ADVANCE DATA SHEET

BCM5812

Security Processor

GENERAL DESCRIPTION

The BCM5812 is a full-feature security processor optimized to provide multi-protocol cryptographic acceleration for cost sensitive VPN and eCommerce applications.

The BCM5812 supports all of the symmetric and asymmetric encryption and authentication algorithms popularly used by IPsec and IKE, including 3DES, AES, RSA, DSA, Diffie-Hellman, SHA-1, MD5, HMAC-SHA-1, and HMAC-MD5. For IPsec ESP processing, the BCM5812 performs single pass encryption combined with HMAC authentication. For SSL and TLS record processing, the BCM5812 provides ARCFOUR encipherment, and computes the SSL MAC and TLS HMAC. The BCM5812 includes a true random number generator.

The BCM5812 offers a complete, single-chip, solution that connects directly to the PCI bus with no need for external interface logic or memory.

The BCM5812 utilizes 0.18 µm semiconductor technology, a high performance bus master interface, and efficient software control structures in an advanced design.

Single-pass IPsec combined encryption and authentication SSL MAC, TLS HMAC for SSL, TLS Record Layer processing Export mode, enables Retail export classification.

FEATURES

- IETF and FIPS compliant algorithms:
  - 3DES-CBC. DES-CBC
  - AES-CBC, AES-CTR with 128, 192, and 256 bit key sizes
  - ARCFOUR
  - MD5, SHA-1 HMAC-MD5, HMAC-SHA-1
  - RSA Public and Private Key operations up to 2048 bit modulus
  - 1024 bit DSA Sign and Verify
  - DIffie-Hellman Key Generation and Agreement up to 2048 bit

FEATURES

- IETF and FIPS compliant algorithms:
  - 3DES-CBC. DES-CBC
  - AES-CBC, AES-CTR with 128, 192, and 256 bit key sizes
  - ARCFOUR
  - MD5, SHA-1 HMAC-MD5, HMAC-SHA-1
  - RSA public and private key operations up to 2048 bit modulus
  - 1024 bit DSA Sign and Verify
  - DIffie-Hellman key generation and agreement up to 2048 bit

- True random number generator
- Modular math functions with up to 2048 bit modulus
- PCI v2.2 32 bit 33 MHz bus interface
- Advanced testability features
  - 100% testability of on-chip RAM cells via BIST
  - JTAG boundary scan for board level testing
- Low-power 1.8V core, 5V tolerant, 3.3V I/O, Power-saving mode
- Small footprint 196-pin FBGA package
- 100% Software Compatible with the BCM5823
- Extensive Software Support
  - Drivers for Linux, VxWorks, Solaris, Windows 2000
  - Full SDK with application library and diagnostics

APPLICATIONS

- VPN residential gateways
- VPN SoHo firewalls, routers and switches
- Secure wireless access points and gateways

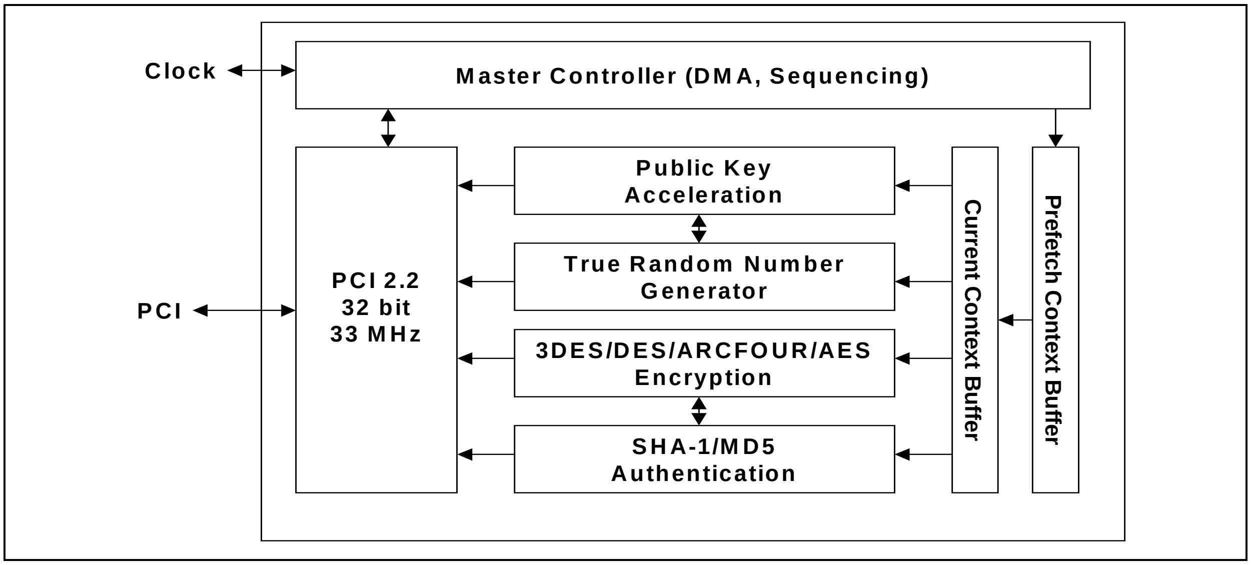


Figure 1: Functional Block Diagram

REVISION HISTORY

Revision	*Date*	*Change Description*
5812-DS01-R	3/11/03	Corrected caption for Figure 41.
5812-DS00-R	2/14/03	Initial release.

Broadcom Corporation
P.O. Box 57013
16215 Alton Parkway
Irvine, California 92619-7013

This data sheet (including, without limitation, the Broadcom component(s) identified herein) is not designed, intended, or certified for use in any military, nuclear, medical, mass transportation, aviation, navigations, pollution control, hazardous substances management, or other high risk application. BROADCOM PROVIDES THIS DATA SHEET "AS-IS", WITHOUT WARRANTY OF ANY KIND. BROADCOM DISCLAIMS ALL WARRANTIES, EXPRESSED AND IMPLIED, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Section 1: Functional Description

OVERVIEW

The BCM5812 is a low-cost, general purpose security processor that incorporates specialized features for IPsec, IKE, SSL, and TLS protocol processing. The BCM5812 is a member of Broadcom's Security Processor family and fully software compatible with the higher performance BCM5821 and BCM5820.

The BCM5812 provides bulk cryptographic acceleration for the 3DES, DES, AES, and ARCFOUR symmetric encryption algorithms, for the SHA-1 and MD5 hash algorithms, and for the HMAC-SHA-1 and HMAC-MD5 keyed authentication algorithms. It provides public key acceleration for the RSA, DSA, and Diffie-Hellman asymmetric algorithms, as well as basic modular math functions. The BCM5812 provides a true random number generator, and can use it to generate on-chip random values for Diffie-Hellman key generation and DSA signatures.

The BCM5812 provides combined encryption and HMAC authentication for single pass IPsec processing. It also provides the SSL MAC and TLS HMAC functions needed for SSL and TLS record layer processing, respectively.

Bulk encryption and public key processing are performed in separate processing functions within the BCM5812. These share a common bus interface, but are otherwise independent. The BCM5812 can thus complete multiple bulk encryption operations while simultaneously executing a relatively slower public key operation, without having to stall bulk processing.

The BCM5812 is ideal as a single-chip, low-cost security solution for both embedded systems and add-in option modules. It interfaces directly to the PCI bus with no need for additional interface logic, operates without external memory, and can derive its clock from the PCI bus. It supports all of the algorithms needed for IPsec, IKE, SSL, and TLS security protocols, and used as well by many other protocols, such as Secure RTP.

PROGRAMMING INTERFACE

The BCM5812 is a bus master PCI device that uses programming structures designed to facilitate pre-fetching for high device utilization. These structures are built by the host processor in main memory and accessed by the BCM5812 using DMA, as diagrammed in Figure 2.

The BCM5812 is controlled using a block of five Control and Status Registers (CSR). The host processor maps the CSR block into PCI memory space, usually at system initialization, by writing the start address of the CSR block into BCM5812 PCI configuration register BAR0. Host software accesses the BCM5812 CSR block by doing PCI memory (slave) reads and writes starting at this address.

The main programming structure is the Master Command Record (MCR), shown in Figure 3 on page 3. The MCR consists of a header word followed by an array of one or more packet descriptors. A single MCR can accommodate up to 65535 packet descriptors. Table 1 on page 2 shows the fields in the header word that would be set by the host processor before giving the MCR to the BCM5812.

The host software gives an MCR to the BCM5812 by writing the address of the MCR structure to either the MCR1@ or MCR2@ CSR, depending on the type of operation (see Table 3 on page 5). The BCM5812 provides a four-level FIFO behind MCR1@ and MCR2@, which permits the host software to push up to four MCR addresses at a time to each. The host software can sense the MCR1_FULL and MCR2_FULL flags in the DMA Status CSR, bits 30 and 27 respectively. These flags are zero when their respective MCR FIFO can accommodate at least one new MCR address.

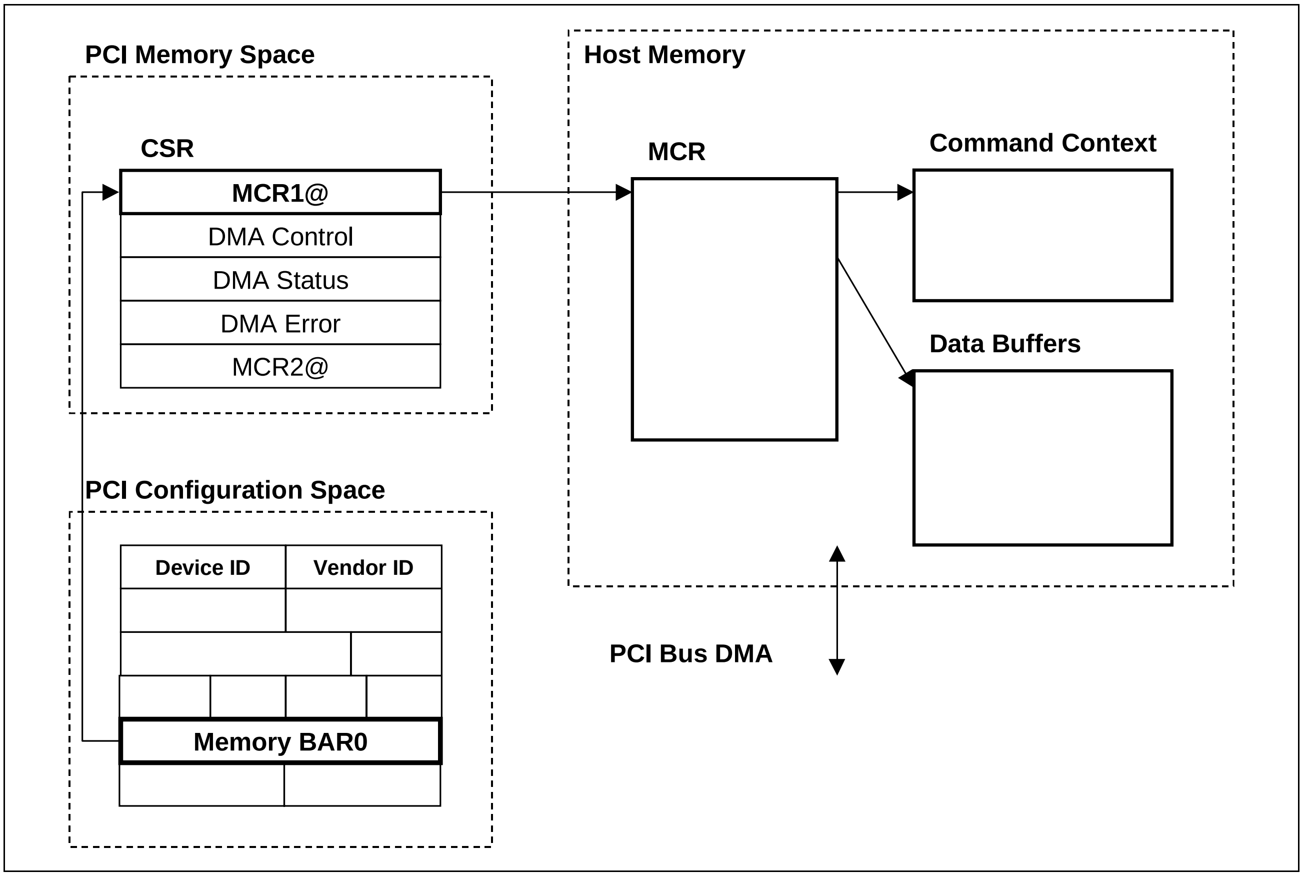


Figure 2: BCM5812 Programming Overview

Table 1: MCR Header Word (Input)

Bits	*Description*
31	Suppress Interrupts
30:16	Reserved
15:0	Number of Packets

Each packet descriptor entry is 32 bytes long, and contains a pointer to a command context structure along with initial input and output fragment chain entries. The chain entry structures are buffer descriptors used to support input and output scatter-gather operations. Each provides the address and length, in bytes, of a buffer fragment, and a pointer to a next fragment chain entry. The packet descriptor also supplies the total length of the input. Figure 4 on page 4 diagrams the format of a packet descriptor.

The command context indicates the actual operation to be performed. The first 32-bit word is common to all command context types, and indicates the operation to be performed and the total length of the command context structure itself, in bytes. Everything else in a command context is operation specific, as is how the BCM5812 interprets the fragment lists. The operations performed by the BCM5812, along with their command context, input, and output, are described under "Cryptographic Operations" on page 5.

All reserved fields must be zero and, unless otherwise noted, all lengths are in bytes.

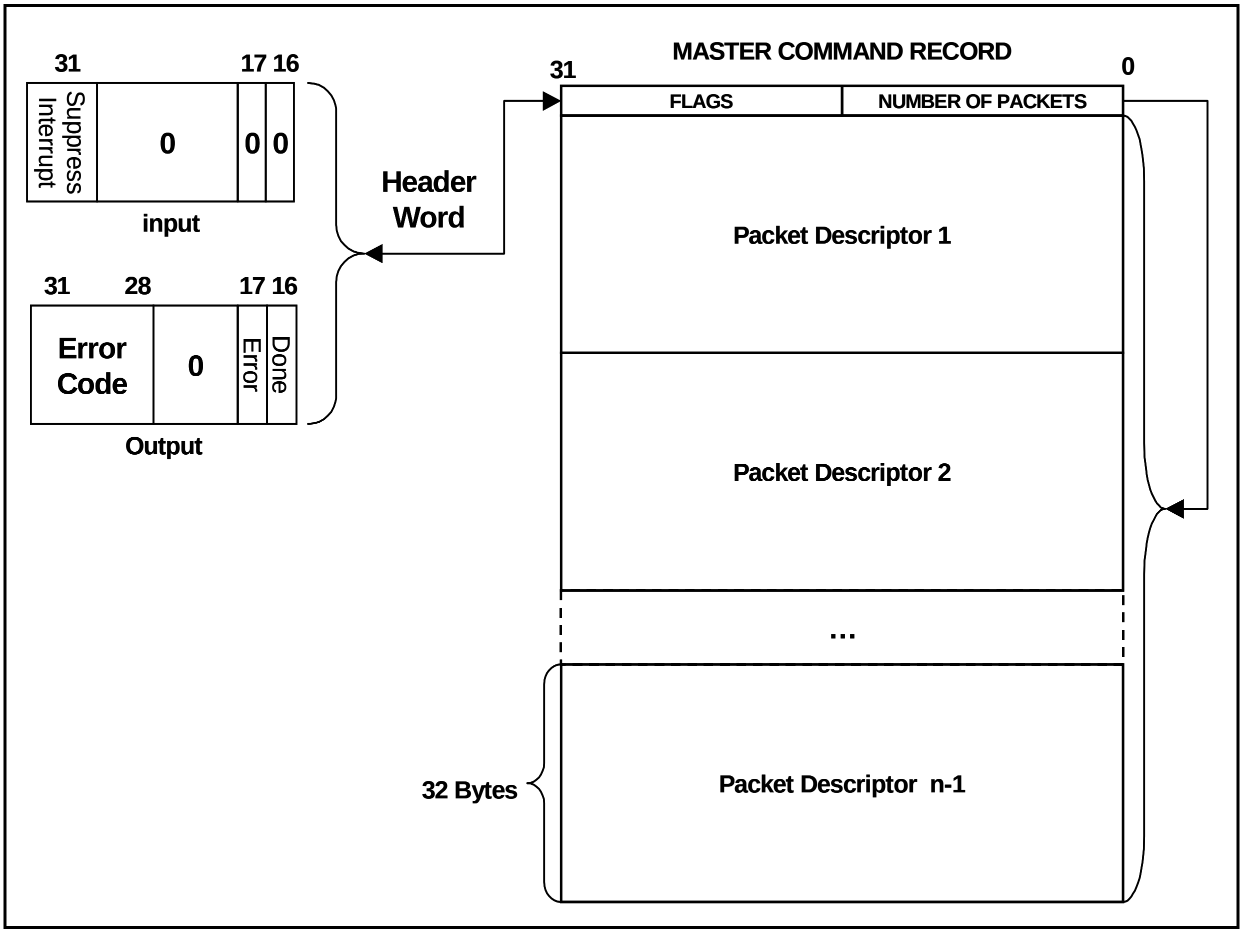


Figure 3: Master Command Record Format

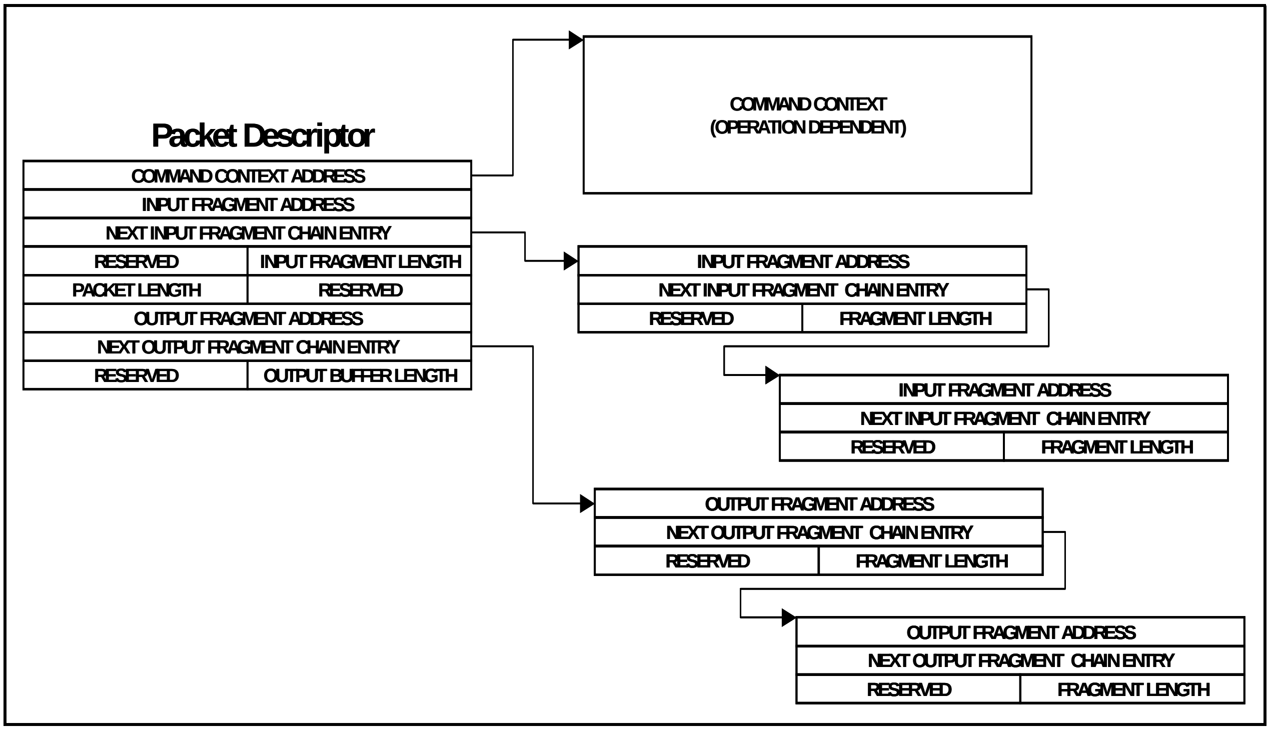


Figure 4: Packet Descriptor Format

The packet descriptors within a single MCR structure can be for different cryptographic operations. In some cases, the output of a packet descriptor cannot be used in the input to the immediate following packet descriptor operation in the list, but can always be used for a subsequent packet descriptor input (see “Chaining Operation Dependencies” on page 70).

After it completes processing on all of the packet descriptors in the MCR, the BCM5812 updates the header word as shown in Table 2. The Done bit (bit 16) is always set. The error code contains an error value if the Error bit (bit 17) is one.

Table 2: MCR Header Word (Output)

Bits	***Description***
31:28	Error Code: • 0000 = Normal Completion • 0001 = Unknown Opcode • 0010 = Not Enough Input Data for DSA Operations • 0011 = Not Enough input Data for Public Key Operations • 0100 = Not Enough Output Data for Public Key Operations • 0101 = Input Fragment Chain too short • 0110 = PCI Read FIFO non-empty
27:18	0
17	Error
16	Done

Table 2: MCR Header Word (Output) (Cont.)

Bits	Description
15:0	Number of Packets (unchanged from input)

CRYPTOGRAPHIC OPERATIONS

Each cryptographic operation performed by the BCM5812 is described in terms of its command context and how it uses input and output buffer fragment lists. Table 3 lists the operations performed by the BCM5812. Operation type codes depend upon whether the command context is programmed using MCR1@ or MCR2@. Subsequent sections describe the command context for each operation in detail.

Table 3: Operation Types

OPCODE	MCR1@ (Symmetric Operations)	MCR2@ (Asymmetric Operations)
0x00	IPsec 3DES/DES combined encryption with authentication	Reserved
0x01	SSL MAC authenticator calculation	Diffie-Hellman public key generation
0x02	TLS HMAC authenticator calculation	Diffie-Hellman shared key generation
0x03	3DES/DES encryption (for SSL and TLS)	RSA public key operation
0x04	ARCFOUR encipherment (for SSL and TLS)	RSA secret key operation
0x05	Hash (Pure MD5 or SHA-1)	DSA signing operation
0x06	Reserved	DSA verification operation
0x40	IPsec AES combined encryption with authentication	Reserved
0x41	Reserved	RNG direct
0x42	Reserved	RNG SHA-1
0x43	Reserved	Modular Addition
0x44	Reserved	Modular Subtraction
0x45	Reserved	Modular Multiplication
0x46	Reserved	Modular Reduction
0x47	Reserved	Modular Exponentiation
0x48	Reserved	Reserved
0x49	Reserved	Double Modular Exponentiation

Programming structures are described relative to a little endian host processor system, where the layout in host memory and on the PCI bus is the same. See “Endian Considerations” on page 42 for a discussion of the differences on big endian systems.

Note

- **An MCR must contain at least one packet descriptor.**
- **The minimum command context length actually read by the BCM5812 is 64 bytes.**

IPSEC 3DES

The BCM5812 performs FIPS-46-3 [11] compliant DES-CBC and 3DES-CBC bulk encryption and decryption, combined with RFC-2104 [10] compliant SHA-1-HMAC or MD5-HMAC. These algorithms are the primary ones used for the IPsec (RFC2401 [2]) ESP and AH Security protocols (RFC2406 [7] and RFC2402 [3], respectively). Cipher Block Chaining Mode (CBC) is diagrammed in Figure 7, taken from NIST Special Publication 800-38A [16].

The SHA-1 provides features specifically to support the IPsec use of 3DES and DES as described in RFC2406 [7], and for HMAC-SHA-1 (RFC-2404 [5]) and HMAC-MD5 (RFC2403 [4]). Figure 5 shows the command context structure used for combined 3DES or DES encryption, with HMAC-SHA-1 or HMAC-MD5 authentication, for IPsec processing. The flags, detailed in Table 4 on page 7, indicate the specific operations to be performed. The command context should always include three DES keys, even for single DES, so that the length of this structure is always 80 bytes.

The command context supplies the keys, initialization vector (IV), and authentication contexts as shown in Figure 5. The opcode for this command, which uses MCR1@, is 0x00.

Authentication contexts are partial HMAC calculations pre-computed using the authentication secret. Figure 6 on page 8 diagrams the computation for the HMAC inner and outer state. A 20 byte field is used for both the HMAC-SHA-1 and HMAC-MD5 states. HMAC-MD5 only uses the first 16 bytes, and the remaining 4 bytes should be set to zeros.

Note

Broadcom supplies a software routine in the Software Reference Library for computing HMAC inner and outer states.

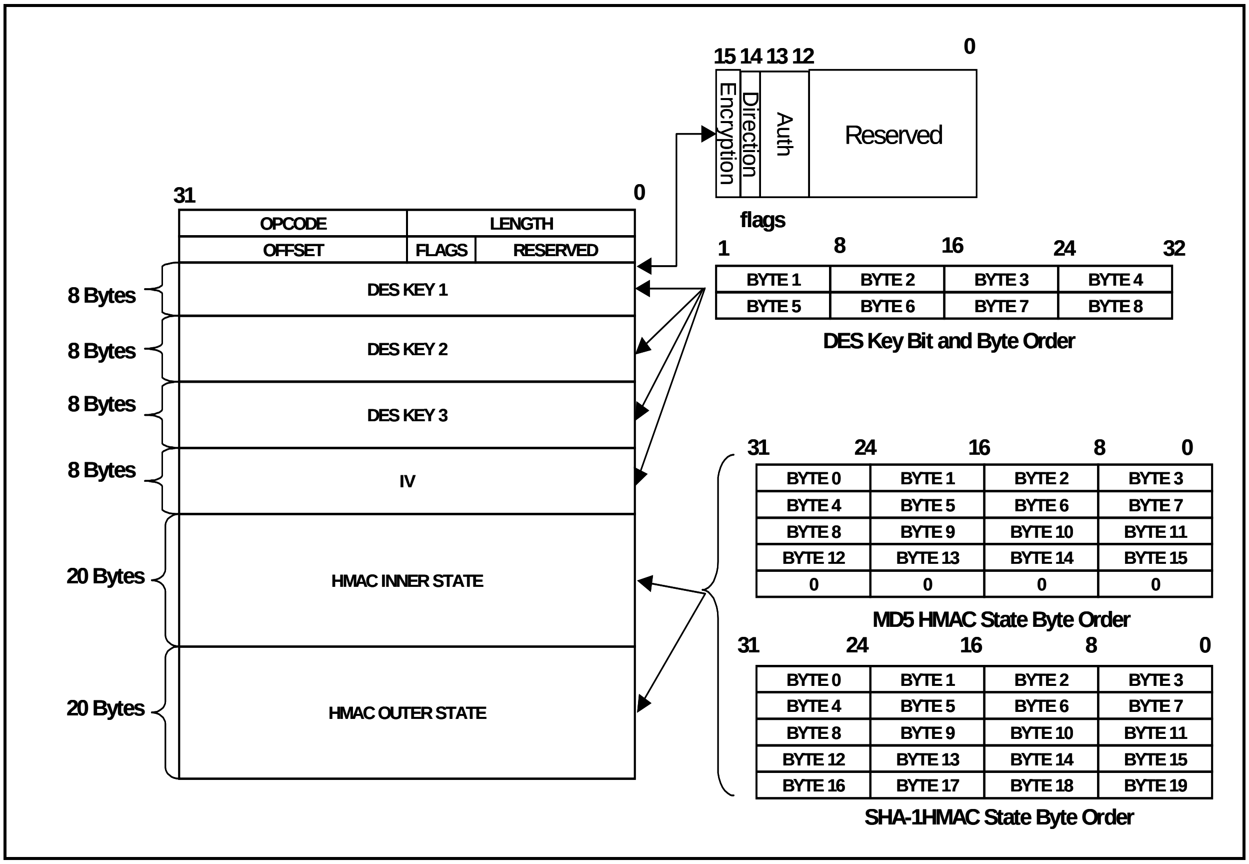

15 14 13 12
0
Encryption
Direction
Auth
Reserved
flags
31
0
OPCODE
LENGTH
OFFSET
FLAGS
RESERVED
8 Bytes
DES KEY 1
8 Bytes
DES KEY 2
8 Bytes
DES KEY 3
8 Bytes
IV
20 Bytes
HMAC INNER STATE
20 Bytes
HMAC OUTER STATE
1 8 16 24 32
BYTE 1 BYTE 2 BYTE 3 BYTE 4
BYTE 5 BYTE 6 BYTE 7 BYTE 8
DES Key Bit and Byte Order
31 24 16 8 0
BYTE 0 BYTE 1 BYTE 2 BYTE 3
BYTE 4 BYTE 5 BYTE 6 BYTE 7
BYTE 8 BYTE 9 BYTE 10 BYTE 11
BYTE 12 BYTE 13 BYTE 14 BYTE 15
0 0 0 0
MD5 HMAC State Byte Order
31 24 16 8 0
BYTE 0 BYTE 1 BYTE 2 BYTE 3
BYTE 4 BYTE 5 BYTE 6 BYTE 7
BYTE 8 BYTE 9 BYTE 10 BYTE 11
BYTE 12 BYTE 13 BYTE 14 BYTE 15
BYTE 16 BYTE 17 BYTE 18 BYTE 19
SHA-1HMAC State Byte Order

Figure 5: IPsec DES/3DES Command Context

Table 4: Flags

Bits	*Definition*
15	Encryption: • 0 = NULL • 1 = 3DES
14	Direction • 0 = Outbound, Encrypt then Authenticate • 1 = Inbound, Authenticate, then Decrypt
13:12	Authentication: • 00 = NULL • 01 = HMAC-MD5 • 10 = HMAC-SHA1 • 11 = Invalid
11:0	Reserved

Note

Authentication without encryption for AH or for ESP with the NULL encryption algorithm is supported, as is NULL authentication for confidentiality-only ESP. However, at least one of encryption or authentication must be defined.

NULL encryption with NULL authentication is specifically disallowed.

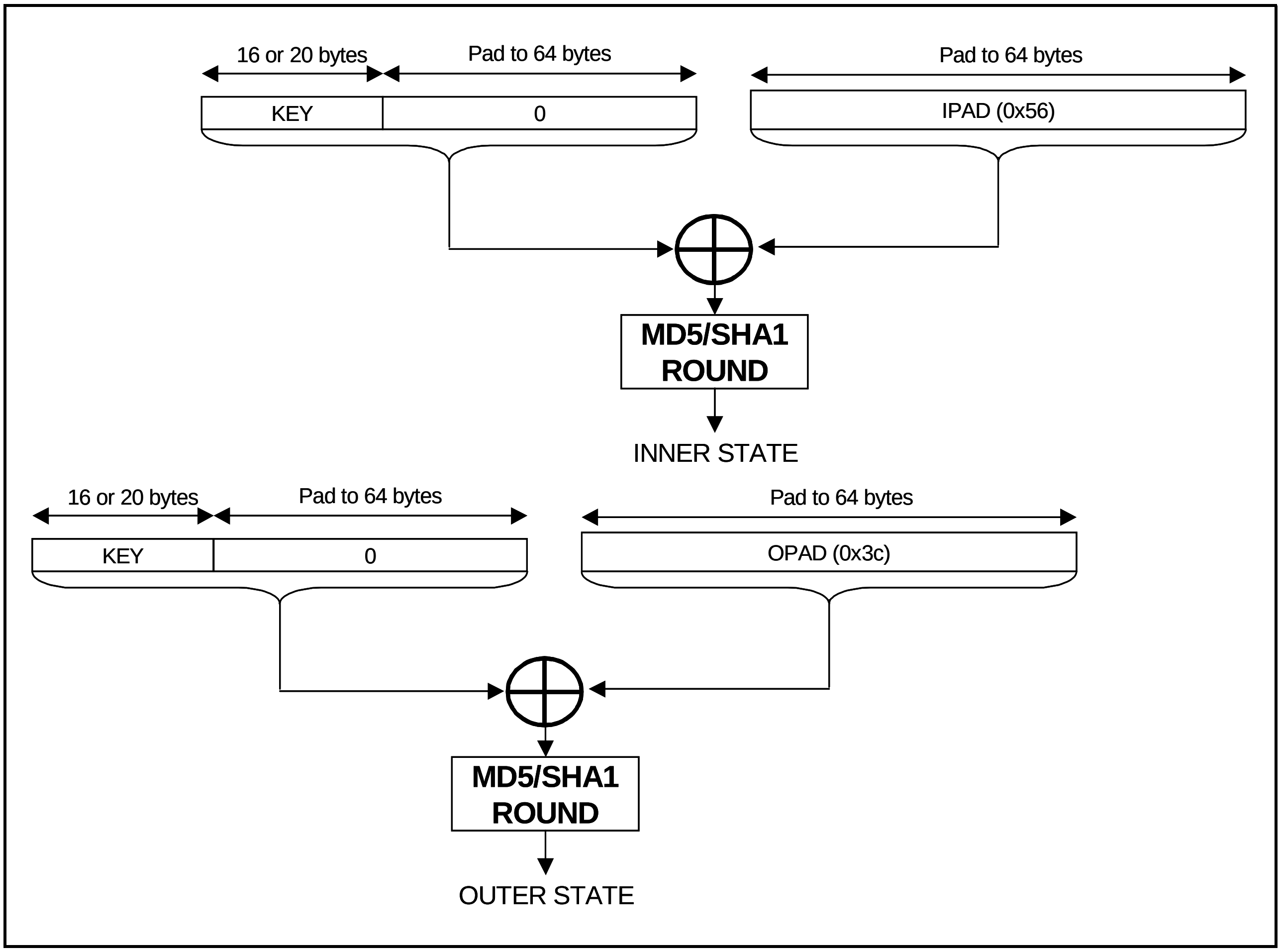


Figure 6: Computed HMAC Inner and Outer State

Note

The byte order for 3DES keys, IV, and HMAC inner and outer states may differ from the host CPU string byte order.

- **DES keys in FIPS-46-3 are numbered left to right, from 1 to 32. The first key byte is in the leftmost byte position of the first 32-bit word of the key.**
- **For example, the test key string <0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef> is represented in the Command Context as two 32-bit words, <0x01234567, 0x89abcdef>.**
- **Single DES uses the same command and structures, with all three keys the same.**
- **The offset value in the command context is in 32-bit words, not in bytes.**

Figure 8 on page 11 shows a packet description for combined IPsec ESP encryption and authentication. The total packet length is supplied in the packet descriptor. The offset is supplied in the command context in terms of 32-bit words (not bytes). The IV is also supplied in the command context, copied there from the packet by the host software.

Note **On some platforms, this may require each 32-bit word to be byte swapped.**

Figure 9 on page 12 shows an example of a logically contiguous input packet that is physically discontinuous in host memory, requiring the BCM5812 to gather three input fragments. Three input fragment chain entry descriptors are shown, organized as a linked list. Each fragment chain entry includes the address and length of the input fragment, along with a next input fragment chain entry pointer.

The first input fragment chain entry is supplied in the packet descriptor structure. Subsequent input fragment chain entries are read in by the BCM5812 as needed.

Note

- **The total of the lengths in the input fragment chain entries must match the packet length.**
- **Input fragment chain entries must be 32-bit aligned in host memory.**
- **Input fragment buffers can be byte aligned in host memory.**
- **The total length of the input for encryption or decryption must be a multiple of the cipher block size (8 bytes for 3DES or DES).**

The length of the data to be encrypted or decrypted is the packet length (from the packet descriptor) minus the offset in bytes (the offset value from the command context must be multiplied by 4).

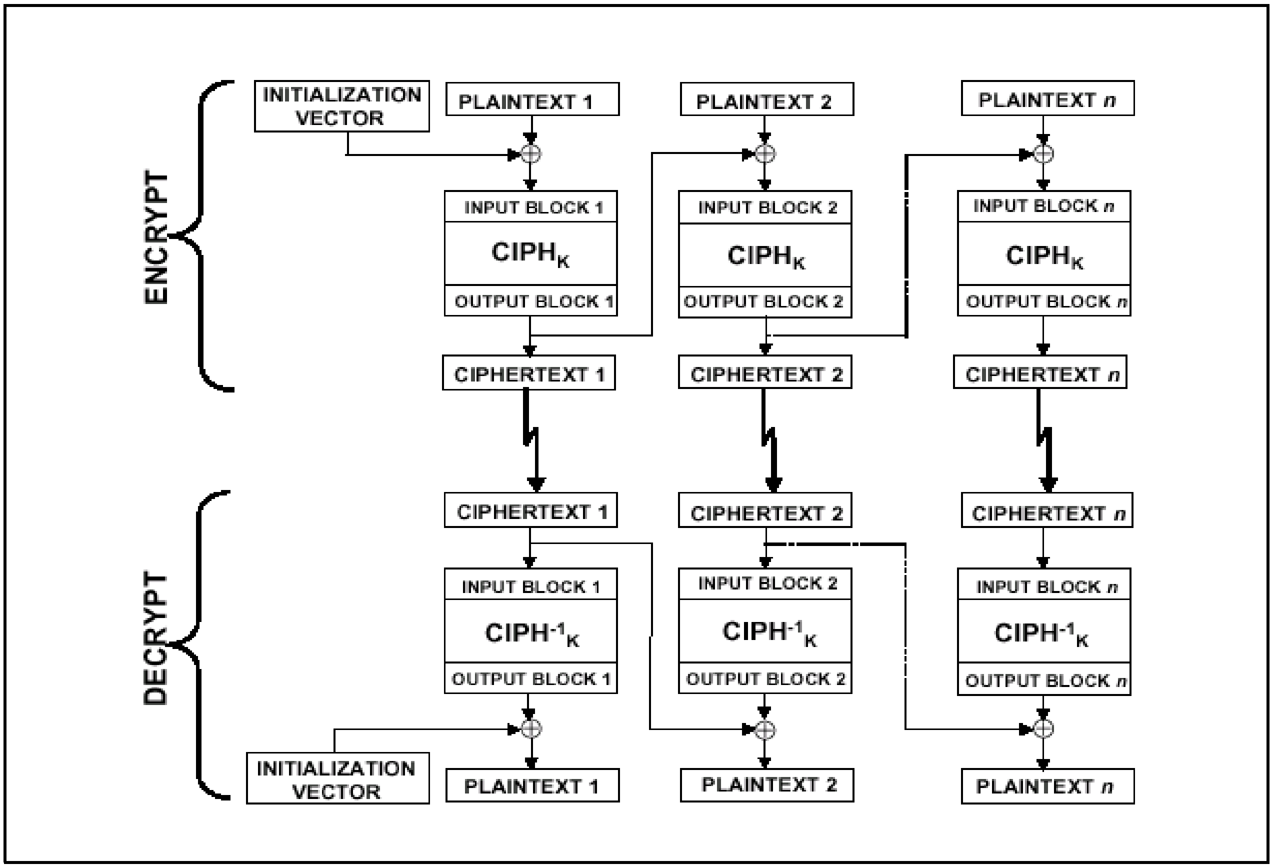


Figure 7: CBC Mode

If authentication is performed, it is done over the entire packet length. If encryption is performed (i.e., the direction in Table 4 on page 7 is outbound), the packet is encrypted beginning at the offset, and then authentication is performed over the entire packet length, including the ciphertext. If decryption is performed (i.e., the direction is inbound), authentication is performed over the entire packet length, and then the packet is decrypted beginning at the offset.

Figure 8 on page 11 also shows the position of the IV in an IP packet that is ESP-encapsulated according to RFC 2405. Figure 8 shows the byte order of the data as little-endian host CPU byte order. The BCM5812 can be configured to interpret these network bytes in big-endian host CPU byte order (See “Endian Considerations” on page 42). For 3DES, the initialization vector (IV) length is 8 bytes.

Figure 10 on page 12 shows the corresponding output buffer described using a list of output fragment chain entries.

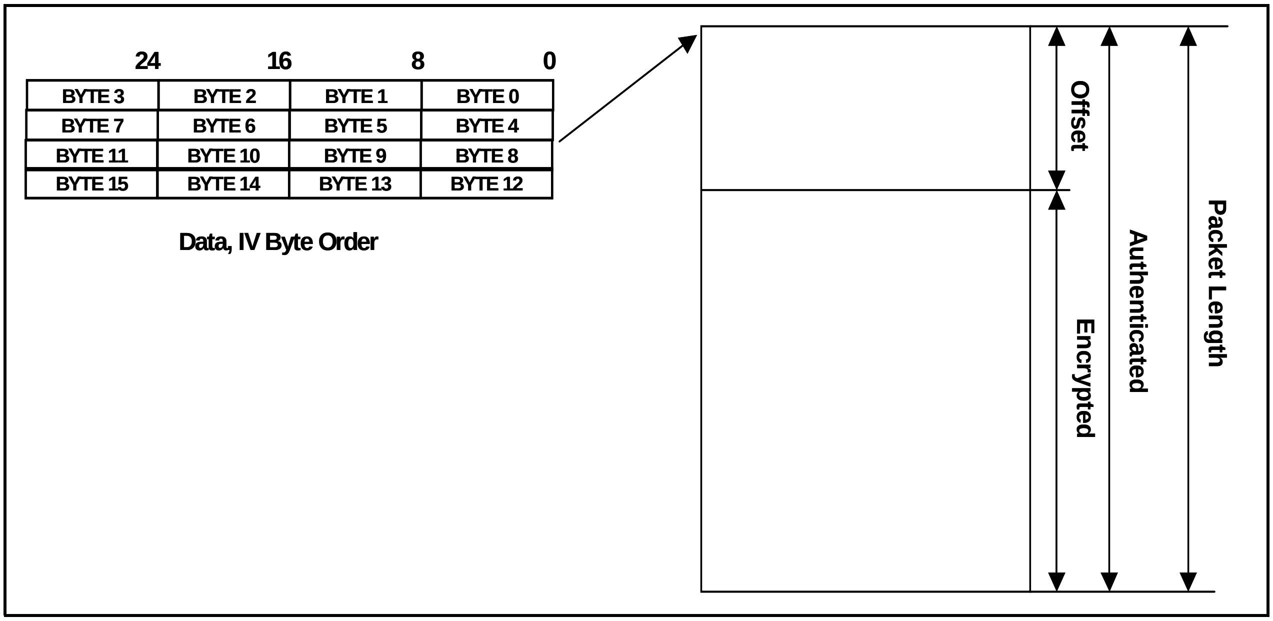
Note

- **Output fragment chain entries must be 32-bit aligned in host memory.**
- **Output fragment buffers must be 32-bit aligned in host memory.**
- **The total encrypted or decrypted length must be a multiple of the cipher block size (8 bytes for 3DES and DES)**

If encryption or decryption is performed, data is written into successive fragment buffers according to each output fragment chain entry’s output fragment length until the total of the encrypted or decrypted length (input packet length minus the offset) is satisfied.

The output byte order of encrypted or decrypted data is the same as that specified or configured for the input.

If authentication is performed, the HMAC value is written to the address specified in the next output fragment chain entry address of the last output fragment chain entry. The full HMAC hash output is always written, which for MD5 is16 bytes, and for SHA-1 is 20 bytes. That is, the returned authentication result is not truncated by the BCM5812 to 12 bytes as used by HMAC-MD5-96 (RFC 2405) or HMAC-SHA1-96 (RFC2404).



24	16	8	0
BYTE 3	BYTE 2	BYTE 1	BYTE 0
BYTE 7	BYTE 6	BYTE 5	BYTE 4
BYTE 11	BYTE 10	BYTE 9	BYTE 8
BYTE 15	BYTE 14	BYTE 13	BYTE 12

Figure 8: Packet Description for IPsec DES/3DES Encryption and Authentication

The output byte order for the HMAC is the same as that specified or configured for encrypted or decrypted data.

If authentication is performed without encryption or decryption, the host software must set the offset in the command context to zero. The BCM5812 writes the HMAC starting at the address given in the next output fragment chain entry in the packet descriptor structure. The output fragment address in the packet descriptor is ignored.

If encryption is performed without authentication, the host software should set the offset in the command context to zero for backward compatibility with the BCM582x. The BCM5812 ignores the next output fragment chain entry pointer in the last output fragment chain entry.

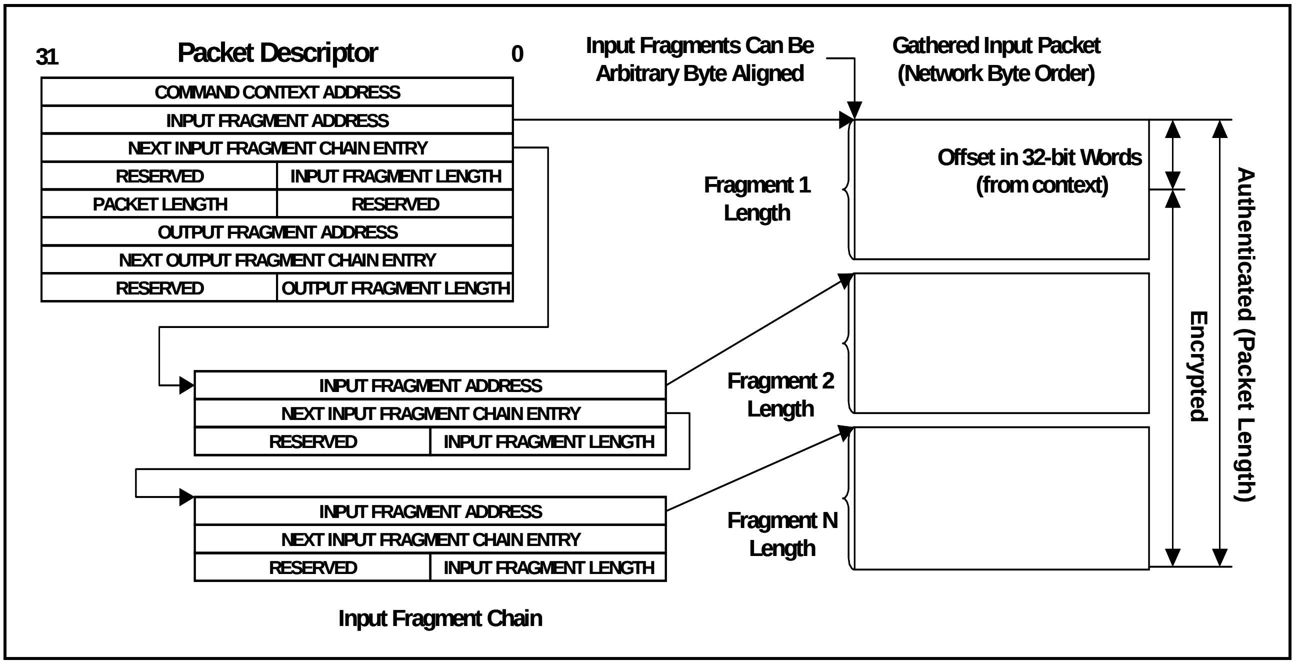


Figure 9: IPsec Packet Description Showing Input Fragments

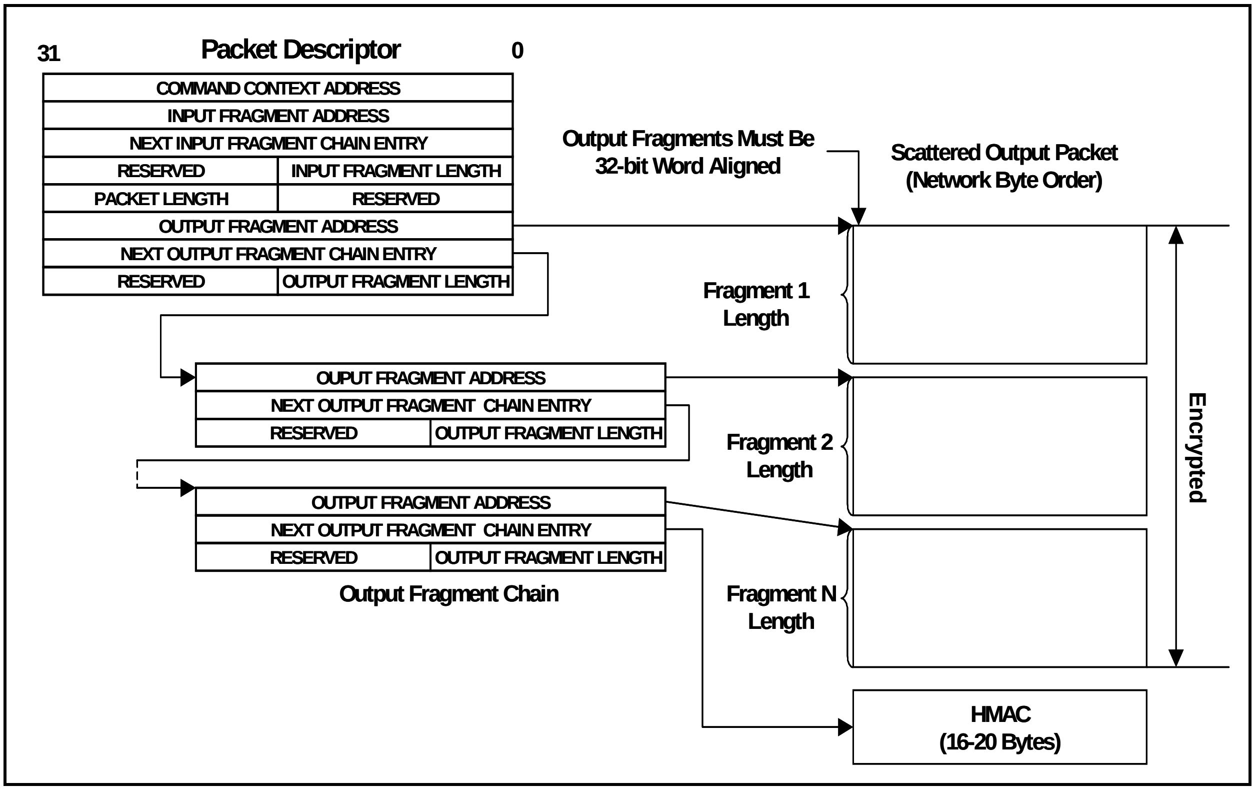


Figure 10: IPsec Packet Description Showing Output Fragments

SSL MAC

The BCM5812 computes the SSL-specific MAC function used in processing SSL protocol records. Figure 11 shows the SSL Record Structure, consisting of a five byte header, application data, and an authentication code computed using the SSL MAC. Normally, the application data and MAC are also encrypted. If a block cipher is used, 1 or more pad bytes would follow the MAC to ensure that the encrypted data is a multiple of the blocksize.

The SSL-specified MAC can be computed using either MD5 or SHA-1. Figure 12 diagrams the SSL MAC computation using MD5. A two level hash scheme is used. First, a buffer is constructed that begins with the MAC secret, which for MD5 is 16 bytes. The MAC secret is followed by 48-pad bytes containing 0x36, followed by a 64-bit record sequence number, followed by the content type from the record header. This is followed by a 16-bit length value, and then the application data payload from the record. The length value is the length of the application data proper, in bytes.

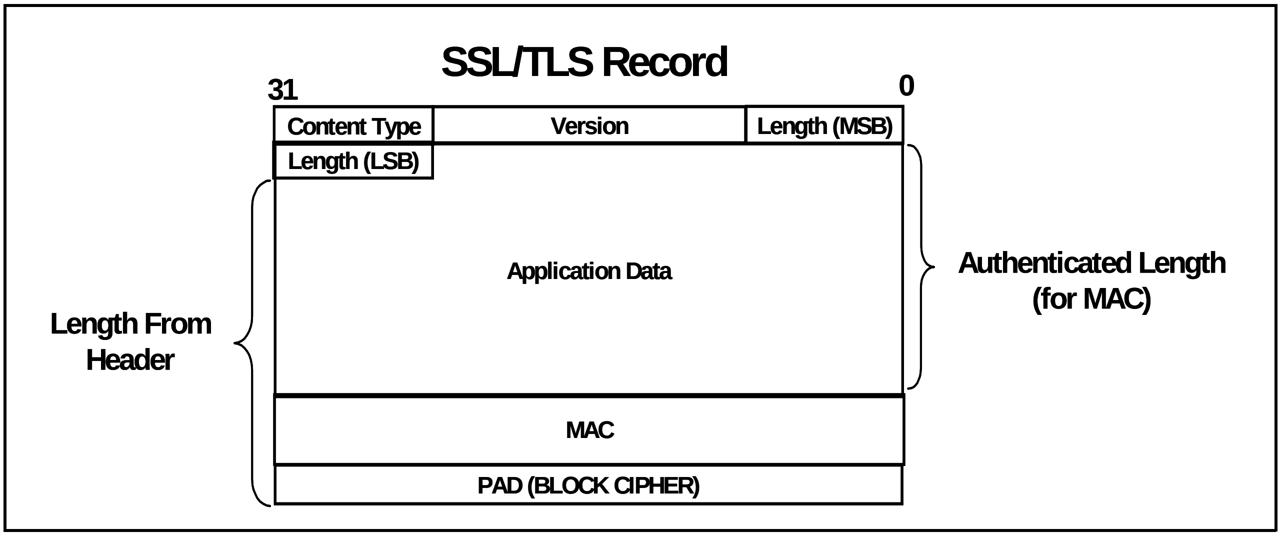


Figure 11: SSL Record Structure

This buffer is hashed using MD5. A second buffer is constructed using the MAC secret and a 48-byte pad field, this time using 0x5C as the pad value, followed by the 16 byte output from the first MD5 hash. This is then hashed using MD5, and the 16-byte output used as the SSL MAC.

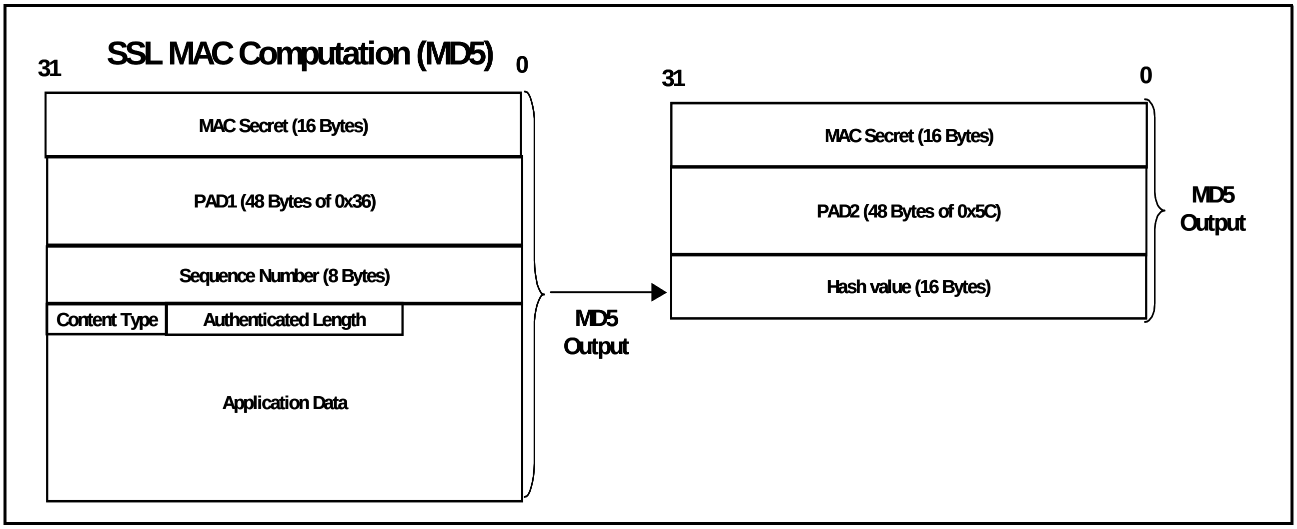


Figure 12: SSL MAC Computation Using MD5

The computation using SHA-1 differs only in the length of the MAC secret (20 bytes) and the number of pad bytes (40 rather than 48). Figure 13 diagrams the SSL MAC computation using SHA-1.

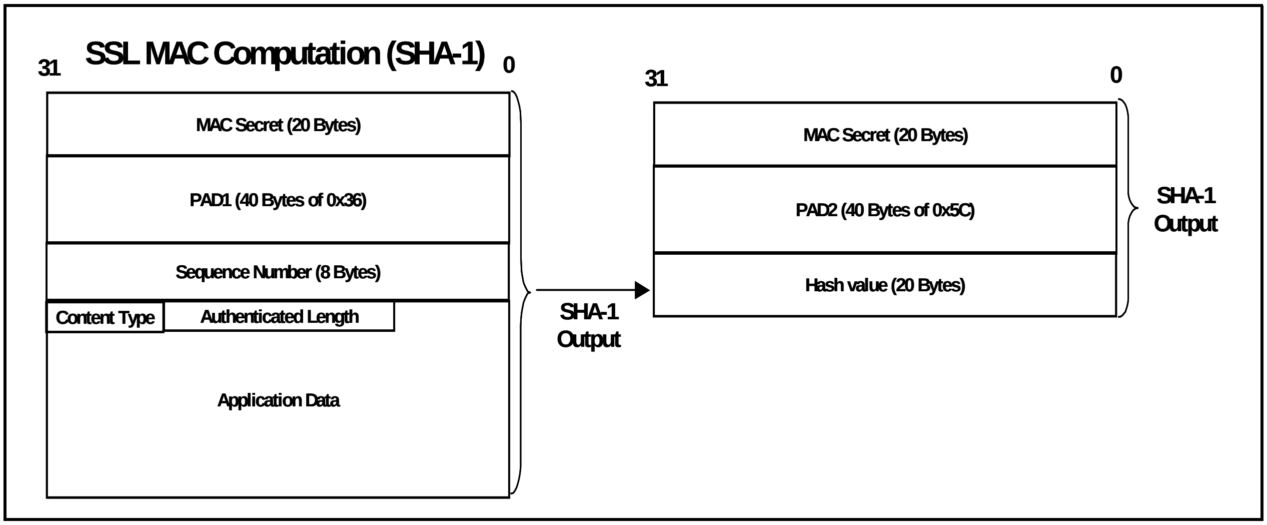


Figure 13: SSL MAC Computation Using SHA-1

Figure 14 on page 15 shows the command context used for computing the SSL MAC. Table 5 shows the codes for MD5 and SHA-1. The SSL MAC command context length is always 88 bytes.

This command uses MCR1@. The opcode for this command is 0x01.

Table 5: SSL MAC Authentication Codes

Bits	Definition
15:14	Reserved
13:12	Authentication: • 00 = Invalid • 01 = MD5 • 10 = SHA-1 • 11 = Invalid
11:0	Reserved

The command context includes the content type, sequence number, and authenticated length arguments used in computation of the SSL MAC. The byte ordering of the HMAC secret is the same as that used for the HMAC inner and outer states for IPsec.

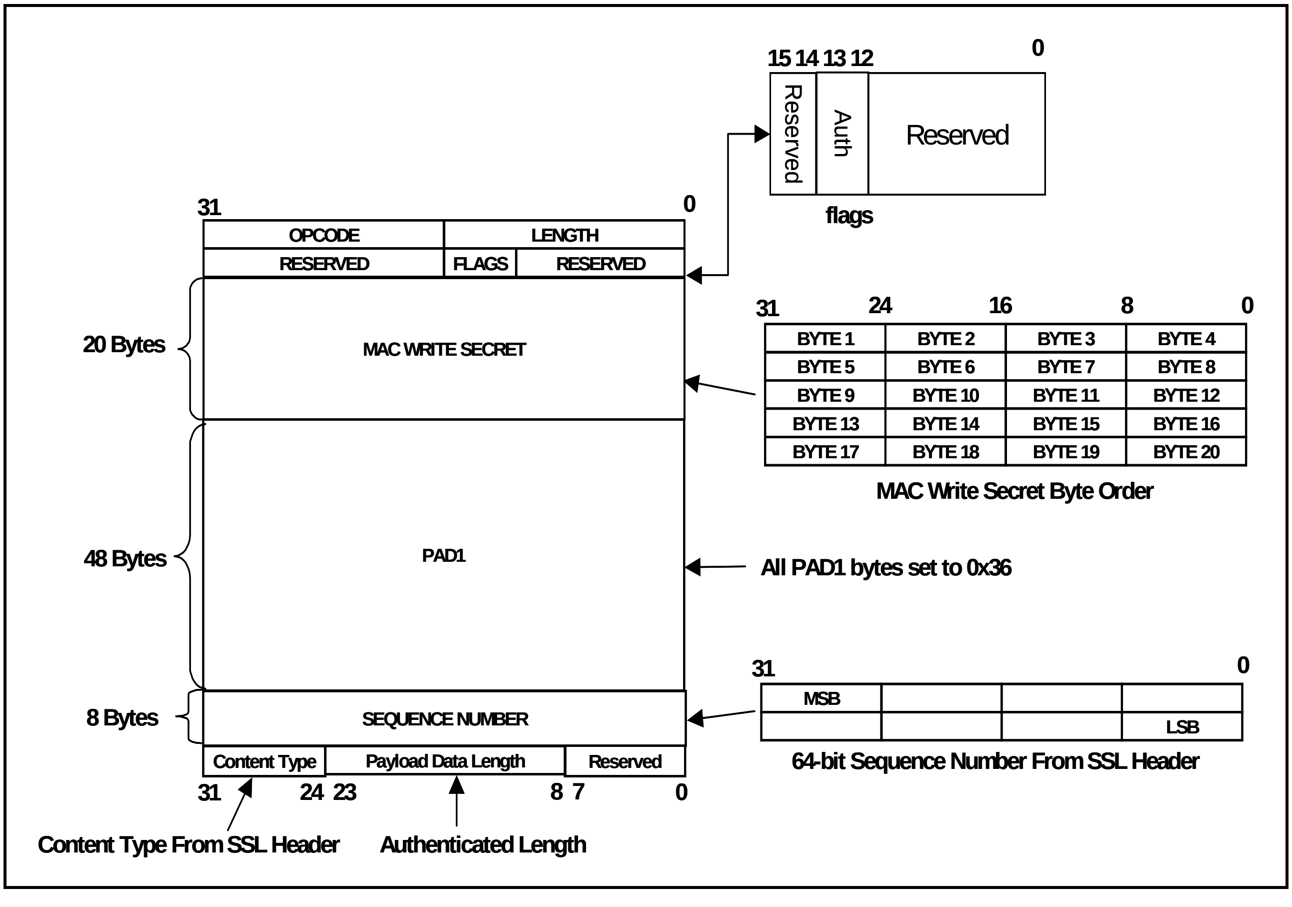


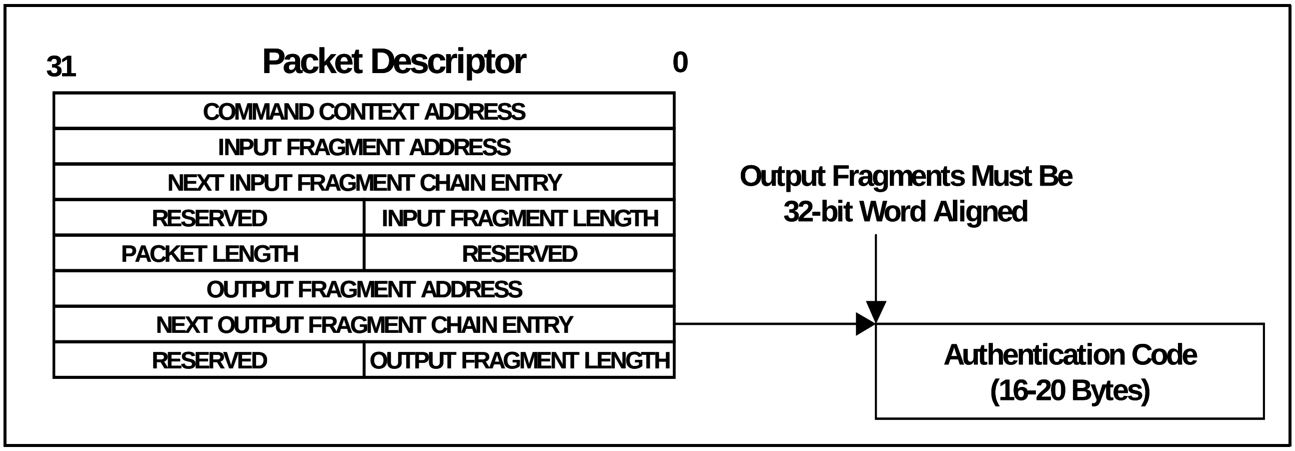
Figure 14: SSL MAC Command Context

The MAC write secret is used to compute the SSL MAC prior to outbound encryption. The MAC read secret would be used instead for authenticating inbound packets after decryption. The full 20 byte field is required. For MD5, the last four bytes must be zero. The full 48 bytes of PAD1 are also required, although only the first 40 are used for SHA-1. The sequence number is input as two 32-bit integer values, the first containing the most significant 32-bits of the 64-bit value.

One or more input fragment chain entry structures are used to describe the input record application data. The only output is the computed MAC, which is written to the address specified in the next output fragment chain entry pointer in the packet descriptor, as shown in Figure 15.

Note

- **The MAC output buffer must be 32-bit aligned in host memory.**
- **Input Fragment Buffers must be byte aligned in host memory.**
- **The total length of the Input Fragments must equal the Payload Data Length.**



Packet Descriptor

31	0
COMMAND CONTEXT ADDRESS	
INPUT FRAGMENT ADDRESS	
NEXT INPUT FRAGMENT CHAIN ENTRY	
RESERVED	INPUT FRAGMENT LENGTH
PACKET LENGTH	RESERVED
OUTPUT FRAGMENT ADDRESS	
NEXT OUTPUT FRAGMENT CHAIN ENTRY	
RESERVED	OUTPUT FRAGMENT LENGTH

Output Fragments Must Be 32-bit Word Aligned

Figure 15: SSL MAC Authentication Output

TLS HMAC

TLS, specified in RFC 2246 [18], is derived from, and in may respects very similar to, SSL. TLS updates the SSL MAC by using the standard HMAC (RFC2104) instead, and also includes the protocol major and minor version numbers in the authentication.

Figure 16 shows the input buffer for the TLS HMAC computation. The MAC secret is used to initialize the HMAC, and does not appear explicitly in the buffer. The major and minor version numbers are included following the content type and before the authenticated length.

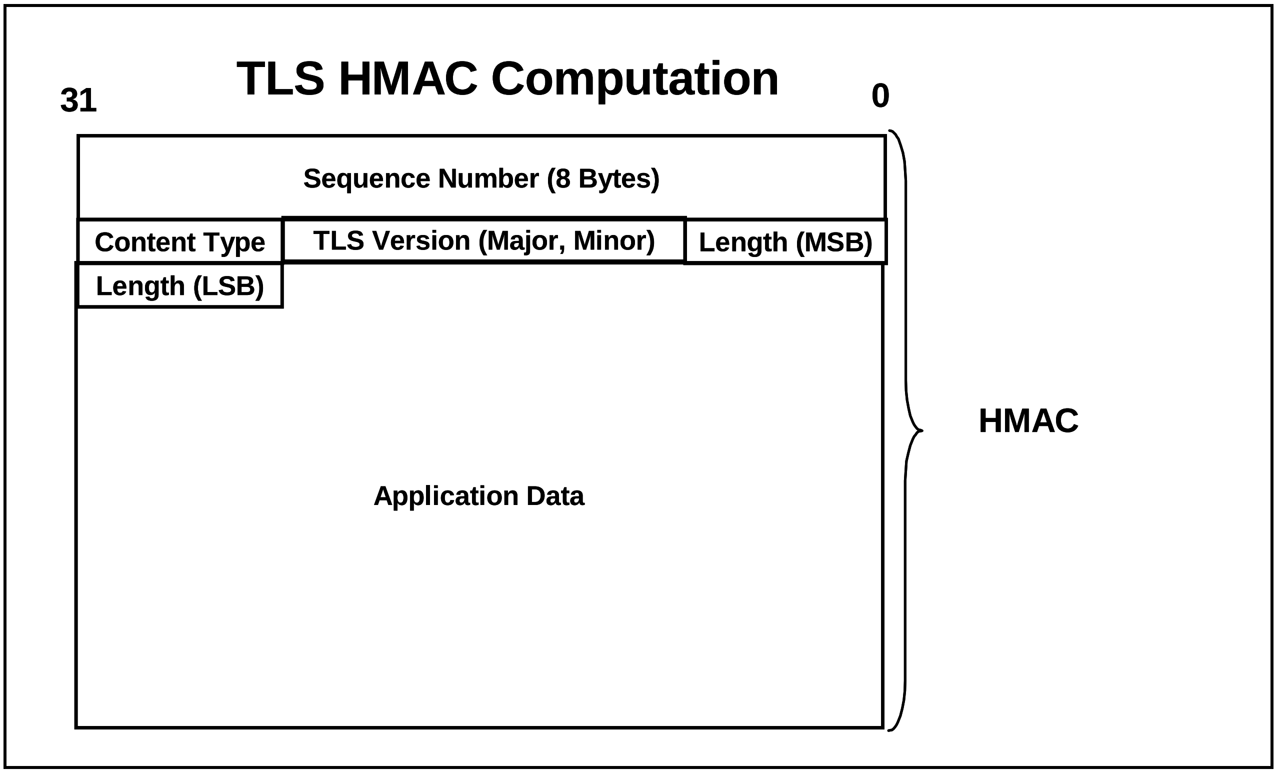


Figure 16: TLS HMAC Computation

Figure 17 diagrams the TLS HMAC command context, which is always 64 bytes in length. The basic hash algorithm, MD5 or SHA-1, is set using the authenticator bits in the flags word, and uses the same values as SSL (Table 5 on page 15). The opcode for this command, which uses MCR1@, is 0x02.

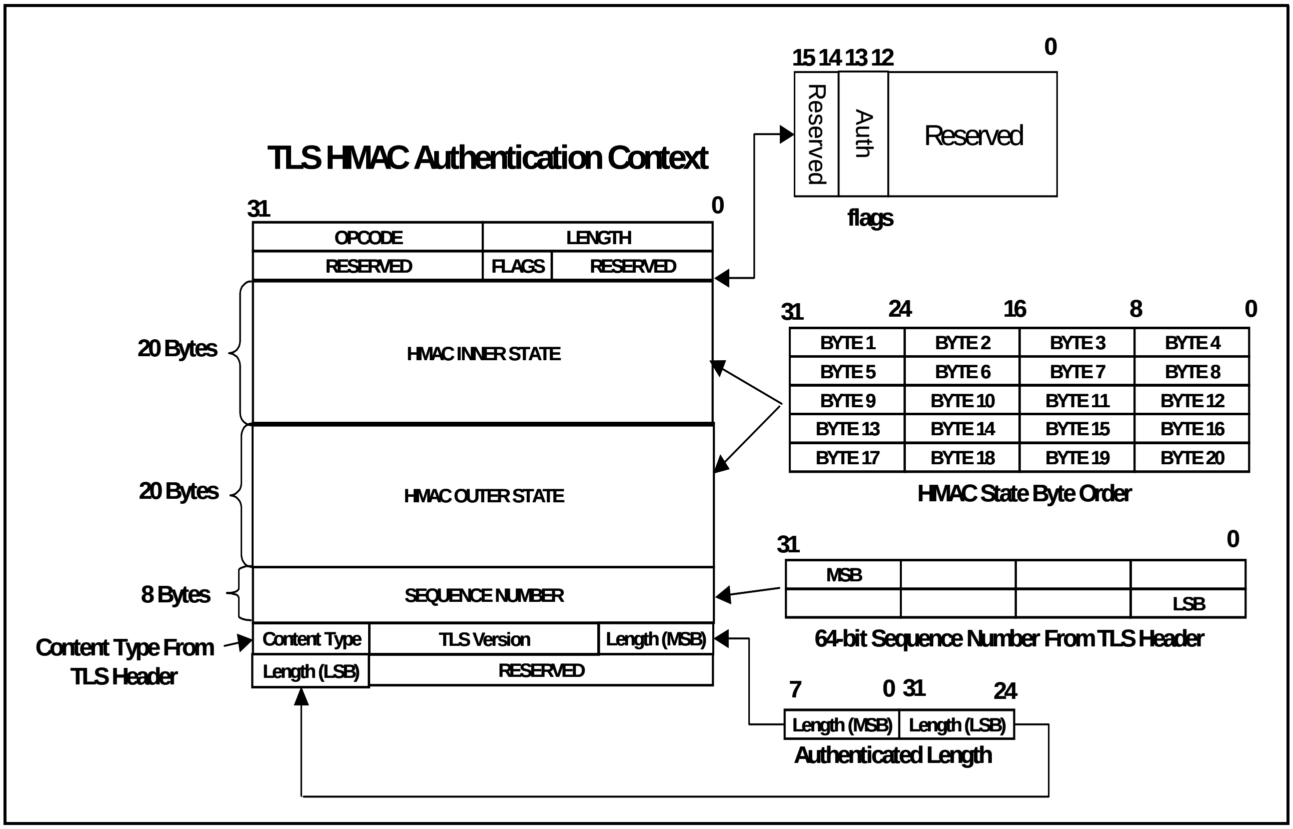


Figure 17: TLS HMAC Command Context

The HMAC inner and outer state are computed exactly the same as for IPsec (Figure 6 on page 8), and the byte ordering in the context is also the same. The TLS version consists of the major in the MSB, and the minor in the LSB.

The input and output fragment processing is the same as for SSL MAC, shown in Figure 15 on page 16.

SSL/TLS DES/3DES

The BCM5812 includes a pure DES/3DES operation for use with SSL and TLS. The command context and packet fragment processing is very similar to that for 3DES IPsec, except that authentication is not performed. Figure 18 shows the command context for SSL/TLS DES/3DES, which is always 64 bytes in length even though only 40 bytes are used. The only control flag, bit 14, indicates direction (zero indicates outbound, the same as in Table 4 on page 7). The reserved fields must be zero. The opcode for this command is 0x03. This command uses MCR1@.

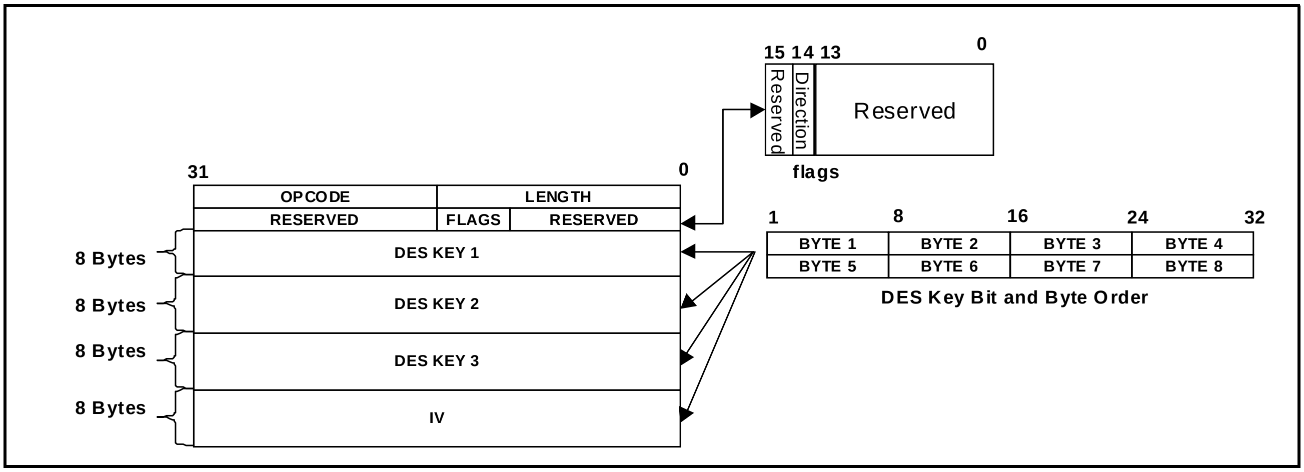


Figure 18: SSL/TLS DES/3DES Command Context

The same packet descriptor and fragment processing formats are used as for IPsec as well.

ARCFOUR

The BCM5812 provides ARCFOUR, a stream cipher that is compatible with the RSA Security RC4™ algorithm.

ARCFOUR is described in Applied Cryptography, Second Edition [17], in terms of pseudo-code using a 256 byte ARCFOUR state array and two variables, i and j.

1. Initialize the ARCFOUR state array `S` such that `S[n]=n`. Set variables `i` and `j` to 0,
2. Initialize another 256 byte array such that `K[n]` contains the n-th byte of the key. If the key contains fewer than 256 bytes, continue by putting the first key byte in the next `K` entry, and so forth, until `K` is filled.
3. Perform the following once, to set up the ARCFOUR state using the key, where swap exchanges the byte values for the specified pair of elements:

```
for (i=0; i<256; i++){
j = (j+S[i]+K[i]) & 0xff;
swap(S[i],S[j]);
}
```

4. Use the ARCFOUR state array to generate the next keystream byte, `O`:

```
i = (i+1) & 0xff;
j = (j+S[i]) & 0xff;
swap(S[i],S[j]);
O = S[(S[i]+S[j]) & 0xff];
```

5. Exclusive-or `O` with input data to encipher the next byte in the cleartext stream, or to decipher the next byte in the ciphertext stream.

Figure 19 shows the Command Context used by the BCM5812 for ARCFOUR. The BCM5812 provides sub-operations for combining the initial ARCFOUR state key step along with encipherment, or just encipherment using a continued ARCFOUR state. Options control whether the BCM5812 writes back the ARCFOUR state or generates keystream only, without input data. These options are described in Table 6 on page 20. The command context length is always 268 bytes.

This command uses MCR1@. The Opcode for this command is 0x04.

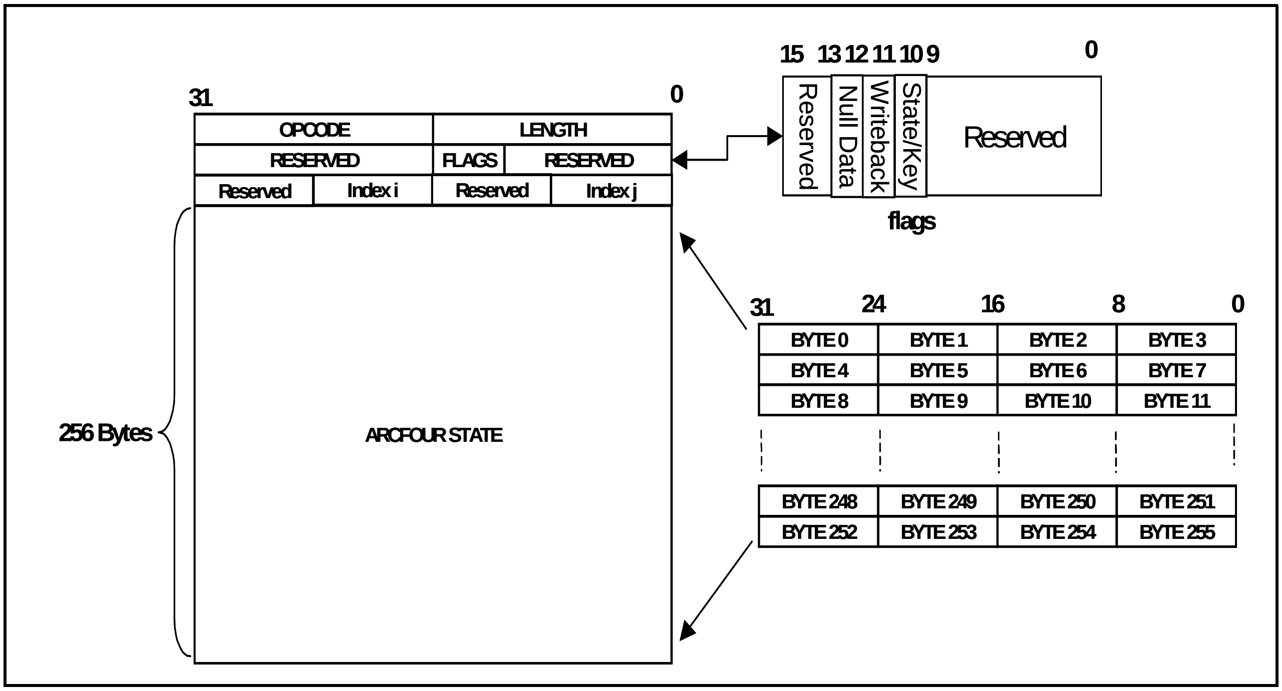


Figure 19: ARCFOUR Command Context

Table 6: ARCFOUR Command Context Flags

Bits	*Definition*
15:13	Reserved
12	Null Data: • 0 = Normal data encipherment • 1 = No input data, just output raw keystream
11	Writeback: • 0 = Don't write back state, used for last encipherment operation in a keystream. • 1 = Write back internal state after encipherment.
10	State/Key: • 0 = Context contains continuation ARCFOUR state for keystream • 1 = Context contains ARCFOUR key input, perform ARCFOUR state initialization first.
9:0	Reserved

Figure 20 shows how the input packet fragment list is used. Figure 21 on page 22 shows the corresponding output packet fragment list usage.

If indicated by the command context flags, the ARCFOUR state at the end of this keystream generation or encipherment operation is written to a fixed-size, 260-byte buffer starting at the output fragment address in the last output fragment chain entry. This can be input directly in the command context of a subsequent ARCFOUR operation to continue the keystream.

In null data mode, the input fragment chain entry in the packet descriptor and any chained elements are ignored. Only the output fragment chained entry values are used. The number of keystream bytes generated is the packet length, which must equal the total of the output fragment lengths.

For normal encipherment, the packet length, total of the input fragment lengths, and total of the output fragment lengths must be equal.

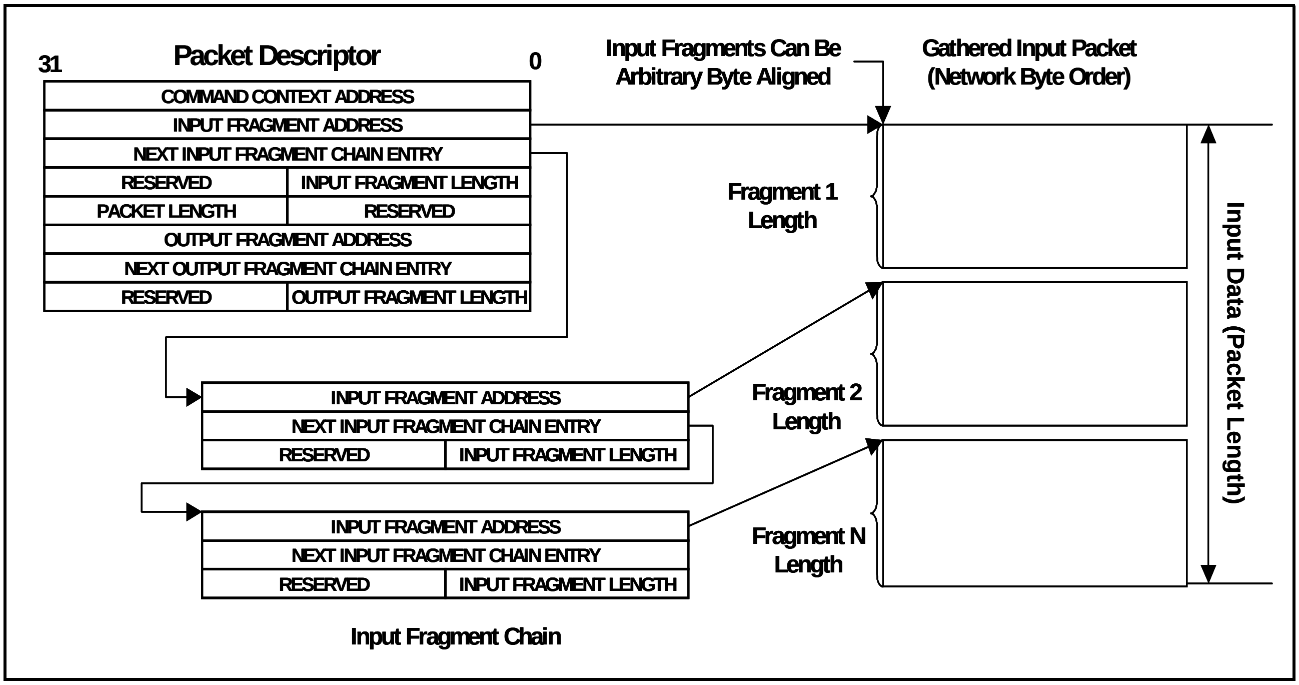


Figure 20: ARCFOUR Packet Description Showing Input Fragments

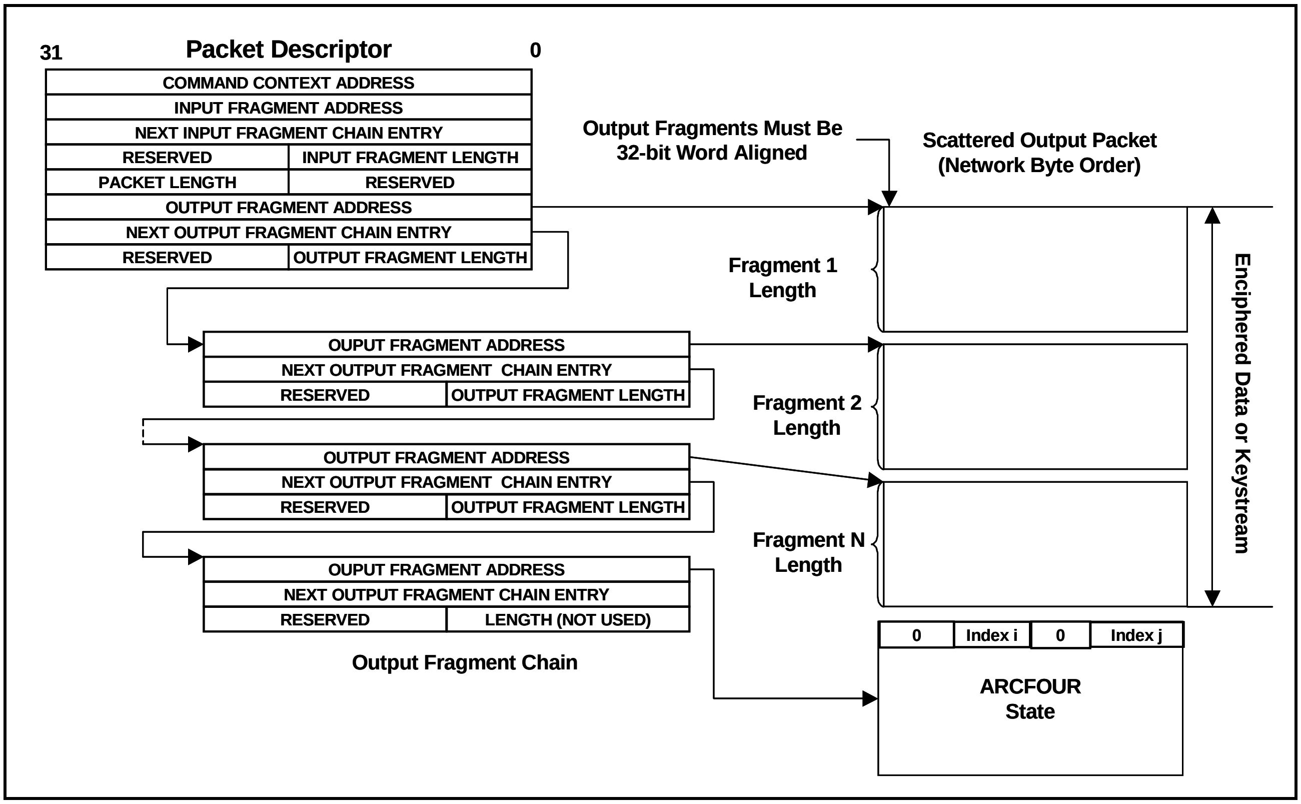


Figure 21: ARCFOUR Packet Description Showing Output Fragments

Note

- **The default byte order for ARCFOUR input and output fragment data is the same as the byte order on a little endian host processor system.**
- **The default ARCFOUR state in the Command Context is byte swapped relative to the input and output fragment data.**
- **The ARCFOUR continuation state is output in the same byte order as the Command Context.**

PURE MD5/SHA-1 HASH

The BCM5812 includes basic MD5 and SHA-1 hash operations. These are useful for various protocol processing functions, such as for computing SSL and TLS finished messages.

Figure 22 shows the command context for Pure MD5/SHA-1 Hash. The flags word authentication value is shown in Table 7. This command context length should be programmed to 8, even though the BCM5812 always reads 64 bytes.

This command uses MCR1@. The opcode for this command is 0x05.

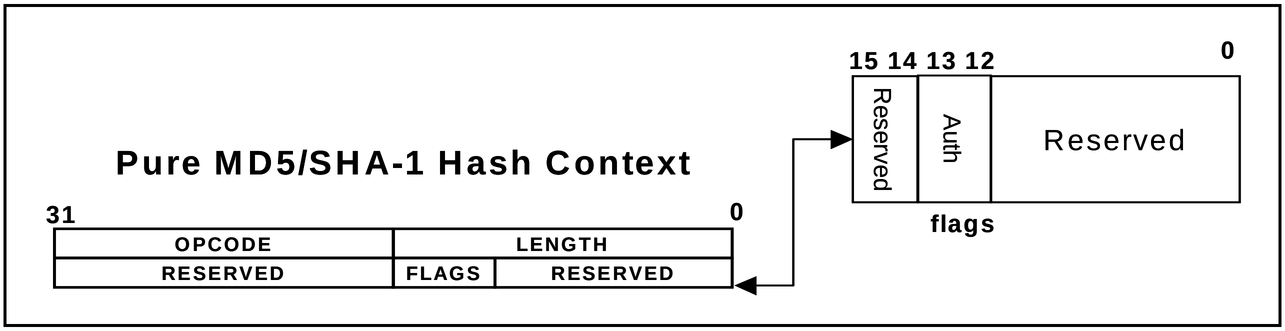


Figure 22: Pure MD5/SHA-1 Hash Command Context

The input structures are the same as for ARCFOUR, Figure 20 on page 21. The packet length must equal the total of the input fragment lengths.

Table 7: Pure MD5/SHA-1 Hash Command Context Flags

Bits	***Definition***
15:14	Reserved
13:12	Authentication: • 00 = Invalid • 01 = MD5 • 10 = SHA-1 • 11 = Invalid
11:0	Reserved

The hash output is written to the address in the next output fragment chained entry in the packet descriptor. The size of the hash written is determined by the algorithm, 16 bytes for MD5 and 20 bytes for SHA-1. The output fragment length for this descriptor must be zero.

IPSEC AES

The BCM5812 provides FIPS-197 [15] compliant AES. It supports the 128 bit blocksize, with key sizes of 128, 192, and 256 bits.

Figure 23 shows the AES command context. AES processing is very similarly to 3DES, with the following differences:

- the block size is always 128 bits, as is the IV
- the key can be 128, 192, or 256 bits
- counter mode (CTR) is supported in addition to CBC.

The AES command context flags are detailed in Table 9 on page 24. The command context size depends on the AES key size, as shown in Table 8. This command uses MCR1@. The opcode for this command is 0x40.

Table 8: AES Command Context Sizes by Key Size

Key Size	*Command Context Length*
128	80
192	88
256	96

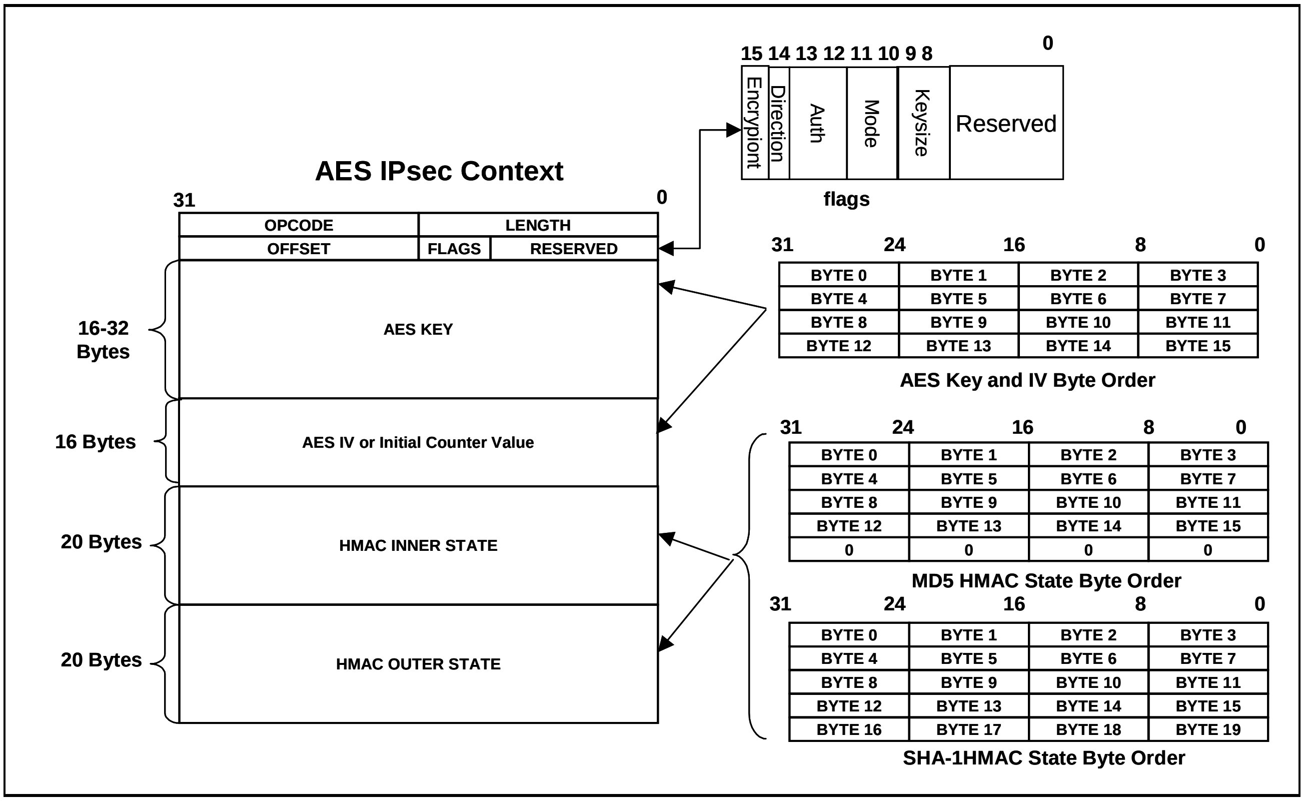

AES IPsec Context
31
0
OPCODE
LENGTH
OFFSET
FLAGS
RESERVED
16-32 Bytes
AES KEY
16 Bytes
AES IV or Initial Counter Value
20 Bytes
HMAC INNER STATE
20 Bytes
HMAC OUTER STATE
15 14 13 12 11 10 9 8
0
Encrypiont
Direction
Auth
Mode
Keysize
Reserved
flags
31 24 16 8 0
BYTE 0 BYTE 1 BYTE 2 BYTE 3
BYTE 4 BYTE 5 BYTE 6 BYTE 7
BYTE 8 BYTE 9 BYTE 10 BYTE 11
BYTE 12 BYTE 13 BYTE 14 BYTE 15
AES Key and IV Byte Order
31 24 16 8 0
BYTE 0 BYTE 1 BYTE 2 BYTE 3
BYTE 4 BYTE 5 BYTE 6 BYTE 7
BYTE 8 BYTE 9 BYTE 10 BYTE 11
BYTE 12 BYTE 13 BYTE 14 BYTE 15
0 0 0 0
MD5 HMAC State Byte Order
31 24 16 8 0
BYTE 0 BYTE 1 BYTE 2 BYTE 3
BYTE 4 BYTE 5 BYTE 6 BYTE 7
BYTE 8 BYTE 9 BYTE 10 BYTE 11
BYTE 12 BYTE 13 BYTE 14 BYTE 15
BYTE 16 BYTE 17 BYTE 18 BYTE 19
SHA-1HMAC State Byte Order

Figure 23: IPsec AES Command Context

The data input and output fragment formats are identical to 3DES (Figure 9 on page 12 and Figure 10 on page 12), except that the encrypted part must be a multiple of the 128 bit blocksize. The AES IV is also 128 bits, and taken from the command context. For counter mode, the 128-bit IV field is used for the initial counter value.

Table 9: AES Command Context Flags

Bits	*Definition*
15	Encryption: • 0 = NULL • 1 = AES

Table 9: AES Command Context Flags

Bits	***Definition***
14	Direction: • 0 = Outbound • 1 = Inbound
13:12	Authentication: • 00 = NULL • 01 = HMAC-MD5 • 10 = HMAC-SHA1 • 11 = Reserved
11:10	Mode: • 00 = CBC • 01 = CTR • 10 = Reserved • 11 = Reserved
9:8	Keysize: • 00 = 128 bits • 01 = 192 bits • 10 = 256 bits • 11 = Reserved
7:0	Reserved

Note

- **The total length of the input for encryption or decryption must be a multiple of the cipher block size (16 bytes for AES).**

Counter mode (CTR) uses a running counter to generate a keystream which is exclusive-ored with the input data, as opposed to Cipher Block Chaining (CBC). CTR is illustrated in Figure 24 on page 26, which is taken from NIST Special Publication 800-38a [16].

Counter mode is used in a number of protocols, including for wireless security and for streaming media.

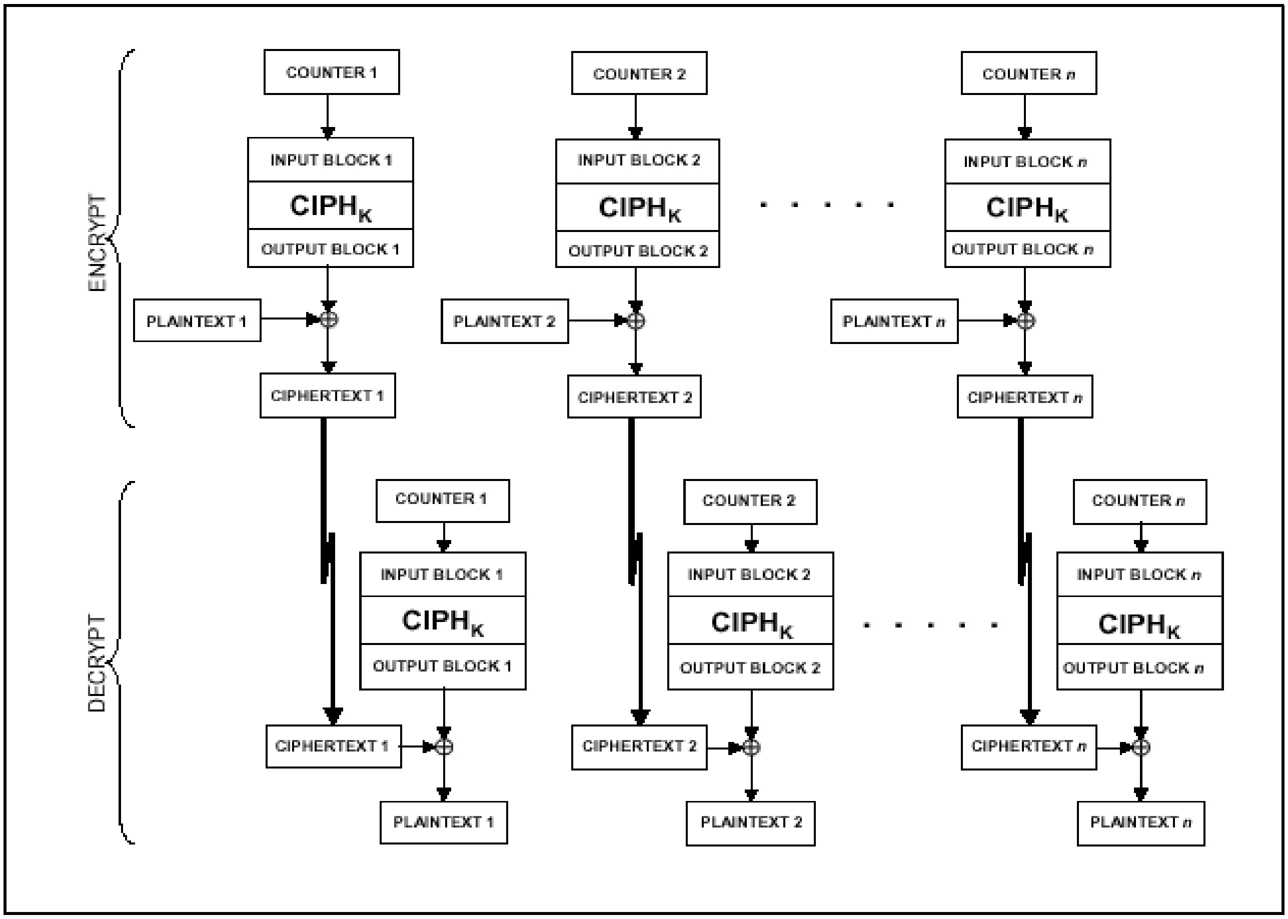


Figure 24: Counter Mode

In counter mode, the BCM5812 takes the initial count value from the IV field. Each subsequent block is incremented by one in the least significant bit position.

DIFFIE-HELLMAN

The Diffie-Hellman public key algorithm is used for key agreement in a number of protocols, including IKE, SSL, and TLS. It is based on the difficulty of calculating discrete logarithms in a finite field, and typically involves the following steps [17]:

1 Alice and Bob agree on parameters for a group, defined in terms of a large prime base, N, and a generator, g. The generator is such that each x less than N generates a different value ($y = g^x \bmod N$).

2 Alice chooses a secret random number, xa and sends Bob, $y_A = g^{xa} \bmod N$ over a public channel.

3 Bob does the same thing, choosing his secret number xb, and sends Alice $y_B = g^{xb} \bmod N$.

4 Alice exponentiates Bob's public number and computes ($y_b^{xa} \bmod N$), which equals $(g^{xa})^{xb} \bmod N$

5 Bob similarly computes ($y_a^{xb} \bmod N$), which equals $(g^{xb})^{xa} \bmod N$, the shared secret.

The BCM5812 provides separate Diffie-Hellman operation codes for generating the public key and for generating the secret key. Figure 25 on page 28 shows the command context, packet descriptor structure, input, and output data for Diffie-Hellman public key generate. This command uses opcode 0x01, and MCR2@.

The modulus and generator lengths can be between 16 and 2048 bits, and are specified in bits in the command context. The modulus and generator fields in the command context must take on one of the parameter field sizes in Table 10.

Note **The BCM5812 interprets all large integer arguments as “BigNums,” arrays of 32-bit “digits” ordered least significant digit first. That is, the digit containing the least significant bit of the large integer is at the lowest ordered index in the array.**

The exponent length is specified in bits in the command context. The BCM5812 can use its internal random number generator to generate a new secret value of length exponent length if generate secret is equal to 0x0001. If generate secret is equal to 0x0000, the host software must supply an exponent length input secret, as a BigNum, using the appropriate parameter size increment. Generate secret should be one of these two values.

Table 10: Allowed Public Key Parameter Field Size Increments

Modulus Length, in Bits	*Parameter Size in 32-bit words*	*Parameter SIze in Bytes*
16-512	16	64
513-768	24	96
769-1024	32	128
1025-1536	48	192
1537-2048	64	256

Figure 25 illustrates the layout of a BigNum argument. A modulus or generator that is less than the parameter field size must start with its least significant bit in the first 32-bit word, and be padded with zeros to the parameter field size in its high order bits.

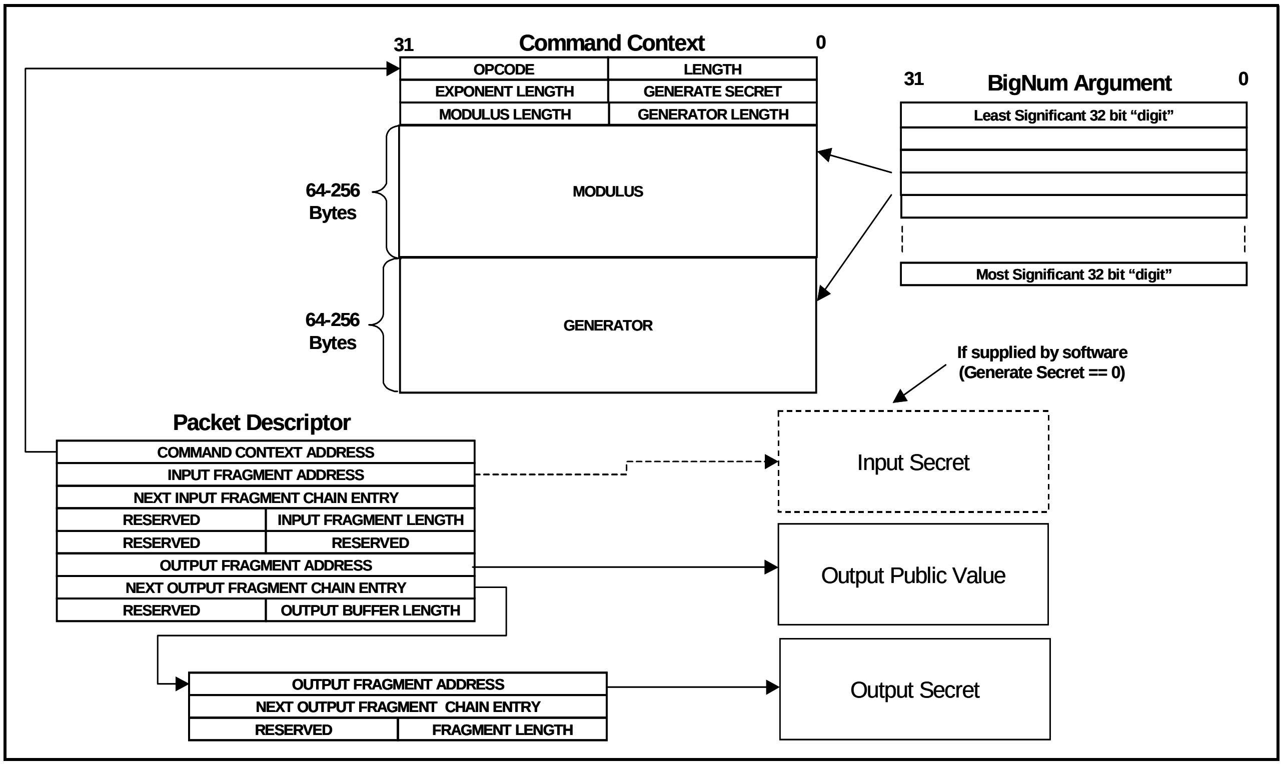


Figure 25: Diffie-Hellman Public Key Generate

The generator and modulus parameter field sizes must match their respective number of bits, and the same parameter field size must be used for both.

The BCM5812 outputs the public value as a BigNum to the first output fragment. This buffer must be the same length in bytes as the modulus parameter field size increment in Table 10 on page 27. It must be 32-bit aligned and contiguous. Output fragmentation is not used.

The secret value is always written to the second output fragment chain entry buffer address, whether supplied by software or generated by the BCM5812. This buffer must be the same length in bytes as the exponent parameter field size increment in Table 10. It must be 32-bit aligned and contiguous. Output fragmentation is not used.

Figure 26 on page 29 shows the command context, packet descriptor structure, input, and output for Diffie-Hellman secret key derivation. This uses MCR2@, and Opcode 0x02.

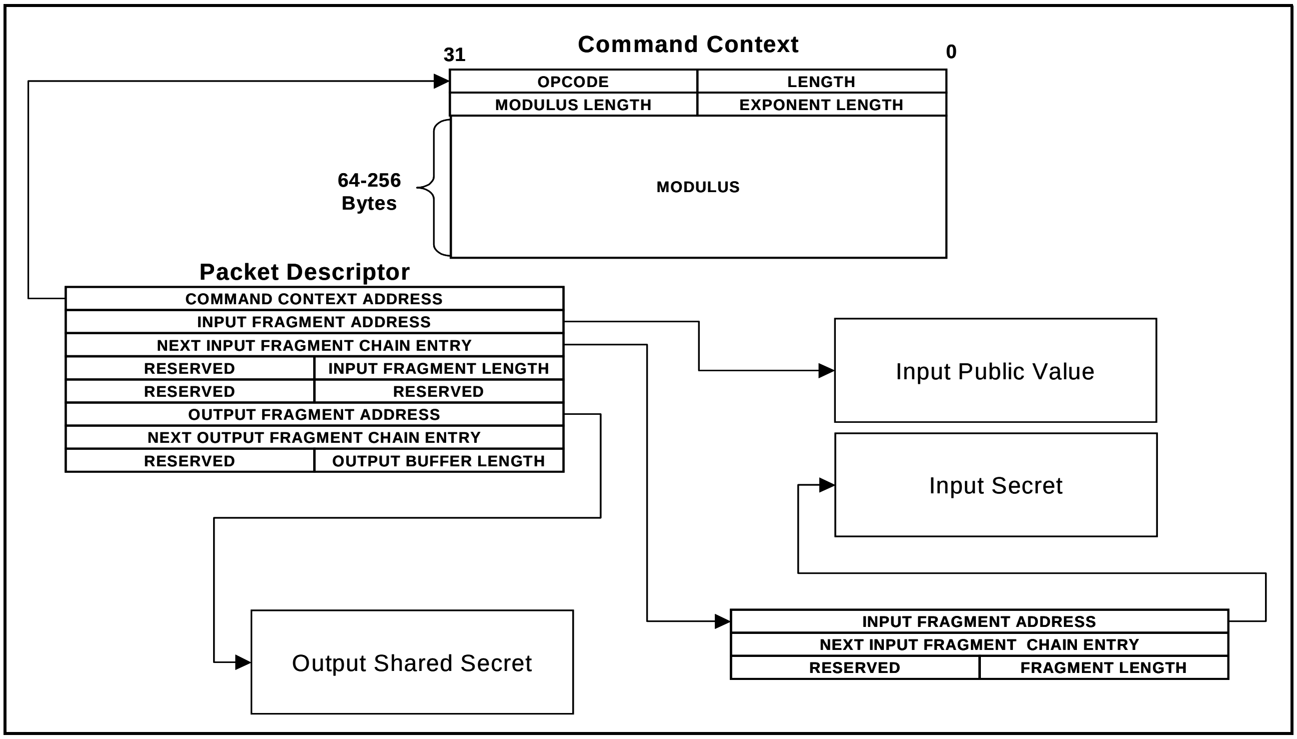


Figure 26: Diffie-Hellman Secret Key

The modulus must be input in the command context as a BigNum in the same manner as for Diffie-Hellman generate. The first Input fragment chain entry buffer address specifies the input secret, and the second input fragment chain entry buffer contains the peer entity's public value. The shared secret is output to the output fragment address in the packet descriptor structure. All of these are BigNum values.

All output buffers must be sized for the appropriate parameter argument parameter field size in Table 10.

The lengths of the Diffie-Hellman generate and shared secret command contexts depend upon the parameter field sizes, and range from 82 to 524 bytes.

RSA

The RSA public key algorithm is used for digital signature authentication and key exchange in IKE, SSL, and TLS. It also widely used in a number of other applications, such as Public Key Infrastructure (PKI) products. RSA is a two-key system, with a public key that can be used to either encrypt a value so that only the holder of the private key can decrypt it, or to decrypt a value that only the holder of the private key could have encrypted [17][19].

RSA uses the product of two large primes, p and q, which must remain secret. The public key consists of n = p * q, and a public exponent e that is relatively prime to (p-1)*(q-1). The public key operation encrypts a message m by exponentiating it modulo n, so that the encrypted value $x = m^e$ mod n. The private key consists of the modulus and a decrypting exponent, d, that is the inverse of e, $d = e^{-1}$ mod (p-1)(q-1). Knowing p and q, it is straight forward to compute d, but otherwise quite difficult. Decryption is simply $m = x^d$ mod n.

Generally, the public exponent e can be chosen to be short, but the private exponent is derived and should be nearly as long as the modulus. The effort to exponentiate is proportional to the logarithm of the exponent; therefore, the private key operation takes considerably longer than the public key operation. This can be speeded up by taking advantage of the knowledge of p and q, and doing exponential using residue arithmetic according to the Chinese Remainder Theorem (CRT). In effect, this breaks d into two components, d_p = d mod (p-1) and d_q = d mod (q-1), does two, half-sized exponentiations modulo p and modulo q, and combining the result using an inverse constant (q^{-1} mod p).

The BCM5812 provides both RSA public key and CRT private key operations. Figure 27 shows the command context for the RSA public key operation. As with Diffie-Hellman, the modulus and exponent are specified in bits, with parameter sizes in the command context, input, and output according to Table 10 on page 27. This operation uses MCR2@, and opcode 0x03.

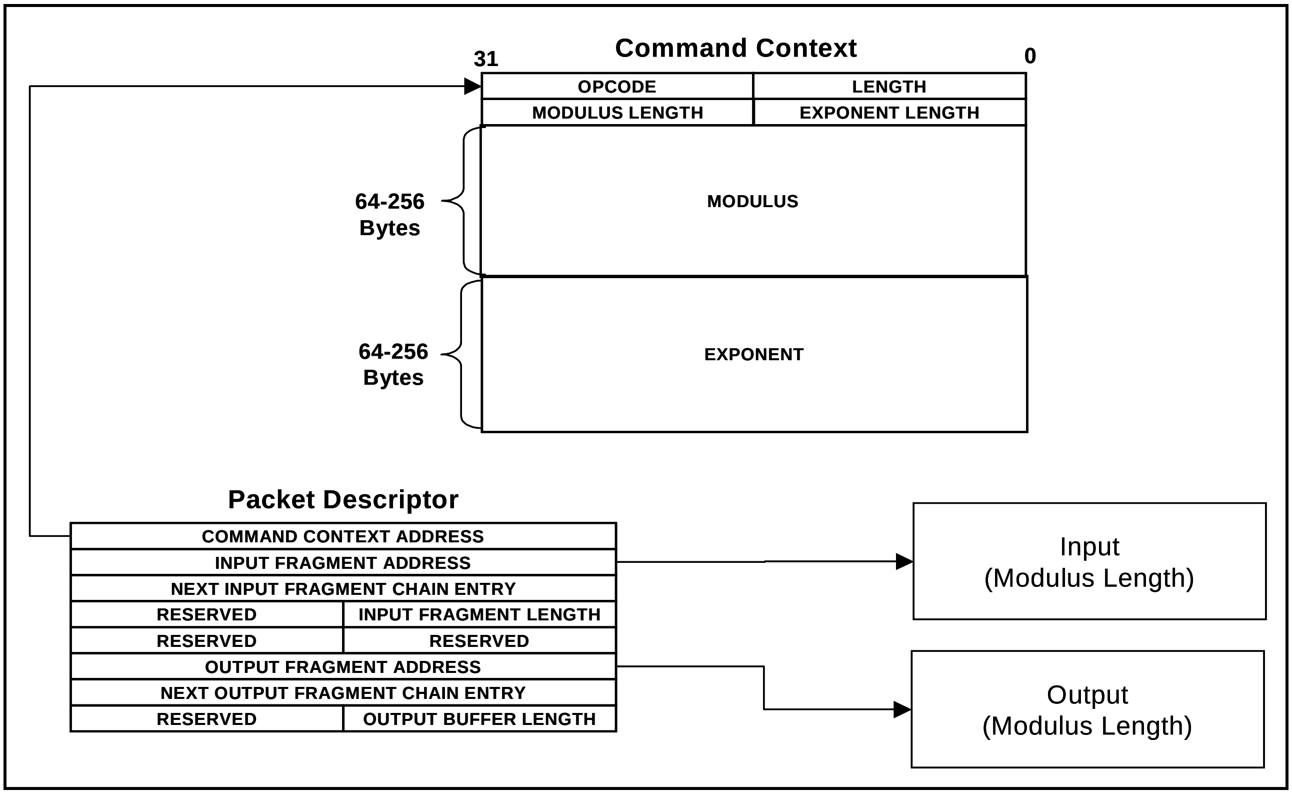


Figure 27: RSA Public Key

The single input buffer length is given in the input fragment length, in bytes, and must be one of the parameter field sizes in Table 10. The input can be byte aligned, and may require padding with zeros to the parameter field size. The output buffer must be the same length, but must be 32-bit aligned. The exponent and modulus parameter field sizes must be the same. The exponent must be smaller than the modulus.

Figure 28 shows the RSA CRT private key operation command context.The five CRT parameters, p, q, d_p, d_q, and q^{-1}, are each half the size of the public key modulus. Table 11 shows the allowed parameter field sizes in 32-bit words and bytes.

Table 11: Allowed RSA Private Key Parameter Size Increments

Modulus Length, in Bits	*Parameter Size in 32-bit words*	*Parameter SIze in Bytes*
16-256	8	32
257-384	12	48
385-512	16	64
513-768	24	96
769-1024	32	128

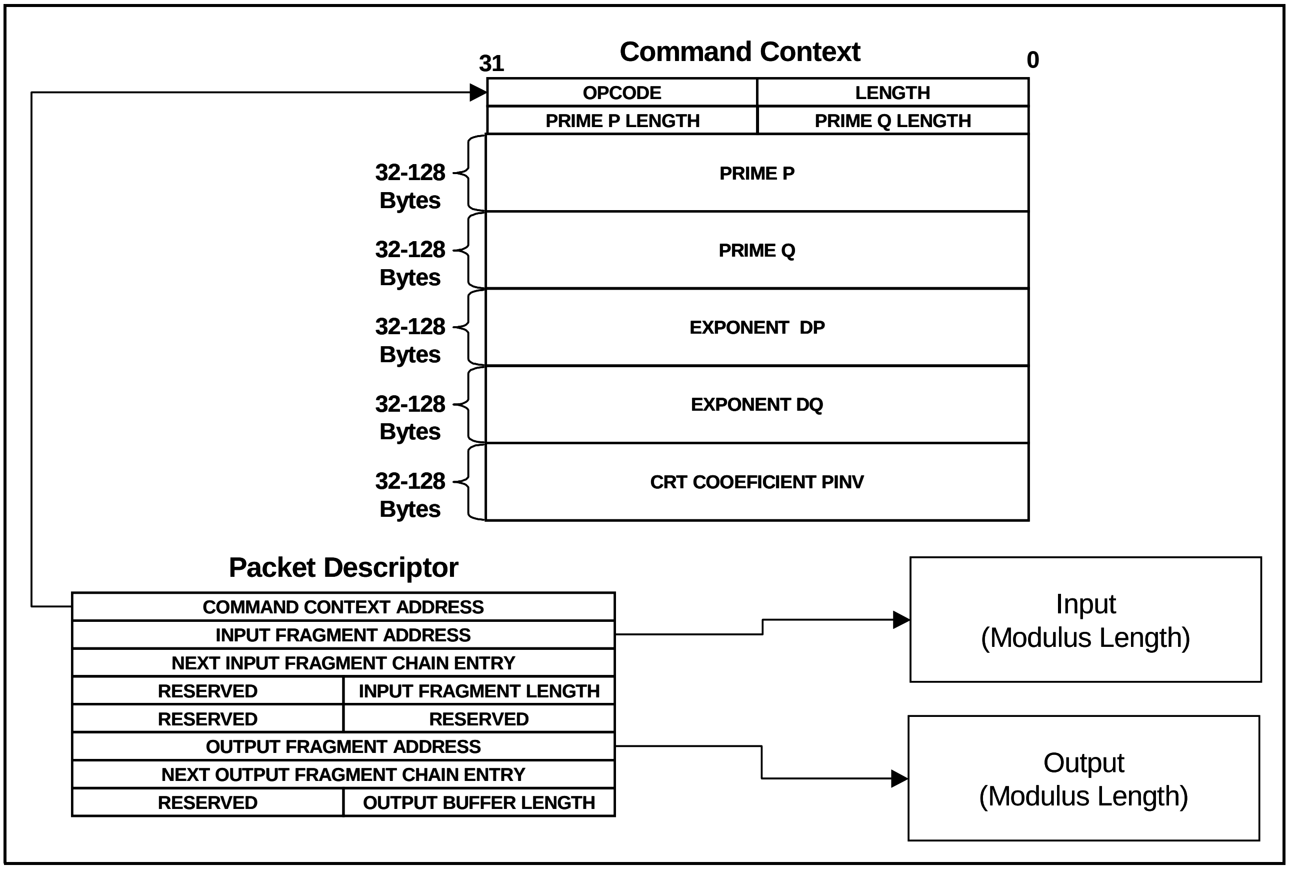


Figure 28: RSA Private Key

The command context lengths depend upon the parameter field sizes, as was the case with Diffie-Hellman. Prime P and Prime Q lengths are in bits.

This command uses MCR2@, and opcode 0x04.

DSA

DSA, also known as the Digital Signature Standard (DSS), is described in FIPS-180-2 [19] as follows.

1 p, which is an L-bit long prime modulus, $2^{L-1} < p < 2^{L}$, where L is an integer multiple of 64 greater than or equal to 512 and less than or equal to 1024.

2 q is a 160-bit prime factor of (p - 1), in other words, $2^{159} < q < 2^{160}$.

3 g = $h^{(p-1)/q}$ mod p, where h is any integer with 1 < h < (p - 1) such that $h^{(p-1)/q}$ mod p is greater than 1 (g has order q mod p).

4 x is a randomly or pseudo randomly generated integer with 0 < x < q.

5 y = g^{x} mod p

6 k, a... randomly or pseudo randomly generated integer with 0 < k < q.

The integers p, q, and g can be public and can be common to a group of users. A user's private and public keys are x and y, respectively. They are normally fixed for a period of time. Parameters x and k are used for signature generation only, and must be kept secret. Parameter k must be regenerated for each signature.

For Alice to sign a message m, and Bob to verify the message:

1 Alice generates a k as above, and uses it to compute r and s, according to the following formula:

$$r = (g^{k} \bmod p) \bmod q, \text{ and}$$
$$s = (k^{-1} (\text{SHA-1}(M) + xr)) \bmod q$$

Alice sends Bob the pair (r,s) as the signature on message M.

2 For Bob to verify the signature, he uses Alice’s public key, y, and message M, along with the public parameters p and q, and performs the following computation:

$$w = s^{-1} \bmod q$$
$$u1 = (\text{SHA-1}(M) * w) \bmod q$$
$$u2 = (r * w) \bmod q$$
$$v = ((g^{u1} * y^{u2}) \bmod p) \bmod q$$

If v equals r, Bob accepts the signature as valid.

Refer to FIPS-180-2 for details.

The BCM5812 provides separate operations for DSA sign and DSA verify. Figure 29 on page 33 shows the command context, input, and output parameter buffers for the DSA sign operation. P, Q, and G, and X correspond to p, q, g, and X in the above description, and are provided in the command context. The modulus P length is supplied in bits.

DSA sign uses MCR2@, and opcode 0x05.

The input message, M, can be supplied directly and hashed by the BCM5812, or it can be supplied as a pre-computed SHA-1 hash value. If supplied as a hash value, it is provided at the first input fragment address, 20 bytes in length, and the hash generated parameter is 0x0000. In this case, the Dlength parameter in the packet descriptor is zero.

If an explicit message is supplied for the BCM5812 to compute the SHA-1 hash, the hash generated parameter should be set to 0x0001. If M is explicitly supplied, the total length in bytes must be supplied in the Dlength field of the packet descriptor structure. M can be supplied in a single fragment of up to 65,535 bytes in length, or in multiple fragments, with the restriction that intermediate fragments, other than the last, must be exactly 64 bytes (512 bits) in length. Hash generated should be one of these two values.

The random number can be optionally generated by the BCM5812 or explicitly provided. If it is generated, generate random number is set to 0x0001. If software provides the random number, generate random number should be 0x0000, and the random number is taken from the 20 byte buffer address in the last input fragment chain entry descriptor. Figure 29 illustrates

DSA sign using a single input message fragment and optional explicitly provided random number. Generate random number should be one of these two values.

The random number is not needed for verification and is usually discarded following creation of the signature. The BCM5812 generates a 20 byte random value, which is not output.

The signature values, r and s, are output in two fragment buffers, as shown in Figure 29.

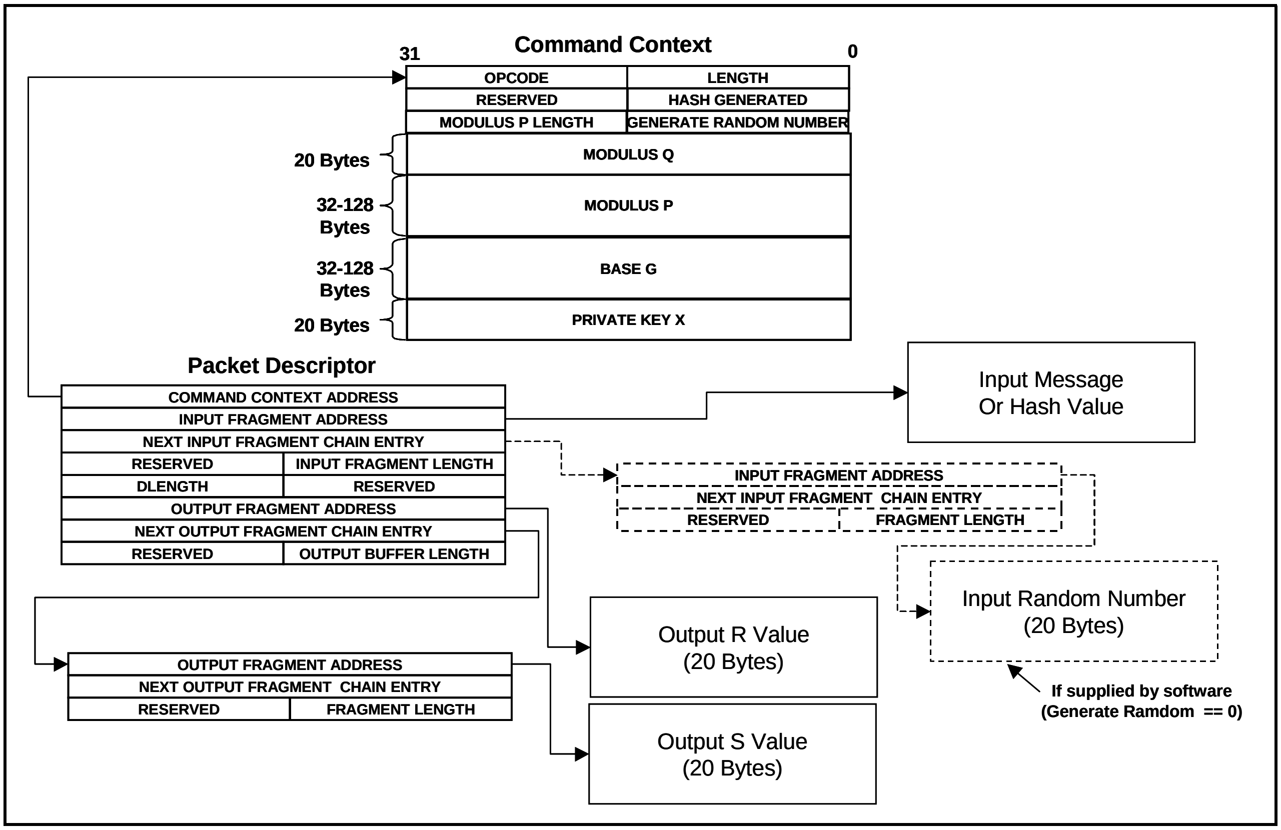


Figure 29: DSA Sign

Figure 30 on page 34 shows the DSA verify operation. As with DSA sign, the message M may be input as a pre-computed SHA-1 hash value or explicitly hashed by the BCM5812. If the hash is supplied, the hash generated parameter must be 0x0000. If the message is supplied, hash generated should be set to 0x0001. The same message fragmentation rules must be followed as for DSA sign, with intermediate fragments other than the last 64 bytes long. The signature values, r and s, are input following the hash or message, with s at the buffer address supplied in the last input fragment chain entry.

DSA verify uses MCR2@, and opcode 0x06.

At least three input fragments must be supplied, and the lengths of the last two (r and s) must be 20 bytes each. The host software can then compare the output V value to the input R value to verify the signature.

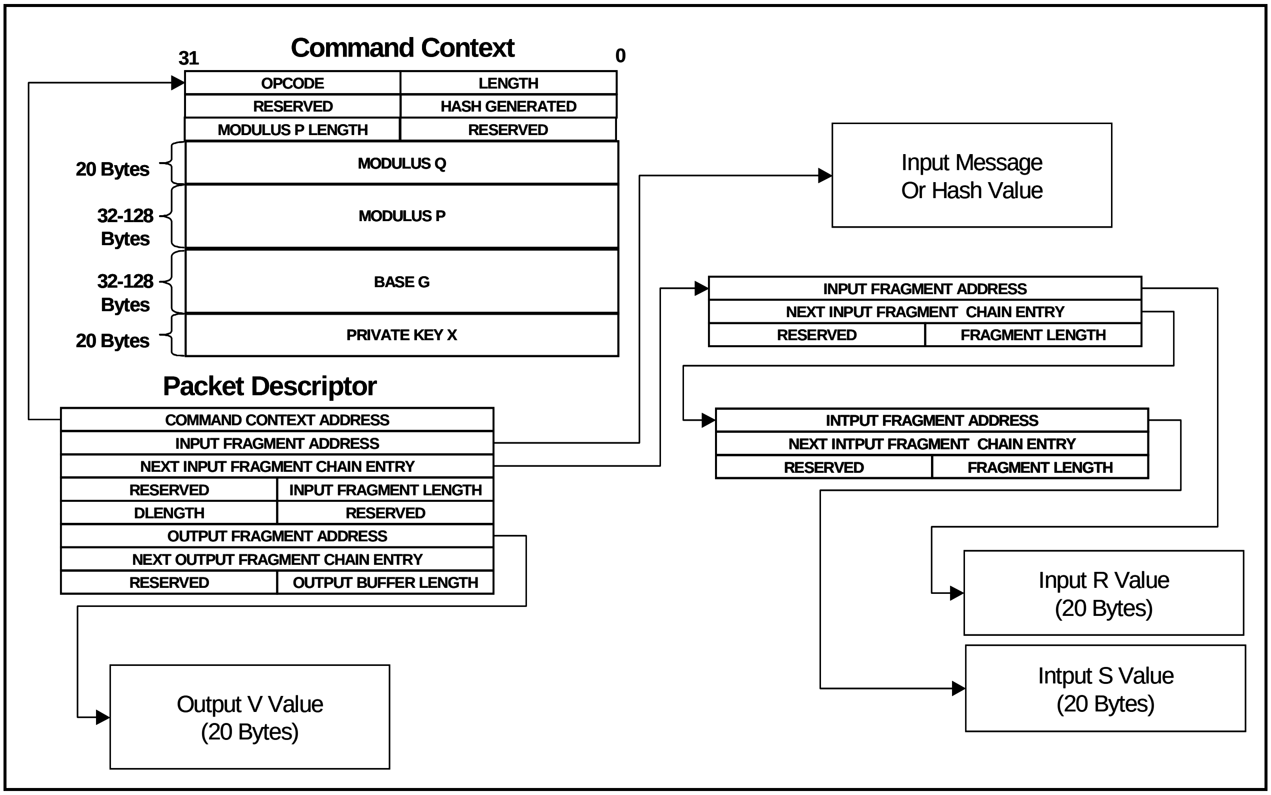


Figure 30: DSA Verify

Random Number Generation

The BCM5812 provides true random number generation using thermal noise to generate a random stream of bits that is collected as 32-bit words in a FIFO. The FIFO feeds into a SHA-1 engine, which hashes 512 bits at a time into a new set of 32-bit numbers which are placed in a second FIFO. Direct RNG output passes the FIPS-140-1 and FIPS-140-2 requirements for self testing. This shows a sufficient randomness, a uniform distribution of results across the number line, and negligible bias. SHA-1 output adds greater assurance that the data is uniformly distributed. The SHA-1 RNG output is used as the internal random source for Diffie-Hellman and DSA functions. The raw RNG data source or the SHA-1 RNG data source can be accessed directly using the random number opcodes.

The direct opcode, 0x40 using MCR2@, provides the raw output from the random number generator. This is usually used for test or certification purposes, where the raw output needs to be examined. The SHA-1 opcode, 0x41 using MCR2@, provides the SHA-1 output.

The number of bits is controlled by the Dlength parameter in the packet descriptor, and must be specified as an integral number of 32-bit words. The output buffer fragment length must be equal to the value of Dlength. Output is to a single, contiguous buffer indicated by the output fragment address in the packet descriptor. output fragmentation is not performed.

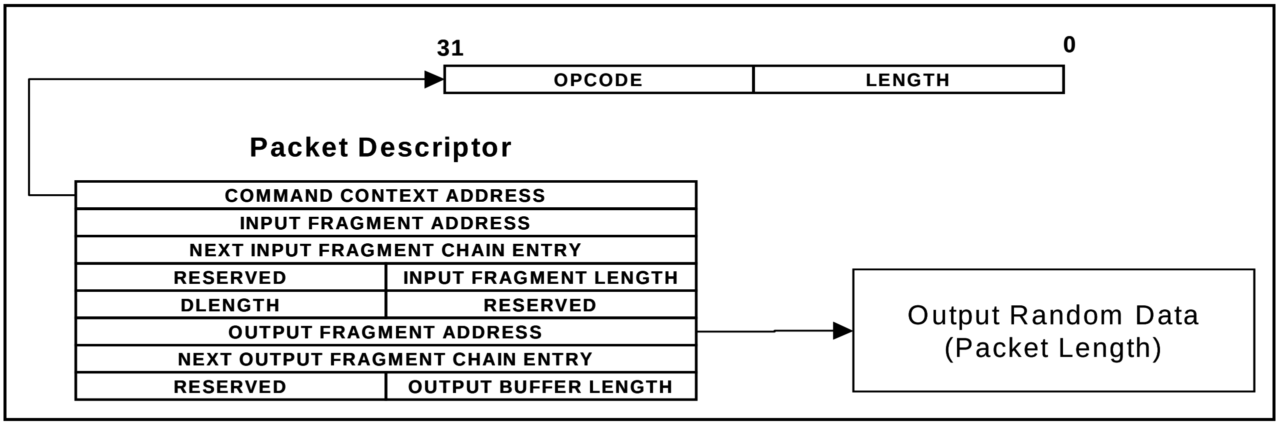


Figure 31: Random Number Generation Command Context

Modular Arithmetic

The BCM5812 provides basic, large integer modular arithmetic functions for add, subtract, multiply, remainder, and exponentiation. These commands all use MCR2@, and the opcodes listed in Table 13 on page 41. With the exception of remainder, all arguments must be less than the modulus. The BCM5812 does not reduce arguments prior to performing the operation.

Figure 32 shows the command context, input, and output structures for computing (A+B) mod N, for N up to 2048 bits. All input and output values must be supplied in a single buffer fragment. The modulus length, in bits, is supplied in the command context. All fragments must be the same size, which must be one of the parameter sizes in Table 13.

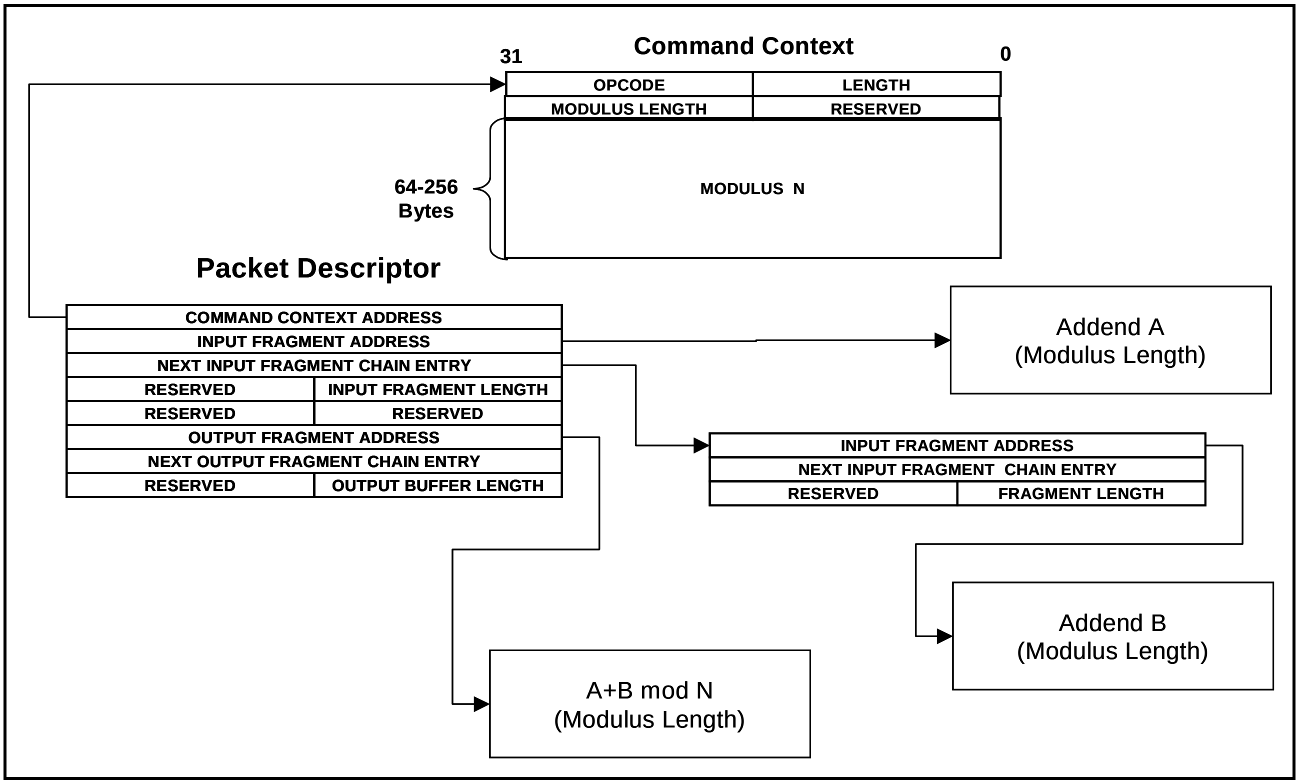


Figure 32: Modular Add

Figure 33 on page 37 shows the structure values for computing (A-B) mod N, and Figure 34 on page 38 for computing (A * B) mod N. Figure 35 on page 39 shows the unary operation A mod N, used to compute the remainder, and Figure 36 on page 40 shows A^E mod N. In all cases, the parameters are otherwise as described above for modular addition.

Figure 37 on page 41 shows the command context, input, and output structures for double modular exponentiation. This function is useful for performing the RSA private key operation when the modulus is the product of three or more primes (referred to as Multi-Prime^{TM}). In this case, the lengths for the two moduli length, in bits, are supplied in the command context as modulus N1 Length and modulus N0 length, respectively. Both values, in bits, must be between 16 and 512. All input and output parameters must be supplied in either 256 (32 byte) or 512 bit (64 byte) buffers.

Table 12: Allowed Modular Arithmetic Parameter Field Size Increments

Modulus Length, in Bits	*Parameter Size in 32-bit words*	*Parameter SIze in Bytes*
16-256	8	32
257-512	16	64
513-768	24	96
769-1024	32	128
1025-1536	48	192
1537-2048	64	256

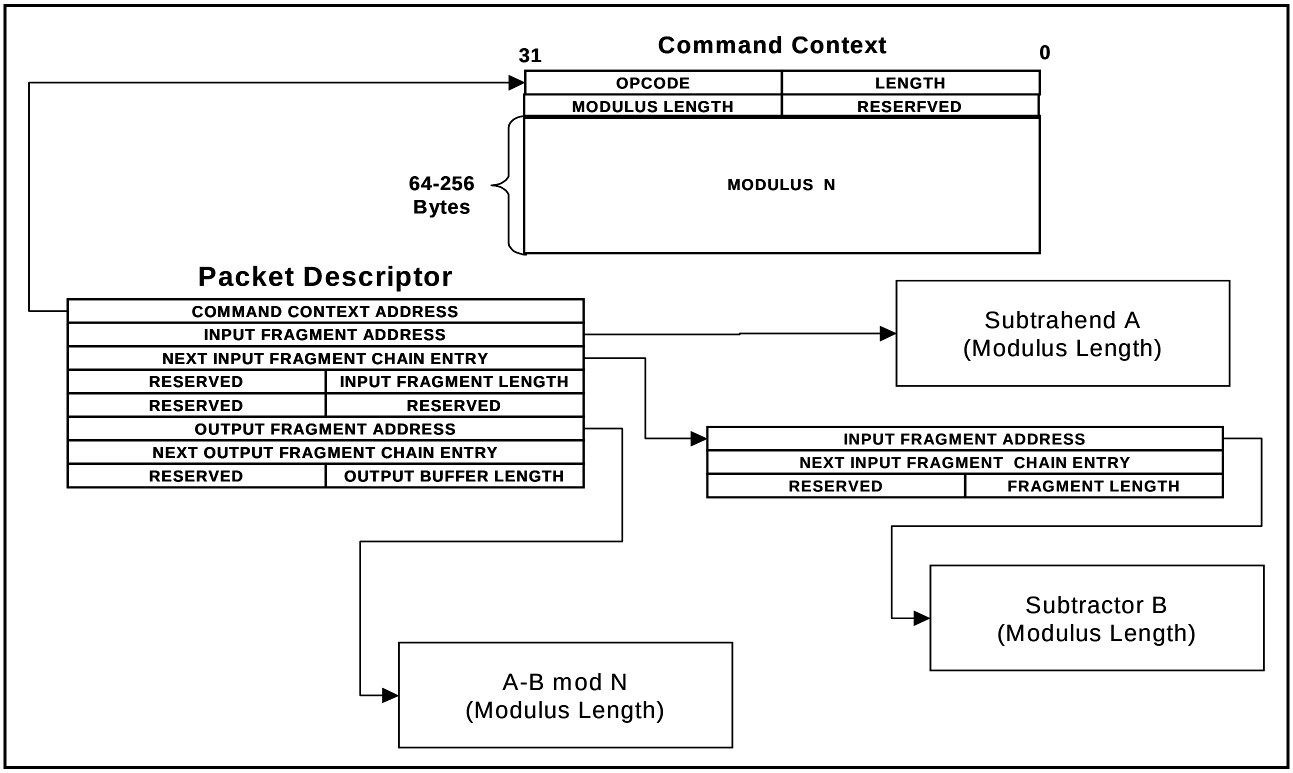


Figure 33: Modular Subtract

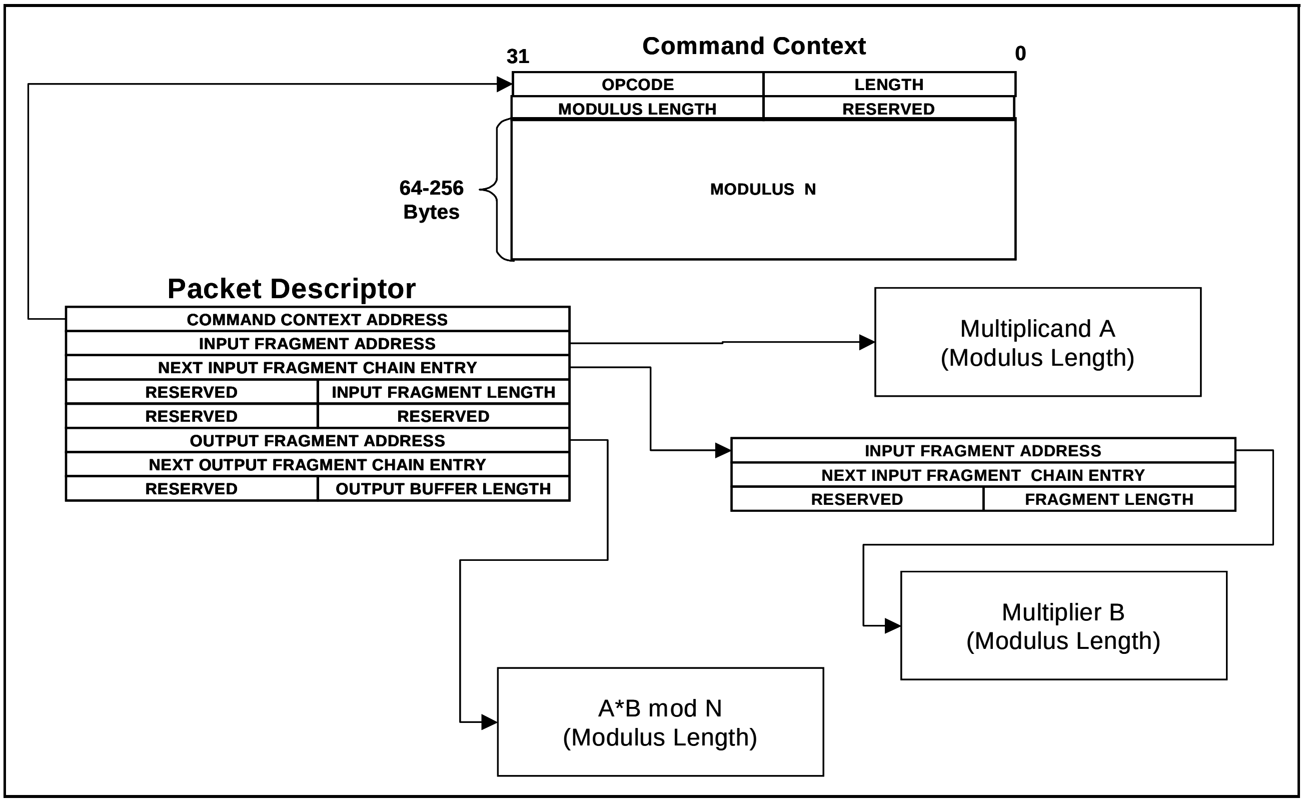


Figure 34: Modular Multiply

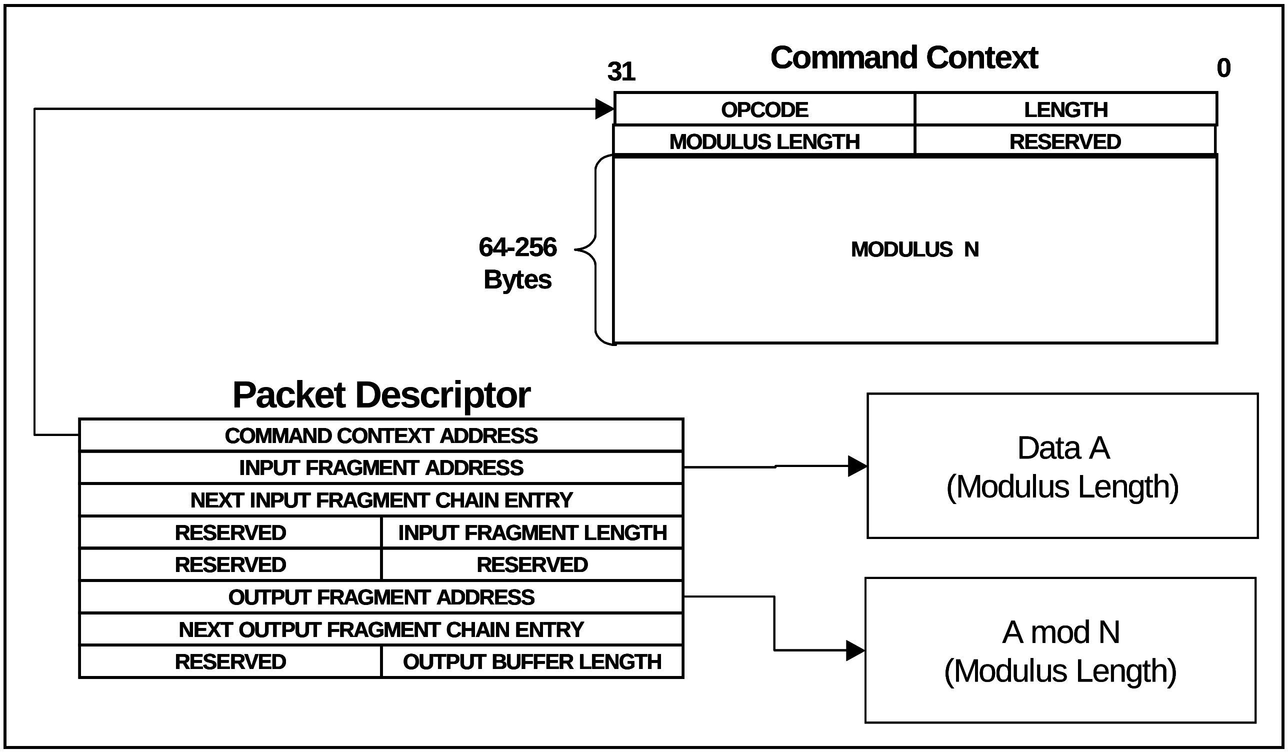


Figure 35: Modular Remainder

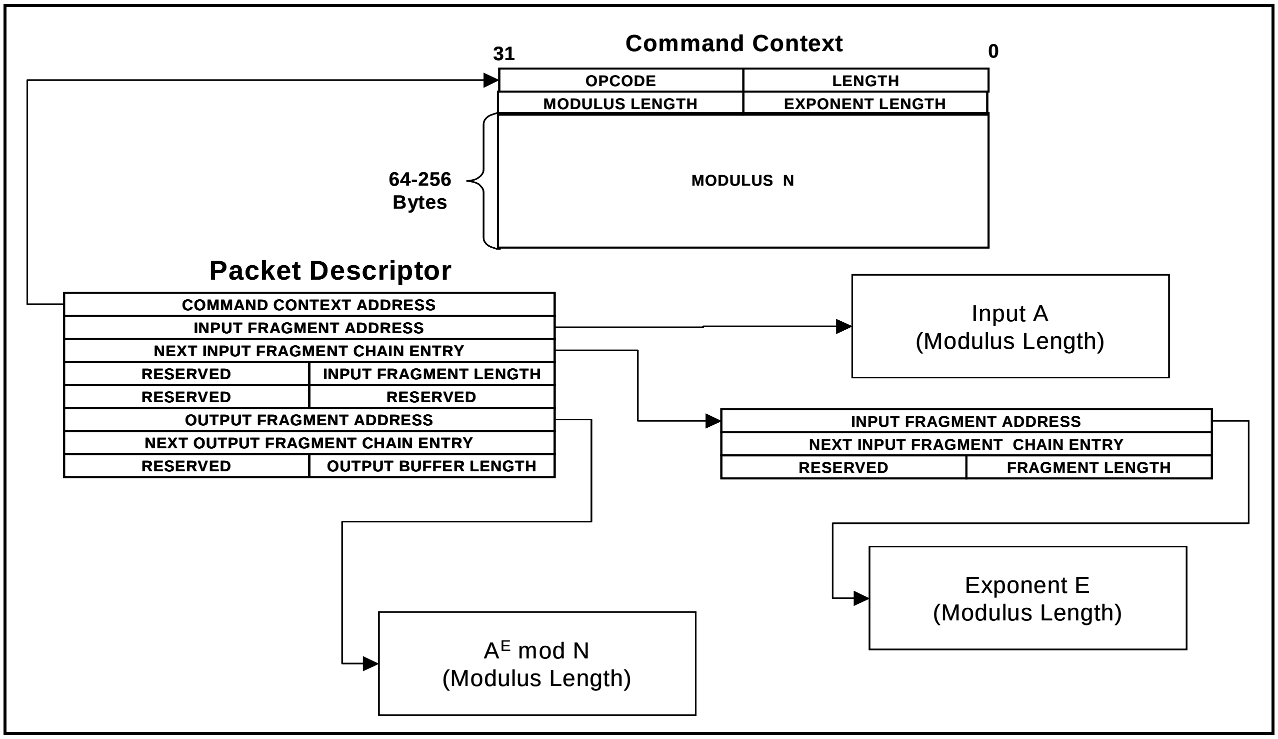


Figure 36: Modular Exponentiation

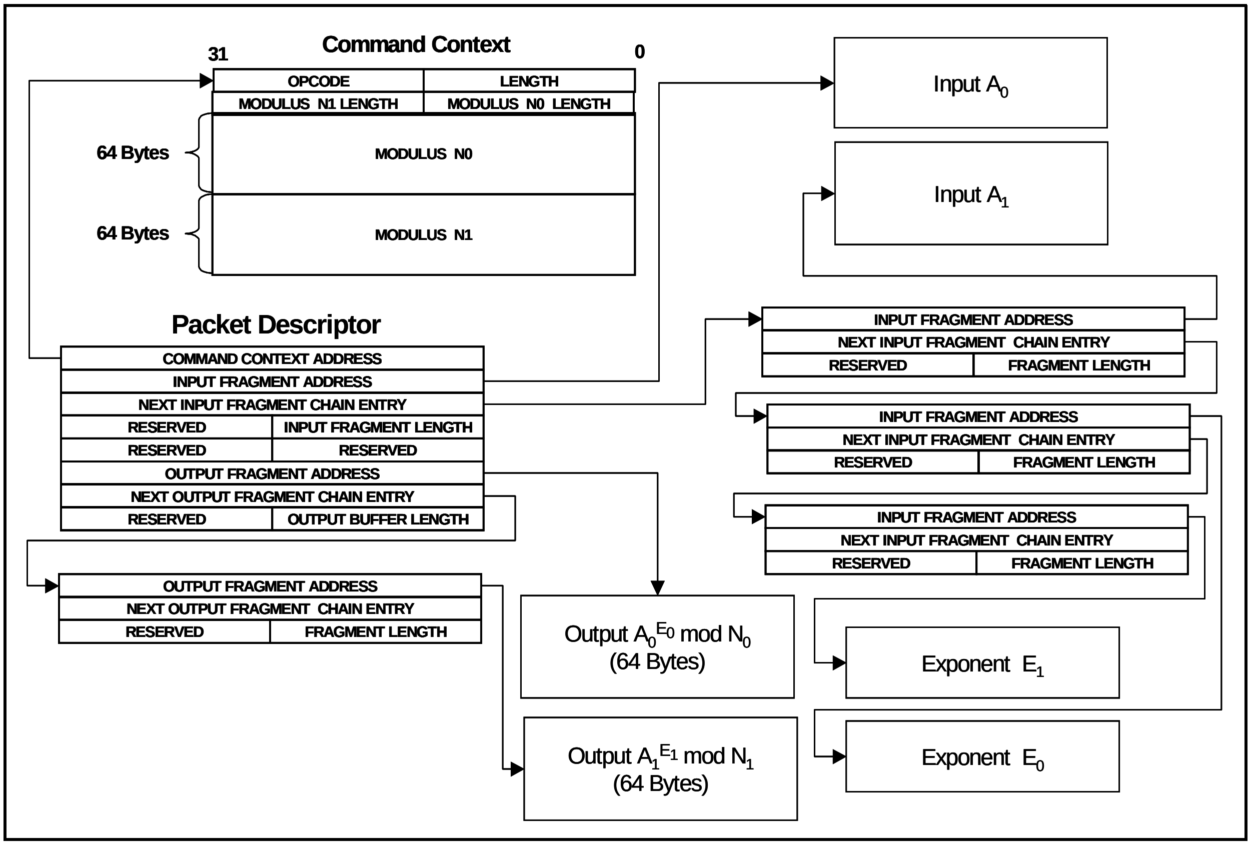


Figure 37: Double Modular Exponentiation

Table 13: Modular Arithmetic Opcodes

Opcode	*Name*	*Function*
0x43	Modular Addition	(A + B) mod N
0x44	Modular Subtraction	(A - B) mod N
0x45	Modular Multiplication	(A * B) mod N
0x46	Modular Reduction	A mod N
0x47	Modular Exponentiation	A^E mod N
0x49	Double Modular Exponentiation	A_1^{E1} mod N1 A_2^{E2} mod N2

INTERRUPT PROCESSING

The BCM5812 generates an interrupt after completion of all packet descriptors for a particular MCR structure if the following two conditions hold:

- interrupts are enabled in the DMA control register (See Table 20 on page 59) (MCR1INT_EN enables interrupts for symmetric operations pushed to MCR1@, and MCR2INT_EN enables interrupts for asymmetric and random number operations pushed to MCR2@), and
- the suppress interrupts bit in the MCR header word (See Table 1 on page 2) is zero. If this bit is set, the BCM5812 does not issue a interrupt upon completion of processing of that MCR structure.

The DMA status register indicates interrupt status via the MCR1_INTR and MCR2_INTR bits, respectively. These bits must be explicitly cleared by writing a 1 into the appropriate bit position.

The BCM5812 also generates an interrupt when either the MCR1@ or MCR2@ FIFO empties, as long as interrupts are enabled for that FIFO. This mechanism is intended to deal with the condition where a sequence of MCRs are pushed with interrupts suppressed. The DMA status register MCR1_ALL_EMPTY and MCR2_ALL_EMPTY bits indicate that their respective interrupt has been issued. These are cleared by explicitly writing a 1 to the bit position, or by writing a 1 to the MCR1_INTR or MCR2_INTR bit, respectively.

The BCM5812 generates an interrupt on DMA error if DMAERR_EN is set in the DMA control register. The DMA status register indicates status of this interrupt using DMAERR_INTR.

EXPORT CONTROL

The BCM5812 export control feature allows strong bulk cryptography to be disabled to facilitate products with retail export classification. Export mode is controlled externally by whether the EXPORT pin (see Table 15 on page 47, Table 16 on page 50, and Table 17 on page 53) is pulled high or low. If EXPORT is high, export mode is enabled, allowing only 56-bit DES and disabling ARCFOUR, AES, and 3DES. If EXPORT is pulled low, all bulk cryptographic functionality is enabled.

EXPORT is internally pulled high, and enabled by default.

ENDIAN CONSIDERATIONS

The BCM5812 is designed to work with both little endian and big endian host processors, in both standard and non-standard PCI bus configurations.

The PCI bus is specified to work naturally with little endian host processors. Figure 38 on page 43 illustrates a typical little endian 64-bit host processor bus configuration. In Figure 38 on page 43, the bytes are labelled from A to F, in the order that they would be addressed by the host processor from memory (i.e., starting with the byte at the little end of the memory word).

Most big endian host processor systems are configured to swap bytes between the host processor or memory and the PCI bus. Figure 39 on page 44 illustrates a typical big endian 64-bit host processor bus configuration. In Figure 39 on page 44, the bytes are also labelled in the order that they would be addressed by the host processor from memory (i.e., starting from the big end of the memory word).

The byte order that is seen at a device on the PCI bus is the same in both cases. When used with a big endian host processor, this is sometimes referred to as a match byte lanes policy.

Unfortunately, this policy produces the wrong result for non-byte stream data, including fields built by the host processor as 16-bit lengths, 32-bit addresses, or 32-bit BigNum elements. These appear in the BCM5812 MCR, command context, and fragment chain entry structures. Ordinarily, host software would have to byte swap such fields in order to undo the effect of the byte swap.

The BCM5812 provides two software enabled endian control flags for adjusting the byte order to match the host processor system configuration and minimize or eliminate the need for byte swapping in software. Table 14 on page 44 describes these flags. The default setting at reset is for a little endian host processor on a typical PCI bus configuration.

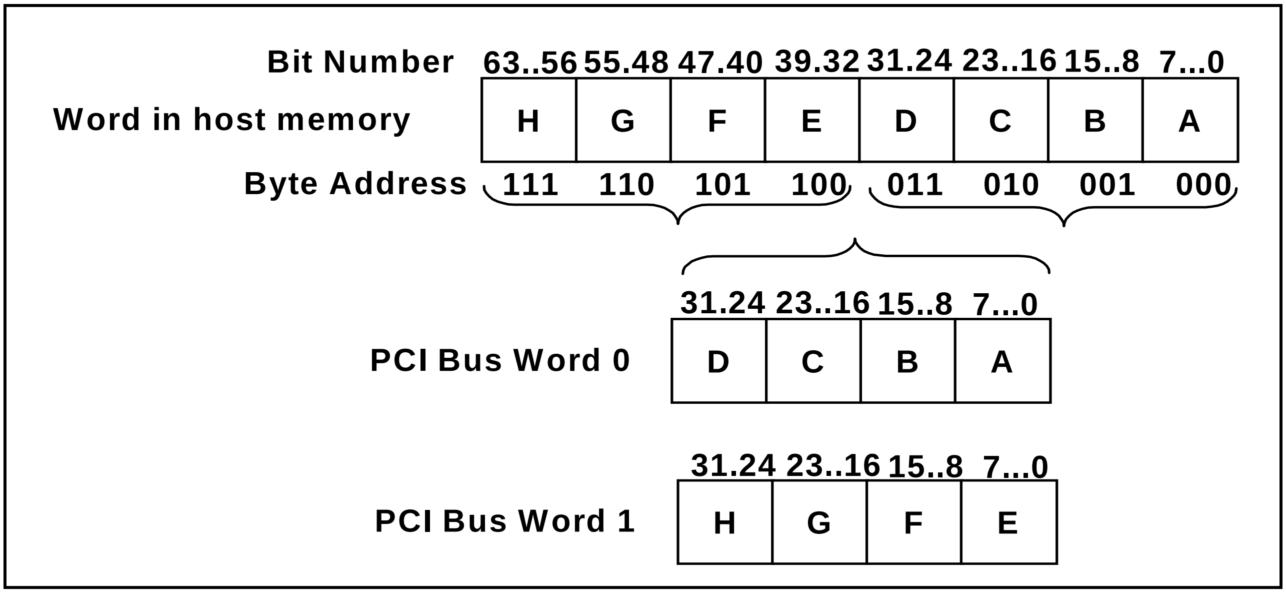


Figure 38: Typical Little Endian Processor PCI Bus Configuration

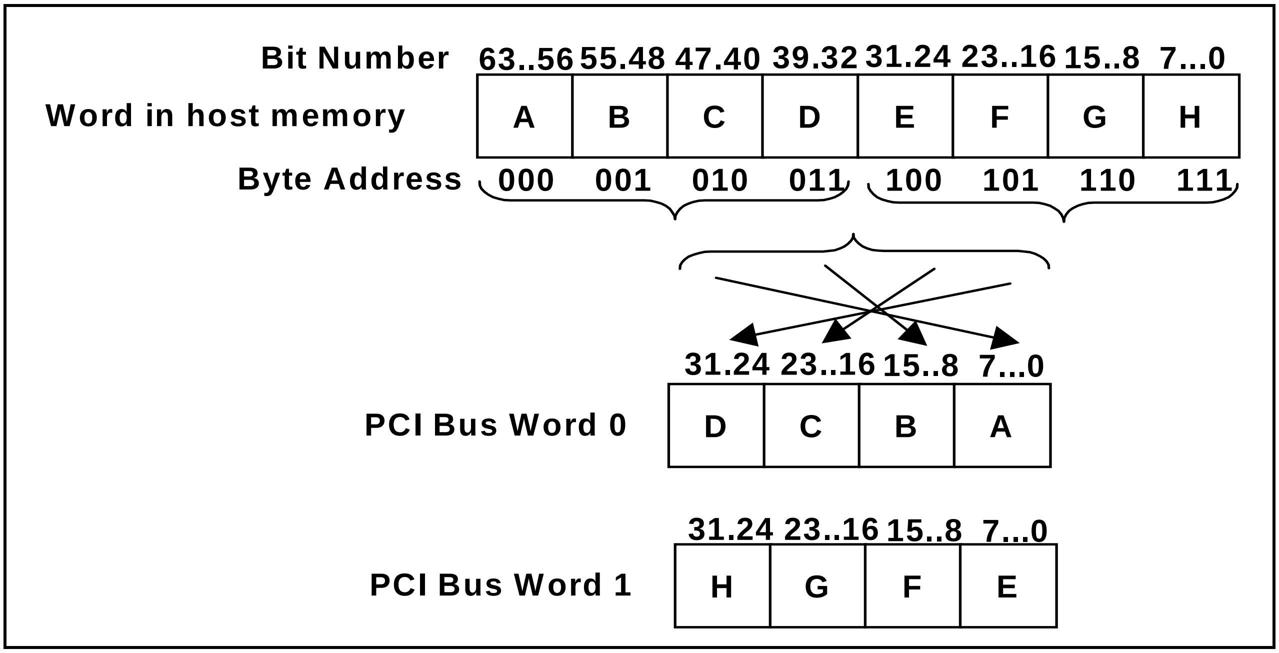


Figure 39: Typical Big Endian Processor PCI Bus Configuration

Table 14: Endian Control Flags

Name	*Use*	*Description*	*Default*
CRYPTONET_LE	Little Endian Host Processor	If zero, swap bytes for MCR, command context, fragment chain entries, BigNum data fragments, and ARCFOUR States output using DMA bus master access.	1
NORMAL_PCI	Conventional PCI Configuration	If zero, swap bytes for all data read or written over the PCI bus, master or slave.	1

For a big endian host processor, clearing CRYPTONET_LE enables most host structures to be constructed without the need for software bye swapping. This applies to the structures shown in Figure 5 on page 7, and to the ARCFOUR State output. Byte stream data is unaffected.

NORMAL_PCI would be used in non-standard PCI bus configurations. Figure 40 shows an example of a match bit lanes bus policy. If NORMAL_PCI is cleared, the BCM5812 byte swaps everything on its way in or out, whether a master or slave access, including the CSRs.

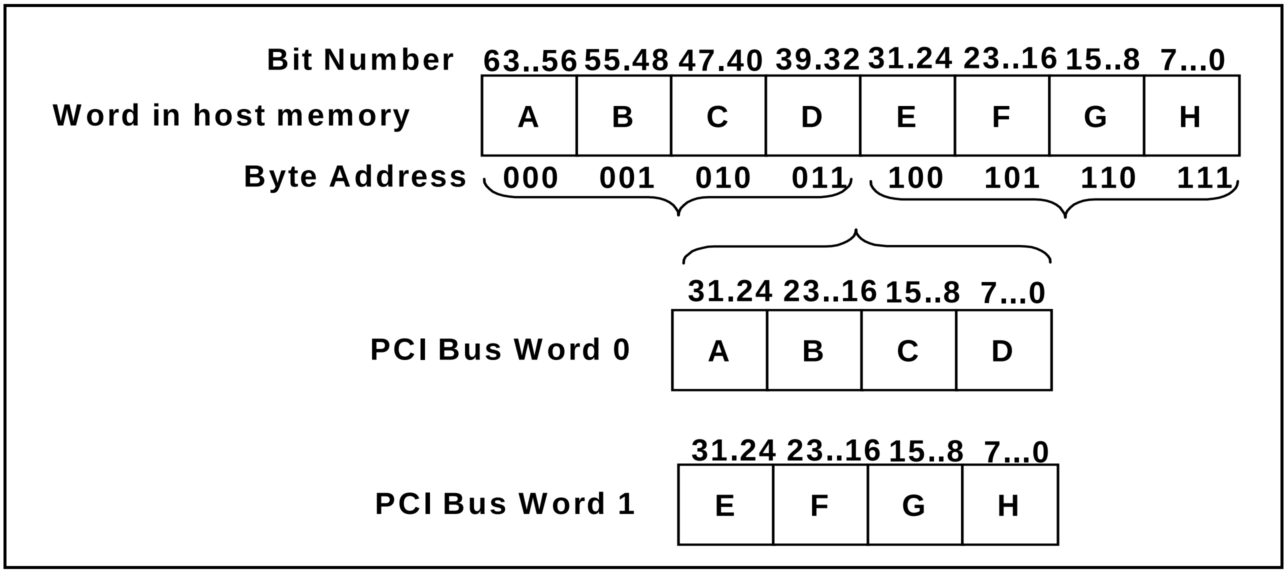


Figure 40: Match Bit Lanes PCI Bus Configuration

Note **At least 50 clock cycles are needed following a change to the CRYPTONET_LE and/or NORMAL_PCI setting.**

Section 2: Hardware

SIGNAL DEFINITION

	A	B	C	D	E	F	G	H	J	K	L	M	N	P	
14	VDDO	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	14
13	EPROM	VDDO	GNDC	GNDC	VDDC	VDDC	SGND	VIO	VDDC	VDDC	VDDC	GNDC	GNDC	VDDO	13
12	EPROM_	PROM_	VDDO	VIO	VDDC	VDDC	VDDC	VDDC	VDDC	VDDC	VDDC	GNDC	VDDO	SGND	12
11	NC5	PROM_	PROM_(	VDDO	VDDC	VDDC	VDDC	VDDC	VDDC	VDDO	VDDO	VDDO	VIO	AD[01]	11
10	NC4	NC8	NC9	NC10	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	VDDC	AD[00]	AD[02]	10
9	NC3	NC7	RNGOSC	VDDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	VDDC	AD[03]	AD[04]	9
8	TCK	TEST	SGND	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	AD[06]	AD[05]	AD[07]	8
7	TMS	TRST#	TDI	TDO	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	VDDO	AD[08]	C/BE#[0]	7
6	NC2	AVDD1	AGND1	AGND1	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	AD[11]	AD[10]	AD[09]	6
5	AVDD1	AVDD1	AVDD2	AVDD2	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	GNDC	VDDO	AD[13]	AD[12]	5
4	EXPORT	NC6	AVDD2	VDDC	GNDC	AD[28]	VDDC	IDSEL	VDDO	TRDY#	VDDC	PAR	AD[15]	AD[14]	4
3	NC1	AGND2	VDDC	PCI_CLK	RST#	AD[27]	AD[24]	AD[22]	AD[19]	IRDY#	LOCK#	VDDC	SERR#	C/BE#[1]	3
2	AGND2	VDDC	VIO	SGND	AD[30]	AD[26]	GNT#	AD[23]	AD[20]	AD[17]	C/BE#[2]	DEVSEL#	VDDC	PERR#	2
1	GNDC	VIO	INTA#	AD[31]	AD[29]	AD[25]	REQ#	C/BE#[3]	AD[21]	AD[18]	AD[16]	FRAME#	STOP#	VDDC	1
	A	B	C	D	E	F	G	H	J	K	L	M	N	P	

Figure 41: 196-pin FBGA Pinout Diagram

BALLOUT BY BALL NUMBER

Table 15: Ballout by Ball Number

Ball	*Signal Name*
A1	GNDC
A2	AGND2
A3	NC1
A4	EXPORT
A5	AVDD1
A6	NC2
A7	TMS
A8	TCK
A9	NC3
A10	NC4
A11	NC5
A12	EEPROM_DI
A13	$\overline{\text{EEPROM}}$
A14	VDDO
B1	VIO
B2	VDDC
B3	AGND2
B4	NC6
B5	AVDD1
B6	AVDD1
B7	$\overline{\text{TRST}}$
B8	TEST
B9	NC7
B10	NC8
B11	EEPROM_DO
B12	EEPROM_SK
B13	VDDO
B14	GNDC
C1	$\overline{\text{INTA}}$
C2	VIO
C3	VDDC
C4	AVDD2
C5	AVDD2
C6	AGND1
C7	TDI
C8	SGND
C9	RNGOSC
C10	NC9
C11	EEPROM_CS
C12	VDDO
C13	GNDC
C14	GNDC
D1	AD[31]
D2	SGND
D3	PCI_CLK
D4	VDDC
D5	AVDD2
D6	AGND1
D7	TDO
D8	GNDC
D9	VDDC
D10	NC10
D11	VDDO
D12	VIO
D13	GNDC
D14	GNDC
E1	AD[29]
E2	AD[30]
E3	$\overline{\text{RST}}$
E4	GNDC
E5	GNDC
E6	GNDC
E7	GNDC
E8	GNDC
E9	GNDC
E10	GNDC
E11	VDDC
E12	VDDC

Table 15: Ballout by Ball Number

Ball	Signal Name
E13	VDDC
E14	GNDC
F1	AD[25]
F2	AD[26]
F3	AD[27]
F4	AD[28]
F5	GNDC
F6	GNDC
F7	GNDC
F8	GNDC
F9	GNDC
F10	GNDC
F11	VDDC
F12	VDDC
F13	VDDC
F14	GNDC
G1	$\overline{\text{REQ}}$
G2	$\overline{\text{GNT}}$
G3	AD[24]
G4	VDDC
G5	GNDC
G6	GNDC
G7	GNDC
G8	GNDC
G9	GNDC
G10	GNDC
G11	VDDC
G12	VDDC
G13	SGND
G14	GNDC
H1	$\overline{\text{C/BE}}$[3]
H2	AD[23]
H3	AD[22]
H4	IDSEL
H5	GNDC
H6	GNDC
H7	GNDC
H8	GNDC
H9	GNDC
H10	GNDC
H11	VDDC
H12	VDDC
H13	VIO
H14	GNDC
J1	AD[21]
J2	AD[20]
J3	AD[19]
J4	VDDO
J5	GNDC
J6	GNDC
J7	GNDC
J8	GNDC
J9	GNDC
J10	GNDC
J11	VDDC
J12	VDDC
J13	VDDC
J14	GNDC
K1	AD[18]
K2	AD[17]
K3	$\overline{\text{IRDY}}$
K4	$\overline{\text{TRDY}}$
K5	GNDC
K6	GNDC
K7	GNDC
K8	GNDC
K9	GNDC
K10	GNDC
K11	VDDO
K12	VDDC
K13	VDDC
K14	GNDC

Table 15: Ballout by Ball Number

Ball	*Signal Name*
L1	AD[16]
L2	$\overline{\text{C/BE}}$[2]
L3	$\overline{\text{LOCK}}$
L4	VDDC
L5	GNDC
L6	GNDC
L7	GNDC
L8	GNDC
L9	GNDC
L10	GNDC
L11	VDDO
L12	VDDC
L13	VDDC
L14	GNDC
M1	$\overline{\text{FRAME}}$
M2	$\overline{\text{DEVSEL}}$
M3	VDDC
M4	PAR
M5	VDDO
M6	AD[11]
M7	VDDO
M8	AD[06]
M9	VDDC
M10	VDDC
M11	VDDO
M12	GNDC
M13	GNDC
M14	GNDC
N1	$\overline{\text{STOP}}$
N2	VDDC
N3	$\overline{\text{SERR}}$
N4	AD[15]
N5	AD[13]
N6	AD[10]
N7	AD[08]
N8	AD[05]
N9	AD[03]
N10	AD[00]
N11	VIO
N12	VDDO
N13	GNDC
N14	GNDC
P1	VDDC
P2	$\overline{\text{PERR}}$
P3	$\overline{\text{C/BE}}$[1]
P4	AD[14]
P5	AD[12]
P6	AD[09]
P7	$\overline{\text{C/BE}}$[0]
P8	AD[07]
P9	AD[04]
P10	AD[02]
P11	AD[01]
P12	SGND
P13	VDDO
P14	GNDC

BALLOUT BY SIGNAL NAME

Table 16: Ballout by Signal Name

Ball	*Signal Name*
N10	AD[00]
P11	AD[01]
P10	AD[02]
N9	AD[03]
P9	AD[04]
N8	AD[05]
M8	AD[06]
P8	AD[07]
N7	AD[08]
P6	AD[09]
N6	AD[10]
M6	AD[11]
P5	AD[12]
N5	AD[13]
P4	AD[14]
N4	AD[15]
L1	AD[16]
K2	AD[17]
K1	AD[18]
J3	AD[19]
J2	AD[20]
J1	AD[21]
H3	AD[22]
H2	AD[23]
G3	AD[24]
F1	AD[25]
F2	AD[26]
F3	AD[27]
F4	AD[28]
E1	AD[29]
E2	AD[30]
D1	AD[31]
C6	AGND1

Table 16: Ballout by Signal Name (Cont.)

Ball	*Signal Name*
D6	AGND1
A2	AGND2
B3	AGND2
A5	AVDD1
B5	AVDD1
B6	AVDD1
C4	AVDD2
C5	AVDD2
D5	AVDD2
P7	$\overline{\text{C/BE}}$[0]
P3	$\overline{\text{C/BE}}$[1]
L2	$\overline{\text{C/BE}}$[2]
H1	$\overline{\text{C/BE}}$[3]
M2	$\overline{\text{DEVSEL}}$
A13	$\overline{\text{EEPROM}}$
C11	EEPROM_CS
A12	EEPROM_DI
B11	EEPROM_DO
B12	EEPROM_SK
A4	EXPORT
M1	$\overline{\text{FRAME}}$
A1	GNDC
B14	GNDC
C13	GNDC
C14	GNDC
D8	GNDC
D13	GNDC
D14	GNDC
E4	GNDC
E5	GNDC
E6	GNDC
E7	GNDC
E8	GNDC
E9	GNDC
E10	GNDC
E14	GNDC

Table 16: Ballout by Signal Name (Cont.)

Ball	***Signal Name***
F5	GNDC
F6	GNDC
F7	GNDC
F8	GNDC
F9	GNDC
F10	GNDC
F14	GNDC
G5	GNDC
G6	GNDC
G7	GNDC
G8	GNDC
G9	GNDC
G10	GNDC
G14	GNDC
H5	GNDC
H6	GNDC
H7	GNDC
H8	GNDC
H9	GNDC
H10	GNDC
H14	GNDC
J5	GNDC
J6	GNDC
J7	GNDC
J8	GNDC
J9	GNDC
J10	GNDC
J14	GNDC
K5	GNDC
K6	GNDC
K7	GNDC
K8	GNDC
K9	GNDC
K10	GNDC
K14	GNDC
L5	GNDC

Table 16: Ballout by Signal Name (Cont.)

Ball	***Signal Name***
L6	GNDC
L7	GNDC
L8	GNDC
L9	GNDC
L10	GNDC
L14	GNDC
M12	GNDC
M13	GNDC
M14	GNDC
N13	GNDC
N14	GNDC
P14	GNDC
G2	$\overline{\text{GNT}}$
H4	IDSEL
C1	$\overline{\text{INTA}}$
K3	$\overline{\text{IRDY}}$
L3	$\overline{\text{LOCK}}$
A3	NC1
D10	NC10
A6	NC2
A9	NC3
A10	NC4
A11	NC5
B4	NC6
B9	NC7
B10	NC8
C10	NC9
M4	PAR
D3	PCI_CLK
P2	$\overline{\text{PERR}}$
G1	$\overline{\text{REQ}}$
C9	RNGOSC
E3	$\overline{\text{RST}}$
N3	$\overline{\text{SERR}}$
C8	SGND
D2	SGND

Table 16: Ballout by Signal Name (Cont.)

Ball	Signal Name
G13	SGND
P12	SGND
N1	$\overline{\text{STOP}}$
A8	TCK
C7	TDI
D7	TDO
B8	TEST
A7	TMS
K4	$\overline{\text{TRDY}}$
B7	$\overline{\text{TRST}}$
B2	VDDC
C3	VDDC
D4	VDDC
D9	VDDC
E11	VDDC
E12	VDDC
E13	VDDC
F11	VDDC
F12	VDDC
F13	VDDC
G4	VDDC
G11	VDDC
G12	VDDC
H11	VDDC
H12	VDDC
J11	VDDC
J12	VDDC
J13	VDDC
K12	VDDC
K13	VDDC
L4	VDDC
L12	VDDC
L13	VDDC
M3	VDDC
M9	VDDC
M10	VDDC

Table 16: Ballout by Signal Name (Cont.)

Ball	Signal Name
N2	VDDC
P1	VDDC
A14	VDDO
B13	VDDO
C12	VDDO
D11	VDDO
J4	VDDO
K11	VDDO
L11	VDDO
M5	VDDO
M7	VDDO
M11	VDDO
N12	VDDO
P13	VDDO
B1	VIO
C2	VIO
D12	VIO
H13	VIO
N11	VIO

SIGNAL DEFINITIONS

Table 17: Signal Definitions

Signal Name	***I/O***	***Description***
AD[31:0]	IO	PCI multiplexed Address/Data bus.
PCI_CLK	I	PCI Clock, 33 MHz.
$\overline{\text{GNT}}$	I	PCI Bus Grant allowing BCM5812 to use the bus.
$\overline{\text{FRAME}}$	IO	PCI Frame, indicates the beginning and duration of a master transfer.
$\overline{\text{IRDY}}$	IO	PCI Initiator ready.
$\overline{\text{TRDY}}$	IO	PCI Target ready.
$\overline{\text{DEVSEL}}$	IO	PCI Device Select, asserted by an access target.
$\overline{\text{STOP}}$	IO	PCI Stop, requesting that the current master stop an active transfer.
$\overline{\text{PERR}}$	IO	PCI Parity Error.
$\overline{\text{SERR}}$	O	PCI System Error, open drain.
PAR	IO	PCI Parity.
$\overline{\text{REQ}}$	O	PCI Bus Request.
$\overline{\text{RST}}$	I	PCI Reset, tri-states all PCI outputs.
$\overline{\text{INTA}}$	O	PCI interrupt output, open drain.
$\overline{\text{C/BE}}$[3:0]	IO	PCI Command/Byte Enable, provides PCI bus command and data byte enables.
IDSEL	I	PCI Initialization Device Request, used for PCI configuration cycles.
$\overline{\text{LOCK}}$	I	PCI lock for atomic operation, must be pulled up to VDDO.
VDDC	–	Core power pins, must be connected to a 1.8V(core) source.
VDDO	–	Peripheral power pins, must be connected to a 3.3V(I/O) source.
GND	–	Core and Peripheral ground pins.
AVCC1	–	Analog VCC for PLL1, must be connected to a quiet 1.8V source.
AGND1	–	Analog GND for PLL1.
AVCC2	–	Analog VCC for PLL2, must be connected to a quiet 1.8V source.
AGND2	–	Analog GND for PLL2.
VIO	–	PCI clamp voltage bias. Connect to 3.3V for 3.3V signaling environments.Connect to 5V for 5V signaling environments.
SGND	–	Clamp ground (substrate ground).
EXPORT	I	EXPORT pin (high = 56-bit DES encryption; disable ARCFOUR; disable AES; low = 3DES strong encryption; enable ARCFOUR; enable AES).
TEST	I	Test pin, must be grounded for regular operation. When TEST is high, all outputs are tri-stated. It is internally pulled down.
$\overline{\text{TRST}}$	I	Must be connected to ground for normal operation. Used for boundary scan JTAG testing. It is internally pulled down.
TMS	I	Test mode select for JTAG boundary scan. Must be connected to VDDO for normal operation. It is internally pulled up.

Table 17: Signal Definitions (Cont.)

Signal Name	*I/O*	*Description*
TCK	I	Test mode clock for JTAG boundary scan. Unused in normal operation; connect to either high or low static level. It is internally pulled up.
TDI	I	Test data in for JTAG boundary scan. Unused in normal operation; connect to either high or low static level. It is internally pulled up.
TDO	O	Test data out for JTAG boundary scan. Unused in normal operation.
RNGOSC	I	Optional random number generator oscillator. It is Ex-ORed with internal oscillator to provide random number source. It is internally pulled down.
$\overline{\text{EEPROM}}$	I	If $\overline{\text{EEPROM}}$ is Low, the PCI configuration space is loaded from attached EEPROM. Otherwise, the PCI configuration space contains default values. It is internally pulled up.
EEPROM_CS	O	Chip Select for the attached EEPROM. It connects to Chip Select pin of the EEPROM.
EEPROM_SK	O	Serial Clock for the attached EEPROM. It connects to Serial Clock pin of the EEPROM.
EEPROM_DI	I	Data In for the attached EEPROM. It connects to Data Out pin of the EEPROM.
EEPROM_DO	O	Data Out for the attached EEPROM. It connects to Data In pin of the EEPROM.
NC1... NC10	–	Test pins, must be left unconnected (floating).

Section 3: Register Details

The BCM5812 registers are divided into two categories.

1 PCI configuration registers implement control and status information that is specific to the PCI bus, as well as registers required by the PCI specification rev. 2.2.

2 DMA control and status registers pertain to master command, data, and command context fetch and write back operations.

Unused or reserved bits are initialized to zero. Unused bits should be written as zeroes. The following mnemonics are used to describe the types of access allowed for each register bit:

- RW – bit is read/write
- WO – bit is write only
- RO – read only bit (i.e. status flag)
- RSVD – reserved bit, ignore upon read, write 0 upon write

A value of X upon reset means that the state of the register is undefined and should not be relied upon after a reset occurs.

PCI CONFIGURATION REGISTERS

The BCM5812 provides PCI 2.2 compliant configuration space registers as shown in Table 18. In addition, the BCM5812 uses PCI Memory as configured in BAR0 for all slave control and status registers (CSRs). The registers use a total memory space of 64 KB in one memory BAR region. This region is non-prefetchable, and must be relocated only in 32-bit space.

Configuration registers that are not shown in Table 18 are reserved.

Table 18: PCI Configuration Registers

Offset	*31 bits 16*		*15 bits 00*	
0x00	Device ID		Vendor ID	
0x04	Status		Command	
0x08	Class code			Rev ID
0x0C	BIST	Header Type	Master Latency Timer	Cache line Size
0x10	Memory BAR0			
0x2C	Subsystem ID		Subsystem Vendor ID	
0x34	Reserved			Capabilities Pointer
0x3C	MAX_LAT	MIN_GNT	Interrupt Pin	Interrupt Line
0x40	Reserved		Retry Timeout	TRDY Timeout
0x48	Power Management Capabilities		Next Capability Pointer (End Of List)	Power Management Capability ID
0x4c	Reserved		Power Management Control/Status	

The various registers within PCI configuration space are shown in Table 19.

An optional serial EEPROM allows the following PCI register parameters to be modified:

- Vendor ID
- Device ID
- Subsystem ID
- Subsystem Vendor ID
- Class Code
- Revision ID
- Header Type
- Maximum Latency
- Minimum Grant
- Interrupt Pin

In addition, the power management register read-only fields can also be updated by the optional EEPROM. For more information, see Appendix C “EEPROM Information” on page 73.

Table 19: PCI Configuration Register Bit Fields

Bits	***Access***	***Reset***	***Purpose***
PCI Vendor ID – 0x00			
15:0	RO	0x14E4	Hardwired vendor identifier (0x14E4), Assigned by PCISIG for Broadcom.
PCI Device ID – 0x02			
31:16	RO	0x5823	Hardwired device identifier (0x5823)
PCI Command Register – 0x04			
15:10	RSVD	0x00	Reserved
9	RW	0	Fast back to back master enable
8	RW	0	System error enable
7	RSVD	0	Reserved
6	RW	0	Parity error enable
5	RSVD	0	Reserved
4	RW	0	Memory write and Invalidate enable
3	RSVD	0	Reserved
2	RW	0	Bus master enable
1	RW	0	Memory access enable
0	RO	0	I/O access enable
PCI Status Register – 0x04			
31	RO	0	Detect parity error
30	RO	0	Signaled system error
29	RO	0	Received master abort status
28	RO	0	Received target abort status
27	RO	0	Signaled target abort status
26:25	RO	01	Device select timing
24	RO	0	Data parity detected

Table 19: PCI Configuration Register Bit Fields (Cont.)

Bits	***Access***	***Reset***	***Purpose***
23	RO	1	Fast back to back capable status
22:23	RSVD	0	Reserved
20	RO	1	Capability list
19:16	RSVD	0x00	Reserved
PCI Rev ID – 0x08			
7:0	RO	0x01/0xE1	Hardwired device revision identifier (0x01 for domestic version and 0xE1 for export version)
PCI Class Code Register – 0x08			
31:8	RO	0x0B4000	Class code value (hardwired) – 0x0B4000 (processor class, coprocessor subclass)
BIST, Header, Master Latency, PCI Cache Line – 0x0C			
31	RO	0	BIST capable, no BIST capability
30	RW	0	BIST start, writing has no effect
29:24	RO	0x00	BIST register, Not supported in BCM5812. Default to 0x00.
23:16	RO	0x00	Header type = 0, single function
15:10	RW	0x0C	Master latency timer. The value in this register defines the maximum length of the current burst should the PCI arbiter logic remove the GNT signal from the BCM5812 device. This value is ignored as long as the GNT signal remains asserted to the BCM5812. Once GNT is removed, the latency timer limit is immediately applied. This register can be programmed with values from 0x00 to 0xfc. Only the six most significant bits are implemented (two LSBs are hardwired to 0). A value of zero would cause the BCM5812 to perform one data phase.
9:8	RO	0x00	
7:0	RW	0x00	Cache line size. The value in this register determines what types of PCI bus master read cycles are generated by the BCM5812 device. When the intended read burst is smaller than a full cacheline and is completely contained within a single cache line, a MEM_READ cycle is generated. When the intended read burst is exactly one full cacheline or crosses only one cacheline boundary, a MEM_READ_LINE cycle is generated. When the intended burst is exactly two full cachelines or crosses multiple cache line boundaries, a MEM_READ_MULTIPLE cycle is used. Setting this register to zero inhibits the BCM5812 from generating any MEM_READ_LINE or MEM_READ_MULTIPLE cycles.
PCI Memory BAR – 0x10			
31:16	RW	0xFFFF	Memory Base Address Register (upper), 64 KB region, non-prefetchable, relocate in 32-bit space only.
15:0	RO	0x0000	Memory Base Address Register (lower), 64 KB region, non-prefetchable, relocate in 32-bit space only.
Subsystem ID, Subsystem Vendor ID – 0x2C			
31:16	RO	0x0200 / 0x0500	Subsystem ID • BCM5812-200 = 0x0200 • BCM5812-500 = 0x0500
15:0	RO	0x14E4	Subsystem Vendor ID (Broadcom PCISIG ID)

Table 19: PCI Configuration Register Bit Fields (Cont.)

Bits	***Access***	***Reset***	***Purpose***
Capabilities Pointer – 0x34			
7:0	RO	0x48	Points to a linked list of new PCI capabilities (point to register 0x48 in the PCI space by default)
PCI MAX_LAT, MIN_GNT, Interrupt – 0x3C			
31:24	RO	0	PCI MAX_LAT parameter
23:16	RO	0	Length of burst period MIN_GNT
15:8	RO	0x1	Interrupt pin register
7:0	RW	0	Interrupt line register
PCI Retry Timeout, TRDY Timeout – 0x40			
15:8	RW	0x80	Retry times. The value in this register defines the number of consecutive retries addressing a particular memory address that the BCM5812 attempts before setting the DMAERR_INTR bit in the BCM5812 DMA status register. This register can be programmed with any value between 0x00 and 0xFF. A value of 0x00 disables this register, i.e. the BCM5812 allows an infinite number of retries.
7:0	RW	0x80	TRDY timeout. The value in this register defines the number of TRDY wait states that the BCM5812 allows before setting the DMAERR_INTR bit in the BCM5812 DMA status register. This value applies to the number of TRDY states encountered before the initial data phase occurs on the PCI bus. This register can be programmed with any value between 0x00 and 0xFF. A value of 0x00 disables this register, i.e. the BCM5812 allows an infinite amount of TRDY wait states.
Next Capabilities Pointer, Power Management Capabilities ID – 0x48			
15:8	RO	0x0	Next capabilities pointer. Default to 0x0 because it is end of capabilities list
7:0	RO	0x1	Power management capabilities ID.
Power Management Capabilities Register – 0x4A			
15:11	RO	0	PME support. BCM5812 does not support $\overline{\text{PME}}$ pin.
10	RO	0	Indicates whether the device supports D2 power management state. BCM5812 does not support this state.
9	RO	0	Indicates whether the device supports D1 power management state. BCM5812 does not support this state.
8:6	RO	0	Auxiliary Current. BCM5812 does not support auxiliary power supply.
5	RO	0	Device specification initialization. It is not necessary for 5823.
4	RSVD	0	Reserved
3	RO	0	Indicates that the device requires the presence of PCI clock for $\overline{\text{PME}}$ operation. BCM5812 does not support PME.
2:0	RO	0x2	Version of PCI Power Management Interface Spec supported. 0x2 means version 1.1 of the spec.
Power Management Control/Status (PMCSR) Register – 0x4C			
15	RO	0	PME Status.
14:13	RO	0	Data scaling factor used when interpreting the value of the data register. BCM5812 does not have data register.
12:9	RO	0	Indicates which data is to be reported via the data register. BCM5812 does not have data register.

Table 19: PCI Configuration Register Bit Fields (Cont.)

Bits	*Access*	*Reset*	*Purpose*
8	RO	0	Enables the device to generate $\overline{\text{PME}}$. BCM5812 does not support $\overline{\text{PME}}$ pin.
7:2	RSVD	0	Reserved
1:0	RW	0	Current power state. Can be set by writing to it. • 00 = D0 • 01 = D1 • 02 = D2 • 03 = D3

DMA CONTROL AND STATUS REGISTERS

The DMA registers control how master command structures, command context, and packet data are fetched and stored back after processing. All of the following registers are located in PCI memory starting at the address programmed by the host processor into BAR0.

Table 20: DMA Control and Status Register Summary

Offset	*31 bits 16*	*15 bits 00*
0x00	Master Command Record 1@	
0x04	DMA Control	
0x08	DMA Status	
0x0C	DMA Error Address	
0x10	Master Command Record 2@	

The various registers within the DMA control and status space are as follows.

Table 21: DMA Control and Status Registers

Bits	*Access*	*Reset*	*Purpose*
DMA Master Command Record 1@ – 0x00			
31:0	RW	X	Writing the address of a valid master command record to this register causes crypto/authentication processing of the packet descriptors within that record to begin. This register must only be written when the MCR_FULL bit of the DMA status register is 0. This register is double buffered, such that the MCR_FULL bit goes to zero very quickly after an initial write to this register. This allows the CPU to write a second MCR address value to this register, effectively queuing up to MCR structures for back to back processing with zero latency. Reset state is unknown. Do not write if PCI master mode is disabled.
DMA Control – 0x04			
31	RW	0	RESET – Software reset. Normally, it is zero. If software detects hanging or other undesirable states of BCM5812, it writes a one to this bit to reset the BCM5812. The BCM5812 can be used after 256 core-clock cycles.

Table 21: DMA Control and Status Registers (Cont.)

Bits	***Access***	***Reset***	***Purpose***
30	RW	0	MCR2INT_EN - Enable interrupt per MCR for MCR2@. An interrupt is generated every time an entire MCR completes processing. This is the preferred operational mode. Resets to 0.
29	RW	0	MCR1INT_EN - Enable interrupt per MCR for MCR1@. An interrupt is generated every time an entire MCR completes processing. This is the preferred operational mode. Resets to 0.
28	RSVD	0	Reserved
27	RW	1	LE_CRYPTONET - Software structures in DMA memory (as seen from the PCI bus) are in: • 1 = little endian format. • 0 = big endian format. BCM5812 uses this bit to correctly interpret the appropriate structures in DMA memory during the BCM5812 bus master operations. Optimally, this bit configuration should match the endianess of the host CPU.
26	RW	1	NORMAL_PCI: • 1 = Normal PCI mode. • 0 = Swapped PCI mode; all PCI bus master memory data is internally byte-swapped by the BCM5812. Used for testing. This bit should normally be set to 1.
25	RW	0	DMAERR_EN - Enable interrupt upon DMA master access error.
24	RSVD	0	Reserved
23	RW	0	RNG_MODE • 0 = 1 bit random number per one slow clock cycle. • 1 = 1 bit random number per two slow clock cycles
22	RW	0	Modulus Normalization • 0 = Modulus normalization is done by BCM5812. Software must not perform modulus normalization function. • 1 = Modulus normalization must be done by software. Provided for backwards compatibility. Should always be 0.
21:17	RSVD	0	Reserved
16	RW	0	DMA Master Write Burst Size Select • 0 = Master write burst size is 128 bytes. • 1 = Master write burst size is 240 bytes.
15:0	RSVD	0	Reserved
DMA Status – 0x08			
31	RO	0	Master access in progress. Resets to 0.
30	RO	0	MCR1_FULL flag = Master command address register is full. When this flag is 1, the CPU must not write to the MCR1@ register. When this flag is 0, the CPU may write a value to the MCR1@ register to request processing of a master command structure. Resets to 0.
29	RW	0	MCR1_INTR = Completion interrupt status of per-MCR interrupt for MCR1@. Cleared by writing a 1 to this bit position. This bit accurately reflects processing status even if the corresponding interrupt bit is disabled (in which case a PCI interrupt is not generated). This bit is sticky until cleared explicitly. Resets to 0.

Table 21: DMA Control and Status Registers (Cont.)

Bits	Access	Reset	Purpose
28	RW	0	DMAERR_INTR = Interrupt status for MCR DMA master access error. Sticky until explicitly cleared by writing a 1 to this bit position. This bit accurately reflects status even if the corresponding interrupt enable bit is off (in which case a PCI interrupt is not generated). Resets to 0.
27	RO	0	MCR2_FULL flag = Master command address register is full. When this flag is 1, the CPU must not write to the MCR2@ register. When this flag is 0, the CPU may write a value to the MCR2@ register to request processing of a master command structure. Resets to 0.
26	RW	0	MCR2_INTR = Completion interrupt status of per-MCR interrupt for MCR2@. Cleared by writing a 1 to this bit position. This bit accurately reflects processing status even if the corresponding interrupt bit is disabled (in which case a PCI interrupt is not generated). This bit is sticky until cleared explicitly. Resets to 0.
25	RO	0	MCR1_ALL_EMPTY = All the MCRs in the MCR1@ register are done. Cleared by writing a 1 to this bit position or by writing a 1 to MCR1_INTR bit position. This bit is sticky until cleared. Reset to 0.
24	RO	0	MCR2_ALL_EMPTY = All the MCRs in the MCR2@ register are done. Cleared by writing a 1 to this bit position or by writing a 1 to MCR2_INTR bit position. This bit is sticky until cleared. Reset to 0.
DMA Error Address – 0x0C			
31:2	RO	X	Address of master access that resulted in a PCI fault (32b word address). Reset state Unknown.
1	RO	X	• 0 = Faulted master access was a write. • 1 = Faulted master access was a read. Reset state Unknown.
DMA Master Command Record 2@ – 0x10			
31:0	RW	X	Writing the address of a valid master command record to this register causes key setup processing of the data within that record to begin. This register must only be written when the 'MCR_FULL' bit of the DMA status register is 0. This register is quadruple buffered because BCM5812 can process two key setup MCRs simultaneously, such that the MCR_FULL bit goes to zero very quickly after two initial writes to this register. This allows the CPU to write third and fourth MCR address values to this register, effectively queuing up four MCR structures for back to back processing with zero latency. Reset state is Unknown. Do not write if PCI master mode is disabled.

Section 4: Electrical and Timing Characteristics

Table 22: Electrical and Timing Specifications

Parameter	***Typical***	***Description***
PCI Compliance	3.3V and 5V	Over the range of 25-33 MHz PCI clocks
Supply Voltage	3.3V ± 5% 1.8V ± 5%	For I/O buffers For the core
I/O Buffers	3.3V	
Operating Temperature	0-70C	Within the commercial temperature range
DC Characteristics for the I/O Pins		Follows the PCI 2.2 DC specifications
The BCM5812 works in both 3.3V and 5V environments		

Table 23: PCI Pin Timing Specifications

Symbol	***Parameter***	***33 MHz***		***Units***
		Min	***Max***	
T_{VAL}	PCI_CLK to Signal Valid Delay - bussed signals	2	11	ns
T_{SU}	Input Setup Time to PCI_CLK - bussed signals	7		ns
$T_{SU(PTP)}$	Input Setup Time to PCI_CLK - point to point signals ($\overline{\text{REQ}}$ and $\overline{\text{GNT}}$)	10, 12		ns
T_{HD}	Input Hold Time from PCI_CLK	0		ns

BCM5812 PCI pins violate T_{HD} slightly (0.6 ns vs. 0 ns in the PIC Spec). T_{VAL}, T_{SU}, and $T_{SU(PTP)}$ are within the PCI timing specifications in the entire operating range.

Table 24: Power Consumption (BCM5812-200)

Parameter	***Max Power***	***Max Current***	***Max Voltage (Vnominal +5%)***
Peripheral	0.05W	14.4 mA	3.46V
Core	0.4W	211 mA	1.89V

Table 25: 196-Pin FBGA Package Thermal Parameters

Parameter	Theta$_{JA}$ (Junction to amb.)	Theta$_{JB}$ (Junction to board)	Theta$_{JC}$ (Junction to case)	Max Ambient Temperature
Assume in still air, no heat sink	33.75 C/W	16.40 C/W	6.5 C/W	70C

Section 5: Performance

SYSTEM THROUGHPUT

System throughput values for the BCM5812 are shown below in Table 26. System values represent measured, memory to memory, in-system throughput on an optimal platform using large buffer sizes and maximum pipelining.

Table 26: BCM5812 System Throughput

Function	***Value***
IPsec 3DES with HMAC-MD5-96 or HMAC-SHA-1-96	50 Mbps
IPsec AES 256-bit key with HMAC-MD5-96 or HMAC-SHA-1-96	50 Mbps
IPsec Null with HMAC-MD5-96	80 Mbps
IPsec Null with HMAC-SHA-1-96	70 Mbps
ARCFOUR	80 Mbps
SSL-MAC using MD5	80 Mbps
SSL-MAC using SHA-1	70 Mbps
TLS-HMAC-MD5	80 Mbps
TLS-HMAC-SHA-1	70 Mbps
Diffie-Hellman (1024-bit Modulus, 180-bit Exponent)	50 Generate+Shared per second
DSA Sign with 1024-bit public key, 160-bit private key	50 per second
DSA Verify with 1024-bit public key, 160-bit private key	30 per second
RSA Private Key CRT, 1024-bit modulus	65 per second
Random number generation	100 Kbps

PACKET PERFORMANCE

IPSEC

Table 28 shows in-system performance for the BCM5812, in Mbps, for 3DES encryption with HMAC-SHA-1 authentication and different data packet sizes. Performance was measured using contiguous data packets (i.e., one input fragment and one output fragment descriptor). Each MCR had 64 packet descriptors for packet sizes up to 1024 bytes, 16 packet descriptors for packet sizes 1400, and 2048, and 4 packet descriptors for the two largest sizes. Different keys (i.e., Security Associations) were used on a per packet basis. Performance is given for both inbound (decrypt) and outbound (encrypt) directions.

Table 27: BCM5812 3DES IPSec Performance by Packet SIze

PCI Width (bits)	*PCI Clock (MHz)*	*DIrection*	*Packet Sizes (Bytes)*									
			64	*128*	*256*	*304*	*512*	*1024*	*1400*	*2048*	*5120*	*10240*
32	33	Outbound	16	24	35	40	45	53	55	59	62	64
		Inbound	18	29	40	43	50	56	57	61	63	65

Table 28 shows the in-system performance for AES under the same conditions. Each MCR had 64 packet descriptors for packet sizes up to 512 bytes, 16 packet descriptors for packet sizes 1024, 1400, and 2048, and 4 packet descriptors for the two largest sizes.

Table 28: BCM5812 AES IPSec Performance by Packet SIze

PCI Width (bits)	*PCI Clock (MHz)*	*DIrection*	*Packet Sizes (Bytes)*									
			64	*128*	*256*	*304*	*512*	*1024*	*1400*	*2048*	*5120*	*10240*
32	33	Outbound	16	25	39	47	58	66	72	76	81	85
		Inbound	19	31	46	50	61	71	74	79	83	86

Section 6: Mechanical Information

Package Drawing

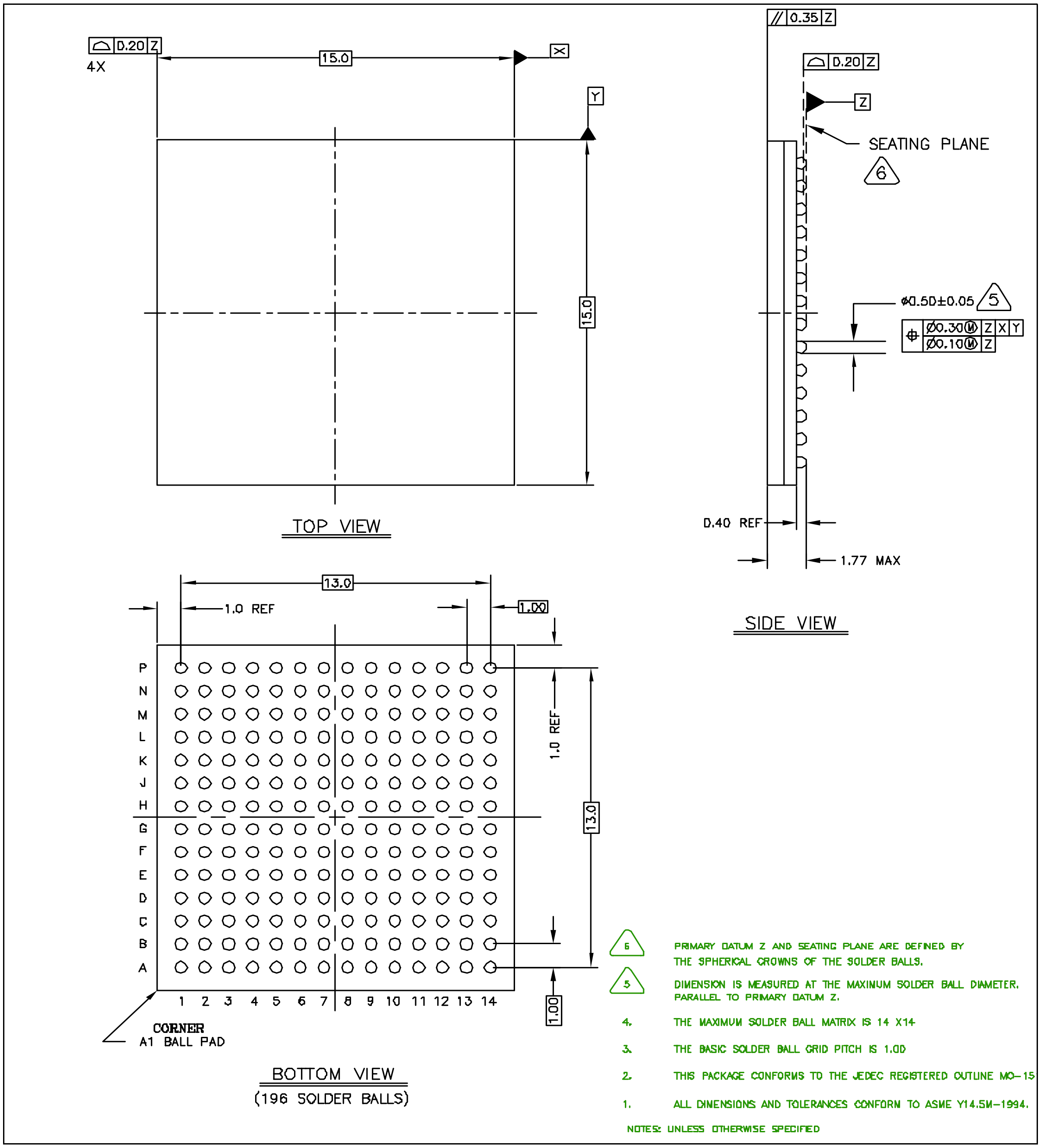

0.20 Z
4X
15.0
X
Y
15.0
TOP VIEW
0.35 Z
0.20 Z
Z
SEATING PLANE
6
⌀0.50±0.05
5
⌀0.30Ⓜ Z X Y
⌀0.10Ⓜ Z
0.40 REF
1.77 MAX
SIDE VIEW
13.0
1.0 REF
1.00
P
N
M
L
K
J
H
G
F
E
D
C
B
A
1.0 REF
13.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14
1.00
CORNER
A1 BALL PAD
BOTTOM VIEW
(196 SOLDER BALLS)
6 PRIMARY DATUM Z AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
5 DIMENSION IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER. PARALLEL TO PRIMARY DATUM Z.
4. THE MAXIMUM SOLDER BALL MATRIX IS 14 X14
3. THE BASIC SOLDER BALL GRID PITCH IS 1.00
2. THIS PACKAGE CONFORMS TO THE JEDEC REGISTERED OUTLINE MO-15
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ASME Y14.5M-1994.
NOTES: UNLESS OTHERWISE SPECIFIED

Figure 42: 196-pin FBGA Package Drawing

Section 7: Ordering Information

ORDERING INFORMATION

Table 29: BCM5812 Ordering Information

Marketing Part Number	Ordering Part Number	Package	Ambient Temperature
BCM5812	BCM5812KFB	196-pin FBGA	0° to 70° C

Appendix A: References

REFERENCED STANDARDS AND TEXTS

[1] Broadcom Corporation, Software Reference Library, BCM508x/BCM582x Security Processors, Document Number 580x_582x-SLR102-R

[2] S. Kent, R. Atkinson. RFC 2401 = Security Architecture for the Internet Protocol. November,1998.

[3] S. Kent, R. Atkinson. RFC 2402 = IP Authentication Header. November, 1998.

[4] C. Madson, R. Glenn, RFC 2403 = Use of HMAC-MD5-96 within ESP and AH. November,1998.

[5] C. Madson, R. Glenn, RFC 2404 = Use of HMAC-SHA-1-96 within ESP and AH. November,1998.

[6] C. Madson, N. Doraswamy, RFC 2405 = The ESP DES-CBC Cipher Algorithm with Explicit IV. November, 1998.

[7] S. Kent, R. Atkinson. RFC 2406 = IP Encapsulating Security Payload (ESP). November, 1998.

[8] R. Glenn, S. Kent, RFC 2410 = The NULL Encryption Algorithm and its use with IPsec. November, 1998.

[9] R. Periera, R. Adams, RFC 2451 = The ESP CBC-Mode CIpher Algorithms. November, 1998.

[10] H. Krawczyk, M. Bellare, R. Canetti, RFC 2104 = HMAC Keyed-Hashing for Message Authentication, February, 1997

[11] National Institute of Standards and Technology, FIPS PUB 46-3 = Data Encryption Standard. October 25, 1999

[12] National Institute of Standards and Technology, FIPS PUB 74 = Guidelines for Implementing and Using the NBS Data Encryption Standard. April 1, 1981

[13] National Institute of Standards and Technology, FIPS PUB 81 = DES Modes of Operation. December 1980

[14] National Institute of Standards and Technology, FIPS PUB 180-2 = Digital Signature Standard. April, 1995

[15] National Institute of Standards and Technology, FIPS PUB 197 = Advanced Encryption Standard. November 26, 2001

[16] M. Dworkin, National Institute of Standards and Technology Special Publication 800-38A = Recommendation for Block Cipher Modes of Operation, Methods and Techniques, December, 2001

[17] B. Schneier, Applied Cryptography, Protocols, Algorithms, and Source Code in C, Second Edition, John Wiley and Sons, 1996

[18] Dirks, T., and Allen, C., RFC2246 = The TLS Protocol, Version 1.0, January, 1999

[19] National Institute of Standards and Technology, FIPS PUB 186-2 = Digital Signature Standard. January, 2000

Appendix B: Programming Considerations

This Appendix summarizes Invalid Operation Conditions, Chaining Restrictions, and Alignment Restrictions mentioned in the text.

INVALID OPERATION CONDITIONS

This section summarizes the conditions that cause unpredictable results being written to memory, or possibly in a hang condition.

ZERO PACKET LENGTHS

IPsec operations should never be supplied a zero packet length parameter in the packet descriptor. The offset field in the command context should never be equal to or greater than the packet length.

ZERO FRAGMENT LENGTHS IN FRAGMENT CHAIN ENTRY DESCRIPTORS

All buffer lengths in input and output fragment chain entries that refer to actual data buffers must contain non-zero values.

ERRONEOUS PARAMETER SPECIFICATIONS

Situations such as illegal authentication specifiers, misaligned structure members, or misaligned output packet payload data, should be guaranteed to never occur.

OUTPUT FRAGMENT ADDRESSES FOR MISALIGNED BUFFERS

All output data should be aligned on 32-bit boundaries. See “Alignment Restrictions” on page 71.

OUTPUT FRAGMENT LENGTHS THAT ARE NOT A MULTIPLE OF 4

All output data buffers must have a length that is multiple of 32-bits. See “Alignment Restrictions” on page 71.

NON-ZERO OFFSET WITH NULL ENCRYPTION

The offset must be zero with NULL encryption specified.

NULL AUTHENTICATION WITH NULL ENCRYPTION

At least one of authentication or encryption must be specified.

Incorrect Data Size for Encryption

Whenever 3DES or AES is enabled, the length of input data to be encrypted or decrypted must be a multiple of the block size, which is eight bytes for 3DES and sixteen bytes for AES. The BCM5812 calculates this length as the packet length from the command context minus the number of bytes (specified in 32-bit words) in the offset field in the command context.

Writing to the MCR Register with PCI Master Mode Disabled

Doing so causes the BCM5812 control microcode to start processing, waiting for a PCI master mode access that never begins.

Modular Arithmetic Operation Restrictions

In modular exponentiation, the exponent length must be less than modulus length. In all modular arithmetic operations, the modulus cannot be less than 16 bits.

In all modular arithmetic operations other than remainder, the arguments must be less than the modulus. In remainder, the input parameter size must be equal to modulus parameter size. Thus, the input argument must be between the modulus and the maximum parameter size.

Chaining Operation Dependencies

BCM5812 can perform chaining operations (The output of the first operation feeds as the input to the second operation within the same MCR) for the most frequent SSL/TLS operations. However, it has limitation how the chaining operations can be done in the other SSL/TLS operations.

SSL-MAC or TLS-HMAC Followed by ARCFOUR

Figure 12 on page 14 shows the structure of SSL or TLS protocol records. For outgoing SSL or TLS record layer processing, which typically requires computing the authentication code over the clear text followed by encryption of the data with the authentication code appended, the BCM5812 allows authentication and encryption to be performed in back to back packet descriptors in the same MCR.

In other words, it is safe to use the SSL-MAC or TLS-HMAC hash output as the data input to ARCFOUR. The authentication code may need its own fragment chain entry to assure compliance with alignment of the authentication code on a 32-bit boundary (see “Alignment Restrictions” on page 71 below).

ARCFOUR or 3DES Followed by SSL-MAC or TLS-HMAC

Incoming SSL or TLS record layer processing typically requires decryption followed by computing the authentication code over the clear text. For the BCM5812, decryption followed by authentication of the decrypted data can be performed in back to back packet descriptors in the same MCR. In other words, it is safe to have the ARCFOUR or 3DES data output as input to the SSL-MAC or TLS-HMAC.

MD5 or SHA-1 Followed by MD5 or SHA-1

SSL or TLS master key and connection key derivations require multiple successive hash operations, with the output of some operations feeding into the input of the next. In general, the BCM5812 should not be used with chained authentication operations where the output of one hash operation is used as input by the immediately following operation in the packet descriptor list. This applies to any of MD5 Hash, SHA-1 Hash, SSL-MAC, or TLS-HMAC followed by MD5 Hash, SHA-1 Hash, SSL-MAC, or TLS-HMAC.

There are three recommendations:

- Interleave an independent operation between two dependent operations. For the SSL key derivation example, perform the SHA-1 operations first, followed by the MD5 operations. If three operations are required to generate 48 bits of key material, the MD5 input would use the SHA-1 output from three operations prior.
- Put the dependent operations in separate MCRs.
- Assure that the output of the first operation is not the first 64 bytes input by the next operation. At an operation transition, the BCM5812 may prefetch up to 64 bytes of input while it stages completion of the previous operation.

3DES/HMAC Followed by 3DES/HMAC

The authentication function performed by 3DES/HMAC-MD5 or SHA-1 can be used for TLS key derivation with the 3DES option bit zero in the flags word in the command context. In this case, the same apply comments as under “MD5 or SHA-1 Followed by MD5 or SHA-1” on page 71.

SSL-MAC or TLS-HMAC Followed by SSL-3DES

In case of a TLS-MAC or SSL-MAC operation followed by an SSL-3DES operation, the chip starts prefetching the input data (64 bytes) for the 3DES before it writes out the hash of the first operation. The input data read of the second operation is suspended as the BCM5812 writes out the hash of the first operation. It then comes back to complete the data input read for SSL-3DES.

In this case, the same suggestions as in “MD5 or SHA-1 Followed by MD5 or SHA-1” on page 71 should be applied.

Alignment Restrictions

Table 30 shows alignment requirements for all memory-resident data in IPsec, SSL, and TLS encryption and authentication operations.

The flexibility with respect to input packet payload data allows extreme combinations to be supported. For instance, a packet with 16,000 bytes of input payload data could be described as a chain of 16,000 descriptors, with each descriptor holding one single byte. The BCM5812 handles such an extreme situation correctly from a functional standpoint, albeit with reduced performance from the huge number of descriptor fetches.

For ARCFOUR encryption, data can be any length, not necessarily multiple of 32-bit words. In this case, the last word of an output data buffer may contain one, two, three, or four bytes of actual ARCFOUR data. The non-ARCFOUR data in the word could be anything and should be ignored by software. The output fragment length for the data buffer should indicate the actual ARCFOUR data length. It may not be multiple of 4 bytes.

Table 30: Alignment Restrictions for IPSec/SSL/TLS Crypto/Authentication Operations

Memory-Resident Data Type	***Alignment Requirement,***	***Size Requirement***
Packet Payload Data		
Input Data Buffers (per Chain Entry)	None (byte)	None (byte)
Output Data Buffers (per Chain Entry)	32-bit	Multiple of 32 bits
Control and Command Structures		
Chain Entry descriptors (Input and Output)	32-bit	Fixed (3 words of 32 bits)
Command Context Structures	32-bit	Fixed (depends on operation) 64-bytes minimum is read
Master Command Record Structures	32-bit	(1 + #pkts∗8) 32-bit words)

Table 31 shows alignment requirements for all memory-resident data in DH/RSA/DSA/modular arithmetic operations.

Table 31: Alignment Restrictions for DH/RSA/DSA/Modular Arithmetic Operations

Memory-Resident Data Type	***Alignment Requirement***	***Size Requirement***
Packet Payload Data		
Input Data Buffers (per Chain Entry)	32-bit	Multiple of 32 bits
Output Data Buffers (per Chain Entry)	32-bit	Multiple of 32 bits
Control and Command Structures		
Chain Entry descriptors (Input and Output)	32-bit	Fixed (3 words of 32 bits)
Command Context Structure	32-bit	Fixed (depends on operation) 64-bytes minimum is read
Master Command Record	32-bit	(1 + #pkts∗8) 32-bit words)

Appendix C: EEPROM Information

DESCRIPTION

The BCM5812 optional serial EEPROM should be an industry standard type 93C46. The part should be configured for 16-bit operation and connected as shown in Figure 43. The EEPROM must be externally programmed, as the BCM5812 does not support write or erase cycles.

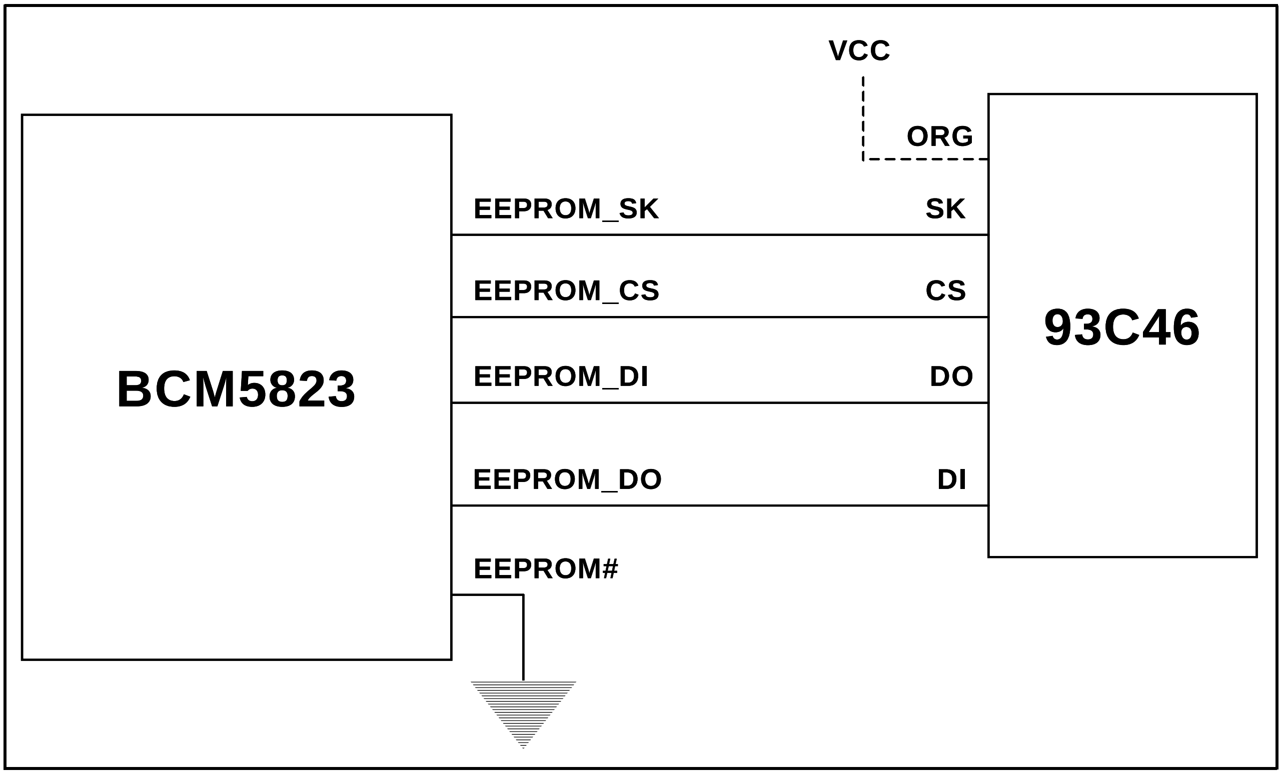


Figure 43: EEPROM Connection

PROGRAMMING

Table 32 shows the EEPROM locations accessed by the BCM5812 and what information should be programmed into each. For reference, the equivalent default value for that location, which the BCM5812 supplies if no EEPROM is present, is also listed. All other locations should be programmed to 0xFF.

Although the BCM5812 makes 16-bit accesses, the EEPROM contents are shown in Table 32 with their respective byte offset, as though for byte programming.

Table 32: EEPROM Programming

EEPROM Address (Byte, in Hex)	***Field***	***PCI Register Offset (bytes)***	***Field Length (bits)***	***Default Value (Hex)***
0x00	Vendor ID (low byte)	0x00	16	0xE4
0x01	Vendor ID (high byte)	0x01		0x14
0x02	Device ID (low byte)	0x02	16	0x23
0x03	Device ID (high byte)	0x03		0x58
0x08	Revision ID	0x08	8	0x01[a]
0x09	Class Code (low byte)	0x09	8	0x00
0x0A	Class Code (middle byte)	0x0A	16	0x40
0x0B	Class Code (high byte)	0x0B		0x0B
0x0C	Header Type	0x0E	8	0x00
0x0D	BIST Capable	0x0F	8	0x00
0x2C	Subsystem Vendor ID (low byte)	0x2C	16	0xE4
0x2D	Subsystem Vendor ID (high byte)	0x2D		0x14
0x2E	Subsystem ID (low byte)	0x2E	16	0x00
0x2F	Subsystem ID (high byte)	0x2F		0x05[b]
0x34	Capabilities Pointer	0x34	8	0x48
0x3D	Interrupt Pin	0x3D	8	0x01
0x3E	Maximum Latency	0x3E	8	0x00
0x3F	Minimum Grant	0x3F	8	0x00
0x48	Power Management Capability ID	0x48	8	0x00
0x49	Next Capability Pointer (EOL)	0x49	8	0x01
0x4A	Power Management Capability (low byte)	0x4A	16	0x02
0x4B	Power Management Capability (high byte)	0x4B		0x00
0x4C	Power Management Control/Status (low byte)	0x4C	16	0x00
0x4D	Power Management Control/Status (high byte)	0x4D		0x00

[a] EXPORT pulled down or open (default). If EXPORT is pulled high, this value is 0xE1

[b] This value is for the BCM5812-500, the BCM5812-200 reports 0x05

If the EEPROM is present, the BCM5812 unconditionally uses the value at the specified address to replace the indicated default PCI configuration value.

Broadcom Corporation

16215 Alton Parkway
P.O. Box 57013
Irvine, CA 92619-7013
Phone: 949-450-8700
Fax: 949-450-8710

Broadcom® Corporation reserves the right to make changes without further notice to any products or data herein to improve reliability, function, or design. Information furnished by Broadcom Corporation is believed to be accurate and reliable. However, Broadcom Corporation does not assume any liability arising out of the application or use of this information, nor the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.