: PKTE_SA_NONCE Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Nonce.

SA Outer Hash Digest Registers

The PKTE_SA_ODIGEST[n] registers are a set of five eight 32-bit write-only registers.

For write operations, these registers contain the pre-computed outer hash digest for IPsec operations with basic HMAC operations with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA).

For MD5, SHA-1, SHA-224 and SHA-256, these read/write registers hold a start hash state, or the interim outer hash digest. They are only used for HMAC processing.

For IPsec, SSL, TLS, DTLS and SRTP operations that make use of MD5, SHA-1, SHA-224 or SHA-256 with HMAC authentication with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the pre-computed outer hash digest. This is the hash of the hash-key padded with 0x5C hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-MD5 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the outer hash pre-compute; this is the hash of the MAC_WRITE_SECRET padded with 0x5C hex. The starting hash byte count is automatically set to 64 decimal / 0x40 hex (to indicate that 64 bytes have already been processed through the hash).

For SSL operations that make use of SSL-MAC-SHA-1 with the PKTE_SA_CMD0.HASHSRC bits = 00 (from SA), these registers hold the MAC_WRITE_SECRET. Note that it is not possible to calculate a hash pre-compute for SHA-1 in combination with SSL-MAC (specification flaw). The packet engine appends the required hash-key pad (0x5C hex) and sets the starting hash byte count automatically to 60 decimal / 0x3C hex (to indicate that 60 bytes have already been prepared for the hash).

The reset value for these registers is zero.

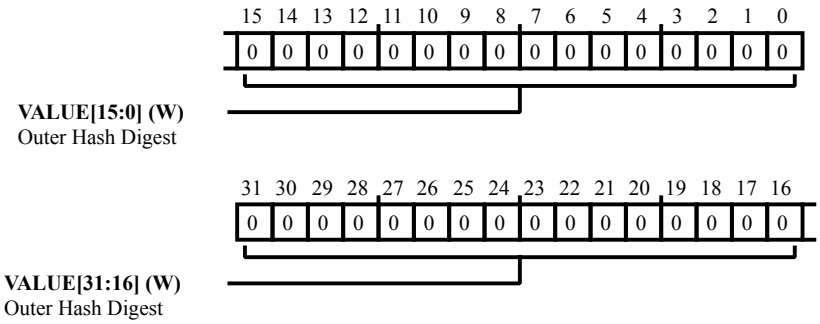

Figure 15-44: PKTE_SA_ODIGEST[n] Register Diagram

Table 15-66: PKTE_SA_ODIGEST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	Outer Hash Digest.

SA Ready Indicator

In direct host mode, a write to the PKTE_SA_RDY register triggers the packet engine to start processing using the command descriptor, SA record and state record in the packet engine registers. This register MUST be written for all direct host mode packet operations. It is intended that this register is written in sequence; as the entire SA record is written.

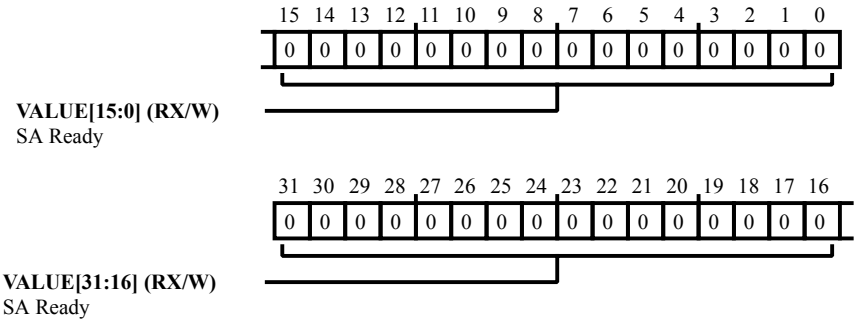

Figure 15-45: PKTE_SA_RDY Register Diagram

Table 15-67: PKTE_SA_RDY Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	SA Ready.

SA Sequence Number Register

The PKTE_SA_SEQNUM[n] registers are a set of two read/write registers and are used for IPsec ESP, SSL, TLS, DTLS operations to specify the anti-replay sequence number value that is to be placed in the ESP header (outbound), or to be checked against for inbound packets. The packet engine manages this counter value for both inbound and outbound operations.

Outbound: The host writes the counter value stored in the SA record to this register to start an IPsec, SSL, TLS, DTLS operation. The packet engine automatically increments the count if header processing is selected. Upon successful completion, the host reads back this value and writes it to the SA record.

Inbound: The host writes the counter value stored in the SA record to this register to start an IPsec or DTLS operation. The packet engine automatically performs the specified inbound processing (per RFC 4303) as it processes the packet. As a result, the expected count value may or may not be updated during processing. Upon successful completion, the host should read back this value and write it to the SA record.

Note: The description is only for the direct host mode. The sequence number for autonomous ring mode and target command mode are updated by the packet engine.

The reset value of this register is zero.

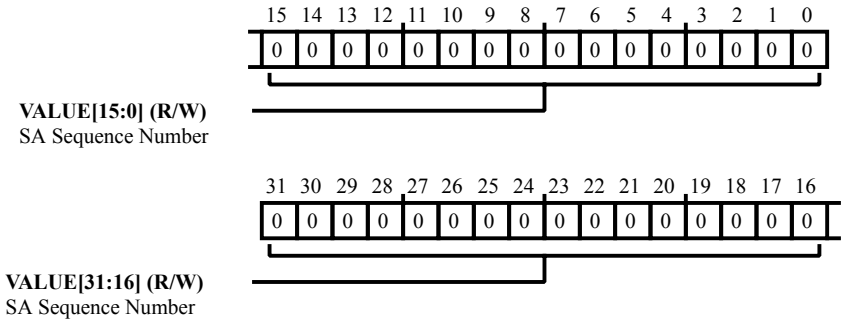

Figure 15-46: PKTE_SA_SEQNUM[n] Register Diagram

Table 15-68: PKTE_SA_SEQNUM[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Sequence Number.

SA Sequence Number Mask Registers

The PKTE_SA_SEQNUM_MSK[n] registers are a set of two read/write registers and are used for IPsec ESP and DTLS operations to specify the anti-replay sequence number mask value for inbound operations. The packet engine manages this counter value automatically.

Inbound: The host writes the counter value stored in the SA record into this register upon starting an IPsec, DTLS operation. The packet engine automatically performs the specified inbound processing (per RFC 4303) as it processes the packet. As a result, the new mask value may or may not be updated during processing. Upon successful completion, the host should read back this value and write it to the SA record.

Outbound: not used.

Note that the above description only applies to the direct host mode, for autonomous ring mode and target command mode the packet engine extracts the sequence number mask from the SA record.

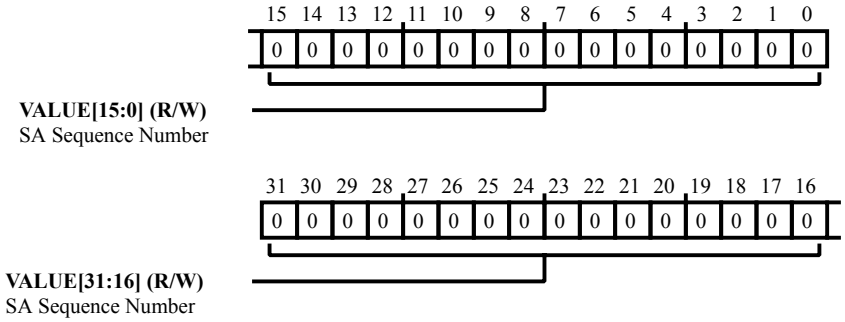

Figure 15-47: PKTE_SA_SEQNUM_MSK[n] Register Diagram

Table 15-69: PKTE_SA_SEQNUM_MSK[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	SA Sequence Number.

SA SPI Register

For IPsec operations, the PKTE_SA_SPI register is written with the SPI (Security Parameters Index) associated with the inbound or outbound flow.

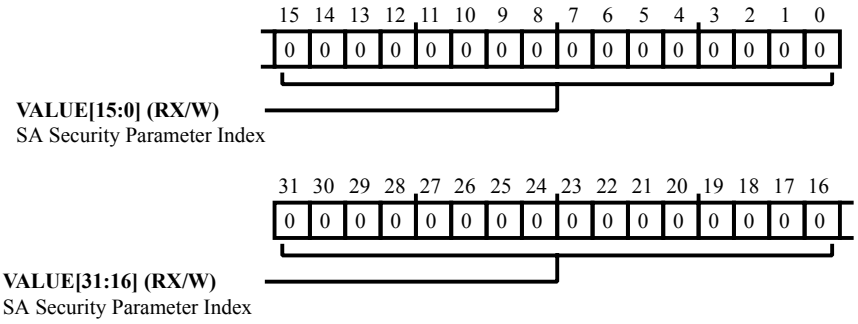

Figure 15-48: PKTE_SA_SPI Register Diagram

Table 15-70: PKTE_SA_SPI Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (RX/W)	VALUE	SA Security Parameter Index.

Packet Engine Source Address

The PKTE_SRC_ADDR register holds the starting (byte) address for the packet to be processed.

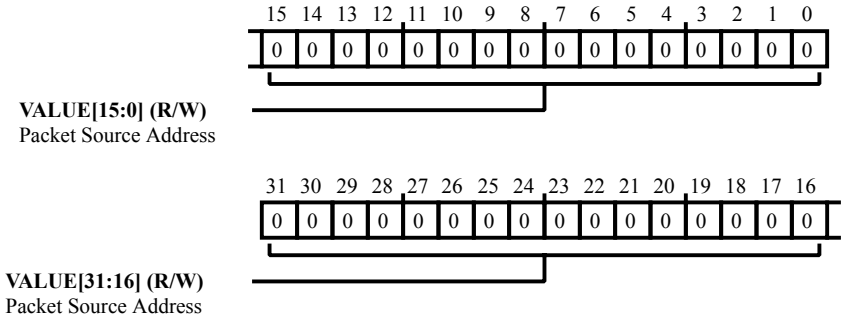

Figure 15-49: PKTE_SRC_ADDR Register Diagram

Table 15-71: PKTE_SRC_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Packet Source Address. The PKTE_SRC_ADDR.VALUE bit field holds the starting (byte) address for the packet to be processed.

Packet Engine Status Register

The PKTE_STAT register is used to provide the status of the packet engine. This register is useful in the direct host mode to determine when data must be written to or read from the packet engine, or for debugging the software when errors occur. This register can be ignored in autonomous ring mode and target command mode where the DMA engine controls the packet data I/O. This is a read-only register. A write to any of the bits has no effect.

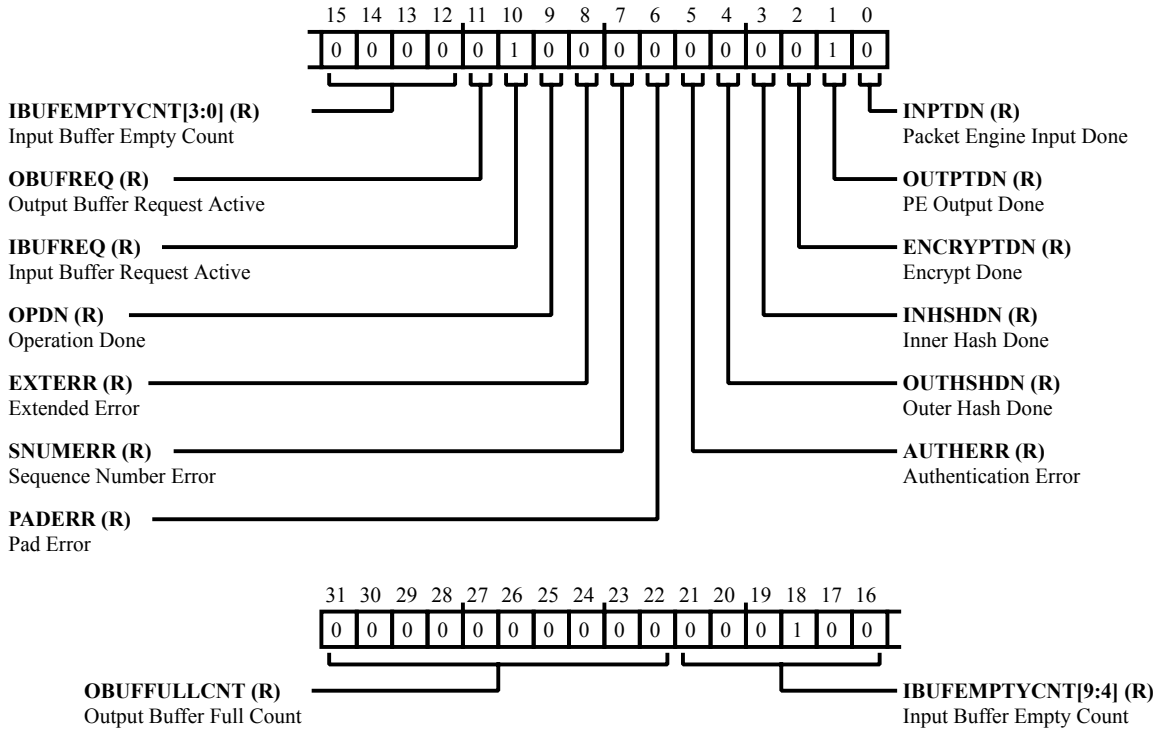

Figure 15-50: PKTE_STAT Register Diagram

Table 15-72: PKTE_STAT Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:22 (R/NW)	OBUFFULLCNT	Output Buffer Full Count. The PKTE_STAT.OBUFFULLCNT bit field indicates the number of 32-bit words that are available in the packet engine output buffer. It works in conjunction with bit 11 from this register. When bit 11 is asserted, to indicate a request for output, the word count matches the specified output buffer threshold setting in the PKTE_BUF_THRESH register. For the last output for a given packet, any value from 1 dword to the full output buffer threshold can be seen. Transfers must be a multiple of full dwords. The application must read the PKTE_LEN field in the result descriptor to determine the exact byte-length of the result.
21:12 (R/NW)	IBUFEMPTYCNT	Input Buffer Empty Count.

Table 15-72: PKTE_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
		The PKTE_STAT.IBUFEMPTYCNT bit field indicates the number of 32-bit empty spaces that are available in the packet engine input buffer. It works in conjunction with the PKTE_STAT.IBUFREQ bit (10) from this register. The value in the register is deducted from the specified packet length, so will never exceed the number of dwords that remain in the packet. For packets smaller than the buffer size, this register typically indicates that buffer space is available for the entire packet (rounded up to the nearest dword). For very large packets, these bits usually have a value around the maximum buffer size, indicating that the full input buffer is available.	
11 (R/NW)	OBUFREQ	Output Buffer Request Active. The PKTE_STAT.OBUFREQ bit indicates that the packet engine requests output data to be read from the output buffer.	
		0	No request for output data
		1	Request for output data
10 (R/NW)	IBUFREQ	Input Buffer Request Active. The PKTE_STAT.IBUFREQ bit indicates that the packet engine requests input data to be written to the input buffer.	
		0	No request for input data
		1	Request for input data
9 (R/NW)	OPDN	Operation Done. The PKTE_STAT.OPDN bit indicates that the packet engine has finished processing a packet when in direct host mode. This bit is zero in autonomous ring mode and target command mode.	
		0	Packet engine is idle
		1	Packet engine has finished processing a packet
8 (R/NW)	EXTERR	Extended Error. The PKTE_STAT.EXTERR bit indicates that an extended error occurred for this packet. For more information, refer to table Extended Error Codes - Status Encoding.	
		0	No extended error
		1	Extended error

Table 15-72: PKTE_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
7 (R/NW)	SNUMERR	Sequence Number Error. For an inbound operation, the PKTE_STAT.SNUMERR bit indicates that there was a fault in the anti-replay sequence number. For an outbound operation, there was a sequence number overflow condition. For more information, refer to table Extended Error Codes - Status Encoding.	
		0	No sequence number error
		1	Input done, all bytes written to input buffer
6 (R/NW)	PADERR	Pad Error. The PKTE_STAT.PADERR bit indicates that an inbound crypto pad fault is detected. For more information about pad verification, refer to the Pad Verification and Consumption section.	
		0	No pad error
		1	Pad error
5 (R/NW)	AUTHERR	Authentication Error. The PKTE_STAT.AUTHERR bit indicates that an inbound ICV (for IPsec) or TAG (for SRTP) or MAC (for SSL/TLS/DTLS) fault is detected; the value carried within the packet did not match the value just computed.	
		0	No authentication error
		1	Authentication error
4 (R/NW)	OUTHSHDN	Outer Hash Done. The PKTE_STAT.OUTHSHDN bit indicates that the outer hash processing for this packet is finished.	
		0	Outer hash busy
		1	Outer hash done
3 (R/NW)	INHSHDN	Inner Hash Done. The PKTE_STAT.INHSHDN bit indicates that the inner hash processing for this packet is finished.	
		0	Inner hash busy
		1	Inner hash done
2 (R/NW)	ENCRYPTDN	Encrypt Done. The PKTE_STAT.ENCRYPTDN bit indicates that the encryption or decryption for this packet is finished.	
		0	Encryption or decryption busy
		1	Encryption or decryption done

Table 15-72: PKTE_STAT Register Fields (Continued)

Bit No. (Access)	Bit Name	Description/Enumeration	
1 (R/NW)	OUTPTDN	PE Output Done. The PKTE_STAT.OUTPTDN bit indicates that the output data for the current packet is read from the packet engine output buffer.	
		0	Output not done, more output bytes available
		1	Output done, all bytes read from the output buffer
0 (R/NW)	INPTDN	Packet Engine Input Done. The PKTE_STAT.INPTDN bit indicates that the number of bytes specified in the command descriptor PKTE_LEN field is written into the packet engine input buffer.	
		0	Input not done, more input bytes expected
		1	Input done, all bytes written to input buffer

Packet Engine State Record Address

The PKTE_STATE_ADDR register holds the start address of the SA state record.

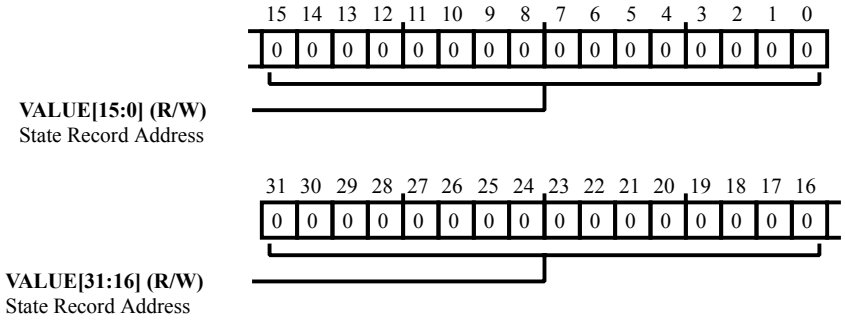

Figure 15-51: PKTE_STATE_ADDR Register Diagram

Table 15-73: PKTE_STATE_ADDR Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	State Record Address. The PKTE_STATE_ADDR.VALUE bit field holds the start address of the SA state record.

State Hash Byte Count Registers

The PKTE_STATE_BYTE_CNT[n] registers are used to enter a starting hash byte count, as well as to read the interim or final byte count.

For some hash operations, these registers are ignored and the byte count is internally set to 64 (0x40 hex) to indicate that the first 64 bytes (512 bits) hash block has been processed using a pre-computed hash state. These operations are:

All IPsec, SSL, TLS, DTLS and SRTP operations that use authentication; the "pre-computed" inner and outer hash digests are loaded from SA words 10 - 19.

Basic operations with PKTE_SA_CMD0.HASHSRC bits = 00 (from SA) specified. For Basic Hash with no HMAC, a pre-computed digest is loaded from SA words 10 - 14. For Basic Hash with HMAC, the inner and outer digests are loaded from SA words 10 - 19.

Note: Protocol operations can not be suspended in mid-packet and resumed later, therefore protocol operations do not use these registers.

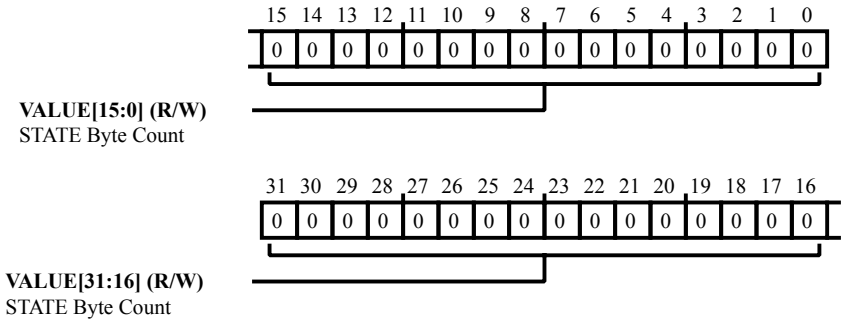

Figure 15-52: PKTE_STATE_BYTE_CNT[n] Register Diagram

Table 15-74: PKTE_STATE_BYTE_CNT[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	STATE Byte Count.

State Inner Digest Registers

The PKTE_STATE_IDIGEST[n] registers consist of eight 32-bit registers. These read/write registers are used to read the interim or final hash digest. The PKTE_STATE_IDIGEST[n] registers are only used with basic operations involving basic hash, and are typically used for operations that must be suspended and resumed in the middle of a hash. The interim hash state can be read from these registers along with the hash byte-count from the previous register. Both can be restored when resuming the hash. The appropriate save hash state (PKTE_SA_CMD0.SVHASH=1) and load hash from state (PKTE_SA_CMD0.HASHSRC=0b10) settings must be used. These registers are a mirror of the SA record inner hash digest register.

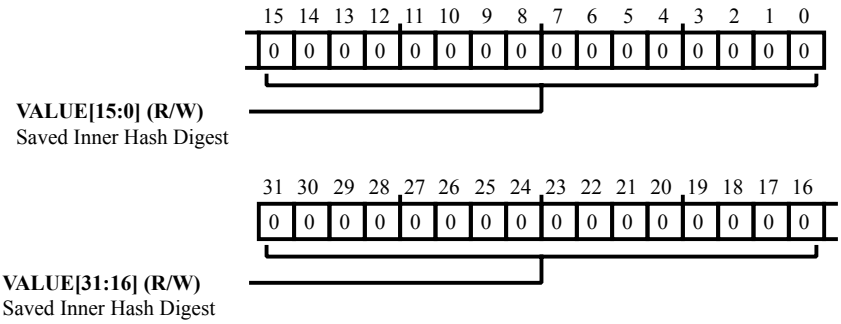

Figure 15-53: PKTE_STATE_IDIGEST[n] Register Diagram

Table 15-75: PKTE_STATE_IDIGEST[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Saved Inner Hash Digest.

State Initialization Vector Registers

The PKTE_STATE_IV[n] consists of four 32-bit registers. These registers are used to enter a starting IV state and to read the interim or final IV. PKTE_STATE_IV0 and PKTE_STATE_IV1 are used with DES/3DES cipher while PKTE_STATE_IV0 to PKTE_STATE_IV3 are used with AES cipher. The reset value of these registers is zero.

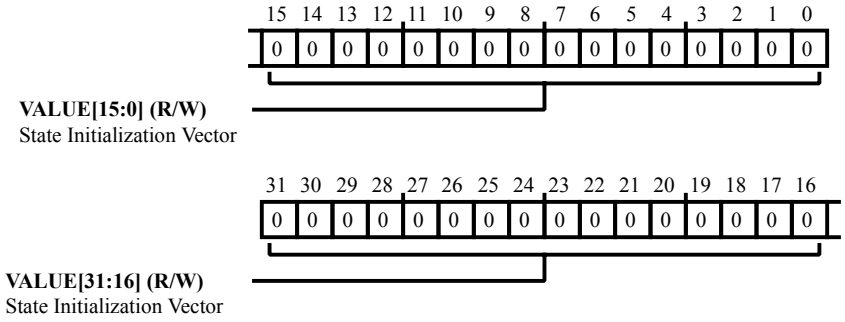

Figure 15-54: PKTE_STATE_IV[n] Register Diagram

Table 15-76: PKTE_STATE_IV[n] Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	State Initialization Vector. The PKTE_STATE_IV[n].VALUE bit field is used to enter a starting IV state and to read the interim or final IV.

Packet Engine User ID

The PKTE_USERID register is a read/write register that gives identification to a command descriptor and the resultant result descriptor. The host is free to use this field for its own purpose. The host can write a unique identifier to the register in direct host mode or includes it as part of the command descriptor in autonomous ring mode. The PKTE_USERID register value passes though the packet engine without alteration to the result descriptor to be read back by the host.

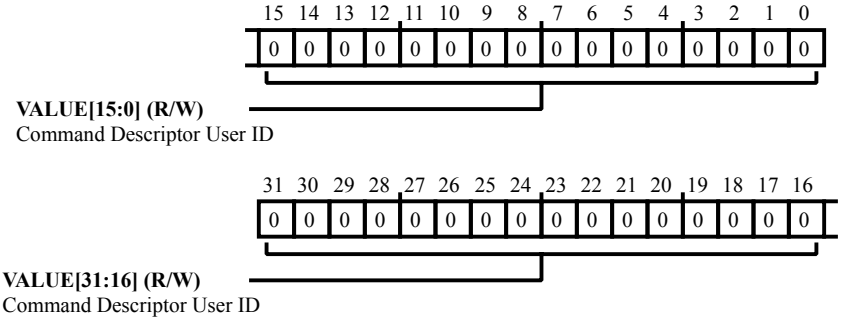

Figure 15-55: PKTE_USERID Register Diagram

Table 15-77: PKTE_USERID Register Fields

Bit No. (Access)	Bit Name	Description/Enumeration
31:0 (R/W)	VALUE	Command Descriptor User ID. The PKTE_USERID.VALUE bit field gives identification to a command descriptor and the resultant result descriptor.