BROADCOM®

DATA SHEET

BCM5820

eCommerce Processor

GENERAL DESCRIPTION

The BCM5820 is a high performance security processor that provides multi-protocol cryptographic acceleration for VPN and eCommerce applications.

The BCM5820 supports all of the symmetric and asymmetric encryption and authentication algorithms popularly used by IPsec and IKE, including 3DES, RSA, DSA, Diffie-Hellman, SHA-1, MD5, HMAC-SHA-1, and HMAC-MD5. For IPsec ESP processing, the BCM5820 performs single pass encryption combined with HMAC authentication. For SSL and TLS record processing, the BCM5820 computes the SSL MAC and TLS HMAC. The BCM5820 includes a True Random Number Generator.

The BCM5820 offers a complete, single-chip, solution that connects directly to the PCI bus with no need for external interface logic or memory.

The BCM5820 utilizes 0.22 µm semiconductor technology, a high performance bus master interface, and efficient software control structures in an advanced design that supports 800 1024-bit RSA Private Key Operations per second and over 290 Mbps sustained in-system bulk throughput for IPsec with 3DES and HMAC-SHA-1.

FEATURES

- Single pass IPsec combined encryption and authentication
- SSL MAC, TLS HMAC for SSL, TLS Record Layer processing
- Export mode, enables Retail export classification
- IETF and FIPS compliant algorithms:
  - 3DES-CBC. DES-CBC
  - 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 32/64 bit 33/66 MHz bus interface
- Advanced Testability Features
  - 100% testability of on-chip RAM cells via BIST
  - JTAG boundary scan for board level testing
- 5V tolerant, 3.3V PCI I/O
- 256 TBGA package
- Extensive Software Support
  - Drivers for Linux, VxWorks, Solaris, Win2K
  - OpenSSL (0.9.6c), Win2K ModExpo Offload
  - Full SDK with Application Library and Diagnostics

APPLICATIONS

- Secure Web Servers
- SSL Proxy Servers
- VPN Appliances
- VPN Firewalls, Routers and Switches
- Secure Wireless Gateways
- Secure Storage Servers

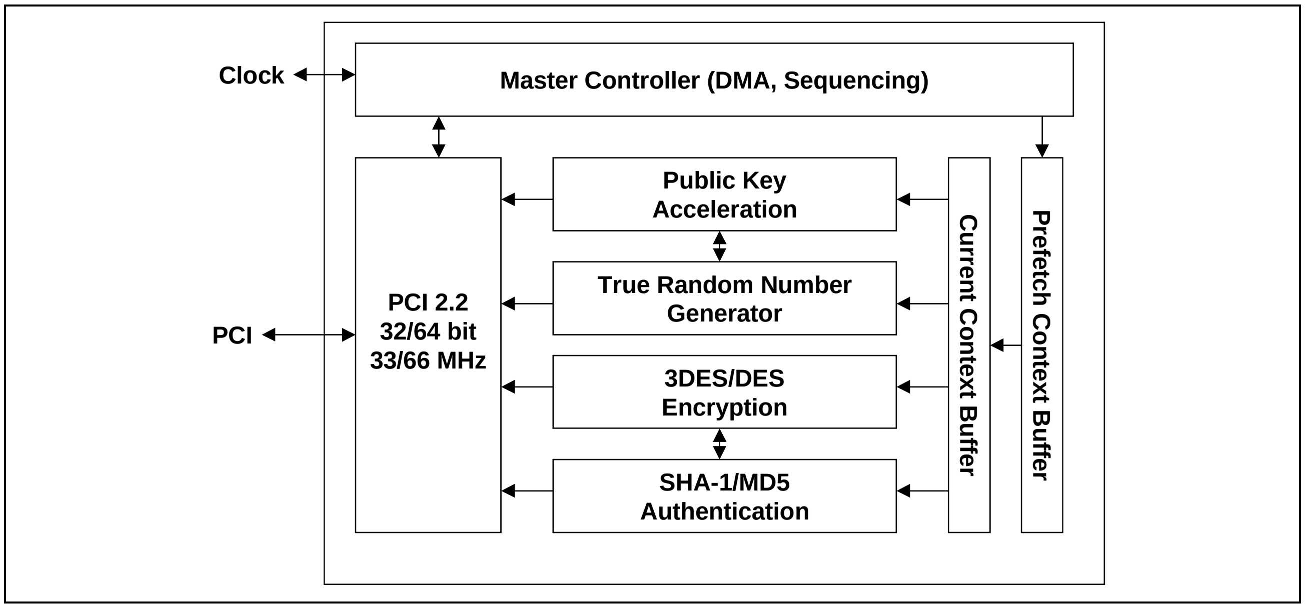


Figure 1: Functional Block Diagram

Revision History

Revision	*Date*	*Change Description*
5820-DS04-R	11/25/02	New Functional Description and Programming section and appendices.
5820-DS03-R	08/10/01	Minor editorial changes. Updated the frequency/performance/power numbers. Added Table 19 on page 51 and Table 20 on page 52.
5820-DS02-R	03/02/01	Updated the frequency/performance/power numbers Added Table 38 and 39 Correct errors.
5820-DS01-R	10/31/00	Inserted new section: Section 2 ”Ballout Assignments” on page 33 Incorporated minor editorial changes.
5820-DS00-R	6/30/00	Initial Release

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

Table of Contents

Section 1: Functional Description 1

Overview 1

Programming Interface 1

Cryptographic Operations 5

IPsec 3DES 6

SSL MAC 12

TLS HMAC 15

SSL/TLS DES/3DES 17

Pure MD5/SHA-1 HASH 18

Diffie-Hellman 19

RSA 22

DSA 24

Random Number Generation 27

Modular Arithmetic 28

Interrupt Processing 32

Export Control 32

Section 2: Ballout Assignments 33

Ballout by Ball Number 34

Ballout by Signal Name 38

Signal Definitions 42

Section 3: Register Details 44

PCI Configuration Registers 45

DMA Control and Status Registers 49

Section 4: Electrical & Timing Specifications 51

General Specifications 51

Section 5: Performance 53

System Throughput 53

Packet Performance 54

IPsec 54

Section 6: Mechanical Information 55

Section 7: Ordering Information 56

Appendix A: References 57

Referenced Standards and Texts 57

Appendix B: Programming Considerations 58

Invalid Operation Conditions 58

Zero Packet Lengths 58

Zero Fragment Lengths in Fragment Chain Entry Descriptors 58

Erroneous Parameter Specifications 58

Output Fragment Addresses for Misaligned Buffers 58

Output Fragment Lengths that are not a Multiple of 4 58

Non-zero Offset with NULL Encryption 58

NULL Authentication with NULL Encryption 58

Incorrect Data Size for Encryption 59

Writing to the MCR Register with PCI Master Mode Disabled 59

Modular Arithmetic Operation Restrictions 59

Chaining Operation Dependencies 59

MD5 or SHA-1 Followed by MD5 or SHA-1 59

3DES/HMAC Followed by 3DES/HMAC 60

SSL-MAC or TLS-HMAC Followed by SSL-3DES 60

Alignment Restrictions 61

List of Figures

Figure 1: Functional Block Diagram i
Figure 2: BCM5820 Programming Overview 2
Figure 3: Master Command Record Format 3
Figure 4: Packet Descriptor Format 3
Figure 5: IPsec DES/3DES Command Context 6
Figure 6: Computed HMAC Inner and Outer State 7
Figure 7: CBC Mode 9
Figure 8: Packet Description for IPsec DES/3DES Encryption and Authentication 10
Figure 9: IPsec Packet Description Showing Input Fragments 10
Figure 10: IPsec Packet Description Showing Output Fragments 11
Figure 11: SSL Record Structure 12
Figure 12: SSL MAC Computation Using MD5 12
Figure 13: SSL MAC Computation Using SHA-1 13
Figure 14: SSL MAC Command Context 14
Figure 15: SSL MAC Authentication Output 14
Figure 16: TLS HMAC Computation 15
Figure 17: TLS HMAC Command Context 16
Figure 18: SSL/TLS DES/3DES Command Context 17
Figure 19: Pure MD5/SHA-1 Hash Command Context 18
Figure 20: Diffie-Hellman Public Key Generate 20
Figure 21: Diffie-Hellman Secret Key 21
Figure 22: RSA Public Key 22
Figure 23: RSA Private Key 23
Figure 24: DSA Sign 25
Figure 25: DSA Verify 26
Figure 26: Random Number Generation Command Context 27
Figure 27: Modular Add 28
Figure 28: Modular Subtract 29
Figure 29: Modular Multiply 29
Figure 30: Modular Remainder 30
Figure 31: Modular Exponentiation 30
Figure 32: 256-Pin TBGA Pinout Diagram 33

Figure 33: 256-pin TBGA Package Drawing 55

List of Tables

Table 1: MCR Header Word (Input).......... 2
Table 2: MCR Header Word (Output).......... 4
Table 3: Operation Types.......... 5
Table 4: Flags.......... 7
Table 5: SSL MAC Authentication Codes.......... 13
Table 6: Pure MD5/SHA-1 Hash Command Context Flags.......... 18
Table 7: Allowed Public Key Parameter Field Size Increments.......... 19
Table 8: Allowed RSA Private Key Parameter Size Increments.......... 23
Table 9: Allowed Modular Arithmetic Parameter Field Size Increments.......... 28
Table 10: Modular Arithmetic Opcodes 31
Table 11: Ballout by Ball Number.......... 34
Table 12: Ballout by Signal Name 38
Table 13: Signal Definitions.......... 42
Table 14: PCI Configuration Registers.......... 45
Table 15: PCI Configuration Register Bit Fields 46
Table 16: DMA Control and Status Register Summary.......... 49
Table 17: DMA Control and Status Registers.......... 49
Table 18: Operating Conditions.......... 51
Table 19: PCI Pin DC Specifications 51
Table 20: PCI Pin Timing Specifications 52
Table 21: Power Consumption 52
Table 22: 256-pin TBGA Package Thermal Parameters.......... 52
Table 23: External Core Clock AC Specifications 52
Table 24: BCM5820 System Throughput 53
Table 25: BCM5820 IPSec Performance by Packet SIze 54
Table 26: Alignment Restrictions for IPSec/SSL/TLS Crypto/Authentication Operations.......... 61
Table 27: Alignment Restrictions for DH/RSA/DSA/Modular Arithmetic Operations 61

Section 1: Functional Description

OVERVIEW

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

The BCM5820 provides bulk cryptographic acceleration for the 3DES and DES with 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 BCM5820 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 BCM5820 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 BCM5820. These share a common bus interface, but are otherwise independent. The BCM5820 can thus complete multiple bulk encryption operations while simultaneously executing relatively slower public key operations, without having to stall bulk processing. In addition, the BCM5820 can execute up to eight public key operations in parallel.

The BCM5820 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.

PROGRAMMING INTERFACE

The BCM5820 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 BCM5820 using DMA, as diagrammed in Figure 2.

The BCM5820 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 BCM5820 PCI Configuration register BAR0. Host software accesses the BCM5820 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 Structure consists of a Header Word followed by an array of one or more Packet Descriptors. A single MCR Structure 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 Structure address to the BCM5820.

The host software gives an MCR to the BCM5820 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 BCM5820 provides a two level FIFO behind

MCR1@ and MCR2@. This permits the host software to pipeline MCR addresses in order to fully utilize the device. 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.

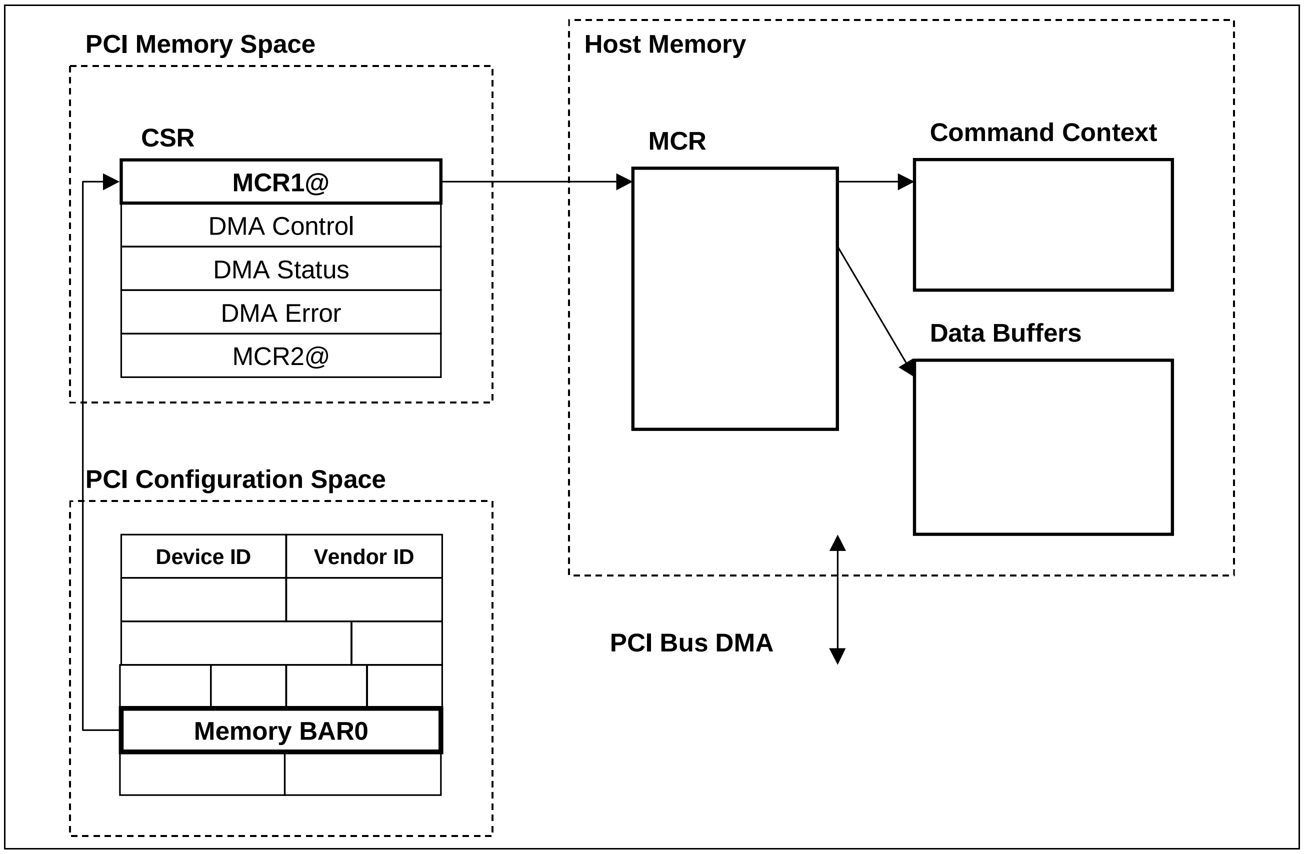


Figure 2: BCM5820 Programming Overview

Table 1: MCR Header Word (Input)

Bits	***Description***
31:16	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 3 diagrams the format of a Packet Descriptor.

The Command Context indicates the 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 BCM5820 interprets the fragment lists. The operations performed by the BCM5820, 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.

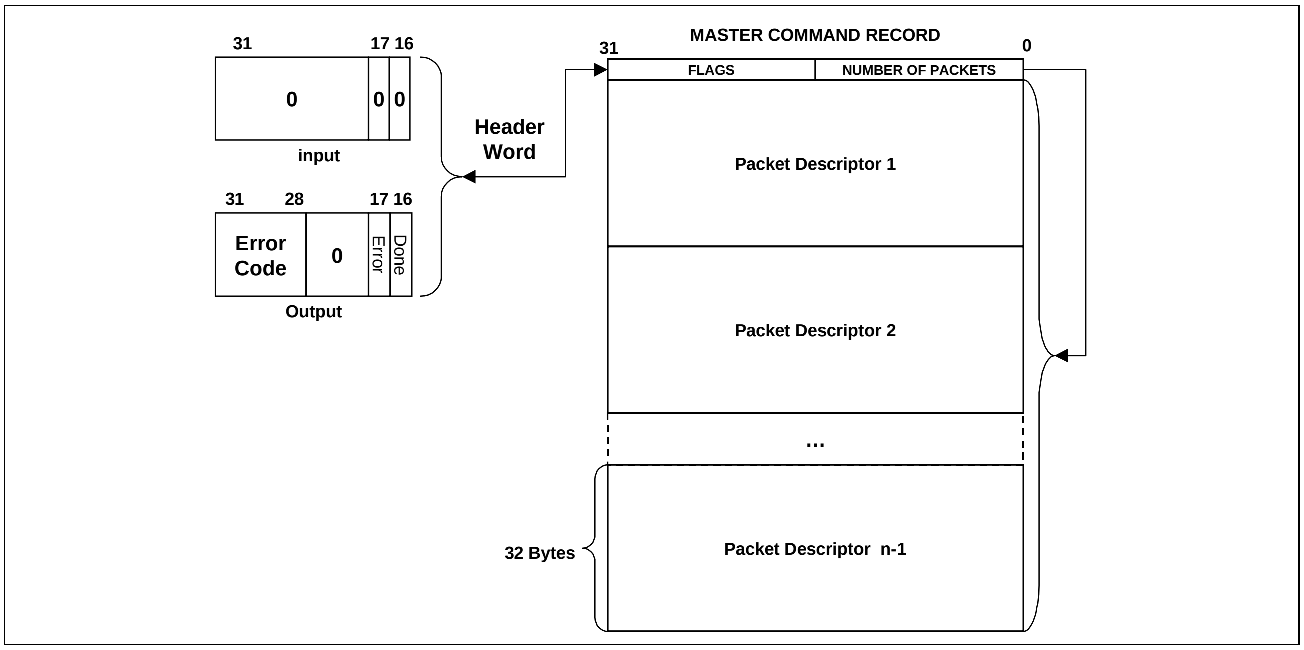


Figure 3: Master Command Record Format

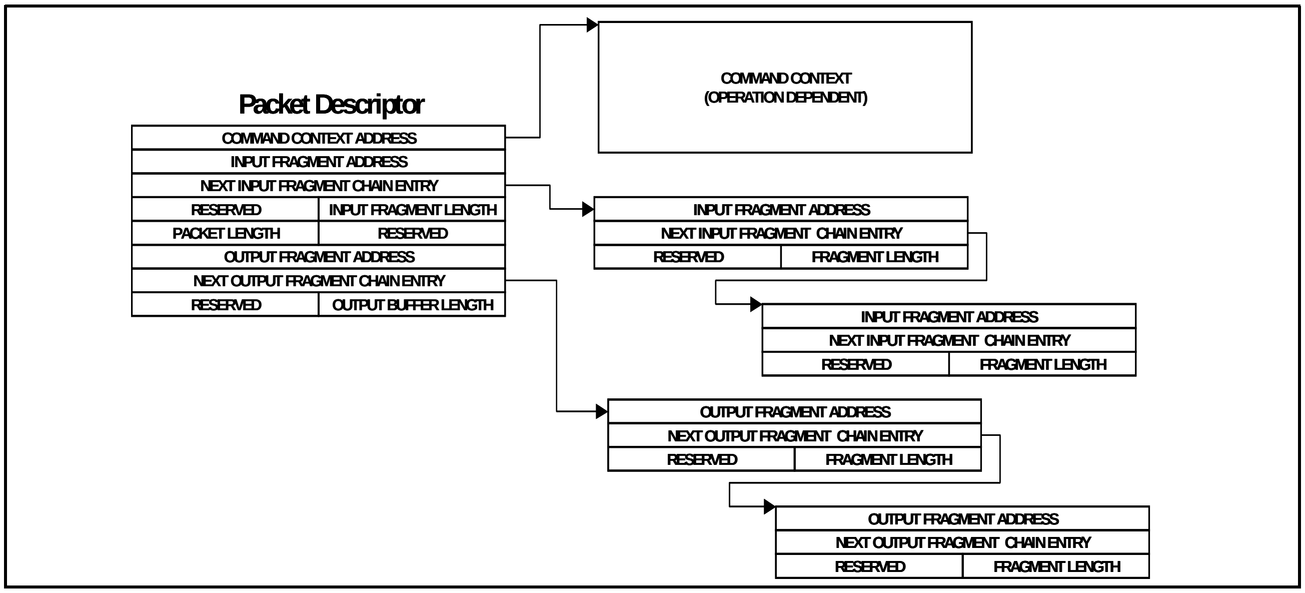


Figure 4: Packet Descriptor Format

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 59).

After it completes processing on all of the Packet Descriptors in the MCR Structure, the BCM5820 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. The BCM5820 can be configured to generate an interrupt after it completes processing an MCR Structure (see "Interrupt Processing" on page 32).

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
15:0	Number of Packets (unchanged from input)

Cryptographic Operations

Each cryptographic operation performed by the BCM5820 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 BCM5820. 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)*
0x0000	IPsec 3DES/DES combined encryption with authentication	Reserved
0x0001	SSL MAC authenticator calculation	Diffie-Hellman public key generation
0x0002	TLS HMAC authenticator calculation	Diffie-Hellman shared key generation
0x0003	3DES/DES encryption (for SSL and TLS)	RSA public key operation
0x0004	Reserved	RSA secret key operation
0x0005	Hash (Pure MD5 or SHA-1)	DSA signing operation
0x0006	Reserved	DSA verification operation
0x0040	Reserved	Reserved
0x0041	Reserved	RNG direct
0x0042	Reserved	RNG SHA-1
0x0043	Reserved	Modular Addition
0x0044	Reserved	Modular Subtraction
0x0045	Reserved	Modular Multiplication
0x0046	Reserved	Modular Reduction
0x0047	Reserved	Modular Exponentiation
0x0048	Reserved	Reserved
0x0049	Reserved	Reserved

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.

Note

- **An MCR must contain at least one Packet Descriptor.**
- **The minimum Command Context length actually read by the BCM5820 is 64 bytes.**

IPSEC 3DES

The BCM5820 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 [15].

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 7 diagrams the computation for the HMAC Inner and Outer State.[1] Note that 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 four bytes should be set to zeros.

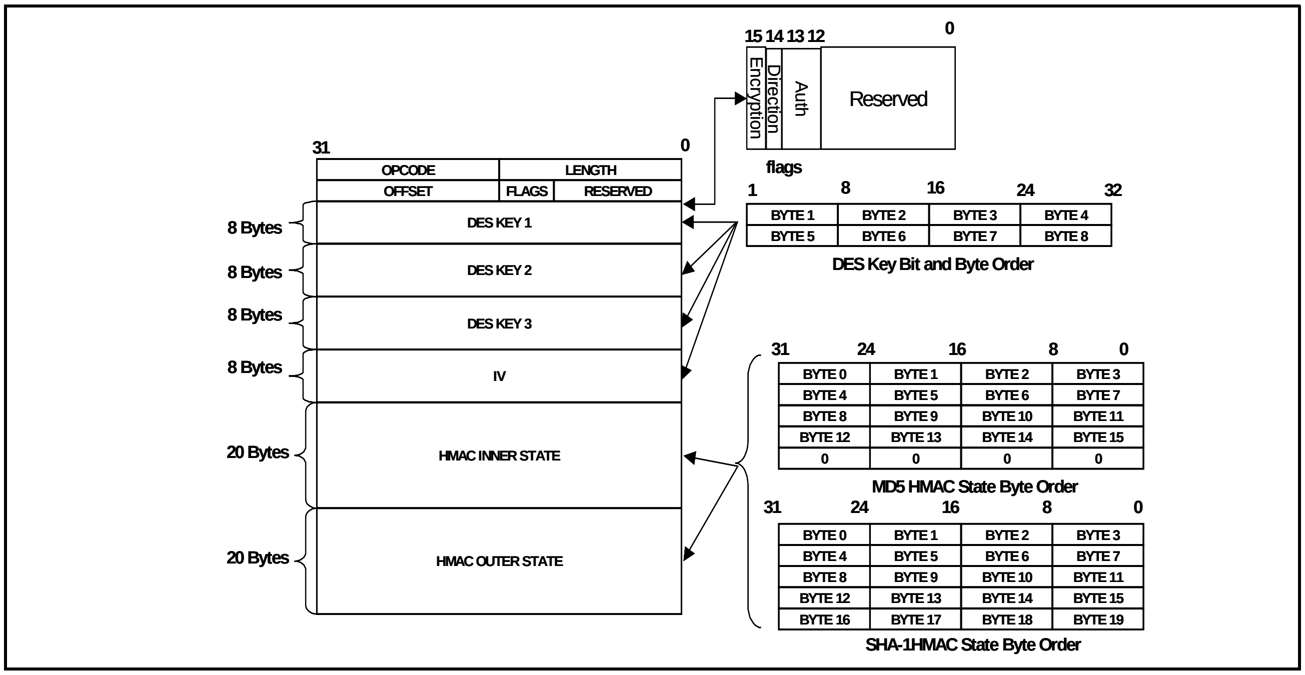

15 14 13 12
0
Encryption
Direction
Auth
Reserved
flags
31
0
OPCODE
LENGTH
OFFSET
FLAGS
RESERVED
8 Bytes
DES KEY 1
8 Bytes
DES KEY 2
8 Bytes
DES KEY 3
8 Bytes
IV
20 Bytes
HMAC INNER STATE
20 Bytes
HMAC OUTER STATE
1 8 16 24 32
BYTE 1 BYTE 2 BYTE 3 BYTE 4
BYTE 5 BYTE 6 BYTE 7 BYTE 8
DES Key Bit and Byte 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

1. Broadcom supplies a software routine in the Software Reference Library for computing HMAC Inner and Outer states.

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.

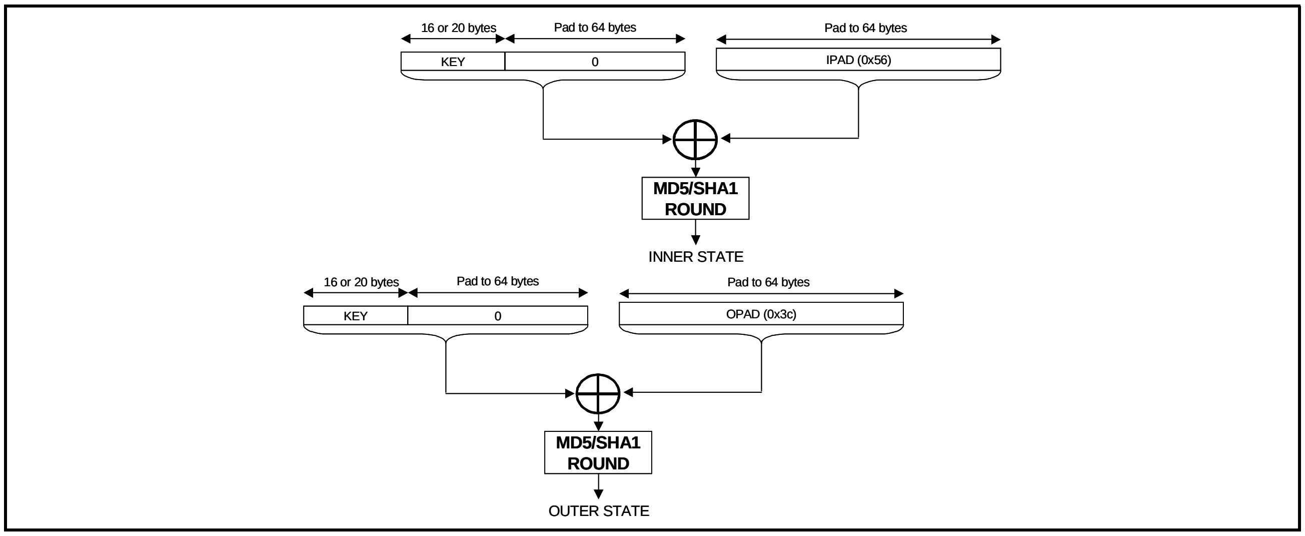


Figure 6: Computed HMAC Inner and Outer State

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 10 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.[2]

Figure 9 on page 10 shows an example of a logically contiguous input packet that is physically discontinuous in host memory, requiring the BCM5820 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 BCM5820 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).

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

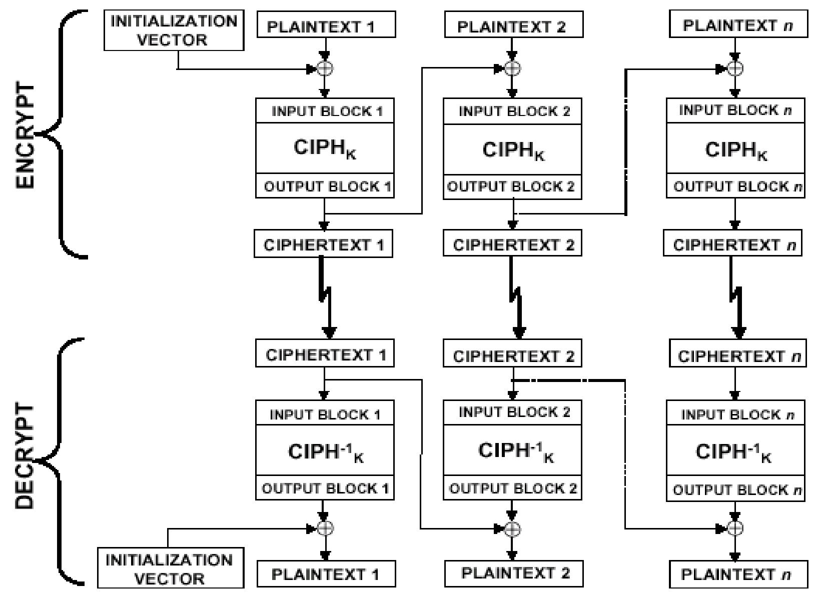


Figure 7: CBC Mode

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., 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 10 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 BCM5820 can be configured to interpret these “network bytes” in big-endian host CPU byte order. For 3DES, the Initialization Vector (IV) length is 8 bytes.

Figure 10 on page 11 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 BCM5820 to 12 bytes as used by HMAC-MD5-96 (RFC 2405) or HMAC-SHA1-96 (RFC2404).

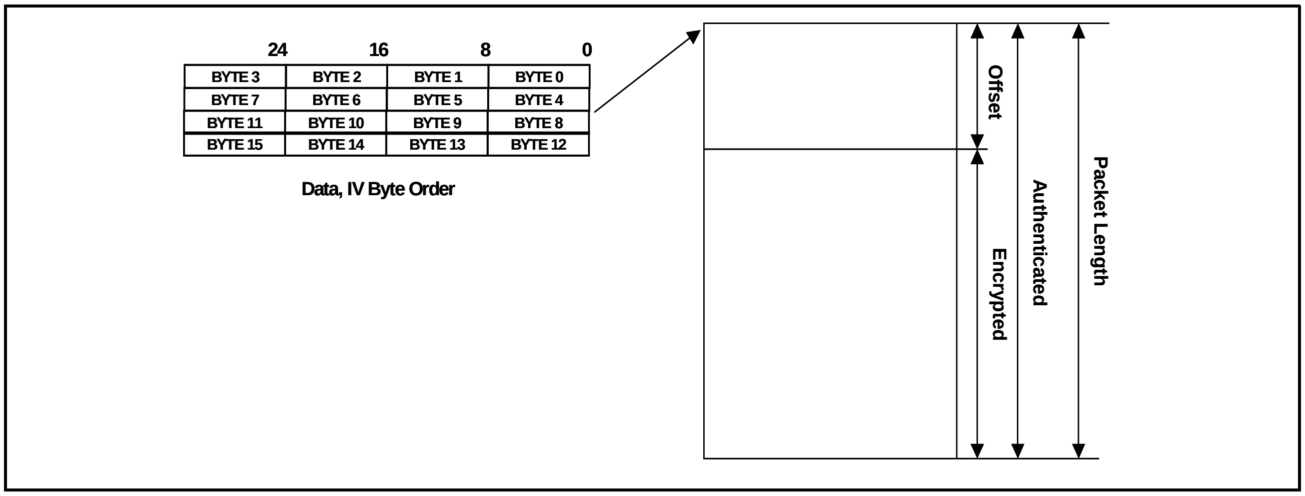


Figure 8: Packet Description for IPsec DES/3DES Encryption and Authentication

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 BCM5820 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 BCM5820 ignores the Next Output Fragment Chain Entry pointer in the last Output Fragment Chain Entry.

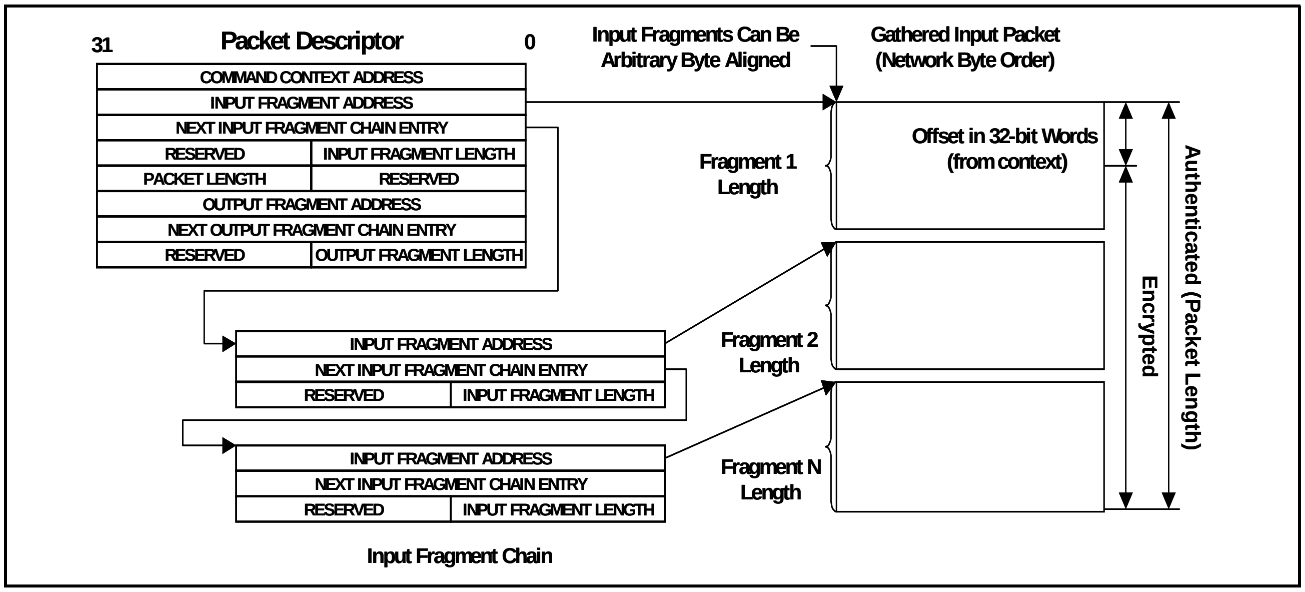


Figure 9: IPsec Packet Description Showing Input Fragments

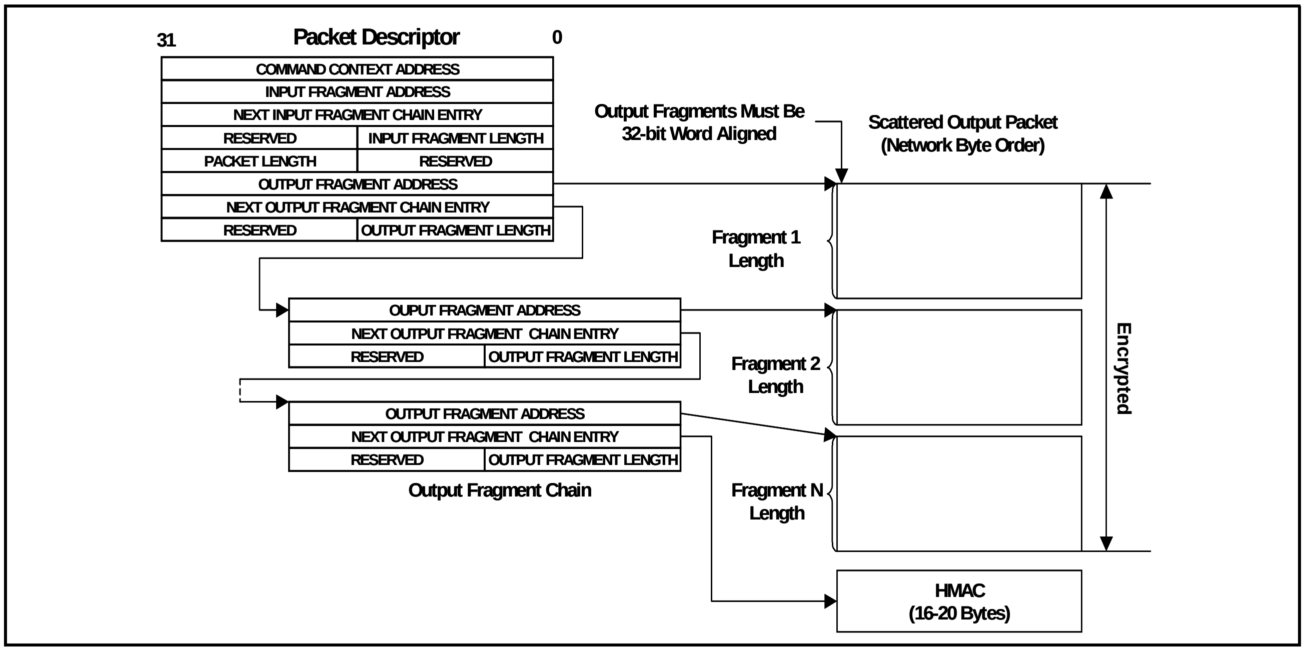


Figure 10: IPsec Packet Description Showing Output Fragments

SSL MAC

The BCM5820 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, one 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.

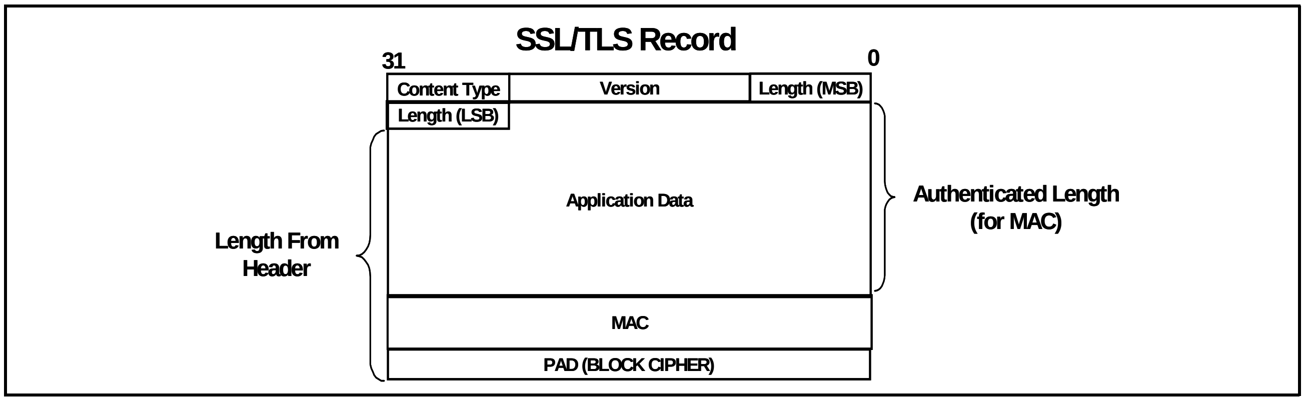


Figure 11: SSL Record Structure

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.

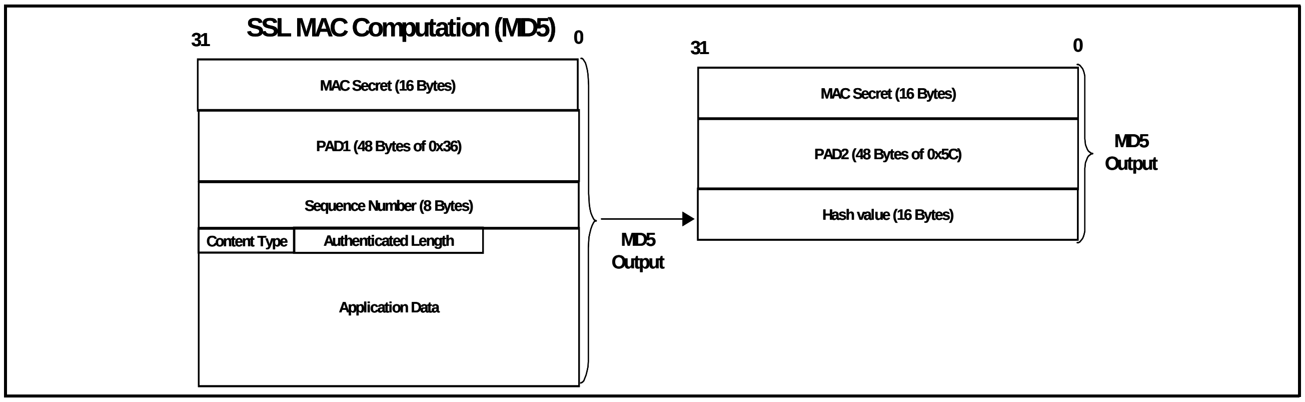


Figure 12: SSL MAC Computation Using MD5

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.

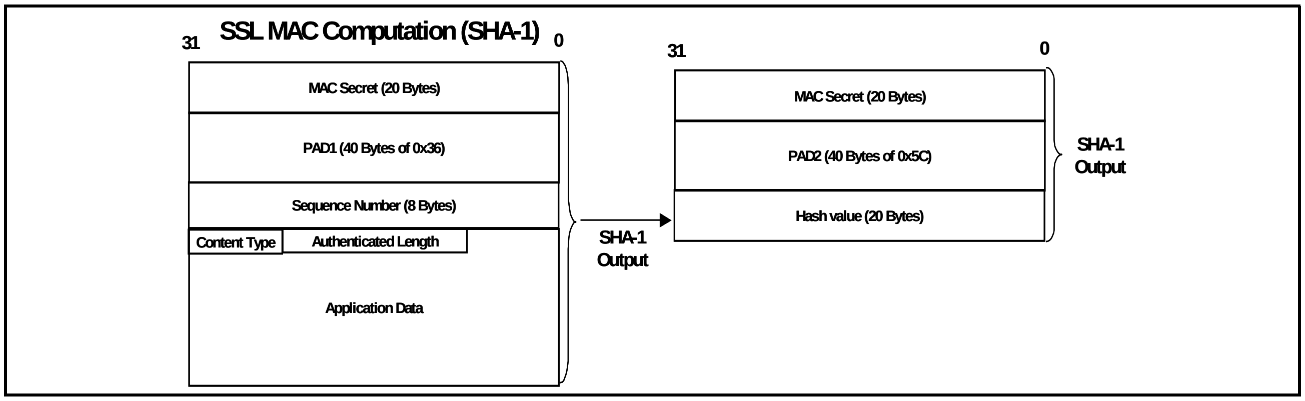


Figure 13: SSL MAC Computation Using SHA-1

Figure 14 on page 14 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. Note that the byte ordering of the HMAC secret is the same as that used for the HMAC Inner and Outer States for IPsec.

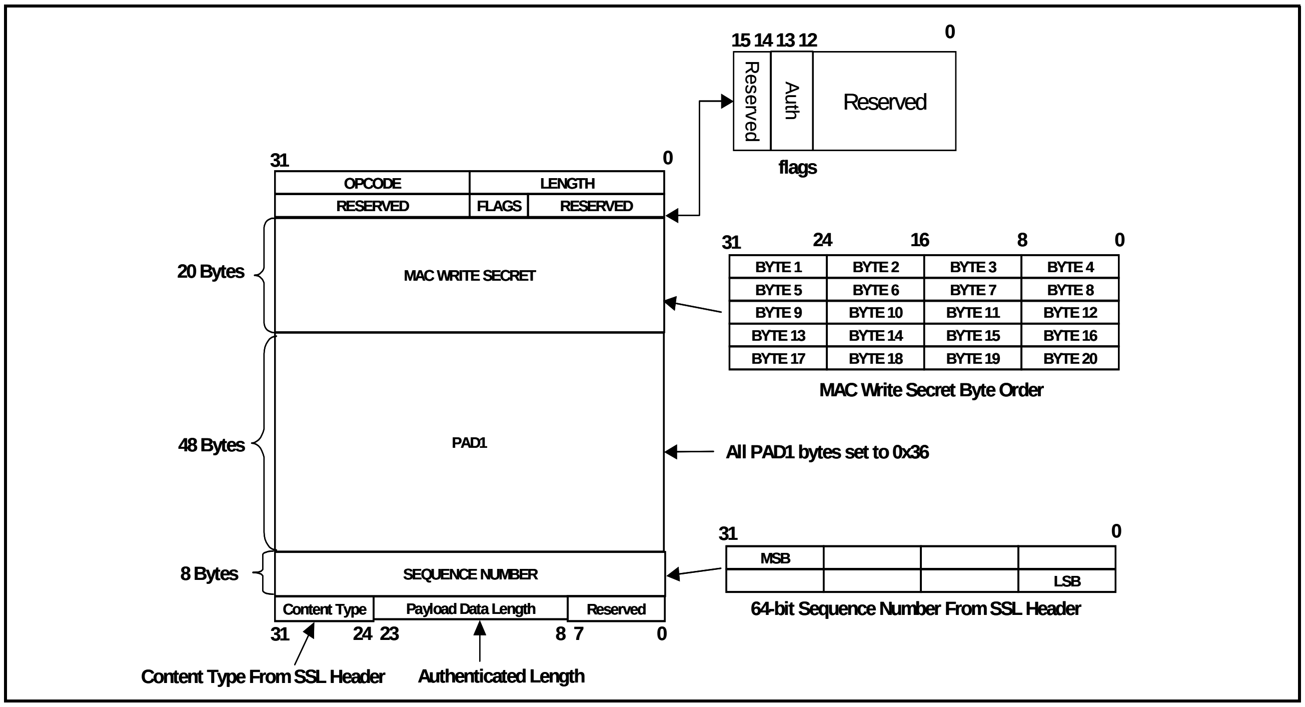

15 14 13 12
0
Reserved
Auth
Reserved
flags
31
0
OPCODE
LENGTH
RESERVED
FLAGS
RESERVED
20 Bytes
MAC WRITE SECRET
48 Bytes
PAD1
8 Bytes
SEQUENCE NUMBER
Content Type
Payload Data Length
Reserved
31 24 23 8 7 0
Content Type From SSL Header
Authenticated Length
31 24 16 8 0
BYTE 1 BYTE 2 BYTE 3 BYTE 4
BYTE 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 BYTE 20
MAC Write Secret Byte Order
All PAD1 bytes set to 0x36
31 0
MSB
LSB
64-bit Sequence Number From SSL Header

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.**

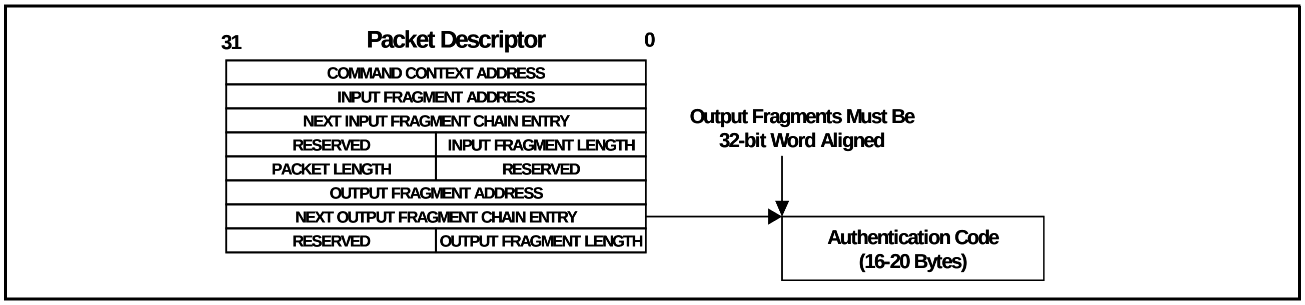


Figure 15: SSL MAC Authentication Output

TLS HMAC

TLS, specified in RFC 2246 [17], 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.

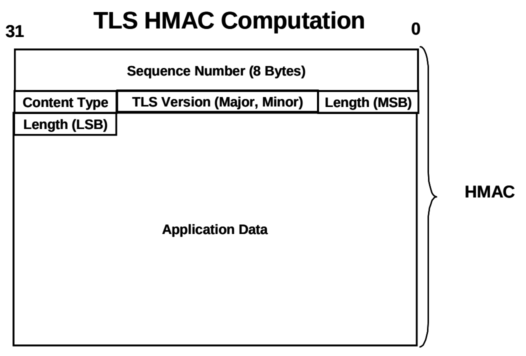


Figure 16: TLS HMAC Computation

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 13). The Opcode for this command, which uses MCR1@, is 0x02.

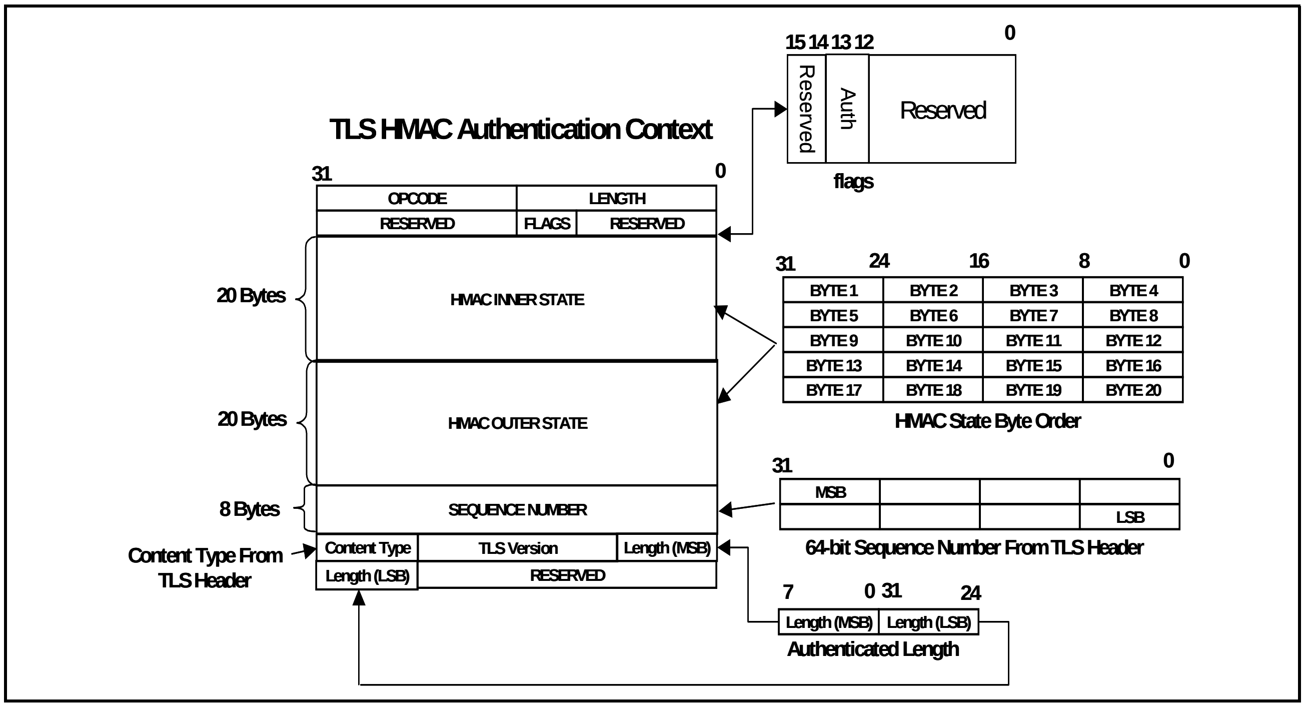


Figure 17: TLS HMAC Command Context

The HMAC Inner and Outer State are computed exactly the same as for IPsec (Figure 6 on page 7), 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 14.

SSL/TLS DES/3DES

The BCM5820 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@.

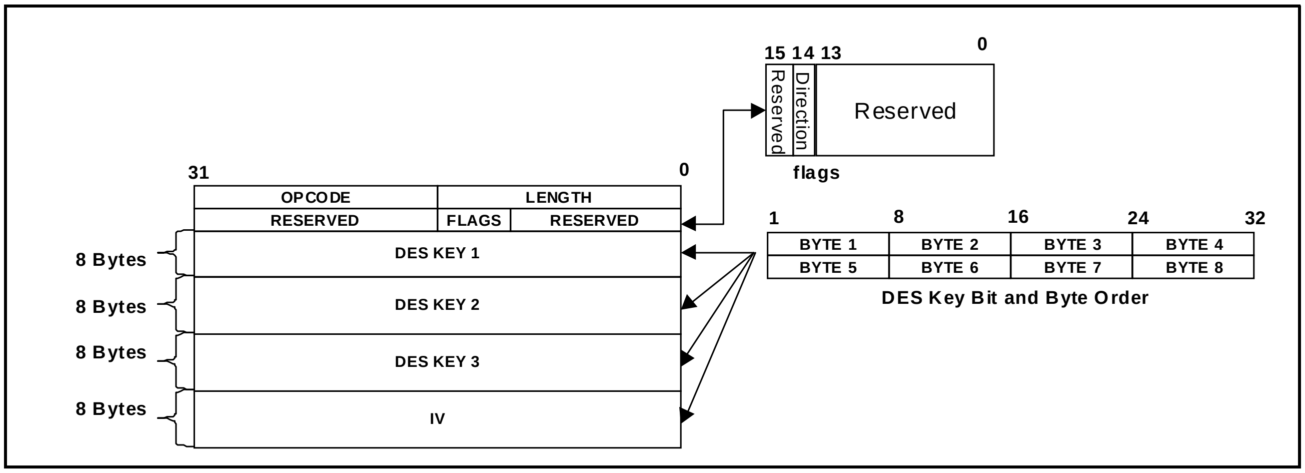


Figure 18: SSL/TLS DES/3DES Command Context

The same Packet Descriptor and Fragment processing formats are used as for IPsec as well.

PURE MD5/SHA-1 HASH

The BCM5820 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 19 shows the Command Context for Pure MD5/SHA-1 Hash. The Flags word authentication value is shown in Table 6. This Command Context length should be programmed to 8, even though the BCM5820 always reads 64 bytes.

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

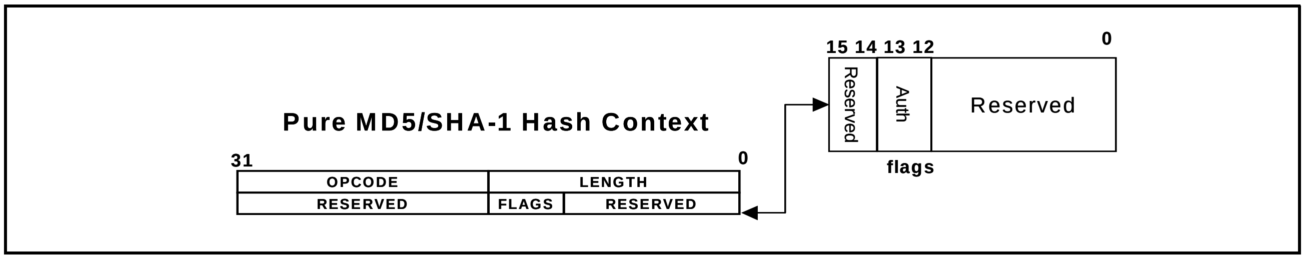


Figure 19: Pure MD5/SHA-1 Hash Command Context

The Packet Length must equal the total of the Input Fragment Lengths.

Table 6: 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.

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 [16]:

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 ($y = g^x$ mod N).

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

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

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

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

The BCM5820 provides separate Diffie-Hellman operation codes for generating the public key and for generating the secret key. Figure 20 on page 20 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 7.

Note

- **The BCM5820 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 BCM5820 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 7: 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 20 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.

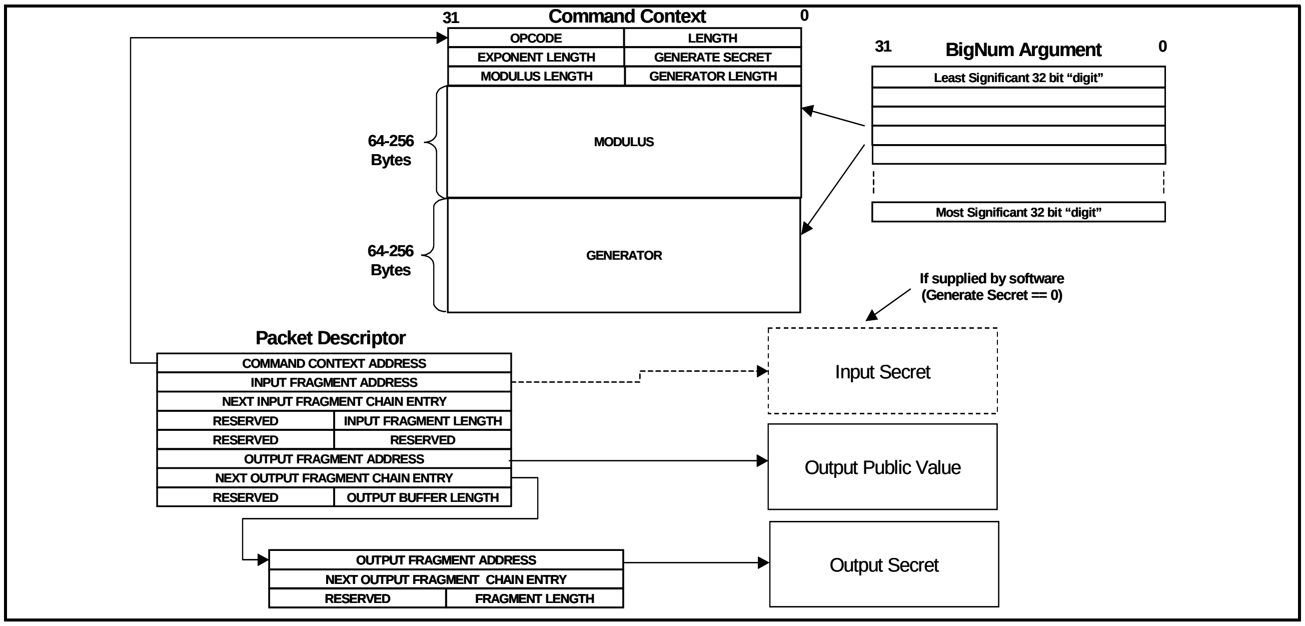


Figure 20: Diffie-Hellman Public Key Generate

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 BCM5820 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 7 on page 19. 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 BCM5820. This buffer must be the same length in bytes as the exponent Parameter Field Size increment in Table 7. It must be 32-bit aligned and contiguous. Output fragmentation is not used.

Figure 21 on page 21 shows the Command Context, Packet Descriptor structure, input, and output for Diffie-Hellman secret key derivation. This uses MCR2@, and Opcode 0x02.

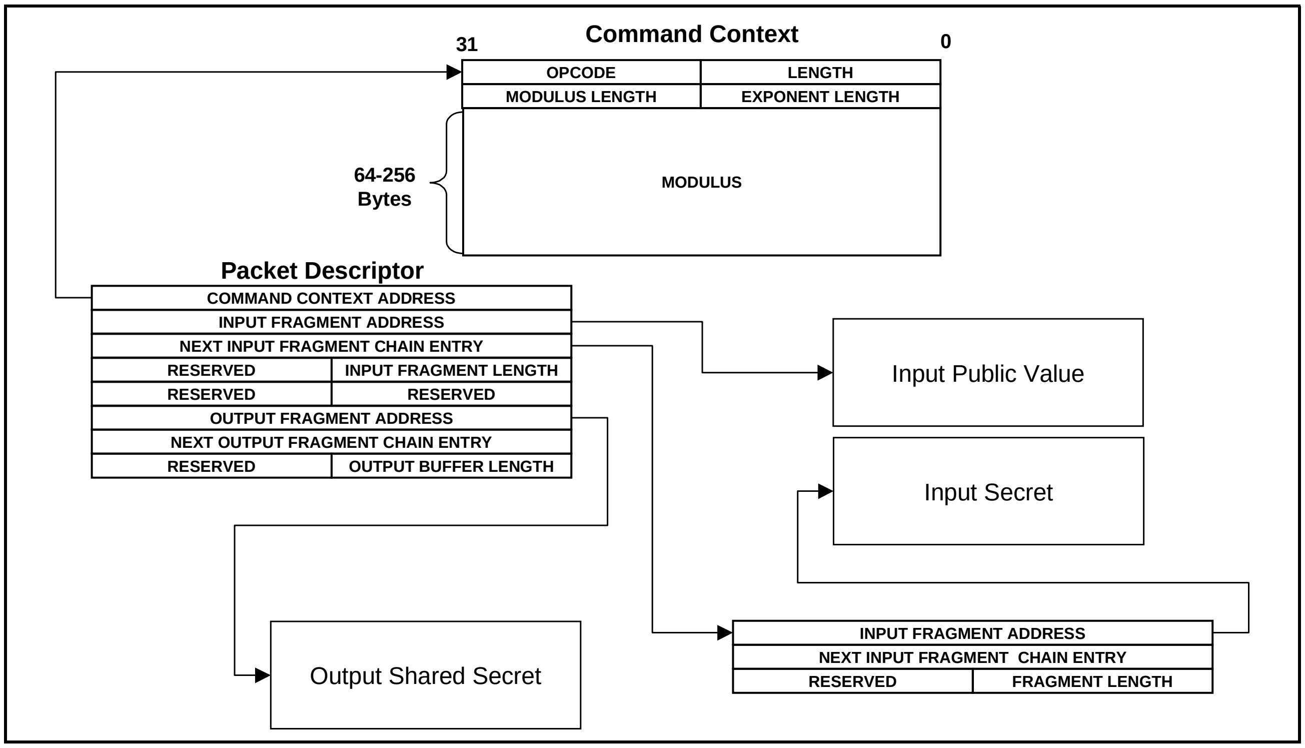


Figure 21: Diffie-Hellman Secret Key

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 7.

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 [16][18].

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. Since the effort to exponentiate is proportional to the logarithm of the exponent, 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. 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 (p^{-1} mod q).

The BCM5820 provides both RSA public key and CRT private key operations. Figure 22 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 7 on page 19. This operation uses MCR2@, and Opcode 0x03.

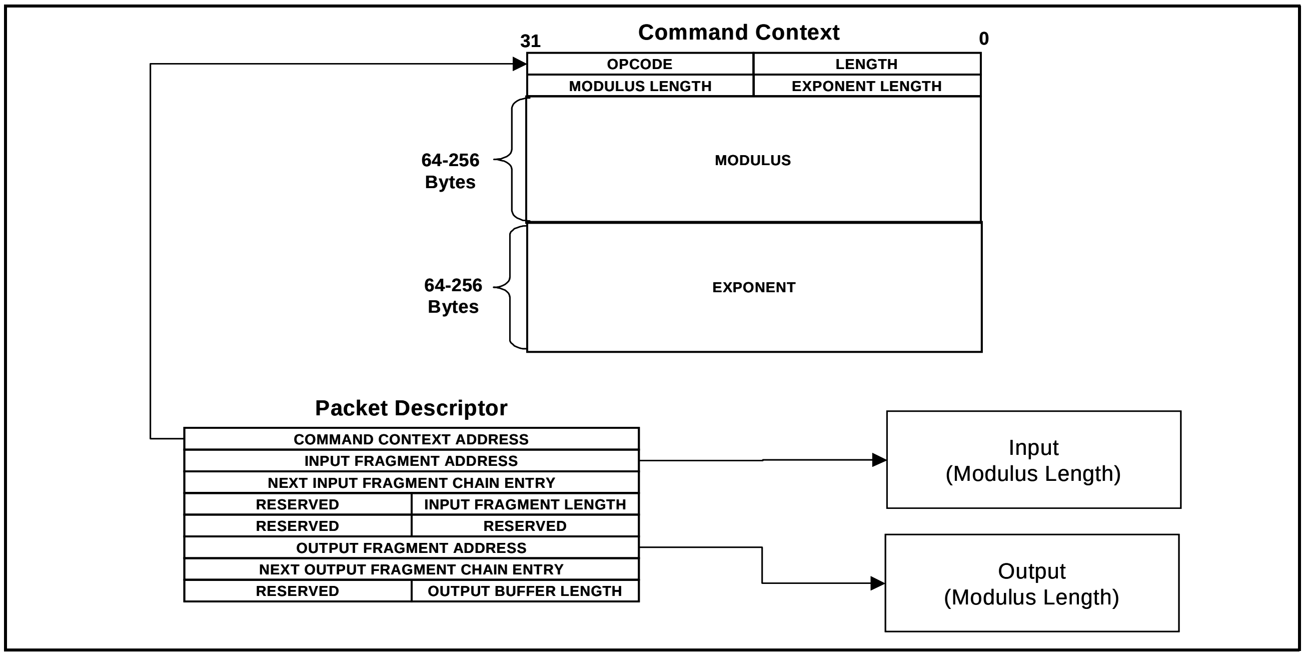


Figure 22: RSA Public Key

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 7. 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 23 shows the RSA CRT Private Key operation Command Context.The five CRT parameters, p, q, d_p, d_q, and p^{-1}, are each half the size of the public key modulus. Table 8 shows the allowed Parameter Field Sizes in 32-bit words and bytes.

Table 8: 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

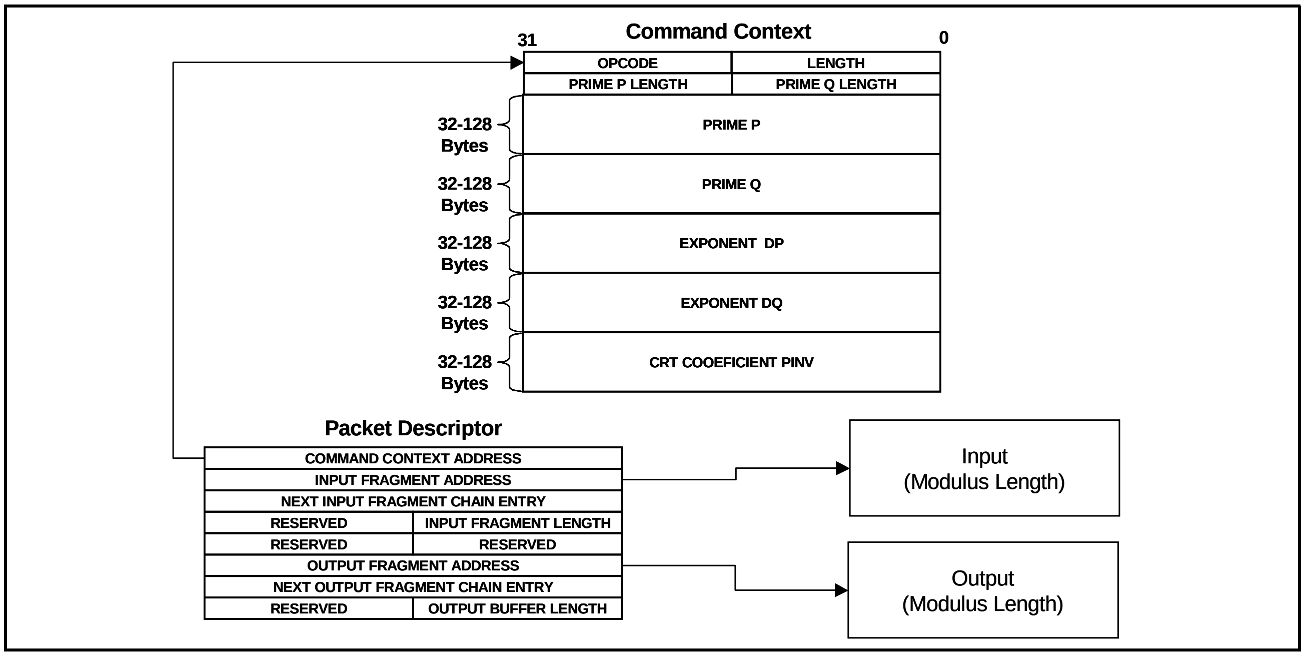


Figure 23: RSA Private Key

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.

Note

- **The BCM5820 provides eight complete modular arithmetic units, organized into two groups of four. In many applications, full utilization can often be achieved by using MCR structures containing four Packet Descriptors each.**

DSA

DSA, also known as the Digital Signature Standard (DSS), is described in FIPS-180-2 [18] 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 BCM5820 provides separate operations for DSA Sign and DSA Verify. Figure 24 on page 25 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 BCM5820, 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 BCM5820 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 BCM5820 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 24 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.

Note that the random number is not needed for verification and is usually discarded following creation of the signature. The BCM5820 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 24.

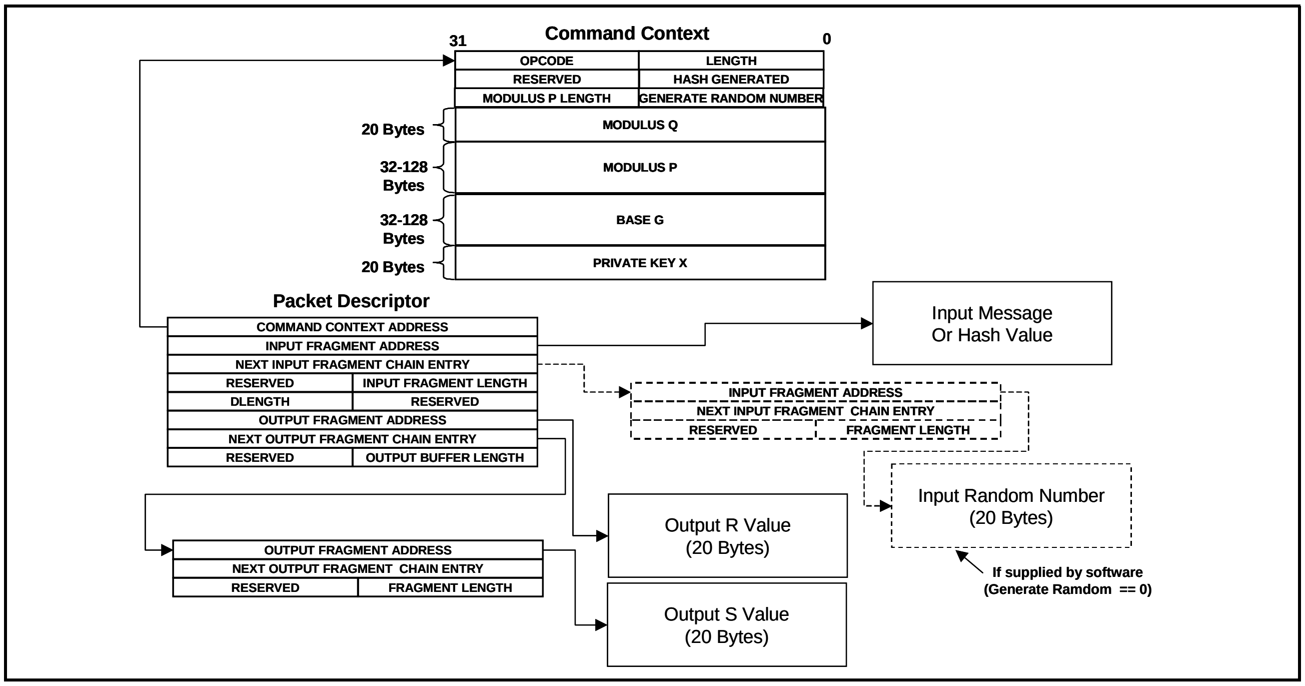


Figure 24: DSA Sign

Figure 25 on page 26 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 BCM5820. 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.

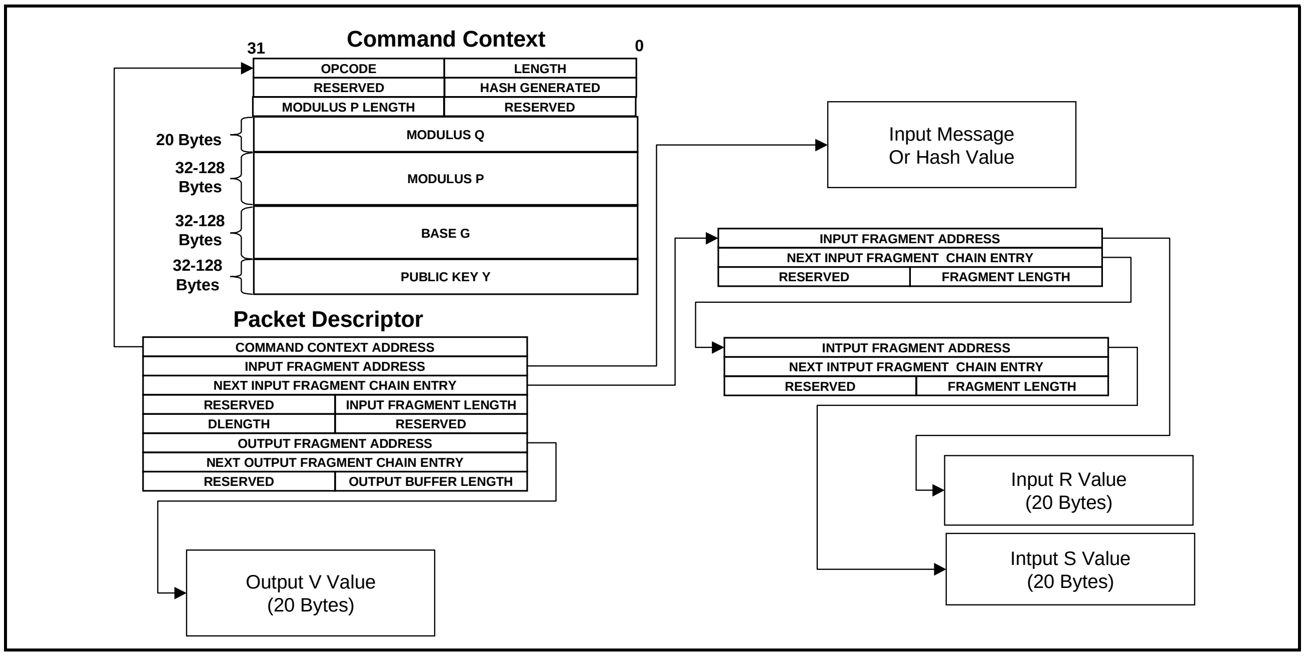


Figure 25: DSA Verify

Random Number Generation

The BCM5820 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.

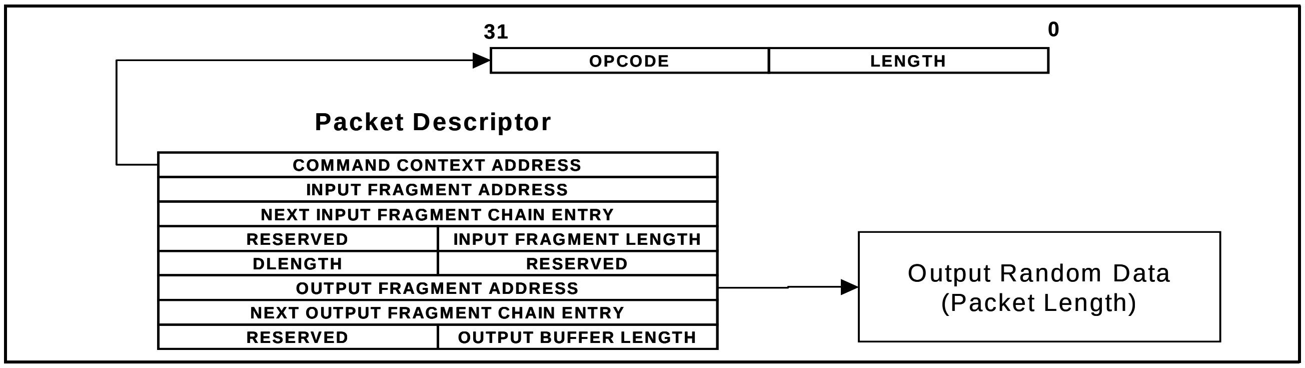


Figure 26: Random Number Generation Command Context

Modular Arithmetic

The BCM5820 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 10 on page 31. With the exception of Remainder, all arguments must be less than the modulus. The BCM5820 does not reduce arguments prior to performing the operation.

Figure 27 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 10.

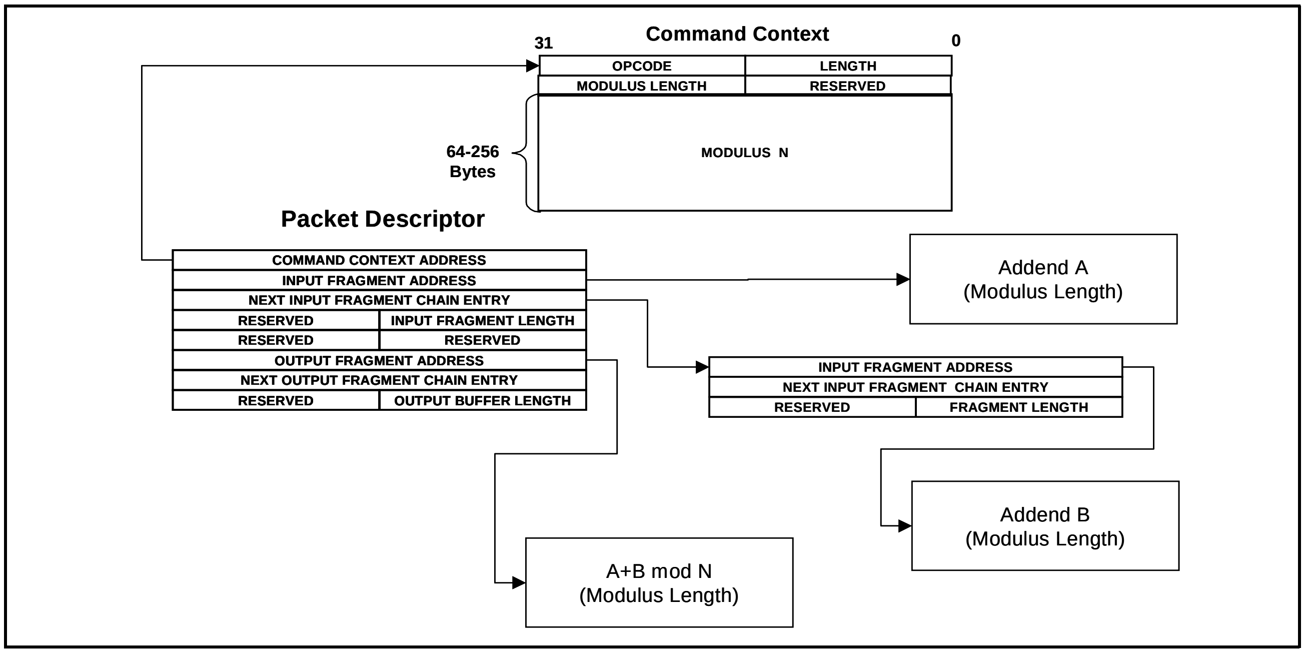


Figure 27: Modular Add

Figure 28 on page 29 shows the structure values for computing (A-B) mod N, and Figure 29 on page 29 for computing (A * B) mod N. Figure 30 on page 30 shows the unary operation A mod N, used to compute the remainder, and Figure 31 on page 30 shows A^E mod N. In all cases, the parameters are otherwise as described above for Modular Addition.

Table 9: 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

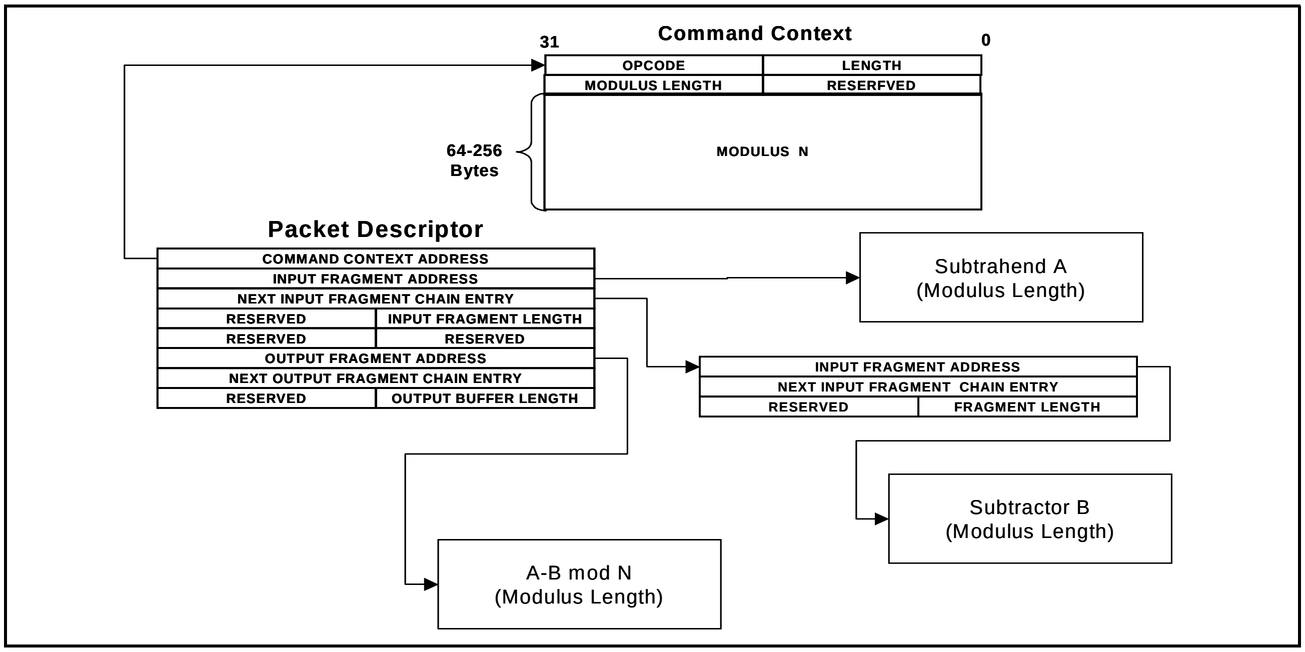


Figure 28: Modular Subtract

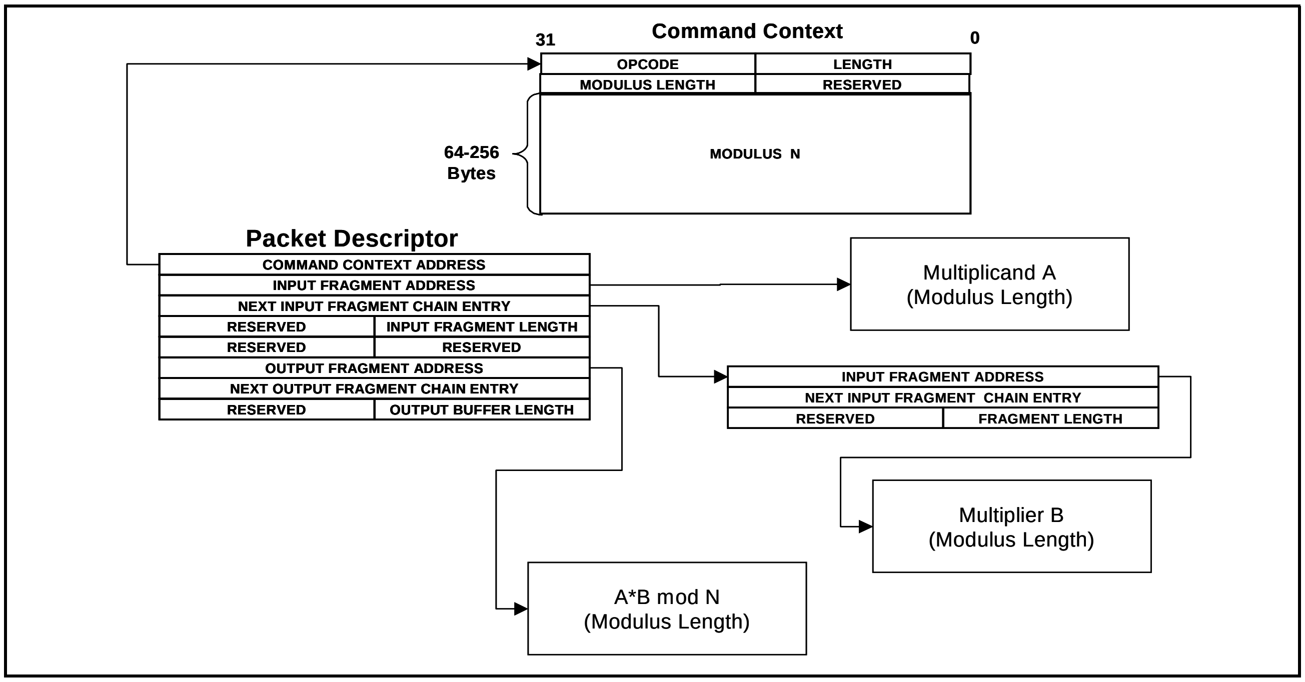


Figure 29: Modular Multiply

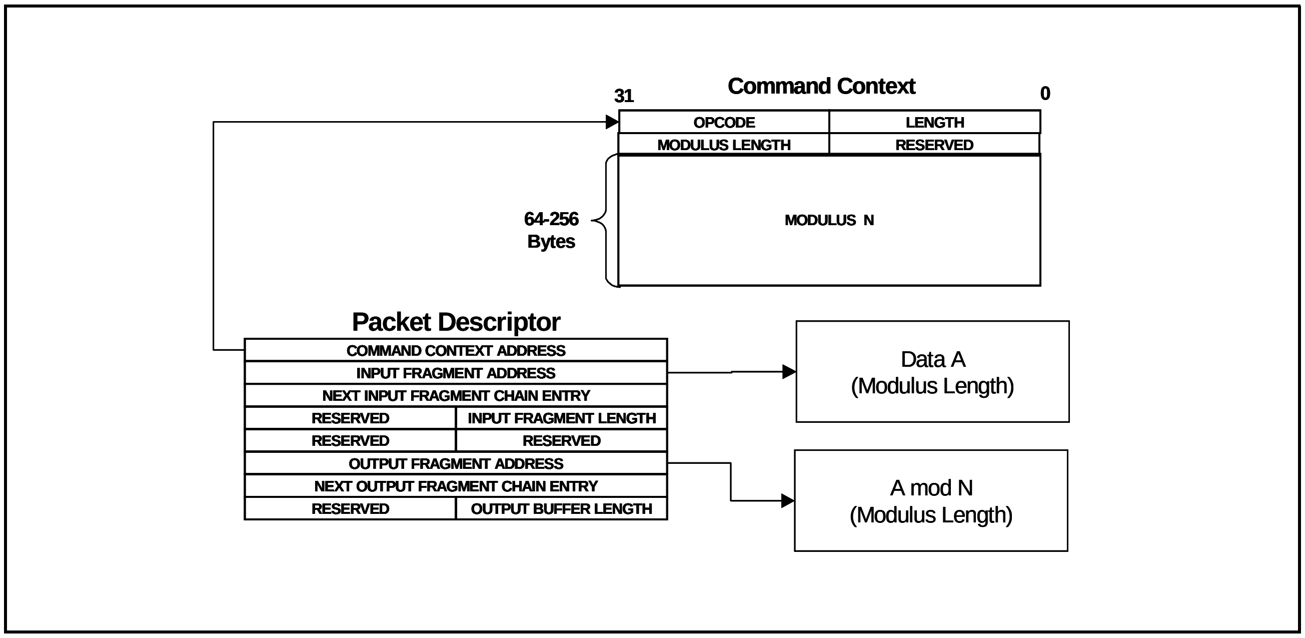


Figure 30: Modular Remainder

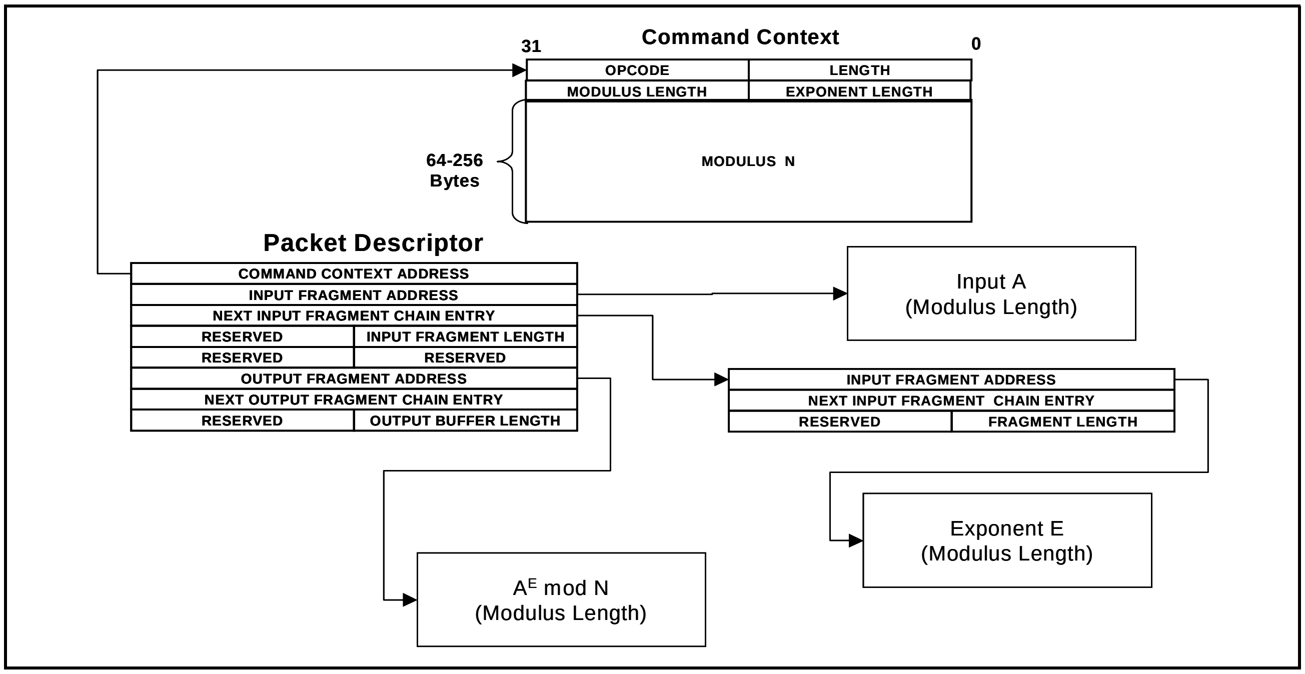


Figure 31: Modular Exponentiation

Table 10: 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

INTERRUPT PROCESSING

The BCM5820 will generate an interrupt after completion of all packet descriptors for a particular MCR Structure if interrupts are enabled in the DMA Control Register (See Table 17 on page 49) (MCR1INT_EN enables interrupts for symmetric operations pushed to MCR1@ and MCR2INT_EN enables interrupts for asymmetric and random number operations pushed to MCR2@). 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 BCM5820 will generate 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 BCM5820 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 11 on page 34, Table 12 on page 38, and Table 13 on page 42) is pulled high or low. If EXPORT is high, export mode is enabled, allowing only 56-bit DES and disabling 3DES. If EXPORT is pulled low, all bulk cryptographic functionality is enabled.

EXPORT is internally pulled high, and enabled by default.

Section 2: Ballout Assignments

	Y	W	V	U	T	R	P	N	M	L	K	J	H	G	F	E	D	C	B	A	
20	VDDC	GND	GND	GND	AD[59]	AD[56]	AD[53]	AD[50]	AD[49]	GND	VDDO	AD[44]	AD[41]	AD[38]	AD[35]	GND	GND	GND	GND	VDDC	20
19	GND	VDDC	GND	GND	AD[60]	AD[57]	AD[54]	AD[51]	SGND	AD[47]	AD[45]	AD[42]	AD[39]	AD[37]	AD[34]	AD[32]	GND	GND	VDDC	GND	19
18	GND	GND	VDDC	GND	AD[61]	AD[58]	AD[55]	AD[52]	VIO	AD[48]	AD[46]	AD[43]	AD[40]	AD[36]	AD[33]	VIO	GND	VDDC	GND	GND	18
17	GND	GND	GND	VDDC	GND	VDDO	VDDO	VDDO	VDDO	GND	VDDO	VDDO	VDDO	VDDO	VDDO	GND	VDDC	GND	GND	GND	17
16	PAR64	AD[63]	AD[62]	GND													GND	NC13	NC12	NC11	16
15	C/BE[6]#	C/BE[5]#	C/BE[4]#	VDDO													VDDO	NC10	NC9	VDDO	15
14	VIO	SGND	C/BE[7]#	VDDO													VDDO	NC8	NC7	GND	14
13	AD[2]	AD[1]	AD[0]	VDDO													GND	NC4	NC3	NC6	13
12	REQ64#	AD[4]	AD[3]	VDDO													VDDC	RNGOSC	NC5	VDDC	12
11	VDDO	AD[5]	ACK64#	VDDO													VDDO	SGND	NC2	VDDO	11
10	VDDO	AD[6]	AD[7]	VDDO													GND	TCK	TEST	GND	10
9	C/BE[0]#	AD[8]	AD[9]	GND													VDDC	TRST#	TMS	GND	9
8	AD[10]	AD[12]	AD[11]	VDDO													GND	TDO	TDI	VDDC	8
7	VDDO	AD[13]	AD[14]	VDDO													VDDO	AGND1	AVCC1	NC1	7
6	AD[15]	C/BE[1]#	PAR	VDDO													VDDO	COCLK	CLKMOD	EXPORT	6
5	GND	SERR#	PERR#	GND													GND	AGND2	AVCC2	GND	5
4	GND	GND	GND	VDDC	GND	GND	VDDO	VDDO	VDDO	VDDO	VDDO	VDDO	VDDO	VDDO	GND	GND	VDDC	GND	GND	GND	4
3	GND	GND	VDDC	GND	LOCK#	TRDY#	C/BE[2]#	AD[18]	AD[20]	AD[23]	REQ#	AD[25]	AD[28]	AD[31]	VIO	PCI_CLK	GND	VDDC	GND	GND	3
2	GND	VDDC	GND	GND	STOP#	IRDY#	AD[16]	AD[19]	AD[21]	IDSEL	C/BE[3]#	AD[24]	AD[27]	AD[30]	INTA#	VIO	GND	GND	VDDC	GND	2
1	VDDC	GND	GND	GND	DEVSEL#	FRAME#	AD[17]	VDDO	AD[22]	VDDO	VDDO	GNT#	AD[26]	AD[29]	SGND	RESET#	GND	GND	GND	VDDC	1
	Y	W	V	U	T	R	P	N	M	L	K	J	H	G	F	E	D	C	B	A	

BROADCOM
BCM5820

Figure 32: 256-Pin TBGA Pinout Diagram

BALLOUT BY BALL NUMBER

Table 11: Ballout by Ball Number

Ball	*Signal Name*
A1	VDDC
A2	GND
A3	GND
A4	GND
A5	GND
A6	EXPORT
A7	NC1
A8	VDDC
A9	GND
A10	GND
A11	VDDO
A12	VDDC
A13	NC6
A14	GND
A15	VDDO
A16	NC11
A17	GND
A18	GND
A19	GND
A20	VDDC
B1	GND
B2	VDDC
B3	GND
B4	GND
B5	AVCC2
B6	CLKMOD
B7	AVCC1
B8	TDI
B9	TMS
B10	TEST
B11	NC2
B12	NC5
B13	NC3
B14	NC7
B15	NC9
B16	NC12
B17	GND
B18	GND
B19	VDDC
B20	GND
C1	GND
C2	GND
C3	VDDC
C4	GND
C5	AGND2
C6	COCLK
C7	AGND1
C8	TDO
C9	TRST#
C10	TCK
C11	SGND
C12	RNGOSC
C13	NC4
C14	NC8
C15	NC10
C16	NC13
C17	GND
C18	VDDC
C19	GND
C20	GND
D1	GND
D2	GND
D3	GND
D4	VDDC
D5	GND
D6	VDDO
D7	VDDO
D8	GND
D9	VDDC
D10	GND
D11	VDDO
D12	VDDC
D13	GND
D14	VDDO
D15	VDDO
D16	GND
D17	VDDC
D18	GND

Table 11: Ballout by Ball Number (Cont.)

Ball	Signal Name
D19	GND
D20	GND
E1	RST#
E2	VIO
E3	PCI_CLK
E4	GND
E17	GND
E18	VIO
E19	AD[32]
E20	GND
F1	SGND
F2	INTA#
F3	VIO
F4	GND
F17	VDDO
F18	AD[33]
F19	AD[34]
F20	AD[35]
G1	AD[29]
G2	AD[30]
G3	AD[31]
G4	VDDO
G17	VDDO
G18	AD[36]
G19	AD[37]
G20	AD[38]
H1	AD[26]
H2	AD[27]
H3	AD[28]
H4	VDDO
H17	VDDO
H18	AD[40]
H19	AD[39]
H20	AD[41]
J1	GNT#
J2	AD[24]
J3	AD[25]
J4	VDDO
J17	VDDO
J18	AD[43]
J19	AD[42]
J20	AD[44]
K1	VDDO
K2	C/[BE[3]#
K3	REQ#
K4	VDDO
K17	VDDO
K18	AD[46]
K19	AD[45]
K20	VDDO
L1	VDDO
L2	IDSEL
L3	AD[23]
L4	VDDO
L17	GND
L18	AD[48]
L19	AD[47]
L20	GND
M1	AD[22]
M2	AD[21]
M3	AD[20]
M4	VDDO
M17	VDDO
M18	VIO
M19	SGND
M20	AD[49]
N1	VDDO
N2	AD[19]
N3	AD[18]
N4	VDDO
N17	VDDO
N18	AD[52]
N19	AD[51]
N20	AD[50]
P1	AD[17]
P2	AD[16]
P3	C/BE[2]#
P4	VDDO
P17	VDDO
P18	AD[55]
P19	AD[54]
P20	AD[53]
R1	FRAME#
R2	IRDY#

Table 11: Ballout by Ball Number (Cont.)

Ball	Signal Name
R3	TRDY#
R4	GND
R17	VDDO
R18	AD[58]
R19	AD[57]
R20	AD[56]
T1	DEVSEL#
T2	STOP#
T3	LOCK#
T4	GND
T17	GND
T18	AD[61]
T19	AD[60]
T20	AD[59]
U1	GND
U2	GND
U3	GND
U4	VDDC
U5	GND
U6	VDDO
U7	VDDO
U8	VDDO
U9	GND
U10	VDDO
U11	VDDO
U12	VDDO
U13	VDDO
U14	VDDO
U15	VDDO
U16	GND
U17	VDDC
U18	GND
U19	GND
U20	GND
V1	GND
V2	GND
V3	VDDC
V4	GND
V5	PERR#
V6	PAR
V7	AD[14]
V8	AD[11]

Table 11: Ballout by Ball Number (Cont.)

Ball	Signal Name
V9	AD[9]
V10	AD[7]
V11	ACK64#
V12	AD[3]
V13	AD[0]
V14	C/BE[7]#
V15	C/BE[4]#
V16	AD[62]
V17	GND
V18	VDDC
V19	GND
V20	GND
W1	GND
W2	VDDC
W3	GND
W4	GND
W5	SERR#
W6	C/BE[1]#
W7	AD[13]
W8	AD[12]
W9	AD[8]
W10	AD[6]
W11	AD[5]
W12	AD[4]
W13	AD[1]
W14	SGND
W15	C/BE[5]#
W16	AD[63]
W17	GND
W18	GND
W19	VDDC
W20	GND
Y1	VDDC
Y2	GND
Y3	GND
Y4	GND
Y5	GND
Y6	AD[15]
Y7	VDDO
Y8	AD[10]
Y9	C/BE[0]#
Y10	VDDO

Table 11: Ballout by Ball Number (Cont.)

Ball	***Signal Name***
Y11	VDDO
Y12	REQ64#
Y13	AD[2]
Y14	VIO
Y15	C/BE[6]#
Y16	PAR64
Y17	GND
Y18	GND
Y19	GND
Y20	VDDC

Ballout by Signal Name

Table 12: Ballout by Signal Name

Ball	Signal Name
V11	ACK64#
V13	AD[0]
W13	AD[1]
Y13	AD[2]
V12	AD[3]
W12	AD[4]
W11	AD[5]
W10	AD[6]
V10	AD[7]
W9	AD[8]
V9	AD[9]
Y8	AD[10]
V8	AD[11]
W8	AD[12]
W7	AD[13]
V7	AD[14]
Y6	AD[15]
P2	AD[16]
P1	AD[17]
N3	AD[18]
N2	AD[19]
M3	AD[20]
M2	AD[21]
M1	AD[22]
L3	AD[23]
J2	AD[24]
J3	AD[25]
H1	AD[26]
H2	AD[27]
H3	AD[28]
G1	AD[29]
G2	AD[30]
G3	AD[31]
E19	AD[32]
F18	AD[33]
F19	AD[34]
F20	AD[35]
G18	AD[36]
G19	AD[37]
G20	AD[38]
H19	AD[39]
H18	AD[40]
H20	AD[41]
J19	AD[42]
J18	AD[43]
J20	AD[44]
K19	AD[45]
K18	AD[46]
L19	AD[47]
L18	AD[48]
M20	AD[49]
N20	AD[50]
N19	AD[51]
N18	AD[52]
P20	AD[53]
P19	AD[54]
P18	AD[55]
R20	AD[56]
R19	AD[57]
R18	AD[58]
T20	AD[59]
T19	AD[60]
T18	AD[61]
V16	AD[62]
W16	AD[63]
C7	AGND1
C5	AGND2
B7	AVCC1
B5	AVCC2
Y9	C/BE[0]#
W6	C/BE[1]#
P3	C/BE[2]#
K2	C/BE[3]#
V15	C/BE[4]#
W15	C/BE[5]#
Y15	C/BE[6]#
V14	C/BE[7]#
B6	CLKMOD

Table 12: Ballout by Signal Name (Cont.)

Ball	Signal Name
C6	COCLK
T1	DEVSEL#
A6	EXPORT
R1	FRAME#
A2	GND
A3	GND
A4	GND
A5	GND
A9	GND
A10	GND
A14	GND
A17	GND
A18	GND
A19	GND
B1	GND
B3	GND
B4	GND
B17	GND
B18	GND
B20	GND
C1	GND
C2	GND
C4	GND
C17	GND
C19	GND
C20	GND
D1	GND
D2	GND
D3	GND
D5	GND
D8	GND
D10	GND
D13	GND
D16	GND
D18	GND
D19	GND
D20	GND
E4	GND
E17	GND
E20	GND
F4	GND
L17	GND
L20	GND
R4	GND
T4	GND
T17	GND
U1	GND
U2	GND
U3	GND
U5	GND
U9	GND
U16	GND
U18	GND
U19	GND
U20	GND
V1	GND
V2	GND
V4	GND
V17	GND
V19	GND
V20	GND
W1	GND
W3	GND
W4	GND
W17	GND
W18	GND
W20	GND
Y2	GND
Y3	GND
Y4	GND
Y5	GND
Y17	GND
Y18	GND
Y19	GND
J1	GNT#
L2	IDSEL
F2	INTA#
R2	IRDY#
T3	LOCK#
A7	NC1
B11	NC2
B13	NC3
C13	NC4
B12	NC5

Table 12: Ballout by Signal Name (Cont.)

Ball	***Signal Name***
A13	NC6
B14	NC7
C14	NC8
B15	NC9
C15	NC10
A16	NC11
B16	NC12
C16	NC13
V6	PAR
Y16	PAR64
E3	PCI_CLK
V5	PERR#
K3	REQ#
Y12	REQ64#
E1	RST#
C12	RNGOSC
W5	SERR#
C11	SGND
F1	SGND
M19	SGND
W14	SGND
T2	STOP#
C10	TCK
B8	TDI
C8	TDO
B10	TEST
B9	TMS
R3	TRDY#
C9	TRST#
A1	VDDC
A8	VDDC
A12	VDDC
A20	VDDC
B2	VDDC
B19	VDDC
C3	VDDC
C18	VDDC
D4	VDDC
D9	VDDC
D12	VDDC
D17	VDDC
U4	VDDC

Table 12: Ballout by Signal Name (Cont.)

Ball	***Signal Name***
U17	VDDC
V3	VDDC
V18	VDDC
W2	VDDC
W19	VDDC
Y1	VDDC
Y20	VDDC
A11	VDDO
A15	VDDO
D6	VDDO
D7	VDDO
D11	VDDO
D14	VDDO
D15	VDDO
F17	VDDO
G4	VDDO
G17	VDDO
H4	VDDO
H17	VDDO
J4	VDDO
J17	VDDO
K1	VDDO
K4	VDDO
K17	VDDO
K20	VDDO
L1	VDDO
L4	VDDO
M4	VDDO
M17	VDDO
N1	VDDO
N4	VDDO
N17	VDDO
P4	VDDO
P17	VDDO
R17	VDDO
U6	VDDO
U7	VDDO
U8	VDDO
U10	VDDO
U11	VDDO
U12	VDDO
U13	VDDO

Table 12: Ballout by Signal Name (Cont.)

Ball	***Signal Name***
U14	VDDO
U15	VDDO
Y7	VDDO
Y10	VDDO
Y11	VDDO
E2	VIO
E18	VIO
F3	VIO
M18	VIO
Y14	VIO

Signal Definitions

Table 13: Signal Definitions

Signal Name	*I/O*	*Description*
AD[63::0]	IO	PCI multiplexed Address/Data bus.
PCI_CLK	I	PCI Clock, 33-66MHz.
GNT#	I	PCI Bus Grant allowing BCM5820 to use the bus.
FRAME#	IO	PCI Frame, indicates the beginning and duration of a master transfer.
IRDY#	IO	PCI Initiator ready.
TRDY#	IO	PCI Target ready.
DEVSEL#	IO	PCI Device Select, asserted by an access target.
STOP#	IO	PCI Stop, requesting that the current master stop an active transfer.
PERR#	IO	PCI Parity Error.
SERR#	O	PCI System Error, open drain.
PAR	IO	PCI Parity.
PAR64	IO	PCI Parity for upper 32 address/data and 4 C/BE signal lines.
REQ#	O	PCI Bus Request.
REQ64#	IO	PCI 64-bit Bus Request.
ACK64#	IO	PCI Bus Acknowledgement in response to REQ64#.
RST#	I	PCI Reset, tri-states all PCI outputs.
INTA#	O	PCI interrupt output, open drain.
C/BE[7::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.
LOCK#	I	PCI lock for atomic operation, must be pulled up to VDDO.
VDDC	Core power pins, must be connected to a 2.5V(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 2.5V source.
AGND1		Analog GND for PLL1.
AVCC2		Analog VCC for PLL2, must be connected to a quiet 2.5V 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).	
COCLK	I	Chip Core Clock, 33-90 MHz.
CLKMOD	I	Clock Mode: High=clock chip from PCI, Low=clock from COCLK. It is internally pulled up.
EXPORT	IO	EXPORT pin (high = 56-bit DES encyrption; low = 3DES strong encryption). If the EXPORT pin is not connected, the signal EXPORT will be driven to high by an on-chip pull-up driver.
TEST	I	Test pin, must be grounded for regular operation. When TEST is high, all outputs are tri-stated. It is internally pulled down.
TRST#	I	Must be connected to ground for normal operation. Used for boundary scan JTAG testing. It is internally pulled down.

Table 13: Signal Definitions (Cont.)

Signal Name	***I/O***	***Description***
TMS	I	Test mode select for JTAG boundary scan. Must be connected to VDDO for normal operation. It is internally pulled up.
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.
NC1......NC13		Test pins, must be left unconnected (floating).

Section 3: Register Details

The BCM5820 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’s 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 BCM5820 provides PCI 2.2 compliant configuration space registers as shown in Table 14. In addition, the BCM5820 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 14 are Reserved.

Table 14: 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 15.

Table 15: PCI Configuration Register Bit Fields

Bits	***Access***	***Reset***	***Purpose***
PCI Vendor ID - 0x00			
15:0	RO	0x14E4	Vendor identifier, default value assigned by PCISIG for Broadcom
PCI Device ID - 0x02			
31:16	RO	0x5820	Device identifier
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
23	RO	1	Fast back to back capable status
22	RSVD	0	Reserved
21	RO	1	66-MHz capable
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 BCM5820. Default to 0x00.
23:16	RO	0x00	Header type = 0, single function

Table 15: PCI Configuration Register Bit Fields (Cont.)

Bits	Access	Reset	Purpose
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 BCM5820 device. This value is ignored as long as the GNT# signal remains asserted to the BCM5820. 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 BCM5820 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 will be generated by the BCM5820 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 BCM5820 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	0x0001	Subsystem ID
15:0	RO	0x14E4	Subsystem Vendor ID (Broadcom PCISIG ID)
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 BCM5820 will attempt before setting the DMAERR_INTR bit in the BCM5820 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 BCM5820 will tolerate 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 BCM5820 will tolerate before setting the DMAERR_INTR bit in the BCM5820 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 BCM5820 will tolerate 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 since 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. BCM5820 does not support PME# pin.

Table 15: PCI Configuration Register Bit Fields (Cont.)

Bits	***Access***	***Reset***	***Purpose***
10	RO	0	Indicates whether the device supports D2 power management state. BCM5820 does not support this state.
9	RO	0	Indicates whether the device supports D1 power management state. BCM5820 does not support this state.
8:6	RO	0	Auxiliary Current. BCM5820 does not support Auxiliary Power Supply.
5	RO	0	Device specification initialization. It is not necessary for BCM5820.
4	RSVD	0	Reserved
3	RO	0	Indicates that the device requires the presence of PCI clock for PME# operation. BCM5820 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. BCM5820 does not have Data register.
12:9	RO	0	Indicates which data is to be reported via the Data register. BCM5820 does not have Data register.
8	RO	0	Enables the device to generate PME#. BCM5820 does not support 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 16: 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 17: 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 Structure to this register causes crypto/authentication processing of the Packet Descriptors within that structure 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 will go 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 BCM5820, it writes a one to this bit to reset the BCM5820. The BCM5820 can be used after 256 core-clock cycles.
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
27:26	RSVD	1	Reserved
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

Table 17: DMA Control and Status Registers (Cont.)

Bits	**Access**	**Reset**	**Purpose**
22	RW	0	Modulus Normalization • 0: Modulus normalization is done by BCM5820. 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. Note that this bit will accurately reflect processing status even if the corresponding interrupt bit is disabled (in which case a PCI interrupt will not be generated). This bit is sticky until cleared explicitly. Resets to 0.
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 will accurately reflect status even if the corresponding interrupt enable bit is off (in which case a PCI interrupt will not be 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. Note that this bit will accurately reflect processing status even if the corresponding interrupt bit is disabled (in which case a PCI interrupt will not be generated). This bit is sticky until cleared explicitly. Resets to 0.
25:24	RSVD	0	Reserved
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 Structure to this register causes key setup processing of the data within that structure 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 because BCM5820 can process two key setup MCRs simultaneously, such that the MCR_FULL bit will go to zero very quickly after two initial writes to this register. This allows the CPU to write an additional MCR address values to this register, effectively queuing two 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 & Timing Specifications

GENERAL SPECIFICATIONS

Table 18: Operating Conditions

Parameter	Typical	Description
PCI Compliance	3.3V and 5V	Over the range of 33-66 MHz PCI clocks
Supply Voltage	3.3V ± 5% 2.5V ± 5%	For I/O buffers For the core
I/O Buffers	3.3V	
Operating Temperature	0-70C	Within the commercial temperature range

Table 19: PCI Pin DC Specifications

Symbol	Parameter	Condition	Min	Max	Units
V_{DCC}	Core Supply Voltage		2.375	2.625	V
V_{IH} (PCI_CLK)	Input High Voltage for PCI_CLK pin		0.61 V_{DCC}	V_{CC} + 0.5	V
V_{OL} (INT#)	Output Low Voltage for INT# pin			0.26 V_{DDC}	V
V_{OL} (SERR#)	Output Low Voltage for SERR# pin			0.26 V_{DDC}	V
V_{IH}	Input High Voltage for all other pins		0.50 V_{CC}	V_{CC} + 0.5	V
V_{IL}	Input Low Voltage		-0.5	0.3V_{CC}	V
V_{IPU}	Input Pull-up Voltage		0.7V_{CC}		V
V_{OH}	Output High Voltage	I_{OUT} = -0.5 mA	0.9V_{CC}		V
V_{OL}	Output Low Voltage	I_{OUT} = -1.5 mA		0.1V_{CC}	V
C_{IN}	Input Pin Capacitance		5	12	pF
C_{CLK}	PCI_CLK Pin Capacitance			8	pF
L_{PIN}	Pin Inductance			20	nH

PCI_CLK pin violates PCI V_{IH} specification at the operating range.
INT# and SERR# violate PCI V_{OL} specification at the operating range.
All other pins are within the PCI DC Specifications.

Table 20: PCI Pin Timing Specifications

Symbol	**Parameter**	**66 MHz**		**33 MHz**		**Units**
		Min	**Max**	**Min**	**Max**	
T_{VAL}	PCI_CLK to Signal Valid Delay - bussed signals	2	6	2	11	ns
T_{SU}	Input Setup Time to PCI_CLK - bussed signals	3		7		ns
$T_{SU\ (PTP)}$	Input Setup Time to PCI_CLK - point to point signals (REQ# and GNT#)	5		10, 12		ns
T_{HD}	Input Hold Time from PCI_CLK	0.5				ns

BCM5820 PCI pins violate T_{HD} slightly (0.5ns vs. 0ns in the PCI Spec).
T_{VAL}, T_{SU} and $T_{SU\ (PTP)}$ are within the PCI Timing Specifications in the entire operating range.

Table 21: Power Consumption

Parameter		**Max Power**	**Max Current**	**Max Voltage (Vnominal +5%)**
Peripheral (66MHz)		0.5W	145mA	3.46V
Core (90MHz)	Bulk Encrypt/Auth	1.7W	650mA	2.62V
	512-bit Public Key Ops	2.3W	880mA	2.62V
	1024-bit Public Key Ops	2.8W	1100mA	2.62V
	2048-bit Public Key Ops	4.0W	1500mA	2.62V

Table 22: 256-pin TBGA Package Thermal Parameters

Parameter	**Theta$_{ja}$ (Junction To Amb.)**	**Theta$_{jb}$ (Junction T-Board)**	**Theta$_{jc}$ (Junction To Case)**	**Max Ambient Temperature**
Assume in still air, no heat sink	11.6 C/W	3.68 C/W	0.35 C/W	70C

Table 23: External Core Clock AC Specifications

Parameter	**Min**	**Max**	**Units**
Oscillator frequency	25	90	MHz
Duty cycle	40	60	%
Rise Time/Fall Time		5	ns

Section 5: Performance

System Throughput

System Throughput values for the BCM5820 are shown below in Table 24. System values represent measured, memory to memory, in-system throughput on an optimal platform using large buffer sizes and maximum pipelining.

Table 24: BCM5820 System Throughput

Function	***Value***
IPsec 3DES with HMAC-MD5-96 or HMAC-SHA-1-96	290 Mbps
IPsec Null with HMAC-MD5-96	450 Mbps
IPsec Null with HMAC-SHA-1-96	375 Mbps
SSL-MAC using MD5	450 Mbps
SSL-MAC using SHA-1	375 Mbps
TLS-HMAC-MD5	450 Mbps
TLS-HMAC-SHA-1	375 Mbps
Diffie-Hellman (1024-bit Modulus, 180-bit Exponent)	1200 Generate+Shared per second
DSA Sign with 1024-bit public key, 160-bit private key	2000 per second
DSA Verify with 1024-bit public key, 160-bit private key	1200 per second
RSA Private Key CRT, 1024-bit modulus	800 per second
Random number generation	1 Mbps

PACKET PERFORMANCE

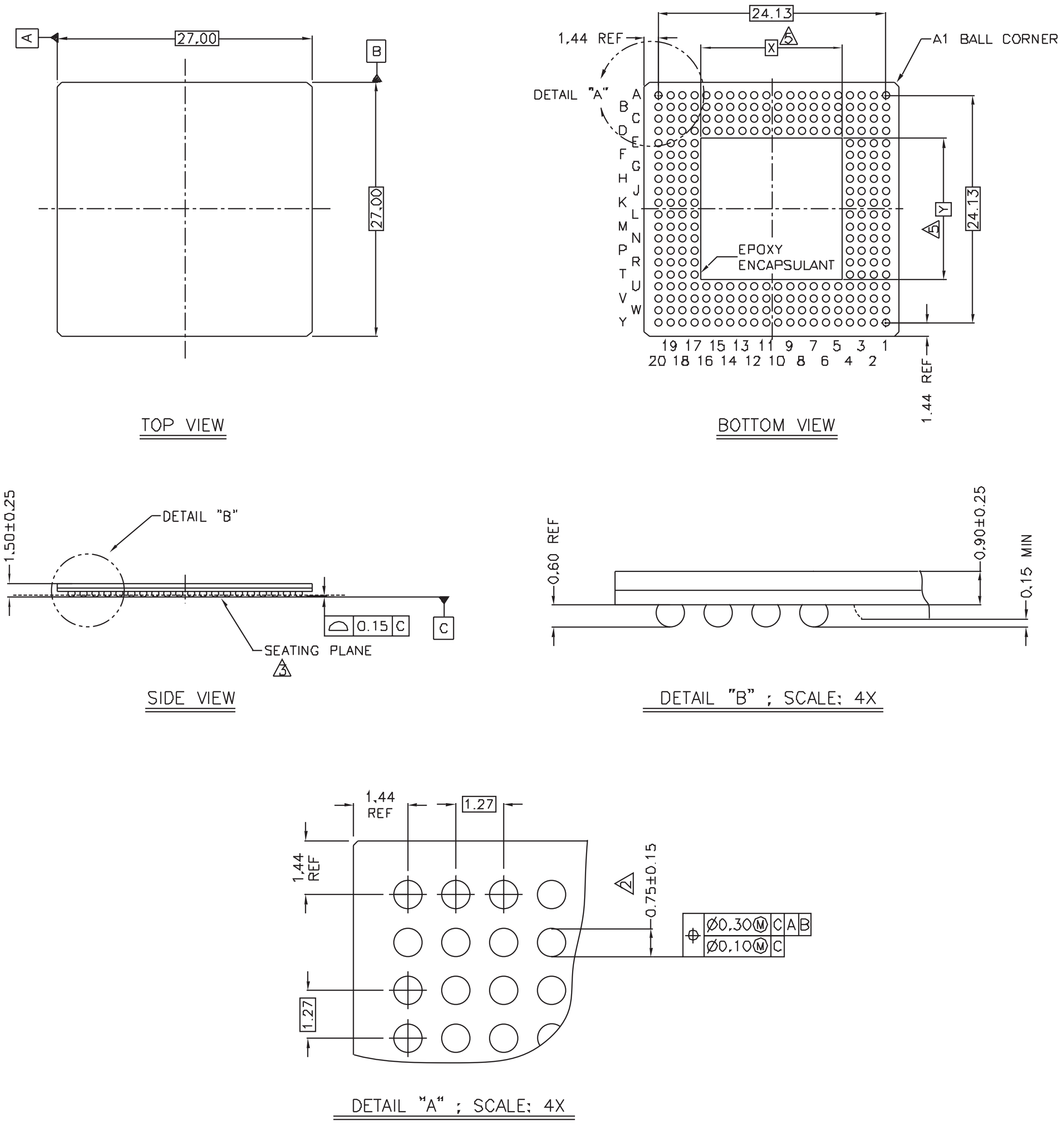
IPSEC

Table 25 shows in-system performance for the BCM5820, 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, except for the 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 25: BCM5820 IPSec Performance by Packet SIze

Core Clock Frequency/PCI Clock Frequency	*Packet DIrection*	*Packet Sizes (Bytes)*					
		64	*128*	*256*	*512*	*1024*	*2048*
33 MHz/33MHz	Outbound	26	45	62	84	101	111
	Inbound	45	64	81	98	109	116
90 MHz/33MHz	Outbound	55	95	132	178	211	233
	Inbound	77	117	154	191	219	236
66 MHz/66MHz	Outbound	50	88	124	168	201	224
	Inbound	87	119	158	193	218	233
90 MHz/66MHz	Outbound	55	97	138	193	240	269
	Inbound	87	119	163	212	248	274

Section 6: Mechanical Information



NOTES

1. DIMENSIONS AND TOLERANCE PER ASME Y 14.5M – 1994
2. DIMENSION IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER PARALLEL TO PRIMARY DATUM C
3. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
4. SOLDER BALL POSITION DESIGNATION PER JESD 95-1, SPP-010.
5. THE ENCAPSULATION SIZE (X,Y) WILL VARY WITH CAVITY SIZE.

Figure 33: 256-pin TBGA Package Drawing

Section 7: Ordering Information

Part Number	***Package***	***Ambient Temperature***	
BCM5820KTB	256-TBGA	0° to 70° C	32° to 158° F

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, RFC2104: 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] M. Dworkin, National Institute of Standards and Technology Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation, Methods and Techniques, December, 2001

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

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

[18] National Institute of Standards and Technology, FIPS PUB 186-2: Digital Signature Standard. Jamuary, 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 will result in 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 61.

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 61.

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 is enabled, the length of input data to be encrypted or decrypted must be a multiple of the block size, which is eight bytes. The BCM5820 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 BCM5820 control microcode to start processing, waiting for a PCI master mode access that will never begin.

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

BCM5820 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.

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 BCM5820 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 BCM5820 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 59.

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 BCM5820 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 59 should be applied.

ALIGNMENT RESTRICTIONS

Table 26 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 BCM5820 will handle such an extreme situation correctly from a functional standpoint, albeit with reduced performance from the huge number of descriptor fetches.

Table 26: 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 will be read
Master Command Record Structures	32-bit	(1 + #pkts*8) 32-bit words)

Table 27 shows alignment requirements for all memory-resident data in DH/RSA/DSA/Modular Arithmetic operations.

Table 27: 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 will be read
Master Command Record	32-bit	(1 + #pkts*8) 32-bit words)

Broadcom Corporation

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