



Freescale Semiconductor, Inc.

MPC1848xxUM
5/2004
REV 1.1

Freescale Semiconductor, Inc.

PRELIMINARY DATA
SUBJECT TO CHANGE WITHOUT NOTICE

MPC184 8xx Security Co-Processor User's Manual

**For More Information On This Product,
Go to: www.freescale.com**

Home Page:

www.freescal.com

email:

support@freescal.com

USA/Europe or Locations Not Listed:

Freescal Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
480-768-2130
support@freescal.com

Europe, Middle East, and Africa:

Freescal Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescal.com

Japan:

Freescal Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 2666 8080
support.japan@freescal.com

Asia/Pacific:

Freescal Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescal.com

For Literature Requests Only:

Freescal Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescalSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescal Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document. Freescal Semiconductor reserves the right to make changes without further notice to any products herein. Freescal Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescal Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescal Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescal Semiconductor does not convey any license under its patent rights nor the rights of others. Freescal Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescal Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescal Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescal Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescal Semiconductor was negligent regarding the design or manufacture of the part.





Overview	1
Signal Descriptions	2
Address Map	3
Execution Units	4
MPC185 Descriptors	5
Crypto-Channels	6
Controller	7
60x Bus Interface Module	8
User's Manual Revision History	A
Index	IND



Freescale Semiconductor, Inc.

1	Overview
2	Signal Descriptions
3	Address Map
4	Execution Units
5	MPC185 Descriptors
6	Crypto-Channels
7	Controller
8	60x Bus Interface Module
A	User's Manual Revision History
IND	Index

Contents

Paragraph Number	Title	Page Number
Chapter 1		
Overview		
1.1	Development History	1-1
1.2	Typical Applications	1-1
1.3	Features	1-1
1.4	Typical System Architecture	1-3
1.5	Architectural Overview	1-4
1.6	Data Packet Descriptors	1-5
1.6.1	External Bus Interface	1-6
1.6.2	The MPC184 Controller	1-6
1.6.3	Host-Managed Register Access	1-7
1.6.4	Static EU Access	1-7
1.6.5	Dynamic EU Access	1-7
1.6.6	Crypto-Channels	1-7
1.7	Execution Units (EUs)	1-8
1.7.1	Public Key Execution Unit (PKEU)	1-9
1.7.1.1	Elliptic Curve Operations	1-9
1.7.1.2	Modular Exponentiation Operations	1-9
1.7.2	Data Encryption Standard Execution Unit (DEU)	1-10
1.7.3	Arc Four Execution Unit (AFEU)	1-10
1.7.4	Advanced Encryption Standard Execution Unit (AESU)	1-10
1.7.5	Message Digest Execution Unit (MDEU) Module	1-10
1.7.6	Random Number Generator (RNG)	1-11
1.7.7	8KB General Purpose RAM (gpRAM)	1-11
1.8	Performance Estimates	1-11

Chapter 2

Signal Descriptions

2.1	Signal Descriptions	2-13
2.2	Pin Assignments	2-17

Contents

Paragraph Number	Title	Page Number
---------------------	-------	----------------

Chapter 3 Address Map

3.1	Address Map	3-1
3.2	Base Address Register	3-5
3.3	Slave Parity Address Register.....	3-6

Chapter 4 Execution Units

4.1	Public Key Execution Units (PKEU).....	4-2
4.1.1	PKEU Register Map	4-2
4.1.2	PKEU Mode Register	4-2
4.1.3	PKEU Key Size Register	4-4
4.1.4	PKEU Data Size Register	4-5
4.1.5	PKEU Reset Control Register.....	4-6
4.1.6	PKEU Status Register	4-7
4.1.7	PKEU Interrupt Status Register	4-8
4.1.8	PKEU Interrupt Control Register	4-10
4.1.9	PKEU EU_GO Register.....	4-11
4.1.10	PKEU Parameter Memories.....	4-12
4.1.10.1	PKEU Parameter Memory A	4-12
4.1.10.2	PKEU Parameter Memory B	4-12
4.1.10.3	PKEU Parameter Memory E	4-12
4.1.10.4	PKEU Parameter Memory N	4-12
4.2	Data Encryption Standard Execution Units (DEU)	4-12
4.2.1	DEU Register Map.....	4-13
4.2.2	DEU Mode Register.....	4-13
4.2.3	DEU Key Size Register	4-14
4.2.4	DEU Data Size Register	4-15
4.2.5	DEU Reset Control Register.....	4-16
4.2.6	DEU Status Register	4-17
4.2.7	DEU Interrupt Status Register	4-18
4.2.8	DEU Interrupt Control Register.....	4-20
4.2.9	DEU EU_GO Register.....	4-22
4.2.10	DEU IV Register.....	4-23
4.2.11	DEU Key Registers.....	4-23
4.2.12	DEU FIFOs	4-23
4.3	ARC Four Execution Unit (AFEU)	4-23
4.3.1	AFEU Register Map	4-24
4.3.2	AFEU Mode Register	4-24
4.3.2.1	Host-provided Context via Prevent Permute	4-24

Contents

Paragraph Number	Title	Page Number
4.3.2.2	Dump Context.....	4-25
4.3.3	AFEU Key Size Register	4-26
4.3.4	AFEU Context/Data Size Register	4-27
4.3.5	AFEU Reset Control Register.....	4-28
4.3.6	AFEU Status Register	4-29
4.3.7	AFEU Interrupt Status Register	4-30
4.3.8	AFEU Interrupt Control Register	4-32
4.3.9	AFEU End of Message Register.....	4-33
4.3.10	AFEU Context	4-34
4.3.10.1	AFEU Context Memory	4-34
4.3.10.2	AFEU Context Memory Pointer Register.....	4-34
4.3.11	AFEU Key Registers	4-35
4.3.12	AFEU FIFOs.....	4-35
4.4	Message Digest Execution Units (MDEU).....	4-35
4.4.1	MDEU Register Map	4-35
4.4.2	MDEU Mode Register	4-36
4.4.2.1	Recommended settings for MDEU Mode Register	4-37
4.4.3	MDEU Key Size Register	4-38
4.4.4	MDEU Data Size Register	4-38
4.4.5	MDEU Reset Control Register	4-39
4.4.6	MDEU Status Register.....	4-40
4.4.7	MDEU Interrupt Status Register.....	4-41
4.4.8	MDEU Interrupt Control Register	4-42
4.4.9	MDEU EU_GO Register	4-44
4.4.10	MDEU Context Registers	4-44
4.4.11	MDEU Key Registers	4-45
4.4.12	MDEU FIFOs	4-46
4.5	Random Number Generator (RNG).....	4-46
4.5.1	Overview.....	4-46
4.5.2	Functional Description.....	4-46
4.5.3	RNG Register Map	4-47
4.5.4	RNG Mode Register	4-47
4.5.5	RNG Data Size Register	4-48
4.5.6	RNG Reset Control Register.....	4-49
4.5.7	RNG Status Register	4-49
4.5.8	RNG Interrupt Status Register	4-51
4.5.9	RNG Interrupt Control Register	4-51
4.5.10	RNG EU_GO Register.....	4-52
4.5.11	RNG FIFO	4-53
4.6	Advanced Encryption Standard Execution Units (AESU)	4-53
4.6.1	AESU Register Map	4-53

Contents

Paragraph Number	Title	Page Number
4.6.2	AESU Mode Register	4-54
4.6.3	AESU Key Size Register	4-55
4.6.4	AESU Data Size Register	4-56
4.6.5	AESU Reset Control Register.....	4-57
4.6.6	AESU Status Register	4-58
4.6.7	AESU Interrupt Status Register.....	4-59
4.6.8	AESU Interrupt Control Register	4-61
4.6.9	AESU End of Message Register.....	4-63
4.6.9.1	AESU Context Registers	4-63
4.6.9.2	Context for CBC Mode.....	4-64
4.6.9.3	Context for Counter Mode.....	4-64
4.6.9.4	AESU Key Registers	4-65
4.6.9.5	AESU FIFOs.....	4-65

Chapter 5 MPC184 Descriptors

5.1	Data Packet Descriptor Overview.....	5-1
5.2	Descriptor Structure	5-1
5.2.1	Descriptor Header	5-2
5.2.2	Descriptor Length and Pointer Fields	5-5
5.3	Descriptor Chaining.....	5-7
5.3.1	Null Fields	5-8
5.4	Descriptor Classes.....	5-9
5.4.1	Static Descriptors.....	5-9
5.4.2	Dynamic Descriptors	5-12

Chapter 6 Crypto-Channels

6.1	Crypto-Channel Registers.....	6-2
6.1.1	Crypto-Channel Configuration Register (CCCR).....	6-2
6.1.2	Crypto-Channel Pointer Status Registers (CCPSR)	6-4
6.1.3	Crypto-Channel Current Descriptor Pointer Register (CDPR).....	6-10
6.1.4	Fetch Register (FR).....	6-11
6.1.5	Descriptor Buffer (DB)	6-12
6.1.5.1	Descriptor Header	6-13
6.1.5.2	Descriptor Length/Pointer Pairs	6-13
6.1.5.3	Next Descriptor Pointer	6-14
6.2	Interrupts.....	6-14
6.2.1	Channel Done Interrupt	6-14
6.2.2	Channel Error Interrupt.....	6-14

Contents

Paragraph Number	Title	Page Number
6.2.3	Channel Reset	6-14
6.2.3.1	Hardware Reset.....	6-15
6.2.3.2	Channel Specific Software Reset.....	6-15

Chapter 7 Controller

7.1	Controller Registers	7-1
7.1.1	EU Assignment Control Registers (EUACR).....	7-1
7.1.2	EU Assignment Status Register (EUASR)	7-2
7.1.3	Interrupt Mask Registers (IMR)	7-3
7.1.4	Interrupt Status Registers.....	7-4
7.1.5	Interrupt Clear Register (ICR)	7-6
7.1.6	ID Register.....	7-8
7.1.7	Master Control Register (MCR)	7-9
7.1.8	Master TEA Error Address Register (MTAR).....	7-10
7.1.9	EU Access.....	7-11
7.1.10	Multiple EU Assignment	7-11
7.1.11	Multiple Channels.....	7-12
7.1.12	Priority Arbitration	7-12
7.1.13	Round Robin Snapshot Arbiters	7-13
7.1.14	Bus Access.....	7-13

Chapter 8 8xx Bus Interface Module

8.1	8xx Interface	8-1
8.1.1	Bus Access.....	8-1
8.1.2	8xx Initiator.....	8-1
8.1.3	Parity Errors.....	8-2
8.1.4	8xx Read	8-2
8.1.4.1	Target Aborts	8-2
8.1.5	Initiator Write.....	8-2
8.1.6	Misaligned Data.....	8-3
8.1.7	8xx Target	8-4



Contents

Paragraph Number	Title	Page Number
---------------------	-------	----------------

Freescale Semiconductor, Inc.

Figures

Figure Number	Title	Page Number
1-1	MPC184 Connected to PowerQuicc 8xx Bus	1-3
1-2	MPC184 Connected to host CPU via PCI bus	1-4
1-3	MPC184 Functional Blocks	1-5
2-1	MPC184 Pin Diagram	2-17
3-1	Base Address Register	3-6
3-2	Slave Parity Address Register	3-6
4-1	PKEU Mode Register: Definition 1	4-3
4-2	PKEU Mode Register: Definition 2	4-3
4-3	PKEU Key Size Register	4-5
4-4	PKEU Data Size Register	4-6
4-5	PKEU Reset Control Register	4-6
4-6	PKEU Status Register	4-7
4-7	PKEU Interrupt Status Register	4-9
4-8	PKEU Interrupt Control Register	4-10
4-9	PKEU EU_GO Register	4-11
4-10	DEU Mode Register	4-14
4-11	DEU Key Size Register	4-15
4-12	DEU Data Size Register	4-16
4-13	DEU Reset Control Register	4-16
4-14	DEU Status Register	4-17
4-15	DEU Interrupt Status Register	4-19
4-16	DEU Interrupt Control Register	4-21
4-17	DEU EU_GO Register	4-23
4-18	AFEU Mode Register	4-25
4-19	AFEU Key Size Register	4-26
4-20	AFEU Data Size Register	4-28
4-21	AFEU Reset Control Register	4-28
4-22	AFEU Status Register	4-29
4-23	AFEU Interrupt Status Register	4-31
4-24	AFEU Interrupt Control Register	4-32
4-25	AFEU End of Message Register	4-34
4-26	MDEU Mode Register	4-36
4-27	MDEU Key Size Register	4-38
4-28	MDEU Data Size Register	4-39
4-29	MDEU Reset Control Register	4-39
4-30	MDEU Status Register	4-40
4-31	MDEU Interrupt Status Register	4-41
4-32	MDEU Interrupt Control Register	4-43
4-33	MDEU EU_GO Register	4-44
4-34	RNG Mode Register	4-48
4-35	RNG Data Size Register	4-48

Figures

Figure Number	Title	Page Number
4-36	RNG Reset Control Register	4-49
4-37	RNG Status Register	4-50
4-38	RNG Interrupt Status Register	4-51
4-39	RNGA Interrupt Control Register	4-52
4-40	RNG EU_GO Register	4-53
4-41	AESU Mode Register	4-54
4-42	AESU Key Size Register	4-56
4-43	AESU Data Size Register	4-57
4-44	AESU Reset Control Register	4-57
4-45	AESU Status Register	4-58
4-46	AESU Interrupt Status Register	4-60
4-47	AESU Interrupt Control Register	4-62
4-48	AESU End of Message Register	4-63
4-49	AESU Context Registers	4-64
5-2	Descriptor Header	5-2
5-1	Example Data Packet Descriptor	5-2
5-3	Op_x sub fields	5-4
5-4	Descriptor Length Field	5-5
5-5	Descriptor Pointer Field	5-6
5-6	Next Descriptor Pointer Field	5-7
5-7	Chain of Descriptors	5-8
6-1	Crypto-Channel Configuration Register	6-2
6-2	Crypto-Channel Pointer Status Register 1	6-5
6-3	Crypto-Channel Pointer Status Register 2	6-5
6-4	Crypto-Channel Current Descriptor Pointer Register	6-10
6-5	Fetch Register	6-12
6-6	Data Packet Descriptor Buffer	6-13
7-1	EU Assignment Control Registers	7-2
7-2	EU Assignment Status Register	7-3
7-3	Interrupt Mask Register 1	7-3
7-4	Interrupt Mask Register 2	7-4
7-5	Interrupt Status Register 1	7-5
7-6	Interrupt Status Register 2	7-5
7-7	Interrupt Clear Register 1	7-6
7-8	Interrupt Clear Register 2	7-7
7-9	ID Register	7-8
7-10	Master Control Register	7-9
7-11	Master TEA Address Register	7-11
8-1	Data Alignment Example	8-4

Tables

Table Number	Title	Page Number
1-1	Example Data Packet Descriptor	1-5
1-2	Estimated Bulk Data Encryption Performance (Mbps)	1-11
2-1	Signal Descriptions	2-13
3-1	Module Base Address Map	3-1
3-2	Preliminary System Address Map Showing All Registers	3-2
3-3	Base Address Register Definition	3-6
3-4	Slave Parity Address Register	3-7
4-1	Mode Register Routine Definitions	4-3
4-2	PKEU Reset Control Register Signals	4-7
4-3	PKEU Status Register Signals	4-8
4-4	PKEU Interrupt Status Register Signals	4-9
4-5	PKEU Interrupt Control Register Signals	4-10
4-6	DEU Mode Register Signals	4-14
4-7	DEU Key Size Register.....	4-15
4-8	DEU Reset Control Register Signals	4-17
4-9	DEU Status Register Signals.....	4-18
4-10	DEU Interrupt Status Register Signals.....	4-19
4-11	DEU Interrupt Control Register Signals	4-21
4-12	AFEU Mode Register Signals.....	4-25
4-13	AFEU Reset Control Register Signals	4-29
4-14	AFEU Status Register Signals	4-30
4-15	AFEU Interrupt Status Register	4-31
4-16	AFEU Interrupt Control Register.....	4-33
4-17	MDEU Mode Register	4-36
4-18	MDEU Reset Control Register Signal	4-39
4-19	MDEU Status Register Signals	4-40
4-20	MDEU Interrupt Status Register Signals	4-42
4-21	MDEU Interrupt Control Register Signals.....	4-43
4-22	RNG Mode Register Definitions.....	4-48
4-23	RNG Reset Control Register Signals	4-49
4-24	RNG Status Register Signals	4-50
4-25	RNG Interrupt Status Register Signals	4-51
4-26	RNG Interrupt Control Register Signals	4-52
4-27	AESU Mode Register Signals.....	4-54
4-28	AESU Reset Control Register Signals	4-57

Tables

Table Number	Title	Page Number
4-29	AESU Status Register Signals	4-59
4-30	AESU Interrupt Status Register Signals	4-60
4-31	AESU Interrupt Control Register Signals	4-62
4-32	Counter Modulus.....	4-65
5-1	Header Bit Definitions	5-3
5-2	EU_Select Values	5-4
5-3	Descriptor Types	5-4
5-6	Descriptor Length/Pointer Mapping	5-6
5-4	Descriptor Length Field Mapping.....	5-6
5-5	Descriptor Pointer Field Mapping.....	5-6
5-7	Descriptor Pointer Field Mapping.....	5-7
5-8	Actual Descriptor DPD_RC4-SA_NEWCTX	5-9
5-9	Actual Descriptor DPD_RC4-SA_LDCTX	5-10
5-10	Actual Descriptor DPD_RC4-SA_CRYPT	5-10
5-11	Actual Descriptor DPD_RC4-SA_CRYPT_ULCTX.....	5-11
5-12	Representative Descriptor DPD_DEU_CTX_CRYPT	5-12
6-1	Crypto-Channel Configuration Register Signals	6-3
6-2	Burst Size Definition.....	6-4
6-3	Crypto-Channel Pointer Status Register 1 Signals.....	6-5
6-4	Crypto-Channel Pointer Status Register 2 Signals.....	6-6
6-5	STATE Field Values	6-7
6-6	Crypto-Channel Pointer Status Register Error Field Definitions.....	6-9
6-7	Crypto-Channel Pointer Status Register PAIR_PTR Field Values.....	6-10
6-8	Crypto-Channel Current Descriptor Pointer Register Signals	6-11
6-9	Fetch Register Signals.....	6-12
7-1	Channel Assignment Value	7-2
7-2	Interrupt Mask, Status, and Clear Register 1 Signals.....	7-7
7-3	Interrupt Mask, Status, and Clear Register 2 Signals.....	7-8
7-4	Master Control Register Signals	7-9
7-5	Master Control Register 2 signals.....	7-10
7-6	Master TEA Address Register Bit Definitions	7-11

Chapter 1

Overview

This chapter provides an overview of the MPC184 Security Processor, including a brief development history, target applications, key features, typical system architecture, device architectural overview, and a performance summary.

1.1 Development History

The MPC184 belongs to the Smart Networks platform's S1 family of security processors developed for the commercial networking market. This product family is derived from security technologies Motorola has developed over the last 30 years, primarily for government applications. The fifth-generation execution units (EU) have been proven in Motorola semi-custom ICs and in other members of the S1 family, including the MPC180, MPC190, and MPC185.

1.2 Typical Applications

The MPC184 is suited for applications such as the following:

- SOHO VPN routers
- Customer Premise Equipment
- eCommerce servers
- Wireless Access Points
- Dedicated Encryption Modules

1.3 Features

The MPC184 is a flexible and powerful addition to any networking or computing system using the Motorola PowerQUICC line of integrated communications processors, or any system supporting 32-bit PCI. The MPC184 is designed to off load computationally intensive security functions, such as key generation and exchange, authentication, and bulk encryption from the host processor.

The MPC184 is optimized to process all the algorithms associated with IPSec, IKE, WTLS/WAP, SSL/TLS, DOCSIS BPI+, 802.16, and 802.11(WEP). In addition, the Motorola family of security co-processors are the only devices on the market capable of executing Elliptic Curve Cryptography which is especially important for secure wireless communications.

Features

MPC184 features include the following:

- Public Key Execution Unit (PKEU) that supports the following:
 - RSA and Diffie-Hellman
 - Programmable field size up to 2048-bits
 - Elliptic curve cryptography
 - F_2m and $F(p)$ modes
 - Programmable field size up to 511-bits
- Data Encryption Standard Execution Unit (DEU)
 - DES, 3DES
 - Two key (K1, K2, K1) or Three Key (K1, K2, K3)
 - ECB and CBC modes for both DES and 3DES
- Advanced Encryption Standard Unit (AESU)
 - Implements the Rijndael symmetric key cipher
 - Key lengths of 128, 192, and 256 bits. Two key
 - ECB, CBC, and Counter modes
- ARC Four Execution Unit (AFEU)
 - Implements a stream cipher compatible with the RC4 algorithm
 - 40- to 128-bit programmable key
- Message Digest Execution Unit (MDEU)
 - SHA with 160-bit or 256-bit message digest
 - MD5 with 128-bit message digest
 - HMAC with either algorithm
- Random number generator (RNG)
- 8xx compliant external bus interface, with master/slave logic.
 - 32-bit address/32 -bit data
 - up to 66MHz operation
- Optional PCI 2.2 compliant external bus interface, with master/slave logic.
 - 32-bit address/data
 - up to 66MHz operation
- 4 Crypto-channels, each supporting multi-command descriptor chains
 - Static and/or dynamic assignment of crypto-execution units via an integrated controller
 - Buffer size of 512 Bytes for each execution unit, with flow control for large data sizes
- 8KB of internal scratch pad memory for key, IV and context storage
- 1.5V supply, 3.3V I/O
- 252MAP BGA, 21x 21mm package body size
- 1.0W power dissipation

1.4 Typical System Architecture

The MPC184 is designed to integrate easily into any system using the 8xx or PCI bus protocol. The MPC184 is ideal in any system using a Motorola PowerQUICC communications processor (as shown in Figure 1) or any system using PCI. The ability of the MPC184 to be a master on the 8xx bus allows the co-processor to off load the data movement bottleneck normally associated with slave devices.

The host processor accesses the MPC184 through its device drivers using system memory for data storage. The MPC184 resides in the memory map of the processor, therefore when an application requires cryptographic functions, it simply creates descriptors for the MPC184 which define the cryptographic function to be performed and the location of the data. The MPC184's mastering capability permits the host processor to set up a crypto-channel with a few short register writes, leaving the MPC184 to perform reads and writes on system memory to complete the required task.

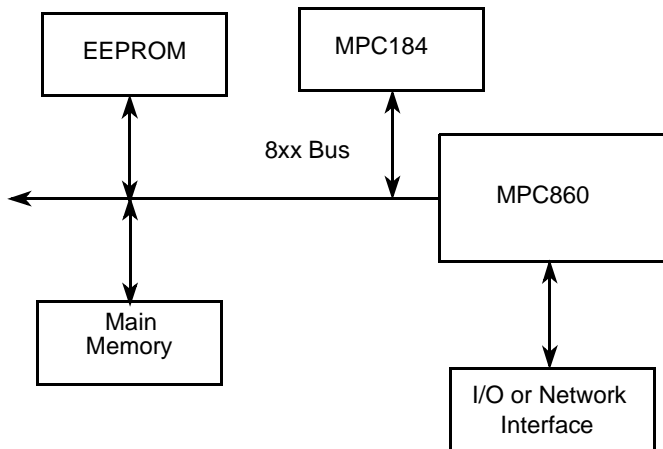


Figure 1-1. MPC184 Connected to PowerQuicc 8xx Bus

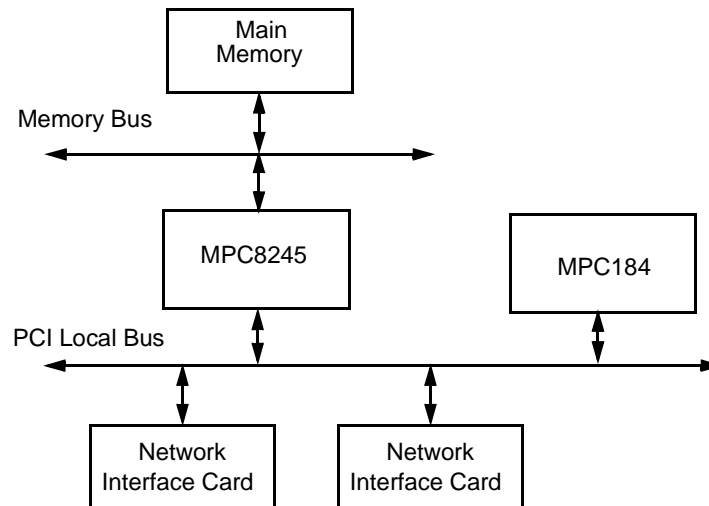


Figure 1-2. MPC184 Connected to host CPU via PCI bus

1.5 Architectural Overview

A block diagram of the MPC184 internal architecture is shown in Figure 1-1. The mode selectable 8xx/PCI bus interface module is designed to transfer 32-bit words between the external bus and any register inside the MPC184. An operation begins with a write of a pointer to a crypto-channel fetch register which points to a data packet descriptor. The channel then requests the descriptor and decodes the operation to be performed. The channel then makes requests of the controller to assign crypto execution units and fetch the keys, IV's and data needed to perform the given operation. The controller satisfies the requests by assigning execution units to the channel and by making requests to the master interface per the programmable priority scheme. As data is processed, it is written to the individual execution units output buffer and then back to system memory via the bus interface module.

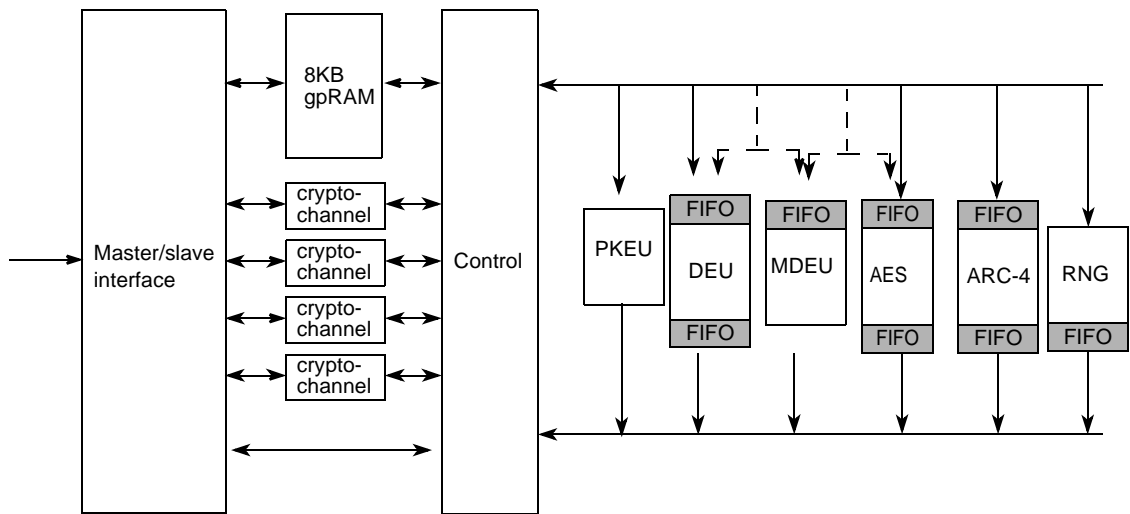


Figure 1-3. MPC184 Functional Blocks

1.6 Data Packet Descriptors

As an IPsec accelerator, the MPC184’s controller has been designed for easy use and integration with existing systems and software. All cryptographic functions are accessible through data packet descriptors, some of which have been defined as multifunction to facilitate IPsec applications. A data packet descriptor is diagrammed in Table 1-1.

Table 1-1. Example Data Packet Descriptor

Field Name	Value/Type	Description
DPD_DES_CTX_CRYPT	tbd	Representative header for DES using Context to Encrypt
LEN_CTXIN PTR_CTXIN	length pointer	Number of bytes to be written Pointer to Context (IV) to be written into DES engine
LEN_KEY PTR_KEY	length pointer	Number of bytes in key Pointer to block cipher key
LEN_DATAIN PTR_DATAIN	length pointer	Number of bytes of data to be ciphered Pointer to data to perform cipher upon
LEN_DATAOUT PTR_DATAOUT	length pointer	Number of bytes of data after ciphering Pointer to location where cipher output is to be written
LEN_CTXOUT PTR_CTXOUT	length pointer	Length of output Context (IV) Pointer to location where altered Context is to be written
nul length nul pointer	length pointer	Zeroes for fixed length descriptor filter Zeroes for fixed length descriptor filter
nul length nul pointer	length pointer	Zeroes for fixed length descriptor filter Zeroes for fixed length descriptor filter
PTR_NEXT	pointer	Pointer to next data packet descriptor

Each data packet descriptor contains the following:

- Header—The header describes the required services and encodes information that indicates which EUs to use and which modes to set.
- Seven data length/data pointer pairs—The data length indicates the number of contiguous bytes of data to be transferred. The data pointer indicates the starting address of the data, key, or context in system memory.
- Next descriptor pointer

A data packet descriptor ends with a pointer to the next data packet descriptor. Therefore, once a descriptor is processed and if the value of this pointer is non-zero, it is used to request a burst read of the next descriptor.

Processing of the next descriptor (and whether or not a done signal is generated) is determined by the programming of crypto-channel's configuration register. Two modes of operation are supported:

- Signal done at end of descriptor
- Signal done at end of descriptor chain

The crypto-channel can signal done via an interrupt or by a write-back of the descriptor header after processing a data packet descriptor. The value written back is identical to that of the header, with the exception that a DONE field is set.

Occasionally, a descriptor field may not be applicable to the requested service. For example, if using DES in ECB mode, the contents of the IV field do not affect the result of the DES computation. Therefore, when processing data packet descriptors, the crypto-channel skips any pointer that has an associated length of zero.

1.6.1 External Bus Interface

The External Bus Interface (EBI) manages communication between the MPC184's internal execution units and the external bus. The interface is mode selectable between the 8xx bus protocols, used by the PowerQuicc family of integrated communications processors, and the PCI 2.2 bus protocol. The MPC184 is unique in its ability to act as a bus master on the 8xx bus. All on-chip resources are memory mapped, and the target accesses and initiator writes from the MPC184 must be addressed on word boundaries. The MPC184 will perform initiator reads on byte boundaries and will adjust the data to place on word boundaries as appropriate. The 8xx bus mastering interface allows the MPC184 to off-load both crypto processing and data movement from the PowerQuicc processor, freeing the CPU for other networking system functions, allowing the chip set to achieve best in class performance levels.

1.6.2 The MPC184 Controller

The MPC184 controller manages on-chip resources, including individual execution units (EUs), FIFOs, the EBI, and the internal buses that connect all the various modules. The controller receives service requests from the EBI and various crypto-channels, and schedules the required activities. The controller can configure each of the on-chip resources in three modes:

- Host-controlled mode—The host is directly responsible for all data movement into and out of the resource.
- Static mode—The user can reserve a specific execution unit to a specific crypto-channel.
- Dynamic mode—A crypto channel can request a particular service from any available execution unit.

1.6.3 Host-Managed Register Access

All EUs can be used entirely through register read/write access. It is strongly recommended that read/write access only be performed on a EU that is statically assigned to an idle crypto-channel. Such an assignment is the only method for the host to inform the controller that a particular EU is in use.

1.6.4 Static EU Access

The Controller can be configured to reserve one or more EUs to a particular crypto-channel. Doing so permits locking the EU to a particular context. When in this mode, the crypto-channel can be used by multiple descriptors representing the same context without unloading and reloading the context at the end of each descriptor. This mode presents considerable performance improvement over dynamic access, but only when the MPC184 is supporting a single context (or a single session is being streamed.)

1.6.5 Dynamic EU Access

Processing begins when a data packet descriptor pointer is written to the next descriptor pointer register of one of the crypto-channels. Prior to fetching the data referred to by the descriptor and based on the services requested by the descriptor header in the descriptor buffer, the controller dynamically reserves usage of an EU to the crypto-channel. If all appropriate EUs are already dynamically reserved by other crypto-channels, the crypto-channel stalls and waits to fetch data until the appropriate EU is available.

If multiple crypto-channels simultaneously request the same EU, the EU is assigned on a round-robin basis. Once the required EU has been reserved, the crypto-channel fetches and loads the appropriate data packets, operates the EU, unloads data to system memory, and releases the EU for use by another crypto-channel. If a crypto-channel attempts to reserve a statically-assigned EU (and no appropriate EUs are available for dynamic assignment), an interrupt is generated and status indicates illegal access. When dynamic assignment is used, each encryption/decryption packet must contain context that is particular to the context being supported.

1.6.6 Crypto-Channels

The MPC184 includes four crypto-channels that manage data and EU function. Each crypto-channel consists of the following:

- Control registers containing information about the transaction in process

Execution Units (EUs)

- A status register containing an indication of the last unfulfilled bus request
- A pointer register indicating the location of a new descriptor to fetch
- Buffer memory used to store the active data packet descriptor (See Section 1.6, “Data Packet Descriptors.”)

Crypto-channels analyze the data packet descriptor header and request from the controller the first required cryptographic service. The controller implements a programmable prioritization scheme that allows the user to dictate the order in which the four crypto-channels are serviced. After the controller grants access to the required EU, the crypto-channel and the controller perform the following steps:

1. Set the appropriate Mode bits available in the EU for the required service.
2. Fetch context and other parameters as indicated in the data packet descriptor buffer and use these to program the EU.
3. Fetch data as indicated and place in either the EU’s input FIFO or the EU itself (as appropriate).
4. Wait for EU to complete processing.
5. Upon completion, unload results and context and write them to external memory as indicated by the data packet descriptor buffer.
6. If multiple services requested, go back to step 2.
7. Reset the appropriate EU if it is dynamically assigned. Note that if statically assigned, a EU is reset only upon direct command written to the MPC184.
8. Perform descriptor completion notification as appropriate. This notification comes in one of two forms—interrupt or header write back modification—and can occur either at the end of every descriptor or at the end of a descriptor chain.

1.7 Execution Units (EUs)

“Execution unit” is the generic term for a functional block that performs the mathematical permutations required by protocols used in cryptographic processing. The EUs are compatible with IPsec, WAP/WTLS, IKE, SSL/TLS and 802.11i processing, and can work together to perform high level cryptographic tasks. The MPC184’s execution units are as follows:

- PKEU for computing asymmetric key mathematics, including Modular Exponentiation (and other Modular Arithmetic functions) or ECC Point Arithmetic
- DEU for performing block symmetric cryptography using DES and 3DES
- AFEU for performing RC-4 compatible stream symmetric cryptography
- AESU for performing the Advanced Encryption Standard algorithm
- MDEU for hashing data
- RNG for random number generation

1.7.1 Public Key Execution Unit (PKEU)

The PKEU is capable of performing many advanced mathematical functions to support both RSA and ECC public key cryptographic algorithms. ECC is supported in both $F(2)^m$ (polynomial-basis) and $F(p)$ modes. This EU supports all levels of functions to assist the host microprocessor to perform its desired cryptographic function. For example, at the highest level, the accelerator performs modular exponentiations to support RSA and performs point multiplies to support ECC. At the lower levels, the PKEU can perform simple operations such as modular multiplies.

1.7.1.1 Elliptic Curve Operations

The PKEU has its own data and control units, including a general-purpose register file in the programmable-size arithmetic unit. The field or modulus size can be programmed to any value between 160 bits and 512 bits in programmable increments of 8, with each programmable value i supporting all actual field sizes from $i \cdot 8 - 7$ to $i \cdot 8$. The result is hardware supporting a wide range of cryptographic security. Larger field / modulus sizes result in greater security but lower performance; processing time is determined by field or modulus size. For example, a field size of 160 is roughly equivalent to the security provided by 1024 bit RSA. A field size set to 208 roughly equates to 2048 bits of RSA security.

The PKEU contains routines implementing the atomic functions for elliptic curve processing—point arithmetic and finite field arithmetic. The point operations (multiplication, addition and doubling) involve one or more finite field operations which are addition, multiplication, inverse, and squaring. Point add and double each use of all four finite field operations. Similarly, point multiplication uses all EC point operations as well as the finite field operations. All these functions are supported both in modular arithmetic as well as polynomial basis finite fields.

1.7.1.2 Modular Exponentiation Operations

The PKEU is also capable of performing ordinary integer modulo arithmetic. This arithmetic is an integral part of the RSA public key algorithm; however, it can also play a role in the generation of ECC digital signatures and Diffie-Hellman key exchanges.

Modular arithmetic functions supported by the MPC184's PKEU include the following:

- $R \cdot 2 \bmod N$
- $A' \cdot E \bmod N$
- $(A \times B) R^{-1} \bmod N$
- $(A \times B) R^{-2} \bmod N$
- $(A+B) \bmod N$
- $(A-B) \bmod N$

Where the following variable definitions: $A' = AR \bmod N$, N is the modulus vector, A and B are input vectors, E is the exponent vector, R is 2^s , where s is the bit length of the N vector rounded up to the nearest multiple of 32.

The PKEU can perform modular arithmetic on operands up to 2048 bits in length. The modulus must be larger than or equal to 129 bits. The PKEU uses the Montgomery

modular multiplication algorithm to perform core functions. The addition and subtraction functions exist to help support known methods of the Chinese Remainder Theorem (CRT) for efficient exponentiation.

1.7.2 Data Encryption Standard Execution Unit (DEU)

The DES execution unit (DEU) performs bulk data encryption/decryption, in compliance with the Data Encryption Standard algorithm (ANSI x3.92). The DEU can also compute 3DES and extension of the DES algorithm in which each 64-bit input block is processed three times. The MPC184 supports 2 key (K1=K3) or 3 key 3DES.

The DEU operates by permuting 64-bit data blocks with a shared 56-bit key and an initialization vector (IV). The MPC184 supports two modes of IV operation: ECB (Electronic Code Book) and CBC (Cipher Block Chaining).

1.7.3 Arc Four Execution Unit (AFEU)

The AFEU accelerates a bulk encryption algorithm compatible with the RC4 stream cipher from RSA Security, Inc. The algorithm is byte-oriented, meaning a byte of plain text is encrypted with a key to produce a byte of ciphertext. The key is variable length and the AFEU supports key lengths from 40 to 128 bits (in byte increments), providing a wide range of security strengths. RC4 is a symmetric algorithm, meaning each of the two communicating parties share the same key.

1.7.4 Advanced Encryption Standard Execution Unit (AESU)

The AESU is used to accelerate bulk data encryption/decryption in compliance with the Advanced Encryption Standard algorithm Rijndael. The AESU executes on 128 bit blocks with a choice of key sizes: 128, 192, or 256 bits.

AES is a symmetric key algorithm, the sender and receiver use the same key for both encryption and decryption. The session key and IV(CBC mode) are supplied to the AESU module prior to encryption. The processor supplies data to the module that is processed as 128 bit input. The AESU operates in ECB, CBC, and counter modes.

1.7.5 Message Digest Execution Unit (MDEU) Module

The MDEU computes a single message digest (or hash or integrity check) value of all the data presented on the input bus, using either the MD5, SHA-1 or SHA-256 algorithms for bulk data hashing. With any hash algorithm, the larger message is mapped onto a smaller output space, therefore collisions are potential, albeit not probable. The 160-bit hash value is a sufficiently large space such that collisions are extremely rare. The security of the hash function is based on the difficulty of locating collisions. That is, it is computation infeasible to construct two distinct but similar messages that produce the same hash output.

- The MD5 generates a 128-bit hash, and the algorithm is specified in RFC 1321.

- SHA-1 is a 160-bit hash function, specified by the ANSI X9.30-2 and FIPS 180-1 standards.
- SHA-256 is a 256-bit hash function that provides 256 bits of security against collision attacks.
- The MDEU also supports HMAC computations, as specified in RFC 2104.

1.7.6 Random Number Generator (RNG)

The RNG is a digital integrated circuit capable of generating 32-bit random numbers. It is designed to comply with FIPS 140-1 standards for randomness and non-determinism.

Because many cryptographic algorithms use random numbers as a source for generating a secret value (a nonce), it is desirable to have a private RNG for use by the MPC184. The anonymity of each random number must be maintained, as well as the unpredictability of the next random number. The FIPS-140 compliant private RNG allows the system to develop random challenges or random secret keys. The secret key can thus remain hidden from even the high-level application code, providing an added measure of physical security.

1.7.7 8KB General Purpose RAM (gpRAM)

The MPC184 contains 8KB of internal general purpose RAM that can be used to store keys, IV's and data. The internal scratch pad allows the user to store frequently used context on chip which increases system performance by minimizing setup time. This feature is especially important when dealing with small packets and in systems where bus bandwidth is limited.

1.8 Performance Estimates

Bulk encryption/authentication performance estimates shown in Table 1-2. include data/key/context reads (from memory to MPC184), security processing (internal to MPC184), and writes of completed data/context to memory by MPC184, using typical bus overhead.

Table 1-2. Estimated Bulk Data Encryption Performance (Mbps)

	DES CBC	3DES CBC	AES 128	AES 256	ARC4	MD5	SHA-1	3DES/ HMAC- SHA-1(Rx)
64 byte	43	36	38	32	43	38	34	29
128 byte	75	55	60	51	75	66	59	50
256 byte	119	76	83	70	118	100	87	74
512 byte	173	95	104	88	171	135	114	97
1024 byte	223	109	118	100	221	163	136	115
1536 byte	247	114	124	105	252	176	144	123



Performance Estimates

The MPC184 supports single pass processing of encryption/message authentication. All performance measurements assume descriptor generation and bus availability (66Mhz, 32bit 8xx bus with typical SDRAM read/write latency) are not constraints.

Chapter 2

Signal Descriptions

This chapter describes the signals used by the MPC184 in 8xx mode, as well as the device pinout.

2.1 Signal Descriptions

Table 2-1 shows the signal descriptions for the MPC184 in 8xx mode.

Table 2-1. Signal Descriptions

Signal Name	Pin Locations	Signal Type	Description
A[0–31]	A5, B5, B6, A6, A7, B7, B8, A8, A9, B9, B10, A10, A11, B11, B12, A12, A13, B13, B14, A14, A15, B15, B16, C16, C15, D15, D16, E16, E15, F15, F16, G16	I/O	Address Bus—Provides the address for the current bus cycle. A0 is the msb.
TSIZ[0-1]	J16, J15	I/O	Transfer Size 0-1—When accessing a slave in the external bus, used by the bus master to indicate the number of operand bytes waiting to be transferred in the current bus cycle.
RD/WR	H16	I/O	Read/Write—Driven by a bus master to indicate the direction of the data transfer. A logic one indicates a read from a slave device and a logic zero indicates a write to a slave device.
BURST	K15	I/O	Burst Transaction—Driven by the bus master to indicate that the current initiated transfer is a burst.
BDIP	K16	I/O	Burst Data in Progress—When accessing a slave device in the external bus, the master on the bus asserts this signal to indicate that the data beat in front of the current one is the one requested by the master. $\overline{\text{BDIP}}$ is negated before the expected last data beat of the burst transfer.
TS	R11	I/O	Transfer Start—Asserted by a bus master to indicate the start of a bus cycle that transfers data to or from a slave device. Driven by the master only when it has gained the ownership of the bus.

Table 2-1. Signal Descriptions (continued)

Signal Name	Pin Locations	Signal Type	Description
\overline{TA}	R10	I/O	Transfer Acknowledge—Indicates that the slave device addressed in the current transaction accepted data sent by the master (write) or has driven the data bus with valid data (read).
\overline{TEA}	T10	I/O	Transfer Error Acknowledge—Indicates that a bus error occurred in the current transaction.
\overline{BI}	R9	I/O	Burst Inhibit—Indicates that the slave device addressed in the current burst transaction cannot support burst transfers.
\overline{IRQ}	D1	O	Interrupt request - Interrupt signal that indicates that one of the modules has asserted its hardware interrupt to indicate that service is needed by the system.
D[0–31]	L16, L15, M16, M15, N16, N15, P16, P15, R15, T15, T14, R14, T13, R13, T12, R12, R6, T5, R5, T4, T3, T2, R2, R1, N1, N2, M1, M2, L1, L2, K1, K2	I/O	Data bus - In write transactions the bus master drives the valid data on this bus. In read transactions the slave drives the valid data on this bus.
DP[0-3]	R16, T11, T6, P2	I	Data Parity — Provides parity generation and checking for transfers to a slave device initiated by the MPC860. Parity generation and checking is not supported for external masters.
\overline{BR}	R7	I/O	Bus Request—Asserted low when a possible master is requesting ownership of the bus.
\overline{BG}	T7	I/O	Bus Grant—Asserted low when the arbiter of the external bus grants the bus to a specific device.
\overline{BB}	T8	I/O	Bus Busy—Asserted low by a master to show that it owns the bus.
\overline{RETRY}	R8	I	Retry—Input used by a slave device to indicate it cannot accept the transaction.
BASE[0:4]	D2, F1, P1, H15, G15	I	Base Address Select - These 5 bits set the initial Base Address for the MPC184 and address the upper 5 bits of the 32-bit address range. After reset the Base Address may be reprogrammed anywhere in the address space via software. As an example, if BASE[0:4] = 00001, the initial Base Address for the MPC184 is 0800_0000.
CLK	H1	I	System Clock input
\overline{RESET}	E1	I	Asynchronous reset signal. Initializes MPC184 to known state.
TPA / NC	G2	O	Test Pad Analog This pin MUST have No Connection

Table 2-1. Signal Descriptions (continued)

Signal Name	Pin Locations	Signal Type	Description
TCK	A3	I	Test Clock If JTAG is NOT used, this pin should be tied to VSS
TDI	C1	I	Test Input If JTAG is NOT used, this pin should be tied to OVDD
TDO	B1	O	Test output If JTAG is NOT used, this pin should be NC
TMS	A2	I	Test Mode Select If JTAG is NOT used, this pin should be tied to OVDD
$\overline{\text{TRST}}$	A4	I	Test Reset If JTAG is NOT used, this pin should be tied to VSS
PLL Range	E2	I	PLL Range 0 (OVSS) = 66-100 MHz PLL band 1 (OVDD) = 33-66 MHz PLL band If operating slower than 33MHz, the PLL must be disabled using the PLL Bypass pin ()
PLL Bypass	P5	I	PLL Bypass 0 (OVSS) = PLL Disabled 1 (OVDD) = PLL Enabled
AVSS	H2	I	Analog PLL Ground
Analog VDD	F2	I	Analog PLL Power (+1.5 V)
VSS	B2, B3, B4, C2, C3, C4, C5, C7, C10, C12, C13, C14, D3, E14, F6, F7, F8, F9, F10, F11, F14, G1, G3, G6, G7, G8, G9, G10, G11, H6, H7, H8, H9, H10, H11, J1, J2, J6, J7, J8, J9, J10, J11, J14, K6, K7, K8, K9, K10, K11, L3, L6, L7, L8, L9, L10, L11, M14, N3, P3, P4, P6, P7, P9, P11, P12, P13, P14, R3, R4, T9	I	Ground



Table 2-1. Signal Descriptions (continued)

Signal Name	Pin Locations	Signal Type	Description
IVDD	E5, E6, E7, E8, E9, E10, E11, E12, F5, F12, G5, G12, H5, H12, J5, J12, K5, K12, L5, L12, M5, M6, M7, M8, M9, M10, M11, M12	I	Core Power (+1.5 V)
OVDD	C6, C8, C9, C11, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, E3, E4, E13, F3, F4, F13, G4, G13, G14, H3, H4, H13, H14, J3, J4, J13, K3, K4, K13, K14, L4, L13, L14, M3, M4, M13, N4, N5, N6, N7, N8, N9, N10, N11, N12, N13, N14, P5, P8, P10	I	I/O Power (+3.3v)

2.2 Pin Assignments

Figure 2-1 shows the pin connections for the MPC184 in 8xx mode.

Figure 2-1. MPC184 Pin Diagram

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A		TMS	TCK	TRST	A0	A3	A4	A7	A8	A11	A12	A15	A16	A19	A20		A
B	TDO	VSS	VSS	VSS	A1	A2	A5	A6	A9	A10	A13	A14	A17	A18	A21	A22	B
C	TDI	VSS	VSS	VSS	VSS	3.3V	VSS	3.3V	3.3V	VSS	3.3V	VSS	VSS	VSS	A24	A23	C
D	IRQ	BASE 0	VSS	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	A25	A26	D
E	RESET	PLL Range	3.3V	3.3V	1.5 V	1.5 V	1.5 V	1.5 V	1.5 V	1.5 V	1.5 V	1.5 V	3.3V	VSS	A28	A27	E
F	BASE 1	Analog Vdd	3.3V	3.3V	1.5 V	VSS	VSS	VSS	VSS	VSS	VSS	1.5 V	3.3V	VSS	A29	A30	F
G	VSS	TPA /NC	VSS	3.3V	1.5 V	VSS	VSS	VSS	VSS	VSS	VSS	1.5 V	3.3V	3.3V	BASE 4	A31	G
H	CLK	AVSS	3.3V	3.3V	1.5 V	VSS	VSS	VSS	VSS	VSS	VSS	1.5 V	3.3V	3.3V	BASE 3	RD/WR	H
J	VSS	VSS	3.3V	3.3V	1.5 V	VSS	VSS	VSS	VSS	VSS	VSS	1.5 V	3.3V	VSS	TSIZ 1	TSIZ 0	J
K	D[30]	D[31]	3.3V	3.3V	1.5 V	VSS	VSS	VSS	VSS	VSS	VSS	1.5 V	3.3V	3.3V	BURST	BDIP	K
L	D[28]	D[29]	VSS	3.3V	1.5 V	VSS	VSS	VSS	VSS	VSS	VSS	1.5 V	3.3V	3.3V	D[01]	D[00]	L
M	D[26]	D[27]	3.3V	3.3V	1.5 V	1.5 V	1.5 V	1.5 V	1.5 V	1.5 V	1.5 V	1.5 V	3.3V	VSS	D[03]	D[02]	M
N	D[24]	D[25]	VSS	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V	D[05]	D[04]	N
P	BASE 2	DP3	VSS	VSS	PLL Bypass	VSS	VSS	3.3V	VSS	3.3V	VSS	VSS	VSS	VSS	D[07]	D[06]	P
R	D[23]	D[22]	VSS	VSS	D[18]	D[16]	BR	RETRY	BI	TA	TS	D[15]	D[13]	D[11]	D[08]	DP0	R
T		D[21]	D[20]	D[19]	D[17]	DP2	BG	BB	VSS	TEA	DP1	D[14]	D[12]	D[10]	D[09]		T
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	



Chapter 3

Address Map

This chapter contains the MPC184 address map. All registers are 32-bit aligned, and are addressed on 32-bit boundaries.

The MPC184’s internal memory resources are within a contiguous block of memory. The size of the internal space is 128 Kilobytes. In 8xx mode, the location of this block within the global 4 Gbyte real memory space is initially mapped to 32 possible address ranges by the setting of 5 base Address pins (see Chapter 2- Signals). The initial base address value established by the base address pins, BASE[0:4], can be over-written with a refined base address by a write to the Base Address Register (see section 3.2). In PCI mode, the Base Address Register is set according to the PCI protocol.

3.1 Address Map

Table 3-1 shows the base address map, and Table 3-2 the precise address map, including all registers in the Execution Units. The 17-bit MPC184 address bus value is shown. Note that these tables show module addresses; the 3 least significant address bits that are used to select bytes within 32-bit-words are not shown.

Table 3-1. Module Base Address Map

MPC184 Address (hex) (AD 16::0)	MPC184 Module	Description	Type
00000-00FFF	Configuration	MPC184 Configuration Setup	Configuration
01000-01FFF	Controller	Arbiter/Controller Control Register Space	Resource Control
02000-02FFF	Channel 1	Crypto-Channel unit 1	Data Control
03000-03FFF	Channel 2	Crypto-Channel unit 2	Data Control
04000-04FFF	Channel 3	Crypto-Channel unit 3	Data Control
05000-05FFF	Channel 4	Crypto-Channel unit 4	Data Control
08000-08FFF	AFEU	ARCFour Execution Unit	CryptoAccelerator
0A000-0AFFF	DEU	DES Execution Unit	CryptoAccelerator
0C000-0CFFF	MDEU	Message Digest Execution Unit	CryptoAccelerator

Table 3-1. Module Base Address Map

MPC184 Address (hex) (AD 16::0)	MPC184 Module	Description	Type
0E000-0EFFF	RNG	Random Number Generator	CryptoAccelerator
10000-10FFF	PKEU	Public Key Execution Unit	CryptoAccelerator
12000-12FFF	AESU	AES Execution Unit	CryptoAccelerator
18000-19FFF	Memory	8K Bytes General Purpose Memory	Memory

Table 3-2 shows the system address map including all functional registers.

Table 3-2. Preliminary System Address Map Showing All Registers

MPC184 Address (hex) (AD 16::0)	MPC184 Module	Description	Type
00F00	Configuration	8xx Mode Base Address	R/W
00800	Configuration	8xx Slave PERR Address	R
01000	Controller	EU Assignment Control	R/W
01008	Controller	Interrupt Mask	R/W
01010	Controller	Interrupt Status	R
01018	Controller	Interrupt Clear	W
01020	Controller	Identification	R
01028	Controller	EU Assignment Status	R
01030	Controller	Master Control	R/W
01038	Controller	Master TEA Address	R
02008	Channel_1	Config register	R/W
02010	Channel_1	Pointer status	R
02040	Channel_1	Current descriptor pointer	R
02048	Channel_1	Fetch register	R/W
02080-020BF	Channel_1	Descriptor buffer[16]	R/W
03008	Channel_2	Config register	R/W
03010	Channel_2	Pointer status	R
03040	Channel_2	Current descriptor pointer	R
03048	Channel_2	Fetch register	R/W
03080-030BF	Channel_2	Descriptor buffer[16]	R/W
04008	Channel_3	Config register	R/W
04010	Channel_3	Pointer status	R
04040	Channel_3	Current descriptor pointer	R

Table 3-2. Preliminary System Address Map Showing All Registers (continued)

MPC184 Address (hex) (AD 16::0)	MPC184 Module	Description	Type
04048	Channel_3	Fetch register	R/W
04080-040BF	Channel_3	Descriptor buffer[16]	R/W
05008	Channel_4	Config register	R/W
05010	Channel_4	Pointer status	R
05040	Channel_4	Current descriptor pointer	R
05048	Channel_4	Fetch register	R/W
05080-050BF	Channel_4	Descriptor buffer[16]	R/W
08000	AFEU	Mode Register	R/W
08008	AFEU	Key Size Register	R/W
08010	AFEU	Context/Data Size Register	R/W
08018	AFEU	Reset Control Register	R/W
08028	AFEU	Status Register	R
08030	AFEU	Interrupt Status Register	R
08038	AFEU	Interrupt Control Register	R/W
08050	AFEU	End of Message Register	W
08100-081FF	AFEU	Context Memory	R/W
08200	AFEU	Context Memory Pointers	R/W
08400	AFEU	Key Register 0	W
08408	AFEU	Key Register 1	W
08800-08FFF	AFEU	FIFO	R/W
0A000	DEU	Mode Register	R/W
0A008	DEU	Key Size Register	R/W
0A010	DEU	Data Size Register	R/W
0A018	DEU	Reset Control Register	R/W
0A028	DEU	Status Register	R
0A030	DEU	Interrupt Status Register	R
0A038	DEU	Interrupt Control Register	R/W
0A050	DEU	EU-Go	W
0A100	DEU	IV Register	R/W
0A400	DEU	Key 1 Register	W
0A408	DEU	Key 2 Register	W
0A410	DEU	Key 3 Register	W
0A800-0AFFF	DEU	FIFO	R/W

Table 3-2. Preliminary System Address Map Showing All Registers (continued)

MPC184 Address (hex) (AD 16::0)	MPC184 Module	Description	Type
0C000	MDEU	Mode Register	R/W
0C008	MDEU	Key Size Register	R/W
0C010	MDEU	Data Size Register	R/W
0C018	MDEU	Reset Control Register	R/W
0C028	MDEU	Status Register	R
0C030	MDEU	Interrupt Status Register	R
0C038	MDEU	Interrupt Control Register	R/W
0C050	MDEU	EU_GO	W
0C100-0C120	MDEU	Context Memory	R/W
0C400-0C47F	MDEU	Key Memory	W
0C800-0CFFF	MDEU	FIFO	W
0E000	RNG	Mode Register	R/W
0E010	RNG	Data Size Register	R/W
0E018	RNG	Reset Control Register	R/W
0E028	RNG	Status Register	R
0E030	RNG	Interrupt Status Register	R
0E038	RNG	Interrupt Control Register	R/W
0E050	RNG	EU_GO	W
0E800-0EFFF	RNG	FIFO	R
10000	PKEU	Mode Register	R/W
10008	PKEU	Key Size Register	R/W
10010	PKEU	Data Size Register	R/W
10018	PKEU	Reset Control Register	R/W
10028	PKEU	Status Register	R
10030	PKEU	Interrupt Status Register	R
10038	PKEU	Interrupt Control Register	R/W
10050	PKEU	EU_GO	W
10200-1023F	PKEU	Parameter Memory A0	R/W
10240-1027F	PKEU	Parameter Memory A1	R/W
10280-102BF	PKEU	Parameter Memory A2	R/W
102C0-102FF	PKEU	Parameter Memory A3	R/W
10300-1033F	PKEU	Parameter Memory B0	R/W
10340-1037F	PKEU	Parameter Memory B1	R/W

Table 3-2. Preliminary System Address Map Showing All Registers (continued)

MPC184 Address (hex) (AD 16::0)	MPC184 Module	Description	Type
10380-103BF	PKEU	Parameter Memory B2	R/W
103C0-103FF	PKEU	Parameter Memory B3	R/W
10400-104FF	PKEU	Parameter Memory E	W
10800-108FF	PKEU	Parameter Memory N	R/W
12000	AESU	Mode Register	R/W
12008	AESU	Key Size Register	R/W
12010	AESU	Data Size Register	R/W
12018	AESU	Reset Control Register	R/W
12028	AESU	Status Register	R
12030	AESU	Interrupt Status Register	R
12038	AESU	Interrupt Control Register	R/W
12050	AESU	End of Message Register	W
12100	AESU	IV Register	R/W
12400-12408	AESU	Key Memory	R/W
12800-12FFF	AESU	FIFO	R/W
18000-19FFF	Memory	General Purpose Memory	R/W

3.2 Base Address Register

This register contains the base address for all MPC184 registers and memory. It is initially set via the BASE[0:4] input pins. The initial setting can be overwritten by writing to this

register.

Figure 3-1. Base Address Register.

	0						31
Field	Reserved						
Reset	0						
R/W	R/W						
Addr	0x 00F00						

	0	4	5		14	15		31
Field	Base Address							
Definition	Base		Program			Fixed		
Reset	Base Pins		0			0		
R/W	R/W							
Addr	0x 00F04							

Table 3-3. Base Address Register Definition

Bits	Name	Reset Value	Description
0:4	Base	Variable	At reset, bits [0:4] are loaded with the value on input pins BASE[0-4]. This value is overwritten upon a write to this register.
5:14	Program	0	At reset, these bits are set to zero. This value is overwritten upon a write to this register.
15:31	Fixed	0	These bits are used within the MPC184 address space. The base address for these bits is always zero. They can not be written.

3.3 Slave Parity Address Register

If enabled by the Parity Enable bit in the Controller's Master Control Register, the MPC184, when acting as a slave, will store the current address on the 8xx bus in the Slave Parity Address Register when it detects a data parity error on a write to any address within the MPC184 memory map. Once an address is stored, it will not be overwritten until the MPC184 is reset. The definition of the fields are as follows.

	0					31
Field	ADDRESS					
Reset	0					
R/W	R					
Addr	0x 00800 (Register 1)					

Figure 3-2. Slave Parity Address Register

Table 3-4., "Slave Parity Address Register" provides the bit definition for this register.



Table 3-4. Slave Parity Address Register

Bits	Name	Reset Value	Description
0:31	Address	0	Slave Parity Address Register: The address on the 8xx bus at the time a data parity error was detected by the MPC184 on a slave write to the MPC184.



Chapter 4

Execution Units

“Execution unit” is the generic term for a functional block that performs the mathematical permutations required by protocols used in cryptographic processing. The EUs are compatible with IPsec, WAP/WTLS, IKE, SSL/TLS and DOCSIS BPI++ processing, and can work together to perform high level cryptographic tasks.

The following Execution Units are used on the MPC184:

- Public Key Execution Unit (PKEU) supporting:
 - RSA and Diffie-Hellman
 - Elliptic curve operations in either F_{2^m} or F_p
- Data Encryption Standard Execution Unit (DEU) supporting:
 - DES
 - 3DES
 - Two key (K1, K2, K1) or Three Key (K1, K2, K3)
 - ECB and CBC modes for both DES and 3DES
- Advanced Encryption Standard Execution Unit (AESU) supporting:
 - 128, 192, or 256 bit keys
 - ECB, CBC, and Counter modes for all key lengths
- ARC Four Execution Unit (AFEU)
 - Implements a stream cipher compatible with the RC-4 algorithm
 - 8- to 128-bit programmable key
- Message Digest Execution Unit (MDEU) supporting:
 - SHA-1, a 160 bit hash function, specified by the ANSI X9.30-2 and FIPS 180-1 standards.
 - The MD5 generates a 128 bit hash, and the algorithm is specified in RFC 1321.
 - SHA-256, a 256-bit hash function that provides 256 bits of security against collision attacks.
 - The MDEU also supports HMAC computations, as specified in RFC 2104.
- Private on-chip random number generator (RNG)

Working together, the EUs can perform high-level cryptographic tasks, such as IPSec encapsulating security protocol (ESP) and digital signature. The remainder of this Chapter provides details about the Execution Units themselves.

NOTE

The execution units used in the MPC184 are identical to those used in previous security processors, and are natively little endian. Register values are shown in a big endian format to assist in debug in an 8xx (big endian) environment. Much of the following detail is required only for debug and operation of the MPC184 in target mode. When operating as an initiator, the device drivers abstract register-level operations, and the crypto-channels and controller operate the execution units.

4.1 Public Key Execution Units (PKEU)

This section contains details about the Public Key Execution Unit (PKEU), including detailed register map, modes of operation, status and control registers, and the parameter RAMs.

4.1.1 PKEU Register Map

The PKEU contains the following registers and parameter memories, which are explained in detail in the following sections.

- PKEU Mode Register
- Key Size Register
- Data Size Register
- Reset Control Register
- Status Register
- Interrupt Status Register
- Interrupt Control Register
- “Go” Register
- Parameter Memory A
- Parameter Memory B
- Parameter Memory E
- Parameter Memory N

4.1.2 PKEU Mode Register

This register specifies the internal PKEU routine to be executed. For the root arithmetic routines, PKEU has the capability to perform arithmetic operations on subsegments of the entire memory. This is particularly useful for operations such as ECDH (elliptic curve Diffie-Hellman) key agreement computation. By using regAsel and regBsel, for example, parameter memory A subsegment 2 can be multiplied into Parameter Memory B subsegment 1. Figure 4-1 and Figure 4-2 detail two definitions.

	0	1	7	8	31
Field	—	MODE			Reserved
Reset	0				0
R/W	R/W				
Addr	PKEU 0x10000				

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	PKEU 0x10004	

Figure 4-1. PKEU Mode Register: Definition 1

	0	1	3	4	7	8	31
Field	—	MODE		RegSEL		Reserved	
Reset	0	0		0		0	
R/W	R/W						
Addr	PKEU 0x10000						

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	PKEU 0x10004	

Figure 4-2. PKEU Mode Register: Definition 2

Table 4-1 lists mode register routine definitions. Parameter memories are referred to for the base address, as show.

Table 4-1. Mode Register Routine Definitions

Routine	Mode [1:3]	Mode [4:5]	Mode [6:7]
Reserved	000	00	00
Clear memory	000	0	01
Modular exponentiation	000	00	10
$R^2 \bmod N$	000	00	11
$R_n R_p \bmod N$	000	01	00
F_p affine point multiplication	000	01	01
F_{2m} affine point multiplication	000	01	10

Table 4-1. Mode Register Routine Definitions (continued)

Routine	Mode [1:3]	Mode [4:5]	Mode [6:7]
F_p projective point multiplication	000	01	11
F_{2m} projective point multiplication	000	10	00
F_p point addition	000	10	01
F_p point doubling	000	10	10
F_{2m} point addition	000	10	11
F_{2m} point doubling	000	11	00
F_{2m} R ² CMD	000	11	01
F_{2m} INV CMD	000	11	10
MOD INV CMD	000	11	11
Modular addition	001	regAsel ¹ 00 = A0 01 = A1 10 = A2 11 = A3	regBsel ¹ 00 = B0 01 = B1 10 = B2 11 = B3
Modular subtraction	010		
Modular multiplication with single reduction	011		
Modular multiplication with double reduction	100		
Polynomial addition	101		
Polynomial multiplication with single reduction	110		
Polynomial multiplication with double reduction	111		

¹ regAsel and regBsel here refer to the specific segment of parameter memory A and B.

4.1.3 PKEU Key Size Register

The Key Size Register reflects the number of significant bytes to be used from PKEU Parameter Memory E in performing modular exponentiation or elliptic curve point multiplication. The minimum value for this register, when performing either modular exponentiation or elliptic curve point multiplication, is 1 byte. (0:15= 0x0100). The maximum legal value is 256 bytes. (0:15= 0x0001). To avoid a key size error, 12:14 must be set to zero, and the value of [0:7, 15] must not be greater than 256.

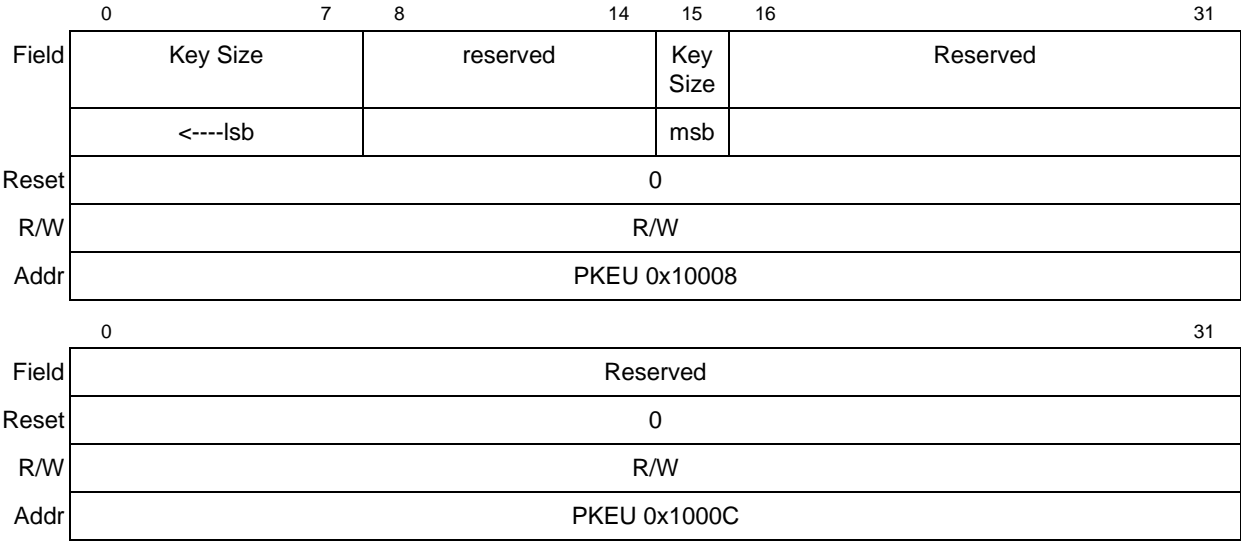


Figure 4-3. PKEU Key Size Register

4.1.4 PKEU Data Size Register

The PKEU Data Size Register specifies, in bits, the size of the significant portion of the modulus or irreducible polynomial. Any value written to this register that is a multiple of 32 bits (i.e. 128 bits, 160 bits,...), will be represented internally as the same value (128 bits, 160 bits,...). Any value written that is not a multiple of 32 bits (i.e. 132bits, 161bits,...), will be represented internally as the next larger 32 bit multiple (160 bits, 196 bits,...). This internal rounding up to the next 32-bit multiple is described for information only. The minimum size valid for all routines to operate properly is 97 bits (internally 128 bits). (0:15= 0x6100) The maximum size to operate properly is 2048 bits (0:15= 0x0008). A

value in bits larger than 2048 will result in a Data Size error.

	0	7	8	11	12	15	16	31	
Field	Data Size			Reserved	Data Size			Reserved	
	<----lsb				msb<-----				
Reset	0								
R/W	R/W								
Addr	PKEU 0x10010								

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	PKEU 0x10014	

Figure 4-4. PKEU Data Size Register

4.1.5 PKEU Reset Control Register

This register, Figure 4-5, contains three reset options specific to the PKEU.

	0	4	5	6	7	8	31
Field	Reserved		RI	MI	SR	Reserved	
Reset	0		0	0	0	0	
R/W	R/W						
Addr	PKEU 0x10018						

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	PKEU 0x1001C	

Figure 4-5. PKEU Reset Control Register

Table 4-2 describes the PKEU Reset Control Register’s signals.

Table 4-2. PKEU Reset Control Register Signals

Bits	Name	Description
0:4	-	Reserved
5	Reset Interrupt	Writing this bit active high causes PKEU interrupts signalling DONE and ERROR to be reset. It further resets the state of the PKEU Interrupt Status Register. 0 Don't reset 1 Reset interrupt logic
6	Module_Init	Module initialization is nearly the same as Software Reset, except that the Interrupt Control register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the PKEU Status Register (Section 4.1.6, "PKEU Status Register"). 0 Don't reset 1 Reset most of PKEU
7	SW_RESET	Software Reset is functionally equivalent to hardware reset (the RESET# pin), but only for the PKEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the PKEU will enter a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the PKEU Status Register will indicate when this initialization routine is complete (Section 4.1.6, "PKEU Status Register"). 0 Don't reset 1 Full PKEU reset
8:31	—	Reserved

4.1.6 PKEU Status Register

This status register contains 5 bits which reflect the state of PKEU internal signals.

The PKEU Status Register is read-only. Writing to this location will result in address error being reflected in the PKEU Interrupt Status Register.

	0	1	2	3	4	5	6	7	8	31
Field	---	Z	Halt	<div></div>	IE	ID	RD	Reserved		
Reset	0	0	0	0	0	0	0	0		
R/W	R									
Addr	PKEU 0x10028									

	0	31
Field	Reserved	
Reset	0	
R/W	R	
Addr	PKEU 0x1002C	

Figure 4-6. PKEU Status Register

Table 4-3 describes the PKEU Status Register's signals.

Table 4-3. PKEU Status Register Signals

Bits	Name	Description
0	----	Reserved
1	Z	Zero: this bit reflects the state of the PKEU Zero Detect bit when last sampled. Only particular instructions within routines cause Zero to be modified, so this bit should be used with great care.
2	Halt	Halt- Indicates that the PKEU has halted due to an error. 0 PKEU not halted 1 PKEU halted Note: Because the error causing the PKEU to stop operating may be masked to the Interrupt Status Register, the Status Register is used to provide a second source of information regarding errors preventing normal operation.
3:4	----	Reserved
5	Interrupt_Error	This status bit reflects the state of the ERROR interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, "Interrupt Status Registers"). 0 PKEU is not signaling error 1 PKEU is signaling error
6	Interrupt_Done	This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, "Interrupt Status Registers"). 0 PKEU is not signaling done 1 PKEU is signaling done
7	Reset_Done	This status bit, when high, indicates that PKEU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel. 0 Reset in progress 1 Reset done
8:31	----	Reserved Note: Some bits in the upper portion of this register are used as state tables for internal PKEU routines. In order to avoid confusion should the user read this register during normal operation, the user is advised that these bits exist, but their specific definition is reserved.

4.1.7 PKEU Interrupt Status Register

The interrupt status register tracks the state of possible errors, if those errors are not masked, via the PKEU interrupt control register. The definition of each bit in the PKEU Interrupt Status Register is shown in Figure 4-7.

	0	1	2	9	10	11	12	13	14	15	16	31
Field	ME	AE	Reserved		Inv	IE	--	CE	KSE	DSE	Reserved	
Reset	0	0	0					0	0	0	0	
R/W	R											
Addr	PKEU 0x10030											

	0	31																														
Field	Reserved																															
Reset	0X0000_0000																															
R/W	R																															
Addr	PKEU 0x10034																															

Figure 4-7. PKEU Interrupt Status Register

Table 4-4 describes PKEU Interrupt Status Register signals.

Table 4-4. PKEU Interrupt Status Register Signals

Bits	Name	Description
0	Mode Error	An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
1	Address Error	Illegal read or write address was detected within the PKEU address space. 0 no error detected 1 address error
2–9	—	Reserved
10	Inversion Error	Indicates that the inversion routine has a zero operand. 0 no inversion error detected 1 inversion error detected
11	Internal Error	An internal processing error was detected while the PKEU was operating. 0 No error detected 1 Internal error Note: This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Control Register or by resetting the PKEU.
12	—	Reserved
13	Context Error	A PKEU Key register, the key size register, the data size register, or mode register was modified while the PKEU was operating. 0 No error detected 1 Context error
14	Key Size Error	Value outside the bounds of 1 - 256 bytes was written to the PKEU key size register 0 no error detected 1 key size error detected

Table 4-4. PKEU Interrupt Status Register Signals (continued)

Bits	Name	Description
15	Data Size Error	Value outside the bounds 97- 2048 bits was written to the PKEU data size register 0 no error detected 1 data size error detected
16-31	—	Reserved

4.1.8 PKEU Interrupt Control Register

The PKEU Interrupt Control Register controls the result of detected errors. For a given error (as defined in Section 4.1.7, “PKEU Interrupt Status Register”), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the PKEU Interrupt Status Register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0	1	2	9	10	11	12	13	14	15	16	31
Field	ME	AE	Reserved	Inv	IE	--	CE	KSE	DSE	Reserved		
Reset	0	0	0				0	0	0	0		
R/W	R/W											
Addr	PKEU 0x10038											
	0											31
Field	Reserved											
Reset	0X0000_0000											
R/W	R/W											
Addr	PKEU 0x1003C											

Figure 4-8. PKEU Interrupt Control Register

Table 4-5 describes PKEU Interrupt Control Register signals.

Table 4-5. PKEU Interrupt Control Register Signals

Bits	Name	Description
0	Mode Error	Mode Error 0 Mode Error enabled 1 Mode Error disabled
1	Address Error	Address Error 0 address error enabled 1 address error disabled
2–9	—	Reserved

Table 4-5. PKEU Interrupt Control Register Signals (continued)

Bits	Name	Description
10	Inversion Error	Inversion Error. 0 Inversion error enabled 1 Inversion error disabled
11	Internal Error	Internal Error 0 Internal Error enabled 1 Internal Error disabled
12	—	Reserved
13	Context Error	Context Error 0 Context Error enabled 1 Context Error disabled
14	Key Size Error	Key Size Error 0 Key Size Error enabled 1 Key Size Error disabled
15	Data Size Error	Data Size Error 0 Data Size Error enabled 1 Data Size Error disabled
16-31	—	Reserved

4.1.9 PKEU EU_GO Register

The EU_GO Register in the PKEU is used to indicate the start of a new computation. Writing to this register causes the PKEU to execute the function requested by the mode register, per the contents of the parameter memories listed below. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. The PKEU EU_GO Register is only used when the MPC184 is operated as a target. The descriptors and crypto-channel activate the PKEU (via an internally generated write to the EU_GO Register) when the MPC184 acts as an initiator.

	0	31
Field	PKEU EU_GO	
Reset	0	
R/W	W	
Addr	PKEU 0x10050	

Figure 4-9. PKEU EU_GO Register

4.1.10 PKEU Parameter Memories

The PKEU uses four 2048-bit memories to receive and store operands for the arithmetic operations the PKEU will be asked to perform. In addition, results are stored in one particular parameter memory.

All these memories store data in the same format: least significant data byte in the least significantly addressed byte, both data significance and addressing significance increasing identically and simultaneously.

4.1.10.1 PKEU Parameter Memory A

This 2048 bit memory is used typically as an input parameter memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function. For elliptic curve routines, this memory is segmented into four 512 bit memories, and is used to specify particular curve parameters and input values.

4.1.10.2 PKEU Parameter Memory B

This 2048 bit memory is used typically as an input parameter memory space, as well as the result memory space. For modular arithmetic routines, this memory operates as one of the operands of the desired function, as well as the result memory space. For elliptic curve routines, this memory is segmented into four 512 bit memories, and is used to specify particular curve parameters and input values, as well as to store result values.

4.1.10.3 PKEU Parameter Memory E

This 2048 bit memory is non-segmentable, and stores the exponent for modular exponentiation, or the multiplier k for elliptic curve point multiplication. This memory space is write only; a read of this memory space will cause address error to be reflected in the PKEU Interrupt Status Register.

4.1.10.4 PKEU Parameter Memory N

This 2048 bit memory is non-segmentable, and stores the modulus for modular arithmetic and F_p elliptic curve routines. For F_{2^m} elliptic curve routines, this memory stores the irreducible polynomial.

4.2 Data Encryption Standard Execution Units (DEU)

This section contains details about the Data Encryption Standard Execution Units (DEU), including detailed register map, modes of operation, status and control registers, and FIFOs.

4.2.1 DEU Register Map

The registers used in the DEU are documented primarily for debug and target mode operations. If the MPC185 requires the use of the DEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user. Each of the two instances of DEU contains the following registers:

- DEU Mode Register
- Key Size Register
- Data Size Register
- Reset Control Register
- Status Register
- Interrupt Status Register
- Interrupt Control Register
- “Go” Register
- IV Register
- Key Registers
- FIFO

4.2.2 DEU Mode Register

The DEU Mode Register contains 3 bits which are used to program the DEU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the MPC184 as an initiator. Burst size is not relevant to target mode operations, where an external host pushes and pulls data from the execution units.

The mode register is cleared when the DEU is reset or re-initialized. Setting a reserved mode bit will generate a data error. If the mode register is modified during processing, a context error will be generated.

	0	4	5	6	7	8	12	13	15	16	31
Field	Reserved		CE	TS	ED	Reserved		Burst Size		Reserved	
Reset	0		0	0	0	0		0		0	
R/W	R/W										
Addr	DEU 0x0A000										

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	DEU 0x0A004	

Figure 4-10. DEU Mode Register

Table 4-6 describes DEU Mode Register signals.

Table 4-6. DEU Mode Register Signals

Bits	Signal	Description
0-4	—	Reserved
5	CBC/ECB	If set, DEU operates in cipher-block-chaining mode. If not set, DEU operates in electronic code book mode. 0 ECB mode 1 CBC mode
6	Triple/Single DES	If set, DEU operates the Triple DES algorithm; if not set, DEU operates the single DES algorithm. 0 single DES 1 triple DES
7	encrypt/decrypt	If set, DEU operates the encryption algorithm; if not set, DEU operates the decryption algorithm. 0 Perform decryption 1 Perform encryption
8-12	—	Reserved
13-15	Burst Size	The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The DEU signals to the crypto-channel that a "Burst Size" amount of data is available to be pushed to or pulled from the FIFO. Note: The inclusion of this field in the DEU Mode Register is to avoid confusing a user who may read this register in debug mode. Burst Size should not be written directly to the DEU.
16-31	—	Reserved

4.2.3 DEU Key Size Register

This value indicates the number of bytes of key memory that should be used in encrypting or decrypting. If the DEU mode register is set for single DES, any value other than 8 bytes

will automatically generate a key size error in the DEU interrupt status register. If the mode bit is set for triple DES, any value other than 16 bytes (112 bits for 2-key triple DES (K1=K3) or 24 bytes (168 bits for 3-key triple DES) will generate an error. Triple DES always uses K1 to encrypt, Key2 to decrypt, K3 to encrypt.

NOTE

Reserved fields must be set to zero to ensure proper operation.

	0	1	2	7	8	31
Field	Reserved	Key Size			Reserved	
		msb<----lsb				
Reset	0	0			0	
R/W	R/W					
Addr	DEU 0x0A008					

	0	31
Field	Reserved	
Reset	0x0000_0000	
R/W	R/W	
Addr	DEU 0x0A00C	

Figure 4-11. DEU Key Size Register

Table 4-7 shows the legal values for DEU key size.

Table 4-7. DEU Key Size Register

Bits	Signal	Description
0-7	Key Size	8 bytes = 0x08 (only legal value if mode is single DES.) 16 bytes= 0x10 (for 2 key 3DES, K1 = K3) 24 bytes= 0x18 (for 3 key 3DES)
7-31	----	Reserved

4.2.4 DEU Data Size Register

This register is used to verify that the data to be processed by the DEU is divisible by the DES algorithm block size of 64-bits. The DEU does not automatically pad messages out to 64-bit blocks, therefore any message processed by the DEU must be divisible by 64-bits or a Data Size Error will occur.

In normal operation, the full message length (data size) to be encrypted or decrypted by the DEU is copied from the descriptor to the DEU Data Size register, however only bits 2:7 are checked to determine if there is a Data Size Error. If 2:7 are all zeroes, the message is evenly divisible into 64-bit blocks. In target mode, the user must write the data size to the data size

register. If the data size written is not divisible by 64-bits (2:7 non-zero), a data size error will occur.

	0	1	2	7	8	31									
Field	Reserved		Data Size			Reserved									
			msb<----lsb												
Reset	0														
R/W	R/W														
Addr	DEU 0x0A010														

	0	31																													
Field	Reserved																														
Reset	0x0000_0000																														
R/W	R/W																														
Addr	DEU 0x0A014																														

Figure 4-12. DEU Data Size Register

4.2.5 DEU Reset Control Register

This register, shown in Figure 4-13, allows 3 levels reset of just DEU, as defined by the 3 self-clearing bits:

	0	4	5	6	7	8	31
Field	Reserved		RI	MI	SR	Reserved	
Reset	0		0	0	0	0	
R/W	R/W						
Addr	DEU 0x0A018						

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	DEU 0x0A01C	

Figure 4-13. DEU Reset Control Register

Table 4-8 describes DEU Reset Control Register signals.

Table 4-8. DEU Reset Control Register Signals

Bits	Signals	Description
0:4	—	Reserved
5	Reset Interrupt	Writing this bit active high causes DEU interrupts signalling DONE and ERROR to be reset. It further resets the state of the DEU Interrupt Status Register. 0 Don't reset 1 Reset interrupt logic
6	Module_Init	Module initialization is nearly the same as Software Reset, except that the Interrupt Control register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the DEU Status Register 0 Don't reset 1 Reset most of DEU
7	SW_RESET	Software Reset is functionally equivalent to hardware reset (the RESET# pin), but only for DEU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the DEU will enter a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the DEU Status Register will indicate when this initialization routine is complete 0 Don't reset 1 Full DEU reset
8:31	—	Reserved

4.2.6 DEU Status Register

This status register, displayed in Figure 4-14, contains 6 bits which reflect the state of DEU internal signals.

The DEU Status Register is read-only. Writing to this location will result in address error being reflected in the DEU interrupt status register.

	0	1	2	3	4	5	6	7	8	31
Field	---	Halt	IFW	OFR	IE	ID	RD	Reserved		
Reset	0	0	0	0	0	0	0	0		
R/W	R									
Addr	DEU 0x0A028									

	0	31
Field	Reserved	
Reset	0	
R/W	R	
Addr	DEU 0x0A02C	

Figure 4-14. DEU Status Register

Table 4-9 describes the DEU Status Register's signals.

Table 4-9. DEU Status Register Signals

Bits	Name	Description
0-1	----	Reserved
2	Halt	<p>Halt- Indicates that the DEU has halted due to an error.</p> <p>0 DEU not halted</p> <p>1 DEU halted</p> <p>Note: Because the error causing the DEU to stop operating may be masked to the Interrupt Status Register, the Status Register is used to provide a second source of information regarding errors preventing normal operation.</p>
3	IFW	<p>Input FIFO Writable- The Controller uses this signal to determine if the DEU can accept the next BURST SIZE block of data.</p> <p>0 DEU Input FIFO not ready</p> <p>1 DEU Input FIFO ready</p> <p>Note: The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The DEU signals to the crypto-channel that a “Burst Size” amount of space is available in the FIFO. The documentation of this bit in the DEU Status Register is to avoid confusing a user who may read this register in debug mode.</p>
4	OFR	<p>Output FIFO Readable- The Controller uses this signal to determine if the DEU can source the next BURST SIZE block of data.</p> <p>0 DEU Output FIFO not ready</p> <p>1 DEU Output FIFO ready</p> <p>Note: The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The DEU signals to the crypto-channel that a “Burst Size” amount of data is available in the FIFO. The documentation of this bit in the DEU Status Register is to avoid confusing a user who may read this register in debug mode.</p>
5	Interrupt_Error	<p>This status bit reflects the state of the ERROR interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, “Interrupt Status Registers”).</p> <p>0 DEU is not signaling error</p> <p>1 DEU is signaling error</p>
6	Interrupt_Done	<p>This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, “Interrupt Status Registers”).</p> <p>0 DEU is not signaling done</p> <p>1 DEU is signaling done</p>
7	Reset_Done	<p>This status bit, when high, indicates that DEU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel.</p> <p>0 Reset in progress</p> <p>1 Reset done</p>
8:31	----	Reserved

4.2.7 DEU Interrupt Status Register

The DEU Interrupt Status Register, shown in Figure 4-15, tracks the state of possible errors, if those errors are not masked, via the DEU interrupt control register. The definition of each bit in the interrupt status register is:

	0	1	2	3	4	5	6	7	9	10	11	12	13	14	15	16	31	
Field	ME	AE	OFE	IFE	--	IFO	OFU	---		KPE	IE	ERE	CE	KSE	DSE	RESERVED		
Reset	0																	
R/W	R																	
Addr	DEU 0x0A030																	
	0																	31
Field	Reserved																	
Reset	0																	
R/W	R																	
Addr	DEU 0x0A034																	

Figure 4-15. DEU Interrupt Status Register

Table 4-10 describes DEU Interrupt Register signals.

Table 4-10. DEU Interrupt Status Register Signals

Bits	Signal	Description
0	Mode Error	An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
1	Address Error	An illegal read or write address was detected within the DEU address space. 0 No error detected 1 Address error
2	Output FIFO Error	The DEU output FIFO was detected non-empty upon write of DEU data size register. 0 No error detected 1 Output FIFO non-empty error
3	Input FIFO Error	The DEU input FIFO was detected non-empty upon generation of DONE interrupt. 0 No error detected 1 Input FIFO non-empty error
4	—	Reserved
5	Input FIFO Overflow	The DEU input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed Note: When operating as a master, the MPC184 implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the MPC184 cannot accept FIFO inputs larger than 512B without overflowing.
6	Output FIFO Underflow	The DEU output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
7-9	—	Reserved

Table 4-10. DEU Interrupt Status Register Signals (continued)

Bits	Signal	Description
10	Key Parity Error	Defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are checked for parity only if the appropriate DEU mode register bit indicates triple DES. Also, key register 3 is checked only if key size reg = 24. Key register 2 is checked only if key size reg = 16 or 24.) 0 No error detected 1 Key parity error
11	Internal Error	An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error Note: This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Control Register or by resetting the DEU.
12	Early Read Error	The DEU IV register was read while the DEU was performing encryption. 0 No error detected 1 Early read error
13	Context Error	A DEU Key register, the key size register, the data size register, the mode register, or IV register was modified while DEU was performing encryption. 0 No error detected 1 Context error
14	Key Size Error	An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for triple DES) was written to the DEU key size register 0 No error detected 1 Key size error
15	Data Size Error	Data Size Error (DSE): A value was written to the DEU Data Size Register that is not a multiple of 64 bits. 0 No error detected 1 Data size error
16-31	—	Reserved

4.2.8 DEU Interrupt Control Register

The interrupt control register controls the result of detected errors. For a given error (as defined in Section 4.2.7, “DEU Interrupt Status Register”), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0	1	2	3	4	5	6	7	9	10	11	12	13	14	15	16	31
Field	ME	AE	OFE	IFE	--	IFO	OFU	---		KPE	IE	ERE	CE	KSE	DSE	RESERVED	
Reset	0																
R/W	R/W																
Addr	DEU 0x0A038																

	0																31
Field	RESERVED																
Reset	0																
R/W	R/W																
Addr	DEU 0x0A03C																

Figure 4-16. DEU Interrupt Control Register
Table 4-11. DEU Interrupt Control Register Signals

Bits	Signal	Description
0	Mode Error	An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
1	Address Error	An illegal read or write address was detected within the DEU address space. 0 Address error enabled 1 Address error disabled
2	Output FIFO Error	The DEU Output FIFO was detected non-empty upon write of DEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
3	Input FIFO Error	The DEU Input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
4	—	Reserved
5	Input FIFO Overflow	The DEU Input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled Note: When operating as a master, the MPC184 implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the MPC184 cannot accept FIFO inputs larger than 512B without overflowing.
6	Output FIFO Underflow	The DEU Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
7-9	—	Reserved

Table 4-11. DEU Interrupt Control Register Signals (continued)

Bits	Signal	Description
10	Key Parity Error	The defined parity bits in the keys written to the key registers did not reflect odd parity correctly. (Note that key register 2 and key register 3 are only checked for parity if the appropriate DEU mode register bit indicates triple DES. 0 Key parity enabled 1 Key parity error disabled
11	Internal Error	An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
12	Early Read Error	The DEU IV Register was read while the DEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
13	Context Error	A DEU key register, the key size register, the data size register, the mode register, or IV register was modified while DEU was performing encryption. 0 Context error enabled 1 Context error disabled
14	Key Size Error	An inappropriate value (8 being appropriate for single DES, and 16 and 24 being appropriate for Triple DES) was written to the DEU key size register 0 Key size error enabled 1 Key size error disabled
15	Data Size Error	Data Size Error (DSE): A value was written to the DEU Data Size Register that is not a multiple of 8 bytes. 0 Data Size error enabled 1 Data size error disabled
16-31	—	Reserved

4.2.9 DEU EU_GO Register

The EU_GO register in the DEU is used to indicate a DES operation may be completed. After the final message block is written to the input FIFO, the EU-GO register must be written. The value in the data size register will be used to determine how many bits of the final message block (always 64) will be processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. Writing to this register is merely a trigger causing the DEU to process the final block of a message, allowing it to signal DONE.

The DEU EU_GO Register is only used when the MPC184 is operated as a target. The descriptors and crypto-channel activate the DEU (via an internally generated write to the EU_Go register) when the MPC184 acts as an initiator.

	0	31
Field	DEU EU_GO	
Reset	0	
R/W	W	
Addr	DEU 0x0A050	

Figure 4-17. DEU EU_GO Register

4.2.10 DEU IV Register

For CBC mode, the initialization vector is written to and read from the DEU IV Register. The value of this register changes as a result of the encryption process and reflects the context of DEU. Reading this memory location while the module is processing data generates an error interrupt.

4.2.11 DEU Key Registers

The DEU uses three write-only key registers to perform encryption and decryption. In Single DES mode, only key register 1 may be written. The value written to key register 1 is simultaneously written to key register 3, auto-enabling the DEU for 112-bit Triple DES if the key size register indicates 2 key 3DES is to be performed (key size = 16 bytes). To operate in 168-bit Triple DES, key register 1 must be written first, followed by the write of key register 2, the key register 3.

Reading any of these memory locations will generate an address error interrupt.

4.2.12 DEU FIFOs

DEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. These FIFOs are multiply addressable, but those multiple addresses point only to the appropriate end of the appropriate FIFO. A write to anywhere in the DEU FIFO address space causes the 32-bit-word to be pushed onto the DEU input FIFO, and a read from anywhere in the DEU FIFO Address space causes a 32-bit-word to be popped off of the DEU output FIFO. Overflows and under flows caused by reading or writing the DEU FIFOs are reflected in the DEU interrupt status register.

4.3 ARC Four Execution Unit (AFEU)

This section contains details about the ARC Four Execution Unit (AFEU), including detailed register map, modes of operation, status and control registers, S-box memory, and FIFOs.

4.3.1 AFEU Register Map

The registers used in the AFEU are documented primarily for debug and target mode operations. If the MPC184 requires the use of the AFEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user. The AFEU contains the following registers:

- AFEU Mode Register
- Key Size Register
- Data Size Register
- Reset Control Register
- Status Register
- Interrupt Status Register
- Interrupt Control Register
- End Of Message Register
- Context Memory
- Context Pointer Register
- Key Registers
- FIFO

4.3.2 AFEU Mode Register

Shown in Figure 4-18, the AFEU Mode Register contains three bits which are used to program the AFEU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the MPC184 as an initiator. Burst size is not relevant to target mode operations, where an external host pushes and pulls data from the execution units.

The mode register is cleared when the AFEU is reset or re-initialized. Setting a reserved mode bit will generate a data error. If the mode register is modified during processing, a context error will be generated.

4.3.2.1 Host-provided Context via Prevent Permute

In the default mode of operation, the host provides the key and key size to the AFEU. The initial memory values in the S-Box are permuted with the key to create new S-Box values, which are used to encrypt the plaintext.

If the 'Prevent Permute' mode bit is set, the AFEU will not require a key. Rather, the host will write the context to the AFEU and message processing will occur using the provided context. This mode is used to resume processing of a message using the already permuted S-Box. The context may be written through the FIFO if the 'context source' mode bit is set.

4.3.2.2 Dump Context

This mode may be independently specified in addition to host-provided context mode. In this mode, once message processing is complete and the output data is read, the AFEU will make the current context data available for reads via the output FIFO.

NOTE

After the initial key permute to generate a context for an AFEU encrypted session, all subsequent messages will re-use that context, such that it is loaded, modified during the encryption, and unloaded, similar to the use of a CBC initialization vector in DES operations. A new context is generated (via key permute) according to a re keying interval specified by the security protocol. Context should never be loaded to encrypt a message if a key is loaded and permuted at the same time.

	0	4	5	6	7	8	12	13	15	16	31	
Field	Reserved		CS	DC	PP	Reserved		Burst Size		Reserved		
Reset	0		0	0	0	0		0		0		
R/W	R/W											
Addr	AFEU 0x08000											
	0											31
Field	Reserved											
Reset	0											
R/W	R/W											
Addr	AFEU 0x08004											

Figure 4-18. AFEU Mode Register

Table 4-12 describes AFEU Mode Register signals.

Table 4-12. AFEU Mode Register Signals

Bits	Signal	Description
0-4	—	Reserved
5	Context Source	If Set, this causes the context to be moved from the input FIFO into the S-box prior to starting encryption/decryption. Otherwise, context should be directly written to the context registers. Context Source is only checked if the Prevent Permute bit is set. 0 Context not from FIFO 1 Context from input FIFO
6	Dump Context	If Set, this causes the context to be moved from the S-box to the output FIFO following assertion AFEU's done interrupt. 0 Do not dump context 1 After cipher, dump context

Table 4-12. AFEU Mode Register Signals

Bits	Signal	Description
7	Prevent Permute	Normally, AFEU receives a key and uses that information to randomize the S-box. If reusing a context from a previous descriptor or if in static assignment mode, this bit should be set to prevent AFEU from re performing this permutation step. 0 Perform S-Box permutation 1 Do not permute
8-12	—	Reserved
13-15	Burst Size	The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/context. The AFEU signals to the crypto-channel that a “Burst Size” amount of data is available to be pushed to or pulled from the FIFO. Note: The inclusion of this field in the AFEU Mode Register is to avoid confusing a user who may read this register in debug mode. Burst Size should not be written directly to the AFEU.
16-31	—	Reserved

4.3.3 AFEU Key Size Register

As displayed in Figure 4-19, this value (1-16) indicates the number of bytes of key memory that should be used in performing S-box permutation. Any key data beyond the number of bytes in the key size register will be ignored. This register is cleared when the AFEU is reset or re-initialized. If the key size is <1 or > 16 is specified, an key size error will be generated. If the Key Size Register is modified during processing, a context error will be generated.

	0	1	2	7	8	31
Field	Reserved	Key Size				Reserved
		msb<-----lsb				
Reset	0					
R/W	R/W					
Addr	AFEU 0x08008					

	0					31
Field	Reserved					
Reset	0x0000_0000					
R/W	R/W					
Addr	AFEU 0x0800C					

Figure 4-19. AFEU Key Size Register

NOTE

The device driver will create properly formatted descriptors for situations requiring an key permute prior to ciphering. When operating the MPC184 as a target (typically debug mode), the user must set the AFEU Mode Register to perform ‘permute with key’, then write the key data to AFEU Key Registers, then write the key size to the key size register. The AFEU will start permuting the memory with the contents of the key registers immediately after the key size is written.

4.3.4 AFEU Context/Data Size Register

The AFEU Context/Data Size Register, shown in Figure 4-20, stores the number of bits in the final message block. This register is cleared when the AFEU is reset or re-initialized. The last message block can be between 8 to 64 bits. If a data size that is not a multiple of 8 bits is written, a data size error will be generated.

The context/data size register is also used to specify the context size. The context size is fixed at 2072 bits (259 bytes). When loading context through the FIFO, all context data must be written prior to writing the context data size. The message data size must be written separately.

NOTE: In target mode, when reloading an existing context, the user must write the context to the Input FIFO, then write the context size (always 2072 bits, 0:15 = 0x1808). The write of the context size indicates to the MPC184 that all context has been loaded. The user then writes the message data size to the Context/Data Size Register. After this write, the user may begin writing message data to the FIFO.

Writing to this register signals the AFEU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error will be generated.

	0	7	8	11	12	15	16	31	
Field	Data Size			Reserved	Data Size		Reserved		
	<----lsb				msb<-----				
Reset	0								
R/W	R/W								
Addr	AFEU 0x08010								

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	AFEU 0x08014	

Figure 4-20. AFEU Data Size Register

4.3.5 AFEU Reset Control Register

This register, as shown in Figure 4-21, allows 3 levels reset that effect the AFEU only, as defined by 3 self-clearing bits. It should be noted that the AFEU executes an internal reset sequence for hardware reset, SW_RESET, or Module Init, which performs proper initialization of the S-Box. To determine when this is complete, observe the RESET_DONE bit in the AFEU Status Register.

	0	4	5	6	7	8	31
Field	Reserved		RI	MI	SR	Reserved	
Reset	0		0	0	0	0	
R/W	R/W						
Addr	AFEU 0x08018						

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	AFEU 0x0801C	

Figure 4-21. AFEU Reset Control Register

Table 4-13 describes AFEU Reset Control Register signals.

Table 4-13. AFEU Reset Control Register Signals

Bits	Signal	Description
0-4	—	Reserved
5	Reset Interrupt	Writing this bit active high causes AFEU interrupts signalling DONE and ERROR to be reset. It further resets the state of the AFEU interrupt status register. 0 Do not reset 1 Reset interrupt logic
6	Module Init	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. 0 Do not reset 1 Reset most of AFEU
7	SW_Reset	Software Reset is functionally equivalent to hardware reset (the RESET# pin), but only for AFEU. All registers and internal state are returned to their defined reset state. On negation of SW_RESET, the AFEU will enter a routine to perform proper initialization of the S-Box. 0 Do not reset 1 Full AFEU reset
8-31	—	Reserved

4.3.6 AFEU Status Register

This status register, shown in Figure 4-22, contains 6 bits which reflect the state of the AFEU internal signals.

The AFEU Status Register is read-only. Writing to this location will result in address error being reflected in the AFEU interrupt status register.

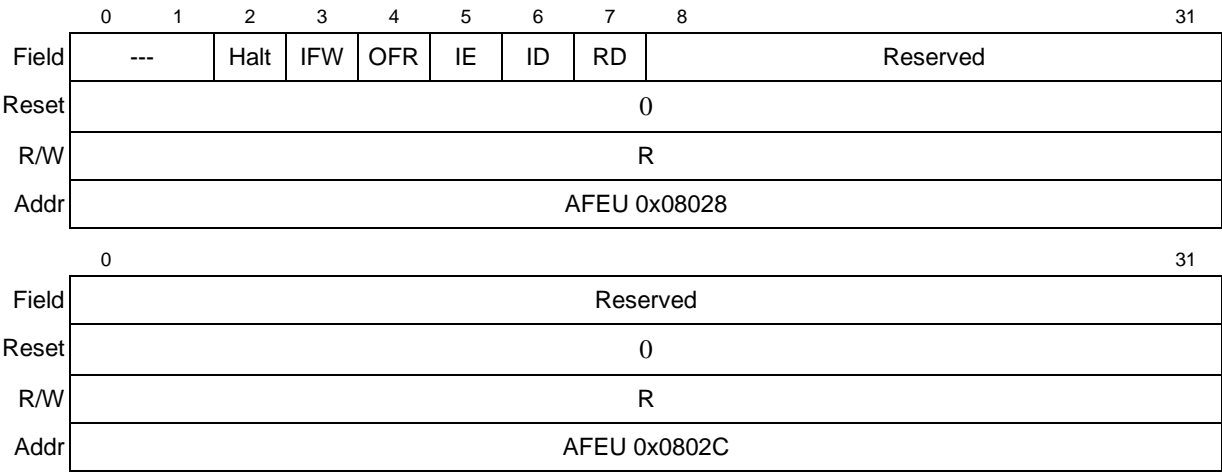


Figure 4-22. AFEU Status Register

Table 4-14 describes AFEU Status Register signals.

Table 4-14. AFEU Status Register Signals

Bits	Signal	Description
0-1	—	Reserved
2	Halt	<p>Halt- Indicates that the AFEU has halted due to an error.</p> <p>0 AFEU not halted 1 AFEU halted</p> <p>Note: Because the error causing the AFEU to stop operating may be masked to the Interrupt Status Register, the Status Register is used to provide a second source of information regarding errors preventing normal operation.</p>
3	IFW	<p>Input FIFO Writable- The Controller uses this signal to determine if the AFEU can accept the next BURST SIZE block of data.</p> <p>0 AFEU Input FIFO not ready 1 AFEU Input FIFO ready</p> <p>Note: The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The AFEU signals to the crypto-channel that a “Burst Size” amount of space is available in the FIFO. The documentation of this bit in the AFEU Status Register is to avoid confusing a user who may read this register in debug mode.</p>
4	OFR	<p>Output FIFO Readable- The Controller uses this signal to determine if the AFEU can source the next BURST SIZE block of data.</p> <p>0 AFEU Output FIFO not ready 1 AFEU Output FIFO ready</p> <p>Note: The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The AFEU signals to the crypto-channel that a “Burst Size” amount of data is available in the FIFO. The documentation of this bit in the AFEU Status Register is to avoid confusing a user who may read this register in debug mode.</p>
5	Interrupt_Error	<p>This status bit reflects the state of the ERROR interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, “Interrupt Status Registers”).</p> <p>0 AFEU is not signaling error 1 AFEU is signaling error</p>
6	Interrupt_Done	<p>This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, “Interrupt Status Registers”).</p> <p>0 AFEU is not signaling done 1 AFEU is signaling done</p>
7	Reset_Done	<p>This status bit, when high, indicates that AFEU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel.</p> <p>0 Reset in progress 1 Reset done</p>
8-31	—	Reserved

4.3.7 AFEU Interrupt Status Register

The interrupt status register, seen in Figure 4-23, tracks the state of possible errors, if those errors are not masked, via the AFEU Interrupt Control Register. The definition of each bit in the interrupt status register is:

	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	31	
Field	ME	AE	OFE	IFE	--	IFO	OFU	----		IE	ERE	CE	KSE	DSE	Reserved		
Reset	0																
R/W	R																
Addr	AFEU 0x08030																
	0																31
Field	Reserved																
Reset	0																
R/W	R																
Addr	AFEU 0x08034																

Figure 4-23. AFEU Interrupt Status Register

Table 4-15 describes AFEU Interrupt Status Register signals.

Table 4-15. AFEU Interrupt Status Register

Bits	Signals	Description
0	Mode Error	An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
1	Address Error	An illegal read or write address was detected within the AFEU address space. 0 No error detected 1 Address error
2	Output FIFO Error	The AFEU output FIFO was detected non-empty upon write of AFEU data size register. 0 No error detected 1 Output FIFO non-empty error
3	Input FIFO Error	The AFEU Input FIFO was detected non-empty upon generation of done interrupt 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
4	—	Reserved
5	Input FIFO Overflow	The AFEU input FIFO has been pushed while full. 1 Input FIFO has overflowed 0 No error detected Note: When operating as a master, the MPC184 implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the MPC184 cannot accept FIFO inputs larger than 512B without overflowing.
6	Output FIFO Underflow	The AFEU output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
7-10	—	Reserved
11	Internal Error	An internal processing error was detected while performing encryption. 0 No error detected 1 Internal error

Table 4-15. AFEU Interrupt Status Register (continued)

Bits	Signals	Description
12	Early Read Error	Early Read Error- the AFEU Context Memory or Control was read while the AFEU was performing encryption. 0 No error detected 1 Early read error
13	Context Error	The AFEU mode register, key register, key size register, data size register, or context memory is modified while AFEU processes data. 0 No error detected 1 Context error
14	Key Size Error	A value outside the bounds 1 - 16 bytes was written to the AFEU key size register 0 No error detected 1 Key size error
15	Data Size Error	An inconsistent value (not a multiple of 8 bits, or larger than 64 bits) was written to the AFEU Data Size Register: 0 No error detected 1 Data size error
16-31	—	Reserved

4.3.8 AFEU Interrupt Control Register

The interrupt control register, shown in Figure 4-24, controls the result of detected errors. For a given error (as defined in Section 4.3.7, “AFEU Interrupt Status Register”), if the corresponding bit in this register is set, the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	31
Field	ME	AE	OFE	IFE	---	IFO	OFU	---		IE	ERE	CE	KSE	DSE	Reserved	
Reset	0															
R/W	R/W															
Addr	AFEU 0x08038															
	0															31
Field	Reserved															
Reset	0															
R/W	R/W															
Addr	AFEU 0x0803C															

Figure 4-24. AFEU Interrupt Control Register

Table 4-16 describes AFEU Interrupt Control Register signals.

Table 4-16. AFEU Interrupt Control Register

Bits	Signals	Description
0	Mode Error	An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
1	Address Error	An illegal read or write address was detected within the AFEU address space. 0 Address error enabled 1 Address error disabled
2	Output FIFO Error	The AFEU Output FIFO was detected non-empty upon write of AFEU data size register 0 Output FIFO non-empty error enabled 1 Output FIFO non-empty error disabled
3	Input FIFO Error	The AFEU input FIFO was detected non-empty upon generation of done interrupt. 0 Input FIFO non-empty error enabled 1 Input FIFO non-empty error disabled
4	—	Reserved
5	Input FIFO Overflow	The AFEU Input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
6	Output FIFO Underflow	The AFEU Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
7-10	—	Reserved
11	Internal Error	An internal processing error was detected while performing encryption. 0 Internal error enabled 1 Internal error disabled
12	Early Read Error	The AFEU Register was read while the AFEU was performing encryption. 0 Early read error enabled 1 Early read error disabled
13	Context Error	An AFEU key register, the key size register, the data size register, the mode register, or context memory was modified while AFEU was performing encryption. 0 Context error enabled 1 Context error disabled
14	Key Size Error	A value outside the bounds 1 - 16 bytes was written to the AFEU key size register 0 Key size error enabled 1 Key size error disabled
15	Data Size Error	An inconsistent value was written to the AFEU Data Size Register: 0 Data Size error enabled 1 Data size error disabled
16-31	—	Reserved

4.3.9 AFEU End of Message Register

The end of message register in the AFEU, displayed in Figure 4-25, is used to indicate an ARC-4 operation may be completed. After the final message block is written to the input FIFO, the end of message register must be written. The value in the data size register will be used to determine how many bits of the final message block (8-64, in multiples of 8) will

be processed. Writing to this register causes the AFEU to process the final block of a message, allowing it to signal DONE. If the ‘dump context’ bit in the AFEU Mode Register is set, the context will be written to the output FIFO following the last message word. A read of this register will always return a zero value.

The AFEU End Of Message Register is only used when the MPC184 is operated as a target. The descriptors and crypto-channel activate the AFEU (via an internally generated write to the end of message register) when the MPC184 acts as an initiator.

	0	31
Field	AFEU End of Message	
Reset	0	
R/W	W	
Addr	AFEU 0x08050	

Figure 4-25. AFEU End of Message Register

4.3.10 AFEU Context

This section provides additional information about the AFEU context memory and its related pointer register.

4.3.10.1 AFEU Context Memory

The S-Box memory consists of 64 32-bit words, each readable and writable. The S-Box contents should not be written with data unless it was previously read from the S-Box. Context data may only be written if the ‘prevent permutation’ mode bit is set (see Figure 4-18) and the context data must be written prior to the message data. If the context registers are written during message processing or the ‘prevent permutation’ bit is not set, a context error will be generated. Reading this memory while the module is not done will generate an error interrupt.

4.3.10.2 AFEU Context Memory Pointer Register

The context memory pointer register holds the internal context pointers that are updated with each byte of message processed. These pointers correspond to the values of I, J, and Sbox[I+1] in the ARC-4 algorithm. If this register is written during message processing, a context error will be generated.

When performing ARC-4 operations, the user has the option of performing a new S-Box permutation per packet, or unloading the contents of the S-box (context) and reloading this context prior to processing of the next packet. The S-Box contents (256bytes) plus the 3 bytes of the context memory pointers are unloaded and reloaded via the AFEU FIFOs.

AFEU Context consists of the contents of the S-Box, as well as three counter values, which indicate the next values to be used from the S-Box. Context must be loaded in the same order in which it was unloaded.

4.3.11 AFEU Key Registers

AFEU uses two write-only key registers to guide initial permutation of the AFEU S-Box, in conjunction with the AFEU key size register. AFEU performs permutation starting with the first byte of key register 0, and uses as many bytes from the two key registers as necessary to complete the permutation. Reading either of these memory locations will generate an address error interrupt.

4.3.12 AFEU FIFOs

AFEU uses an input FIFO/output FIFO pair to hold data before and after the encryption process. These FIFOs are multiply addressable, but those multiple addresses point only to the appropriate end of the appropriate FIFO. A write to anywhere in the AFEU FIFO address space causes the 32-bit-word to be pushed onto the AFEU input FIFO, and a read from anywhere in the AFEU FIFO Address space causes a 32-bit-word to be popped off of the AFEU output FIFO. Overflows and under flows caused by reading or writing the AFEU FIFOs are reflected in the AFEU interrupt status register.

4.4 Message Digest Execution Units (MDEU)

This section contains details about the Message Digest Execution Units (MDEU), including detailed register map, modes of operation, status and control registers, and FIFOs.

4.4.1 MDEU Register Map

The registers used in the MDEU are documented primarily for debug and target mode operations. If the MPC184 requires the use of the MDEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

Each of the 2 instances of MDEU contains the following registers:

- MDEU Mode Register
- Key Size Register
- Data Size Register
- Reset Control Register
- Status Register
- Interrupt Status Register
- Interrupt Control Register

- “Go” Register
- Context Registers
- Key Registers
- MDEU Input FIFO

4.4.2 MDEU Mode Register

The MDEU Mode Register, shown in Figure 4-26, contains 8 bits which are used to program the MDEU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the MPC184 as an initiator. Burst size is not relevant to target mode operations, where an external host pushes and pulls data from the execution units.

The mode register is cleared when the MDEU is reset or re-initialized. Setting a reserved mode bit will generate a data error. If the mode register is modified during processing, a context error will be generated.

	0	1	2	3	4	5	6	7	8	12	13	15	16	31
Field	Cont	--		INT	HMAC	PD	ALG		—		BURST SIZE		—	
Reset	0													
R/W	R/W													
Addr	MDEU 0x0C000													

	0	31																												
Field	Reserved																													
Reset	0																													
R/W	R/W																													
Addr	MDEU 0x0C004																													

Figure 4-26. MDEU Mode Register

Table 4-17 describes MDEU Mode Register signals.

Table 4-17. MDEU Mode Register

Bits	Signal	Description
0	Cont	Continue (Cont): Used during HMAC/HASH processing when the data to be hashed is spread across multiple descriptors. 0 = Don't Continue- operate the MDEU in auto completion mode. 1 = Preserve context to operate the MDEU in Continuation mode.
1–2	—	Reserved

Table 4-17. MDEU Mode Register (continued)

Bits	Signal	Description
3	INT	Initialization Bit (INT): Cause an algorithm-specific initialization of the digest registers. Most operations will require this bit to be set. Only static operations that are continuing from a know intermediate hash value would not initialize the registers. 0 Do not initialize 1 Initialize the selected algorithm's starting registers
4	HMAC	Identifies the hash operation to execute: 0 Perform standard hash 1 Perform HMAC operation. This requires a key and key length information.
5	PD	If set, configures the MDEU to automatically pad partial message blocks. 0 Do not autopad 1 Perform automatic message padding whenever an incomplete message block is detected.
6–7	ALG	Message Digest algorithm selection 00 = SHA-160 algorithm (full name for SHA-1) 01 = SHA-256 algorithm 10 = MD5 algorithm 11 = Reserved
8–12	—	Reserved
13–15	BURST SIZE	The implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/context. The MDEU signals to the crypto-channel that a “Burst Size” amount of data is available to be pushed to the FIFO. Note: The inclusion of this field in the MDEU Mode Register is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the MDEU.
16–31	—	Reserved

4.4.2.1 Recommended settings for MDEU Mode Register

The most common task likely to be executed via the MDEU is HMAC generation. HMACs are used to provide message integrity within a number of security protocols, including IPSec, and SSL/TLS. When the HMAC is being generated by a single dynamic descriptor (the MDEU acting as sole or secondary EU), the following Mode Register bit settings should be used:

Continue-Off, Initialize-On, HMAC-On, Autopad-On

When the HMAC is being generated for a message that is spread across a chain of static descriptors, the following Mode Register bit settings should be used:

First Descriptor:

Continue-On, Initialize-On, HMAC-On, Autopad-Off

Middle Descriptor(s):

Continue-On, Initialize-Off, HMAC-Off, Autopad-Off

Final Descriptor

Continue-Off, Initialize-Off, HMAC-On, Autopad-On

Additional information on descriptors can be found in Chapter 5.

4.4.3 MDEU Key Size Register

Displayed in Figure 4-27, this value indicates the number of bits of key memory that should be used in HMAC generation. MDEU supports at most 512 bits of key. MDEU will generate a key size error if the value written to this register exceeds 512 bits, or if a non-zero value is written when the MDEU Mode Register indicates no HMAC.

Talos

	0	1	7	8	31
Field	--	Key Size		Reserved	
		msb<----lsb			
Reset	0				
R/W	R/W				
Addr	MDEU 0x0C008				

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	MDEU 0x0C00C	

Figure 4-27. MDEU Key Size Register

4.4.4 MDEU Data Size Register

The MDEU Data Size Register, shown in Figure 4-28, stores the size of the last block of data (in bits) to be processed. The first three bits are used to check for a bit offset in the last byte of the message. Since the engine does not support bit offsets, any value other than ‘0’ in these positions will cause a data size error. The next three bits are used to identify the ending byte location in the last 8-byte dword. This is used to add the data padding when auto padding is selected. This register is cleared when the MDEU is reset, re-initialized, and at the end of processing the complete message.

NOTE

Writing to the data size register will allow the MDEU to enter auto-start mode. Therefore, the required context data should be written prior to writing the data size.

	0	1	2	7	8	31
Field	Reserved		Data Size		Reserved	
		msb<----lsb				
Reset	0					
R/W	R/W					
Addr	MDEU 0x0C010					

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	MDEU 0x0C014	

Figure 4-28. MDEU Data Size Register

4.4.5 MDEU Reset Control Register

This register, shown in Figure 4-29, allows 3 levels reset of just the MDEU, as defined by the 3 self-clearing bits:

	0	4	5	6	7	8	31
Field	Reserved		RI	MI	SR	Reserved	
Reset	0	0	0	0	0		
R/W	R/W						
Addr	MDEU 0x0C018						

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	MDEU 0x0C01C	

Figure 4-29. MDEU Reset Control Register

Table 4-18 describes MDEU Reset Control Register signals.

Table 4-18. MDEU Reset Control Register Signal

Bits	Signal	Description
0-4	—	Reserved
5	Reset Interrupt	Writing this bit active high causes MDEU interrupts signalling DONE and ERROR to be reset. It further resets the state of the MDEU Interrupt Status Register. 0 No reset 1 Reset interrupt logic

Table 4-18. MDEU Reset Control Register Signal (continued)

Bits	Signal	Description
6	Module Init	Module initialization is nearly the same as software reset, except that the MDEU Interrupt Control Register remains unchanged. 0 No reset 1 Reset most of MDEU
7	SW_RESET	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the MDEU. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full MDEU reset
8-31	—	Reserved

4.4.6 MDEU Status Register

This status register, as seen in Figure 4-30, contains 5 bits which reflect the state of the MDEU internal signals.

The MDEU Status Register is read-only. Writing to this location will result in address error being reflected in the MDEU Interrupt Status Register.

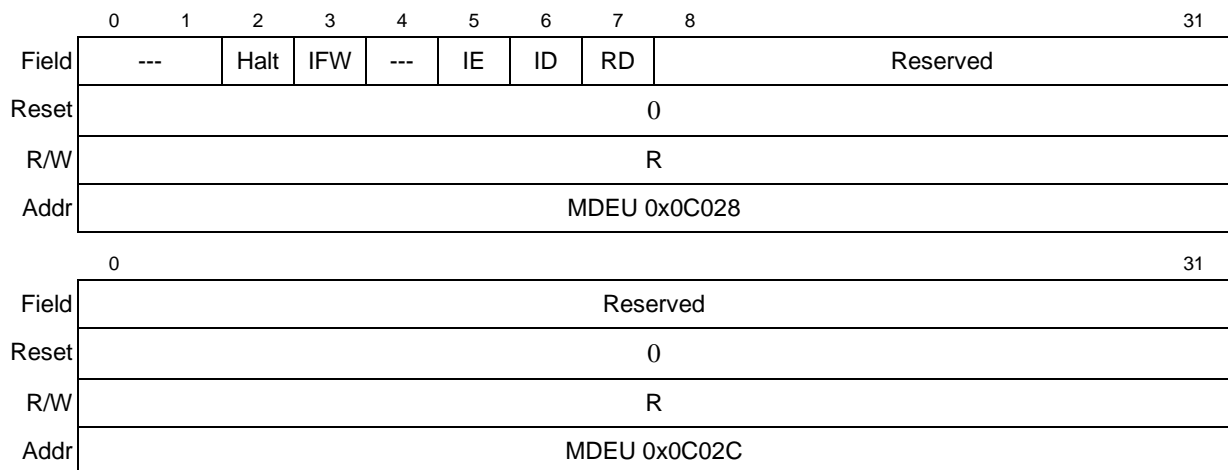

Figure 4-30. MDEU Status Register

Table 4-14 describes MDEU Status Register signals.

Table 4-19. MDEU Status Register Signals

Bits	Signal	Description
0-1	—	Reserved
2	Halt	Halt- Indicates that the MDEU has halted due to an error. 0 MDEU not halted 1 MDEU halted Note: Because the error causing the MDEU to stop operating may be masked to the Interrupt Status Register, the Status Register is used to provide a second source of information regarding errors preventing normal operation.

Table 4-19. MDEU Status Register Signals

Bits	Signal	Description
3	IFW	Input FIFO Writable- The Controller uses this signal to determine if the MDEU can accept the next BURST SIZE block of data. 0 MDEU Input FIFO not ready 1 MDEU Input FIFO ready Note: The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The MDEU signals to the crypto-channel that a "Burst Size" amount of space is available in the FIFO. The documentation of this bit in the MDEU Status Register is to avoid confusing a user who may read this register in debug mode.
4	—	Reserved
5	Interrupt_Error	This status bit reflects the state of the ERROR interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, "Interrupt Status Registers"). 0 MDEU is not signaling error 1 MDEU is signaling error
6	Interrupt_Done	This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, "Interrupt Status Registers"). 0 MDEU is not signaling done 1 MDEU is signaling done
7	Reset_Done	This status bit, when high, indicates that MDEU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel. 0 Reset in progress 1 Reset done
8-31	—	Reserved

4.4.7 MDEU Interrupt Status Register

The interrupt status register tracks the state of possible errors, if those errors are not masked, via the MDEU Interrupt Control Register. The definition of each bit in the interrupt status register is shown in Figure 4-31.

	0	1	2	4	5	6	10	11	12	13	14	15	16	31
Field	ME	AE	Reserved	IFO	•	Reserved	IE	ERE	CE	KSE	DSE	Reserved		
Reset	0													
R/W	R													
Addr	MDEU 0x0C030													
	0													31
Field	RESERVED													
Reset	0													
R/W	R													
Addr	MDEU 0x0C034													

Figure 4-31. MDEU Interrupt Status Register

Table 4-20 describes MDEU Interrupt Status Register signals.

Table 4-20. MDEU Interrupt Status Register Signals

Bits	Signal	Description
0	Mode Error	An illegal value was detected in the mode register. Note: writing to reserved bits in mode register is likely source of error. 0 No error detected 1 Mode error
1	Address Error	An illegal read or write address was detected within the MDEU address space. 0 No error detected 1 Address Error
2-4	—	Reserved
5	Input FIFO Overflow	the MDEU Input FIFO has been pushed while full. 0 No overflow detected 1 Input FIFO has overflowed Note: When operating as a master, the MPC184 implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the MPC184 cannot accept FIFO inputs larger than 512B without overflowing.
6-10	—	Reserved
11	Internal Error	Indicates the MDEU has been locked up and requires a reset before use. 0 No internal error detected 1 Internal error detected Note: This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Error Interrupt Control Register or by resetting the MDEU.
12	Early Read Error	The MDEU context was read before the MDEU completed the hashing operation. 0 No error detected 1 Early read error
13	Context Error	The MDEU key register, key size register, or data size register was modified while MDEU was hashing. 0 No error detected 1 Context error
14	Key Size Error	A value greater than 512 bits was written to the MDEU key size register. 0 No error detected 1 Key size error
15	Data Size Error	A value not a multiple of 512 bits while the MDEU Mode Register autopad bit is negated. 0 No error detected 1 Data size error
16-31	—	Reserved

4.4.8 MDEU Interrupt Control Register

The MDEU Interrupt Control Register, shown in Figure 4-32, controls the result of detected errors. For a given error (as defined in Section 4.4.7, “MDEU Interrupt Status Register”),

if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0	1	2	4	5	6	10	11	12	13	14	15	16	31
Field	DE	AE	RESERVED	IFO	RESERVED		IE	ERE	CE	KSE	DSE	RESERVED		
Reset	0													
R/W	R/W													
Addr	MDEU 0x0C038													

	0	31																										
Field	RESERVED																											
Reset	0																											
R/W	R/W																											
Addr	MDEU 0x0C03C																											

Figure 4-32. MDEU Interrupt Control Register

Table 4-21 describes MDEU Interrupt Status Register signals.

Table 4-21. MDEU Interrupt Control Register Signals

Bits	Signal	Description
0	Mode Error	An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
1	Address Error	An illegal read or write address was detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
2-4	—	Reserved
5	Input FIFO Overflow	The MDEU Input FIFO has been pushed while full. 0 Input FIFO overflow error enabled 1 Input FIFO overflow error disabled
6-10	—	Reserved
11	Internal Error	An internal processing error was detected while performing hashing. 0 Internal error enabled 1 Internal error disabled
12	Early Read Error	The MDEU Register was read while the MDEU was performing hashing. 0 Early read error enabled 1 Early read error disabled
13	Context Error	The MDEU key register, the key size register, the data size register, or the mode register, was modified while the MDEU was performing hashing. 0 Context error enabled 1 Context error disabled

Table 4-21. MDEU Interrupt Control Register Signals

Bits	Signal	Description
14	Key Size Error	A value outside the bounds 512 bits was written to the MDEU key size register 0 Key size error enabled 1 Key size error disabled
15	Data Size Error	An inconsistent value was written to the MDEU Data Size Register: 0 Data Size error enabled 1 Data size error disabled
16-31	—	Reserved

4.4.9 MDEU EU_GO Register

The EU_GO Register in the MDEU, see Figure 4-33, is used to indicate an authentication operation may be completed. After the final message block is written to the input FIFO, the EU-GO Register must be written. The value in the data size register will be used to determine how many bits of the final message block (always 512) will be processed. Note that this register has no data size, and during the write operation, the host data bus is not read. Hence, any data value is accepted. Normally, a write operation with a zero data value is performed. Moreover, no read operation from this register is meaningful, but no error is generated, and a zero value is always returned. Writing to this register is merely a trigger causing the MDEU to process the final block of a message, allowing it to signal DONE.

The MDEU EU_GO Register is only used when the MPC184 is operated as a target. The descriptors and crypto-channel activate the MDEU (via an internally generated write to the EU_Go register) when the MPC184 acts as an initiator.

	0	31
Field	MDEU EU_GO	
Reset	0	
R/W	W	
Addr	MDEU 0x0C050	

Figure 4-33. MDEU EU_GO Register

4.4.10 MDEU Context Registers

For MDEU, context consists of the hash plus the message length count. Write access to this register block allows continuation of a previous hash. Reading these registers provide the resulting message digest or HMAC, along with an aggregate bitcount.

NOTE

SHA-1 and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the five registers A, B, C, D, and E upon writing to or reading from the MDEU context if the MDEU mode register indicates MD5 is the hash of choice. Most other endian considerations are performed as 8 byte swaps. In this case, 4-byte endianness swapping is performed within the A, B, C, D, and E fields as individual registers. Reading this memory location while the module is not done will generate an error interrupt.

Bits	0	31	0	31		
Name	A		Context offset\$100	B		Context offset\$104
Reset (MD5, SHA-1)	0x01234567			0x89abcdef		
Reset (SHA-256)	0x67e6096a			0x85ae67bb		
Name	C		Context offset\$108	D		Context offset\$10C
Reset (MD5, SHA-1)	0xfedcba98			0x76543210		
Reset (SHA-256)	0x72f36e3c			0x3af54fa5		
Name	E		Context offset\$110	F		Context offset\$114
Reset (MD5, SHA-1)	0xf0e1d2c3			0x00000000		
Reset (SHA-256)	0x7f520e51			0x8c68059b		
Name	G		Context offset\$118	H		Context offset\$11C
Reset (MD5, SHA-1)	0x00000000			0x00000000		
Reset (SHA-256)	0xabd9831f			0x19cde05b		
Name	Message Length Count					Context offset\$120
Reset	0					

4.4.11 MDEU Key Registers

The MDEU maintains sixteen 32-bit registers for writing an HMAC key. The IPAD and OPAD operations are performed automatically on the key data when required. Reading any of these memory locations will generate an address error interrupt.

NOTE

SHA-1 and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU mode register indicates MD5 is the hash of choice.

4.4.12 MDEU FIFOs

MDEU uses an input FIFO to hold data to be hashed. The input FIFO is multiply addressable, but those multiple addresses point only to the write (push) end of the FIFO. A write to anywhere in the MDEU FIFO address space causes the 32-bit-words to be pushed onto the MDEU input FIFO, and a read from anywhere in the MDEU FIFO address space causes the address error bit of the interrupt status register to be set.

NOTE

SHA-1 and SHA-256 are big endian. MD5 is little endian. The MDEU module internally reverses the endianness of the key upon writing to or reading from the MDEU key registers if the MDEU mode register indicates MD5 is the hash of choice.

4.5 Random Number Generator (RNG)

This section contains details about the Random Number Generator (RNG), including detailed register map, modes of operation, status and control registers, and FIFOs.

4.5.1 Overview

The RNG is an execution unit capable of generating 32-bit random numbers. It is designed to comply with the FIPS-140 standard for randomness and non-determinism. A linear feedback shift register (LFSR) and cellular automata shift register (CASR) are operated in parallel to generate pseudo-random data.

4.5.2 Functional Description

The RNG consists of six major functional blocks:

- Bus interface unit (BIU)
- Linear feedback shift register (LFSR)
- Cellular automata shift register (CASR)
- Clock controller
- Two ring oscillators

The states of the LFSR and CASR are advanced at unknown frequencies determined by the two ring oscillator clocks and the clock control. When a read is performed, the oscillator

clocks are halted and a collection of bits from the LFSR and CASR are XORED together to obtain the 64-bit random output.

4.5.3 RNG Register Map

The registers used in the MDEU are documented primarily for debug and target mode operations. If the MPC184 requires the use of the MDEU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

The single RNG contains the following registers:

- RNG mode register
- Data size register
- Reset control register
- Status register
- Interrupt status register
- Interrupt control register
- RNG output FIFO

4.5.4 RNG Mode Register

The RNG Mode Register is used to control the RNG. One operational mode, randomizing, is defined. Writing any other value than 0 to 0:12 results in a data error interrupt that's reflected in the RNG Interrupt Status Register. The mode register also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the MPC184 as an initiator. Burst size is not relevant to target mode operations, where an external host pushes and pulls data from the execution units.

The mode register is cleared when the RNG is reset or re-initialized. The RNG mode register is shown in Figure 4-34.

	0	12	13	15	16	31
Field	Reserved		Burst Count		Reserved	
Reset	0					
R/W	R/W					
Addr	0x0E000					

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	0x0E004	

Figure 4-34. RNG Mode Register

Table 4-22. RNG Mode Register Definitions

Bits	Signal	Description
0:12	—	Reserved, must be set to zero.
13:15	Burst Count	The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/context. The RNG signals to the crypto-channel that a “Burst Size” amount of data is available to be pulled from the FIFO. Note: The inclusion of this field in the RNG Mode Register is to avoid confusing a user who may read this register in debug mode. Burst Size should not be written directly to the RNG.
16:31	—	Reserved

4.5.5 RNG Data Size Register

The RNG Data Size Register is used to tell the RNG to begin generating random data. The actual contents of the data size register does not affect the operation of the RNGA. After a reset and prior to the first write of data size, the RNG builds entropy without pushing data onto the FIFO. Once the data size register is written, the RNG will begin pushing data onto the FIFO. Data will be pushed onto the FIFO every 256 cycles until the FIFO is full. The RNG will then attempt to keep the FIFO full.

	0	31
Field	RNG Data Size	
Reset	0	
R/W	R/W	
Addr	0x0E010	

Figure 4-35. RNG Data Size Register

4.5.6 RNG Reset Control Register

This register, shown in Figure 4-36, contains three reset options specific to the RNG.

	0	4	5	6	7	8	31
Field	Reserved			RI	MI	SR	Reserved
Reset	0			0	0	0	0
R/W	R/W						
Addr	0x0E018						

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	RNG 0x0E01C	

Figure 4-36. RNG Reset Control Register

Table 4-23 describes RNG reset control register signals.

Table 4-23. RNG Reset Control Register Signals

Bits	Signal	Description
0-4	—	Reserved
5	Reset Interrupt	Writing this bit active high causes RNG interrupts signalling DONE and ERROR to be reset. It further resets the state of the RNG interrupt status register. 0 No reset 1 Reset interrupt logic
6	Module Init	This reset value performs enough of a reset to prepare the RNG for another request, without forcing the internal control machines and the output FIFO to be reset, thereby invalidating stored random numbers or requiring reinvocation of a warm-up period. Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. 0 No reset 1 Reset most of RNG
7	SW_RESET	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for the RNG. All registers and internal state are returned to their defined reset state. 0 No reset 1 Full RNG reset
8-31	—	Reserved

4.5.7 RNG Status Register

This RNG Status Register, Figure 4-37, contains 4 bits which reflect the state of the RNG internal signals.

The RNG Status Register is read-only. Writing to this location will result in an address error being reflected in the RNG interrupt status register.

	0	1	2	3	4	5	6	7	8		31
Field	---	Halt	--	OFR	IE	--	RD	Reserved			
Reset	0										
R/W	R										
Addr	RNG 0x0E028										

	0		31																											
Field	Reserved																													
Reset	0																													
R/W	R																													
Addr	RNG 0x0E02C																													

Figure 4-37. RNG Status Register

Table 4-24 describes RNG Status Register signals.

Table 4-24. RNG Status Register Signals

Bits	Signal	Description
0-1	—	Reserved
2	Halt	Halt- Indicates that the RNG has halted due to an error. 0 RNG not halted 1 RNG halted Note: Because the error causing the RNG to stop operating may be masked to the Interrupt Status Register, the Status Register is used to provide a second source of information regarding errors preventing normal operation.
3	—	Reserved
4	OFR	Output FIFO Readable- The Controller uses this signal to determine if the RNG can source the next BURST SIZE block of data. 0 RNG Output FIFO not ready 1 RNG Output FIFO ready
5	Interrupt_Error	This status bit reflects the state of the ERROR interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, "Interrupt Status Registers"). 0 RNG is not signaling error 1 RNG is signaling error
6	---	Reserved
7	RESET_DONE	This status bit, when high, indicates that the RNG has completed its reset sequence. 0 Reset in progress 1 Reset done
8-31	—	Reserved

4.5.8 RNG Interrupt Status Register

The RNG Interrupt Status Register tracks the state of possible errors, if those errors are not masked, via the RNG interrupt control register. The definition of each bit in the interrupt status register is shown in Figure 4-38.

	0	1	2		5	6	7		10	11	12		31
Field	ME	AE	Reserved			OFU	Reserved			IE	Reserved		
Reset	0												
R/W	R												
Addr	RNG 0x0E030												

	0												31
Field	Reserved												
Reset	0												
R/W	R												
Addr	RNG 0x0E034												

Figure 4-38. RNG Interrupt Status Register

Table 4-25 describes RNG interrupt status register signals.

Table 4-25. RNG Interrupt Status Register Signals

Bits	Signal	Description
0	Mode Error	Indicates that the host has attempted to write an illegal value to the Mode register 0 = Valid Data 1 = Invalid Data Error
1	Address Error	An illegal read or write address was detected within the RNG address space. 0 No error detected 1 Address error
2-5	----	Reserved
6	Output FIFO Underflow	The RNG Output FIFO has been read while empty. 0 No overflow detected 1 Output FIFO has under flowed
7-10	---	Reserved
11	Internal Error	0 No internal error detected 1 Internal error
12-31	----	Reserved

4.5.9 RNG Interrupt Control Register

The RNG Interrupt Control Register controls the result of detected errors. For a given error (as defined in Section 4.5.8, “RNG Interrupt Status Register”), if the corresponding bit in this register is set, then the error is disabled; no error interrupt occurs and the interrupt status

register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.

	0	1	2	5	6	7	10	11	12	31
Field	ME	AE	RESERVED	OFU	Reserved	IE	RESERVED			
Reset	0									
R/W	R/W									
Addr	RNG 0x0E038									

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	RNG 0x0E03C	

Figure 4-39. RNGA Interrupt Control Register

Table 4-26 describes RNG interrupt status register signals.

Table 4-26. RNG Interrupt Control Register Signals

Bits	Signal	Description
0	Mode Error	An illegal value was detected in the mode register. 0 Mode error enabled 1 Mode error disabled
1	Address Error	An illegal read or write address was detected within the MDEU address space. 0 Address error enabled 1 Address error disabled
2-5	----	Reserved
6	Output FIFO Underflow	RNG Output FIFO has been read while empty. 0 Output FIFO underflow error enabled 1 Output FIFO underflow error disabled
7-10	—	Reserved
11	Internal Error	An internal processing error was detected while generating random numbers. 0 Internal error enabled 1 Internal error disabled
12-31	----	Reserved

4.5.10 RNG EU_GO Register

The RNG EU_Go is a writable location but serves no function in the RNG. It is documented for the sake of consistency with the other EU's.

	0	31
Field	RNG EU_GO	
Reset	0	
R/W	W	
Addr	RNG 0x0E050	

Figure 4-40. RNG EU_GO Register

4.5.11 RNG FIFO

RNG uses an output FIFO to collect periodically sampled random 64-bit-words, with the intent that random data always be available for reading. The FIFO is multiply addressed, but those multiple addresses point only to the appropriate end of the output FIFO. A read from anywhere in the RNG FIFO address space causes a 32-bit-word to be popped off of the RNG output FIFO. Under flows caused by reading or writing the RNG output FIFO are reflected in the RNG interrupt status register. Also, a write to the RNG output FIFO space will be reflected as an addressing error in the RNG interrupt status register.

4.6 Advanced Encryption Standard Execution Units (AESU)

This section contains details about the Advanced Encryption Standard Execution Units (AESU), including detailed register map, modes of operation, status and control registers, and FIFOs.

4.6.1 AESU Register Map

The registers used in the AESU are documented primarily for debug and target mode operations. If the MPC184 requires the use of the AESU when acting as an initiator, accessing these registers directly is unnecessary. The device drivers and the on-chip controller will abstract register level access from the user.

The AESU contains the following registers:

- AESU Mode Register
- Key Size Register
- Data Size Register
- Reset Control Register
- Status Register
- Interrupt Status Register
- Interrupt Control Register
- End Of Message Register

- IV Registers
- Key Registers
- AESU FIFOs

4.6.2 AESU Mode Register

The AESU Mode Register, shown in Figure 4-41, contains 4 bits which are used to program the AESU. It also reflects the value of burst size, which is loaded by the crypto-channel during normal operation with the MPC184 as an initiator. Burst size is not relevant to target mode operations, where an external host pushes and pulls data from the execution units.

The mode register is cleared when the AESU is reset or re-initialized. Setting a reserved mode bit will generate a data error. If the mode register is modified during processing, a context error will be generated.

Figure 4-41. AESU Mode Register

Table 4-6 describes AESU mode register signals.

Table 4-27. AESU Mode Register Signals

Bits	Signal	Description
0-3	—	Reserved
4	RDK	Restore Decrypt Key (RDK): Specifies that key data write will contain pre-expanded key (decrypt mode only). 0 Expand the user key prior to decrypting the first block 1 Do not expand the key. The expanded decryption key will be written following the context switch.
5-6	CM	Cipher Mode: Controls which cipher mode the AESU will use in processing: 00 ECB -Electronic Codebook mode. 01 CBC- Cipher Block Chaining mode. 10 Reserved 11 CTR- Counter Mode.
7	Encrypt/Decrypt	If set, AESU operates the encryption algorithm; if not set, AESU operates the decryption algorithm. Note: This bit is ignored if CM is set to “11” - CTR Mode. 0 Perform decryption 1 Perform encryption
8-12	—	Reserved
13-15	Burst Size	The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/context. The AESU signals to the crypto-channel that a “burst size” amount of data is available to be pushed to or pulled from the FIFO. Note: The inclusion of this field in the AESU mode register is to avoid confusing a user who may read this register in debug mode. Burst size should not be written directly to the AESU.
16:31	—	Reserved

NOTE: Restore Decrypt Key

In most networking applications, the decryption of an AES protected packet will be performed as a single operation. However, if circumstances dictate that the decryption of a message should be split across multiple descriptors, the AESU allows the user to save the decrypt key, and the active AES context, to memory for later re-use. This saves the internal AESU processing overhead associated with regenerating the decryption key schedule (~12 AESU clock cycles for the first block of data to be decrypted.)

The use of RDK is completely optional, as the Input time of the preserved decrypt key may exceed the ~12 cycles required to restore the decrypt key for processing the first block.

To use RDK, the following procedure is recommended:

The descriptor type used in decryption of the first portion of the message is “0100- AESU Key Expand Output”. The description mode must be “Decrypt”. See Chapter 5 “Descriptors” for more information. The descriptor will cause the MPC184 to write the contents of the Context registers and the key registers (containing the expanded decrypt key) to memory.

To process the remainder of the message, use a “normal” descriptor type (descriptor type selected based on need for simultaneous HMAC generation, etc.), and set the “restore decrypt key” mode bit. Load the context registers and the expanded decrypt key with previously saved key and context data from the first message. The key size is written as before (16, 24, or 32 bytes).

4.6.3 AESU Key Size Register

The AESU Key Size Register stores the number of bytes in the key (16,24,32). Any key data beyond the number of bytes in the key size register will be ignored. This register is cleared when the AESU is reset or re-initialized. If a key size other than 16, 24, or 32 bytes is specified, an illegal key size error will be generated. If the key size register is modified during processing, a context error will be generated.

	0	1	2	7	8	31		
Field	Reserved		Key Size			Reserved		
			msb<-----lsb					
Reset	0							
R/W	R/W							
Addr	AESU 0x12008							

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	AESU 0x1200C	

Figure 4-42. AESU Key Size Register

4.6.4 AESU Data Size Register

This AESU Data Size Register is used to verify that the data to be processed by the AESU is divisible by the AES algorithm block size of 128-bits. The AESU does not automatically pad messages out to 128-bit blocks, therefore any message processed by the AESU must be divisible by 128-bits or a data size error will occur.

In normal operation, the full message length to be encrypted or decrypted with the AESU is copied from the descriptor to the AESU data size register, however only bits 1:7 are checked to determine if there is a data size error. If 1:7 are all zeroes, the message is evenly divisible into 128-bit blocks.

This register is cleared when the AESU is reset or re-initialized. If a data size other than 128-bits is specified, an illegal data size error will be generated. Writing to this register signals the AESU to start processing data from the input FIFO as soon as it is available. If the value of data size is modified during processing, a context error will be generated.

	0	1	7	8	31
Field	---	Data Size		Reserved	
	---	msb<-----lsb			
Reset	0				
R/W	R/W				
Addr	AESU 0x12010				

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	AESU 0x12014	

Figure 4-43. AESU Data Size Register

4.6.5 AESU Reset Control Register

This register allows 3 levels reset of just AESU, as defined by the 3 self-clearing bits:

	0	4	5	6	7	8	31
Field	Reserved		RI	MI	SR	Reserved	
Reset	0		0	0	0	0	
R/W	R/W						
Addr	AESU 0x12018						

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	AESU 0x1201C	

Figure 4-44. AESU Reset Control Register

Table 4-28 describes AESU reset control register signals.

Table 4-28. AESU Reset Control Register Signals

Bits	Signals	Description
0:4	—	Reserved
5	Reset Interrupt	Writing this bit active high causes AESU interrupts signalling DONE and ERROR to be reset. It further resets the state of the AESU interrupt status register. 0 Don't reset 1 Reset interrupt logic

Table 4-28. AESU Reset Control Register Signals

Bits	Signals	Description
6	Module_Init	Module initialization is nearly the same as software reset, except that the interrupt control register remains unchanged. This module initialization includes execution of an initialization routine, completion of which is indicated by the RESET_DONE bit in the AESU status register 0 Don't reset 1 Reset most of AESU
7	SW_RESET	Software reset is functionally equivalent to hardware reset (the RESET# pin), but only for AESU. All registers and internal state are returned to their defined reset state. Upon negation of SW_RESET, the AESU will enter a routine to perform proper initialization of the parameter memories. The RESET_DONE bit in the AESU status register will indicate when this initialization routine is complete 0 Don't reset 1 Full AESU reset
8:31	—	Reserved

4.6.6 AESU Status Register

AESU status register is a read-only register that reflects the state of six status outputs. Writing to this location will result in an address error being reflected in the AESU interrupt status register.

	0	1	2	3	4	5	6	7	8	31	
Field	---	Halt	IFW	OFR	IE	ID	RD	Reserved			
Reset	0										
R/W	W										
Addr	AESU 0x12028										
	0										31
Field	Reserved										
Reset	0										
R/W	R										
Addr	AESU 0x0A02C										

Figure 4-45. AESU Status Register

Table 4-29 describes AESU status register signals.

Table 4-29. AESU Status Register Signals

Bits	Signal	Description
0-1	—	Reserved
2	Halt	<p>Halt- Indicates that the AESU has halted due to an error.</p> <p>0 AESU not halted</p> <p>1 AESU halted</p> <p>Note: Because the error causing the AESU to stop operating may be masked to the interrupt status register, the status register is used to provide a second source of information regarding errors preventing normal operation.</p>
3	IFW	<p>Input FIFO Writable- The Controller uses this signal to determine if the AESU can accept the next BURST SIZE block of data.</p> <p>0 AESU Input FIFO not ready</p> <p>1 AESU Input FIFO ready</p> <p>Note: The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The AESU signals to the crypto-channel that a 'burst size' amount of space is available in the FIFO. The documentation of this bit in the AESU status register is to avoid confusing a user who may read this register in debug mode.</p>
4	OFR	<p>Output FIFO Readable- The controller uses this signal to determine if the AESU can source the next burst size block of data.</p> <p>0 AESU Output FIFO not ready</p> <p>1 AESU Output FIFO ready</p> <p>Note: The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The AESU signals to the crypto-channel that a "Burst Size" amount of data is available in the FIFO. The documentation of this bit in the AESU Status Register is to avoid confusing a user who may read this register in debug mode.</p>
5	Interrupt_Error	<p>This status bit reflects the state of the ERROR interrupt signal, as sampled by the controller interrupt status register (Section 7.1.4, "Interrupt Status Registers").</p> <p>0 AESU is not signaling error</p> <p>1 AESU is signaling error</p>
6	Interrupt_Done	<p>This status bit reflects the state of the DONE interrupt signal, as sampled by the Controller Interrupt Status Register (Section 7.1.4, "Interrupt Status Registers").</p> <p>0 AESU is not signaling done</p> <p>1 AESU is signaling done</p>
7	Reset_Done	<p>This status bit, when high, indicates that AESU has completed its reset sequence, as reflected in the signal sampled by the appropriate crypto-channel.</p> <p>0 Reset in progress</p> <p>1 Reset done</p>
8-31	—	Reserved

4.6.7 AESU Interrupt Status Register

The AESU interrupt status register tracks the state of possible errors, if those errors are not masked, via the AESU interrupt control register. The definition of each bit in the interrupt status register is shown in Figure 4-46.

	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	31
Field	ME	AE	OFE	IFE	RSV	IFO	OFU	---		IE	ERE	CE	KSE	DSE	Reserved	
Reset	0															
R/W	R															
Addr	AESU 0x12030															

	0																31
Field	RESERVED																
Reset	0																
R/W	R																
Addr	AESU 0x12034																

Figure 4-46. AESU Interrupt Status Register

Table 4-30 describes AESU interrupt register signals.

Table 4-30. AESU Interrupt Status Register Signals

Bits	Signal	Description
0	Mode Error	Mode Error. Indicates that invalid data was written to a register or a reserved mode bit was set. 0 = Valid Data 1 = Reserved or invalid mode selected
1	Address Error	An illegal read or write address was detected within the AESU address space. 0 No error detected 1 Address error
2	Output FIFO Error	The AESU output FIFO was detected non-empty upon write of AESU data size register. 0 No error detected 1 Output FIFO non-empty error
3	Input FIFO Error	The AESU input FIFO was detected non-empty upon generation of done interrupt. 0 No error detected 1 Input FIFO non-empty error
4	—	Reserved
5	Input FIFO Overflow	The AESU Input FIFO has been pushed while full. 0 No error detected 1 Input FIFO has overflowed Note: When operating as a master, the MPC184 implements flow-control, and FIFO size is not a limit to data input. When operated as a target, the MPC184512B cannot accept FIFO inputs larger than 512 Bytes without overflowing.
6	Output FIFO Underflow	The AESU Output FIFO has been read while empty. 0 No error detected 1 Output FIFO has underflow error
7-10	—	Reserved

Table 4-30. AESU Interrupt Status Register Signals (continued)

Bits	Signal	Description
11	Internal Error	An internal processing error was detected while the AESU was processing. 0 No error detected 1 Internal error Note: This bit will be asserted any time an enabled error condition occurs and can only be cleared by setting the corresponding bit in the Interrupt Control Register or by resetting the AESU.
12	Early Read Error	The AESU IV Register was read while the AESU was processing. 0 No error detected 1 Early read error
13	Context Error	An AESU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while AESU was processing 0 No error detected 1 Context error
14	Key Size Error	An inappropriate value (not 16, 24 or 32bytes) was written to the AESU Key Size Register 0 No error detected 1 Key size error
15	Data Size Error	Data Size Error (DSE): A value was written to the AESU Data Size Register that is not a multiple of 128 bits. 0 No error detected 1 Data size error
16-31	—	Reserved

4.6.8 AESU Interrupt Control Register

The AESU Interrupt Control Register, shown in Figure 4-47, controls the result of detected errors. For a given error (as defined in Section 4.6.7, “AESU Interrupt Status Register”), if the corresponding bit in this register is set, then the error is ignored; no error interrupt occurs and the interrupt status register is not updated to reflect the error. If the corresponding bit is not set, then upon detection of an error, the interrupt status register is updated to reflect the error, causing assertion of the error interrupt signal, and causing the module to halt processing.



Freescal Semiconductor, Inc.

Advanced Encryption Standard Execution Units (AESU)

	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	31
Field	ME	AE	OFE	IFE	RSV	IFO	OFU	RSV	IE	ERE	CE	KSE	DSE	Reserved		
Reset	0															
R/W	R/W															
Addr	AES 0x12038															

	0	31
Field	Reserved	
Reset	0	
R/W	R/W	
Addr	AESU 0x1203C	

Figure 4-47. AESU Interrupt Control Register

Table 4-31 describes the AESU interrupt control register signals.

Table 4-31. AESU Interrupt Control Register Signals

Bits	Signal	Description
0	Mode Error	Mode Error: Indicates that invalid data was written to a register or a reserved mode bit was set. 0 = Mode error enabled 1 = Mode error disabled
1	Address Error	An illegal read or write address was detected within the AESU address space. 1 Address error disabled 0 Address error enabled
2	Output FIFO Error	The AESU Output FIFO was detected non-empty upon write of AESU data size register 1 Output FIFO non-empty error disabled 0 Output FIFO non-empty error enabled
3	Input FIFO Error	The AESU Input FIFO was detected non-empty upon generation of done interrupt 1 Input FIFO non-empty error disabled 0 Input FIFO non-empty error enabled
4	—	Reserved
5	Input FIFO Overflow	The AESU Input FIFO has been pushed while full. 1 Input FIFO overflow error disabled 0 Input FIFO overflow error enabled
6	Output FIFO Underflow	The AESU Output FIFO has been read while empty. 1 Output FIFO underflow error disabled 0 Output FIFO underflow error enabled
7-10	—	Reserved
11	Internal Error	An internal processing error was detected while the AESU was processing. 1 Internal error disabled 0 Internal error enabled
12	Early Read Error	The AESU IV Register was read while the AESU was processing. 1 Early read error disabled 0 Early read error enabled

Table 4-31. AESU Interrupt Control Register Signals (continued)

Bits	Signal	Description
13	Context Error	An AESU Key Register, the Key Size Register, Data Size Register, Mode Register, or IV Register was modified while the AESU was processing. 1 Context error disabled 0 Context error enabled
14	Key Size Error	An inappropriate value (non 16, 24 or 32 bytes) was written to the AESU key size register 1 Key size error disabled 0 Key size error enabled
15	Data Size Error	Data Size Error: Indicates that the number of bits to process is out of range. 0 = Data size error enabled 1 = Data size error disabled
16-31	Data Error	Reserved

4.6.9 AESU End of Message Register

The AESU End Of Message Register, shown in Figure 4-48, is used to indicate an AES operation may be completed. After the final message block is written to the input FIFO, the end of message register must be written. The value in the data size register will be used to determine how many bits of the final message block (always 128) will be processed. Writing to this register causes the AESU to process the final block of a message, allowing it to signal DONE. A read of this register will always return a zero value. The AESU end of message register is only used when the MPC184 is operated as a target. The descriptors and crypto-channel activate the AESU (via an internally generated write to the end of message register) when the MPC184 acts as an initiator.

	0	31
Field	AESU End of Message	
Reset	0	
R/W	W	
Addr	AESU 0x12050	

Figure 4-48. AESU End of Message Register

4.6.9.1 AESU Context Registers

There are 3 64-bit context data registers that allow the host to read/write the contents of the context used to process the message. The context must be written prior to the key data. If the context registers are written during message processing, a context error will be generated. All context registers are cleared when a hard/soft reset or initialization is performed.

The context registers must be read when changing context and restored to their original values to resume processing an interrupted message (CBC and CTR modes) . Although

there are 7 64-bit context register fields, only those fields containing data must be read and restored during context switching.

Context should be loaded with the lower bytes in the lowest 64-bit context register. The Context registers are summarized in Figure 4-49.

Context Register (64-bits each)							
Cipher Mode	1	2	3	4	5	6	7
ECB	—	—	—	—	—	—	—
CBC	IV1 ¹	IV2 ¹	—	—	—	—	—
CTR	Counter ¹		Counter Modulus ¹ (msb<--lsb)	—	—	—	—

¹ Must be written at the start of a new message

Figure 4-49. AESU Context Registers

4.6.9.2 Context for CBC Mode

Within the Context register, for use in CBC mode, are two 64-bit context data registers that allow the host to read/write the contents of the initialization vector (IV):

IV1 holds the *least* significant bytes of the initialization vector (bytes 1-8).

IV2 holds the *most* significant bytes of the initialization vector (bytes 9-16).

The IV must be written prior to the message data. If the IV registers are written during message processing, or the **CBC** mode bit is not set, a context error will be generated.

The IV registers may only be read after processing has completed, as indicated by the assertion of Interrupt_Done DONE in the AESU status register as shown in Section 4.6.6, “AESU Status Register”. If the IV registers are read prior to assertion of Interrupt_Done, an early read error will be generated.

The IV registers must be read when changing context and restored to resume processing an interrupted message (**CBC** mode only).

4.6.9.3 Context for Counter Mode

In counter mode, a random 128-bit initial counter value is incremented modulo 2^n with each block processed. The modulus size can be set between 2^8 through 2^{128} , by powers of 8. The running counter is encrypted and eXclusive-ORed with the plaintext to derive the ciphertext, or with the ciphertext to recover the plaintext.

In CTR mode, the block counter is incremented modulo 2^M . The value of M is specified by writing to Context Register 3 as described in Table 4-32

Table 4-32. Counter Modulus

Value Written	Modulus
8	2^8
16	2^{16}
24	2^{24}
32	2^{32}
40	2^{40}
48	2^{48}
56	2^{56}
64	2^{64}
72	2^{72}
80	2^{80}
88	2^{88}
96	2^{96}
104	2^{104}
112	2^{112}
120	2^{120}
128	2^{128}

4.6.9.4 AESU Key Registers

The AESU Key Registers hold from 16, 24, or 32 bytes of key data, with the first 8 bytes of key data written to Key 1. Any key data written to bytes beyond the value written to the key size register will be ignored. The key data registers are cleared when the AESU is reset or re-initialized. If these registers are modified during message processing, a context error will be generated.

The key data registers may be read when changing context in decrypt mode. To resume processing, the value read must be written back to the key registers and the “restore decrypt key” bit must be set in the mode register. This eliminates the overhead of expanding the key prior to starting decryption when switching context.

4.6.9.5 AESU FIFOs

The AESU fetches data 128 bits at a time from the input FIFO. During processing, the input data is encrypted or decrypted with the key and initialization vector (**CBC** mode only) and the results are placed in the output FIFO. The output size is the same as the input size.



Writing to the FIFO address space places 32 bits of message data into the input FIFO. The input FIFO may be written any time the IFW signal is asserted (as indicated in the AESU status register). This will indicate that the number of bytes of available space is at or above the threshold specified in the mode register. There is no limit on the total number of bytes in a message. The number of bits in the final message block must be set in the data size register.

Reading from the FIFO address space will pop 32 bits of message data from the output FIFO. The output FIFO may be read any time the OFR signal is asserted (as indicated in the AESU status register). This will indicate that the number of bytes in the output FIFO is at or above the threshold specified in the mode register.

Chapter 5

MPC184 Descriptors

5.1 Data Packet Descriptor Overview

The MPC184 has 8xx bus mastering capability to off-load data movement and encryption operations from the PowerQuicc host processor. As the system controller, the host processor maintains a record of current secure sessions and the corresponding keys and contexts of those sessions. Once the host has determined a security operation is required, it can either directly write keys, context, and data to the MPC184 (MPC184 in target mode), or the host can create a 'data packet descriptor' to guide the MPC184 through the security operation, with the MPC184 acting as a bus master. The descriptor can be created in main memory, any memory local to the MPC184, or written directly to the data packet descriptor buffer in the MPC184 crypto-channel.

5.2 Descriptor Structure

The MPC184 data packet descriptors are conceptually similar to descriptors used by most devices with DMA capability. See Figure 5-1 for a conceptual data packet descriptor. The descriptors are fixed length (64 bytes), and consist of 16 32-bit fields. Descriptors begin with a header, which describes the security operation to be performed and the mode the execution unit will be set to while performing the operation.

The header is followed by seven data length/data pointer pairs. Data length indicates the amount of contiguous data to be transferred. This amount cannot exceed 32k bytes. The data pointer refers to the address of the data which the MPC184 fetches. Data in this case is broadly interpreted to mean keys, context, additional pointers, or the actual plain text to be permuted.

Figure 5-1 shows an example data packet descriptor. The descriptor consists of 16 32-bit fields, and if written to a cache line boundary, would be fetched in a total of four 4-beat 8xx bus bursts.

	0	31
Word 1	Descriptor Header	
Word 2	Length 1 (Key Length)	
Word 3	Pointer 1 (Key Location)	
Word 4	Length 2 (IV Length)	
Word 5	Pointer 2 (IV Location)	
Word 6	Length 3 (Data-in Length)	
Word 7	Pointer 3 (Data-in Location)	
Word 8	Length 4 (Data-Out Length)	
Word 9	Pointer 4 (Data-out Location)	
Word 10	Length 5 (NULL)	
Word 11	Pointer 5 (NULL)	
Word 12	Length 6 (NULL)	
Word 13	Pointer 6 (NULL)	
Word 14	Length 7 (NULL)	
Word 15	Pointer 7 (NULL)	
Word 16	Next Descriptor Pointer	

Figure 5-1. Example Data Packet Descriptor

5.2.1 Descriptor Header

Descriptors are created by the host to guide the MPC184 through required cryptographic operations. The descriptor header defines the operations to be performed, mode for each operation, and internal addressing used by the controller and channel for internal data movement. The MPC184 device drivers allow the host to create proper headers for each cryptographic operation. Figure 5-2 shows the descriptor header.

	0	11	12	23	24	27	28	29	30	31	
Field	Op_0			Op_1			DESC_TYPE		RSVD	ST	DN
Reset	0x0000_0000										
R/W	R/W										
Addr	Channel_1 0x02080, Channel_2 0x03080, Channel_3 0x04080, Channel_4 0x05080										

Figure 5-2. Descriptor Header

Table 5-1 defines the header bits.

Table 5-1. Header Bit Definitions

Bits	Name	Description
0:11	Op_0	Op_0 contains two sub fields, EU_Select and Mode_Data. Figure 5-3 shows the sub field detail. EU_SELECT[0:3] - Programs the channel to select a primary EU of a given type. Table 5-2 lists the possible EU_SELECT values. MODE_DATA[4:11] - Programs the primary EU mode data. The mode data is specific to the chosen EU. This data is passed directly to bits 0:7 of the specified EU mode register.
12:23	Op_1	Op_1 contains two sub fields, EU_Select and Mode_Data. Figure 5-3 shows the sub field detail. EU_SELECT[12:15] — Programs the channel to select a secondary EU of a given type. Table 5-2 lists the possible EU_SELECT values. MODE_DATA[16:23] —Programs the secondary EU mode data. The mode data is specific to the chosen EU. This data is passed directly to bits 0:7 of the specified EU mode register. Note: The MDEU is the only valid secondary EU. Values for Op1 EU_SELECT other than 'MDEU' or 'No secondary CHA selected' will result in an 'Unrecognized Header' error condition. Selecting MDEU for both primary and secondary EU will also create an error condition.
24:27	Desc_Type	Descriptor Type —Each type of descriptor determines the following attributes for the corresponding data length/pointer pairs: the direction of the data flow; which EU is associated with the data; and which internal EU address is used. Table 5-3 lists the valid types of descriptors.
28:29	---	Reserved- set to zero
30	ST	Snoop type — Selects which of the two types of available snoop modes applies to the descriptor. 0 Snoop output data mode. 1 Snoop input data mode. In 'Snoop input data mode', while the bus transaction to write data into the input FIFO of the primary EU is in progress, the secondary EU (always MDEU) will snoop the same data into its input FIFO. In 'Snoop output data mode', the secondary EU (always MDEU) will snoop data into its input FIFO during the bus transaction to read data out of the output FIFO of the primary EU.
31	DN	DONE_NOTIFICATION_FLAG — Done Notification Flag. Setting this bit indicates whether to perform notification upon completion of this descriptor. The notification can take the form of an interrupt or modified header write back or both depending upon the state of the INTERRUPT_ENABLE and WRITEBACK_ENABLE control bits in Crypto Channel Configuration Register. 0 Do not signal DONE upon completion of this descriptor (unless globally programmed to do so via the Crypto Channel Configuration Register.) 1 Signal DONE upon completion of this descriptor Note: The MPC184 can be programmed to perform DONE notification upon completion of each descriptor, upon completion of any descriptor, or completion of the final descriptor in a chain. This bit provides for the second case. When the Crypto-Channel is requesting a write of the Descriptor Header back to system memory, the most significant byte (Big Endian) of the header will always read as set to 0xFF, and the remaining 24 bits will not be changed.

Figure 5-3 shows the two sub fields of Op_x.

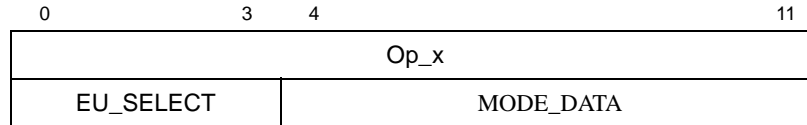


Figure 5-3. Op_x sub fields

Op0 EU_SELECT values of 'no primary EU selected' or 'reserved EU' will result in an 'unrecognized header error' condition during processing of the descriptor header. Also, the primary EU selected by the Op0 EU_SELECT field may only be DEU, AESU or AFEU when a valid secondary EU is selected. For this case, all other values of Op0 EU_SELECT will result in an 'Unrecognized header' error condition. The full range of permissible EU_Select values is shown in Table 5-2.

Table 5-2. EU_Select Values

Value	EU Select:
0000	No EU selected.
0001	AFEU
0010	DEU
0011	MDEU
0100	RNG
0101	PKEU
0110	AESU
Others	Reserved EU

Table 5-3 shows the permissible values for the descriptor type field in the descriptor header.

Table 5-3. Descriptor Types

Value	Descriptor Type	Notes
0000	Reserved	—
0001	common_nonsnoop_no_afeu	Common, nonsnooping, non-PKEU, non-AFEU
0010	hmac_snoop_no_afeu	Snooping, HMAC, non-AFEU
0011	non_hmac_snoop_no_afeu	Snooping, non-HMAC, non-AFEU
0100	aseu_key_expand_output	Non-snooping, non HMAC, AESU, expanded key out
0101	common_nonsnoop_afeu	Common, nonsnooping, AFEU
0110	hmac_snoop_afeu	Snooping, HMAC, AFEU (no context out)
0111	non_hmac_snoop_afeu	Snooping, non-HMAC, AFEU
1000	pkeu_mm	PKEU-MM
1001	pkeu_ec	PKEU-EC
1010	pkeu_static_ec_point	PKEU Static-EC Point (completes operand loading and executes)
1011	pkeu_static_ec_parameter	PKEU Static-EC Parameter (pre loads EC operands)

Table 5-3. Descriptor Types (continued)

1100	Reserved	-
1101	Reserved	-
1110	hmac_snoop_afeu_key_in	AFEU Context Out Available
1111	hmac_snoop_afeu_ctx_in	AFEU Context Out Available

5.2.2 Descriptor Length and Pointer Fields

The length and pointer fields represent one of seven data length/pointer pairs. Each pair defines a block of data in system memory. The length field gives the length of the block in bytes. The maximum allowable number of bytes is 32K bytes. A value of zero loaded into the length field indicates that this length/pointer pair should be skipped and processing continue with the next pair.

The pointer field contains the address, in 8xx address space, of the first byte of the data block. Transfers from the 8xx bus with the pointer address set to zero will have the length value written to the EU, and no data fetched from the 8xx bus.

NOTE

Certain public key operations require information about data length, but not the data itself.

Figure 5-4 shows the descriptor length field.

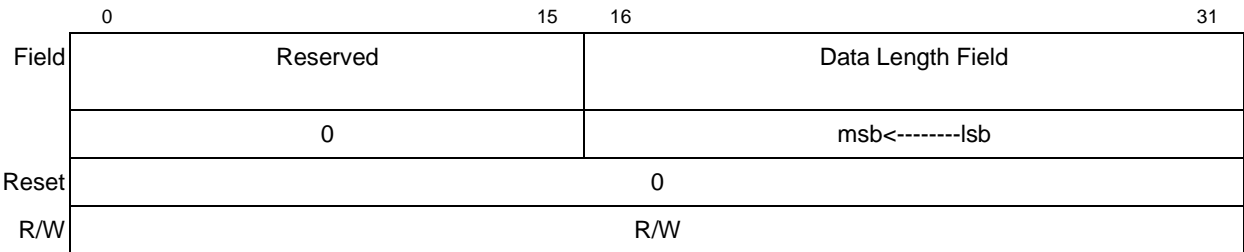


Figure 5-4. Descriptor Length Field

Table 5-4 shows the descriptor length field mapping.

Table 5-4. Descriptor Length Field Mapping

Bits	Name	Reset Value	Description
0:15	--	0	Reserved, set to zero
16:31	Data Field Length	0	The maximum length this field can be set to 32K bytes. Under host control, a channel can be temporarily locked static, and data only" descriptors can be chained to fetch blocks larger than 32K bytes in 32K byte sub-blocks without key/context switching, until the large original block has been completely ciphered. Length fields also indicate the size of items to be written back to memory upon completion of security processing in the MPC184.

Figure 5-5 shows the descriptor pointer field.

Field	0	31
Reset	0	
R/W	R/W	

Figure 5-5. Descriptor Pointer Field

Table 5-5 shows the descriptor pointer field mapping.

Table 5-5. Descriptor Pointer Field Mapping

Bits	Name	Reset Value	Description
0:31	Data Field Pointer	0	<p>The Data Pointer Field contains the address, in 8xx address space, of the first byte of the data packet for either read or write back. Transfers from the 8xx bus with Pointer address set to zero will be skipped.</p> <p>WARNING</p> <p>MPC184-initiated 8xx writes can occur only on 32-bit-word boundaries, but reads can occur on any byte boundary. Writing back a header read from a non-32-bit-word boundary will yield unpredictable results.</p>

Table 5-6 shows how the length/pointer pairs should be used with the various descriptor types to load keys, context, and data into the Execution Units, and how the required outputs should be unloaded.

Table 5-6. Descriptor Length/Pointer Mapping

Descriptor Type	L/P 1	L/P 2	L/P 3	L/P 4	L/P 5	L/P 6	L/P 7
0000	Null	Null	Null	Null	Null	Null	Null
0001	Null	IV	Key	Data In	Data Out	IV Out	MAC Out
0010	HMAC Key	HMAC Data	Key	IV	Data In	Data Out	HMAC/Context Out
0011	MD Ctx In	IV	Key	Data In	Data Out	IV Out	MD/Context Out
0100	Null	IV	Key	Data In	Data Out	IV Out	Key Out via FIFO
0101	Null	IV in via FIFO	Key	Data In	Data Out	IV Out via FIFO	MD/Context Out

Descriptor Type	L/P 1	L/P 2	L/P 3	L/P 4	L/P 5	L/P 6	L/P 7
0110	HMAC Key	HMAC Data	Key	IV in via FIFO	Data In	Data Out	HMAC/Context Out
0111	MD Ctx In	IV in via FIFO	Key	Data In	Data Out	IV Out via FIFO	MD/Context Out
1000	B	A	E	N	B out	Null	Null
1001	B	A	Key	N	B1 out	Null	Null
1010	A0	A1	A2	B1 Out	B2 Out	B3 Out	Null
1011	A3	B0	B1	Key	N	Null	Null
1100	Null	Null	Null	Null	Null	Null	Null
1101	Null	Null	Null	Null	Null	Null	Null
1110	HMAC Key	HMAC Data	Key	Data In	Data Out	IV Out via FIFO	HMAC/Context Out
1111	HMAC Key	HMAC Data	IV	Data In	Data Out	IV Out via FIFO	HMAC/Context Out

5.3 Descriptor Chaining

Following the length/pointer pairs is the ‘Next Descriptor’ field, which contains the pointer to the next descriptor in memory. Upon completion of processing of the current descriptor, this value, if non-zero, is used to request a 8xx burst read of the next-data-packet descriptor. This automatic load of the next descriptor is referred to as descriptor chaining. Figure 5-6 displays the next descriptor pointer field.

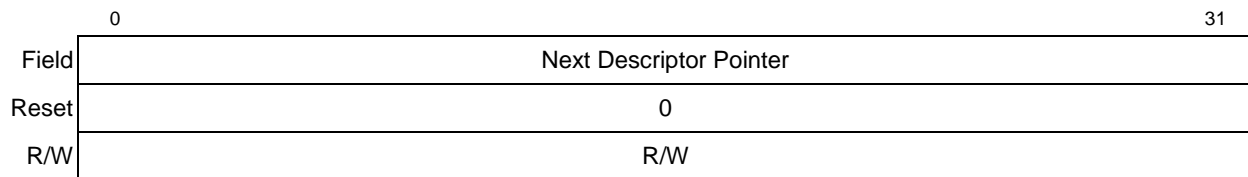


Figure 5-6. Next Descriptor Pointer Field

Table 5-7 describes the descriptor pointer field mapping.

Table 5-7. Descriptor Pointer Field Mapping

Bits	Name	Reset Value	Description
0:31	Next Descriptor Pointer	0	<p>The Next Descriptor Pointer Field contains the address, in 8xx address space, of the next descriptor to be fetched if descriptor chaining is enabled.</p> <p>Warning</p> <p>The Next Descriptor Pointer Address must be modulo-4 aligned if Write back is enabled as the method of DONE notification.</p>

Descriptor chaining provides a measure of ‘de coupling’ between host CPU activities and the status of the MPC184. Rather than waiting for the MPC184 to signal DONE, and arbitrating for the 8xx bus in order to write directly to the next-data-packet descriptor in the crypto-channel, the host can simply create new descriptors in memory, and chain them to

Descriptor Chaining

descriptors which have not yet been fetched by the MPC184 by filling the next-data-packet field with the address of the newly created descriptor. Whether or not processing continues automatically following next-descriptor fetch and whether or not an interrupt is generated depends on the programming of the Crypto-Channel's Configuration Register.

See Section 6.1.1, "Crypto-Channel Configuration Register (CCCR)," in the Crypto-Channels chapter for additional information on how the MPC184 can be programmed to signal and act upon completion of a descriptor.

NOTE

It is possible to insert a descriptor into an existing chain; however, great care must be taken when doing so.

Figure 5-7 shows a conceptual chain, or 'linked list,' of descriptors.

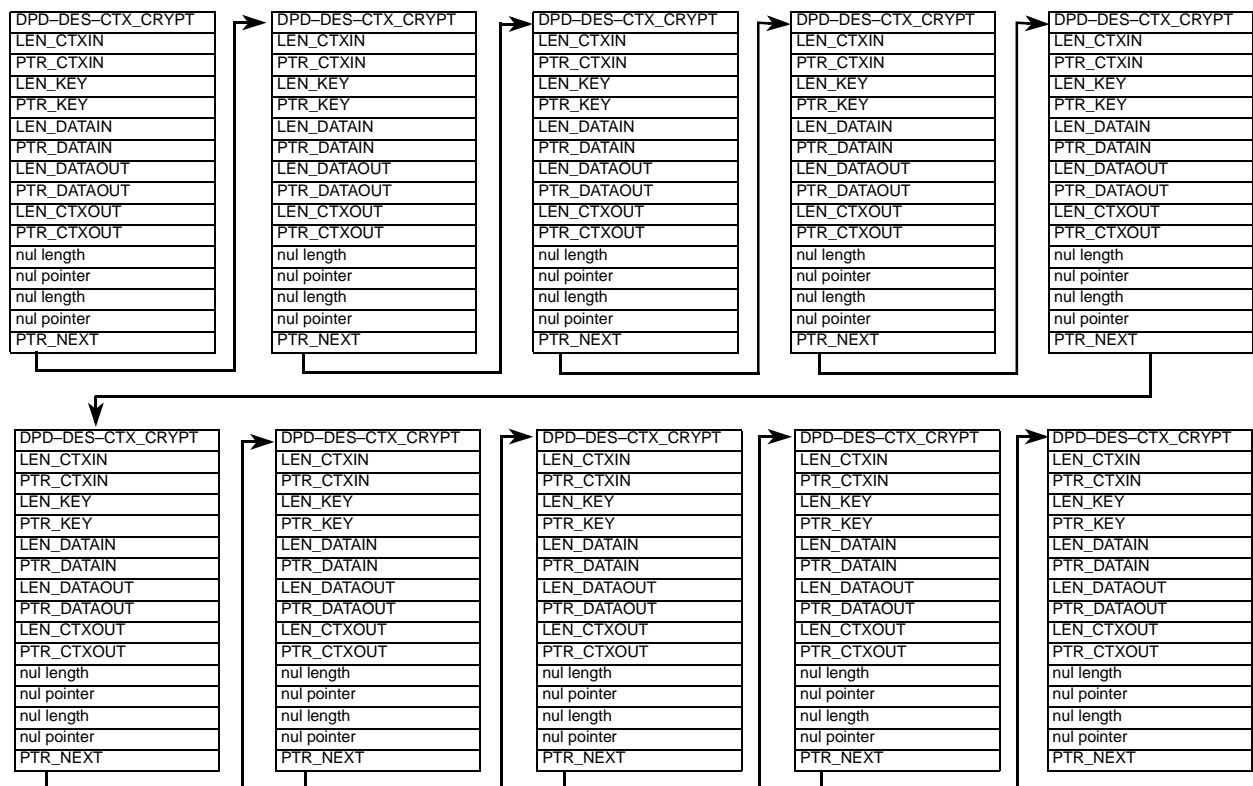


Figure 5-7. Chain of Descriptors

5.3.1 Null Fields

On occasion, a descriptor field may not be applicable to the requested service. With seven length/pointer pairs, it is possible that not all descriptor fields will be required to load the required keys, context, and data. (Some operations don't require context, others may only need to fetch a small, contiguous block of data.) Therefore, when processing data packet descriptors, the MPC184 will skip entirely any pointer that has an associated length of zero.

5.4 Descriptor Classes

The MPC184 has two general classes of descriptors: static, which refers to a relatively unchanging usage of MPC184 resources, and dynamic, which refers to a continually changing usage model.

5.4.1 Static Descriptors

Recall that the MPC184 has 6 execution units and 4 crypto-channels. The EUs can be statically assigned or dedicated to a particular crypto-channel. Certain combinations of EUs can be statically assigned to the same crypto-channel to facilitate multi-operation security processes, such as IPsec ESP mode. When the system traffic model permits its use, static assignment can offer significant performance improvements over dynamic assignment by avoid key and context switching per packet.

Static descriptors split the operations to be performed during a security operation into separate descriptors. The first descriptor is typically only used to set the EU mode, and load the key and context. The second (and multiple subsequent) descriptor contains length/pointer pairs to the data to be permuted. Because the key and context are unchanging over multiple packets (or descriptors), the series of short reads and writes required to set-up and tear down a session are avoided. This savings, along with the crypto-channel having dedicated execution units, represents a significant performance improvement.

For example, statically assigning AFEU to a particular crypto-channel permits AFEU to retain state between data packets. The following descriptors, displayed in Table 5-8 through Table 5-11, support state-retention. Table 5-8 defines the DPD_RC4-SA_NEWCTX descriptor.

Table 5-8. Actual Descriptor DPD_RC4-SA_NEWCTX

Field	Value/ Type	Description
DPD_RC4-SA_NEWCTX ¹	TBD	Packet Command "DPD_RC4-SA_NEWCTX"
LEN_KEY	Length	Number of bytes of key to be written
PTR_KEY	Pointer	Pointer to where the RC4 key is stored
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused

Table 5-8. Actual Descriptor DPD_RC4-SA_NEWCTX (continued)

Field	Value/ Type	Description
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
PTR_NEXT	Pointer	Pointer to Next Descriptor

¹ DPD_RC4-SA_NEWCTX writes the key at PTR_KEY to be written to AFEU and causes the initial context-permutation to occur

Table 5-9 defines the DPD_RC4-SA_LDCTX descriptor.

Table 5-9. Actual Descriptor DPD_RC4-SA_LDCTX

Field	Value / Type	Description
DPD_RC4-SA_LDCTX ¹	TBD	Packet Command "DPD_RC4-SA_LDCTX"
LEN_CTXIN	Length	Number of bytes to be written (should always be 257)
PTR_CTXIN	Pointer	Pointer to context to be written into AFEU
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
PTR_NEXT	Pointer	Pointer to Next Descriptor

¹ DPD_RC4-SA_LDCTX rewrites the permuted context memory from a previously used RC4 context.

Table 5-10 defines the DPD_RC4-SA_CRYPT descriptor.

Table 5-10. Actual Descriptor DPD_RC4-SA_CRYPT

Field	Value / Type	Description
DPD_RC4-SA_CRYPT ¹	TBD	Packet Command "DPD_RC4-SA_CRYPT"
LEN_DATAIN	Length	Number of bytes to be ciphered
PTR_DATAIN	Pointer	Pointer to location containing data to be ciphered

Table 5-10. Actual Descriptor DPD_RC4-SA_CRYPT (continued)

Field	Value / Type	Description
LEN_DATAOUT	Length	Bytes to be written (should be equal to DATAIN_LEN)
PTR_DATAOUT	Pointer	Pointer to location where ciphered data is to be written
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
PTR_NEXT	Pointer	Pointer to Next Descriptor

¹ DPD_RC4-SA_CRYPT performs the cipher function on a block of data, neither loading nor unloading the current context.

Table 5-11 defines the DPD_RC4-SA_CRYPT_ULCTX descriptor.

Table 5-11. Actual Descriptor DPD_RC4-SA_CRYPT_ULCTX

Field	Value / Type	Description
DPD_RC4-SA_CRYPT_ULCTX ¹	TBD	Packet Command “DPD_RC4-SA_CRYPT_ULCTX”
LEN_DATAIN	Length	Number of bytes to be ciphered
PTR_DATAIN	Pointer	Pointer to location containing data to be ciphered
LEN_DATAOUT	Length	Bytes to be written (should be equal to DATAIN_LEN)
PTR_DATAOUT	Pointer	Pointer to location where ciphered data is to be written
LEN_CTXOUT	Length	Length of AFEU context written (should be 256 bytes)
PTR_CTXOUT	Pointer	Location where AFEU context is to be written
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused

Table 5-11. Actual Descriptor DPD_RC4-SA_CRYPT_ULCTX

Field	Value / Type	Description
Nul pointer	Nul	Unused
PTR_NEXT	Pointer	Pointer to Next Descriptor

¹ DPD_RC4-SA_CRYPT_ULCTX unloads the context from the AFEU into system memory. For ArcFour cryptographic computations, architectural implementation details prevent a stand alone unload-context descriptor. Context unload must always follow ciphering.

5.4.2 Dynamic Descriptors

In a typical networking environment, packets from innumerable sessions can arrive randomly. The host must determine which security association applies to the current packet and encrypt or decrypt without any knowledge of the security association of the previous or next packet. This situation calls for the use of dynamic descriptors.

When under dynamic assignment, an EU must be used under the assumption that a different crypto-channel (with a different context) may have just used the EU and that another crypto-channel (with yet another context) may use that EU immediately after the current crypto-channel has released the EU. Therefore, for dynamic-assignment use, there is a set of data packet descriptors defined that sets up the appropriate context, performs the cipher function, and then saves the context to system memory.

The descriptor shown in Table 5-12 completely sets up the DEU for an encryption operation; loads the keys, context, and data; writes the permuted data back to memory; and writes the altered context (IV) back to memory. (This may be necessary when DES is operating in CBC mode.) Upon completion of the descriptor, the DEU is cleared and released.

Table 5-12. Representative Descriptor DPD_DEU_CTX_CRYPT

Field	Value / Type	Description
DPD_DEU_CTX_CRYPT ¹	TBD	Representative header "DPD_DEU_CTX_CRYPT"
LEN_CTXIN	Length	Number of bytes to be written (0 or 8 bytes)
PTR_CTXIN	Pointer	Pointer to context (IV) to be written into DEU
LEN_KEY	Length	Number of bytes in key (8, 16, or 24 bytes)
PTR_KEY	Pointer	Pointer to block cipher key
LEN_DATAIN	Length	Number of bytes of data to be ciphered (multiples of 8 bytes)
PTR_DATAIN	Pointer	Pointer to data to perform cipher upon
LEN_DATAOUT	Length	Number of bytes of data after ciphering
PTR_DATAOUT	Pointer	Pointer to location where cipher output is to be written
LEN_CTXOUT	Length	Length of output context (IV) (0 or 8 bytes)
PTR_CTXOUT	Pointer	Location where DEU context is to be written



Table 5-12. Representative Descriptor DPD_DEU_CTX_CRYPT

Field	Value / Type	Description
Nul length	Nul	Unused
Nul pointer	Nul	Unused
Nul length	Nul	Unused
Nul pointer	Nul	Unused
PTR_NEXT	Pointer	Pointer to Next Descriptor

¹ DPD_DEU_CTX_CRYPT represents Descriptors that write the key at PTR_KEY to DEU, writes the IV located at PTR_CTXIN to DEU, performs the cipher on data fetched from PTR_DATAIN, and writes the result to memory at PTR_DATAOUT.



Chapter 6

Crypto-Channels

A crypto-channel manages data associated with the one of more execution units (EUs) on the MPC184. Control and data information for a given task is stored in the form of 16 32-bit word descriptors in system memory or in the crypto-channel itself. The descriptor describes how the EU should be initialized, where to fetch the data to be ciphered and where to store the ciphered data the EU outputs. Through a series of requests to the controller, the crypto-channel decodes the contents of the descriptors to perform the following functions:

- Request assignment of one or more of the several EUs on the MPC184 for the exclusive use of the channel.
- Automatically initialize mode registers in the assigned EU upon notification of completion of the EU reset sequence.
- Transfer data packets (up to 32K bytes) from system memory (8xx Read) into assigned EU input registers and FIFOs (EU Write).
- Transfer data packets (up to 32K bytes) from assigned EU output registers and FIFOs (EU Read) to system memory space (8xx Write).
- Automatically initialize the key size register in the assigned EU after requesting a write to EU key address space.
- Automatically initialize data size register in the assigned EU before requesting a write to EU FIFO address space.
- Automatically initialize the EU_GO register (where applicable) in the assigned EU upon completion of last EU write indicated by the descriptor. The channel will wait for a indication from the EU that processing of input data is complete before proceeding with further activity after writing EU_GO. Request assignment of the MDEU when the descriptor header calls for multi-operation processing. The MDEU will be configured to snoop input or output data intended for the primary assigned EU.
- Reset assigned EU(s).
- Release assigned EU(s).
- Automatically fetch the next descriptor from system memory and start processing, when chaining is enabled. Descriptor chains can be of unlimited size

- Provide feedback to host, via interrupt, when a descriptor, or a chain of descriptors, has been completely processed.
- Provide feedback to host, via modified descriptor header write back to system memory, when a descriptor, or a chain of descriptors, has been completely processed.
- Provide feedback to host, via interrupt, when descriptor processing is halted due to an error.
- Detect static assignment of EU(s) by the controller and alter descriptor processing flow to skip EU Request and EU Release steps. The channel will also automatically reset the EU_DONE interrupt after receiving indication that processing of input data has been completed by the EU.

The channel will wait indefinitely for the controller to complete a requested activity before continuing to process a descriptor.

6.1 Crypto-Channel Registers

Each crypto-channel contains the following registers:

- Crypto-Channel Configuration Register (CCCR)
- Crypto-Channel Pointer Status Register (CCPSR)
- Current Descriptor Pointer Register (CDPR)
- Fetch Register (FR)
- Descriptor Buffer Register (DBR)

6.1.1 Crypto-Channel Configuration Register (CCCR)

This register contains five operational bits permitting configuration of the crypto-channel as shown in Figure 6-1. Table 6-1 describes the CCCR.

	0																												31		
Field	Reserved																														
Reset	0																														
R/W	R/W																														
Addr	Channel_1 0x02008, Channel_2 0x03008, Channel_3 0x04008, Channel_4 0x05008																														

	0					20	21	23	24					26	27	28	29	30	31												
Field	Reserved					Burst Size		Reserved				WE		NE	NT	CDIE	R														
Reset	0																														
R/W	R/W																														
Addr	Channel_1 0x0200C, Channel_2 0x0300C, Channel_3 0x0400C, Channel_4 0x0500C																														

Figure 6-1. Crypto-Channel Configuration Register

Table 6-1 describes the CCCR signals.

Table 6-1. Crypto-Channel Configuration Register Signals

Bits	Name	Reset Value	Description
0:20	Reserved	0	Reserved, set to zero
21:23	Burst Size	0	The MPC184 implements flow control to allow larger than FIFO sized blocks of data to be processed with a single key/IV. The channel programs the various execution units to advise on space or data available in the FIFO via this field. The size of the burst is given in Table 6-2
24:26	Reserved	0	Reserved, set to zero
27	WRITEBACK_ENABLE	0	<p>Writeback_Enable. This bit determines if the crypto-channel is allowed to notify the host of the completion of descriptor processing by setting (writing back) a DONE bit in the descriptor header. This enables the host to poll the memory location of the original descriptor header to determine if that descriptor has been completed.</p> <p>0 Descriptor header writeback notification is disabled. 1 Descriptor header writeback notification is enabled.</p> <p>Header write back notification will occur at the end of every descriptor if NOTIFICATION_TYPE is set to end-of-descriptor and Writeback_Enable is set. Write back will occur only after the last descriptor in the chain (Next Descriptor Pointer is NIL) if NOTIFICATION_TYPE is set to end-of-chain.</p> <p>WARNING</p> <p>The MPC184 is capable ONLY of performing initiator write cycles to 32-bit-word aligned addresses. Enabling header write back when the MPC184 fetches a descriptor from a non-aligned location will yield unpredictable results.</p>
28	NEXT_ENABLE	0	<p>Fetch Next Descriptor Enable. This bit determines if the crypto-channel is allowed to request a transfer of the next descriptor, in a multi-descriptor chain, into its descriptor buffer.</p> <p>0 Disable fetching of next descriptor when crypto-channel has finished processing the current one. 1 Enable fetching of next descriptor when crypto-channel has finished processing the current one.</p> <p>The address of the next descriptor in a multi-descriptor chain is either the contents of the next descriptor pointer in the descriptor buffer or the contents of the fetch register. Only if both of these registers are NIL upon completion of the descriptor currently being processed will that descriptor be considered the end of the chain.</p>
29	NOTIFICATION_TYPE	0	<p>Channel DONE Notification Type. This bit controls when the crypto-channel will generate Channel DONE Notification.</p> <p>0 End-of-chain: The crypto-channel will generate channel done notification (if enabled) when it completes the processing of the last descriptor in a descriptor chain. The last descriptor is identified by having NIL loaded into both the next descriptor pointer in the descriptor buffer and the fetch register. 1 End-of-descriptor: The crypto-channel will generate channel done notification (if enabled) at the end of every data descriptor it processes</p> <p>Channel DONE notification can take the form of an interrupt or modified header writeback or both, depending on the state of the INTERRUPT_ENABLE and WRITEBACK_ENABLE control bits.</p>

Table 6-1. Crypto-Channel Configuration Register Signals (continued)

Bits	Name	Reset Value	Description
30	CDIE	0	Channel DONE Interrupt Enable. This bit determines whether or not the crypto-channel is allowed to assert interrupts to notify the host that the channel has completed descriptor processing. 0 Channel Done interrupt disabled 1 Channel Done interrupt enabled When CDIE is set, the NOTIFICATION_TYPE control bit determines when the CHANNEL_DONE interrupt is asserted. Channel error interrupts are asserted as soon as the error is detected. Refer to Section 6.2, "Interrupts," for complete description of crypto-channel interrupt operation.
31	RESET	0	Reset Crypto-Channel. This bit allows the crypto-channel to be software reset. 0 Automatically cleared by the crypto-channel when reset sequence is complete. Refer to Section 6.2.3, "Channel Reset," for complete description of crypto-channel reset operation. 1 Reset the registers and internal state of the crypto-channel, any EU assigned to the crypto-channel and the controller state associated with the crypto-channel.

Table 6-2 defines the burst size according to the value displayed in bits 21 through 23.

Table 6-2. Burst Size Definition

Value	Number of dwords in burst
000	1
001	4
010	8
011	12
100	16
101	20
110	24
111	32

6.1.2 Crypto-Channel Pointer Status Registers (CCPSR)

These registers contains status fields and counters which provide the user with status information regarding the channel's actual processing of a given descriptor. Figure 6-2 shows the bit positions of Crypto-Channel Pointer Status Register 1.

	0	23	24	31
Field	Reserved			State
Reset	0			
R/W	R			
Addr	Channel_1 0x02010, Channel_2 0x03010, Channel_3 0x04010, Channel_4 0x05010			

Figure 6-2. Crypto-Channel Pointer Status Register 1

Table 6-3 describes the Crypto-Channel Pointer Status Register fields.

Table 6-3. Crypto-Channel Pointer Status Register 1 Signals

Bits	Name	Reset Value	Description
0:23	Reserved	0	Reserved, set to zero
24:31	STATE	0	State of the crypto-channel state machine. This field reflects the state of the crypto-channel control state machine. The value of this field indicates exactly which stage the crypto-channel is in the sequence of fetching and processing data descriptors. Table 6-5 shows the meaning of all possible values of the STATE field. Note: State is documented for information only. The User will not typically care about the crypto-channel state machine.

Figure 6-3 shows the bit positions of Crypto-Channel Pointer Status Register 2.

	0	4	5	6	7	8	9	10	11	12	13	14	15	16	23	24	31
Field	Reserved	Stat	MI	MO	PR	SR	PG	SG	PRD	SRD	PD	SD	ERROR		PAIR_PTR		
Reset	0																
R/W	R																
Addr	Channel_1 0x02014, Channel_2 0x03014, Channel_3 0x04014, Channel_4 0x05014																

Figure 6-3. Crypto-Channel Pointer Status Register 2

Table 6-4 describes the Crypto-Channel Pointer Status Register fields.

Table 6-4. Crypto-Channel Pointer Status Register 2 Signals

Bits	Name	Reset Value	Description
0:4	Reserved	0	Reserved, set to zero
5	Static	0	Crypto-Channel Static Mode Enable. 0 Crypto-channel is operating in dynamic mode. 1 Crypto-channel is operating in static mode. The STATIC bit is set when descriptor processing is initiated and the EUs indicated in the descriptor header register are already assigned to the channel. This bit is cleared when descriptor processing is initiated for the next descriptor and no EUs are assigned to the channel.
6	Multi_EU_IN	0	If enabled, the secondary assigned EU will receive the same data as the primary assigned EU. 0 Data input snooping by secondary EU disabled. 1 Data input snooping by secondary EU enabled.
7	Multi_EU_OUT	0	If enabled, the secondary assigned EU will received data generated as output by the primary assigned EU. 0 Data output snooping by secondary EU disabled. 1 Data output snooping by secondary EU enabled.
8	PRI_REQ	0	Request primary EU assignment. 0 Primary EU Assignment Request is inactive. 1 The crypto-channel is requesting assignment of primary EU to the channel. The channel will assert the EU request signal indicated by the op0 field in the Descriptor Header register as long as this bit remains set. The PRI_REQ bit is set when descriptor processing is initiated in dynamic mode and the Op_0 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the PRI_GRANT bit.
9	SEC_REQ	0	Request secondary EU assignment. 0 Secondary EU Assignment Request is inactive. 1 The crypto-channel is requesting assignment of secondary EU to the channel. The channel will assert the EU request signal indicated by the Op_1 field in the descriptor header register as long as this bit remains set. The SEC_REQ bit is set when descriptor processing is initiated in dynamic mode and the Op_1 field in the descriptor header contains a valid EU identifier. This bit is cleared when the request is granted, which will be reflected in the status register by the setting the SEC_GRANT bit.
10	PRI_GRANT	0	Primary EU granted. The PRI_GRANT bit reflects the state of the EU grant signal for the requested primary EU from the controller. 0 The primary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested primary EU to the channel.
11	SEC_GRANT	0	Secondary EU granted. The SEC_GRANT bit reflects the state of the EU grant signal for the requested secondary EU from the controller. 0 The secondary EU grant signal is inactive. 1 The EU grant signal is active indicating the controller has assigned the requested secondary EU to the channel.

Table 6-4. Crypto-Channel Pointer Status Register 2 Signals (continued)

Bits	Name	Reset Value	Description
12	PRI_RESET_DONE	0	Primary EU reset done. The PRI_RST_DONE bit reflects the state of the reset done signal from the assigned primary EU. 0 The assigned primary EU reset done signal is inactive. 1 The assigned primary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
13	SEC_RESET_DONE	0	Secondary EU reset done. The SEC_RST_DONE bit reflects the state of the reset done signal from the assigned secondary EU. 0 The assigned secondary EU reset done signal is inactive. 1 The assigned secondary EU reset done signal is active indicating its reset sequence has completed and it is ready to accept data.
14	PRI_DONE	0	Primary EU done. The PRI_DONE bit reflects the state of the done interrupt from the assigned primary EU. 0 The assigned primary EU done interrupt is inactive. 1 The assigned primary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.
15	SEC_DONE	0	Secondary EU done. The SEC_DONE bit reflects the state of the done interrupt from the assigned secondary EU. 0 The assigned secondary EU done interrupt is inactive. 1 The assigned secondary EU done interrupt is active indicating the EU has completed processing and is ready to provide output data.
16:23	ERROR	0	Crypto-channel error status. This field reflects the error status of the crypto-channel. When a channel error interrupt is generated, this field will reflect the source of the error. The bits in the ERROR field are registered at specific stages in the descriptor processing flow. Once registered, an error can only be cleared only by resetting the crypto-channel or writing the appropriate registers to initiate the processing of a new descriptor. Table 6-6 lists the conditions which can cause a crypto-channel error and how they are represented in the ERROR field.
24:31	PAIR_PTR	7	Descriptor buffer register length/pointer pair. This field indicates which of the length/pointer pairs are currently being processed by the channel. Table 6-7 shows the meaning of all possible values of the PAIR_PTR field.

Table 6-5 shows the values of crypto-channel states.

Table 6-5. STATE Field Values

Value	Crypto-Channel State
0x00	Idle
0x01	Process_header
0x02	Fetch_descriptor
0x03	Channel_done
0x04	Channel_done_irq
0x05	Channel_done_writeback
0x06	Channel_done_notification

Table 6-5. STATE Field Values (continued)

Value	Crypto-Channel State
0x07	Channel_error
0x08	Request_pri_eu
0x09	Inc_data_pair_pointer
0x0A	Delay_data_pair_update
0x0B	Evaluate_data_pairs
0x0C	Write_reset_pri
0x0D	Release_pri_eu
0x0E	Write_reset_sec
0x0F	Release_sec_eu
0x10	Process_data_pairs
0x11	Write_mode_pri
0x12	Write_mode_sec
0x13	Write_datasize_pri
0x14	Delay_rng_done
0x15	Write_datasize_sec_multi_eu_in
0x16	Trans_request_read_multi_eu_in
0x17	Delay_pri_sec_done
0x18	Trans_request_read
0x19	Write_key_size
0x1A	Write_eu_go
0x1B	Delay_pri_done
0x1C	Write_reset_irq_pri
0x1D	Write_reset_irq_sec
0x1E	Write_datasize_sec_snoopout
0x1F	Trans_request_write_snoopout
0x20	Delay_sec_done
0x21	Trans_request_write
0x22	Evaluate_reset
0x23	Reset_write_reset_pri
0x24	Reset_release_pri_eu
0x25	Reset_write_reset_sec
0x26	Reset_release_sec_eu
0x27	Reset_channel
0x28	Write_datasize_pri_post

Table 6-5. STATE Field Values (continued)

Value	Crypto-Channel State
0x29	Reset_release_all
0x2A	Reset_release_all_delay
0x2B	Request_sec_eu
0x2C	Write_datasize_sec
0x2D	Write_pri_eu_go_multi_eu_out
0x2E	Write_sec_eu_go_multi_eu_out
0x2F	Write_pri_eu_go_multi_eu_in
0x30	Write_sec_eu_go_multi_eu_in
0x31	Write_datasize_pri_delay
0x32- 0xFF	Reserved

Table 6-6 shows the bit positions of each potential error. Multiple errors are possible.

Table 6-6. Crypto-Channel Pointer Status Register Error Field Definitions

Value	Error
b00000000	No error detected
bxxxxxx1	EU error detected. An EU assigned to this channel has generated an error interrupt. This error may also be reflected in the controller's interrupt status register.
bxxxxxx1x	Static assignment error. Either the channel is statically assigned, but not to an EU requested by the descriptor, or the dynamic assignment request is unfillable because all suitable EUs are otherwise statically assigned.
bxxxxx1xx	Illegal descriptor header
bxxxx1xxx	Parity error. A parity error was detected on the 8xx bus by the controller on behalf of this channel.
bxxx1xxx	Pointer not complete. Caused by an invalid write to the next descriptor register in the descriptor buffer, or to the fetch register.
bxx1xxxx	TEA- A transfer error acknowledge was received from the 8xx bus interface. When the MPC184, while acting as a 8xx bus master, detects a TEA, the controller passes the TEA error to the channel in use. The channel halts and outputs an interrupt. The channel can only be restarted by resetting the channel or the whole MPC184.
bx1xxxxx	Reserved
b1xxxxxx	Reserved

NOTE

EU error bit (ERROR[0]) can only be cleared by first clearing the error source in the assigned EU which caused it to be set.

Table 6-7 shows the possible values of the PAIR_PTR field in the CCPSR.

Table 6-7. Crypto-Channel Pointer Status Register PAIR_PTR Field Values

Value	Error
0x01	Processing Header/Length/Pointer Pair 1
0x02	Processing Length/Pointer Pair 2
0x03	Processing Length/Pointer Pair 3
0x04	Processing Length/Pointer Pair 4
0x05	Processing Length/Pointer Pair 5
0x06	Processing Length/Pointer Pair 6
0x07	Complete (or not yet begun) processing of Header and Length/Pointer pairs
0x08-FF	Reserved

6.1.3 Crypto-Channel Current Descriptor Pointer Register (CDPR)

The CDPR, shown in Figure 6-4, contains the address of the descriptor which the crypto-channel is currently processing. This register, along with the PAIR_PTR in the CCPSR, can be used to determine if a new descriptor can be safely inserted into a chain of descriptors.

	0	31
Field	Reserved	
Reset	0x0000_0000	
R/W	R	
Addr	Channel_1 0x02040, Channel_2 0x03040, Channel_3 0x04040, Channel_4 0x05040	

	0	31
Field	Current Descriptor Pointer Address	
Reset	0x0000_0000	
R/W	R	
Addr	Channel_1 0x02044, Channel_2 0x03044, Channel_3 0x04044, Channel_4 0x05044	

Figure 6-4. Crypto-Channel Current Descriptor Pointer Register

The bits in the current descriptor pointer register perform the functions described in Table 6-8.

Table 6-8. Crypto-Channel Current Descriptor Pointer Register Signals

Bits	Name	Reset Value	Description
0:31	CUR_DES_PTR_ADRS	0	<p>Pointer to system memory location of the current descriptor. This field reflects the starting location in system memory of the descriptor currently loaded into the DB. This value is updated whenever the crypto-channel requests a fetch of a descriptor from the controller. Either the value of the fetch register or of word 16 of the DB is transferred to the current descriptor pointer register immediately after the fetch is completed.</p> <p>This address will be used as destination of the write back of the modified header word, if header write back notification is enabled. If a descriptor is written directly into the descriptor buffer, the host is responsible for writing a meaningful pointer value into the CURRENT_DESCRIPTOR_POINTER field.</p>

6.1.4 Fetch Register (FR)

The FR, displayed in Figure 6-5, contains the address of the first byte of a descriptor to be processed. In typical operation, the host CPU will create a descriptor in memory containing all relevant mode and location information for the MPC184, and then “launch” the MPC184 by writing the address of the descriptor to the fetch register.

Writes to the FR, while the channel is already processing a different descriptor, will be registered and held pending until the channel finishes processing the current descriptor or chain of descriptors. When the end of the current descriptor or chain of descriptors is reached, the descriptor pointed to by the FR will be treated as the next descriptor in a multi-descriptor chain. In this case, the FR must be written to before the channel begins end of descriptor notification. If the register is written after notification has begun, the descriptor will not be considered part of the current chain and will be fetched as a new stand alone descriptor or start of chain after the notification process has completed.

In summary, a channel is initiated by a direct write to the FR, and the channel always checks the FR before determining if it has truly reached the end of a chain.

NOTE

End of descriptor notification consists of modified header writeback or channel DONE interrupt. The fetch address must be modulo-4 aligned if writeback is enabled as the method of DONE notification.

	0	31
Field	Reserved	
Reset	0x0000_0000	
R/W	R/W	
Addr	Channel_1 0x02048, Channel_2 0x03048, Channel_3 0x04048, Channel_4 0x05048	

	0	31
Field	Fetch Address	
Reset	0x0000_0000	
R/W	R/W	
Addr	Channel_1 0x0204C, Channel_2 0x0304C, Channel_3 0x0404C, Channel_4 0x0504C	

Figure 6-5. Fetch Register

Table 6-9 describes the fetch register signals.

Table 6-9. Fetch Register Signals

Bits	Name	Reset Value	Description
0:31	FETCH ADRS	0x0000_0000	Pointer to system memory location of a descriptor the host wants the MPC184 to fetch.

6.1.5 Descriptor Buffer (DB)

The descriptor buffer (DB) actually consists of 32 word aligned registers, and contains the current descriptor being processed by the crypto-channel. This field is R/W enabled, however in typical operation, the DB is filled by a write from the MPC184 Eos controller, acting as an initiator on the PCI bus. (In host controlled mode, the host processor can write the entire descriptor to the DB rather than creating the descriptor in memory.)

The first word of the DB contains the header of the descriptor under processing. The DB uses information in the descriptor header to request and program other on-chip resources in order to complete the required security operation.

Words 2–15 contain fields for data length/data pointer pairs. Each pair consists of a length register, which specifies the size if the data in bytes, and a pointer register which specifies the address of the first byte of the data in system memory space.

Word 16 contains an extra register referred to as the “Next Descriptor Pointer” register, which contains a pointer to the ‘next descriptor’ to be processed, if any. The pointer is set to zero for a single descriptor or the end of a multi-descriptor chain. A descriptor is considered DONE only when the contents of word 16 have been processed by the channel. Additional information on the descriptor format and field values can be found in 184.

	0	31
Word 1	Descriptor Header	
Word 2	Length 1 (Key Length)	
Word 3	Pointer 1 (Key Location)	
Word 4	Length 2 (IV Length)	
Word 5	Pointer 2 (IV Location)	
Word 6	Length 3 (Data-in Length)	
Word 7	Pointer 3 (Data-in Location)	
Word 8	Length 4 (Data-Out Length)	
Word 9	Pointer 4 (Data-out Location)	
Word 10	Length 5 (NULL)	
Word 11	Pointer 5 (NULL)	
Word 12	Length 6 (NULL)	
Word 13	Pointer 6 (NULL)	
Word 14	Length 7 (NULL)	
Word 15	Pointer 7 (NULL)	
Word 16	Next Descriptor Pointer	
Address	Channel_1 0x02080, Channel_2 0x03080, Channel_3 0x04080, Channel_4 0x05080	

Figure 6-6. Data Packet Descriptor Buffer

6.1.5.1 Descriptor Header

Descriptors are created by the host to guide the MPC184 through required crypto-graphic operations. The descriptor header defines the operations to be performed, mode for each operation, and internal addressing used by the controller and channel for internal data movement. The MPC184 device drivers allow the host to create proper headers for each crypto-graphic operation. See Chapter 5, “MPC184 Descriptors” for a full description of the descriptor header.

6.1.5.2 Descriptor Length/Pointer Pairs

The length and pointer fields represent one of seven data length/pointer pairs. Each pair defines a block of data in system memory. The length field gives the length of the block in bytes. The maximum allowable number of bytes is 32K bytes. A value of zero loaded into the length field indicates that this length/pointer pair should be skipped and processing continue with the next pair.

The pointer field contains the address, in 8xx address space, of the first byte of the data block. Transfers from the 8xx bus with the pointer address set to zero will have the length value written to the EU, and no data fetched from the 8xx bus.

6.1.5.3 Next Descriptor Pointer

Following the length/pointer pairs is the ‘Next Descriptor’ field, which contains the pointer to the next descriptor in memory. Upon completion of processing of the current descriptor, this value, if non-zero, is used to request a 8xx burst read of the next-data-packet descriptor. This automatic load of the next descriptor is referred to as descriptor chaining. Chapter 5, “Descriptors” contains a full description of the next descriptor pointer.

NOTE

The next descriptor pointer address must be modulo-4 aligned if write back is enabled as the method of DONE notification.

6.2 Interrupts

The crypto-channel can assert both DONE and ERROR interrupts to the controller. When the interrupt generation conditions have been met, the crypto-channel will assert the appropriate interrupt. The status of the registered crypto-channel interrupts are available in the controller interrupt status register. The registered interrupts can be cleared by writing to the controller interrupt clear register. The crypto-channel does not have an internal interrupt mask bit and interrupts are always asserted to the controller. The controller can be programmed to mask channel interrupts to the host via its interrupt mask register (IMR). See Section 7.1.3, “Interrupt Mask Registers (IMR).”

6.2.1 Channel Done Interrupt

The Channel DONE Interrupt is generated when the crypto-channel has completed processing of a single descriptor or the end of a Chain of descriptors and the Channel DONE Interrupt Enable bit in the CCCR (see Figure 6-1 on page 6-2) is set. Which one of these conditions is responsible for the interrupt depends upon the state of the NOTIFICATION_TYPE bit in the control register, or the DONE_NOTIFICATION_FLAG in the descriptor header.

6.2.2 Channel Error Interrupt

The Channel Error Interrupt is generated when an error condition occurs during descriptor processing. The channel error interrupt will be asserted as soon as the error condition is detected. The type of error condition is reflected in the ERROR field of the Crypto-Channel Pointer Status Register (CCPSR). Refer to Table 6-6 for a complete listing of error types.

6.2.3 Channel Reset

There are two ways to reset the crypto-channel:

- Asynchronous hardware reset
- Software reset

The implications of the two reset methods are described in the following sections:

6.2.3.1 Hardware Reset

The RESET# pin clears all MPC184 registers, including those in the channels, and initializes them to their reset values. Writing the software reset bit in the master control register (Section 7.1.7, “Master Control Register (MCR)”) has the same effect on the crypto-channels as a hardware reset.

6.2.3.2 Channel Specific Software Reset

Software reset is asserted when the host sets the RESET bit in the Crypto-Channel Configuration Register (CCCR). The effect of software reset on the channel varies according to what the channel is doing when the bit is set:

- If the RESET bit is set while the crypto-channel is requesting a EU assignment from the controller, the crypto-channel will cancel its request by asserting the release output signals. The crypto-channel will then reset all the registers, clear the RESET bit and return the control state machine to the idle state.
- If the RESET bit is set after the crypto-channel has been dynamically assigned a EU, the channel will request a write from the controller to set the software reset bit of the EU. A write to reset the secondary (MDEU) EU will also be requested if one has been reserved for snooping. The crypto-channel will then assert the appropriate release output signal to notify the controller that the channel has finished with the reserved EU(s). The crypto-channel will then reset all the registers, clear the RESET bit and return the control state machine to the idle state.
- Setting the RESET bit in the control register while channel is statically assigned to a EU will not cause the channel to reset the assigned EU. It is the host's responsibility to reset the assigned EU in this case.

NOTE

The CCCR and the descriptor buffer registers remain unchanged after software reset.



.interrupts

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Chapter 7 Controller

The controller of the MPC184 is responsible for overseeing the operations of the execution units (EUs), the interface to the host processor, and the management of the crypto-channels. The controller interfaces to the host via the 8xx bus interface and to the channels and EUs via internal buses. All transfers between the host and the EUs are moderated by the controller. Some of the main functions of the controller are as follows:

- Arbitrate and control accesses to the 8xx bus
- Control the internal bus accesses to the EUs
- Arbitrate and assign EUs to the crypto-channels
- Monitor interrupts from channels and pass to host
- Realign initiator read data to dword boundary

7.1 Controller Registers

The Controller contains the following registers, which are described in detail in the following sections.

- EU Assignment Control Register
- EU Assignment Status Register
- Interrupt Mask Register
- Interrupt Status Register
- Interrupt Clear Register
- ID Register
- Master Control Register
- Master Transfer Error Acknowledge (TEA) Address Register

7.1.1 EU Assignment Control Registers (EUACR)

These registers, shown in Figure 7-1, is used to make a static assignment of a EU to a particular crypto-channel. When assigned in this fashion, the EU is inaccessible to any other crypto-channel.

	0	3	4	7	8	11	12	15	16	19	20	23	24	27	28	31
Field	Reserved	RNG			Reserved	PKEU			Reserved	MDEU			Reserved	AFEU		
Reset	0xF	0x0			0xF	0x0			0xF	0x0			0xF	0x0		
R/W	R/W															
Addr	0x 01000															

	0	3	4	7	8	11	12	15	16																				31		
Field	Reserved	DEU			Reserved	AESU			Reserved																						
Reset	0xF	0x0			0xF	0x0			0xFF00																						
R/W	R/W																														
Addr	0x 01004																														

Figure 7-1. EU Assignment Control Registers

Table 7-1. Channel Assignment Value

Value	Channel
0x0	No channel assigned
0x1	Channel 1
0x2	Channel 2
0x3	Channel 3
0x4	Channel 4
0x5-0xE	Undefined
0xF	Unavailable

NOTE

Writing any of the defined values shown in Table 7-1 to any of the fields in the EU assignment control register statically assigns the EU to that specific channel. To release, the host must write 0x0 to the specified EU field of the EUACR.

7.1.2 EU Assignment Status Register (EUASR)

This EUASR, displayed in Figure 7-2, is used to check the assignment status (static or dynamic) of an EU to a particular crypto-channel. When an EU is already assigned, it is inaccessible to any other crypto-channel.

A four-bit field (see Table 7-1) indicates the channel to which an EU is assigned, whether statically or dynamically.

	0	3	4	7	8	11	12	15	16	19	20	23	24	27	28	31						
Field	Reserved	RNG			Reserved			PKEU			Reserved			MDEU			Reserved			AFEU		
Reset	0xF	0x0			0xF			0x0			0xF			0x0			0xF			0x0		
R/W	R/W																					
Addr	0x 01028																					

	0	3	4	7	8	11	12	15	16																	31
Field	Reserved	DEU			Reserved			AESU			Reserved															
Reset	0xF	0x0			0xF			0x0			0xFF00															
R/W	R/W																									
Addr	0x 0102C																									

Figure 7-2. EU Assignment Status Register

7.1.3 Interrupt Mask Registers (IMR)

The MPC184 controller generates the single interrupt output from all possible interrupt sources. These sources can be masked by the Interrupt Mask Registers. If unmasked, the interrupt source value, when active, is captured into the interrupt status registers. Figure 7-3 shows the bit positions of each potential interrupt source. Each interrupt source is individually masked by setting it's corresponding bit.

A complete definition of the bits that can be masked by Interrupt Mask Register 1 is shown in Figure 7-3.

	0	1	2	3	4	5						11	12	13	14	15
Field	CHA_2		CHA_1		A-Err	Reserved						CHA_4		CHA_3		
Definition	Err	Dn	Err	Dn								Err	Dn	Err	Dn	
Reset	0x0000															
R/W	R/W															
Addr	0x 01008															

	16															31
Field	Reserved															
Definition																
Reset	0x0000															
R/W	R/W															
Addr	0x 01008															

Figure 7-3. Interrupt Mask Register 1

A complete definition of the bits that can be masked by Interrupt Mask Register 2 is shown in Figure 7-4.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved		PKEU		Reserved		RNG		Reserved		AFEU		Reserved		MDEU	
Definition			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	R/W															
Addr	0x 0100C															

	16	17	18	19	20	21	22	23	24	25	26	27	28		31	
Field	Reserved		AESU		Reserved		DEU		GIE	TEA	DPE	APE	Reserved			
Definition			Err	Dn			Err	Dn								
Reset	0x0000															
R/W	R/W															
Addr	0x 0100C															

Figure 7-4. Interrupt Mask Register 2

7.1.4 Interrupt Status Registers

The ISR contains fields representing all possible sources of interrupts. The Interrupt Status Registers are cleared either by a reset, or by writing the appropriate bits active in the Interrupt Clear Registers. Figure 7-5 shows the bit positions of each potential interrupt source.

A complete definition of the signal states reported by Interrupt Status Register 1 is shown in Figure 7-5.

	0	1	2	3	4	5		11	12	13	14	15		
Field	CHA_2		CHA_1		A-Err	Reserved					CHA_4		CHA_3	
Definition	Err	Dn	Err	Dn							Err	Dn	Err	Dn
Reset	0x0000													
R/W	R/W													
Addr	0x 01010													

	16													31
Field	Reserved													
Definition														
Reset	0x0000													
R/W	R/W													
Addr	0x 01010													

Figure 7-5. Interrupt Status Register 1

A complete definition of the bits that can be masked by Interrupt Status Register 2 is shown in Figure 7-6.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved		PKEU		Reserved		RNG		Reserved		AFEU		Reserved		MDEU	
Definition			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	R/W															
Addr	0x 01014															

	16	17	18	19	20	21	22	23	24	25	26	27	28			31
Field	Reserved		AESU		Reserved		DEU		GIE	TEA	DPE	APE	Reserved			
Definition			Err	Dn			Err	Dn								
Reset	0x0000															
R/W	R/W															
Addr	0x 01014															

Figure 7-6. Interrupt Status Register 2

7.1.5 Interrupt Clear Register (ICR)

The Interrupt Control Register provides a means of clearing the Interrupt Status Register. When a bit in the ICR is written with a 1, the corresponding bit in the ISR is cleared, clearing the interrupt output pin \overline{IRQ} (assuming the cleared bit in the ISR is the only interrupt source). If the input source to the ISR is a steady-state signal that remains active, the appropriate ISR bit, and subsequently \overline{IRQ} , will be reasserted shortly thereafter. Figure 7-5 shows the bit positions of each interrupt source that can be cleared by this register. The complete bit definitions for the ICR can be found in Figure 7-5.

When an ICR bit is written, it will automatically clear itself one cycle later. That is, it is not necessary to write a “0” to a bit position which has been written with a “1.”

NOTE

Interrupts are registered and sent based upon the conditions which cause them. If the cause of an interrupt is not removed, the interrupt will return a few cycles after it has been cleared using the ICR.

A complete definition of the signal states reported by Interrupt Clear Register 1 is shown in Figure 7-7.

	0	1	2	3	4	5		11	12	13	14	15			
Field	CHA_2		CHA_1		A-Err	Reserved					CHA_4		CHA_3		
Definition	Err	Dn	Err	Dn							Err	Dn	Err	Dn	
Reset	0x0000														
R/W	R/W														
Addr	0x 01018														
	16														31
Field	Reserved														
Definition															
Reset	0x0000														
R/W	R/W														
Addr	0x 01018														

Figure 7-7. Interrupt Clear Register 1

A complete definition of the bits that can be masked by Interrupt Clear Register 2 is shown in Figure 7-8.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Field	Reserved		PKEU		Reserved		RNG		Reserved		AFEU		Reserved		MDEU	
Definition			Err	Dn			Err	Dn			Err	Dn			Err	Dn
Reset	0x0000															
R/W	R/W															
Addr	0x 0101C															

	16	17	18	19	20	21	22	23	24	25	26	27	28		31	
Field	Reserved		AESU		Reserved		DEU		--	TEA	DPE	APE	Reserved			
Definition			Err	Dn			Err	Dn								
Reset	0x0000															
R/W	R/W															
Addr	0x 0101C															

Figure 7-8. Interrupt Clear Register 2

Table 7-2 describes the interrupt mask, status, and clear register signals.

Table 7-2. Interrupt Mask, Status, and Clear Register 1 Signals

Bits	Name	Reset Value	Description
0:1, 2:3, 12:13, 14:15	CH_Err_Dn	0	Each of the 4 channels has Error & Done bits. 0 No error detected. 1 Error detected. Indicates that execution unit status register must be read to determine exact cause of the error. 0 Not DONE. 1 DONE bit indicates that the interrupting channel or EU has completed its operation.
4	A-Err	0	EU Assignment Error bit. This bit indicates that a static assignment of a EU was attempted on a EU which is currently in use. 0 No error detected. 1 EU Assignment Error detected.
Multiple	Reserved	0	Reserved, set to zero.

Table 7-2 describes the interrupt mask, status, and clear register signals.

Table 7-3. Interrupt Mask, Status, and Clear Register 2 Signals

Bits	Name	Reset Value	Description
Multiple	Reserved	0	Reserved, set to zero.
2:3, 6:7, 10:11, 14:15, 18:19, 22:23	EU_Err_Dn	0	Each of the execution units has Error & Done bits. 0 No error detected. 1 Error detected. Indicates that execution unit status register must be read to determine exact cause of the error. 0 Not DONE. 1 DONE bit indicates that the interrupting channel or EU has completed its operation.
24	GIE	0	Global Interrupt Enable (GIE). Starting with the MPC184VFB, the GIE bit reflects the individual interrupt source when the interrupt status register is enabled at reset. The GIE bit resets to disabled, it allows the user to selectively mask individual interrupt sources in the interrupt mask register before enabling the remaining unmasked interrupt sources. Note: This bit is reserved in the PPC184VF and the PPC184VFA.
25	TEA	0	Transfer Error Acknowledge. Set when the MPC184 as a master receives a Transfer Error Acknowledge. 0 No error detected. 1 TEA detected on 8xx bus.
26	DPE	0	Data Parity Error. Set when the MPC184 detects a slave data parity error. 0 No error detected. 1 DPE detected on 8xx bus.
27	APE	0	Address Parity Error. Set when the MPC184 detects a slave address parity error. 0 No error detected. 1 APE detected on 8xx bus.

7.1.6 ID Register

The Read-Only ID Register, displayed in Figure 7-9, contains a 32-bit value that uniquely identifies the version of the MPC184. The value of this register is always 0x1000_0000, indicating that this is the first version of the MPC184.

	0	31
Field	Revision ID	
Reset	0x1000_0000	
R/W	R	
Addr	0x 01020	

Figure 7-9. ID Register

7.1.7 Master Control Register (MCR)

The MCR, shown in Figure 7-10, controls certain functions in the controller and provides a means for software to reset the MPC184.

	0	1	2	6	7	8	31
Field	PE	PT	Reserved		SwR	Reserved	
Reset	0x0000_0041						
R/W	R/W						
Addr	Master Control Register 1 0x 01030						

	0	7	8	15	16	23	24	31
Field	CHA3_EU_PR_CNT		CHA4_EU_PR_CNT		CHA3_BUS_PR_CNT		CHA4_BUS_PR_CNT	
Field	msb<-----lsb		msb<-----lsb		msb<-----lsb		msb<-----lsb	
Reset	0x0000_0000							
R/W	R/W							
Addr	Master Control Register 2 0x 01034							

Figure 7-10. Master Control Register

Table 7-4 describes the Master Control Register signals.

Table 7-4. Master Control Register Signals

Bits	Name	Reset Value	Description
0	PE	0	Parity Enable: Parity Enable (bit 7): Writing '1' to this bit will enable parity to be checked for Master Read Data.
1	PT	0	Parity Type: (bit 6): In 8xx mode this bit defines whether the parity generated and checked is even (bit = 1) or odd (bit = 0).
2:6	Reserved	0	Reserved
7	SWR	0	Software Reset. Writing 1 to this bit will cause a global software reset. Upon completion of the reset, this bit will be automatically cleared and zero will be written to all locations of the gpRAM. 0 Don't reset 1 Global Reset
8:31	Reserved	0	Reserved- Note: Do not change the reset values in this reserved field.

Table 7-5 describes the Master Control Register 2 signals.

Table 7-5. Master Control Register 2 signals

Bits	Name	Reset Value	Description
0:7	CHN3_EU_PR_CNT	0	Channel 3 EU Priority Counter. This counter is used by the controller to determine when Channel 3 has been denied access to a requested EU long enough to warrant immediate elevation to top priority. Note: If set to zero, the CHA4_EU_PR_CTR must also be set to zero, and the controller will assign EU's on a pure round robin basis. If set to non-zero, CHN4_EU_PR_CTR must also be set to a different, non-zero value.
8:15	CHN4_EU_PR_CNT	0	Channel 4 EU Priority Counter. This counter is used by the controller to determine when Channel 4 has been denied access to a requested EU long enough to warrant immediate elevation to top priority. Note: If set to zero, the CHN3_EU_PR_CTR must also be set to zero, and the controller will assign EU's on a pure round robin basis. If set to non-zero, CHA3_EU_PR_CTR must also be set to a different, non-zero value.
16:23	CHN3_BUS_PR_CNT	0	Channel 3 Bus Priority Counter. This counter is used by the controller to determine when Channel 3 has been denied access to the PCI bus long enough to warrant immediate elevation to top priority. Note: If set to zero, the CHN4_BUS_PR_CTR must also be set to zero, and the controller will assign access to the PCI bus on a pure round robin basis. If set to non-zero, the user has the option to set to the same value as CHN4_BUS_PR_CTR.
24:31	CHN4_BUS_PR_CNT	0	Channel 4 Bus Priority Counter. This counter is used by the controller to determine when Channel 4 has been denied access to a needed on-chip resource long enough to warrant immediate elevation to top priority. Note: If set to zero, the CHN3_BUS_PR_CTR must also be set to zero, and the controller will assign access to the PCI bus on a pure round robin basis. If set to non-zero, the user has the option to set to the same value as CHN3_BUS_PR_CTR.

7.1.8 Master TEA Error Address Register (MTAR)

A Transfer Error Acknowledge (TEA) signal indicates a fatal error has occurred during the data phase of a 8xx bus transaction. Invalid data may have been received and stored prior to the receipt of the TEA. This register saves the address of the transaction whose data phase was terminated with a TEA. The channel that was initiating the transaction will be evident from that channel's error interrupt. The host CPU may chose to reset the channel reporting the TEA, reset the whole MPC184, or reset the entire system with a machine check error. In any case, the host may chose to preserve this TEA information prior to reset to assist in debug.

The MTAR only holds the address of the first TEA reported, in the event multiple TEAs are received before the first is cleared.

	0	31
Field	Reserved	
Reset	0x0000_0000	
R/W	R	
Addr	0x 01038	

	0	31
Field	Address	
Reset	0x0000_0000	
R/W	R	
Addr	0x 0103C	

Figure 7-11. Master TEA Address Register

Table 7-6 defines the Master TEA Address Register bits.

Table 7-6. Master TEA Address Register Bit Definitions

Bits	Name	Reset Value	Description
0:31	Address	0	Target address of the transaction when TEA was received.

7.1.9 EU Access

Assignment of a EU function to a channel is done either statically or dynamically. In the case of static assignment, a EU is assigned to a channel via the EU Assignment Control Register (EUACR). Once a EU is statically assigned to a channel, it will remain that way until the EUACR is written and the assignment is removed. In the case of dynamic assignment, the channel requests a EU function, the controller checks to see if the requested EU function is available, and if it is, the controller grants the channel assignment of the EU. Note that the channel does not need to know which EU it receives, only that the function is assigned.

For the case of a EU function for which multiple EUs exist (e.g., the PKEU function), the controller will implement a priority scheme to find the first available EU. The priority will be from PKEU1 to PKEU2. That is, if a PKEU function is requested, the controller will first check PKEU1, and then PKEU2, until it finds an unused PKEU.

If a EU is available for a channel when requested, the controller will assert the grant signal pertaining to the request from the channel. The grant signal will remain asserted until the channel issues the done signal.

7.1.10 Multiple EU Assignment

In some cases, a channel may request two EUs. The channel will do this by first requesting the primary EU, then requesting the secondary EU. Once the controller has granted both

EUs, this channel is then capable of requesting that the secondary EU snoop the bus. Snooping is described in Table 6-4.

In all cases, the controller assigns the primary EU to a requesting channel as the EUs become available. The controller does not wait until both EUs are available before issuing any grants to a channel which is requesting two EU functions.

7.1.11 Multiple Channels

Since there are multiple channels in the MPC184, the controller must arbitrate for access to the execution units. To accomplish this, the controller implements an arbiter for each channel.

Each arbiter acts on either a weighted priority-based or round-robin scheme, depending on the values of `CHA3_EU_PR_CNT` and `CHA4_EU_PR_CNT`. If both `CHA3_EU_PR_CNT` and `CHA4_EU_PR_CNT` are set to a non-zero value, the arbiter will implement the weighted priority scheme. Otherwise, the arbitration will be round-robin. Setting only one of the `CHA_EU_PR_CNT` fields to a non-zero value will result in unpredictable operation.

7.1.12 Priority Arbitration

When arbitrating on the priority scheme, the priority will be as follows:

- Channel 1 -- Highest priority
- Channel 2 -- Second highest priority, unless `CHA3_EU_PR_CNT` or `CHA4_EU_PR_CNT` expired
- Channel 3 -- Third priority, unless `CHA4_EU_PR_CNT` expired.
- Channel 4 -- Lowest priority, until `CHA4_EU_PR_CNT` expired

For channels 1-4, the priority is channel 1, channel 2, channel 3, and channel 4, in that order. In order to prevent channels 3 and 4 from being locked out, the `CHA3_EU_PR_CNT` and `CHA4_EU_PR_CNT` fields are implemented in the Master Control Register. The value of these fields determines how many times channel 3 or channel 4 can be refused access to an EU in favor of a higher priority channel. A counter is implemented in the arbiter for each of these entities. When the channel has lost arbitration the number of times specified in its `CHA_EU_PR_CNT` field, then that channel has the 2nd highest priority when the requested EU becomes available. `CHA1` always has the highest priority, but it cannot make back to back requests, so the 2nd highest priority channel will be serviced upon completion of the current `CHA1` operation.

It is permissible for the `CHA_EU_PR_CNT` values to be different from the `CHA_BUS_PR_CNT` values, i.e., EU access may be prioritized, while bus access is pure round robin, and vice-versa.

7.1.13 Round Robin Snapshot Arbiters

The controller implements eight ‘snapshot’ arbiters, one for each EU function, and one for the 8xx bus. Each arbiter takes a snapshot of the requests for its function. If there are requests, then the arbiter satisfies those requests via a round-robin scheme as the resource becomes available. When all requests have been satisfied, the arbiter takes another snapshot.

7.1.14 Bus Access

Bus access is granted via the same scheme that is used for granting EUs. When the CHA_BUS_PR_CNT values of both channel 3 and 4 are set to zero, round robin operation is in effect. In this case, the snapshot arbiter samples the requests for the bus, then grants those requests as the bus becomes available. For example, if channels 1, 2, and 4 are requesting bus access at a given time, the snapshot arbiter will register the three requests and ignore further requests. The buses will be granted to channel 1 until its transfer is completely satisfied. Then the buses will be granted to channel 2 until channel 2’s transfer is completely satisfied. Finally, the buses will be granted to channel 4 until that transfer is completely satisfied. Then another snapshot of requests will be taken.

When arbitrating on the priority scheme, the priority will be as follows:

- Channel 1 -- Highest priority
- Channel 2-- Second priority, unless CHA3_BUS_PR_CNT or CHA4_BUS_PR_CNT expired.
- Channel 3 -- Third priority, unless CHA4_BUS_PR_CNT expired.
- Channel 4 -- Lowest priority, until CHA4_BUS_PR_CNT expired

For channels 1-4, the priority is channel 1, channel 2, channel 3, and channel 4, in that order. In order to prevent channels 3 and 4 from being locked out, the CHA3_BUS_PR_CNT and CHA4_BUS_PR_CNT fields are implemented in the master control register. The value of these fields determines how many times channel 3 or channel 4 can be refused access to the 8xx184 in favor of a higher priority channel. A counter is implemented in the arbiter for each of these entities. When the channel has lost arbitration the number of times specified in its CHA_BUS_PR_CNT field, then that channel has the 2nd highest priority when the BUS becomes available. CHA1 always has the highest priority, but it cannot make back to back requests, so the 2nd highest priority channel will be serviced upon completion of the current CHA1 operation.

It is permissible for the CHA_BUS_PR_CNT values to be different from the CHA_EU_PR_CNT values, i.e., EU access may be prioritized, while bus access is pure round robin, and vice-versa.



THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 8

8xx Bus Interface Module

8.1 8xx Interface

The 8xx Bus Interface Module handles the interface between the system and the internal modules of the MPC184. The 8xx Bus Protocol is used in systems implementing the PowerQuicc 1 architecture. The interface is capable of up to 66MHz operation, and operates at 3.3v I/O.

See the MPC184 Hardware Reference Manual for additional information regarding connecting the MPC184 to the 8xx bus of a PowerQuicc 1 processor.

8.1.1 Bus Access

The controller in the MPC184 has the ability to be a 8xx initiator or a 8xx target. This means that the controller can issue read and write commands to the 8xx bus, and it can also be written to and read from by the host.

The controller is the sole bus master in the MPC184. All other modules are slave only devices. A channel may request access to system resources including the 8xx bus. In these cases, the channel must provide the starting address of the transfer for the bus(es) requested. All subsequent addresses are generated by the controller. All addresses will be sequential.

8.1.2 8xx Initiator

The mechanism for transferring data to and from the 8xx bus as an initiator is either through a 8xx read request or a 8xx write request to the 8xx IF block. As an initiator, the MPC184 uses both single and burst read/writes. Target accesses take priority over initiator accesses. It is possible that an initiator access can be interrupted (internal to the MPC184) by a target access. This occurs when a request has been made to the 8xx IF for initiator access and a target access occurs before the MPC184 is granted access to the 8xx bus. In this case, the controller saves the state of the initiator transfer, satisfies the target access, then resumes with the initiator transfer at the point where the initiator transfer was interrupted.

8.1.3 Parity Errors

The 8xx Interface Module can be programmed to check for even or odd parity errors when transmitting and receiving data, if programmed to do so by the PT and PE bits in the Master Control Register. If a master parity error occurs, the controller routes the parity error indication to the appropriate channel and the channel generates an error interrupt. If the MPC184 detects a parity error when acting as a slave, the address on the 8xx bus is stored in the Slave Parity Error Registers, and an interrupt is generated.

8.1.4 8xx Read

The sequence for 8xx read access is as follows:

- Channel asserts its 8xx read request.
- Channel furnishes address and transfer length.
- Controller acknowledges request to channel.
- Controller asserts request to 8xx interface module.
- Controller waits for 8xx read to begin.
- When 8xx read begins, controller receives data from the 8xx interface module and performs a master write to the appropriate internal address using the address supplied by the channel. Data always goes through the misalignment block.
- Transfer continues until the 8xx burst read is completed and the controller has written all data to the appropriate internal address. The 8xx interface module will continue making 8xx bus requests until the full data length has been read. The 8xx interface module will also make single reads when the amount of data remaining to be fetched is less than a 8xx burst (less than a full cache line.)

8.1.4.1 Target Aborts

It is possible for the intended target of an MPC184 master initiated transaction to terminate the transfer due to an error. Every time a Transfer Error Acknowledge is received from a target, the TEA bit in the Controller's Interrupt Status Register is set, and, unless masked by the Interrupt Mask Register, the MPC184 channel which requested the transfer will signal interrupt via the Interrupt Status Register. The host will be able to determine which channel generated the interrupt by checking the ISR for the channel ERROR bit. The controller will also log the target address which terminated with a TEA in the Master TEA Error Address Register.

8.1.5 Initiator Write

Initiator writes are performed by transferring data from one of the EUs to the output FIFO in the controller, then transferring the data from the FIFO to the 8xx bus when the 8xx is granted to the controller. The sequence for a 8xx write access is as follows:

- Channel requests the 8xx bus from controller.
- Controller acknowledges request to channel.
- Channel furnishes address, transfer length.
- Controller loads the write data into its FIFO, and waits for the 8xx bus to become available.
- When the 8xx bus becomes available, controller writes data from its FIFO to the 8xx interface module.
- Transfer continues until the 8xx burst write is completed and the controller has read all data from the appropriate internal address. The 8xx interface module will continue making 8xx bus requests until the full data length has been written. The 8xx interface module will also make single reads when the amount of data remaining to be written is less than a 8xx burst (less than a full cache line.) All MPC184 initiator writes must be 32-bit word aligned.

NOTE

It is probable that several 8xx bursts will be required to complete any given request. When a channel has been granted access to the 8xx bus, no other requests to the 8xx bus will be acknowledged until that transfer has been fully satisfied; i.e., all bytes have been transferred.

8.1.6 Misaligned Data

The controller has the ability to initiate a 8xx read. In some cases, the address for the read may not be aligned to a modulo 4 boundary. In these cases, the controller will realign the data to a modulo 4 boundary as it comes in from the 8xx bus.

The data alignment block will fetch the data from the 8xx bus at modulo 4 addresses. This block will continue fetching data until the number of bytes left to read is equal to or less than the width of the master data bus.

An example of misaligned data is shown in Figure . Note that the data alignment block aligns the read data to a quad-word address boundary ('h100) for cycle 2. Then, with only 6 bytes remaining, the data alignment block aligns the 6 bytes to the word address boundary and fills the last two bytes with zeroes.

10 Byte Initiator Read from Addr 0x102

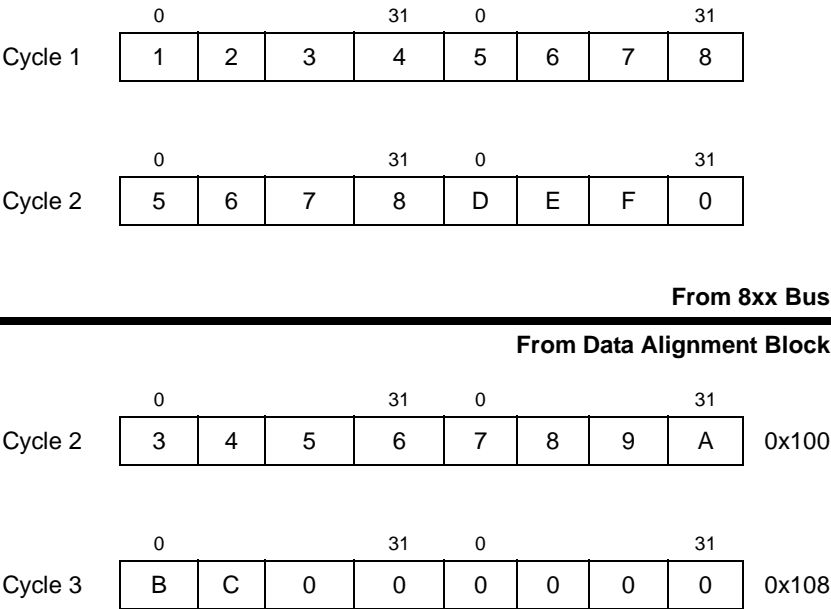


Figure 8-1. Data Alignment Example

8.1.7 8xx Target

The controller also acts as a 8xx target. As a target, the controller simply responds to read and write commands from the 8xx bus. When a write command is received from the 8xx bus, the controller takes the data from the 8xx interface module and sends it to whichever internal location is indicated by the address. For a read, the controller goes to the internal location and fetches the requested data from the specified address.

Target accesses from the 8xx bus are given priority over all other bus accesses in the MPC184.



THIS PAGE INTENTIONALLY LEFT BLANK



3xx Interface

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Appendix A

User’s Manual Revision History

for the

MPC184 8xx Security Co-Processor

User’s Manual

This appendix provides a list of the major differences between revisions. Note that the list only covers the major changes to the user’s manual.

Major changes to the *MPC184 8xx Security Co-Processor User’s Manual* from Revision 1. to Revision 1.1 were to include the GIE bit in the Interrupt Mask Register 2 and Interrupt Status Register 2, details are as follows:

Section, Page	Changes
7.1.3, 7-3	Figure 7-4, the global interrupt enable (GIE), bit 31, was added to the interrupt mask register 2.
7.1.4, 7-4	Figure 7-6, the global interrupt enable (GIE), bit 31, was added to the interrupt status register 2.
7.1.5, 7-6	Table 7-3, a description was added. for the GIE, bit 31.



Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Index

A

Address map, 3-1, 3-5
 AFEU
 context, 4-34
 context memory, 4-34
 context pointer register, 4-34
 data size register, 4-27
 FIFOs, 4-35
 interrupt control register, 4-32
 interrupt status register, 4-30
 key registers, 4-35
 key size register, 4-26
 mode register, 4-24
 register map, 4-24
 reset control register, 4-28
 status register, 4-29, 4-40, 4-49, 4-58
 An (address bus) signals, 2-13
 ARC Four execution unit (AFEU), 4-23

B

BB (bus busy) signal, 2-14
 BDIP (burst data in progress) signal, 2-13
 BG (bus grant) signal, 2-14
 BI (burst inhibit) signal, 2-14
 BR (bus request) signal, 2-14
 BURST (burst transfer) signal, 2-13
 Bus access, 8-1

C

Channel reset, 6-14
 Controller registers, 7-1
 Crypto-channel
 configuration register, 6-2
 current descriptor pointer register, 6-4
 pointer status register, 6-4
 Crypto-channel registers, 6-2

D

Data encryption standard execution units (DEU), 4-12, 4-53
 Descriptor
 buffer, 6-4

 chaining, 5-6
 classes, 5-9
 Descriptor structure, 5-1
 DEU
 FIFOs, 4-23
 interrupt control register, 4-20, 4-61
 interrupt status register, 4-18, 4-59
 IV register, 4-23
 key registers, 4-23
 key size register, 4-14, 4-15, 4-55
 mode register, 4-13, 4-54
 register map, 4-13, 4-53
 reset control register, 4-16, 4-57
 Dn (data bus) signals, 2-14
 DPn (parity bus) signals, 2-14
 Dynamic descriptors, 5-12

E

EU
 access, 7-11
 assignment control register, 7-1
 assignment status register, 7-2

F

Fetch register, 6-11

I

ID register, 7-8
 Initiator write, 8-2
 Interrupt
 clear register, 7-6
 mask register, 7-3
 status register, 7-4
 Interrupts
 channel done, 6-14
 channel error, 6-14
 general, 6-14

K

KR/RETRY (kill reservation/retry) signal, 2-14

M

Master control register, 7-9

MDEU

- context registers, 4-44
- data size register, 4-38
- FIFOs, 4-46
- interrupt control register, 4-42
- interrupt status register, 4-41
- key registers, 4-45
- key size register, 4-37, 4-38
- mode register, 4-36
- register map, 4-35
- reset control register, 4-39

Message digest execution unit (MDEU), 4-35

Misaligned data, 8-3

Multiple channels, 7-12

Multiple EU assignment, 7-11

N

Null fields, 5-8

P

Parity errors, 8-2

PCI

- initiator, 8-1
- local bus interface, 8-1
- read, 8-2
- target, 8-4

PKEU

- data size register, 4-5
- EU_GO register, 4-11, 4-22, 4-33, 4-44, 4-52
- interrupt control register, 4-10
- interrupt status register, 4-8
- key size register, 4-4
- mode register, 4-2
- parameter memory A, 4-12
- parameter memory B, 4-12
- parameter memory E, 4-12
- parameter memory N, 4-12
- register map, 4-2
- reset control register, 4-6
- status register, 4-7, 4-17

Public key execution unit (PKEU), 4-2

R

Random number generator (RNG)
overview, 4-46

RD/WR (read/write) signal, 2-13

Registers

AFEU

- context pointer, 4-34
- data size, 4-27

interrupt control, 4-32

interrupt status, 4-30

key, 4-35

key size, 4-26

mode, 4-24

reset control, 4-28

status, 4-29, 4-40, 4-49, 4-58

crypto-channel

configuration, 6-2

current descriptor pointer, 6-4

general, 6-2

pointer status, 6-4

DEU

interrupt control, 4-20, 4-61

interrupt status, 4-18, 4-59

IV, 4-23

key, 4-23

key size, 4-14, 4-15, 4-55

mode, 4-13, 4-54

reset control, 4-16, 4-57

EU assignment control, 7-1

EU assignment status, 7-2

fetch, 6-11

ID, 7-8

interrupt

clear, 7-6

mask, 7-3

status, 7-4

master control, 7-9

MDEU

context registers, 4-44

data size, 4-38

interrupt control, 4-42

interrupt status, 4-41

key, 4-45

key size, 4-37, 4-38

mode, 4-36

reset control, 4-39

PKEU

EU_GO, 4-11, 4-22, 4-33, 4-44, 4-52

interrupt control, 4-10

interrupt status, 4-8

key size, 4-4

reset control, 4-6

status, 4-7, 4-17

PKEU data size, 4-5

RNG

data size, 4-48

interrupt control, 4-51

interrupt status, 4-51

mode, 4-47

reset control, 4-49

RNG

data size register, 4-48

- FIFO, 4-53
- functional description, 4-46
- interrupt control register, 4-51
- interrupt status register, 4-51
- mode register, 4-47
- register map, 4-47
- reset control register, 4-49
- RSV (reservation transfer) signal, 2-14

S

Signal descriptions, 2-13

Signals

- An, 2-13
- BB, 2-14
- BDIP, 2-13
- BG, 2-14
- BI, 2-14
- BR, 2-14
- BURST, 2-13
- Dn, 2-14
- DPn, 2-14
- KR/RETRY, 2-14
- RD/WR, 2-13
- RSV, 2-14
- TA, 2-14
- TEA, 2-14
- TS, 2-13
- TSIZn, 2-13
- Snapshot arbiters, 7-13
- Software reset, 6-15
- Static descriptors, 5-9

T

- TA (transfer acknowledge) signal, 2-14
- Target aborts, 8-2
- TEA (transfer error acknowledge) signal, 2-14
- TS (transfer start) signal, 2-13
- TSIZn (transfer size) signals, 2-13





Overview	1
Signal Descriptions	2
Address Map	3
Execution Units	4
MPC185 Descriptors	5
Crypto-Channels	6
Controller	7
60x Bus Interface Module	8
User's Manual Revision History	A
Index	IND



Freescale Semiconductor, Inc.

1	Overview
2	Signal Descriptions
3	Address Map
4	Execution Units
5	MPC185 Descriptors
6	Crypto-Channels
7	Controller
8	60x Bus Interface Module
A	User's Manual Revision History
IND	Index