

# **QorIQ LS1043A Data Path Acceleration Architecture (DPAA) Reference Manual**

**Supports**  
LS1023A  
LS1043A

LS1043ADPAARM  
Rev. 0  
04/2016



**How to Reach Us:**

**Home Page:**

[nxp.com](http://nxp.com)

**Web Support:**

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP 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 that may be provided in NXP 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. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:  
[nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

Freescale, the Freescale logo, Layerscape, and QorIQ are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA and ARM Powered are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates.

© 2016 NXP B.V.

# Contents

Paragraph Number	Title	Page Number
	<b>Terminology, Conventions, and Resources</b>	
	Terms, Acronyms, and Abbreviations .....	lxxxiii
	Notational Conventions .....	lxxxvi
	Suggested Reading.....	lxxxvii
	<b>Chapter 1</b>	
	<b>Data Path Acceleration Architecture (DPAA) Overview</b>	
1.1	Purpose of this manual.....	1-1
1.2	General DPAA Functionality .....	1-1
1.2.1	Packet Distribution and Queue/Congestion Management.....	1-2
1.3	DPAA Programming Model.....	1-2
1.4	DPAA Terms and Definitions .....	1-3
1.5	Major DPAA Components .....	1-4
1.5.1	Queue Manager (QMan).....	1-4
1.5.2	Buffer Manager (BMan) .....	1-5
1.5.3	Frame Manager (FMan).....	1-5
1.5.3.1	FMan Network Interfaces .....	1-6
1.5.3.2	FMan Parse Function.....	1-6
1.5.3.3	FMan Distribution and Policing .....	1-7
1.5.3.3.1	FMan Policier .....	1-7
1.5.4	Security Engine (SEC).....	1-8
1.6	Data Formats Used in the DPAA .....	1-9
1.6.1	Frame Descriptor (FD).....	1-9
1.6.1.1	FD Format.....	1-9
1.6.1.2	Frame Descriptor (FD) Considerations .....	1-10
1.6.2	Multi-Buffer Frames .....	1-11
1.6.2.1	Scatter/Gather Entry Format.....	1-11
1.6.2.2	Multi-Buffer Frame Considerations.....	1-11
1.6.2.3	Situations Where Multi-Buffer-Frame Processing Stops .....	1-12
1.6.3	Single-Buffer Frames.....	1-13
1.6.4	Compound Frames .....	1-13
1.6.4.1	When to Use Compound Frames.....	1-13
1.6.4.2	Compound Frame Considerations .....	1-13
1.6.5	Simple Frames .....	1-14
1.6.6	Frame Format Codes.....	1-15
1.6.7	Frame Formats Supported by Accelerators .....	1-15
1.6.8	Special Values and Exceptions .....	1-15
1.6.9	Releasing Buffers to the BMan.....	1-16
1.7	Accessing Memory Using Isolation Context Identifier (ICID) .....	1-16

# Contents

Paragraph Number	Title	Page Number
1.8	Packet Walk-Through Example .....	1-17
1.9	Specific DPAA Implementation Details .....	1-18
1.9.1	Queue Manager (QMan) Implementation.....	1-18
1.9.2	Buffer Manager (BMan) Implementation.....	1-18
1.9.3	Frame Manager (FMan) Implementation .....	1-19
1.9.4	Security and Encryption Engine (SEC) Implementation.....	1-19

## Chapter 2 Memory Map

2.1	Memory Map Overview.....	2-1
2.1.1	Considerations When Accessing Reserved Registers and Bits .....	2-1
2.1.2	DPAA Address Map .....	2-1

## Chapter 3 Queue Manager (QMan)

3.1	QMan Introduction .....	3-1
3.1.1	QMan Overview .....	3-1
3.1.2	QMan Features.....	3-2
3.2	QMan Memory Map and Register Definition.....	3-3
3.2.1	QMan Software Portals (QCSP) Memory Map.....	3-3
3.2.2	QMan Configuration and Control Register Memory Map .....	3-8
3.2.3	QMan Software Portals (QCSP) Register Descriptions .....	3-13
3.2.3.1	QCSP Enqueue Command Ring Registers (QCSP <sub>i</sub> _EQCR0-7).....	3-13
3.2.3.2	QCSP Dequeue Response Ring Registers (QCSP <sub>i</sub> _DQRR0-15).....	3-15
3.2.3.3	QCSP Message Ring Registers (QCSP <sub>i</sub> _MR0-7) .....	3-18
3.2.3.4	QCSP Management Command Registers (QCSP <sub>i</sub> _CR).....	3-19
3.2.3.5	QCSP Management Response Registers (QCSP <sub>i</sub> _RR0-1) Format.....	3-20
3.2.3.6	QCSP Enqueue Command Ring (EQCR) Producer Index Registers .....	3-21
3.2.3.6.1	QCSP EQCR Producer Index Cache-Enabled Registers (QCSP <sub>i</sub> _EQCR_PI_CENA) .....	3-22
3.2.3.6.2	QCSP EQCR Producer Index Cache-Inhibited Registers (QCSP <sub>i</sub> _EQCR_PI_CINH) .....	3-22
3.2.3.7	QCSP EQCR Consumer Index Registers .....	3-23
3.2.3.7.1	QCSP EQCR Consumer Index Cache-Enabled Registers (QCSP <sub>i</sub> _EQCR_CI_CENA).....	3-24
3.2.3.7.2	QCSP EQCR Consumer Index Cache-Inhibited Registers (QCSP <sub>i</sub> _EQCR_CI_CINH).....	3-25
3.2.3.8	QCSP Dequeue Response Ring (DQRR) Producer Index Registers.....	3-25

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
3.2.3.8.1	QCSP DQRR Producer Index Cache-Enabled Registers ( <i>QCSPi_DQRR_PI_CENA</i> ).....	3-26
3.2.3.8.2	QCSP DQRR Producer Index Cache-Inhibited Registers ( <i>QCSPi_DQRR_PI_CINH</i> ).....	3-27
3.2.3.9	QCSP DQRR Consumer Index Registers ( <i>QCSPi_DQRR_CI_CENA</i> ).....	3-28
3.2.3.9.1	QCSP DQRR Consumer Index Cache-Enabled Registers ( <i>QCSPi_DQRR_CI_CENA</i> ) .....	3-28
3.2.3.9.2	QCSP DQRR Consumer Index Cache-Inhibited Registers ( <i>QCSPi_DQRR_CI_CINH</i> ) .....	3-29
3.2.3.10	QCSP DQRR Discrete Consumption Acknowledgment and Park ( <i>QCSPi_DQRR_DCAP</i> ).....	3-29
3.2.3.11	QCSP DQRR Static Dequeue Command Register (SDQCR) .....	3-31
3.2.3.12	QCSP DQRR Volatile Dequeue Command Register ( <i>QCSPi_DQRR_VDQCR</i> ) .....	3-34
3.2.3.13	QCSP DQRR Pull Dequeue Command Register ( <i>QCSPi_DQRR_PDQCR</i> ) .....	3-35
3.2.3.14	QCSP MR Producer Index Registers.....	3-36
3.2.3.14.1	QCSP MR Producer Index Cache-Enabled Registers ( <i>QCSPi_MR_PI_CENA</i> )..... 3-37	3-37
3.2.3.14.2	QCSP MR Producer Index Cache-Inhibited Registers ( <i>QCSPi_MR_PI_CINH</i> ) .... 3-37	3-37
3.2.3.15	QCSP MR Consumer Index Registers.....	3-38
3.2.3.15.1	QCSP MR Consumer Index Cache-Enabled Registers ( <i>QCSPi_MR_CI_CENA</i> ) ... 3-38	3-38
3.2.3.15.2	QCSP MR Consumer Index Cache-Inhibited Registers ( <i>QCSPi_MR_CI_CINH</i> ) ... 3-39	3-39
3.2.3.16	QCSP Read-Only Ring Indices Cache-Enabled Registers ( <i>QCSPi_RORI_CENA</i> ).... 3-39	3-39
3.2.3.17	QMan Software Portal Configuration Register ( <i>QCSPi_CFG</i> ).....	3-41
3.2.3.18	QCSP EQCR Interrupt Threshold Register ( <i>QCSPi_EQCR_ITR</i> ) .....	3-44
3.2.3.19	QCSP DQRR Interrupt Threshold Register ( <i>QCSPi_DQRR_ITR</i> ) .....	3-45
3.2.3.20	QCSP MR Interrupt Threshold Register ( <i>QCSPi_MR_ITR</i> ) .....	3-45
3.2.3.21	QCSP Interrupt Status Register ( <i>QCSPi_ISR</i> ) .....	3-46
3.2.3.22	QCSP Interrupt Enable Registers ( <i>QCSPi_IER</i> ) .....	3-47
3.2.3.23	QCSP Interrupt Status Disable Registers ( <i>QCSPi_ISDR</i> ) .....	3-48
3.2.3.24	QCSP Interrupt Inhibit Registers ( <i>QCSPi_IIR</i> ).....	3-48
3.2.3.25	QCSP Interrupt Time-Out Period Registers ( <i>QCSPi_ITPR</i> ) .....	3-49
3.2.4	QMan Configuration and Control Register Descriptions .....	3-50
3.2.4.1	QCSP ICID Configuration Registers ( <i>QCSPi_ICID_CFG</i> ) .....	3-50
3.2.4.2	QCSP IO Configuration Registers ( <i>QCSPi_IO_CFG</i> ) .....	3-50
3.2.4.3	QCSP Dynamic Debug Configuration Registers ( <i>QCSPi_DD_CFG</i> ) .....	3-51
3.2.4.4	QMan Dynamic Debug Configuration Register ( <i>QMANTDDCFG</i> ).....	3-52

# Contents

Paragraph Number	Title	Page Number
3.2.4.5	QCSP Dynamic Debug Internal Halt Request Status Registers (QCSP_DD_IHRSR_i) 3-53	3-53
3.2.4.6	DCP Dynamic Debug Internal Halt Request Status Register (DCP_DD_IHRSR)3-54	3-54
3.2.4.7	QCSP Dynamic Debug Internal Halt Request Force Registers (QCSP_DD_IHRFR_i) 3-55	3-55
3.2.4.8	DCP Dynamic Debug Internal Halt Request Force Register (DCP_DD_IHRFR) 3-55	3-55
3.2.4.9	QCSP Dynamic Debug Halt Acknowledge Status Registers (QCSP_DD_HASR_i)... 3-56	3-56
3.2.4.10	DCP Dynamic Debug Halt Acknowledge Status Register (DCP_DD_HASR) .... 3-57	3-57
3.2.4.11	DCP Configuration Registers (DCPi_CFG)..... 3-58	3-58
3.2.4.12	DCP Dynamic Debug Configuration Registers (DCPi_DD_CFG)..... 3-59	3-59
3.2.4.13	DCP Dequeue Latency Monitor Configuration Registers (DCPi_DLM_CFG).... 3-60	3-60
3.2.4.14	DCP Dequeue Latency Monitor Average Registers (DCPi_DLM_AVG)..... 3-61	3-61
3.2.4.15	PFDR Free Pool Count Register (PFDR_FPC) ..... 3-62	3-62
3.2.4.16	PFDR Free Pool Head Pointer Register (PFDR_FP_HEAD) ..... 3-62	3-62
3.2.4.17	PFDR Free Pool Tail Pointer Register (PFDR_FP_TAIL) ..... 3-63	3-63
3.2.4.18	PFDR Free Pool Low Watermark Interrupt Threshold (PFDR_FP_LWIT)..... 3-63	3-63
3.2.4.19	PFDR Configuration (PFDR_CFG) ..... 3-64	3-64
3.2.4.20	SFDR Configuration Register (SFDR_CFG) ..... 3-64	3-64
3.2.4.21	SFDR In Use Register (SFDR_IN_USE) ..... 3-65	3-65
3.2.4.22	Work Queue Class Scheduler Configuration Registers (WQ_CS_CFG <i>i</i> ) ..... 3-65	3-65
3.2.4.23	WQ Default Enqueue WQID Register (WQ_DEF_ENQ_WQID) ..... 3-67	3-67
3.2.4.24	WQ Channel Dynamic Debug Configuration Registers..... 3-67	3-67
3.2.4.24.1	WQ Channel Software Portal Dynamic Debug Configuration Registers (WQ_SC_DD_CFG_ <i>i</i> ) ..... 3-67	3-67
3.2.4.24.2	WQ Channel Pool Dynamic Debug Configuration Registers (WQ_PC_DD_CFG_ <i>i</i> ) ..... 3-68	3-68
3.2.4.24.3	WQ Channel DCP Dynamic Debug Configuration Registers (WQ_DCx_DD_CFG_ <i>i</i> )..... 3-70	3-70
3.2.4.25	CM Configuration Register (CM_CFG)..... 3-71	3-71
3.2.4.26	CEETM Configuration Index Register (CEETM_CFG_IDX)..... 3-72	3-72
3.2.4.27	CEETM Configuration Shaper Pre-Scaler Register (CEETM_CFG_PRES)..... 3-72	3-72
3.2.4.28	CEETM XSFDR In Use Register (CEETM_XSFDR_IN_USE) ..... 3-73	3-73
3.2.4.29	QMan Error Capture Status Register (QMAN_ECSR) ..... 3-74	3-74
3.2.4.30	QMan Error Capture Information Registers (QMAN_ECIR and QMAN_ECIR2)..... 3-75	3-75
3.2.4.31	QMan ECC Error Address Register (QMAN_EADR)..... 3-76	3-76
3.2.4.32	QMan ECC Error Data Registers (QMAN_EDATA <i>i</i> )..... 3-77	3-77
3.2.4.33	QMan Single-Bit ECC Error Threshold Register (QMAN_SBET) ..... 3-78	3-78
3.2.4.34	QMan Single Bit ECC Error Count Registers (QMAN_SBEC <i>i</i> ) ..... 3-79	3-79
3.2.4.35	QMan Management Command/Result Register (QMAN_MCR) ..... 3-80	3-80

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
3.2.4.36	QMan Management Command Parameter 0 Register (QMAN_MCP0).....	3-81
3.2.4.37	QMan Management Command Parameter 1 Register (QMAN_MCP1).....	3-83
3.2.4.38	QMan Management Command Result Registers (QMAN_MR <i>i</i> ) .....	3-83
3.2.4.39	QMan Miscellaneous Configuration Register (QMAN_MISC_CFG).....	3-84
3.2.4.40	QMan Idle Status Register (QMAN_IDLE_STAT).....	3-85
3.2.4.41	QMan IP Block Revision 1 Register (QMAN_IP_REV_1).....	3-86
3.2.4.42	QMan IP Block Revision 2 Register (QMAN_IP_REV_2).....	3-86
3.2.4.43	Data Structure Base Address Registers .....	3-87
3.2.4.43.1	Data Structure Extended Base Address Registers (FQD_BARE).....	3-87
3.2.4.43.2	Data Structure Extended Base Address Registers (PFDR_BARE).....	3-88
3.2.4.43.3	Data Structure Base Address Registers (FQD_BAR) .....	3-88
3.2.4.43.4	Data Structure Base Address Registers (PFDR_BAR) .....	3-89
3.2.4.44	Data Structure Attributes Registers .....	3-89
3.2.4.44.1	Data Structure Attributes Register (FQD_AR).....	3-90
3.2.4.44.2	Data Structure Attributes Register (PFDR_AR).....	3-90
3.2.4.45	QMan Software Portal (QCSP) Base Address Registers.....	3-91
3.2.4.45.1	QCSP Extended Base Address Register (QCSP_BARE).....	3-92
3.2.4.45.2	QCSP Base Address Register (QCSP_BAR) .....	3-92
3.2.4.46	Initiator Scheduling Configuration (CI_SCHED_CFG) .....	3-93
3.2.4.47	QMan Source ID Register (QMAN_SRCIDR) .....	3-93
3.2.4.48	QMan Isolation Context ID Register (QMAN_ICIDR) .....	3-94
3.2.4.49	Initiator Read Latency Monitor Configuration Register (CI_RLM_CFG) .....	3-95
3.2.4.50	Initiator Read Latency Monitor Average (CI_RLM_AVG) .....	3-95
3.2.4.51	QMan Error Interrupt Status Register (QMAN_ERR_ISR).....	3-96
3.2.4.52	QMan Error Interrupt Enable Register (QMAN_ERR_IER) .....	3-99
3.2.4.53	QMan Error Interrupt Status Disable Register (QMAN_ERR_ISDR).....	3-100
3.2.4.54	QMan Error Interrupt Inhibit Register (QMAN_ERR_IIR).....	3-100
3.2.4.55	QMan Error Halt Enable Register (QMAN_ERR_HER).....	3-101
3.3	Functional Description.....	3-101
3.3.1	Frames.....	3-103
3.3.1.1	Frames and the QMan.....	3-103
3.3.1.2	Frame Descriptors (FDs) .....	3-103
3.3.1.2.1	Definition of Frame Descriptors (FDs) .....	3-103
3.3.1.2.2	Structure of Frame Descriptors (FDs) .....	3-103
3.3.1.3	Frame Queues (FQs).....	3-105
3.3.1.3.1	Definition of Frame Queues (FQs).....	3-105
3.3.1.3.2	Frame Queues (FQs) and the QMan.....	3-105
3.3.1.3.3	Structure of Frame Queues (FQs).....	3-105
3.3.1.4	Frame Queue Descriptors (FQDs) .....	3-110
3.3.1.4.1	Definition of a Frame Queue Descriptor (FQD) .....	3-110
3.3.1.4.2	Frame Queue Descriptors (FQDs) and the QMan .....	3-110

# Contents

Paragraph Number	Title	Page Number
3.3.1.4.3	Structure of a Frame Queue Descriptor (FQD) .....	3-111
3.3.1.5	Frame Queue State.....	3-119
3.3.2	Work Queues (WQs) and Channels .....	3-122
3.3.2.1	Definition of a Work Queue (WQ) .....	3-122
3.3.2.2	Definition of a Channel .....	3-122
3.3.2.3	Types of Channels.....	3-122
3.3.2.4	Work Queue Channel Assignments .....	3-123
3.3.3	Enqueue Operations .....	3-124
3.3.4	Dequeue Operations.....	3-124
3.3.4.1	Dequeue Availability .....	3-124
3.3.4.2	Dequeue Commands.....	3-125
3.3.4.3	Dequeueing in the Held Active State .....	3-125
3.3.5	Traffic Class (TC) Flow Control.....	3-126
3.3.6	Frame Queue (FQ) Flow Control.....	3-127
3.3.6.1	FQ Flow Control and FQ States .....	3-127
3.3.6.2	FQ Flow Control Exceptions .....	3-127
3.3.7	Dequeue Scheduling .....	3-128
3.3.7.1	WQ Channel Scheduling .....	3-128
3.3.7.2	WQ Class Scheduling .....	3-128
3.3.7.3	Intra-WQ Class Scheduling .....	3-130
3.3.8	Software Portals .....	3-132
3.3.8.1	Enqueue Command Ring (EQCR).....	3-133
3.3.8.1.1	EQCR Errors.....	3-133
3.3.8.1.2	EQCR Production Notification.....	3-134
3.3.8.1.3	EQCR Consumption Notification.....	3-136
3.3.8.1.4	EQCR Interrupts .....	3-136
3.3.8.1.5	Speculative Reads of the EQCR Cache-Enabled Entries when Using Implicit Production Notification .....	3-137
3.3.8.2	Dequeue Response Ring (DQRR) .....	3-138
3.3.8.2.1	DQRR Production Notification .....	3-139
3.3.8.2.2	DQRR Consumption Notification .....	3-139
3.3.8.2.3	DQRR Dequeue Commands.....	3-141
3.3.8.2.4	DQRR Dequeue Command Portal Selection.....	3-142
3.3.8.2.5	DQRR Dequeue Dispatcher.....	3-143
3.3.8.2.6	Dequeue Dispatcher Operation—Dequeueing from One or More Channels....	3-147
3.3.8.2.7	Dequeue Dispatcher Operation—Dequeueing from a Specific WQ .....	3-150
3.3.8.2.8	Active and Suspended Frame Queues .....	3-155
3.3.8.2.9	Dequeue Command Behavior when No Frames are Available for Dequeue ..	3-156
3.3.8.2.10	Dequeue Command Behavior with Parked and Retired Frame Queues.....	3-156
3.3.8.2.11	DQRR Interrupts.....	3-157
3.3.8.3	Message Ring (MR).....	3-157

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
3.3.8.3.1	MR Production Notification .....	3-158
3.3.8.3.2	MR Consumption Notification .....	3-158
3.3.8.4	Management Command Register (CR) .....	3-159
3.3.8.5	Management Response Registers (RR0/RR1).....	3-160
3.3.8.6	DQRR Entry Stashing.....	3-161
3.3.8.7	EQCR_CI Stashing.....	3-161
3.3.8.8	Dequeued Frame Data, Annotation, and Context Stashing .....	3-162
3.3.8.8.1	Cache warming stashing in an AXI platform .....	3-163
3.3.8.8.2	Cache warming stashing controls .....	3-163
3.3.8.8.3	FQD Context_A Field used for dequeued Frame Data, Annotation, and Context Stashing control .....	3-164
3.3.8.8.4	Dropping of Dequeued Frame Data, Annotation, and FQ Context Stashes ....	3-165
3.3.8.9	Stash Transaction Scheduling .....	3-166
3.3.8.10	Software Portals Virtualization.....	3-166
3.3.9	Software Portal Commands and Responses.....	3-166
3.3.9.1	Enqueue Command.....	3-167
3.3.9.2	Frame Dequeue Response.....	3-171
3.3.9.3	ERN Message Response .....	3-173
3.3.9.4	FQ State Change Notification Message Response .....	3-175
3.3.9.5	Frame Queue Management Commands.....	3-178
3.3.9.5.1	Initialize Frame Queues (FQ) .....	3-178
3.3.9.5.2	Query FQ Programmable Fields.....	3-182
3.3.9.5.3	Query FQ Non-Programmable Fields.....	3-185
3.3.9.5.4	Alter FQ State Commands.....	3-187
3.3.9.5.5	Query WQ Length .....	3-191
3.3.9.6	Congestion Group Management Commands.....	3-193
3.3.9.6.1	Initialize/Modify a CGR .....	3-193
3.3.9.6.2	Query CGR .....	3-196
3.3.9.6.3	Query Congestion State .....	3-199
3.3.9.7	Customer Edge Egress Traffic Management (CEETM) Management Commands..... 3-200	
3.3.9.7.1	CEETM Logical FQ Mapping Table (LFQMT) Configure.....	3-201
3.3.9.7.2	CEETM Logical FQ Mapping Table (LFQMT) Query .....	3-203
3.3.9.7.3	CEETM Class Queue (CQ) Configure .....	3-204
3.3.9.7.4	CEETM Class Queue (CQ) Query .....	3-206
3.3.9.7.5	CEETM Dequeue Context Table (DCT) Configure .....	3-208
3.3.9.7.6	CEETM Dequeue Context Table (DCT) Query .....	3-210
3.3.9.7.7	CEETM Class Scheduler Configure.....	3-211
3.3.9.7.8	CEETM Class Scheduler Query .....	3-213
3.3.9.7.9	CEETM Channel Mapping Configure.....	3-215
3.3.9.7.10	CEETM Channel Mapping Query .....	3-217

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
3.3.9.7.11	CEETM Sub-Portal Mapping Configure .....	3-218
3.3.9.7.12	CEETM Sub-Portal Mapping Query .....	3-220
3.3.9.7.13	CEETM Shaper Configure .....	3-221
3.3.9.7.14	CEETM Shaper Query.....	3-224
3.3.9.7.15	CEETM Traffic Class Flow Control Configure .....	3-226
3.3.9.7.16	CEETM Traffic Class Flow Control Query.....	3-228
3.3.9.7.17	CEETM Class Congestion Group Record (CCGR) CM Configure .....	3-229
3.3.9.7.18	CEETM Class Congestion Group Record (CCGR) CM Query .....	3-232
3.3.9.7.19	CEETM Query Congestion State.....	3-233
3.3.9.7.20	CEETM Class Queue (CQ) Peek / Pop / XSFDR Read .....	3-235
3.3.9.7.21	CEETM Statistics Query/Write .....	3-237
3.3.10	Direct Connect Portals (DCPs) .....	3-239
3.3.11	Algorithmic Sequencers.....	3-240
3.3.11.1	Portal Service Selection.....	3-240
3.3.11.2	Multi-Way Resource Arbiter (MRA).....	3-240
3.3.11.3	Work Queue Semaphore and Context Manager .....	3-241
3.3.12	Frame Queue Descriptor Cache .....	3-241
3.3.12.1	FQD cache operation summary .....	3-241
3.3.12.1.1	On-demand FQD cache eviction .....	3-241
3.3.12.1.2	Post-dequeue FQD cache eviction and Lock in Cache.....	3-242
3.3.13	Order Restoration.....	3-242
3.3.13.1	Position of Order Definition Points within QMan.....	3-243
3.3.13.2	Position of Order Restoration Points within QMan.....	3-243
3.3.13.3	Special Cases for Order Restoration.....	3-244
3.3.13.4	Order Definition Point Implementation.....	3-246
3.3.13.5	Order Restoration Point Implementation.....	3-247
3.3.13.5.1	Order Restoration Point (ORP) Descriptor.....	3-247
3.3.13.5.2	Using Sequence Numbers.....	3-247
3.3.13.5.3	ORP Behavior in Extreme Conditions.....	3-248
3.3.14	Congestion Management and Avoidance .....	3-250
3.3.14.1	Random Early Discard (RED).....	3-251
3.3.14.2	Weighted Random Early Discard (WRED) .....	3-251
3.3.14.3	Congestion Group Tail Drop .....	3-252
3.3.14.4	FQ Tail Drop .....	3-253
3.3.14.5	Enqueue Rejections .....	3-253
3.3.14.6	Congestion State Change Notifications (CSCN) .....	3-254
3.3.14.7	Congestion Group Record (CGR) .....	3-255
3.3.15	Dynamic Debug (DD).....	3-257
3.3.15.1	Data Path Frame Marking.....	3-257
3.3.15.2	Debug Trace Points.....	3-257
3.3.15.3	QMan Dynamic Frame Marking .....	3-258

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
3.3.15.4	Debug Halt.....	3-258
3.3.16	Error Management and Recovery .....	3-260
3.3.17	System Interfaces .....	3-260
3.3.18	System Target interface.....	3-260
3.3.19	System Initiator Interface.....	3-260
3.3.19.1	SrcID and ICID .....	3-261
3.3.19.2	Initiator Scheduling and Priority .....	3-261
3.3.19.3	CPC Stashing of QMan Private Data Structures .....	3-262
3.3.20	Customer Edge Egress Traffic Management (CEETM).....	3-263
3.3.20.1	CEETM Overview .....	3-263
3.3.20.2	CEETM Features .....	3-264
3.3.20.3	Example CEETM scheduling hierarchy for one logical network interface.....	3-265
3.3.20.4	CEETM Class Queue (CQ) .....	3-269
3.3.20.4.1	CEETM Class Queue Descriptor.....	3-270
3.3.20.5	Enqueuing onto a CEETM Class Queue .....	3-272
3.3.20.6	CEETM Class Queue Channel and Class Scheduler.....	3-272
3.3.20.6.1	CEETM Weighted Scheduling among Grouped Classes.....	3-275
3.3.20.7	CEETM Channel Shapers.....	3-276
3.3.20.8	CEETM Channel Scheduler .....	3-277
3.3.20.8.1	CEETM channel scheduler Shaped Fair Queueing (ShFQ) .....	3-279
3.3.20.8.2	CEETM LNI shapers .....	3-280
3.3.20.9	CEETM dequeue requests and scheduling decision flow.....	3-280
3.3.20.10	CEETM dequeue context.....	3-281
3.3.20.11	CEETM Class Congestion Management and Avoidance (CCM).....	3-281
3.3.20.11.1	CEETM Congestion State Change Notifications (CEETM CSCN).....	3-283
3.3.20.11.2	CEETM Class Congestion Group Record (CCGR) .....	3-283
3.3.20.12	CEETM Sub-Portal Mapping and Physical Interface Switchover .....	3-286
3.3.20.13	CEETM traffic class flow control.....	3-287
3.3.20.14	CEETM Configuration Summary.....	3-288
3.3.20.15	Detailed Description of TM Algorithms used in CEETM.....	3-290
3.3.20.15.1	Weighted Bandwidth Fair Scheduling .....	3-290
3.4	Initialize the QMan .....	3-296

## Chapter 4

### Buffer Manager (BMan)

4.1	BMan Overview.....	4-1
4.1.1	BMan Features Overview .....	4-1
4.1.2	BMan Features Summary .....	4-4
4.2	BMan Memory Map and Register Definition.....	4-4
4.2.1	BMan Software Portal Memory Map .....	4-4

# Contents

Paragraph Number	Title	Page Number
4.2.2	BMan Configuration and Control Register Memory Map.....	4-6
4.2.3	BMan Software Portal (BCSP) Register Descriptions .....	4-8
4.2.3.1	BCSP Command Registers (BCSP $n$ _CR) .....	4-9
4.2.3.2	BCSP Response Registers (BCSP $n$ _RR $m$ ).....	4-10
4.2.3.3	BCSP Release Command Ring Registers (BCSP $n$ _RCR $m$ ) .....	4-11
4.2.3.4	BCSP Release Command Ring (RCR) Producer Index Registers (BCSP $n$ _RCR_PI_CENA,BCSP $n$ _RCR_PI_CINH) .....	4-12
4.2.3.5	BCSP RCR Consumer Index Registers (BCSP $n$ _RCR_CI_CENA, BCSP $n$ _RCR_CI_CINH) .....	4-13
4.2.3.6	BCSP RCR Interrupt Threshold Register (BCSP $i$ _RCR_ITR) .....	4-15
4.2.3.7	BCSP Configuration Registers (BCSP $n$ _CFG) .....	4-15
4.2.3.8	BCSP Depletion State Change Interrupt Enable Registers (BCSP $n$ _SCNm) .....	4-16
4.2.3.9	BCSP Functional Interrupt Management Registers (BCSP $n$ _ISR, BCSP $n$ _IER, BCSP $n$ _ISDR, BCSP $n$ _IFR, BCSP $n$ _IIR) .....	4-17
4.2.4	BMan Configuration and Control Register Descriptions .....	4-20
4.2.4.1	Block Revision Registers (BMAN_IP_REV_1, BMAN_IP_REV_2).....	4-20
4.2.4.2	Memory Configuration Registers (FBPR_BAR, FBPR_BARE, FBPR_AR, BMAN_ICIDR, BMAN_SRCID) .....	4-21
4.2.4.3	Hardware and Software Portal Depletion Threshold Registers (BMAN_POOL $n$ _SWDET, BMAN_POOL $n$ _SWDXT) .....	4-23
4.2.4.4	FBPR Pool Low Watermark Interrupt Threshold Register (FBPR_FP_LWIT) ....	4-25
4.2.4.5	Hardware and Software Portal Depletion Count Register (BMAN_POOL $n$ _SDCNT, BMAN_POOL $n$ _HDCNT).....	4-25
4.2.4.6	Pool Content Registers (BMAN_POOL $n$ _CONTENT, FBPR_FPC) .....	4-26
4.2.4.7	Free List Head Pointer Registers (BMAN_POOL $n$ _HDPTR, FBPR_HDPTR) ...	4-27
4.2.4.8	Command Performance Monitor Configuration Registers (CMD_PM $n$ _CFG) ...	4-27
4.2.4.9	Free List Performance Monitor Configuration Register (BMAN_FL_PM $n$ _CFG).....	4-29
4.2.4.10	Idle State and Stop Registers (STATE_IDLE, STATE_STOP) .....	4-29
4.2.4.11	Error Interrupt Management Registers (BMAN_ERR_ISR, BMAN_ERR_IER, BMAN_ERR_ISDR, BMAN_ERR_IFR, BMAN_ERR_IIR) .....	4-31
4.2.4.12	Single Bit ECC Error Threshold Register (BMAN_SBET) .....	4-33
4.2.4.13	Single Bit ECC Error Count Register (BMAN_SBEC0) .....	4-33
4.2.4.14	Error Capture Information Register (BMAN_ECIR) .....	4-34
4.2.4.15	Corruption Error Capture and Address Registers (BMAN_CECR, BMAN_CEAR) ...	4-35
4.2.4.16	Access Error Capture and Address Registers (BMAN_AECR, BMAN_AEAR) .	4-36
4.2.4.17	BMan Debug: Command FIFO Disable Register (DEBUG_CFIFO).....	4-37
4.2.4.18	BMan Debug: State Machine Disable (DEBUG_FSM) .....	4-38
4.3	BMan Functional Description.....	4-38
4.3.1	Direct Connect Portals (DCPs) .....	4-38

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
4.3.2	External Memory Interfaces .....	4-38
4.3.3	Software Portal Components .....	4-38
4.3.3.1	Command Register (CR) Functionality .....	4-39
4.3.3.1.1	Writing a Command into the CR .....	4-39
4.3.3.1.2	Benefits of Using Alternating Polarity .....	4-40
4.3.3.1.3	Acquire and Query Commands .....	4-40
4.3.3.2	Response Register (RR) Functionality .....	4-41
4.3.3.2.1	Determining the Appropriate Response Register .....	4-41
4.3.3.2.2	Acquire Response .....	4-42
4.3.3.2.3	Query Response .....	4-43
4.3.3.2.4	Determining the Expected Polarity of the Valid Bit .....	4-44
4.3.3.3	Release Command Ring (RCR) Functionality .....	4-45
4.3.3.3.1	Issuing a Command Using the RCR .....	4-45
4.3.3.3.2	Release Commands .....	4-47
4.3.4	Interrupts .....	4-49
4.3.4.1	Interrupt Control Logic .....	4-50
4.3.4.2	Error Interrupt Sources .....	4-50
4.3.4.3	Functional Interrupt Source, per Software Portal .....	4-52
4.3.4.4	Reaction to Multi-Bit ECC Interrupts (MBEI) .....	4-53
4.3.4.5	Effect of FBPR Depletion on Release Commands .....	4-53
4.3.5	Buffer Pool State .....	4-53
4.3.5.1	Buffer Availability State .....	4-53
4.3.5.2	Buffer Depletion State .....	4-54
4.3.5.3	Changing Depletion Thresholds .....	4-54
4.3.6	Free Buffer Proxy Records (FBPRs) .....	4-55
4.3.6.1	FBPR Format .....	4-55
4.3.7	Performance Monitor .....	4-56
4.3.7.1	Command Performance Monitoring .....	4-56
4.3.7.2	Free List Fetch/Flush Performance Monitoring .....	4-57
4.4	BMan Initialization Tips .....	4-57

## Chapter 5

### Frame Manager (FMan)

5.1	Frame Manager Overview .....	5-1
5.1.1	FMan Terms, Acronyms, and Abbreviations .....	5-1
5.1.2	FMan Features Summary .....	5-3
5.2	FMan aggregate rate .....	5-5
5.3	Frame Manager High-Level Functional Description .....	5-5
5.3.1	FMan Hardware Ports Types .....	5-6
5.3.2	FMan Network Interfaces .....	5-6

# Contents

Paragraph Number	Title	Page Number
5.3.3	FMan-to-SoC Interfaces .....	5-6
5.3.4	FMan Module Architecture .....	5-7
5.3.5	FMan User-Configurable Pipeline Architecture—Introducing the NIA .....	5-10
5.3.5.1	Packet Flow in the FMan Configurable Pipeline Architecture.....	5-10
5.3.5.2	Role of the NIA in the FMan Configurable Pipeline Architecture.....	5-11
5.3.5.3	Operational mode bits.....	5-11
5.3.6	FMan Multitasking—Introducing the Task and TNUM .....	5-12
5.3.7	Virtual Storage Profiles.....	5-12
5.3.8	ICID .....	5-12
5.3.9	Rx frame replicator .....	5-12
5.3.10	FMan Hardware Ports, QMan sub-portals .....	5-13
5.3.11	FMan Timestamp .....	5-13
5.3.12	Partitioning.....	5-13
5.3.12.1	FMan Memory Map and partitioning .....	5-13
5.3.12.2	FIFO partitioning .....	5-14
5.3.12.3	KeyGen, Policer, storage profile scheme partitioning .....	5-14
5.3.12.4	Custom classifier partitioning.....	5-14
5.3.12.5	Congestion group partitioning .....	5-14
5.3.13	FMan Internal Memory.....	5-15
5.3.13.1	FMan Internal Memory for Tx, Rx, and O/H FIFOs.....	5-15
5.3.13.2	FMan Internal Memory for Custom Classifier .....	5-15
5.3.14	FMan Parser .....	5-16
5.3.15	FMan Classification, Distribution.....	5-16
5.3.16	FMan Policer.....	5-17
5.3.17	DPAA congestion management .....	5-18
5.3.17.1	Congestion management on Rx .....	5-18
5.3.18	Congestion management on Tx .....	5-18
5.3.19	FMan Functional Flows .....	5-18
5.3.19.1	Receive (Rx) Flows .....	5-19
5.3.19.1.1	Configurable Receive (Rx) Flows .....	5-19
5.3.19.1.2	Receive (Rx) Flow Example with Custom Classifier.....	5-21
5.3.19.2	Transmit (Tx) Flow Example .....	5-25
5.3.19.3	FMan Offline Port Flow Example .....	5-27
5.3.19.4	Independent Mode (IM) Flow .....	5-28
5.3.19.5	Host Command Flow .....	5-29
5.4	Frame Manager Detailed Functional Description.....	5-31
5.4.1	FMan Hardware Ports.....	5-31
5.4.1.1	FMan PortIDs .....	5-32
5.4.2	FMan Detailed Memory Map .....	5-33
5.4.2.1	Hardware Port Pages in the FMan Memory Map .....	5-34
5.4.2.2	Examples for the Evaluation of Addresses of FMan Registers .....	5-36

# Contents

Paragraph Number	Title	Page Number
5.4.2.2.1	Calculate the SoC Absolute Address for an FMan Register .....	5-36
5.4.2.2.2	Calculate the SoC Absolute Address for FMan Hardware Port Page Register .	5-37
5.4.3	FMan Frame Internal Context (IC) .....	5-37
5.4.3.1	Override IC from FQD or Data Buffer .....	5-39
5.4.3.1.1	Frame Queue Descriptor (FQD) Context A .....	5-39
5.4.3.1.2	Frame Queue Descriptor Context B .....	5-44
5.4.3.2	Frame Descriptor (FD) .....	5-45
5.4.3.2.1	FD Command/Status Word.....	5-46
5.4.3.3	Internal Context Action Descriptor (ICAD) .....	5-57
5.4.3.3.1	Internal Context Action Descriptor (ICAD) for Rx.....	5-58
5.4.3.3.2	Internal Context Action Descriptor (ICAD) for Tx or Offline Port .....	5-58
5.4.4	Next Invoked Action (NIA) .....	5-59
5.4.4.1	NIA Register Configuration—Rx Flows .....	5-61
5.4.5	FMan Debug .....	5-63
5.4.5.1	FMan Debug Overview .....	5-63
5.4.5.2	FMan Debug Features Summary .....	5-63
5.4.5.3	FMan Debug Glossary .....	5-63
5.4.5.4	FMan Packet Debug .....	5-64
5.4.5.4.1	Flow Debug .....	5-64
5.4.5.4.2	Trace .....	5-65
5.4.5.4.3	Debug Traps.....	5-66
5.4.5.5	Debug Programming Model .....	5-66
5.4.5.5.1	Generic Debug Control Register Model .....	5-66
5.4.5.5.2	Generic Debug Trap Configuration Register (GDTCR) Model .....	5-68
5.4.5.5.3	Generic Debug Value Register (DVR) Model .....	5-69
5.4.5.5.4	Generic Debug Trap Mask Register Model.....	5-69
5.4.5.6	DMA Trace Write .....	5-69
5.4.6	Implementing Statistics Counters .....	5-70
5.4.7	Error Handling .....	5-70
5.4.7.1	Local Catastrophic Errors .....	5-70
5.4.7.2	Serious Errors .....	5-71
5.4.7.3	Operational Errors .....	5-71
5.4.8	Configuration Register Access .....	5-71
5.4.9	Graceful Stop .....	5-72
5.4.9.1	Tx Port Graceful Stop .....	5-72
5.4.9.2	Rx Port Graceful Stop .....	5-72
5.4.10	Dynamic Changes .....	5-72
5.5	Frame Manager—Buffer Manager Interface (BMI) .....	5-73
5.5.1	Frame Manager BMI Introduction.....	5-73
5.5.1.1	Acronyms.....	5-73
5.5.1.2	Terminology .....	5-74

# Contents

Paragraph Number	Title	Page Number
5.5.2	Frame Manager BMI Overview.....	5-75
5.5.2.1	Frame Manager BMI Features.....	5-75
5.5.2.2	Frame Manager BMI Modes of Operation .....	5-78
5.5.3	BMI Input NIA .....	5-78
5.5.4	Frame Manager BMI Memory Map and Register Definition.....	5-80
5.5.4.1	Storage Profiles.....	5-87
5.5.4.1.1	Storage Profile Virtualization per port (for Rx and Offline ports) .....	5-88
5.5.4.1.2	Hardware port storage profiles .....	5-89
5.5.4.1.3	Virtual storage profiles .....	5-90
5.5.4.2	Congestion Group Priority Mapping table .....	5-93
5.5.4.3	PFC priority mapping to QMan traffic classes .....	5-93
5.5.4.4	FMan BMI Common Registers Description.....	5-93
5.5.4.4.1	BMI Initialization Register (FMBM_INIT) .....	5-93
5.5.4.4.2	BMI Configuration 1 Register (FMBM_CFG1) .....	5-94
5.5.4.4.3	BMI Configuration 2 Register (FMBM_CFG2) .....	5-96
5.5.4.4.4	Interrupt Event Register (FMBM_IER).....	5-96
5.5.4.4.5	Interrupt Enable Register (FMBM_IER).....	5-97
5.5.4.4.6	Interrupt Force Register (FMBM_IFR).....	5-98
5.5.4.4.7	Debug Trap Counter Register (FMBM_DTC) .....	5-99
5.5.4.4.8	Debug Compare Value Register (FMBM_DCV) .....	5-99
5.5.4.4.9	Debug Compare Mask (FMBM_DCM) .....	5-100
5.5.4.4.10	Global Debug Enable (FMBM_GDE).....	5-100
5.5.4.4.11	Port Parameters Register (FMBM_PP_1–63) .....	5-101
5.5.4.4.12	Port FIFO Size Register (FMBM_PFS_1–63) .....	5-103
5.5.4.4.13	SPICID Register (FMBM_SPICID_1–63).....	5-106
5.5.4.5	Rx Port Register Descriptions .....	5-108
5.5.4.5.1	Rx Configuration Register (FMBM_RCFG).....	5-108
5.5.4.5.2	Rx Status Register (FMBM_RST).....	5-109
5.5.4.5.3	Rx DMA Attributes Register (FMBM_RDA).....	5-110
5.5.4.5.4	Rx FIFO Parameters Register (FMBM_RFP).....	5-112
5.5.4.5.5	Rx Frame End Data Register (FMBM_RFED) .....	5-113
5.5.4.5.6	Rx Internal Context Parameters (FMBM_RICP) .....	5-115
5.5.4.5.7	Rx Internal Margins Register (FMBM_RIM) .....	5-116
5.5.4.5.8	Rx External Buffer Margins Register (FMBM_REBM) .....	5-117
5.5.4.5.9	Rx Frame Next Engine Register (FMBM_RFNE) .....	5-118
5.5.4.5.10	Rx Frame Attributes Register (FMBM_RFCA) .....	5-119
5.5.4.5.11	Rx Frame Parser Next Engine Register (FMBM_RFPNE) .....	5-120
5.5.4.5.12	Rx Parsing Start Offset Register (FMBM_RPSO) .....	5-121
5.5.4.5.13	Rx Policer Profile Register (FMBM_RPP) .....	5-121
5.5.4.5.14	Rx Parse Result Initialization Register (FMBM_RPRI) .....	5-122
5.5.4.5.15	Rx Frame Queue ID Register (FMBM_RFQID).....	5-123

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.5.4.5.16	Rx Error Frame Queue ID Register (FMBM_REFQID).....	5-124
5.5.4.5.17	Rx Frame Status Discard Mask Register (FMBM_RFSDM).....	5-125
5.5.4.5.18	Rx Frame Status Error Mask Register (FMBM_RFSEM) .....	5-125
5.5.4.5.19	Rx Frame Enqueue Next Engine Register (FMBM_RFENE).....	5-126
5.5.4.5.20	Rx Continuous Mode Next Enqueue Register (FMBM_RCMNE).....	5-127
5.5.4.5.21	Rx External Buffers Manager Pool Information Register (FMBM_REBMPI)..... 5-127	
5.5.4.5.22	Rx Allocate Counter Register (FMBM_RACNT).....	5-129
5.5.4.5.23	Receive Congestion Group Map Register (FMBM_RCGM).....	5-130
5.5.4.5.24	BMan Pool Depletion Register (FMBM_RMPD).....	5-131
5.5.4.5.25	Statistics Counters Register (FMBM_RSTC) .....	5-132
5.5.4.5.26	Rx Frame Counter Register (FMBM_RFRC) .....	5-133
5.5.4.5.27	Rx Bad Frames Counter Register (FMBM_RBFC) .....	5-133
5.5.4.5.28	Rx Large Frames Counter Register (FMBM_RLFC).....	5-134
5.5.4.5.29	Rx Filter Frames Counter Register (FMBM_RFFC) .....	5-135
5.5.4.5.30	Rx Frames Discard Counter Register (FMBM_RFDC) .....	5-135
5.5.4.5.31	Rx Frames List DMA Error Counter Register (FMBM_RFLDEC).....	5-136
5.5.4.5.32	Rx Out of Buffers Discard Counter Register (FMBM_RODC).....	5-137
5.5.4.5.33	Rx Buffers Deallocate Counter Register (FMBM_RBDC).....	5-138
5.5.4.5.34	RX Prepare to Enqueue Counter (FMBM_RPEC).....	5-138
5.5.4.5.35	Rx Performance Counters Register (FMBM_RPC) .....	5-139
5.5.4.5.36	Rx Performance Count Parameters Register (FMBM_RPCP).....	5-139
5.5.4.5.37	Rx Cycle Counter Register (FMBM_RCCN) .....	5-141
5.5.4.5.38	Rx Tasks Utilization Counter Register (FMBM_RTUC) .....	5-142
5.5.4.5.39	Rx Receive Queue Utilization Counter Register (FMBM_RRQUC) .....	5-142
5.5.4.5.40	Rx DMA Utilization Counter Register (FMBM_RDUC) .....	5-143
5.5.4.5.41	Rx FIFO Utilization Counter Register (FMBM_RFUC) .....	5-144
5.5.4.5.42	Rx Pause Activation Counter Register (FMBM_RPAC) .....	5-144
5.5.4.5.43	Rx Debug Configuration Register (FMBM_RDCFG) .....	5-145
5.5.4.5.44	Rx General Purpose Register (FMBM_RGPR).....	5-147
5.5.4.6	Tx Port Register Descriptions.....	5-147
5.5.4.6.1	Tx Configuration Register (FMBM_TCFG) .....	5-147
5.5.4.6.2	Tx Status Register (FMBM_TST) .....	5-148
5.5.4.6.3	Tx DMA Attributes Register (FMBM_TDA) .....	5-149
5.5.4.6.4	Tx FIFO Parameters Register (FMBM_TFP) .....	5-149
5.5.4.6.5	Tx Frame End Data Register (FMBM_TFED).....	5-151
5.5.4.6.6	Tx Internal Context Parameters Register (FMBM_TICP) .....	5-152
5.5.4.6.7	Tx Frame Dequeue Next Engine Register (FMBM_TFDNE) .....	5-153
5.5.4.6.8	Tx Frame Attributes Register (FMBM_TFCA) .....	5-154
5.5.4.6.9	Tx Confirmation Frame Queue ID Register (FMBM_TCFQID).....	5-155
5.5.4.6.10	Tx Error Frame Queue ID Register (FMBM_TEFQID) .....	5-156

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.5.4.6.11	Tx Frame Enqueue Next Engine Register (FMBM_TFENE) .....	5-156
5.5.4.6.12	Tx Rate Limiter Scale Register (FMBM_TRLMTS) .....	5-157
5.5.4.6.13	Tx Rate Limiter Register (FMBM_TRLMT) .....	5-158
5.5.4.6.14	Tx Custom Classifier Base Register (FMBM_TCCB) .....	5-159
5.5.4.6.15	Tx Frame Next Engine Register (FMBM_TFNE) .....	5-160
5.5.4.6.16	Tx Priority based Flow Control (PFC) Mapping Registers (FMBM_TPFCM0) .....	5-160
5.5.4.6.17	Tx Continuous Mode Next Enqueue Register (FMBM_TCMNE) .....	5-161
5.5.4.6.18	Tx Statistics Counters Register (FMBM_TSTC) .....	5-162
5.5.4.6.19	Tx Frame Counter Register (FMBM_TFRC) .....	5-163
5.5.4.6.20	Tx Frames Discard Counter Register (FMBM_TFDC) .....	5-163
5.5.4.6.21	Tx Frames Length Error Discard Counter Register (FMBM_TFLEDC) .....	5-164
5.5.4.6.22	Tx Frames Unsupported Format Discard Counter Register (FMBM_TFUFD) .....	5-165
5.5.4.6.23	Tx Buffers Deallocate Counter Register (FMBM_TBDC) .....	5-165
5.5.4.6.24	Tx Performance Counters Register (FMBM_TPC) .....	5-166
5.5.4.6.25	Tx Performance Count Parameters Register (FMBM_TPCP) .....	5-166
5.5.4.6.26	Tx Cycle Counter Register (FMBM_TCCN) .....	5-168
5.5.4.6.27	Tx Tasks Utilization Counter Register (FMBM_TTUC) .....	5-169
5.5.4.6.28	Tx Transmit Confirm Queue Utilization Counter Register (FMBM_TTCQUC) .....	5-169
5.5.4.6.29	Tx DMA Utilization Counter Register (FMBM_TDUC) .....	5-170
5.5.4.6.30	Tx FIFO Utilization Counter Register (FMBM_TFUC) .....	5-171
5.5.4.6.31	Tx Debug Configuration Register (FMBM_TDCFG) .....	5-171
5.5.4.6.32	Tx General Purpose Register (FMBM_TGPR) .....	5-173
5.5.4.7	Offline Port/Host Command Port Registers Description .....	5-174
5.5.4.7.1	Offline Port/Host Command (O/H) Configuration Register (FMBM_OCFG) .....	5-174
5.5.4.7.2	O/H Status Register (FMBM_OST) .....	5-175
5.5.4.7.3	O/H DMA Attributes Register (FMBM_ODA) .....	5-176
5.5.4.7.4	O/H Internal Context Parameters Register (FMBM_OICP) .....	5-177
5.5.4.7.5	O/H Frame Dequeue Next Engine Register (FMBM_OFDNE) .....	5-179
5.5.4.7.6	O/H Frame Next Engine Register (FMBM_OFNE) .....	5-179
5.5.4.7.7	O/H Frame Attributes Register (FMBM_OFCA) .....	5-180
5.5.4.7.8	O/H Frame Parser Next Engine Register (FMBM_OFPNE) .....	5-181
5.5.4.7.9	O/H Parsing Start Offset Register (FMBM_OPSO) .....	5-182
5.5.4.7.10	O/H Policer Profile Register (FMBM_OPP) .....	5-182
5.5.4.7.11	O/H Custom Classifier Base Register (FMBM_OCCB) .....	5-183
5.5.4.7.12	O/H Internal Margins Register (FMBM_OIM) .....	5-183
5.5.4.7.13	O/H FIFO Parameters Register (FMBM_OFP) .....	5-184
5.5.4.7.14	O/H Frame End Data Register (FMBM_OFED) .....	5-185
5.5.4.7.15	O/H Parse Result Initialization Register (FMBM_OPRI) .....	5-186

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.5.4.7.16	O/H Frame Queue ID Register (FMBM_OFQID) .....	5-187
5.5.4.7.17	O/H Error Frame Queue ID Register (FMBM_OEFQID) .....	5-188
5.5.4.7.18	O/H Frame Status Discard Mask Register (FMBM_OFSDM) .....	5-189
5.5.4.7.19	O/H Frame Status Error Mask Register (FMBM_OFSEM).....	5-189
5.5.4.7.20	O/H Frame Enqueue Next Engine Register (FMBM_OFENE) .....	5-190
5.5.4.7.21	O/H Rate Limiter Scale Register (FMBM_ORLMTS).....	5-191
5.5.4.7.22	O/H Rate Limiter Register (FMBM_ORLMT) .....	5-192
5.5.4.7.23	O/H Continuous Mode Next Enqueue Register (FMBM_OCMNE) .....	5-193
5.5.4.7.24	FMBM_OCGM - Observed Congestion Group Map.....	5-194
5.5.4.7.25	O/H Statistics Counters Register (FMBM_OSTC) .....	5-195
5.5.4.7.26	O/H Frame Counter Register (FMBM_OFRC).....	5-196
5.5.4.7.27	O/H Frames Discard Counter Register (FMBM_OFDC).....	5-196
5.5.4.7.28	O/H Frames Length Error Discard Counter Register (FMBM_OFLEDC)....	5-197
5.5.4.7.29	O/H Frames Unsupported Format Discard Counter Register (FMBM_OFUFDC) ..	5-198
5.5.4.7.30	O/H Filter Frames Counter Register (FMBM_OFFC) .....	5-198
5.5.4.7.31	O/H Frames WRED Discard Counter Register (FMBM_OFWDC) .....	5-199
5.5.4.7.32	O/H Frames List DMA Error Counter Register (FMBM_OFLDEC) .....	5-200
5.5.4.7.33	O/H Buffers Deallocate Counter Register (FMBM_OBDC) .....	5-201
5.5.4.7.34	O/H Out of Buffers Discard Counter Register (FMBM_OODC) .....	5-201
5.5.4.7.35	O/H Prepare to Enqueue Counter (FMBM_OPEC) .....	5-202
5.5.4.7.36	O/H Performance Counters Register (FMBM_OPC).....	5-203
5.5.4.7.37	O/H Performance Count Parameters Register (FMBM_OPCP) .....	5-203
5.5.4.7.38	O/H Cycle Counter Register (FMBM_OCCN) .....	5-205
5.5.4.7.39	O/H Tasks Utilization Counter Register (FMBM_OTUC) .....	5-205
5.5.4.7.40	O/H DMA Utilization Counter Register (FMBM_ODUC).....	5-206
5.5.4.7.41	O/H FIFO Utilization Counter Register (FMBM_OFUC) .....	5-207
5.5.4.7.42	O/H Debug Configuration Register (FMBM_ODCFG).....	5-207
5.5.4.7.43	O/H General Purpose Register (FMBM_OGPR) .....	5-209
5.5.5	Frame Manager BMI Functional Description.....	5-209
5.5.5.1	Introduction to BMI Functional Description .....	5-209
5.5.5.2	Introduction to BMI Data flows .....	5-210
5.5.5.3	BMI Rx Flow—Normal Mode .....	5-211
5.5.5.3.1	Initialization.....	5-211
5.5.5.3.2	BMI ‘Rx Frame’ .....	5-212
5.5.5.3.3	Rx BMI ‘Discard Frame’ or ‘Prepare to Enqueue Frame’ .....	5-212
5.5.5.3.4	BMI ‘Release Internal Buffers’ .....	5-213
5.5.5.4	BMI Rx Flow—Independent Mode .....	5-214
5.5.5.5	BMI Tx Flow—Normal Mode .....	5-214
5.5.5.5.1	Tx BMI ‘Allocate internal IC’ and QMI dequeue .....	5-214
5.5.5.5.2	BMI ‘Transmit Frame’ or ‘Process and Transmit Frame’ .....	5-215

# Contents

Paragraph Number	Title	Page Number
5.5.5.6	BMI Tx Flow—Independent Mode .....	5-216
5.5.5.7	BMI Offline Port Flow .....	5-216
5.5.5.7.1	Initialization.....	5-216
5.5.5.7.2	Offline BMI ‘Allocate internal IC’ and QMI dequeue .....	5-216
5.5.5.7.3	BMI Offline ‘frame fetch’ .....	5-217
5.5.5.7.4	BMI Offline ‘Discard Frame’ or ‘Prepare to Enqueue Frame’ .....	5-218
5.5.5.7.5	BMI Offline ‘Release Internal Buffers’ .....	5-219
5.5.5.7.6	Illegal combinations .....	5-219
5.5.5.8	BMI Host Command Flow .....	5-220
5.5.6	BMI Internal Operation Details .....	5-220
5.5.6.1	Operational Mode bits .....	5-220
5.5.6.2	Buffer Pool Selection for Rx and O/H.....	5-221
5.5.6.2.1	Backup Pools .....	5-221
5.5.6.3	Restrictions on a scatter/gather list .....	5-221
5.5.6.4	Internal and External Margins .....	5-222
5.5.6.4.1	Internal Margins.....	5-222
5.5.6.4.2	External Margins .....	5-222
5.5.6.4.3	External memory accesses optimization.....	5-224
5.5.6.5	Conditions that enable Tx checksum generation .....	5-224
5.5.6.6	Conditions that enable Offline checksum validation.....	5-225
5.5.6.7	Transmit Confirmation Type (TXCT) .....	5-225
5.5.6.8	Rx Normal mode and Offline - Enqueue or discard decision logic .....	5-226
5.5.6.9	Tx confirmation enqueue and buffer deallocation decision.....	5-228
5.5.6.10	Offline Port Enqueue Decision Flow.....	5-231
5.5.6.11	Host command enqueue decision flow .....	5-232
5.5.6.12	DMA resource Load balancing.....	5-232
5.5.6.13	FMan DMA Priority Elevation.....	5-233
5.5.6.14	ICID (Isolation Context Identifier).....	5-233
5.5.6.15	Congestion or Rejection Handling.....	5-234
5.5.6.16	Pause Frames for Flow Control .....	5-234
5.5.6.17	Tx and O/H Rate Limiter .....	5-235
5.5.6.18	Internal FIFO Configuration Requirements.....	5-236
5.5.6.18.1	Internal FIFO for Rx Ports.....	5-236
5.5.6.18.2	Internal FIFO for Tx Ports .....	5-237
5.5.6.18.3	Internal FIFO for O/H Ports .....	5-237
5.5.6.19	Hardware Assist for IEEE 1588-Compliant Timestamping .....	5-237
5.5.6.20	Error Handling .....	5-238
5.5.6.20.1	Non-Recoverable Errors .....	5-238
5.5.6.20.2	Network Errors .....	5-238
5.5.6.20.3	Buffer Depletion .....	5-239
5.5.6.20.4	Queue Error .....	5-239

# Contents

Paragraph Number	Title	Page Number
5.5.6.20.5	Unsupported Frames.....	5-239
5.5.6.20.6	Timestamp Errors .....	5-240
5.5.6.20.7	DMA Error .....	5-240
5.5.6.21	Graceful Stop .....	5-241
5.5.6.22	Debug Capabilities.....	5-241
5.5.6.22.1	Flow Initialization.....	5-241
5.5.6.22.2	Flow Trap.....	5-241
5.5.6.22.3	Flow Trace .....	5-242
5.5.6.22.4	Trap Action .....	5-243
5.5.7	Frame Manager BMI Initialization/Application Information .....	5-243
5.5.7.1	Initialization of Common Registers.....	5-243
5.5.7.2	Rx Port Initialization Steps in Normal Mode .....	5-244
5.5.7.3	Rx Port Initialization Steps in Independent Mode.....	5-245
5.5.7.4	Tx Port Initialization Steps in Normal Mode (Example).....	5-245
5.5.7.5	Tx Port Initialization Steps in Independent Mode .....	5-245
5.5.7.6	Initialization Steps for Offline Ports .....	5-246
5.5.7.7	Initialization Steps of Host Command Ports .....	5-246
5.6	Frame Manager—Queue Manager Interface (QMI).....	5-247
5.6.1	QMI Overview .....	5-247
5.6.1.1	QMI Features Summary .....	5-247
5.6.2	QMI Input NIA .....	5-248
5.6.3	QMI Memory Map and Register Definition .....	5-248
5.6.3.1	Register offsets .....	5-249
5.6.3.2	QMI Register Descriptions .....	5-252
5.6.3.2.1	QMI General Configuration Register (FMQM_GC).....	5-252
5.6.3.2.2	Error Interrupt Event Register (FMQM_EIE) .....	5-252
5.6.3.2.3	Error Interrupt Enable Register (FMQM_EIEN) .....	5-253
5.6.3.2.4	Error Interrupt Force Register (FMQM{EIF}) .....	5-254
5.6.3.2.5	Global Status Register (FMQM_GS) .....	5-255
5.6.3.2.6	Enqueue Total Frame Counter Register (FMQM_ETFC) .....	5-256
5.6.3.2.7	Dequeue Total Frame Counter Register (FMQM_DTFC) .....	5-256
5.6.3.2.8	Dequeue Counter 0 Register (FMQM_DC0) .....	5-257
5.6.3.2.9	Debug Trace Configuration Register (FMQM_DTRC) .....	5-257
5.6.3.2.10	Enqueue Frame Descriptor Dynamic Debug Register (FMQM_EFDDD) .....	5-258
5.6.3.2.11	Debug Trap Configuration <i>n</i> Register (FMQM_DTC <i>n</i> ) .....	5-259
5.6.3.2.12	Debug Trap Value <i>n</i> Register (FMQM_DTV <i>n</i> ).....	5-261
5.6.3.2.13	Debug Trap Mask <i>n</i> Register (FMQM_DTM <i>n</i> ) .....	5-261
5.6.3.2.14	Debug Trap Counter <i>n</i> Register (FMQM_DTC <i>n</i> ) .....	5-261
5.6.3.2.15	PortID <i>n</i> Configuration Register (FMQM_PnC).....	5-262
5.6.3.2.16	PortID <i>n</i> Status Register (FMQM_PnS) .....	5-263
5.6.3.2.17	PortID <i>n</i> Task Status Register (FMQM_PnTS) .....	5-263

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.6.3.2.18	PortID <i>n</i> Enqueue NIA Register (FMQM_PnEN) .....	5-264
5.6.3.2.19	PortID <i>n</i> Enqueue NIA Register (FMQM_PnEN) .....	5-265
5.6.3.2.20	PortID <i>n</i> Enqueue Total Frame Counter Register (FMQM_PnETFC).....	5-266
5.6.3.2.21	PortID <i>n</i> Dequeue NIA Register (FMQM_PnDN).....	5-266
5.6.3.2.22	PortID <i>n</i> Dequeue NIA Register (FMQM_PnDN).....	5-267
5.6.3.2.23	PortID <i>n</i> Dequeue Config Registers (FMQM_PnDC) .....	5-267
5.6.3.2.24	PortID <i>n</i> Dequeue Total Frame Counter Register (FMQM_PnDTFC) .....	5-269
5.6.3.2.25	PortID <i>n</i> Dequeue FQID Not Override Counter Register (FMQM_PnDFNOC)..... 5-270	
5.6.3.2.26	PortID <i>n</i> Dequeue Confirm Counter Register (FMQM_PnDCC) .....	5-270
5.6.4	QMI Functional Description.....	5-270
5.6.4.1	Enqueue Operation .....	5-270
5.6.4.2	Dequeue Operation .....	5-271
5.6.4.3	FMan Hardware Ports, QMan sub-portals - details .....	5-271
5.6.4.4	QMI parameters which affect performance .....	5-272
5.6.4.5	FMan Software Reset Operation .....	5-273
5.6.4.6	QMI Graceful Stop Operation .....	5-273
5.6.4.7	QMI Debug Functionality.....	5-273
5.6.4.7.1	Packet Processing Flow Debug .....	5-273
5.6.4.7.2	QMI Performance Monitoring.....	5-276
5.6.5	QMI Initialization Sequence .....	5-277
5.6.5.1	Initializing the Common QMI Registers .....	5-277
5.6.5.2	Initializing a Specific PortID .....	5-277
5.6.5.2.1	Initializing an Rx Port.....	5-277
5.6.5.2.2	Initializing an Tx/Host Command/Offline Parsing Port.....	5-278
5.7	Frame Manager—Frame Processing Manager .....	5-279
5.7.1	FPM Overview.....	5-279
5.7.1.1	Frames and the FPM.....	5-279
5.7.2	FPM Features .....	5-279
5.7.3	Frame Manager—FPM Memory Map and Register Definition .....	5-280
5.7.3.1	FPM PortID Control Register (FMFP_PRC) .....	5-281
5.7.3.2	FPM Maximum Dispatches Register (FMFP_MXD).....	5-282
5.7.3.3	FPM Dispatch Thresholds1 Register (FMFP_DIST1) .....	5-283
5.7.3.4	FPM Dispatch Thresholds2 Register (FMFP_DIST2) .....	5-284
5.7.3.5	FMan Error Pending Interrupt Register (FM_EPI) .....	5-284
5.7.3.6	FMan Rams Interrupt Enable Register (FM_RIE) .....	5-288
5.7.3.7	FPM FMan Controller Event 0-3 Registers (FMFP_FCEV).....	5-288
5.7.3.8	FPM FMan Controller Event Enable0-3 Registers (FMFP_CEE) .....	5-289
5.7.3.9	FPM TimeStamp Control1 Register (FMFP_TSC1) .....	5-290
5.7.3.10	FPM TimeStamp Control2 Register (FMFP_TSC2) .....	5-290
5.7.3.11	FPM Time Stamp Register (FMFP_TSP).....	5-291

# Contents

Paragraph Number	Title	Page Number
5.7.3.12	FPM Time Stamp Fraction Register (FMFP_TS <sub>F</sub> ) .....	5-291
5.7.3.13	FMan Rams Control and Event Register (FM_RCR).....	5-292
5.7.3.14	FPM External Requests Control Register (FMFP_EXTC) .....	5-293
5.7.3.15	FPM Data Ram Data0-3 Register (FMFP_DRD <sub>n</sub> ).....	5-293
5.7.3.16	FM Double ECC Error Source Register (FM_DECSES) .....	5-294
5.7.3.17	FPM Data RAM Access Register (FMFP_DRA).....	5-296
5.7.3.18	FMan IP Block Revision 1 Register (FM_IP_REV_1) .....	5-296
5.7.3.19	FMan IP Block Revision 2 Register (FM_IP_REV_2) .....	5-297
5.7.3.20	FMan Reset Command Register (FM_RSTC) .....	5-298
5.7.3.21	FMan Controller Debug Control Register (FMFP_CLDC).....	5-298
5.7.3.22	FMan Normal Pending Interrupt Register (FM_NPI) .....	5-301
5.7.3.23	FPM Event and Enable Register (FMFP_EE) .....	5-304
5.7.3.24	FPM CPU Event 0-3 Registers (FMFP_CEvn) .....	5-305
5.7.3.25	FPM PortID Status 0-49 Registers (FMFP_PSn) .....	5-306
5.7.3.26	FMan Controller Flow AB Control Register (FMFP_CLFABC).....	5-307
5.7.3.27	FMan Controller Flow C Control Register (FMFP_CLFCC) .....	5-308
5.7.3.28	FMan Controller Flow A Value Register (FMFP_CLFAVAL).....	5-309
5.7.3.29	FMan Controller Flow B Value Register (FMFP_CLFBVAL) .....	5-309
5.7.3.30	FMan Controller Flow C Value Register (FMFP_CLFCVAL) .....	5-310
5.7.3.31	FMan Controller Flow A Mask Register (FMFP_CLFAMSK) .....	5-310
5.7.3.32	FMan Controller Flow B Mask Register (FMFP_CLFBMSK).....	5-311
5.7.3.33	FMan Controller Flow C Mask Register (FMFP_CLFCMSK).....	5-311
5.7.3.34	FMan Controller Flow A Match Count Register (FMFP_CLFAMC).....	5-312
5.7.3.35	FMan Controller Flow B Match Count Register (FMFP_CLFBMC) .....	5-312
5.7.3.36	FMan Controller Flow C Match Count Register (FMFP_CLFCMC) .....	5-313
5.7.3.37	FM_DECCEH - FM Double ECC Error Halt Register .....	5-313
5.7.3.38	FPM TNUM Status 0-127 Registers (FMFP_TS <sub>n</sub> ) .....	5-315
5.7.4	Functional Description.....	5-317
5.7.4.1	Order Definition and Restoration .....	5-317
5.7.4.2	Timestamp and Prescale (TSP).....	5-317
5.7.4.2.1	TSP Example .....	5-318
5.7.5	Initialization Information.....	5-318
5.8	Frame Manager—DMA .....	5-319
5.8.1	FMan DMA Overview .....	5-319
5.8.2	FMan DMA Feature Summary .....	5-319
5.8.3	Introduction about the FMan DMA .....	5-320
5.8.4	FMan DMA Memory Map/Register Definition.....	5-320
5.8.4.1	FMan DMA Status Register (FMDM_SR).....	5-321
5.8.4.2	FMan DMA Mode Register (FMDM_MR).....	5-322
5.8.4.3	FMan DMA Threshold Register (FMDM_TR).....	5-324
5.8.4.4	FMan DMA Hysteresis Registers (FMDM_HY) .....	5-326

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.8.4.5	FMan DMA SOS Emergency Threshold Register (FMDM_SETR).....	5-327
5.8.4.6	FMan DMA Transfer Address High Register (FMDM_TAH).....	5-327
5.8.4.7	FMan DMA Transfer Address Low Register (FMDM_TAL).....	5-328
5.8.4.8	FMan DMA Transfer Communication ID Register (FMDM_TCID) .....	5-328
5.8.4.9	FMan DMA buffer Watchdog Counter Value (FMDM_WCR).....	5-329
5.8.4.10	FMan DMA buffer Base in FMan Memory Value Register (FMDM_EBCR)....	5-330
5.8.4.11	FMan DMA Debug Counter (FMDM_DCR).....	5-330
5.8.4.12	FMan DMA Emergency Smoother Register (FMDM_EMSR).....	5-330
5.8.4.13	FMan DMA PortID Registers (FMDM_PortIDn).....	5-331
5.8.5	FMan DMA Functional Description.....	5-332
5.8.5.1	Bus Error.....	5-332
5.8.5.2	ECC Error .....	5-333
5.8.5.3	Halt .....	5-333
5.8.5.4	Reset .....	5-333
5.8.6	FMan DMA Initialization .....	5-333
5.8.7	FMan DMA Debug Functionality.....	5-333
5.9	Frame Manager—Parser .....	5-335
5.9.1	Parser Overview.....	5-335
5.9.2	Parser Features Summary .....	5-335
5.9.3	Parser Memory Map/Register Definition .....	5-336
5.9.3.1	Parser Register Descriptions.....	5-341
5.9.3.1.1	Port <i>x</i> Parse Memory Direct Access Registers .....	5-341
5.9.3.1.2	Parse Memory Read/Write Direct Access Registers—Global Configuration .	5-343
5.9.3.1.3	Interrupt Registers .....	5-346
5.9.3.1.4	Parse Statistic Registers .....	5-349
5.9.3.1.5	Parser Debug Registers.....	5-356
5.9.4	Parser Functional Description.....	5-359
5.9.4.1	Benefits of Using Hard Header Examination Sequences (HXSs) .....	5-359
5.9.4.2	Benefits of Using Soft Examination Sequences .....	5-360
5.9.4.3	Parse Array Functionality Overview .....	5-360
5.9.4.4	Parser Inputs/Outputs.....	5-360
5.9.4.4.1	Possible Parser NIA Action Codes .....	5-360
5.9.4.4.2	Parser Inputs/Outputs from/to FMan Memory .....	5-361
5.9.4.4.3	Purpose of the Status Region of the Frame Descriptor (FD[STATUS]).....	5-362
5.9.4.4.4	Hard Parser Next Instruction Address (HPNIA) .....	5-363
5.9.4.4.5	FMan Controller Activation for Parsing.....	5-364
5.9.4.5	Parse Tree and Hard Header Examination Sequences (HXSs).....	5-364
5.9.4.5.1	What is a Parse Shell? .....	5-366
5.9.4.5.2	Soft Sequence Attachment.....	5-366
5.9.4.5.3	Line-up Enable Confirmation Mask .....	5-371
5.9.4.5.4	Line-Up Confirmation Vector.....	5-371

# Contents

Paragraph Number	Title	Page Number
5.9.4.6	Parse Array .....	5-372
5.9.4.7	Hard Header Examination Sequences (HXSs) .....	5-378
5.9.4.7.1	L2 HXS - Ethernet .....	5-378
5.9.4.7.2	L2 HXS—VLAN .....	5-380
5.9.4.7.3	L2 HXS—LLC + SNAP .....	5-383
5.9.4.7.4	L2 HXS—PPPoE+PPP .....	5-384
5.9.4.7.5	L2 HXS—MPLS .....	5-385
5.9.4.7.6	L2 HXS—L2 Results .....	5-387
5.9.4.7.7	L3 HXS .....	5-388
5.9.4.7.8	L3 HXS—IPv4 .....	5-388
5.9.4.7.9	L3 HXS—IPv6 .....	5-390
5.9.4.7.10	L3 HXS - GRE version 0 .....	5-393
5.9.4.7.11	L3 HXS—Minimal Encapsulation .....	5-394
5.9.4.7.12	L3 HXS—Other L3 Shell .....	5-395
5.9.4.7.13	L3 HXS—L3 Results .....	5-395
5.9.4.7.14	L4 HXS—TCP .....	5-398
5.9.4.7.15	L4 HXS—UDP .....	5-399
5.9.4.7.16	L4 HXS—IPSec .....	5-401
5.9.4.7.17	L4 HXS—SCTP .....	5-401
5.9.4.7.18	L4 HXS—DCCP .....	5-402
5.9.4.7.19	L4 HXS—Other L4 Shell .....	5-403
5.9.4.7.20	L4 HXS—L4 Results .....	5-403
5.9.4.7.21	Shim Result .....	5-404
5.9.4.8	256 Limit Event Handling .....	5-405
5.9.4.9	Soft HXS Parse Array updates .....	5-407
5.9.4.10	Parse Result .....	5-407
5.9.4.11	Soft Parser .....	5-409
5.9.4.12	Parsing Exception handling .....	5-409
5.9.4.12.1	Invalid Soft Parser Instruction .....	5-409
5.9.4.12.2	Parse Cycle Limit .....	5-410
5.9.4.12.3	Parse Error Status .....	5-410
5.9.4.12.4	Soft Parser Parsing Events .....	5-410
5.9.4.12.5	Frame Received on a Disabled Port or Unsupported Port .....	5-410
5.9.4.12.6	Frame Received with 256 Limit or Error(s) in Parse Result/FD[STATUS] .....	5-411
5.9.4.13	Parser Debug Functionality .....	5-411
5.9.4.13.1	Parser Trace .....	5-411
5.9.4.13.2	Debug Traps .....	5-412
5.9.5	Parser Initialization Information .....	5-412
5.9.5.1	Start-Up or Restart .....	5-412
5.10	Frame Manager—Key Generator .....	5-413
5.10.1	KeyGen Overview .....	5-413

## Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.10.1.1	KeyGen Features Summary .....	5-413
5.10.2	KeyGen Memory Map/Register Definition .....	5-414
5.10.2.1	KeyGen Indirect Memory Space .....	5-415
5.10.2.2	KeyGen Memory Map .....	5-415
5.10.3	KeyGen Detailed Register Definitions .....	5-418
5.10.3.1	KeyGen General Configuration Register (FMKG_GCR) .....	5-418
5.10.3.2	KeyGen Error Event Register (FMKG_EER) .....	5-418
5.10.3.3	KeyGen Error Event Enable Register (FMKG_EEER) .....	5-419
5.10.3.4	KeyGen Scheme Error Event Register (FMKG_SEER) .....	5-420
5.10.3.5	KeyGen Global Status Register (FMKG_GSR) .....	5-420
5.10.3.6	KeyGen Total Packet Counter Register (FMKG_TPC) .....	5-421
5.10.3.7	KeyGen Soft Error Capture Register (FMKG_SERC) .....	5-421
5.10.3.8	KeyGen Frame Data Offset Register (FMKG_FDOR) .....	5-422
5.10.3.9	KeyGen Global Default Value Register (FMKG_GDV<i>R) .....	5-422
5.10.3.10	KeyGen Force Error Event Register (FMKG_FEER) .....	5-423
5.10.3.11	KeyGen Action Register (FMKG_AR) .....	5-424
5.10.3.12	KeyGen Scheme Entry Memory Map .....	5-426
5.10.3.12.1	KeyGen Scheme Entry MODE Register (AWR1_RFMODE_FMKG_SE_MODE) 5-426	
5.10.3.12.2	KeyGen Scheme Entry Extract Known Fields Command Register (AWR2_RFMODE_FMKG_SE_EKFC) .....	5-427
5.10.3.12.3	KeyGen Scheme Entry Extract Known Default Value Register (AWR3_RFMODE_FMKG_SE_EKDV) .....	5-431
5.10.3.12.4	KeyGen Scheme Bit Mask Command High Register (AWR4_RFMODE_FMKG_SE_BMCH) .....	5-433
5.10.3.12.5	KeyGen Scheme Bit Mask Command Low Register (AWR5_RFMODE_FMKG_SE_BMCL) .....	5-435
5.10.3.12.6	KeyGen Scheme Entry Frame Queue Base Register (AWR6_RFMODE_FMKG_SE_FQB) .....	5-436
5.10.3.12.7	KeyGen Scheme Entry Hash Configuration Register (AWR7_RFMODE_FMKG_SE_HC) .....	5-437
5.10.3.12.8	KeyGen Scheme Entry Policer Profile Command Register (AWR8_RFMODE_FMKG_SE_PPC) .....	5-438
5.10.3.12.9	KeyGen Scheme Entry Generic Extract Command Register (AWR<9+i>_RFMODE_FMKG_SE_GEC<i>) .....	5-439
5.10.3.12.10	KeyGen Scheme Entry Statistic Packet Counter Register (AWR17_RFMODE_FMKG_SE_SPC) .....	5-443
5.10.3.12.11	KeyGen Scheme Entry Default Value Register (AWR<18+i>_RFMODE_FMKG_SE_DV<i>) .....	5-443
5.10.3.12.12	KeyGen Scheme Entry Custom Classifier Bit Select Register (AWR20_RFMODE_FMKG_SE_CCBS) .....	5-444

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.10.3.12.13	KeyGen Scheme Entry Match Vector Register (AWR21_RFMODE_FMKG_SE_MV) .....	5-444
5.10.3.12.14	KeyGen Scheme Entry Operational Mode bits Register (AWR21_RFMODE_FMKG_SE_OM) .....	5-445
5.10.3.12.15	KeyGen Scheme Entry Virtual Storage Profile Register (AWR8_RFMODE_FMKG_SE_VSP) .....	5-446
5.10.3.13	KeyGen Classification Plan Entry Memory Map .....	5-447
5.10.3.13.1	KeyGen Classification Plan Entry Register (AWR<1+i>_RFMODE_FMKG_CPE<i>) .....	5-448
5.10.3.14	KeyGen Port Partition Configuration .....	5-448
5.10.3.14.1	KeyGen Port Entry Scheme Partition Register (AWR1_RFMODE_FMKG_PE_SP) .....	5-448
5.10.3.14.2	KeyGen Port Entry Classification Plan Partition Register (AWR2_RFMODE_FMKG_PE_CPP) .....	5-449
5.10.3.15	KeyGen Debug Registers .....	5-450
5.10.3.15.1	KeyGen Debug Control Register (FMKG_DCR) .....	5-450
5.10.3.15.2	KeyGen Debug Flow <j> Trap Counter Register (FMKG_D<j>TC) .....	5-451
5.10.3.15.3	KeyGen Debug Flow <j> Trap <k> Configuration Register (FMKG_D<j>T<k>CR) .....	5-451
5.10.3.15.4	KeyGen Debug Flow <j> Trap <k> Value Register (FMKG_D<j>T<k>VR) .....	5-454
5.10.3.15.5	KeyGen Debug Flow <j> Trap <k> Mask Register (FMKG_D<j>T<k>MR) .....	5-454
5.10.4	KeyGen Functional Description .....	5-454
5.10.4.1	Scheme Selection.....	5-457
5.10.4.1.1	Direct Scheme Selection.....	5-457
5.10.4.1.2	Indirect Scheme Selection .....	5-457
5.10.4.2	Key Generation .....	5-460
5.10.4.2.1	Extract Known Field Commands .....	5-460
5.10.4.2.2	Generic Known Field Commands .....	5-461
5.10.4.2.3	Field Validation.....	5-461
5.10.4.3	Hash Value Generation .....	5-461
5.10.4.3.1	Symmetric Hash Function .....	5-462
5.10.4.4	Keygen Distribution Function Value (KDFV).....	5-462
5.10.4.4.1	OR Data Vector Generation .....	5-463
5.10.4.5	Frame Queue ID (FQID) Generation.....	5-463
5.10.4.6	Policer Profile Number (PNUM) Generation .....	5-464
5.10.4.7	KeyGen Storage Profile ID Generation .....	5-464
5.10.4.8	Custom Classifier Base (CCBASE) Update .....	5-464
5.10.4.9	Next Invoke Action (NIA) Generation .....	5-466
5.10.4.10	KeyGen Inputs and Outputs.....	5-466
5.10.4.10.1	KeyGen NIA Action Codes .....	5-466
5.10.4.10.2	KeyGen Inputs from FMan Memory .....	5-467

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.10.4.10.3	KeyGen Outputs to FMan Memory.....	5-467
5.10.4.11	KeyGen Debug Functionality .....	5-468
5.10.4.11.1	Debug Trace.....	5-469
5.10.4.11.2	Debug Traps.....	5-469
5.10.5	KeyGen Initialization.....	5-470
5.10.5.1	Initialize the KeyGen Port Partition .....	5-470
5.10.5.2	Initialize the KeyGen Scheme Entry .....	5-470
5.10.5.2.1	Initialize a Full Scheme Entry .....	5-470
5.10.5.2.2	Initialize a Partial Scheme Entry .....	5-471
5.10.5.3	Initialize the KeyGen Classification Plan Group.....	5-471
5.10.5.3.1	Initialize a Full Classification Plan Group .....	5-471
5.10.5.3.2	Initialize a Partial Classification Plan Group .....	5-471
5.10.6	KeyGen Events .....	5-472
5.11	Frame Manager—Policer .....	5-473
5.11.1	Policer Overview .....	5-473
5.11.2	Policer Features.....	5-474
5.11.3	Policer Modes of Operation.....	5-476
5.11.3.1	Disabled Mode.....	5-476
5.11.3.2	Active Mode .....	5-476
5.11.4	Policer Memory Map and Register Definitions .....	5-476
5.11.4.1	FMan Policer General Configuration Register (FMPL_GCR).....	5-479
5.11.4.2	FMan Policer Global Status Register (FMPL_GSR).....	5-480
5.11.4.3	FMan Policer Event Register (FMPL_EVR).....	5-482
5.11.4.4	FMan Policer Interrupt Enable Register (FMPL_IER) .....	5-482
5.11.4.5	FMan Policer Interrupt Force Register (FMPL_IFR).....	5-483
5.11.4.6	FMan Policer Error Event Register (FMPL_EEVR).....	5-483
5.11.4.7	FMan Policer Error Interrupt Enable Register (FMPL_EIER).....	5-484
5.11.4.8	FMan Policer RED Packet Counter Register (FMPL_RPC).....	5-485
5.11.4.9	FMan Policer YELLOW Packet Counter Register (FMPL_YPC).....	5-485
5.11.4.10	FMan Policer Recolored RED Packet Counter Register (FMPL_RRPC).....	5-486
5.11.4.11	FMan Policer Recolored YELLOW Packet Counter Register (FMPL_RYPC) ..	5-486
5.11.4.12	FMan Policer Total Packet Counter Register (FMPL_TPC) .....	5-487
5.11.4.13	FMan Policer Frame Length Mismatch Counter Register (FMPL_FLMC).....	5-487
5.11.4.14	FMan Policer Profile Action Register (FMPL_PAR).....	5-488
5.11.4.15	FMan Policer Profile Entry Access Word Registers (FMPL_PE*).....	5-489
5.11.4.16	FMan Policer Soft Error Capture Register (FMPL_SERC) .....	5-490
5.11.4.17	FMan Policer Uninitialized Profile Capture Register (FMPL_UPCR) .....	5-491
5.11.4.18	FMan Policer Debug Trace Configuration Register (FMPL_DTRCR).....	5-492
5.11.4.19	FMan Policer Flow A Debug Trap Configuration Registers (FMPL_FADBTCR $n$ ).....	5-494
5.11.4.20	FMan Policer Flow A Debug Value Registers (FMPL_FADBVALR $n$ ).....	5-495

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.11.4.21	FMan Policer Flow A Debug Trap Mask Registers (FMPL_FADBTMR $n$ ) .....	5-496
5.11.4.22	FMan Policer Flow A Debug Trap Match Counter Register (FMPL_FADBTMC)..... 5-496	
5.11.4.23	FMan Policer Flow B Debug Trap Configuration Registers (FMPL_FBDBTCR $n$ )..... 5-497	
5.11.4.24	FMan Policer Flow B Debug Value Registers (FMPL_FBDBVALR $n$ ) .....	5-498
5.11.4.25	FMan Policer Flow B Debug Trap Mask Registers (FMPL_FBDhtmR $n$ ) .....	5-499
5.11.4.26	FMan Policer Flow B Debug Trap Match Counter Register (FMPL_FBDbtmc)..... 5-499	
5.11.4.27	FMan Policer Flow C Debug Trap Configuration Registers (FMPL_FCDBTCR $n$ )..... 5-500	
5.11.4.28	FMan Policer Flow C Debug Value Registers (FMPL_FCDBVALR $n$ ) .....	5-501
5.11.4.29	FMan Policer Flow C Debug Trap Mask Registers (FMPL_FCDBTMR $n$ ) .....	5-502
5.11.4.30	FMan Policer Flow C Debug Trap Match Counter Register (FMPL_FCDBTMC)..... 5-502	
5.11.4.31	FMan Policer Default Profile Mapping Register (FMPL_DPMR) .....	5-502
5.11.4.32	FMan Policer Profile Mapping Registers (FMPL_PMR $n$ ).....	5-504
5.11.4.33	Policer Profile Entry Memory Map .....	5-505
5.11.4.33.1	FMan Policer Profile Entry MODE Configuration Word Register (MODE)..	5-506
5.11.4.33.2	GREEN Next Invoked Action Configuration Word Register (GNIA).....	5-509
5.11.4.33.3	YELLOW Next Invoked Action Address Configuration Word Register (YNIA) .... 5-509	
5.11.4.33.4	RED Next Invoked Action Configuration Word Register (RNIA) .....	5-510
5.11.4.33.5	Committed Information Rate Configuration Word Register (CIR) .....	5-511
5.11.4.33.6	Committed Burst Size Configuration Word Register (CBS) .....	5-511
5.11.4.33.7	Peak/Excess Information Rate Configuration Word Register (PIR_EIR).....	5-512
5.11.4.33.8	Peak/Excess Burst Size Configuration Word Register (PBS_EBS) .....	5-512
5.11.4.33.9	Last Timestamp Variable Register (LTS) .....	5-513
5.11.4.33.10	Committed Token-Bucket Status Variable Register (CTS) .....	5-513
5.11.4.33.11	Peak/Excess Token-Bucket Status Variable Register (PTS_ETS) .....	5-514
5.11.4.33.12	GREEN Packet Counter Variable Register (GPC) .....	5-514
5.11.4.33.13	YELLOW Packet Counter Variable Register (YPC).....	5-515
5.11.4.33.14	RED Packet Counter Variable Register (RPC) .....	5-515
5.11.4.33.15	Recolored YELLOW Packet Counter Variable Register (RYPC).....	5-515
5.11.4.33.16	Recolored RED Packet Counter Variable Register (RRPC).....	5-516
5.11.5	Policer Functional Description .....	5-516
5.11.5.1	Policer Flow Description .....	5-516
5.11.5.1.1	Receiving an Incoming Packet .....	5-516
5.11.5.1.2	Avoiding Calculation Errors in Policer Profile Using Refresh Packets .....	5-518
5.11.5.1.3	Atomic Profile Update.....	5-518
5.11.5.2	Policer Profile Operation Modes .....	5-518

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.11.5.2.1	Pass-Through Modes .....	5-519
5.11.5.2.2	Color-Aware Pass-Through Mode .....	5-519
5.11.5.2.3	Color-Blind Pass-Through Mode .....	5-520
5.11.5.3	Traffic Metering and Marking Modes .....	5-520
5.11.5.3.1	Token Bucket Metering Principles .....	5-520
5.11.5.3.2	Rate Measurement Implementation by Time-Stamp .....	5-521
5.11.5.3.3	RFC-2698 Profile Mode Description .....	5-522
5.11.5.3.4	RFC-2698 Profile Mode Configuration.....	5-522
5.11.5.3.5	RFC-2698 Profile Mode Metering and Marking.....	5-524
5.11.5.3.6	RFC-4115 Profile Mode Description.....	5-525
5.11.5.3.7	RFC-4115 Profile Mode Configuration.....	5-525
5.11.5.3.8	RFC-4115 Profile Mode Metering and Marking.....	5-526
5.11.5.4	Functional Description—Next Invoked Action (NIA) .....	5-527
5.11.5.5	Debug Support .....	5-527
5.11.5.5.1	Packet-Based Debug Trace.....	5-528
5.11.5.5.2	Debug Flow Traps .....	5-528
5.11.5.5.3	Debug Trace.....	5-529
5.11.5.5.4	Flow-Based Debug Trace .....	5-532
5.11.6	Initialization Information.....	5-533
5.11.6.1	Reset Initialization .....	5-533
5.11.6.1.1	Hard Reset Sequence .....	5-533
5.11.6.1.2	Activating the PRAM Self Initialization .....	5-533
5.11.6.1.3	FMan Soft Reset Initialization.....	5-533
5.11.6.2	Profile Entry Initialization .....	5-533
5.11.6.3	PRAM Re-Initialization after Software Reset .....	5-534
5.11.6.3.1	Specific Profile Re-initialization .....	5-535
5.11.6.3.2	Full PRAM Re-Initialization .....	5-535
5.11.6.4	Profile Mapping Initialization.....	5-536
5.11.6.4.1	Next Invoked Action (NIA).....	5-536
5.11.6.5	Profile Concatenation Initialization .....	5-537
5.11.6.6	Writing to the Statistic Counters.....	5-538
5.11.7	Application Information .....	5-538
5.11.7.1	Load Spreading and Policing Traffic per Core .....	5-538
5.11.7.2	Profile Concatenation for Combined PACKET/BYTE Based Policing .....	5-540
5.11.7.3	Profile Concatenation for Aggregating Multiple Streams .....	5-540
5.11.7.4	Profile Concatenation with Per Color Aggregation.....	5-541
5.12	Frame Manager—FMan Controller .....	5-543
5.12.1	FMan Controller Overview.....	5-543
5.12.2	FMan Controller Features Summary .....	5-544
5.12.3	FMan Controller NIA—Action Codes .....	5-547
5.12.4	FMan Controller Memory Map/Register Definition .....	5-550

# Contents

Paragraph Number	Title	Page Number
5.12.4.1	FMan Controller Configuration Data Address Register (FMCDADDR).....	5-550
5.12.4.2	FMan Controller Configuration Data, Data Register (FMCDDATA) .....	5-551
5.12.4.3	FMan Controller Configuration Data Ready Register (FMCDREADY) .....	5-551
5.12.4.4	FMan Controller Configuration Data Download Flow .....	5-551
5.12.5	FMan Controller Functional Description.....	5-552
5.12.5.1	Matching Table Structure.....	5-552
5.12.5.2	TTL/Hop Limit Equals One Identification .....	5-553
5.12.5.3	Look-Up on Internal Context Fields.....	5-553
5.12.5.4	Using KeyGen HASH Result for Indexed Look-Up .....	5-553
5.12.5.5	Header Manipulation (HM) .....	5-555
5.12.5.6	Statistics Gathering in Custom Classifier .....	5-556
5.12.5.7	Custom Classifier and HM Generic Flow .....	5-557
5.12.6	Custom Classifier Action Descriptors (ADs) .....	5-558
5.12.6.1	New Classification Result.....	5-558
5.12.6.2	Keep Classification Result.....	5-561
5.12.6.3	Table Descriptor.....	5-563
5.12.7	First Custom Classifier Action Descriptor Location .....	5-568
5.12.8	Custom Classifier Search Tree Example .....	5-569
5.12.9	Header Manipulation Table Descriptor (HMTD) .....	5-569
5.12.10	Header Manipulation Command Descriptors (HMCDs).....	5-570
5.12.10.1	Header Removal Command Descriptor.....	5-572
5.12.10.2	Local Header Insert Command Descriptor .....	5-573
5.12.10.3	Internal Header Insert Command Descriptor.....	5-574
5.12.10.4	Local Header Replace Command Descriptor .....	5-574
5.12.10.5	Internal Header Replace Command Descriptor .....	5-575
5.12.10.6	Protocol Specific Header Removal Command Descriptor .....	5-576
5.12.10.7	Internal Protocol Specific Header Insert Command Descriptor .....	5-577
5.12.10.8	VLAN Priority Update Command Descriptor .....	5-579
5.12.10.8.1	IP_DSCP_to_VPri_Table .....	5-580
5.12.10.9	Local IPv4 Update Command Descriptor.....	5-580
5.12.10.10	Local TCP/UDP Update Command Descriptor.....	5-582
5.12.10.11	Local IPv6 Update Command Descriptor.....	5-583
5.12.10.12	Internal IP Header Replace Command Descriptor .....	5-585
5.12.10.13	UDP/TCP Checksum Calculation Command Descriptor .....	5-586
5.12.10.14	Remove Header till known parser result Command Descriptor .....	5-587
5.12.10.15	Local Insert UDP or UDP-Lite Header Command Descriptor .....	5-588
5.12.10.16	Local Insert L3 (IPv4 or IPv6) Header Command Descriptor .....	5-589
5.12.10.17	Remove/Local Insert CAPWAP Header Command Descriptor .....	5-590
5.12.10.18	Replace Field in Header Command Descriptor .....	5-591
5.12.11	Restrictions on the Usage of HMCDs.....	5-592
5.12.12	Parse After Header Manipulation .....	5-593

# Contents

Paragraph Number	Title	Page Number
5.12.13	Deep Sleep Auto Response.....	5-596
5.12.13.1	Introduction to Deep Sleep Auto Response.....	5-596
5.12.13.2	Auto Response Pass Filter .....	5-597
5.12.13.3	Soft Parser for Deep Sleep Auto Response .....	5-598
5.12.13.4	Deep Sleep Auto Response Programming Model .....	5-598
5.12.13.4.1	Filtering Tables Formats .....	5-602
5.12.13.4.2	Deep Sleep Auto Response Statistics table .....	5-603
5.12.13.5	Address Resolution Protocol (ARP) Deep Sleep Auto Response .....	5-605
5.12.13.5.1	Introduction to ARP Deep Sleep Auto Response .....	5-605
5.12.13.5.2	Functional Description of ARP Deep Sleep Auto Response.....	5-606
5.12.13.5.3	Deep Sleep Auto Response ARP Descriptor .....	5-607
5.12.13.5.4	Deep Sleep Auto Response ARP Data Structures .....	5-608
5.12.13.6	Internet Control Message Protocol (ICMP) Deep Sleep Auto Response .....	5-610
5.12.13.6.1	Introduction to ICMP Deep Sleep Auto Response.....	5-610
5.12.13.6.2	IPv4 ICMP Deep Sleep Auto Response Functional Description .....	5-610
5.12.13.6.3	Deep Sleep Auto Response ICMPv4 Descriptor .....	5-611
5.12.13.6.4	Deep Sleep Auto Response ICMPv4 Data Structures .....	5-612
5.12.13.7	Internet Control Message Protocol (ICMPv6) Echo Auto Response .....	5-613
5.12.13.7.1	Introduction to ICMPv6 Echo Deep Sleep Auto Response.....	5-613
5.12.13.7.2	ICMPv6 Echo Auto Response Functional Description .....	5-614
5.12.13.7.3	Deep Sleep Auto Response ICMPv6 Descriptor .....	5-615
5.12.13.7.4	Deep Sleep Auto Response ICMPv6 Data Structures .....	5-615
5.12.13.8	Neighbor Discovery Protocol Auto Response.....	5-617
5.12.13.8.1	Introduction to IPv6 Neighbor Discovery Deep Sleep Auto Response.....	5-617
5.12.13.8.2	Neighbor Discovery Auto Response Functional Description.....	5-618
5.12.13.8.3	Deep Sleep Auto Response Neighbor Discovery Descriptor .....	5-621
5.12.13.8.4	Deep Sleep Auto Response Neighbor Discovery Data Structures .....	5-622
5.12.13.9	SNMP Deep Sleep Auto Response.....	5-624
5.12.13.9.1	Introduction to SNMP Deep Sleep Auto Response.....	5-624
5.12.13.9.2	Key features of SNMP Deep Sleep Auto Response .....	5-624
5.12.13.9.3	Functional description of SNMP Deep Sleep Auto Response .....	5-625
5.12.13.9.4	SNMP Deep Sleep Auto Response Programming model.....	5-627
5.12.14	FMan Controller Important Often-Overlooked Details .....	5-631
5.12.14.1	Dynamic Update of Custom Classifier and HM Tables .....	5-632
5.12.14.1.1	Direct Table Access Direct Access Sync Flow.....	5-632
5.12.14.1.2	Direct Table access Host Command Sync Flow .....	5-633
5.12.14.1.3	Full Host Command Flow .....	5-633
5.12.15	Ethernet Independent Mode (IM) .....	5-634
5.12.15.1	IM Introduction.....	5-634
5.12.15.2	IM Features .....	5-634
5.12.15.3	Frame Transmitter Overview .....	5-634

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
5.12.15.4	Frame Receiver Overview .....	5-636
5.12.15.5	IM Programming Model .....	5-637
5.12.15.5.1	Parameter RAM.....	5-637
5.12.15.5.2	Rx Queue Descriptor .....	5-639
5.12.15.5.3	RxBD .....	5-640
5.12.15.5.4	Tx Queue Descriptor .....	5-642
5.12.15.5.5	TxBD .....	5-643
5.12.15.6	IM Application Notes .....	5-643
5.12.16	Host Commands.....	5-644
5.12.16.1	Host Command Introduction .....	5-644
5.12.16.2	Host Command Features.....	5-645
5.12.16.3	Host Command Flow .....	5-645
5.12.16.4	Host Command Programming Model.....	5-646
5.12.16.4.1	Host Command Opcode Register (HCOR) .....	5-647
5.12.16.4.2	Host Command Action Register (HCAR) .....	5-648
5.12.16.4.3	Host Command Extra Register (HCER).....	5-649
5.12.16.4.4	Host Command Sequence Number (HCSN) .....	5-651
5.12.16.4.5	Host Command Data Area.....	5-652
5.12.16.5	Host Command Application Notes .....	5-653
5.12.17	IP Acceleration Programming Model Additions .....	5-653
5.12.17.1	IP Fragmentation .....	5-653
5.12.17.1.1	IP Fragmentation Functional Description.....	5-654
5.12.17.1.2	IP Fragmentation Data Structures.....	5-659
5.12.17.2	IP Reassembly .....	5-661
5.12.17.2.1	Overview .....	5-661
5.12.17.2.2	Terminology.....	5-661
5.12.17.2.3	IP Reassembly Functional Description.....	5-662
5.12.17.2.4	IP Reassembly Data Structures.....	5-667
5.12.17.3	IPsec.....	5-683
5.12.17.3.1	IPsec Features .....	5-683
5.12.17.3.2	IPsec Introduction.....	5-684
5.12.17.4	Table Descriptors Additions .....	5-685
5.12.17.4.1	IP Manipulation Custom-Classification Table Descriptor.....	5-685
5.12.17.4.2	IP Fragmentation Table Descriptor.....	5-688
5.12.17.4.3	IP Reassembly Table Descriptor.....	5-691
5.12.17.4.4	IPsec Manipulation Table Descriptor .....	5-693
5.12.18	Statistics Gathering .....	5-695
5.12.18.1	Frame and Byte Count Statistics.....	5-695
5.12.18.2	Frame Length Range Statistics (FLR) .....	5-695
5.12.18.3	Conditional Statistics (Fman_v3 only) .....	5-696
5.12.18.3.1	Configuring the Conditional Statistics.....	5-697

# Contents

Paragraph Number	Title	Page Number
5.12.18.3.2	Conditional Statistics Dynamic Changes.....	5-698
5.12.18.3.3	Conditional Statistics Restrictions .....	5-698
5.12.18.4	Statistics Table Descriptor (STD) .....	5-698
5.12.19	Next Generation CAPWAP Programming Model Additions .....	5-699
5.12.19.1	CAPWAP Fragmentation and CAPWAP Reassembly .....	5-700
5.12.19.2	Next Generation CAPWAP Table Descriptor Additions .....	5-700
5.12.19.2.1	CAPWAP Fragmentation Condition Check Custom-Classification Table Descriptor 5-700	
5.12.19.2.2	CAPWAP Reassembly Table Descriptor .....	5-702
5.12.19.2.3	CAPWAP Fragmentation Table Descriptor .....	5-703
5.12.19.2.4	CAPWAP Manipulation Table Descriptor.....	5-705
5.12.20	Frame Replicator Support.....	5-707
5.12.20.1	Frame Replicator Source Table Descriptor (FRSTD) .....	5-709
5.12.21	Pre/Post BMI Fetch NIAs .....	5-710
5.12.22	FMan Controller Event Register.....	5-712
5.12.23	FMan-Controller Parameters Page per Port.....	5-712
5.12.24	General FMan-Controller Application Notes .....	5-713

## Chapter 6

### Multirate Ethernet Media Access Controller (mEMAC)

6.1	mEMAC Overview .....	6-1
6.2	mEMAC Features Summary.....	6-1
6.3	mEMAC External Signals Descriptions .....	6-2
6.3.1	mEMAC Detailed Signal Descriptions.....	6-3
6.4	mEMAC Memory Map/Register Definition.....	6-4
6.4.1	mEMAC Top-Level Memory Map .....	6-5
6.4.2	mEMAC Detailed Memory Map .....	6-5
6.4.3	mEMAC Memory-Mapped Register Descriptions .....	6-10
6.4.3.1	mEMAC General Control and Status Registers .....	6-10
6.4.3.1.1	Command and Configuration Register (COMMAND_CONFIG) .....	6-10
6.4.3.1.2	First MAC Lower Address Register (MAC_ADDR_0).....	6-12
6.4.3.1.3	First MAC Upper Address Register (MAC_ADDR_1) .....	6-13
6.4.3.1.4	Maximum Frame Length Register (MAXFRM) .....	6-13
6.4.3.1.5	Receive FIFO Sections Register (RX_FIFO_SECTIONS).....	6-14
6.4.3.1.6	Transmit FIFO Sections Register (TX_FIFO_SECTIONS).....	6-14
6.4.3.1.7	Hash Table Control Register (HASHTABLE_CTRL) .....	6-15
6.4.3.1.8	Interrupt Event Register (IEVENT) .....	6-16
6.4.3.1.9	Transmit Inter-Packet Gap Length Register (TX_LENGTH).....	6-17
6.4.3.1.10	Interrupt Mask Register (IMASK) .....	6-18
6.4.3.1.11	CL01 Pause Quanta Register (CL01_PAUSE_QUANTA) .....	6-19

# Contents

Paragraph Number	Title	Page Number
6.4.3.1.12	CL23 Pause Quanta Register (CL23_PAUSE_QUANTA) .....	6-20
6.4.3.1.13	CL45 Pause Quanta Register (CL45_PAUSE_QUANTA) .....	6-20
6.4.3.1.14	CL67 Pause Quanta Register (CL67_PAUSE_QUANTA) .....	6-20
6.4.3.1.15	CL01 Pause Quanta Threshold Register (CL01_PAUSE_THRESH) .....	6-21
6.4.3.1.16	CL23 Pause Quanta Threshold Register (CL23_PAUSE_THRESH) .....	6-21
6.4.3.1.17	CL45 Pause Quanta Threshold Register (CL45_PAUSE_THRESH) .....	6-21
6.4.3.1.18	CL67 Pause Quanta Threshold Register (CL67_PAUSE_THRESH) .....	6-22
6.4.3.1.19	Receive Pause Status Register (RX_PAUSE_STATUS) .....	6-22
6.4.3.1.20	2nd MAC Lower Address Register (MAC_ADDR_2) .....	6-23
6.4.3.1.21	2nd MAC Upper Address Register (MAC_ADDR_3) .....	6-23
6.4.3.1.22	3rd MAC Lower Address Register (MAC_ADDR_4) .....	6-23
6.4.3.1.23	3rd MAC Upper Address Register (MAC_ADDR_5) .....	6-24
6.4.3.1.24	4th MAC Lower Address Register (MAC_ADDR_6) .....	6-24
6.4.3.1.25	4th MAC Upper Address Register (MAC_ADDR_7) .....	6-24
6.4.3.1.26	5th MAC Lower Address Register (MAC_ADDR_8) .....	6-25
6.4.3.1.27	5th MAC Upper Address Register (MAC_ADDR_9) .....	6-25
6.4.3.1.28	6th MAC Lower Address Register (MAC_ADDR_10) .....	6-25
6.4.3.1.29	6th MAC Upper Address Register (MAC_ADDR_11) .....	6-26
6.4.3.1.30	7th MAC Lower Address Register (MAC_ADDR_12) .....	6-26
6.4.3.1.31	7th MAC Upper Address Register (MAC_ADDR_13) .....	6-26
6.4.3.1.32	8th MAC Lower Address Register (MAC_ADDR_14) .....	6-27
6.4.3.1.33	8th MAC Upper Address Register (MAC_ADDR_15) .....	6-27
6.4.3.1.34	EEE Low Power Wakeup Timer Register (LPWAKE_TIMER) .....	6-27
6.4.3.1.35	Transmit EEE Low Power Timer Register (SLEEP_TIMER) .....	6-28
6.4.3.1.36	Statistics Configuration Register (STATN_CONFIG) .....	6-28
6.4.3.2	mEMAC Statistics Counter Registers .....	6-28
6.4.3.2.1	Receive Ethernet Octets Counter (REOCT $n$ ) .....	6-29
6.4.3.2.2	Receive Octets Counter Register (ROCT $n$ ) .....	6-29
6.4.3.2.3	Receive Alignment Error Counter Register (RALN $n$ ) .....	6-29
6.4.3.2.4	Receive Valid Pause Frame Counter Register (RXPF $n$ ) .....	6-30
6.4.3.2.5	Receive Frame Counter Register (RFRM $n$ ) .....	6-30
6.4.3.2.6	Receive Frame Check Sequence Error Counter Register (RFCS $n$ ) .....	6-30
6.4.3.2.7	Receive VLAN Frame Counter Register (RVLAN $n$ ) .....	6-31
6.4.3.2.8	Receive Frame Error Counter Register (RERR $n$ ) .....	6-31
6.4.3.2.9	Receive Unicast Frame Counter Register (RUCAn) .....	6-31
6.4.3.2.10	Receive Multicast Frame Counter Register (RMCan) .....	6-32
6.4.3.2.11	Receive Broadcast Frame Counter Register (RBCAn) .....	6-32
6.4.3.2.12	Receive Dropped Packets Counter Register (RDRP $n$ ) .....	6-32
6.4.3.2.13	Receive Packets Counter Register (RPKT $n$ ) .....	6-33
6.4.3.2.14	Receive Undersized Packet Counter Register (RUND $n$ ) .....	6-33
6.4.3.2.15	Receive 64-Octet Packet Counter Register (R64 $n$ ) .....	6-33

# Contents

Paragraph Number	Title	Page Number
6.4.3.2.16	Receive 65- to 127-Octet Packet Counter Register (R127n).....	6-34
6.4.3.2.17	Receive 128- to 255-Octet Packet Counter Register (R255n).....	6-34
6.4.3.2.18	Receive 256- to 511-Octet Packet Counter Register (R511n).....	6-34
6.4.3.2.19	Receive 512- to 1023-Octet Packet Counter Register (R1023n).....	6-35
6.4.3.2.20	Receive 1024- to 1518-Octet Packet Counter Register (R1518n).....	6-35
6.4.3.2.21	Receive 1519- to Max-Octet Packet Counter Register (R1519Xn) .....	6-35
6.4.3.2.22	Receive Oversized Packet Counter Register (ROVRn) .....	6-36
6.4.3.2.23	Receive Jabber Packet Counter Register (RJBRn).....	6-36
6.4.3.2.24	Receive Fragment Packet Counter Register (RFRGn) .....	6-36
6.4.3.2.25	Receive Control Packet Counter Register (RCNPn) .....	6-37
6.4.3.2.26	Receive Dropped Not Truncated Packets Counter Register (RDRNTPn) .....	6-37
6.4.3.2.27	Transmit Ethernet Octets Counter (TEOCTn).....	6-37
6.4.3.2.28	Transmit Octets Counter Register (TOCTn) .....	6-38
6.4.3.2.29	Transmit Valid Pause Frame Counter Register (TXPFn) .....	6-38
6.4.3.2.30	Transmit Frame Counter Register (TFRMn).....	6-38
6.4.3.2.31	Transmit Frame Check Sequence Error Counter Register (TFCSn) .....	6-39
6.4.3.2.32	Transmit VLAN Frame Counter Register (TVLANn) .....	6-39
6.4.3.2.33	Transmit Frame Error Counter Register (TERRn) .....	6-39
6.4.3.2.34	Transmit Unicast Frame Counter Register (TUCAn).....	6-40
6.4.3.2.35	Transmit Multicast Frame Counter Register (TMCAAn) .....	6-40
6.4.3.2.36	Transmit Broadcast Frame Counter Register (TBCAn) .....	6-41
6.4.3.2.37	Transmit Packets Counter Register (TPKTn) .....	6-41
6.4.3.2.38	Transmit Undersized Packet Counter Register (TUNDn) .....	6-41
6.4.3.2.39	Transmit 64-Octet Packet Counter Register (T64n).....	6-42
6.4.3.2.40	Transmit 65- to 127-Octet Packet Counter Register (T127n) .....	6-42
6.4.3.2.41	Transmit 128- to 255-Octet Packet Counter Register (T255n) .....	6-42
6.4.3.2.42	Transmit 256- to 511-Octet Packet Counter Register (T511n).....	6-43
6.4.3.2.43	Transmit 512- to 1023-Octet Packet Counter Register (T1023n) .....	6-43
6.4.3.2.44	Transmit 1024- to 1518-Octet Packet Counter Register (T1518n) .....	6-43
6.4.3.2.45	Transmit 1519- to TX_MTU-Octet Packet Counter Register (T1519Xn) .....	6-44
6.4.3.2.46	Transmit Control Packet Counter Register (TCNPn) .....	6-44
6.4.3.3	Line Interface Control Registers.....	6-44
6.4.3.3.1	Interface Mode Register (IF_MODE) .....	6-44
6.4.3.3.2	Interface Status Register (IF_STATUS) .....	6-45
6.4.3.4	MDIO Ethernet Management Interface Registers .....	6-46
6.4.3.4.1	MDIO Configuration Register (MDIO_CFG).....	6-46
6.4.3.4.2	MDIO Control Register (MDIO_CTL) .....	6-48
6.4.3.4.3	MDIO Data Register (MDIO_DATA) .....	6-49
6.4.3.4.4	MDIO Register Address Register (MDIO_ADDR) .....	6-49
6.5	mEMAC Functional Description .....	6-50
6.5.1	MAC Address Insertion .....	6-50

# Contents

Paragraph Number	Title	Page Number
6.5.2	CRC-32 Calculation.....	6-50
6.5.3	Unicast destination address recognition .....	6-50
6.5.4	MDIO Ethernet Management Interface usage .....	6-50
6.5.4.1	Clause 45 Read Flow .....	6-50
6.5.4.2	Clause 45 Write Flow .....	6-51
6.5.4.3	Clause 22 Read Flow .....	6-51
6.5.4.4	Clause 22 Write Flow .....	6-51
6.5.5	Graceful stop.....	6-51
6.6	Initialization Information.....	6-52

## Chapter 7

### IEEE 1588 Timer Module

7.1	Overview.....	7-1
7.1.1	Features.....	7-1
7.2	Modes of Operation .....	7-1
7.3	External Signals Description .....	7-1
7.3.1	Detailed Signal Descriptions .....	7-2
7.4	Memory Map/Register Definition .....	7-2
7.4.1	Top-Level Module Memory Map .....	7-2
7.4.2	Detailed Memory Map.....	7-3
7.4.3	Memory-Mapped Register Descriptions.....	7-4
7.4.3.1	General Control and Status Registers .....	7-4
7.4.3.1.1	Module ID Register (TMR_ID).....	7-4
7.4.3.1.2	Controller ID Register (TMR_ID2).....	7-5
7.4.3.2	Hardware Assist for IEEE1588 Compliant Timestamping.....	7-5
7.4.3.2.1	Timer Control Register (TMR_CTRL) .....	7-5
7.4.3.2.2	Timer Event Register (TMR_TEVENT) .....	7-7
7.4.3.2.3	Timer Event Mask Register (TMR_TEMASK) .....	7-8
7.4.3.2.4	Timer Status Register (TMR_STAT) .....	7-9
7.4.3.2.5	Timer Counter Register (TMR_CNT_H/L).....	7-10
7.4.3.2.6	Timer Drift Compensation Addend Register (TMR_ADD).....	7-11
7.4.3.2.7	Timer Accumulator Register (TMR_ACC).....	7-12
7.4.3.2.8	Timer Prescale Register (TMR_PRSC).....	7-12
7.4.3.2.9	Timer Offset Register (TMROFF_H/L) .....	7-13
7.4.3.2.10	Alarm Time Comparator Register (TMR_ALARM1–2_H/L) .....	7-14
7.4.3.2.11	Timer Fixed Interval Period Register (TMR_FIPER1–3) .....	7-14
7.4.3.2.12	External Trigger Stamp Register (TMR_ETTS1–2_H/L) .....	7-16
7.5	Functional Description.....	7-17
7.5.1	1588 Timer Module Operation .....	7-17
7.5.1.1	Initialization Sequence.....	7-17

# Contents

<b>Paragraph Number</b>	<b>Title</b>	<b>Page Number</b>
7.5.1.1.1	Hardware Controlled Initialization .....	7-17
7.5.1.1.2	User Initialization .....	7-17
7.5.1.2	Soft Reset and Reconfiguring Procedure.....	7-18
7.5.1.3	Interrupt Handling .....	7-19
7.6	Initialization/Application Information.....	7-19
7.6.1	Hardware Assist for IEEE 1588 Compliant Timestamping.....	7-19
7.6.1.1	Features.....	7-19
7.6.1.2	Time-Stamping of Packets.....	7-22
7.6.1.3	PTP Packets .....	7-22
7.6.1.4	Time-Stamp For External Trigger .....	7-23
7.6.1.5	User Time Adjustments .....	7-23
7.6.1.5.1	1588 Timer Counter Setup Delays .....	7-23
7.6.1.5.2	1588 Timer Alarm Delays .....	7-23
7.6.1.5.3	Time-Stamp Delays for Supported Modes .....	7-24

## Appendix A Revision History

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
1-1	QorIQ Data Path Acceleration Architecture (DPAA) .....	1-4
1-2	Frame Descriptor (FD) .....	1-9
1-3	Scatter/Gather Table Entry Format .....	1-11
1-4	Simplified Representation of a Long, Multi-Buffer, Simple Frame .....	1-12
1-5	Simplified Representation of a Compound Frame .....	1-14
3-1	QMan Overview Block Diagram .....	3-2
3-2	QCSP Enqueue Command Ring Registers ( $QCSPi_EQCR0-7$ ) .....	3-14
3-3	QCSP Dequeue Response Ring Registers ( $QCSPi_DQRR0-15$ ) (Entries 0 to 7) .....	3-16
3-4	QCSP Dequeue Response Ring Registers ( $QCSPi_DQRR0-15$ ) (Entries 8 to 15) .....	3-17
3-5	QCSP Message Ring Registers ( $QCSPi_MR0-7$ ) .....	3-18
3-6	QCSP Management Command Registers ( $QCSPi_CR$ ) .....	3-19
3-7	QCSP Management Response Register 0 ( $QCSPi_RR0$ ) .....	3-20
3-8	QCSP Management Response Register 1 ( $QCSPi_RR1$ ) .....	3-21
3-9	QCSP EQCR Producer Index Registers ( $QCSPi_EQCR_PI_CENA$ ) .....	3-22
3-10	QCSP EQCR Producer Index Register ( $QCSPi_EQCR_PI_CINH$ ) .....	3-22
3-11	QCSP EQCR Consumer Index Register ( $QCSPi_EQCR_CI_CENA$ ) .....	3-24
3-12	QCSP EQCR Consumer Index Register ( $QCSPi_EQCR_CI_CINH$ ) .....	3-25
3-13	QCSP DQRR Producer Index Register ( $DQRRi_PI_CENA$ ) .....	3-26
3-14	QCSP DQRR Producer Index Cache-Inhibited Registers ( $QCSPi_DQRR_PI_CINH$ ) .....	3-27
3-15	QCSP DQRR Consumer Index Cache-Enabled Registers ( $QCSPi_DQRR_CI_CENA$ ) .....	3-28
3-16	QCSP DQRR Consumer Index Cache-Inhibited Registers ( $QCSPi_DQRR_CI_CINH$ ) .....	3-29
3-17	DQRR Discrete Consumption Acknowledgment Register ( $QCSPi_DQRR_DCAP$ ) .....	3-30
3-18	QCSP DQRR Static Dequeue Command Register ( $QCSPi_DQRR_SDQCR$ ) .....	3-32
3-19	QCSO DQRR Volatile Dequeue Command Register ( $QCSPi_DQRR_VDQCR$ ) .....	3-34
3-20	QCSP DQRR Pull Dequeue Command Register ( $QCSPi_DQRR_PDQCR$ ) .....	3-35
3-21	QCSP MR Producer Index Cache-Enabled Registers ( $QCSPi_MR_PI_CENA$ ) .....	3-37
3-22	QCSP MR Producer Index Cache-Inhibited Registers ( $QCSPi_MR_PI_CINH$ ) .....	3-37
3-23	QCSP MR Consumer Index Cache-Enabled Registers ( $QCSPi_MR_CI_CENA$ ) .....	3-38
3-24	QCSP MR Consumer Index Cache-Inhibited Registers ( $QCSPi_MR_CI_CINH$ ) .....	3-39
3-25	QCSP Read-Only Ring Indices Cache-Enabled Registers ( $QCSPi_RORI_CENA$ ) .....	3-40
3-26	QCSP Configuration Register ( $QCSPi_CFG$ ) .....	3-41
3-27	QCSP EQCR Interrupt Threshold Register ( $QCSPi_EQCR_ITR$ ) .....	3-44
3-28	QCSP DQRR Interrupt Threshold Register ( $QCSPi_DQRR_ITR$ ) .....	3-45
3-29	QCSP MR Interrupt Threshold Register ( $QCSPi_MR_ITR$ ) .....	3-45
3-30	QCSP Interrupt Status Register ( $QCSPi_ISR$ ) .....	3-46
3-31	QCSP Interrupt Enable Register ( $QCSPi_IER$ ) .....	3-47
3-32	QCSP Interrupt Enable Registers ( $QCSPi_IER$ ) .....	3-48
3-33	QCSP Interrupt Inhibit Register ( $QCSPi_IIR$ ) .....	3-48
3-34	QCSP Interrupt Time Out Period Registers ( $QCSPi_ITPR$ ) .....	3-49
3-35	QCSP ICID Configuration ( $QCSPi_ICID_CFG$ ) .....	3-50
3-36	QCSP IO Configuration Registers ( $QCSPi_IO_CFG$ ) .....	3-50

# Figures

Figure Number	Title	Page Number
3-37	QCSP Dynamic Debug Configuration Registers (QCSP <sub>i</sub> _DD_CFG).....	3-51
3-38	QMan Dynamic Debug Configuration Register (QMAN_DD_CFG).....	3-52
3-39	QCSP Dynamic Debug Internal Halt Request Status Registers (QCSP_DD_IHRSR_i) .....	3-54
3-40	DCP Dynamic Debug Internal Halt Request Status Register (DCP_DD_IHRSR) .....	3-54
3-41	QCSP Dynamic Debug Internal Halt Request Force Registers (QCSP_DD_IHRFR_i).....	3-55
3-42	DCP Dynamic Debug Internal Halt Request Force Register (DCP_DD_IHRFR).....	3-56
3-43	QCSP Dynamic Debug Halt Acknowledge Status Registers (QCSP_DD_HASR_i) .....	3-56
3-44	DCP Dynamic Debug Halt Acknowledge Status Register (DCP_DD_HASR).....	3-57
3-45	DCP Configuration Registers (DCPi_CFG) .....	3-58
3-46	DCP Dynamic Debug Configuration Registers (DCPi_DD_CFG) .....	3-59
3-47	DCP Dequeue Latency Monitor Configuration (DCPi_DLM_CFG).....	3-60
3-48	DCP Dequeue Latency Monitor Average Registers (DCPi_DLM_AVG).....	3-61
3-49	PFDR Free Pool Count Register (PFDR_FPC) .....	3-62
3-50	PFDR Free Pool Head Pointer Register (PFDR_FP_HEAD).....	3-62
3-51	PFDR Free Pool Tail Pointer Register (PFDR_FP_TAIL) .....	3-63
3-52	PFDR Free Pool Low Watermark Interrupt Threshold (PFDR_FP_LWIT) .....	3-63
3-53	PFDR Configuration (PFDR_CFG).....	3-64
3-54	SFDR Configuration Register (SFDR_CFG) .....	3-64
3-55	SFDR In Use Register (SFDR_IN_USE) .....	3-65
3-56	Work Queue Class Scheduler Configuration (WQ_CS_CFG <sub>i</sub> ) .....	3-66
3-57	WQ Default Enqueue WQID Register (WQ_DEF_ENQ_WQID).....	3-67
3-58	WQ Channel Software Portal Dynamic Debug Configuration Registers (WQ_SC_DD_CFG <sub>i</sub> ) 3-67	
3-59	WQ Channel Pool Dynamic Debug Configuration Registers (WQ_PC_DD_CFG <sub>i</sub> ).....	3-68
3-60	WQ Channel DCP Dynamic Debug Configuration Registers (WQ_DCx_DD_CFG <sub>i</sub> ).....	3-70
3-61	CM Configuration Register (CM_CFG) .....	3-71
3-62	CEETM Configuration Index Register (CEETM_CFG_IDX) .....	3-72
3-63	CEETM Configuration Shaper Pre-Scaler Register (CEETM_CFG_PRES) .....	3-72
3-64	CEETM XSFDR In Use Register (CEETM_XSFDR_IN_USE) .....	3-73
3-65	QMan Error Capture Status Register (QMAN_ECSR) .....	3-74
3-66	QMan Error Capture Information Register (QMAN_ECIR) .....	3-75
3-67	QMan Error Capture Information Register 2 (QMAN_ECIR2).....	3-76
3-68	QMan Error Address Register (QMAN_EADR).....	3-76
3-69	QMan Error Data Registers (QMAN_EDATA <sub>i</sub> ).....	3-78
3-70	QMan Single Bit Error Threshold Register (QMAN_SBET).....	3-78
3-71	QMan Single Bit Error Count Registers (QMAN_SBECi) .....	3-79
3-72	QMan Management Command/Result Register (QMAN_MCR) .....	3-80
3-73	QMan Management Command Parameter 0 Register (QMAN_MCP0) .....	3-81
3-74	QMan Management Command Parameter 1 Register (QMAN_MCP1) .....	3-83
3-75	QMan Management Command Result Registers (QMAN_MR <sub>i</sub> ).....	3-83
3-76	QMan Miscellaneous Configuration Register (QMAN_MISC_CFG) .....	3-84

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
3-77	QMan Idle Status Register (QMAN_IDLE_STAT) .....	3-85
3-78	QMan IP Block Revision 1 Register (QMAN_IP_REV_1) .....	3-86
3-79	QMan IP Block Revision 2 Register (QMAN_IP_REV_2) .....	3-86
3-80	Data Structure Extended Base Address Registers (FQD_BARE) .....	3-87
3-81	Data Structure Extended Base Address Registers (PFDR_BARE) .....	3-88
3-82	Data Structure Base Address Registers (FQD_BAR) .....	3-88
3-83	Data Structure Base Address Registers (PFDR_BAR) .....	3-89
3-84	Data Structure Attribute Register (FQD_AR) .....	3-90
3-85	Data Structure Attribute Register (PFDR_AR) .....	3-90
3-86	QCSP Extended Base Address Register (QCSP_BARE) .....	3-92
3-87	QCSP Base Address Register (QCSP_BAR) .....	3-92
3-88	Initiator Scheduling Configuration Register (CI_SCHED_CFG) .....	3-93
3-89	QMan Source ID Register (QMAN_SRCIDR) .....	3-93
3-90	QMan Isolation Context Identifier Register (QMAN_ICIDR) .....	3-94
3-91	Initiator Read Latency Monitor Configuration Register (CI_RLM_CFG) .....	3-95
3-92	Initiator Read Latency Monitor Average Register (CI_RLM_AVG) .....	3-96
3-93	QMan Error Interrupt Status Register (QMAN_ERR_ISR) .....	3-97
3-94	QMan Error Interrupt Enable Register (QMAN_ERR_IER) .....	3-99
3-95	QMan Error Interrupt Status Disable Register (QMAN_ERR_ISDR) .....	3-100
3-96	QMan Error Interrupt Inhibit Register (QMAN_ERR_IIR) .....	3-100
3-97	QMan Error Halt Enable Register (QMAN_ERR_HER) .....	3-101
3-98	QMan Detailed Block Diagram .....	3-102
3-99	Frame Descriptor Format (FD) .....	3-104
3-100	Example—Frame Queue Structure Walk-Through .....	3-108
3-101	Example—Frame Queue Structure Walk-Through (continued) .....	3-109
3-102	Example—Frame Queue Structure Walk-Through (continued) .....	3-110
3-103	Frame Queue Descriptor (FQD) Format .....	3-112
3-104	Frame Queue State Diagram .....	3-121
3-105	Software Portal Dequeue Dispatcher and WQ Class Schedulers .....	3-146
3-106	Dequeue Dispatcher Operation Flow Diagram—Dequeueing from One or More Channels .....	3-153
3-107	Dequeue Dispatcher Operation Flow Diagram—Dequeueing from a Specified WQ .....	3-154
3-108	FQD Context_A for dequeued Frame Data, Annotation, and Context Stashing control .....	3-164
3-109	Enqueue Command Format .....	3-168
3-110	Frame Dequeue Response Format .....	3-171
3-111	ERN Message Response Format .....	3-174
3-112	FQ State Change Notification Message Response Format .....	3-176
3-113	Initialize FQ Command Format .....	3-180
3-114	Initialize FQ Response Format .....	3-182
3-115	Query FQ Programmable Fields Command Format .....	3-183
3-116	Query FQ Programmable Fields Response Format .....	3-183
3-117	Query FQ Non-Programmable Fields Command Format .....	3-185

# Figures

Figure Number	Title	Page Number
3-118	Query FQ Non-Programmable Fields Response Format .....	3-185
3-119	Alter FQ State Command Format .....	3-189
3-120	Alter FQ State Response Format.....	3-190
3-121	Query WQ Length Command Format .....	3-192
3-122	Query WQ Length Response Format.....	3-192
3-123	Initialize/Modify CGR Command Format.....	3-194
3-124	Initialize/Modify CGR Response Format .....	3-196
3-125	Query CGR Command Format .....	3-197
3-126	Query CGR Response Format.....	3-197
3-127	Query Congestion State Command Format .....	3-199
3-128	Query Congestion State Response Format.....	3-200
3-129	LFQMT Configure Command Format.....	3-201
3-130	LFQMT Configure Response Format .....	3-202
3-131	LFQMT Query Command Format .....	3-203
3-132	LFQMT Query Response Format .....	3-203
3-133	CEETM CQ Configure Command Format .....	3-205
3-134	CEETM CQ Configure Response Format .....	3-205
3-135	CEETM CQ Query Command Format .....	3-206
3-136	CEETM CQ Query Response Format.....	3-207
3-137	CEETM DCT Configure Command Format.....	3-208
3-138	CEETM DCT Configure Response Format .....	3-209
3-139	CEETM DCT Query Command Format .....	3-210
3-140	CEETM DCT Query Response Format .....	3-210
3-141	CEETM Class Scheduler Configure Command Format .....	3-211
3-142	CEETM Class Scheduler Configure Response Format .....	3-213
3-143	CEETM Class Scheduler Query Command Format .....	3-213
3-144	CEETM Class Scheduler Query Response Format.....	3-214
3-145	CEETM Channel Mapping Configure Command Format .....	3-215
3-146	CEETM Channel Mapping Configure Response Format .....	3-216
3-147	CEETM Channel Mapping Query Command Format .....	3-217
3-148	CEETM Channel Mapping Query Response Format .....	3-218
3-149	CEETM Sub-Portal Mapping Configure Command Format .....	3-218
3-150	CEETM Sub-Portal Mapping Configure Response Format .....	3-219
3-151	CEETM Sub-Portal Mapping Query Command Format .....	3-220
3-152	CEETM Sub-Portal Mapping Query Response Format.....	3-221
3-153	CEETM Shaper Configure Command Format.....	3-222
3-154	CEETM Shaper Configure Response Format.....	3-223
3-155	CEETM Shaper Query Command Format.....	3-224
3-156	CEETM Shaper Query Response Format .....	3-225
3-157	CEETM Traffic Class Flow Control Configure Command Format.....	3-226
3-158	CEETM Traffic Class Flow Control Configure Response Format .....	3-227

# Figures

Figure Number	Title	Page Number
3-159	CEETM Traffic Class Flow Control Query Command Format .....	3-228
3-160	CEETM Traffic Class Flow Control Query Response Format .....	3-229
3-161	CEETM CCCR CM Configure Command Format.....	3-229
3-162	CEETM CCCR CM Configure Response Format .....	3-231
3-163	CEETM CCCR CM Query Command Format.....	3-232
3-164	CEETM CCCR CM Query Response Format .....	3-233
3-165	CEETM Query Congestion State Command Format.....	3-234
3-166	CEETM Query Congestion State Response Format .....	3-234
3-167	CEETM CQ Peek/Pop/XSFDR Read Command Format.....	3-235
3-168	CEETM CQ Peek/Pop/XSFDR Read Response Format .....	3-236
3-169	CEETM Statistics Query/Write Command Format .....	3-237
3-170	CEETM Statistics Query/Write Response Format.....	3-238
3-171	Order Restoration: Elapsed Time View of Packet Processing .....	3-245
3-172	Order Restoration: Elapsed Time View of Packet Processing (continued).....	3-246
3-173	Order Restoration Sequence Number Windows .....	3-250
3-174	WRED Curve .....	3-252
3-175	Example CEETM scheduling hierarchy for one logical network interface .....	3-267
3-176	CEETM Class Queue Channel.....	3-273
3-177	CEETM unshaped class queue channel .....	3-274
3-178	CEETM shaper token bucket update .....	3-277
3-179	CEETM Channel Scheduler.....	3-278
3-180	CEETM channel scheduler shaped fair queueing (ShFQ) .....	3-279
3-181	CEETM configuration summary 1 .....	3-288
3-182	CEETM configuration summary 2.....	3-289
3-183	CEETM default configuration after reset.....	3-289
3-184	WBFS: Class (Queue) Scheduling Mechanism & Packet Queueing Mechanism .....	3-293
3-185	WBFS Mechanism Example: Overweight class dequeue & underweight class enqueue... 3-295	
4-1	Buffer Manager (BMan) Block Diagram .....	4-3
4-2	BCSP Command Registers (BCSPn_CR).....	4-9
4-3	BCSP Response Registers (BCSPn_RRM).....	4-10
4-4	BCSP Release Command Ring Registers (BCSPn_RCRm).....	4-11
4-5	BCSP RCR Producer Index Cache-Enabled and Cache-Inhibited Registers (BCSPn_RCR_PI_CENA, BCSPn_RCR_PI_CINH).....	4-13
4-6	BCSP RCR Consumer Index Cache-Enabled and Cache-Inhibited Registers (BCSPn_RCR_CI_CENA, BCSPn_RCR_CI_CINH).....	4-14
4-7	BCSP RCR Interrupt Threshold Register (BCSPn_RCR_ITR) .....	4-15
4-8	BMan Software Portal Configuration Registers (BCSPn_CFG) .....	4-15
4-9	BCSP SCN0 Depletion State Change Interrupt Enable Registers (BCSPn_SCN0).....	4-16
4-10	BCSP SCN1 Depletion State Change Interrupt Enable Registers (BCSPn_SCN1).....	4-17
4-11	BCSP Interrupt Status Registers (BCSPn_ISR).....	4-18

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
4-12	BCSP Interrupt Enable and Interrupt Status Disable Registers (BCSPn_IER, BCSPn_ISDR) ... 4-18	
4-13	BCSP Interrupt Force Register (BCSPn_IFR).....	4-18
4-14	BCSP Interrupt Inhibit Registers (BCSPn_IIR).....	4-19
4-15	BMan IP Block Revision 1Register (BMAN_IP_REV_1).....	4-20
4-16	BMan IP Block Revision 2 Register (BMAN_IP_REV_2).....	4-20
4-17	Data Structure Extended Base Address Registers (FBPR_BARE).....	4-21
4-18	Data Structure Base Address Registers (FBPR_BAR) .....	4-22
4-19	Data Structure Attribute Registers (FBPR_AR) .....	4-22
4-20	BMan Isolation Context Identifier Register (BMAN_ICIDR) .....	4-23
4-21	BMan Source ID Register (BMAN_SRCIDR) .....	4-23
4-22	Buffer Manager (BMan) threshold depletion state changes .....	4-24
4-23	BMan Software Portal Depletion Threshold Register (BMAN_POOLn_SWDET, BMAN_POOLn_SWDXT, BMAN_POOLn_HWDET, BMAN_POOLn_HWDXT) ....	4-24
4-24	FBPR Free Pool Low Watermark Interrupt Threshold Register (FBPR_FP_LWIT) .....	4-25
4-25	BMan Software Portal Depletion Count Registers (BMAN_POOLn_SDCNT, BMAN_POOLn_HDCNT) .....	4-26
4-26	BMan Pool Content Register (BMAN_POOLn_CONTENT, FBPR_FPC).....	4-26
4-27	BMan Pool Content Registers (BMAN_POOLn_HDPTR, FBPR_HDPTR) .....	4-27
4-28	BMAN Performance Monitor Configuration Registers (CMD_PMn_CFG) .....	4-28
4-29	BMAN Performance Monitor Configuration Registers (CMD_PMn_CFG_CFIFO) .....	4-28
4-30	BMAN Performance Monitor Configuration Register (BMAN_FL_PM_CFG) .....	4-29
4-31	BMAN Idle Status Register (STATE_IDLE) .....	4-29
4-32	BMAN Stop Register (STATE_STOP) .....	4-30
4-33	BMan Error Interrupt Status Register (BMAN_ERR_ISR).....	4-31
4-34	BMan Error Interrupt Enable Register (BMAN_ERR_IER).....	4-31
4-35	BMan Error Interrupt Force Register (BMAN_ERR_IFR) .....	4-32
4-36	BMan Error Interrupt Inhibit Register (BMAN_ERR_IIR) .....	4-32
4-37	BMan Single Bit Error Threshold Register (BMAN_SBET) .....	4-33
4-38	BMan Single Bit Error Count Registers (BMAN_SBEC0-1) .....	4-33
4-39	Free Buffer Proxy Record Free Pool Count Register (BMAN_ECIR).....	4-34
4-40	BMan Error Fetch Capture Register (BMAN_CECR) .....	4-35
4-41	BMan Corruption Error Address Register (BMAN_CEAR) .....	4-36
4-42	BMan Error Fetch Capture Register (BMAN_AECR) .....	4-36
4-43	BMan Error Fetch Address Register (BMAN_AEAR) .....	4-37
4-44	BMAN Command FIFO Disable Registers (DEBUG_CFIFO) .....	4-37
4-45	BMAN State Machine Disable Registers (DEBUG_FSM) .....	4-38
4-46	Acquire Command Format.....	4-40
4-47	Acquire Response Format .....	4-42
4-48	Query Response Format.....	4-44
4-49	Release Command Format .....	4-48

# Figures

Figure Number	Title	Page Number
4-50	Interrupt Control Logic Flow .....	4-50
4-51	Buffer Manager (BMan) interrupt state changes .....	4-50
4-52	Free Buffer Proxy Record (FBPR) Using Buffer Pointers (BPs).....	4-56
5-1	FMan Interfaces Block Diagram.....	5-7
5-2	FMan Block Diagram.....	5-8
5-3	FMan User-Configurable Pipeline Example.....	5-11
5-4	Configurable Receive (Rx) Flows.....	5-20
5-5	FMan Receive (Rx) Functional Flow (Example).....	5-22
5-6	FMan Transmit (Tx) Functional Flow (Example) .....	5-25
5-7	FMan Offline Port Functional Flow (Example for Offline Parsing).....	5-27
5-8	Independent Mode Flow (Rx and Tx).....	5-28
5-9	FMan Host Command Flow.....	5-30
5-10	FMan Hardware Port Pages Address Space.....	5-36
5-11	FMan Hardware Port Pages Address Space.....	5-37
5-12	FQD[Context A] for FMan dequeue .....	5-39
5-13	Context A—A0 Field Description for Tx Port.....	5-41
5-14	Context A—A0 Field Description for Offline Port.....	5-41
5-15	Context A—A2 Field Description for Tx Port.....	5-42
5-16	Context A—A2 Field Description for Offline Port.....	5-43
5-17	Context B Structure.....	5-44
5-18	Frame Descriptor.....	5-45
5-19	Rx FD Status .....	5-46
5-20	Tx FD Command .....	5-50
5-21	Tx FD Status (Confirmation) .....	5-50
5-22	Offline Port FD Command.....	5-52
5-23	Offline Port FD Status.....	5-52
5-24	Host Command FD Command.....	5-56
5-25	Host Command FD Status.....	5-57
5-26	Internal Context Action Descriptor (ICAD) for Rx .....	5-58
5-27	Internal Context Action Descriptor (ICAD) for Tx or Offline .....	5-58
5-28	FMan Next Invoked Action (NIA) Structure .....	5-60
5-29	Rx Flows .....	5-62
5-30	Trace First 4 Bytes .....	5-65
5-31	Generic Debug Trace Configuration Register Model .....	5-67
5-32	Generic Debug Trap Configuration Register (GDTCR) Model.....	5-68
5-33	Generic Debug Trap Value Register (DVR) Model .....	5-69
5-34	Generic Debug Trap Mask Register Model .....	5-69
5-35	FMan Hardware Port Pages Address Space.....	5-80
5-36	Examples of virtual profile selection .....	5-89
5-37	BMI Initialization Register (FMBM_INIT).....	5-93
5-38	BMI Configuration 1 Register (FMBM_CFG1).....	5-94

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
5-39	BMI Configuration 2 Register (FMBM_CFG2).....	5-96
5-40	Interrupt Event Register (FMBM_IEVR).....	5-97
5-41	Interrupt Enable Register (FMBM_IER).....	5-97
5-42	Interrupt Force Register (FMBM_IFR).....	5-98
5-43	Debug Trap Counter Register (FMBM_DTC).....	5-99
5-44	Debug Compare Value Register (FMBM_DCV).....	5-100
5-45	Debug Compare Mask (FMBM_DCM).....	5-100
5-46	Global Debug Enable (FMBM_GDE).....	5-101
5-47	Port Parameters Register (FMBM_PP_1-63).....	5-102
5-48	Port FIFO Size Register (FMBM_PFS_1-63).....	5-105
5-49	Storage Profile and Isolation Context Identifier Register (FMBM_SPICID).....	5-107
5-50	Rx Configuration Register (FMBM_RCFG).....	5-108
5-51	Rx Status Register (FMBM_RST).....	5-109
5-52	Rx DMA Attributes Register (FMBM_RDA).....	5-111
5-53	Rx FIFO Parameters Register (FMBM_RFP).....	5-112
5-54	Rx Frame End Data Register (FMBM_RFED).....	5-113
5-55	Rx Internal Context Parameters (FMBM_RICP).....	5-115
5-56	Rx Internal Margins Register (FMBM_RIM).....	5-116
5-57	Rx External Buffer Margins Register (FMBM_REBM).....	5-117
5-58	Rx Frame Next Engine Register (FMBM_RFNE).....	5-118
5-59	Rx Frame Attributes Register (FMBM_RFCA).....	5-119
5-60	Rx Frame Parser Next Engine Register (FMBM_RFPNE).....	5-120
5-61	Rx Parsing Start Offset Register (FMBM_RPSO).....	5-121
5-62	Rx Policer Profile Register (FMBM_RPP).....	5-121
5-63	Rx Parse Result Initialization Register (FMBM_RPRI).....	5-122
5-64	Rx Frame Queue ID Register (FMBM_RFQID).....	5-124
5-65	Rx Error Frame Queue ID Register (FMBM_REFQID).....	5-124
5-66	Rx Frame Status Discard Mask Register (FMBM_RFSDM).....	5-125
5-67	Rx Frame Status Error Mask Register (FMBM_RFSEM).....	5-126
5-68	Rx Frame Enqueue Next Engine Register (FMBM_RFENE).....	5-126
5-69	Rx Continuous Mode Next Engine (FMBM_RCMNE).....	5-127
5-70	External Buffers Manager Pool Information Register (FMBM_REBMPI).....	5-128
5-71	Allocate Counter Register (FMBM_RACNT).....	5-129
5-72	Receive Congestion Group Map Register (FMBM_RCGM).....	5-130
5-73	BMan Pool Depletion Register (FMBM_RMPD).....	5-131
5-74	Rx Statistics Counters Register (FMBM_RSTC).....	5-132
5-75	Rx Frame Counter Register (FMBM_RFRC).....	5-133
5-76	Rx Bad Frames Counter Register (FMBM_RBFC).....	5-134
5-77	Rx Large Frames Counter Register (FMBM_RLFC).....	5-134
5-78	Rx Filter Frames Counter Register (FMBM_RFFC).....	5-135
5-79	Rx Frames Discard Counter Register (FMBM_RFDC).....	5-136

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
5-80	Rx Frames List DMA Error Counter Register (FMBM_RFLDEC) .....	5-136
5-81	Rx Out of Buffers Discard Counter Register (FMBM_RODC) .....	5-137
5-82	Rx Buffers Deallocate Counter Register (FMBM_RBDC) .....	5-138
5-83	Rx Prepare to Enqueue (FMBM_RPEC) .....	5-138
5-84	Rx Performance Counters Register (FMBM_RPC) .....	5-139
5-85	Rx Performance Count Parameters Register (FMBM_RPCP) .....	5-140
5-86	Rx Cycle Counter Register (FMBM_RCCN) .....	5-141
5-87	Rx Tasks Utilization Counter Register (FMBM_RTUC) .....	5-142
5-88	Rx Receive Queue Utilization Counter Register (FMBM_RRQUC) .....	5-142
5-89	Rx DMA Utilization Counter Register (FMBM_RDUC) .....	5-143
5-90	Rx FIFO Utilization Counter Register (FMBM_RFUC) .....	5-144
5-91	Rx Pause Activation Counter Register (FMBM_RPAC) .....	5-144
5-92	Rx Debug Configuration Register (FMBM_RDCFG) .....	5-145
5-93	Rx General Purpose Register (FMBM_RGPR) .....	5-147
5-94	Tx Configuration Register (FMBM_TCFG) .....	5-147
5-95	Tx Status Register (FMBM_TST) .....	5-148
5-96	Tx DMA Attributes Register (FMBM_TDA) .....	5-149
5-97	Tx FIFO Parameters Register (FMBM_TFP) .....	5-150
5-98	Tx Frame End Data Register (FMBM_TFED) .....	5-151
5-99	Tx Internal Context Parameters Register (FMBM_TICP) .....	5-152
5-100	Tx Frame Dequeue Next Engine Register (FMBM_TFDNE) .....	5-153
5-101	Tx Frame Attributes Register (FMBM_TFCA) .....	5-154
5-102	Tx Confirmation Frame Queue ID Register (FMBM_TCFQID) .....	5-155
5-103	Tx Error Frame Queue ID Register (FMBM_TEFAQID) .....	5-156
5-104	Tx Frame Enqueue Next Engine Register (FMBM_TFENE) .....	5-156
5-105	Tx Rate Limiter Scale Register (FMBM_TRLMTS) .....	5-157
5-106	Tx Rate Limiter Register (FMBM_TRLMT) .....	5-158
5-107	Tx Custom Classifier Base Register (FMBM_TCCB) .....	5-159
5-108	Tx Frame Next Engine Register (FMBM_TFNE) .....	5-160
5-109	Tx PFC Mapping Register 0 (FMBM_TPFCM0) .....	5-161
5-110	Tx Continuous Mode Next Engine (FMBM_TCMNE) .....	5-162
5-111	Tx Statistics Counters Register (FMBM_TSTC) .....	5-162
5-112	Tx Frame Counter Register (FMBM_TFRC) .....	5-163
5-113	Tx Frames Discard Counter Register (FMBM_Tfdc) .....	5-163
5-114	Tx Frames Length Error Discard Counter Register (FMBM_Tfledc) .....	5-164
5-115	Tx Frames Unsupported Format Discard Counter Register (FMBM_Tfufdc) .....	5-165
5-116	Tx Buffers Deallocate Counter Register (FMBM_TBDC) .....	5-165
5-117	Tx Performance Counters Register (FMBM_TPC) .....	5-166
5-118	Tx Performance Count Parameters Register (FMBM_TPCP) .....	5-167
5-119	Tx Cycle Counter Register (FMBM_TCCN) .....	5-168
5-120	Tx Tasks Utilization Counter Register (FMBM_TTUC) .....	5-169

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
5-121	Tx Transmit Confirm Queue Utilization Counter Register (FMBM_TTCQUC) .....	5-169
5-122	Tx DMA Utilization Counter Register (FMBM_TDUC) .....	5-170
5-123	Tx FIFO Utilization Counter Register (FMBM_TFUC) .....	5-171
5-124	Tx Debug Configuration Register (FMBM_TDCFG) .....	5-172
5-125	Tx General Purpose Register (FMBM_TGPR) .....	5-173
5-126	Offline Port/Host Command (O/H) Configuration Register (FMBM_OCFG).....	5-174
5-127	O/H Status Register (FMBM_OST).....	5-175
5-128	O/H DMA Attributes Register (FMBM_ODA).....	5-176
5-129	O/H Internal Context Parameters Register (FMBM_OICP).....	5-177
5-130	O/H Frame Dequeue Next Engine Register (FMBM_OFDNE) .....	5-179
5-131	O/H Frame Dequeue Next Engine Register (FMBM_OFNE) .....	5-179
5-132	O/H Frame Attributes Register (FMBM_OFCA) .....	5-180
5-133	O/H Frame Dequeue Next Engine Register (FMBM_OFPNE).....	5-181
5-134	O/H Parsing Start Offset Register (FMBM_OPPO) .....	5-182
5-135	O/H Policer Profile Register (FMBM_OPP) .....	5-182
5-136	O/H Custom Classifier Base Register (FMBM_OCCB) .....	5-183
5-137	O/H Internal Margins Register (FMBM_OIM) .....	5-184
5-138	O/H FIFO Parameters Register (FMBM_OFP) .....	5-185
5-139	O/H Frame End Data Register (FMBM_OFED) .....	5-186
5-140	O/H Parse Result Initialization Register (FMBM_OPRI) .....	5-186
5-141	O/H Frame Queue ID Register (FMBM_OFQID).....	5-187
5-142	O/H Error Frame Queue ID Register (FMBM_OEFQID).....	5-188
5-143	O/H Frame Status Discard Mask Register (FMBM_OFSDM).....	5-189
5-144	O/H Frame Status Error Mask Register (FMBM_OFSEM) .....	5-190
5-145	O/H Frame Enqueue Next Engine Register (FMBM_OFENE).....	5-190
5-146	O/H Rate Limiter Scale Register (FMBM_ORLMTS).....	5-191
5-147	O/H Rate Limiter Register (FMBM_ORLMT) .....	5-192
5-148	O/H Continuous Mode Next Engine (FMBM_OCMNE).....	5-194
5-149	FMBM_OCGM - Observed Congestion Group Map .....	5-195
5-150	O/H Statistics Counters Register (FMBM_OSTC).....	5-195
5-151	O/H Frame Counter Register (FMBM_OFRC) .....	5-196
5-152	O/H Frames Discard Counter Register (FMBM_OFDC) .....	5-197
5-153	O/H Frames Length Error Discard Counter Register (FMBM_OFLEDC) .....	5-197
5-154	O/H Frames Unsupported Format Discard Counter Register (FMBM_OFUFDC).....	5-198
5-155	O/H Filter Frames Counter Register (FMBM_OFFC) .....	5-199
5-156	O/H Frames WRED Discard Counter Register (FMBM_OFWDC) .....	5-199
5-157	O/H Frames List DMA Error Counter Register (FMBM_OFLDEC).....	5-200
5-158	O/H Buffers Deallocation Counter Register (FMBM_OBDC).....	5-201
5-159	O/H Out of Buffers Discard Counter Register (FMBM_OODC).....	5-202
5-160	O/H prepare to enqueue Counter Register (FMBM_OPEC) .....	5-202
5-161	O/H Performance Counters Register (FMBM_OPC) .....	5-203

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
5-162	O/H Performance Count Parameters Register (FMBM_OPCP).....	5-204
5-163	O/H Cycle Counter Register (FMBM_OCCN) .....	5-205
5-164	O/H Tasks Utilization Counter Register (FMBM_OTUC).....	5-205
5-165	O/H DMA Utilization Counter Register (FMBM_ODUC) .....	5-206
5-166	O/H FIFO Utilization Counter Register (FMBM_OFUC) .....	5-207
5-167	O/H Debug Configuration Register (FMBM_ODCFG) .....	5-208
5-168	O/H General Purpose Register (FMBM_OGPR).....	5-209
5-169	Maximum Frame Size to Fit in One External Buffer.....	5-221
5-170	External Buffer Start Margin Calculation .....	5-223
5-171	Internal and External Margins, Single Buffer Frame Format .....	5-223
5-172	Internal and External Margins, Scatter/Gather Frame Format.....	5-223
5-173	Tx Confirmation Enqueue Decision Flow .....	5-229
5-174	Tx Confirmation Enqueue Decision Flow for FMan_v3 .....	5-230
5-175	Host Command Enqueue Decision Flow .....	5-232
5-176	BMI Token Bucket Mechanism .....	5-236
5-177	QMI Connection Block Diagram .....	5-247
5-178	FMan Hardware Port Pages Address Space.....	5-249
5-179	QMI General Configuration Register (FMQM_GC) .....	5-252
5-180	Error Interrupt Event Register (FMQM_EIE) .....	5-253
5-181	Error Interrupt Enable Register (FMQM_EIEN).....	5-253
5-182	Error Interrupt Force Register (FMQM{EIF}) .....	5-254
5-183	QMI Global Status Register (FMQM_GS).....	5-255
5-184	Enqueue Total Frame Counter Register (FMQM_ETFC) .....	5-256
5-185	Dequeue Total Frame Counter Register (FMQM_DTFC).....	5-256
5-186	Dequeue Counter 0 Register (FMQM_DC0).....	5-257
5-187	Debug Trace Configuration Register (FMQM_DTRC).....	5-257
5-188	Enqueue Frame Descriptor Dynamic Debug Register (FMQM_EFDDD).....	5-259
5-189	Debug Trap Configuration <i>n</i> Register (FMQM_DTC <i>n</i> ) .....	5-259
5-190	Debug Trap Value <i>n</i> Register (FMQM_DTV <i>n</i> ) .....	5-261
5-191	Debug Trap Mask <i>n</i> Register (FMQM_DTM <i>n</i> ).....	5-261
5-192	Debug Trap Counter <i>n</i> Register (FMQM_DTC <i>n</i> ) .....	5-262
5-193	PortID <i>n</i> Configuration Register (FMQM_PnC) .....	5-262
5-194	PortID <i>n</i> Status Register (FMQM_PnS) .....	5-263
5-195	PortID <i>n</i> Task Status Register (FMQM_PnTS) .....	5-264
5-196	PortID <i>n</i> Enqueue NIA Register (FMQM_PnEN) .....	5-264
5-197	PortID <i>n</i> Enqueue NIA Register (FMQM_PnEN) .....	5-265
5-198	PortID <i>n</i> Enqueue Total Frame Counter Register (FMQM_PnETFC) .....	5-266
5-199	PortID <i>n</i> Dequeue NIA Register (FMQM_PnDN) .....	5-266
5-200	PortID <i>n</i> Dequeue NIA Register (FMQM_PnDN) .....	5-267
5-201	PortID <i>n</i> Dequeue Config Register (FMQM_PnDC) .....	5-267
5-202	PortID <i>n</i> Dequeue Total Frame Counter Register (FMQM_PnDTFC).....	5-269

# Figures

Figure Number	Title	Page Number
5-203	PortID $n$ Dequeue FQID Not Override Counter Register (FMQM_PnDFNOC) .....	5-270
5-204	PortID $n$ Dequeue Confirm Counter Register (FMQM_PnDCC) .....	5-270
5-205	FMan hardware ports and QMan sub-portals .....	5-272
5-206	FPM PortID Control Register (FMFP_PRC).....	5-282
5-207	FPM Maximum Dispatches Register (FMFP_MXD).....	5-283
5-208	FPM Dispatch Thresholds 1 Register (FMFP_DIST1).....	5-283
5-209	FPM Dispatch Thresholds 2 Register (FMFP_DIST2).....	5-284
5-210	FMan Error Pending Interrupt Register (FM_EPI).....	5-285
5-211	FMan RAMs Interrupt Enable Register (FM_RIE) .....	5-288
5-212	FPM FMan Controller Event 0–3 Registers (FMFP_FCE $n$ ) .....	5-289
5-213	FPM FMan Controller Event Enable 0–3 Registers (FMFP_CEE $n$ ) .....	5-289
5-214	FPM TimeStamp Control 1 Register (FMFP_TSC1) .....	5-290
5-215	FPM TimeStamp Control 2 Register (FMFP_TSC2) .....	5-290
5-216	FPM Time Stamp Register (FMFP_TSP) .....	5-291
5-217	FPM TimeStamp Fraction Register (FMFP_TS $f$ ) .....	5-291
5-218	FMan RAMs Control and Event Register (FM_RCR) .....	5-292
5-219	FPM External Requests Control Register (FMFP_EXTC) .....	5-293
5-220	FPM Data RAM Data 0–3 Registers (FMFP_DRD $n$ ) .....	5-294
5-221	FM_DECES Configuration .....	5-294
5-222	FPM Data RAM Access Register (FMFP_DRA) .....	5-296
5-223	FMan IP Block Revision 1 Register (FM_IP_REV_1) .....	5-297
5-224	FMan IP Block Revision 2 Register (FM_IP_REV_2) .....	5-297
5-225	FMan Reset Command Register (FM_RSTC) .....	5-298
5-226	FPM Controller Debug Control Register (FMFP_CLDC) .....	5-298
5-227	FMan Normal Pending Interrupt Register (FM_NPI) .....	5-301
5-228	FPM Event and Enable Register (FMFP_EE) .....	5-304
5-229	FPM CPU Event 0–3 Registers (FMFP_CE $Vn$ ) .....	5-305
5-230	FPM PortID Status 0–49 Registers (FMFP_PS $n$ ) .....	5-306
5-231	FMan Controller Flow AB Control Register (FMFP_CLFABC) .....	5-307
5-232	FMan Controller Flow C Control Register (FMFP_CLFCC) .....	5-308
5-233	FMan Controller Flow A Value Register (FMFP_CLFAVAL) .....	5-309
5-234	FMan Controller Flow B Value Register (FMFP_CLFBVAL) .....	5-309
5-235	FMan Controller Flow C Value Register (FMFP_CLFCVAL) .....	5-310
5-236	FMan Controller Flow A Mask Register (FMFP_CLFAMSK) .....	5-310
5-237	FMan Controller Flow B Mask Register (FMFP_CLFBMSK) .....	5-311
5-238	FMan Controller Flow C Mask Register (FMFP_CLFCMSK) .....	5-311
5-239	FMan Controller Flow A Match Count Register (FMFP_CLFAMC) .....	5-312
5-240	FMan Controller Flow B Match Count Register (FMFP_CLFBMC) .....	5-312
5-241	FMan Controller Flow C Match Count Register (FMFP_CLFCMC) .....	5-313
5-242	FM_DECCH Configuration .....	5-313
5-243	FPM TNUM Status 0–127 Registers (FMFP_TS $n$ ) .....	5-315

# Figures

Figure Number	Title	Page Number
5-244	Timestamp & Prescale Logic .....	5-317
5-245	DMA in the FMan Block Diagram .....	5-319
5-246	FMan DMA Status Register (FMDM_SR) .....	5-321
5-247	FMan DMA Mode Register (FMDM_MR) .....	5-323
5-248	DMA Read Data Path .....	5-324
5-249	DMA Write Data Path .....	5-325
5-250	DMA Command Queue .....	5-325
5-251	FMan DMA Threshold Register (FMDM_TR) .....	5-325
5-252	FMan DMA Hysteresis Register (FMDM_HY) .....	5-326
5-253	FMan DMA SOS Emergency Threshold Register (FMDM_SETR) .....	5-327
5-254	FMan DMA Transfer Address High Register (FMDM_TAH) .....	5-327
5-255	FMan DMA Transfer Address Low Register (FMDM_TAL) .....	5-328
5-256	FMan DMA Transfer Communication ID Register (FMDM_TCID) .....	5-328
5-257	FMan DMA buffer Watchdog Counter Value (FMDM_WCR) .....	5-329
5-258	FMan DMA buffer Base in FMan Memory Value Register (FMDM_EBCR) .....	5-330
5-259	FMan DMA Debug Counter (FMDM_DCR) .....	5-330
5-260	FMan DMA Emergency Smoother Register (FMDM_EMSR) .....	5-331
5-261	Hardware Port (PortID) Registers (FMDM_PortID) .....	5-332
5-262	Port $x$ Configuration Access Control Register (FMPR_PxCAC) .....	5-342
5-263	Port $x$ Configured TPID Register (FMPR_PxCTPID) .....	5-343
5-264	Soft Examination Parameter Array W $x$ Register (FMPR_SXPAW $\bar{x}$ ) .....	5-343
5-265	Rx Parsing Cycle Limit Register (FMPR_RPCLIM) .....	5-344
5-266	Rx Parse Internal Memory Access Control Register (FMPR_RPIMAC) .....	5-344
5-267	Parse Memory ECC Error Capture Register (FMPR_PMEEC) .....	5-345
5-268	Parser Event Register (FMPR_PEVR) .....	5-346
5-269	Parser Event Enable Register (FMPR_PEVER) .....	5-347
5-270	Parser Error Register (FMPR_PERR) .....	5-348
5-271	Parser Error Enable Register (FMPR_PERER) .....	5-348
5-272	Per Port Parser Statistic Control Register (FMPR_PPSC) .....	5-349
5-273	Parse Dispatch Statistic Register (FMPR_PDS) .....	5-350
5-274	L2 Result Returned Statistic Register (FMPR_L2RRS) .....	5-350
5-275	L3 Result Returned Statistic Register (FMPR_L3RRS) .....	5-351
5-276	L4 Result Returned Statistic Register (FMPR_L4RRS) .....	5-351
5-277	Shim Result Returned Statistic Register (FMPR_SRRES) .....	5-351
5-278	L2 Result Returned with Error Statistic Register (FMPR_L2RRES) .....	5-352
5-279	L3 Result Returned with Error Statistic Register (FMPR_L3RRES) .....	5-352
5-280	L4 Result Returned with Error Statistic Register (FMPR_L4RRES) .....	5-352
5-281	Shim Result Returned with Error Statistic Register (FMPR_SRRES) .....	5-353
5-282	Soft Parser Cycle Statistic Register (FMPR_SPCS) .....	5-353
5-283	Soft Parser Stall Cycle Statistic Register (FMPR_SPSCS) .....	5-353
5-284	HXS Cycle Statistic Register (FMPR_HXSCS) .....	5-354

# Figures

Figure Number	Title	Page Number
5-285	FMan Memory Read Cycle Statistic Register (FMPR_MRCS) .....	5-354
5-286	FMan Memory Write Cycle Statistic Register (FMPR_MWCS) .....	5-354
5-287	FMan Memory Read Stall Cycle Statistic Register Format.....	5-355
5-288	FMan Memory Write Stalled Cycle Statistic Register (FMPR_MWSCS) .....	5-355
5-289	<b>FPM Command Stall Cycle Statistic Register (FMPR_FCSCS)</b> .....	<b>5-355</b>
5-290	Parser Debug Flow <i>x</i> Trap Event Statistic Registers (FMPR_PDxTES) .....	5-356
5-291	Parser Debug Control Register (FMPR_PDC) .....	5-356
5-292	Parser Debug Flow <i>x</i> Trap <i>y</i> Configuration Registers (FMPR_PDxTyC).....	5-357
5-293	Parser Debug Flow <i>x</i> Trap <i>y</i> Value Registers (FMPR_PDxTyV).....	5-358
5-294	Parser Debug Flow <i>x</i> Trap <i>y</i> Mask Registers (FMPR_PDxTyM) .....	5-359
5-295	HPNIA Flow .....	5-364
5-296	Hard Parse Tree.....	5-365
5-297	Shell .....	5-366
5-298	Soft Extension .....	5-367
5-299	Line-Up Confirmation Example .....	5-372
5-300	Ethernet Frame Formats.....	5-379
5-301	VLAN Tag.....	5-381
5-302	MPLS (RFC3032).....	5-386
5-303	GRE (RFC2890) .....	5-393
5-304	Minimal Encapsulation for IP .....	5-394
5-305	TCP (RFC793) .....	5-399
5-306	UDP (RFC768) .....	5-400
5-307	AH (RFC2402).....	5-401
5-308	ESP (RFC2406).....	5-401
5-309	SCTP (RFC2960).....	5-402
5-310	DCCP (RFC4340) .....	5-402
5-311	KeyGen General Configuration Register (FMKG_GCR) .....	5-418
5-312	KeyGen Error Event Register (FMKG_EER) .....	5-418
5-313	KeyGen Error Event Enable Register (FMKG_EEER).....	5-419
5-314	KeyGen Scheme Error Event Register (FMKG_SEER).....	5-420
5-315	KeyGen Global Status Register (FMKG_GSR) .....	5-420
5-316	KeyGen Total Packet Counter Register (FMKG_TPC).....	5-421
5-317	KeyGen Soft Error Capture Register (FMKG_SERC) .....	5-421
5-318	KeyGen Frame Data Offset Register (FMKG_FDOR) .....	5-422
5-319	KeyGen Global Default Value Register (FMKG_GDV<i>R).....	5-423
5-320	KeyGen Force Error Event Register (FMKG_FEER) .....	5-423
5-321	KeyGen Action Register (FMKG_AR) .....	5-424
5-322	KeyGen Scheme Entry MODE Register (AWR1_RFMODE_FMKG_SE_MODE) .....	5-426
5-323	KeyGen Scheme Entry Extract Known Fields Command Register (AWR2_RFMODE_FMKG_SE_EKFC) .....	5-427

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
5-324	KeyGen Scheme Entry Extract Known Default Value Register (AWR3_RFMODE_FMKG_SE_EKDV) .....	5-431
5-325	KeyGen Scheme Bit Mask Command High Register (AWR4_RFMODE_FMKG_SE_BMCH) .....	5-434
5-326	KeyGen Scheme Bit Mask Command Low Register (AWR5_RFMODE_FMKG_SE_BMCL). .... 5-435	
5-327	Bit Mask Command Example .....	5-436
5-328	KeyGen Scheme Entry Frame Queue Base Register (AWR6_RFMODE_FMKG_SE_FQB).... 5-436	
5-329	KeyGen Scheme Entry Hash Configuration Register (AWR7_RFMODE_FMKG_SE_HC)..... 5-437	
5-330	KeyGen Scheme Entry Policer Profile Command Register (AWR8_RFMODE_FMKG_SE_PPC) .....	5-438
5-331	KeyGen Scheme Entry Generic Extract Command Register (AWR<9+i>_RFMODE_FMKG_SE_GEC<i>) .....	5-440
5-332	Key Generation Command Description .....	5-442
5-333	KeyGen Scheme Entry Statistic Packet Counter Register (AWR17_RFMODE_FMKG_SE_SPC) .....	5-443
5-334	KeyGen Scheme Entry Default Value Register (AWR<18+i>_RFMODE_FMKG_SE_DV<i>) .. 5-443	
5-335	KeyGen Scheme Entry Custom Classifier Bit Select Register (AWR20_RFMODE_FMKG_SE_CCBS) .....	5-444
5-336	KeyGen Scheme Entry Match Vector Register (AWR21_RFMODE_FMKG_SE_MV) ...	5-444
5-337	KeyGen Scheme Entry Match Vector Register (AWR21_RFMODE_FMKG_SE_OM) ...	5-445
5-338	KeyGen Scheme Entry Virtual Storage Profile Command Register (AWR8_RFMODE_FMKG_SE_VSP) .....	5-446
5-339	Classification Plan Tables .....	5-447
5-340	KeyGen Classification Plan Entry Register (AWR<1+i>_RFMODE_FMKG_CPE<i>) ..	5-448
5-341	KeyGen Port Entry Scheme Partition Register (AWR1_RFMODE_FMKG_PE_SP) .....	5-448
5-342	KeyGen Port Entry Classification Plan Partition Register (AWR2_RFMODE_FMKG_PE_CPP) .....	5-449
5-343	KeyGen Debug Control Register (FMKG_DCR) .....	5-450
5-344	KeyGen Debug Flow <j> Trap Counter Register (FMKG_D<j>TC) .....	5-451
5-345	KeyGen Debug Flow <j> Trap <k> Configuration Register (FMKG_D<j>T<k>CR) ..	5-452
5-346	KeyGen Debug Flow <j> Trap <k> Value Register (FMKG_D<j>T<k>VR) .....	5-454
5-347	KeyGen Debug Flow <j> Trap <k> Mask Register (FMKG_D<j>T<k>MR) .....	5-454
5-348	KeyGen Functional Flow .....	5-456
5-349	Scheme Selection Flow .....	5-458
5-350	FQID Generation Flow .....	5-463
5-351	Custom Classifier Offset Calculation.....	5-465
5-352	KeyGen NIA Bit Mapping .....	5-466

# Figures

Figure Number	Title	Page Number
5-353	FMan Policer General Configuration Register (FMPL_GCR) .....	5-479
5-354	FMan Policer Global Status Register (FMPL_GSR) .....	5-480
5-355	FMan Policer Event Register (FMPL_EVR) .....	5-482
5-356	FMan Policer Interrupt Enable Register (FMPL_IER) .....	5-482
5-357	FMan Policer Interrupt Force Register (FMPL_IFR) .....	5-483
5-358	FMan Policer Error Event Register (FMPL_EEVR) .....	5-483
5-359	FMan Policer Error Interrupt Enable Register (FMPL_EIER) .....	5-484
5-360	FMan Policer RED Packet Counter Register (FMPL_RPC) .....	5-485
5-361	FMan Policer YELLOW Packet Counter Register (FMPL_YPC) .....	5-485
5-362	FMan Policer Recolored RED Packet Counter Register (FMPL_RRPC) .....	5-486
5-363	FMan Policer Recolored YELLOW Packet Counter Register (FMPL_RYPC) .....	5-486
5-364	FMan Policer Total Packet Counter Register (FMPL_TPC) .....	5-487
5-365	FMan Policer Frame Length Mismatch Counter Register (FMPL_FLMC) .....	5-487
5-366	FMan Policer Profile Action Register (FMPL_PAR) .....	5-488
5-367	FMan Policer Profile Entry Access Word Registers (FMPL_PE*) .....	5-490
5-368	FMan Policer Soft Error Capture Register (FMPL_SERC) .....	5-490
5-369	FMan Policer Uninitialized Profile Capture Register (FMPL_UPCR) .....	5-491
5-370	FMan Policer Debug Trace Configuration Register (FMPL_DTRCR) .....	5-492
5-371	FMan Policer Flow A Debug Trap Configuration Registers (FMPL_FADBTCR $n$ ) .....	5-494
5-372	FMan Policer Flow A Debug Value Registers (FMPL_FADBVALR $n$ ) .....	5-495
5-373	FMan Policer Flow A Debug Trap Mask Registers (FMPL_FADBTMR $n$ ) .....	5-496
5-374	FMan Policer Flow A Debug Trap Match Counter Register (FMPL_FADBTMC) .....	5-496
5-375	FMan Policer Flow B Debug Trap Configuration Registers (FMPL_FBDBTCR $n$ ) .....	5-497
5-376	FMan Policer Flow B Debug Value Registers (FMPL_FBDBVALR $n$ ) .....	5-498
5-377	FMan Policer Flow B Debug Trap Mask Registers (FMPL_FBDBTMR $n$ ) .....	5-499
5-378	FMan Policer Flow B Debug Trap Match Counter Register (FMPL_FBDBTMC) .....	5-499
5-379	FMan Policer Flow C Debug Trap Configuration Registers (FMPL_FCDBTCR $n$ ) .....	5-500
5-380	FMan Policer Flow C Debug Value Registers (FMPL_FCDBVALR $n$ ) .....	5-501
5-381	FMan Policer Flow C Debug Trap Mask Registers (FMPL_FCDBTMR $n$ ) .....	5-502
5-382	FMan Policer Flow C Debug Trap Match Counter Register (FMPL_FCDBTMC) .....	5-502
5-383	FMan Policer Default Profile Mapping Register (FMPL_DPMR) .....	5-503
5-384	FMan Policer Profile Mapping Registers (FMPL_PMR $n$ ) .....	5-504
5-385	FMan Policer Profile Entry MODE Configuration Word Register (MODE) .....	5-506
5-386	GREEN Next Invoked Action Configuration Word Register (GNIA) .....	5-509
5-387	YELLOW Next Invoked Action Address Configuration Word Register (YNIA) .....	5-509
5-388	RED Next Invoked Action Configuration Word (RNIA) .....	5-510
5-389	Committed Information Rate Configuration Word (CIR) .....	5-511
5-390	Committed Burst Size Configuration Word (CBS) .....	5-511
5-391	Peak/Excess Information Rate Configuration Word (PIR_EIR) .....	5-512
5-392	Peak/Excess Burst Size Configuration Word Register (PBS_EBS) .....	5-512
5-393	Last Timestamp Variable Register (LTS) .....	5-513

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
5-394	Committed Token-Bucket Status Variable Register (CTS).....	5-513
5-395	Peak/Excess Token-Bucket Status Variable Register (PTS_ETS) .....	5-514
5-396	GREEN Packet Counter Variable Register (GPC).....	5-514
5-397	YELLOW Packet Counter Variable Register (YPC) .....	5-515
5-398	RED Packet Counter Variable Register (RPC) .....	5-515
5-399	Recolored YELLOW Packet Counter Variable Register (RYPC) .....	5-515
5-400	Recolored RED Packet Counter Variable Register (RRPC) .....	5-516
5-401	Token Bucket Concept .....	5-521
5-402	RFC-2968 Token Buckets Implementation.....	5-523
5-403	Load Spreading Application Example .....	5-539
5-404	Double Policing Scheme by Profile Concatenation .....	5-540
5-405	Aggregating Multiple Streams by Profile Concatenation .....	5-541
5-406	FMan Controller Configuration Data, Address Register .....	5-550
5-407	FMan Controller Configuration Data, Data Register .....	5-551
5-408	FMan Controller Configuration Data Ready Register .....	5-551
5-409	Matching Table Structure .....	5-553
5-410	KeyGen/Custom Classifier HASH Look-Up Flow.....	5-554
5-411	IC Look-Up with Aging mechanism.....	5-555
5-412	Basic Header Manipulation Flow .....	5-556
5-413	Custom Classifier and HM Generic Flow Example.....	5-557
5-414	New Classification Result Action Descriptor (AD) (Type = 00).....	5-558
5-415	Keep Classification Action Descriptor (Type = 10).....	5-561
5-416	Table Descriptor (Type = 01) .....	5-563
5-417	HM Table Descriptor (Type = 01).....	5-569
5-418	General Header Manipulation Command Descriptor (HMCD).....	5-571
5-419	Header Removal Command .....	5-572
5-420	Header Removal Command Descriptor .....	5-572
5-421	Local Header Insert Command Descriptor .....	5-573
5-422	Internal Header Insert Command Descriptor .....	5-574
5-423	Local Header Replace Command Descriptor.....	5-574
5-424	Internal Header Replace Command Descriptor .....	5-575
5-425	Protocol Specific Header Removal Command Descriptor .....	5-576
5-426	Internal Protocol Specific Header Insert Command Descriptor .....	5-578
5-427	VLAN Priority Update Command Descriptor .....	5-579
5-428	IP_DSCP_to_VPri_Table.....	5-580
5-429	Local IPv4 Update Command Descriptor .....	5-581
5-430	Local TCP/UDP Update Command Descriptor .....	5-583
5-431	Local IPv6 Update Command Descriptor .....	5-584
5-432	Internal IP Header replace Command Descriptor .....	5-585
5-433	UDP/TCP Checksum Calculation Command Descriptor .....	5-587
5-434	Remove header till known parser result Command Descriptor .....	5-588

# Figures

Figure Number	Title	Page Number
5-435	Local Insert UDP or UDP-Lite Header Command Descriptor .....	5-588
5-436	Local Insert L3 Header Command Descriptor .....	5-589
5-437	Remove/Local Insert CAPWAP Header Command Descriptor.....	5-591
5-438	Replace Field in Header Command Descriptor .....	5-591
5-439	Parse after HM Generic Flow Example .....	5-595
5-440	Deep Sleep Auto Response main flow.....	5-597
5-441	Auto Response Programming Model.....	5-602
5-442	TCP/UDP port table entry .....	5-603
5-443	Deep Sleep Auto Response Statistics table.....	5-604
5-444	Deep Sleep ARP Flow Illustration.....	5-607
5-445	ARP Descriptor .....	5-607
5-446	Deep Sleep Auto Response ARP Data Structures illustration .....	5-609
5-447	ICMPv4 Descriptor .....	5-611
5-448	Deep Sleep Auto Response ICMPv4 Data Structures illustration .....	5-613
5-449	ICMPv6 Descriptor .....	5-615
5-450	Deep Sleep Auto Response ICMPv6 Data Structures illustration .....	5-617
5-451	ND Descriptor .....	5-621
5-452	Deep Sleep Auto Response Neighbor Discovery Data Structures illustration .....	5-624
5-453	SNMP message format.....	5-625
5-454	SNMP Descriptor .....	5-627
5-455	OIDs table entry .....	5-630
5-456	SNMP statistics table .....	5-631
5-457	Old AD First 4 Bytes Temporary Value (Type = 11) .....	5-632
5-458	Independent Mode Flow .....	5-635
5-459	Transmit Structures .....	5-636
5-460	Receive Structures.....	5-637
5-461	Independent Mode Register .....	5-638
5-462	Rx Queue Descriptor.....	5-639
5-463	RxBD data structure.....	5-640
5-464	Tx Queue Descriptor .....	5-642
5-465	TxBD Data Structure.....	5-643
5-466	Host Command Flow .....	5-646
5-467	Data Frame Description for Host Command (External Memory) .....	5-647
5-468	Host Command Opcode Register.....	5-647
5-469	Policer Profile Host Command Action Register .....	5-649
5-470	KeyGen Scheme/Classification Plan/Port Partition Host Command Action Register .....	5-649
5-471	HCAR for Aging Support .....	5-649
5-472	KeyGen Scheme Host Command Extra Register .....	5-649
5-473	Policer Profile Host Command Extra Register .....	5-650
5-474	HCER for Dynamic Update of Custom Classifier Tables Command .....	5-650
5-475	HCER for Aging Support.....	5-651

# Figures

Figure Number	Title	Page Number
5-476	Single Buffer Frame Fragmentation.....	5-656
5-477	S/G Frame Fragmentation .....	5-657
5-478	Data Frame Descriptor For IP Fragmentation Host Command .....	5-659
5-479	IP Fragmentation Host Command Action Register .....	5-660
5-480	IP Fragmentation Host Command Extra Register.....	5-660
5-481	Example Per-Port IP Reassembly Data Structures.....	5-667
5-482	IP Reassembly Data Structures Overview .....	5-668
5-483	Example of a Reassembled IP Frame .....	5-669
5-484	IP Reassembly Parameters Table .....	5-671
5-485	IP Reassembly Common Parameters Table .....	5-676
5-486	RFD Index Pool and RFD Pool.....	5-679
5-487	Internal Buffer Pool Management Structure .....	5-680
5-488	IP reassembly TimeOut Entry .....	5-681
5-489	Data Frame Descriptor For IP Reassembly TimeOut Configuration Host Command.....	5-681
5-490	IP Manipulation Table Descriptor .....	5-686
5-491	IP Fragmentation Table Descriptor .....	5-688
5-492	IP Reassembly Table Descriptor .....	5-691
5-493	IPsec Manipulation Table Descriptor.....	5-693
5-494	Frame and Byte Count Statistics .....	5-695
5-495	Frame Length Range Statistics.....	5-696
5-496	Conditional Statistics .....	5-697
5-497	Statistics Table Descriptor (Type = 01).....	5-698
5-498	CAPWAP Fragmentation Condition Check Table Descriptor .....	5-701
5-499	CAPWAP Reassembly Table Descriptor .....	5-702
5-500	CAPWAP Fragmentation Table Descriptor .....	5-703
5-501	CAPWAP Manipulation Table Descriptor .....	5-705
5-502	CC list of AD for Frame Replication .....	5-708
5-503	CC list of AD for Frame Replicator (with “Keep” ADs).....	5-709
5-504	Frame Replicator Source Table Descriptor (Type = 01) .....	5-709
5-505	FPM FMan Controller Event Register .....	5-712
6-1	Ethernet MAC Block Diagram .....	6-1
6-2	Command and Configuration Register (COMMAND_CONFIG).....	6-10
6-3	First MAC Lower Address Register (MAC_ADDR_0) .....	6-12
6-4	First MAC Upper Address Register (MAC_ADDR_1).....	6-13
6-5	Maximum Frame Length Register (MAXFRM).....	6-13
6-6	RX_FIFO_SECTIONS Register Definition.....	6-14
6-7	TX_FIFO_SECTIONS Register Definition .....	6-14
6-8	HASHTABLE_CTRL Register Definition .....	6-15
6-9	Interrupt Event Register (IEVENT) .....	6-16
6-10	Transmit Inter-Packet Gap Length Register (TX_LENGTH) .....	6-17
6-11	Interrupt Mask Register (IMASK) .....	6-18

# Figures

Figure Number	Title	Page Number
6-12	CL01_PAUSE_QUANTA Register .....	6-19
6-13	CL23_PAUSE_QUANTA Register .....	6-20
6-14	CL45_PAUSE_QUANTA Register .....	6-20
6-15	CL67_PAUSE_QUANTA Register .....	6-20
6-16	CL01_PAUSE_THRESH Register .....	6-21
6-17	CL23_PAUSE_THRESH Register .....	6-21
6-18	CL45_PAUSE_THRESH Register .....	6-21
6-19	CL67_PAUSE_THRESH Register .....	6-22
6-20	RX_PAUSE_STATUS Register .....	6-22
6-21	2nd MAC Lower Address Register (MAC_ADDR_2).....	6-23
6-22	2nd MAC Upper Address Register (MAC_ADDR_3).....	6-23
6-23	3rd MAC Lower Address Register (MAC_ADDR_4).....	6-23
6-24	3rd MAC Upper Address Register (MAC_ADDR_5).....	6-24
6-25	4th MAC Lower Address Register (MAC_ADDR_6).....	6-24
6-26	4th MAC Upper Address Register (MAC_ADDR_7).....	6-24
6-27	5th MAC Lower Address Register (MAC_ADDR_8).....	6-25
6-28	5th MAC Upper Address Register (MAC_ADDR_9).....	6-25
6-29	6th MAC Lower Address Register (MAC_ADDR_10).....	6-25
6-30	6th MAC Upper Address Register (MAC_ADDR_11).....	6-26
6-31	7th MAC Lower Address Register (MAC_ADDR_12).....	6-26
6-32	7th MAC Upper Address Register (MAC_ADDR_13).....	6-26
6-33	8th MAC Lower Address Register (MAC_ADDR_14).....	6-27
6-34	8th MAC Upper Address Register (MAC_ADDR_15).....	6-27
6-35	EEE Low Power Wakeup Timer Register (LPWAKE_TIMER) .....	6-27
6-36	Transmit EEE Low Power Timer Register (SLEEP_TIMER).....	6-28
6-37	STATN_CONFIG Register .....	6-28
6-38	Receive Ethernet Octets Counter (REOCT $n$ ) .....	6-29
6-39	Receive Octets Counter Register (ROCT $n$ ) .....	6-29
6-40	Receive Alignment Error Counter Register (RALN $n$ ) .....	6-29
6-41	Receive Valid Pause Frame Counter Register (RXPF $n$ ) .....	6-30
6-42	Receive Frame Counter Register (RFRM $n$ ).....	6-30
6-43	Receive Frame Check Sequence Error Counter Register (RFCS $n$ ) .....	6-30
6-44	Receive VLAN Frame Counter Register (RVLAN $n$ ) .....	6-31
6-45	Receive Frame Error Counter Register (RERR $n$ ) .....	6-31
6-46	Receive Unicast Frame Counter Register (RUCAn) .....	6-31
6-47	Receive Multicast Frame Counter Register (RMCA $n$ ) .....	6-32
6-48	Receive Broadcast Frame Counter Register (RBCA $n$ ) .....	6-32
6-49	Receive Dropped Packets Counter Register (RDRP $n$ ) .....	6-32
6-50	Receive Packets Counter Register (RPKT $n$ ) .....	6-33
6-51	Receive Undersized Packet Counter Register (RUND $n$ ) .....	6-33
6-52	Receive 64-Octet Packet Counter Register (R64 $n$ ).....	6-33

# Figures

<b>Figure Number</b>	<b>Title</b>	<b>Page Number</b>
6-53	Receive 65- to 127-Octet Packet Counter Register (R127 <i>n</i> ) .....	6-34
6-54	Receive 128- to 255-Octet Packet Counter Register (R255 <i>n</i> ) .....	6-34
6-55	Receive 256- to 511-Octet Packet Counter Register (R511 <i>n</i> ) .....	6-34
6-56	Receive 512- to 1023-Octet Packet Counter Register (R1023 <i>n</i> ) .....	6-35
6-57	Receive 1024- to 1518-Octet Packet Counter Register (R1518 <i>n</i> ) .....	6-35
6-58	Receive 1519- to Max-Octet Packet Counter Register (R1519X <i>n</i> ) .....	6-35
6-59	Receive Oversized Packet Counter Register (ROVR <i>n</i> ) .....	6-36
6-60	Receive Jabber Packet Counter Register (RJBR <i>n</i> ) .....	6-36
6-61	Receive Fragment Packet Counter Register (RFRG <i>n</i> ) .....	6-36
6-62	Receive Control Packet Counter Register (RCNP <i>n</i> ) .....	6-37
6-63	Receive Dropped Not Truncated Packets Counter Register (RDRNTP <i>n</i> ) .....	6-37
6-64	Transmit Ethernet Octets Counter (TEOCT <i>n</i> ) .....	6-37
6-65	Transmit Octets Counter Register (TOCT <i>n</i> ) .....	6-38
6-66	Transmit Valid Pause Frame Counter Register (TXPF <i>n</i> ) .....	6-38
6-67	Transmit Frame Counter Register (TFRM <i>n</i> ) .....	6-38
6-68	Transmit Frame Check Sequence Error Counter Register (TFCS <i>n</i> ) .....	6-39
6-69	Transmit VLAN Frame Counter Register (TVLAN <i>n</i> ) .....	6-39
6-70	Transmit Frame Error Counter Register (TERR <i>n</i> ) .....	6-39
6-71	Transmit Unicast Frame Counter Register (TUCA <i>n</i> ) .....	6-40
6-72	Transmit Multicast Frame Counter Register (TMCA <i>n</i> ) .....	6-40
6-73	Transmit Broadcast Frame Counter Register (TBCA <i>n</i> ) .....	6-41
6-74	Transmit Packets Counter Register (TPKT <i>n</i> ) .....	6-41
6-75	Transmit Undersized Packet Counter Register (TUND <i>n</i> ) .....	6-41
6-76	Transmit 64-Octet Packet Counter Register (T64 <i>n</i> ) .....	6-42
6-77	Transmit 65- to 127-Octet Packet Counter Register (T127 <i>n</i> ) .....	6-42
6-78	Transmit 128- to 255-Octet Packet Counter Register (T255 <i>n</i> ) .....	6-42
6-79	Transmit 256- to 511-Octet Packet Counter Register (T511 <i>n</i> ) .....	6-43
6-80	Transmit 512- to 1023-Octet Packet Counter Register (T1023 <i>n</i> ) .....	6-43
6-81	Transmit 1024- to 1518-Octet Packet Counter Register (T1518 <i>n</i> ) .....	6-43
6-82	Transmit 1519- to TX_MTU-Octet Packet Counter Register (T1519X <i>n</i> ) .....	6-44
6-83	Transmit Control Packet Counter Register (TCNP <i>n</i> ) .....	6-44
6-84	IF_MODE Register .....	6-45
6-85	IF_STATUS Register .....	6-45
6-86	MDIO_CFG Register Definition .....	6-46
6-87	MDIO_CTL Register Definition .....	6-48
6-88	MDIO_DATA Register Definition .....	6-49
6-89	MDIO_ADDR Register Definition .....	6-49
7-1	TMR_ID Register Definition .....	7-4
7-2	TMR_ID2 Register Definition .....	7-5
7-3	TMR_CTRL Register Definition .....	7-5
7-4	TMR_TEVENT Register Definition .....	7-8

# Figures

Figure Number	Title	Page Number
7-5	TMR_TEMASK Register Definition.....	7-9
7-6	TMR_STAT Register Definition .....	7-10
7-7	TMR_CNT_H Register Definition .....	7-10
7-8	TMR_ADD Register Definition.....	7-11
7-9	TMR_ACC Register Definition .....	7-12
7-10	TMR_PRSC Register Definition .....	7-13
7-11	TMROFF_H/L Register Definition .....	7-13
7-12	TMR_ALARM1-2_H/L Register Definition .....	7-14
7-13	TMR_FIPER $n$ Register Definition .....	7-15
7-14	TMR_ETTS1-2_H/L Register Definition.....	7-16
7-15	1588 Timer Block Diagram .....	7-21
7-16	PTP Packet Format.....	7-23
7-17	SFD Delay From Pin to Detection .....	7-24

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
i	Terms, Acronyms, and Abbreviations.....	lxxxiii
ii	Notational Conventions.....	lxxxvi
iii	Related Documentation Types .....	lxxxvii
1-1	DPAA Offload Functions .....	1-2
1-2	DPAA Terms and Definitions .....	1-3
1-3	Parser Header Types.....	1-6
1-4	Post-Parsing Treatment Options .....	1-7
1-5	SEC Crypto Hardware Accelerators (CHAs).....	1-8
1-6	FD Field Description.....	1-9
1-7	Scatter/Gather Table Entry Field Descriptions .....	1-11
1-8	Simple Frame Types.....	1-14
1-9	Frame Format Codes.....	1-15
1-10	Frame Format Support Matrix .....	1-15
2-1	DPAA-Block Base Address Map .....	2-2
3-1	QMan Software Portals (QCSP) Memory Map .....	3-3
3-2	QMan Configuration and Control Register Memory Map.....	3-8
3-3	QCSP <i>i</i> _EQCR0-7 Field Descriptions .....	3-15
3-4	QCSP <i>i</i> _DQRR0-15 Field Descriptions.....	3-17
3-5	QCSP <i>i</i> _MR0-7 Field Descriptions.....	3-19
3-6	QCSP <i>i</i> _CR Field Descriptions.....	3-20
3-7	QCSP <i>i</i> _RR0-1 Field Descriptions .....	3-21
3-8	QCSP <i>i</i> _EQCR_PI_CENA Field Descriptions .....	3-22
3-9	QCSP <i>i</i> _EQCR_PI_CINH Field Descriptions .....	3-23
3-10	QCSP <i>i</i> _EQCR_CI_CENA Field Descriptions.....	3-24
3-11	QCSP <i>i</i> _EQCR_CI_CINH Field Descriptions.....	3-25
3-12	QCSP <i>i</i> _DQRR_PI_CENA Field Descriptions.....	3-26
3-13	QCSP <i>i</i> _DQRR_PI_CINH Field Descriptions.....	3-27
3-14	QCSP <i>i</i> _DQRR_CI_CENA Field Descriptions .....	3-29
3-15	QCSP <i>i</i> _DQRR_CI_CINH Field Descriptions .....	3-29
3-16	QCSP <i>i</i> _DQRR_DCAP Field Descriptions .....	3-30
3-17	QCSP <i>i</i> _DQRR_SDQCR Field Descriptions.....	3-32
3-18	QCSP <i>i</i> _DQRR_VDQCR Field Descriptions .....	3-35
3-19	QCSP <i>i</i> _DQRR_PDQCR Field Descriptions.....	3-36
3-20	QCSP <i>i</i> _MR_PI_CENA Field Descriptions .....	3-37
3-21	QCSP <i>i</i> _MR_PI_CINH Field Descriptions.....	3-38
3-22	QCSP <i>i</i> _MR_CI_CENA Field Descriptions .....	3-39
3-23	QCSP <i>i</i> _MR_CI_CINH Field Descriptions .....	3-39
3-24	QCSP <i>i</i> _RORI_CENA Field Descriptions.....	3-40
3-25	QCSP <i>i</i> _CFG Field Descriptions .....	3-41
3-26	QCSP <i>i</i> _EQCR_ITR Field Descriptions .....	3-44
3-27	QCSP <i>i</i> _DQRR_ITR Field Descriptions.....	3-45

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
3-28	QCSP <i>i</i> _MR_ITR Field Descriptions .....	3-45
3-29	QCSP <i>i</i> _ISR Field Descriptions.....	3-46
3-30	QCSP <i>i</i> _IER Field Descriptions.....	3-47
3-31	QCSP <i>i</i> _ISDR Field Descriptions .....	3-48
3-32	QCSP <i>i</i> _IIR Field Descriptions .....	3-48
3-33	QCSP <i>i</i> _ITPR Field Descriptions .....	3-49
3-34	QCSP <i>i</i> _ICID_CFG Field Descriptions .....	3-50
3-35	QCSP <i>i</i> _IO_CFG Field Descriptions .....	3-51
3-36	QCSP <i>i</i> _DD_CFG Field Descriptions.....	3-52
3-37	QMAN_DD_CFG Field Descriptions .....	3-53
3-38	QCSP_DD_IHRSR_i Field Descriptions.....	3-54
3-39	DCP_DD_IHRSR Field Descriptions.....	3-55
3-40	QCSP_DD_IHRFR_i Field Descriptions.....	3-55
3-41	DCP_DD_IHRFR Field Descriptions.....	3-56
3-42	QCSP_DD_HASR_i Field Descriptions.....	3-57
3-43	DCP_DD_HASR Field Descriptions .....	3-57
3-44	DCPi_CFG Field Descriptions.....	3-58
3-45	DCPi_DD_CFG Field Descriptions.....	3-59
3-46	DCPi_DLM_CFG Field Descriptions.....	3-60
3-47	DCPi_DLM_AVG Field Descriptions .....	3-62
3-48	PFDR_FPC Field Descriptions .....	3-62
3-49	PFDR_FP_HEAD Field Descriptions.....	3-63
3-50	PFDR_FP_TAIL Field Descriptions .....	3-63
3-51	Register PFDR_FP_LWIT Field Descriptions.....	3-63
3-52	Register PFDR_CFG Field Descriptions .....	3-64
3-53	SFDR_CFG Field Descriptions .....	3-64
3-54	SFDR_IN_USE Field Descriptions .....	3-65
3-55	WQ_CS_CFG <i>i</i> Field Descriptions .....	3-66
3-56	WQ_DEF_ENQ_WQID Field Descriptions .....	3-67
3-57	WQ_SC_DD_CFG_i Field Descriptions .....	3-68
3-58	WQ_PC_DD_CFG_i Field Descriptions .....	3-69
3-59	WQ_DCx_DD_CFG_i Field Descriptions.....	3-70
3-60	CM_CFG Field Descriptions .....	3-72
3-61	CEETM_CFG_IDX Field Descriptions.....	3-72
3-62	CEETM_CFG_PRES Field Descriptions .....	3-73
3-63	CEETM_XSFDR_IN_USE Field Descriptions .....	3-73
3-64	QMAN_ECSR Field Descriptions .....	3-75
3-65	QMAN_ECIR Field Descriptions .....	3-75
3-66	QMAN_ECIR2 Field Descriptions .....	3-76
3-67	QMAN_EADR Field Descriptions .....	3-77
3-68	QMAN_EDATA <i>i</i> Field Descriptions .....	3-78

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
3-69	QMAM_SBET Field Descriptions .....	3-79
3-70	QMAM_SBE <i>Ci</i> Field Descriptions.....	3-80
3-71	QMAM_MCR Field Descriptions .....	3-81
3-72	QMAM_MCP0 Field Descriptions.....	3-82
3-73	QMAM_MCP1 Field Descriptions.....	3-83
3-74	QMAM_MR <i>i</i> Field Descriptions.....	3-84
3-75	QMAM_MISC_CFG Field Descriptions .....	3-85
3-76	QMAM_IDLE_STAT Field Descriptions.....	3-85
3-77	QMAM_IP_REV_1 Field Descriptions .....	3-86
3-78	QMAM_IP_REV_2 Field Descriptions .....	3-86
3-79	FQD_BARE Field Descriptions.....	3-88
3-80	PFDR_BARE Field Descriptions.....	3-88
3-81	FQD_BAR Field Descriptions .....	3-89
3-82	PFDR_BAR Field Descriptions .....	3-89
3-83	FQD_AR Field Descriptions.....	3-90
3-84	PFDR_AR Field Descriptions.....	3-91
3-85	QCSP_BARE Field Descriptions.....	3-92
3-86	QCSP_BAR Field Descriptions .....	3-92
3-87	CI_SCHED_CFG Field Descriptions .....	3-93
3-88	QMAM_SRCIDR Field Descriptions.....	3-93
3-89	(QMAM_ICIDR) Field Descriptions .....	3-94
3-90	CI_RLM_CFG Field Descriptions.....	3-95
3-91	CI_RLM_AVG Field Descriptions .....	3-96
3-92	QMAM_ERR_ISR Field Descriptions .....	3-97
3-93	QMAM_ERR_IER Field Descriptions .....	3-100
3-94	QMAM_ERR_ISDR Field Descriptions .....	3-100
3-95	QMAM_ERR_IIR Field Descriptions .....	3-101
3-96	QMAM_ERR_HER Field Descriptions .....	3-101
3-97	Frame Descriptor (FD) .....	3-104
3-98	Frame Queue Descriptor (FQD) Format Description .....	3-112
3-99	Work Queue (WQ) Channel Assignments in the QMan .....	3-123
3-100	WQ Class Scheduler .....	3-129
3-101	Dequeue Dispatcher Service Priorities .....	3-143
3-102	FQD Context_A field for dequeued Frame Data, Annotation, and Context Stashing control..... 3-164	
3-103	Enqueue Command Format .....	3-169
3-104	Frame Dequeue Response Format .....	3-172
3-105	ERN Message Response Format.....	3-174
3-106	FQ State Change Notification Message Response Format .....	3-177
3-107	Initialize FQ Command Format .....	3-180
3-108	Initialize FQ Response Format .....	3-182

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
3-109	Query FQ Programmable Fields Command Format .....	3-183
3-110	Query FQ Programmable Fields Response Format .....	3-184
3-111	Query FQ Non-Programmable Fields Command Format.....	3-185
3-112	Query FQ Non-Programmable Fields Response Format .....	3-186
3-113	Alter FQ State Command Format .....	3-189
3-114	Alter FQ State Response Format.....	3-191
3-115	Query WQ Length Command Format .....	3-192
3-116	Query WQ Length Response Format.....	3-193
3-117	Initialize/Modify CGR Command Format.....	3-194
3-118	Initialize/Modify CGR Response Format .....	3-196
3-119	Query CGR Command Format .....	3-197
3-120	Query CGR Response Format.....	3-198
3-121	Query Congestion State Command Format .....	3-199
3-122	Query Congestion State Response Format.....	3-200
3-123	LFQMT Configure Command Format.....	3-201
3-124	LFQMT Configure Response Format .....	3-202
3-125	LFQMT Query Command Format .....	3-203
3-126	LFQMT Query Response Format .....	3-204
3-127	CEETM CQ Configure Command Format .....	3-205
3-128	CEETM CQ Configure Response Format .....	3-206
3-129	CEETM CQ Query Command Format .....	3-206
3-130	CEETM CQ Query Response Format.....	3-207
3-131	CEETM DCT Configure Command Format.....	3-208
3-132	CEETM DCT Configure Response Format .....	3-209
3-133	CEETM DCT Query Command Format.....	3-210
3-134	CEETM DCT Query Response Format .....	3-211
3-135	CEETM Class Scheduler Configure Command Format .....	3-211
3-136	CEETM Class Scheduler Configure Response Format .....	3-213
3-137	CEETM Class Scheduler Query Command Format .....	3-213
3-138	CEETM Class Scheduler Query Response Format.....	3-214
3-139	CEETM Channel Mapping Configure Command Format.....	3-215
3-140	CEETM Channel Mapping Configure Response Format .....	3-216
3-141	CEETM Channel Mapping Query Command Format .....	3-217
3-142	CEETM Channel Mapping Query Response Format .....	3-218
3-143	CEETM Sub-Portal Mapping Configure Command Format .....	3-219
3-144	CEETM Sub-Portal Mapping Configure Response Format .....	3-219
3-145	CEETM Sub-Portal Mapping Query Command Format .....	3-220
3-146	CEETM Sub-Portal Mapping Query Response Format.....	3-221
3-147	CEETM Shaper Configure Command Format.....	3-222
3-148	CEETM Shaper Configure Response Format.....	3-224
3-149	CEETM Shaper Query Command Format.....	3-224

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
3-150	CEETM Shaper Query Response Format .....	3-225
3-151	CEETM Traffic Class Flow Control Configure Command Format.....	3-227
3-152	CEETM Traffic Class Flow Control Configure Response Format.....	3-228
3-153	CEETM Traffic Class Flow Control Query Command Format.....	3-228
3-154	CEETM Traffic Class Flow Control Query Response Format .....	3-229
3-155	CEETM CCCR CM Configure Command Format.....	3-230
3-156	CEETM CCCR CM Configure Response Format .....	3-232
3-157	CEETM CCCR CM Query Command Format.....	3-232
3-158	CEETM CCCR CM Query Response Format .....	3-233
3-159	CEETM Query Congestion State Command Format.....	3-234
3-160	CEETM Query Congestion State Response Format .....	3-235
3-161	CEETM CQ Peek/Pop/XSFDR Read Command Format.....	3-235
3-162	CEETM CQ Peek/Pop/XSFDR Read Response Format .....	3-236
3-163	CEETM Statistics Query/Write Command Format .....	3-237
3-164	CEETM Statistics Query/Write Response Format.....	3-239
3-165	Congestion Group Record (CGR).....	3-255
3-166	CEETM Class Queue Descriptor (CQD) Format Description.....	3-270
3-167	Logical Frame Queue ID to CEETM Class Queue mapping.....	3-272
3-168	WBFS weight codes to weight mapping.....	3-275
3-169	CEETM Class Congestion Group Record (CCGR) .....	3-284
3-170	Example of Max-Min Fairness .....	3-291
4-1	BMan Software Portal Memory Map.....	4-5
4-2	BMan Configuration, Control, and Status Register Memory Map .....	4-6
4-3	BCSPn_CR Field Descriptions .....	4-9
4-4	BCSPn_RRm Field Descriptions .....	4-10
4-5	BCSPn_RCRm Field Descriptions.....	4-12
4-6	BCSPn_RCR_PI_CENA, BCSPn_RCR_PI_CINH Field Descriptions .....	4-13
4-7	BCSPn_RCR_CI_CENA, BCSPn_RCR_CI_CENA Field Descriptions .....	4-14
4-8	BCSPn_RCR_ITR Field Descriptions .....	4-15
4-9	BCSPn_CFG Field Descriptions.....	4-16
4-10	BCSPn_SCNm Field Descriptions.....	4-17
4-11	BCSPn_ISR, BCSPn_IER, BCSPn_IFR, BCSPn_ISDR Field Descriptions .....	4-18
4-12	BCSPn_IIR Field Descriptions .....	4-19
4-13	Register BMAN_IP_REV_1 Field Descriptions .....	4-20
4-14	Register BMAN_IP_REV_2 Field Descriptions .....	4-20
4-15	Register FBPR_BARE Field Descriptions .....	4-22
4-16	Register FBPR_BAR Field Descriptions .....	4-22
4-17	Register FBPR_AR Field Descriptions .....	4-22
4-18	BMAN_ICIDR Field Descriptions .....	4-23
4-19	BMAN_SRCIDR Field Descriptions.....	4-23

# Tables

Table Number	Title	Page Number
4-20	BMAN_POOLn_SWDET, BMAN_POOLn_SWDXT, BMAN_POOLn_HWDET, BMAN_POOLn_HWDXT Field Descriptions .....	4-25
4-21	FBPR_FP_LWIT Field Descriptions .....	4-25
4-22	BMAN_POOLn_SDCNT, BMAN_POOLn_HDCNT Field Descriptions .....	4-26
4-23	BMAN_POOLn_CONTENT Field Descriptions .....	4-26
4-24	BMAN_POOLn_HDPTR, FBPR_HDPTR Field Descriptions .....	4-27
4-25	CMD_PMn_CFG Field Descriptions.....	4-28
4-26	CMD_PMn_CFG_CFIFO Field Descriptions .....	4-29
4-27	BMAN_FL_PM_CFG Field Descriptions .....	4-29
4-28	STATE_IDLE Field Descriptions .....	4-30
4-29	STATE_STOP Field Descriptions.....	4-30
4-30	BMAN_ERR_ISR Field Descriptions .....	4-32
4-31	BMAN_ERR_IIR Field Descriptions .....	4-33
4-32	BMAN_SBET Field Descriptions .....	4-33
4-33	BMAN_SBEC0-1 Field Descriptions.....	4-34
4-34	BMAN_ECIR Field Descriptions .....	4-34
4-35	BMAN_CECR Field Descriptions.....	4-35
4-36	BMAN_CEAR Field Descriptions.....	4-36
4-37	BMAN_AECR Field Descriptions.....	4-37
4-38	BMAN_AEAR Field Descriptions .....	4-37
4-39	DEBUG_CFIFO Field Descriptions .....	4-37
4-40	DEBUG_FSM Field Descriptions .....	4-38
4-41	Acquire Command Format Field Descriptions .....	4-41
4-42	Internal Storage Register Conditions/Results .....	4-41
4-43	Acquire Response Format Field Descriptions .....	4-43
4-44	Query Response Format.....	4-44
4-45	Determining the Expected Polarity of the Valid Bit .....	4-45
4-46	Release Command Format .....	4-48
4-47	Additional BMan Error Event Registers.....	4-51
4-48	Additional BMan Error Event Registers .....	4-52
5-1	Mapping features to devices .....	5-1
5-2	FMan Terms, Acronyms, and Abbreviations .....	5-1
5-3	FMan aggregate rate.....	5-5
5-4	Supported FMan Hardware Ports.....	5-6
5-5	Mapping Different FMans to Devices.....	5-15
5-6	Parser Header Types.....	5-16
5-7	Classification types .....	5-17
5-8	FMan Receive (Rx) Functional Flow (Example).....	5-23
5-9	FMan Transmit (Tx) Functional Flow (Example) .....	5-26
5-10	FMan Offline Port Functional Flow (Example for Offline Parsing).....	5-27
5-11	Independent Rx Flow .....	5-29

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-12	Independent Tx Flow .....	5-29
5-13	HOST Command Flow .....	5-31
5-14	Hardware PortIDs .....	5-31
5-15	FMan Hardware Ports in Devices .....	5-32
5-16	FMan base address in SoC Memory map .....	5-33
5-17	FMan Memory Map Regions .....	5-34
5-18	FMan Hardware Port Page Memory Map .....	5-35
5-19	Internal Context (IC) .....	5-37
5-20	FQD[Context A] for FMan dequeue description .....	5-39
5-21	Context A—A0 Field Descriptions for Tx Port .....	5-41
5-22	Context A—A0 Field Descriptions for Offline Port .....	5-42
5-23	Context A—A2 Field Descriptions for Tx Port .....	5-42
5-24	Context A—A2 Field Descriptions for Offline Port .....	5-43
5-25	Context B - field bit Descriptions .....	5-44
5-26	Frame Descriptor Fields .....	5-45
5-27	Rx FD Status Field Description .....	5-47
5-28	Tx FD Command/Status Field Description .....	5-51
5-29	Offline Port FD Command/Status Field Description .....	5-53
5-30	If IP acceleration is enabled: {IPR,IPRE} coding .....	5-54
5-31	Host Command FD Command/Status Field Description .....	5-57
5-32	ICAD for Rx Fields .....	5-58
5-33	ICAD for Tx or Offline Fields .....	5-59
5-34	NIA Field Descriptions .....	5-60
5-35	NIA Codes Cross References .....	5-60
5-36	FMan Debug Terms .....	5-63
5-37	Trace NIA Field Descriptions .....	5-65
5-38	Generic Debug Trace Configuration Model Field Descriptions .....	5-67
5-39	GDTCR Field Descriptions .....	5-68
5-40	Generic Debug Trap Value Register (DVR) Model Field Descriptions .....	5-69
5-41	Generic Debug Trap Mask Register Model Field Descriptions .....	5-69
5-42	Types of FMan Errors .....	5-70
5-43	Glossary of Acronyms and Abbreviations .....	5-73
5-44	Terms and Descriptions .....	5-74
5-45	BMI Input Actions Codes (NIAs) .....	5-79
5-46	BMI Memory Map .....	5-81
5-47	BMI Detailed Memory Map .....	5-82
5-48	Number of virtual storage profiles .....	5-88
5-49	Hardware port storage profile .....	5-90
5-50	Virtual Storage Profile .....	5-92
5-51	Congestion group priority mapping table .....	5-93
5-52	FMBM_INIT Field Descriptions .....	5-94

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-53	FMBM_CFG1 Field Descriptions .....	5-95
5-54	FMBM_CFG2 Field Descriptions .....	5-96
5-55	FMBM_IEVR Field Descriptions .....	5-97
5-56	FMBM_IER Field Descriptions .....	5-98
5-57	FMBM_IFR Field Descriptions .....	5-98
5-58	FMBM_DTC Field Descriptions .....	5-99
5-59	FMBM_DCV Field Descriptions .....	5-100
5-60	FMBM_DCM Field Descriptions .....	5-100
5-61	FMBM_GDE Field Descriptions .....	5-101
5-62	FMBM_PP Field Descriptions .....	5-102
5-63	FMBM_PFS Reset Values in FMan_v3 .....	5-105
5-64	FMBM_PFS Field Descriptions .....	5-106
5-65	FMBM_SPICID Field Descriptions .....	5-107
5-66	FMBM_RCFG Field Descriptions .....	5-108
5-67	FMBM_RST Reset Values .....	5-110
5-68	FMBM_RST Field Descriptions .....	5-110
5-69	FMBM_RDA Field Descriptions .....	5-111
5-70	FMBM_RFP Field Descriptions .....	5-113
5-71	FMBM_RFED Field Descriptions .....	5-114
5-72	FMBM_RICP Field Descriptions .....	5-115
5-73	FMBM_RIM Field Descriptions .....	5-117
5-74	FMBM_REBM Field Descriptions .....	5-118
5-75	FMBM_RFNE Field Descriptions .....	5-119
5-76	FMBM_RFCA Field Descriptions .....	5-119
5-77	FMBM_RFPNE Field Descriptions .....	5-120
5-78	FMBM_RPSO Field Descriptions .....	5-121
5-79	FMBM_RPP Field Descriptions .....	5-122
5-80	FMBM_RPRI Reset Values .....	5-123
5-81	FMBM_RPRI Field Descriptions .....	5-123
5-82	FMBM_RFQID Field Descriptions .....	5-124
5-83	FMBM_REFQID Field Descriptions .....	5-125
5-84	FMBM_RFSDM Field Descriptions .....	5-125
5-85	FMBM_RFSEM Field Descriptions .....	5-126
5-86	FMBM_RFENE Field Descriptions .....	5-127
5-87	FMBM_RCMNE Descriptions .....	5-127
5-88	FMBM_REBMPI<nn>_y Field Descriptions .....	5-128
5-89	FMBM_RACNT Field Descriptions .....	5-130
5-90	FMBM_RCGM Field Descriptions .....	5-131
5-91	FMBM_RMPD Field Descriptions .....	5-131
5-92	FMBM_RSTC Field Descriptions .....	5-133
5-93	FMBM_RFRC Field Descriptions .....	5-133

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-94	FMBM_RBFC Field Descriptions .....	5-134
5-95	FMBM_RLFC Field Descriptions .....	5-135
5-96	FMBM_RFFC Field Descriptions .....	5-135
5-97	FMBM_RFDC Field Descriptions.....	5-136
5-98	FMBM_RLFDEC Field Descriptions .....	5-137
5-99	FMBM_RODC Field Descriptions .....	5-137
5-100	FMBM_RBDC Field Descriptions .....	5-138
5-101	FMBM_RPEC Field Descriptions .....	5-139
5-102	FMBM_RPC Field Descriptions.....	5-139
5-103	FMBM_RPCP Field Descriptions .....	5-140
5-104	FMBM_RCCN Field Descriptions .....	5-141
5-105	FMBM_RTUC Field Descriptions.....	5-142
5-106	FMBM_RRQUC Field Descriptions .....	5-143
5-107	FMBM_RDUC Field Descriptions .....	5-143
5-108	FMBM_RFUC Field Descriptions.....	5-144
5-109	FMBM_RFUC Field Descriptions.....	5-145
5-110	FMBM_RDCFG Field Descriptions .....	5-146
5-111	FMBM_RGPR Field Descriptions.....	5-147
5-112	FMBM_TCFG Field Descriptions .....	5-148
5-113	FMBM_TST Field Descriptions .....	5-148
5-114	FMBM_TDA Field Descriptions .....	5-149
5-115	FMBM_TFP Reset Values .....	5-150
5-116	FMBM_TFP Field Descriptions .....	5-150
5-117	FMBM_TFED Field Descriptions .....	5-152
5-118	FMBM_TICP Field Descriptions.....	5-153
5-119	FMBM_TFDNE Field Descriptions .....	5-154
5-120	FMBM_TFCA Field Descriptions .....	5-154
5-121	FMBM_TCFQID Field Descriptions.....	5-155
5-122	FMBM_TEFQID Field Descriptions .....	5-156
5-123	FMBM_TFENE Field Descriptions.....	5-157
5-124	FMBM_TRLMTS Field Descriptions .....	5-157
5-125	FMBM_TRLMT Field Descriptions.....	5-159
5-126	FMBM_TCCB Field Descriptions .....	5-160
5-127	FMBM_TFNE Field Descriptions .....	5-160
5-128	FMBM_TPFCM0 Descriptions .....	5-161
5-129	FMBM_TCMNE Descriptions .....	5-162
5-130	FMBM_TSTC Field Descriptions .....	5-162
5-131	FMBM_TFRC Field Descriptions .....	5-163
5-132	FMBM_TFDC Field Descriptions .....	5-164
5-133	FMBM_TFLEDC Field Descriptions .....	5-164
5-134	FMBM_TFUFDCE Field Descriptions .....	5-165

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-135	FMBM_TBDC Field Descriptions.....	5-166
5-136	FMBM_TPC Field Descriptions.....	5-166
5-137	FMBM_TPCC Field Descriptions.....	5-167
5-138	FMBM_TCCN Field Descriptions.....	5-168
5-139	FMBM_TTUC Field Descriptions.....	5-169
5-140	FMBM_TTCQUC Field Descriptions .....	5-170
5-141	FMBM_TDUC Field Descriptions .....	5-170
5-142	FMBM_TFUC Field Descriptions .....	5-171
5-143	FMBM_TDCFG Field Descriptions .....	5-172
5-144	FMBM_TGPR Field Descriptions .....	5-173
5-145	FMBM_OCFG Field Descriptions.....	5-174
5-146	FMBM_OST Reset Values.....	5-175
5-147	FMBM_OST Field Descriptions.....	5-175
5-148	FMBM_ODA Field Descriptions.....	5-176
5-149	FMBM_OICP Field Descriptions .....	5-178
5-150	FMBM_OFDNE Field Descriptions .....	5-179
5-151	FMBM_OFNE Field Descriptions .....	5-180
5-152	FMBM_OFCA Field Descriptions.....	5-180
5-153	FMBM_OFPNE Field Descriptions.....	5-181
5-154	FMBM_OPSO Field Descriptions .....	5-182
5-155	FMBM_OPP Field Descriptions .....	5-183
5-156	FMBM_OCCB Field Descriptions .....	5-183
5-157	FMBM_OIM Field Descriptions .....	5-184
5-158	FMBM_OFP Field Descriptions .....	5-185
5-159	FMBM_OFED Field Descriptions.....	5-186
5-160	FMBM_OPRI Reset Values .....	5-187
5-161	FMBM_OPRI Field Descriptions .....	5-187
5-162	FMBM_OFQID Field Descriptions .....	5-188
5-163	FMBM_OEFQID Field Descriptions.....	5-188
5-164	FMBM_OFSDM Field Descriptions .....	5-189
5-165	FMBM_OFSEM Field Descriptions .....	5-190
5-166	FMBM_OFENE Field Descriptions .....	5-191
5-167	FMBM_ORLMTS Field Descriptions .....	5-191
5-168	FMBM_ORLMT Field Descriptions .....	5-193
5-169	FMBM_OCMNE Descriptions .....	5-194
5-170	FMBM_OCGM Field Descriptions .....	5-195
5-171	FMBM_OSTC Field Descriptions .....	5-196
5-172	FMBM_OFRC Field Descriptions.....	5-196
5-173	FMBM_OFDC Field Descriptions.....	5-197
5-174	FMBM_OFLED C Field Descriptions.....	5-198
5-175	FMBM_OFUFDC Field Descriptions .....	5-198

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-176	FMBM_OFFC Field Descriptions .....	5-199
5-177	FMBM_RFWDC Field Descriptions .....	5-200
5-178	FMBM_OLFDEC Field Descriptions.....	5-200
5-179	FMBM_OBDC Field Descriptions .....	5-201
5-180	FMBM_OODC Field Descriptions .....	5-202
5-181	FMBM_OPEC Field Descriptions .....	5-203
5-182	FMBM_OPC Field Descriptions .....	5-203
5-183	FMBM_OCP Field Descriptions .....	5-204
5-184	FMBM_OCCN Field Descriptions .....	5-205
5-185	FMBM_OTUC Field Descriptions .....	5-206
5-186	FMBM_ODUC Field Descriptions .....	5-206
5-187	FMBM_OFUC Field Descriptions.....	5-207
5-188	FMBM_ODCFG Field Descriptions.....	5-208
5-189	FMBM_OGPR Field Descriptions.....	5-209
5-190	BMI Rx/O/H Flow - Enqueue or discard decision logic.....	5-227
5-191	Offline Port Enqueue Decision Flow .....	5-231
5-192	BMI Trace Information .....	5-242
5-193	BMI Prefix Description.....	5-243
5-194	BMI Port ID Description.....	5-243
5-195	QMI Memory Map.....	5-249
5-196	Register FMQM_GC Field Descriptions .....	5-252
5-197	Register FMQM_EIE Field Descriptions .....	5-253
5-198	Register FMQM_EIEN Field Descriptions.....	5-254
5-199	FMQM{EIF Field Descriptions .....	5-254
5-200	Register FMQM_GS Field Descriptions.....	5-255
5-201	Register FMQM_ETFC Field Descriptions .....	5-256
5-202	Register FMQM_DTFC Field Descriptions .....	5-256
5-203	Register FMQM_DC0 Field Descriptions .....	5-257
5-204	Register FMQM_DTRC Field Descriptions .....	5-257
5-205	Register FMQM_EFDDD Field Descriptions .....	5-259
5-206	FMQM_DTCn Field Descriptions .....	5-260
5-207	FMQM_DTVn Field Descriptions .....	5-261
5-208	FMQM_DTMn Field Descriptions .....	5-261
5-209	FMQM_DTCn Field Descriptions .....	5-262
5-210	FMQM_PnC Field Descriptions .....	5-262
5-211	FMQM_PnS Field Descriptions.....	5-263
5-212	FMQM_PnTS Field Descriptions .....	5-264
5-213	FMQM_PnEN Field Descriptions .....	5-265
5-214	FMQM_PnEN Field Descriptions .....	5-265
5-215	FMQM_PnETFC Field Descriptions .....	5-266
5-216	FMQM_PnDN Field Descriptions .....	5-266

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-217	FMQM_PnDN Field Descriptions .....	5-267
5-218	FMQM_PnDC Reset Values in FMan_v3 .....	5-268
5-219	FMQM_PnDC Field Descriptions .....	5-268
5-220	FMQM_PnDTFC Field Descriptions.....	5-269
5-221	FMQM_PnDFNOC Field Descriptions .....	5-270
5-222	FMQM_PnDCC Field Descriptions.....	5-270
5-223	QMI Trace Size .....	5-274
5-224	FMan QMI Trace Debug Format .....	5-275
5-225	Trapped Data .....	5-276
5-226	QMI PortID Counters.....	5-277
5-227	FPM Memory Map.....	5-280
5-228	FMFP_PRC Field Description .....	5-282
5-229	FMFP_MXD Field Descriptions.....	5-283
5-230	FMFP_DIST1 Field Descriptions .....	5-283
5-231	FMFP_DIST2 Field Descriptions .....	5-284
5-232	FM_EPI Field Descriptions .....	5-285
5-233	FM_RIE Field Descriptions .....	5-288
5-234	FMFP_FCEV $n$ Field Descriptions.....	5-289
5-235	ReFMFP_CEE $n$ Field Descriptions .....	5-289
5-236	FMFP_TSC1 Descriptions .....	5-290
5-237	FMFP_TSC2 Field Descriptions.....	5-291
5-238	FMFP_TSP Field Descriptions .....	5-291
5-239	FMFP_TSFI Field Descriptions .....	5-292
5-240	FM_RCR Field Descriptions .....	5-292
5-241	FMFP_EXTC Field Descriptions.....	5-293
5-242	FMFP_DRD $n$ Field Descriptions.....	5-294
5-243	Register FM_DECCES Bits Description .....	5-294
5-244	FMFP_DRA Field Descriptions.....	5-296
5-245	Register FM_IP_REV_1 Bits Description .....	5-297
5-246	FM_IP_REV_2 Field Descriptions .....	5-297
5-247	FM_RSTC Field Descriptions .....	5-298
5-248	FMFP_CLDC Field Descriptions .....	5-299
5-249	FMan FPM Debug Trace Format .....	5-300
5-250	FM_NPI Field Descriptions .....	5-301
5-251	FMFP_EE Field Descriptions .....	5-304
5-252	FMFP_CE $Vn$ Field Descriptions .....	5-306
5-253	FMFP_PSn Field Descriptions.....	5-306
5-254	FMFP_CLFABC Field Descriptions.....	5-307
5-255	FMFP_CLFCC Field Descriptions .....	5-309
5-256	FMFP_CLFAVAL Field Descriptions .....	5-309
5-257	FMFP_CLFBVAL Field Descriptions .....	5-310

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-258	FMFP_CLFCVAL Field Descriptions .....	5-310
5-259	FMFP_CLFAMSK Field Descriptions .....	5-310
5-260	FMFP_CLFBMSK Field Descriptions .....	5-311
5-261	FMFP_CLFCMSK Field Descriptions .....	5-311
5-262	FMFP_CLFAMC Field Descriptions.....	5-312
5-263	FMFP_CLFBMC Field Descriptions.....	5-312
5-264	FMFP_CLFCMC Field Descriptions.....	5-313
5-265	Register FM_DECCEH Bits Description .....	5-313
5-266	FMFP_TS $n$ Field Descriptions .....	5-316
5-267	FMan DMA Memory Map.....	5-320
5-268	Register FMDM_SR Bits Description .....	5-321
5-269	Register FMDM_MR Bits Description.....	5-323
5-270	Register FMDM_TR Bits Description.....	5-326
5-271	Register FMDM_HY Bits Description .....	5-326
5-272	Register FMDM_SETR Bits Description .....	5-327
5-273	Register FMDM_TAH Bits Description .....	5-328
5-274	Register FMDM_TAL Bits Description.....	5-328
5-275	Register FMDM_TCID Bits Description.....	5-329
5-276	Register FMDM_WCR Bits Description .....	5-329
5-277	Register FMDM_EBCR Bits Description.....	5-330
5-279	Register FMDM_DCR Bits Description.....	5-330
5-280	Register FMDM_EMSP Bits Description .....	5-331
5-281	Register FMDM_PortID<math>i</math> (0.. 31) Bits Description.....	5-332
5-282	Parser Overall Memory Map.....	5-337
5-283	Parser Individual Register Memory Map.....	5-337
5-284	Parse Internal Memory Map .....	5-340
5-285	Port $x$ Header Examination Configuration.....	5-341
5-286	FMPR_PxCAC Field Descriptions .....	5-342
5-287	FMPR_PxCTPID Field Descriptions .....	5-343
5-288	FMPR_SXPAW $x$ Field Descriptions .....	5-343
5-289	FMPR_RPCLIM Field Descriptions.....	5-344
5-290	FMPR_RPIMAC Field Descriptions .....	5-345
5-291	FMPR_PMEEC Field Descriptions .....	5-345
5-292	FMPR_PEVER Field Descriptions .....	5-347
5-293	FMPR_PEVER Field Descriptions .....	5-347
5-294	FMPR_PERR Field Descriptions.....	5-348
5-295	FMPR_PERER Field Descriptions .....	5-349
5-296	FMPR_PPSC Fields Description .....	5-350
5-297	Register FMPR_PDS Bits Description .....	5-350
5-298	FMPR_L2RRS Field Descriptions.....	5-350
5-299	FMPR_L3RRS Field Descriptions.....	5-351

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-300	FMPR_L4RRS Field Descriptions.....	5-351
5-301	FMPR_SRRES Field Descriptions.....	5-351
5-302	FMPR_L2RRES Field Descriptions .....	5-352
5-303	FMPR_L3RRES Field Descriptions .....	5-352
5-304	FMPR_L4RRES Field Descriptions .....	5-352
5-305	FMPR_SRRES Field Descriptions .....	5-353
5-306	FMPR_SPCS Field Descriptions .....	5-353
5-307	FMPR_SPSCS Field Descriptions .....	5-354
5-308	FMPR_HXSCS Descriptions.....	5-354
5-309	FMPR_MRCS Field Descriptions .....	5-354
5-310	FMPR_MWCS Field Descriptions .....	5-355
5-311	FMPR_MRSCS Field Descriptions .....	5-355
5-312	FMPR_MWSCS Field Descriptions .....	5-355
5-313	FMPR_FCSCS Field Descriptions.....	5-356
5-314	FMPR_PDxTES Field Descriptions.....	5-356
5-315	FMPR_PDC Field Descriptions .....	5-357
5-316	FMPR_PDxTyC Field Descriptions .....	5-358
5-317	FMPR_PDxTyV Field Descriptions .....	5-359
5-318	FMPR_PDxTyM Field Descriptions .....	5-359
5-319	Possible NIA Action Codes .....	5-360
5-320	Parser Inputs/Outputs from/to FMan Memory.....	5-362
5-321	FD[STATUS] Bits .....	5-362
5-322	HPNIA Next Processing Modules .....	5-363
5-323	Hard HXS—Soft Sequence Attachment.....	5-368
5-324	Ethernet HXS Configuration.....	5-368
5-325	PPPoE+PPP HXS Configuration .....	5-368
5-328	IPv4 HXS Configuration.....	5-369
5-326	VLAN HXS Configuration .....	5-369
5-327	MPLS HXS Configuration.....	5-369
5-329	IPv6 HXS Configuration.....	5-370
5-330	TCP HXS Configuration.....	5-370
5-331	UDP HXS Configuration .....	5-370
5-332	Line-up Enable Confirmation Mask .....	5-371
5-333	Parse Array.....	5-373
5-334	Parse Array Field Descriptions .....	5-373
5-335	Ethernet—EtherType to Next HXS Mapping .....	5-380
5-336	VLAN—EtherType to Next HXS Mapping.....	5-382
5-337	802.3/SNAP—EtherType to Next HXS Mapping .....	5-384
5-338	Ethernet with PPPoE Frame Format .....	5-385
5-339	PPPoE+PPP—Protocol to Next HXS Mapping.....	5-385
5-340	MPLS Label to Next HXS Mapping .....	5-386

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-341	L2R Decoded .....	5-387
5-342	L2R Result Codes .....	5-387
5-343	IPv4—Protocol to Next HXS Mapping .....	5-390
5-344	IPv6 Extension Header Parsing.....	5-392
5-345	IPv6—Next Header to Next HXS Mapping .....	5-392
5-346	GRE—Protocol Type to Next HXS Mapping.....	5-394
5-347	Min. Encap.—Protocol to Next HXS Mapping .....	5-395
5-348	L3R Decoded .....	5-396
5-349	First L3R Info Results Decoded.....	5-396
5-350	Last L3R Info Results Decoded .....	5-397
5-351	L3R Error Result Codes .....	5-397
5-352	L4R Decoded .....	5-403
5-353	L4R Result Codes .....	5-404
5-354	ShimR Decoded .....	5-405
5-355	Parse Array Required Field Population .....	5-407
5-356	Parse Result.....	5-408
5-357	Parse Result Field Description .....	5-408
5-358	FMan Parser Debug Trace Format.....	5-412
5-359	KeyGen Memory Map .....	5-415
5-360	FMKG_GCR Field Descriptions .....	5-418
5-361	FMKG_EER Field Descriptions .....	5-419
5-362	FMKG_EEER Field Descriptions.....	5-419
5-363	FMKG_SEER Field Descriptions .....	5-420
5-364	FMKG_GSR Field Descriptions .....	5-420
5-365	FMKG_TPC Field Descriptions .....	5-421
5-366	FMKG_SERC Field Descriptions.....	5-421
5-367	FMKG_FDOR Field Descriptions .....	5-422
5-368	FMKG_GDV<i>R Field Descriptions.....	5-423
5-369	FMKG_FEER Field Descriptions .....	5-423
5-370	FMKG_AR Field Descriptions .....	5-424
5-371	AWR1_RFMODE_FMKG_SE_MODE Field Descriptions.....	5-426
5-372	AWR2_RFMODE_FMKG_SE_EKFC Field Descriptions .....	5-427
5-373	AWR3_RFMODE_FMKG_SE_EKDV Field Descriptions .....	5-431
5-374	AWR4_RFMODE_FMKG_SE_BMCH Field Descriptions.....	5-434
5-375	AWR5_RFMODE_FMKG_SE_BMCL Field Descriptions .....	5-435
5-376	AWR6_RFMODE_FMKG_SE_FQB Field Descriptions.....	5-437
5-377	AWR7_RFMODE_FMKG_SE_HC Field Descriptions.....	5-437
5-378	Register AWR8_RFMODE_FMKG_SE_PPC Bits Description .....	5-438
5-379	AWR<9+i>_RFMODE_FMKG_SE_GEC<i> (0 ..7) Field Descriptions .....	5-440
5-380	AWR17_RFMODE_FMKG_SE_SPC Field Descriptions .....	5-443
5-381	AWR<18+i>_RFMODE_FMKG_SE_DV<i> Field Descriptions .....	5-443

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-382	AWR20_RFMODE_FMKG_SE_CCBS Field Descriptions .....	5-444
5-383	AWR21_RFMODE_FMKG_SE_MV Field Descriptions .....	5-445
5-384	AWR21_RFMODE_FMKG_SE_OM Field Descriptions .....	5-445
5-385	Register AWR8_RFMODE_FMKG_SE_VSP Bits Description .....	5-446
5-386	AWR<1+i>_RFMODE_FMKG_CPE<i> Field Descriptions .....	5-448
5-387	AWR1_RFMODE_FMKG_PE_SP Field Descriptions .....	5-449
5-388	AWR2_RFMODE_FMKG_PE_CPP Field Descriptions .....	5-449
5-389	FMKG_DCR Field Descriptions .....	5-450
5-390	FMKG_D<j>TC Field Descriptions .....	5-451
5-391	FMKG_D<j>T<k>CR Field Descriptions .....	5-452
5-392	FMKG_D<j>T<k>VR Field Descriptions .....	5-454
5-393	FMKG_D<j>T<k>MR Field Descriptions .....	5-454
5-394	LECM Configuration .....	5-458
5-395	Scheme Match Vector Configuration .....	5-459
5-396	HXS Configuration .....	5-459
5-397	KeyGen NIA Action Code (AC) Fields Description .....	5-466
5-398	KeyGen Read Data .....	5-467
5-399	KeyGen Write Data .....	5-467
5-400	Frame Descriptor (FD) Status Entry .....	5-468
5-401	Action Descriptor (AD) Entry .....	5-468
5-402	FMan KeyGen Trace Debug Format .....	5-469
5-403	Types of KeyGen Events .....	5-472
5-404	Policer Memory Map .....	5-476
5-405	FMPL_GCR Field Descriptions .....	5-480
5-406	FMPL_GSR Field Descriptions .....	5-481
5-407	FMPL_EVR Field Descriptions .....	5-482
5-408	FMPL_IER Field Descriptions .....	5-483
5-409	FMPL_IFR Field Descriptions .....	5-483
5-410	MPL_EEVR Field Descriptions .....	5-484
5-411	FMPL_EIER Field Descriptions .....	5-484
5-412	FMPL_RPC Field Descriptions .....	5-485
5-413	FMPL_YPC Field Descriptions .....	5-485
5-414	FMPL_RRPC Field Descriptions .....	5-486
5-415	FMPL_RYPC Field Descriptions .....	5-486
5-416	FMPL_TPC Field Descriptions .....	5-487
5-417	MPL_FLMC Field Descriptions .....	5-487
5-418	FMPL_PAR Field Descriptions .....	5-488
5-419	FMPL_PE* Field Descriptions .....	5-490
5-420	FMPL_SERC Field Descriptions .....	5-491
5-421	FMPL_UPCR Field Descriptions .....	5-492
5-422	FMPL_DTRCR Field Descriptions .....	5-493

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-423	FMPL_FADBT $n$ Field Descriptions.....	5-494
5-424	FMPL_FADBVAL $n$ Field Descriptions .....	5-495
5-425	FMPL_FADBTMR $n$ Field Descriptions.....	5-496
5-426	FMPL_FADBTMC Field Descriptions.....	5-496
5-427	FMPL_FBDBTCR $n$ Field Descriptions.....	5-497
5-428	FMPL_FBDVALR $n$ Field Descriptions .....	5-498
5-429	FMPL_FBDVTMR $n$ Field Descriptions.....	5-499
5-430	FMPL_FBDTMC Field Descriptions.....	5-499
5-431	FMPL_FCDBTCR $n$ Field Descriptions.....	5-500
5-432	FMPL_FCDBVALR $n$ Field Descriptions .....	5-501
5-433	FMPL_FCDBTMR $n$ Field Descriptions.....	5-502
5-434	FMPL_FCDBTMC Field Descriptions.....	5-502
5-435	FMPL_DPMR Field Descriptions .....	5-503
5-436	Port ID to FMPL_PMR $n$ Mapping .....	5-504
5-437	FMPL_PMR $n$ Field Descriptions .....	5-505
5-438	MODE Field Descriptions .....	5-506
5-439	GNIA Field Descriptions .....	5-509
5-440	YNIA Field Descriptions .....	5-510
5-441	RNIA Field Descriptions .....	5-510
5-442	CIR Field Descriptions.....	5-511
5-443	CBS Field Descriptions.....	5-511
5-444	PIR_EIR Field Descriptions.....	5-512
5-445	PBS_EBS Field Descriptions.....	5-512
5-446	LTS Field Descriptions .....	5-513
5-447	CTS Field Descriptions.....	5-513
5-448	PTS_ETS Field Descriptions .....	5-514
5-449	GPC Field Descriptions .....	5-514
5-450	YPC Field Descriptions .....	5-515
5-451	RPC Field Descriptions.....	5-515
5-452	RPC Field Descriptions .....	5-516
5-453	RRPC Field Descriptions .....	5-516
5-454	FMan Policer Debug Trace Format.....	5-530
5-455	Debug Flags and PNUM Dump Field Descriptions.....	5-531
5-456	Debug Mode Dump Field Descriptions .....	5-531
5-457	NIA Example .....	5-536
5-458	FMan controller Dispatch Commands .....	5-547
5-459	FMan controller Configuration Data Download Registers .....	5-550
5-460	FMCDADDR Field Descriptions.....	5-550
5-461	FMCDATA Field Descriptions.....	5-551
5-462	FMCDREADY Field Descriptions .....	5-551
5-463	FMan Controller Configuration Data Download Flow .....	5-552

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-464	New Classification Action Descriptor (Type = 00) .....	5-559
5-465	Keep Classification Action Descriptor (Type=10).....	5-562
5-466	Table Descriptor (Type = 01) .....	5-564
5-467	\Operation Code Description .....	5-566
5-468	HM Table Descriptor (Type = 01).....	5-570
5-469	Header Manipulation Command Descriptor OPCODES .....	5-571
5-470	Header Removal Command Descriptor Field Descriptions.....	5-573
5-471	Local Header Insert Command Descriptor Field Descriptions .....	5-573
5-472	Internal Header Insert Command Descriptor Field Descriptions.....	5-574
5-473	Local Header Replace Command Descriptor Field Descriptions .....	5-575
5-474	Internal Header Replace Command Descriptor Description.....	5-575
5-475	Protocol Specific Header Removal Command Descriptor Descriptions .....	5-577
5-476	Internal Protocol Specific Header Insert Command Descriptor Descriptions .....	5-578
5-477	VLAN Priority Update Command Descriptor Descriptions .....	5-579
5-478	IP_DSCP_to_VPri_Table.....	5-580
5-479	Local IPv4 Update Command Descriptor .....	5-581
5-480	Local TCP/UDP Update Command Descriptor .....	5-583
5-481	Local IPv6 Update Command Descriptor .....	5-584
5-482	Internal IP Header replace Command Descriptor .....	5-586
5-483	UDP/TCP Checksum Calculation Command Descriptor .....	5-587
5-484	Remove header till known parser result Command Descriptor .....	5-588
5-485	Local Insert UDP or UDP-Lite Header Command Descriptor .....	5-589
5-486	Local Insert L3 Header Command Descriptor .....	5-590
5-487	Remove/Local Insert CAPWAP Header Command Descriptor.....	5-591
5-488	Replace Field in Header Command Descriptor .....	5-592
5-489	Auto Response Common Parameters Descriptor .....	5-598
5-490	TCP/UDP port table entry fields .....	5-603
5-491	Deep Sleep Auto Response Statistics table fields .....	5-604
5-492	ARP Descriptor .....	5-608
5-493	VLAN-IPv4 Binding List Structure .....	5-608
5-494	ARP Statistics Counters .....	5-609
5-495	ICMPv4 Descriptor .....	5-611
5-496	VLAN-IPv4 Binding List Structure .....	5-612
5-497	ICMPv4 Statistics Counters .....	5-612
5-498	ICMPv6 Descriptor .....	5-615
5-499	VLAN and IPv6 Addresses List Structure .....	5-616
5-500	ICMPv6 Statistics Counters .....	5-616
5-501	Neighbor Solicitation Response Cases .....	5-621
5-502	Neighbor Discovery Descriptor .....	5-622
5-503	VLAN and IPv6 Addresses List Structure .....	5-622
5-504	Neighbor Discovery Statistics Counters .....	5-623

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
5-505	SNMP Descriptor fields .....	5-628
5-506	IPv4 Addresses Table entry .....	5-629
5-507	IPv6 Addresses Table entry .....	5-629
5-508	OIDs table entry fields .....	5-630
5-509	SNMP statistics table fields .....	5-631
5-510	Old AD First 4 Bytes Temporary Value (Type = 11) .....	5-633
5-511	Independent Mode Global Parameter RAM .....	5-638
5-512	Independent Mode Register Field Descriptions .....	5-638
5-513	Rx Queue Descriptor Field Descriptions .....	5-639
5-514	RxBD Data Structure Field Descriptions .....	5-641
5-515	Tx Queue Descriptor Field Descriptions .....	5-642
5-516	TxBD Data Structure Field Descriptions .....	5-643
5-517	Host Command Opcode Register Description .....	5-647
5-518	KeyGen Scheme Host Command Extra Register Field Descriptions .....	5-650
5-519	Policer Profile Host Command Extra Register Field Descriptions .....	5-650
5-520	HCER for Dynamic Update of Custom Classifier Tables Command description .....	5-651
5-521	Host Command Data Area Description .....	5-652
5-522	IP Fragmentation Host Command Action Register .....	5-660
5-523	IP Fragmentation Host Command Extra Register .....	5-661
5-524	IP Reassembly Parameters Table .....	5-672
5-525	IP Reassembly Common Parameters Table .....	5-677
5-526	Internal Buffer Pool Management Structure .....	5-680
5-527	IP Reassembly Timeout Configuration Host Command HCAR and HCER .....	5-683
5-528	RFC 6040 IP in IP Decapsulation behaviors .....	5-685
5-529	IP Manipulation Table Descriptor .....	5-686
5-530	IP Fragmentation Table Descriptor .....	5-689
5-531	IP Reassembly Table Descriptor .....	5-691
5-532	IPsec Manipulation Table Descriptor (Type = 01) .....	5-693
5-533	Statistics Table Descriptor fields description (Type = 01) .....	5-699
5-534	CAPWAP Fragmentation Condition Check Table Descriptor .....	5-701
5-535	CAPWAP Reassembly Table Descriptor .....	5-702
5-536	CAPWAP Fragmentation Table Descriptor .....	5-704
5-537	CAPWAP Manipulation Table Descriptor (Type = 01) .....	5-705
5-538	Frame Replicator Source Table Descriptor (Type = 01) .....	5-710
5-539	FPM FMan Controller Event Register Field Descriptions .....	5-712
5-540	Parameter Page per Port .....	5-712
5-541	TNUMs Parked by the FMan Controller .....	5-713
6-1	Ethernet MAC Network Interface Signals .....	6-3
6-2	Ethernet MAC Signals—Detailed Signal Descriptions .....	6-3
6-3	Ethernet MAC Memory Map Summary .....	6-5
6-4	Module Memory Map .....	6-5

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
6-5	COMMAND_CONFIG Field Descriptions .....	6-10
6-6	MAC_ADDR_0 Field Descriptions.....	6-13
6-7	MAC_ADDR_1 Field Descriptions.....	6-13
6-8	MAXFRM Field Descriptions .....	6-13
6-9	RX_FIFO_SECTIONS Field Descriptions.....	6-14
6-10	TX_FIFO_SECTIONS Field Descriptions .....	6-15
6-11	HASHTABLE_CTRL Field Descriptions.....	6-15
6-12	IEVENT Field Descriptions.....	6-16
6-13	TX_LENGTH Field Descriptions .....	6-18
6-14	IMASK Field Descriptions .....	6-18
6-15	CL01_PAUSE_QUANTA Field Descriptions .....	6-19
6-16	CL23_PAUSE_QUANTA Field Descriptions .....	6-20
6-17	CL45_PAUSE_QUANTA Field Descriptions .....	6-20
6-18	CL67_PAUSE_QUANTA Field Descriptions .....	6-20
6-19	CL01_PAUSE_THRESH Field Descriptions .....	6-21
6-20	CL23_PAUSE_THRESH Field Descriptions .....	6-21
6-21	CL45_PAUSE_THRESH Field Descriptions .....	6-22
6-22	CL67_PAUSE_THRESH Field Descriptions .....	6-22
6-23	RX_PAUSE_STATUS Field Descriptions .....	6-22
6-24	MAC_ADDR_2 Field Descriptions.....	6-23
6-25	MAC_ADDR_3 Field Descriptions.....	6-23
6-26	MAC_ADDR_4 Field Descriptions.....	6-23
6-27	MAC_ADDR_5 Field Descriptions.....	6-24
6-28	MAC_ADDR_6 Field Descriptions.....	6-24
6-29	MAC_ADDR_7 Field Descriptions.....	6-24
6-30	MAC_ADDR_8 Field Descriptions.....	6-25
6-31	MAC_ADDR_9 Field Descriptions.....	6-25
6-32	MAC_ADDR_10 Field Descriptions.....	6-25
6-33	MAC_ADDR_11 Field Descriptions.....	6-26
6-34	MAC_ADDR_12 Field Descriptions.....	6-26
6-35	MAC_ADDR_13 Field Descriptions.....	6-26
6-36	MAC_ADDR_14 Field Descriptions.....	6-27
6-37	MAC_ADDR_15 Field Descriptions.....	6-27
6-38	LPWAKE_TIMER Field Descriptions.....	6-27
6-39	SLEEP_TIMER Field Descriptions .....	6-28
6-40	STATN_CONFIG Field Descriptions .....	6-28
6-41	REOCTn Field Description.....	6-29
6-42	ROCTn Field Description .....	6-29
6-43	RALNn Field Description .....	6-29
6-44	RXPFn Field Description .....	6-30
6-45	RFRMn Field Description.....	6-30

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
6-46	RFC <i>S</i> n Field Description .....	6-30
6-47	RVLAN <i>n</i> Field Description .....	6-31
6-48	RERR <i>n</i> Field Description .....	6-31
6-49	RUC <i>An</i> Field Description.....	6-32
6-50	RMCA <i>n</i> Field Description.....	6-32
6-51	RBC <i>An</i> Field Description.....	6-32
6-52	RDRP <i>n</i> Field Description .....	6-33
6-53	RPKT <i>n</i> Field Description.....	6-33
6-54	RUND <i>n</i> Field Description .....	6-33
6-55	R64 <i>n</i> Field Description .....	6-34
6-56	R127 <i>n</i> Field Description .....	6-34
6-57	R255 <i>n</i> Field Description .....	6-34
6-58	R511 <i>n</i> Field Description .....	6-35
6-59	R1023 <i>n</i> Field Description .....	6-35
6-60	R1518 <i>n</i> Field Description .....	6-35
6-62	ROVR <i>n</i> Field Description.....	6-36
6-63	RJBR <i>n</i> Field Description .....	6-36
6-61	R1519X <i>n</i> Field Description .....	6-36
6-64	RFRG <i>n</i> Field Description .....	6-37
6-65	RCNP <i>n</i> Field Description .....	6-37
6-66	RDRNTP <i>n</i> Field Description .....	6-37
6-67	TEOCT <i>n</i> Field Description .....	6-38
6-68	TOCT <i>n</i> Field Description .....	6-38
6-69	TXPF <i>n</i> Field Description .....	6-38
6-70	TFRM <i>n</i> Field Description .....	6-39
6-71	TFCS <i>n</i> Field Description .....	6-39
6-72	TVLAN <i>n</i> Field Description .....	6-39
6-73	TERR <i>n</i> Field Description.....	6-40
6-74	TUCA <i>n</i> Field Description.....	6-40
6-75	TMCA <i>n</i> Field Description .....	6-40
6-76	TBC <i>An</i> Field Description .....	6-41
6-77	TPKT <i>n</i> Field Description.....	6-41
6-78	TUND <i>n</i> Field Description.....	6-41
6-79	T64 <i>n</i> Field Description .....	6-42
6-80	T127 <i>n</i> Field Description .....	6-42
6-81	T255 <i>n</i> Field Description .....	6-42
6-82	T511 <i>n</i> Field Description .....	6-43
6-83	T1023 <i>n</i> Field Description .....	6-43
6-84	T1518 <i>n</i> Field Description .....	6-43
6-85	T1519X <i>n</i> Field Description .....	6-44
6-86	TCNP <i>n</i> Field Description.....	6-44

# Tables

<b>Table Number</b>	<b>Title</b>	<b>Page Number</b>
6-87	IF_MODE Field Description .....	6-45
6-88	IF_STATUS Field Description .....	6-46
6-89	MDIO_CFG Field Descriptions .....	6-47
6-90	MDIO_CTL Field Descriptions .....	6-48
6-91	MDIO_DATA Field Descriptions .....	6-49
6-92	MDIO_ADDR Field Descriptions .....	6-50
7-1	1588 Timer External Interface Signal Summary.....	7-1
7-2	1588 Timer—Detailed Signal Descriptions .....	7-2
7-3	Module Memory Map Summary.....	7-3
7-4	Module Memory Map .....	7-3
7-5	TMR_ID Field Descriptions .....	7-5
7-6	TMR_ID2 Field Descriptions .....	7-5
7-7	TMR_CTRL Register Field Descriptions .....	7-6
7-8	TMR_TEVENT Register Field Descriptions.....	7-8
7-9	TMR_TEMASK Register Field Descriptions.....	7-9
7-10	TMR_STAT Register Field Descriptions .....	7-10
7-11	TMR_CNT_H/L Register Field Descriptions.....	7-11
7-12	TMR_ADD Register Field Descriptions.....	7-12
7-13	TMR_ACC Register Field Descriptions .....	7-12
7-14	TMR_PRSC Register Field Descriptions .....	7-13
7-15	TMROFF_H/L Register Field Descriptions .....	7-14
7-16	TMR_ALARM <sub>n</sub> _H/L Register Field Descriptions .....	7-14
7-17	TMR_FIPER Register Field Descriptions .....	7-16
7-18	TMR_ETTS1-2_H Register Field Descriptions .....	7-17
7-19	Steps for Minimum Register Initialization.....	7-17
7-20	Timer Clock Domain Register List.....	7-18
7-21	Platform Clock (ipg_clk) Domain Register List.....	7-18
7-22	1588 Timer Module Interrupts .....	7-19
7-23	PTP Payload Special Fields.....	7-22
7-24	Ethernet Mode and 1588 Support .....	7-24

# Terminology, Conventions, and Resources

## Terms, Acronyms, and Abbreviations

This table shows common terms, acronyms, and abbreviations used in this document.

**Table i. Terms, Acronyms, and Abbreviations**

Term	Meaning
ADB	Allowable disconnect boundary
ATM	Asynchronous transfer mode
ATMU	Address translation and mapping unit
BD	Buffer descriptor
BIST	Built-in self test
BRI	Basic rate interface
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CCB	Core complex bus
CCSR	Configuration control and status register
CEPT	Conference des administrations Europeanes des Postes et Telecommunications (European Conference of Postal and Telecommunications Administrations)
COL	Collision
CRC	Cyclic redundancy check
CRS	Carrier sense
DPLL	Digital phase-locked loop
DUART	Dual universal asynchronous receiver/transmitter
EA	Effective address
ECC	Error checking and correction
EEST	Enhanced Ethernet serial transceiver
EHPI	Enhanced host port interface
EPROM	Erasable programmable read-only memory
FCS	Frame-check sequence
GCI	General circuit interface

**Table i. Terms, Acronyms, and Abbreviations (continued)**

Term	Meaning
GMII	Gigabit media independent interface
GPCM	General-purpose chip-select machine
GPIO	General-purpose I/O
GPR	General-purpose register
IDL	Inter-chip digital link
IPG	Interpacket gap
IrDA	Infrared Data Association
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
LAE	Local access error
LAW	Local access window
LBC	Local bus controller
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least significant byte
lsb	Least significant bit
LSU	Load/store unit
MAC	Multiply accumulate, media access control
MDI	Medium-dependent interface
MESI	Modified/exclusive/shared/invalid—cache coherency protocol
MII	Media independent interface
MMU	Memory management unit
MSB	Most significant byte
msb	Most significant bit
NMSI	Nonmultiplexed serial interface
No-op	No operation
OCeaN	On-chip network
OSI	Open systems interconnection
PCMCIA	Personal Computer Memory Card International Association
PCS	Physical coding sublayer
PIC	Programmable interrupt controller
PMA	Physical medium attachment

**Table i. Terms, Acronyms, and Abbreviations (continued)**

Term	Meaning
PMD	Physical medium dependent
POR	Power-on reset
PRI	Primary rate interface
RGMII	Reduced gigabit media independent interface
RIO	Abbreviation occasionally used to refer to the RapidIO interface
RTOS	Real-time operating system
RWITM	Read with intent to modify
RWM	Read modify write
Rx	Receive
RxBD	Receive buffer descriptor
SCP	Serial control port
SDLC	Synchronous data link control
SDMA	Serial DMA
SFD	Start frame delimiter
SI	Serial interface
SIU	System interface unit
SMC	Serial management controller
SNA	Systems network architecture
SPI	Serial peripheral interface
SPR	Special-purpose register
SRAM	Static random access memory
TAP	Test access port
TBI	Ten-bit interface
TDM	Time-division multiplexed
TLB	Translation lookaside buffer
TSA	Time-slot assigner
TSEC	Three-speed Ethernet controller
Tx	Transmit
TxBD	Transmit buffer descriptor
UART	Universal asynchronous receiver/transmitter
UPM	User-programmable machine
UTP	Unshielded twisted pair
ZBT	Zero bus turnaround

# Notational Conventions

This table shows notational conventions used in this document.

**Table ii. Notational Conventions**

Convention	Definition
<b>General</b>	
Cleared	When a bit takes the value zero, it is said to be cleared.
Set	When a bit takes the value one, it is said to be set.
<b>mnemonics</b>	Instruction mnemonics are shown in lowercase bold.
<i>italics</i>	Italics can indicate the following: · Variable command parameters, for example, <b>bcctrx</b> · Titles of publications · Internal signals, for example, <u>core int</u>
0x0	Prefix to denote hexadecimal number
0b0	Prefix to denote binary number
rA, rB	Instruction syntax used to identify a source GPR
rD	Instruction syntax used to identify a destination GPR
REGISTER[FIELD]	Abbreviations for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In some contexts, such as signal encodings, an unitalicized x indicates a don't care.
x	An italicized x indicates an alphanumeric variable
n	An italicized n indicates a numeric variable
¬	NOT logical operator
&	AND logical operator
	OR logical operator
	Concatenation, for example, TCR[WPEXT]    TCR[WP]
<b>Signals</b>	
OVERBAR	An overbar indicates that a signal is active-low
lowercase_italics	Lowercase italics is used to indicate internal signals
lowercase_plaintext	Lowercase plain text is used to indicate signals that are used for configuration.
<b>Register Access</b>	
Reserved	Ignored for the purposes of determining access type
R/W	Indicates that all non-reserved fields in a register are read/write
R	Indicates that all non-reserved fields in a register are read only
W	Indicates that all non-reserved fields in a register are write only
w1c	Indicates that all non-reserved fields in a register are cleared by writing ones to them

## Suggested Reading

This table shows related documentation types that may be helpful. NXP documentation is available from the sources listed on the second page of this manual.

**Table iii. Related Documentation Types**

Resource	Purpose
SoC reference manual	Provides details about individual implementations of an SoC
SoC hardware specifications	Provides specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations
SoC chip errata	Provides additional or corrective information for a particular device mask set. Individual errata items are published cumulatively.
Application note	Addresses specific design issues useful to programmers and engineers working with our processors
<i>EREF: A Programmer's Reference Manual for Freescale Power Architecture Processors</i>	Provides a higher-level view of the programming model as it is defined by Book E, the Freescale Book E implementation standards.

Additional literature is published as new processors become available. For a current list of documentation, visit [www.nxp.com](http://www.nxp.com).



# **Chapter 1**

## **Data Path Acceleration Architecture (DPAA) Overview**

### **1.1 Purpose of this manual**

This manual describes the core set of DPAA functionality implemented in the QorIQ LS1043A processor.

### **1.2 General DPAA Functionality**

The QorIQ data path acceleration architecture (DPAA) provides the infrastructure to support simplified sharing of networking interfaces and accelerators by multiple CPU cores. These resources are abstracted into enqueue/dequeue operations by means of a common DPAA Queue Manager (QMan) driver.

The DPAA also significantly reduces software overhead associated with high-touch, packet-forwarding operations, including the following:

- Traditional routing and bridging
- Firewall
- VPN termination for both IPsec and SSL VPNs
- Intrusion detection/prevention (IDS/IPS)
- Network anti-virus (AV)

The DPAA generally leaves software in control of protocol processing, while reducing CPU overhead through off-load functions that fall into two broad categories:

- Packet Distribution and Queue/Congestion Management
- Content Processing Acceleration

## 1.2.1 Packet Distribution and Queue/Congestion Management

This table lists some packet distribution and queue/congestion management offload functions.

**Table 1-1. DPAA Offload Functions**

Function Type	Definition
Data buffer management	Supports allocation and deallocation of buffers belonging to pools originally created by software with configurable depletion thresholds. Implemented in the Buffer Manager (BMan).
Queue management	Supports queuing and quality-of-service scheduling of frames to CPUs, network interfaces and DPAA logic modules, maintains packet ordering within flows. Implemented in a module called the Queue Manager (QMan). The QMan, besides providing flow-level queuing, is also responsible for congestion management functions such as RED/WRED, congestion notifications and tail discards.
Packet distribution	Supports in-line packet parsing and general classification to enable policing and QoS-based packet distribution to the CPUs for further processing of the packets. This function is implemented in the Frame Manager (FMan).
Policing	Supports in-line rate-limiting by means of two-rate, three-color marking (RFC 2698). Up to 256 policing profiles are supported. This function is implemented in the Frame Manager (FMan).

## 1.3 DPAA Programming Model

The DPAA assumes the existence of network flows, each of which is a series of packets that have the same packet processing and ordering requirements. Software has full flexibility in how each flow is defined, from a broad grouping of packets down to an individual session that is more synonymous with the standard networking definition of a flow.

In general, packets arriving from the network are categorized into broader flows to be distributed to the appropriate cores as defined by packet parsing and classification rules (for example, control plane traffic), while packets being sent to the hardware accelerators are categorized into more granular flows (for example, a specific IPsec tunnel). Control software would set up these flows through the processor by updating or creating data structures describing how packets in the flow should be treated.

The DPAA prescribes specific data structures to be created by the control processor at flow establishment time, and also defines how datapath processors should interact with these data structures to move packets through the off-load engines and outbound network interfaces with the least CPU overhead.

The DPAA also provides for a uniform programming interface for the hardware accelerators and network interfaces. Rather than porting multiple distinct device drivers to control and datapath software, the user ports a unified QMan driver.

The DPAA is not an all-or-nothing programming model. It is possible to use legacy implementations of packet classification and buffer management and still take advantage of the queue interface driver's simplified, software-friendly interface. Operations are encoded in architected messages which are placed on queues, and responses (normal and error) flow back to software using the same queue structures.

## 1.4 DPAA Terms and Definitions

This table shows common DPAA terms, their definitions, and graphic representations of those terms.

**Table 1-2. DPAA Terms and Definitions**

Term	Definition	Graphic Representation
Buffer	Region of contiguous memory, allocated by software, and managed by the DPAA BMan	
Buffer pool	Set of buffers with common characteristics (mainly size, alignment, access control)	
Frame	Single buffer or list of buffers that hold data, for example, packet payload, header, and other control information	
Frame queue (FQ)	FIFO of frames	
Work queue (WQ)	FIFO of FQs	
Channel	Set of eight WQs with hardware provided prioritized access	
Dedicated channel	A channel statically assigned to a particular end point, from which that end point can dequeue frames. End point may be a CPU, FMan, or SEC.	—
Pool channel	A channel statically assigned to a group of end points, from which any of the end points may dequeue frames.	—

## 1.5 Major DPAA Components

The Data Path Acceleration Architecture (DPAA) includes the following major components:

- Queue Manager (QMan)
- Buffer Manager (BMan)
- Frame Manager (FMan)
- Security Engine (SEC)

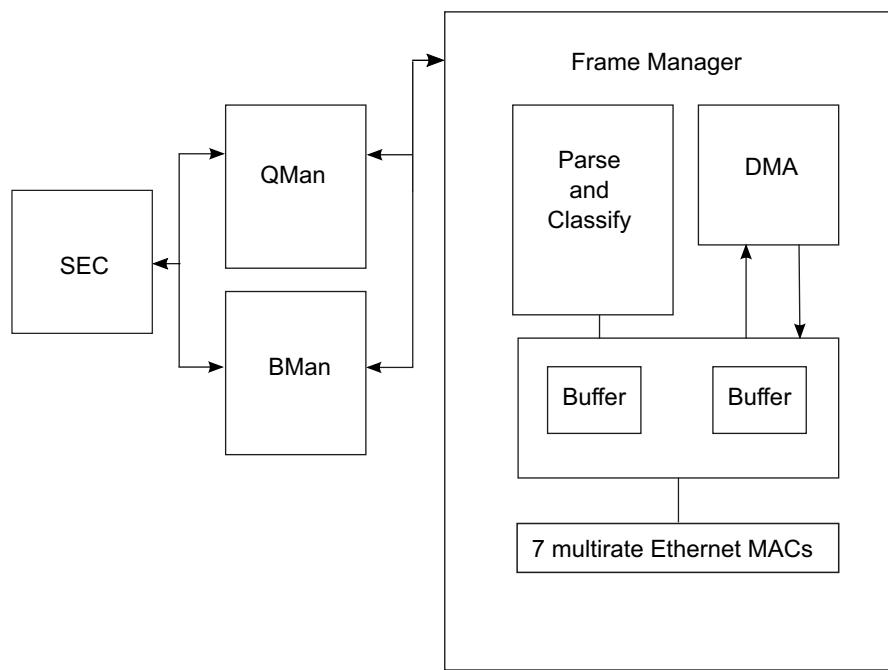


Figure 1-1. QorIQ Data Path Acceleration Architecture (DPAA)

### 1.5.1 Queue Manager (QMan)

The Queue Manager (QMan) is the component in the DPAA that allows for simplified sharing of network interfaces and hardware accelerators by multiple CPU cores. It also provides a simple and consistent message and data passing mechanism for dividing processing tasks among multiple CPU cores.

The QMan offers the following features:

- Common interface between software and all hardware
  - controls the prioritized queuing of data between multiple processor cores, network interfaces, and hardware accelerators
  - supports both dedicated and pool channels, allowing both push and pull models of multicore load spreading

- Atomic access to common queues without software locking overhead
- Mechanisms to guarantee order preservation with atomicity and order restoration following parallel processing on multiple CPUs
- Two level queuing hierarchy with one or more channels per endpoint, eight WQs per channel, and numerous frame queues per work queue
- Priority and work conserving fair scheduling between the work queues and the frame queues
- Lossless flow control for ingress network interfaces
- Congestion avoidance (RED/WRED) and congestion management with tail discard and up to 256 congestion groups. Each congestion group is composed of a user-configured number of FQs. Congestion notification is sent to an endpoint when the sum of the buffer occupancies of the group's frame queues reaches a pre-determined congestion threshold. Congestion group status has hysteresis built in.

See [Chapter 3, "Queue Manager \(QMan\)."](#)

### 1.5.2 Buffer Manager (BMan)

The buffer manager (BMan) manages pools of buffers on behalf of software for both hardware (accelerators and network interfaces) and software use.

The BMan offers the following features:

- Common interface for software and hardware
- Guarantees atomic access to shared buffer pools
- Supports 64 buffer pools. Software and hardware buffer consumers can request different size buffers and buffers in different memory partitions
- Supports depletion thresholds with congestion notifications
- On-chip per pool buffer stockpile to minimize access to memory for buffer pool management
- LIFO (last in first out) buffer allocation policy
  - Optimizes cache usage and allocation
  - A released buffer is immediately used for receiving new data

See [Chapter 4, "Buffer Manager \(BMan\)."](#)

### 1.5.3 Frame Manager (FMan)

The Frame Manager (FMan) combines the Ethernet network interfaces with packet distribution logic to provide intelligent distribution and queuing decisions for incoming traffic at line rate. This integration allows the FMan to perform configurable parsing and classification of the incoming frame with the purpose of selecting the appropriate input frame queue for expedited processing by a CPU or pool of CPUs.

See [Chapter 5, "Frame Manager \(FMan\)."](#)

### 1.5.3.1 FMan Network Interfaces

The FMan\_v3, which is the version of FMan used on the LS1043A, uses a multirate Ethernet MAC (mEMAC) supporting 100Mbps/1Gbps/2.5Gbps/10Gbps rates.

See the device reference manual for the exact number of interfaces.

The Ethernet controllers support the following:

- Programmable CRC generation and checking
- RMON statistics
- Jumbo frames of up to 9600 bytes
- Designed to comply with IEEE Std. 802.3, IEEE Std. 802.3u, IEEE Std. 802.3x, IEEE Std. 802.3z, IEEE Std. 802.3ac, IEEE Std. 802.3ab, and IEEE-1588 v2 (clock synchronization over Ethernet). In FMan\_v3: IEEE Std. 803.3az and IEEE Std. 802.1Qbb.

### 1.5.3.2 FMan Parse Function

The FMan parse function is called the Parser. The primary function of the packet parse logic is to identify the incoming frame for the purpose of determining the desired treatment to apply. This parse function can parse many standard protocols, including options and tunnels, and supports a generic configurable capability to allow proprietary or future protocols to be parsed.

There are several types of parser headers, shown in this table.

**Table 1-3. Parser Header Types**

Header Type	Definition
Self-describing	Announced by proprietary values of Ethertype, protocol identifier, next header, and other standard fields. They are self-describing in that the frame contains information that describes the presence of the proprietary header.
Non-self-describing	Does not contain any information that indicates the presence of the header. For example, a frame that always contains a proprietary header before the Ethernet header would be non-self-describing. Both self-describing and non-self-describing headers are supported by means of parsing rules in the FMan.
Proprietary	Can be defined as being self-describing or non-self-describing

The underlying notion is that different frames may require different treatment (see [Table 1-4, “Post-Parsing Treatment Options”](#)), and only through detailed parsing of the frame can proper treatment be determined. Parse results can (optionally) be passed to software.

See [Section 5.9, “Frame Manager—Parser.”](#)

### 1.5.3.3 FMan Distribution and Policing

After parsing is complete, choose one of the treatments described in this table.

**Table 1-4. Post-Parsing Treatment Options**

Treatment	Function	Benefits
Hash	<ul style="list-style-type: none"> <li>Hashes selected fields in the frame as part of a spreading mechanism</li> <li>The result is a specific frame queue identifier.</li> <li>To support added control, this FQ ID can be indexed by values found in the frame, such as TOS or p-bits, or any other desired field(s).</li> </ul>	Useful when spreading traffic while obeying QoS constraints is required
Classification look-up	<ul style="list-style-type: none"> <li>Looks up certain fields in the frame to determine subsequent action to take, including policing</li> <li>The FMan contains internal memory that holds small tables for this purpose.</li> <li>The user configures the sets of lookups to perform, and the parse results dictate which one of those sets to use.</li> <li>Lookups can be chained together such that a successful look-up can provide key information for a subsequent look-up. After all the look-ups are complete, the final classification result provides either a hash key to use for spreading, or a FQ ID directly.</li> </ul>	<ul style="list-style-type: none"> <li>Useful when hash distribution is insufficient and a more detailed examination of the frame is required</li> <li>Can determine whether policing is required and the policing context to use</li> </ul>

The subsequent choice of FQ could depend on various conditions, including the following:

- The flow being either directed to a particular CPU
- Quality of service consideration
- Control plane/data plane traffic

For example, the most obvious scenario is the distribution of flows on FQs based on their DSCP or IP precedence bits. Thus, up to eight different FQs would be created. CPUs that service those FQs could then preferentially schedule these queues or let the QMan perform preferential scheduling for the CPUs.

#### 1.5.3.3.1 FMan Policer

Key benefits of the FMan Policer are as follows:

- Because the FMan has up to 256 policing profiles, any FQ or group of FQs can be policed to either drop or mark packets if the flow exceeds a preconfigured rate.
- Policing and classification can be used in conjunction for mitigating Distributed Denial of Service Attack (DDOS).
- The policing is based on two-rate, three-color marking algorithm (RFC2698). The sustained and peak rates as well as the burst sizes are user-configurable. Hence, the policing function can rate-limit traffic to conform to the rate the flow is mapped to at flow set-up time. By prioritizing and policing traffic prior to software processing, CPU cycles can be focused on the important and urgent traffic ahead of other traffic.

See [Section 5.11, "Frame Manager—Policer."](#)

## 1.5.4 Security Engine (SEC)

The SEC is a QorIQ crypto-acceleration engine with the following functionalities:

- Off-loading cryptographic algorithms
- Offers header and trailer processing for several established security protocols
- Includes one or more descriptor controllers (DECOs), which are updated versions of the previous SEC crypto-channels

The descriptor controllers (DECOs) are responsible for header and trailer processing, and managing context and data flow into the crypto hardware accelerators (CHAs) assigned to it for the length of an operation.

The DECOs can perform header and trailer processing, as well as single pass encryption/integrity checking for the following security protocols:

- IPsec
- SSL/TLS
- SRTP
- IEEE Std. 802.1AE MACSec
- IEEE Std. 802.16e WiMax MAC layer
- 3GPP RLC encryption/decryption

In prior versions of the SEC, the individual algorithm accelerators are referred to as execution units (EUs). In the current version of the SEC, these are referred to as crypto hardware accelerators (CHAs) (to distinguish them from prior implementations). Specific CHAs available to the descriptor controllers (DECOs) are listed in this table.

**Table 1-5. SEC Crypto Hardware Accelerators (CHAs)**

Crypto Hardware Accelerator	Abbreviation
Advanced encryption standard accelerator	AESA
Alleged RC four hardware accelerator	AFHA
Cyclic redundancy check accelerator	CRCA
Data encryption standard accelerator	DESA
Kasumi hardware accelerator	KFHA
SNOW 3 G hardware accelerator	STHA
Message digest hardware accelerator	MDHA
Public key hardware accelerator	PKHA
Random number generator	RNG4
ZUC authentication accelerator	ZUCA
ZUC encryption accelerator	ZUCE

SEC is also part of the QorIQ Trust Architecture, which gives the processor the ability to provide secure boot protection, runtime code integrity protection, and session key protection. The Trust Architecture is described in SoC reference manual.

See the *QorIQ LS1043A Security (SEC) Reference Manual* for more details.

## 1.6 Data Formats Used in the DPAA

Data-to-be-processed is passed within the DPAA in distinct units known as “frames.” The actual data is held in buffers in memory and a description of the frame is what is passed within the DPAA.

### 1.6.1 Frame Descriptor (FD)

A frame descriptor (FD) is the basic element that is queued by the QMan. Figure 1-2 shows the layout of this data structure.

#### NOTE

The use and interpretation of some fields is specific to the producer or consumer of the FD; see the following chapters for details on FD implementation:

QMan: [Section 3.3.1.2, "Frame Descriptors \(FDs\)"](#)

FMan: [Section 5.4.3.2, "Frame Descriptor \(FD\)"](#)

#### 1.6.1.1 FD Format

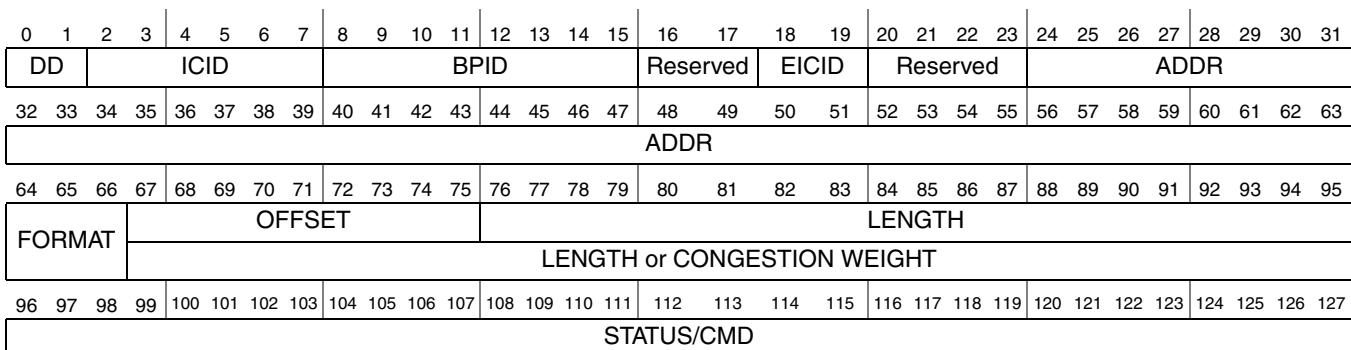


Figure 1-2. Frame Descriptor (FD)

Table 1-6. FD Field Description

FD Bits	FD Name	FD Field Description
0–1	DD	Dynamic Debug marking code point. A frame with a non-zero debug code point can generate a trace event at various enqueue and/or dequeue points within QMan. Individual QMan users have different mechanisms and criteria for setting non-zero DD values.
2–7	ICID	Frame ICID (6 least significant bits of complete ICID) The ICID is set when the FD is enqueued. It is used by a hardware module that dequeues the FD as part of the memory access control mechanism supported by the SMMU.

**Table 1-6. FD Field Description (continued)**

<b>FD Bits</b>	<b>FD Name</b>	<b>FD Field Description</b>
8–15	BPID	Buffer Pool ID The Buffer Pool ID is the number of the BMan buffer pool to which the buffer at ADDR should be released, if it is to be released to BMan
16–17	—	Reserved
18–19	EICID	Frame Extended ICID (2 most significant bits of complete ICID) The complete 8-bit ICID value is a concatenation of {EICID, ICID}. See <a href="#">Section 1.7, "Accessing Memory Using Isolation Context Identifier (ICID)"</a> , for more information on the use of this field.
20–23	—	Reserved
24–63	ADDR	Address The memory address of the start of the buffer holding the frame data or the buffer containing the scatter/gather list describing the frame.
64–66	FORMAT	A coded value that defines the format of the OFFSET and LENGTH fields, and of the frame referenced by this FD. See <a href="#">Section 1.6.6, "Frame Format Codes"</a> , for a description of these codes.
Optional	67–75	OFFSET This field is valid only when FORMAT bits 65–66 are 00, otherwise, a frame offset value of 0 is implied. Frame offset is the number of bytes from the frame address (ADDR) at which actual data begins. Note that this field is not present in all FDs
	76–95	LENGTH Total number of valid bytes of data in the frame. Note that this field is not present in all FDs
	67–95	CONGESTION WEIGHT A value used by the QMan for certain congestion management/avoidance calculations. Note that this field is not present in all FDs.
96–127	STATUS/CMD	Status or Command This field is provided to allow the sender of a frame to communicate some out-of-band information to the receiver of the frame. For instance, this field can be used to communicate a command describing what processing is to be performed to a hardware accelerator or the error/output status after a hardware accelerator has finished processing a frame.

### 1.6.1.2 Frame Descriptor (FD) Considerations

To support the diverse needs of the various accelerators in the DPAA, multiple frame formats are defined. The FD[FORMAT] designates the frame format being described by the FD.

Depending on the situation, one of the following may be stored:

- Data beginning at an offset from the actual start of the buffer, saving room at the beginning of the buffer (for adding new headers, for example)
- Large amount of data in contiguous memory

Because these cases tend to be mutually exclusive, formats are defined that trade off between the size of FD[LENGTH] and FD[OFFSET]. Depending on the FD[FORMAT], one of the following is present in the FD:

- FD[LENGTH] and FD[OFFSET]
- Just FD[LENGTH]
- Just FD[CONGESTION WEIGHT]

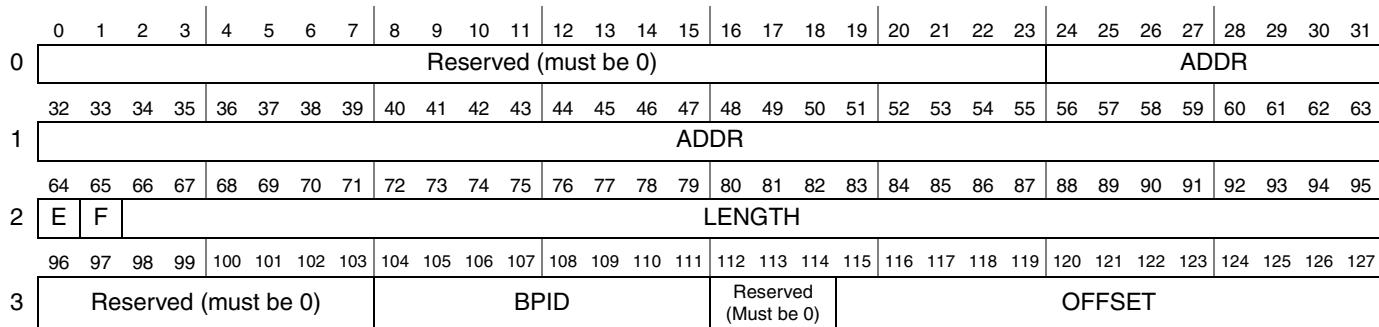
These fields occupy the same bits within the FD.

## 1.6.2 Multi-Buffer Frames

The actual data in a frame is held in one or more memory buffers. If multiple buffers are required, a scatter/gather table (also known as a block vector or IO vector table) is used to describe the buffers in a multi-buffer frame. [Figure 1-3](#) and [Table 1-7](#) define the format of each entry in a scatter/gather table.

### 1.6.2.1 Scatter/Gather Entry Format

Each multi-buffer frame consists of at least one table with at least one entry of this form.



**Figure 1-3. Scatter/Gather Table Entry Format**

**Table 1-7. Scatter/Gather Table Entry Field Descriptions**

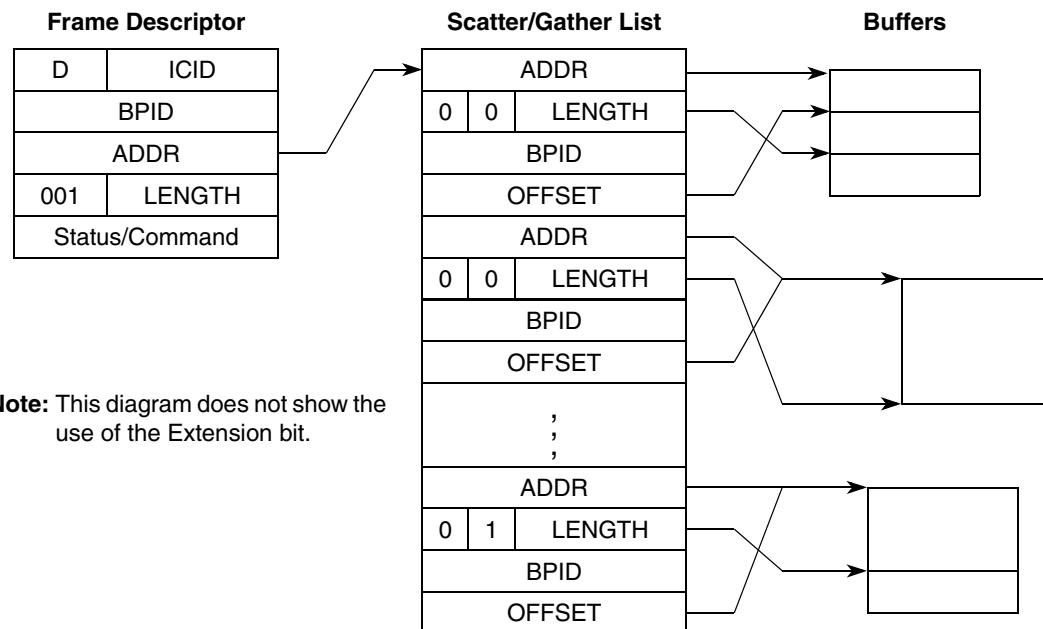
Bits	Name	Description
0–23	—	Reserved
24–63	ADDR	Address of the buffer referenced by this table entry. This buffer may contain data or it may contain another scatter/gather table.
64	E	Extension bit. If set, this table entry points to a buffer that contains another scatter/gather table.
65	F	Final bit. If set, this is the last table entry for this frame. <b>Note:</b> The E (extension) bit takes precedence over the F (final) bit. If both are set the F bit is ignored and processing continues with the entries in the scatter/gather table in the referenced buffer.
66–95	LENGTH	Depending on the context this either specifies the number of valid bytes of data in the referenced buffer or the size of the referenced buffer (in the case that empty buffers are being provided to an accelerator). <b>Note:</b> If the E bit is set, LENGTH is ignored.
96–103	—	Reserved
104–111	BPID	Buffer pool ID. The number of the BMan buffer pool to which this buffer must be released (if released to the BMan)
112–114	—	Reserved
115–127	OFFSET	An offset in bytes from the ADDR at which valid data starts. <b>Note:</b> If the E (extension) bit is set, then “valid data” is a scatter/gather table.

### 1.6.2.2 Multi-Buffer Frame Considerations

Important multi-buffer frame considerations are as follows:

- The first scatter/gather table in a multi-buffer frame is held in the buffer referenced by the FD. In the case of a short, multi-buffer frame, the table starts at OFFSET bytes from the beginning of the buffer.
- “Trees” or hierarchy are not supported, because for simple, multi-buffer frames, processing never returns to the table in the original buffer.
- If the E bit is set, it indicates that the table entry points to another buffer containing more table entries. When a scatter/gather table entry with the E bit set is encountered, processing proceeds from the first entry in the new table found in the new buffer.  
This new table may begin at some offset from the start of the buffer as defined by the OFFSET field in the entry with the E bit set.
- In simple, multi-buffer frames, the LENGTH field in a table entry with its E bit set can be ignored.
- The E bit takes precedence over the F bit (that is, if both are set in an entry, the F bit is ignored).

This figure shows a simplified representation of a long, simple frame using a scatter/gather table.



**Figure 1-4. Simplified Representation of a Long, Multi-Buffer, Simple Frame**

### 1.6.2.3 Situations Where Multi-Buffer-Frame Processing Stops

Multi-buffer frame processing stops in the following situations:

- When the data referenced by an entry with the F bit is processed regardless of the LENGTH indicated in the FD for the frame.  
In some cases, a mismatch (less data found in the frame compared to the FD length) may be an error condition for the accelerator module that is performing the processing, and it is reported as such.
- When the number of bytes specified by the overall length in the frame’s FD have been processed.  
In this case, it is not necessary to have encountered an entry with the “F” bit set and it is not considered an error when this occurs.

### 1.6.3 Single-Buffer Frames

Most frame formats consist of a single, distinct unit of data such as a single packet or protocol data unit (PDU). The FD[LENGTH] for these formats represents the total amount of valid data in the frame regardless of whether it is held in a single or in multiple buffers using scatter/gather. FD[OFFSET], on the other hand, represents the offset to the start of valid data or the scatter/gather table in the first buffer.

### 1.6.4 Compound Frames

Frames may also consist of multiple, related, distinct units such as the encrypted form of a packet along with the decrypted form of the same packet. These are known as compound frames, which consist of two or more simple frames. Each simple frame in a compound frame can in turn be stored in a single buffer or in multiple buffers. Compound frames exist so that all of the related data can be passed in a single unit within the DPAA. If an FD has a FORMAT code that identifies it as a compound frame then the FD[CONGESTION WEIGHT] is used to by the QMan for active queue management calculations such as WRED.

#### 1.6.4.1 When to Use Compound Frames

Compound frames are used for the following purposes:

- To pass multiple, distinct pieces of data either to or from a hardware accelerator. For instance, it may be required to pass frames containing both encrypted and decrypted forms of a packet.
- To supply empty buffers to a hardware accelerator into which it may place its output. In this case, it is not desirable to use the BMan. Any LENGTH and OFFSET fields that refer to a specific buffer have special meaning: OFFSET specifies the byte offset from ADDR where the accelerator should start storing data and LENGTH gives the number of bytes of data which can be stored in the remainder of the buffer.

Consider the following when using compound frames:

- When multiple input or output frames are described by a compound frame, their order is consumer-dependent.
- If a compound frame is used to pass empty buffers to a consumer for its output, those buffers are in the first frame of the compound frame.
- If a module uses a compound frame to return the input frame as well as its output frame, the output frame is the first frame of the compound frame and the input frame is the second frame.

#### 1.6.4.2 Compound Frame Considerations

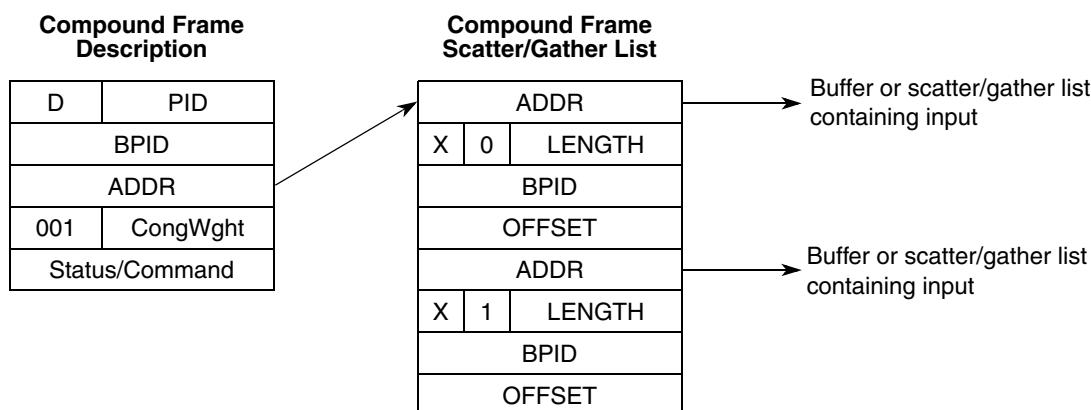
Important compound frame considerations are as follows:

- Compound frames use the same scatter/gather table format as simple frames, but with slightly different semantics for the fields in the table entries. See [Figure 1-3](#), “Scatter/Gather Table Entry Format.”
- The buffer referenced by FD[ADDR] of a compound frame must contain a scatter/gather table (the compound scatter/gather table). A compound frame contains FD[CONGESTION WEIGHT] but

does not contain FD[LENGTH] or FD[OFFSET]. The compound scatter/gather table must start at FD[ADDR].

- Each entry in the compound scatter/gather table references a simple frame. ADDR, LENGTH, BPID, and OFFSET fields in the compound scatter/gather table entry replace the corresponding fields in an FD.
- The E bit is set if the simple frame occupies multiple buffers. The E bits in multiple entries in the compound scatter/gather table may be set, because each simple frame in the compound frame may occupy multiple buffers. In this case, the buffer at ADDR contains a scatter/gather table.
- The scatter/gather table(s) that make up a simple frame that is part of a compound frame follow the same semantics as described in [Section 1.6.2.1, "Scatter/Gather Entry Format."](#)
- There is no equivalent in the compound scatter/gather table for the FORMAT field in the FD. The entries in a compound scatter/gather table cannot themselves be compound frames; the E bit is used to indicate a multi-buffer frame, and the LENGTH and OFFSET fields supported in the scatter/gather entry are a superset of the fields in an FD.
- The F bit must be set in the last entry of compound frame scatter/gather table.

This figure shows a representation of a compound frame.



**Figure 1-5. Simplified Representation of a Compound Frame**

## 1.6.5 Simple Frames

Simple frames can be either short or long (see [Table 1-8](#)). Regardless of the length of the frame, *data* is stored in a single-buffer frame, whereas *scatter/gather tables* are stored in a multi-buffer frame.

**Table 1-8. Simple Frame Types**

Simple Frame Type	Length	Requirement
Short <sup>1</sup>	No more than 1 Mbyte–1 byte-long	The data or scatter/gather table can be stored starting at an offset from the beginning of the buffer referenced by the FD.
Long	As much as 512 Mbytes–1 byte-long	The data or scatter/gather table must be stored starting at the beginning of the buffer referenced by the FD.

<sup>1</sup> Because network data (packets) are typically much less than 64 Kbytes in size, the short format is the most useful for the majority of network processing needs.

## 1.6.6 Frame Format Codes

This table defines the frame format codes and provides a brief description of the frame format.

**Table 1-9. Frame Format Codes**

Value	Frame Type	Size/Presence of LENGTH, OFFSET, and CONGESTION WEIGHT Fields (Total of 29 Bytes)		
		OFFSET	LENGTH	CONGESTION WEIGHT
'000'	Short, single-buffer, simple	9 bytes	20 bytes	—
'001'	Compound	—	—	29 bytes
'010'	Long, single-buffer, simple	—	29 bytes	—
'011'	Not defined			
'100'	Short, multi-buffer, simple	9 bytes	20 bytes	—
'101'	Not defined			
'110'	Long, multi-buffer, simple	—	29 bytes	—
'111'	Not defined			

## 1.6.7 Frame Formats Supported by Accelerators

Hardware accelerators (FMan and SEC) support a subset of the DPAA frame formats. This table summarizes the frame formats supported by various accelerators.

**Table 1-10. Frame Format Support Matrix**

Module	Short, Single-Buffer		Long, Single-Buffer		Short, Multi-Buffer		Long, Multi-Buffer		Compound Frames	
	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
FMan	Yes	Yes	No	No	SG table <sup>1</sup> limited to 16 entries; E bit not supported	SG table limited to 16 entries; E bit not supported	No	No	No	No
SEC	Yes	Yes	Yes	Yes	Yes	SG table limited to one buffer; E bit not supported	Yes	SG table limited to one buffer; E bit not supported	Yes	Yes

<sup>1</sup> Scatter/Gather entry table

## 1.6.8 Special Values and Exceptions

The ADDR, BPID, and LENGTH fields can have special meanings under certain conditions, as follows:

- FDs
- Compound frame scatter/gather tables
- Multi-buffer scatter/gather tables

Exceptions and special behaviors are as follows:

- When the LENGTH field that describes a specific buffer is zero, this indicates that the buffer contains no data. If an accelerator finds a LENGTH of zero it should not access (read or write) the memory at ADDR.
- The LENGTH field in a scatter/gather table entry of a simple multi-buffer frame with its E (extension) bit set is an exception, because the value of this field is ignored.
- FDs, compound frame, or multi-buffer frame scatter/gather entries with ADDR/BPID/LENGTH=0 encoding indicate that the FD or SG entry does not convey buffer information, that is, the scatter/gather entry is unused and is therefore skipped during input and/or output processing.

### 1.6.9 Releasing Buffers to the BMan

Accelerators only release buffers to the BMan that they have processed or, in the case of a buffer with a zero LENGTH, that they have passed over during processing. The exact configuration is dependent on the specific accelerator. Buffers with a zero ADDR, BPID, and LENGTH are not released.

#### **NOTE: Requirement**

Because processing stops when the frame's total length of data is processed or after the data referenced by a scatter/gather entry with its F (final) bit set is processed, ensure that all buffers in a frame are released.

For example, software must ensure that there are no entries in a scatter/gather table for a multi-buffer frame that are beyond the frame's total length or described in entries after the entry with the F bit set.

## 1.7 Accessing Memory Using Isolation Context Identifier (ICID)

The isolation context identifier (ICID) maps an incoming transaction from IO device to one of the context and it maps to StreamID as described in ARM documentation.

By using different ICIDs, a single hardware module can perform memory accesses on behalf of different requestors with the mapping and access controls appropriate to that requestor.

### **ICID Requirements**

Some important ICID requirements are as follows:

- The requestor must communicate the ICID to the hardware module. In the DPAA, this is done using a complete 8-bit ICID value, formed by concatenating ICID and EICID from the FD with the latter used as the 2 most significant bits (msbs): {EICID, ICID}
- Because ICID is in the FD, the module can use an ICID for each frame to access the memory.
- Some hardware modules may make different types of memory accesses as a result of dequeuing a frame. For instance, they may access per queue context/state descriptors as well as reading and writing frame data. These modules may have different ICID values for these different types of access.
- For software portals, the ICID is set by the QMan from values configured for the portal to ensure that accesses by hardware modules on behalf of software are controlled. In other words, software

running on a core cannot get a hardware module to make accesses to memory unless that access is permitted by configuration for that ICID value in an FD.

## 1.8 Packet Walk-Through Example

The following example walks a packet through a DPAA-enabled, QorIQ processor to illustrate how the DPAA offloads and accelerates packet forwarding while leaving key decisions under software control:

1. Datapath processing begins when Ethernet frames arrive at a network interface, assumed to be one of the Ethernet MACs (mEMACs) within a Frame Manager (FMan). Alternatively, packets can arrive across a peripheral bus from an external network interface, in which case, the same packet walk-through stages can be applied with the help of a CPU acting as an I/O processor.
2. After the FMan receives the Ethernet frame, it requests one or more buffers from the hardware Buffer Manager (BMan) to store the frame. BMan maintains pools of buffers, each with software-defined characteristics, and FMan is initialized to request a buffer from the most appropriate pool. If a sufficiently large buffer cannot be found for the incoming frame, FMan stores the frame across several smaller buffers and create a scatter/gather list for these buffers.
3. FMan's configurable parsing and filing capabilities can perform initial classification, sufficient to steer a packet toward a control processor, or toward one of the datapath processors for flow-specific processing (or additional classification by means of software). The steering of a packet towards a processor or a group of processors could be also based on differentiating flows by means of the Quality of Service attributes of the flow (for example, DSCP, IP precedence, or by user-defined proprietary headers).
4. Steering is accomplished by the FMan issuing an enqueue command to the QMan with the designated FQ ID, along with frame parameters such as FQ ID and Color Marking. In this example, the FMan's initial classification causes it to select a FQ ID that the Queue Manager uses to steer the FQ to the dedicated channel of logical CPU#3, a CPU operating in a datapath role. Potentially, the FQ could be selected based on the QoS requirements of the flow with a policing profile associated with the selection.
5. Continuing the example, QMan places the FQ onto WQ#5 of CPU#3's dedicated channel. The FQ was queued to CPU#3 due to user configuration decisions that all packets belonging to this specific flow should be processed by CPU#3, possibly to take advantage of special processing instructions locked in CPU#3's private cache, or due to user-defined load balancing. The FQ was placed in WQ#5 for QoS reasons, as the amount of traffic processed from each WQ relative to other work queues is also user-configurable. All FQs on a WQ have equal priority, but the amount of traffic that the CPU draws from each is user-configurable to allow fairness and appropriate bandwidth allocation.
6. Once configured, QMan can take care of the packet/frame level scheduling requirements of the CPU, that is, QMan would appropriately schedule the WQ and FQ within the WQ. QMan can be configured to perform a stashing operation in response to a CPU (or co-processor) accessing a FQ.
7. CPU#3 performs protocol processing on the packet, including modification of the packet data in the buffer. Any modifications which change the length of the packet would be noted by means of changes to the frame. The CPU determines that the packet requires IPsec ESP processing, and enqueues the frame back to the QMan using the FQ ID associated with the packet's specific ESP tunnel.

8. The QMan uses this Frame Queue ID to determine that the next consumer of the Frame Queue is the SEC, and places the Frame Queue onto Work Queue#5 of the SEC's dedicated channel. The SEC pulls Frame Queues from the Work Queues in its dedicated channel according to user-defined weights in a WFQ model.
9. The SEC dequeues and processes data from the Frame Queue, adding the tunnel IP header, ESP header, IV, trailer, and HMAC, and writing encrypted data to either the original buffers, or new buffers that the SEC requests from BMan. The SEC updates the frame description (length change due to the addition of the headers, trailers, and HMAC) and enqueues the FQ back to QMan using a configured FQ ID. In this case, the FQ ID causes the QMan to enqueue the FQ to Work Queue #5 of logical CPU#4's dedicated channel. CPU#4 dequeues the FQ, determines the outbound interface for the newly encrypted packet, updates the tunnel IP header, and enqueues the frame back to QMan on a new FQ ID. This FQ ID causes the QMan to enqueue the FQ to a channel serviced by FMan, which dequeues the FQ, transmits one or more packets from that FQ out the appropriate mEMAC, and releases the buffers back to BMan.

The processing pipeline used in this example is not required by the QorIQ DPAA. The initial classification could have caused the packet to be steered toward a CPU dedicated to fine-grained classification, or to a pool channel of CPUs, any of which could have performed the operations described. CPU#3 could have added the ESP header and trailer to the packet and sent it to the SEC for crypto-only processing. Following SEC processing, the flow was steered to CPU#4, however it could just as easily have been steered back to CPU#3, or to a pool channel. At any FQ ID transition, the relative priority of the flow could have been elevated or reduced by enqueueing it to a different work queue.

## 1.9 Specific DPAA Implementation Details

This section describes the DPAA functionality for the LS1043A.

### 1.9.1 Queue Manager (QMan) Implementation

The QMan is implemented as described in [Chapter 3, "Queue Manager \(QMan\),"](#) with the following implementation parameters:

- QMan block base address: 188\_0000h
- 512 frame queue (FQ) descriptors in cache
- 2-K SFDRs
- 256 congestion groups

### 1.9.2 Buffer Manager (BMan) Implementation

The BMan is implemented as described in [Chapter 4, "Buffer Manager \(BMan\),"](#) with the following implementation parameters:

- BMan block base address: 189\_0000h
- 64 buffer pools

### 1.9.3 Frame Manager (FMan) Implementation

The FMan is implemented as described in the [Chapter 5, "Frame Manager \(FMan\),"](#) with the following implementation parameters:

- FMan block base address: 1A0\_0000h
- Seven multirate Ethernet MACs (mEMACs) on the Frame Manager
  - See the SerDes protocol section of the *QorIQ LS1043A Reference Manual* for configuration options
  - Block base addresses are as follows:
    - FM1 mEMAC1: E\_0000h
    - FM1 mEMAC2: E\_2000h
    - FM1 mEMAC3: E\_4000h
    - FM1 mEMAC4: E\_6000h
    - FM1 mEMAC5: E\_8000h
    - FM1 mEMAC6: E\_A000h
    - FM1 mEMAC9: F\_0000h
  - Supports 1 host command and 3 offline ports:
    - Host command: 02h
    - Offline port 3: 03h
    - Offline port 4: 04h
    - Offline port 5: 05h
- FM1 Dedicated MDIO1: 1AF\_C000h
- FM1 Dedicated MDIO2: 1AF\_D000h
- One FMan Controller complex
- 384-Kbyte internal FMan memory
- 64-Kbyte FMan Controller configuration data
- Up to 32 Keygen schemes
- Up to 256 Policier profiles
- Up to 84 entries in FMan DMA command queue
- Up to 128 TNUMs
- 1 FMan debug flow

### 1.9.4 Security and Encryption Engine (SEC) Implementation

The Security and Encryption Engine (SEC) is implemented as described in the *LS1043A Security (SEC) Reference Manual* with the following implementation parameters:

- SEC block base address: 170\_0000h



# **Chapter 2**

## **Memory Map**

### **2.1 Memory Map Overview**

There are several address domains within DPAA-enabled SoCs, including the following:

- Logical, virtual, and physical (real) address spaces within the core(s)
- Internal local address space
- Internal configuration, control, and status register (CCSR) address space, which is a special-purpose subset of the internal local address space
- Internal debug, control, and status register (DCSR) address space, which is another special-purpose set of registers mapped in the internal local address space

See the memory map chapter of the QorIQ LS1043A Reference Manual to properly contextualize this DPAA memory map information.

#### **2.1.1 Considerations When Accessing Reserved Registers and Bits**

Some important things to consider when accessing reserved registers/bits are as follows:

- Reserved registers and bits in the DPAA memory space are not guaranteed to have predictable values. In general, reads of reserved bits return zeros, but software should not rely on the value of any reserved bit being a zero or remaining consistent.
- Unless otherwise specified, when writing registers in DPAA memory space, reserved bits must be written with zeros. Writing ones to a reserved bit may result in undefined operation.

#### **2.1.2 DPAA Address Map**

The full register address of any DPAA register consists of all of the following:

- CCSR window base address, specified in CCSRBAR (default address 0\_0100\_0000h)
- Block base address
- The specific register's offset within that block

## Memory Map

This table shows the location of the block base addresses for the DPAA space within CCSR of this DPAA-enabled processor.

**Table 2-1. DPAA-Block Base Address Map**

Block Base Address (Hex)	Block	Section/Page
170_0000h–17F_FFFFh	Security (SEC)	See the <i>QorIQ LS1043A Security (SEC) Reference Manual</i>
180_0000h–187_FFFFh	Reserved	—
188_0000h–188_FFFFh	Queue Manager (QMan)	<a href="#">3.2/3-3</a>
189_0000h–189_FFFFh	Buffer Manager (BMan)	<a href="#">4.2/4-4</a>
18A_0000h–19F_FFFFh	Reserved	—
1A0_0000h–1AF_FFFFh	Frame Manager (FMan) <b>Note:</b> See the FMan Memory map for the many sub-blocks.	<a href="#">5.4.2/5-33</a>

# **Chapter 3**

## **Queue Manager (QMan)**

### **3.1 QMan Introduction**

The Queue Manager (QMan) acts as a central resource in the multicore datapath infrastructure (DPI), managing the queueing of data between multiple processor cores, network interfaces, and hardware accelerators in a multicore SoC.

#### **3.1.1 QMan Overview**

The QMan provides queuing of data between the network interfaces, hardware off-load accelerators, and cores in a multicore device. It manages the data while it is queued and provides various queue-related functionality, such as congestion management (tail drop, RED/WRED) and prioritized scheduling of data from queues. It can also provide traffic shaping functionality on egress traffic to network interfaces. It also implements mechanisms that support preserving the order of data (packets) as they are processed in the SoC or the restoration of data (packets) to their original order after they are processed in the system and potentially misordered. QMan provides mechanisms to load spread processing of data across multiple processing elements, such as a pool of cores. The QMan also provides a simplified low-latency interface for software running on a core to retrieve data from queues as compared to standard buffer descriptor list/ring based DMAs.

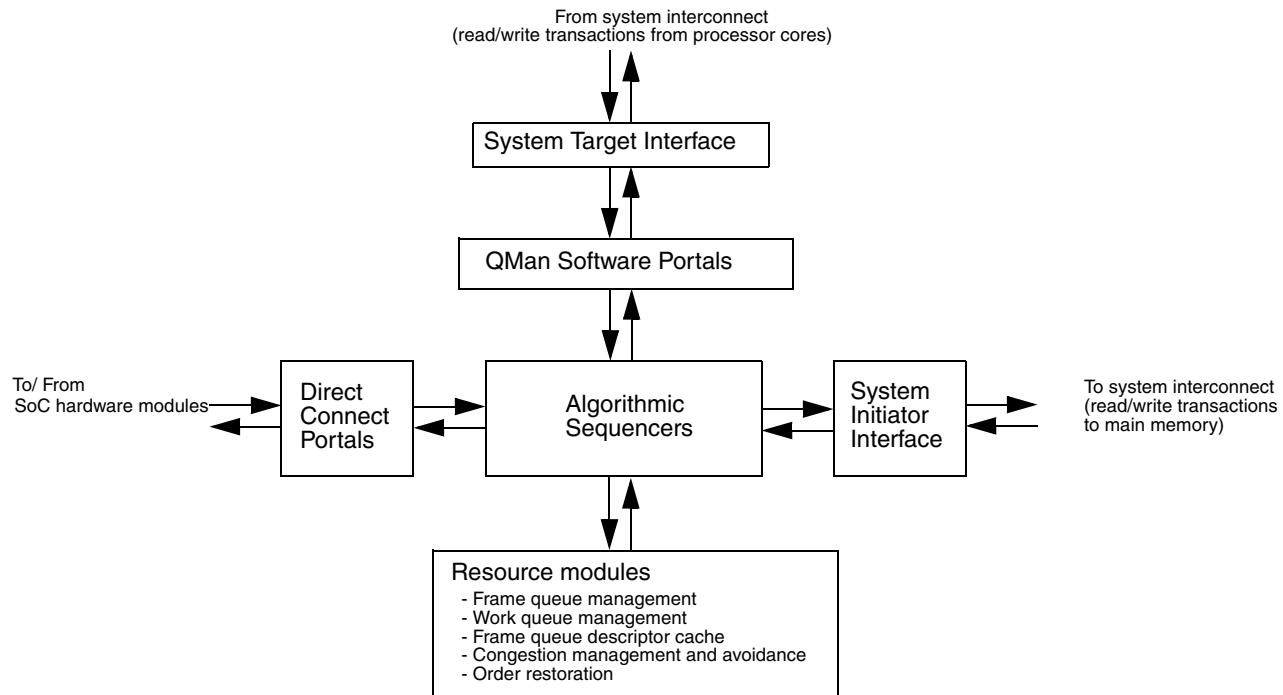


Figure 3-1. QMan Overview Block Diagram

### 3.1.2 QMan Features

- Two types of portals provide for communication with the QMan
  - Software portals used by data path software executing on a processor core. A total of 10 software portals are provided. These portals are memory mapped in the system.
  - 2 (SEC and FMan) direct connect portals (DCP) used by hardware modules with a dedicated interface connected directly to QMan.
- Two-level queueing architecture; frames are enqueued onto frame queues that are in turn enqueued onto WQs. This is referred to as FQ/WQ scheduling mode. An alternate mode referred to as customer edge egress traffic management (CEETM) scheduling mode is available on the egress side of certain direct connect portals.
- Work queues are grouped into channels with priority scheduling provided between the frame queues on a channel.
- Intra-WQ class scheduling controls the amount of data to be pulled from a frame queue (across one or multiple frames) when it is selected for presentation to a portal.
- Some WQ channels are dedicated to a particular portal, while others are used as pool channels from which multiple software portals may dequeue. A total of 15 pool channels are supported.
- Supports customer edge egress traffic management (CEETM) that provides hierarchical class based scheduling and traffic shaping:
  - Available as an alternate to FQ/WQ scheduling mode on the egress side of specific direct connect portals
  - Enhanced class based scheduling supporting 16 class queues per channel

- Token bucket based dual rate shaping representing committed rate (CR) and excess rate (ER)
- Congestion avoidance mechanism equivalent to that provided by FQ congestion groups
- Supports efficient order restoration for frames from a common source (order definition point) that may be misordered on their way to a destination (order restoration point).
- Supports 256 congestion groups within the QMan where one or more frame queues are configured for a congestion avoidance mechanism, such as RED/WRED.
- Supports tail drop congestion management on a per-frame queue basis.
- A total of 16 algorithmic sequencers are provided, allowing multiple enqueue/dequeue operations to execute simultaneously.
- A 512 deep internal memory is used to cache frame queue descriptors for improved performance.
- A 2048 deep internal memory is used to store frame descriptors at the head and tail of frame queues for improved performance.
- The AXI target interface provides access to the QMan’s software portals from processor cores.
- The AXI initiator interface is used by the QMan to read and write frame queue descriptors from/to main memory. All data accessed in main memory is private to the QMan, and treated as non-coherent when transferred between QMan and main memory.

## 3.2 QMan Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all registers. The following tables list the QMan registers and their offsets.

### 3.2.1 QMan Software Portals (QCSP) Memory Map

The software portals of the QMan are mapped in a 128 MB area of system memory space, aligned on a 128 MB boundary.

The software portals are accessible as a target using a window configured for this purpose in the system memory map. See also [Section 3.3.18, “System Target interface.”](#)

The software portals are divided into a cache-enabled area and a cache-inhibited area, with many of the registers and ring entries accessible in both areas, and some only in the cache-inhibited area, see [Section 3.3.8, “Software Portals,”](#) for more detail on how this works.

The address offsets shown in this table refer to the 27-bit address offset from the base address programmed in the LAWBAR register used for the QMan.

**Table 3-1. QMan Software Portals (QCSP) Memory Map**

Offset	Register	Access	Reset Value	Section/Page
<b>QMan Software Portal 0, Enqueue Command Ring (EQCR), Cache-Enabled Area</b>				
0x000_0000	QCSP0_EQCR0—QMan Software Portal 0, EQCR, Entry 0	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x000_0040	QCSP0_EQCR1—QMan Software Portal 0, EQCR, Entry 1	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x000_0080	QCSP0_EQCR2—QMan Software Portal 0, EQCR, Entry 2	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>

**Table 3-1. QMan Software Portals (QCSP) Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x000_00C0	QCSP0_EQCR3—QMan Software Portal 0, EQCR, Entry 3	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x000_0100	QCSP0_EQCR4—QMan Software Portal 0, EQCR, Entry 4	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x000_0140	QCSP0_EQCR5—QMan Software Portal 0, EQCR, Entry 5	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x000_0180	QCSP0_EQCR6—QMan Software Portal 0, EQCR, Entry 6	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x000_01C0	QCSP0_EQCR7—QMan Software Portal 0, EQCR, Entry 7	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
<b>QMan Software Portal 0, Dequeue Response Ring (DQRR), Cache-Enabled Area</b>				
0x000_1000	QCSP0_DQRR0—QMan Software Portal 0, DQRR, Entry 0	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1040	QCSP0_DQRR1—QMan Software Portal 0, DQRR, Entry 1	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1080	QCSP0_DQRR2—QMan Software Portal 0, DQRR, Entry 2	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_10C0	QCSP0_DQRR3—QMan Software Portal 0, DQRR, Entry 3	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1100	QCSP0_DQRR4—QMan Software Portal 0, DQRR, Entry 4	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1140	QCSP0_DQRR5—QMan Software Portal 0, DQRR, Entry 5	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1180	QCSP0_DQRR6—QMan Software Portal 0, DQRR, Entry 6	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_11C0	QCSP0_DQRR7—QMan Software Portal 0, DQRR, Entry 7	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1200	QCSP0_DQRR8—QMan Software Portal 0, DQRR, Entry 8	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1240	QCSP0_DQRR9—QMan Software Portal 0, DQRR, Entry 9	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1280	QCSP0_DQRR10—QMan Software Portal 0, DQRR, Entry 10	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_12C0	QCSP0_DQRR11—QMan Software Portal 0, DQRR, Entry 11	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1300	QCSP0_DQRR12—QMan Software Portal 0, DQRR, Entry 12	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1340	QCSP0_DQRR13—QMan Software Portal 0, DQRR, Entry 13	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_1380	QCSP0_DQRR14—QMan Software Portal 0, DQRR, Entry 14	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x000_13C0	QCSP0_DQRR15—QMan Software Portal 0, DQRR, Entry 15	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
<b>QMan Software Portal 0, Message Ring (MR), Cache-Enabled Area</b>				
0x000_2000	QCSP0_MR0—QMan Software Portal 0, Message Ring, Entry 0	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x000_2040	QCSP0_MR1—QMan Software Portal 0, Message Ring, Entry 1	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x000_2080	QCSP0_MR2—QMan Software Portal 0, Message Ring, Entry 2	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x000_20C0	QCSP0_MR3—QMan Software Portal 0, Message Ring, Entry 3	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x000_2100	QCSP0_MR4—QMan Software Portal 0, Message Ring, Entry 4	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x000_2140	QCSP0_MR5—QMan Software Portal 0, Message Ring, Entry 5	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x000_2180	QCSP0_MR6—QMan Software Portal 0, Message Ring, Entry 6	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x000_21C0	QCSP0_MR7—QMan Software Portal 0, Message Ring, Entry 7	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>

**Table 3-1. QMan Software Portals (QCSP) Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
<b>QMan Software Portal 0, Command and Response Ring Index Registers, Cache-Enabled Area</b>				
0x000_3000	QCSP0_EQCR_PI_CENA—Software Portal 0, Enqueue Command Ring, Producer Index Register, Cache-Enabled	R/W	0x0000_0008	<a href="#">3.2.3.6.1/3-22</a>
0x000_3040	QCSP0_EQCR_CI_CENA—Software Portal 0, Enqueue Command Ring, Consumer Index Register, Cache-Enabled	R	0x0000_8808	<a href="#">3.2.3.7.1/3-24</a>
0x000_3100	QCSP0_DQRR_PI_CENA—Software Portal 0, Dequeue Response Ring, Producer Index Register, Cache-Enabled	R	0x0000_0010	<a href="#">3.2.3.8.1/3-26</a>
0x000_3140	QCSP0_DQRR_CI_CENA—Software Portal 0, Dequeue Response Ring, Consumer Index Register, Cache-Enabled	R/W	0x0000_0000	<a href="#">3.2.3.9.1/3-28</a>
0x000_3300	QCSP0_MR_PI_CENA—Software Portal 0, Message Ring, Producer Index Register, Cache-Enabled	R	0x0000_0008	<a href="#">3.2.3.14.1/3-37</a>
0x000_3340	QCSP0_MR_CI_CENA—Software Portal 0, Message Ring, Consumer Index Register, Cache-Enabled	R/W	0x0000_0000	<a href="#">3.2.3.15/3-38</a>
0x000_3400	QCSP0_RORI_CENA—Software Portal 0, Read Only Ring Indices Register, Cache-Enabled	R	Special	<a href="#">3.2.3.16/3-39</a>
<b>QMan Software Portal 0, Management Registers, Cache-Enabled Area</b>				
0x000_3800	QCSP0_CR—Software Portal 0, Management Command Register	R/W	0x0000_0000	<a href="#">3.2.3.4/3-19</a>
0x000_3900	QCSP0_RR0—Portal 0, Management Response Register 0	R	0x0000_0000	<a href="#">3.2.3.5/3-20</a>
0x000_3940	QCSP0_RR1—Portal 0, Management Response Register 1	R	0x0000_0000	<a href="#">3.2.3.5/3-20</a>
0x000_4000–0x000_FFFF	Reserved, QMan Software Portal 0, cache-enabled area	—	—	—
<b>QMan Software Portals 1 and Above, Cache-Enabled Area</b>				
0x001_0000–0x001_FFFF	QMan Software Portal 1, cache-enabled area (64 KB)	—	—	—
0x002_0000 and above	QMan Software Portal cache-enabled area, 64 KB per portal In the LS1043A: 10 software portals occupy 0x000_0000 to 0x009_FFFF	—	—	—
0x002_0000–0x3FF_FFFF	64 KB blocks corresponding to non-existent software portals are reserved.	—	—	—
<b>QMan Software Portal 0, Enqueue Command Ring (EQCR), Cache-Inhibited Area</b>				
0x400_0000	QCSP0_EQCR0—QMan Software Portal 0, EQCR, Entry 0	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x400_0040	QCSP0_EQCR1—QMan Software Portal 0, EQCR, Entry 1	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x400_0080	QCSP0_EQCR2—QMan Software Portal 0, EQCR, Entry 2	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x400_00C0	QCSP0_EQCR3—QMan Software Portal 0, EQCR, Entry 3	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x400_0100	QCSP0_EQCR4—QMan Software Portal 0, EQCR, Entry 4	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x400_0140	QCSP0_EQCR5—QMan Software Portal 0, EQCR, Entry 5	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
0x400_0180	QCSP0_EQCR6—QMan Software Portal 0, EQCR, Entry 6	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>

**Table 3-1. QMan Software Portals (QCSP) Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x400_01C0	QCSP0_EQCR7—QMan Software Portal 0, EQCR, Entry 7	R/W	0x0000_0000	<a href="#">3.2.3.1/3-13</a>
<b>QMan Software Portal 0, Dequeue Response Ring (DQRR), Cache-Inhibited Area</b>				
0x400_1000	QCSP0_DQRR0—QMan Software Portal 0, DQRR, Entry 0	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1040	QCSP0_DQRR1—QMan Software Portal 0, DQRR, Entry 1	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1080	QCSP0_DQRR2—QMan Software Portal 0, DQRR, Entry 2	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_10C0	QCSP0_DQRR3—QMan Software Portal 0, DQRR, Entry 3	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1100	QCSP0_DQRR4—QMan Software Portal 0, DQRR, Entry 4	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1140	QCSP0_DQRR5—QMan Software Portal 0, DQRR, Entry 5	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1180	QCSP0_DQRR6—QMan Software Portal 0, DQRR, Entry 6	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_11C0	QCSP0_DQRR7—QMan Software Portal 0, DQRR, Entry 7	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1200	QCSP0_DQRR8—QMan Software Portal 0, DQRR, Entry 8	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1240	QCSP0_DQRR9—QMan Software Portal 0, DQRR, Entry 9	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1280	QCSP0_DQRR10—QMan Software Portal 0, DQRR, Entry 10	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_12C0	QCSP0_DQRR11—QMan Software Portal 0, DQRR, Entry 11	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1300	QCSP0_DQRR12—QMan Software Portal 0, DQRR, Entry 12	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1340	QCSP0_DQRR13—QMan Software Portal 0, DQRR, Entry 13	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_1380	QCSP0_DQRR14—QMan Software Portal 0, DQRR, Entry 14	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
0x400_13C0	QCSP0_DQRR15—QMan Software Portal 0, DQRR, Entry 15	R	0x0000_0000	<a href="#">3.2.3.2/3-15</a>
<b>QMan Software Portal 0, Message Ring (MR), Cache-Inhibited Area</b>				
0x400_2000	QCSP0_MR0—QMan Software Portal 0, Message Ring, Entry 0	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x400_2040	QCSP0_MR1—QMan Software Portal 0, Message Ring, Entry 1	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x400_2080	QCSP0_MR2—QMan Software Portal 0, Message Ring, Entry 2	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x400_20C0	QCSP0_MR3—QMan Software Portal 0, Message Ring, Entry 3	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x400_2100	QCSP0_MR4—QMan Software Portal 0, Message Ring, Entry 4	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x400_2140	QCSP0_MR5—QMan Software Portal 0, Message Ring, Entry 5	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x400_2180	QCSP0_MR6—QMan Software Portal 0, Message Ring, Entry 6	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
0x400_21C0	QCSP0_MR7—QMan Software Portal 0, Message Ring, Entry 7	R	0x0000_0000	<a href="#">3.2.3.3/3-18</a>
<b>QMan Software Portal 0, EQCR Index Registers, Cache-Inhibited Area</b>				
0x400_3000	QCSP0_EQCR_PI_CINH—Software Portal 0, Enqueue Command Ring, Producer Index Register, Cache-Inhibited	R/W	0x0000_0008	<a href="#">3.2.3.6.2/3-22</a>
0x400_3040	QCSP0_EQCR_CI_CINH—Software Portal 0, Enqueue Command Ring, Consumer Index Register, Cache-Inhibited	R	0x0000_8808	<a href="#">3.2.3.7.2/3-25</a>
0x400_3080	QCSP0_EQCR_ITR—Software Portal 0, EQCR Interrupt Threshold	R/W	0x0000_0000	<a href="#">3.2.3.18/3-44</a>

**Table 3-1. QMan Software Portals (QCSP) Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
<b>QMan Software Portal 0, DQRR Index and Command Registers, Cache-Inhibited Area</b>				
0x400_3100	QCSP0_DQRR_PI_CINH—Software Portal 0, Dequeue Response Ring, Producer Index Register, Cache-Inhibited	R	0x0000_0010	<a href="#">3.2.3.8.2/3-27</a>
0x400_3140	QCSP0_DQRR_CI_CINH—Software Portal 0, Dequeue Response Ring, Consumer Index Register, Cache-Inhibited	R/W	0x0000_0000	<a href="#">3.2.3.9.2/3-29</a>
0x400_3180	QCSP0_DQRR_ITR—Software Portal 0, DQRR Interrupt Threshold	R/W	0x0000_0000	<a href="#">3.2.3.19/3-45</a>
0x400_31C0	QCSP0_DQRR_DCAP—Software Portal 0, Dequeue Response Ring, Discrete Consumption Acknowledgment and Park Register	W	0x0000_0000	<a href="#">3.2.3.10/3-29</a>
0x400_3200	QCSP0_DQRR_SDQCR—Software Portal 0, Static Dequeue Command Register	R/W	0x0000_0000	<a href="#">3.2.3.11/3-31</a>
0x400_3240	QCSP0_DQRR_VDQCR—Software Portal 0, Volatile Dequeue Command Register	R/W	0x0000_0000	<a href="#">3.2.3.12/3-34</a>
0x400_3280	QCSP0_DQRR_PDQCR—Software Portal 0, Pull Dequeue Command Register	R/W	0x0000_0000	<a href="#">3.2.3.13/3-35</a>
<b>QMan Software Portal 0, MR Index Registers, Cache-Inhibited Area</b>				
0x400_3300	QCSP0_MR_PI_CINH—Software Portal 0, Message Ring, Producer Index Register, Cache-Inhibited	R	0x0000_0008	<a href="#">3.2.3.14.2/3-37</a>
0x400_3340	QCSP0_MR_CI_CINH—Software Portal 0, Message Ring, Consumer Index Register, Cache-Inhibited	R/W	0x0000_0000	<a href="#">3.2.3.15.2/3-39</a>
0x400_3380	QCSP0_MR_ITR—Software Portal 0, MR Interrupt Threshold	R/W	0x0000_0000	<a href="#">3.2.3.20/3-45</a>
<b>QMan Software Portal 0, Configuration Registers, Cache-Inhibited Area</b>				
0x400_3500	QCSP0_CFG—Software Portal 0, Configuration Register	R/W	0x0000_0000	<a href="#">3.2.3.17/3-41</a>
<b>QMan Software Portal 0, Interrupt Registers, Cache-Inhibited Area</b>				
0x400_3600	QCSP0_ISR—Software Portal 0, Interrupt Status Register	w1c	0x0000_0000	<a href="#">3.2.3.21/3-46</a>
0x400_3640	QCSP0_IER—Software Portal 0, Interrupt Enable Register	R/W	0x0000_0000	<a href="#">3.2.3.22/3-47</a>
0x400_3680	QCSP0_ISDR—Software Portal 0, Interrupt Status Disable	R/W	0x0000_0000	<a href="#">3.2.3.23/3-48</a>
0x400_36C0	QCSP0_IIR—Software Portal 0, Interrupt Inhibit Register	R/W	0x0000_0000	<a href="#">3.2.3.24/3-48</a>
0x400_3740	QCSP0_ITPR—Software Portal 0 Interrupt Time out Period Register	R/W	0x0000_0000	<a href="#">3.2.3.25/3-49</a>
<b>QMan Software Portal 0, Management Registers, Cache-Inhibited Area</b>				
0x400_3800	QCSP0_CR—Software Portal 0, Management Command Register	R/W	0x0000_0000	<a href="#">3.2.3.4/3-19</a>
0x400_3900	QCSP0_RR0—Portal 0, Management Response Register 0	R	0x0000_0000	<a href="#">3.2.3.5/3-20</a>
0x400_3940	QCSP0_RR1—Portal 0, Management Response Register 1	R	0x0000_0000	<a href="#">3.2.3.5/3-20</a>
0x400_4000–0x400_FFFF	Reserved, QMan Software Portal 0, cache-inhibited area	—	—	—
<b>QMan Software Portals 1 and Above, Cache-Inhibited Area</b>				
0x401_0000–0x401_FFFF	QMan Software Portal 1, cache-inhibited area (64 KB)	—	—	—

**Table 3-1. QMan Software Portals (QCSP) Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x402_0000 and above	QMan Software Portal cache-inhibited area, 64 KB per portal In the LS1043A: 10 software portals occupy 0x400_0000 to 0x409_FFFF	—	—	—
0x402_0000–0x7FF_FFFF	64 KB blocks corresponding to non-existent software portals are reserved.	—	—	—

### 3.2.2 QMan Configuration and Control Register Memory Map

The QMan occupies a 8 KB aligned block in the configuration, control, and status register (CCSR) space of the system. CCSR space is a fixed 2 MB in size, and all registers in the system are accessed using a 21-bit address offset from CCSRBAR. QMan's 8 KB block begins at offset 0x08\_8000 within CCSR space. The address offsets shown in [Table 3-2](#) refer to the 13-bit address offset within the QMan's 8 KB block, which is used to address the individual registers within the QMan.

#### NOTE

It is a programming error to attempt 8- or 16-bit accesses in these 32-bit wide registers.

**Table 3-2. QMan Configuration and Control Register Memory Map**

Offset	Register	Access	Reset Value	Section/Page
<b>Dynamic Debug (DD) Configuration Registers</b>				
0x200	QMAM_DD_CFG—QMan DD Configuration	R/W	0x0000_0000	<a href="#">3.2.4.4/3-52</a>
0x220	DCP_DD_IHRSR—DCP DD Internal Halt Request Status	w1c	0x0000_0000	<a href="#">3.2.4.6/3-54</a>
0x224	DCP_DD_IHRFR—DCP DD Internal Halt Request Force	W	0x0000_0000	<a href="#">3.2.4.8/3-55</a>
0x228	DCP_DD_HASR—Direct Connect Portal DD Halt Acknowledge Status	R	0x0000_0000	<a href="#">3.2.4.10/3-57</a>
0x240	QCSP_DD_IHRSR_0—S/W Portal DD Internal Halt Request Status	w1c	0x0000_0000	<a href="#">3.2.4.5/3-53</a>
0x244	QCSP_DD_IHRFR_0— S/W Portal DD Internal Halt Request Force	W	0x0000_0000	<a href="#">3.2.4.7/3-55</a>
0x248	QCSP_DD_HASR_0— S/W Portal DD Halt Acknowledge Status	R	0x0000_0000	<a href="#">3.2.4.9/3-56</a>
0x250	QCSP_DD_IHRSR_1— S/W Portal DD Internal Halt Request Status	w1c	0x0000_0000	<a href="#">3.2.4.5/3-53</a>
0x254	QCSP_DD_IHRFR_1— S/W Portal DD Internal Halt Request Force	W	0x0000_0000	<a href="#">3.2.4.7/3-55</a>
0x258	QCSP_DD_HASR_1— S/W Portal DD Halt Acknowledge Status	R	0x0000_0000	<a href="#">3.2.4.9/3-56</a>
0x260	QCSP_DD_IHRSR_2— S/W Portal DD Internal Halt Request Status	w1c	0x0000_0000	<a href="#">3.2.4.5/3-53</a>
0x264	QCSP_DD_IHRFR_2— S/W Portal DD Internal Halt Request Force	W	0x0000_0000	<a href="#">3.2.4.7/3-55</a>
0x268	QCSP_DD_HASR_2—S/W Portal DD Halt Acknowledge Status	R	0x0000_0000	<a href="#">3.2.4.9/3-56</a>
<b>Direct Connect Portal (DCP) Configuration Registers</b>				
0x300	DCP0_CFG—DCP0 Configuration	R/W	0x0000_0000	<a href="#">3.2.4.11/3-58</a>
0x304	DCP0_DD_CFG—DCP0 Dynamic Debug Configuration	R/W	0x0000_0000	<a href="#">3.2.4.12/3-59</a>

**Table 3-2. QMan Configuration and Control Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x308	DCP0_DLM_CFG—DCP0 Dequeue Latency Monitor Configuration	R/W	0x0000_0000	<a href="#">3.2.4.13/3-60</a>
0x30C	DCP0_DLM_AVG—DCP0 Dequeue Latency Monitor Average	R/W	0x0000_0000	<a href="#">3.2.4.14/3-61</a>
0x320	DCP2_CFG—DCP2 Configuration	R/W	0x0000_0000	<a href="#">3.2.4.11/3-58</a>
0x324	DCP2_DD_CFG—DCP2 Dynamic Debug Configuration	R/W	0x0000_0000	<a href="#">3.2.4.12/3-59</a>
0x328	DCP2_DLM_CFG—DCP2 Dequeue Latency Monitor Configuration	R/W	0x0000_0000	<a href="#">3.2.4.13/3-60</a>
0x32C	DCP2_DLM_AVG—DCP2 Dequeue Latency Monitor Average	R/W	0x0000_0000	<a href="#">3.2.4.14/3-61</a>
<b>Packed Frame Descriptor Record (PFDR) Manager Query Registers</b>				
0x400	PFDR_FPC—PFDR Free Pool Count	R	0x0000_0000	<a href="#">3.2.4.15/3-62</a>
0x404	PFDR_FP_HEAD—PFDR Free Pool Head Pointer	R	0x0000_0000	<a href="#">3.2.4.16/3-62</a>
0x408	PFDR_FP_TAIL—PFDR Free Pool Tail Pointer	R	0x0000_0000	<a href="#">3.2.4.17/3-63</a>
0x410	PFDR_FP_LWIT—PFDR Free Pool Low Watermark Interrupt Threshold	R/W	0x0000_0000	<a href="#">3.2.4.18/3-63</a>
0x414	PFDR_CFG—PFDR Configuration	R/W	0x0000_0000	<a href="#">3.2.4.19/3-64</a>
<b>Single Frame Descriptor Record (SFDR) Manager Registers</b>				
0x500	SFDR_CFG—SFDR Configuration Register	R/W	0x0000_0000	<a href="#">3.2.4.20/3-64</a>
0x504	SFDR_IN_USE—SFDR In Use Register	R	0x0000_0001	<a href="#">3.2.4.21/3-65</a>
<b>Work Queue Semaphore and Context Manager Registers</b>				
0x600	WQ_CS_CFG0—Work Queue Class Scheduler Configuration 0	R/W	0x0000_0000	<a href="#">3.2.4.22/3-65</a>
0x604	WQ_CS_CFG1—Work Queue Class Scheduler Configuration 1	R/W	0x0000_0000	<a href="#">3.2.4.22/3-65</a>
0x608	WQ_CS_CFG2—Work Queue Class Scheduler Configuration 2	R/W	0x0000_0000	<a href="#">3.2.4.22/3-65</a>
0x610	WQ_CS_CFG4—Work Queue Class Scheduler Configuration 4	R/W	0x0000_0000	<a href="#">3.2.4.22/3-65</a>
0x630	WQ_DEF_ENQ_WQID—Work Queue Default Enqueue WQID	R/W	0x0000_0000	<a href="#">3.2.4.23/3-67</a>
0x680	WQ_PC_DD_CFG_0—WQ Pool Channel Dynamic Debug Config 0	R/W	0x0000_0000	<a href="#">3.2.4.24/3-67</a>
0x684	WQ_PC_DD_CFG_1—WQ Pool Channel Dynamic Debug Config 1	R/W	0x0000_0000	<a href="#">3.2.4.24/3-67</a>
0x6C0	WQ_DC0_DD_CFG_0—WQ DCP0 Chan. Dynamic Debug Config 0	R/W	0x0000_0000	<a href="#">3.2.4.24/3-67</a>
0x6C4	WQ_DC0_DD_CFG_1—WQ DCP0 Chan. Dynamic Debug Config 1	R/W	0x0000_0000	<a href="#">3.2.4.24/3-67</a>
0x6C8	WQ_DC0_DD_CFG_2—WQ DCP0 Chan. Dynamic Debug Config 2	R/W	0x0000_0000	<a href="#">3.2.4.24/3-67</a>
0x6CC	WQ_DC0_DD_CFG_3—WQ DCP0 Chan. Dynamic Debug Config 3	R/W	0x0000_0000	<a href="#">3.2.4.24/3-67</a>
0x740	WQ_DC2_DD_CFG_0—WQ DCP2 Chan. Dynamic Debug Config 0	R/W	0x0000_0000	<a href="#">3.2.4.24/3-67</a>
<b>Congestion Manager (CM) Registers</b>				
0x800	CM_CFG—CM Configuration Register	R/W	0x0000_0000	<a href="#">3.2.4.25/3-71</a>
<b>Customer Edge Egress Traffic Management Configuration (CEETM_CFG) Registers</b>				
0x900	CEETM_CFG_IDX—CEETM Configuration Index Register	R/W	0x0000_0000	<a href="#">3.2.4.26/3-72</a>

**Table 3-2. QMan Configuration and Control Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x904	CEETM_CFG_PRES—CEETM Configuration Pre-Scaler Register	R/W	0x0000_0000	<a href="#">3.2.4.27/3-72</a>
0x908	CEETM_XSFDR_IN_USE—CEETM XSFDR In Use Register	R	0x0000_0001	<a href="#">3.2.4.28/3-73</a>
<b>QMan Error Capture Registers</b>				
0xA00	QMANT_ECSR—QMan Error Capture Status Register	w1c	0x0000_0000	<a href="#">3.2.4.29/3-74</a>
0xA04	QMANT_ECIR—QMan Error Capture Information Register	R	0x0000_0000	<a href="#">3.2.4.30/3-75</a>
0xA08	QMANT_EADR—QMan ECC Error Address Register	R	0x0000_0000	<a href="#">3.2.4.31/3-76</a>
0xA0C	QMANT_ECIR2—QMan Error Capture Information Register 2	R	0x0000_0000	<a href="#">3.2.4.30/3-75</a>
0xA10	QMANT_EDATA0—QMan ECC Error Data 0 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA14	QMANT_EDATA1—QMan ECC Error Data 1 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA18	QMANT_EDATA2—QMan ECC Error Data 2 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA1C	QMANT_EDATA3—QMan ECC Error Data 3 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA20	QMANT_EDATA4—QMan ECC Error Data 4 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA24	QMANT_EDATA5—QMan ECC Error Data 5 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA28	QMANT_EDATA6—QMan ECC Error Data 6 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA2C	QMANT_EDATA7—QMan ECC Error Data 7 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA30	QMANT_EDATA8—QMan ECC Error Data 8 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA34	QMANT_EDATA9—QMan ECC Error Data 9 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA38	QMANT_EDATA10—QMan ECC Error Data 10 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA3C	QMANT_EDATA11—QMan ECC Error Data 11 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA40	QMANT_EDATA12—QMan ECC Error Data 12 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA44	QMANT_EDATA13—QMan ECC Error Data 13 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA48	QMANT_EDATA14—QMan ECC Error Data 14 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA4C	QMANT_EDATA15—QMan ECC Error Data 15 Register	R	0x0000_0000	<a href="#">3.2.4.32/3-77</a>
0xA70	QMANT_SBT—QMan Single Bit ECC Error Threshold Register	R/W	0x0000_0000	<a href="#">3.2.4.33/3-78</a>
0xA80	QMANT_SBEC0—QMan Single Bit ECC Error Count 0 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xA84	QMANT_SBEC1—QMan Single Bit ECC Error Count 1 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xA88	QMANT_SBEC2—QMan Single Bit ECC Error Count 2 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xA8C	QMANT_SBEC3—QMan Single Bit ECC Error Count 3 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xA90	QMANT_SBEC4—QMan Single Bit ECC Error Count 4 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xA94	QMANT_SBEC5—QMan Single Bit ECC Error Count 5 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xA98	QMANT_SBEC6—QMan Single Bit ECC Error Count 6 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xA9C	QMANT_SBEC7—QMan Single Bit ECC Error Count 7 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>

**Table 3-2. QMan Configuration and Control Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0xAA0	QMAM_SBEC8—QMan Single Bit ECC Error Count 8 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xAA4	QMAM_SBEC9—QMan Single Bit ECC Error Count 9 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xAA8	QMAM_SBEC10—QMan Single Bit ECC Error Count 10 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xAAC	QMAM_SBEC11—QMan Single Bit ECC Error Count 11 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xAB0	QMAM_SBEC12—QMan Single Bit ECC Error Count 12 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xAB4	QMAM_SBEC13—QMan Single Bit ECC Error Count 13 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
0xAB8	QMAM_SBEC14—QMan Single Bit ECC Error Count 14 Register	RR	0x0000_0000	<a href="#">3.2.4.34/3-79</a>
<b>QMan Initialization and Debug Control Registers</b>				
0xB00	QMAM_MCR—QMan Management Command/Result Register	R/W	0x0000_0000	<a href="#">3.2.4.35/3-80</a>
0xB04	QMAM_MCP0—QMan Management Command Parameter 0 Register	R/W	0x0000_0000	<a href="#">3.2.4.36/3-81</a>
0xB08	QMAM_MCP1—QMan Management Command Parameter 1 Register	R/W	0x0000_0000	<a href="#">3.2.4.37/3-83</a>
0xB20	QMAM_MR0—QMan Management Return Register 0	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB24	QMAM_MR1—QMan Management Return Register 1	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB28	QMAM_MR2—QMan Management Return Register 2	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB2C	QMAM_MR3—QMan Management Return Register 3	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB30	QMAM_MR4—QMan Management Return Register 4	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB34	QMAM_MR5—QMan Management Return Register 5	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB38	QMAM_MR6—QMan Management Return Register 6	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB3C	QMAM_MR7—QMan Management Return Register 7	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB40	QMAM_MR8—QMan Management Return Register 8	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB44	QMAM_MR9—QMan Management Return Register 9	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB48	QMAM_MR10—QMan Management Return Register 10	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB4C	QMAM_MR11—QMan Management Return Register 11	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB50	QMAM_MR12—QMan Management Return Register 12	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB54	QMAM_MR13—QMan Management Return Register 13	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB58	QMAM_MR14—QMan Management Return Register 14	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xB5C	QMAM_MR15—QMan Management Return Register 15	R	0x0000_0000	<a href="#">3.2.4.38/3-83</a>
0xBE0	QMAM_MISC_CFG—QMan Miscellaneous Configuration Register	R/W	0x0000_0000	<a href="#">3.2.4.39/3-84</a>
0xBF4	QMAM_IDLE_STAT—QMan Idle Status Register	R	0x0000_0003	<a href="#">3.2.4.40/3-85</a>
<b>QMan ID/Revision Registers</b>				
0xBF8	QMAM_IP_REV_1—QM IP Block Revision 1 register	R	0x0A01_0302	<a href="#">3.2.4.41/3-86</a>
0xBFC	QMAM_IP_REV_2—QM IP Block Revision 2 register	R	0x0000_0000	<a href="#">3.2.4.42/3-86</a>

**Table 3-2. QMan Configuration and Control Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
<b>QMan Initiator Interface Memory Window Configuration Registers</b>				
0xC00	FQD_BARE—FQD Extended Base Address Register	R/W	0x0000_0000	<a href="#">3.2.4.43/3-87</a>
0xC04	FQD_BAR—Frame Queue Descriptor (FQD) Base Address Register	R/W	0x0000_0000	<a href="#">3.2.4.43/3-87</a>
0xC10	FQD_AR—FQD Attributes Register	R/W	0x0000_0000	<a href="#">3.2.4.44/3-89</a>
0xC20	PFDR_BARE—PFDR Extended Base Address Register	R/W	0x0000_0000	<a href="#">3.2.4.43/3-87</a>
0xC24	PFDR_BAR—Packed Frame Descriptor Record (PFDR) Base Addr	R/W	0x0000_0000	<a href="#">3.2.4.43/3-87</a>
0xC30	PFDR_AR—PFDR Attributes Register	R/W	0x0000_0000	<a href="#">3.2.4.44/3-89</a>
0xC80	QCSP_BARE—QCSP Extended Base Address	R/W	0x0000_0000	<a href="#">3.2.4.45/3-91</a>
0xC84	QCSP_BAR—QMan Software Portal Base Address	R/W	0x0000_0000	<a href="#">3.2.4.45/3-91</a>
0xD00	CI_SCHED_CFG—Initiator Scheduling Configuration	R/W	0x0000_0000	<a href="#">3.2.4.46/3-93</a>
0xD04	QMANT_SRCIDR—QMan Source ID Register	R	0x0000_003C	<a href="#">3.2.4.47/3-93</a>
0xD08	QMANT_ICIDR—QMan Isolation Context Identifier Register	R/W	0x0000_0000	<a href="#">3.2.4.48/3-94</a>
0xD10	CI_RLM_CFG—Initiator Read Latency Monitor Configuration	R/W	0x0000_0000	<a href="#">3.2.4.49/3-95</a>
0xD14	CI_RLM_AVG—Initiator Read Latency Monitor Average	R/W	0x0000_0000	<a href="#">3.2.4.50/3-95</a>
<b>QMan Interrupt and Error Registers</b>				
0xE00	QMANT_ERR_ISR—QMan Error Interrupt Status Register	w1c	0x0080_0000	<a href="#">3.2.4.51/3-96</a>
0xE04	QMANT_ERR_IER—QMan Error Interrupt Enable Register	R/W	0x0000_0000	<a href="#">3.2.4.52/3-99</a>
0xE08	QMANT_ERR_ISDR—QMan Error Interrupt Status Disable Register	R/W	0x0000_0000	<a href="#">3.2.4.53/3-100</a>
0xE0C	QMANT_ERR_IIR—QMan Error Interrupt Inhibit Register	R/W	0x0000_0000	<a href="#">3.2.4.54/3-100</a>
0xE14	QMANT_ERR_HER—QMan Error Halt Enable Register	R/W	0x0000_0000	<a href="#">3.2.4.55/3-101</a>
<b>QMan Software Portal Configuration Registers</b>				
0x1000	QCSP0_LIO_CFG—QMan Software Portal 0 LIO Configuration	R/W	0x0000_0000	<a href="#">3.2.4.1/3-50</a>
0x1004	QCSP0_IO_CFG—QMan Software Portal 0 IO Configuration	R/W	0x0000_0000	<a href="#">3.2.4.2/3-50</a>
0x100C	QCSP0_DD_CFG—Software Portal 0 Dynamic Debug Configuration	R/W	0x0000_0000	<a href="#">3.2.4.3/3-51</a>
0x1010 and above	QCSPi_LIO_CFG, QCSPi_IO_CFG, and QCSPi_DD_CFG registers for each software portal In the LS1043A: 10 software portals occupy 0x1000 to 0x109C	—	—	—
<b>Software Portal Dedicated WQ Channel Dynamic Debug Configuration Registers</b>				
0x1E00	WQ_SC_DD_CFG_0—WQ S/W Channel Dynamic Debug Config 0	R/W	0x0000_0000	<a href="#">3.2.4.24/3-67</a>

**Table 3-2. QMan Configuration and Control Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x1E04	WQ_SC_DD_CFG_1—WQ S/W Channel Dynamic Debug Config 1	R/W	0x0000_0000	<a href="#">3.2.4.24/3-67</a>
0x1E08 and above	WQ_SC_DD_CFG_i registers, one for every two software portals In the LS1043A: 10 software portals occupy 0x1E00 to 0x1E10	—	—	—

### 3.2.3 QMan Software Portals (QCSP) Register Descriptions

This section describes the QMan software portals registers.

#### 3.2.3.1 QCSP Enqueue Command Ring Registers (QCSP*i*\_EQCR0–7)

Use the QCSP*i*\_EQCR0–7 to issue enqueue commands from software to the QMan. Each entry can be accessed using only one system bus transaction. The QCSP*i*\_EQCR0–7 consist of eight, separate 64-byte

## Queue Manager (QMan)

entries that are organized as a circular FIFO. See [Section 3.3.8.1, “Enqueue Command Ring \(EQCR\),”](#) for more information.

Offset 0x000_0000 (QCSP<i>_EQCR0)	Access: R/W
offset 0x10000	
range i=0..9	
0x000_0040 (QCSP<i>_EQCR1)	
offset 0x10000	
range i=0..9	
0x000_0080 (QCSP<i>_EQCR2)	
offset 0x10000	
range i=0..9	
0x000_00C0 (QCSP<i>_EQCR3)	
offset 0x10000	
range i=0..9	
0x000_0100 (QCSP<i>_EQCR4)	
offset 0x10000	
range i=0..9	
0x000_0140 (QCSP<i>_EQCR5)	
offset 0x10000	
range i=0..9	
0x000_0180 (QCSP<i>_EQCR6)	
offset 0x10000	
range i=0..9	
0x000_01C0 (QCSP<i>_EQCR7)	
offset 0x10000	
range i=0..9	

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VERB	COMMAND_DATA																														
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**Figure 3-2. QCSP Enqueue Command Ring Registers (QCSP*i*\_EQCR0-7)**

**Table 3-3. QCSP*i*\_EQCR0-7 Field Descriptions**

Field	Description
0 VERB	<p>The verb byte is subdivided into the following separate fields.</p> <p>Bit 0 (msb): Valid bit</p> <p>The function of this bit is determined by the EQCR production notification mode programmed by software for each portal; see <a href="#">Section 3.2.3.17, “QMan Software Portal Configuration Register (QCSP<i>i</i>_CFG).</a></p> <p>If EQCR production notification mode is set to valid bit mode, this bit must be set to the current valid bit polarity (1 or 0) in every command issued in an EQCR entry, and QMan uses this bit (along with the presence of a non-zero command verb) to determine when a valid entry is placed in the ring. The expected polarity of this bit, which indicates a valid entry, toggles each time the ring wraps around from entry 7 to entry 0, and this is referred to as an alternating polarity valid bit. See <a href="#">Section 3.3.8.1, “Enqueue Command Ring (EQCR),”</a> for more details.</p> <p>If EQCR production notification mode is set to one of the two producer index write modes, this bit is not used and is ignored by the QMan.</p> <p>Bits 1-7 (lsbs): Command verb</p> <p>Indicates the type of command being issued. See <a href="#">Section 3.3.9.1, “Enqueue Command,”</a> for a description of each of the possible commands.</p>
1-31 COMMAND_DATA	<p>Command data</p> <p>The information carried in these bytes is dependent on the type of command. See <a href="#">Section 3.3.9.1, “Enqueue Command,”</a> for a description of each of the possible commands.</p>
32-63	Reserved

### 3.2.3.2 QCSP Dequeue Response Ring Registers (QCSP*i*\_DQRR0-15)

Use the QMan software portal dequeue response ring registers (QCSP*i*\_DQRR0-15) to carry responses from the QMan to software. The content of the responses is described in [Section 3.3.9, “Software Portal Commands and Responses.”](#) Each entry can be accessed using only one bus transaction, and any writes to these ring entries are ignored by the QMan. The QCSP*i*\_DQRR0-15 consist of separate 64-byte entries that are organized as a circular FIFO. See [Section 3.3.8.2, “Dequeue Response Ring \(DQRR\),”](#) for more

## Queue Manager (QMan)

information. [Figure 3-3](#) shows the register for entries 0–7, and [Figure 3-4](#) shows the register for entries 8–15.

Offset 0x000_1000 (QCSP<i>_DQRR0)	Access: R
offset 0x10000	
range i=0..9	
0x000_1040 (QCSP<i>_DQRR1)	
offset 0x10000	
range i=0..9	
0x000_1080 (QCSP<i>_DQRR2)	
offset 0x10000	
range i=0..9	
0x000_10C0 (QCSP<i>_DQRR3)	
offset 0x10000	
range i=0..9	
0x000_1100 (QCSP<i>_DQRR4)	
offset 0x10000	
range i=0..9	
0x000_1140 (QCSP<i>_DQRR5)	
offset 0x10000	
range i=0..9	
0x000_1180 (QCSP<i>_DQRR6)	
offset 0x10000	
range i=0..9	
0x000_11C0 (QCSP<i>_DQRR7)	
offset 0x10000	
range i=0..9	

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VERB	RESPONSE_DATA																														
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**Figure 3-3. QCSP Dequeue Response Ring Registers (QCSP*i*\_DQRR0-15) (Entries 0 to 7)**

Offset 0x000\_1200 (QCSP<i>\_DQRR8) Access: R  
 offset 0x10000  
 range i=0..9  
 0x000\_1240 (QCSP<i>\_DQRR9)  
 offset 0x10000  
 range i=0..9  
 0x000\_1280 (QCSP<i>\_DQRR10)  
 offset 0x10000  
 range i=0..9  
 0x000\_12C0 (QCSP<i>\_DQRR11)  
 offset 0x10000  
 range i=0..9  
 0x000\_1300 (QCSP<i>\_DQRR12)  
 offset 0x10000  
 range i=0..9  
 0x000\_1340 (QCSP<i>\_DQRR13)  
 offset 0x10000  
 range i=0..9  
 0x000\_1380 (QCSP<i>\_DQRR14)  
 offset 0x10000  
 range i=0..9  
 0x000\_13C0 (QCSP<i>\_DQRR15)  
 offset 0x10000  
 range i=0..9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VERB																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-4. QCSP Dequeue Response Ring Registers (QCSP*i*\_DQRR0-15) (Entries 8 to 15)Table 3-4. QCSP*i*\_DQRR0-15 Field Descriptions

Field	Description
0 VERB	In the response ring, the VERB byte is subdivided into two separate fields. Bit 0 (msb): Valid This bit indicates to software whether this entry is valid or not. This bit is provided in all ring entries, and may be used or ignored by software. If the valid bit is not used, software must read the producer index (PI) of the ring to determine when the QMan has added entries to the ring. The polarity of this bit, which indicates a valid entry, toggles each time the ring wraps around from the last entry to entry 0, and this is referred to as an alternating polarity valid bit. See <a href="#">Section 3.3.8.2, “Dequeue Response Ring (DQRR),”</a> for more details. Bits 1–7 (lsbs): Response verb Identifies the type of response in each ring entry. See <a href="#">Section 3.3.9.2, “Frame Dequeue Response”</a> for a description of the possible responses.

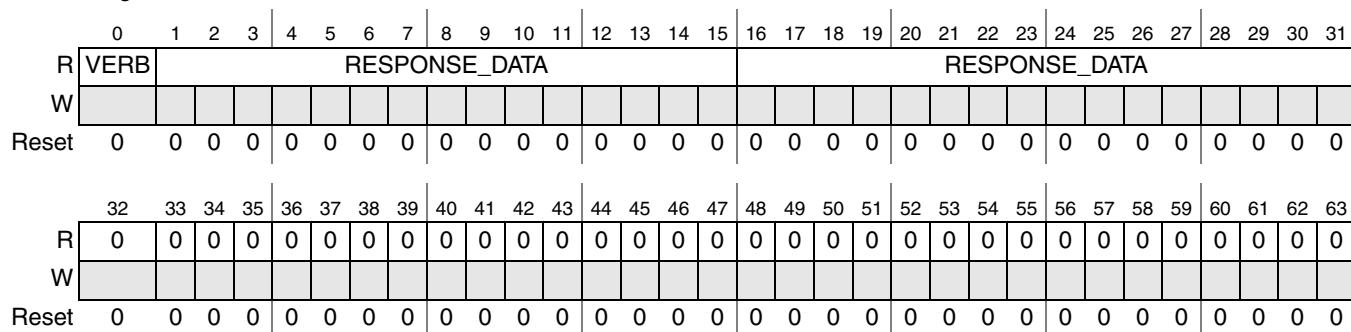
**Table 3-4. QCSP*i*\_DQRR0-15 Field Descriptions (continued)**

Field	Description
1-31 RESPONSE_DATA	Response data The information carried in these bytes is dependent on the type of response. See <a href="#">Section 3.3.9.2, “Frame Dequeue Response,”</a> for a description of the possible responses.
32-63	Reserved

### **3.2.3.3 QCSP Message Ring Registers (QCSP*i*\_MR0-7)**

Use the QMan software portal message ring registers (QCSPi\_MR0-7) to carry responses from the QMan to software. The content of the responses is described in [Section 3.3.9, “Software Portal Commands and Responses.”](#) Each entry can be accessed using only one bus transaction, and any writes to these ring entries are ignored by the QMan. The QCSPi\_MR0-7 consist of eight, separate 64-byte entries that are organized as a circular FIFO. See [Section 3.3.8.3, “Message Ring \(MR\),”](#) for more information.

Offset 0x000\_2000 (QCSP<i>\_MR0)  
offset 0x10000  
range i=0..9  
0x000\_2040 (QCSP<i>\_MR1)  
offset 0x10000  
range i=0..9  
0x000\_2080 (QCSP<i>\_MR2)  
offset 0x10000  
range i=0..9  
0x000\_20C0 (QCSP<i>\_MR3)  
offset 0x10000  
range i=0..9  
0x000\_2100 (QCSP<i>\_MR4)  
offset 0x10000  
range i=0..9  
0x000\_2140 (QCSP<i>\_MR5)  
offset 0x10000  
range i=0..9  
0x000\_2180 (QCSP<i>\_MR6)  
offset 0x10000  
range i=0..9  
0x000\_21C0 (QCSP<i>\_MR7)  
offset 0x10000  
range i=0..9



**Figure 3-5. QCSP Message Ring Registers (QCSP*i* MR0-7)**

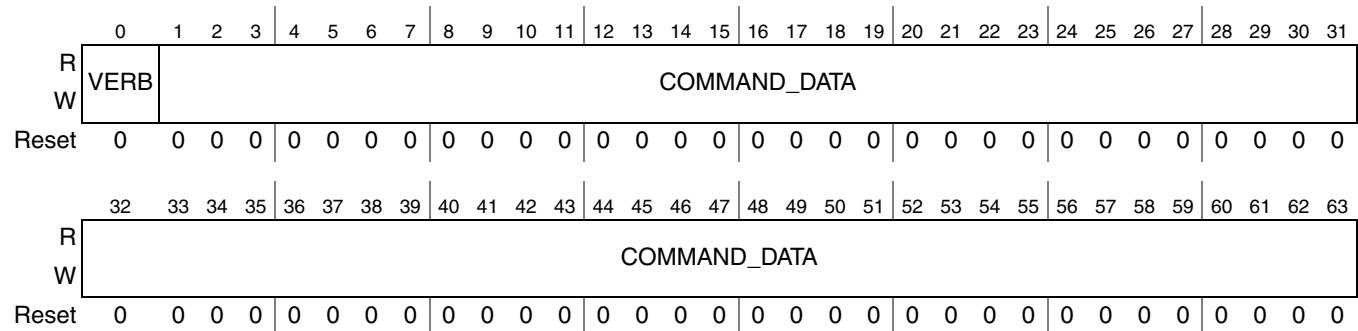
**Table 3-5. QCSP*i*\_MR0-7 Field Descriptions**

Field	Description
0 VERB	<p>In the response ring, the VERB byte is subdivided into two separate fields.</p> <p>Bit 0 (msb): Valid</p> <ul style="list-style-type: none"> <li>This bit indicates to software whether this entry is valid or not.</li> <li>This bit is provided in all ring entries, and may be used or ignored by software. If the valid bit is not used, software must read the producer index (PI) of the ring to determine when the QMan has added entries to the ring. The polarity of this bit, which indicates a valid entry, toggles each time the ring wraps around from the last entry to entry 0, and this is referred to as an alternating polarity valid bit. See <a href="#">Section 3.3.8.3, “Message Ring (MR),”</a> for more details.</li> </ul> <p>Bits 1–7 (lsbs): Response verb</p> <p>Identifies the type of response in each ring entry. See <a href="#">Section 3.3.9.3, “ERN Message Response,”</a> and <a href="#">Section 3.3.9.4, “FQ State Change Notification Message Response,”</a> for a description of each of the possible responses.</p>
1–31 RESPONSE_DATA	<p>Response data</p> <p>The information carried in these bytes is dependent on the type of response. See <a href="#">Section 3.3.9.3, “ERN Message Response,”</a> and <a href="#">Section 3.3.9.4, “FQ State Change Notification Message Response,”</a> for a description of each of the possible responses.</p>
32–63	Reserved

### 3.2.3.4 QCSP Management Command Registers (QCSPi\_CR)

The QCSP management command registers ( $\text{QCSP}_i\text{-CR}$ ) can be accessed using only one bus transaction. See [Section 3.3.8.4, “Management Command Register \(CR\),”](#) for more information.

Offset 0x000\_3800 (QCSP<i>\_CR) Access: R/W  
offset 0x10000  
range i=0..9



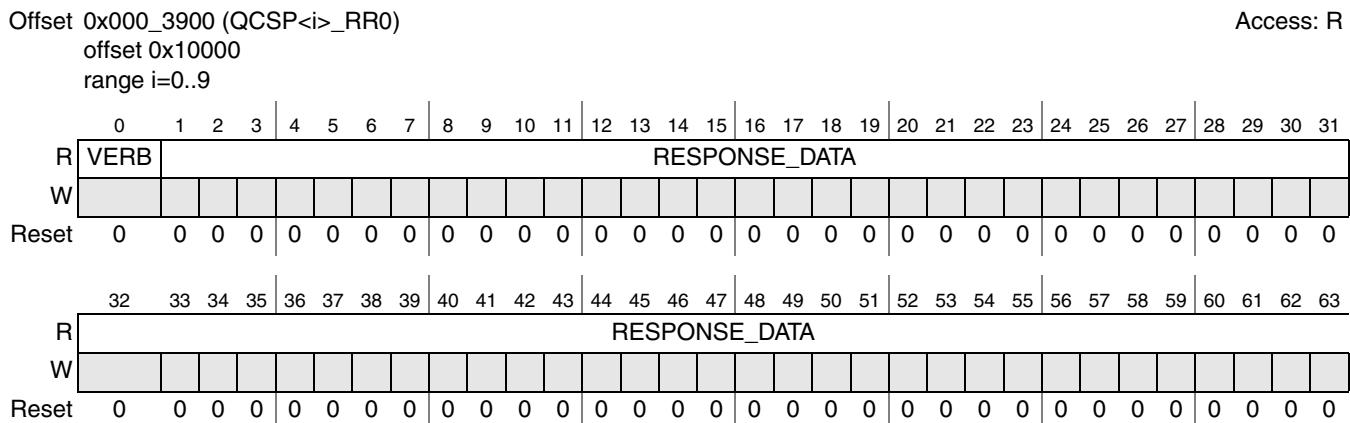
**Figure 3-6. QCSP Management Command Registers (QCSP*i* CR)**

**Table 3-6. QCSP*i*\_CR Field Descriptions**

Field	Description
0 VERB	<p>The VERB byte is subdivided into the following separate fields.</p> <p>Bit 0 (msb): Valid This bit indicates to the QMan whether this CR command is valid or not. The expected polarity of this bit, which indicates a valid command, toggles each time that a new command is written to the CR, and this is referred to as an alternating polarity valid bit. This bit also indicates which of the two management response registers (RR0/1) are used to return the response to this command. See <a href="#">Section 3.3.8.4, “Management Command Register (CR),”</a> for more details.</p> <p>Bit 1–7 (lsbs): Command verb Indicates the type of command being issued. See <a href="#">Section 3.3.9.5, “Frame Queue Management Commands,”</a> and <a href="#">Section 3.3.9.6, “Congestion Group Management Commands,”</a> for a description of each of the possible commands. In the CR, all commands must have a non-zero value in bits 1–2; command verbs 0x01 to 0x1f are invalid in the CR. A write to the CR with bits 1–2 = 0 and bits 3–7 non-zero is treated as follows:</p> <ul style="list-style-type: none"> <li>• The write data is stored, but the QMan takes no action on the command.</li> <li>• The RR0/RR1 registers are not updated. The previous response remains valid in RR0 or RR1.</li> <li>• An invalid command verb (ICVI) error interrupt is raised in QMAN_ERR_ISR, if enabled.</li> </ul> <p>Any CR command with bits 1–2 non-zero is dispatched for execution and results in a response in RR0 or RR1. That response may be “invalid command” if the command verb does not correspond to one of the defined FQ or congestion group management commands.</p>
1–63 COMMAND_DATA	<p>Command data The information carried in these bytes is dependent on the type of command. See <a href="#">Section 3.3.9.5, “Frame Queue Management Commands,”</a> and <a href="#">Section 3.3.9.6, “Congestion Group Management Commands,”</a> for a description of each of the possible commands.</p>

### 3.2.3.5 QCSP Management Response Registers (QCSP*i*\_RR0–1) Format

The QMan software portal management response registers (QCSP*i*\_RR0–1) are described in more detail in [Section 3.3.8.5, “Management Response Registers \(RR0/RR1\),”](#)

**Figure 3-7. QCSP Management Response Register 0 (QCSP*i*\_RR0)**

Offset 0x000\_3940 (QCSP*i*\_RR1)

offset 0x10000

range i=0..9

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	VERB																																
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
R																																	
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**Figure 3-8. QCSP Management Response Register 1 (QCSP*i*\_RR1)****Table 3-7. QCSP*i*\_RR0-1 Field Descriptions**

Field	Description
0 VERB	The verb byte is subdivided into two separate fields. Bit 0 (msb): RR identification bit This bit is the same as QCSP <i>i</i> _CR[VALID] for the command which initiated this response. As such, this bit is always 0 in RR0, and is always 1 when a valid response is present in RR1. Bit 1–7 (lsbs): Response verb Indicates the type of response. See <a href="#">Section 3.3.9.5, “Frame Queue Management Commands,”</a> and <a href="#">Section 3.3.9.6, “Congestion Group Management Commands,”</a> for a description of each of the possible responses.
1-63 RESPONSE_DATA	Response data The information carried in these bytes is dependent on the type of response. See <a href="#">Section 3.3.9.5, “Frame Queue Management Commands,”</a> and <a href="#">Section 3.3.9.6, “Congestion Group Management Commands,”</a> for a description of each of the possible responses.

### 3.2.3.6 QCSP Enqueue Command Ring (EQCR) Producer Index Registers

If the EQCR production notification mode is programmed by software to use one of the two producer index write modes, the EQCR\_PI value is used by software to notify the QMan that one or more valid entries have been written into the EQCR ring. EQCR\_PI is a single value stored in the QMan and is available in one of the following separate locations:

- Cache-enabled register (QCSP*i*\_EQCR\_PI\_CENA)
- Cache-inhibited register (QCSP*i*\_EQCR\_PI\_CINH)

One of these registers is read/write, and the other register is read only. The EQCR production notification mode programmed by software determines which of the two registers is read/write and which one is read only (see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSP\*i\*\\_CFG\)”](#)).

If EQCR production notification mode is programmed for valid bit mode rather than one of the two producer index write modes, then the QMan implicitly updates the EQCR\_PI value using the valid bits provided in the ring entries, and in this mode, both of the EQCR\_PI registers are read only.

### **3.2.3.6.1 QCSP EQCR Producer Index Cache-Enabled Registers (QCSP*i*\_EQCR\_PI\_CENA)**

When accessed in the cache-enabled location, QCSP*i*\_EQCR\_PI\_CENA is located in word 0 (the word at address offset 0) of the cache line.

Offset 0x000_3000 (QCSP<i>_EQCR_PI_CENA)			Access: R/W																											
offset 0x10000																														
range i=0..9																														
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VP	PI		
W	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

**Figure 3-9. QCSP EQCR Producer Index Registers (QCSP*i*\_EQCR\_PI\_CENA)**

**Table 3-8. QCSP*i*\_EQCR\_PI\_CENA Field Descriptions**

Field	Description
0-27	Reserved
28 VP	<p>Current valid bit polarity at producer index</p> <p>This bit is read only and is 1 after reset. Indicates the polarity that the QMan is currently expecting to see in the alternating polarity valid bit in the EQCR entry currently indexed by EQCR_PI.</p> <ul style="list-style-type: none"> <li>• If this bit is 1, the QMan is expecting a 1 to indicate a valid entry, and 0 indicates an invalid entry.</li> <li>• If this bit is 0, the QMan is expecting a 0 to indicate a valid entry, and 1 indicates an invalid entry.</li> </ul> <p>This bit toggles every time that the QMan's EQCR_PI value wraps from 7 back to 0. Reading this bit can be used as a debug or verification assist, or to re-synchronize software's copy of this bit with the QMan's, if needed. If EQCR production notification mode is programmed to valid bit mode, the QMan uses the value indicated in this bit to determine when valid entries have been written into the EQCR ring. If EQCR production notification is not in valid bit mode, the value in this bit is not used by the QMan.</p>
29-31 PI	<p>Producer index</p> <p>This field is read only. It indicates the ring entry in which the next valid command will be written. When EQCR_PI and EQCR_CI are equal, the QMan considers the ring to be empty and waits for EQCR_PI to be advanced. This field may be read/write or read only depending on the programmed EQCR production notification mode.</p> <ul style="list-style-type: none"> <li>• If QCSPi_CFG[EPM] is 0: PI is read/write in EQCR_PI_CINH and read only in QCSPi_EQCR_PI_CENA.</li> <li>• If QCSPi_CFG[EPM] is 1: PI is read/write in QCSPi_EQCR_PI_CENA and read only in EQCR_PI_CINH.</li> <li>• If QCSPi_CFG[EPM] is 2 or 3: PI is read only in both EQCR_PI_CINH and QCSPi_EQCR_PI_CENA.</li> </ul>

### **3.2.3.6.2 QCSP EQCR Producer Index Cache-Inhibited Registers (QCSP*i* EQCR PI CINH)**

Offset 0x400_3000 (QCSP<i>_EQCR_PI_CINH)																								Access: R/W															
offset 0x10000																																							
range i=0..9																																							
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VP	PI										
W																																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0								

**Figure 3-10. QCSP EQCR Producer Index Register (QCSP*i*\_EQCR\_PI\_CINH)**

**Table 3-9. QCSP*i*\_EQCR\_PI\_CINH Field Descriptions**

<b>Field</b>	<b>Description</b>
0-27	Reserved
28 VP	<p>Current valid bit polarity at producer index</p> <p>This bit is read only and is 1 after reset. Indicates the polarity that the QMan is currently expecting to see in the alternating polarity valid bit in the EQCR entry currently indexed by EQCR_PI</p> <ul style="list-style-type: none"> <li>If this bit is 1, the QMan is expecting a 1 to indicate a valid entry, and 0 indicates an invalid entry.</li> <li>If this bit is 0, the QMan is expecting a 0 to indicate a valid entry, and 1 indicates an invalid entry.</li> </ul> <p>This bit toggles every time that the QMan's EQCR_PI value wraps from 7 back to 0. Reading this bit can be used as a debug or verification assist, or to re-synchronize software's copy of this bit with the QMan's, if needed. If EQCR production notification mode is programmed to valid bit mode, the QMan uses the value indicated in this bit to determine when valid entries have been written into the EQCR ring. If EQCR production notification is not in valid bit mode, the value in this bit is not used by the QMan.</p>
29-31 PI	<p>Producer index</p> <p>This field is read only. It indicates the ring entry in which the next valid command will be written. When EQCR_PI and EQCR_CI are equal, the QMan considers the ring to be empty and waits for EQCR_PI to be advanced. This field may be read/write or read only depending on the programmed EQCR production notification mode.</p> <ul style="list-style-type: none"> <li>If QCSP<i>i</i>_CFG[EPM] is 0: PI is read/write in EQCR_PI_CINH and read only in QCSP<i>i</i>_EQCR_PI_CENA.</li> <li>If QCSP<i>i</i>_CFG[EPM] is 1: PI is read/write in QCSP<i>i</i>_EQCR_PI_CENA and read only in EQCR_PI_CINH.</li> <li>If QCSP<i>i</i>_CFG[EPM] is 2 or 3: PI is read only in both EQCR_PI_CINH and QCSP<i>i</i>_EQCR_PI_CENA.</li> </ul>

### 3.2.3.7 QCSP EQCR Consumer Index Registers

The EQCR\_CI value indicates which entry in the ring is consumed next by the QMan. Software reads this value to determine when it is safe to add new entries to the ring. EQCR\_CI is a single, read-only value stored in the QMan, and is available in one of the following separate locations:

- Separate, cache-enabled register (QCSP*i*\_EQCR\_CI\_CENA)
- Combined with the other read-only indices in a single, cache-enabled register (QCSP*i*\_RORI\_CENA)
- Cache-inhibited register (QCSP*i*\_EQCR\_CI\_CINH)

These registers are by definition read-only, however QCSP*i*\_EQCR\_CI\_CENA is also writeable when EQCR\_CI stashing is enabled, although writes to this register have no effect other than for coherency maintenance, see [Section 3.3.8.7, “EQCR\\_CI Stashing.”](#)

### 3.2.3.7.1 QCSP EQCR Consumer Index Cache-Enabled Registers (QCSP*i*\_EQCR\_CI\_CENA)

When accessed in the cache-enabled location, QCSP*i*\_EQCR\_CI\_CENA is located in word 0 (the word at address offset 0) of the cache line.

Offset 0x000_3040 (QCSP<i>_EQCR_CI_CENA)	Access: R
offset 0x10000	
range i=0..9	
R	0 1 2 3   4 5 6 7   8 9 10 11   12 13 14 15   16 17 18 19   20 21 22 23   24 25 26 27   28 29 30 31
W	0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0   VP 0 0 0   PB 0 0 0   0 0 0 0   VC CI
Reset	0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0   1 0 0 0   1 0 0 0   0 0 0 0   1 0 0 0

Figure 3-11. QCSP EQCR Consumer Index Register (QCSP*i*\_EQCR\_CI\_CENA)

Table 3-10. QCSP*i*\_EQCR\_CI\_CENA Field Descriptions

Field	Description
0-15	Reserved
16 VP	Current valid bit polarity at producer index This bit is 1 after reset. This bit is a copy of EQCR_PI[VP]; see <a href="#">Section 3.2.3.6.1, “QCSP EQCR Producer Index Cache-Enabled Registers (QCSP<i>i</i>_EQCR_PI_CENA).</a>
17-19	Reserved
20 PB	PFDR enqueues blocked This bit is 1 after reset, and negates when PFDR initialization begins. This bit is asserted when enqueues are blocked due to insufficient PFDR; see <a href="#">Section 3.2.4.19, “PFDR Configuration (PFDR_CFG).</a>
21-27	Reserved
28 VC	Current valid bit polarity at consumer index This bit is 1 after reset. Indicates the polarity that the QMan is currently expecting to see in the alternating polarity valid bit in the EQCR entry currently indexed by EQCR_CI. <ul style="list-style-type: none"> <li>• If this bit is 1, QMan is expecting a 1 to indicate a valid entry, and 0 indicates an invalid entry.</li> <li>• If this bit is 0, QMan is expecting a 0 to indicate a valid entry, and 1 indicates an invalid entry.</li> </ul> This bit toggles every time that the QMan’s EQCR_CI value wraps from 7 back to 0. Reading this bit can be used as a debug or verification assist, or to re-synchronize software’s copy of this bit with the QMan’s if needed. If EQCR production notification mode is programmed to valid bit mode, the QMan uses the value indicated in this bit to determine when valid entries have been written into the EQCR ring. If EQCR production notification is not in valid bit mode, the value in this bit is not used by the QMan.
29-31 CI	Consumer index This field indicates the ring entry that consumed next by the QMan. When the difference between EQCR_PI and EQCR_CI is greater than or equal to the maximum number of entries that the ring can hold, which for EQCR is 7, the ring is full and software must not add any new entries to the ring until the QMan has consumed an entry and advanced CI.

### 3.2.3.7.2 QCSP EQCR Consumer Index Cache-Inhibited Registers (QCSP*i*\_EQCR\_CI\_CINH)

Offset 0x400\_3040 (QCSP&lt;i&gt;\_EQCR\_CI\_CINH)

Access: R

offset 0x10000

range i=0..9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VP	0	0	0	PB	0	0	0	0	0	0	0	VC	CI				
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	

Figure 3-12. QCSP EQCR Consumer Index Register (QCSP*i*\_EQCR\_CI\_CINH)Table 3-11. QCSP*i*\_EQCR\_CI\_CINH Field Descriptions

Field	Description
0-15	Reserved
16 VP	Current valid bit polarity at producer index This bit is 1 after reset. This bit is a copy of EQCR_PI[VP]; see <a href="#">Section 3.2.3.6.1, “QCSP EQCR Producer Index Cache-Enabled Registers (QCSP<i>i</i>_EQCR_PI_CENA).</a>
17-19	Reserved
20 PB	PFDR enqueues blocked This bit is 1 after reset, and negates when PFDR initialization begins. This bit is asserted when enqueues are blocked due to insufficient PFDR; see <a href="#">Section 3.2.4.19, “PFDR Configuration (PFDR_CFG).</a>
21-27	Reserved
28 VC	Current valid bit polarity at consumer index This bit is 1 after reset. Indicates the polarity that the QMan is currently expecting to see in the alternating polarity valid bit in the EQCR entry currently indexed by EQCR_CI. <ul style="list-style-type: none"> <li>• If this bit is 1, QMan is expecting a 1 to indicate a valid entry, and 0 indicates an invalid entry.</li> <li>• If this bit is 0, QMan is expecting a 0 to indicate a valid entry, and 1 indicates an invalid entry.</li> </ul> This bit toggles every time that the QMan’s EQCR_CI value wraps from 7 back to 0. Reading this bit can be used as a debug or verification assist, or to re-synchronize software’s copy of this bit with the QMan’s if needed. If EQCR production notification mode is programmed to valid bit mode, the QMan uses the value indicated in this bit to determine when valid entries have been written into the EQCR ring. If EQCR production notification is not in valid bit mode, the value in this bit is not used by the QMan.
29-31 CI	Consumer index This field indicates the ring entry that consumed next by the QMan. When the difference between EQCR_PI and EQCR_CI is greater than or equal to the maximum number of entries that the ring can hold, which for EQCR is 7, the ring is full and software must not add any new entries to the ring until the QMan has consumed an entry and advanced CI.

### 3.2.3.8 QCSP Dequeue Response Ring (DQRR) Producer Index Registers

The DQRR\_PI value is used by the QMan to notify software that one or more valid entries have been written into the DQRR. Software may either read this value or use the alternating polarity valid bit in each entry to determine when an entry in the ring is valid and ready to be consumed. DQRR\_PI is a single, read-only value stored in the QMan, and is available as one of the following separate locations:

- Separate, cache-enabled register (QCSP*i*\_DQRR\_PI\_CENA)

- Combined with the other read-only indices in a single, cache-enabled register (QCSP*i*\_RORI\_CENA)
- Cache-inhibited register (QCSP*i*\_DQRR\_PI\_CINH)

### 3.2.3.8.1 QCSP DQRR Producer Index Cache-Enabled Registers (QCSP*i*\_DQRR\_PI\_CENA)

When accessed in the cache-enabled location, QCSP*i*\_DQRR\_PI\_CENA is located in word 0 (the word at address offset 0) of the cache line.

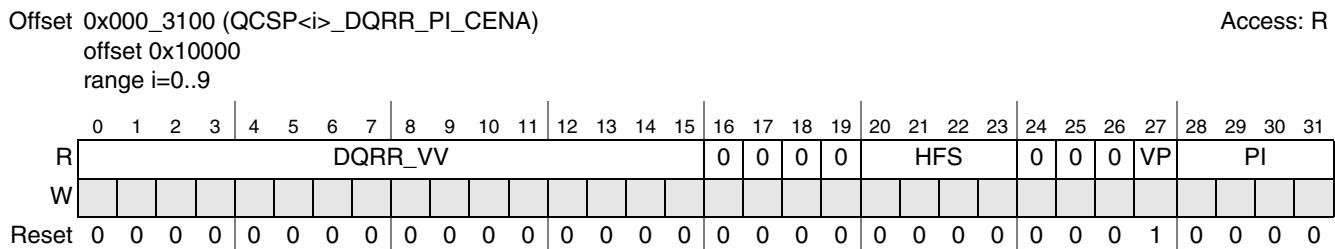


Figure 3-13. QCSP DQRR Producer Index Register (DQRR*i*\_PI\_CENA)

Table 3-12. QCSP*i*\_DQRR\_PI\_CENA Field Descriptions

Field	Description
0-15 DQRR_VV	DQRR valid vector This field contains a bitwise vector indicating the validity status of each DQRR entry. The QMan maintains these state bits for the purpose of discrete consumption acknowledgments, and the ability to read these bits is provided as a debug and verification assist. The state bit for an entry is asserted by the QMan when that entry is produced into the ring, and is cleared by the QMan when a consumption acknowledgment is received from software. Bits 0–15 = state of DQRR entry 0–15. For each bit: 0 Entry is invalid 1 Entry is valid
16-19	Reserved
20-23 HFS	Held FQ status This field contains a bit indicating the occupancy status of the four portal-specific cache locations (PSCL) in the DQRR; see <a href="#">Section 3.3.8.2.8, “Active and Suspended Frame Queues.”</a> The ability to read these bits is provided as a debug and verification assist. Bit 20: PSCL 0 status: 0 Empty 1 Occupied Bit 21: PSCL 1 status Bit 22: PSCL 2 status Bit 23: PSCL 3 status
24-26	Reserved

**Table 3-12. QCSP*i*\_DQRR\_PI\_CENA Field Descriptions (continued)**

Field	Description
27 VP	<p>Current valid bit polarity at producer index</p> <p>This bit is 1 after reset. Indicates the polarity that the QMan is currently using in the alternating polarity valid bit in the DQRR entries. This bit toggles every time that the QMan's DQRR_PI value wraps from 15 back to 0.</p> <ul style="list-style-type: none"> <li>• If this bit is 1, QMan is writing a 1 to indicate a valid entry, and 0 indicates an invalid entry.</li> <li>• If this bit is 0, QMan is writing a 0 to indicate a valid entry, and 1 indicates an invalid entry.</li> </ul> <p>Reading this bit can be used as a debug or verification assist, or to re-synchronize software's copy of this bit with the QMan's, if needed.</p>
28-31 PI	<p>Producer index</p> <p>This field indicates the ring entry in which the next valid response is to be written. When PI and CI are equal, the ring is empty, and the QMan advances PI after adding a new entry to the ring.</p>

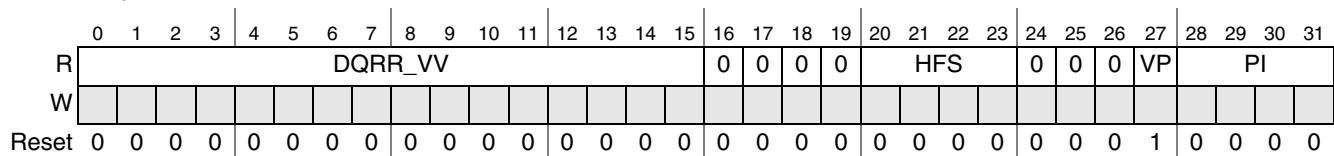
### **3.2.3.8.2 QCSP DQRR Producer Index Cache-Inhibited Registers (QCSP*i*\_DQRR\_PI\_CINH)**

Offset 0x400\_3100 (QCSP<i>\_DQRR\_PI\_CINH)

Access: R

offset 0x10000

range i=0..9



**Figure 3-14. QCSP DQRR Producer Index Cache-Inhibited Registers (QCSP*i*\_DQRR\_PI\_CINH)**

**Table 3-13. QCSP*i*\_DQRR\_PI\_CINH Field Descriptions**

Field	Description
0-15 DQRR_VV	<p>DQRR valid vector</p> <p>This field contains a bitwise vector indicating the validity status of each DQRR entry. The QMan maintains these state bits for the purpose of discrete consumption acknowledgments, and the ability to read these bits is provided as a debug and verification assist.</p> <p>The state bit for an entry is asserted by the QMan when that entry is produced into the ring, and is cleared by the QMan when a consumption acknowledgment is received from software.</p> <p>Bits 0–15 = state of DQRR entry 0–15.</p> <p>For each bit:</p> <ul style="list-style-type: none"> <li>0 Entry is invalid</li> <li>1 Entry is valid</li> </ul>
16-19	Reserved
20-23 HFS	<p>Held FQ status</p> <p>This field contains a bit indicating the occupancy status of the four portal-specific cache locations (PSCL) in the DQRR; see <a href="#">Section 3.3.8.2.8, “Active and Suspended Frame Queues.”</a> The ability to read these bits is provided as a debug and verification assist.</p> <p>Bit 20: PSCL 0 status:</p> <ul style="list-style-type: none"> <li>0 Empty</li> <li>1 Occupied</li> </ul> <p>Bit 21: PSCL 1 status</p> <p>Bit 22: PSCL 2 status</p> <p>Bit 23: PSCL 3 status</p>

**Table 3-13. QCSP*i*\_DQRR\_PI\_CINH Field Descriptions (continued)**

Field	Description
24-26	Reserved
27 VP	<p>Current valid bit polarity at producer index</p> <p>This bit is 1 after reset. Indicates the polarity that the QMan is currently using in the alternating polarity valid bit in the DQRR entries. This bit toggles every time that the QMan's DQRR_PI value wraps from 15 back to 0.</p> <ul style="list-style-type: none"> <li>• If this bit is 1, QMan is writing a 1 to indicate a valid entry, and 0 indicates an invalid entry.</li> <li>• If this bit is 0, QMan is writing a 0 to indicate a valid entry, and 1 indicates an invalid entry.</li> </ul> <p>Reading this bit can be used as a debug or verification assist, or to re-synchronize software's copy of this bit with the QMan's, if needed.</p>
28-31 PI	<p>Producer index</p> <p>This field indicates the ring entry in which the next valid response is to be written. When PI and CI are equal, the ring is empty, and the QMan advances PI after adding a new entry to the ring.</p>

### **3.2.3.9 QCSP DQRR Consumer Index Registers (QCSP*i* DQRR CI CENA)**

If the DQRR consumption notification mode is programmed by software to use one of the two consumer index write modes, the DQRR\_CI value is used by software to notify the QMan that one or more entries have been consumed from the DQRR. DQRR\_CI is a single value stored in QMan, and is available in one of the following separate locations:

- Cache-enabled register (QCSPI\_DQRR\_CI\_CENA)
  - Cache-inhibited register (QCSPI\_DQRR\_CI\_CINH)

One of these registers is read/write, and the other register is read only. The DQRR consumption notification mode programmed by software determines which of the two registers is write/read and which one is read only (see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\)”\).](#)

If DQRR consumption notification mode is programmed for discrete consumption acknowledgment mode rather than one of the two consumer index write modes, then the QMan implicitly updates the DQRR\_CI value using the discrete consumption acknowledgments provided by software (see [Section 3.3.8.2.2, “DQRR Consumption Notification”](#)), and in this mode, both of the DQRR\_CI registers are read only.

### **3.2.3.9.1 QCSP DQRR Consumer Index Cache-Enabled Registers (QCSP*i*\_DQRR\_CI\_CENA)**

When accessed in the cache-enabled location, QCSP*i*\_DQRR\_CI\_CENA is located in word 0 (the word at address offset 0) of the cache line.

Offset 0x000\_3140 (QCSP< i>\_DQRR\_CI\_CENA)  
offset 0x10000  
range i=0..9

Access: R/W or R

**Figure 3-15. QCSP DQRR Consumer Index Cache-Enabled Registers (QCSP*i*\_DQRR\_CI\_CENA)**

**Table 3-14. QCSP*i*\_DQRR\_CI\_CENA Field Descriptions**

Field	Description
0-27	Reserved
28-31 CI	<p>Consumer index</p> <p>This field indicates the ring entry that is next to be consumed by software. When the difference between DQRR_PI and DQRR_CI is greater than or equal to the maximum number of entries that the ring can hold, which for DQRR is programmed in the DQRR_MF field of the configuration register of each software portal (see <a href="#">Section 3.2.3.17, “QMan Software Portal Configuration Register (QCSP<i>i</i>_CFG)</a>”), then the ring is full and the QMan does not add any new entries to the ring until software has consumed one or more entries and advanced CI.</p> <p>This field may be read/write or read only depending on the programmed DQRR consumption notification mode.</p> <ul style="list-style-type: none"> <li>If QCSP<i>i</i>_CFG[DCM] is 0: CI is R/W in DQRR_CI_CINH and read only in DQRR_CI_CENA.</li> <li>If QCSP<i>i</i>_CFG[DCM] is 1: CI is R/W in DQRR_CI_CENA and read only in DQRR_CI_CINH.</li> <li>If QCSP<i>i</i>_CFG[DCM] is 2 or 3: CI is read only in both DQRR_CI_CINH and DQRR_CI_CENA.</li> </ul>

### 3.2.3.9.2 QCSP DQRR Consumer Index Cache-Inhibited Registers (QCSP*i*\_DQRR\_CI\_CINH)

Offset 0x400_3140 (QCSP<i>_DQRR_CI_CINH)	Access: R/W or R																																																																																																																																							
offset 0x10000																																																																																																																																								
range i=0..9																																																																																																																																								
R	<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td> </tr> <tr> <td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> <tr> <td>W</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	W																																		Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																																																																																																									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																							
W																																																																																																																																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																						

**Figure 3-16. QCSP DQRR Consumer Index Cache-Inhibited Registers (QCSP*i*\_DQRR\_CI\_CINH)****Table 3-15. QCSP*i*\_DQRR\_CI\_CINH Field Descriptions**

Field	Description
0-27	Reserved
28-31 CI	<p>Consumer index</p> <p>This field indicates the ring entry that is next to be consumed by software. When the difference between DQRR_PI and DQRR_CI is greater than or equal to the maximum number of entries that the ring can hold, which for DQRR is programmed in the DQRR_MF field of the configuration register of each software portal (see <a href="#">Section 3.2.3.17, “QMan Software Portal Configuration Register (QCSP<i>i</i>_CFG)</a>”), then the ring is full and the QMan does not add any new entries to the ring until software has consumed one or more entries and advanced CI.</p> <p>This field may be read/write or read only depending on the programmed DQRR consumption notification mode.</p> <ul style="list-style-type: none"> <li>If QCSP<i>i</i>_CFG[DCM] is 0: CI is R/W in DQRR_CI_CINH and read only in DQRR_CI_CENA.</li> <li>If QCSP<i>i</i>_CFG[DCM] is 1: CI is R/W in DQRR_CI_CENA and read only in DQRR_CI_CINH.</li> <li>If QCSP<i>i</i>_CFG[DCM] is 2 or 3: CI is read only in both DQRR_CI_CINH and DQRR_CI_CENA.</li> </ul>

### 3.2.3.10 QCSP DQRR Discrete Consumption Acknowledgment and Park (QCSP*i*\_DQRR\_DCAP)

If the DQRR consumption notification mode is programmed for discrete consumption acknowledgment mode rather than one of the two consumer index modes, then the QMan software portal DQRR discrete consumption acknowledgment and park register (QCSP*i*\_DQRR\_DCAP), shown in [Figure 3-17](#), can be

used by software to issue discrete consumption acknowledgments to QMan to indicate that it has consumed one or more entries from the DQRR; see [Section 3.3.8.2.2, “DQRR Consumption Notification.”](#)

When a write to `QCSPi_DQRR_DCAP` is received by the QMan, one or more DQRR entries are marked as consumed within the QMan, and the `DQRR_CI` value is implicitly advanced as a result of these consumption acknowledgments.

If the DQRR consumption notification mode is programmed by software to use one of the two consumer index modes, then the `QCSPi_DQRR_DCAP` register is not used for consumption notifications, and writes to it have no effect on the `DQRR_CI` value.

In any of the DQRR consumption notification modes, a write to `QCSPi_DQRR_DCAP` can be used to request that a FQ that is in the Held Active or Held Suspended states be parked (rather than rescheduled) when it is released from the portal (that is, when the last DQRR entry associated with that held FQ is consumed). The QMan maintains a park request bit for each held FQ (up to four are possible), which is asserted when a park request is received in a write to `DQRR_DCAP` (a write with `PK = 1`) in which `DCAP_CI` indexes a DQRR entry associated (that is, containing a frame dequeued from) with that held FQ. When a held FQ is released, the park request bit for the held FQ is used to determine whether the released FQ is parked or rescheduled. Therefore, to successfully park a held FQ, it is necessary (and sufficient) for software to issue a park request on any one (or more) DQRR entry/entries containing a frame dequeued from that FQ. When a FQ on which a park request has been issued ultimately reaches the parked state, a FQPN message is produced in the MR of the software portal (see [Section 3.3.9.4, “FQ State Change Notification Message Response”](#)) to notify software that the FQ is now in the parked state.

`DQRR_DCAP` is contained in a 32-bit cache-inhibited, write-only register.

Offset 0x400_31C0 ( <code>QCSP&lt;i&gt;_DQRR_DCAP</code> )																															Access: W
offset 0x10000																															
range i=0..9																															
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
W	CI_VECTOR																S				PK				DCAP_CI						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**Figure 3-17. DQRR Discrete Consumption Acknowledgment Register (`QCSPi_DQRR_DCAP`)**

**Table 3-16. `QCSPi_DQRR_DCAP` Field Descriptions**

Field	Description
0-15 CI_VECTOR	Consumer index vector Valid only if <code>QCSP<i>i</i>_DQRR_DCAP[S] = 1</code> . Valid only if DQRR consumption notification mode = discrete consumption acknowledgment mode. This field contains a bitwise vector indicating each of the DQRR entries being consumed by this write to <code>DQRR_DCAP</code> . Use of this field allows one or more DQRR entries to be consumed with a single write to <code>DQRR_DCAP</code> . Bits 0–15 = discrete consumption acknowledgment for DQRR entries 0–15. For each bit: 0 This entry is not consumed. 1 This entry is consumed.
16-22	Reserved

**Table 3-16. QCSP*i*\_DQRR\_DCAP Field Descriptions (continued)**

<b>Field</b>	<b>Description</b>
23 S	Select 0 The DCAP_CI field is used to acknowledge consumption of a single DQRR entry at a time; CI_VECTOR is ignored. 1 The CI_VECTOR field is used to acknowledge consumption of one or more DQRR entries at a time; DCAP_CI and PK are ignored.
24	Reserved
25 PK	Park Valid only if QCSP <i>i</i> _DQRR_DCAP[S] = 0. Valid for any of the DQRR consumption notification modes. This bit affects only FQ that are currently in the Held Active or Held Suspended state in the DQRR. 0 Do not park the held FQ associated with the DQRR entry indexed by DCAP_CI. When the last DQRR entry associated with that held FQ is consumed, the FQ is released and rescheduled. 1 Park the held FQ associated with the DQRR entry indexed by DCAP_CI. When the last DQRR entry associated with that held FQ is consumed, the FQ is released and parked rather than being rescheduled.
26-27	Reserved
28-31 DCAP_CI	Discrete consumption acknowledgment and park consumer index Valid only if QCSP <i>i</i> _DQRR_DCAP[S] = 0 <ul style="list-style-type: none"> <li>• If DQRR consumption notification mode = discrete consumption acknowledgment mode: This field indicates the ring entry that is being consumed, and to which the PK bit applies, for this write to DQRR_DCAP.</li> <li>• If DQRR consumption notification mode = one of the two consumer index modes: This field indicates the ring entry to which the PK bit applies for this write to DQRR_DCAP.</li> </ul>

### 3.2.3.11 QCSP DQRR Static Dequeue Command Register (SDQCR)

When the DQRR is configured in Push mode, this register is used to issue commands for scheduled dequeues, that is, from one or more channels or from a specific WQ.

The source of dequeue operations is selected by the SS bit in this register. SS = 0 indicates a channel based dequeue, in which case a WQ for dequeue is selected from among the WQ channels associated with this portal. SS = 1 indicates a specific WQ dequeue, in which case QMan's selection algorithms are bypassed and the WQ for dequeue is specified directly in the command.

A channel based dequeue (SS = 0) command written to this register is persistent, therefore as long as a non-null channel based dequeue command is present in this register, and as long as there is room in the DQRR to accept a response, QMan will continually repeat the execution of this command and place the resulting frame dequeue responses in the DQRR.

A specific WQ dequeue (SS = 1) command written to this register operates as a one-shot command, when SDQCR is written with SS = 1 and DCT non-zero, QMan executes the requested dequeue exactly once, delivering from 0 to 3 frames from the head of the specified WQ.

QMan internally keeps two copies of SDQCR, one to store a channel based dequeue command and the other to store a specific WQ dequeue command. So, when a specific WQ dequeue command is written, the current channel based dequeue command remains saved internally. QMan will execute the one shot specific WQ dequeue command, then automatically revert to executing the channel based dequeue command. This means software does not have to read, save, and restore the current channel based dequeue

## Queue Manager (QMan)

command, the one shot specific WQ dequeue command can be written at any time, and QMan will execute it exactly once and then revert to the channel based dequeue command automatically.

When SDQCR is read, the contents of the channel based dequeue command are always returned, even if a specific WQ command has been written and is pending. Specific WQ dequeue commands are always write-only, they cannot be read.

The command in SDQCR can be changed on the fly, and the token can be used in the response to associate frame dequeue responses with changes in the command written to the SDQCR, and this for both channel based and specific WQ dequeue commands. Dequeue operations can be halted or paused by software by either writing a null channel based dequeue command to SDQCR or by not advancing DQRR\_CI. If a new command is written to SDQCR when the DQRR is stopped (software has let it become full), then the new command will take effect immediately when enough entries in DQRR are consumed to allow the new command to be dispatched.

SDQCR is contained in a 32 bit cache-inhibited register.

**Figure 3-18. QCSP DQRR Static Dequeue Command Register (QCSP\_I\_DQRR\_SDQCR)**

**Table 3-17. QCSP*i*\_DQRR\_SDQCR Field Descriptions**

Field	Description
0	Reserved
1 SS	<p>Source Select. Along with DQ_SRC, specifies the source of dequeue operations.</p> <p>0 = Dequeue from the WQ channels associated with this portal.            The DQ_SRC field specifies which of the 16 available channels should participate in the dequeue selection.</p> <p>1 = Dequeue from a specific work queue (WQ) in the channels associated with this portal.            The WQ is specified in the DQ_SRC field.</p>
2 FC	<p>Dequeue Command Frame Count.</p> <p>Indicates the maximum number of frames the requester is willing to accept in response to each repeated execution of this dequeue command.</p> <p>0 = Dequeue at most one frame in response to this dequeue command.            QMan will repeat execution of the dequeue command as long as there is at least 1 free entry in the DQRR.</p> <p>1 = Dequeue up to 3 frames in response to this dequeue command.            QMan will repeat execution of the dequeue command as long as there are at least 3 free entries in the DQRR. If fewer than 3 free entries are available in DQRR, dispatch of the command is withheld until software consumes some ring entries. This mode enables QMan to use memory bandwidth more efficiently when performing dequeue operations.</p>

**Table 3-17. QCSPi\_DQRR\_SDQCR Field Descriptions (continued)**

Field	Description
3 DP	Dedicated Channel Precedence. Only valid if SS = 0. 0 = The dedicated channel in this portal does not have precedence over the pool channels, it is included with the pool channels in the round robin selection in the WQ channel scheduler. 1 = The dedicated channel in this portal has precedence over the pool channels, it does not participate in the round robin selection with the pool channels, and it will be selected whenever its advertised service priority is equal to or higher than that of all pool channels selected in the dequeue command.
4-5	Reserved
6-7 DCT	Dequeue Command Type. See <a href="#">Section 3.3.8.2.6, “Dequeue Dispatcher Operation—Dequeueing from One or More Channels”</a> and <a href="#">Section 3.3.8.2.7, “Dequeue Dispatcher Operation—Dequeueing from a Specific WQ”</a> for a detailed description of these command types and how they operate. 0 = Null Command. No dequeue operations are executed. 1 = Dequeue with priority precedence, and Intra-WQ class scheduling respected. 2 = Dequeue with active FQ precedence, and Intra-WQ class scheduling respected. 3 = Dequeue with active FQ precedence, and override Intra-WQ class scheduling.
8-15 TOKEN	Dequeue Command Token Opaque token returned in the DQRR entry of all dequeued frames that resulted from this command.
16-31 DQ_SRC	Dequeue Source.  If SS = 0, DQ_SRC specifies which of the 16 available channels should participate in the dequeue selection. Each bit will disable (0) or enable (1) the corresponding channel for participation in the dequeue selection. Bit 16: enable/disable dequeue selection on the WQ channel dedicated to this portal. Bit 17: enable/disable dequeue selection on Pool Channel 1. ... Bit 31: enable/disable dequeue selection on Pool Channel 15. It is an error to issue a dequeue command in SDQCR with SS = 0 and with all bits of DQ_SRC that correspond to valid channels set to 0. For example, in an SoC with only 3 pool channels, one or more of bits 16 to 19 must be 1. In an SoC in which all 15 pool channels are implemented, all bits of DQ_SRC are valid and only DQ_SRC = 0 is considered an error. If the software portal is idle and does not contain an active FQ, then such an erroneous command will never be executed and no dequeues will occur. If the portal contains an active FQ when this erroneous command is written to SDQCR, then an error will be indicated in QMAN_ERR_ISR[IDS1], and no dequeue will occur.  If SS = 1, DQ_SRC specifies the WQ from which to dequeue. Bit 16-23: not used. Bits 24-27: Identifies one of the 16 available channels in a software portal, decoded as follows: 0 = dequeue from the WQ channel dedicated to this portal. 1 = dequeue from Pool Channel 1. ... 15 = dequeue from Pool Channel 15. Bit 28: not used. Bits 29-31: Identifies the WQ (0-7) within the channel selected by bits 24-27. An error will be indicated in QMAN_ERR_ISR[IDS1] (and no dequeue will occur) It is an error to issue a dequeue command in SDQCR with SS = 1 and a non-existent pool channel specified in bits 24-27 of the command, for SoC in which fewer than 15 pool channels are implemented. If the software portal is idle and does not contain an active FQ, then such an erroneous command will never be executed and no dequeues will occur. If the portal contains an active FQ when this erroneous command is written to SDQCR, then an error will be indicated in QMAN_ERR_ISR[IDS1], and no dequeue will occur.

### **3.2.3.12 QCSP DQRR Volatile Dequeue Command Register (QCSP*i*\_DQRR\_VDQCR)**

When the DQRR is configured in Push mode, the QMan software portal DQRR volatile dequeue command register (`QCSPi_DQRR_VDQCR`), shown in [Figure 3-19](#), is used to issue commands for unscheduled dequeues (that is, from a specific parked or retired FQ).

There are two possible options when a non-null FQID is present in `QCSPi_DQRR_VDQCR`, as follows:

- $\text{QCSPi\_DQRR\_VDQCR[P]} = 0$ —the QMan executes the command in VDQCR rather than the one in SDQCR. Additionally, execution from VDQCR takes precedence over SDQCR, and continues until NUM\_FRAMES is met or until the specified FQ becomes empty. When the VDQCR command reaches its end, the QMan writes  $\text{QCSPi\_DQRR\_VDQCR[FQID]} = 0$  and reverts to executing dequeues using the command in SDQCR (if it is non-null).
  - $\text{QCSPi\_DQRR\_VDQCR[P]} = 1$ —the QMan executes the command in VDQCR on a best effort basis. A dequeue using  $\text{QCSPi\_DQRR\_VDQCR}$  occurs each time there is room in the DQRR and there are no scheduled dequeues to be done in the channels or in the WQ specified in SDQCR. Such dequeues from  $\text{QCSPi\_DQRR\_VDQCR}$  continue until NUM\_FRAMES is met or until the specified FQ becomes empty. When the VDQCR command reaches its end, the QMan writes  $\text{QCSPi\_DQRR\_VDQCR[FQID]} = 0$  and reverts to executing dequeues from SDQCR only (if it is non-null).

Software can stop the dequeue from this register at any time by writing `QCSPi_DQRR_VDQCR[FQID] = 0`. If an error is detected in the VDQCR dequeue command, the command is terminated (FQID cleared to 0), an error interrupt is raised, and a null response (DQRR entry with 0 frames delivered (bit 3 of STAT field = 0) and FQ Empty = 0 (bit 0 of STAT field)) is produced in the DQRR. Possible errors are as follows:

- Attempted dequeue from a FQ that is not in either the Parked or Retired state.
  - Access to FQD is disabled in FQD\_AR[EN].
  - The FQID specified is outside of the valid range of FQID programmed in FQD\_AR[SIZE].

VDQCR is contained in a 32-bit cache-inhibited register.

**Figure 3-19. QCSO DQRR Volatile Dequeue Command Register (QCSPi DQRR VDQCR)**

**Table 3-18. QCSP*i*\_DQRR\_VDQCR Field Descriptions**

Field	Description
1 P	Precedence 0 VDQCR has precedence over SDQCR. 1 SDQCR has precedence over VDQCR.
2 E	Exact This bit is ignored if NUM_FRAMES is zero. 1 NUM_FRAMES must be met exactly.
	0 NUM_FRAMES may be exceeded by 0, 1, or 2 for efficiency (to prevent leaving a partial PFDR). Note that there must be at least 3 free entries in the DQRR before a VDQCR command is dispatched; therefore, the value of DQRR_MF configured for this portal must be 3 or greater.
2-7 NUM_FRAMES	Number of frames that are dequeued from the specified FQ <ul style="list-style-type: none"> <li>If NUM_FRAMES is non-zero—specifies the maximum number of frames to be dequeued before the volatile command is terminated; range is 1 to 63 frames.</li> <li>If NUM_FRAMES is zero—indicates that the volatile command is not terminate until the specified FQ becomes empty.</li> </ul>
8-31 FQID	Frame queue ID Specifies the FQID from which to dequeue. If 0, indicates a null command, no dequeues are performed. Note that FQID 0 is reserved, no unscheduled dequeues can be performed from this FQ.

### 3.2.3.13 QCSP DQRR Pull Dequeue Command Register (QCSP*i*\_DQRR\_PDQCR)

When the DQRR is configured in Pull mode, the QMan software portal DQRR pull dequeue command register (QCSPi\_DQRR\_PDQCR), shown in [Figure 3-20](#), is used to issue commands for both scheduled and unscheduled dequeues—that is, from one or more channels or from a specific WQ (scheduled), or from a specific parked or retired FQ (unscheduled).

In Pull mode, the QMan executes each dequeue command written to QCSP*i*\_DQRR\_PDQCR exactly once, and software must write a new command to QCSP*i*\_DQRR\_PDQCR to initiate each subsequent dequeue operation.

If an error is detected in the PDQCR dequeue command, the command is ignored without any dequeues occurring and an error interrupt is raised. Possible errors are when an unscheduled dequeue command is issued, and they are the same as those that apply to the FQID in the QCSP*i*\_DQRR\_VDQCR when the DQRR is configured in Push mode.

PDQCR is contained in a 32 bit cache-inhibited register.

**Figure 3-20. QCSP DQRR Pull Dequeue Command Register (QCSP*i*\_DQRR\_PDQCR)**

**Table 3-19. QCSP*i*\_DQRR\_PDQCR Field Descriptions**

<b>Field</b>	<b>Description</b>
0 SU	Scheduled/Unscheduled 0 Perform a scheduled dequeue, from one or more channels or from a specific WQ. 1 Perform an unscheduled dequeue, from a specific parked or retired FQ.
1 SS	Source select Valid only if QCSP <i>i</i> _DQRR_PDQCR[SU] = 0. Along with QCSP <i>i</i> _DQRR_PDQCR[DQ_SRC], specifies the source of dequeue operations. 0 Dequeue from the WQ channels associated with this portal. QCSP <i>i</i> _DQRR_PDQCR[DQ_SRC] specifies which of the 16 available channels should participate in the dequeue selection. 1 Dequeue from a specific work queue (WQ) in the channels associated with this portal. The WQ is specified in QCSP <i>i</i> _DQRR_PDQCR[DQ_SRC].
2 FC	Dequeue command frame count Valid for both QCSP <i>i</i> _DQRR_PDQCR[SU] = 0 or 1. See the QCSP <i>i</i> _DQRR_SDQCR[FC] in <a href="#">Section 3.2.3.11, “QCSP DQRR Static Dequeue Command Register (SDQCR),”</a> for a description of this field.
3 DP	Dedicated channel precedence Only valid if QCSP <i>i</i> _DQRR_PDQCR[SU]= 0 and QCSP <i>i</i> _DQRR_PDQCR[SS] = 0. See QCSP <i>i</i> _DQRR_SDQCR[DP] in <a href="#">Section 3.2.3.11, “QCSP DQRR Static Dequeue Command Register (SDQCR),”</a> for a description of this field.
4-5	Reserved
6-7 DCT	Dequeue command type Valid only if QCSP <i>i</i> _DQRR_PDQCR[SU] = 0. See QCSP <i>i</i> _DQRR_SDQCR[DCT] in <a href="#">Section 3.2.3.11, “QCSP DQRR Static Dequeue Command Register (SDQCR),”</a> for a description of this field.
8-31 DQ_SRC	Dequeue source Valid only if QCSP <i>i</i> _DQRR_PDQCR[SU] = 0. Bits 8-15: Not used Bits 16-31: Specifies the DQ_SRC. See QCSP <i>i</i> _DQRR_SDQCR[DQ_SRC] in <a href="#">Section 3.2.3.11, “QCSP DQRR Static Dequeue Command Register (SDQCR),”</a> for a description of this field. An error is indicated in QMAN_ERR_ISR[IDS1] if a dequeue command is issued with an invalid DQ_SRC field.
8-31 FQID	Frame queue ID. Valid only if QCSP <i>i</i> _DQRR_PDQCR[SU] = 1. Specifies the FQID from which to dequeue. If 0, indicates a null command, no dequeues are performed. Note that FQID 0 is reserved, no unscheduled dequeues can be performed from this FQ.

### 3.2.3.14 QCSP MR Producer Index Registers

The MR\_PI value is used by QMan to notify software that one or more valid entries have been written into the MR. Software may either read this value, or use the alternating polarity valid bit in each entry to determine when an entry in the ring is valid and ready to be consumed.

MR\_PI is a single read-only value stored in QMan, and this single value is available as one of the following separate locations:

- Separate cache-enabled register (QCSP*i*\_MR\_PI\_CENA)
- Combined with the other read-only indices in a single cache-enabled register (QCSP*i*\_RORI\_CENA)

- Cache-inhibited register (QCSP*i*\_MR\_PI\_CINH)

When accessed in the cache-enabled MR\_PI\_CENA location, the register is located in word 0 (the word at address offset 0) of the cache line.

### 3.2.3.14.1 QCSP MR Producer Index Cache-Enabled Registers (QCSP*i*\_MR\_PI\_CENA)

Offset 0x000_3300 (QCSP <i>&lt;i&gt;</i> _MR_PI_CENA)																														Access: R			
offset 0x10000																																	
range i=0..9																																	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VP	PI			
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0		

Figure 3-21. QCSP MR Producer Index Cache-Enabled Registers (QCSP*i*\_MR\_PI\_CENA)

Table 3-20. QCSP*i*\_MR\_PI\_CENA Field Descriptions

Field	Description																														
0-27	Reserved																														
28 VP	Current valid bit polarity at producer index This bit is 1 after reset. Indicates the polarity that the QMan is currently using in the alternating polarity valid bit in the MR entries. This bit toggles every time that the QMan's MR_PI value wraps from 7 back to 0. <ul style="list-style-type: none"> <li>If this bit is 1, QMan is writing a 1 to indicate a valid entry, and 0 indicates an invalid entry.</li> <li>If this bit is 0, QMan is writing a 0 to indicate a valid entry, and 1 indicates an invalid entry.</li> </ul> Reading this bit can be used as a debug or verification assist or to re-synchronize software's copy of this bit with the QMan's, if needed.																														
29-31 PI	Producer index This field indicates the ring entry in which the next valid response will be written. When PI and CI are equal, the ring is empty, and the QMan advances PI after adding a new entry to the ring.																														

### 3.2.3.14.2 QCSP MR Producer Index Cache-Inhibited Registers (QCSP*i*\_MR\_PI\_CINH)

Offset 0x400_3300 (QCSP <i>&lt;i&gt;</i> _MR_PI_CINH)																															Access: R	
offset 0x10000																																
range i=0..9																																
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VP	PI		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

Figure 3-22. QCSP MR Producer Index Cache-Inhibited Registers (QCSP*i*\_MR\_PI\_CINH)

**Table 3-21. QCSP*i*\_MR\_PI\_CINH Field Descriptions**

Field	Description
0-27	Reserved
28 VP	<p>Current valid bit polarity at producer index</p> <p>This bit is 1 after reset. Indicates the polarity that the QMan is currently using in the alternating polarity valid bit in the MR entries. This bit toggles every time that the QMan's MR_PI value wraps from 7 back to 0.</p> <ul style="list-style-type: none"> <li>• If this bit is 1, QMan is writing a 1 to indicate a valid entry, and 0 indicates an invalid entry.</li> <li>• If this bit is 0, QMan is writing a 0 to indicate a valid entry, and 1 indicates an invalid entry.</li> </ul> <p>Reading this bit can be used as a debug or verification assist or to re-synchronize software's copy of this bit with the QMan's, if needed.</p>
29-31 PI	<p>Producer index</p> <p>This field indicates the ring entry in which the next valid response will be written. When PI and CI are equal, the ring is empty, and the QMan advances PI after adding a new entry to the ring.</p>

### **3.2.3.15 QCSP MR Consumer Index Registers**

The MR\_CI value is used by software to notify the QMan that one or more entries have been consumed from the MR. MR\_CI is a single value stored in the QMan, and this single value is available in one of the following separate locations:

- Cache-enabled register (QCSPi\_MR\_CI\_CENA)
  - Cache-inhibited register (QCSPi\_MR\_CI\_CINH)

One of these registers is read/write, and the other register is read only. The MR consumption notification mode programmed by software determines which of the two registers is write/read and which one is read only; see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\).”](#)

### **3.2.3.15.1 QCSP MR Consumer Index Cache-Enabled Registers (QCSP<sub>i</sub>\_MR\_CI\_CENA)**

When accessed in the cache-enabled QCSP*i*\_MR\_CI\_CENA location, the register is located in word 0 (the word at address offset 0) of the cache line.

Offset 0x000\_3340 (QCSP<i>\_MR\_CI\_CENA)  
offset 0x10000  
range i=0..9

Access: R/W or R

**Figure 3-23. QCSP MR Consumer Index Cache-Enabled Registers (QCSP*i*\_MR\_CI\_CENA)**

**Table 3-22. QCSP*i*\_MR\_CI\_CENA Field Descriptions**

Field	Description
0-28	Reserved
29-31 CI	<p>Consumer index</p> <p>This field indicates the ring entry that is next to be consumed by software. When the difference between MR_PI and MR_CI is greater than or equal to the maximum number of entries that the ring can hold, which for MR is 7, then the ring is full and the QMan does not add any new entries to the ring until software has consumed one or more entries and advanced CI.</p> <p>This field may be R/W or read only depending on the programmed MR consumption notification mode:</p> <ul style="list-style-type: none"> <li>• If MR consumption notification mode is 0: CI is R/W in QCSP<i>i</i>_MR_CI_CINH and read only in QCSP<i>i</i>_MR_CI_CENA.</li> <li>• If MR consumption notification mode is 1: CI is R/W in QCSP<i>i</i>_MR_CI_CENA and read only in QCSP<i>i</i>_MR_CI_CINH.</li> </ul>

### 3.2.3.15.2 QCSP MR Consumer Index Cache-Inhibited Registers (QCSP*i*\_MR\_CI\_CINH)

Offset 0x400\_3340 (QCSP&lt;i&gt;\_MR\_CI\_CINH)

Access: R/W or R

offset 0x10000  
range i=0..9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CI	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**Figure 3-24. QCSP MR Consumer Index Cache-Inhibited Registers (QCSP*i*\_MR\_CI\_CINH)****Table 3-23. QCSP*i*\_MR\_CI\_CINH Field Descriptions**

Field	Description
0-28	Reserved
29-31 CI	<p>Consumer index</p> <p>This field indicates the ring entry that is next to be consumed by software. When the difference between MR_PI and MR_CI is greater than or equal to the maximum number of entries that the ring can hold, which for MR is 7, then the ring is full and the QMan does not add any new entries to the ring until software has consumed one or more entries and advanced CI.</p> <p>This field may be R/W or read only depending on the programmed MR consumption notification mode:</p> <ul style="list-style-type: none"> <li>• If MR consumption notification mode is 0: CI is R/W in QCSP<i>i</i>_MR_CI_CINH and read only in QCSP<i>i</i>_MR_CI_CENA.</li> <li>• If MR consumption notification mode is 1: CI is R/W in QCSP<i>i</i>_MR_CI_CENA and read only in QCSP<i>i</i>_MR_CI_CINH.</li> </ul>

### 3.2.3.16 QCSP Read-Only Ring Indices Cache-Enabled Registers (QCSP*i*\_RORI\_CENA)

The QMan software portal read-only ring indices cache-enabled registers (QCSP*i*\_RORI\_CENA) carry a read-only copy of the producer and consumer indices of the three rings in the software portal. Each of the indices carried in these registers are available in one of the following separate locations:

- Three, separate cache-enabled registers
- Single, cache-enabled register with all indices combined (QCSP*i*\_RORI\_CENA)
- Three, separate cache-inhibited registers

## Queue Manager (QMan)

Placing these indices together in these registers allows all indices to be read with a single bus transaction for debug purposes.

Offset 0x000_3400 (QCSP<i>_RORI_CENA)	Access: R
offset 0x10000	
range i=0..9	
R	0 1 2 3   4 5 6 7   8 9 10 11   12 13 14 15 EQCR_CI_CENA   DQRR_PI_CENA   MR_PI_CENA   0 0 0 0
W	
Reset	0x0000_8808_0000_0010_0000_0008_0000_0000
R	16 17 18 19   20 21 22 23   24 25 26 27   28 29 30 31 EQCR_PI_CENA   DQRR_CI_CENA   MR_CI_CENA   0 0 0 0
W	
Reset	0x0000_0008_0000_0000_0000_0000_0000_0000
R	32 33 34 35   36 37 38 39   40 41 42 43   44 45 46 47 0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0
W	
Reset	0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0
R	48 49 50 51   52 53 54 55   56 57 58 59   60 61 62 63 0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0
W	
Reset	0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0

**Figure 3-25. QCSP Read-Only Ring Indices Cache-Enabled Registers (QCSP*i*\_RORI\_CENA)**

**Table 3-24. QCSP*i*\_RORI\_CENA Field Descriptions**

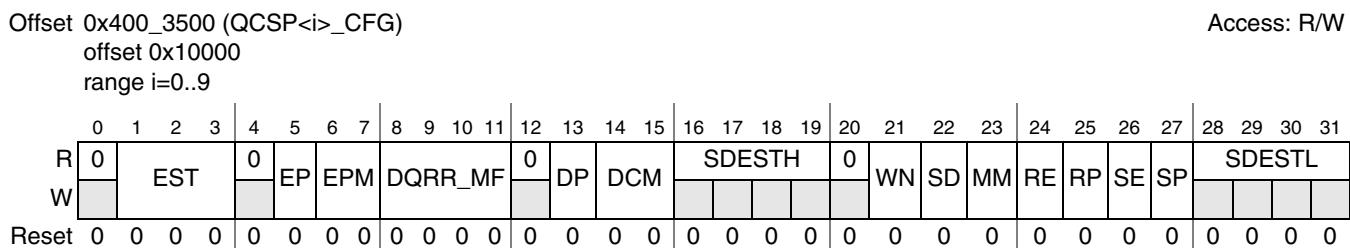
Field	Description
0-3 EQCR_CI_CENA	EQCR consumer index, cache-enabled The contents of this word is identical to that of the cache-inhibited register where this value is available; see <a href="#">Section 3.2.3.7.1, “QCSP EQCR Consumer Index Cache-Enabled Registers (QCSP<i>i</i>_EQCR_CI_CENA).</a>
4-7 DQRR_PI_CENA	DQRR producer index, cache-enabled The contents of this word is identical to that of the cache-inhibited register where this value is available; see <a href="#">Section 3.2.3.8.1, “QCSP DQRR Producer Index Cache-Enabled Registers (QCSP<i>i</i>_DQRR_PI_CENA).</a>
8-11 MR_PI_CENA	MR producer index, cache-enabled The contents of this word is identical to that of the cache-inhibited register where this value is available; see <a href="#">Section 3.2.3.14.1, “QCSP MR Producer Index Cache-Enabled Registers (QCSP<i>i</i>_MR_PI_CENA).</a>
12-15	Reserved
16-19 EQCR_PI_CENA	EQCR producer index, cache-enabled The contents of this word is identical to that of the cache-inhibited register where this value is available; see <a href="#">Section 3.2.3.6.1, “QCSP EQCR Producer Index Cache-Enabled Registers (QCSP<i>i</i>_EQCR_PI_CENA).</a>
20-23 DQRR_CI_CENA	DQRR consumer index, cache-enabled The contents of this word is identical to that of the cache-inhibited register where this value is available; see <a href="#">Section 3.2.3.9.1, “QCSP DQRR Consumer Index Cache-Enabled Registers (QCSP<i>i</i>_DQRR_CI_CENA).</a>

**Table 3-24. QCSP*i*\_RORI\_CENA Field Descriptions (continued)**

Field	Description
24-27 MR_CI_CENA	MR consumer index, cache-enabled The contents of this word is identical to that of the cache-inhibited register where this value is available; see <a href="#">Section 3.2.3.15.1, “QCSP MR Consumer Index Cache-Enabled Registers (QCSP<i>i</i>_MR_CI_CENA)</a> .
28-63	Reserved

### 3.2.3.17 QMan Software Portal Configuration Register (QCSP*i*\_CFG)

The QMan software portal configuration register (QCSP*i*\_CFG) should be mapped into a cache-inhibited page within each software portal.

**Figure 3-26. QCSP Configuration Register (QCSP*i*\_CFG)****Table 3-25. QCSP*i*\_CFG Field Descriptions**

Field	Description
0	Reserved
1-3 EST	EQCR_CI stashing threshold 0 EQCR_CI stashing is disabled. 1-7 EQCR_CI stashing is enabled; threshold setting 1 to 7 The QMan stashes the new QCSP <i>i</i> _EQCR_CI_CENA value for this portal (see <a href="#">Section 3.2.3.7.1, “QCSP EQCR Consumer Index Cache-Enabled Registers (QCSP<i>i</i>_EQCR_CI_CENA)</a> ) into a processor core’s cache after the EQCR_CI value (QCSP <i>i</i> _EQCR_CI_CENA[CI]) is incremented, if this field is non-zero and if the current EQCR_CI value minus the last stashed EQCR_CI value is greater than or equal to the configured EST value. A setting of EST = 1 results in a new stash transaction for every EQCR entry consumed by QMan, higher values of EST result in a stash transaction for every n EQCR entries consumed, n = 2 to 7. See <a href="#">Section 3.3.8.7, “EQCR_CI Stashing”</a>
4	Reserved
5 EP	EQCR_CI stashing priority Valid only if EST is non-zero. 0 EQCR_CI stash transactions for this software portal is signaled with lower priority. 1 EQCR_CI stash transactions for this software portal is signaled with higher priority.

**Table 3-25. QCSP*i*\_CFG Field Descriptions (continued)**

<b>Field</b>	<b>Description</b>
6-7 EPM	<p>EQCR production notification mode</p> <p>00 PI write mode, cache-inhibited. EQCR production notifications are issued by writing a producer index (PI) value in QCSP<i>i</i>_EQCR_PI_CINH; see <a href="#">Section 3.2.3.6.2, “QCSP EQCR Producer Index Cache-Inhibited Registers (QCSP<i>i</i>_EQCR_PI_CINH)</a>.</p> <p>01 PI write mode, cache-enabled EQCR production notifications are issued by writing a producer index (PI) value in QCSP<i>i</i>_EQCR_PI_CENA; see <a href="#">Section 3.2.3.6.1, “QCSP EQCR Producer Index Cache-Enabled Registers (QCSP<i>i</i>_EQCR_PI_CENA)</a>.</p> <p>10 or 11 Valid bit mode EQCR production notifications are derived implicitly by the QMan using the alternating polarity valid bit in each EQCR entry. See <a href="#">Section 3.3.8.1, “Enqueue Command Ring (EQCR)</a>.</p>
8-11 DQRR_MF	<p>DQRR max fill</p> <p>This field configures the maximum number of valid entries that the QMan populates in the DQRR at any time. A value of 0, the default (reset) value, effectively disables the DQRR, whereas a value of 15 allows up to 15 entries in the ring.</p> <p>Internally, the QMan computes its DQRR full indication as DQRR_PI – DQRR_CI &gt;= DQRR_MF, and the QMan adds entries to the ring as long as it is not full.</p>
12	Reserved
13 DP	<p>DQRR push/pull mode</p> <p>0 Push mode In push mode, QCSP<i>i</i>_DQRR_SDQCR and QCSP<i>i</i>_DQRR_VDQCR (see <a href="#">Section 3.2.3.11, “QCSP DQRR Static Dequeue Command Register (SDQCR)</a>, and <a href="#">Section 3.2.3.12, “QCSP DQRR Volatile Dequeue Command Register (QCSP<i>i</i>_DQRR_VDQCR)</a>) provide the dequeue commands that drive the responses returned in the DQRR.</p> <p>1 Pull mode In pull mode, the QCSP<i>i</i>_DQRR_PDQCR (see <a href="#">Section 3.2.3.13, “QCSP DQRR Pull Dequeue Command Register (QCSP<i>i</i>_DQRR_PDQCR)</a>) provides the dequeue commands that drive the responses returned in the DQRR.</p>
14-15 DCM	<p>DQRR consumption notification mode</p> <p>00 CI write mode, cache-inhibited DQRR consumption notifications are issued by writing a Consumer Index (CI) value in the cache-inhibited DQRR_CI_CINH register; see <a href="#">Section 3.2.3.9.2, “QCSP DQRR Consumer Index Cache-Inhibited Registers (QCSP<i>i</i>_DQRR_CI_CINH)</a>.</p> <p>01 CI write mode, cache-enabled DQRR consumption notifications are issued by writing a Consumer Index (CI) value in the cache-enabled DQRR_CI_CENA register; see <a href="#">Section 3.2.3.9.2, “QCSP DQRR Consumer Index Cache-Inhibited Registers (QCSP<i>i</i>_DQRR_CI_CNH)</a>.</p> <p>10 or 11 Discrete consumption acknowledgment (DCA) mode This mode allows software to issue discrete (rather than cumulative) consumption notifications to the QMan, via QCSP<i>i</i>_DQRR_DCAP and/or by adding consumption acknowledgments to enqueue commands issued in the EQCR. See <a href="#">Section 3.3.8.2, “Dequeue Response Ring (DQRR)</a>.</p>
20	Reserved

**Table 3-25. QCSP*i*\_CFG Field Descriptions (continued)**

<b>Field</b>	<b>Description</b>
21 WN	Writes Non-cacheable The EQCR and CR are 64 byte structures writeable by software which reside in both the cache-enabled and cache-inhibited areas of the portal. This bit specifies the area of the software portal memory map (see <a href="#">Section 3.2.1, “QMan Software Portals (QCSP) Memory Map”</a> ) in which writes to these structures are enabled. 0 EQCR and CR writes are enabled in the cache-enabled portion of the memory map. Writes to the cache-inhibited portion will be ignored. 1 EQCR and CR writes are enabled in the cache-inhibited portion of the memory map. Writes to the cache-enabled portion will be ignored.
22 SD	Dequeued frame data, annotation, and FQ context stashing drop enable Valid only if QCSP <i>i</i> _CFG[SE] = 1. See <a href="#">Section 3.3.8.4, “Dropping of Dequeued Frame Data, Annotation, and FQ Context Stashes.”</a> 0 Dequeued frame data, annotation, and FQ context stash transactions for this software portal are never dropped by the QMan. If the target SRQ is full a sequencer will stall until each stash transaction can be completed. 1 Dequeued frame data, annotation, and FQ context stash transactions for this software portal are dropped by the QMan if the target SRQ is almost full, to prevent the QMan sequencer stalling. Stash transactions that are dropped result in a fetch from main memory when a core reads the addressed coherency granule. <b>Note:</b> Reserved.
23 MM	MR consumption notification mode 0 CI write mode, cache-inhibited MR consumption notifications are issued by writing a consumer index (CI) value in the cache-inhibited MR_CI_CINH register; see <a href="#">Section 3.2.3.15.2, “QCSP MR Consumer Index Cache-Inhibited Registers (QCSPi_MR_CI_CINH).”</a> 1 CI write mode, cache-enabled MR consumption notifications are issued by writing a consumer index (CI) value in the cache-enabled MR_CI_CENA register; see <a href="#">Section 3.2.3.15.1, “QCSP MR Consumer Index Cache-Enabled Registers (QCSPi_MR_CI_CENA).”</a>
24 RE	Dequeue response ring (DQRR) entry stashing enable 0 DQRR entries of this software portal are not stashed in a processor core's cache. 1 DQRR entries of this software portal are stashed in a processor core's cache. See <a href="#">Section 3.3.8.6, “DQRR Entry Stashing.”</a>
25 RP	Dequeue response ring (DQRR) entry stashing priority Valid only if QCSP <i>i</i> _CFG[RE] = 1. 0 DQRR entry stash transactions for this software portal are signaled with lower priority. 1 DQRR entry stash transactions for this software portal are signaled with higher priority.
26 SE	Dequeued frame data, annotation, and FQ context stashing enable 0 Frame data, annotations, and FQ context for frames dequeued via this portal are not stashed in a processor core's cache. 1 Frame data, annotations, and FQ context for frames dequeued via this portal may be stashed in a processor core's cache, under control of the Context_A field in the FQD of the FQ on which the frame arrived. See <a href="#">Section 3.3.8.8, “Dequeued Frame Data, Annotation, and Context Stashing.”</a> <b>Note:</b> Reserved.

**Table 3-25. QCSP*i*\_CFG Field Descriptions (continued)**

Field	Description
27-SP	<p>Dequeued frame data, annotation, and FQ context stashing priority Valid only if SE = 1.</p> <ul style="list-style-type: none"> <li>0 Dequeued frame data, annotation, and FQ context stash transactions for this software portal are signaled with lower priority.</li> <li>1 Dequeued frame data, annotation, and FQ context stash transactions for this software portal are signaled with higher priority.</li> </ul> <p><b>Note:</b> Reserved.</p>
16-19 SDESTH 28-31 SDESTL	<p>Stashing destination This is a read-only copy of the SDEST value programmed for this portal; see <a href="#">Section 3.2.4.2, “QCSP IO Configuration Registers (QCSPi_IO_CFG).”</a></p> <p>Bits 16-19 carry the msbits and 28-31 carry the lsbits of the SDEST value from QCSP_IO_CFG.</p>

### 3.2.3.18 QCSP EQCR Interrupt Threshold Register (QCSP*i*\_EQCR\_ITR)

The QMan software portal EQCR interrupt threshold register (QCSP*i*\_EQCR\_ITR) should be mapped into a cache-inhibited page within each software portal.

Offset 0x400\_3080 (QCSP<i>\_EQCR\_ITR)  
offset 0x10000  
range i=0..9

Access: R/W

**Figure 3-27. QCSP EQCR Interrupt Threshold Register (QCSP*i*\_EQCR\_ITR)**

**Table 3-26. QCSP*i*\_EQCR\_ITR Field Descriptions**

Field	Description
0-28	Reserved
29-31 EQCR_IT	EQCR interrupt threshold Sets the threshold for EQCR ring interrupt generation (ISR[EQRI]) The EQCR ring interrupt asserts when the ring contains fewer than EQCR_IT entries.

### 3.2.3.19 QCSP DQRR Interrupt Threshold Register (QCSP*i*\_DQRR\_ITR)

The QMan software portal DQRR interrupt threshold register (QCSP*i*\_DQRR\_ITR) should be mapped into a cache-inhibited page within each software portal.

Offset 0x400\_3180 (QCSP<i>\_DQRR\_ITR)  
offset 0x10000  
range i=0..9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DQRR_IT		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Figure 3-28. QCSP DQRR Interrupt Threshold Register (QCSP*i*\_DQRR\_ITR)

Table 3-27. QCSP*i*\_DQRR\_ITR Field Descriptions

Field	Description
0-27	Reserved
28-31 DQRR_IT	DQRR interrupt threshold Sets the threshold for DQRR interrupt generation (ISR[DQRI]). The DQRR interrupt asserts when the ring contains greater than DQRR_IT entries. The number of entries in the ring is calculated as DQRR_PI - DQRR_CI.

### 3.2.3.20 QCSP MR Interrupt Threshold Register (QCSP*i*\_MR\_ITR)

The QMan software portal MR interrupt threshold register (QCSP*i*\_MR\_ITR) should be mapped into a cache-inhibited page within each software portal.

Offset 0x400\_3380 (QCSP<i>\_MR\_ITR)  
offset 0x10000  
range i=0..9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MR_IT	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Figure 3-29. QCSP MR Interrupt Threshold Register (QCSP*i*\_MR\_ITR)

Table 3-28. QCSP*i*\_MR\_ITR Field Descriptions

Field	Description
0-28	Reserved
29-31 MR_IT	MR interrupt threshold Sets the threshold for MR interrupt generation (QCSP <i>i</i> _ISR[MRI]). The MR interrupt asserts when the ring contains greater than MR_IT entries.

### 3.2.3.21 QCSP Interrupt Status Register (QCSP*i*\_ISR)

The QMan software portal interrupt status registers (QCSP*i*\_ISR) should be mapped into a cache-inhibited page within each software portal.

These registers contain functional interrupts that pertain to software portals. A separate interrupt signal and QMAN\_ERR\_ISR are provided for reporting error conditions (see [Section 3.2.4.51, “QMan Error Interrupt Status Register \(QMAN\\_ERR\\_ISR\)”](#)).

Any asserted bit in this register can assert the portal’s interrupt line if enabled to do so in QCSP*i*\_IER of the portal. An asserted bit remains asserted until cleared by writing 1 to it.

Offset 0x400_3600 (QCSP<i>_ISR)																															Access: w1c
offset 0x10000																															
range i=0..9																															
R	0	0	0	0	0	0	0	0	0	CCSCI	CSCI	EQCI	EQRI	DQRI	MRI															DQ_AVAIL	
W												w1c	w1c	w1c	w1c	w1c															w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**Figure 3-30. QCSP Interrupt Status Register (QCSP*i*\_ISR)**

**Table 3-29. QCSP*i*\_ISR Field Descriptions**

Field	Description
0-9	Reserved
10 CCSCI	CEETM Congestion state change notifications (CEETM CSCN) interrupt This interrupt is asserted when the state of a CEETM class congestion group (CCG) changes from not congested to congested (or vice-versa), and that CCG is configured to notify this portal of the state change. See <a href="#">Section 3.3.20.11.1, “CEETM Congestion State Change Notifications (CEETM CSCN)”</a> .
11 CSCI	Congestion state change notifications (CSCN) interrupt This interrupt is asserted when the state of a FQ congestion group changes from not congested to congested (or vice-versa), and that congestion group is configured to notify this portal of the state change. See <a href="#">Section 3.3.14.6, “Congestion State Change Notifications (CSCN)”</a> .
12 EQCI	Enqueue command dispatched interrupt This interrupt is asserted when an enqueue command is dispatched for execution and an interrupt on command dispatch has been requested in that command; see <a href="#">Section 3.3.8.1.4, “EQCR Interrupts,”</a> and <a href="#">Section 3.3.9.1, “Enqueue Command.”</a>
13 EQRI	EQCR ring interrupt This interrupt is asserted when the number of entries in this portal’s EQCR is below a programmed threshold; see <a href="#">Section 3.2.3.18, “QCSP EQCR Interrupt Threshold Register (QCSP<i>i</i>_EQCR_ITR)”</a> .
14 DQRI	DQRR non-empty interrupt. This interrupt is asserted when the number of entries in this portal’s DQRR exceeds a programmed threshold (see <a href="#">Section 3.2.3.19, “QCSP DQRR Interrupt Threshold Register (QCSP<i>i</i>_DQRR_ITR)”</a> ), or when the ring has been non-empty for longer than a programmed time out period (see <a href="#">Section 3.2.3.25, “QCSP Interrupt Time-Out Period Registers (QCSP<i>i</i>_ITPR)”</a> ).

**Table 3-29. QCSP*i*\_ISR Field Descriptions (continued)**

Field	Description
15 MRI	<p>MR non-empty interrupt</p> <p>This interrupt is asserted when the number of entries in this portal's MR exceeds a programmed threshold (see <a href="#">Section 3.2.3.20, “QCSP MR Interrupt Threshold Register (QCSPi_MR_ITR)”,</a> or when the ring has been non-empty for longer than a programmed time out period (see <a href="#">Section 3.2.3.25, “QCSP Interrupt Time-Out Period Registers (QCSPi_ITPR)”,</a>).</p>
16-31 DQ_AVAIL	<p>This field contains 1 bit used for dequeue available notification in each of the channels available for consumption by the software portal.</p> <ul style="list-style-type: none"> <li>Bit 16: One or more frames are available to be dequeued from the channel dedicated to this software portal.</li> <li>Bit 17: One or more frames are available to be dequeued from pool channel 1.</li> <li>...</li> <li>Bit 31: One or more frames are available to be dequeued from pool channel 15.</li> </ul> <p>A DQ_AVAIL bit is asserted when one or more non-empty WQs are present in the channel assigned to that bit, or when an active FQ is present in the portal and that FQ was dequeued from the channel associated with that bit. When the last FQ is pulled from its WQ, all WQs in a channel may become empty, but that last FQ may remain active in the portal if it is not yet empty, and the DQ_AVAIL bit remains asserted (in this case).</p> <p>Using these bits together with the interrupt enables in QCSPi_IER and/or QCSPi_ISDR, software can decide which software portals get notified of dequeue availability for their dedicated channel and each of up to 15 pool channels. In SoCs that contain fewer than 15 pool channels, the right-most bits of DQ_AVAIL are not used. The number of pool channels available is specified in <a href="#">Section 3.3.2.4, “Work Queue Channel Assignments.”</a></p>

### **3.2.3.22 QCSP Interrupt Enable Registers (QCSP*i*\_IER)**

The QMan software portal interrupt enable registers (QCSP*i*\_IER) should be mapped into a cache-inhibited page within each software portal.

Each of the `QCSPi_IER` provides the ability to individually enable each interrupt source to assert the interrupt signal associated with that portal. If a bit location is asserted in both `QCSPi_ISR` and `QCSPi_IER`, the interrupt is asserted. Clearing an interrupt by clearing the interrupt source if applicable, and then clearing the associated bit in the `QCSPi_ISR`.

**Figure 3-31. QCSP Interrupt Enable Register (QCSP*i*\_IER)**

**Table 3-30. QCSP*i*\_IER Field Descriptions**

Field	Description
0-31	Interrupt enable bit for each bit in QMAN_ERR_ISR All fields in this register carry the same assignment as in QMAN_ERR_ISR; see <a href="#">Section 3.2.3.21, “QCSP Interrupt Status Register (QCSPi_ISR).”</a>

### 3.2.3.23 QCSP Interrupt Status Disable Registers (QCSP*i*\_ISDR)

The QMan software portal interrupt status disable registers (QCSP*i*\_ISDR) should be mapped into a cache-inhibited page within each software portal.

QCSP*i*\_ISDR control reporting of interrupts in QMAN\_ERR\_ISR. A bit in QCSP*i*\_ISR is never asserted if the corresponding bit in QCSP*i*\_ISDR is set (interrupt status is disabled).

Offset 0x400_3680 (QCSP<i>_ISDR)																Access: R/W																	
offset 0x10000																																	
range i=0..9																																	
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
W											CCSCI	CSCI	EQCI	EQRI	DQRI	MRI	DQ_AVAIL																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-32. QCSP Interrupt Enable Registers (QCSP*i*\_IER)

Table 3-31. QCSP*i*\_ISDR Field Descriptions

Field	Description
0-31	Interrupt status disable bit for each bit in the QMAN_ERR_ISR All fields in this register carry the same assignment as in the QMAN_ERR_ISR; see <a href="#">Section 3.2.3.21, “QCSP Interrupt Status Register (QCSP<i>i</i>_ISR).”</a>

### 3.2.3.24 QCSP Interrupt Inhibit Registers (QCSP*i*\_IIR)

The QMan software portal interrupt inhibit registers (QCSP*i*\_IIR) should be mapped into a cache-inhibited page within each software portal.

Each of the QCSP*i*\_IIR allows the corresponding portal’s interrupt signal to be inhibited without modifying the enable or the status disable registers.

Offset 0x400_36C0 (QCSP<i>_IIR)																Access: R/W																	
offset 0x10000																																	
range i=0..9																																	
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
W																															I		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-33. QCSP Interrupt Inhibit Register (QCSP*i*\_IIR)

Table 3-32. QCSP*i*\_IIR Field Descriptions

Field	Description
0-30	Reserved
31 I	Interrupt inhibit 0 Interrupt not inhibited. The corresponding QMan interrupt signal may assert. 1 Interrupt inhibited. The corresponding QMan interrupt signal does not assert.

### 3.2.3.25 QCSP Interrupt Time-Out Period Registers (QCSP*i*\_ITPR)

The QMan software portal interrupt time-out period registers (QCSP*i*\_ITPR) should be mapped into a cache-inhibited page within each software portal. Use these registers to configure a time-out period for the DQRR and MR non-empty interrupts (the DQRI and MRI interrupts in the portal's QCSP*i*\_ISR) in each software portal; these interrupts are described in [Section 3.2.3.21, “QCSP Interrupt Status Register \(QCSP\*i\*\\_ISR\).”](#)

This time-out period applies when the threshold for the DQRI and/or the MRI is greater than 0. These thresholds are described in [Section 3.2.3.19, “QCSP DQRR Interrupt Threshold Register \(QCSP\*i\*\\_DQRR\\_ITR\),”](#) and [Section 3.2.3.20, “QCSP MR Interrupt Threshold Register \(QCSP\*i\*\\_MR\\_ITR\).”](#) In that case, the DQRR or MR may contain one or more entries without causing the interrupt to assert, and this behavior is useful for interrupt coalescing. However, when this is used, there is also an upper bound set on how long one or more entries may be in the ring without alerting software via interrupt, and that is the purpose of the interrupt time-out period.

For each software portal, the QMan maintains a 20-bit counter that is used to count the DQRI time-out period, and another that counts the MRI time out period. The behavior of the DQRI (and MRI) time-out counter is as follows:

- The counter is reset when the DQRR is empty or when the DQRI bit is asserted in QCSP*i*\_ISR.
- The counter advances when the DQRR is non-empty and the DQRI bit in QCSP*i*\_ISR is not asserted. It stops counting when its value equals the time out period programmed in QCSP*i*\_ITPR.
- When the counter reaches the time out period value programmed in QCSP*i*\_ITPR, the DQRI bit in the QCSP*i*\_ISR is asserted (unless this ISR bit is disabled in the QCSP*i*\_ISDR).

Note that the same time-out period value (programmed in QCSP*i*\_ITPR) is used for both the DQRIs and the MRIs. The interrupt time-out counters operate at the QMan's internal clock frequency, and the value specified in QCSP*i*\_ITPR is the 12 msbs of the 20-bit time out period value, such that time-out periods can be specified from 0 up to 1048320 (0xF\_FF00) QMan clock cycles in increments of 256.

Offset 0x400_3740 (QCSP <i>i</i> _ITPR)		Access: R/W	
		offset 0x10000	range i=0..9
R	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 1 2 3   4 5 6 7   8 9 10 11   12 13 14 15   16 17 18 19   20 21 22 23   24 25 26 27   28 29 30 31	ITP
W	[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 3-34. QCSP Interrupt Time Out Period Registers (QCSP*i*\_ITPR)

Table 3-33. QCSP*i*\_ITPR Field Descriptions

Field	Description
0-19	Reserved
20-31 ITP	DQRR and MR non-empty Interrupt time out period Sets the time out period for the DQRIs and MRIs in each software portal.

### 3.2.4 QMan Configuration and Control Register Descriptions

See [Section 3.2.2, “QMan Configuration and Control Register Memory Map,”](#) for a general overview of the QMan configuration and control registers.

#### 3.2.4.1 QCSP ICID Configuration Registers (QCSP*i*\_ICID\_CFG)

Offset 0x1000 (QCSP<i>\_LIO\_CFG) Access: R/W  
 offset 0x10  
 range i=0..9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved				EICID		ICID																									
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-35. QCSP ICID Configuration (QCSP*i*\_ICID\_CFG)

Table 3-34. QCSP*i*\_ICID\_CFG Field Descriptions

Field	Description
0-7	Reserved
8-9 EICID	Extended Isolation Context ID (4 msbs of complete ICID) The complete 10-bit ICID value is a concatenation of {EICID, ICID}. This field contains the extended ICID offset value that is used for all frames enqueued via this software portal. The QMan automatically inserts the EICID value into the frame descriptor (FD) of frames that are enqueued via this software portal. The EICID field in the FD provided by software in the enqueue command is overwritten by the value from this register. See <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs),”</a>
10-15 ICID	Isolation Context ID (6 lsbs of complete ICID) This field contains the ICID value that is used for all frames enqueued via this software portal. The QMan automatically inserts this ICID value into the frame descriptor (FD) of frames that are enqueued via this software portal. The ICID field in the FD provided by software in the enqueue command is overwritten by the value from this register. See <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs),”</a>
16-31	Reserved

#### 3.2.4.2 QCSP IO Configuration Registers (QCSP*i*\_IO\_CFG)

Offset 0x1004 (QCSP<i>\_IO\_CFG) Access: R/W  
 offset 0x10  
 range i=0..9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	SDEST						0	0	0	0	Reserved													
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

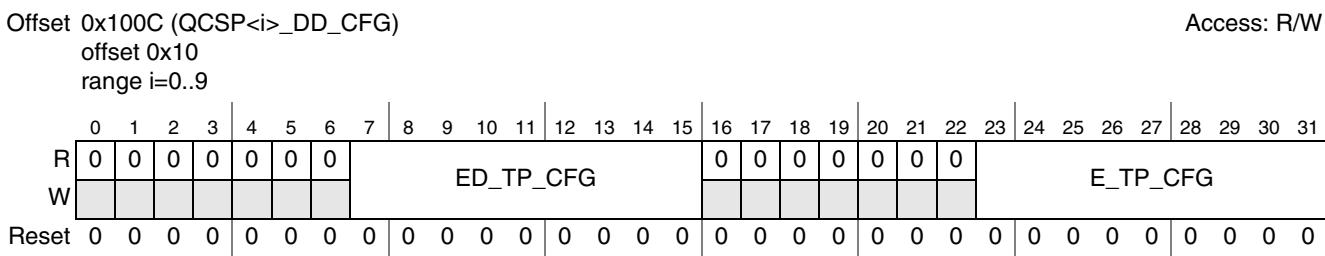
Figure 3-36. QCSP IO Configuration Registers (QCSP*i*\_IO\_CFG)

**Table 3-35. QCSP*i*\_IO\_CFG Field Descriptions**

Field	Description
0-7	Reserved
8-15 SDEST	<p>Stashing destination</p> <p>In a SoC that supports write and/or read allocate stashing to core or core cluster backside caches (e.g. using Power cores or the ACP port of ARM core clusters):</p> <p>This value is used to direct all stashing transactions initiated on behalf of this software portal (DQRR entry stashing, EQCR_CI stashing, dequeued frame data/annotation/flow context stashing) to one of the available stashing destinations in the SoC. A stashing destination may be the L1 or L2 backside cache of a particular core, or the L2 cache of a particular cluster of cores.</p> <p>In a SoC that does not support write or read allocate stashing to core backside caches:</p> <p>This value is not used. Software portal specific stashing may still be supported, but such stashes are all directed at a single common platform cache in the system.</p>
16-31	Reserved

### 3.2.4.3 QCSP Dynamic Debug Configuration Registers (QCSP*i*\_DD\_CFG)

The QMan software portal dynamic debug configuration registers (QCSP*i*\_DD\_CFG) are used to configure dynamic debug (DD) for the portals (see [Section 3.3.15, “Dynamic Debug \(DD\)”](#)).

**Figure 3-37. QCSP Dynamic Debug Configuration Registers (QCSP*i*\_DD\_CFG)**

**Table 3-36. QCSP*i*\_DD\_CFG Field Descriptions**

Field	Description
0-6	Reserved
7-15 ED_TP_CFG	<p>Enqueue deferred complete trace point configuration</p> <p>For each portal, this field configures the dynamic debug trace point that can generate a trace event and/or halt the portal when the deferred enqueue of a marked frame received via this portal completes (that is, at the time when the frame is enqueued onto the FQ after being delayed due to order restoration).</p> <p>3 bits are used to configure tracing at this trace point for each of three different dynamic debug (DD) codes in a frame. The location of the DD bits in a frame's FD is described in <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs).”</a></p> <p>Note that DD code = 00 means no tracing for that frame, so no configuration bits are provided for this code.</p> <ul style="list-style-type: none"> <li>Bits 7-9: Trace point configuration for frame DD code = 01</li> <li>Bits 10-12: Trace point configuration for frame DD code = 10</li> <li>Bits 13-15: Trace point configuration for frame DD code = 11</li> </ul> <p>For each of the three frame DD codes listed above, the configuration bits are defined as follows (x = don't care):</p> <ul style="list-style-type: none"> <li>• xx0 = Trace disabled</li> <li>• x01 = Trace enabled, terse output</li> <li>• x11 = Trace enabled, verbose output</li> <li>• 0xx = Portal halt disabled</li> <li>• 1xx = Portal halt enabled</li> </ul>
16-22	Reserved
23-31 E_TP_CFG	<p>Enqueue command trace point configuration</p> <p>For each portal, this field configures the dynamic debug trace point that can generate a trace event and/or halt the portal when an enqueue command containing a marked frame received via this portal is executed.</p> <p>Three bits are used to configure tracing at this trace point for each of three different dynamic debug (DD) codes in a frame. The frame DD codes are the same as those used in the ED_TP_CFG trace point.</p> <ul style="list-style-type: none"> <li>Bits 23-25: Trace point configuration for frame DD code = 01</li> <li>Bits 26-28: Trace point configuration for frame DD code = 10</li> <li>Bits 29-31: Trace point configuration for frame DD code = 11</li> </ul> <p>For each of the three frame DD codes listed above, the definition of the configuration bits is identical to that for QCSP<i>i</i>_DD_CFG[ED_TP_CFG].</p>

### 3.2.4.4 QMan Dynamic Debug Configuration Register (QMAN\_DD\_CFG)

See [Section 3.3.15, “Dynamic Debug \(DD\),”](#) for more information on dynamic debug (DD).

Offset 0x200 (QMAN_DD_CFG)																															Access: R/W	
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	L	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	MDD	M_CFG		
Reset																															All zeros	

**Figure 3-38. QMan Dynamic Debug Configuration Register (QMAN\_DD\_CFG)**

**Table 3-37. QMAN\_DD\_CFG Field Descriptions**

<b>Field</b>	<b>Description</b>
0 L	Lossy mode This read-only bit indicates the mode in which dynamic debug trace events from the QMan are currently configured; see <a href="#">Section 3.3.15.2, “Debug Trace Points.”</a> 0 Lossless mode 1 Lossy mode
1 F	Full. This read-only bit indicates the current status of the dynamic debug trace event output FIFO buffer; see <a href="#">Section 3.3.15.2, “Debug Trace Points.”</a> 0 FIFO is not full. 1 FIFO is full.
2-25	Reserved
26-27 MDD	Order restoration deferral marking dynamic debug code This field contains the dynamic debug (DD) code that is placed in a frame's FD when it is marked by the QMan as a result of order restoration deferral marking, as configured in QMAN_DD_CFG[M_CFG]. The location of the DD bits in a frame's FD is described in <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs).”</a>
28-31 M_CFG	Order restoration deferral marking configuration This field configures the dynamic debug (DD) codes in a frame for which QMan marks (or re-marks) a frame when its enqueue is deferred due to order restoration. See <a href="#">Section 3.3.15, “Dynamic Debug (DD).”</a> Bit 28: Marking configuration for frame DD code = 00 Bit 29: Marking configuration for frame DD code = 01 Bit 30: Marking configuration for frame DD code = 10 Bit 31: Marking configuration for frame DD code = 11 For each of the frame DD codes listed above, the configuration bit is defined as follows: 0 Marking disabled; QMan does not mark frames carrying this DD code. 1 Marking enabled; QMan marks a frame carrying this DD code if its enqueue is deferred due to order restoration. The DD field in the frame's FD is overwritten with the MDD value from this register.

### **3.2.4.5 QCSP Dynamic Debug Internal Halt Request Status Registers (QCSP\_DD\_IHRSR\_*i*)**

The QMan software portal dynamic debug internal halt request status registers (QCSP\_DD\_IHRSR<sub>*i*</sub>) are used to indicate to software (or to a debugger reading this register) the internal halt request status of each of the QMan's portals. An internal halt request can be generated by the arrival of a marked frame at one of the QMan's internal debug trace points (see [Section 3.3.15.4, “Debug Halt”](#)). Only the internal halt request status is shown, the status of the external halt request is not indicated in this register.

When a bit is asserted, it remains asserted until cleared by writing 1 to it. Clearing a bit removes the internal halt request for that portal.

Offset 0x240 (QCSP\_DD\_IHRSR\_0) Access: w1c  
 0x250 (QCSP\_DD\_IHRSR\_1)  
 0x260 (QCSP\_DD\_IHRSR\_2)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																															
W																															

Reset All zeros

**Figure 3-39. QCSP Dynamic Debug Internal Halt Request Status Registers (QCSP\_DD\_IHRSR<sub>i</sub>)****Table 3-38. QCSP\_DD\_IHRSR<sub>i</sub> Field Descriptions**

Field	Description
0-31 IHRS	Internal halt request status for QMan software portals In QCSP_DD_IHRSR_0: bit 0-31 = Internal halt request status for software portal 0 to 31 In QCSP_DD_IHRSR_1: bit 0-31 = Internal halt request status for software portal 32 to 63 In QCSP_DD_IHRSR_2: bit 0-31 = Internal halt request status for software portal 64 to 95 For each bit: 0!Internal halt request not asserted 1!Internal halt request asserted Bits that correspond to non-existent portals have no effect.

### 3.2.4.6 DCP Dynamic Debug Internal Halt Request Status Register (DCP\_DD\_IHRSR)

The direct connect portal dynamic debug internal halt request status register (DCP\_DD\_IHRSR) is used to indicate to software (or to a debugger reading this register) the internal halt request status of each of the QMan's direct connect portals. An internal halt request can be generated by the arrival of a marked frame at one of the QMan's internal debug trace points (see [Section 3.3.15.4, “Debug Halt”](#)). Only the internal halt request status is shown, the status of the external halt request is not indicated in this register.

DCP\_DD\_IHRSR indicates the internal halt request status of all direct connect portals. When a bit in this register is asserted, it remains asserted until cleared by writing a 1 to it. Clearing a bit removes the internal halt request for that portal.

Offset 0x220 (DCP\_DD\_IHRSR) Access: w1c

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	IHRS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	w1c																															

Reset All zeros

**Figure 3-40. DCP Dynamic Debug Internal Halt Request Status Register (DCP\_DD\_IHRSR)**

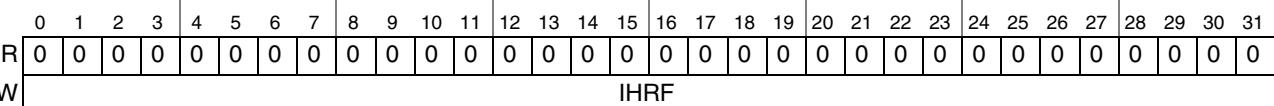
**Table 3-39. DCP\_DD\_IHRSR Field Descriptions**

Field	Description
0-2 IHRS	Internal halt request status for direct connect portals Bit 0: Internal Halt Request Status for DCP0 Bit 1: Internal Halt Request Status for DCP1 Bit 2: Internal Halt Request Status for DCP2  For each bit: 0 Internal halt request not asserted 1 Internal halt request asserted
3-31	Reserved. Any bit that corresponds to a non-existent portal is reserved.

### 3.2.4.7 QCSP Dynamic Debug Internal Halt Request Force Registers (QCSP\_DD\_IHRFR\_i)

Software can use the QMan software portal dynamic debug internal halt request force registers (QCSP\_DD\_IHRFR\_i) to assert an internal halt request to any of the QMan’s software portals. Normally, an internal halt request is generated only by the arrival of a marked frame at one of the QMan’s internal debug trace points (see [Section 3.3.15.4, “Debug Halt”](#)).

Writing a 1 to any valid bit in QCSP\_DD\_IHRFR sets the corresponding bit in QCSP\_DD\_IHRSR.

Offset 0x244 (QCSP_DD_IHRFR_0)	Access: W
0x254 (QCSP_DD_IHRFR_1)	
0x264 (QCSP_DD_IHRFR_2)	
	
Reset	All zeros

**Figure 3-41. QCSP Dynamic Debug Internal Halt Request Force Registers (QCSP\_DD\_IHRFR\_i)****Table 3-40. QCSP\_DD\_IHRFR\_i Field Descriptions**

Field	Description
0-31 IHRF	Internal halt request force for QMan software portals All fields in this register carry the same assignment as in IHRSR; see <a href="#">Section 3.2.4.5, “QCSP Dynamic Debug Internal Halt Request Status Registers (QCSP_DD_IHRSR_i)”</a> .

### 3.2.4.8 DCP Dynamic Debug Internal Halt Request Force Register (DCP\_DD\_IHRFR)

Software can use the direct connect portal dynamic debug internal halt request force register (DCP\_DD\_IHRFR) to assert an internal halt request to any of the QMan’s portals. Normally, an internal halt request is generated only by the arrival of a marked frame at one of the QMan’s internal debug trace points (see [Section 3.3.15.4, “Debug Halt”](#)).

Writing a 1 to any valid bit in DCP\_DD\_IHRFR sets the corresponding bit in DCP\_DD\_IHRSR.

**Figure 3-42. DCP Dynamic Debug Internal Halt Request Force Register (DCP\_DD\_IHRFR)**

**Table 3-41. DCP\_DD\_IHRFR Field Descriptions**

Field	Description
0-2 IHRF	<p>Internal halt request force for direct connect portals</p> <p>All fields in this register carry the same assignment as in IHRSR; see <a href="#">Section 3.2.4.6, “DCP Dynamic Debug Internal Halt Request Status Register (DCP_DD_IHRSR).”</a></p>
3-31	Reserved. Any bit that corresponds to a non-existent portal is reserved.

### **3.2.4.9 QCSP Dynamic Debug Halt Acknowledge Status Registers (QCSP DD HASR<sub>i</sub>)**

The QMan software portal dynamic debug halt acknowledge status registers (QCSP\_DD\_HASR<sub>*i*</sub>) are used to indicate to software (or to a debugger reading this register) the halt acknowledge status of each of the QMan’s software portals. When a portal enters the halted state as a result of an error, or an internal halt request, an external debug halt request, a halt acknowledge signal is sent to the SoC debug logic, and the corresponding bit in this register is asserted (see [Section 3.3.15.4, “Debug Halt”](#)).

If a software portal has been halted as a result of an internal halt request, that software portal's bit is asserted in the corresponding IHRSR register as well. If a software portal is halted but its IHRSR bit is negated, the software portal has been halted either by the SoC debug logic or by an error. In the case of an error, the error is indicated in the QMAN\_ERR\_ISR, and all software portals are halted if the corresponding bit in QMAN\_ERR\_HER is set.

Offset 0x248 (QCSP_DD_HASR_0)	Access: R
0x258 (QCSP_DD_HASR_1)	
0x268 (QCSP_DD_HASR_2)	
0 1 2 3   4 5 6 7   8 9 10 11   12 13 14 15   16 17 18 19   20 21 22 23   24 25 26 27   28 29 30 31	
R	HAS
W	
Reset	All zeros

**Figure 3-43. QCSP Dynamic Debug Halt Acknowledge Status Registers (QCSP\_DD\_HASR\_i)**

**Table 3-42. QCSP\_DD\_HASR\_i Field Descriptions**

Field	Description
0-31 HAS	Halt acknowledge status for QMan software portals In QCSP_DD_HASR_0: bit 0-31 = Halt acknowledge status for software portal 0 to 31 In QCSP_DD_HASR_1: bit 0-31 = Halt acknowledge status for software portal 32 to 63 In QCSP_DD_HASR_2: bit 0-31 = Halt acknowledge status for software portal 64 to 95 For each bit: 0 Software portal is not in the halted state 1 Software portal is in the halted state Bits that correspond to non-existent portals have no effect.

### 3.2.4.10 DCP Dynamic Debug Halt Acknowledge Status Register (DCP\_DD\_HASR)

Use the direct connect portal dynamic debug halt acknowledge status (DCP\_DD\_HASR) to indicate to software (or to a debugger reading this register) the halt acknowledge status of each of the QMan's direct connect portals. When a portal enters the halted state as a result of an error, or an internal halt request, or an external debug halt request, a halt acknowledge signal is sent to the SoC debug logic, and the corresponding bit in this register is asserted (see [Section 3.3.15.4, “Debug Halt”](#)).

If a direct connect portal has been halted as a result of an internal halt request, that direct connect portal's bit is asserted in the corresponding IHRSR register as well. If a direct connect portal is halted but its IHRSR bit is negated, the direct connect portal has been halted either by the SoC debug logic or by an error. In the case of an error, the error is indicated in the QMAN\_ERR\_ISR, and all direct connect portals are halted if the corresponding bit in QMAN\_ERR\_HER is set.

Offset 0x228 (DCP_DD_HASR)																															Access: R
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HAS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

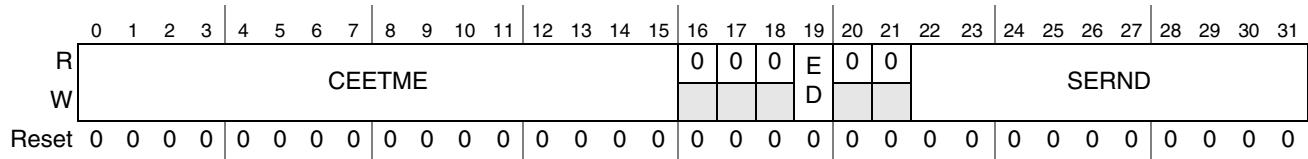
**Figure 3-44. DCP Dynamic Debug Halt Acknowledge Status Register (DCP\_DD\_HASR)****Table 3-43. DCP\_DD\_HASR Field Descriptions**

Field	Description
0-2 HAS	Halt acknowledge status for direct connect portals Bit 0: Halt acknowledge status for DCP0 Bit 1: Halt acknowledge status for DCP1 Bit 2: Halt acknowledge status for DCP2  For each bit: 0 Direct connect portal is not in the halted state 1 Direct connect portal is in the halted state
6-31	Reserved. Any bit that corresponds to a non-existent portal is reserved.

### 3.2.4.11 DCP Configuration Registers (DCPi\_CFG)

Use the direct connect portal configuration registers (DCPi\_CFG) to configure the portals.

Offset 0x300 (DCP<i>\_CFG) Access: R/W  
offset 0x10  
range i=0..2



**Figure 3-45. DCP Configuration Registers (DCPi\_CFG)**

**Table 3-44. DCPI\_CFG Field Descriptions**

Field	Description
0-15 CEETME	<p>Customer Edge Egress Traffic Management (CEETM) Enable.</p> <p>Each bit is used to individually enable CEETM scheduling mode on each sub-portal of this DCP portal.</p> <ul style="list-style-type: none"> <li>Bit 0 = CEETM enable for sub-portal 0.</li> <li>0 = CEETM is disabled on this sub-portal, regular FQ/WQ scheduling mode is used.</li> <li>1 = CEETM is enabled on this sub-portal, CEETM scheduling mode is used.</li> </ul> <p>Bit 1 = CEETM enable for sub-portal 1.</p> <p>...</p> <p>Bit 15 = CEETM enable for sub-portal 15.</p> <p>CEETM may be supported on only a subset of all DCP portals in the SoC, or on none at all, for portals in which CEETM is not supported this entire field is reserved. See <a href="#">Section 3.3.20.2, “CEETM Features.”</a></p> <p>For DCP portals on which CEETM is supported, setting a bit to 1 in this field enables CEETM scheduling/shaping mode for that sub-portal. Setting a bit to 0 enables regular FQ/WQ scheduling mode for that sub-portal. See <a href="#">Section 3.3.20.1, “CEETM Overview.”</a></p> <p>Note that most DCP portals contain fewer than 16 sub-portals, bits corresponding to unused sub-portals are reserved.</p>
16-18	Reserved
19 ED	<p>Enqueue rejection notification (ERN) destination</p> <p>0 Send ERN to software: enqueue rejection notifications (ERN) that result from enqueues requested by the hardware module using this DCP are sent to software via the message ring (MR) of the software portal selected by the SERND field.</p> <p>This should be used with hardware modules for which enqueue rejections are expected only as a result of an error condition, not as a result of congestion management or avoidance.</p> <p>1 Send ERN to hardware: enqueue rejection notifications (ERN) that result from enqueues requested by the hardware module using this DCP are sent back to the hardware block via this DCP.</p> <p>This should be used with hardware modules for which enqueue rejections may occur as a result of congestion management or avoidance, as well as an error condition. This is the recommended setting for any hardware module which has the ability to receive enqueue rejections, such as FMan.</p> <p>Note that the ED bit is only writeable in a DCP which is used by a hardware module that has the ability to receive enqueue rejections, such as FMan. In a DCP used by a hardware module which cannot receive enqueue rejections, such as the security engine, the ED bit is not writeable and will always read back as ED = 0, i.e. ERN are always sent to a software portal. <a href="#">Section 3.3.10, “Direct Connect Portals (DCPs)”</a> lists the connectivity between QMan and the hardware modules which use a DCP.</p>

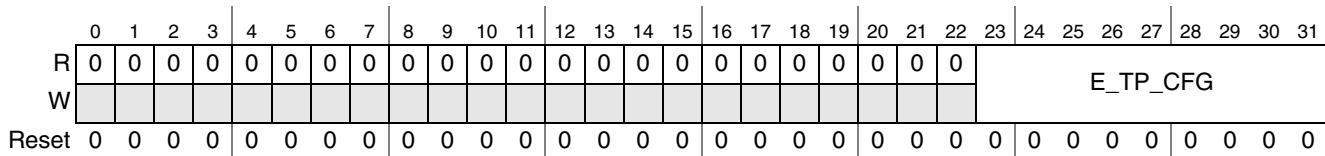
**Table 3-44. DCPI\_CFG Field Descriptions (continued)**

Field	Description
20-21	Reserved
22-31 SERND	Software enqueue rejection notification (ERN) destination This field is valid only if ED = 0. Indicates which software portal should receive the enqueue rejection notifications (ERN) that result from enqueues requested by the hardware module using this DCP. Value 0 to (num_software_portals - 1): ERN for this DCP are sent to the MR of the software portal whose value is configured in SERND. Value greater than or equal to num_software_portals: Reserved If a reserved value is programmed, ERN for this DCP are sent to software portal 0.

### 3.2.4.12 DCP Dynamic Debug Configuration Registers (DCPi\_DD\_CFG)

Use the direct connect portal dynamic debug configuration registers (DCPi\_DD\_CFG) to configure dynamic debug (DD) for the DCPs (see [Section 3.3.15, “Dynamic Debug \(DD\)”](#)).

Offset 0x304 (DCP*<i>*\_DD\_CFG) Access: R/W  
offset 0x10  
range i=0..2

**Figure 3-46. DCP Dynamic Debug Configuration Registers (DCPi\_DD\_CFG)****Table 3-45. DCPI\_DD\_CFG Field Descriptions**

Field	Description
0-22	Reserved
23-31 E_TP_CFG	Enqueue command trace point configuration For each DCP, this field configures the dynamic debug trace point that can generate a trace event and/or halt the portal when an enqueue command containing a marked frame received via this portal is executed. 3 bits are used to configure tracing at this trace point for each of three different dynamic debug (DD) codes in a frame. The location of the DD bits in a frame's FD is described in <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs)”</a> . Note that DD code = 00 means no tracing for that frame, so no configuration bits are provided for this DD code. Bits 23-25: Trace point configuration for frame DD code = 01 Bits 26-28: Trace point configuration for frame DD code = 10 Bits 29-31: Trace point configuration for frame DD code = 11 For each of the three frame DD codes listed above, the configuration bits are defined as follows (x = don't care): xx0 = Trace disable x01 = Trace enabled, terse output x11 = Trace enabled, verbose output 0xx = Portal halt disabled 1xx = Portal halt enabled

### 3.2.4.13 DCP Dequeue Latency Monitor Configuration Registers (DCPi\_DLM\_CFG)

Use the direct connect portal dequeue latency monitor configuration registers (DCPi\_DLM\_CFG) to configure the dequeue latency monitor for each portal. The monitor generates four signals (A, B, C, and D), which are asserted when a dequeue completes and takes as long as or longer than ( $\geq$ ) the corresponding threshold. These signals are received and counted by the central performance monitor module in the SoC. The latency measure starts at the cycle in which the QMan receives a dequeue command in the portal and ends at cycle in which the dequeue response becomes available in the portal.

Offset 0x308 (DCP*i*\_DLM\_CFG) Access: R/W  
 offset 0x10  
 range i=0..

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DLM_TH_A	DLM_TH_B	DLM_TH_C	DLM_TH_D																												
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-47. DCP Dequeue Latency Monitor Configuration (DCPi\_DLM\_CFG)

Table 3-46. DCPI\_DLM\_CFG Field Descriptions

Field	Description
0-3 DLM_TH_A	Specifies a threshold which is used for comparison in the generation of the DCP Dequeue Latency A Performance Monitor signal. The threshold is in units of QMan clock cycles. Note that 9 cycles (code 0) is absolute best case latency by design. 0: 9 cycles 1: 10 cycles 2: 11 cycles 3: 12 cycles 4: 14 cycles 5: 16 cycles 6: 18 cycles 7: 20 cycles 8: 24 cycles 9: 32 cycles 10: 48 cycles 11: 64 cycles 12: 128 cycles 13: 256 cycles 14: 512 cycles 15: 1024 cycles
4-7 DLM_TH_B	Specifies a threshold which is used for comparison in the generation of the DCP Dequeue Latency B Performance Monitor signal. Coding is the same as for DLM_TH_A.
8-11 DLM_TH_C	Specifies a threshold which is used for comparison in the generation of the DCP Dequeue Latency C Performance Monitor signal. Coding is the same as for DLM_TH_A.

**Table 3-46. DCP*i*\_DLM\_CFG Field Descriptions (continued)**

Field	Description
12-15 DLM_TH_D	Specifies a threshold which is used for comparison in the generation of the DCP Dequeue Latency D Performance Monitor signal. Coding is the same as for DLM_TH_A.
16-31 SM	Dequeue Latency Monitoring Sub Portal Enable Mask. This field configures whether or not the dequeue latency measurement is performed for the dequeue commands issued via the corresponding sub portals. This field determines which sub portals contribute to both the performance monitor counters controlled by the DLM_TH_A to DLM_TH_D fields, as well as the weighted moving average presented in the DCP <i>i</i> _DLM_AVG register. Bit 16: 1 = Enable monitoring for dequeue commands made via sub portal 0 Bit 17: 1 = Enable monitoring for dequeue commands made via sub portal 1 ..... Bit 30: 1 = Enable monitoring for dequeue commands made via sub portal 14 Bit 31: 1 = Enable monitoring for dequeue commands made via sub portal 15 Bits in this field associated with sub portals that are not present in a particular DCP portal have no effect. The number of sub portals in each DCP portal is listed in <a href="#">Section 3.3.10, “Direct Connect Portals (DCPs)”</a> .

### 3.2.4.14 DCP Dequeue Latency Monitor Average Registers (DCP*i*\_DLM\_AVG)

The direct connect portal dequeue latency monitor average registers (DCP*i*\_DLM\_AVG) contain an exponentially weighted moving average (EWMA) of dequeue latency samples for dequeue commands received on the sub-portals, which are enabled in the sub-portal enable mask in DCP0-3\_DLM\_CFG[SM]. Note that the threshold settings in DCP*i*\_DLM\_CFG have no effect on this value. The average is calculated with every latency sample:

- $\text{EWMA}_{\text{new}} = (1 - \alpha) * \text{EWMA}_{\text{current}} + \text{Sample} * \alpha$ ; (where  $\alpha = 1/256$ )

The average value is implemented using a 12-bit integer portion and an 8-bit fractional portion. The latency samples are 12-bit integer values (measuring latencies from 0 to 4095 QMan clock cycles) and a power of 2 weight (1/256) is used to allow the average to be implemented without using a multiplier. The actual implementation is as follows:

- `Avg_latency_20_bit <= Avg_latency_20_bit - (Avg_latency_20_bit >> 8) + sample_12_bit;`

Note that after reset, the average value is 0, following which a sufficient number of samples must be taken to arrive at a statistically significant average value reading. To speed up the process of arriving at a significant average value, initialize the average to an expected value by writing to this register.

Offset 0x30C (DCP<i>\_DLM\_AVG)  
Access: R/W  
offset 0x10  
range i=0..2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0																				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

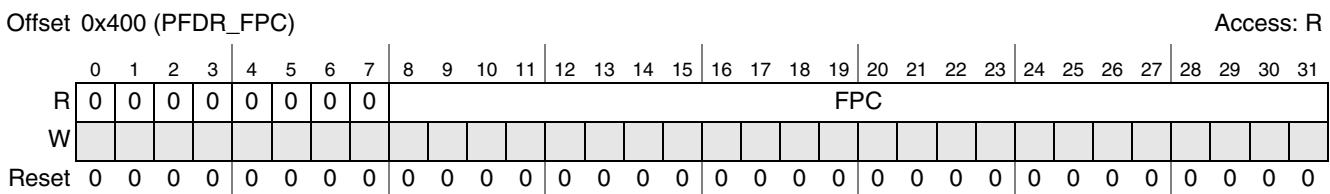
**Figure 3-48. DCP Dequeue Latency Monitor Average Registers (DCP*i*\_DLM\_AVG)**

**Table 3-47. DCP*i*\_DLM\_AVG Field Descriptions**

Field	Description
0-11	Reserved
12-23 DLM_AVG_INT	Integer portion of the measured average latency value
24-31 DLM_AVG_FRACT	Fractional portion of the measured average latency value

**3.2.4.15 PFDR Free Pool Count Register (PFDR\_FPC)**

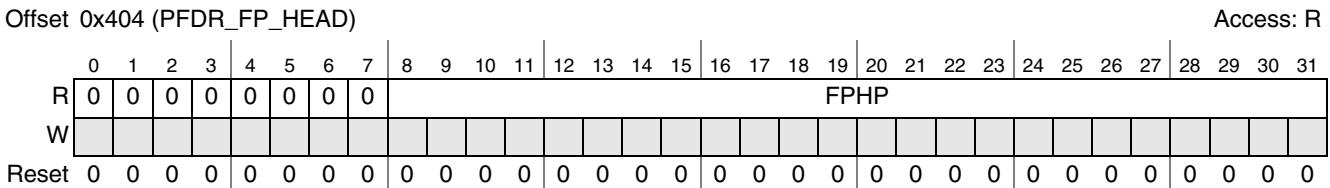
Use the packed frame descriptor record free pool count register (PFDR\_FPC) to read a snapshot of the packed frame descriptor record (PFDR) free pool size at a given time. PFDR\_FPC only shows the PFDRs that are currently part of the free pool in main memory, which does not include the PFDRs held internally to the PFDR manager, or PFDRs that are currently in use.

**Figure 3-49. PFDR Free Pool Count Register (PFDR\_FPC)****Table 3-48. PFDR\_FPC Field Descriptions**

Field	Description
0-7	Reserved
8-31 FPC	Free pool count Total packed frame descriptor record free pool count in external memory

**3.2.4.16 PFDR Free Pool Head Pointer Register (PFDR\_FP\_HEAD)**

The packed frame descriptor record free pool head pointer register (PFDR\_FP\_HEAD) allows the PFDR free pool head pointer to be read.

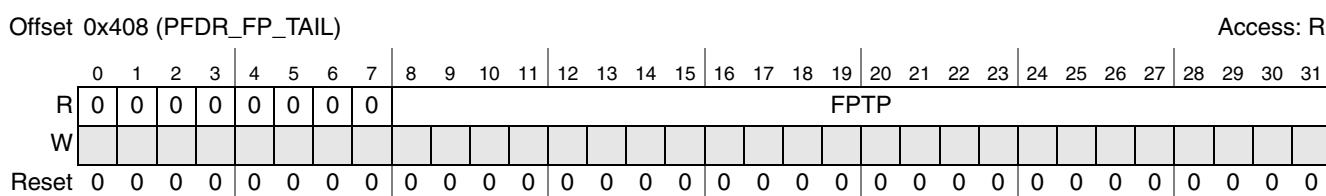
**Figure 3-50. PFDR Free Pool Head Pointer Register (PFDR\_FP\_HEAD)**

**Table 3-49. PFDR FP HEAD Field Descriptions**

Field	Description
0-7	Reserved
8-31 FPHP	Packed frame descriptor record free pool head pointer

### 3.2.4.17 PFDR Free Pool Tail Pointer Register (PFDR FP TAIL)

The packed frame descriptor record free pool tail pointer register (PFDR\_FP\_TAIL) allows the PFDR free pool tail pointer to be read.

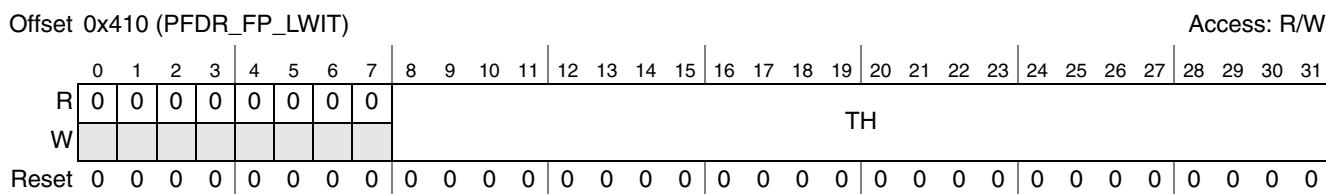


**Figure 3-51. PFDR Free Pool Tail Pointer Register (PFDR\_FP\_TAIL)**

**Table 3-50. PFDR FP TAIL Field Descriptions**

Field	Description
0-7	Reserved
8-31 FPTP	Packed frame descriptor record free pool tail pointer

### 3.2.4.18 PFDR Free Pool Low Watermark Interrupt Threshold (PFDR\_FP\_LWIT)



**Figure 3-52. PFDR Free Pool Low Watermark Interrupt Threshold (PFDR\_FP\_LWIT)**

**Table 3-51. Register PFDR\_FP\_LWIT Field Descriptions**

Field	Description
0-7	Reserved
8-31 TH	PFDR low watermark interrupt threshold An interrupt can be asserted (if enabled in QMAN_ERR_IER) when PFDR_FPC is below this threshold value.

### 3.2.4.19 PFDR Configuration (PFDR\_CFG)

Offset 0x414 (PFDR_CFG)																													Access: R/W			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	K				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-53. PFDR Configuration (PFDR\_CFG)

Table 3-52. Register PFDR\_CFG Field Descriptions

Field	Description
0-23	Reserved
24-31 K	PFDR base constant Used to configure the dynamic allocation policy for PFDRs K is used to account for PFDRs that may be required to complete any currently executing operations in the sequencers. Write a value of 64 to this register at initialization time. This field, combined with the current number of outstanding entries held awaiting order restoration, is compared to the current PFDR free pool size to determine whether further frame enqueues are processed. If there are insufficient PFDRs in the free pool, enqueues from both software and DCPs are blocked until more PFDRs become available. An interrupt can be asserted (if enabled in QMAN_ERR_IER) when enqueues are blocked due to insufficient PFDRs.

### 3.2.4.20 SFDR Configuration Register (SFDR\_CFG)

Use the single frame descriptor record (SFDR) configuration register to configure the allocation policy for SFDRs.

Offset 0x500 (SFDR_CFG)																														Access: R/W		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TH			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-54. SFDR Configuration Register (SFDR\_CFG)

Table 3-53. SFDR\_CFG Field Descriptions

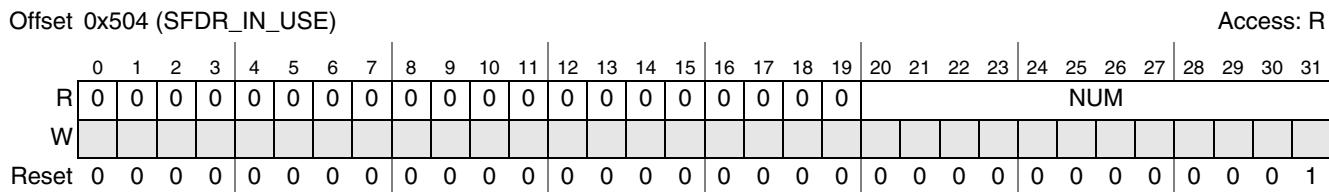
Field	Description
0 RM	SFDR reservation mode 0 Reserved SFDRs can be used by FQs from high priority WQs (WQ 0 and 1 in a channel), and by FQs whose Force_SFDR_Allocate bit is set (see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs)”</a> ). 1 Reserved SFDRs can be used only by FQs whose Force_SFDR_Allocate bit is set. <b>Note:</b> The number of reserved SFDRs is set by SFDR_CFG[TH].

**Table 3-53. SFDR\_CFG Field Descriptions (continued)**

Field	Description
1-21	Reserved
22-31 TH	SFDR reservation threshold TH sets the number of SFDRs reserved for use by certain FQs. The FQs that may use these reserved SFDRs are configured by SFDR_CFG[RM]. If the number of free SFDR drops below this threshold, allocation requests for FQs that are not allowed to use the reserved SFDRs fail, forcing the use of PFDRs instead of SFDRs for those FQs. This threshold applies to SFDR allocation for both enqueue (new RA_SFDRs) and dequeue (new OD_SFDRs). Default (reset) value of the threshold is 0, which means no reservation and SFDRs are allocated on a first-come, first-served basis. Setting the threshold to a value that is higher than the number of SFDRs available (see <a href="#">Section 3.3.1.3.3, “Structure of Frame Queues (FQs)”</a> ) results in all SFDRs being reserved. Note that SFDR allocation is prevented for all FQs if there are fewer than three free SFDRs available. This is done to guarantee that three SFDRs can always be allocated in consecutive cycles. Therefore, to reserve $n$ number of SFDRs, set the value in SFDR_CFG[TH] to $n + 3$ .

### 3.2.4.21 SFDR In Use Register (SFDR\_IN\_USE)

The SFDR in use register (SFDR\_IN\_USE) reports the number of SFDR currently in use.

**Figure 3-55. SFDR In Use Register (SFDR\_IN\_USE)****Table 3-54. SFDR\_IN\_USE Field Descriptions**

Field	Description
0-19	Reserved
20-31 NUM	SFDR in use number NUM reports the number of SFDRs currently in use. The minimum value is 1, because the SFDR manager always keeps one free SFDR in reserve for low latency access.

### 3.2.4.22 Work Queue Class Scheduler Configuration Registers (WQ\_CS\_CFG*i*)

Use the work queue class scheduler configuration registers (WQ\_CS\_CFG*i*) to configure the WQ class schedulers for all work queue (WQ) channels in the QMan.

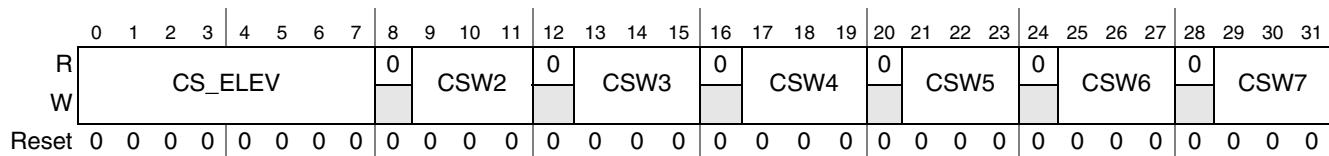
The WQ class schedulers perform scheduling of WQs within a channel (see [Section 3.3.7.2, “WQ Class Scheduling”](#)). The channels in the QMan are grouped into WQ class scheduler configurations. Each channel’s configuration is mapped to one register, as follows:

- WQ\_CS\_CFG0: Configuration for all software portal dedicated channels
- WQ\_CS\_CFG1: Configuration for all software portal pool channels
- WQ\_CS\_CFG2: Configuration for all dedicated channels used by DCP0

## Queue Manager (QMan)

- WQ\_CS\_CFG4: Configuration for the dedicated channels used by DCP2

Offset 0x600 (WQ\_CS\_CFG0)  
 0x604 (WQ\_CS\_CFG1)  
 0x608 (WQ\_CS\_CFG2)  
 0x610 (WQ\_CS\_CFG4) Access: R/W



**Figure 3-56. Work Queue Class Scheduler Configuration (WQ\_CS\_CFGi)**

**Table 3-55. WQ\_CS\_CFG*i* Field Descriptions**

Field	Description
0-7 CS_ELEV	WQ class scheduler priority elevation weight This field is used by the WQ class scheduler to set the weight for elevation of the low priority tier over the medium priority tier. Setting this field to 0 disables the elevation. For non-zero values, the low priority tier is elevated over the medium priority tier 1 in WQ_CS_CFG <i>i</i> [CS_ELEV] + 1 times.
8	Reserved
9-11 CSW2	WQ class scheduler Weight for WQ <i>n</i> ( <i>n</i> = 2-7) Weight assigned to WQ number <i>n</i> in each channel. Assigned weight = CSW <i>n</i> + 1, with a range of 1-8. See <a href="#">Section 3.3.7.2, “WQ Class Scheduling,”</a> for more information.
12	Reserved
13-15 CSW3	WQ class scheduler weight for WQ <i>n</i> ( <i>n</i> = 2-7)
16	Reserved
17-19 CSW4	WQ class scheduler weight for WQ <i>n</i> ( <i>n</i> = 2-7)
20	Reserved
21-23 CSW5	WQ class scheduler weight for WQ <i>n</i> ( <i>n</i> = 2-7)
24	Reserved
25-27 CSW6	WQ class scheduler weight for WQ <i>n</i> ( <i>n</i> = 2-7)
28	Reserved
29-31 CSW7	WQ class scheduler weight for WQ <i>n</i> ( <i>n</i> = 2-7)

### 3.2.4.23 WQ Default Enqueue WQID Register (WQ\_DEF\_ENQ\_WQID)

Offset 0x630 (WQ\_DEF\_ENQ\_WQID) Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-57. WQ Default Enqueue WQID Register (WQ\_DEF\_ENQ\_WQID)

Table 3-56. WQ\_DEF\_ENQ\_WQID Field Descriptions

Field	Description
0-15	Reserved
16-31 WQID	<p>Default enqueue WQ ID</p> <p>During an enqueue operation, if the QMan encounters a FQ whose destination WQ specifies an invalid (reserved) channel, the FQ is enqueued onto the default WQ specified in this register, and an error interrupt is asserted (if enabled).</p> <p>If an invalid WQID is written to this register, and an enqueue is attempted to a FQ configured with an invalid destination WQ (invalid DEST_WQ in the FQD), the default WQ becomes one of the WQs in channel 0. In this case, the 3 lsbs of the default WQID (selecting one of eight WQs within channel 0) are taken from the 3 lsbs of the invalid WQID specified in this register.</p> <p>The assignment of valid WQ IDs is shown in <a href="#">Section 3.3.2.4, “Work Queue Channel Assignments.”</a></p>

### 3.2.4.24 WQ Channel Dynamic Debug Configuration Registers

Use the work queue channel dynamic debug configuration registers to configure the dynamic debug trace point provided at the output of each WQ channel in the QMan (see [Section 3.3.15, “Dynamic Debug \(DD\)”](#)).

Each of these registers can contain one or two dynamic debug trace point configuration fields. The register figures show the general format of each of these registers, and the specific fields used (one or two) in each register are shown in the field descriptions tables.

#### 3.2.4.24.1 WQ Channel Software Portal Dynamic Debug Configuration Registers (WQ\_SC\_DD\_CFG\_i)

Offset 0x1E00 (WQ\_SC\_DD\_CFG\_<i>) Access: R/W

offset 4

range i=0..4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	D_TP_CFG1								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Figure 3-58. WQ Channel Software Portal Dynamic Debug Configuration Registers (WQ\_SC\_DD\_CFG\_i)

**Table 3-57. WQ\_SC\_DD\_CFG\_i Field Descriptions**

Field	Description
0-6	Reserved
7-15 D_TP_CFG1	<p>Dequeue trace point configuration, for the dedicated channel of odd numbered software portals.</p> <p>The specific register and associated channel are as follows:</p> <ul style="list-style-type: none"> <li>In WQ_SC_DD_CFG_i: Dequeue trace point configuration for software portal ((i*2)+1) dedicated channel</li> <li>For example:</li> <li>In WQ_SC_DD_CFG_0: Dequeue trace point configuration for software portal 1 dedicated channel</li> <li>Registers and fields corresponding to non-existent software portals are reserved.</li> <li>The definition of this field is the same as in WQ_SC_DD_CFG_[D_TP_CFG0].</li> </ul>
16-22	Reserved
23-31 D_TP_CFG0	<p>Dequeue trace point configuration, for the dedicated channel of even numbered software portals.</p> <p>The specific register and associated channel are as follows:</p> <ul style="list-style-type: none"> <li>In WQ_SC_DD_CFG_i: Dequeue trace point configuration for software portal (i*2) dedicated channel</li> <li>For example:</li> <li>In WQ_SC_DD_CFG_0: Dequeue trace point configuration for software portal 0 dedicated channel</li> <li>Registers and fields corresponding to non-existent software portals are reserved.</li> </ul> <p>For each WQ channel, this field configures the DD trace point that can generate a trace event and/or halt a portal when a marked frame is dequeued from a WQ in this channel.</p> <p>3 bits are used to configure tracing at this trace point for each of the three different DD codes in a frame. The location of the DD bits in a frame's FD is described in <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs).”</a> Note that when DD code = 00, there is no tracing for that frame, so no configuration bits are provided for this code.</p> <ul style="list-style-type: none"> <li>Bit 23-25: Trace point configuration for frame DD code = 01</li> <li>Bit 26-28: Trace point configuration for frame DD code = 10</li> <li>Bit 29-31: Trace point configuration for frame DD code = 11</li> </ul> <p>For each of the three frame DD codes listed above, the configuration bits are defined as follows (x = don't care):</p> <ul style="list-style-type: none"> <li>xx0 Trace disabled</li> <li>x01 Trace enabled, terse output</li> <li>x11 Trace enabled, verbose output</li> <li>0xx Portal halt disabled</li> <li>1xx Portal halt enabled</li> </ul> <p><b>Note:</b></p>

### 3.2.4.24.2 WQ Channel Pool Dynamic Debug Configuration Registers (WQ\_PC\_DD\_CFG\_i)

Offset 0x680 (WQ_PC_DD_CFG_<i>)	Access: R/W																																																																																																																																		
offset 4																																																																																																																																			
range i=0..7																																																																																																																																			
<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td> </tr> <tr> <td>R</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="8" style="text-align: center;">D_TP_CFG1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>D_TP_CFG0</td><td></td></tr> <tr> <td>W</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr> <td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	R	0	0	0	0	0	0	0	D_TP_CFG1								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D_TP_CFG0		W																																		Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																																																																																																				
R	0	0	0	0	0	0	0	D_TP_CFG1								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D_TP_CFG0																																																																																																				
W																																																																																																																																			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																				

**Figure 3-59. WQ Channel Pool Dynamic Debug Configuration Registers (WQ\_PC\_DD\_CFG\_i)**

**Table 3-58. WQ\_PC\_DD\_CFG\_i Field Descriptions**

<b>Field</b>	<b>Description</b>										
0-6	Reserved										
7-15 D_TP_CFG1	<p>Dequeue trace point configuration</p> <p>The specific register and associated channel are as follows:</p> <ul style="list-style-type: none"> <li>In WQ_PC_DD_CFG_0: Dequeue trace point configuration for pool channel 1</li> <li>In WQ_PC_DD_CFG_1: Dequeue trace point configuration for pool channel 3</li> <li>In WQ_PC_DD_CFG_2: Dequeue trace point configuration for pool channel 5</li> <li>In WQ_PC_DD_CFG_3: Dequeue trace point configuration for pool channel 7</li> <li>In WQ_PC_DD_CFG_4: Dequeue trace point configuration for pool channel 9</li> <li>In WQ_PC_DD_CFG_5: Dequeue trace point configuration for pool channel 11</li> <li>In WQ_PC_DD_CFG_6: Dequeue trace point configuration for pool channel 13</li> <li>In WQ_PC_DD_CFG_7: Dequeue trace point configuration for pool channel 15</li> </ul> <p>The definition of this field is the same as that for D_TP_CFG0 described below.</p>										
16-22	Reserved										
23-31 D_TP_CFG0	<p>Dequeue trace point configuration</p> <p>The specific register and associated channel are as follows:</p> <ul style="list-style-type: none"> <li>In WQ_PC_DD_CFG_0: Reserved</li> <li>In WQ_PC_DD_CFG_1: Dequeue trace point configuration for pool channel 2</li> <li>In WQ_PC_DD_CFG_2: Dequeue trace point configuration for pool channel 4</li> <li>In WQ_PC_DD_CFG_3: Dequeue trace point configuration for pool channel 6</li> <li>In WQ_PC_DD_CFG_4: Dequeue trace point configuration for pool channel 8</li> <li>In WQ_PC_DD_CFG_5: Dequeue trace point configuration for pool channel 10</li> <li>In WQ_PC_DD_CFG_6: Dequeue trace point configuration for pool channel 12</li> <li>In WQ_PC_DD_CFG_7: Dequeue trace point configuration for pool channel 14</li> </ul> <p>For each WQ channel, this field configures the DD trace point that can generate a trace event and/or halt a portal when a marked frame is dequeued from a WQ in this channel.</p> <p>3 bits are used to configure tracing at this trace point for each of the three different DD codes in a frame. The location of the DD bits in a frame's FD is described in <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs)”</a>. Note that when DD code = 00, there is no tracing for that frame, so no configuration bits are provided for this DD code.</p> <ul style="list-style-type: none"> <li>Bit 23-25: Trace point configuration for frame DD code = 01</li> <li>Bit 26-28: Trace point configuration for frame DD code = 10</li> <li>Bit 29-31: Trace point configuration for frame DD code = 11</li> </ul> <p>For each of the three frame DD codes listed above, the configuration bits are defined as follows (x = don't care):</p> <table> <tr> <td>xx0</td> <td>Trace disabled</td> </tr> <tr> <td>x01</td> <td>Trace enabled, terse output</td> </tr> <tr> <td>x11</td> <td>Trace enabled, verbose output</td> </tr> <tr> <td>0xx</td> <td>Portal halt disabled</td> </tr> <tr> <td>1xx</td> <td>Portal halt enabled</td> </tr> </table>	xx0	Trace disabled	x01	Trace enabled, terse output	x11	Trace enabled, verbose output	0xx	Portal halt disabled	1xx	Portal halt enabled
xx0	Trace disabled										
x01	Trace enabled, terse output										
x11	Trace enabled, verbose output										
0xx	Portal halt disabled										
1xx	Portal halt enabled										

### 3.2.4.24.3 WQ Channel DCP Dynamic Debug Configuration Registers (WQ\_DCx\_DD\_CFG\_i)

Offset 0x6C0 (WQ\_DC0\_DD\_CFG\_&lt;i&gt;)

Access: R/W

offset 4

range i=0..7

0x740 (WQ\_DC2\_DD\_CFG\_&lt;i&gt;)

offset 4

range i=0..0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	D_TP_CFG1								0	0	0	0	0	0	0	0	D_TP_CFG0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-60. WQ Channel DCP Dynamic Debug Configuration Registers (WQ\_DCx\_DD\_CFG\_i)

Table 3-59. WQ\_DCx\_DD\_CFG\_i Field Descriptions

Field	Description
0-6	Reserved
7-15 D_TP_CFG1	<p>Dequeue trace point configuration</p> <p>The specific register and associated channel are as follows:</p> <ul style="list-style-type: none"> <li>In WQ_DC0_DD_CFG_0: Dequeue trace point configuration for DCP 0, channel 1</li> <li>In WQ_DC0_DD_CFG_1: Dequeue trace point configuration for DCP 0, channel 3</li> <li>In WQ_DC0_DD_CFG_2: Dequeue trace point configuration for DCP 0, channel 5</li> <li>In WQ_DC0_DD_CFG_3: Dequeue trace point configuration for DCP 0, channel 7</li> <li>In WQ_DC0_DD_CFG_4: Dequeue trace point configuration for DCP 0, channel 9</li> <li>In WQ_DC0_DD_CFG_5: Dequeue trace point configuration for DCP 0, channel 11</li> <li>In WQ_DC0_DD_CFG_6: Dequeue trace point configuration for DCP 0, channel 13</li> <li>In WQ_DC0_DD_CFG_7: Dequeue trace point configuration for DCP 0, channel 15</li> </ul> <p>In WQ_DC2_DD_CFG_0: Dequeue trace point configuration for DCP 2, channel 1: Reserved</p> <p>The definition of this field is the same as that for D_TP_CFG0 described below.</p>

**Table 3-59. WQ\_DCx\_DD\_CFG\_i Field Descriptions (continued)**

Field	Description										
16-22	Reserved										
23-31 D_TP_CFG0	<p>Dequeue trace point configuration</p> <p>The specific register and associated channel are as follows:</p> <ul style="list-style-type: none"> <li>In WQ_DC0_DD_CFG_0: Dequeue trace point configuration for DCP 0, channel 0</li> <li>In WQ_DC0_DD_CFG_1: Dequeue trace point configuration for DCP 0, channel 2</li> <li>In WQ_DC0_DD_CFG_2: Dequeue trace point configuration for DCP 0, channel 4</li> <li>In WQ_DC0_DD_CFG_3: Dequeue trace point configuration for DCP 0, channel 6</li> <li>In WQ_DC0_DD_CFG_4: Dequeue trace point configuration for DCP 0, channel 8</li> <li>In WQ_DC0_DD_CFG_5: Dequeue trace point configuration for DCP 0, channel 10</li> <li>In WQ_DC0_DD_CFG_6: Dequeue trace point configuration for DCP 0, channel 12</li> <li>In WQ_DC0_DD_CFG_7: Dequeue trace point configuration for DCP 0, channel 14</li> <li>In WQ_DC2_DD_CFG_0: Dequeue trace point configuration for DCP 2, channel 0</li> </ul> <p>For each WQ channel, this field configures the DD trace point that can generate a trace event and/or halt a portal when a marked frame is dequeued from a WQ in this channel.</p> <p>3 bits are used to configure tracing at this trace point for each of the three different DD codes in a frame. The location of the DD bits in a frame's FD is described in <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs).”</a> Note that when DD code = 00, there is no tracing for that frame, so no configuration bits are provided for this DD code.</p> <ul style="list-style-type: none"> <li>Bit 23-25: Trace point configuration for frame DD code = 01</li> <li>Bit 26-28: Trace point configuration for frame DD code = 10</li> <li>Bit 29-31: Trace point configuration for frame DD code = 11</li> </ul> <p>For each of the three frame DD codes listed above, the configuration bits are defined as follows (x = don't care):</p> <table border="0"> <tr> <td>xx0</td> <td>Trace disabled</td> </tr> <tr> <td>x01</td> <td>Trace enabled, terse output</td> </tr> <tr> <td>x11</td> <td>Trace enabled, verbose output</td> </tr> <tr> <td>0xx</td> <td>Portal halt disabled</td> </tr> <tr> <td>1xx</td> <td>Portal halt enabled</td> </tr> </table>	xx0	Trace disabled	x01	Trace enabled, terse output	x11	Trace enabled, verbose output	0xx	Portal halt disabled	1xx	Portal halt enabled
xx0	Trace disabled										
x01	Trace enabled, terse output										
x11	Trace enabled, verbose output										
0xx	Portal halt disabled										
1xx	Portal halt enabled										

### 3.2.4.25 CM Configuration Register (CM\_CFG)

Use the CM configuration register (CM\_CFG) to configure the pre-scaler for the WRED byte count averaging timer.

Offset 0x800 (CM_CFG)																	Access: R/W														
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	PRES															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**Figure 3-61. CM Configuration Register (CM\_CFG)**

**Table 3-60. CM\_CFG Field Descriptions**

Field	Description
0-15	Reserved
16-31 PRES	WRED block averaging timer pre-scaler The CM counts PRES + 1 times at the QMan clock frequency, before incrementing the internal CurrentTime counter, which is then used in calculating Congestion Group Record time stamps; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR).”</a> QMan operates at 1/2 the frequency of the platform clock; for example, if the platform clock is 800 MHz, the QMan clock frequency is 400 MHz, and to get a time stamp interval of 1usec at this frequency a PRES value of 399 would be used. For SoC in which one or more DCP portals support CEETM (see <a href="#">Section 3.3.20.2, “CEETM Features”</a> ), this same pre-scaler value is also used to calculate the CEETM Class Congestion Group Record time stamps, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a>

### 3.2.4.26 CEETM Configuration Index Register (CEETM\_CFG\_IDX)

Note this register exists only in SoC in which one or more DCP portals support CEETM, see [Section 3.3.20.2, “CEETM Features.”](#)

Offset 0x900 (CEETM_CFG_IDX)																															Access: R/W			
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
W	[REDACTED]	DCPID																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 3-62. CEETM Configuration Index Register (CEETM\_CFG\_IDX)****Table 3-61. CEETM\_CFG\_IDX Field Descriptions**

Field	Description
0-27	Reserved
28-31 DCPID	CEETM portal ID. In a SoC which contains one or more instances of CEETM logic (each connected to a different DCP portal), this field selects which portal's CEETM registers are addressed by reads and writes to CEETM configuration registers (i.e. registers in the address offset range 0x904 to 0x9FC). This register must be written with the desired DCP ID before attempting reads or writes to the CEETM configuration registers of that portal.

### 3.2.4.27 CEETM Configuration Shaper Pre-Scaler Register (CEETM\_CFG\_PRES)

Note this register exists once for each DCP portal which supports CEETM, see [Section 3.3.20.2, “CEETM Features.”](#) The portal whose register is visible at this location is selected in CEETM\_CFG\_IDX[DCP\_ID].

Offset 0x904 (CEETM_CFG_PRES)																															Access: R/W			
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
W	[REDACTED]	PRES																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 3-63. CEETM Configuration Shaper Pre-Scaler Register (CEETM\_CFG\_PRES)**

**Table 3-62. CEETM\_CFG\_PRES Field Descriptions**

Field	Description
0-15	Reserved
16-31 PRES	<p>CEETM shaper pre-scaler value.</p> <p>This pre-scaler value determines the credit update reference period, i.e. the period at which updates are done to add tokens to each CEETM shaper bucket, see <a href="#">Section 3.3.20.7, “CEETM Channel Shapers.”</a></p> <p>The credit update reference period is implemented using an internal 22 bit reference counter, with the PRES value from this register added to the reference counter once every QMan clock cycle. Therefore the credit update reference period is calculated as:</p> $\text{credit update reference period} = (4194304/\text{PRES}) * \text{QMan clock period}$ <p>The user should start by selecting an appropriate credit update reference period, and then deriving the PRES value required to achieve that reference period for a given QMan clock period.</p> $\text{PRES} = (4194304 / \text{credit update reference period}) * \text{QMan clock period}$ <p>A credit update reference period of 1000ns is suggested, and should serve well for most purposes. This yields a PRES value of 10486 for a 400 MHz QMan clock, or 12583 for a 333 MHz QMan clock.</p> <p>Note that if PRES = 0 the internal reference counter will not increment and the shaper token buckets will never be updated, therefore PRES must always be set to a non-zero value. Note also that the suggested value of 1000ns is the minimum recommended value for the credit update reference period, using a value lower than 1000ns runs the risk of producing incorrect shaper output rates due to insufficient time allowed to update all of the shaper token buckets.</p> <p>The credit update reference period and the shaper's configured token credit rate (CRTCR or ERTCR, see <a href="#">Section 3.3.9.7.13, “CEETM Shaper Configure”</a>) determine the shaper's output rate.</p> $\text{shaper output rate (in bits/sec)} = \text{token credit rate} * 8 / \text{credit update reference period}$ <p>The token credit rate is specified in bytes, with an 11 bit integer and a 13 bit fractional part. Therefore a credit update reference period of 1000ns yields shaper output rates of 0 to 16.384 Gbps in increments of 977 bps.</p>

### 3.2.4.28 CEETM\_XSFDR In Use Register (CEETM\_XSFDR\_IN\_USE)

Note this register exists once for each DCP portal which supports CEETM, see [Section 3.3.20.2, “CEETM Features.”](#) The portal whose register is visible at this location is selected in CEETM\_CFG\_IDX[DCP\_ID].

Offset 0x908 (CEETM_XSFDR_IN_USE)																															Access: R	
R																																
W																																
Reset																																

**Figure 3-64. CEETM\_XSFDR In Use Register (CEETM\_XSFDR\_IN\_USE)****Table 3-63. CEETM\_XSFDR\_IN\_USE Field Descriptions**

Field	Description
0-18	Reserved
19-31 NUM	XSFDR in use number NUM reports the number of XSFDRs currently in use by the CEETM logic of the portal selected by CEETM_CFG_IDX[DCP_ID]. The minimum value is 1, because the hardware always keeps one free XSFDR in reserve for low latency access.

### 3.2.4.29 QMan Error Capture Status Register (QMAM\_ECSR)

When an error condition is detected within QMan, the bit in QMAN\_ERR\_ISR corresponding to that error is asserted (if not disabled by QMAN\_ERR\_ISDR), and an interrupt is asserted (if enabled by QMAN\_ERR\_IER). For most of these error conditions, the QMan provides the ability to capture some information related to the error, for example, the portal in which an invalid command is received, or the address at which an ECC error occurred.

A number of registers can be used to capture the error information, but information for only one error can be captured at any time. The QMan error capture status register (QMAM\_ECSR), shown in [Figure 3-65](#), is used to indicate which error has been captured in the various error capture registers.

When all bits in QMAN\_ECSR (not including ME) are negated, the error capture logic is armed. The contents of all error capture registers is invalid at this point. The first error that occurs causes the associated bit in QMAN\_ECSR to be asserted, and causes that error's information to be captured in the associated error capture registers.

When a bit in QMAN\_ECSR is asserted, the error capture logic is frozen. The bit that is asserted identifies which fields in which error capture registers are valid. Subsequent errors assert the ME bit in this register and may assert other error bits in QMAN\_ERR\_ISR, but information in the error capture registers is saved only for the first error.

If multiple errors happen in the same clock cycle, the error capture logic selects one of the asserted errors for capture in the error capture registers, and the ME bit is asserted at the same time. As such, only one error bit (plus the ME bit if multiple errors have occurred) can ever be set at a time in QMAN\_ECSR. When all of the valid error information has been read, the error capture logic can be re-armed by clearing QMAN\_ECSR, which is done by writing a one to the error bit location which is asserted, and also to the ME bit if it is asserted. If the original source of the interrupt remains asserted, its bit in this register reasserts immediately after the error capture logic is re-armed. If an error source remains asserted for multiple cycles, that constitutes multiple errors and the ME bit is asserted.

Offset 0xA00 (QMAM_ECSR)																Access: w1c			
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
	ME	0	0	0	0	0	MBEI	SBEI	0	0	0	0	0	0	IFSI	ICVI			
W	w1c						w1c	w1c							w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
R	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
	0	0	0	0	IDDI	IDFI	IDSI	IDQI	0	0	0	IECE	IEOI	IESI	IECI	IEQI			
W					w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Figure 3-65. QMan Error Capture Status Register (QMAM\_ECSR)

**Table 3-64. QMAN\_ECSR Field Descriptions**

Field	Description
0 ME	<p>Multiple errors</p> <p>0 No multiple errors occurred since the last time the error capture logic was armed. If one of the other bits in this register is asserted and is 0, only one error occurred, and the error information for that error is captured. No error information is lost.</p> <p>1 Multiple errors occurred since the last time the error capture logic was armed. If this bit is asserted, another bit in this register must also be asserted. Information for the first error that occurred is captured, however, assertion of this bit indicates that one or more additional errors have occurred for which information was not captured. The multiple errors may be errors of the same or different types; if different types of errors occurred, multiple bits are asserted in QMAN_ERR_ISR (see <a href="#">Section 3.2.4.51, “QMan Error Interrupt Status Register (QMAN_ERR_ISR)”</a>), but if multiple errors of the same type occurred only 1 bit is set in QMAN_ERR_ISR.</p>
1-31	<p>Error capture status bit for each bit in the QMAN_ERR_ISR</p> <p>All defined bits in this register carry the same assignment as in the QMAN_ERR_ISR; see <a href="#">Section 3.2.4.51, “QMan Error Interrupt Status Register (QMAN_ERR_ISR)”</a>.</p> <p>Note that not all bits that are defined in QMAN_ERR_ISR are defined in this register, those errors from QMAN_ERR_ISR that are not present in this register do not cause any error capture to occur. Only those errors defined in this register affect the error capture logic, including the ME bit.</p>

### 3.2.4.30 QMan Error Capture Information Registers (QMAN\_ECIR and QMAN\_ECIR2)

When a bit in QMAN\_ECSR is asserted, the QMan error capture information registers (QMAN\_ECIR and QMAN\_ECIR2) may contain (depending on which error has been captured) some information related to the captured error.

Offset 0xA04 (QMAN_ECIR)																														Access: R	

**Figure 3-66. QMan Error Capture Information Register (QMAN\_ECIR)****Table 3-65. QMAN\_ECIR Field Descriptions**

Field	Valid for QMAN_ECSR Error Bits	Description
0-7	—	Reserved
8-31 FQID	IEQI IECI IESI IEOI IECE IDQI IDFI IFSI	Frame queue ID When appropriate for the error condition, the FQ ID related to the error is captured in this field. For an IECE error, the LFQID will be captured here, see the IECE bit definition in <a href="#">Section 3.2.4.51, “QMan Error Interrupt Status Register (QMAN_ERR_ISR)”</a> .

Offset 0xA0C (QMAM_ECIR2)																														Access: R		
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PORTAL	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-67. QMan Error Capture Information Register 2 (QMAM\_ECIR2)

Table 3-66. QMAN\_ECSR Field Descriptions

Field	Valid for QMAN_ECSR Error Bits	Description
0 T	Same as PORTAL field	Portal type When the PORTAL field is valid, this bit indicates the type of portal identified in the PORTAL field. 0 Software portal 1 Direct connect portal
1-21	—	Reserved
22-31 PORTAL	IEQI IESI IEOI IDQI IDS1 IDFI IDDI ICVI IFSI	Portal number When appropriate for the error condition, the portal number on which the error was encountered is captured in this field.

### 3.2.4.31 QMan ECC Error Address Register (QMAM\_EADR)

When the multi-bit ECC error occurred (MBEI) bit or single-bit error occurrence count exceeds programmed threshold (SBEI) bit is set in QMAN\_ECSR, the QMan internal memory on which the error was detected, and the address and data of the error, are saved in a series of error capture registers.

The QMan ECC error address register (QMAM\_EADR), shown in Figure 3-68, indicates which of the QMan internal memories was being read when the error was captured, and the address at which the error was detected.

Offset 0xA08 (QMAM_EADR)																														Access: R		
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	MEM_ID				0	0	0	0	0	0	0	0	0	0	0	0																EADR
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-68. QMan Error Address Register (QMAM\_EADR)

**Table 3-67. QMAN\_EADR Field Descriptions**

<b>Field</b>	<b>Description</b>
0-2	Reserved
3-7 MEM ID	<p>Memory ID</p> <p>This field indicates the QMan internal memory on which an ECC error occurred and was captured. It is valid if the error captured in QMAN_ECSR is either MBEI or SBEI; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a></p> <ul style="list-style-type: none"> <li>0 FQD cache tag memory 0</li> <li>1 FQD cache tag memory 1</li> <li>2 FQD cache tag memory 2</li> <li>3 FQD cache tag memory 3</li> <li>4 FQD cache memory</li> <li>5 SFDR memory</li> <li>6 WQ context memory</li> <li>7 Congestion group record (CGR) memory</li> <li>8 Internal order restoration list memory</li> <li>9 Software portal ring memory</li> <li>10 Egress traffic management class queue descriptor memory</li> <li>11 Egress traffic management extended SFDR memory</li> <li>12 Egress traffic management logical FQ mapping memory</li> <li>13 Egress traffic management dequeue context memory</li> <li>14 Egress traffic management class congestion group record memory</li> <li>15 Egress traffic management class queue channel shaping memory</li> <li>16 Egress traffic management class queue channel scheduling memory</li> <li>17 Egress traffic management dequeue statistics memory</li> </ul>
8-15	Reserved
16-31 EADR	<p>Error Address.</p> <p>The following lists each of QMan's internal memories and the number of address bits used for each:</p> <ul style="list-style-type: none"> <li>FQD cache memory up to 2304 deep, 12 address bits</li> <li>FQD cache tag memory up to 512 deep, 9 address bits</li> <li>SFDR memory: up to 2048 deep, 11 address bits</li> <li>WQ context memory: up to 464 deep, 9 address bits</li> <li>CGR memory: up to 256 deep, 8 address bits</li> <li>Internal order restoration list memory: up to 256 deep, 8 address bits</li> <li>Software portal ring memory: up to 15 address bits</li> <li>CEETM class queue descriptor memory up to 2048 deep, 11 address bits</li> <li>CEETM extended SFDR memory up to 4096 deep, 12 address bits</li> <li>CEETM logical FQ mapping memory up to 4096 deep, 12 address bits</li> <li>CEETM dequeue context memory up to 4096 deep, 12 address bits</li> <li>CEETM class congestion group record memory up to 2048 deep, 11 address bits</li> <li>CEETM class queue channel shaping memory up to 144 deep, 8 address bits</li> <li>CEETM class queue channel scheduling memory up to 128 deep, 7 address bits</li> <li>CEETM dequeue statistics memory up to 512 deep, 9 address bits</li> </ul>

### **3.2.4.32 QMan ECC Error Data Registers (QMAN\_EDATA*i*)**

The QMan ECC error data registers (QMAN\_EDATA*i*) contain the read data on which the ECC error was detected in one of the QMan internal memories.

When the multi-bit ECC error occurred bit (MBEI) or the single-bit error occurrence count exceeds programmed threshold bit (SBEI) is set in QMAN\_ECSR, the QMan internal memory on which the error was detected, and the address and data of the error, are saved in a series of error capture registers.

Offset 0xA10 (QMANT\_EDATA<i>)

Access: R

offset 4  
range i=0..15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	EDATA																																
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-69. QMan Error Data Registers (QMANT\_EDATA*i*)Table 3-68. QMAN\_EDATA*i* Field Descriptions

Field	Description
0-31 EDATA	Error data These 16 registers contain up to 512 bits of data read from a QMan internal memory when an ECC error is the first error detected. When QMAN_ECSR indicates a memory that contains fewer than 512 data bits, the data is right-justified in these registers such that the 32 lsbs from the memory are always in QMAN_EDATA15, the next 32 bits from the memory are in QMAN_EDATA14, and so on. The following lists each of the QMan's internal memories and the number of data bits used for each: FQD cache memory: 512 data bits each FQD cache tag memory: 24 data bits SFDR memory: 128 data bits WQ context memory: 84 data bits Congestion group record memory: 240 or more data bits Internal order restoration List memory: 302 data bits Software portal ring memory: 256 data bits CEETM class queue descriptor memory 181 data bits CEETM extended SFDR memory 140 data bits CEETM logical FQ mapping memory 25 data bits CEETM dequeue context memory 96 data bits CEETM class congestion group record memory 396 data bits CEETM class queue channel shaping memory 146 data bits CEETM class queue channel scheduling memory 256 data bits CEETM dequeue statistics memory 88 data bits

### 3.2.4.33 QMan Single-Bit ECC Error Threshold Register (QMANT\_SBET)

Offset 0xA70 (QMANT\_SBET)

Access: R/W

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	ECDD																																
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-70. QMan Single Bit Error Threshold Register (QMANT\_SBET)

**Table 3-69. QMAN\_SBET Field Descriptions**

Field	Description
0-14 ECDD	<p>Error correction and detection disable</p> <p>This field allows ECC error correction and detection to be individually disabled for each of QMan's internal memories.</p> <p>0 ECC correction and ECC error reporting are enabled in the memory associated with this bit.      1 ECC correction and ECC error reporting are disabled in the memory associated with this bit.</p> <p>The memory associated with each bit is as follows:</p> <ul style="list-style-type: none"> <li>bit 0 FQD cache memory</li> <li>bit 1 FQD cache tag memory</li> <li>bit 2 SFDR memory</li> <li>bit 3 WQ context memory</li> <li>bit 4 Congestion group record memory</li> <li>bit 5 Internal order restoration list memory</li> <li>bit 6 Software portal ring memory</li> <li>bit 7 Egress traffic management class queue descriptor memory</li> <li>bit 8 Egress traffic management extended SFDR memory</li> <li>bit 9 Egress traffic management logical FQ mapping memory</li> <li>bit 10Egress traffic management dequeue context memory</li> <li>bit 11Egress traffic management class congestion group record memory</li> <li>bit 12Egress traffic management class queue channel shaping memory</li> <li>bit 13Egress traffic management class queue channel scheduling memory</li> <li>bit 14Egress traffic management dequeue statistics memory</li> </ul>
15-23	Reserved
24-31 SBET	<p>Single-bit error threshold</p> <p>Threshold value for the number of single-bit ECC errors in any of the QMan internal memories that are detected before reporting an error. When the count in one of the QMAN_SBEC registers is greater than QMAN_SBET, an error is reported in QMAN_ERR_ISR (unless it is disabled in QMAN_ERR_ISDR). A threshold of 0xFF disables reporting of single-bit ECC error interrupts.</p>

### **3.2.4.34 QMan Single Bit ECC Error Count Registers (QMAN\_SBEC<sub>i</sub>)**

Offset 0xA80 (QMAN\_SBEC<i>)

Access: RR

offset 4

range i=0..14

**Figure 3-71. QMan Single Bit Error Count Registers (QMAN\_SBEC<sub>i</sub>)**

**Table 3-70. QMAN\_SBEC*i* Field Descriptions**

Field	Description
0-23	Reserved
24-31 SBEC	<p>Single-bit error count</p> <p>Provides a count of the number of single-bit ECC errors that have occurred when reading from one of the QMan internal memories. The QMan contains one QMAN_SBEC register for each internal memory.</p> <p>QMAN_SBEC0: FQD cache memory</p> <p>QMAN_SBEC1: FQD cache tag memory</p> <p>QMAN_SBEC2: SFDR memory</p> <p>QMAN_SBEC3: WQ context memory</p> <p>QMAN_SBEC4: Congestion group record memory</p> <p>QMAN_SBEC5: Internal order restoration list memory</p> <p>QMAN_SBEC6: Software portal ring memory</p> <p>QMAN_SBEC7: Egress traffic management class queue descriptor memory</p> <p>QMAN_SBEC8: Egress traffic management extended SFDR memory</p> <p>QMAN_SBEC9: Egress traffic management logical FQ mapping memory</p> <p>QMAN_SBEC10: Egress traffic management dequeue context memory</p> <p>QMAN_SBEC11: Egress traffic management class congestion group record memory</p> <p>QMAN_SBEC12: Egress traffic management class queue channel shaping memory</p> <p>QMAN_SBEC13: Egress traffic management class queue channel scheduling memory</p> <p>QMAN_SBEC14: Egress traffic management dequeue statistics memory</p> <p>This register is cleared on read. When the count reaches its max value (0xFF), it sticks at that value without rollover until this register is read.</p>

### 3.2.4.35 QMan Management Command/Result Register (QMAN\_MCR)

Use the QMan management command/result register (QMAN\_MCR) for a variety of QMan management, initialization, and debug-assist features.

Offset 0xB00 (QMAN_MCR)																													Access: R/W		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMD_RSLT		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																															

Reset 0

**Figure 3-72. QMan Management Command/Result Register (QMAN\_MCR)**

**Table 3-71. QMAN\_MCR Field Descriptions**

Field	Description
0-7 CMD_ RSLT	<p>QMan management command/result When a valid command code is written in this byte, the QMan performs the requested command and returns a result code in this byte upon command completion.</p> <p>Reading this register while a command is in progress returns the command that was written. Reading this register after a command has completed returns one of the result codes listed below. Software must poll this byte to determine when a command is complete and a new command can subsequently be issued. Before the command field is written, any required command parameters must be written into the QMAN_MCP<i>i</i> registers. If a command results in returned information, the returned data is held in one or more of the QMAN_MR<i>i</i> registers depending on the command.</p> <p>Command codes:</p> <ul style="list-style-type: none"> <li>0x00: Null command</li> <li>0x01: Initialize packed frame descriptor record (PFDR) free pool. Parameter 0 and 1 are required. No return data</li> <li>0x02: Read one PFDR and return the contents. Parameter 0 is required. Return data in QMAN_MR0–15</li> <li>0x03: Read one SFDR and return the contents. Parameter 0 is required. Return data in QMAN_MR12–15</li> <li>0x10: Query FQD cache fill levels. No parameters required. Return data in QMAN_MR0–6</li> <li>0x11: Query FQD cache tags. Parameter 0 is required. Return data in QMAN_MR0</li> <li>0x12: Query FQD cache contents. Parameter 0 is required. Return data in QMAN_MR0–15</li> <li>0x20: Query WQ XON/XOFF state. Parameter 0 is required. Return data in QMAN_MR15</li> <li>All other command codes: Reserved.</li> </ul> <p>Result codes:</p> <ul style="list-style-type: none"> <li>0x00: Waiting for a command</li> <li>0xF0: Command completed without error, no result returned. Valid for command codes 0x01</li> <li>0xF1: Command completed without error, result returned in QMAN_MR<i>i</i>. Valid for command codes 0x02 to 0x03, 0x10 to 0x12, 0x20</li> <li>0xF4: Command aborted with error, invalid channel number specified in PARAM0. Valid for command code 0x20</li> <li>0xF8: Command aborted with error, this occurs if access to PFDR memory is disabled (PFDR_AR[EN]==0), or if the PFDR index specified is outside of the valid range programmed in PFDR_AR[SIZE]. See <a href="#">Section 3.2.4.44, “Data Structure Attributes Registers.”</a> Valid for command codes 0x01 and 0x02</li> <li>0xFF: Command aborted with error, start index &gt; end index, or start index = 0, or end index &gt; 0xFF_FEFF. Valid for command codes 0x01</li> </ul>
8-31	Reserved

### 3.2.4.36 QMan Management Command Parameter 0 Register (QMAN\_MCP0)

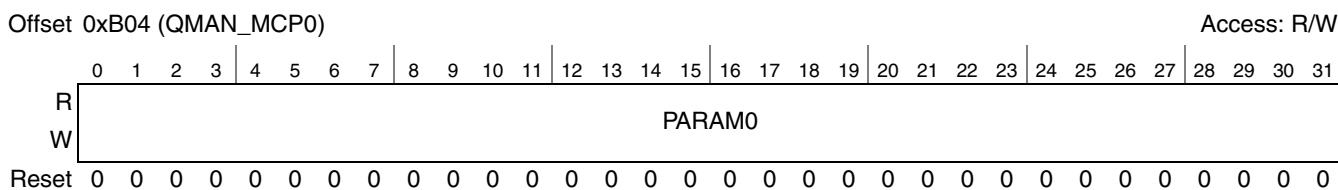
**Figure 3-73. QMan Management Command Parameter 0 Register (QMAN\_MCP0)**

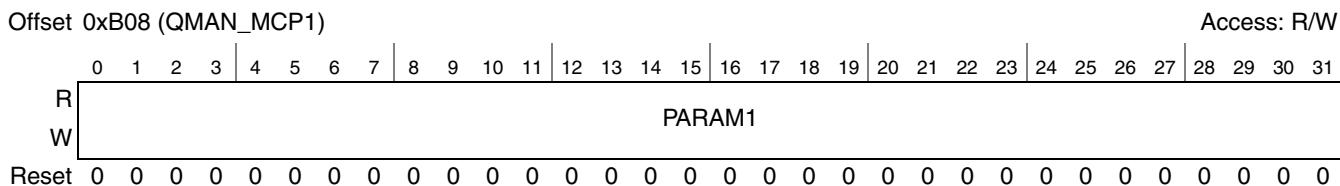
Table 3-72. QMAN\_MCP0 Field Descriptions

Field	Description
0-31 PARAM0	<p>Generic parameter for QMan management command</p> <p>When command code 0x01 is written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>Bits 0-7: Not used</li> <li>Bits 8-31: First PFDR index to be added to the PFDR free pool.</li> </ul> <p>The 3 lsbs (bits 29-31) of this index are not used, because PFDR are always added to the free pool in blocks of eight. For example, if 16 (0x00_0010) is written to this field, PFDR 16 to 23 are the first eight PFDR added to the free pool. The first eight and the last 256 PFDR indices are reserved; therefore this field must be set within the range 0x00_0008 to 0xFF_FEFF.</p> <p>When command code 0x02 is written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>Bits 0-7: Not used</li> <li>Bits 8-31: contains the index of the PFDR that is to be read</li> </ul> <p>When command code 0x03 is written to QMAN_MCR</p> <ul style="list-style-type: none"> <li>Bits 0-20: Not used</li> <li>Bits 21-31: SFDR index (0-2047) to be read</li> </ul> <p>When command code 0x11 or 0x12 is written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>Bits 0-19: Not used</li> <li>Bits 20-31: FQD cache location being queried. FQD cache location addresses are defined as follows:</li> </ul> <ul style="list-style-type: none"> <li>address 0x000 to 0x1FF = hash mapped way 0 FQD Cache locations (up to 512 locations)</li> <li>address 0x200 to 0x3FF = hash mapped way 1 FQD Cache locations (up to 512 locations)</li> <li>address 0x400 to 0x5FF = hash mapped way 2 FQD Cache locations (up to 512 locations)</li> <li>address 0x600 to 0x7FF = hash mapped way 3 FQD Cache locations (up to 512 locations)</li> <li>address 0x800 to 0x83F = up to 64 fully associative overflow locations</li> <li>address 0xA00 to 0xA7F = up to 128 locations reserved for PSCL belonging to the direct connect portals (2 PSCL per sub-portal). The address for the DCP PSCL is formed as {5'b10100, sub_portal_id[5:0], pscl_id[0:0]}, with sub_portal_id values from 0 to (total number of DCP sub-portals - 1). The sub_portal_id values are assigned in linear fashion starting with sub-portal 0 of DCP 0 (which always uses sub_portal_id = 0), followed by the remaining sub-portals of DCP 0, then followed by sub-portals 0 to j-1 of DCP 1, then followed by sub-portals 0 to k-1 of DCP 2, and so on, until the last sub-portal of the last DCP portal, which uses the highest numbered sub_portal_id. The number of DCP portals, and the number of sub-portals in each of those portals, is defined in <a href="#">Section 3.3.10, “Direct Connect Portals (DCPs).”</a> The assignment of sub_portal_id values is illustrated as follows: <ul style="list-style-type: none"> <li>DCP 0, sub-portals 0 to i-1 (i = number of DCP 0 sub-portals): sub_portal_id = 0 to i-1</li> <li>DCP 1, sub-portals 0 to j-1 (j = number of DCP 1 sub-portals): sub_portal_id = i to i+j-1</li> <li>DCP 2, sub-portals 0 to k-1 (k = number of DCP 2 sub-portals): sub_portal_id = i+j to i+j+k-1</li> <li>etc...</li> </ul> </li> <li>address 0xC00 to 0xDFF = up to 512 locations reserved for PSCL belonging to SWP (4 PSCL per software portal). The address for software portal PSCLs is formed as {3'b110, portal_id[6:0], pscl_id[1:0]}, with portal_id values from 0 to (number of software portals – 1).</li> </ul> <p><b>Note:</b> This table continues on the next page....</p> <p>Note that the PARAM0 field must be written before writing a command to QMAN_MCR (if required) and also must not change while the command is underway.</p>

**Table 3-72. QMAN\_MCP0 Field Descriptions (continued)**

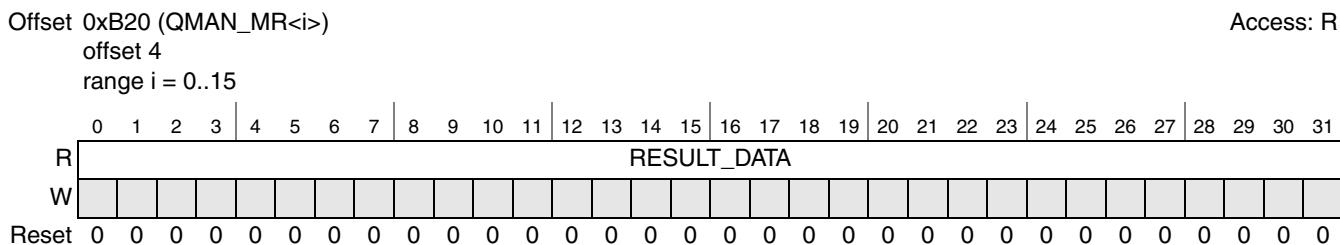
Field	Description
0-31 PARAM0 (continued)	<p>Generic parameter for QMan management command</p> <p>When command code 0x20 is written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>Bits 0-15: Not used.</li> <li>Bits 16-28: Channel number. Specifies the channel whose WQ XON/XOFF states should be queried. These bits correspond to the msbs of a 16 bit WQ ID value. See <a href="#">Section 3.3.2.4, “Work Queue Channel Assignments”</a> for the channel numbers and WQ IDs assigned to direct connect portals. Note that only the dedicated channels serviced by direct connect portals can be queried, an attempt to query any other channel number will result in an error.</li> <li>Bits 29-31 (lsbs): Reserved</li> </ul> <p>Note that the PARAM0 field must be written before writing a command to QMAN_MCR (if required) and also must not change while the command is underway.</p>

### 3.2.4.37 QMan Management Command Parameter 1 Register (QMAN\_MCP1)

**Figure 3-74. QMan Management Command Parameter 1 Register (QMAN\_MCP1)****Table 3-73. QMAN\_MCP1 Field Descriptions**

Field	Description
0-31 PARAM1	<p>Generic parameter for QMan management command</p> <p>When command code 0x01 is written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>Bits 0-7: Not used</li> <li>Bits 8-31: Last PFDR index to be added to the PFDR free pool.</li> </ul> <p>The 3 lsbs (bits 29-31) of this index are not used, because PFDR are always added to the free pool in blocks of eight. For example, if 431 (0x00_01AF) is written to this field, PFDR 424 to 431 are the last eight PFDR added to the free pool. The first 8 and the last 256 PFDR indices are reserved, therefore this field must be set within the range 0x00_0008 to 0xFF_FEFF.</p> <p>Note that the PARAM1 field must be written before writing a command to QMAN_MCR (if required) and also must not change while the command is underway.</p>

### 3.2.4.38 QMan Management Command Result Registers (QMAN\_MR*i*)

**Figure 3-75. QMan Management Command Result Registers (QMAN\_MR*i*)**

**Table 3-74. QMAN\_MR*i* Field Descriptions**

Field	Description
0-31 RESULT_DATA	<p>Allows a QMan management command to return up to 512 bits (64 bytes) of result data. The QMAN_MR<i>i</i> are only valid when a result code 0xF1 is present in the QMAN_MCR.</p> <p>When command code 0x02 was written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>QMAN_MR<i>i</i> are valid (full 64-byte result)</li> </ul> <p>When command code 0x03 was written to QMAN_MCR</p> <ul style="list-style-type: none"> <li>QMAN_MR<i>i</i> are not used</li> </ul> <p>QMAN_MR<i>i</i> are valid. The full SFDR record contains a FD (see <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs)”</a>), and is held in bits 0-127 of the concatenated registers.</p> <p>When command code 0x10 was written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>QMAN_MR0 contains the current fill level of FQD cache hash mapped way 0, in bits 22-31.</li> <li>QMAN_MR1 contains the current fill level of FQD cache hash mapped way 1, in bits 22-31.</li> <li>QMAN_MR2 contains the current fill level of FQD cache hash mapped way 2, in bits 22-31.</li> <li>QMAN_MR3 contains the current fill level of FQD cache hash mapped way 3, in bits 22-31.</li> <li>QMAN_MR4 contains the current fill level of the FQD cache overflow way, in bits 25-31.</li> <li>QMAN_MR5 contains the fill level for DCP PSCLs (i.e. how many DCP PSCL are in use), in bits 22-31.</li> <li>QMAN_MR6 contains the fill level for SWP PSCLs (i.e. how many SWP PSCL are in use), in bits 22-31.</li> <li>QMAN_MR7-15 are not used.</li> </ul> <p>When command code 0x11 was written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>QMAN_MR0: bit 0: 0 = This FQD cache location is not in use. 1 = This FQD cache location is in use.</li> <li>QMAN_MR0: bit 6-7 = State of this FQD cache location (Valid only for the hash mapped and overflow ways) <ul style="list-style-type: none"> <li>0 = Free</li> <li>1 = In use</li> <li>2 = Evictable</li> <li>3 = Reserved</li> </ul> </li> <li>QMAN_MR0: bits 8-31 = FQID of the FQD stored at this cache location, if bit 0 = 1.</li> <li>QMAN_MR1-15: Not used.</li> </ul> <p>When command code 0x12 was written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>QMAN_MR<i>i</i> contain the full 64 byte FQD read from the FQD cache location specified.</li> </ul> <p>When command code 0x20 was written to QMAN_MCR:</p> <ul style="list-style-type: none"> <li>QMAN_MR0-14: Not used.</li> <li>QMAN_MR15: bits 0-23: Not used.</li> <li>QMAN_MR15: bits 24-31 = XON/XOFF state of each WQ in the channel that was queried. <ul style="list-style-type: none"> <li>bit 31 (lsb): 0 = WQ0 in the channel is currently enabled (XON). 1 = WQ0 is disabled (XOFF).</li> <li>....</li> <li>bit 24 (msb): 0 = WQ7 in the channel is currently enabled (XON). 1 = WQ7 is disabled (XOFF).</li> </ul> </li> </ul>

### 3.2.4.39 QMan Miscellaneous Configuration Register (QMAN\_MISC\_CFG)

Offset 0xBE0 (QMAN_MISC_CFG)																															Access: R/W	
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

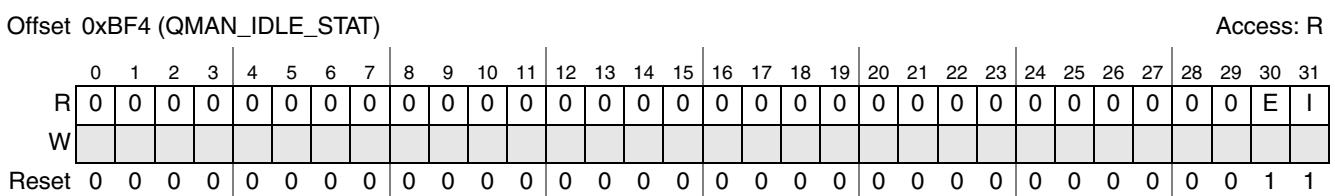
**Figure 3-76. QMan Miscellaneous Configuration Register (QMAN\_MISC\_CFG)**

**Table 3-75. QMAN\_MISC\_CFG Field Descriptions**

Field	Description
0-29	Reserved
30 WPM	<p>Waterfall Power Management enable</p> <p>Setting this bit to 1 enables waterfall power management mode when selecting a software portal to be serviced for dequeue, see <a href="#">Section 3.3.8.2.4, “DQRR Dequeue Command Portal Selection.”</a></p> <p>When this bit is 0, the default portal selection algorithm (round-robin among eligible software portals) is used.</p>
31 FCM	<p>FQD cache mode</p> <p>See <a href="#">Section 3.3.12, “Frame Queue Descriptor Cache.”</a></p> <p>0 Hash mapped</p> <p>1 Direct mapped</p> <p>Hash mapped mode is the default; in this mode, allocation of FQDs into the cache is based on a hash algorithm optimized for maximum cache fill when FQID values are non-contiguous and spread widely over the total FQD space.</p> <p>Direct mapped mode is intended for when an application wants to allocate a large number of consecutive FQIDs into FQD cache and ensure that they all successfully allocate.</p>

### **3.2.4.40 QMan Idle Status Register (QMAN\_IDLE\_STAT)**

Software can use the QMan idle status register (QMAN\_IDLE\_STAT) to determine when the QMan is both idle and empty, which is information that can be used to control power management functions in the SOC.



**Figure 3-77. QMan Idle Status Register (QMAN\_IDLE\_STAT)**

**Table 3-76. QMAN\_IDLE\_STAT Field Descriptions**

Field	Description
0-29	Reserved
30 E	Empty
	This bit is asserted when all WQs in the QMan are empty.
31 I	Idle This bit is asserted when the QMan is idle. The QMan is considered idle when the following conditions are met:
	<ul style="list-style-type: none"> <li data-bbox="287 1434 1356 1448">• No commands or responses are pending, all FIFOs, command rings, response rings, and command registers are empty. This also includes no pending commands in the BMan RCR rings and CR command registers.</li> </ul>
	<ul style="list-style-type: none"> <li data-bbox="287 1448 1356 1461">• No interrupts are pending, that is, the QCSP<i>i</i>_ISR, BCSP<i>i</i>_ISR (in BMan), and QMAN_ERR_ISR registers are all 0.</li> </ul>
	<ul style="list-style-type: none"> <li data-bbox="287 1461 1356 1474">• All Algorithmic Sequencers (AS) are in their idle state.</li> </ul>
	<ul style="list-style-type: none"> <li data-bbox="287 1474 1356 1486">• No I/O transactions are pending on the system interfaces or in a QMan configuration and control register.</li> </ul>

### 3.2.4.41 QMan IP Block Revision 1 Register (QMANT\_IP\_REV\_1)

The QMan IP block revision 1 register (QMANT\_IP\_REV\_1) provides the unique IP block ID for the QMan block, as well as major and minor revision numbers.

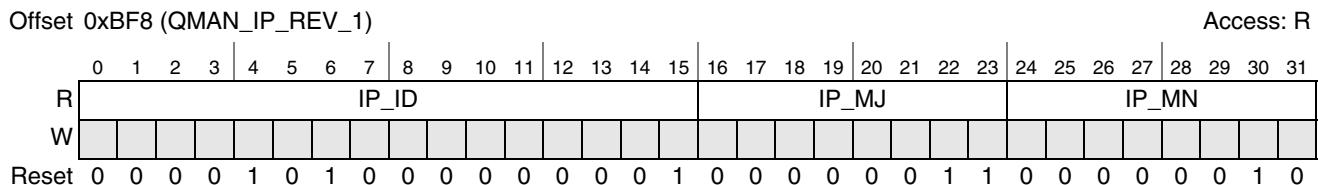


Figure 3-78. QMan IP Block Revision 1 Register (QMANT\_IP\_REV\_1)

Table 3-77. QMAN\_IP\_REV\_1 Field Descriptions

Field	Description																												
0-15 IP_ID	IP Block ID For the QMan, this value is 0xA01.																												
16-23 IP_MJ	Major revision This is currently set to 0x03.																												
24-31 IP_MN	Minor revision This is currently set to 0x02.																												

### 3.2.4.42 QMan IP Block Revision 2 Register (QMANT\_IP\_REV\_2)

The QMan IP block revision 2 register (QMANT\_IP\_REV\_2) provides QMan block integration and configuration options.

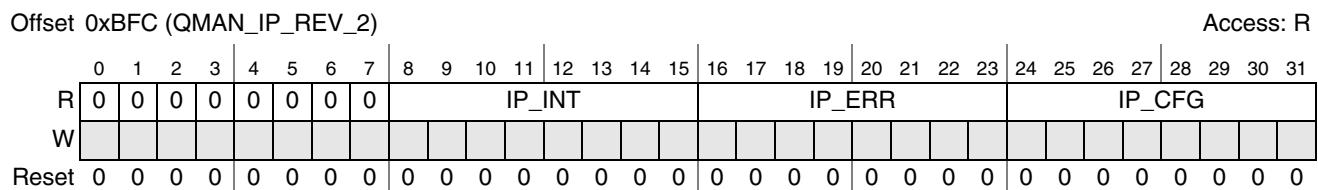


Figure 3-79. QMan IP Block Revision 2 Register (QMANT\_IP\_REV\_2)

Table 3-78. QMAN\_IP\_REV\_2 Field Descriptions

Field	Description																													
0-7	Reserved																													
8-15 IP_INT	Integration options This is currently set to 0x00.																													

**Table 3-78. QMAN\_IP\_REV\_2 Field Descriptions (continued)**

Field	Description
16-23 IP_ERR	Errata revision level This is currently set to 0x00.
24-31 IP_CFG	Configuration options. This is currently set to 0x00. QMan version and options are identified below as IP_MJ.IP_MN.IP_ERR.IP_CFG. Note that the short list of features is not exhaustive.  3.2.0.0: 10 SW portals, 2 DCP portals (0, 2), AXI interfaces, 512 deep FQD cache, 1 DCP with CEETM (LS1043A)

### 3.2.4.43 Data Structure Base Address Registers

The QMan contains two different sets of base address registers (FQD and PFDR), and each one provides a portion of the base address in system memory of one of the private data structures accessed and maintained by the QMan. The base address and attributes of each of these data structures must be initialized by software (see [Section 3.4, “Initialize the QMan”](#)) before traffic (enqueues and dequeues) can flow through the QMan.

The two sets of base address registers are described as follows:

- FQD registers—The area reserved for FQD with the FQD\_BAR/FQD\_BARE/FQD\_AR registers is used solely for accessing FQDs.  
The area reserved for PFDR with the FQD\_BAR/FQD\_BARE/FQD\_AR registers is used for PFDR, as well as for order restoration list (ORL) and enqueue rejection notification list (ERNL) records when these lists extend into main memory.
- PFDR registers—Together, the PFDR\_BAR, PFDR\_BARE, and PFDR\_AR registers describe regions of main memory that have been set aside for use by the QMan to store its private data structures, and as such, these are non-coherent ( $M = 0$ ) memory regions and should be protected from access by anyone other than the QMan. The QMan’s algorithmic sequencers (AS) access these data structures using their index only; the initiator interface module shifts and adds to create the byte address from the index provided by the AS and the programmed base address and size of the data structure’s memory window.

#### 3.2.4.43.1 Data Structure Extended Base Address Registers (FQD\_BARE)

Offset 0xC00 (FQD_BARE)																																Access: R/W
																																EBA
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**Figure 3-80. Data Structure Extended Base Address Registers (FQD\_BARE)**

**Table 3-79. FQD\_BARE Field Descriptions**

Field	Description
0-15	Reserved
16-31 EBA	<p>Extended base address</p> <p>The FQD_BARE and FQD_BAR registers together identify the 36 most significant address bits of the 48-bit base address of the memory window that has been allocated by software for the corresponding data structure (FQD or PFDR) in system memory. BARE contains the 16 most significant (left-most) address bits, and BAR contains the next 20 most significant address bits.</p> <p>The specified base address should be aligned to the window size programmed in FQD_AR[SIZE]. Depending on the size of the window, a number of the lsbs of BAR and BARE are unused; for example, for a minimum sized 4 KB window, all bits of FQD_BAR and FQD_BARE are used, and for an 8 KB window, FQD_BAR[19] is unused, and so on.</p>

### **3.2.4.43.2 Data Structure Extended Base Address Registers (PFDR\_BARE)**

Offset 0xC20 (PFDR\_BARE)

Access: R/W

**Figure 3-81. Data Structure Extended Base Address Registers (PFDR\_BARE)**

**Table 3-80. PFDR BARE Field Descriptions**

Field	Description
0-15	Reserved
16-31 EBA	<p>Extended base address</p> <p>The PFDR_BARE and PFDR_BAR registers together identify the 36 most significant address bits of the 48-bit base address of the memory window that has been allocated by software for the corresponding data structure (FQD or PFDR) in system memory. BARE contains the 16 most significant (left-most) address bits, and BAR contains the next 20 most significant address bits.</p> <p>The specified base address should be aligned to the window size programmed in PFDR_AR[SIZE]. Depending on the size of the window, a number of the lsbs of BAR and BARE are unused; for example, for a minimum sized 4 KB window, all bits of PFDR_BAR and PFDR_BARE are used, and for an 8 KB window, PFDR_BAR[19] is unused, and so on.</p>

### **3.2.4.43.3 Data Structure Base Address Registers (FQD\_BAR)**

Offset 0xC04 (FQD\_BAR)

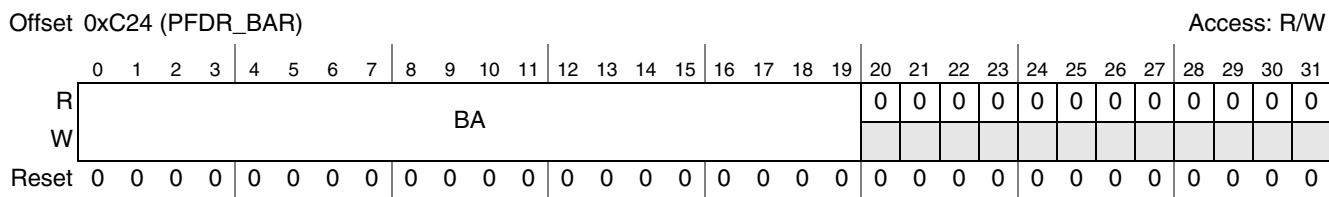
Access: R/W

**Figure 3-82. Data Structure Base Address Registers (FQD BAR)**

**Table 3-81. FQD\_BAR Field Descriptions**

Field	Description
0-19 BA	Base address The FQD_BARE and FQD_BAR registers together identify the 36 most significant address bits of the 48-bit base address of the memory window that has been allocated by software for the corresponding data structure (FQD or PFDR) in system memory. FQD_BARE contains the 16 most significant (left-most) address bits, and FQD_BAR contains the next 20 most significant address bits. The specified base address should be aligned to the window size programmed in FQD_AR[SIZE]. Depending on the size of the window, a number of the lsbs of FQD_BAR and FQD_BARE are unused; for example, for a minimum sized 4 KB window, all bits of FQD_BAR and FQD_BARE are used, and for an 8 KB window, FQD_BAR[19] is unused, and so on.
20-31	Reserved

### 3.2.4.43.4 Data Structure Base Address Registers (PFDR\_BAR)

**Figure 3-83. Data Structure Base Address Registers (PFDR\_BAR)****Table 3-82. PFDR\_BAR Field Descriptions**

Field	Description
0-19 BA	Base address The PFDR_BARE and PFDR_BAR registers together identify the 36 most significant address bits of the 48-bit base address of the memory window that has been allocated by software for the corresponding data structure (FQD or PFDR) in system memory. PFDR_BARE contains the 16 most significant (left-most) address bits, and PFDR_BAR contains the next 20 most significant address bits. The specified base address should be aligned to the window size programmed in PFDR_AR[SIZE]. Depending on the size of the window, a number of the lsbs of PFDR_BAR and PFDR_BARE are unused; for example, for a minimum sized 4 KB window, all bits of PFDR_BAR and PFDR_BARE are used, and for an 8 KB window, PFDR_BAR[19] is unused, and so on.
20-31	Reserved

### 3.2.4.44 Data Structure Attributes Registers

The QMan contains two different attributes registers (FQD\_AR and PFDR\_AR), and each one provides the window size and attributes in system memory of one of the private data structures accessed and maintained by the QMan. The base address and attributes of each of these data structures must be initialized by software (see [Section 3.4, “Initialize the QMan”](#)) before traffic (enqueues and dequeues) can begin flowing through the QMan.

### 3.2.4.44.1 Data Structure Attributes Register (FQD\_AR)

Offset 0xC10 (FQD_AR)																															Access: R/W

Figure 3-84. Data Structure Attribute Register (FQD\_AR)

Table 3-83. FQD\_AR Field Descriptions

Field	Description
0 EN	Data structure access enable 0 This memory window, and access to its corresponding data structure, is disabled. 1 This memory window, and access to its corresponding data structure, is enabled.
1 P	Data structure transaction priority 0 Transactions for this data structure are signaled with lower priority. 1 Transactions for this data structure are signaled with higher priority.
2 SE	CPC stash enable. See <a href="#">Section 3.3.19.3, “CPC Stashing of QMan Private Data Structures.”</a> 0 Transactions for this data structure are signaled with no request for explicit CPC stashing. 1 Transactions for this data structure may be signaled with a request for explicit CPC stashing.
3 SO	CPC stash optimize 0 All FQD writes may be signaled with a request for explicit CPC stashing. 1 Only FQD writes that occur because of a failed attempt to allocate a FQ into FQD cache may be signaled with a request for explicit CPC stashing. Other FQD writes (for example, writing the FQD of an empty or retired FQ) are not signaled with a request for explicit CPC stashing. This setting can be used to reduce the amount of CPC stashing requests by removing some that are not needed.
4-25	Reserved
26-31 SIZE	Identifies the size of the window from the base address. Window size is $2^{(\text{SIZE}+1)}$ bytes. 000000–001010 Reserved 0010114 KB 0011008 KB 00110116 KB ..... $2^{(\text{SIZE}+1)}$ bytes 0111011 GB 011110–111111 Reserved All main memory-space data structures accessed by the QMan are 64 bytes in size. FQD uses a 24-bit pointer, therefore, the maximum possible window size needed for these data structures is 1 GB.

### 3.2.4.44.2 Data Structure Attributes Register (PFDR\_AR)

Offset 0xC30 (PFDR_AR)																															Access: R/W

Figure 3-85. Data Structure Attribute Register (PFDR\_AR)

**Table 3-84. PFDR\_AR Field Descriptions**

Field	Description
0 EN	Data structure access enable 0 This memory window, and access to its corresponding data structure, is disabled. 1 This memory window, and access to its corresponding data structure, is enabled.
1 P	Data structure transaction priority 0 Transactions for this data structure are signaled with lower priority. 1 Transactions for this data structure are signaled with higher priority.
2 SE	CPC stash enable. See <a href="#">Section 3.3.19.3, “CPC Stashing of QMan Private Data Structures.”</a> 0 Transactions for this data structure are signaled with no request for explicit CPC stashing. 1 Transactions for this data structure may be signaled with a request for explicit CPC stashing.
3-25	Reserved
26-31 SIZE	Identifies the size of the window from the base address. Window size is $2^{(\text{SIZE}+1)}$ bytes. 000000–001010 Reserved 0010114 KB 0011008 KB 00110116 KB ..... $2^{(\text{SIZE}+1)}$ bytes 0111011 GB 011110–111111 Reserved All main memory-space data structures accessed by QMan are 64 bytes in size. PFDR uses a 24-bit pointer, therefore, the maximum possible window size needed for these data structures is 1 GB.

### 3.2.4.45 QMan Software Portal (QCSP) Base Address Registers

Together, the QCSP\_BAR and QCSP\_BARE registers define the base address of the 128 MB aligned area of system memory that has been assigned for the QMan’s software portals.

If addresses issued by the QMan as an initiator are not translated before reaching the system interconnect, this base address must be the same as that programmed into the local access window (LAW) registers assigned to the QMan’s target interface.

If address translation is used between the QMan and the system interconnect, the base address programmed in QCSP\_BARE and QCSP\_BAR may be different from the physical address programmed in the LAW of the QMan’s target interface (if such a configuration is desirable), as long as the translated physical address that reaches the system interconnect is the same as that programmed into the LAW registers assigned to the QMan’s target interface.

The base address programmed in QCSP\_BAR and QCSP\_BARE is not used to decode transactions received by QMan; this is already done by the LAW in the SoC system interconnect. Instead, this base address is used to form the full address that is used when issuing DQRR entry or EQCR\_CI stashing transactions; see [Section 3.3.8.6, “DQRR Entry Stashing,”](#) and [Section 3.3.8.7, “EQCR\\_CI Stashing.”](#) When such transactions are issued, they must target address locations in the software portals of the QMan’s target interface, which is why the base address of the software portals is needed within QMan as well as in the LAW registers of the SoC.

**NOTE**

QCSP\_BAR and QCSP\_BARE together specify a 48-bit base address, whereas the LAW base address registers in the SoC may use an address range that is smaller (fewer address bits). In such a case, the msbs of QCSP\_BAR/QCSP\_BARE that are unused should be left at their default value of 0.

### 3.2.4.45.1 QCSP Extended Base Address Register (QCSP\_BARE)

Offset 0xC80 (QCSP_BARE)																																	Access: R/W								
R								EBA																W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-86. QCSP Extended Base Address Register (QCSP\_BARE)

Table 3-85. QCSP\_BARE Field Descriptions

Field	Description																																		
0-15	Reserved																																		
16-31 EBA	Extended base address The QCSP_BARE and QCSP_BAR registers together identify the most significant address bits of the 48-bit base address of the 128 MB memory window that has been allocated by software for the QMan software portals in system memory. QCSP_BARE contains the 16 most significant (left-most) address bits, and QCSP_BAR contains the next 5 most significant address bits.																																		

### 3.2.4.45.2 QCSP Base Address Register (QCSP\_BAR)

Offset 0xC84 (QCSP_BAR)																																		Access: R/W							
R								BA																W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-87. QCSP Base Address Register (QCSP\_BAR)

Table 3-86. QCSP\_BAR Field Descriptions

Field	Description																																	
0-4 BA	Base address See the description above for QCSP_BARE[EBA]																																	
11-31	Reserved																																	

### 3.2.4.46 Initiator Scheduling Configuration (CI\_SCHED\_CFG)

Offset 0xD00 (CI_SCHED_CFG)																																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SRQ_W	0	RW_W	0	BMAN_W					
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3-88. Initiator Scheduling Configuration Register (CI\_SCHED\_CFG)

Table 3-87. CI\_SCHED\_CFG Field Descriptions

Field	Description
0-20	Reserved
21-23 SRQ_W	Stash request queues (SRQ) scheduling weight Weight assigned to all of the SRQs as a group in the initiator transaction request scheduler. The SRQs carry all stashing transactions from the QMan. The assigned weight is SRQ_W + 1, range 1 to 8. See <a href="#">Section 3.3.8.9, “Stash Transaction Scheduling,”</a> and <a href="#">Section 3.3.19.2, “Initiator Scheduling and Priority.”</a>
24	Reserved
25-27 RW_W	QMan read/write transaction scheduling weight Weight assigned to the read and write transactions to the QMan’s private data structures issued by the algorithmic sequencers (AS), the PFDR manager, and the ordered list manager (OLM) in the initiator transaction request scheduler. The assigned weight is RW_W + 1, range 1 to 8. See also <a href="#">Section 3.3.19.2, “Initiator Scheduling and Priority.”</a>
28	Reserved
29-31 BMAN_W	BMan fetch/flush sequencer scheduling weight Weight assigned to the BMan’s fetch/flush sequencer in the initiator transaction request scheduler. The assigned weight is BMAN_W + 1, range 1 to 8. See also <a href="#">Section 3.3.19.2, “Initiator Scheduling and Priority.”</a>

### 3.2.4.47 QMan Source ID Register (QMANT\_SRCIDR)

Offset 0xD04 (QMANT_SRCIDR)																																		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																																		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-89. QMan Source ID Register (QMANT\_SRCIDR)

Table 3-88. QMAN\_SRCIDR Field Descriptions

Field	Description
0-23	Reserved
24-31 SRCID	QMan’s source ID This value is the source ID that is used with the QMan’s initiator transactions. This value is not programmable. This register allows software to read the fixed source ID value used in this SoC.

### 3.2.4.48 QMan Isolation Context ID Register (QMAM\_ICIDR)

Offset 0xD08 (QMAM_ICIDR)		Access: R/W																															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	WC	RA	WA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 3-90. QMan Isolation Context Identifier Register (QMAM\_ICIDR)

Table 3-89. (QMAM\_ICIDR) Field Descriptions

Field	Description
0-8, 12-19	Reserved
9 WC	<p>Write Allocate stash Coherency mode.</p> <p>If write allocate stashing is enabled for one or more software portals (see <a href="#">Section 3.3.8.6, “DQRR Entry Stashing”</a> and <a href="#">Section 3.3.8.7, “EQCR_CI Stashing”</a>), this bit determines the coherency of the write allocate transactions used for stashing issued from QMan:</p> <ul style="list-style-type: none"> <li>0 = Write allocate stash transactions are issued as non-coherent</li> <li>1 = Write allocate stash transactions are issued as coherent</li> </ul> <p>The coherency requirement of QMan transactions is indicated by side-band signals on the interface, and the default setting of 0 means that the write allocate stash transactions will be issued as cacheable and non-coherent (non-shareable). Setting this bit to 1 will cause write allocate stash transactions to be issued as cacheable and coherent (outer-shareable if an ACP port is used).</p>
10 RA	<p>Read Allocate stash steering mode.</p> <p>If read allocate (also known as cache warming) stashing is enabled for one or more software portals (see <a href="#">Section 3.3.8.8, “Dequeued Frame Data, Annotation, and Context Stashing”</a>), this bit determines how the read allocate transactions used for stashing are issued from QMan:</p> <ul style="list-style-type: none"> <li>0 = Read allocate stash transactions are issued with stash indicator side-band signal</li> <li>1 = Read allocate stash transactions are issued the same as all other reads, with no side-band stash signal</li> </ul> <p>The default setting of 0 means that the read allocate stash transactions may be steered by SoC logic to a different path than the regular system interconnect used by all other transactions, for example to the L2 cache of a core cluster via its ACP port. Setting this bit to 1 means that read allocate stash transactions will go through the same path as all other reads issued by QMan.</p>
11 WA	<p>Write Allocate stash steering mode.</p> <p>If write allocate stashing is enabled for one or more software portals (see <a href="#">Section 3.3.8.6, “DQRR Entry Stashing”</a> and <a href="#">Section 3.3.8.7, “EQCR_CI Stashing”</a>), this bit determines how the write allocate transactions used for stashing are issued from QMan:</p> <ul style="list-style-type: none"> <li>0 = Write allocate stash transactions are issued with stash indicator side-band signal</li> <li>1 = Write allocate stash transactions are issued the same as all other writes, with no side-band stash signal</li> </ul> <p>The default setting of 0 means that the write allocate stash transactions may be steered by SoC logic to a different path than the regular system interconnect used by all other transactions, for example to the L2 cache of a core cluster via its ACP port. Setting this bit to 1 means that write allocate stash transactions will go through the same path as all other writes issued by QMan.</p>
20-31 QMAM_ICID	<p>QMan’s Isolation context identifier</p> <p>This value is attached to every transaction initiated by the QMan when accessing its private data structures (PFDR and FQD).</p> <p>Not used outside of the QMan.</p> <p><b>Note:</b> Only 8 lsbs of this value are used.</p>

### 3.2.4.49 Initiator Read Latency Monitor Configuration Register (CI\_RLM\_CFG)

The initiator read latency monitor generates four signals (A, B, C, and D) that are asserted when a read initiated by the QMan completes and takes as long as or longer than ( $\geq$ ) the corresponding threshold. These signals are received and counted by the central performance monitor module in the SoC. The latency measure is from the cycle in which the QMan launches a read to the cycle in which the read response is received.

Offset 0xD10 (CI_RLM_CFG)																															
0 1 2 3				4 5 6 7				8 9 10 11				12 13 14 15				16 17 18 19				20 21 22 23				24 25 26 27				28 29 30 31			
R	W	RLM_TH_A	RLM_TH_B	RLM_TH_C	RLM_TH_D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset		0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	

Figure 3-91. Initiator Read Latency Monitor Configuration Register (CI\_RLM\_CFG)

Table 3-90. CI\_RLM\_CFG Field Descriptions

Field	Description
0-3 RLM_TH_A	Specifies a threshold that is used for comparison in the generation of the initiator read latency A performance monitor signal. The threshold is in units of QMan clock cycles. 0: 6 cycles 1: 8 cycles 2: 12 cycles 3: 16 cycles 4: 24 cycles 5: 32 cycles 6: 48 cycles 7: 64 cycles 8: 96 cycles 9: 128 cycles 10: 192 cycles 11: 256 cycles 12: 384 cycles 13: 512 cycles 14: 768 cycles 15: 1024 cycles
4-7 RLM_TH_B	Specifies a threshold that is used for comparison in the generation of the initiator read latency B performance monitor signal. Coding is the same as for CI_RLM_CFG[RLM_TH_A].
8-11 RLM_TH_C	Specifies a threshold that is used for comparison in the generation of the initiator read latency C performance monitor signal. Coding is the same as for CI_RLM_CFG[RLM_TH_A].
12-15 RLM_TH_D	Specifies a threshold that is used for comparison in the generation of the initiator read latency D performance monitor signal. Coding is the same as for CI_RLM_CFG[RLM_TH_A].
16-31	Reserved

### 3.2.4.50 Initiator Read Latency Monitor Average (CI\_RLM\_AVG)

The initiator read latency monitor average register (CI\_RLM\_AVG) contains an exponentially weighted moving average (EWMA) of read latency samples for reads initiated by the QMan. The average is calculated with every latency sample:

- $EWMA_{new} = (1 - \alpha) * EWMA_{current} + Sample * \alpha$ ; (where  $\alpha = 1/256$ )

A 12-bit integer portion and an 8-bit fractional portion implements the average value. The latency samples are 12-bit integer values (measuring latencies from 0 to 4095 QMan clock cycles) and a power of 2 weight (1/256) is used to allow the average to be implemented without using a multiplier. The actual implementation is as follows:

- $\text{Avg\_latency\_20\_bit} \leq \text{Avg\_latency\_20\_bit} - (\text{Avg\_latency\_20\_bit} \gg 8) + \text{sample\_12\_bit};$

Note that after reset, the average value is 0, following which a sufficient number of samples must be taken to arrive at a statistically significant average value reading. To speed up the process of arriving at a significant average value, the average can be initialized to an expected value by writing to this register.

**Figure 3-92. Initiator Read Latency Monitor Average Register (CI\_RLM\_AVG)**

**Table 3-91. CI\_RLM\_AVG Field Descriptions**

Field	Description
0-11	Reserved
12-23 RLM_AVG_INT	Integer portion of the measured average latency value
24-31 RLM_AVG_FRACT	Fractional portion of the measured average latency value

### 3.2.4.51 QMan Error Interrupt Status Register (QMAN\_ERR\_ISR)

The QMan contains one dedicated interrupt line for signaling error conditions to software. The QMan error interrupt status register (QMAM\_ERR\_ISR) identifies the source of the error interrupt within the QMan.

Any asserted bit in this register can assert an interrupt line if enabled to do so in the QMAN\_ERR\_IER register. When a bit in this register is asserted, it remains asserted until cleared by writing 1 to it. If the original source of the interrupt remains asserted, its bit in this register reasserts immediately after having been cleared.

Offset 0xE00 (QMAM_ERR_ISR)															Access: w1c				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
R	0	0	CIDE	CTDE	CITT	PLWI	MBEI	SBEI	PEBI	0	0	0	0	0	IFSI	ICVI			
W			w1c						w1c	w1c									
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0		
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
R	0	0	0	0	IDDI	IDFI	IDSI	IDQI	0	0	0	IECE	IEOI	IESI	IECI	IEQI			
W					w1c	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Figure 3-93. QMan Error Interrupt Status Register (QMAM\_ERR\_ISR)

Table 3-92. QMAN\_ERR\_ISR Field Descriptions

Field	Description
0-1	Reserved
2 CIDE	Initiator data error This interrupt is asserted if a beat of read data or, in the P1023, a write response with an error indication asserted is received by QMan as an initiator. Transactions initiated by either the QMan or BMan cause this bit to assert. Because QMan provides the initiator interface for both the QMan and BMan, the QMan reports this error on behalf of both, and the BMan does not report this error.
3 CTDE	Target data error This interrupt is not used (reserved).
4 CITT	Invalid target transaction This interrupt is asserted if a transaction with an unrecognized or reserved attribute (such as Target ID, Type, Length, Size, Burst Mode) is received by the QMan as a target.
5 PLWI	PFDR low watermark interrupt This interrupt is asserted if the PFDR free pool occupancy is below a programmable threshold; see <a href="#">Section 3.2.4.18, “PFDR Free Pool Low Watermark Interrupt Threshold (PFDR_FP_LWIT).”</a>
6 MBEI	Multi-bit ECC error interrupt This interrupt is asserted if a multi-bit ECC error is detected in one or more of the QMan internal memories. The particular memory that encountered the error, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.31, “QMan ECC Error Address Register (QMAM_EADR).”</a>
7 SBEI	Single-bit ECC error interrupt This interrupt is asserted if the number of single-bit ECC errors detected in one or more of the QMan internal memories exceeds a programmable threshold. The threshold is described in <a href="#">Section 3.2.4.33, “QMan Single-Bit ECC Error Threshold Register (QMAM_SBET).”</a> The particular memory that encountered the error, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.31, “QMan ECC Error Address Register (QMAM_EADR).”</a>
8 PEBI	PFDR enqueues blocked interrupt This interrupt is asserted if enqueues become blocked due to insufficient PFDR remaining in the PFDR free pool. See <a href="#">Section 3.2.4.19, “PFDR Configuration (PFDR_CFG).”</a> Note that this bit is asserted after reset, and remains so until sufficient PFDR are initialized and this bit is cleared by writing a 1 in this bit location.
9-13	Reserved

**Table 3-92. QMAN\_ERR\_ISR Field Descriptions (continued)**

Field	Description
14 IFSI	Invalid FQ flow control state interrupt This interrupt is asserted if a FQ flow control command is received which specifies a FQ that is in the Out Of Service, Retired, or Parked state; see <a href="#">Section 3.3.6, “Frame Queue (FQ) Flow Control.”</a>
15 ICVI	Invalid command verb interrupt This interrupt is asserted if an unrecognized command verb is received in any software portal. This includes enqueue commands written to the CR register, or management commands written to the EQCR ring. The portal in which the error was detected, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a> Note this error is also triggered if a CEETM management command is issued (see <a href="#">Section 3.3.9.7, “Customer Edge Egress Traffic Management (CEETM) Management Commands”</a> ) with an invalid DCPIID value.
16-19	Reserved
20 IDDI	Invalid dequeue direct connect portal interrupt Dequeue command error; this interrupt is asserted if a dequeue command (or a WQ or FQ flow control command) with an invalid format is received from a hardware block connected to one of the QMan’s direct connect portals. The portal in which the error was detected, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a>
21 IDFI	Invalid dequeue FQ Interrupt Dequeue command error; this interrupt is asserted if an unscheduled dequeue command is received that specifies a FQ that is not in the parked or retired state. The portal in which the error was detected, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a>
22 IDSI	Invalid dequeue source interrupt Dequeue command error; this interrupt is asserted if the DQ_SRC field is invalid in a software portal scheduled (SU = 0), dequeue command from SDQCR or PDQCR. See the DQ_SRC description in <a href="#">Section 3.2.3.11, “QCSP DQRR Static Dequeue Command Register (SDQCR).”</a> The portal in which the error was detected, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a>
23 IDQI	Invalid dequeue queue interrupt Dequeue command error; this interrupt is asserted if an invalid FQID is received in an unscheduled dequeue command or in a FQ flow control command from a software portal or a direct connect portal. The error is triggered if any of the following conditions are true: <ul style="list-style-type: none"> <li>• FQD_AR[EN]==0.</li> <li>• FQID is outside of the valid range programmed in FQD_AR[SIZE].</li> </ul> The portal in which the error was detected, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a>
24-26	Reserved
27 IECE	Invalid enqueue configuration error Enqueue command error; this interrupt is asserted if an enqueue is performed to a CEETM LFQID (see <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue”</a> ), and the CCGID found in the LFQMT entry does not match the CCGID found in the CQD. Note that the enqueue in this case is not rejected (detection of this configuration error occurs beyond the point where the enqueue can be rejected), the consequence of this error is incorrect congestion management for this class queue, since a different CCG will be updated for enqueues to and dequeues from this class queue (see <a href="#">Section 3.3.20.11, “CEETM Class Congestion Management and Avoidance (CCM)”</a> ). The LFQID at which the enqueue was performed is captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a>

**Table 3-92. QMAN\_ERR\_ISR Field Descriptions (continued)**

Field	Description
28 IEOI	Invalid enqueue overflow interrupt Enqueue command error; this interrupt is asserted if an enqueue is rejected, because enqueueing this frame would cause the FQ's byte count (FQD[BYTE_CNT]) or frame count (FQD[FRM_CNT]) fields to overflow. The portal in which the error was detected, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a>
29 IESI	Invalid enqueue state interrupt Enqueue command error; this interrupt is asserted if an enqueue is rejected because the FQ and/or the ORP specified in the enqueue command is in the Retired or Out Of Service state. The portal in which the error was detected, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a>
30 IECI	Invalid enqueue channel interrupt Enqueue error, this interrupt is asserted if an invalid channel number is found in a FQ's DEST_WQ field. When this happens, the enqueue is redirected to a default WQ; see <a href="#">Section 3.2.4.23, “WQ Default Enqueue WQID Register (WQ_DEF_ENQ_WQID).”</a> The FQ on which this error was detected is captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a>
31 IEQI	Invalid enqueue queue interrupt Enqueue command error, this interrupt is asserted if an enqueue is rejected due to an invalid FQID or ORP received in the enqueue command from a software portal or a direct connect portal. The error is triggered if any of the following conditions are true: <ul style="list-style-type: none"> <li>• FQD_AR[EN]==0.</li> <li>• FQID is outside of the valid range programmed in FQD_AR[SIZE].</li> <li>• ORP pointer is outside of the valid range programmed in FQD_AR[SIZE] (software portals only).</li> <li>• FQID destined for CEETM (F0_0000 and above) but not in valid range for any of the portals that support CEETM, see <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue.”</a></li> </ul> The portal in which the error was detected, and some related error information, are captured when the first such error occurs; see <a href="#">Section 3.2.4.29, “QMan Error Capture Status Register (QMAN_ECSR).”</a>

### 3.2.4.52 QMan Error Interrupt Enable Register (QMAN\_ERR\_IER)

The QMan error interrupt enable register (QMAN\_ERR\_IER) individually enables each interrupt source to assert the error interrupt signal. If a bit location is asserted in both QMAN\_ERR\_ISR and QMAN\_ERR\_IER, the interrupt is asserted. To clear an interrupt, clear the interrupt source, if applicable, and then clear the associated bit in QMAN\_ERR\_ISR.

Offset 0xE04 (QMAN_ERR_IER)																Access: R/W	
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
W			CIDE	CTDE	CITT	PLWI	MBEI	SBEI	PEBI	0	0	0	0	0	0	IFSI	ICVI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
W					IDDI	IDFI	IDSI	IDQI	0	0	0	IECE	IEOI	IESI	IECI	IEQI	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 3-94. QMan Error Interrupt Enable Register (QMAN\_ERR\_IER)**

**Table 3-93. QMAN\_ERR\_IER Field Descriptions**

Field	Description
0-31	Interrupt enable bit for each bit in the QMAN_ERR_ISR All fields in this register carry the same assignment as in the QMAN_ERR_ISR; see <a href="#">Section 3.2.4.51, “QMan Error Interrupt Status Register (QMAN_ERR_ISR).”</a>

### 3.2.4.53 QMan Error Interrupt Status Disable Register (QMAN\_ERR\_ISDR)

The QMan error interrupt status disable register (QMAN\_ERR\_ISDR) controls reporting of interrupts in the QMAN\_ERR\_ISR. A bit in QMAN\_ERR\_ISR is never asserted if the corresponding bit in the QMAN\_ERR\_ISDR register is set (interrupt status is disabled).

**Figure 3-95. QMan Error Interrupt Status Disable Register (QMAN\_ERR\_ISDR)**

**Table 3-94. QMAN ERR ISDR Field Descriptions**

Field	Description
0-31	Interrupt status disable bit for each bit in the QMAN_ERR_ISR All fields in this register carry the same assignment as in the QMAN_ERR_ISR; see <a href="#">Section 3.2.4.51, “QMan Error Interrupt Status Register (QMAN_ERR_ISR).”</a>

### 3.2.4.54 QMan Error Interrupt Inhibit Register (QMAN\_ERR\_IIR)

The QMan error interrupt inhibit register (QMAM\_ERR\_IIR) can be used to inhibit the QMan error interrupt signal without modifying the enable or the status disable registers.

**Figure 3-96. QMan Error Interrupt Inhibit Register (QMAN\_ERR\_IIR)**

**Table 3-95. QMAN\_ERR\_IIR Field Descriptions**

Field	Description
0-30	Reserved
31 I	Interrupt inhibit 0 Interrupt not inhibited. The QMan error interrupt signal may assert. 1 Interrupt inhibited. The QMan error interrupt signal does not assert.

### 3.2.4.55 QMan Error Halt Enable Register (QMAN\_ERR\_HER)

The QMan error halt enable register (QMAN\_ERR\_HER) individually enables each error source to halt the QMan. If a bit location is asserted in both QMAN\_ERR\_ISR and QMAN\_ERR\_HER, a halt request is asserted to all of the QMan’s software portals and DCPs simultaneously (see [Section 3.3.15.4, “Debug Halt”](#)).

When asserted, this halt request remains asserted until removed by software. To remove this halt request, clear the error source, if applicable, and then clear the associated bit in QMAN\_ERR\_ISR.

Offset 0xE14 (QMAN_ERR_HER)																Access: R/W	
R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
W			CIDE	CTDE	CITT	PLWI	MBEI	SBEI	PEBI	0	0	0	0	0	0	IFSI	ICVI
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
W	0	0	0	0	IDDI	IDFI	IDSI	IDQI	0	0	0	IECE	IEOI	IESI	IECI	IEQI	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 3-97. QMan Error Halt Enable Register (QMAN\_ERR\_HER)****Table 3-96. QMAN\_ERR\_HER Field Descriptions**

Field	Description
0-31	Halt enable bit for each bit in the QMAN_ERR_ISR All fields in this register carry the same assignment as the QMAN_ERR_ISR (see <a href="#">Section 3.2.4.51, “QMan Error Interrupt Status Register (QMAN_ERR_ISR)”</a> ). <b>Exception:</b> if QMAN_ERR_ISR[CIDE] was asserted by bad read data destined for the BMan, the QMan is not halted even if QMAN_ERR_HER[CIDE] is asserted.

## 3.3 Functional Description

This figure shows the detailed QMan block diagram.

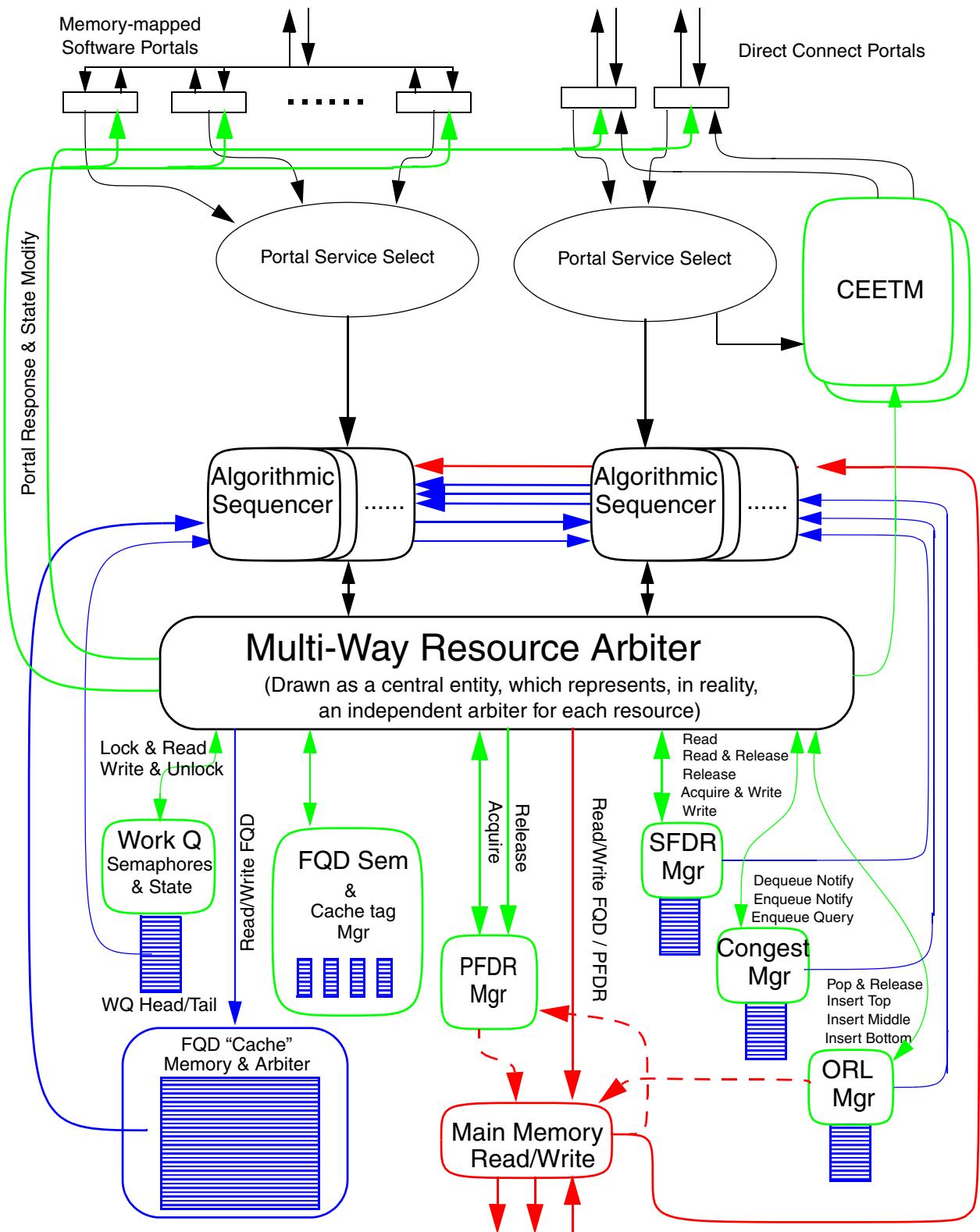


Figure 3-98. QMan Detailed Block Diagram

### 3.3.1 Frames

The basic piece of information that is queued by the QMan is a frame. In its simplest form, a frame consists of a buffer in memory, but it can also consist of multiple buffers that are tied together using a scatter/gather list.

#### 3.3.1.1 Frames and the QMan

A frame is described to the QMan using a well-defined frame descriptor (FD) format (see [Section 3.3.1.2, “Frame Descriptors \(FDs\)”](#)).

The QMan does *not* do the following with frames:

- Impose a format to support frames consisting of multiple buffers. All that is strictly required is that the source and destination of the frame agree upon the format.
- Make use of the buffer memory in the frame itself to implement queueing. Instead, the QMan uses proxy structures stored in both internal and external memory for storing the FD information. This means that the QMan imposes no constraints on the actual buffer(s) in the frame and provides a high degree of compatibility with existing software with regard to buffer format.

#### 3.3.1.2 Frame Descriptors (FDs)

##### 3.3.1.2.1 Definition of Frame Descriptors (FDs)

Frame descriptors (FDs) are carried in the QMan enqueue and dequeue commands and responses. They are 16-byte data structures that exist as out-of-band information used to identify a frame, but are not stored in memory within the frame itself.

FDs are passed to the QMan when a frame is enqueued, and passed from the QMan when a frame is dequeued.

##### 3.3.1.2.2 Structure of Frame Descriptors (FDs)

The sender of a frame creates the fields of the FD, which are interpreted by the receiver of the frame. Senders and receivers of frames are the processor cores and the direct-connected hardware modules attached to the QMan, the QMan itself is neither a sender nor a receiver of frames. In general, the entire FD is passed transparently through the QMan from sender to receiver. The QMan does, however, interpret certain fields of the FD to perform certain functions (see [Section 3.3.14, “Congestion Management and Avoidance”](#)).

#### NOTE

Only the FD fields interpreted by the QMan are described here.

## Queue Manager (QMan)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DD	ICID					BPID					Reserved			EICID	Reserved			ADDR													
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
ADDR																															
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
FORMAT	OFFSET					LENGTH					LENGTH or CONGESTION WEIGHT																				
	STATUS/CMD																														
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

Figure 3-99. Frame Descriptor Format (FD)

Table 3-97. Frame Descriptor (FD)

FD Field	FD Field Description
0-1 DD	Dynamic Debug marking code. This field is interpreted by QMan as follows 00 Disables debug trace for this frame 01,10,11Enables debug trace on this frame A frame with a non-zero debug code can generate a trace event at various enqueue and/or dequeue points within the QMan (see <a href="#">Section 3.3.15, “Dynamic Debug (DD)”</a> ).
2-7 ICID	Isolation context identifier (6 lsbs of complete ICID) This field is overwritten by the QMan in FDs that are provided in enqueue commands received via software portals (see <a href="#">Section 3.3.19.1, “SrcID and ICID”</a> ). The QMan does not use this field in any other situation.
8-15 BPID	Frame buffer pool ID This field is not used by the QMan
16-17	Reserved
18-19 EICID	Extended isolation context identifier (2 msbs of complete ICID) The complete, 8-bit ICID value is a concatenation of {EICID, ICID}. This field is overwritten by the QMan in FDs that are provided in enqueue commands received via software portals (see <a href="#">Section 3.3.19.1, “SrcID and ICID”</a> ). The QMan does not use this field in any other situation.
20-23	Reserved
24-63 ADDR	Frame address ADDR is the memory address of the start of the buffer holding the frame data or the buffer containing the scatter/gather list describing the frame.
64-66 FORMAT	Frame Format A coded value that defines the format of the OFFSET and LENGTH fields, and of the frame referenced by this FD. This value is used to determine the format of the LENGTH field used for congestion management and avoidance. QMan interprets this field as follows: <ul style="list-style-type: none"><li>• If bits 65-66 = 00, this FD contains a 9-bit OFFSET and a 20-bit LENGTH,</li><li>• else, this FD contains only a 29-bit LENGTH or CONGESTION WEIGHT.</li></ul>
67-75 OFFSET	Frame offset This field is valid only when FORMAT bits 65-66 = 00, otherwise, a frame offset value of 0 is implied. Frame offset is the number of bytes from the frame address (ADDR) at which actual data begins. If a frame carries a non-zero offset, additional information, called the frame annotation, may be stored in the buffer carrying the frame, from ADDR to ADDR + OFFSET – 1.

**Table 3-97. Frame Descriptor (FD)**

FD Field	FD Field Description
76-95 or 67-95 LENGTH or CONGESTION WEIGHT	Frame length or congestion weight Length of the data in the frame, or congestion weight of the frame, in bytes. This is interpreted as either a 20- or 29-bit value, depending on the value found in the FORMAT field. The QMan uses this value to perform congestion management and avoidance calculations for this frame (see <a href="#">Section 3.3.14, “Congestion Management and Avoidance”</a> ). Note that even if all congestion management and avoidance features are disabled, the FD length value is still used to perform a default FQ tail drop check to avoid overflow of the FQD byte count, see <a href="#">Section 3.3.14.4, “FQ Tail Drop.”</a>
96-127 STATUS /CMD	Frame status/command This field allows the sender of a frame to communicate some out-of-band information to the receiver of the frame. The QMan does not use this field.

### 3.3.1.3 Frame Queues (FQs)

#### 3.3.1.3.1 Definition of Frame Queues (FQs)

A frame queue (FQ) is the basic queuing structure used by the QMan. It comprises a list of FDs, so it can be thought of as a queue of frames.

The FQ on which the frame is placed is specified by the sender when a frame is passed to the QMan. Because frames are in a list in the FQ, an inherent order is imposed. FQs are FIFO structures, and new frames are added (enqueued) at the tail of the FQ and removed (dequeued) from the head of the FQ. FQs provide the basic packet ordering functionality in the architecture.

#### 3.3.1.3.2 Frame Queues (FQs) and the QMan

The QMan uses a structure called a frame queue descriptor (FQD) to maintain FQs. This structure is described in [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\)”](#).

FQs must be created by software. When software creates a FQ, the QMan allocates the FQD structure. In addition to the linking data for the list of FDs that form the FQ, the FQD also includes a number of other fields that are initialized when software creates the FQ. Software specifies the values for many of these fields.

Frames can be enqueued on a FQ by multiple sources. The QMan arbitrates between these actions, serializes them, and ensures that the FQ is not corrupted. Similarly, frames can be enqueued at the same time that a frame is being dequeued.

#### 3.3.1.3.3 Structure of Frame Queues (FQs)

FQs are built using the following types of data structures:

- Singular frame descriptor record (SFDR)—The SFDR can contain exactly one FD. SFDRs are 16 bytes and are stored exclusively in internal memory within the QMan, and thus are accessed with very low latency. The QMan contains internal memory for storage of up to 2048 SFDRs, which are stored in a single-port internal SRAM. The SFDRM tracks which SFDRs are used through a free list vector.

- Packed frame descriptor record (PFDR)—The PFDRs can contain up to three FDs. PFDRs are 64 bytes and are stored in system memory external to the QMan. PFDRs are linked together to form the linked list of frames in the FQ. The QMan manages a pool of PFDRs from an area of memory allocated for this purpose by software using the PFDR base address and attributes registers. There are two resource interfaces within the PFDRM, one for the acquisition of PFDR and one for the release of PFDR. These distinct interfaces allow the “drop and go” release interface to continue to operate even when the blocking acquire interface is held because of a lack of immediately available PFDRs.  
The memory area reserved for PFDR, and the PFDR free pool, must be initialized by software before the QMan can forward any traffic. PFDR free pool initialization is done using QMAN\_MCR and QMAN\_MCP0–1 (see [Section 3.4, “Initialize the QMan”](#)). If the PFDR free pool becomes depleted during operation, an error interrupt is raised, and software can add more PFDRs to the free pool during run time using the same method as for the initialization. The range of valid PFDRs is 0x8–0xFFFFEFFF.

Software must allocate an area of main memory for PFDR storage, and the allocated area must be communicated to the QMan by programming the PFDR base address and attributes registers; see [Section 3.2.4.43, “Data Structure Base Address Registers.”](#) The PFDR storage area in main memory is intended to be private to the QMan, and should therefore be configured as inaccessible to all processor cores and other hardware modules in the system. Accesses by the QMan accesses to PFDR storage in system memory are performed as non-coherent, because the QMan always assumes that it has private ownership of this area, and accesses to it do not need to be snooped by other entities in the system.

SFDRs and PFDRs can build FQs in the following ways:

- Up to three “on deck” frames at the head of the FQ can be held in SFDRs. These are the next frames removed from the FQ when a dequeue occurs.
- Up to two “recently arrived” frames at the tail of the FQ can also be held in SFDRs. Frames may be placed in these SFDR when an enqueue occurs.
- A linked list of PFDR contains all frames in the FQ that are not currently stored in the on-deck or recently-arrived SFDRs.

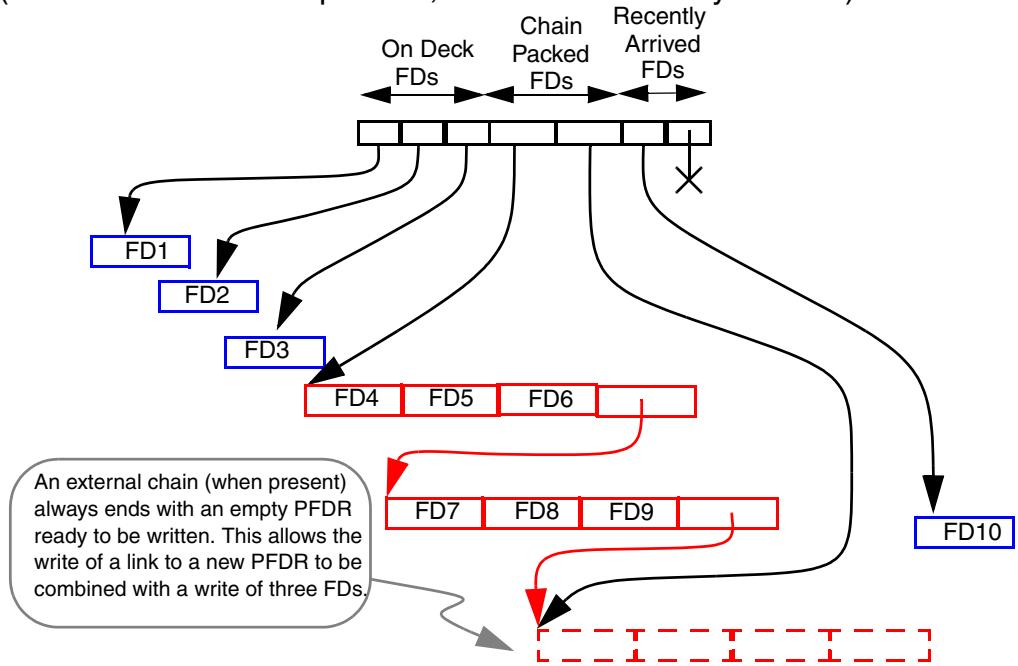
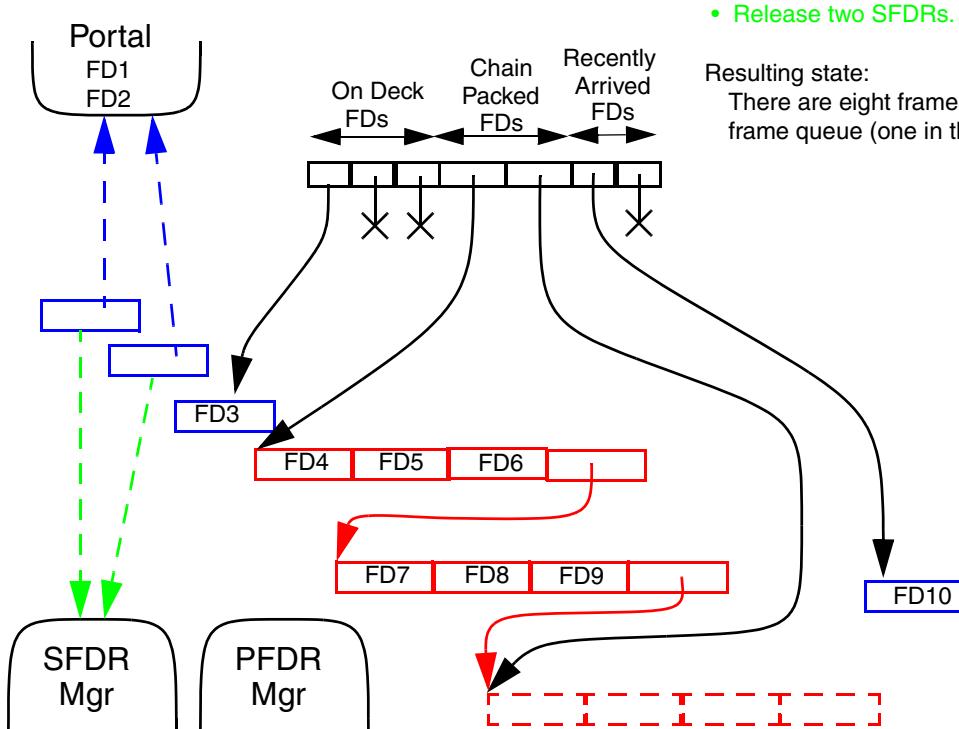
The following are some basic operations that occur on a FQ:

- When the frame in the last on-deck SFDR is consumed, the head PFDR can be read, and its contents transferred into three newly allocated SFDRs, which become the new on-deck SFDRs.
- When there are two recently-arrived frames in SFDRs and a third arrives, all three are transferred into a PFDR in external memory. At the same time, an extra empty PFDR is pre-linked to the one being written; this is done to avoid the inefficiency of another write to this PFDR (the one currently being written) later on when the next PFDR (the empty one being pre-linked) is ready to be written to external memory.
- As long as free SFDRs are available, external memory access is required only when a FQ reaches a length of more than five frames; this is a side benefit, not the motivation of the structure.
- If no SFDRs are available at a given time, repeated reads of the same head PFDR (for dequeue) or partial CG writes into the tail PFDR (for enqueue) may be required.

The example walk-through diagrams in [Figure 3-100](#) through [Figure 3-102](#) help illustrate these basic frame queue operations.

**Initial State (Arbitrary):**

There are 10 frames on the frame queue.  
(three in the “on deck” position, and one in “recently arrived”)

**Operation #1: Dequeue Two**

## Action:

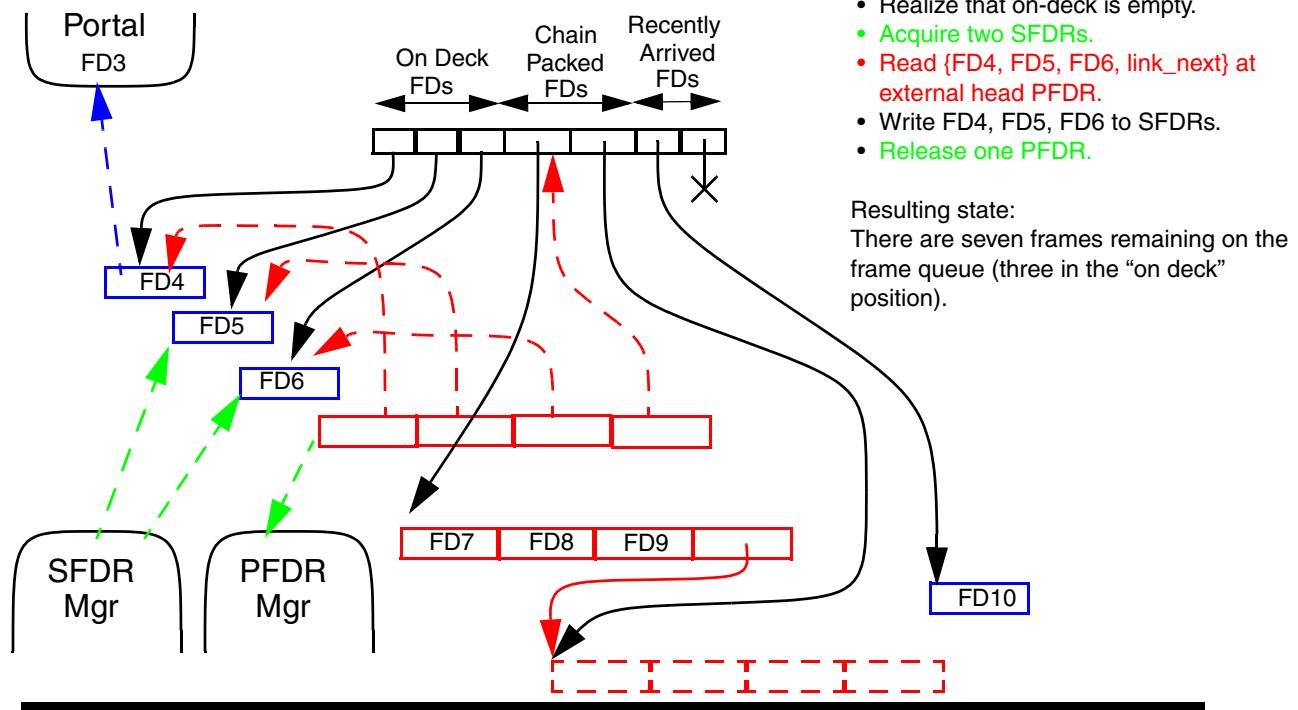
- Read and deliver FD1 and FD2 to the portal.
- Release two SFDRs.

## Resulting state:

There are eight frames remaining on the frame queue (one in the on deck position).

Figure 3-100. Example—Frame Queue Structure Walk-Through

### Operation #2: Dequeue One FD



### Operation #3: Enqueue One FD

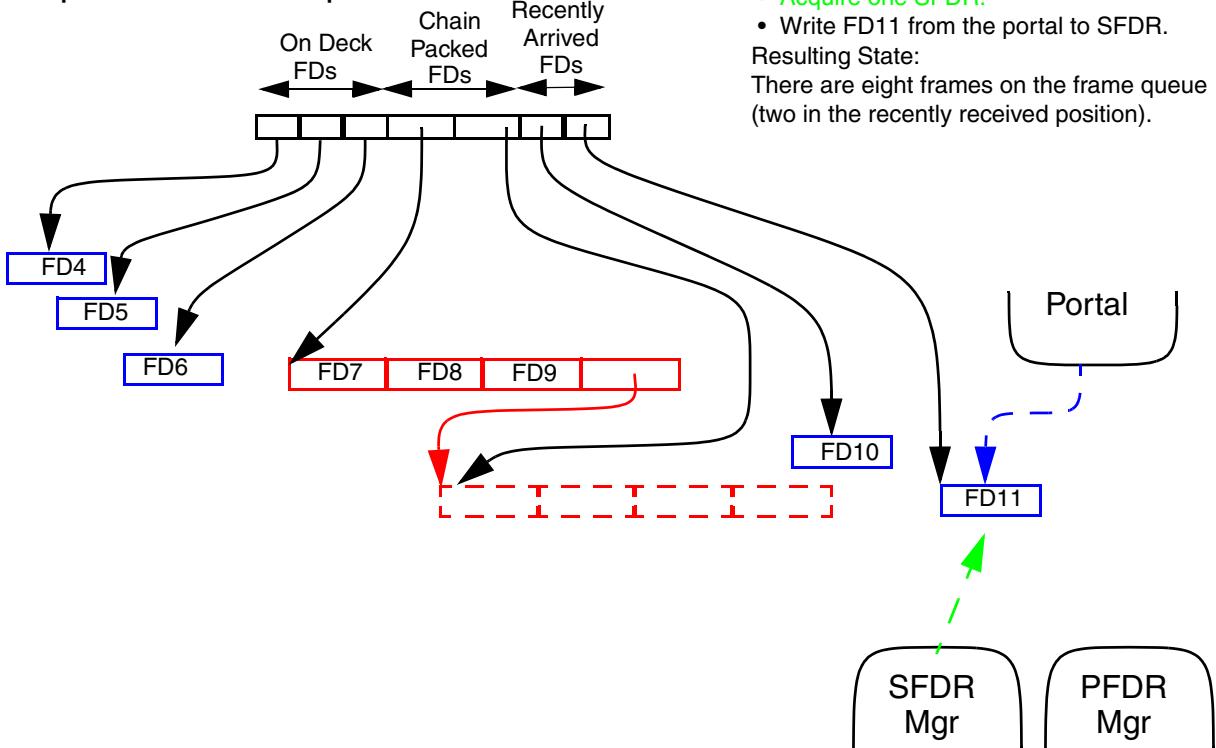


Figure 3-101. Example—Frame Queue Structure Walk-Through (continued)

## Operation #4: Enqueue One FD

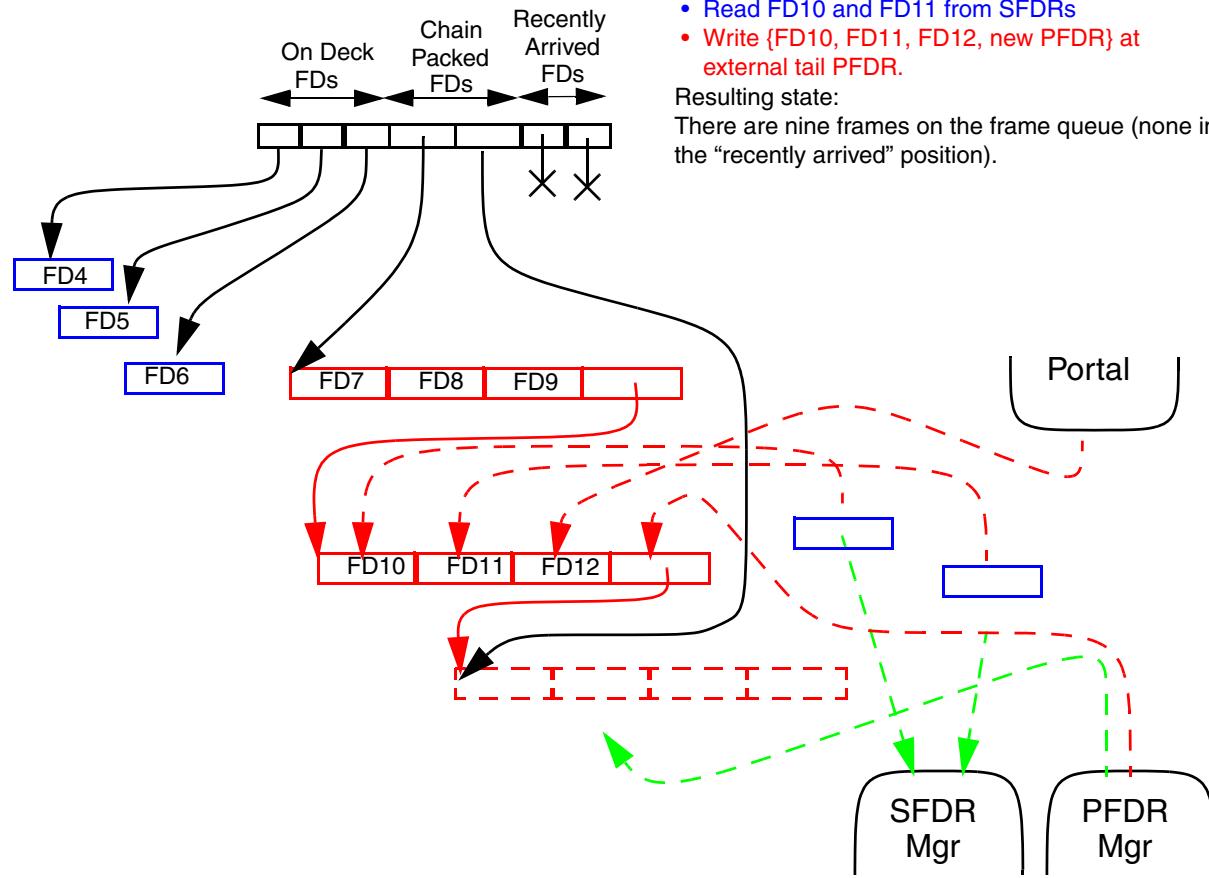


Figure 3-102. Example—Frame Queue Structure Walk-Through (continued)

### 3.3.1.4 Frame Queue Descriptors (FQDs)

#### 3.3.1.4.1 Definition of a Frame Queue Descriptor (FQD)

Frame queue descriptors (FQDs) are 64-byte data structures that are managed by the QMan only, and are not directly accessible by software or other hardware modules.

#### 3.3.1.4.2 Frame Queue Descriptors (FQDs) and the QMan

The user must initialize FQDs indirectly, and can examine them, by using specific software commands (see Section 3.3.9.5, “Frame Queue Management Commands”).

An area of main memory must be allocated by software for FQD storage. This allocated area must be communicated to the QMan by programming the FQD base and attributes registers; see Section 3.2.4.43, “Data Structure Base Address Registers.” Additionally, software should clear to 0 all of the memory that has been allocated for FQD storage, which allows the QMan to properly react with a rejection if an enqueue is attempted to a FQ that has not yet been initialized.

The FQD storage area in main memory is intended to be private to the QMan, and should therefore be configured as inaccessible to all processor cores and other hardware blocks in the system. Accesses by the QMan to FQD storage in system memory is performed as non-coherent, because the QMan always assumes that it has private ownership of this area and accesses to it do not need to be snooped by other entities in the system.

### 3.3.1.4.3 Structure of a Frame Queue Descriptor (FQD)

Even though FQDs are not directly accessible to software, they do contain some fields that must be configured by software when a FQ is initialized and brought into service. Certain fields may also be modified when a FQ is rescheduled. These software configurable fields are identified below. A number of queue management commands are available via the software portals to allow software to program these fields (see [Section 3.3.9.5, “Frame Queue Management Commands”](#)).

## Queue Manager (QMan)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31							
Reserved								FQD_LINK																														
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63							
ORPRWS	OA	ODP_SEQ															ORP_NESN																					
OLWS	ORP_EA_HSEQ															ORP_EA_TSEQ																						
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127							
ORP_EA_HPTR															ORP_EA_TPTR															ORP_EA_TPTR								
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159							
ORP_EA_TPTR															PFDR_HPTR															PFDR_HPTR								
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191							
PFDR_HPTR								PFDR_TPTR																														
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223							
CONTEXT_A (msbs)																																						
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255							
CONTEXT_A (lsbs)																																						
256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287							
CONTEXT_B																																						
288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319							
FQ_CTRL								FE	R	STATE		DEST_WQ																										
320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351							
ICS_SURP								IS	ICS_CRED																													
352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383							
BYTE_CNT																																						
384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415							
CONG_ID								FRM_CNT																														
416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447							
NRA	OAC	C	RA1_SFDR_PTR															X	IT					RA2_SFDR_PTR														
448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479							
NOD	OAL	OD1_SFDR_PTR															OD2_SFDR_PTR																					
480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511							
NPC			OD3_SFDR_PTR															TD_MANT					TD_EXP															

Figure 3-103. Frame Queue Descriptor (FQD) Format

Table 3-98. Frame Queue Descriptor (FQD) Format Description

Field	Software Config?	Description
0-7	—	Reserved
8-31 FQD_LINK	—	Link to the next FQD in a queue of FQDs, used for Work Queues.

**Table 3-98. Frame Queue Descriptor (FQD) Format Description (continued)**

<b>Field</b>	<b>Software Config?</b>	<b>Description</b>
32-34 ORPRWS	Yes	<p>Order restoration point (ORP) restoration window size 3-bit coded value</p> <ul style="list-style-type: none"> <li>0 Window size is 32 frames.</li> <li>1 Window size is 64 frames.</li> <li>2 Window size is 128 frames.</li> <li>3 Window size is 256 frames.</li> <li>4 Window size is 512 frames.</li> <li>5 Window size is 1024 frames.</li> <li>6 Window size is 2048 frames.</li> <li>7 Window size is 4096 frames.</li> </ul> <p>See <a href="#">Section 3.3.13, “Order Restoration,”</a> for a description of this and the other ORP related fields resident in the FQD.</p>
35 OA	Yes	<p>ORP auto advance NESN window size</p> <ul style="list-style-type: none"> <li>0 NESN auto advance is disabled.</li> <li>1 Auto advance NESN window size is the same as the ORP restoration window size configured in the ORPRWS field.</li> </ul>
36-49 ODP_SEQ	—	<p>Order definition point sequence number</p> <p>Returned in the dequeue response for frames from this FQ. Increments for every frame dequeued from this FQ.</p>
50-63 ORP_NESN	—	<p>Order restoration point next expected sequence number.</p> <p>Sequence number of the next frame expected at this ORP. If the next frame enqueued through this ORP carries this sequence number, it is enqueued without delay. If the next frame enqueued through this ORP does not carry this sequence number, its enqueue is delayed and it is added to the order restoration list (ORL).</p>
64-65 OLWS	Yes	<p>ORP acceptable late arrival window size</p> <ul style="list-style-type: none"> <li>0 Disabled. Late arrivals are always rejected.</li> <li>1 Window size is 32 frames.</li> <li>2 Window size is the same as the ORP restoration window size configured in the ORPRWS field.</li> <li>3 Window size is 8192 frames. Late arrivals are always accepted.</li> </ul>
66-80 ORP_EA_HSEQ	—	<p>Order restoration point early arrival head sequence number</p> <p>Bit 66: NLIS: not last in sequence</p> <p>This bit is used to delineate multiple fragments from an original single frame. If asserted, indicates that more fragments using the same sequence number are coming in later ORL records. This bit must be negated in the ORL record which carries the last fragment of the original frame, or in an ORL record which carries a non-fragmented frame.</p> <p>Bits 67-80: Sequence number of the frame at the head of the ORL.</p>
81-95 ORP_EA_TSEQ	—	<p>Order restoration point early arrival tail sequence number</p> <p>Bit 81: NLIS: not last in sequence.</p> <p>See description of ORP_EA_HPTR[NLIS].</p> <p>Bits 82-95: Sequence number of the frame at the tail of the ORL.</p>
96-119 ORP_EA_HPTR	—	<p>Order restoration point early arrival head pointer</p> <p>Pointer to the ORL Record containing the frame at the head of the ORL.</p> <p>If the pointer value is 0xFF_FF00 to 0xFF_FFFF, the ORL record is stored in internal memory, otherwise it is stored in external memory; see <a href="#">Section 3.3.13, “Order Restoration,”</a></p>

**Table 3-98. Frame Queue Descriptor (FQD) Format Description (continued)**

Field	Software Config?	Description
120-143 ORP_EA_TPTR	—	Order restoration point early arrival tail pointer Pointer to the ORL Record containing the frame at the tail of the ORL. If the pointer value is 0xFF_FF00 to 0xFF_FFFF, the ORL record is stored in internal memory, otherwise it is stored in external memory; see <a href="#">Section 3.3.13, “Order Restoration.”</a>
144-167 PFDR_HPTR	—	Packed frame descriptor record (PFDR) head pointer Pointer to the first record on the linked list of PFDRs for this FQ. The PFDR at the head of the list contains 1-3 FDs, which normally follow the “on-deck” FDs within the FQ.
168-191 PFDR_TPTR	—	Packed frame descriptor record (PFDR) tail pointer Pointer to the last record on the linked list of PFDRs for this FQ. The tail PFDR may contain 0, 1, or 2 valid FDs. It is never filled (never contains 3 valid FDs) because a new empty PFDR is always attached at the same time as the last FD of a PFDR is written. Under ideal circumstances (where SFDRs can always be allocated) there are always 0 valid FDs in the tail PFDR. In these circumstances, 2 recently arrived FDs (stored in SFDRs and indexed by RA1_SFDR_PTR and RA2_SFDR_PTR), a currently arriving FD, and a link to the new empty tail PFDR are all written into the current tail PFDR in a single 64 byte write. In cases where an SFDR could not be allocated for a recently arrived FD, 1 or 2 FD will be written directly into the tail PFDR instead of going into RA1_SFDR_PTR and RA2_SFDR_PTR. The IT bit is used to track whether the recently arrived FDs are currently stored in SFDRs or in the tail PFDR.
192-255 CONTEXT_A	Yes	Frame queue context A CONTEXT_A is a FQ attribute, it is initialized by software and is stored in the FQD. When the QMan dequeues a frame from a FQ it has access to the context field, and provides this value in the dequeue response in the DCPs. CONTEXT_A is not made available in the dequeue response in a software portal, however, this field may be used for stashing of dequeued frame data and FQ context if this feature is enabled in the FQ_CTRL field; see <a href="#">Section 3.3.8.8, “Dequeued Frame Data, Annotation, and Context Stashing.”</a> For FQs serviced by hardware modules (through DCPs), or when the “Context_A used for stashing” bit in FQ_CTRL is set to 0, this context value is treated as an opaque quantity by the QMan (that is, the QMan does not interpret the context value in anyway).
256-287 CONTEXT_B	Yes	Frame queue context B Context B is a FQ attribute. It is a 32-bit value initialized by software and is stored in the FQD. When the QMan dequeues a frame from a FQ, it has access to this context field and passes it in the dequeue response.

**Table 3-98. Frame Queue Descriptor (FQD) Format Description (continued)**

Field	Software Config?	Description
288-298 FQ_CTRL	Yes	<p>Frame queue control</p> <p>These bits are used to control the behavior of the FQ during enqueue and dequeue operations.</p> <p>Bit 288: Congestion_Group_Enable</p> <ul style="list-style-type: none"> <li>0 WRED and CS tail drop congestion management disabled for this FQ, CONG_ID is not used.</li> <li>1 WRED and CS tail drop congestion management enabled for this FQ, CONG_ID identifies the congestion group to which this FQ belongs.</li> </ul> <p>Bit 289: Tail_Drop_Enable</p> <ul style="list-style-type: none"> <li>0 FQ Tail drop congestion management disabled for this FQ, TD_EXP and TD_MANT are not used.</li> <li>1 FQ Tail drop congestion management enabled for this FQ, TD_EXP and TD_MANT specify the threshold used for tail drop.</li> </ul> <p>Bit 290: ORP enable</p> <ul style="list-style-type: none"> <li>0 Order restoration using this FQID as an ORP is disabled.</li> <li>1 Order restoration using this FQID as an ORP is enabled.</li> </ul> <p>Bit 291: Context_A used for stashing..</p> <ul style="list-style-type: none"> <li>0 The Context_A field in this FQD is not used for stashing, and is not interpreted by QMan. Frame data, annotation, and context stashing is disabled for this FQ.</li> <li>1 The Context_A field in this FQD is used for stashing, and is interpreted by the QMan when a dequeue destined to a software portal occurs from this FQ (see <a href="#">Section 3.3.8.8.3, “FQD Context_A Field used for dequeued Frame Data, Annotation, and Context Stashing control.”</a>)</li> </ul> <p>Bit 292: CPC stash enable</p> <p>If this bit is asserted (and the FQD_AR[SE] bit is also asserted), 64 byte writes of this FQD will be signaled with an explicit request to stash into the CPC. See also <a href="#">Section 3.3.19.3, “CPC Stashing of QMan Private Data Structures.”</a></p> <p>Bit 293-294: Reserved</p>

**Table 3-98. Frame Queue Descriptor (FQD) Format Description (continued)**

Field	Software Config?	Description
288-298 FQ_CTRL (continued)	Yes	<p>Frame queue control      These bits are used to control the behavior of the FQ during enqueue and dequeue operations.</p> <p>Bit 295: Force_SFDR_Allocate.          If this bit is asserted in the FQD, it causes the priority mechanism on SFDR allocation (see <a href="#">Section 3.2.4.20, “SFDR Configuration Register (SFDR_CFG)”</a>) to be overridden independent of the destination WQ of this FQ, so to the SFDR manager this FQ would always appear as a high priority FQ. This can be used on a few high-bandwidth FQ to improve overall system performance.</p> <p>Bit 296: Avoid_Blocking          This bit is only valid if Hold_Active and FE are both 0, if Hold_Active = 1 or if a “Force Eligible FQ” command has been issued on this FQ then Avoid_Blocking has no effect. If 1, indicates that this FQ should be rescheduled from the Active state to avoid blocking when the DQRR is full. Normally, if a FQ has not yet met its intra-WQ class scheduling consumption allowance, it will remain Active even if DQRR becomes full, waiting for more room in DQRR to become available before dequeuing more frames from the same FQ. This bit will cause the FQ to be rescheduled if DQRR becomes full, even if its consumption allowance has not yet been met (which will cause an addition to the surplus in ICS_SURP for this FQ).</p> <p>When to use the Avoid_Blocking bit:          - Should be used only on a FQ whose DEST_WQ is in a pool channel.          - Should be used only on a FQ that is expected to carry more traffic than a single core can handle.          In such a case, use of the Avoid_Blocking bit in the FQD will allow the FQ to be suspended from a software portal, and rescheduled to another software portal servicing the same pool channel, when the DQRR of the first portal becomes full.</p> <p>Bit 297: Hold_Active.          If 1, indicates that this FQ should be placed in the Held Active state in a portal when selected for dequeue. Otherwise, the FQ will be placed in the Active state.          Applies to any FQ destined to a software portal. Also applies to any FQ destined to a DCP used by a hardware module which has the ability to release held active FQs, such as DCE. In a FQ destined to any DCP used by a hardware module which does not support held active FQs, such as FMan or SEC, this bit is unused and has no effect. <a href="#">Section 3.3.10, “Direct Connect Portals (DCPs)”</a> lists the connectivity between QMan and the hardware modules which use a DCP.</p> <p>Bit 298: Lock_in_Cache.          If 1, indicates that the FQ is not subject to eviction from FQD cache. Once it makes it into the FQD cache, this FQ will remain there until it is retired. See also <a href="#">Section 3.3.12.1, “FQD cache operation summary.”</a></p>
299 FE	—	<p>Force eligible pending      When a “Force Eligible FQ” command is issued, this bit is set to indicate the need to move this FQ to the Held Active state when selected for a scheduled dequeue. See <a href="#">Section 3.3.9.5.4, “Alter FQ State Commands.”</a></p>
300 R	—	<p>Retirement pending      When a “Retire FQ” command is issued but the FQ is in a state that cannot be changed immediately, this bit is set to indicate the need to move to the retired state as soon as possible.</p>

**Table 3-98. Frame Queue Descriptor (FQD) Format Description (continued)**

Field	Software Config?	Description
301-303 STATE	—	FQ state. See also <a href="#">Section 3.3.1.5, “Frame Queue State.”</a> 0 Out Of Service 1 Retired 2 Tentatively Scheduled 3 Truly Scheduled 4 Parked 5 Active, Held Active, or Held Suspended 6, 7 are not used.
304-319 DEST_WQ	Yes	Destination WQ See <a href="#">Section 3.3.2.4, “Work Queue Channel Assignments,”</a> for a description of the channel number assignments used in this field. Bits 304-316: Channel number Bits 317-319: WQ number within the channel (0 to 7)
320-335 ICS_SURP	—	Intra-WQ class scheduling surplus or deficit This context determines the eligible number of bytes to consume from the frame queue for the current dequeue event. See <a href="#">Section 3.3.7.3, “Intra-WQ Class Scheduling.”</a>
336 IS	—	Intra-WQ class scheduling surplus or deficit identifier Determines whether the ICS_SURP field contains a surplus (IS = 0) or a deficit (IS = 1).
337-351 ICS_CRED	Yes	Intra-WQ class scheduling credit This context determines the eligible number of bytes to consume from the frame queue for the current dequeue event. See <a href="#">Section 3.3.7.3, “Intra-WQ Class Scheduling.”</a>
352-383 BYTE_CNT	—	Frame queue byte count Total number of bytes in all frames on this frame queue.
384-391 CONG_ID	Yes	Congestion group ID If the Congestion_Group_Enable bit is set in the FQ_CTRL field, this field identifies the Congestion Group to which this FQ belongs. See <a href="#">Section 3.3.14, “Congestion Management and Avoidance.”</a> Bit 0-7: Index (0 to 255) of the congestion group to which this FQ belongs
392-415 FRM_CNT	—	Frame queue frame count Total number of frames on this frame queue
416-417 NRA	—	Number of recently arrived frames at the tail of the FQ Valid values are 0 to 2. The FD of these frames are stored either in SFDR indexed by RA1_SFDR_PTR and RA2_SFDR_PTR, or in the tail PFDR indexed by PFDR_TPTR. The IT bit tracks whether the 1 or 2 recently arrived FDs are currently stored in SFDRs or in the tail PFDR.

**Table 3-98. Frame Queue Descriptor (FQD) Format Description (continued)**

Field	Software Config?	Description
418-419 OAC	Yes	<p>Overhead accounting control</p> <p>Bit 418: Overhead accounting enable for Intra-WQ class scheduling (1 = enabled, 0 = disabled).</p> <p>When enabled, the OAL value from the FQD is added to the actual length of each dequeued frame when performing intra-WQ class scheduling calculations.</p> <p>See <a href="#">Section 3.3.7.3, “Intra-WQ Class Scheduling.”</a></p> <p>Bit 419: Overhead accounting enable for Congestion Group (1 = enabled, 0 = disabled).</p> <p>When enabled, and if Congestion_Group_enable is asserted in the FQ_CTRL field, the OAL value from the FQD is added to the actual length of each enqueued and dequeued frame when updating congestion group byte counts, thereby affecting WRED and CS tail drop threshold comparisons.</p> <p>See <a href="#">Section 3.3.14, “Congestion Management and Avoidance.”</a></p> <p>For each enqueue and dequeue operation on which overhead accounting is enabled, an overhead accounted frame length (OAFL) is calculated using the actual frame length (AFL) and the OAL value from the FQD as follows:</p> <ul style="list-style-type: none"> <li>if (AFL &gt; 16383) then OAFL = AFL, overhead accounting effect is negligible</li> <li>else if (AFL + OAL &lt; 0) then OAFL = 0, negative OAFL is capped at 0</li> <li>else OAFL = AFL + OAL.</li> </ul>
420 C	—	FQD in Cache. This is a status bit that can be queried by software. 0 This FQD is currently in external memory. 1 This FQD is currently in FQD cache or in a PSCL.
421-431 RA1_SFDR_PTR	—	SFDR pointer for recently arrived frame number 1 Points to an SFDR containing the FD of a frame recently arrived at the tail of the FQ. Only valid when NRA is 1 or 2 and IT is 0.
432 X	—	FQ XOFF This bit indicates the XON/XOFF state of the FQ. 0 XON 1 XOFF It is modified by QMan as a result of FQ flow control commands received; see <a href="#">Section 3.3.6, “Frame Queue (FQ) Flow Control.”</a>
433 IT	—	Recently arrived frames In Tail PFDR 0 Recently arrived frames (NRA = 1 or 2) are stored in SFDRs. 1 Recently arrived frames (NRA = 1 or 2) are stored in the tail PFDR.
434-436	—	Reserved
437-447 RA2_SFDR_PTR	—	SFDR pointer for recently arrived frame number 2 Points to an SFDR containing the FD of a frame recently arrived at the tail of the FQ. Only valid when NRA is 2 and IT is 0.
448-449 NOD	—	Number of on-deck frames in SFDRs at the head of the FQ Valid values are 0 to 3. The FD of these frames are stored in SFDR indexed by OD1_SFDR_PTR, OD2_SFDR_PTR, and OD3_SFDR_PTR. If NOD = 0 but the FQ is not empty, then the frames at the head of the FQ are stored in the head PFDR rather than in SFDRs.
450-452, 464-468 OAL	Yes	Overhead accounting length This is an 8-bit, 2's complement value (range -128 to +127) representing a fixed per-frame overhead to be added to the actual length of a frame when performing certain calculations and/or threshold comparisons using frame length. Use of this OAL value is controlled by the OAC bits described above.

**Table 3-98. Frame Queue Descriptor (FQD) Format Description (continued)**

Field	Software Config?	Description
453-463 OD1_SFDR_PTR	—	SFDR pointer for on-deck frame number 1 Points to an SFDR containing the first on-deck FD at the head of the FQ. Only valid when NOD is 1, 2, or 3.
469-479 OD2_SFDR_PTR	—	SFDR pointer for on-deck frame number 2 Points to an SFDR containing the 2nd on-deck FD at the head of the FQ. Only valid when NOD is 2 or 3.
480-481 NPC	—	Number of previously consumed frames in the head PFDR Valid values are 0 to 2, are non-zero only when NOD is 0. This field supports consuming a subset of the FDs from within the head PFDR when SFDRs are not available to be allocated. Under ideal circumstances (where SFDRs can always be allocated) this value is constantly zero because all 3 FDs from the head are moved into on-deck SFDRs, and the PFDR is deallocated.
482-484	—	Reserved
485-495 OD3_SFDR_PTR	—	SFDR pointer for on-deck frame number 3 Points to an SFDR containing the 3rd on-deck FD at the head of the FQ. Only valid when NOD is 3.
496-498	—	Reserved
499-506 TD_MANT	Yes	Tail drop threshold exponent and mantissa The threshold in bytes at which tail drop from this FQ occurs is $TD\_MANT * 2^{TD\_EXP}$ .
507-511 TD_EXP	Yes	Tail drop is only enabled if the FQ_CTRL[Tail_Drop_enable] bit is 1. There is no hysteresis on tail drop congestion management, if the byte count of the FQ has exceeded the tail drop threshold and Tail Drop is enabled then all subsequent enqueues are rejected; see also <a href="#">Section 3.3.14, “Congestion Management and Avoidance.”</a> These fields must be programmed such that $TD\_MANT * 2^{TD\_EXP} \leq 0xE0000000$

### 3.3.1.5 Frame Queue State

There are many different FQ states, which include the following:

- Out of Service State—The FQ is not assigned for any specific purpose. It may be in held in a free pool or be in the process of being put into service. All commands to this FQ fail, except for the Initialize command.
- Parked State—The consumer (rather than the QMan) has scheduling control of the FQ. Consumers may dequeue from FQs in this state, and producers may also enqueue to them.
- Tentatively Scheduled State—The consumer gives scheduling control of the FQ to the QMan, but the eligibility criteria (for True Scheduling) is not currently met. Producers may enqueue to FQs in this state.
- Truly Scheduled State—The consumer gives scheduling control of the FQ to the QMan and the eligibility criteria is currently met. Therefore, the FQ is queued in a WQ and is waiting to become the active or held-active FQ on a portal. Producers may enqueue to FQs in this state.
- Active State—The FQ is selected for dequeue on a specific portal and is released from the portal when it is no longer eligible for dequeue. Consumers can dequeue from FQs in this state, and

producers can also enqueue to them. While in the active state, a FQ is temporarily coupled to a single portal or sub-portal and cannot be scheduled or consumed via any other portal.

- Held Active State—The FQ is selected for dequeue on a specific portal and is held in the portal when it is no longer eligible for dequeue. It is released from the portal after all frames dequeued from it are consumed from the portal.

Consumers can dequeue from FQs in this state, and producers may enqueue to them. While in this state, a FQ is temporarily coupled to a single portal or sub-portal and cannot be scheduled or consumed via any other portal.

- Held Suspended State—An FQ moves to this state from the Held Active state when it is no longer eligible for dequeue, and all frames previously dequeued from it have not yet been consumed from the portal. It is released from the portal after all frames dequeued from it are consumed from the portal.

No further dequeues can occur from FQs in this state, however, producers can still enqueue to them. While in this state, a FQ is temporarily coupled to a single portal or sub-portal and cannot be scheduled or consumed via any other portal.

- Retired State—A precursor to the Out of Service State. No enqueues succeed to a FQ in this state. An FQ is placed in this state upon command at the earliest possible opportunity. When a “retire” command is issued, but the FQ is in a state that cannot be changed immediately, a “retirement pending flag” is set to indicate the need to move to the retired state as soon as possible.

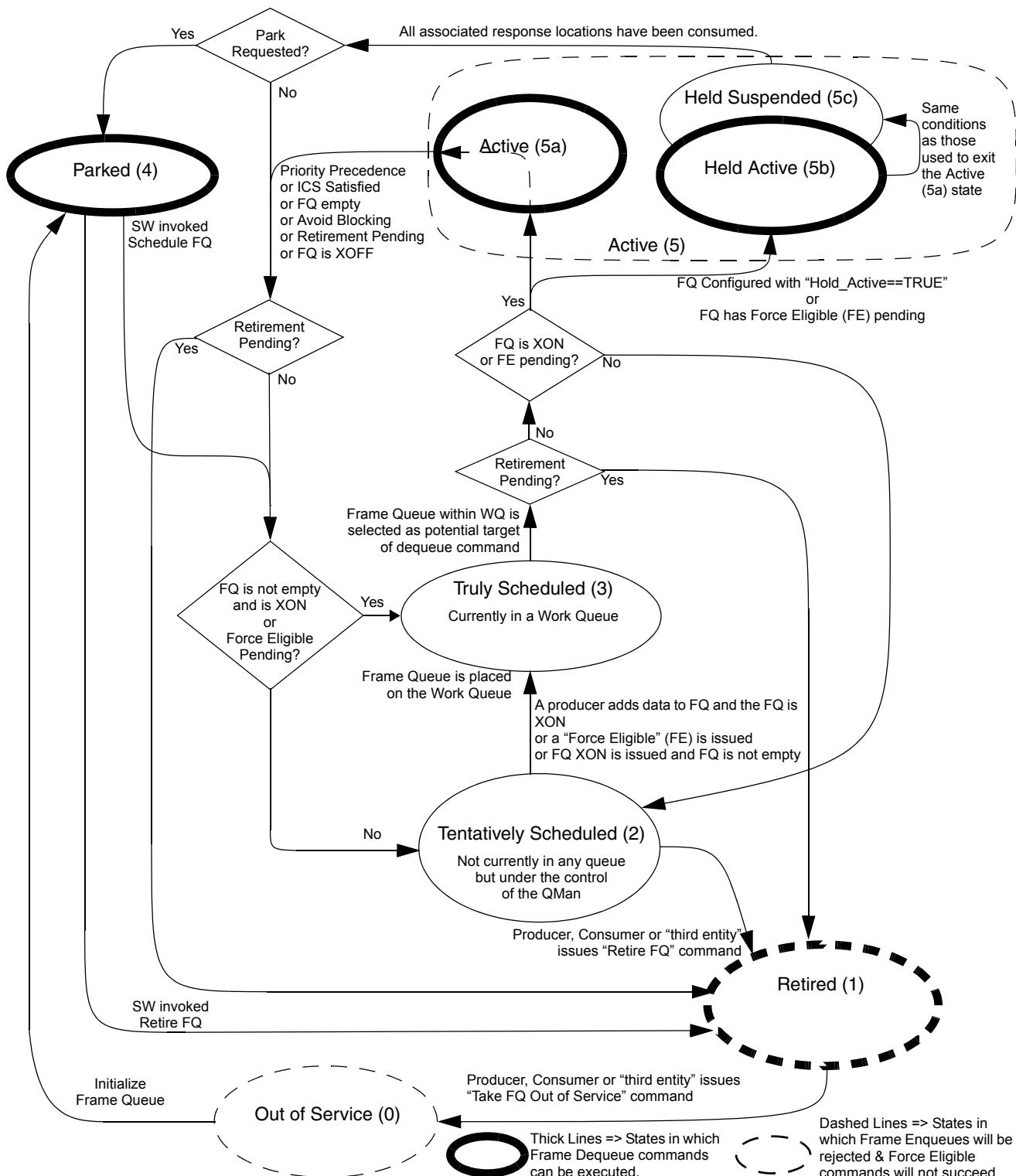


Figure 3-104. Frame Queue State Diagram

## 3.3.2 Work Queues (WQs) and Channels

### 3.3.2.1 Definition of a Work Queue (WQ)

When the frames on a FQ are ready to be processed, the FQ is enqueued onto a work queue (WQ). A WQ consists of a linked list of FQs, and adding a FQ to a WQ is done by linking the FQD to the tail of the WQ. The WQ on which a FQ is to be placed (i.e. its destination WQ) is specified when the FQ is initialized by software, see [Section 3.3.9.5.1, “Initialize Frame Queues \(FQ\).”](#) Some key facts:

- There is no requirement that a FQ be on a WQ.
- If no frames are queued on a FQ, it is not on a WQ.
- Normally, when frames are added to an otherwise empty FQ, it is enqueueued on a WQ.
- FQs are configured to be enqueueued on one and only one WQ.
- The FQ’s destination WQ is configured when the FQ is initialized, but can be subsequently changed by software.

### 3.3.2.2 Definition of a Channel

WQs are organized into channels. The WQs within a channel have a priority relative to each other. Each channel consists of eight WQs, and thus, there are eight possible priorities in a channel.

In general, a given channel is serviced by a single type of entity. That is, a given channel might be serviced by crypto lookaside accelerators and another channel by a processor core. Serviced in this context refers to the entity which dequeues frames from the channel’s WQs. [Section 3.3.2.4, “Work Queue Channel Assignments,”](#) specifies the mapping between channels and the entities that service them.

Dequeuing a frame implies that the QMan selected a WQ from its non-empty WQs, and removed the first frame from the first FQ on that WQ. The algorithm used to select which WQ to service is described in [Section 3.3.7, “Dequeue Scheduling.”](#)

The QMan also allows entities to specify which WQ should be selected from a channel when a dequeue command is issued. This allows the consumers of frames to implement any selection algorithm desired.

### 3.3.2.3 Types of Channels

There are two types of WQ channels defined in the QMan:

- Dedicated channels, which are always serviced by a single entity
- Pool channels, which are serviced by a pool of like entities, such as a pool of processor cores

From the point of view of enqueueing, there is no difference between a dedicated channel and a pool channel.

From the point of view of dequeuing, the only difference between dedicated and pool channels is that pool channels may have multiple entities dequeuing from them at the same time, and the QMan must arbitrate between and serialize these accesses to the queues. WQs on pool channels are serviced by like modules, so a simple, first-come-first-served algorithm can be used to select which entity should be provided the next frame.

Each software portal is assigned a dedicated channel. There are also several pool channels that are shared by all software portals and from which all processor cores can draw frames. When packets are to be load-spread across processor cores, they are enqueued on FQs that are destined to a pool channel, which can then be serviced by a pool of processor cores.

### 3.3.2.4 Work Queue Channel Assignments

In general, a WQ channel is dedicated to a specific portal, with the exception of the pool channels, which can be serviced by any of the software portals.

Each FQ in the system is configured (when the FQ is initialized or rescheduled) to be enqueued onto one of these WQs when its enqueue eligibility criteria are met.

The assignment of WQ channels to portals is shown in [Table 3-99](#). The resulting destination WQ ID, which are used in the DEST\_WQ field of the FQD of every FQ, are also shown in the table. If the QMan encounters a FQ configured to be enqueued onto any reserved channel, the FQ is instead enqueued onto a default WQ reserved for such errors (see [Section 3.2.4.23, “WQ Default Enqueue WQID Register \(WQ\\_DEF\\_ENQ\\_WQID\)”](#)), and an error interrupt is asserted (if enabled).

**Table 3-99. Work Queue (WQ) Channel Assignments in the QMan**

Channel Number	Destination WQ ID	Description
000h to 3FFh	0000h to 1FFFh	Dedicated channels serviced by Software Portals. Channel 000h is serviced exclusively by software portal 0 Channel 001h is serviced exclusively by software portal 1 etc... Channels corresponding to non-existent software portals are reserved.
400h to 40Fh	2000h to 207Fh	Pool channels that can be serviced by any of the Software Portals. Each of these channels can be serviced by one or more of the software portals, and each software portal may service its dedicated channel as well as one or more pool channels. Channel 400h is reserved. Channel 401h is assigned to Pool Channel 1. ... Channel 40Fh is assigned to Pool Channel 15 if there are 15 pool channels, otherwise this channel is reserved. Channels corresponding to non-existent pool channels are reserved.

**Table 3-99. Work Queue (WQ) Channel Assignments in the QMan (continued)**

Channel Number	Destination WQ ID	Description
800h to 80Fh	4000h to 407Fh	<p>Dedicated channels serviced by Direct Connect Portal 0, connected to FMan 0. These 16 channels are assigned in incrementing order to each sub-portal (SP) in the portal; see also <a href="#">Section 3.3.10, “Direct Connect Portals (DCPs)”</a>:</p> <ul style="list-style-type: none"> <li>Channel number 800h is assigned to SP0 in the portal</li> <li>...</li> <li>Channel number 80Fh is assigned to SP15 in the portal.</li> </ul> <p>Note that when configured in FQ/WQ scheduling mode, SP 0 and SP1 contain additional performance optimizations compared to the other SPs, therefore FMan should always be configured to use SP 0 and/or SP 1 for its 10 GE port egress traffic if the SP used by the 10 GE ports are configured for FQ/WQ scheduling mode.</p> <p>Any SP in this portal which is configured in CEETM scheduling mode (see <a href="#">Section 3.3.20.1, “CEETM Overview”</a>) does not use the WQ channel assigned to it in this table, it will instead use the CEETM LNI for which it has been configured, see <a href="#">Section 3.3.9.7.11, “CEETM Sub-Portal Mapping Configure.”</a></p>
840h	4200h to 4207h	Dedicated channel serviced by Direct Connect Portal 2, connected to Security Engine

### 3.3.3 Enqueue Operations

An entity, such as a network interface or a processor core, enqueues a frame by passing the frame descriptor (FD), and the ID of the FQ onto which the frame should be placed, to the QMan. Issue an Enqueue command from the entity to QMan to accomplish this. When responding to the receipt of an Enqueue command, the QMan appends this frame to the tail of the FQ that was specified. If the FQ is not already enqueued on its WQ, the QMan adds it to the tail of the WQ.

Enqueue commands are issued to the QMan via either a software portal (see [Section 3.3.8, “Software Portals”](#)) or a direct connect portal (DCP) (see [Section 3.3.10, “Direct Connect Portals \(DCPs\)”](#)).

### 3.3.4 Dequeue Operations

#### 3.3.4.1 Dequeue Availability

Whenever a WQ in a channel contains one or more FQs containing one or more frames (which means it is “non-empty”), the QMan signals the entity servicing that channel that a frame is available to be dequeued. The mapping between channels and entities assigned to service them is shown in [Section 3.3.2.4, “Work Queue Channel Assignments.”](#)

For entities using a direct connect portal (DCP), a dedicated signal is used to communicate the dequeue availability status of each of the channels serviced by that entity.

For software portals, dequeue availability is signaled using an interrupt. Each software portal is assigned a dedicated interrupt line for this purpose. Each software portal can service its dedicated WQ channel as well as any of the pool channels.

The interrupt enable register (see [Section 3.2.3.22, “QCSP Interrupt Enable Registers \(QCSPi\\_IER\)”](#)) is used to determine which of the pool channels signal dequeue availability to each software portal. Note that it is valid for more than one software portal to service the same pool channel.

### 3.3.4.2 Dequeue Commands

Direct connected hardware modules pull frames from queues by issuing dequeue commands to the QMan. In software portals, a single dequeue command can be repeated autonomously, allowing the QMan to push dequeued frames at software via the portal. These dequeue commands are issued via the same portals as the enqueue commands; however, in general, portals contain separate resources allowing them to handle enqueues and dequeues simultaneously. The QMan uses hierarchical queueing, so a dequeue command results in a FQ being dequeued from a WQ, and the frame(s) at the head of the FQ being dequeued from that FQ. Up to three frames from the same FQ can be requested in one dequeue command, and the frames are returned to the requesting entity via the same portal.

The QMan supports the following different types of frame dequeue commands:

- The requester specifies in the dequeue command which channel it wishes to service. The channels that are available for service are indicated using the dequeue availability signaling described in [Section 3.3.4.1, “Dequeue Availability.”](#) Hardware modules always service one or more dedicated channels, and they specify one of their dedicated channels as the one to be serviced. Software running on cores can specify multiple channels in the dequeue command (for example, their dedicated channel and any or all of the pool channels), and the QMan selects which of these channels to service. In both cases, after a particular channel is chosen for service, the QMan selects a WQ within that channel (see [Section 3.3.7, “Dequeue Scheduling”](#)), identifies the FQ at the head of that WQ, and dequeues the frame(s) at the head of that FQ.
- The requester specifies the channel and one of the WQs within that channel that it wishes to service. This allows requesters, such as software running on a core or a network interface, to implement more complex WQ selection algorithms than those supported by the QMan.
- The requester directly specifies the FQ from which it wishes to receive frames. This dequeue command type bypasses all of the queueing provided by the QMan, and is intended for use with dedicated FQs that are used for out-of-band communication between processing entities.

See [Section 3.3.8.2.3, “DQRR Dequeue Commands,”](#) for more detail on the available dequeue command types.

### 3.3.4.3 Dequeueing in the Held Active State

After the head frame is removed from the FQ and the dequeue response is returned, the FQ may be held in the portal in anticipation of more frames being dequeued from it; in this case, the FQ is said to be “held active” in that portal. Whether the FQ remains active or not is dependent on the configuration of the FQ. When initializing a FQ, software must determine whether that FQ is held active or not after a dequeue.

One consequence of holding the FQ is that no other block is able to receive and process frames from that FQ while it is held active in a portal, but it helps ensure packet ordering in the system.

If a previous FQ is held active, the next frame(s) can be dequeued from that FQ without selecting another FQ. Dequeueing from the active FQ in a portal is an option on the dequeue command, and permits hardware accelerators or software to selectively process multiple frames from the same flow sequentially, and can be used to improve processing efficiency.

An entity using a portal can explicitly release an active FQ in that portal by requesting the QMan to do so, or it can implicitly release an active FQ because it requests the QMan to select a new FQ from a channel. Regardless of whether the release is automatic (as a result of FQ configuration) or explicit (as a result of a request performed by a portal command), the FQ may be rescheduled, and as a result, returned to its WQ. When rescheduling a FQ, the QMan evaluates the eligibility of the FQ to be placed on a WQ, and as a result, the FQ may be left off the WQ and placed in a Tentatively Scheduled state if it is currently empty, or it may be placed on its WQ if it contains more frames to be dequeued. FQ states and the flow between those states is further described in [Section 3.3.1.5, “Frame Queue State.”](#)

### **3.3.5 Traffic Class (TC) Flow Control**

Traffic class flow control is available only on the DCPs, and is not available on software portals.

The traffic class flow control feature allows a hardware module connected to the QMan via a DCP to disable (XOFF) and re-enable (XON) any traffic class (TC) for dequeue within any of the channels assigned to that portal. When the state of a TC is XOFF (disabled), no frames can be dequeued from that TC. Note that enqueues are not affected by the XON/XOFF state of a TC.

The XON/XOFF state of a DCP’s TCs is determined by flow control commands that are sent from a direct connected hardware module to the QMan. On a DCP which supports multiple sub-portals, flow control commands can be applied individually to each sub-portal.

The QMan internally stores the new XON/XOFF state of each TC every time a TC flow control command is received. The initial state of all TCs is XON (enabled for dequeue); therefore, any direct-connected hardware modules that do not support the TC flow control feature default to having all TCs enabled.

The QMan supports a total of 8 traffic classes, numbered TC0 to TC7.

If the sub-portal on which a traffic class flow control command is received is configured in FQ/WQ scheduling mode, then a TC is equivalent to a WQ, and flow control on  $TC_n$  is applied to  $WQ_n$  in a WQ channel ( $n = 0$  to 7). When a traffic class flow control command is received on a particular sub-portal of a DCP, flow control from the 8 TCs in the command is applied to the 8 WQs in the WQ channel used by that sub-portal. The assignment of channels to DCP sub-portals is defined in [Section 3.3.2.4, “Work Queue Channel Assignments.”](#)

If the sub-portal on which a traffic class flow control command is received is configured in CEETM scheduling mode, then traffic class flow control is applied to the class queues in all class queue channels attached to the LNI used by that sub-portal, see [Section 3.3.20.13, “CEETM traffic class flow control.”](#)

When a channel-based dequeue command ( $SU = 0, SS = 0$ ) is received, any TC in the requested channel whose state is XOFF does not participate in the class scheduler selection (see [Section 3.3.7.2, “WQ Class Scheduling”](#) or [Section 3.3.20.6, “CEETM Class Queue Channel and Class Scheduler”](#)), and is not selected for dequeue.

When a specific WQ dequeue command is received ( $SU = 0, SS = 1$ ), and if the state of the specified WQ is XOFF, a null dequeue response (no frames dequeued) is returned; therefore, it appears as if the specified WQ is empty. Note that specific WQ dequeue commands do not apply when CEETM scheduling mode is used.

### 3.3.6 Frame Queue (FQ) Flow Control

The FQ flow control feature allows software or a hardware module connected to the QMan via a DCP to disable (XOFF) and re-enable (XON) for dequeue any FQ in the system, although in practice, only FQs that are consumed by a particular portal (software or DCP) should be flow-controlled by that portal. When the state of a FQ is XOFF (disabled), no frames can be dequeued from that FQ (with some exceptions; see [Section 3.3.6.2, “FQ Flow Control Exceptions”](#)). Enqueues are not affected by the XON/XOFF state of a FQ.

The XON/XOFF state of all FQs is determined by flow control commands that are sent from a direct-connected hardware module and/or from a software portal to the QMan. Software issues FQ flow control commands via the CR in a software portal (see [Section 3.3.8.4, “Management Command Register \(CR\)”](#)), using an Alter FQ State command with the appropriate FQ flow control command verb (see [Section 3.3.9.5.4, “Alter FQ State Commands”](#)).

The QMan stores the new XON/XOFF state of a FQ in the FQD (see [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\)”](#)) when a FQ flow control command is received from either a software portal or a direct-connected hardware module. The initial state of a FQ after it is initialized is XON (enabled for dequeue); therefore, any software applications or direct-connected hardware modules that do not support the FQ flow control feature default to having all FQs enabled.

#### 3.3.6.1 FQ Flow Control and FQ States

In certain states, a FQ behaves in the following ways (see [Section 3.3.1.5, “Frame Queue State,”](#) for detailed descriptions of each state):

- Out Of Service, Retired, or Parked states—The command does not take effect and an error interrupt is asserted (see QMAN\_ERR\_ISR[IFSI] in [Section 3.2.4.51, “QMan Error Interrupt Status Register \(QMAN\\_ERR\\_ISR\)”](#)).
- Switched to XOFF while it is in the Tentatively Scheduled state—The FQ remains in this state (in other words, is not placed on its destination WQ) until it is switched back to XON.
- Switched to XOFF while it is in the Truly Scheduled state (while it is on a WQ), or in the Active state (while it is active in a portal)—When this FQ is later selected as the target of a dequeue command, no dequeue occurs, and the FQ moves to the Tentatively Scheduled state.  
The response in this case depends on the type of dequeue command used. In the DQRR of a software portal configured in push mode (using SDQCR), no response is seen and this FQ does not appear in the DQRR.  
In the DQRR of a software portal configured in pull mode (using PDQCR with SU = 0), a null response (a dequeue response with STAT[3] = 0) is returned in the DQRR. In a DCP, a null response (dequeue response with 0 frames delivered) is returned.

#### 3.3.6.2 FQ Flow Control Exceptions

- An exception occurs if a Retire FQ and/or Force Eligible FQ command (see [Section 3.3.9.5.4, “Alter FQ State Commands”](#)) has been executed on the FQ as well as FQ XOFF  
In this case of multiple conflicting commands on the same FQ, Retirement Pending has first precedence, followed by Force Eligible, and then XOFF, which has lowest precedence.

In the particular case of Retirement Pending = 0, Force Eligible = 1, and XOFF = 1, when this FQ is selected for dequeue, a null dequeue response (0 frames delivered) is returned even if the FQ is not empty; this satisfies the Force Eligible request while still respecting the XOFF state of the FQ. See the FQ state diagram in [Section 3.3.1.5, “Frame Queue State”](#) for more detail on how the XON/XOFF state of a FQ affects its state transitions.

- FQ flow control does not apply to unscheduled FQs, because FQ XON/XOFF commands do not take effect on FQs in the Parked or Retired states. If an unscheduled dequeue command (using VDQCR, or using SU = 1 in PDQCR) is received for a Parked or Retired FQ, and the specified FQ is currently disabled for dequeue (XOFF), then the X bit in the FQD is immediately cleared, setting the FQ to XON, and the dequeue proceeds as usual. This can happen if, for example, a Retire FQ command is executed on a Tentatively Scheduled FQ that is currently XOFF.
- Execution of FQ flow control commands is done by the same algorithmic sequencers that process enqueue and dequeue operations. Execution of occasional FQ flow control commands is easily processed using otherwise spare cycles, and thus creates no noticeable effect; however, executing frequent FQ flow control commands on large numbers of FQs should be avoided in applications requiring very high frame rate throughput.

### 3.3.7 Dequeue Scheduling

#### 3.3.7.1 WQ Channel Scheduling

WQ Channel scheduling applies only to software portals. It is the selection of a channel to be serviced when the dequeue command used specifies a scheduled dequeue from one or more of the channels available to the software portal.

The dequeue command uses a 16-bit field to specify whether or not each of the channels available (the portal’s dedicated channel and the pool channels) is enabled to participate in the dequeue selection. The QMan’s dequeue dispatcher selects a channel, from among those enabled in the command, from which to dequeue. The dequeue dispatcher and its associated WQ channel scheduler are described in [Section 3.3.8.2.5, “DQRR Dequeue Dispatcher.”](#)

#### 3.3.7.2 WQ Class Scheduling

Use the WQ class scheduler when selecting a work queue (WQ) within a channel for a dequeue operation in both software and direct connect portals. WQs are grouped into channels, each of which contains eight WQs numbered 0 to 7. The 3 lsbs of the destination WQ ID (see [Section 3.3.2.4, “Work Queue Channel Assignments”](#)) identify the WQ number within the channel.

The WQ class scheduler is used to select the next WQ to be serviced in a channel when that channel has been selected for dequeue by a consumer. The operation of the WQ class scheduler is illustrated in [Table 3-100](#).

**Table 3-100. WQ Class Scheduler**

Work Queue # within a Channel	Assigned Priority	First Level Schedulers	Second Level Scheduler	
0	High Priority 0	—	Strict priority in the following order: High Priority 0 High Priority 1 Medium Priority Tier Low Priority Tier	
1	High Priority 1	—		
2	Medium Priority	Weighted Interleaved Round Robin between WQ 2, 3, and 4 selects one WQ in the Medium Priority Tier.	With programmable elevation of the Low Priority Tier over the Medium Priority Tier	
3	Medium Priority			
4	Medium Priority			
5	Low Priority			
6	Low Priority	Weighted Interleaved Round Robin between WQ 5, 6, and 7 selects one WQ in the Low Priority Tier.		
7	Low Priority			

The two high priority WQs have absolute strict priority over the other six WQs in a channel, and high priority 0 has absolute strict priority over high priority 1.

The different levels of schedulers are described as follows:

- First level—Weighted interleaved round robin (WIRR) is used to select one of three WQs for service in each of the medium and low priority tiers. The relative weight (range 1–8) of each of the three queues in each tier is programmable (see [Section 3.2.4.22, “Work Queue Class Scheduler Configuration Registers \(WQ\\_CS\\_CFGi\)”](#)). The number of selections made from each of the three queues in each tier is tracked using a selection counter. The WIRR algorithm used within each of the two tiers is summarized as follows:
  - All three selection counters start at 0.
  - A WQ is eligible for selection if it is non-empty and its selection counter is not greater than its programmed weight value (0–7, corresponds to relative weight of 1–8).
  - A work-conserving, round robin selection is made among all eligible WQs among the three in a tier. When a WQ from this tier is selected for dequeue (when the FQ at the head of this WQ moves to the Active or Held Active state), the selection counter for the selected WQ is incremented. The selection counters are 1 bit wider than the weight values, such that a count of 8 can be represented when the weight value is 7.
  - In any cycle in which all three WQs are not eligible, the three selection counters are reset to 0. If one or more WQ are non-empty but not eligible because their selection counters have reached their programmed weight, the reset occurs in the cycle immediately following the last selection, such that there is no dead cycle when resetting the selection counters.
- Second level—The low priority tier can be elevated over the medium priority tier to avoid starvation of the low priority tier with a programmable weight used to decide how often elevation occurs. The elevation weight is programmable, and allows the low priority tier to be elevated above

the medium priority tier one in CS\_ELEV + 1 times, where the range of CS\_ELEV values is 1–255. A CS\_ELEV value of 0 disables the elevation. An elevation pending counter is used to track the number of times that a medium priority selection is made while a low priority WQ is pending. The operation of the elevation algorithm is as follows:

- The elevation pending counter increments when a WQ is selected for dequeue from the medium priority tier while one or more WQs in the low priority tier are non-empty.
- The elevation pending counter is reset when a WQ is selected for dequeue from the low priority tier, or when all WQs in the low priority tier are empty.
- If the value of the elevation pending counter is greater than or equal to the programmed CS\_ELEV value + 1, and the two high priority WQs are empty, the scheduler selects a WQ from the low priority tier over the medium priority tier.

The QMan maintains the required scheduling state on a per-channel basis.

Note that the WQ class scheduler's history registers (used for WIRR and starvation avoidance in the mid and low priority tiers) are only updated when a scheduled dequeue from a channel is performed. When a dequeue command specifies a scheduled dequeue from a specific WQ, or an unscheduled dequeue from a specific FQ, the WQ class scheduler's history registers are not updated. Also, when a channel-based dequeue command is received, any WQ in the requested channel whose state is XOFF (see [Section 3.3.5, “Traffic Class \(TC\) Flow Control”](#)) does not participate in the WQ class scheduler selection, and is not selected for dequeue.

### 3.3.7.3 Intra-WQ Class Scheduling

Intra-WQ class scheduling is the consumption amount and scheduling of a FQ within a WQ, which involves the following:

- Determining the number of bytes of data that can be consumed from a FQ at the head of a WQ
- Rescheduling the FQ back onto the tail of the same WQ when a FQ's consumption allowance is met

Intra-WQ class scheduling is implemented using a modified deficit round robin algorithm. This algorithm is based on credit and surplus tracking on a per-FQ basis. Each FQ is assigned a static credit value that is the number of bytes assigned as eligible consumption allowance for every dequeue opportunity. A dynamic surplus value keeps track of unused consumption allowance in past dequeue opportunities for that FQ. The surplus can be positive or negative.

When a FQ is initially selected for dequeue, a new consumption allowance is calculated (consumption allowance = credit + surplus), and 1 frame or 1–3 frames (whichever is selected in the dequeue command; see [Section 3.3.8.2.3, “DQRR Dequeue Commands”](#)) is always consumed even if the number of bytes dequeued is greater than the consumption allowance. If the consumption allowance is not satisfied after the first dequeue command, the FQ remains eligible for selection by a subsequent dequeue command. 1 frame or 1–3 frames are consumed for each subsequent dequeue command executed on that FQ, with the number of bytes consumed being tracked and updated after each dequeue command executed on the same FQ. After the total number of bytes consumed from the FQ is greater than or equal to the consumption allowance that was initially calculated, the consumption allowance is satisfied and any bytes in excess of the allowance are stored as a negative surplus.

An active FQ may be released (and thus become ineligible for further dequeues) because a FQ from a higher priority WQ has become available, or because its consumption allowance is satisfied:

- If released due to priority precedence, any remaining unused consumption allowance is stored as a positive surplus, which increases the consumption allowance for this FQ the next time it is selected for dequeue.
- If released due to consumption allowance being satisfied, any consumption in excess of the allowance is stored as a negative surplus, which decreases the consumption allowance (relative to its static credit) for this FQ the next time it is selected for dequeue.

If the surplus value reaches its maximum positive or negative value, it sticks at that value without rolling over until a surplus of the opposite polarity (negative or positive) is added, which reduces the surplus below maximum. If the FQ is empty after consumption, no surplus (positive or negative) is accumulated, and the surplus value is set to 0. When a FQ transitions to the parked or retired states (see [Section 3.3.1.5, “Frame Queue State”](#)) the surplus is set to 0, and any unscheduled dequeues that occur on a parked or retired FQ do not affect the surplus.

Each FQ can be configured by software to add or subtract a fixed number of bytes to/from the actual length of each frame dequeued from that FQ when calculating the remaining consumption allowance. This can be used to account for a fixed per-frame overhead, such as media transmission overhead, when using intra-WQ class scheduling to limit the amount of data dequeued from a FQ. The Overhead Accounting Control (OAC) and Overhead Accounting Length (OAL) fields in the FQD are used to configure this behavior. Note that, if enabled, overhead accounting applies to each dequeued frame; therefore, if execution of a dequeue command results in 2 or 3 frames being dequeued, the overhead accounting length is added (or subtracted) 2 or 3 times when calculating the remaining intra-WQ class scheduling allowance after the dequeue.

During FQ initialization, software must appropriately specify the intra-WQ class scheduling credit amount (it is stored in FQD[ICS\_CRED]), and the QMan hardware must initialize the surplus value to 0. The surplus value is updated only by hardware, and the surplus field in the FQD (ICS\_SURP) is used to store and track the consumption allowance:

- When a FQ is initially selected for dequeue ->  
Consumption allowance = ICS\_CRED + ICS\_SURP, and this is stored as a new ICS\_SURP.
- When a frame is dequeued ->  
Remaining allowance = ICS\_SURP - (frame length in bytes + OAL), and this is stored as new ICS\_SURP.
- When a frame is dequeued ->  
Consumption allowance satisfied if (ICS\_SURP <= 0).

This way, the ICS\_SURP field always contains the correct surplus, so when the FQ is suspended, this field does not need to be written again,

For a system where weighted fairness is not required, two configurations are possible. A basic round robin between FQs can be done by setting the credit amount (ICS\_CRED) to 0. In that configuration, 1–3 frames per FQ is consumed at every dequeue opportunity. Conversely, if the credit amount is set to its maximum value (15'h7FFF), this effectively disables intra-WQ class scheduling and causes this FQ to never be rescheduled due to weighted fairness, that is, its consumption allowance is never satisfied.

When dequeue operations are performed, intra-WQ class scheduling of FQs operates in conjunction with WQ priority, and with the dequeue command type being executed, to determine when an Active FQ is rescheduled or when a Held Active FQ becomes Held Suspended. See [Section 3.3.8.2.6, “Dequeue Dispatcher Operation—Dequeueing from One or More Channels,”](#) and [Section 3.3.8.2.7, “Dequeue Dispatcher Operation—Dequeueing from a Specific WQ”](#) for more detail on the interactions involved in dequeue operations.

### 3.3.8 Software Portals

Software portals are used by software running on processor cores to communicate with the QMan.

All of QMan’s software portals are mapped into a 128 MB region of memory. Within this region, each software portal occupies an area that is intended to be private to each processor core using it, although the QMan itself does not enforce any particular connection between software portals and processor cores. The QMan contains a total of 10 software portals intended to be used as follows:

- Two software portals for use by each processor core
- One software portal for use by Hypervisor
- One software portal for use by an external processor accessing the QMan

The memory area occupied by each software portal is divided into two separate regions.

The first is a 64 KB region, aligned on a 64 KB boundary, that contains registers and structures that are 64 bytes in size and are should be accessed using a single bus transaction. This area of the portal should be configured as cache-enabled in the processor core using the portal.

The cache-enabled area of the portal contains the following major structures:

- One enqueue command ring (EQCR)
- One dequeue response ring (DQRR)
- One message ring (MR)
- One management command register (CR)
- A set of management response registers (RR0/RR1)

Note that although the locations in the cache-enabled area are nominally 64 bytes, the EQCR, DQRR, MR, and CR all occupy only the first 32 bytes of their 64-byte location, allowing them to fit into a single cache line (and thus be accessed using a single bus transaction) on systems using either 32 or 64 byte cache lines. Some of the responses returned in the RR0/RR1 registers occupy all 64 bytes, requiring 2 cache lines on systems with 32-byte cache lines.

The second memory area is a 64 KB region, aligned on a 64 KB boundary, that contains only 32-bit wide registers that are intended to be accessed using 4 byte transactions. Note that all of the registers in the cache-inhibited area are 32 bits wide, and must only be written or read as a full, 32-bit value. It is a programming error to attempt 8- or 16-bit accesses. This area should be configured as cache-inhibited and guarded in the processor core using the portal.

[Section 3.2.1, “QMan Software Portals \(QCSP\) Memory Map”](#) contains a detailed memory map of all of the software portal’s resources, and [Section 3.2.3, “QMan Software Portals \(QCSP\) Register](#)

Descriptions,” contains a detailed description of the various ring entries and registers used. The following subsections contain a more detailed functional description of the resources used by the software portals.

### 3.3.8.1 Enqueue Command Ring (EQCR)

From the QMan’s perspective, the enqueue command ring (EQCR) is read-only. The QMan does not modify the data in EQCR entries in any way.

The EQCR is used to issue frame enqueue commands from software to the QMan. The EQCR is an 8-entry, circular FIFO contained in the cache-enabled area of the QMan’s software portal (see [Section 3.2.3.1, “QCSP Enqueue Command Ring Registers \(QCSPi\\_EQCR0-7\)](#)). Each entry is 64 bytes in size and carries an enqueue command as described in [Section 3.3.9.1, “Enqueue Command.”](#) Software produces entries into the EQCR ring by writing enqueue commands into ring entries, and the QMan consumes and executes these enqueue commands by reading them from the ring entries.

Because the EQCR entries are intended to be cache-enabled, to issue a command, software must do the following steps:

1. Zero and allocate the entry in the processor’s cache without reading it (Use DC ZVA for example)
2. Write the command
3. Flush the entry from cache and write it into the EQCR stored in QMan (Use DC CVAC for example)

Some additional steps may be required depending on the mode of production notification that is selected.

EQCR entries are also readable to accommodate a castout that may occur before the cache line flush while the command is being written into cache. A read of any EQCR entry returns the entire entry with all data present as last written.

Software can issue multiple commands into the ring at a time, but must flow-control its production into the EQCR to prevent the ring from overflowing. The ring is full when the difference between its producer and consumer indices is 7 or more ( $\text{full} = \text{EQCR\_PI} - \text{EQCR\_CI} \geq 7$ ). As such, the ring can contain a maximum of 7 valid entries.

Enqueue commands issued to the EQCR are never rejected due to resource depletion. If the number PFDRs required to complete an enqueue operation is insufficient, the QMan stops consuming entries from the EQCR until more PFDRs become available.

#### 3.3.8.1.1 EQCR Errors

If an error is detected in the command contained in an EQCR entry, the following occurs:

- The command is consumed from the ring without being executed
- An error interrupt is raised (if enabled)
- An enqueue rejection notification (ERN) may be produced if appropriate

This happens to allow software to recover the FD that was sent in the enqueue command. Possible errors are listed in [Section 3.2.4.51, “QMan Error Interrupt Status Register \(QMAM\\_ERR\\_ISR\)”](#).

### 3.3.8.1.2 EQCR Production Notification

Production notification from software to the QMan is tracked using a producer index value (EQCR\_PI) stored in the QMan. To software, the value of EQCR\_PI indicates the ring entry in which the next valid command should be placed. The QMan detects that the ring contains valid entries to be consumed when EQCR\_PI and EQCR\_CI are not equal.

After reset, EQCR\_PI is set to 0 and must be incremented when valid commands are written into the ring, wrapping from 7 back to 0 when the end of the ring is reached. The modes that can be selected for the update method of this value are as follows; each mode is programmable in a control register within each portal (see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\)”](#)):

- PI write mode, cache-inhibited
  - EQCR\_PI is a 3-bit value in a cache-inhibited, read/write register.
  - The register is mapped on a word boundary, and software issues a cache-inhibited store (i.e. write) to this word to advance EQCR\_PI. Software must issue an appropriate synchronizing instruction between the cache line flush, which writes the EQCR entry, and the cache-inhibited write that updates EQCR\_PI. The synchronizing instruction used must ensure ordering between cache-enabled and cache-inhibited stores.
- PI write mode, cache-enabled
  - EQCR\_PI is a 3-bit value in a cache-enabled, read/write register.
  - The register is mapped on a 64-byte boundary, and software issues a store (i.e. write) followed by a cache line flush to advance EQCR\_PI. Software must issue an appropriate synchronizing instruction between the cache line flush, which writes the EQCR entry and the cache-enabled write that updates EQCR\_PI. The synchronizing instruction used must ensure ordering between cache-enabled stores.
- Valid bit mode
  - An alternating polarity valid bit and the presence of a non-zero command verb are used in each EQCR entry to indicate a valid ring entry. This option removes the need for software to explicitly update EQCR\_PI with a write.
  - The EQCR\_PI value updated by QMan is available as a read-only value in both its cache-inhibited and its cache-enabled locations. The current valid bit polarity tracked by the QMan can also be read by software. Note that these read-only values are intended for debug purposes only, they should not be used by software to obtain real time ring status. The PI and current valid bit polarity values should always be maintained and tracked by software. This is explained in more detail later in this section.

In valid bit mode, because the production notification to QMan is driven by data in the entry itself (the command verb and its valid bit) rather than an explicit write to EQCR\_PI, there are extra steps required to issue a command in the ring. To write a command into an EQCR entry, software must do the following:

1. Zero out the verb, or the entire command cache line (Use DC ZVA for example)
2. Write all words of the command other than word 0
3. Issue a synchronizing instruction (Use DMB st for example)
4. Write word 0 (which contains the command verb and its valid bit)

## 5. Flush the command from cache to the QMan (Use DC CVAC for example)

The requirement of the word containing the command verb being written last, and a synchronizing instruction being needed before writing word 0 are the only steps unique to valid bit mode. These additional steps ensure that a castout of a partially completed command from the processor's cache always write a 0 (null) command verb, thus preventing the QMan from attempting to execute this partial command. A command verb of 00h is interpreted as an entry not yet produced (no command present) even if the current valid bit polarity is 0.

In the two PI write modes, the valid bit in the EQCR entries is ignored by QMan. When valid bit mode is used, the alternating polarity valid bit works as follows: When software writes a command verb into an EQCR entry, it also sets the valid bit (the msb of the command verb) to the current valid bit polarity. For example, the very first command written after reset must have a 1 in its valid bit, because this is the valid bit polarity after reset. Software must track the current valid bit polarity (as mentioned, it is 1 after reset), and must toggle this current polarity each time that the ring wraps from entry 7 to entry 0. Therefore, the first eight commands issued to EQCR have a valid bit of 1, the next eight have a valid bit of 0, the next eight have a valid bit of 1, and so on. The use of the alternating polarity valid bit is used to protect against the possibility of a speculative read of an EQCR entry followed by a castout of that entry from the processor's cache (see also [Section 3.3.8.1.5, “Speculative Reads of the EQCR Cache-Enabled Entries when Using Implicit Production Notification”](#)).

The QMan also locally tracks the current expected valid bit polarity. In valid bit mode, QMan's internal EQCR\_PI is advanced implicitly in the portal logic when a write with both a valid bit of the expected polarity and a non-null command verb is received at the entry currently pointed at by EQCR\_PI. When EQCR\_PI is advanced, to handle potential misordering of writes to the ring entries the portal logic checks subsequent locations in the ring and advances EQCR\_PI to either the next entry containing a non-valid command (wrong polarity valid bit and/or null command verb) or until the ring is full, whichever comes first.

When in valid bit mode, the internal EQCR\_PI and valid bit polarity values used by QMan can be read by software for debug purposes, however these values should never be used by software to obtain real time ring status. One reason for this is that reading the EQCR\_PI value from QMan can have a performance impact by creating additional unnecessary bus transactions in the system, such reads are unnecessary because software is the producer of entries into the EQCR ring, and is able to track the PI and valid bit polarity values without reading them from QMan. Another reason is that the implicitly advanced EQCR\_PI value internal to QMan (advanced when a EQCR write is received) will always lag the EQCR\_PI value tracked by software (advanced when a EQCR write is launched). Therefore reading the EQCR\_PI value from QMan when in valid bit mode can result in a stale value being read since the read of EQCR\_PI can pass the write of the EQCR entry on its way to QMan. Of course this potential of read passing write can be accounted for by using an appropriate synchronizing instruction between the two, but this would lead to an even greater potential performance impact. So for these reasons the real time ring status (EQCR\_PI value and valid bit polarity) should always be maintained and tracked by software, without reading them from QMan.

**NOTE**

Switching between the different production notification modes should never be done dynamically; it should only be done as an initialization step when the ring is empty.

It is possible and safe to switch modes when the ring is empty, as long as certain precautions are taken. In the two PI write modes, the current expected valid bit polarity is always tracked and updated by QMan as enqueue commands are produced and consumed in the ring, even though this value is not used to drive the consumption of commands in these modes. This means that if EQCR has been operating in one of the PI write modes, and then software wishes to switch to valid bit mode once the ring becomes empty, it must ensure that all valid bits presently sitting in the ring are of the correct polarity before making the switch; otherwise, one or more stale enqueue commands sitting in EQCR entries may be re-executed because they contain a valid bit of the expected polarity. The only safe way to make such a mode switch is to always write correct polarity valid bits into the EQCR even when in one of the PI write modes, in which case a subsequent mode switch when the ring is empty works smoothly. If this method is never used, maintaining correct valid bits when in PI write mode is not necessary.

### **3.3.8.1.3    EQCR Consumption Notification**

Consumption notification from the QMan to software is tracked using a read-only consumer index value (EQCR\_CI) stored in the QMan. EQCR\_CI is a 3-bit value that is 0 after reset, and indicates the ring entry that is next to be consumed by the QMan. Software uses EQCR\_CI to determine when the ring is full and cannot accept a new entry.

The EQCR\_CI value is available for read in three different locations in the portal: one of the locations is word aligned and located in the cache-inhibited region of the portal, and the other two locations are 64-byte aligned and located in the cache-enabled region of the portal. See [Section 3.2.3.7, “QCSP EQCR Consumer Index Registers,”](#) and [Section 3.2.3.16, “QCSP Read-Only Ring Indices Cache-Enabled Registers \(QCSPi\\_RORI\\_CENA\).”](#) Also, in one of the cache-enabled locations (RORI\_CENA), EQCR\_CI is combined with the other read-only indices from the DQRPs and MRs such that all indices can be obtained with a single, cache-enabled read.

Another option is to enable EQCR\_CI stashing (see [Section 3.3.8.7, “EQCR\\_CI Stashing”](#)) to allow EQCR\_CI values to be stashed by the QMan into a processor core’s cache.

### **3.3.8.1.4    EQCR Interrupts**

Interrupts can be generated from the EQCR in the following cases:

- Interrupt on command dispatch

This is a per-command bit in the enqueue command itself.

This interrupt occurs after the enqueue command has been read from the EQCR entry, and dispatched to an algorithmic sequencer (AS) for execution. At this point the EQCR entry is consumed, and the interrupt is raised if the appropriate bit was set in the enqueue command.

Note that at this point the enqueue command is not complete, and may not complete for some time if it needs to be deferred due to order restoration; however, it has executed to the point where it is committed and its order is guaranteed within the QMan.

- Interrupt when EQCR occupancy is below a threshold  
This interrupt occurs when the ring contains fewer entries than set in a programmable threshold (See [Section 3.2.3.18, “QCSP EQCR Interrupt Threshold Register \(QCSPi\\_EQCR\\_ITR\)”](#)). Ring occupancy is equal to EQCR\_PI minus EQCR\_CI, and this interrupt is asserted when ring occupancy is less than the threshold. A threshold of 0 disables the interrupt.
- In the case where EQCR production notification uses one of the two PI write modes, the QMan’s detection of the valid bit and non-null command verbs is disabled, and command consumption is driven solely from the EQCR\_PI values written by software. In that case, a null command found in an entry that is supposed to be valid triggers the QMan’s invalid command interrupt.

### **3.3.8.1.5 Speculative Reads of the EQCR Cache-Enabled Entries when Using Implicit Production Notification**

The purpose of the valid bit mode for EQCR production notification is to allow QMan to implicitly determine when valid entries have been placed in the ring without having software issue explicit writes to update the ring’s producer index (PI) value.

The first requirement for implicit production notification is that QMan must look for a non-zero command verb to protect against the castout of a partially completed command, as explained in [Section 3.3.8.1.2, “EQCR Production Notification.”](#) In addition to this, the QMan is designed to respond without error in the case where, after software has issued an EQCR ring entry using cache line flush, a speculative read followed by a castout occurs, which overwrites the ring entry. Such an event is unlikely, because a castout of the entry should not occur unless the entry is explicitly modified by software, however, this event has been considered and the use of an alternating polarity valid bit is intended to allow EQCR (and CR) production notification to operate correctly even under this unlikely worst case scenario.

There are three cases to be considered that depend on the order in which the related operations occur. The cases shown are intended to illustrate why the valid bit is used. Also note that the same reasoning applies to both the management command register (CR) and the ring entries in the EQCR:

- Case 1: Command flushed, command consumed, speculative read, castout  
In this case if the command verb reads back as written, the castout causes the command to be repeated erroneously.  
This case could be fixed by making the command verb always read back as 0.
- Case 2: Command flushed, speculative read, castout, command consumed  
In this case the command verb must be read as it was written, so that the castout does not corrupt the valid command that has not yet been consumed. So the fix for Case 1 breaks Case 2.  
A fix for both cases 1 and 2 would be to have the QMan clear the command verb when it consumes the command, and then return the command verb unchanged for all reads.  
So in Case 1 the command verb would read as 0, and in case 2 it would read as non-zero, and would be written correctly by the castout in both cases.
- Case 3: Command flushed, speculative read, command consumed, castout  
The fix for Case 2 does not work for case 3. In case 3, the speculative read reads a valid non-zero command and stores it in the processor’s cache. Then QMan consumes the command and clears it in its local backing store, but it is still valid non-zero in the processor’s cache. Then, the castout writes this valid command again, causing it to be erroneously repeated.

A solution to address these cases is to add a valid bit in each EQCR ring entry. An absolute polarity valid bit, that is, a valid bit in which a fixed polarity of either 1 or 0 always indicates a valid entry, would require that the QMan clear the valid bit to its invalid state after consuming an entry from the ring, and this would still cause an erroneously repeated command in Case 3 above.

A solution which addresses all three cases is to use an alternating polarity valid bit as well as a non-zero command verb to drive the detection of valid commands. The use of an alternating polarity valid bit to indicate valid ring entries works reliably for all three speculative read cases, has the advantage of being simple to implement, and that the same solution works for the EQCR, the DQRR, and the management CR, so it is symmetric in the enqueue and dequeue directions.

### 3.3.8.2 Dequeue Response Ring (DQRR)

The dequeue response ring (DQRR) is used to issue frame dequeue responses from the QMan to software. The DQRR is a 16-entry, circular FIFO contained in the cache-enabled area of the QMan’s software portal (see [Section 3.2.3.2, “QCSP Dequeue Response Ring Registers \(QCSPi\\_DQRR0-15\).”](#)) Each entry is 64 bytes in size and carries a frame dequeue response as described in [Section 3.3.9.2, “Frame Dequeue Response.”](#) The QMan produces entries into the DQRR by writing frame dequeue responses into the ring entries, and software consumes and processes these dequeued frames by reading them from the ring entries.

From the software’s perspective, DQRR entries are read-only. Because these entries are cache-enabled, software must invalidate them in the processor’s cache before reading them. If enabled to do so, the QMan stashes the DQRR entries into a processor’s cache when they are produced, placing the entries in the processor’s cache and removing the need to invalidate the entries before reading them.

The DQRR can be programmed to operate in one of two modes, Push mode or Pull mode. In Push mode, a single pre-loaded dequeue command can be used to provide all dequeues. As long as a valid dequeue command is present, there are frames available to be dequeued, and there is room in the DQRR to accept the response to the command, the QMan executes the pre-loaded dequeue command and push frame dequeue responses into the DQRR. In pull mode, the QMan executes each command written by software exactly once, and place the resulting response into the DQRR. Software must write a new command to initiate each subsequent dequeue operation. More detail on these two modes and on the dequeue commands used can be found in [Section 3.3.8.2.3, “DQRR Dequeue Commands.”](#)

The max fill level of the DQRR, called DQRR\_MF, is a 4 bit value programmable on a per-portal basis (see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\)”](#)). DQRR\_MF configures the maximum number of valid entries that QMan populates in the ring at any time. A value of 0, the default (reset) value, effectively disables the DQRR, a value of 15 allows up to 15 entries in the ring. DQRR\_MF is a static configuration value, it should not be changed dynamically while traffic is flowing.

Note that the QMan always waits for sufficient space to be available in the DQRR before executing any dequeue command. The number of frames that may be returned by a dequeue command in SDQCR or PDQCR can be set to either 1 frame only or to 1–3 frames using the FC field in the dequeue command. Therefore, if DQRR\_MF is set to 1 or 2, only dequeue commands with FC = 0 (dequeue at most one frame) can be used, if a dequeue command with FC = 1 is used, all dequeues are blocked. For commands in VDQCR, there must always be at least three free entries in the DQRR before the command is dispatched, therefore DQRR\_MF must be 3 or greater to enable the use of VDQCR.

### 3.3.8.2.1 DQRR Production Notification

Production notification from the QMan to software includes both an alternating polarity valid bit in each DQRR entry and a read-only, 4-bit producer index (DQRR\_PI).

The value of DQRR\_PI indicates the ring entry in which the next valid response is placed. The ring contains valid entries to be consumed when DQRR\_PI and DQRR\_CI are not equal.

After reset, DQRR\_PI is set to 0, and it is incremented by the QMan when valid entries are written into the ring, wrapping from 15 back to 0 when the end of the ring is reached. DQRR\_PI is a 4 bit, read-only value. The same value is available in three different locations, one cache-inhibited and two cache-enabled locations. Also, in one of the cache-enabled locations (RORI\_CENA), DQRR\_PI is combined with the other read-only indices from the EQCRs and MRs such that all indices can be obtained with a single, cache-enabled read. See [Section 3.2.3.8.2, “QCSP DQRR Producer Index Cache-Inhibited Registers \(QCSPi\\_DQRR\\_PI\\_CINH\),”](#) and [Section 3.2.3.16, “QCSP Read-Only Ring Indices Cache-Enabled Registers \(QCSPi\\_RORI\\_CENA\).”](#)

An alternating polarity valid bit is also provided in each entry, and allows software to detect when entries are added to the DQRR by polling on the valid bit in the next expected ring entry, without having to explicitly read the DQRR\_PI value from the QMan.

The alternating polarity valid bit works as follows: When QMan writes a frame dequeue response into a DQRR entry, it also sets the valid bit (the msb of the response verb) to the current valid bit polarity. For example, the very first response written after reset has a 1 in its valid bit, because this is the valid bit polarity after reset. The QMan keeps track of the current valid bit polarity (as mentioned, it is 1 after reset), and toggles this current polarity each time that the ring wraps from entry 15 to entry 0. Therefore, the first 16 responses issued to DQRR have a valid bit of 1, the next 16 have a valid bit of 0, the next 16 have a valid bit of 1, and so on.

It is important to note that the DQRR\_PI value is advanced by QMan immediately after a valid entry has been placed in the ring. If stashing of DQRR entries is disabled this works as intended, since software must read DQRR\_PI before reading any DQRR entries. However if stashing of DQRR entries is enabled, the update of DQRR\_PI occurs before the associated stash of any DQRR entries, which creates a race condition between software’s read of DQRR\_PI and subsequent read of the next DQRR entry, and the stash of that DQRR entry arriving in the processor’s cache. For this reason, DQRR production notification to software should be done using only the valid bit in the DQRR entries, not the DQRR\_PI value, when DQRR entry stashing is enabled.

### 3.3.8.2.2 DQRR Consumption Notification

Consumption notification from software to QMan is tracked using a Consumer Index value (DQRR\_CI) stored in QMan. DQRR\_CI is a 4 bit value that indicates the ring entry which is next to be consumed by software. QMan uses DQRR\_CI to determine when the ring is full and cannot accept a new entry.

After reset DQRR\_CI is set to 0, and it must be incremented when responses are consumed from the ring, wrapping from 15 back to 0 when the end of the ring is reached. There are three modes that can be selected for the update method of this value. The mode is programmable in a control register within each portal; see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\).”](#)

- CI write mode, cache-inhibited.

- In this mode, DQRR\_CI is a 4 bit value in a cache-inhibited read/write register.
- The register is mapped on a word boundary, and software issues a cache-inhibited store to this word to advance DQRR\_CI after it has consumed a DQRR entry. Writes to DQRR\_CI do not need to occur after consumption of each entry, a single write can be used to advance DQRR\_CI after several entries have been consumed, thus amortizing the cost of the write over multiple frame responses in DQRR. This is also referred to as cumulative consumption acknowledgment.
- CI write mode, cache-enabled
  - In this mode, DQRR\_CI is a 4-bit value in a cache-enabled read/write register.
  - The register is mapped on a 64-byte boundary, and software issues store followed by cache line flush to advance DQRR\_CI. As with the cache-inhibited mode, the write to this register can be amortized over the consumption of several DQRR entries.
- Discrete consumption acknowledgment (DCA) mode
  - This mode allows software to issue discrete (rather than cumulative) consumption acknowledgments to QMan, via the DQRR\_DCAP register, and/or by adding an embedded DCA to enqueue commands issued in the EQCR.
  - DQRR\_DCAP is a cache-inhibited read/write register. The register is mapped on a word boundary, and software issues a cache-inhibited store to this word to issue discrete consumption acknowledgments. See [Section 3.2.3.10, “QCSP DQRR Discrete Consumption Acknowledgment and Park \(QCSPi\\_DQRR\\_DCAP\).”](#)
  - Discrete Consumption Acknowledgment can also be embedded in the enqueue commands issued by software in the EQCR; see [Section 3.3.9.1, “Enqueue Command.”](#)

In all modes, QMan maintains a 16 bit Valid Vector (DQRR\_VV) indicating the state (Valid or not) of each DQRR entry. When DCA mode is used and a DCA is received by QMan (as either a write to the DQRR\_DCAP register or an embedded DCA in an enqueue command), the entry or entries indicated in the DCA are marked as consumed in the DQRR\_VV, that is, their entries are marked as invalid. At the same time, DQRR\_CI is advanced from its current value to the point to the next valid entry in the DQRR, or until it is equal to DQRR\_PI, in which case the ring is empty. Note that DQRR\_CI never jumps over a valid entry, if a DCA is received which consumes some entries but does not consume the entry currently pointed at by DQRR\_CI, then the DQRR\_CI value does not change even though other entries have been marked as consumed. Only when the entry pointed at by DQRR\_CI is consumed will DQRR\_CI advance.

In DCA mode, the DQRR\_CI value updated by QMan is available as a read-only value in both its cache-inhibited and its cache-enabled locations. Also, the DQRR\_VV is readable in all modes for debug/verification assist purposes.

When DCA mode is selected, a single DQRR entry consumption can be embedded in a frame enqueue command issued to the EQCR. This reduces the number of writes required to issue DQRR consumption acknowledgments, in that the frame that is dequeued in each DQRR entry can be used to consume that DQRR entry when the frame is re-enqueued via the EQCR. In this way the only additional writes needed for DQRR consumption acknowledgment are for any DQRR entries whose frames are not re-enqueued, such as frames discarded by software.

The use of a DCA embedded in an enqueue command also provides an assist for implementing an order preservation paradigm for frame processing. A DCA embedded in an enqueue command is processed by QMan after the enqueue command has been read from the EQCR and has been placed in line (in the software portal EQCR/CR dispatch FIFO) to be dispatched for execution to an Algorithmic Sequencer (AS). At this point the enqueue command is consumed from the EQCR, and if an embedded DCA is used the DQRR entry associated with the frame being enqueued is also consumed. If the FQ from which this frame was dequeued is ready to be released but is being held in the portal (is in the held suspended state), and if this DCA consumes the last DQRR entry containing a frame from this FQ, then the held FQ is released and rescheduled (or parked). Delaying the release of this held FQ until the enqueue of the last frame which was dequeued from it is in line to be dispatched enables order preservation by preventing any future enqueues to the same FQ from passing this one.

### 3.3.8.2.3 DQRR Dequeue Commands

The DQRR can be programmed to operate in one of two modes, Push mode or Pull mode (see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\).”](#))

In Push mode, two registers are provided in which dequeue commands can be placed, they are the Static and the Volatile dequeue command registers; see [Section 3.2.3.11, “QCSP DQRR Static Dequeue Command Register \(SDQCR\),”](#) and [Section 3.2.3.12, “QCSP DQRR Volatile Dequeue Command Register \(QCSPi\\_DQRR\\_VDQCR\).”](#)

The SDQCR is used to issue commands for scheduled dequeues, from one or more channels or from a specific WQ. It is referred to as static because as long as a non-null command is present in this register, there are frames available to be dequeued, and there is room in the DQRR to accept the response to the command, the QMan continually repeats the execution of this command and pushes the resulting frame dequeue responses into the DQRR.

The VDQCR is used to issue commands for unscheduled dequeues, from a specific parked or retired FQ. It is referred to as volatile because its execution is set to terminate after a specified number of frames have been dequeued or when the FQ becomes empty, at which point the command in VDQCR is cleared and a new command must be written to issue more unscheduled dequeues.

When a valid command is present in both SDQCR and VDQCR, the QMan executes the command from one or the other, the selection of which one to use is determined by the Precedence bit in the VDQCR.

In pull mode, the SDQCR and VDQCR registers are disabled, and a separate Pull dequeue command register (see [Section 3.2.3.13, “QCSP DQRR Pull Dequeue Command Register \(QCSPi\\_DQRR\\_PDQCR\)”](#)) is used. In this mode, the QMan executes each command written by software into PDQCR exactly once, and places the resulting response into the DQRR. Software must write a new command to PDQCR to initiate each subsequent dequeue operation.

In either mode, a dequeue command is dispatched only if there is room in the DQRR to accept the response.

In both the SDQCR and PDQCR, a Dequeue Command Type (DCT) field is used to specify exactly how dequeue operations should occur as a result of the command. There are 3 different types of dequeue command, and the details of their operation is described in [Section 3.3.8.2.6, “Dequeue Dispatcher Operation—Dequeueing from One or More Channels.”](#)

### 3.3.8.2.4 DQRR Dequeue Command Portal Selection

A software portal dequeue command is ready for execution when the following conditions are met:

- When a valid dequeue command is present in SDQCR, VDQCR, or PDQCR.
- For a SDQCR scheduled dequeue, when there is something to be dequeued, i.e. when one of the WQ channels targeted by the dequeue command is non-empty, or if there is a currently active FQ in the portal.
- For a scheduled dequeue (SDQCR and PDQCR), if there is at least one free PSCL available. See [Section 3.3.8.2.8, “Active and Suspended Frame Queues.”](#) When there is enough room in the DQRR to accept the response.
- When there is no command already in progress for this software portal. A maximum of 1 DQRR command in progress per portal is allowed.
- When there is no halt request for this portal.

These conditions can be re-phrased from the point of view of the each dequeue command type.

- For SDQCR scheduled dequeues, execution is ready if there is something available to be dequeued, and if there is a free PSCL available.
- For PDQCR scheduled dequeues, execution is ready if there is a free PSCL available. WQ channel status is not checked, PDQCR execution will proceed and will return a null response if an empty channel is targeted.
- For unscheduled dequeues from VDQCR or PDQCR, channel availability and PSCL are not used. Unscheduled dequeues do not use a PSCL, see [Section 3.3.8.2.10, “Dequeue Command Behavior with Parked and Retired Frame Queues.”](#) When a software portal contains a dequeue command that meets the above conditions, and is thus ready for execution, that dequeue command is dispatched to one of the available algorithmic sequencers for execution, at which point the DQRR dequeue dispatcher (described in [Section 3.3.8.2.5, “DQRR Dequeue Dispatcher”](#)) is invoked to select the WQ channel, the WQ within the channel, and the FQ from which to dequeue.

At any given time, several software portals may contain a dequeue command that is ready for execution. In this case, selection of a software portal for dequeue command execution proceeds in a simple round-robin fashion, with each portal getting a chance to execute one dequeue command (resulting in 0 to 3 frames delivered into that portal’s DQRR) in turn.

When waterfall power management is enabled (see the WPM bit in [Section 3.2.4.39, “QMan Miscellaneous Configuration Register \(QMAM\\_MISC\\_CFG\)”](#)), the default round-robin selection between software portals is replaced with an algorithm that supports water fall power management (WPM) in QMan in conjunction with the processor cores.

Waterfall power management is a mechanism that works between the QMan and the e6500’s Drowsy Core mode. The basic idea is that when using multiple cores and pool channels to forward frames, and the QMan receives less traffic than required to keep all cores busy, it no longer assigns to the higher numbered cores (i.e. software portals) in a pool. When these higher numbered cores look for new work in their DQRR, they no longer find work and should eventually execute a WAIT instruction which after a time out places the core into its drowsy mode. Then when enough traffic arrives to fill the DQRR of one portal QMan will resume sending traffic to the higher numbered portals, and the cores using those portals will be woken out

of their drowsy state by either a stash transaction (see [Section 3.3.8.6, “DQRR Entry Stashing”](#)) or an interrupt received from QMan.

To fill the DQRR rings in a waterfall fashion, when WPM is enabled and a dequeue command is dispatched for execution, the portal selection round-robin history is always set to select (at the next selection) portal 0, rather than setting it to select the next portal in line as is done by default.

This produces a strict priority (rather than round robin) selection which will favor lower numbered portals until their DQRR becomes full. Note however that even when DQRPs are not full, traffic may be dispatched to a few low numbered portals, not just portal 0, because QMan always attempts to dispatch dequeue commands in parallel for different portals to different sequencers.

WPM should only be enabled when a pool of cores all executing the same or similar forwarding software are used to receive dequeue traffic from one or more pool channels, because when WPM is enabled dequeue traffic to low numbered portals may starve any dequeue traffic to higher numbered portals. In an application where one or a few cores are needed for something other than datapath forwarding and require a relatively small amount of dequeue traffic, and use of WPM is still desired, then those cores can be assigned to the lowest numbered portals 0 to n, with the datapath cores assigned to portals n+1 to m. In this case the non-datapath cores can still receive their occasional dequeues while the datapath cores are assigned their dequeue traffic in a waterfall fashion.

### **3.3.8.2.5 DQRR Dequeue Dispatcher**

For each software portal there is logic that performs WQ Channel Scheduling and Active Frame Queue management. This logic is collectively referred to as the Dequeue Dispatcher and operates independently for each software portal such that decisions made for one software portal do not affect the others (except indirectly when software portals are consuming from the same Pool Channel).

The major component of Dequeue Dispatcher is the WQ Channel Scheduler. It is responsible for determining the next channel to service in the case where the Dequeue Dispatcher decides not to continue to dequeue frames from the currently active Frame Queue. It also provides Dequeue Dispatcher with information (the highest priority non-empty WQ in all relevant channels) to help decide whether to continue with the currently active Frame Queue or not.

The WQ Channel Scheduler in turn relies upon WQ Class Schedulers that are implemented for each channel. A WQ Class Scheduler (see [Section 3.3.7.2, “WQ Class Scheduling”](#)) for each channel decides upon the most urgent WQ within that channel to service and the service priority at which it should be treated. The WQ class schedulers constantly supply to the WQ channel scheduler the highest priority work from the channels. This appears as one of 5 distinct service priorities and an indication of the queue within that priority, as shown in [Table 3-101](#).

**Table 3-101. Dequeue Dispatcher Service Priorities**

Service Priority Provided by WQ Class Scheduler to WQ Channel Scheduler	WQ # within Service Priority Provided by WQ Class Scheduler to WQ Channel Scheduler	WQ within Channel
0 (Highest)	0	WQ 0
1 (High)	0	WQ 1

**Table 3-101. Dequeue Dispatcher Service Priorities (continued)**

<b>Service Priority Provided by WQ Class Scheduler to WQ Channel Scheduler</b>	<b>WQ # within Service Priority Provided by WQ Class Scheduler to WQ Channel Scheduler</b>	<b>WQ within Channel</b>
2 (Elevated Low)	0, 1, or 2	WQ 5,6, or 7 (temporarily elevated when the low priority tier is starved)
3 (Medium)	0, 1, or 2	WQ 2,3,or 4
4 (Low)	0, 1, or 2	WQ 5,6, or 7

Using the service priority and WQ number within that service priority provided from each channel, the WQ channel scheduler chooses the channel with the highest priority of work available.

If more than one channel has work at that same priority, the channel is chosen by round-robin based on the last channel serviced for that portal and for that particular service priority level (the last channel serviced history is held for each of the 5 service priorities in each software portal). It is important to note that the round robin is independent for the Elevated Low service priority despite the fact that it represents a servicing of the same WQs as in the Low priority. This is necessary such that when a low priority is constantly being elevated for a given channel, the servicing of channels at Low priority (not elevated) is not affected by the channel selection made at the Elevated Low service priority.

Note that the WQ channel scheduler's history registers are only updated when a scheduled dequeue from a channel is performed. When a dequeue command specifies a scheduled dequeue from a specific WQ, or an unscheduled dequeue from a specific FQ, the WQ channel scheduler's history registers are not updated.

The dedicated channel for each portal can (optionally within the Dequeue Command) be given precedence over the pool channels, meaning that it is selected whenever its advertised service priority is equal to or higher than that of all pool channels selected in the dequeue command. When this option is used in the dequeue command, selection of the dedicated channel does not affect the last channel serviced history for that service priority, this is done in order not to bias the round robin selection from among the pool channels.

For the active FQ in the portal the Dequeue Dispatcher maintains the following state:

- whether or not there is an active FQ
- whether the active FQ is Active or Held Active
- the service priority of the active FQ
- whether or not Intra-WQ class scheduling consumption allowance is satisfied
- where the FQ's FQD is stored (which of the 4 PSCL locations; see [Section 3.3.8.2.8, “Active and Suspended Frame Queues”](#))

Within the Dequeue Dispatcher there exists a special-case optimization that is not visible to the user (except that performance is not degraded as it might have been had this not been implemented). It allows a currently active FQ with intra-WQ class scheduling satisfied to remain active (with a newly calculated Intra-WQ class scheduling consumption allowance) in the case where it would be the FQ selected if it were able to be rescheduled instantly.

Figure 3-105 is a block diagram showing the dequeue dispatcher and the WQ class schedulers.

## Queue Manager (QMan)

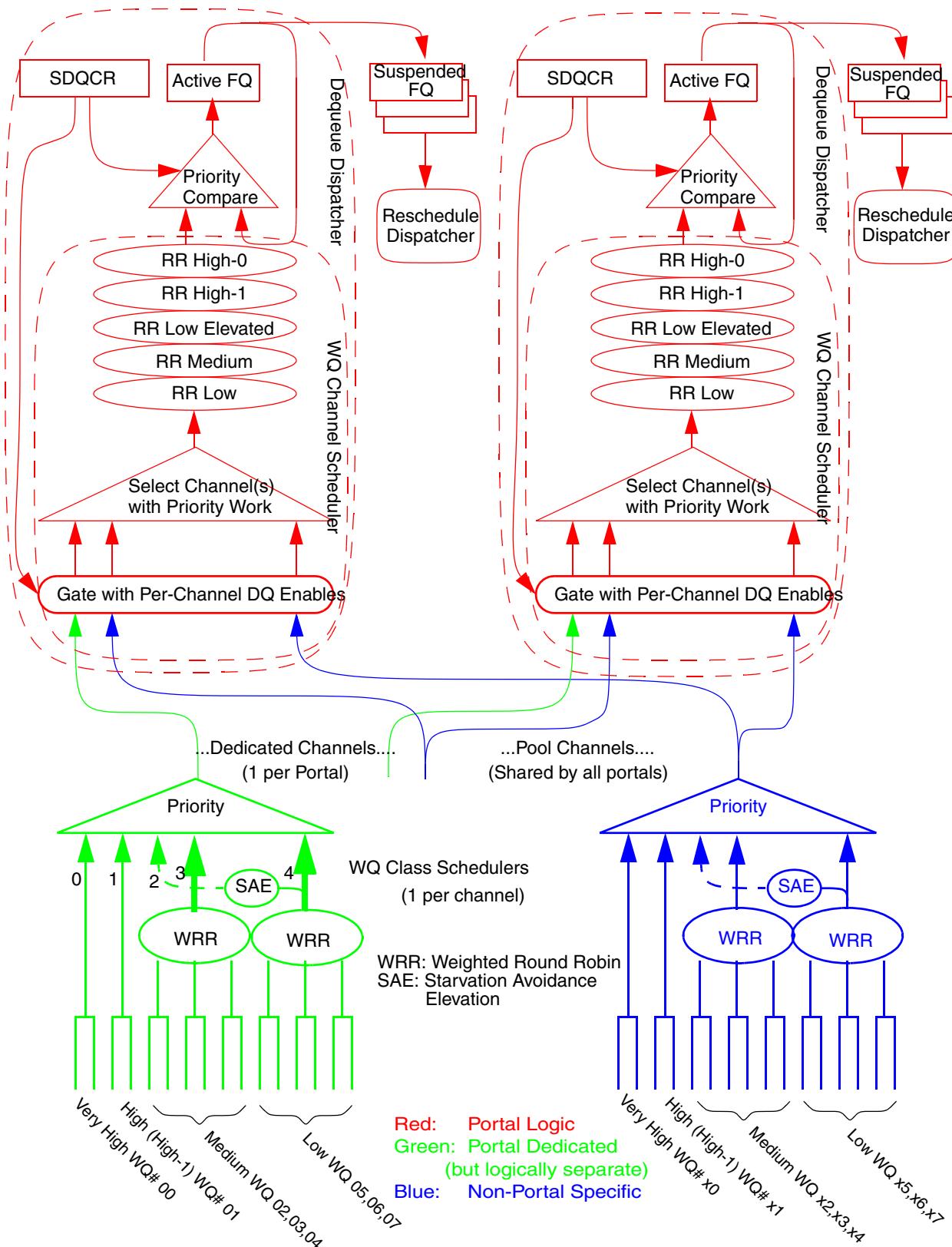


Figure 3-105. Software Portal Dequeue Dispatcher and WQ Class Schedulers

### 3.3.8.2.6 Dequeue Dispatcher Operation—Dequeueing from One or More Channels

This section describes the sequence of operations performed when a dequeue command specifies a scheduled dequeue from one or more channels within a portal.

In both the SDQCR and PDQCR (see [Section 3.3.8.2.3, “DQRR Dequeue Commands”](#)), a source select bit (SS) is used to specify whether a scheduled dequeue consumes from one or more channels in the portal (SS = 0), or from a specific WQ (SS = 1). This section describes dequeue operation for SS = 0, and [Section 3.3.8.2.7, “Dequeue Dispatcher Operation—Dequeueing from a Specific WQ,”](#) describes the operation for SS = 1.

In both the SDQCR and PDQCR, a dequeue command type field (DCT) is used to specify exactly how the dequeue dispatcher operations should occur as a result of the command. There are three different types of dequeue commands, and the details of their operation when dequeuing from one or more channels is described below, and is illustrated with a flow diagram in [Figure 3-106](#). These descriptions refer to FQ states, which are illustrated and described in more detail in [Section 3.3.1.5, “Frame Queue State.”](#)

#### Dequeue Dispatcher Operation—Dequeue Command Type 1

Dequeue with priority precedence and intra-WQ class scheduling respected—With this dequeue command type, a FQ from the highest priority non-empty WQ is always selected for dequeue. The algorithm for selecting the FQ from which to dequeue is as follows:

1. If there is no active or held active FQ in the portal, then a new FQ is selected for dequeue.
  - The QMan selects the highest priority non-empty WQ in the channel or channels specified in the dequeue command, and selects the FQ at the head of that WQ.
  - The selected FQ moves to either the active or held active state, as indicated by the hold active bit in its FQD. If a force eligible command was issued on the selected FQ, it moves to the held active state regardless of the hold active bit in its FQD.
  - Either one frame or one to three frames (as selected in the dequeue command) are dequeued from the selected FQ and placed in the DQRR. Then the post dequeue response operations are executed (see description below, under “Dequeue Dispatcher Operation—Post Dequeue Response Operations”).
2. If there is an active or held active FQ in the portal, then the priority of the active or held active FQ’s destination WQ is compared with the priority of the other WQ available for dequeue to determine whether or not a new FQ is selected for dequeue.
  - If there are one or more non-empty WQ in the channel(s) selected by the dequeue command whose priority is higher than that of the destination WQ of the Active or Held Active FQ, then QMan determines that there is higher priority work available. The currently Active or Held Active FQ is suspended, and a new FQ is selected for dequeue as in step 1.
  - If the priority of the Active or Held Active FQ’s destination WQ is greater than or equal to the priority of all non-empty WQ in the channel(s) selected by the dequeue command, then QMan determines that there is no higher priority work available, and the Active or Held Active FQ remains eligible for dequeue, subject to the intra-WQ class scheduling check described in step 3.

3. If there is an Active or Held Active FQ in the portal, and the priority evaluation in step 2 determines that there is no higher priority work available, then dequeue dispatcher checks to see if the Active or Held Active FQ's intra-WQ class scheduling consumption allowance was satisfied after the previous dequeue from this FQ.
  - If No, then the Active or Held Active FQ is re-selected for dequeue, and the sequence moves to step 6.
  - If Yes, then dequeue dispatcher checks to see if there is any work available at the same priority. If there is work available at the same priority (there are one or more non-empty WQ at the same priority), then the currently Active or Held Active FQ is suspended, and a new FQ is selected for dequeue as in step 1.  
If there is no work available at the same priority (remember that a check for higher priority work was already done in step 2), then the Active or Held Active FQ is re-selected for dequeue, its intra-WQ class scheduling consumption allowance is re-calculated as if this FQ was newly selected, and the sequence moves to step 6. This last case is an optimization allowing the FQ to remain Active or Held Active if there is no higher or equal priority work available.
4. If there is an Active FQ in the portal, and it was suspended as a result of the evaluations in step 2 or step 3, then the Active FQ is rescheduled.
5. If there is a Held Active FQ in the portal, and it was suspended as a result of the evaluations in step 2 or step 3, then the Held Active FQ is moved to the Held Suspended state.
6. If there is an Active or Held Active FQ in the portal, and it was re-selected for dequeue as a result of the evaluations in step 2 and step 3, then the dequeue operation proceeds on this FQ.
  - Either 1 frame or 1–3 frames (as selected in the dequeue command) are dequeued from the selected FQ and placed in the DQRR. Then, the post dequeue response operations are executed (see description below, under “Dequeue Dispatcher Operation—Post Dequeue Response Operations”).

## **Dequeue Dispatcher Operation—Dequeue Command Type 2**

Dequeue with active FQ precedence, and Intra-WQ Class Scheduling respected—With this dequeue command type, if an Active or Held Active FQ is present in the portal it is always selected for dequeue, overriding the priority of any other WQ in the channel(s) selected by the dequeue command. The algorithm for selecting the FQ from which to dequeue is as follows:

1. If there is no Active or Held Active FQ in the portal, then a new FQ is selected for dequeue, as in step 1 of Dequeue command type 1 described above.
2. If there is an Active or Held Active FQ in the portal, and the Intra-WQ Class Scheduling consumption allowance of this FQ was satisfied after the previous dequeue from this FQ, then the priority of the Active or Held Active FQ's destination WQ is compared with the priority of the other WQ available for dequeue to determine whether or not a new FQ is selected for dequeue.
  - If there are one or more non-empty WQ in the channel(s) selected by the dequeue command whose priority is higher than or equal to that of the destination WQ of the Active or Held Active FQ, then QMan determines that there is higher or equal priority work available. The currently Active or Held Active FQ is suspended, and a new FQ is selected for dequeue as in step 1.

- If the priority of the Active or Held Active FQ’s destination WQ is greater than the priority of all non-empty WQ in the channel(s) selected by the dequeue command, then QMan determines that there is no higher or equal priority work available. The active FQ is re-selected for dequeue, its intra-WQ class scheduling consumption allowance is re-calculated as if this FQ was newly selected, and the sequence moves to step 6.
3. If there is an Active or Held Active FQ in the portal, and the Intra-WQ Class Scheduling consumption allowance of this FQ was not satisfied after the previous dequeue from this FQ, then the Active or Held Active FQ is re-selected for dequeue without priority evaluation, and the sequence moves to step 6.
  4. If there is an Active FQ in the portal, and it was suspended as a result of the evaluation in step 2, then the Active FQ is rescheduled.
  5. If there is a Held Active FQ in the portal, and it was suspended as a result of the evaluation in step 2, then the Held Active FQ is moved to the Held Suspended state.
  6. If there is an Active or Held Active FQ in the portal, and it was re-selected for dequeue as a result of the evaluations in step 2 or step 3, then the dequeue operation proceeds on this FQ.
    - Either 1 frame or 1 to 3 frames (as selected in the dequeue command) are dequeued from the selected FQ and placed in the DQRR. Then the post dequeue response operations are executed (see description below, under “Dequeue Dispatcher Operation—Post Dequeue Response Operations”).

### **Dequeue Dispatcher Operation—Dequeue Command Type 3**

Dequeue with active FQ precedence, and override Intra-WQ Class Scheduling—With this dequeue command type, if an Active or Held Active FQ is present in the portal it is always selected for dequeue, overriding both the priority of any other WQ in the channel(s) selected by the dequeue command, and the Intra-WQ Class Scheduling configured for the FQ. The algorithm for selecting the FQ from which to dequeue is as follows:

1. If there is no Active or Held Active FQ in the portal, then a new FQ is selected for dequeue, as in step 1 of Dequeue command type 1 described above.
2. If there is an Active or Held Active FQ in the portal, then the Active or Held Active FQ is re-selected for dequeue.
  - Either 1 frame or 1 to 3 frames (as selected in the dequeue command) is dequeued from the selected FQ and placed in the DQRR. Then the post dequeue response operations are executed (see description below, under “Dequeue Dispatcher Operation—Post Dequeue Response Operations”).

### **Dequeue Dispatcher Operation—Post Dequeue Response Operations**

After the response to a dequeue command (1 frame or 1 to 3 frames) has been placed into the DQRR, a common set of operations is performed, these are described below:

1. If as a result of this dequeue the FQ has become empty, then this FQ is either rescheduled (if it was moved to the Active state when selected) or moved to the Held Suspended state (if it was moved to the Held Active state when selected).

2. If as a result of this dequeue the DQRR has become full (contains fewer entries than required to execute the next dequeue command), and the FQ is configured with Hold Active = 0 and Avoid Blocking = 1 in its FQD, and this FQ was not presented as a result of a Force Eligible command, then this FQ is rescheduled. Note that such a FQ is never in the Held Active state.
3. If neither of the previous two conditions are met, then this FQ remains in its current state (Active or Held Active), and is available to be selected by the next dequeue command. If as a result of this dequeue the Intra-WQ Class Scheduling consumption allowance of the selected FQ is satisfied, then a bit is set to record the fact that Intra-WQ Class Scheduling has been satisfied for this FQ.

### **3.3.8.2.7 Dequeue Dispatcher Operation—Dequeueing from a Specific WQ**

This section describes in detail the sequence of operations performed when a dequeue command specifies a scheduled dequeue from a specific WQ.

In both the SDQCR and PDQCR (see [Section 3.3.8.2.3, “DQRR Dequeue Commands”](#)), a Source Select (SS) bit is used to specify whether a scheduled dequeue consumes from one or more channels in the portal (SS = 0), or from a specified WQ (SS = 1). This section describes dequeue operation for SS = 1, and [Section 3.3.8.2.6, “Dequeue Dispatcher Operation—Dequeueing from One or More Channels”](#) describes the operation for SS = 0.

In both the SDQCR and PDQCR, a Dequeue Command Type (DCT) field is used to specify exactly how the dequeue dispatcher operations should occur as a result of the command. There are 3 different types of dequeue command, and the details of their operation when dequeuing from a specific WQ is described below, and is illustrated with a flow diagram in [Figure 3-107](#). These descriptions refer to FQ states, refer to [Section 3.3.1.5, “Frame Queue State”](#) for an illustration and more detail on these FQ states.

#### **Dequeue Dispatcher Operation—Dequeue Command Type 1**

Dequeue with priority precedence, and intra-WQ class scheduling respected

#### **Dequeue Dispatcher Operation—Dequeue Command Type 2**

Dequeue with active FQ precedence, and Intra-WQ Class Scheduling respected—When dequeuing from a specific WQ (SS = 1 in the command), these 2 commands produce exactly the same result. Both priority precedence and active FQ precedence do not apply, because the user is requesting a specific WQ. The resulting operation when SS = 1 and DCT = 1 or 2 is:

“Dequeue from the specified WQ, with Intra-WQ Class Scheduling respected”.

The algorithm for selecting the FQ from which to dequeue is as follows:

1. If there is no Active or Held Active FQ in the portal, then a new FQ is selected for dequeue.
  - The QMan selects the FQ at the head of the WQ specified in the dequeue command.
  - The selected FQ moves to either the Active or Held Active state, as indicated by the Hold Active bit in its FQD. If a Force Eligible command was issued on the selected FQ, it moves to the Held Active state regardless of the Hold Active bit in its FQD.
  - Either 1 frame or 1 to 3 frames (as selected in the dequeue command) are dequeued from the selected FQ and placed in the DQRR. Then the post dequeue response operations are executed

(see description below, under “Dequeue Dispatcher Operation—Post Dequeue Response Operations”).

2. If there is an Active or Held Active FQ in the portal, then the WQ ID of the Active or Held Active FQ’s destination WQ is compared with the WQ ID specified in the dequeue command to determine whether or not a new FQ is selected for dequeue.
  - If the two WQ ID are not the same, the currently Active or Held Active FQ is suspended, and a new FQ is selected for dequeue as in step 1.
  - If the two WQ ID are the same, the Active or Held Active FQ remains eligible for dequeue, subject to the intra-WQ class scheduling check described in step 3.
3. If there is an Active or Held Active FQ in the portal, and the WQ ID of its destination WQ is the same as the one requested in the dequeue command, then dequeue dispatcher checks to see if the Active or Held Active FQ’s intra-WQ class scheduling consumption allowance was satisfied after the previous dequeue from this FQ.
  - If No, then the Active or Held Active FQ is re-selected for dequeue, and the sequence moves to step 6.
  - If Yes, then dequeue dispatcher checks to see if the specified WQ is currently empty (that is, if there are any other FQ on this WQ that have frames to be dequeued).
    - If the WQ is not empty, then the currently Active or Held Active FQ is suspended, and a new FQ is selected for dequeue as in step 1.
    - If the WQ is empty, then the currently Active or Held Active FQ is re-selected for dequeue, its intra-WQ class scheduling consumption allowance is re-calculated as if this FQ was newly selected, and the sequence moves to step 6. This last case is an optimization allowing the FQ to remain Active or Held Active if there is no other work available on the specified WQ.
4. If there is an Active FQ in the portal, and it was suspended as a result of the evaluations in step 2 or step 3, then the Active FQ is rescheduled.
5. If there is a Held Active FQ in the portal, and it was suspended as a result of the evaluations in step 2 or step 3, then the Held Active FQ is moved to the Held Suspended state.
6. If there is an Active or Held Active FQ in the portal, and it was re-selected for dequeue as a result of the evaluations in step 2 and step 3, then the dequeue operation proceeds on this FQ.
  - Either 1 frame or 1 to 3 frames (as selected in the dequeue command) are dequeued from the selected FQ and placed in the DQRR. Then the post dequeue response operations are executed (see description below, under “Dequeue Dispatcher Operation—Post Dequeue Response Operations”).

### **Dequeue Dispatcher Operation—Dequeue Command Type 3**

Dequeue with active FQ precedence, and override Intra-WQ Class Scheduling—When dequeuing from a specific WQ (SS = 1 in the command), active FQ precedence does not apply since the user is requesting a specific WQ. The resulting operation when SS = 1 and DCT = 3 is:

“Dequeue from the specified WQ, and override Intra-WQ Class Scheduling”.

The algorithm for selecting the FQ from which to dequeue is as follows:

1. If there is no Active or Held Active FQ in the portal, then a new FQ is selected for dequeue, as in step 1 of Dequeue command type 1 described above.

2. If there is an Active or Held Active FQ in the portal, then the WQ ID of the Active or Held Active FQ's destination WQ is compared with the WQ ID specified in the dequeue command to determine whether or not a new FQ is selected for dequeue.
  - If the two WQ ID are not the same, the currently Active or Held Active FQ is suspended, and a new FQ is selected for dequeue as in step 1.
  - If the two WQ ID are the same, the Active or Held Active FQ is re-selected for dequeue, and the sequence moves to step 5.
3. If there is an Active FQ in the portal, and it was suspended as a result of the evaluation in step 2, then the Active FQ is rescheduled.
4. If there is a Held Active FQ in the portal, and it was suspended as a result of the evaluation in step 2, then the Held Active FQ is moved to the Held Suspended state.
5. If there is an Active or Held Active FQ in the portal, and it was re-selected for dequeue as a result of the evaluation in step 2, then the dequeue operation proceeds on this FQ.
  - Either 1 frame or 1 to 3 frames (as selected in the dequeue command) are dequeued from the selected FQ and placed in the DQRR. Then the post dequeue response operations are executed (see description below, under “Dequeue Dispatcher Operation—Post Dequeue Response Operations”).

## **Dequeue Dispatcher Operation—Post Dequeue Response Operations**

After the response to a dequeue command (one frame or one to three frames) has been placed into the DQRR, a common set of operations is performed, these are described below:

1. If as a result of this dequeue the FQ has become empty, then this FQ is either rescheduled (if it was moved to the Active state when selected) or moved to the Held Suspended state (if it was moved to the Held Active state when selected).
2. If as a result of this dequeue the DQRR has become full (contains fewer entries than required to execute the next dequeue command), and the FQ is configured with Hold Active = 0 and Avoid Blocking = 1 in its FQD, and this FQ was not presented as a result of a Force Eligible command, then this FQ is rescheduled. Note that such a FQ is never in the Held Active state.
3. If neither of the previous two conditions are met, then this FQ remains in its current state (Active or Held Active), and is available to be selected by the next dequeue command. If as a result of this dequeue the Intra-WQ Class Scheduling consumption allowance of the selected FQ is satisfied, then a bit is set to record the fact that Intra-WQ Class Scheduling has been satisfied for this FQ.

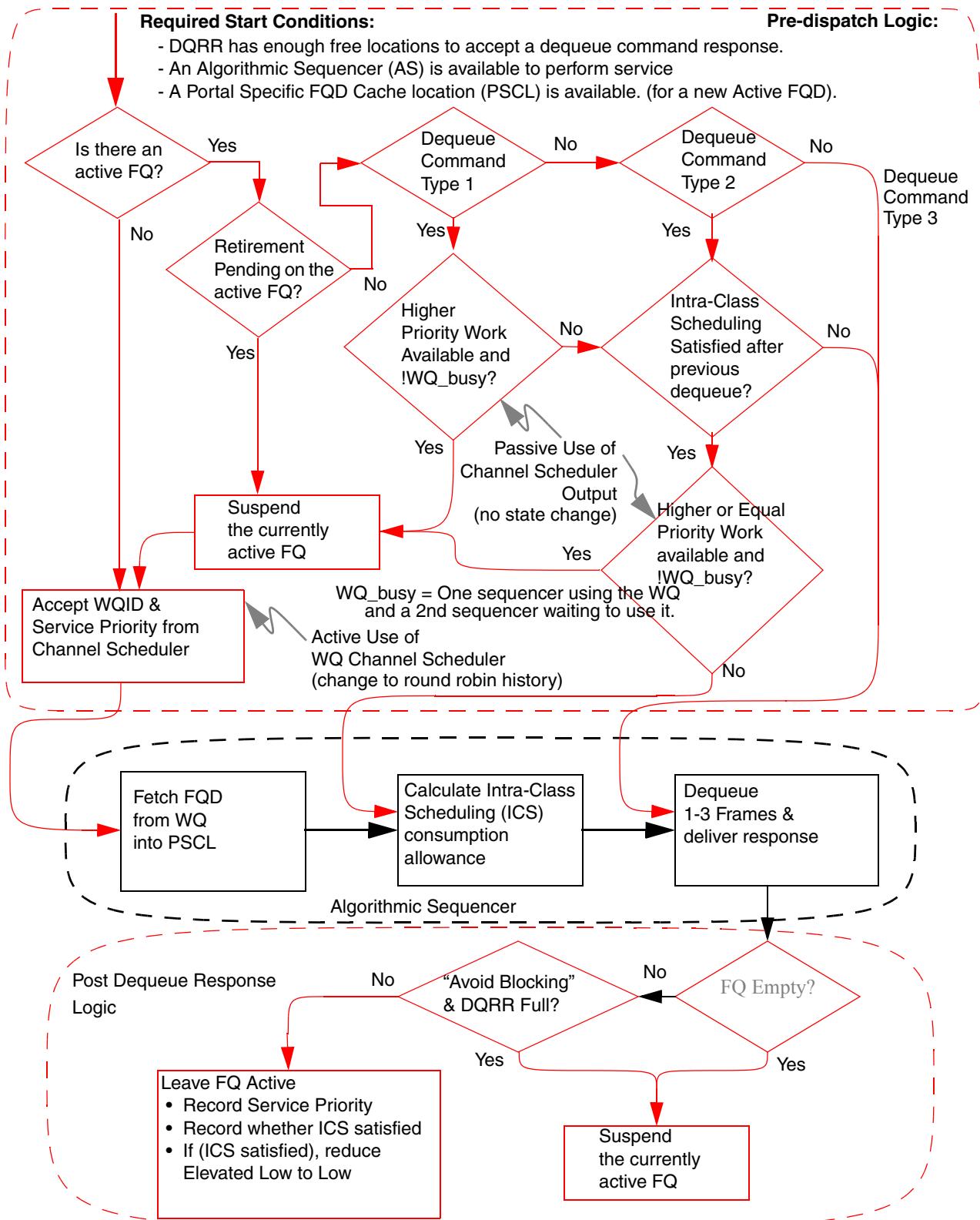


Figure 3-106. Dequeue Dispatcher Operation Flow Diagram—Dequeueing from One or More Channels

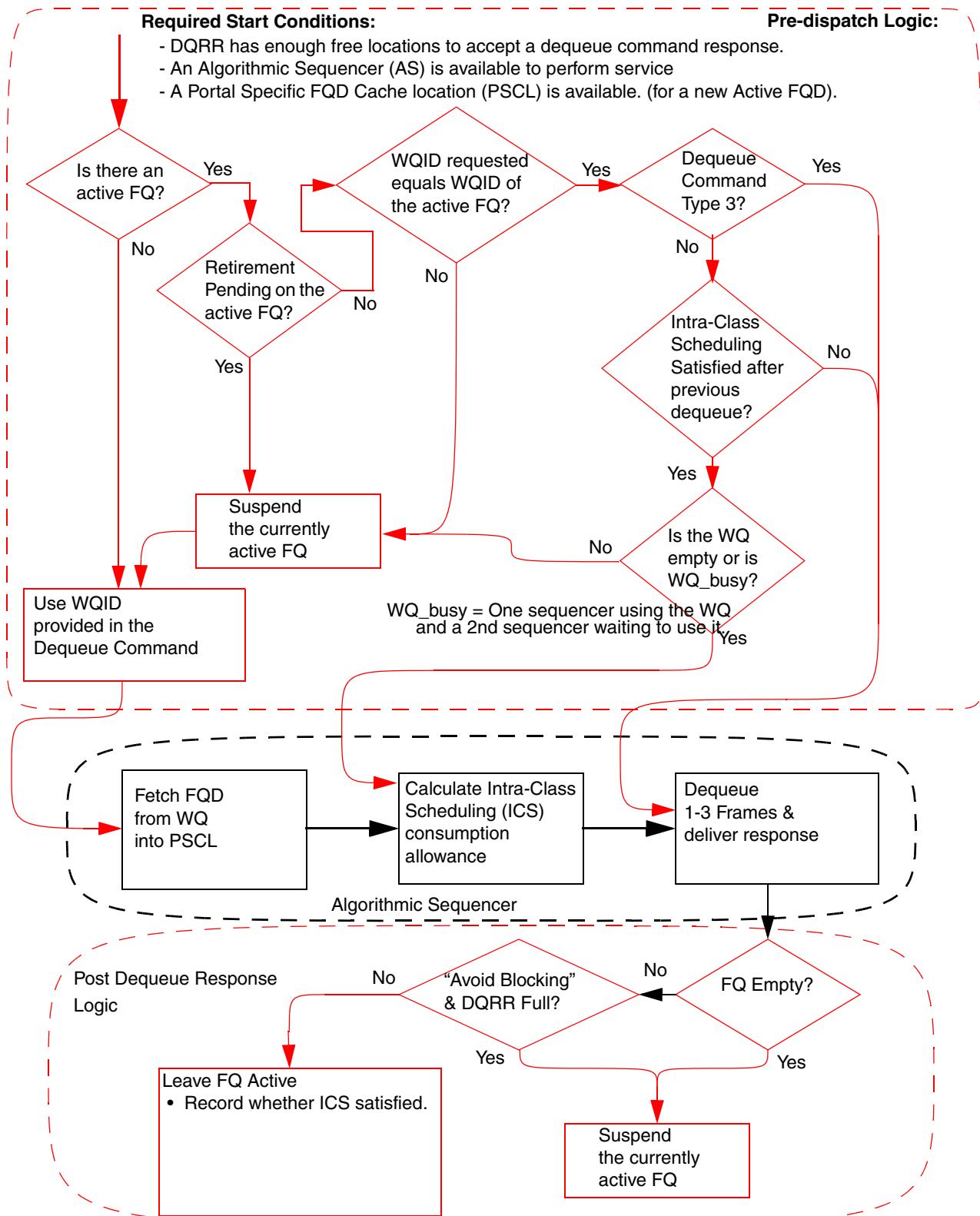


Figure 3-107. Dequeue Dispatcher Operation Flow Diagram—Dequeueing from a Specified WQ

### 3.3.8.2.8 Active and Suspended Frame Queues

#### What is a PSCL?

Each software portal owns and manages four portal-specific cache locations (PSCL). A PSCL is a temporary holding memory for FQDs that have been dequeued by the portal. A FQD may be placed in a PSCL after a dequeue occurs, and PSCL are searched along with all other FQD cache locations when an enqueue operation is executing, to allow enqueues to occur on a FQ stored in a PSCL.

One of the PSCLs is used to hold the FQD of a FQ currently in the Active or Held Active state, that is, the FQ from which a subsequent dequeue can occur. Any of the four available PSCLs may be used for this purpose. Each PSCL carries state information indicating whether or not it contains the Active FQD for this portal. There can be only one Active FQ in each portal at any time.

#### Active and Held Active State Considerations

Note that the Active and Held Active states are mutually exclusive and are associated with a portal. Each portal may contain one FQ occupying either the Active or Held Active state, but these two states are never occupied in the same portal at the same time. When a FQ is selected for dequeue, it is placed in the Held Active state if configured to do so (via the Hold\_Active bit in its FQD) or if a Force Eligible command was performed on it, otherwise it is placed in the Active state.

#### FQs in the Active State

If a FQ selected for dequeue is in the Active (not Held Active) state, when dequeues from this FQ are complete (such as when a new FQ is selected for dequeue due to priority precedence, or this FQ's Intra-WQ Class Scheduling consumption allowance is satisfied), this Active FQ is released and rescheduled, and it cannot be held suspended or parked. The release of the Active FQ occurs without waiting for its associated entries in the DQRR to be consumed.

#### FQs in the Held Active State

If a FQ that was selected for dequeue is Held Active, when dequeues from this FQ are complete it is moved from the Held Active to the Held Suspended state, and the state of the PSCL containing its FQD is changed accordingly. Any or all of the four PSCLs may be in the Held Suspended state at any time. FQs that are in the Held Suspended state can no longer be dequeued from; they are simply being held, pending receipt of a consumption notification from software. The release of a held-active or held-suspended FQ occurs when all DQRR entries associated with that FQ (there may be one or more frames in the DQRR associated with a held FQ) have been consumed. When released, a held FQ is either rescheduled or parked.

#### FQs in the Active State

When a FQ in the Active state is released as described above, the portal logic marks this FQ as suspended while pending a reschedule, and the FQ continues to occupy its PSCL until it is rescheduled by an AS. This is done as an optimization to allow a new Active FQ to be brought into the portal and dequeued from before the reschedule of the formerly-active FQ occurs.

## PSCLs and the DQRR

Using the four PSCLs, the DQRR can contain up to four FQs that are held in the portal at any given time. Each of the PSCLs carries information indicating the state of the FQ that it contains (Empty, Active, Held Active, Held Suspended, or Suspended Pending Re-Schedule). For any non-empty PSCL, the state field in the FQD that it contains (see [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\)”](#)) carries the same state value (5 = Active), and the exact state of the FQ is indicated only by the state of its PSCL.

After all four of the PSCLs in a portal are occupied, no new dequeues can occur until at least one of the four held FQs is released and rescheduled. If all FQs are configured with Hold\_active = 0, or if multiple dequeues are occurring from the same, held-active FQ, up to 15 frames can be placed in the DQRR. If however, all FQs are configured to be held active but only a few frames are pulled from each FQ (for example due to Intra-WQ Class Scheduling), then the number of PSCL limits the number of frames in the DQRR, and in the minimum case of one frame per dequeue the DQRR may hold only four frames before dequeues are stopped due to depletion of PSCLs.

### 3.3.8.2.9 Dequeue Command Behavior when No Frames are Available for Dequeue

In push mode, a command in VDQCR stops automatically when the specified FQ goes empty, so a VDQCR command issued to a FQ that is already empty terminates after one attempt and returns a null response (a single DQRR entry with STAT[3] = 0). The FQ empty and VDQCR command expired bits in the STAT field of the response (see [Section 3.3.9.2, “Frame Dequeue Response”](#)) is asserted in this response.

For the command in SDQCR, dispatching of the command to an AS will be stopped when there is nothing available for dequeue, and then automatically re-started when the specified channel(s) or WQ becomes non-empty. This means the SDQCR command will wait until something becomes available for dequeue, and will be silent until then. This is done using the same dequeue availability signals that drive the dequeue availability interrupts.

In pull mode, each command issued to PDQCR should return a response in DQRR, therefore a null response (a single frame dequeue response with STAT[3] = 0) will be returned for each write to PDQCR which dispatches a dequeue command and which returns no frames because all of the specified channels (or specified WQ or specified FQ) are empty. If a scheduled dequeue was requested, the FQID returned will be 0. If an unscheduled dequeue was requested, FQID will be valid and the FQ empty bit in the STAT field will be set.

In either mode, if a Force Eligible command is executed on a FQ which is empty, then a null response (a single frame dequeue response with STAT[3] = 0) will appear in the DQRR, along with the Force Eligible bit set in the STAT field of the response, to indicate this to software.

### 3.3.8.2.10 Dequeue Command Behavior with Parked and Retired Frame Queues

Frame Queues which are in the Parked or Retired state (see [Section 3.3.1.5, “Frame Queue State”](#)) are never scheduled for dequeue via QMan’s WQs and channels, even if these FQ contain one or more frames. Instead, the intended consumer must maintain the knowledge that these FQ exist in the system and must dequeue from them using specific commands to QMan. The consumer can request a dequeue from a parked or retired FQ by using a dequeue command which specifies an unscheduled dequeue, and using a specific FQID in the command.

In the case of a retired FQ, the consumer (which is always a software portal) is informed of the presence of this FQ, and whether or not it contains any frames to be dequeued, via a FQRN or FQRNI message delivered in the Message Ring (MR) (see [Section 3.3.9.4, “FQ State Change Notification Message Response”](#)).

In the case of a parked FQ, the FQ may be initialized in the parked state and left in this state permanently, an example application for such a FQ would be inter-processor communications. In this case the intended consumer must either be informed by some out of band means that such a FQ contains frames ready to be dequeued, or it may poll (by issuing an unscheduled dequeue command to) the FQ occasionally to see if any frames have arrived on this FQ. An unscheduled dequeue command issued on a parked and empty FQ will return a null dequeue response, with 0 frames delivered.

When an unscheduled dequeue command is executed by QMan, the FQ stays active only for the duration of a single dequeue operation, with 1 to 3 frames dequeued, and then is released from the portal. As such, the PSCL in the portal are not used, and Intra-WQ Class scheduling does not apply to these FQ. Such a dequeue does not displace any FQ in the PSCLs, and can occur even if all PSCL are occupied.

### 3.3.8.2.11 DQRR Interrupts

Interrupts can be generated from the DQRR in the following cases:

- Interrupt when DQRR is non-empty.  
This interrupt occurs when the ring contains more entries than set in a programmable threshold (see [Section 3.2.3.19, “QCSP DQRR Interrupt Threshold Register \(QCSPi\\_DQRR\\_ITR\)”](#)), or when the ring has been non-empty for longer than a programmed time out period (see [Section 3.2.3.25, “QCSP Interrupt Time-Out Period Registers \(QCSPi\\_ITPR\)”](#)). Ring occupancy is equal to DQRR\_PI minus DQRR\_CI, and this interrupt is asserted when ring occupancy is greater than the threshold. A threshold of 15 means the interrupt will assert only when the time out period is reached.
- Interrupt on dequeue availability. This is described in [Section 3.3.4.1, “Dequeue Availability.”](#)

### 3.3.8.3 Message Ring (MR)

The Message Ring (MR) is used to issue messages other than frame dequeue responses from QMan to software.

The MR is an 8 entry circular FIFO contained in the cache-enabled area of QMan’s software portal, its format is described in [Section 3.2.3.3, “QCSP Message Ring Registers \(QCSPi\\_MR0-7\)”](#). Each entry is 64 bytes in size and carries either a ERN message as described in [Section 3.3.9.3, “ERN Message Response,”](#) or a FQ state change notification message as described in [Section 3.3.9.4, “FQ State Change Notification Message Response.”](#) QMan produces entries into the MR by writing messages into the ring entries, and software consumes and processes these messages by reading them from the ring entries.

From the software side MR entries are read-only. Since these entries are cache-enabled, software must invalidate them in the processor’s cache before reading them. QMan will not stash the MR entries into a processor’s cache.

Each MR (one per software portal) is backed internally by an ERN queue, described in [Section 3.3.14.5, “Enqueue Rejections.”](#) This allows a larger number of messages to be queued for a software portal, with

the first 7 of these messages stored in the MR and the remainder extending into the ERN queue. As software consumes messages from the MR, messages are moved from the ERN queue to the MR as needed until the ERN queue is empty. The ERN queue itself may use a limited amount of internal memory, and may then extend into system memory (using QMan's reserved PFDR area). As a consequence, if MR messages are allowed to build up indefinitely then PFDR resources can become exhausted and cause enqueues to stall. Therefore software must consume MR messages on a regular basis to avoid depleting the available PFDR resources.

### 3.3.8.3.1 MR Production Notification

Production notification from QMan to software includes both an alternating polarity valid bit in each MR entry and a read-only 3 bit Producer Index (MR\_PI).

The value of MR\_PI indicates the ring entry in which the next valid response will be placed. The ring contains valid entries to be consumed when MR\_PI and MR\_CI are not equal.

After reset MR\_PI is set to 0, and it is incremented by QMan when valid entries are written into the ring, wrapping from 7 back to 0 when the end of the ring is reached. MR\_PI is a 3 bit, read-only value. The same value is available in three different locations, a cache-inhibited location and two cache-enabled locations. Also, in one of the cache-enabled locations (RORI\_CENA), MR\_PI is combined with the other read-only indices from the EQCRs and DQRPs such that all indices can be obtained with a single, cache-enabled read. See [Section 3.2.3.14.1, “QCSP MR Producer Index Cache-Enabled Registers \(QCSPi\\_MR\\_PI\\_CENA\)”](#) and [Section 3.2.3.16, “QCSP Read-Only Ring Indices Cache-Enabled Registers \(QCSPi\\_RORI\\_CENA\).”](#)

An alternating polarity valid bit is also provided in each entry, and allows software to detect when entries are added to the MR by polling on the valid bit in the next expected ring entry, without having to explicitly read the MR\_PI value from QMan. The alternating polarity valid bit in the MR works the same way as that in the DQRR; see also [Section 3.3.8.2.1, “DQRR Production Notification.”](#)

### 3.3.8.3.2 MR Consumption Notification

Consumption notification from software to QMan is tracked using a Consumer Index value (MR\_CI) stored in QMan. MR\_CI is a 3 bit value that indicates the ring entry which is next to be consumed by software. QMan uses MR\_CI to determine when the ring is full and cannot accept a new entry.

After reset MR\_CI is set to 0, and it must be incremented when responses are consumed from the ring, wrapping from 7 back to 0 when the end of the ring is reached. There are two modes that can be selected for the update method of this value. The mode is programmable in a control register within each portal; see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\).”](#)

- CI write mode, cache-inhibited.
  - In this mode, MR\_CI is a 3 bit value in a cache-inhibited read/write register.
  - The register is mapped on a word boundary, and software issues a cache-inhibited store to this word to advance MR\_CI after it has consumed a MR entry. Writes to MR\_CI do not need to occur after consumption of each entry, a single write can be used to advance MR\_CI after several entries have been consumed, thus amortizing the cost of the write over multiple message responses in MR.

- CI write mode, cache-enabled.
  - In this mode, MR\_CI is a 3 bit value in a cache-enabled read/write register.
  - The register is mapped on a 64 byte boundary, and software issues store followed by cache line flush to advance MR\_CI. As with the cache-inhibited mode, the write to this register can be amortized over the consumption of several MR entries.

### 3.3.8.4 Management Command Register (CR)

The Management Command Register (CR) is used by software to issue management commands from software to QMan. The CR is 64 bytes in size and is contained in the cache-enabled area of QMan’s software portal, its format is described in [Section 3.2.3.4, “QCSP Management Command Registers \(QCSPi\\_CR\).”](#) The different commands that can be issued by writing to the CR are described in [Section 3.3.9.5, “Frame Queue Management Commands”](#) and in [Section 3.3.9.6, “Congestion Group Management Commands.”](#)

The CR is also readable, to accommodate a castout that may occur while a command is being written into cache. A read of the CR returns all data as last written. From the QMan side, the CR is read-only, QMan does not modify the data in the CR in any way.

Only one command at a time can be issued to the CR. Once a command is issued, software must wait for the CR to become available again before issuing a new command to the CR. The indication to software that the CR is available to receive a new command is when the response to the previous command becomes valid in the Management Response Register (RR0 or 1) associated with the command.

Both an alternating polarity valid bit and the presence of a non-zero command verb are used to indicate when a valid and complete command has been written to the CR.

To write a command into the CR, software must zero the command cache line, write all words of the command other than word 0, issue a synchronizing instruction, write word 0 (which contains the command verb and its valid bit), then flush the command from cache to QMan. The steps of writing word 0 last and using a synchronizing instruction before writing this word are needed to ensure that a castout of a partially completed command from the processor’s cache will always write a 0 (null) command verb, thus preventing QMan from attempting to execute this partial command.

The alternating polarity valid bit in the CR works as follows: When software writes a command verb into the CR, it also sets the valid bit (the msb of the command verb) to the current valid bit polarity. For example, the very first command written after reset must have a 1 in its valid bit, since this is the valid bit polarity after reset. Software must track the current valid bit polarity (as mentioned, it is 1 after reset), and must toggle this current polarity each time a new command is written to the CR. Therefore the first command issued to CR will have a valid bit of 1, the next command will have a valid bit of 0, the next one will have a valid bit of 1, and so on. The use of the alternating polarity valid bit is used to protect against the possibility of a speculative read of the CR followed by a castout of from the processor’s cache (see also [Section 3.3.8.1.5, “Speculative Reads of the EQCR Cache-Enabled Entries when Using Implicit Production Notification”](#)).

QMan also locally tracks the current valid bit polarity. QMan will recognize a complete command ready for execution when a write to the CR with both a valid bit of the expected polarity and a non-null command

verb is received. If a write to the CR is received in which the verb contains zero, or the valid bit is of the wrong polarity, QMan will store the written data in the CR but will not execute the command.

### 3.3.8.5 Management Response Registers (RR0/RR1)

The management response registers (RR0 and RR1) are used by software to read the response to management commands issued from software to QMan in the CR. Each RR is 64 bytes in size and is contained in the cache-enabled area of QMan’s software portal, its format is described in [Section 3.2.3.5, “QCSP Management Response Registers \(QCSPi\\_RR0–1\) Format.”](#) The different responses that can be received in the RR are described in [Section 3.3.9.5, “Frame Queue Management Commands,”](#) and in [Section 3.3.9.6, “Congestion Group Management Commands.”](#)

Every command issued in the CR with a non-zero value in command verb bits 1–2 results in a response. The response to a CR command is always returned in either RR0 or RR1. The RR which is used for the response to a command is indicated by the polarity of the valid bit in that command, therefore if the valid bit in the command is 0 then RR0 is used to return the response to that command, and if the valid bit in the command is 1 then RR1 is used to return the response to that command. In this way, the RR used to return a response alternates with every command issued, starting with RR1 after reset (since the polarity of the valid bit is 1 after reset).

QMan contains only one internal storage register for management command responses. When this internal register contains a valid response it is visible in either the RR0 or the RR1 location, as indicated by the valid bit polarity in the command which initiated the response. Whenever the response in RR0 is valid, a read of RR1 will return all 0 data, and conversely when RR1 contains a valid response a read of RR0 will return all 0 data. After reset, and whenever a write of a valid command to the CR is detected, the response held in the internal register is cleared, and reads from both RR locations will return all 0 data. Once a command completes and its response is placed in the internal response register, this response will be available for read in the RR location indicated by the command’s valid bit polarity. This response may be read multiple times, and remains available for read until a new command with a valid bit of the expected polarity and a non-zero command verb is written to the CR.

If software ever loses synchronization with QMan and needs to determine the expected polarity of the valid bit for the next CR command, this can be done by reading both RR and seeing which of the two contains a valid response. The expected valid bit polarity for the next CR command is 0 if RR1 contains a valid response, and is 1 if RR0 contains a valid response. If both RR contain all 0, this indicates either that no command has been executed since reset (in which case the expected valid bit polarity is 1), or that a command is currently in progress.

The use of the two alternating response registers is provided as an assist in handling the weak ordering inherent to the Power architecture. As mentioned above, while a CR command is pending, or is being executed but is not yet complete, any read of the RR will return a null (all 0) response, until the command completes. If a read arrives after the command has completed, it returns the valid response to that command. However when a new command is issued, followed by a read of its response, the read may bypass the write of the new command, in which case QMan must return a null response for that read since it is intended for the response of the new command that has not yet arrived, rather than the current command which has already completed.

By alternating the response register location used by each command, QMan can tell the difference between a read of the previously completed response and the read of a response whose command has not yet arrived at the CR. The read of the valid response in one RR location will return that response correctly, but the read of the other RR location will always return a null response. When the write of the new command arrives at the CR, both RR locations will become null, and then when this new command completes its response will become available in its associated RR location.

If only a single RR were used to return all responses, a synchronization instruction would be required between the instruction which flushes and writes the command to the CR and the subsequent read of its response. By using the alternating response registers this synchronization instruction is not needed.

### 3.3.8.6 DQRR Entry Stashing

As mentioned earlier, the DQRR is intended to be mapped into a cache-enabled area in the MMU of the processor core servicing the software portal. The reason for this is to improve performance by performing full cache line sized transactions when accessing the entries of the DQRR.

A disadvantage of this approach is that software must explicitly manage the cached copies of the DQRR entries, specifically the cached copy of each DQRR entry must be invalidated before reading a new response in that entry location.

The requirement for software to manage the cached copies of the DQRR can be eliminated by enabling stashing of DQRR entries in QMan. The purpose of DQRR entry stashing is to allow QMan to place (stash) valid DQRR entries directly into the L2 frontside cache, rather than having the processor core read the DQRR entries from QMan. Whenever a new valid DQRR entry is successfully stashed by QMan, that entry in cache is automatically invalidated and replaced with the new entry, and the software reading that DQRR entry can pick it up directly from cache without having to explicitly invalidate the cached copy. To use DQRR entry stashing, the memory area containing QMan's DQRR must be configured as cache-enabled.

When DQRR entry stashing is enabled, QMan will issue a 64-byte write with stash when a response becomes valid in a DQRR entry. The data issued with this write contains the entire DQRR entry. The address used for this write is that of the DQRR entry itself, which in fact means that QMan is performing a write to its own DQRR location. As a result of this QMan will also receive the write transaction targeted to that DQRR entry, but since the DQRR is read-only this received write transaction will be ignored.

### 3.3.8.7 EQCR\_CI Stashing

As mentioned in [Section 3.3.8.1, “Enqueue Command Ring \(EQCR\),”](#) software must flow control its production into the EQCR to prevent the enqueue command ring from overflowing. To do this the EQCR\_CI value must be read by software on a regular basis.

The performance of the EQCR\_CI polling process can be improved by enabling EQCR\_CI stashing in QMan. This works in the same way as the DQRR entry stashing described in the previous section.

The purpose of EQCR\_CI stashing is to allow QMan to place (stash) the EQCR\_CI value directly into a cache at regular intervals, rather than having the processor core read the EQCR\_CI value from QMan. As mentioned in [Section 3.3.8.1.3, “EQCR Consumption Notification,”](#) the same EQCR\_CI value is provided in three different register locations, the location used for EQCR\_CI stashing purposes is the 64 byte aligned

cache-enabled EQCR\_CI\_CENA location. When QMan stashes the EQCR\_CI value, the entire contents of the EQCR\_CI\_CENA register is stashed at the address of the EQCR\_CI\_CENA location.

Whenever a new EQCR\_CI value is successfully stashed by QMan, that value in cache is automatically invalidated and replaced with the new value, and the software reading the EQCR\_CI\_CENA register can pick it up directly from cache without having to explicitly invalidate the cached copy. To use ECQR\_CI stashing, the memory area containing EQCR\_CI\_CENA must be configured as cache-enabled.

When EQCR\_CI stashing is enabled, QMan will issue a 16-byte write with stash after one or more EQCR ring entries are consumed by QMan. The transaction issued is the same as that used for DQRR entry stashing except that the address of the transaction will be that of the EQCR\_CI\_CENA register rather than that of a DQRR entry. The registers used to configure QMan's stashing operations are described in [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\),”](#) [Section 3.2.4.45, “QMan Software Portal \(QCSP\) Base Address Registers,”](#) and [Section 3.2.4.1, “QCSP ICID Configuration Registers \(QCSPi\\_ICID\\_CFG\).”](#)

When EQCR\_CI stashing is enabled (QCSPi\_CFG[EST] != 0), the value in EQCR\_CI\_CENA seen by software must be kept coherent with any stashing writes that may be in flight, therefore a shadow copy of EQCR\_CI\_CENA is created, and this copy is updated only by writes received to it from the system bus.

When EQCR\_CI stashing is enabled, this shadow copy of EQCR\_CI\_CENA contents is returned in response to a read (for example if a core reads EQCR\_CI\_CENA after an invalidation or castout of the cache line containing EQCR\_CI\_CENA has occurred), rather than the actual values of the signals that are behind the EQCR\_CI\_CENA. This way, only values that have been seen as written to EQCR\_CI\_CENA from the system bus are returned in response to a read, and coherency is maintained.

This means that maintaining coherency on EQCR\_CI\_CENA, when EQCR\_CI stashing is enabled, relies on EQCR\_CI stashing writes issued by QMan being reflected back to QMan via the system bus. In general all EQCR\_CI stashing writes issued by QMan are reflected back to QMan as a target, therefore the EQCR\_CI\_CENA shadow copy is updated regularly, although it will often lag the true EQCR\_CI value. The actual values in use by QMan are available in EQCR\_CI\_CINH in case software ever needs to read them.

Note that the data written to EQCR\_CI\_CENA has no effect, it does not modify the actual EQCR\_CI value, or any other internal values within QMan, it only updates the shadow copy of EQCR\_CI\_CENA. So there is no reason for software to write to EQCR\_CI\_CENA, the only writes that are expected to EQCR\_CI\_CENA are those that reflect back to QMan as a result of EQCR\_CI stashing writes that are issued by QMan, and those that may occur as the result of eviction from a processor core's cache.

### **3.3.8.8 Dequeued Frame Data, Annotation, and Context Stashing**

Note: This feature relies on the ability of reads from main memory to allocate into cache. In a SoC where AXI is used as QMan's system interconnect, read allocate functionality does not exist, therefore this entire section does not apply in such systems.

In addition to stashing DQRR entries, QMan can be enabled to stash a small amount of data from the head of the frame, from the annotation area of the frame (see [Section 3.3.1, “Frames”](#)), and a small amount of FQ context data, into a cache when a frame is dequeued and placed in the DQRR. This option can improve

processor core performance by pre-positioning critical data locally in the cache (also known as cache warming), on software's behalf, where it is accessible without the latency of a read from main memory.

To accomplish this cache warming, QMan will issue a read with allocate transaction on AXI. Generically this can be referred to as read allocate stashing, as opposed to write allocate stashing which is used for DQRR entry and EQCR\_CI stashing.

### 3.3.8.1 Cache warming stashing in an AXI platform

In an AXI based SoC, the cache warming is done by issuing a read transaction with read allocate attribute asserted. This will result in a read from main memory with a read response returned to QMan, and with the read data being snarfed on the way by a cache in the system. The read data that returns to QMan is ignored.

In a SoC that uses ARM core clusters with ACP ports, the cache warming stashing reads issued by QMan will use the ACP port of a specific core cluster, and will trigger a read from the cluster to main memory which will then allocate into the L2 of that cluster when the read returns (assuming of course that the data was not already in cache). The read data will then also be returned to QMan via the ACP port of the cluster, and again this returning read data is ignored by QMan.

### 3.3.8.2 Cache warming stashing controls

The source of the address for the cache warming read transaction depends on the data being stashed. To stash data at the head of a frame, and/or the frame annotation, the address is formed from the buffer address and offset values found in the frame's FD (the frame head data starts at buffer address + offset, the frame annotation starts at buffer address + 0). The amount of data (number of 64 byte aligned coherency granules) to be stashed from the head of the frame and the frame annotation is programmable as part of the Context\_A field in the FQD of the FQ on which that frame arrived. The address and amount used for FQ context stashing is also found in the Context\_A field of the FQD. See [Section 3.3.8.3, “FQD Context\\_A Field used for dequeued Frame Data, Annotation, and Context Stashing control.”](#)

Each software portal is provided with an enable bit (see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\)”](#)) for dequeued frame data, annotation, and context stashing in that portal. In order for these stashing transactions to occur, they must be enabled in both the FQD on which a frame arrives and in the portal in which the frame dequeue command was issued.

When multiple items are configured to be stashed by QMan for a particular FQ selected for dequeue, QMan will perform the stash transactions in the following order:

1. Frame Context stash transaction(s) (if any are requested in FQD context\_A)
2. Frame data stash transaction(s) (if any are requested in FQD context\_A)
3. Frame annotation stash transaction(s) (if any are requested in FQD context\_A)
4. DQRR entry stash transaction (if enabled in the portal).

The reasoning behind this ordering is to get all data needed to process the frame into the core's cache before it begins processing the frame, which is triggered by the receipt of the DQRR entry. Note that to ensure this ordering, the priority of DQRR entry stashing and of dequeued frame data, annotation and context stashing should be set to the same value for a particular software portal (see the RP and SP bits in [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\).”](#))

When a new FQ is selected for dequeue in a software portal and the first frame is dequeued from that FQ, stash transactions are issued for all items for which stashing is enabled: FQ context, frame data, frame annotation, and DQRR entry. After the first frame is dequeued, one or two more frames may be dequeued in response to the same dequeue command. Also, once the first dequeue command on that FQ is complete, the FQ may remain in the active or held active state in the portal, allowing subsequent dequeues to be performed on the same FQ. When any subsequent frames are dequeued, new stash transactions are issued for frame data, frame annotation, and DQRR entry, but not for the FQ context, since this context has already been stashed once for this FQ.

When stashing frame data, QMan will only stash data from the buffer indexed by the address in the FD. If the frame is stored in a single buffer this will be data from the frame itself, and if the frame is stored using scatter-gather this data will be the contents of the scatter/gather table. In the scatter/gather case QMan will not descend into the data buffer(s) indexed by the table.

### 3.3.8.8.3 FQD Context\_A Field used for dequeued Frame Data, Annotation, and Context Stashing control

This section describes the contents of the Context\_A field of a FQD when this field is enabled for use as a stashing control field (when the “Context\_A used for stashing” bit in the FQ\_CTRL field of the FQD is asserted). See also [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\).”](#)

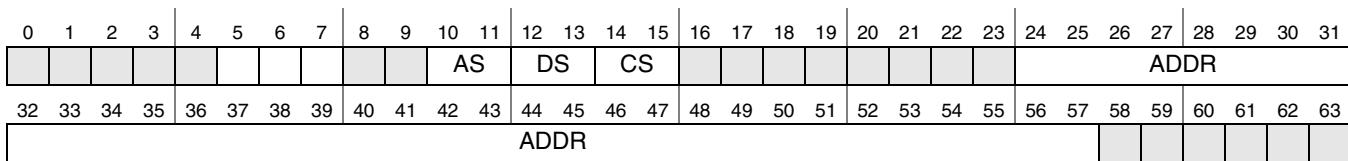


Figure 3-108. FQD Context\_A for dequeued Frame Data, Annotation, and Context Stashing control

Table 3-102. FQD Context\_A field for dequeued Frame Data, Annotation, and Context Stashing control

Field	Description
0-4, 8-9, 16-23	Reserved
10-11 AS	Frame Annotation Stashing Size. Number of 64 byte coherency granules (0, 1, 2, or 3) of Frame Annotation to be stashed.
12-13 DS	Frame Data Stashing Size. Number of 64 byte coherency granules (0, 1, 2, or 3) of Frame Data to be stashed.
14-15 CS	FQ Context Stashing Size. Number of 64 byte coherency granules (0, 1, 2, or 3) of FQ context to be stashed.
24-63 ADDR	FQ Context Address The 40 bit memory address of the first 64 byte coherency granule containing the FQ context information to be stashed. The 6 lsbs (bits 58-63) of this address are not used, QMan issues all stashing transaction as 64 byte transactions aligned on a 64 byte boundary.

The three stashing size parameters provided in FQD[Context\_A] always indicate the maximum amount of data that will be stashed, but QMan also performs a few checks on the frame for which stashing is to be

done, which may result in less data stashed than specified in the stashing size requested in FQD[Context\_A], or in no stashing at all.

For frame annotation stashing, QMan checks the following:

- If the frame format field in the FD specifies a format that does not include an offset field, no frame annotation stashing is done.
- If the offset in the FD is zero, no frame annotation stashing is done.
- If the amount of offset specified in the FD is less than the amount of stashing requested in FQD[Context\_A[AS]], then QMan will stash the coherency granules (CG) containing (buffer address + 0) to (buffer address + offset - 1).

For frame data stashing, QMan checks the following:

- Frame data stashing is done for all frame formats. This can be used to stash actual frame data (for single buffer frames), or the first few entries of a scatter/gather table (for multi-buffer or compound frames).
- If the length in the FD is zero, no frame data stashing is done.
- If the length specified in the FD is less than the amount of stashing requested in FQD[Context\_A[DS]], then QMan will stash the coherency granules (CG) containing (buffer address + offset) to (buffer address + offset + length - 1).

For FQ context stashing, QMan checks the following:

- FQ context stashing is done for all frame formats, and is independent of offset or length.

### **3.3.8.8.4 Dropping of Dequeued Frame Data, Annotation, and FQ Context Stashes**

DQRR entry and EQCR\_CI stashes, when enabled by software, will always be completed by QMan, they will never be dropped. This is because software might typically poll on a DQRR entry and/or an EQCR\_CI\_CENA register in its local cache without invalidating it, therefore the write-with-allocate stash transaction must be issued to invalidate this location in the core's cache, to ensure forward progress in software (even if the snarf is unsuccessful, the invalidate, once issued, is guaranteed to take effect, as described earlier in [Section 3.3.8.6, “DQRR Entry Stashing”](#)). Therefore if a DQRR entry stash needs to be sent as part of a dequeue operation, but QMan's internal stash transaction resources are full, then the algorithmic sequencer performing the dequeue will stall until the DQRR entry stash can be sent.

In contrast, dequeued frame data, annotation, and FQ context stashes (cache warming stash transactions) can be dropped by QMan without causing errors. The effect of a dropped cache warming stash in QMan is the same as an unsuccessful snarf by the processor, the affected location will be invalid in the core's cache, and if the snarf was unsuccessful (due either to the processor being unable to accept the snarf data, or to QMan dropping and never issuing the cachewarming read transaction), then when software reads that piece of data or context the core will read it from main memory rather than from local cache. Therefore dropped cache warming stash transactions should be minimized but are tolerable.

When enabled, dropping of dequeued frame data, annotation, and FQ context stashes in Qman is done by reserving 3 of the 12 entries in each Stash Request Queue (SRQ; see [Section 3.3.8.9, “Stash Transaction Scheduling”](#)) for DQRR entry stashes, this has the effect of allowing the critical DQRR entry stashes to proceed even when dequeued frame data, annotation, and FQ context stashes are not getting through.

When a sequencer performing a dequeue operation needs to send a dequeued frame data, annotation, or FQ context stash transaction that targets an SRQ that contains 9 or more entries, and dropping of such stash transactions is enabled in the software portal in which the dequeue operation is occurring, then that stash transaction is dropped by QMan without stalling the sequencer. This offers the potential benefit of fewer stalls in the algorithmic sequencers.

By default, dropping of dequeued frame data, annotation, and FQ context stashes is disabled. It can be enabled on a per software portal basis, using the SD bit described in [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\).”](#)

- If QCSPi\_CFG[SD] = 0: then no stash transactions are dropped by QMan, and sequencers performing dequeue operations will stall when attempting to issue stash transactions when the targeted SRQ is full.
- If QCSPi\_CFG[SD] = 1: then DQRR entry and EQCR\_CI stash transactions are never dropped, but dequeued frame data, annotation, and FQ context stash transactions may be dropped when the targeted SRQ is almost full, to reduce the potential for stalling of the algorithmic sequencers.

### **3.3.8.9 Stash Transaction Scheduling**

The QMan contains a single Stash Request Queue (SRQ) which is used to carry the stash transactions issued by QMan to the system initiator interface. Each of the software portals within QMan uses this SRQ to forward its stash transactions. The SRQ is 12 entries deep.

12Stash transactions from QMan flow on the same initiator interface as the regular read and write transactions from QMan. The initiator interface contains separate queues for carrying stash transactions (from the SRQ) and regular read and write transactions, and it must schedule transactions from these sources for transmission onto the system interconnect. See [Section 3.3.19.2, “Initiator Scheduling and Priority.”](#)

### **3.3.8.10 Software Portals Virtualization**

Each of QMan’s software portals are located in a separate region of memory, allowing the access permissions for each portal to be configured separately.

In general each of the software portals should be made accessible only to one processor core, and a single piece of interface software (such as a device driver) on that processor core should be used to manage the portal’s resources.

It is possible, and in some cases may be desirable, for a single processor core to manage more than one QMan software portal. An example is when multiple virtual cores are mapped onto a single processor core, with each virtual core running a different OS or software thread. In this case it is expected that each virtual core will have exclusive use of a separate software portal. Any number of QMan software portals can be managed by a processor core, limited of course by the total available number of QMan software portals.

## **3.3.9 Software Portal Commands and Responses**

This section describes the format of commands and responses used in the software portals. Commands from software to QMan are issued using either the Enqueue Command Ring (EQCR), the Management

Command Register (CR), or one of the dequeue command registers. Responses from QMan to software are carried in the Dequeue Response Ring (DQRR), the Message Ring (MR), or in the Management Response Register (RR0/1).

All commands and responses passed between software and QMan are 64 bytes in length, and the memory area containing the data structures in which these commands and responses are carried is intended to be configured as cache-enabled. The various commands and responses are described in the following sub-sections, and the format of each command and response is shown with byte granularity, from byte 0 to byte 63.

### 3.3.9.1 Enqueue Command

The enqueue command is used to enqueue a single frame onto a FQ, with full flexibility in specifying all needed parameters for the enqueue operation. Enqueue commands are carried from software to QMan in the EQCR; see [Section 3.2.3.1, “QCSP Enqueue Command Ring Registers \(QCSPi\\_EQCR0–7\).”](#)

If the enqueue is successful, no response is generated. If the enqueue is rejected (for example due to RED/WRED/tail drop rejection), a notification of this rejection will be issued by QMan using an Enqueue Rejection Notification (ERN) which will be sent to software via the Message Ring (MR); see also [Section 3.3.14.5, “Enqueue Rejections”](#) and [Section 3.3.9.3, “ERN Message Response.”](#)

If the original frame received by software as the result of a dequeue must be fragmented into multiple frames before transmission, multiple enqueue commands must be used to send all of the fragments. In this case, if order restoration is used, each of the fragments carries the same sequence number as the original frame, and the Not Last In Sequence bit (in the msb of the sequence number) is used to indicate that more fragments using the same sequence number are coming in later commands. This bit must be negated in the command which carries the last fragment of the original frame.

The enqueue command also supports the ability to remove a frame’s sequence number from an ORP without enqueueing it. This would be used when software has received a frame that needs to be discarded but whose sequence number is expected to arrive at an ORP.

The enqueue command also supports the ability to issue a DQRR discrete consumption acknowledgment (DCA) along with the enqueue command. A DQRR DCA is used to notify QMan that an entry in the DQRR has been consumed by software, and using a DCA embedded in an enqueue command provides an optimization which reduces the number of writes required to issue DQRR consumption acknowledgments, in that each frame that is dequeued in a DQRR entry can be used to consume that DQRR entry when the frame is re-enqueued via the EQCR.

The use of a DQRR DCA embedded in an enqueue command also provides an assist for implementing an order preservation paradigm for frame processing; see [Section 3.3.8.2.2, “DQRR Consumption Notification.”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	DCA	SEQNUM		ORP				FQID				TAG			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FD															
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-109. Enqueue Command Format

**Table 3-103. Enqueue Command Format**

Field	Description
0 VERB	<p>Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.1, “QCSP Enqueue Command Ring Registers (QCSPi_EQCR0–7).”</a></p> <p>Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed.</p> <p>Bit 1-2: Must be 0 for an enqueue command. If a command is written into an EQCR entry with these bits non-zero, the command is consumed from EQCR without being executed, and an ICVI interrupt is asserted in QMAN_ERR_ISR. No enqueue rejection notification (ERN) is produced.</p> <p>Bit 3-4: FC: Frame Color. This field carries Frame Color information for the frame to be enqueued.</p> <ul style="list-style-type: none"> <li>00: Green</li> <li>01: Yellow</li> <li>10: Red</li> <li>11: Override. Enqueue regardless of RED/WRED/tail drop thresholds.</li> </ul> <p>Frame color is provided to QMan whenever an enqueue operation is performed. It is used by QMan to determine whether the enqueue operation succeeds. For instance, color is used in the RED/WRED calculations. It is not stored with the frame, after QMan uses color to determine whether an enqueue is successful the color value is discarded.</p> <p>Those entities which do not explicitly determine the color of frames should use a color of green for the frames which they enqueue.</p> <p>Bit 5: Interrupt on command dispatch. 1 = Assert interrupt when the command is dispatched for execution. 0 = no interrupt.</p> <p>Bit 6: Order Restoration enable. 1 = Use order restoration for this enqueue. The SEQNUM and ORP fields in the command are valid. 0 = Order restoration not used. The SEQNUM and ORP fields in the command are not used.</p> <p>Bit 7: Enqueue enable. 1 = Perform an enqueue. The FQID and FD fields in the command are valid. 0 = Do not perform an enqueue. The FQID and FD fields in the command are not used.</p> <p>Valid command encodings for bits 6-7 are: 0 = Invalid command. If bit 1-2 of the command verb are 0, and bit 6-7 are zero, an ICVI interrupt is asserted in QMAN_ERR_ISR and an enqueue rejection notification (ERN) is also generated. Note however that in valid bit mode, a write with all bits of the command verb (bits 1-7) equal to 0 is treated as a castout of a partially completed command (rather than an invalid command), and the ring entry is considered not yet produced (EQCR_PI is not advanced). See <a href="#">Section 3.3.8.1.2, “EQCR Production Notification.”</a> 1 = Enqueue to specified FQ. 2 = ORP operation without enqueue. Bit 0 in the SEQNUM field identifies the operation type. 3 = Enqueue to specified FQ through the specified Order Restoration Point (ORP).</p>

**Table 3-103. Enqueue Command Format (continued)**

Field	Description
1 DCA	<p>DQRR Discrete Consumption Acknowledgment (DCA). This byte is used only if the DQRR consumption notification mode of this portal is programmed for DCA mode (see <a href="#">Section 3.2.3.17, “QMan Software Portal Configuration Register (QCSPi_CFG)”</a>). If the DQRR consumption notification mode is programmed for one of the two consumer index write modes, then this byte is ignored by QMan.</p> <p>Bit 0 (msb): DCA_EN: DCA Enable. If this bit is asserted, and the DQRR consumption notification mode of this portal is programmed for DCA mode, then the PK and DCAP_CI fields in this byte are valid and used to acknowledge consumption of a single DQRR entry. Note that for a DCA embedded in an enqueue command, only a single DQRR entry at a time can be acknowledged. If this bit is negated, the PK and DCAP_CI fields are ignored and no DQRR entries are consumed.</p> <p>Bit 1: PK: Park: See the description of the PK bit in the DQRR_DCAP register in <a href="#">Section 3.2.3.10, “QCSP DQRR Discrete Consumption Acknowledgment and Park (QCSPi_DQRR_DCAP)”</a>.</p> <p>Bit 2-3: Reserved</p> <p>Bit 4-7 (lsbs): DCAP_CI: Discrete Consumption Acknowledgment and Park Consumer Index. See the description of the DCAP_CI field in the DQRR_DCAP register in <a href="#">Section 3.2.3.10, “QCSP DQRR Discrete Consumption Acknowledgment and Park (QCSPi_DQRR_DCAP)”</a>.</p>
2-3 SEQNUM	<p>Order Restoration Sequence Number. See <a href="#">Section 3.3.13.5.2, “Using Sequence Numbers”</a>. This field is used only when an enqueue through an order restoration point (ORP) is performed, VERB bit 6 = 1.</p> <p>bit 0 (msb): ORP operation type. Valid only if VERB bit 6-7 = 2. 0 = Remove the specified sequence number from the specified ORP without enqueue. This is used by software when a frame is to be discarded and its sequence number needs to be removed from an ORP. 1 = Advance NESN to indicated sequence number. See <a href="#">Section 3.3.13.5.2, “Using Sequence Numbers”</a>.</p> <p>bit 1: NLIS: Not Last In Sequence. This bit is used to delineate enqueue commands that carry multiple fragments from an original single frame. If asserted, this bit indicates that more fragments using the same sequence number are coming in later commands. This bit must be negated in the command which carries the last fragment of the original frame, or in a command which carries any non-fragmented frame. Note that if this bit is asserted in a Remove Sequence Number command (VERB bits 6-7 = 2, and SEQNUM bit 0 = 0), the specified sequence number is not removed, since NLIS = 1 implies that one or more commands with the same sequence number will follow.</p> <p>bits 2-15: Sequence number. Sequence numbers are used by QMan to determine whether the enqueue is performed immediately (because the frame is the next in sequence for that ORP) or delayed until other frames have passed through the ORP.</p>
4	Reserved (for possible future FQID expansion)
5-7 ORP	<p>Order Restoration Point ID. This field is used only when VERB bit 6 is asserted. Identifies the ORP to be used for this command. If the specified ORP is disabled (that is, the ORP Enable bit is 0 in the FQ_CTRL field of the FQD specified by this ORP value), then this command is rejected and returned to software in an ERN message; see <a href="#">Section 3.3.9.3, “ERN Message Response”</a>.</p>
8	Reserved (for possible future FQID expansion)
9-11 FQID	Frame Queue ID. If VERB bit 7 is asserted, identifies the FQ onto which the specified frame should be enqueued.
12-15 TAG	Enqueue command Tag. This tag is returned in the Enqueue Rejection Notification (ERN; see <a href="#">Section 3.3.14.5, “Enqueue Rejections”</a> ) if this enqueue command is rejected, and can be used to associate an ERN with the enqueue command that has been rejected, for example, the tag could contain the address of a callback routine in an upper layer of software. For enqueue commands that are successful, this tag is not used.

**Table 3-103. Enqueue Command Format (continued)**

Field	Description
16-31 FD	Frame descriptor Frame descriptor (FD) for the frame to be enqueued. This FD must always be valid in a valid enqueue command; if it is not, an error interrupt will be signaled. See <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs)”</a> for a description of the contents of a FD.
32-63	Reserved

### 3.3.9.2 Frame Dequeue Response

The frame dequeue response is used to present a single frame which has been dequeued from one of QMan’s frame queues (FQ) to software. Frame dequeue responses are carried from QMan to software in the DQRR; see [Section 3.2.3.2, “QCSP Dequeue Response Ring Registers \(QCSPi\\_DQRR0-15\)”](#).

Frame dequeue responses into the DQRR are produced by QMan as the result of the execution of a dequeue command, and dequeue commands are provided to QMan via the SDQCR and VDQCR registers (when the DQRR is programmed in Push mode) or via the PDQCR register (when the DQRR is programmed in Pull mode). See [Section 3.2.3.11, “QCSP DQRR Static Dequeue Command Register \(SDQCR\),”](#) [Section 3.2.3.12, “QCSP DQRR Volatile Dequeue Command Register \(QCSPi\\_DQRR\\_VDQCR\),”](#) and [Section 3.2.3.13, “QCSP DQRR Pull Dequeue Command Register \(QCSPi\\_DQRR\\_PDQCR\).”](#)

The DQRR consists of 16 separate 64 byte entries which are organized as a circular FIFO. Each DQRR entry occupies 64 bytes, and the format of a single DQRR entry for a frame dequeue response is shown below with byte granularity, from byte 0 to byte 63.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	STAT	SEQNUM		TOK					FQID			CNTXTB			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
FD															
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-110. Frame Dequeue Response Format**

**Table 3-104. Frame Dequeue Response Format**

<b>Byte Field</b>	<b>Description</b>
0 VERB	<p>Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.2, “QCSP Dequeue Response Ring Registers (QCSPi_DQRR0-15).”</a></p> <p>Bit 1-7 (lsbs): Response Verb. Identifies the type of response in each ring entry. For a frame dequeue response, this field is set to 60h.</p>
1 STAT	<p>Status. This field carries a number of status bits relating to the dequeue response.</p> <p>Bit 0 (msb): FQ empty. 1 = The FQ from which this frame was dequeued was empty after this dequeue.</p> <p>Bit 1: FQ held active. 1 = The FQ from which this frame was dequeued was held active after this dequeue. This bit is always 0 in a null dequeue response due to an error (see the FQID field description), or for an unscheduled dequeue (bit 6 = 1).</p> <p>Bit 2: FQ Force Eligible. 1 = The FQ from which this frame was dequeued was scheduled as a result of a Force Eligible command. This bit is always 0 in a null dequeue response due to an error (see the FQID field description), or for an unscheduled dequeue (bit 6 = 1).</p> <p>Bit 3: Number of frames delivered in this response. 0 = No frame delivered, a null response, the FD field in this response is invalid. 1 = One frame delivered, the FD field contains a valid frame descriptor.</p> <p>Bit 4-5: Reserved</p> <p>Bit 6: Scheduled/Unscheduled Command Type. If the DQRR is programmed in Push mode, this bit indicates which of the two dequeue command registers provided the command which resulted in the dequeue of this frame. 0 = This is a response to a dequeue command dispatched from SDQCR 1 = This is a response to a dequeue command dispatched from VDQCR. If the DQRR is programmed in Pull mode, this bit indicates whether this frame dequeue response resulted from a scheduled or an unscheduled dequeue command, it indicates the value of the SU bit in the PDQCR command.</p> <p>Bit 7 (lsbit): Dequeue command expired. If this is a response to a channel based dequeue command dispatched from SDQCR (DQRR in Push mode, STAT[6] = 0, and SS=0 in the dequeue command), then this bit is always 0.  If this is a response to a specific WQ dequeue command dispatched from SDQCR (DQRR in Push mode, STAT[6] = 0, and SS=1 in the dequeue command), then this bit indicates whether or not the specific WQ dequeue command in SDQCR is expired. 0 = specific WQ dequeue command in SDQCR is not expired, each command may deliver from 0 to 3 frames, so more responses for this command are coming. 1 = specific WQ dequeue command in SDQCR is expired, and this is the last response for that command.  If this is a response to a dequeue command dispatched from VDQCR (DQRR in Push mode and STAT[6] = 1), this bit indicates whether or not the command in VDQCR is expired. The current VDQCR command expires when the specified FQ becomes empty or when the requested NUM_FRAMES in the command is met. 0 = command in VDQCR is not expired. 1 = command in VDQCR is expired, and this is the last response for that command.  If the DQRR is programmed in Pull mode, this bit indicates whether or not the command in PDQCR is expired. 0 = command in PDQCR is not expired, each PDQCR command may deliver from 0 to 3 frames, so more responses for this command are coming. 1 = command in PDQCR is expired, and this is the last response for that command.</p>

**Table 3-104. Frame Dequeue Response Format (continued)**

Byte Field	Description
2-3 SEQNUM	<p>bits 0-1 (msbs): Reserved bits 2-15: Order Restoration Sequence Number.</p> <p>This sequence number is assigned by the order definition point (ODP) within QMan, and is intended for use with a future enqueue of the frame contained in this dequeue response, when order restoration is being used.</p> <p>Every frame in a dequeue response carries a sequence number, which may be used or ignored by the receiver of the response. This field may be ignored if the receiver of the dequeue response is not using order restoration, either globally (all frames are processed without order restoration), or selectively (frames from certain flows require order restoration, others do not).</p> <p>This field is invalid if no frame was dequeued, as indicated by bit 3 of the STAT field.</p>
4 TOK	<p>Dequeue Command Token</p> <p>Opaque token returned in the DQRR entry of all dequeued frames that resulted from the execution of an SDQCR command, this token is copied from the TOKEN field in the SDQCR register; see <a href="#">Section 3.2.3.11, “QCSP DQRR Static Dequeue Command Register (SDQCR).”</a></p> <p>For frame dequeue responses resulting from the execution of dequeue commands from the VDQCR or PDQCR registers, this field is not valid.</p>
5-7	Reserved
8	Reserved (for possible future FQID expansion)
9-11 FQID	<p>FQID identifies the FQ from which the frame in this response was dequeued.</p> <p>If STAT bit 0 (FQ Empty) is 0 and STAT bit 3 (number of frames) is also 0, this indicates a null dequeue response. Such a response can be generated as the result of an error, or because a FQ which reached the head of a WQ was not eligible for dequeue (for example, Retirement Pending or FQ XOFF was asserted on this FQ). In the case of a null dequeue response due to an error, the FQID field is not valid, and the specific error that occurred will be indicated in QMAN_ERR_ISR.</p> <p>If order restoration is to be used with this frame, then this field also identifies the ODP from which the sequence number was assigned, and also (usually, but not necessarily) the ORP that should be used when this frame is later enqueued. See also <a href="#">Section 3.3.13, “Order Restoration.”</a></p>
12-15 CNTXTB	<p>Frame Queue Context B.</p> <p>Context B is a Frame Queue attribute, it is a 32 bit value initialized by software and is stored in the FQ Descriptor. When QMan dequeues a frame from a FQ it has access to this context field, and passes it in the dequeue response.</p>
16-31 FD	<p>Frame descriptor</p> <p>Frame descriptor (FD) for the frame that has been dequeued.</p> <p>This field is invalid if no frame was dequeued, as indicated by bit 3 of the STAT field.</p> <p>See <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs)”</a> for a description of the contents of a FD.</p>
32-63	Reserved

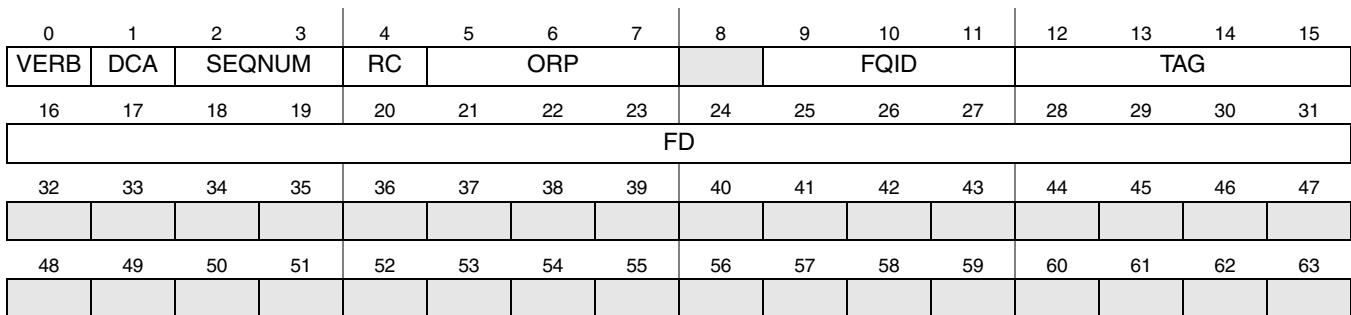
### 3.3.9.3 ERN Message Response

This section describes the format of an Enqueue Rejection Notification (ERN) message response. The ERN message is used to notify software that the enqueue of a frame from an enqueue command previously issued to and consumed from the EQCR was rejected due to an error or due to congestion management and avoidance. It may also be used to convey an ERN message to software on behalf of a hardware block whose direct connect portal (DCP) is configured to do so; see [Section 3.2.4.11, “DCP Configuration Registers \(DCPi\\_CFG\).”](#)

ERN messages are carried from QMan to software in the MR; see [Section 3.2.3.2, “QCSP Dequeue Response Ring Registers \(QCSPi\\_DQRRO-15\).”](#)

When an enqueue rejection occurs, all fields from the enqueue command that was attempted, including the FD, are copied and made available in the ERN message response.

See also [Section 3.3.14.5, “Enqueue Rejections.”](#)



**Figure 3-111. ERN Message Response Format**

**Table 3-105. ERN Message Response Format**

Byte Field	Description
0 VERB	<p>Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.2, “QCSP Dequeue Response Ring Registers (QCSPi_DQRR0-15).”</a></p> <p>Bit 1-7 (lsbs): Response Verb. Identifies the type of response in each ring entry. For an ERN message response that results from a rejected enqueue command received from a software portal, the command verb from the enqueue command whose frame enqueue has been rejected is copied into this field; see <a href="#">Section 3.3.9.1, “Enqueue Command”</a> for a description of the valid command verbs for an enqueue command in a software portal (issued via the EQCR). Note that bits 6 and 7 (the two lsbs) of this field will indicate which of the other fields in the ERN message are valid. For an ERN message response that results from a rejected enqueue command received from a direct connect portal (DCP), this response verb is set to 20h. This value uniquely identifies an ERN that originated from a DCP, because it is not a valid enqueue command verb for an enqueue command issued via the EQCR.</p>
1 DCA	<p>If this ERN resulted from a software portal enqueue (Response Verb not equal to 20h): The non-reserved bits of this field are copied directly from the enqueue command; see <a href="#">Section 3.3.9.1, “Enqueue Command.”</a></p> <p>If this ERN resulted from a DCP enqueue (Response Verb = 20h): Bit 0-1 (msbs) of this field carries the frame color from the enqueue command that was rejected. Bit 2-4 of this field is reserved. Bit 5-7 (lsbs) of this field identifies the direct connect portal (DCP) which originated the enqueue of the frame that has been rejected.</p>
2-3 SEQNUM	<p>If this ERN resulted from a software portal enqueue (Response Verb not equal to 20h): Bit 0 (msb): This bit is valid only if VERB bit 6-7 = 2. Bits 1-15: These bits are copied directly from the enqueue command; see <a href="#">Section 3.3.9.1, “Enqueue Command.”</a></p> <p>If this ERN resulted from a DCP enqueue (Response Verb = 20h): This field is not used.</p>

**Table 3-105. ERN Message Response Format**

Byte Field	Description
4 RC	<p>Rejection Code Bit 0-3 (msbs): RC: Rejection Code.</p> <p>This field is used to identify the reason for the rejection of the frame in this ERN response.</p> <ul style="list-style-type: none"> <li>0 = Frame enqueue rejected due to Congestion Group tail drop threshold exceeded.</li> <li>1 = Frame enqueue rejected due to WRED congestion avoidance.</li> <li>2 = Frame enqueue rejected due to an error condition, the specific error is indicated in the QMAN_ERR_ISR.</li> <li>3 = Frame enqueue rejected, attempted enqueue with order restoration (or attempted Remove Sequence Number ORP operation) with a sequence number within the Early Arrival Rejection Window of the ORP; see <a href="#">Section 3.3.13.5.3, “ORP Behavior in Extreme Conditions.”</a></li> <li>4 = Frame enqueue rejected, attempted enqueue with order restoration (or attempted Remove Sequence Number ORP operation) with a sequence number within the Late Arrival Rejection Window of the ORP; see <a href="#">Section 3.3.13.5.3, “ORP Behavior in Extreme Conditions.”</a></li> <li>5 = Frame enqueue rejected due to FQ tail drop threshold exceeded.</li> <li>6 = Frame enqueue rejected, this enqueue was pending in the order restoration list (ORL) of a FQ that has been retired.</li> <li>7 = Frame enqueue (or ORP operation without enqueue) rejected, because the specified ORP was disabled, Order Restoration in the command was enabled, but ORP Enable bit in the FQD indexed by ORP was 0.</li> <li>8-15 = Reserved</li> </ul> <p>Bit 4-7 (lsbs): Reserved (for possible future FQID expansion).</p>
5-7 ORP	<p>If this ERN resulted from a software portal enqueue (Response Verb not equal to 20h):</p> <p>This field is copied directly from the enqueue command; see <a href="#">Section 3.3.9.1, “Enqueue Command,”</a> unless Order Restoration is in use, and software has used one or more Advance NESN commands (see bit 0 of SEQNUM in <a href="#">Section 3.3.9.1, “Enqueue Command”</a>), in which case this field is not used.</p> <p>If this ERN resulted from a DCP enqueue (Response Verb = 20h):</p> <p>This field is not used.</p>
8	Reserved (for possible future FQID expansion)
9-11 FQID	This field is copied directly from the enqueue command; see <a href="#">Section 3.3.9.1, “Enqueue Command.”</a>
12-15 TAG	This field is copied directly from the enqueue command; see <a href="#">Section 3.3.9.1, “Enqueue Command.”</a>
16-31 FD	<p>Frame descriptor Frame descriptor (FD) for the frame whose enqueue was rejected.</p> <p>If this ERN resulted from a software portal enqueue (Response Verb not equal to 20h):</p> <p>This field is copied directly from the enqueue command; see <a href="#">Section 3.3.9.1, “Enqueue Command,”</a> with the exception of the ICID and EICID fields, which are overwritten by QMan as described in <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs).”</a></p> <p>If this ERN resulted from a DCP enqueue (Response Verb = 20h):</p> <p>This field is copied directly from the enqueue command received in the direct connect portal.</p>
32-63	Reserved

### 3.3.9.4 FQ State Change Notification Message Response

This section describes the format of a Frame Queue State Change Notification message response. These messages are used to notify software that a FQ on which a Retire FQ command was executed has now reached the Retired state (See [Section 3.3.9.5.4, “Alter FQ State Commands”](#)), or that a FQ on which a Park was requested has now reached the Parked state (See [Section 3.2.3.10, “QCSP DQRR Discrete Consumption Acknowledgment and Park \(QCSPi\\_DQRR\\_DCAP\)”](#)).

## Queue Manager (QMan)

FQ State Change Notification messages are carried from QMan to software in the MR; see [Section 3.2.3.2, “QCSP Dequeue Response Ring Registers \(QCSPi\\_DQRRO-15\).”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	FQS								FQID		CNTXTB				
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-112. FQ State Change Notification Message Response Format**

**Table 3-106. FQ State Change Notification Message Response Format**

Byte Field	Description
0 VERB	<p>Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.2, “QCSP Dequeue Response Ring Registers (QCSPi_DQRR0-15).”</a></p> <p>Bit 1-7 (lsbs): Response Verb.</p> <p>Identifies the type of response in each ring entry.</p> <p>21h = FQRN (Frame Queue Retirement Notification).</p> <p>This message is generated when a FQ on which a Retire FQ command was previously executed while the FQ was in the Truly Scheduled or Active states reaches the Retired state.</p> <p>If FQS[Active ORL Present flag] is asserted, this message will be followed (in the MR) by one or more ERN messages containing frames whose enqueue is rejected due to the retirement of this FQ. The last ERN message caused by the retirement of this FQ will be followed by a FQRL message.</p> <p>22h = FQRNI (Frame Queue Retirement Notification Immediate).</p> <p>This message is generated when a Retire FQ command is executed on a FQ in the Tentatively Scheduled or Parked states. In such a case the FQ is retired immediately when the command is executed, and notification that the FQ has reached the retired state is provided in the response to the command itself (see <a href="#">Section 3.3.9.5.4, “Alter FQ State Commands”</a>), as well as in this message.</p> <p>If FQS[Active ORL Present flag] is asserted, this message will be followed (in the MR) by one or more ERN messages containing frames whose enqueue is rejected due to the retirement of this FQ. The last ERN message caused by the retirement of this FQ will be followed by a FQRL message.</p> <p>23h = FQRL (Frame Queue Retirement Last).</p> <p>This message is generated when a FQ has reached the Retired state (either immediately after execution of a Retire FQ command, or at a later time when a FQRN message is produced), and the ORP contained in that FQ's FQD carries one or more frames awaiting enqueue on its ORL list. The FQRL message follows the last ERN message containing a frame from the retired FQ's ORL list, and serves as an indication to software that all frames that were present on the retired FQ's ORL list have now been returned to software via ERN messages.</p> <p>24h= FQPN (Frame Queue Parked Notification).</p> <p>This message is generated when a FQ on which a Park request was issued using DQRR_DCAP (see <a href="#">Section 3.2.3.10, “QCSP DQRR Discrete Consumption Acknowledgment and Park (QCSPi_DQRR_DCAP)”</a>) has reached the Parked state.</p>
1 FQS	<p>FQ Status. This field is valid only for Response Verb = FQRN, FQRNI, or FQPN, it will be set to all 0 for a FQRL message.</p> <p>Bit 0-5 (msbs): Reserved</p> <p>Bit 6: Active ORL Present flag:</p> <ul style="list-style-type: none"> <li>0 = The Order Restoration Point (ORP) contained in the FQD of the FQ that has reached the Retired or Parked state does not carry any frames awaiting enqueue.</li> <li>1 = The Order Restoration Point (ORP) contained in the FQD of the FQ that has reached the Retired or Parked state carries one or more frames awaiting enqueue on its Order Restoration List (ORL).</li> </ul> <p>If the FQ is now Retired (if this is a FQRN or FQRNI message), the frames that were awaiting enqueue on this ORL will be rejected and returned to software via ERN messages; see <a href="#">Section 3.3.9.3, “ERN Message Response.”</a> These ERN messages will be placed in the MR immediately after this FQRN/FQRNI message.</p> <p>If the FQ is now Parked (if this is a FQPN message), no frames are rejected, the ORP is not changed in any way, and all frames remain on the ORL awaiting enqueue.</p> <p>Bit 7 (lsbit): FQ Not Empty flag:</p> <ul style="list-style-type: none"> <li>0 = The FQ that has reached the Retired or Parked state is empty.</li> <li>1 = The FQ that has reached the Retired or Parked state is not empty. If the FQ is now Retired (if this is a FQRN or FQRNI message), it must be emptied, by dequeuing the frames that it carries, before being put out of service.</li> </ul>

**Table 3-106. FQ State Change Notification Message Response Format (continued)**

Byte Field	Description
2-7	Reserved
8	Reserved (for possible future FQID expansion)
9-11 FQID	FQ ID. ID of the FQ that has reached the retired or parked state.
12-15 CNTXTB	Frame Queue Context B. Context B field from the FQD that has reached the retired or parked state. Context B is a Frame Queue attribute, it is a 32 bit value initialized by software and is stored in the FQ Descriptor.
16-63	Reserved

### 3.3.9.5 Frame Queue Management Commands

This section describes the Frame Queue (FQ) management commands that are available in a software portal, as well as the responses to those commands. These commands are issued via the CR, and responses to the commands are returned in RR0 / RR1. All commands issued via the CR result in a response returned in RR0 / RR1. See also [Section 3.3.8.4, “Management Command Register \(CR\)”](#) and [Section 3.3.8.5, “Management Response Registers \(RR0/RR1\)”](#).

#### 3.3.9.5.1 Initialize Frame Queues (FQ)

This command is used to initialize or modify one or more frame queues (FQ).

Every software configurable field in the FQD (see [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\)”](#)) can be initialized with a value provided in the command. A write enable mask is provided in the command to allow software to selectively initialize only certain fields.

If the FQ being initialized is in the Out Of Service state, all software configurable fields whose write enable bit is asserted are initialized with the value provided in the command, any software configurable fields whose write enable bit is negated are reset to 0, and all hardware managed fields in the FQD are reset to 0. Then the newly initialized FQ is moved to the Parked state. If the command also specifies that the FQ should be scheduled, it will be immediately moved from the Parked to the Tentatively Scheduled state.

If the FQ being initialized is in the Parked state, all software configurable fields whose write enable bit is asserted are initialized with the value provided in the command, any software configurable fields whose write enable bit is negated remain unchanged, and all hardware managed fields in the FQD also remain unchanged. The newly modified FQ remains in the Parked state, unless the command specifies that the FQ should be scheduled, in which case it will be immediately moved from the Parked to the Tentatively Scheduled (if the FQ is empty) or Truly Scheduled (if the FQ is not empty) state.

To support guaranteed locking of a FQ into FQD cache, the LIC\_Immediate (see the FQ\_CTRL field in the command format below) feature of this command can be used, in conjunction with setting of the FQ\_CTRL[Lock\_in\_Cache] bit in the FQD (see [Section 3.3.1.4.3, “Structure of a Frame Queue Descriptor \(FQD\)”](#)).

Note that some fields in the Initialize FQ command are overloaded, that is, used to initialize two different FQD fields. In an overloaded command field, only one of the two FQD fields can be initialized at a time, and if both need to be initialized then two Initialize FQ commands are needed. This is best illustrated with an example, using the TD\_THRESH/OAC field in the command:

- If the FQ is Out of Service, and WE\_MASK bit 13 = 1, TD\_THRESH is initialized and OAC is reset to 0.
- If the FQ is Out of Service, and WE\_MASK bit 13 = 0, and WE\_MASK bit 7 = 1, OAC is initialized and TD\_THRESH is reset to 0.
- If the FQ is Out of Service, and WE\_MASK bit 13 = 0, and WE\_MASK bit 7 = 0, TD\_THRESH and OAC are both reset to 0.
- If the FQ is Parked, and WE\_MASK bit 13 = 1, TD\_THRESH is initialized and OAC is left unchanged.
- If the FQ is Parked, and WE\_MASK bit 13 = 0, and WE\_MASK bit 7 = 1, OAC is initialized and TD\_THRESH is left unchanged.
- If the FQ is Parked, and WE\_MASK bit 13 = 0, and WE\_MASK bit 7 = 0, TD\_THRESH and OAC are both left unchanged.

Therefore if both TD\_THRESH and OAC must be initialized, a first Initialize FQ command must be issued which initializes one of the two fields and leaves the FQ in the Parked state. Then a second Initialize FQ command must be issued which initializes the other FQD field.

An initialize FQ command is only valid on a FQ in the Out Of Service or Parked states. If a FQ specified in the command is in any other state, the initialize command will fail. The command will also fail if LIC\_Immediate is requested in the command but the FQ was unable to be placed into the FQD cache. When an initialize FQ command fails for any reason, it is immediately terminated without modifying the FQ, and the response to the command will indicate an error. If multiple FQ are being initialized with a single command, and an error is encountered on one of the FQ being initialized, then FQs up to the one that encountered the error will be initialized. The FQ on which the error occurred, and all subsequent FQs, will not be initialized. Initialization of multiple FQ with a single command should only be used when all FQ to be initialized are in a well known state, such as after the memory area occupied by these FQ has been cleared to 0 as described below.

Before this command can be used, the memory area to be used by QMan for FQD storage must be allocated by software, cleared to 0 by software, and the base address and size of this area must be programmed into QMan's FQD\_BAR/BARE/AR configuration registers; see [Section 3.2.4.43, “Data Structure Base Address Registers.”](#) The correct operation of this command relies on software clearing the FQD memory area before first use, such that QMan will find all FQD to be in the Out Of Service state (State field = 0 in the FQD) the first time an Initialize FQ command is executed on each FQ.

The format of both the command (written to the CR) and the response (read from RR0 / RR1) are shown below with byte granularity.

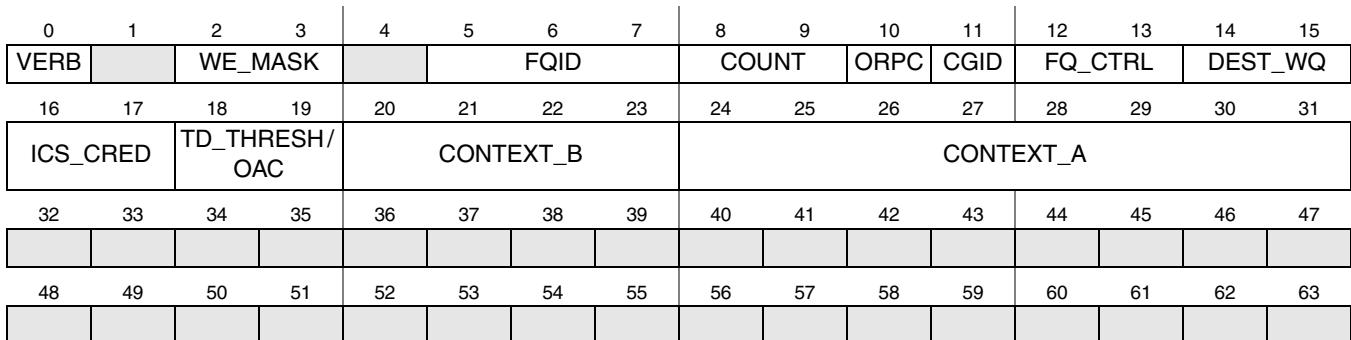


Figure 3-113. Initialize FQ Command Format

Table 3-107. Initialize FQ Command Format

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. Initialize FQ and leave Parked: Verb = 40h. Initialize FQ and Schedule: Verb = 41h.
1	Reserved
2-3 WE_MASK	Write Enable Mask. This is a 16 bit field containing individual bits which enable or disable the update to a specific FQD field. For each bit, 0 = write disabled, 1 = write enabled. Bit 0-6 (msbs): Reserved Bit 7: Write enable for OAC, valid only if bit 13 = 0. Bit 8: Write enable for ORPC Bit 9: Write enable for CGID Bit 10: Write enable for FQ_CTRL Bit 11: Write enable for DEST_WQ Bit 12: Write enable for ICS_CRED Bit 13: Write enable for TD_THRESH Bit 14: Write enable for CONTEXT_B Bit 15: Write enable for CONTEXT_A
4	Reserved (for possible future FQID expansion)
5-7 FQID	Frame Queue ID. ID of the first FQ to be initialized.
8-9 COUNT	Number of consecutive FQID to be initialized. This field enables a number of consecutive FQ to be initialized with the same values. The number of FQ initialized will be COUNT + 1, allowing from 1 to 64K FQ to be initialized with a single command. The FQs initialized will range from FQID to FQID + COUNT.
10 ORPC	ORP Configuration. Bit 0-1 (msbs): Reserved Bit 2-4: Value to be written to ORPRWS field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a> Bit 5: Value to be written to OA field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a> Bit 6-7: Value to be written to OLWS field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>

**Table 3-107. Initialize FQ Command Format (continued)**

Byte Field	Description
11 CGID	Congestion Group ID Value to be written to the CONG_ID field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a> Note that when dynamically changing the CGID on FQ that have been in service, both the CGR byte counts in both the old and new CGRs will be updated when this command is executed.
12-13 FQ_CTRL	Frame Queue Control. Bit 0 (msbit): LIC_Immediate 0: The initialize FQ command proceeds as usual, therefore this FQ may be placed in FQD cache immediately or it may not. If FQ_CTRL[Lock_in_Cache] (i.e. bit 15 of this field) is 1, then whenever this FQ does make it into cache it will remain locked there until retirement. 1: If FQ_CTRL[Lock_in_Cache] (i.e. bit 15 of this field) is also 1, then this initialize FQ command will fail, with a specific result code and no effect on the FQ, if the FQ does not successfully allocate into FQD cache immediately. This is intended for use with a FQ that must be guaranteed to be in cache from initialization until retirement. Bit 1-4: Reserved Bit 5-15: Value to be written to the FQ_CTRL field of the FQD.
14-15 DEST_WQ	Destination WQ Value to be written to the DEST_WQ field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
16-17 ICS_CRED	Intra-WQ Class Scheduling Credit Bit 0 (msb): Reserved Bit 1-15 (lsbs): Value to be written to the ICS_CRED field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
18-19 TD_THRESH / OAC	Tail Drop Threshold / Overhead Accounting Control If WE_MASK bit 13 = 1, then this field carries the Tail Drop Threshold to be written to the FQD. Bit 0-2 (msbs): Reserved Bit 3-10: Value to be written to the TD_MANT field of the FQD. Bit 11-15: Value to be written to the TD_EXP field of the FQD. If WE_MASK bit 13 = 0 and WE_MASK bit 7 = 1, then this field carries the Overhead Accounting Control and Length fields to be written to the FQD. Bit 0-1 (msbs): Value to be written to the OAC field of the FQD. Bit 2-7: Reserved Bit 8-15: Value to be written to the OAL field of the FQD.
20-23 CONTEXT_B	Frame Queue Context B. Value to be written to the CONTEXT_B field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
24-31 CONTEXT_A	Frame Queue Context A. Value to be written to the CONTEXT_A field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
32-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT														
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-114. Initialize FQ Response Format****Table 3-108. Initialize FQ Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. F1h = Error, command failed due to invalid FQID. This occurs if access to FQD memory is disabled (FQD_AR[EN]==0), or if one or more FQID specified in the command is outside of the valid range programmed in FQD_AR[SIZE]. See <a href="#">Section 3.2.4.44, “Data Structure Attributes Registers.”</a> This also occurs if the specified FQID falls into the range of LfqIDs used by CEETM (F0_0000 to FF_FFFF), see <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue.”</a> F2h = Error, command failed due to invalid FQ state. This occurs if one or more of the FQ specified in the command are in a state other than Out Of Service or Parked. F5h = Error, command failed due to LIC_Immediate requested but the FQ could not be allocated immediately into FQD cache. See the FQ_CTRL field in the command. FFh = Error, unrecognized or invalid command was received.
2-63	Reserved

### 3.3.9.5.2 Query FQ Programmable Fields

This command is used to query the software programmable fields of a Frame Queue (FQ), i.e those fields that were written using an Initialize FQ command.

This command returns a non-atomic snapshot of the FQ at the time that the command executes, the snapshot is not synchronized to any particular enqueues or dequeues that may be underway.

The format of both the command (written to the CR) and the response (read from RR0 / RR1) are shown below with byte granularity.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB				FQID											
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-115. Query FQ Programmable Fields Command Format

Table 3-109. Query FQ Programmable Fields Command Format

Byte Field	Description														
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. Query FQ Programmable Fields: Verb = 44h.														
1-3	Reserved														
4	Reserved (for possible future FQID expansion)														
5-7 FQID	Frame Queue ID. ID of the FQ to be queried.														
8-63	Reserved														

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT									ORPC	CGID	FQ_CTRL	DEST_WQ		
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ICS_CRED	TD_THRESH	CONTEXT_B				CONTEXT_A									
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
OAC															
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-116. Query FQ Programmable Fields Response Format

**Table 3-110. Query FQ Programmable Fields Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPI_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. Query FQ Programmable Fields: Verb = 44h.
1 RSLT	Result. F0h = OK, the command completed successfully. F1h = Error, command failed due to invalid FQID. This occurs if access to FQD memory is disabled (FQD_AR[EN]==0), or if the FQID specified is outside of the valid range programmed in FQD_AR[SIZE]. See <a href="#">Section 3.2.4.44, “Data Structure Attributes Registers.”</a> This also occurs if the specified FQID falls into the range of LFQIDs used by CEETM (F0_0000 to FF_FFFF), see <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue.”</a> FFh = Error, unrecognized or invalid command was received.
2-9	Reserved
10 ORPC	ORP Configuration. Bit 0-1 (msbs): Reserved Bit 2-4: Value read from the ORPRWS field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a> Bit 5: Value read from the OA field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a> Bit 6-7: Value read from the OLWS field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
11 CGID	Congestion Group ID Value read from the CONG_ID field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
12-13 FQ_CTRL	Frame Queue Control. Bit 0-4 (msbs): Reserved Bit 5-15: Value read from the FQ_CTRL field of the FQD.
14-15 DEST_WQ	Destination WQ Value read from the DEST_WQ field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
16-17 ICS_CRED	Intra-WQ Class Scheduling Credit Bit 0 (msb): Reserved Bit 1-15 (lsbs): Value read from the ICS_CRED field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
18-19 TD_THRESH	Tail Drop Threshold Bit 0-2 (msbs): Reserved Bit 3-10: Value read from the TD_MANT field of the FQD. Bit 11-15: Value read from the TD_EXP field of the FQD.
20-23 CONTEXT_B	Frame Queue Context B. Value read from the CONTEXT_B field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
24-31 CONTEXT_A	Frame Queue Context A. Value read from the CONTEXT_A field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
32-33 OAC	Overhead Accounting Control Bit 0-1 (msbs): Value read from the OAC field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a> Bit 2-7: Reserved Bit 8-15: Value read from the OAL field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
34-63	Reserved

### 3.3.9.5.3 Query FQ Non-Programmable Fields

This command is used to query the hardware managed fields of a Frame Queue (FQ). It returns a non-atomic snapshot of the FQ at the time that the command executes, the snapshot is not synchronized to any particular enqueues or dequeues that may be underway.

The format of both the command (written to the CR) and the response (read from RR0 / RR1) are shown below with byte granularity.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB						FQID									
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-117. Query FQ Non-Programmable Fields Command Format

Table 3-111. Query FQ Non-Programmable Fields Command Format

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. Query FQ Non-Programmable Fields: Verb = 45h.
1-3	Reserved
4	Reserved (for possible future FQID expansion)
5-7 FQID	Frame Queue ID. ID of the FQ to be queried.
8-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT		STATE			FQD_LINK		ODP_SEQ	ORP_NESN	ORP_EA_HSEQ	ORP_EA_TSEQ				
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	ORP_EA_HPTR			ORP_EA_TPTR				PFDR_HPTR				PFDR_TPTR			
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
				IS	ICS_SURP			BYTE_CNT				FRM_CNT			
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
				RA1_SFDR	RA2_SFDR			OD1_SFDR		OD2_SFDR	OD3_SFDR				

Figure 3-118. Query FQ Non-Programmable Fields Response Format

**Table 3-112. Query FQ Non-Programmable Fields Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. Query FQ Non-Programmable Fields: Verb = 45h.
1 RSLT	Result. F0h = OK, the command completed successfully. F1h = Error, command failed due to invalid FQID. This occurs if access to FQD memory is disabled (FQD_AR[EN]==0), or if the FQID specified is outside of the valid range programmed in FQD_AR[SIZE]. See <a href="#">Section 3.2.4.44, “Data Structure Attributes Registers.”</a> This also occurs if the specified FQID falls into the range of LFQIDs used by CEETM (F0_0000 to FF_FFFF), see <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue.”</a> FFh = Error, unrecognized or invalid command was received.
2	Reserved
3 STATE	Frame Queue State. Bit 0-2 (msbs): Reserved Bit 3: Value read from the FE field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a> Bit 4: Value read from the R field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a> Bit 5-7: Value read from the STATE field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
4	Reserved (for possible future FQID expansion)
5-7 FQ_LINK	Frame Queue Link. Value read from the FQD_LINK field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
8-9 ODP_SEQ	Bit 0-1 (msbs): Reserved Bit 2-15: Value read from ODP_SEQ field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
10-11 ORP_NESN	Bit 0-1 (msbs): Reserved Bit 2-15: Value read from ORP_NESN of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
12-13 ORP_EA_HSEQ	Bit 0 (msb): Reserved Bit 1-15: Value read from the ORP_EA_HSEQ field of the FQD.
14-15 ORP_EA_TSEQ	Bit 0 (msb): Reserved Bit 1-15: Value read from the ORP_EA_TSEQ field of the FQD.
16	Reserved
17-19 ORP_EA_HPTR	Value read from the ORP_EA_HPTR field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
20	Reserved
21-23 ORP_EA_TPTR	Value read from the ORP_EA_TPTR field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
24	Reserved
25-27 PFDR_HPTR	Value read from the PFDR_HPTR field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
28	Reserved

**Table 3-112. Query FQ Non-Programmable Fields Response Format (continued)**

<b>Byte Field</b>	<b>Description</b>
29-31 PFDR_TPTR	Value read from the PFDR_TPTR field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
32-36	Reserved
37 IS	Bit 0-6 (msbs): Reserved Bit 7 (lsb): Value read from the IS field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
38-39 ICS_SURP	Value read from the ICS_SURP field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
40-43 BYTE_CNT	Value read from the BYTE_CNT field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
44	Reserved
45-47 FRM_CNT	Value read from the FRM_CNT field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
48-51	Reserved
52-53 RA1_SFDR	Bit 0-1 (msbs): Value read from the NRA field of the FQD Bit 2-3: Reserved Bit 4: Value read from the C field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a> Bit 5-15: Value read from the RA1_SFDR_PTR field of the FQD.
54-55 RA2_SFDR	Bit 0 (msb): Value read from the X field of the FQD. Bit 1: Value read from the IT field of the FQD. Bit 2-4: Reserved Bit 5-15: Value read from the RA2_SFDR_PTR field of the FQD.
56-57	Reserved
58-59 OD1_SFDR	Bit 0-1 (msbs): Value read from the NOD field of the FQD. Bit 2-4: Reserved Bit 5-15: Value read from the OD1_SFDR_PTR field of the FQD.
60-61 OD2_SFDR	Bit 0-4: Reserved Bit 5-15: Value read from the OD2_SFDR_PTR field of the FQD.
62-63 OD3_SFDR	Bit 0-1 (msbs): Value read from the NPC field of the FQD. Bit 2-4: Reserved Bit 5-15: Value read from the OD3_SFDR_PTR field of the FQD.

### 3.3.9.5.4 Alter FQ State Commands

A group of commands are available to alter the state of a FQ. FQ states are described in [Section 3.3.1.5, “Frame Queue State.”](#) Each of these commands is identified by a different command verb, the command verb and the current FQ state in which it is legal to issue each command are listed in the table below. The description of each command is as follows:

- **Schedule FQ:** This command will move a FQ from the Parked state to either the Tentatively Scheduled (if the FQ is empty) or the Truly Scheduled (if the FQ is not empty) state.
- **Force Eligible FQ:** This command will assert the FE (Force Eligible Pending) bit in the FQ’s FQD. If the FQ is in the Tentatively Scheduled state, it will be placed on its destination WQ and moved to the Truly Scheduled state. If the FQ is in the Parked state and is scheduled (using a Schedule FQ

command) after the FE bit is asserted, it will be placed on its destination WQ and moved to the Truly Scheduled state. When selected for dequeue, the FQ will be moved to the Held Active state, giving software the opportunity to Park the FQ if desired. The FE bit is cleared after the FQ has been selected for dequeue and is then released from the portal (that is, moved out of the Held Suspended state). Note that for an unscheduled dequeue (from a Parked or Retired FQ) the FE bit has no effect, an unscheduled dequeue will ignore the FE bit in the FQD and will not modify it.

- **Retire FQ:** This command is used to move a FQ to the Retired state. If the FQ is in the Tentatively Scheduled or Parked states when the command is executed, the FQ is immediately moved to the Retired state and an “OK” response is returned for this command. An FQRNI message is also produced, and one or more ERN messages followed by a FQRL message may be produced on the MR of the software portal which issued the Retire FQ command, if the retired FQ contains an ORP with one or more frames awaiting enqueue on its ORL list. See [Section 3.3.9.4, “FQ State Change Notification Message Response.”](#) If the FQ is in the Truly Scheduled or Active states when the command is executed, the FQ’s state is not changed, a “Retirement Pending” response is returned for this command, and the R (Retirement Pending) bit is asserted in the FQ’s FQD. Then when the FQ is being moved between certain states the R bit is checked and if asserted the FQ is moved to the Retired state at that time. When the FQ with a Retirement Pending eventually reaches the Retired state, a FQRN message is sent to software on the MR of the software portal which issued the Retire FQ command; see [Section 3.3.9.4, “FQ State Change Notification Message Response.”](#) This deferred notification is done by overwriting the DEST\_WQ field in the FQD with the portal number, which is then used to direct the FQRN message to the right software portal when it becomes retired, this is valid since DEST\_WQ will no longer be used once the FQ becomes retired) If a Retire FQ command is received for a FQ which is in the Truly Scheduled or Active states, and the R (Retirement Pending) bit is already asserted in the FQ’s FQD (due to a previous Retire FQ command on this FQ), then this retire FQ command will fail with an “invalid state” error response.
- **Take FQ Out of Service:** This command will move a FQ from the Retired to the Out of Service state.
- **Retire FQ with Context\_B:** This command is used to move a FQ to the Retired state. This command operates in exactly the same way as the Retire FQ command described earlier, but with the addition of a Context\_B field in the command which carries a value to be written into the Context\_B field of the FQD of the FQ to be retired. This allows software to specify the Context\_B value that will be returned in the FQRN, FQRNI, and FQRL messages that may be generated when the FQ reaches the Retired state. This might be particularly useful when retiring a FQ whose consumer is a direct connected hardware module, in which case the Context\_B value required by the consumer of the FQ while it is in service might be different than the Context\_B value that the software performing FQ retirement operations may want to use. A key requirement is that the new context\_B value written to the FQD by this command applies for retirement only, and must not appear in any dequeued frames, in particular those dequeued by a direct connected hardware module. Therefore execution of a Retire FQ command (with or without Context\_B) on a FQ currently in the Active state will result in that FQ being suspended when the next dequeue operation is attempted on it and sees that retirement is pending, which prevents further dequeues from this FQ from occurring after the Retire FQ command has executed.

- **FQ XON:** This command will set the XON/XOFF state of a FQ to XON, enabled for dequeue. If the FQ is already XON, this will have no effect. See [Section 3.3.6, “Frame Queue \(FQ\) Flow Control.”](#)
- **FQ XOFF:** This command will set the XON/XOFF state of a FQ to XOFF, disabled for dequeue. If the FQ is already XOFF, this will have no effect. See [Section 3.3.6, “Frame Queue \(FQ\) Flow Control.”](#)

It is worthwhile to note that both the Force Eligible FQ and Retire FQ commands can cause a bit to be set in the FQD (the FE and the R bit respectively), which in turn can cause delayed actions to occur on the FQ well after the execution of these commands. So in a case where conflicting commands are issued on the same FQ, it is possible to have both the FE and R bits set in a FQD. In such a case, retirement of a FQ (the R bit) will always take precedence over a Force Eligible command (the FE bit), and this is illustrated in the FQ state diagram in [Section 3.3.1.5, “Frame Queue State.”](#) When the R bit is found asserted during any dequeue operation, the FE bit is ignored and not modified.

The format of both the command (written to the CR) and the response (read from RR0 / RR1) are shown below with byte granularity.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB					FQID			CNT							
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
				CONTEXT_B											
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

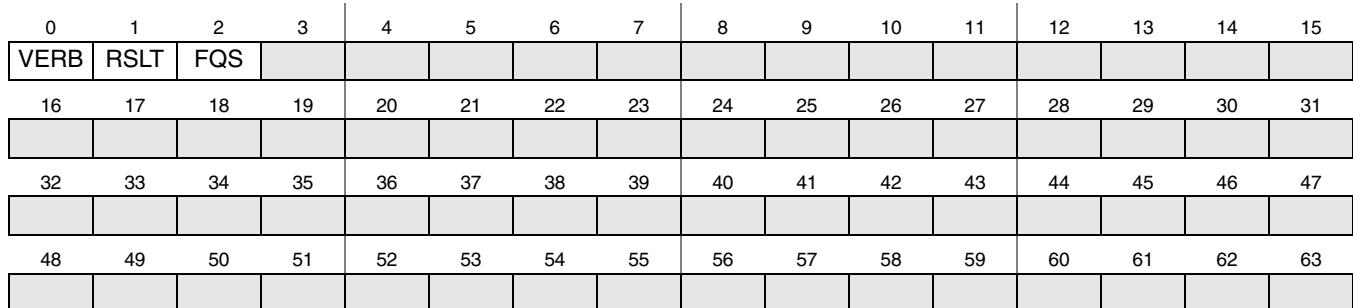
**Figure 3-119. Alter FQ State Command Format**

**Table 3-113. Alter FQ State Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. Schedule FQ: Verb = 48h. Valid only on a Parked FQ. Force Eligible FQ: Verb = 49h. Valid on a FQ in any state other than Out Of Service or Retired. Retire FQ: Verb = 4Ah. Valid on a FQ in any state other than Out Of Service or Retired. Take FQ out of service: Verb = 4Bh. Valid only on a Retired FQ. Retire FQ with Context_B:Verb = 4Ch. Valid on a FQ in any state other than Out Of Service or Retired. FQ XON:Verb = 4Dh. Valid on a FQ in any state other than Out Of Service (OOS), Retired, or Parked FQ XOFF:Verb = 4Eh. Valid on a FQ in any state other than OOS, Retired, or Parked.
1-3	Reserved
4	Reserved (for possible future FQID expansion)
5-7 FQID	Frame Queue ID. ID of the FQ on which to execute the command.
8	Reserved

**Table 3-113. Alter FQ State Command Format (continued)**

Byte Field	Description
9 CNT	Count. This field is used only with the two FQ flow control commands FQ XON and FQ XOFF. Number of consecutive FQID to which the XON or XOFF command applies. The number of FQ modified will be CNT + 1, allowing from 1 to 256 FQ to be switched to XON or XOFF with a single command. The FQs modified will range from FQID to FQID + CNT.
10-19	Reserved
20-23 CONTEXT_B	Frame Queue Context B. This field is valid only for the Retire FQ with Context_B command. Value to be written to the CONTEXT_B field of the FQD; see <a href="#">Section 3.3.1.4, “Frame Queue Descriptors (FQDs).”</a>
24-63	Reserved

**Figure 3-120. Alter FQ State Response Format**

**Table 3-114. Alter FQ State Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPI_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. F1h = Error, command failed due to invalid FQID. This occurs if access to FQD memory is disabled (FQD_AR[EN]==0), or if the FQID specified is outside of the valid range programmed in FQD_AR[SIZE]. See <a href="#">Section 3.2.4.44, “Data Structure Attributes Registers.”</a> This also occurs if the specified FQID falls into the range of LFQIDs used by CEETM (F0_0000 to FF_FFFF), see <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue.”</a> F2h = Error, command failed due to invalid FQ state. This occurs if the FQ specified in the command is in an illegal state for the command to be executed. The valid states for each command are listed in <a href="#">Table 3-113</a> . If the command was FQ XON or FQ XOFF, see also <a href="#">Section 3.3.6.1, “FQ Flow Control and FQ States.”</a> For a Retire FQ or Retire FQ with Context_B command, this also occurs if the FQ is in the Truly Scheduled or Active states and the R (Retirement Pending) bit is already asserted in the FQ's FQD. F3h = Error, Take FQ out of service command failed due to non-empty FQ. This occurs if the FQ which the command is attempting to take out of service is not empty, in which case the FQ remains in its current state and this error is returned. Software must empty a retired FQ before taking it out of service. F8h = Retirement Pending. This occurs if a Retire FQ command is executed on a FQ that is currently in the Truly Scheduled or Active state. When this response is received, software should expect to receive a FQRN message in the MR when this FQ eventually reaches the Retired state; see <a href="#">Section 3.3.9.4, “FQ State Change Notification Message Response.”</a> FFh = Error, unrecognized or invalid command was received.
2 FQS	FQ Status. This field is valid only in response to a Retire FQ command, and then only if the RSLT code is F0h, indicating that the FQ was successfully retired. Bit 0-5 (msbs): Reserved Bit 6: Active ORL Present flag: 0 = The Order Restoration Point (ORP) contained in the FQD of the FQ that has reached the Retired state does not carry any frames awaiting enqueue. 1 = The Order Restoration Point (ORP) contained in the FQD of the FQ that has reached the Retired state carries one or more frames awaiting enqueue on its Order Restoration List (ORL). The frames that were awaiting enqueue on this ORL will be rejected and returned to software via ERN messages; see <a href="#">Section 3.3.9.3, “ERN Message Response.”</a> The last ERN message produced by this retired FQ will be followed by a FQRL message; see <a href="#">Section 3.3.9.4, “FQ State Change Notification Message Response.”</a> Bit 7 (lsbit): FQ Not Empty flag: 0 = The FQ that has reached the Retired state is empty. 1 = The FQ that has reached the Retired state is not empty. It must be emptied, by dequeuing the frames that it carries, before being put out of service.
3-63	Reserved

### 3.3.9.5.5 Query WQ Length

This command is used by software to query the length of the WQs in a particular channel. In this context the length of a WQ is defined as the number of FQs contained in that WQ.

## Queue Manager (QMan)

This command returns a non-atomic snapshot of the WQ lengths at the time that the command executes, the snapshot is not synchronized to any particular enqueues or dequeues that may be underway.

The format of both the command (written to the CR) and the response (read from RR0 / RR1) are shown below with byte granularity.

**Figure 3-121. Query WQ Length Command Format**

**Table 3-115. Query WQ Length Command Format**

Byte Field	Description
0 VERB	<p>Bit 0 (msb): Valid bit.  See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a></p> <p>Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed.</p> <p>Query length of WQs in a specified channel: Verb = 46h.</p> <p>Query length of WQs in the channel dedicated to this software portal: Verb = 47h.</p>
1	Reserved
2-3 CH	<p>Channel Number.</p> <p>Used only when querying a specified channel. Not used when querying the dedicated channel of this portal.</p> <p>Specifies the channel ID whose WQ lengths should be queried.</p> <p>Bit 0-12 (msbs): Channel number. These bits correspond to the msbs of a 16 bit WQ ID value. See <a href="#">Section 3.3.2.4, “Work Queue Channel Assignments”</a> for the channel numbers and WQ IDs assigned to the various software and direct connect portals.</p> <p>Bit 13-15 (lsbs): Reserved</p>
4-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT	CH													
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
	WQ0_LEN			WQ1_LEN			WQ2_LEN			WQ3_LEN					
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	WQ4_LEN			WQ5_LEN			WQ6_LEN			WQ7_LEN					

**Figure 3-122. Query WQ Length Response Format**

**Table 3-116. Query WQ Length Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPI_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. F4h = Error, an invalid channel number was specified in the command. FFh = Error, unrecognized or invalid command was received.
2-3 CH	Channel Number. This field in the response carries the same value as in the command.
4-32 36, 40, 44, 48, 52, 56, 60	Reserved
33-35 WQ0_LEN	WQ0 Length. Returns the number of FQ currently enqueued on WQ0 of the channel that was queried.
37-39 WQ1_LEN	WQ1 Length. Returns the number of FQ currently enqueued on WQ1 of the channel that was queried.
41-43 WQ2_LEN	WQ2 Length. Returns the number of FQ currently enqueued on WQ2 of the channel that was queried.
45-47 WQ3_LEN	WQ3 Length. Returns the number of FQ currently enqueued on WQ3 of the channel that was queried.
49-51 WQ4_LEN	WQ4 Length. Returns the number of FQ currently enqueued on WQ4 of the channel that was queried.
53-55 WQ5_LEN	WQ5 Length. Returns the number of FQ currently enqueued on WQ5 of the channel that was queried.
57-59 WQ6_LEN	WQ6 Length. Returns the number of FQ currently enqueued on WQ6 of the channel that was queried.
61-63 WQ7_LEN	WQ7 Length. Returns the number of FQ currently enqueued on WQ7 of the channel that was queried.

### 3.3.9.6 Congestion Group Management Commands

This section describes the congestion group record (CGR) management commands that are available in a software portal, as well as the responses to those commands. These commands are issued via the CR, and responses to the commands are returned in RR0/RR1. All commands issued via the CR result in a response returned in RR0/RR1. See also [Section 3.3.8.4, “Management Command Register \(CR\),”](#) and [Section 3.3.8.5, “Management Response Registers \(RR0/RR1\).”](#)

#### 3.3.9.6.1 Initialize/Modify a CGR

This command is used to initialize or modify one congestion group record (CGR).

Every software configurable field in the CGR (see [Section 3.3.14.7, “Congestion Group Record \(CGR\)”](#)) can be initialized with a value provided in the command. A write enable mask is provided in the command to allow software to selectively initialize only certain fields.

If the Command Verb is Initialize, all software configurable fields whose write enable bit is asserted are initialized with the value provided in the command, any software configurable fields whose write enable bit is negated are reset to 0, and all hardware managed fields in the CGR are reset to initial values (usually 0). Then the newly initialized CGR is ready to be used. Before any CGR is used this command must be executed on that CGR.

If the Command Verb is Modify, all software configurable fields whose write enable bit is asserted are initialized with the value provided in the command, any software configurable fields whose write enable bit is negated remain unchanged, and all hardware managed fields are updated depending on the change to the software configurable fields (for example, if a threshold change causes a Congestion State change, then this is applied to the hardware fields).

The format of both the command (written to the CR) and the response (read from RR0 / RR1) are shown below with byte granularity.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		WE_MASK		WR_PARM_G				WR_PARM_Y				WR_PARM_R			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
WR_E_N_G	WR_E_N_Y	WR_E_N_R	CSCN_EN	CSCN_TARG_UPD_CTRL				CSTD_EN		CS_THRES	MODE				CGID
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-123. Initialize/Modify CGR Command Format**

**Table 3-117. Initialize/Modify CGR Command Format**

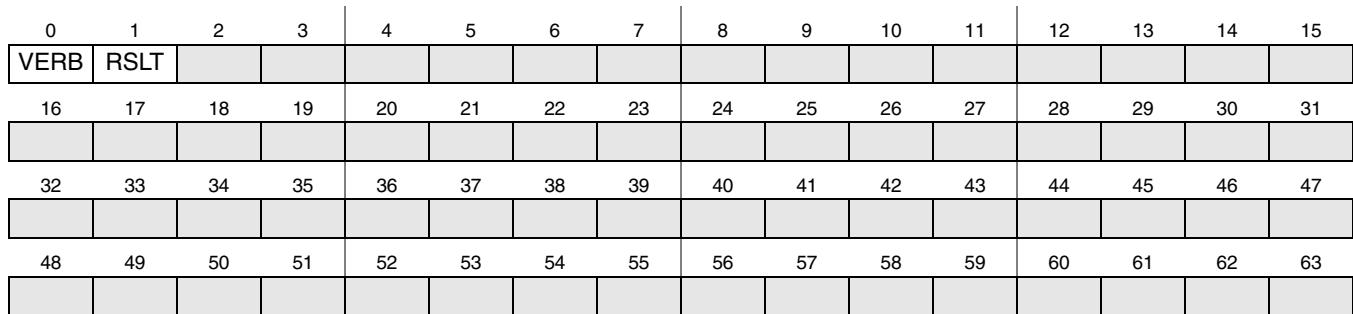
Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR)”</a> . Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. Initialize CGR: Verb = 50h. Modify CGR: Verb = 51h.
1	Reserved

**Table 3-117. Initialize/Modify CGR Command Format (continued)**

Byte Field	Description
2-3 WE_MASK	<p>Write Enable Mask.</p> <p>This is a 16 bit field containing individual bits which enable or disable the update to a specific CGR field. For each bit, 0 = write disabled, 1 = write enabled.</p> <ul style="list-style-type: none"> <li>Bit 0-4 (msbs): Reserved</li> <li>Bit 5: Write enable for WR_PARM_G</li> <li>Bit 6: Write enable for WR_PARM_Y</li> <li>Bit 7: Write enable for WR_PARM_R</li> <li>Bit 8: Write enable for WR_EN_G</li> <li>Bit 9: Write enable for WR_EN_Y</li> <li>Bit 10: Write enable for WR_EN_R</li> <li>Bit 11: Write enable for CSCN_EN</li> <li>Bit 12: Write enable for CSCN_TARG_UPD_CTRL</li> <li>Bit 13: Write enable for CSTD_EN</li> <li>Bit 14: Write enable for CS_THRES</li> <li>Bit 15: Write enable for MODE.</li> </ul>
4-7 WR_PARM_G	<p>WRED Green Parameters</p> <p>Value to be written to WR_PARM_G in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a></p>
8-11 WR_PARM_Y	<p>WRED Yellow Parameters</p> <p>Value to be written to WR_PARM_Y in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a></p>
12-15 WR_PARM_R	<p>WRED Red Parameters</p> <p>Value to be written to WR_PARM_R in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a></p>
16 WR_EN_G	<p>WREN Green Enable</p> <p>Bit 0-6: Reserved</p> <p>Bit 7: Value to be written to WR_EN_G field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a></p>
17 WR_EN_Y	<p>WREN Yellow Enable</p> <p>Bit 0-6: Reserved</p> <p>Bit 7: Value to be written to WR_EN_Y field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a></p>
18 WR_EN_R	<p>WREN Red Enable</p> <p>Bit 0-6: Reserved</p> <p>Bit 7: Value to be written to WR_EN_R field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a></p>
19 CSCN_EN	<p>Congestion State Change Notification Enable</p> <p>Bit 0-6: Reserved</p> <p>Bit 7: Value to be written to CSCN_EN field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a></p>
20-21 CSCN_TARG_UPD_CTRL	<p>Congestion State Change Notification Target update control</p> <p>The CSCN_TARG_SWP and CSCN_TARG_DCP fields, as stored internally in the CGR (see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a>”), contain one bit per portal, for both software and direct connect portals. The Initialize/Modify CGR command allows software to modify exactly one of these bits per command issued.</p> <ul style="list-style-type: none"> <li>bit 0: Value to be written to the CSCN_TARG_SWP or CSCN_TARG_DCP portal bit.</li> <li>bit 1: Portal type, 0 = software portal, 1 = DCP portal.</li> </ul> <p>bits 6-15: Portal number whose CSCN_TARG bit is to be written with the value in bit 0 of this field.</p> <p>Note that if the command verb is Initialize CGR, the portal whose bit is indexed in bits 6-15 is updated with the value from bit 0, and all other CSCN_TARG bits are cleared to 0. If the portal number is out of range (portal does not exist), all CSCN_TARG bits are cleared to 0.</p> <p>If the command verb is Modify CGR, the portal whose bit is indexed in bits 6-15 is updated with the value from bit 0, and all other CSCN_TARG bits remain unchanged. If the portal number is out of range (portal does not exist), no CSCN_TARG bits are modified.</p>
22-23	Reserved

**Table 3-117. Initialize/Modify CGR Command Format (continued)**

Byte Field	Description
24 CSTD_EN	Congestion State Tail Drop Enable Bit 0-6: Reserved Bit 7: Value to be written to CSTD_EN field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a>
25	Reserved
26-27 CS_THRES	Congestion State Threshold Bit 0-2: Reserved Bit 3-15: Value to be written to CS_THRES field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a>
28 MODE	Congestion Group Mode Bit 0-6: Reserved Bit 7: Value to be written to MODE field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)</a>
29	Reserved
30	Reserved for future CGID growth
31 CGID	Index (0 to 255) of the CGR to be initialized or modified
32-63	Reserved

**Figure 3-124. Initialize/Modify CGR Response Format****Table 3-118. Initialize/Modify CGR Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-63	Reserved

### 3.3.9.6.2 Query CGR

This command is used to query the entire contents of a Congestion Group Record (CGR).

This command returns a non-atomic snapshot of the CGR at the time that the command executes, the snapshot is not synchronized to any particular enqueues or dequeues that may be underway.

The format of both the command (written to the CR) and the response (read from RR0 / RR1) are shown below with byte granularity.

**Figure 3-125. Query CGR Command Format**

**Table 3-119. Query CGR Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a>
	Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed.
	Query CGR: Verb = 58h.
1-29	Reserved
30	Reserved (for possible future CGID expansion)
31 CGID	Index (0 to 255) of the CGR to be queried
32-63	Reserved

**Figure 3-126. Query CGR Response Format**

**Table 3-120. Query CGR Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-3	Reserved
4-7 WR_PARM_G	WRED Green Parameters Value read from WR_PARM_G in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
8-11 WR_PARM_Y	WRED Yellow Parameters Value read from WR_PARM_Y in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
12-15 WR_PARM_R	WRED Red Parameters Value read from WR_PARM_R in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
16 WR_EN_G	WREN Green Enable Bit 0-6: Reserved Bit 7: Value read from WR_EN_G field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
17 WR_EN_Y	WREN Yellow Enable Bit 0-6: Reserved Bit 7: Value read from WR_EN_Y field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
18 WR_EN_R	WREN Red Enable Bit 0-6: Reserved Bit 7: Value read from WR_EN_R field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
19 CSCN_EN	Congestion State Change Notification Enable Bit 0-6: Reserved Bit 7: Value read from CSCN_EN field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
20-23 CSCN_TARG_DCP	Congestion State Change Notification Target bit mask for DCP portals. Value read from CSCN_TARG_DCP field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
24 CSTD_EN	Congestion State Tail Drop Enable Bit 0-6: Reserved Bit 7: Value read from CSTD_EN field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
25 CS	Congestion State Bit 0-6: Reserved Bit 7: Value read from CS field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
26-27 CS_THRES	Congestion State Threshold Bit 0-2: Reserved Bit 3-15: Value read from CS_THRES field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
28 MODE	Congestion Group Mode Bit 0-6: Reserved Bit 7: Value to be written to MODE field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
29-34	Reserved

**Table 3-120. Query CGR Response Format (continued)**

Byte Field	Description
35-39 I_BCNT	Instantaneous Group Byte Count Value read from I_BCNT field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
40-42	Reserved
43-47 A_BCNT	Average Group Byte Count Value read from A_BCNT field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>
48-63 CSCN_TARG _SWP	Congestion State Change Notification Target bit mask for software portals. Value read from CSCN_TARG_SWP field in CGR; see <a href="#">Section 3.3.14.7, “Congestion Group Record (CGR)”</a>

### 3.3.9.6.3 Query Congestion State

This command is used to query the state of all Congestion Groups in QMan at once.

This command returns a snapshot of the state of each of QMan’s congestion groups at the time that the command executes. It is used by software in response to receiving a CSCN interrupt. See [Section 3.3.14.6, “Congestion State Change Notifications \(CSCN\).”](#)

The format of both the command (written to the CR) and the response (read from RR0 / RR1) are shown below with byte granularity.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-127. Query Congestion State Command Format****Table 3-121. Query Congestion State Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. Query Congestion State: Verb = 59h.
1-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT														
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
CGS[0:127]															
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CGS[128:255]															

**Figure 3-128. Query Congestion State Response Format****Table 3-122. Query Congestion State Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0-1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-31	Reserved
32-63 CGS[0:255]	Congestion State. This is a 256 bit field containing 1 bit indicating the state of each of the congestion groups in QMan. Bit 0 (msb): State of Congestion Group 0: 0 = not in congestion, 1 = in congestion. .... Bit 255(lsb): State of Congestion Group 255: 0 = not in congestion, 1 = in congestion

### 3.3.9.7 Customer Edge Egress Traffic Management (CEETM) Management Commands

This section describes the CEETM management commands that are available in a software portal, as well as the responses to those commands. These commands are issued via the CR, and responses to the commands are returned in RR0/RR1. All commands issued via the CR result in a response returned in RR0/RR1. See also [Section 3.3.8.4, “Management Command Register \(CR\),”](#) and [Section 3.3.8.5, “Management Response Registers \(RR0/RR1\).”](#)

The command verbs used for CEETM management are the following:

- Command verbs: 70h, 71h: CEETM LFQMT configure / query.
- Command verbs: 72h, 73h: CEETM CQ configure / query.
- Command verbs: 74h, 75h: CEETM dequeue context configure / query.
- Command verbs: 76h, 77h: CEETM class scheduler configure / query.
- Command verbs: 78h, 79h: CEETM mapping/shaper/flow control configure / query.
- Command verbs: 7Ah, 7Bh: CEETM class congestion group record (CCGR) configure / query.
- Command verb 7Ch: CEETM CQ Peek / Pop / XSFDR read.

- Command verb 7Dh: CEETM Statistics query / write.

### 3.3.9.7.1 CEETM Logical FQ Mapping Table (LFQMT) Configure

This command is used to configure entries in the CEETM LFQMT. After reset the LFQMT is initialized by hardware with default values as described in the table below, this command can be used to change the default configuration. See also [Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue.”](#)

Note that the LFQMT also contains a CCGID value which is not writeable by this command. When this command is executed, the CQID value from the command is used to read the CCGID value from the CQD and write it into the LFQMT along with the CQID and DCTIDX values. This ensures that the CCGID values are consistent between the LFQMT and the CQD. However this does mean that whenever a CQ configure command (see [Section 3.3.9.7.3, “CEETM Class Queue \(CQ\) Configure”](#)) is used to modify the CCGID used by a CQ, it must then be followed by a LFQMT configure command to each LFQMT entry that contains that CQID, in order to update the CCGID values in those LFQMT entries. The CEETM logic will also assert an error interrupt if an enqueue is received in which the CCGID values in the LFQMT and CQD do not match see [Section 3.3.20.11, “CEETM Class Congestion Management and Avoidance \(CCM\).”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB				LFQID				CQID			DCTIDX				
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

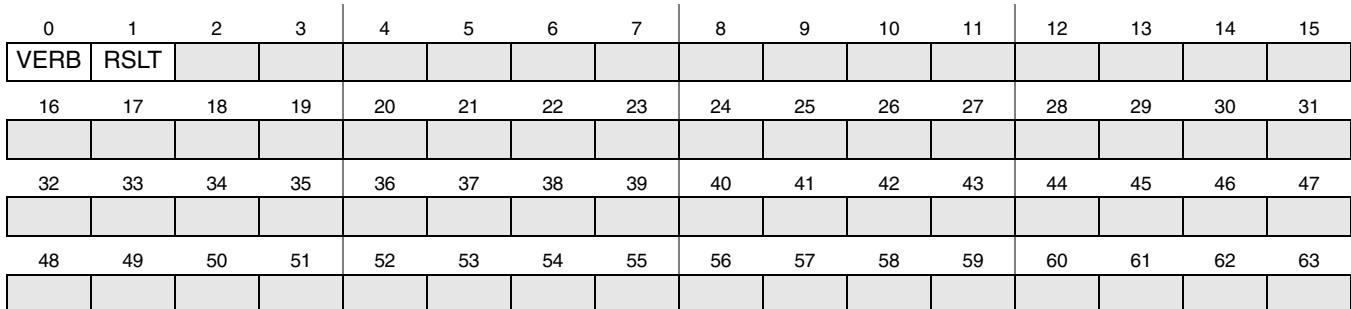
Figure 3-129. LFQMT Configure Command Format

Table 3-123. LFQMT Configure Command Format

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPI_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. LFQMT Configure: Verb = 70h.
1-4	Reserved
5-7 LFQID	Logical FQID serves as an index into the LFQMT, thus this field provides the address in the table at which the access will occur. See <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue”</a> which describes the LFQID values which are reserved for customer edge egress traffic management (CEETM). Bits 0-3: Must be 0Fh (msbits of LFQID). Bits 4-7: LFQMT table select. Note that this is equivalent to DCPIID in other commands, for example the CEETM CQ configure command. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s LFQMT table is to be addressed by this command. Bits 8-11: Not used, must be zero. Bits 12-23: LFQMT table index (lsbits of LFQID) (32 channel CEETM). Bits 14-23: LFQMT table index (lsbits of LFQID) (8 channel CEETM).

**Table 3-123. LFQMT Configure Command Format**

Byte Field	Description
8-9	Reserved
10-11 CQID	Bits 0-6: Reserved Bits 7-15: CQID value to be written to the LFQMT (32 channel CEETM). Bits 9-15: CQID value to be written to the LFQMT (8 channel CEETM). After reset, this field is set equal to the lsbits of the LFQMT table index, such that consecutive LFQMT entries contain consecutive CQID values.
12-13	Reserved
14-15 DCTIDX	Bits 0-3: Reserved Bits 4-15: Dequeue Context Table Index value to be written to the LFQMT (32 channel CEETM). Bits 6-15: Dequeue Context Table Index value to be written to the LFQMT (8 channel CEETM). After reset, this field is set equal to the lsbits of the LFQMT table index, such that consecutive LFQMT entries contain consecutive DCTIDX values
16-63	Reserved

**Figure 3-130. LFQMT Configure Response Format****Table 3-124. LFQMT Configure Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. F1h = Error, invalid LFQID. The LFQID supplied in this command must be a valid CEETM LFQID targeting a DCP portal on which CEETM is supported, see <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue.”</a> FFh = Error, unrecognized or invalid command was received.
2-63	Reserved

### 3.3.9.7.2 CEETM Logical FQ Mapping Table (LFQMT) Query

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB					LFQID										
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-131. LFQMT Query Command Format

Table 3-125. LFQMT Query Command Format

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. LFQMT Query: Verb = 71h.
1-4	Reserved
5-7 LFQID	Logical FQID serves as an index into the LFQMT, thus this field provides the address in the table at which the access will occur. See <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue”</a> which describes the LFQID values which are reserved for customer edge egress traffic management (CEETM). Bits 0-3: Must be 0Fh (msbits of LFQID). Bits 4-7: LFQMT table select. Note that this is equivalent to DCPIID in other commands, for example the CEETM CQ configure command. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s LFQMT table is to be addressed by this command. Bits 8-11: Not used, must be zero. Bits 12-23: LFQMT table index (lsbits of LFQID) (32 channel CEETM). Bits 14-23: LFQMT table index (lsbits of LFQID) (8 channel CEETM).
8-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
VERB	RSLT									CQID				DCTIDX		
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
		CCGID														
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	

Figure 3-132. LFQMT Query Response Format

**Table 3-126. LFQMT Query Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. F1h = Error, invalid LFQID. The LFQID supplied in this command must be a valid CEETM LFQID targeting a DCP portal on which CEETM is supported, see <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue.”</a> FFh = Error, unrecognized or invalid command was received.
2-9	Reserved
10-11 CQID	Bits 0-6: Reserved Bits 7-15: CQID value read from the LFQMT.
12-13	Reserved
14-15 DCTIDX	Bits 0-3: Reserved Bits 4-15: Dequeue Context Table Index value read from the LFQMT.
16-17	Reserved
18-19 CCGID	Bits 0-11: Reserved Bits 12-15: CCGID value read from the LFQMT. After reset, this field is set equal to the lsbits of the CQID value, such that consecutive CQD use consecutive CCGID values.
20-63	Reserved

### 3.3.9.7.3 CEETM Class Queue (CQ) Configure

This is used to configure the software programmable fields in the class queue descriptor (CQD) of a CQ. After reset all CQD are initialized by hardware with default values as described in the table below, this command can be used to change the default configuration. See also [Section 3.3.20.4, “CEETM Class Queue \(CQ\).”](#)

The hardware managed fields in the CQD are not modified by this command. The CQ must be empty (no frames on it) when this command is executed. Once a CQ has been used to carry traffic, it must be drained before it can be re-configured. If this command is issued on a CQ that is not empty, the command will fail and no modification will be done to the CQD.

Note that CCGID values in the LFQMT (see [Section 3.3.9.7.1, “CEETM Logical FQ Mapping Table \(LFQMT\) Configure”](#)) must be kept consistent with the CCGID values in CQDs that are configured using the CQ configure command. This means that whenever a CQ configure command is used to modify the CCGID used by a CQ, it must then be followed by a LFQMT configure command to each LFQMT entry that contains that CQID, in order to update the CCGID values in those LFQMT entries. The CEETM logic will assert an error interrupt if an enqueue is received in which the CCGID values in the LFQMT and CQD do not match see [Section 3.3.20.11, “CEETM Class Congestion Management and Avoidance \(CCM\).”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		CQID		DCPID		CCGID									
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-133. CEETM CQ Configure Command Format

Table 3-127. CEETM CQ Configure Command Format

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPI_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM CQ Configure: Verb = 72h.
1	Reserved
2-3 CQID	Bits 0-6: Reserved Bits 7-15: CQID value of the CQ to be configured (32 channel CEETM). Bits 9-15: CQID value of the CQ to be configured (8 channel CEETM).
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5	Reserved
6-7 CCGID	Bits 0-11: Reserved Bits 12-15: CCGID value to be written to the CQD. See <a href="#">Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue.”</a> After reset, this field is set equal to the lsbits of the CQID value, such that consecutive CQD contain consecutive CCGID values.
6-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT														
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

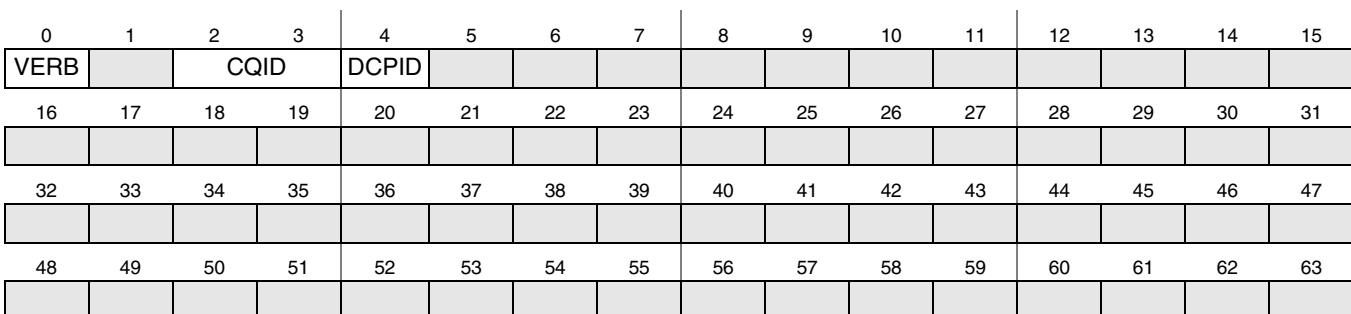
Figure 3-134. CEETM CQ Configure Response Format

**Table 3-128. CEETM CQ Configure Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPI_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. F3h = Error, command failed due to non-empty CQ. FFh = Error, unrecognized or invalid command was received.
2-63	Reserved

**3.3.9.7.4 CEETM Class Queue (CQ) Query**

This is used to read all of the fields (both software programmed and hardware managed) in a particular CQD.

**Figure 3-135. CEETM CQ Query Command Format****Table 3-129. CEETM CQ Query Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPI_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM CQ Query: Verb = 73h.
1	Reserved
2-3 CQID	Bits 0-6: Reserved Bits 7-15: CQID value of the CQ to be queried (32 channel CEETM). Bits 9-15: CQID value of the CQ to be queried (8 channel CEETM).
4 DCPIP	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT			CCGID		STATE		PFDR_HPTR		PFDR_TPTR					
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
OD1_XSFDR	OD2_XSFDR	OD3_XSFDR	OD4_XSFDR	OD5_XSFDR	OD6_XSFDR	RA1_XSFDR	RA2_XSFDR								
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
	FRM_CNT														
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-136. CEETM CQ Query Response Format

Table 3-130. CEETM CQ Query Response Format

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-5	Reserved
6-7 CCGID	Bits 0-11: Reserved Bits 12-15: Value from the CCGID field in the CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
8-9 STATE	Bits 0-6: Reserved Bits 7: Value from the FIP field in the CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a> Bits 8: Value from the IT field in the CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a> Bits 9-11: Value from the NOD field in the CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a> Bits 12-13: Value from the NRA field in the CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a> Bits 14-15: Value from the NPR field in the CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
10-12 PFDR_HPTR	Bits 0-23: Value from PFDR_HPTR in the CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
13-15 PFDR_TPTR	Bits 0-23: Value from PFDR_TPTR in the CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
16-17 OD1_XSFDR	Bits 0-3: Reserved Bits 4-15: Value from OD1_XSFDR_PTR in CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
18-19 OD2_XSFDR	Bits 0-3: Reserved Bits 4-15: Value from OD2_XSFDR_PTR in CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
20-21 OD3_XSFDR	Bits 0-3: Reserved Bits 4-15: Value from OD3_XSFDR_PTR in CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
22-23 OD4_XSFDR	Bits 0-3: Reserved Bits 4-15: Value from OD4_XSFDR_PTR in CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
24-25 OD5_XSFDR	Bits 0-3: Reserved Bits 4-15: Value from OD5_XSFDR_PTR in CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
26-27 OD6_XSFDR	Bits 0-3: Reserved Bits 4-15: Value from OD6_XSFDR_PTR in CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>

**Table 3-130. CEETM CQ Query Response Format (continued)**

Byte Field	Description
28-29 RA1_XSFDR	Bits 0-3: Reserved Bits 4-15: Value from RA1_XSFDR_PTR in CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
30-31 RA2_XSFDR	Bits 0-3: Reserved Bits 4-15: Value from RA2_XSFDR_PTR in CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
32	Reserved
33-35 FRM_CNT	Value from FRM_CNT in the CQD, see <a href="#">Section 3.3.20.4.1, “CEETM Class Queue Descriptor.”</a>
36-63	Reserved

### 3.3.9.7.5 CEETM Dequeue Context Table (DCT) Configure

This is used to configure entries in the CEETM DCT. After reset the DCT is initialized by hardware with default values as described in the table below, this command can be used to change the default configuration. See [Section 3.3.20.10, “CEETM dequeue context.”](#)

The DCT is indexed by the DCTIDX value that was programmed into the LFQMT, and contains the dequeue context (Context\_A and Context\_B) values that will be provided in the dequeue response during dequeues from a CEETM CQ.

Note that the LFQMT is indexed by a CEETM FQID when frames are enqueued onto an CEETM CQ, and that different LFQMT entries may use the same CQID but different DCTIDX values. This allows the same CQ to carry frames with different dequeue contexts.

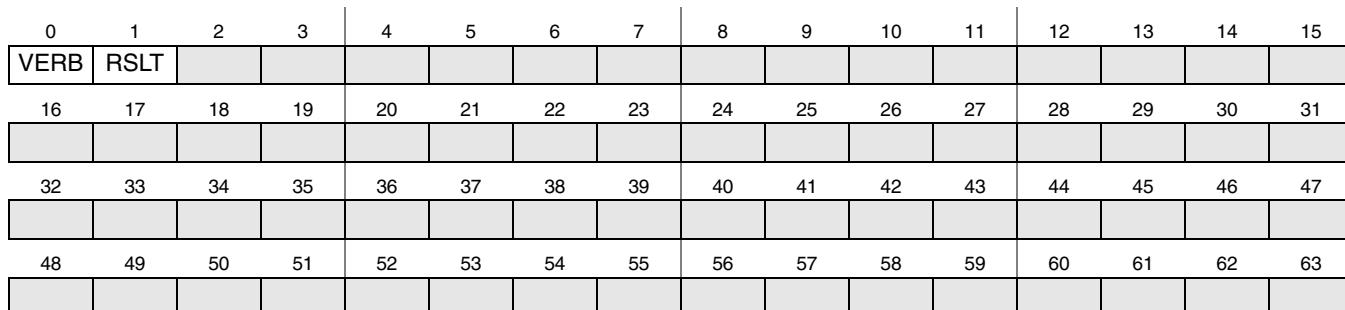
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	DCTIDX	DCPID													
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
				CONTEXT_B				CONTEXT_A							
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-137. CEETM DCT Configure Command Format****Table 3-131. CEETM DCT Configure Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM DCT Configure: Verb = 74h.
1	Reserved

**Table 3-131. CEETM DCT Configure Command Format**

Byte Field	Description
2-3 DCTIDX	Bits 0-3: Reserved Bits 4-15: DCTIDX (32 channel CEETM) Bits 6-15: DCTIDX (8 channel CEETM) serves as an index into the DCT, thus this field provides the address in the table at which the access will occur.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal's CEETM is addressed by this command.
5-19	Reserved
20-23 CONTEXT_B	Context_B value to be written to the DCT. After reset, this field is initialized to 0.
24-31 CONTEXT_A	Context_A value to be written to the DCT. After reset, this field is initialized to 0.
32-63	Reserved

**Figure 3-138. CEETM DCT Configure Response Format****Table 3-132. CEETM DCT Configure Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-63	Reserved

### 3.3.9.7.6 CEETM Dequeue Context Table (DCT) Query

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		DCTIDX		DCPID											
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-139. CEETM DCT Query Command Format

Table 3-133. CEETM DCT Query Command Format

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM DCT Query: Verb = 75h.
1	Reserved
2-3 DCTIDX	Bits 0-3: Reserved Bits 4-15: DCTIDX (32 channel CEETM) Bits 6-15: DCTIDX (8 channel CEETM) serves as an index into the DCT, thus this field provides the address in the table at which the access will occur.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT														
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
				CONTEXT_B				CONTEXT_A							
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-140. CEETM DCT Query Response Format

**Table 3-134. CEETM DCT Query Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-19	Reserved
20-23 CONTEXT_B	Context_B value read from the DCT.
24-31 CONTEXT_A	Context_A value read from the DCT.
32-63	Reserved

### 3.3.9.7.7 CEETM Class Scheduler Configure

This is used to configure the software programmable parameters that are used by the CEETM class schedulers. After reset all class schedulers are initialized by hardware with default values as described in the table below, this command can be used to change the default configuration. See [Section 3.3.20.6, “CEETM Class Queue Channel and Class Scheduler.”](#)

Note that configuring a class scheduler with this command resets the weighted bandwidth fair scheduling (WBFS) context of that class scheduler. The WBFS deficit values can be queried, see [Section 3.3.9.7.8, “CEETM Class Scheduler Query.”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		CQCID		DCPID							GPC	CREM	EREM		
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W8	W9	W10	W11	W12	W13	W14	W15								
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-141. CEETM Class Scheduler Configure Command Format****Table 3-135. CEETM Class Scheduler Configure Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Class Scheduler Configure: Verb = 76h.
1	Reserved

**Table 3-135. CEETM Class Scheduler Configure Command Format**

Byte Field	Description
2-3 CQCID	Class Queue Channel ID Bits 0-10: Reserved Bits 11-15: CQ channel ID (32 channel CEETM) Bits 13-15: CQ channel ID (8 channel CEETM) identifies the class scheduler (one per CQ channel) to be configured. Note that this field corresponds to the msbits of CQID, which identify the class queue channel. The lsbits of CQID identify the CQ within a CQ channel, and are not used for scheduler configuration.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal's CEETM is addressed by this command.
5-10	Reserved
11 GPC	Group Priority Configuration. After reset, this field is initialized to all 0. Bit 0: Reserved Bit 1: Combine groups flag. When set to 1, groups A and B are combined into a single weighted scheduling group of 8 class queues, whose priority is set by the configured group A priority. Bits 2-4: Group B priority. Group B is positioned directly below the CQ specified in this field in the strict priority order. Bits 5-7: Group A priority. Group A is positioned directly below the CQ specified in this field in the strict priority order. Note that if both groups A and B are set to the same priority value, and they are not combined, then group A has higher priority than group B. For example, if the two WBFS groups are not combined, and the group A and B priority values are both set to 4, then the class scheduler strict priority order will be: CQ0, CQ1, CQ2, CQ3, CQ4, group A, group B, CQ5, CQ6, CQ7.
12-13 CREM	CR Eligibility Mask. After reset, this field is initialized to all 0. Bits 0-5: Reserved. Bits 6: group B eligible Bits 7: group A eligible Bits 8-15: strict priority CQs 0-7 eligible Note these bits should be left as all 0 if the CQ channel is configured as unshaped, see <a href="#">Section 3.3.9.7.9, "CEETM Channel Mapping Configure."</a>
14-15 EREM	ER Eligibility Mask. After reset, this field is initialized to all 0. Bits 0-5: Reserved. Bits 6: group B eligible Bits 7: group A eligible Bits 8-15: strict priority CQs 0-7 eligible Note these bits should be left as all 0 if the CQ channel is configured as unshaped, see <a href="#">Section 3.3.9.7.9, "CEETM Channel Mapping Configure."</a>
16-23 W8-15	Weight for class queues 8 to 15 in the channel being configured. After reset, these fields are initialized to all 0. These weights are used by the weighted scheduling (WBFS) algorithm in the CEETM class schedulers, see <a href="#">Section 3.3.20.6.1, "CEETM Weighted Scheduling among Grouped Classes."</a> Each 8 bit weight code is formatted as: bit 0-4 (5 msbits): weight code y value bit 5-7 (3 lsbits): weight code x value And the formula for converting weight code to effective weight is: $\text{effective weight} = 2^x / (1 - (y/64))$
24-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT														
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-142. CEETM Class Scheduler Configure Response Format****Table 3-136. CEETM Class Scheduler Configure Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-63	Reserved

### 3.3.9.7.8 CEETM Class Scheduler Query

This is used to read the CEETM class scheduler configuration, as well as the hardware managed context used by the class scheduler.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		CQCID		DCPID											
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-143. CEETM Class Scheduler Query Command Format****Table 3-137. CEETM Class Scheduler Query Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Class Scheduler Query: Verb = 77h.
1	Reserved

**Table 3-137. CEETM Class Scheduler Query Command Format**

Byte Field	Description
2-3 CQCID	Class Queue Channel ID Bits 0-10: Reserved Bits 11-15: CQ channel ID (32 channel CEETM) Bits 13-15: CQ channel ID (8 channel CEETM) identifies the class scheduler (one per CQ channel) whose configuration is to be read Note that this field corresponds to the msbits of CQID, which identify the class queue channel. The lsbits of CQID identify the CQ within a CQ channel, and are not used for scheduler configuration.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal's CEETM is addressed by this command.
5-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT										GPC	CREM	EREM		
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W8	W9	W10	W11	W12	W13	W14	W15								WBFSLIST
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
	D8			D9				D10					D11		
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	D12			D13				D14					D15		

**Figure 3-144. CEETM Class Scheduler Query Response Format****Table 3-138. CEETM Class Scheduler Query Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0-1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-10	Reserved
11 GPC	Group Priority Configuration. Same bit definitions as in <a href="#">Table 3-135</a> .
12-13 CREM	CR Eligibility Mask. Same bit definitions as in <a href="#">Table 3-135</a> .
14-15 EREM	ER Eligibility Mask. Same bit definitions as in <a href="#">Table 3-135</a> .
16-23 W8-15	Configured scheduling weight for class queues 8 to 15 in the channel being queried. Same bit definitions as in <a href="#">Table 3-135</a> .
29-31 WBFSLIST	Weighted scheduling list for class queues 8-15 in this channel (3 bits per CQ). This is a hardware managed context that can be queried for debug purposes.

**Table 3-138. CEETM Class Scheduler Query Response Format (continued)**

<b>Byte Field</b>	<b>Description</b>
33-35 D8	Weighted scheduling Deficit for class queue 8 in this channel. This is a hardware managed context that can be queried for debug purposes.
37-39 D9	Weighted scheduling Deficit for class queue 9 in this channel.
41-43 D10	Weighted scheduling Deficit for class queue 10 in this channel.
45-47 D11	Weighted scheduling Deficit for class queue 11 in this channel.
49-51 D12	Weighted scheduling Deficit for class queue 12 in this channel.
53-55 D13	Weighted scheduling Deficit for class queue 13 in this channel.
57-59 D14	Weighted scheduling Deficit for class queue 14 in this channel.
61-63 D15	Weighted scheduling Deficit for class queue 15 in this channel.

### 3.3.9.7.9 CEETM Channel Mapping Configure

This is used to configure the software programmable parameters that are used by the CEETM channel schedulers. After reset all channel schedulers are initialized by hardware with default values as described in the table below, this command can be used to change the default configuration. See [Section 3.3.20.8, “CEETM Channel Scheduler.”](#)

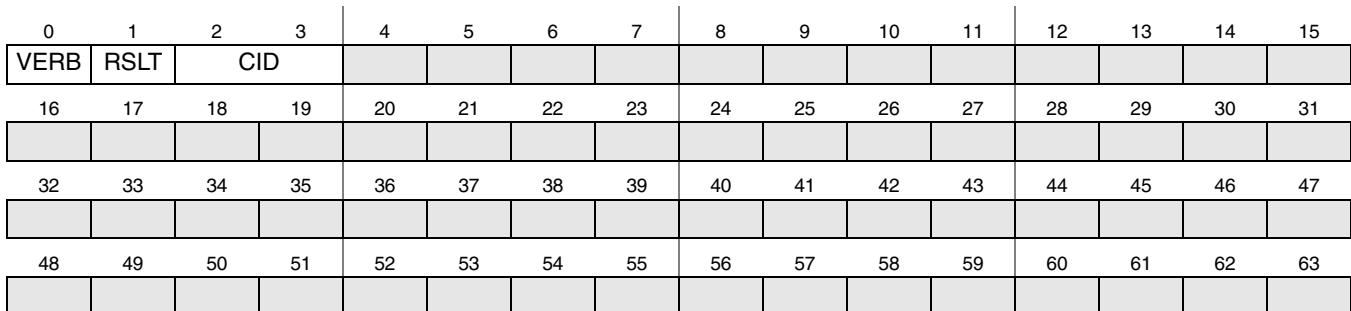
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		CID		DCPID	MAP										
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-145. CEETM Channel Mapping Configure Command Format****Table 3-139. CEETM Channel Mapping Configure Command Format**

<b>Byte Field</b>	<b>Description</b>
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Mapping/Shaper/Flow Control Configure: Verb = 78h.
1	Reserved

**Table 3-139. CEETM Channel Mapping Configure Command Format**

Byte Field	Description
2-3 CID	Command ID. Bits 0-3: Command type: 0 = Channel Mapping Configure/Query. Bits 4-10: Reserved Bits 11-15: CQ channel ID (32 channel CEETM) Bits 13-15: CQ channel ID (8 channel CEETM) Identifies the CQ channel to be configured. Note that this field corresponds to the msbits of CQID, which identify the class queue channel. The lsbits of CQID identify the CQ within a CQ channel, and are not used for this command.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal's CEETM is addressed by this command.
5 MAP	Channel Mapping. Bit 0: Shaped. 0 = Unshaped channel, 1 = Shaped channel. After reset this field is initialized to 0. If this bit is 0, this CQ channel will be treated as an unshaped channel by the channel scheduler, giving it higher priority than shaped channels, see <a href="#">Section 3.3.20.8, “CEETM Channel Scheduler.”</a> Bits 1-4: Reserved Bits 5-7: LNI ID: Configures which LNI (i.e. which channel scheduler) the CQ channel specified in CID belongs to. Note that LNI 0 and LNI 1 are optimized for use with 10 GE Ethernet ports, therefore CQ channels whose egress traffic is destined for a 10 GE port on FMan should always use either LNI 0 or LNI 1. After reset, this field is set equal to the lsbits of the CQ channel ID value (CID bits 11-15), such that consecutive channels belong to consecutive LNIs.
6-63	Reserved

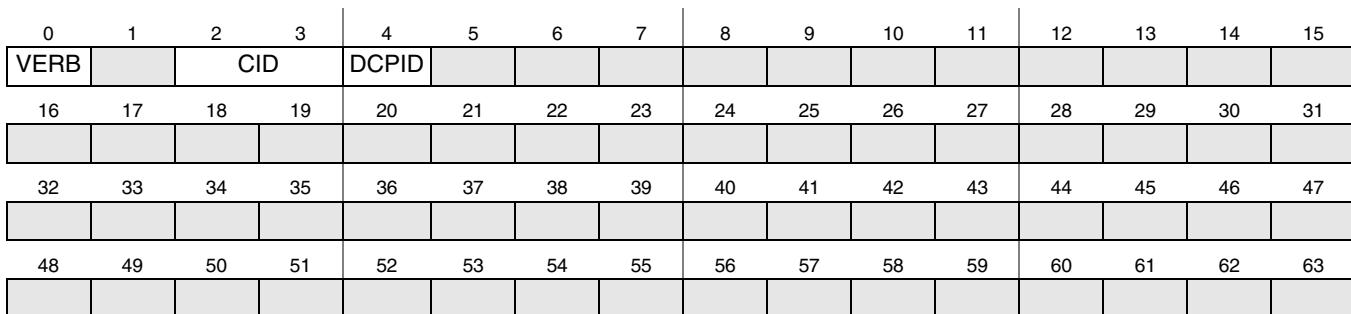
**Figure 3-146. CEETM Channel Mapping Configure Response Format****Table 3-140. CEETM Channel Mapping Configure Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.

**Table 3-140. CEETM Channel Mapping Configure Response Format (continued)**

Byte Field	Description
2-3 CID	Command ID. This response field carries the same CID value as in the command.
4-63	Reserved

### 3.3.9.7.10 CEETM Channel Mapping Query

**Figure 3-147. CEETM Channel Mapping Query Command Format****Table 3-141. CEETM Channel Mapping Query Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPI_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Mapping/Shaper/Flow Control Query: Verb = 79h.
1	Reserved
2-3 CID	Command ID. Bits 0-3: Command type: 0 = Channel Mapping Configure/Query. Bits 4-10: Reserved Bits 11-15: CQ channel ID (32 channel CEETM) Bits 13-15: CQ channel ID (8 channel CEETM) Identifies the CQ channel to be queried. Note that this field corresponds to the msbits of CQID, which identify the class queue channel. The lsbits of CQID identify the CQ within a CQ channel, and are not used for this command.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT	CID		MAP											
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-148. CEETM Channel Mapping Query Response Format

Table 3-142. CEETM Channel Mapping Query Response Format

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-3 CID	Command ID. This response field carries the same CID value as in the command.
4	Reserved
5 MAP	Channel Mapping. Same bit definitions as in <a href="#">Table 3-139., “CEETM Channel Mapping Configure Command Format.”</a>
6-63	Reserved

### 3.3.9.7.11 CEETM Sub-Portal Mapping Configure

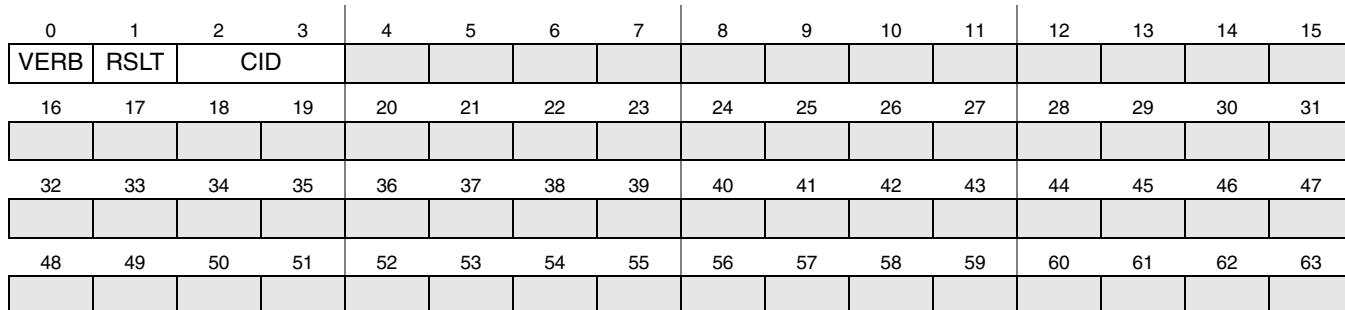
This is used to configure the CEETM sub-portal to LNI mapping. After reset all settings are initialized by hardware with default values as described in the table below, this command can be used to change the default configuration. See [Section 3.3.20.12, “CEETM Sub-Portal Mapping and Physical Interface Switchover.”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		CID		DCPID	MAP										
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-149. CEETM Sub-Portal Mapping Configure Command Format

**Table 3-143. CEETM Sub-Portal Mapping Configure Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPI_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Mapping/Shaper/Flow Control Configure: Verb = 78h.
1	Reserved
2-3 CID	Command ID. Bits 0-3: Command type: 1 = Sub-Portal Mapping Configure/Query. Bits 4-11: Reserved Bits 12-15: SPID, identifies the sub-portal to be configured.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5 MAP	Sub-Portal Mapping. Bits 0-4: Reserved Bits 5-7: LNI ID: Logical Network Interface (LNI) attached to this sub-portal. Valid only for sub-portals in which CEETM scheduling/shaping mode has been enabled (via DCPI_CFG[CEETME]). See <a href="#">Section 3.3.20.1, “CEETM Overview,”</a> and <a href="#">Section 3.3.20.12, “CEETM Sub-Portal Mapping and Physical Interface Switchover.”</a> Note that LNI 0 and LNI 1 are optimized for use with 10 GE Ethernet ports, therefore sub-portals whose egress traffic is destined for a 10 GE port on FMan should always use either LNI 0 or LNI 1. After reset, this field is set equal to the lsbits of the SPID value (CID bits 12-15), such that consecutive sub-portals are connected to consecutive LNIs.
6-63	Reserved

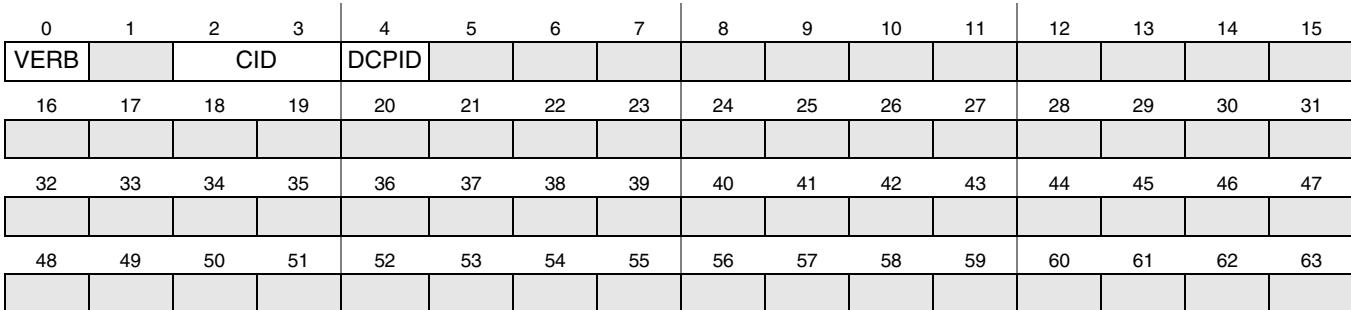
**Figure 3-150. CEETM Sub-Portal Mapping Configure Response Format****Table 3-144. CEETM Sub-Portal Mapping Configure Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPI_RR0-1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.

**Table 3-144. CEETM Sub-Portal Mapping Configure Response Format (continued)**

Byte Field	Description
2-3 CID	Command ID. This response field carries the same CID value as in the command.
4-63	Reserved

### 3.3.9.7.12 CEETM Sub-Portal Mapping Query

**Figure 3-151. CEETM Sub-Portal Mapping Query Command Format****Table 3-145. CEETM Sub-Portal Mapping Query Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Mapping/Shaper/Flow Control Query: Verb = 79h.
1	Reserved
2-3 CID	Command ID. Bits 0-3: Command type: 1 = Sub-Portal Mapping Configure/Query. Bits 4-11: Reserved Bits 12-15: SPID, identifies the sub-portal to be queried.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT	CID		MAP											
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-152. CEETM Sub-Portal Mapping Query Response Format****Table 3-146. CEETM Sub-Portal Mapping Query Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-3 CID	Command ID. This response field carries the same CID value as in the command.
4	Reserved
5 MAP	Sub-Portal Mapping. Same bit definitions as in <a href="#">Table 3-143., “CEETM Sub-Portal Mapping Configure Command Format.”</a>
6-63	Reserved

### 3.3.9.7.13 CEETM Shaper Configure

This is used to configure the software programmable parameters that are used by the CEETM shapers. After reset all shapers are initialized by hardware with default values as described in the table below, this command can be used to change the default configuration. See [Section 3.3.20.7, “CEETM Channel Shapers”](#) and [Section 3.3.20.8.2, “CEETM LNI shapers.”](#)

Configuring a shaper with this command also fills that shaper’s token bucket to its token bucket limit (TBL). The token bucket accumulated credit values can be queried, see [Section 3.3.9.7.14, “CEETM Shaper Query.”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		CID		DCPID	CPL	CRTCR				ERTCR			CRTBL	ERTBL	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
MPS															
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 3-153. CEETM Shaper Configure Command Format

Table 3-147. CEETM Shaper Configure Command Format

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Mapping/Shaper/Flow Control Configure: Verb = 78h.
1	Reserved
2-3 CID	Command ID. Bits 0-3: Command type: 2 = Channel Shaper Configure/Query. 3 = LNI Shaper Configure/Query. If Command type = Channel Shaper Configure/Query: Bits 4-10: Reserved Bits 11-15: CQ channel ID (32 channel CEETM) Bits 13-15: CQ channel ID (8 channel CEETM) Identifies the CQ channel shaper to be configured. Note that this field corresponds to the msbits of CQID, which identify the class queue channel. The lsbits of CQID identify the CQ within a CQ channel, and are not used for this command. If Command type = LNI Shaper Configure/Query: Bits 4-12: Reserved Bits 13-15: LNI ID, identifies the LNI shaper to be configured.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5 CPL	Bit 0: Shaper coupling: 0 = CR and ER shapers are not coupled, 1 = CR and ER shapers are coupled, such that once the CR bucket is full (tokens = token bucket limit), additional CR tokens are added instead to the ER bucket for the same channel or LNI, up to the ER shaper’s token bucket limit. See <a href="#">Section 3.3.20.7, “CEETM Channel Shapers.”</a> Note that shaper coupling is incompatible with infinite token credit rate, therefore if either CR or ER are set for infinite token credit rate (all 1’s in CRTCR or ERTCR) then shaper coupling should be set to 0. Bits 1-2: Reserved. Bits 3-7: Overhead Accounting Length (OAL). This value is added to the actual length of each frame when performing shaper calculations. Only valid for LNI Shaper Configure. The value programmed for a given LNI will also apply to all channels that are mapped to that LNI (see <a href="#">Section 3.3.9.7.9, “CEETM Channel Mapping Configure”</a> ). After reset this field is set to 0.

**Table 3-147. CEETM Shaper Configure Command Format**

Byte Field	Description
6-8 CRTCR	CR token credit rate, for the shaper indicated in CID. After reset this field is set to 0. This field, together with the credit update reference period (see <a href="#">Section 3.2.4.27, “CEETM Configuration Shaper Pre-Scaler Register (CEETM_CFG_PRES)”</a> ) determines the shaper’s output rate. This field is not used in a channel shaper if the channel is configured as unshaped, see <a href="#">Section 3.3.9.7.9, “CEETM Channel Mapping Configure”</a> . Setting this field to all 1’s configures an infinite token credit rate. Setting CRTCR and CRTBL to 0 disables the CR rate in this shaper. Bits 0-10: Integer portion of the credit rate. Bits 11-23: Fractional portion of the credit rate.
9-11 ERTCR	ER token credit rate, for the shaper indicated in CID. After reset this field is set to 0. This field, together with the credit update reference period (see <a href="#">Section 3.2.4.27, “CEETM Configuration Shaper Pre-Scaler Register (CEETM_CFG_PRES)”</a> ) determines the shaper’s output rate. This field is not used in a channel shaper if the channel is configured as unshaped, see <a href="#">Section 3.3.9.7.9, “CEETM Channel Mapping Configure”</a> . Setting this field to all 1’s configures an infinite token credit rate. Setting ERTCR and ERTBL to 0 disables the ER rate in this shaper. Bits 0-10: Integer portion of the credit rate. Bits 11-23: Fractional portion of the credit rate.
12-13 CRTBL	CR Token Bucket Limit (TBL), for the shaper indicated in CID. After reset this field is set to 0. TBL is related to maximum burst size, see <a href="#">Section 3.3.20.7, “CEETM Channel Shapers.”</a> TBL is also used as a weight in the channel scheduler fair queueing algorithms, see <a href="#">Section 3.3.20.8.1, “CEETM channel scheduler Shaped Fair Queueing (ShFQ).”</a>
14-15 ERTBL	ER Token Bucket Limit (TBL), for the shaper indicated in CID. After reset this field is set to 0. TBL is related to maximum burst size, see <a href="#">Section 3.3.20.7, “CEETM Channel Shapers.”</a> TBL is also used as a weight in the channel scheduler fair queueing algorithms, see <a href="#">Section 3.3.20.8.1, “CEETM channel scheduler Shaped Fair Queueing (ShFQ).”</a>
16 MPS	Bit 0: Reserved. Bits 1-7: Minimum Packet Size if a dequeued frame length is less than this value, the length will be rounded up to this value when performing shaper calculations. The frame length is rounded up to this value before any OAL value is applied. Only valid for LNI Shaper Configure. The value programmed for a given LNI will also apply to all channels that are mapped to that LNI (see <a href="#">Section 3.3.9.7.9, “CEETM Channel Mapping Configure”</a> ). Disabled if set to 0.
17-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT	CID													
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

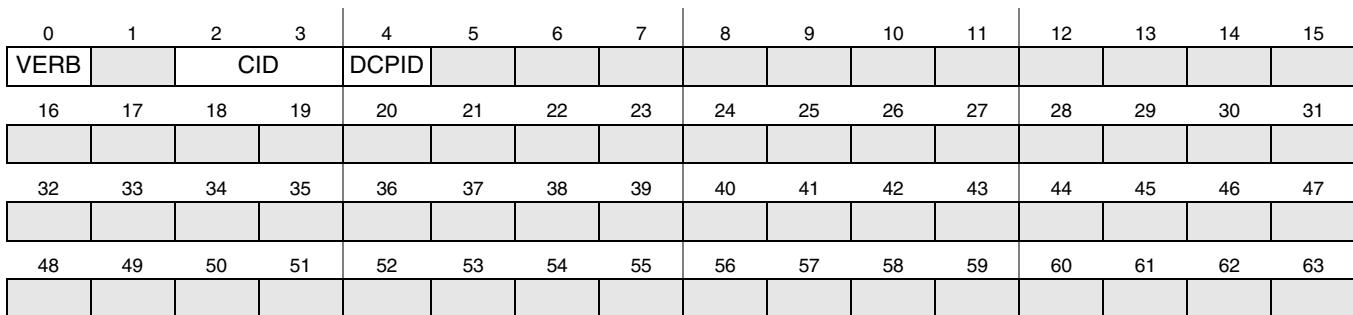
**Figure 3-154. CEETM Shaper Configure Response Format**

**Table 3-148. CEETM Shaper Configure Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-3 CID	Command ID. This response field carries the same CID value as in the command.
4-63	Reserved

**3.3.9.7.14 CEETM Shaper Query**

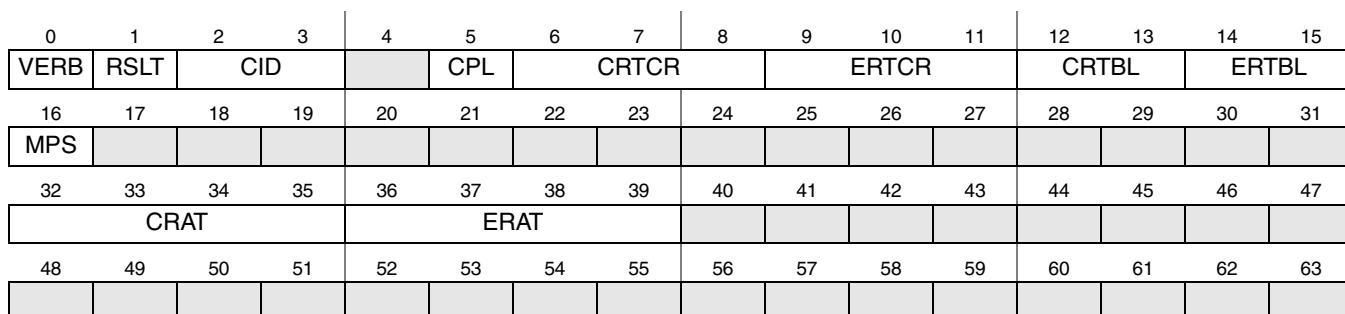
This is used to read the CEETM shaper configuration, as well as the hardware managed context used by the CEETM shapers.

**Figure 3-155. CEETM Shaper Query Command Format****Table 3-149. CEETM Shaper Query Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Mapping/Shaper/Flow Control Query: Verb = 79h.
1	Reserved

**Table 3-149. CEETM Shaper Query Command Format**

Byte Field	Description
2-3 CID	<p>Command ID.</p> <p>Bits 0-3: Command type:</p> <ul style="list-style-type: none"> <li>2 = Channel Shaper Configure/Query.</li> <li>3 = LNI Shaper Configure/Query.</li> </ul> <p>If Command type = Channel Shaper Configure/Query:</p> <ul style="list-style-type: none"> <li>Bits 4-10: Reserved</li> <li>Bits 11-15: CQ channel ID (32 channel CEETM)</li> <li>Bits 13-15: CQ channel ID (8 channel CEETM)</li> <li>Identifies the CQ channel shaper to be queried.</li> <li>Note that this field corresponds to the msbits of CQID, which identify the class queue channel.</li> <li>The lsbits of CQID identify the CQ within a CQ channel, and are not used for this command.</li> </ul> <p>If Command type = LNI Shaper Configure/Query:</p> <ul style="list-style-type: none"> <li>Bits 4-12: Reserved</li> <li>Bits 13-15: LNI ID, identifies the LNI shaper to be queried.</li> </ul>
4 DCPID	<p>Bits 0-3: Reserved</p> <p>Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal's CEETM is addressed by this command.</p>
5-63	Reserved

**Figure 3-156. CEETM Shaper Query Response Format****Table 3-150. CEETM Shaper Query Response Format**

Byte Field	Description
0 VERB	<p>Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a></p> <p>Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.</p>
1 RSLT	<p>Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.</p>
2-3 CID	Command ID. This response field carries the same CID value as in the command.
4	Reserved
5 CPL	<p>Shaper Coupling and OAL. Same bit definitions as in <a href="#">Table 3-147., “CEETM Shaper Configure Command Format.”</a></p>

**Table 3-150. CEETM Shaper Query Response Format (continued)**

Byte Field	Description
6-8 CRTCR	CR token credit rate, for the channel or LNI shaper specified in CID. Same bit definitions as in <a href="#">Table 3-147., “CEETM Shaper Configure Command Format.”</a>
9-11 ERTCR	ER token credit rate, for the channel or LNI shaper specified in CID. Same bit definitions as in <a href="#">Table 3-147., “CEETM Shaper Configure Command Format.”</a>
12-13 CRTBL	CR token bucket limit, for the channel or LNI shaper specified in CID. Same bit definitions as in <a href="#">Table 3-147., “CEETM Shaper Configure Command Format.”</a>
14-15 ERTBL	ER token bucket limit, for the channel or LNI shaper specified in CID. Same bit definitions as in <a href="#">Table 3-147., “CEETM Shaper Configure Command Format.”</a>
16 MPS	Minimum packet size Same bit definitions as in <a href="#">Table 3-147., “CEETM Shaper Configure Command Format.”</a>
32-35 CRAT	CR Accumulated Tokens, for the channel or LNI shaper specified in CID This is a hardware managed context that can be queried for debug purposes. Bits 0-1: Reserved Bit 2: Sign Bits 3-18: credit, integer portion Bits 19-31: credit, fractional portion
36-39 ERAT	ER Accumulated Tokens, for the channel or LNI shaper specified in CID. This is a hardware managed context that can be queried for debug purposes. Bits 0-1: Reserved Bit 2: Sign Bits 3-18: credit, integer portion Bits 19-31: credit, fractional portion
40-63	Reserved

### 3.3.9.7.15 CEETM Traffic Class Flow Control Configure

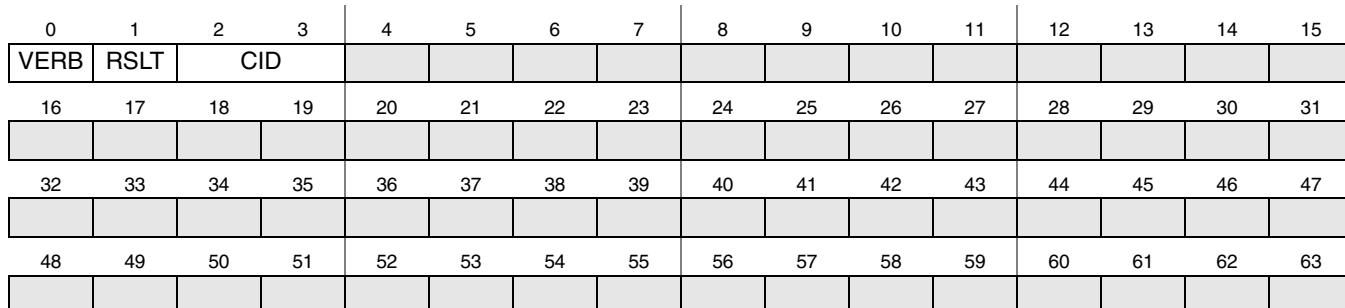
This is used to configure CEETM traffic class flow control for each LNI. After reset all settings are initialized by hardware with default values as described in the table below, this command can be used to change the default configuration. See [Section 3.3.20.13, “CEETM traffic class flow control.”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		CID		DCPID											
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LUNITCFCC															
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-157. CEETM Traffic Class Flow Control Configure Command Format**

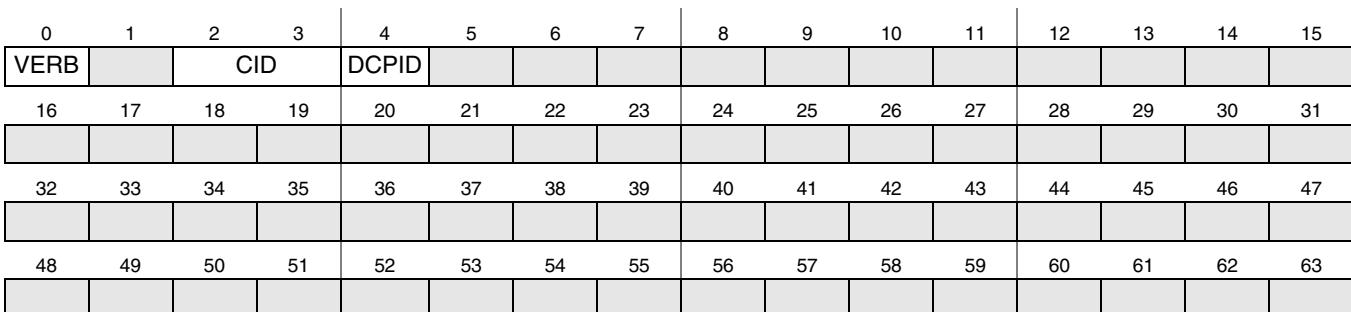
**Table 3-151. CEETM Traffic Class Flow Control Configure Command Format**

Byte Field	Description															
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPI_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Mapping/Shaper/Flow Control Configure: Verb = 78h.															
1	Reserved															
2-3 CID	Command ID. Bits 0-3: Command type: 4 = Traffic Class Flow Control Configure/Query. Bits 4-12: Reserved Bits 13-15: LNI ID, identifies the LNI to be configured.															
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.															
5-15	Reserved															
16-23 LNITCFCC	LNI Traffic Class Flow Control Configuration, for the LNI specified in CID. This fields configures how traffic class flow control is applied to each CQ in all CQ channels that are used by this LNI. The same flow control applies to all CQ channels used by this LNI.  If a traffic class flow control command (TCFCC) is received on the DCP portal using this instance of CEETM (i.e. DCP portal number = DCPID), and the sub-portal on which the command is received is configured to use CEETM scheduling mode (see <a href="#">Section 3.2.4.11, “DCP Configuration Registers (DCPi_CFG).”</a> ), then the flow control state (XON or XOFF) from each of the 8 traffic classes in that command is applied to the CQs which are configured to belong to each of those traffic classes.  The traffic class flow control configuration for each CQ uses 4 bits as follows: bit 0: Traffic class flow control enable for this CQ: 0 = disabled, 1 = enabled. bits 1-3: Identifies the traffic class (0 to 7) to which this CQ belongs. The 4 bits for each CQ are mapped within this 64 bit field as follows: Bits 0-3: Traffic Class Flow Control Configuration for CQ0 in all channels used by this LNI. Bits 4-7: Traffic Class Flow Control Configuration for CQ1 in all channels used by this LNI. .... Bits 56-59: Traffic Class Flow Control Configuration for CQ14 in all channels used by this LNI. Bits 60-63: Traffic Class Flow Control Configuration for CQ15 in all channels used by this LNI. After reset, this field is initialized to all 0, such that flow control is disabled on all CQs.															
24-63	Reserved															

**Figure 3-158. CEETM Traffic Class Flow Control Configure Response Format**

**Table 3-152. CEETM Traffic Class Flow Control Configure Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPI_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-3 CID	Command ID. This response field carries the same CID value as in the command.
4-63	Reserved

**3.3.9.7.16 CEETM Traffic Class Flow Control Query****Figure 3-159. CEETM Traffic Class Flow Control Query Command Format****Table 3-153. CEETM Traffic Class Flow Control Query Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPI_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Mapping/Shaper/Flow Control Query: Verb = 79h.
1	Reserved
2-3 CID	Command ID. Bits 0-3: Command type: 4 = Traffic Class Flow Control Configure/Query. Bits 4-12: Reserved Bits 13-15: LNI ID, identifies the LNI to be queried.
4 DCPIP	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT	CID													
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
LNITCFCC															
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-160. CEETM Traffic Class Flow Control Query Response Format****Table 3-154. CEETM Traffic Class Flow Control Query Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-3 CID	Command ID. This response field carries the same CID value as in the command.
4-15	Reserved
16-23 LNITCFCC	LNI Traffic Class Flow Control Configuration, for the LNI specified in CID. Same bit definitions as in <a href="#">Table 3-151., “CEETM Traffic Class Flow Control Configure Command Format.”</a>
24-63	Reserved

### 3.3.9.7.17 CEETM Class Congestion Group Record (CCGR) CM Configure

This is used to configure the software programmable parameters in a CEETM Class Congestion Group Record (CCGR) that are used for congestion management (CM). After reset all CCGRs are initialized by hardware with a default value of 0, this command can be used to change the default configuration. See [Section 3.3.20.11, “CEETM Class Congestion Management and Avoidance \(CCM\).”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	CCGRID	DCPID		WE_MASK	CTL	CDV	CSCN_TUPD	OAL		CS_THRES					
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CS_THRES_X	TD_THRES	WR_PARM_G				WR_PARM_Y				WR_PARM_R					
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

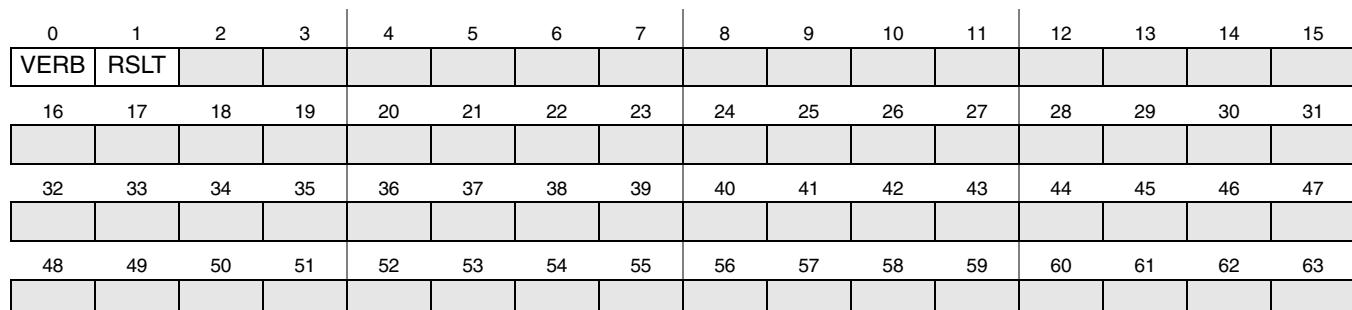
**Figure 3-161. CEETM CCCR CM Configure Command Format**

**Table 3-155. CEETM CCGR CM Configure Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM CCGR Configure: Verb = 7Ah.
1	Reserved
2-3 CCGRID	Class Congestion Group Record ID Bits 0-1: Command Type: 0 = CCGR CM Configure. Bits 2-6: Reserved Bits 7-11: CCG channel to be configured (32 channel CEETM). Bits 9-11: CCG channel to be configured (8 channel CEETM). Note that this field corresponds to the msbits of CQID, which identify the class queue channel. See <a href="#">Section 3.3.20.11, “CEETM Class Congestion Management and Avoidance (CCM).”</a> Bits 12-15: CCGID, identifies the CCG within the CCG channel.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5	Reserved
6-7 WE_MASK	Write Enable Mask. This is a 16 bit field containing individual bits which enable or disable the update to a specific CCGR field. For each bit, 0 = write disabled, 1 = write enabled. Bit 0 (msb): Write enable for CDV Bit 1: Write enable for TD_MODE Bit 2: Write enable for TD_THRES Bit 3: Write enable for CS_THRES_X Bit 4: Write enable for OAL Bit 5: Write enable for WR_PARM_G Bit 6: Write enable for WR_PARM_Y Bit 7: Write enable for WR_PARM_R Bit 8: Write enable for WR_EN_G Bit 9: Write enable for WR_EN_Y Bit 10: Write enable for WR_EN_R Bit 11: Write enable for CSCN_EN Bit 12: Write enable for CSCN_TUPD Bit 13: Write enable for TD_EN Bit 14: Write enable for CS_THRES Bit 15: Write enable for MODE.
8 CTL	Control. Carries several single bit CCGR fields. Bit 0: Reserved Bit 1: WR_EN_G field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a> Bit 2: WR_EN_Y field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a> Bit 3: WR_EN_R field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a> Bit 4: TD_EN field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a> Bit 5: TD_MODE field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a> Bit 6: CSCN_EN field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a> Bit 7: MODE field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a>
9 CDV	Congestion State Change Notification (CSCN) for DCP portals, Virtual CGID. CSCN_DCP_VCGID field of the CCGR, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a>

**Table 3-155. CEETM CCGR CM Configure Command Format**

Byte Field	Description
10-11 CSCN_TUPD	Congestion State Change Notification Target Update The CSCN_TARG_SWP and CSCN_TARG_DCP fields, as stored internally in the CCGR (see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR)”</a> ), contain one bit per software or DCP portal. This configuration command allows software to modify exactly one of these bits per command issued. bit 0: Value to be written to the CSCN_TARG_SWP or CSCN_TARG_DCP portal bit. bit 1: Portal type, 0 = software portal, 1 = DCP portal. bit 2-5: Reserved. bits 6-15: Portal number whose CSCN_TARG bit is to be written with the value in bit 0 of this field. If the portal number field contains an out of range value (portal does not exist), no CSCN_TARG bits are modified.
12 OAL	Overhead Accounting Length OAL field of the CCGR, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR)”</a> .
13	Reserved
14-15 CS_THRES	Bit 0-2: Reserved Bit 3-15: CS_THRES field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR)”</a> .
16-17 CS_THRES_X	Bit 0-2: Reserved Bit 3-15: CS_THRES_X field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR)”</a> .
18-19 TD_THRES	Bit 0-2: Reserved Bit 3-15: TD_THRES field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR)”</a> .
20-23 WR_PARM_G	CCGR WR_PARM_G field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR)”</a> .
24-27 WR_PARM_Y	CCGR WR_PARM_Y field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR)”</a> .
28-31 WR_PARM_R	CCGR WR_PARM_R field, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR)”</a> .
32-63	Reserved

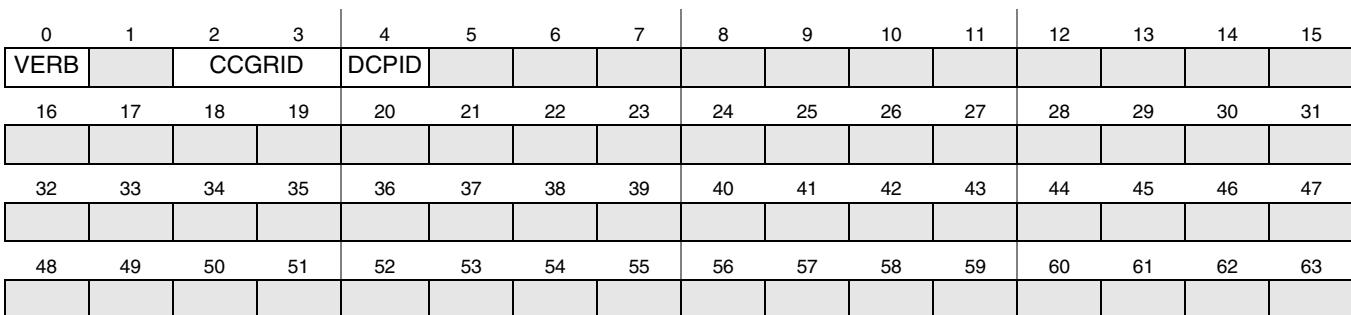
**Figure 3-162. CEETM CCGR CM Configure Response Format**

**Table 3-156. CEETM CCGR CM Configure Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPI_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-63	Reserved

**3.3.9.7.18 CEETM Class Congestion Group Record (CCGR) CM Query**

This is used to read the configuration and state fields within a CEETM CCGR record that relate to congestion management (CM).

**Figure 3-163. CEETM CCGR CM Query Command Format****Table 3-157. CEETM CCGR CM Query Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPI_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM CCGR Query: Verb = 7Bh.
1	Reserved
2-3 CCGRID	Class Congestion Group Record ID Bits 0-1: Command Type: 0 = CCGR CM Query. Bits 2-6: Reserved Bits 7-11: CCG channel to be queried (32 channel CEETM). Bits 9-11: CCG channel to be queried (8 channel CEETM). Note that this field corresponds to the msbits of CQID, which identify the class queue channel. See <a href="#">Section 3.3.20.11, “CEETM Class Congestion Management and Avoidance (CCM).”</a> Bits 12-15: CCGID, identifies the CCG within the CCG channel.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT							CTL	CDV			OAL		CS_THRES	
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CS_THRES_X	TD_THRES			WR_PARM_G				WR_PARM_Y				WR_PARM_R			
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
CSCN_TARG_DCP	DCP_LSN			I_CNT								A_CNT			
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CSCN_TARG_SWP															

**Figure 3-164. CEETM CCGR CM Query Response Format****Table 3-158. CEETM CCGR CM Query Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-7	Reserved
8-31	These bytes carry the software configurable CCGR fields, same definitions as in <a href="#">Table 3-155., “CEETM CCGR CM Configure Command Format,”</a> except for bytes 10 and 11, which are reserved in the query response.
32-33 CSCN_TARG_DCP	Congestion State Change Notification Target bit mask for DCP portals. Value read from the CSCN_TARG_DCP field in the CCGR, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a>
34 DCP_LSN	Congestion State Change Notification for DCP portals, Last State Notified. bit 0-6 (msbs): Reserved. bit 7 (lsb): Value read from the CSCN_DCP_LSN field in the CCGR, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a>
35-39 I_CNT	Instantaneous Count (byte or frame). Value read from the I_CNT field of the CCGR, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a>
40-42	Reserved
43-47 A_CNT	Average Count (byte or frame). Value read from the A_CNT field of the CCGR, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a>
48-63 CSCN_TARG_SWP	Congestion State Change Notification Target bit mask for software portals. Value read from the CSCN_TARG_SWP field in the CCGR, see <a href="#">Section 3.3.20.11.2, “CEETM Class Congestion Group Record (CCGR).”</a>

### 3.3.9.7.19 CEETM Query Congestion State

This command is used to query the state of the CEETM CCGs. The state of 256 CCGs can be queried by a single command.

This command returns a snapshot of the state of the 256 CCGs at the time that the command executes. It is used by software in response to receiving a CEETM CSCN interrupt. See [Section 3.3.20.11.1, “CEETM Congestion State Change Notifications \(CEETM CSCN\).”](#)

The format of both the command (written to the CR) and the response (read from RR0 / RR1) are shown below with byte granularity.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		CCGRID		DCPID											
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-165. CEETM Query Congestion State Command Format**

**Table 3-159. CEETM Query Congestion State Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPI_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM CCGR Query: Verb = 7Bh.
1	Reserved
2-3 CCGRID	Class Congestion Group Record ID Bits 0-1: Command Type: 3 = CEETM Query Congestion State. Bits 2-14: Reserved Bits 15: Identifies which group of 256 CCGs is to be queried: 0 = Query congestion state of CCGs 0 to 255 (0 to 128 for 8 channel CEETM). 1 = Query congestion state of CCGs 256 to 511 (not used for 8 channel CEETM).
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal’s CEETM is addressed by this command.
5-63	Reserved

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	RSLT														
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
CCG_STATE															
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CCG_STATE															

**Figure 3-166. CEETM Query Congestion State Response Format**

**Table 3-160. CEETM Query Congestion State Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-31	Reserved
32-63 CCG_STATE	Congestion state for a group of 256 CCGs. The specific group queried is identified as n, where n = bit 15 of CCGRID in the command. For each bit the state meaning is: 0 = not in congestion, 1 = in congestion. Bit 0 (msbit): State of CCG number (n*256)+0 (CCG 0 if n = 0, CCG 256 if n = 1) Bit 1: State of CCG number (n*256)+1 .... Bit 254: State of CCG number (n*256)+254 Bit 255 (lsbit): State of CCG number (n*256)+255 (CCG 255 if n = 0, CCG 511 if n = 1)

**3.3.9.7.20 CEETM Class Queue (CQ) Peek / Pop / XSFDR Read**

This command can be used to examine and optionally remove (dequeue) a frame at the head of a class queue. Examination of a frame without removal may be useful for debugging, while dequeuing of a frame may be used to drain a CQ if needed, for example if the link which was consuming a CQ has been disabled. XSFDR read may be useful for debug, note that reading a XSFDR that has never been used in a CQ may result in ECC errors due to uninitialized memory.

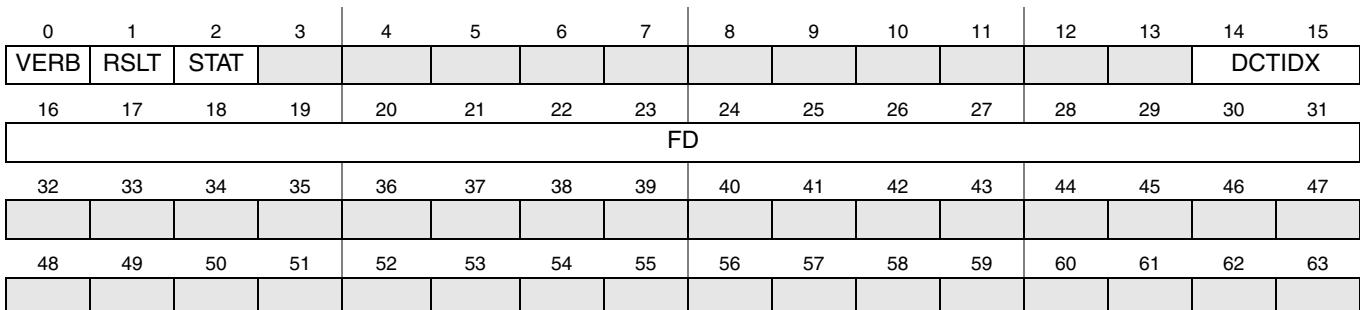
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB		CQID		DCPID	CT	XSFDR									
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

**Figure 3-167. CEETM CQ Peek/Pop/XSFDR Read Command Format****Table 3-161. CEETM CQ Peek/Pop/XSFDR Read Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM CQ Peek/Pop: Verb = 7Ch.
1	Reserved

**Table 3-161. CEETM CQ Peek/Pop/XSFDR Read Command Format**

Byte Field	Description
2-3 CQID	Bits 0-6: Reserved Bits 7-15: CQID value of the CQ targeted by this command (32 channel CEETM). Valid if CT = 0 or 1. Bits 9-15: CQID value of the CQ targeted by this command (8 channel CEETM). Valid if CT = 0 or 1.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal's CEETM is addressed by this command.
5 CT	Bits 0-5: Reserved. Bits 6-7: Command Type 0 = Peek, returns the FD at the head of the CQ without removing it from the CQ. 1 = Pop, returns the FD at the head of the FQ and removes it from the CQ, i.e. dequeues 1 frame. 2 = XSFDR read, allows reading the contents of any particular XSFDR. 3 = Reserved.
6-7 XSFDR	Bits 0-3: Reserved Bits 4-15: XSFDR pointer to be read (32 channel CEETM), valid only if CT = 2. Bits 6-15: XSFDR pointer to be read (8 channel CEETM), valid only if CT = 2.
8-63	Reserved

**Figure 3-168. CEETM CQ Peek/Pop/XSFDR Read Response Format****Table 3-162. CEETM CQ Peek/Pop/XSFDR Read Response Format**

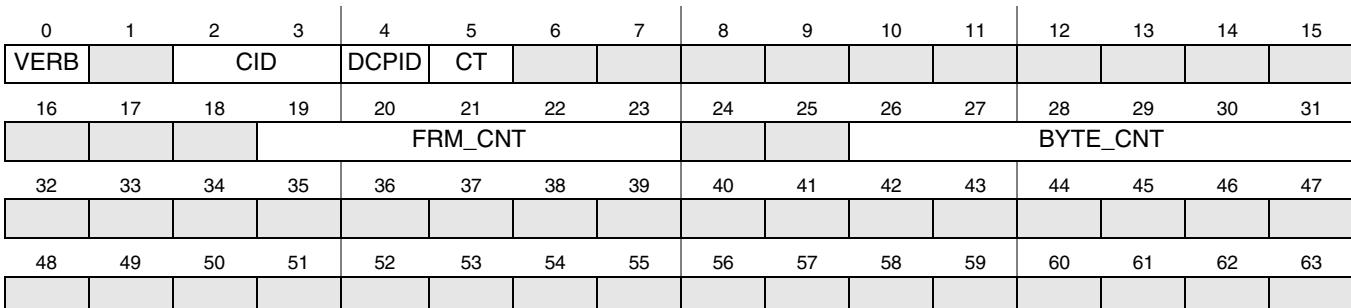
Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.

**Table 3-162. CEETM CQ Peek/Pop/XSFDR Read Response Format (continued)**

Byte Field	Description
2 STAT	Status. Valid only if CT was 0 or 1 in the command. Bits 0-4: Reserved. Bit 5: Retry required. 0 = No prefetch in progress, bits 6 and 7 indicate what is returned. 1 = Prefetch in progress, FD not yet available, the FD and DCTIDX fields in this response are invalid. Bits 6 and 7 are both 0 when this bit is 1. Software may re-issue this command sometime later, once the prefetch is complete a response with this bit set to 0 will be returned. Bit 6: CQ Empty. 1 = The CQ was empty after this command. Valid only for a pop command, i.e. if CT = 1 in the command. Bit 7: Number of frames delivered in this response. 0 = No frame delivered, the FD and DCTIDX fields in this response are invalid. 1 = One frame delivered, the FD and DCTIDX fields are valid.
3-13	Reserved
14-15 DCTIDX	Bits 0-3: Reserved. Bits 4-15: Dequeue Context Table Index that was stored with the FD.
16-31 FD	Frame descriptor (FD) See <a href="#">Section 3.3.1.2, “Frame Descriptors (FDs)”</a> for a description of the contents of a FD.
32-63	Reserved

### 3.3.9.7.21 CEETM Statistics Query/Write

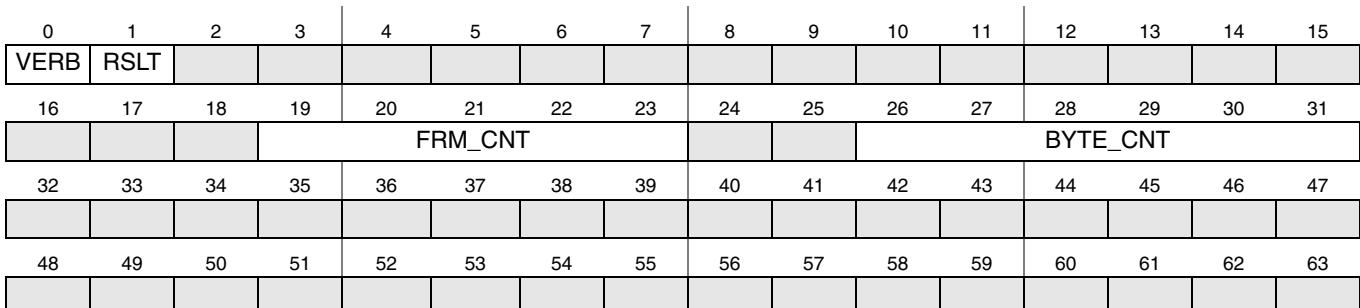
This command can be used to query (and optionally clear) the statistics counters provided with CEETM. A query command will leave the counters unchanged as a result of the query, a query and clear command will clear the counters to 0 after the query. The counters can also be written, although this is typically useful only for verification. After reset all counters are cleared to 0.

**Figure 3-169. CEETM Statistics Query/Write Command Format****Table 3-163. CEETM Statistics Query/Write Command Format**

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See <a href="#">Section 3.2.3.4, “QCSP Management Command Registers (QCSPi_CR).”</a> Bit 1-7 (lsbs): Command Verb. Specifies the command to be executed. CEETM Statistics Query/Write: Verb = 7Dh.
1	Reserved

**Table 3-163. CEETM Statistics Query/Write Command Format**

Byte Field	Description
2-3 CID	Bits 0-6: Reserved Bits 7-15: CQID or CCGRID targeted by this command, dependent on the value of CT. If CT = query or write dequeue statistics: Bits 7-15: CQID value of the CQ targeted by this command (32 channel CEETM). Bits 9-15: CQID value of the CQ targeted by this command (8 channel CEETM). If CT = query or write reject statistics: Bits 7-11: CCG channel to be written (32 channel CEETM). Bits 9-11: CCG channel to be written (8 channel CEETM). Note that this field corresponds to the msbits of CQID, which identify the class queue channel. See <a href="#">Section 3.3.20.11, “CEETM Class Congestion Management and Avoidance (CCM)”</a> . Bits 12-15: CCGID, identifies the CCG within the CCG channel.
4 DCPID	Bits 0-3: Reserved Bits 4-7: CEETM portal ID. In a SoC which contains multiple instances of CEETM logic (each connected to a different DCP portal), this field selects which portal's CEETM is addressed by this command.
5 CT	Bits 0-4: Reserved. Bits 5-7: Command Type 0 = Query dequeue statistics. CID carries the CQID to be queried. 1 = Query and clear dequeue statistics. CID carries the CQID to be queried 2 = Write dequeue statistics. CID carries the CQID to be written. 3 = Query reject statistics. CID carries the CCGRID to be queried. 4 = Query and clear reject statistics. CID carries the CCGRID to be queried 5 = Write reject statistics. CID carries the CCGRID to be written. 6,7 = Reserved. (FFh result code will be returned in the response)
6-18	Reserved
19-23 FRM_CNT	Frame Count. If CT indicates a write command type, the value in this field will be written to the statistics frame counter targeted by this command. For query command types this field is not used.
24-25	Reserved
26-31 BYTE_CNT	Byte Count. If CT indicates a write command type, the value in this field will be written to the statistics byte counter targeted by this command. For query command types this field is not used.
32-63	Reserved

**Figure 3-170. CEETM Statistics Query/Write Response Format**

**Table 3-164. CEETM Statistics Query/Write Response Format**

Byte Field	Description
0 VERB	Bit 0 (msb): RR identification bit. See <a href="#">Section 3.2.3.5, “QCSP Management Response Registers (QCSPi_RR0–1) Format.”</a> Bit 1-7 (lsbs): Response Verb. Indicates the command which initiated this response. The response verb carries the same value as the command verb.
1 RSLT	Result. F0h = OK, the command completed successfully. FFh = Error, unrecognized or invalid command was received.
2-18	Reserved
19-23 FRM_CNT	Frame Count. If CT indicates a query command type, the value of the statistics frame counter targeted by this command is returned in this field. This represents the number of frames that have been counted for this statistic since the last time the counter was cleared. If any statistics count reaches its maximum value, it will stick at that value until cleared or written. For dequeue statistics, this is a cumulative count of the number of frames dequeued from each class queue. For reject statistics, this is a cumulative count of all frame enqueues rejected in all CQs belonging to a particular congestion group, due to either WRED or tail drop. Note that enqueues rejected due to error are not counted.
24-25	Reserved
26-31 BYTE_CNT	Byte Count. If CT indicates a query command type, the value of the statistics byte counter targeted by this command is returned in this field. This represents the number of bytes in all frames that have been counted for this statistic since the last time the counter was cleared. If any statistics count reaches its maximum value, it will stick at that value until cleared or written. For dequeue statistics, this is a cumulative count of the number of bytes dequeued from each class queue. For reject statistics, this is a cumulative count of the number of bytes in all frames whose enqueue was rejected in all CQs belonging to a particular congestion group, due to either WRED or tail drop. Note that enqueues rejected due to error are not counted.
32-63	Reserved

### 3.3.10 Direct Connect Portals (DCPs)

Use direct connect portals (DCPs) to connect one or more hardware modules within the SoC directly to the QMan without going across the system interconnect of the SoC.

The QMan allocates the DCP interfaces as follows:

- One for the FMan
- One for the security engine (SEC)

The DCPs are numbered 0, 2, with DCP0 used by the FMan, DCP1 not used, and DCP2 used by SEC.

A DCP contains one or more sub-portals (SP) to which dequeue commands can be issued.

For DCP0 and DCP1, each SP is connected to a dedicated WQ channel (see [Section 3.3.2, “Work Queues \(WQs\) and Channels”](#)) within QMan, and uses a dedicated set of resources allowing a separate active FQ for each. Each SP in the portal (and thus each dedicated WQ channel) is intended to be associated with a separate transmit link or destination. The separate WQ channel per SP is provided to avoid head of line

blocking between the destinations, because the connected hardware block can issue dequeue commands to the separate sub portals independently.

DCP0 contains 16 SPs for use by the FMan, and DCP1 is not used.

For DCP2, a single WQ channel is serviced using 2 SPs. Each SP can hold an active FQ, allowing SEC to have several active FQ from the same channel open at the same time for dequeue.

Enqueue operations performed through a direct connect portal do not use QMan's order restoration feature.

### **3.3.11 Algorithmic Sequencers**

Algorithmic sequencers (AS) are used within QMan to execute the commands received across the various portals. The AS are divided into separate pools, and within each of these pools a command from any of the portals attached to that pool can be executed by any of the AS in the pool. The number of AS allocated to each pool and the portals attached to each pool as follows:

- Software Portals: 4 AS allocated
- Direct Connect Portal Enqueues: 4 AS allocated
- Direct Connect Portal Dequeues: 4 AS allocated
- FQ re-scheduling: 4 AS allocated

The use of separate pools of AS provides isolation between the functions performed using their attached portals, for example execution of commands from a DCP is not affected by commands issued by software. It also allows several commands to be issued in the same cycle. Another advantage is that the AS in each pool can be optimized to execute only the commands required by the portals which are serviced in that pool, making each AS smaller than it would be if all commands were supported by all AS.

#### **3.3.11.1 Portal Service Selection**

Each of the AS pools is fronted by logic which performs portal service selection, which consists of selecting the next portal containing a command waiting to be executed, and assigning that command to one of the AS in the pool when one becomes available.

The portal service selection algorithm used in each of the pools is a simple round robin among the portals in that pool, with no programmability.

#### **3.3.11.2 Multi-Way Resource Arbiter (MRA)**

The MRA is a central entity that provides an independent arbiter between each of the algorithmic sequencers (AS) and the portals and resource managers (SFDR Mgr, PFDR Mgr, WQ Sem Mgr, and so on).

Each algorithmic sequencer can make a request to any one of the resources at any time. The MRA will determine which algorithmic sequencer should gain access to any particular resource next. Priority is given to the algorithmic sequencer that currently has control of the resource (with the understanding that the algorithmic sequencer will not abuse this privilege, and will not continue making chained requests indefinitely). Selection is done in a work conserving round-robin fashion if the current requester no longer needs access to a resource.

### 3.3.11.3 Work Queue Semaphore and Context Manager

The WQSM provides a semaphore for the WQ so that any algorithmic sequencer knows it is accessing a WQ uniquely and indivisibly. Additionally, the WQSM holds the WQ context (WQ head, WQ tail and length) of each WQ in a 4KB single-port internal SRAM.

## 3.3.12 Frame Queue Descriptor Cache

QMan contains an internal memory used to store local copies of frame queue descriptors (FQD) for fast access and improved performance, this memory is called the FQD Cache. The FQD cache is sized to hold up to 512 FQDs.

The FQD semaphore and cache manager (FQDSC) block within QMan performs two main functions, it manages indivisible access to FQDs (FQD semaphore management) and manages the allocation, querying, and retirement of entries in the FQD cache (FQD cache management). These two functions are collocated in the same resource block because of the tight coupling between them.

All algorithmic sequencers (AS) must interact with FQDSC before and after accessing any FQD. A semaphore lock must be obtained before accessing any FQD, and this lock must be released after the required operations on the FQD are complete and it has been written back either to the FQD cache or to main memory. Before accessing a FQD, the FQD cache must be queried to determine where the most recent copy of the FQD is located, and a new FQD cache entry may be allocated for this FQD or its existing entry in the cache may be retired. All accesses to a FQD, either in FQD cache or in main memory, must be done while that FQD is under semaphore lock.

### 3.3.12.1 FQD cache operation summary

A FQ may be placed in the FQD cache when in the Parked, Truly Scheduled, or Tentatively Scheduled states, whether it is empty or not (See [Section 3.3.1.5, “Frame Queue State”](#) for a description of the states of a FQ). A FQ that is in any other state will not be placed in the FQD cache.

#### 3.3.12.1.1 On-demand FQD cache eviction

This means that any FQ, once initialized, may be placed in the cache. If a large number of FQs have been initialized they may fill up the cache, thus an on-demand eviction policy is used to allow new FQs to be inserted into FQD cache by displacing FQs that are in cache but are currently empty. FQs that are in FQD cache but that are empty (with the exception of FQs configured with `FQ_CTRL[Lock_in_Cache] = 1`, and FQs whose Force Eligible (FE) bit is asserted) are marked as evictable in the cache, and only FQD cache entries in the evictable state are eligible for on-demand eviction.

Whenever the state of a cache entry is set to evictable, the FQD is written to both the FQD cache and to main memory, this avoids having to flush the FQD from cache to main memory if eviction occurs later.

A FQD cache entry in the evictable state remains valid and useable in the cache, but may be evicted when another FQ attempts to go into cache, i.e. if there are no free entries, but there are one or more evictable entries available when an attempt is made to place another FQ into cache. If there are no free entries and more than one evictable entry, then a pseudo Least Recently Used (pseudo-LRU) algorithm, modelled after that used by the e500mc cache, is used to select one of the evictable entries for eviction.

A key requirement is that a FQ cannot be evicted while it is in use by one of the algorithmic sequencers, whether or not it is empty at that time. This requirement is met by ensuring that the state of a cache entry is never evitable while the FQ stored in that entry is in use by a sequencer.

### 3.3.12.1.2 Post-dequeue FQD cache eviction and Lock in Cache

Once a FQ has been placed in the FQD cache, it can only be removed either by on-demand eviction as described above, or when it transitions to either the Active or Retired state. A FQ that is retired is on its way to becoming out of service, therefore its place in cache is released for use by another FQ. When a FQ reaches the Active state, a FQ that is in cache has its FQD moved from the FQD cache to a PSCL (See [Section 3.3.8.2.8, “Active and Suspended Frame Queues”](#)), this is also referred to as post-dequeue eviction from FQD cache.

What happens to a FQD cache entry after a post-dequeue eviction depends on the setting of the FQ\_CTRL[Lock\_in\_Cache] bit (see [Section 3.3.1.4.3, “Structure of a Frame Queue Descriptor \(FQD\)”](#)) in the FQD that was stored in that entry. If FQ\_CTRL[Lock\_in\_Cache] = 0, then the FQD cache entry that was occupied by the now Active FQ is free to be used by another FQ. However if FQ\_CTRL[Lock\_in\_Cache] = 1, the FQD cache entry that was occupied by the now Active FQ is reserved for this FQ, and cannot be used by any other FQ.

Once all dequeues from a FQ are complete (specifically, when the FQ is rescheduled), QMan will attempt to place the FQ into FQD cache, if it is in one of the 3 allowed cacheable states. If the FQ is configured with FQ\_CTRL[Lock\_in\_Cache] = 0, or if it was not in cache before it was moved to the Active state for dequeue, then the attempt to place this FQ in cache may not succeed if the cache is full or almost full. However if the FQ is configured with FQ\_CTRL[Lock\_in\_Cache] = 1, and it was in cache before it was moved to the Active state for dequeue, then QMan will place it in the entry that was reserved for it when it was initially moved to Active. The reservation guarantees that this FQ will return to FQD cache after dequeue.

## 3.3.13 Order Restoration

Order Restoration is a function within QMan which restores the relative temporal order of a flow of frames (sequence of frames) to that observed and recorded at a logical “point” earlier in the datapath (within QMan or elsewhere) referred to as an Order Definition Point (ODP).

The term Order Definition Point has been so named because of the geometric connotations of the term “point.” As in geometry where lines or paths between two spaces may pass through different “points” on a plane that separates the two spaces, different groups of frames may pass through different ODPs as control of the frame passes from one processing stage to the next. Flows of frames that belong together and need to be kept relatively ordered should be directed through the same ODP.

The order restoration algorithm can be applied to multiple flows (that passed through different ODPs) independently and concurrently such that the remedies applied (that is, delays) to mis-ordered frames within one flow do not impact frames of unrelated flows. The OR algorithm is applied at logical points within the datapath referred to as Order Restoration Points (ORPs).

The order restoration mechanism built into QMan’s software portals allows multiple frames from the same source frame queue to be processed in parallel on 2 or more processor cores, while providing the ability to

restore these frames back to the order in which they appeared in the source frame queue before they are enqueued onto their destination frame queue(s).

When one or more frames that are awaiting enqueue need to be held pending the arrival of one or more other frames with a lower sequence number, the FD and some context for the held frames is stored in an Order Restoration List (ORL).

QMan contains an internal SRAM dedicated to the storage of 256 ORL records, and when the internal ORL records are exhausted the ORL lists can extend into main memory. These internal storage records are shared for use as both ORL records and as Enqueue Rejection Notification (ERN) list records. When extended to main memory, ORL records are stored in the same area of memory reserved for PFDR (see [Section 3.2.4.43, “Data Structure Base Address Registers](#)), PFDR are allocated from the PFDR manager as needed and then used for the purpose of storing the ORL records and building the ORL lists.

### **3.3.13.1 Position of Order Definition Points within QMan**

Since frames of an ordered flow must pass through an ODP before they are allowed to diverge through parallel processing elements, the simple intuitive place for an ODP would be at the receive interface. However, since the order restoration involves holding (delaying) frames that complete processing relatively early, having ODPs that are too coarse (such as one ODP for an entire Rx link) may have non-optimal consequences. For instance if frames are to be classified into different flows and queued at different priorities to the processing elements, the effect of that priority queueing may be negated by the delay that is required to restore order at the corresponding ORP. A more subtle consequence, increased average system latency, can appear even when frames are classified into flows of equal priority. This increased latency is a result of the times at which a frame is delayed unnecessarily to restore relative order with another frame despite the fact that there is no ordering requirements between the two frames. In order to facilitate “fine-grained” flows QMan fully implements the function of an ODP at the head of each FQ that is consumed via a software portal.

### **3.3.13.2 Position of Order Restoration Points within QMan**

The QMan fully implements the function of ORPs lying in a logical plane between the Enqueue Command Ring (EQCR) of the software portals and targeted Frame Queues (order restoration is not supported for Direct Connect Portals). When issuing an enqueue command software can indicate to QMan that the actual enqueue to the targeted FQ must be performed in order relative to other frames of the same flow. Software does so by indicating within the enqueue command that the frame must pass through the specified ORP that is unique to that flow and specifies the relative sequence number of that frame that was assigned at the ODP. Each ORP is associated with one ODP. It is important to note that there is no association between an ORP and any destination FQ.

Note that the design choice of implementing QMan ORP functionality on this plane (as described above) is because typically the FQs targeted by enqueue commands from software are the per-priority/per-link output queues that buffer the transmit ready frames. If the function of an ORP was performed after rather than before entering these queues the QOS benefit of priority-based queueing would be negated and head of line blocking would occur between links resulting in unused transmit opportunities (bandwidth) when in fact there was traffic waiting to be transmitted. Although with this design choice the relative order

between frames in different link/priority queues is not preserved, this is clearly allowable, desirable and intentional.

It is important to restate that an ORP is not associated with a destination FQ. To illustrate that fact, it can be stated that frames going through an ORP may be enqueued on different destination queues (as would be the case for IP forwarding), and frames going into the same destination queue did not necessarily pass through the same ORP.

### **3.3.13.3 Special Cases for Order Restoration**

The order restoration algorithm accommodates special treatment that may have to be given to individual frames within the flow as follows.

If an individual frame within a flow does not need to be ordered relative to the rest of the flow it can be exempted from any remedy action (delay) but it must still pass through the ORP with its sequence number.

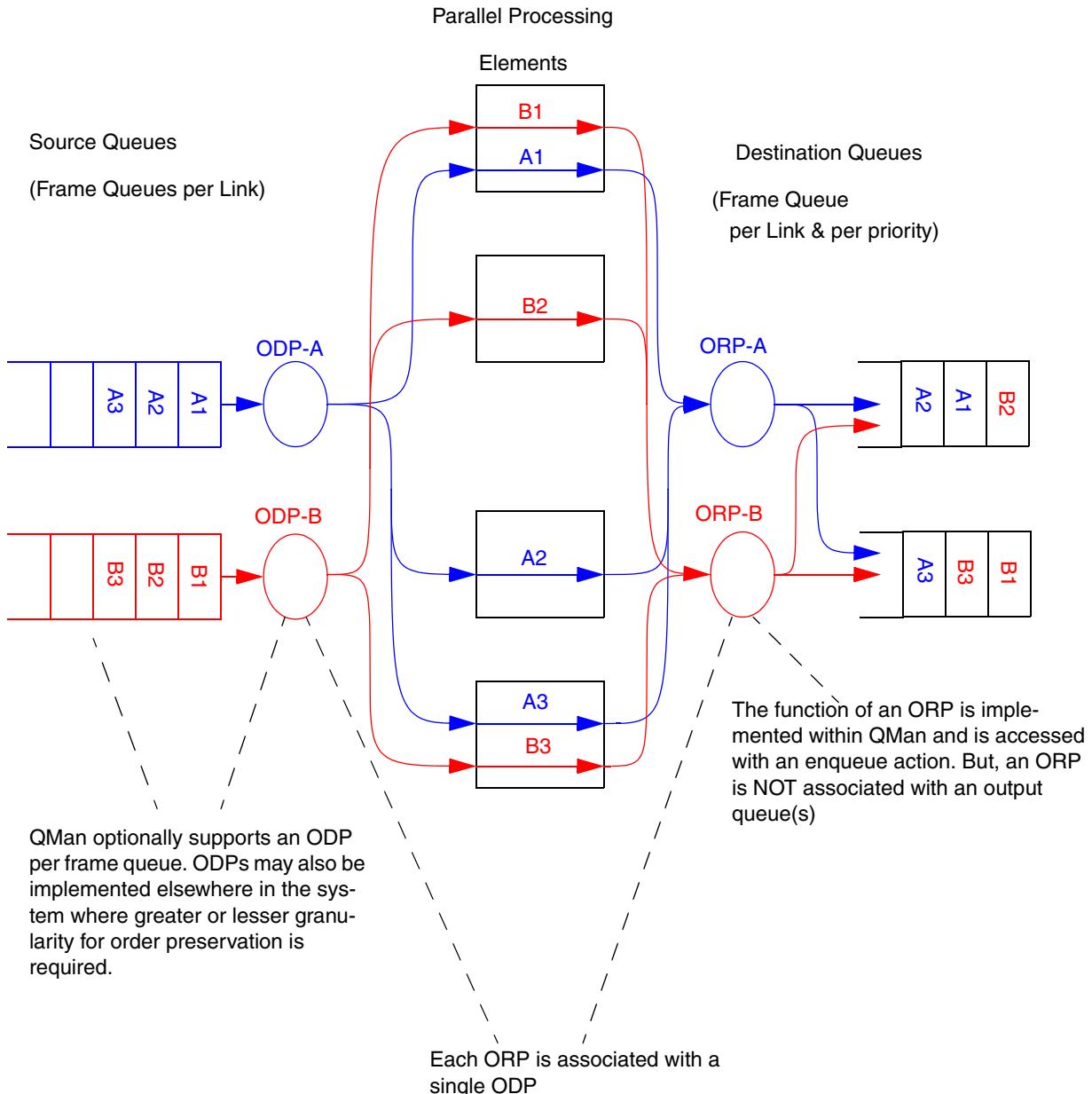
If an individual frame within a flow is being removed from a flow an indication of removal (in lieu of the frame) can be passed to the ORP with the sequence number of the removed frame. Removal of a specific frame from an ordered flow may be as a result of discard, redirection, or excessive processing to the degree that it is not desirable to delay the other frames of the flow.

If processing of an individual frame spawns or is replaced by a group of adjacent frames (as would be the case when an IP Packet is fragmented), the frames of the group may be passed through the ORP with an identical sequence number. All but the last frame of the group must indicate that one or more frames with the same sequence number are still to come. The OR algorithm shall restore order of the group relative to other frames of the flow. However, software is responsible for ensuring that the frames within the group (which all carry the same sequence number) reach the ORP in order and as such the frames of the group are best to be enqueued via the same portal.

Of course, when an entire flow of frames does not require order or when order has been preserved through the system by some other means enqueues can be issued without specifying an ORP.

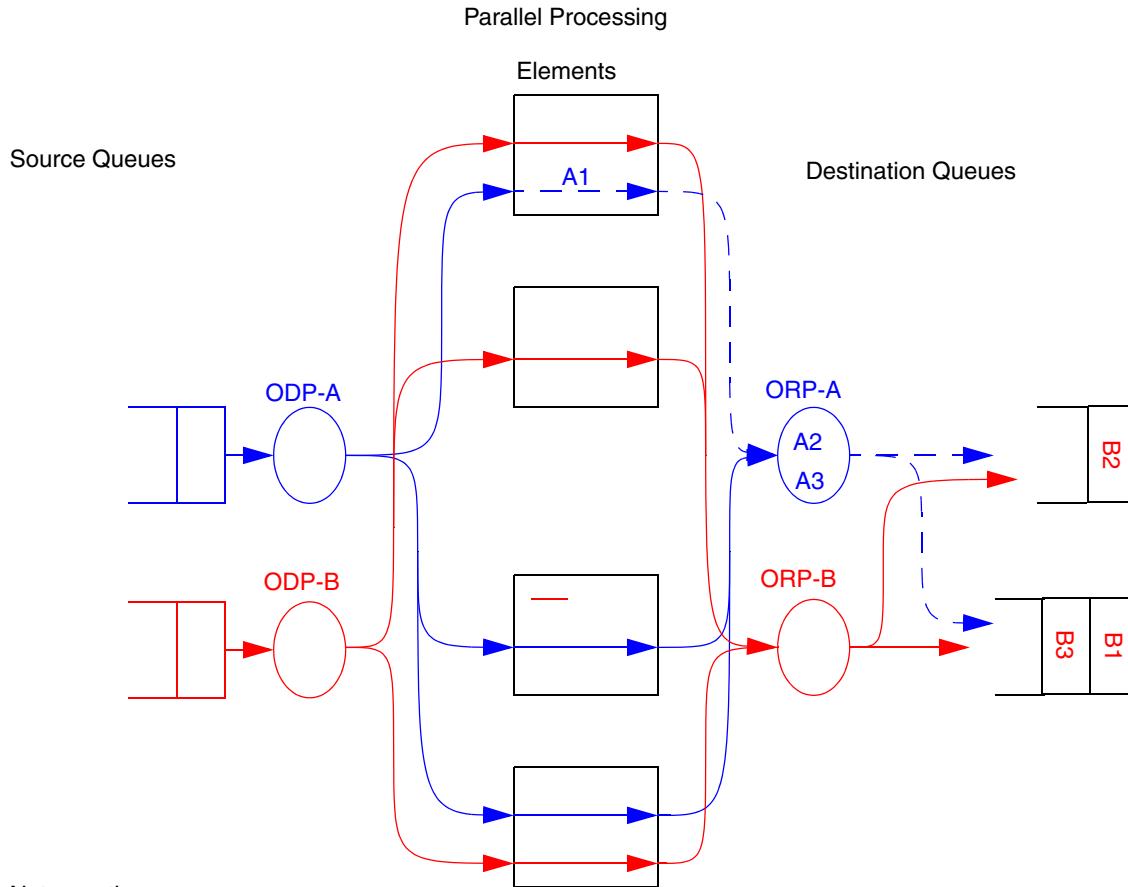
**Elapsed Time View of Packet Processing**

(Source, Processing Path &amp; Destination)

**Figure 3-171. Order Restoration: Elapsed Time View of Packet Processing**

Interim Point in Time (Same scenario as previous page)

(Processing of frame A1 is relatively slow. Frames A2 &amp; A3 are early arrivals at ORP-A)

**Figure 3-172. Order Restoration: Elapsed Time View of Packet Processing (continued)**

### 3.3.13.4 Order Definition Point Implementation

A sequence number field (ODP\_SEQ, 14 bits) is included in the Frame Queue Descriptor to support the ODP; see [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\)](#). As frame descriptors (FDs) are dequeued from a FQ, the sequence number within the FQD is incremented by 1 for each frame passed. A consequence of this implementation is that the sequence number does not have to be placed within the frame in order to convey it to the consumer, it is instead provided alongside the FD in the dequeue response in a software portal; see [Section 3.3.9.2, “Frame Dequeue Response](#). If there is no requirement to have an ODP at the head of a particular FQ, the consumer can simply ignore the sequence number.

### 3.3.13.5 Order Restoration Point Implementation

The function of an ORP is implemented by a mechanism within QMan. The context/state of an ORP is held in a ORP Descriptor and possibly an attached ordered list of deferred enqueue command descriptions (when the ORP currently is holding “early arrivals”).

#### 3.3.13.5.1 Order Restoration Point (ORP) Descriptor

Space is reserved in the 64-byte FQD for an ORP Descriptor to co-reside with a Frame Queue Descriptor; see [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\)”](#). When an ODP is implemented using the ODP resources provided by QMan, that is, at the head of a FQ consumed by software portals (using the sequence numbers in the dequeue responses), then the associated ORP can be co-resident within the FQD of that FQ, meaning that the ORP used when enqueueing a frame (see [Section 3.3.9.1, “Enqueue Command”](#)) can be the same as the FQID from which the frame was dequeued. However such a co-resident configuration is not recommended, as explained below.

When an ODP is implemented elsewhere in the system (for example, using sequence numbers provided by software), then the associated ORP should not be co-resident within the FQD of any FQ in use in the system, it should instead reside within its own separate FQD, with the co-resident FQ initialized in the Parked state and the remaining Frame Queue Descriptor fields unused. Furthermore, having an ORP co-resident within the FQD of a FQ which is in use can lead to reduced performance in certain cases because of contention for access to the FQD, therefore it is generally recommended to always implement an ORP in its own separate FQD in the Parked state. Note that since FQID 0 is reserved, its FQD should not be used to hold ORP context either, therefore an ORP index of 0 is reserved as well.

The essential context value always maintained for an ORP is the “next expected sequence number” (NESN). All enqueue commands that respect an order restoration point (that logically pass through that ORP) must be coupled with a sequence number and are judged by comparing that sequence number to the next expected sequence number.

Additional fields within the ORP Descriptor are used to maintain an ordered list of the deferred enqueue commands associated with the “early arrivals.”

#### 3.3.13.5.2 Using Sequence Numbers

All sequence numbers are in modulo  $2^{14}$  space. This means that all sequence numbers are 14 bits (values of 0 to 16383). Furthermore, all comparisons of sequence numbers are made using modulo arithmetic, with NESN at the center of the number line. This means any number that can be reached by adding  $8191$  ( $2^{13}-1$ ) to NESN is considered greater than NESN. All other numbers are considered less than. Any statements of greater-than or less-than in the discussion of sequence numbers should be assumed to be in this modulo sense of the terms.

Any frame arriving at an ORP with a sequence number that is greater than NESN shall be considered an “early arrival”. Any frame arriving at an ORP with a sequence number that is less than NESN shall be considered a “late arrival”.

Early arrivals may occur because some of the frames are processed (by software) in less time than others. Late arrivals occur when the NESN is advanced despite the fact that the next expected frame(s) have not been delivered for enqueue. Note that if the design were to accommodate only strict ordering and only

advanced NESN when the expected frame arrived, there would be no such thing as a late arrival. However, the design does support the concept of unblocking of held early arrivals due to exceptional advances of NESN.

An exceptional advance of NESN may occur as a result of:

- The skew (difference in sequence numbers between NESN and that of the held-frame that is most early) exceeds an ORP specific threshold, at which point QMan decides to give-up on some of the missing frames and autonomously advances the NESN to bring the skew within threshold.
- Software issues a command to unblock all held frames earlier than a given sequence number and assign that value to NESN. This is done using an Enqueue Command with VERB bits 6-7 = 2, and with the msb of the SEQNUM field asserted; see [Section 3.3.9.1, “Enqueue Command.”](#)

In both cases, an exceptional advance of NESN will cause any deferred enqueues that are currently held, but whose sequence number is less than the new NESN value, to become unblocked (or bumped) and immediately enqueued to their destination FQ. Note that these unblocked enqueues will be processed in order relative to each other, but that order is no longer maintained between these unblocked enqueues and any enqueues whose sequence number is equal to or greater than the new NESN value, that is, between unblocked enqueues that are now outside of the Restoration Window and enqueues which are still within the Restoration Window (see description of Restoration Window below). Also note that any enqueues not yet arrived at the ORP will be treated as late arrivals if/when they do arrive, that is, either rejected or enqueued immediately depending on ORP configuration (see Acceptable Late Arrival Window described below).

Note: It is expected that the sequence numbers used in order restoration operations, on a per ORP basis, will be unique (excluding of course rollover of the 14 bit sequence number, which must eventually occur) unless a frame is to be fragmented by software. If a frame is fragmented by software, the fragments must all have the same sequence number, must be enqueued in order to the same ORP and same FQID, with all fragments except the last having NLIS set to 1, and with the last fragment having NLIS set to 0. Fragment mis-ordering (for example a fragment which is not the last one having NLIS set to 0) may cause unpredictable behavior of the affected ORP and must be avoided.

### **3.3.13.5.3 ORP Behavior in Extreme Conditions**

The OR algorithm can in theory accommodate and correct an infinite amount of mis-order skew. However, from an implementation perspective and network behavior perspective that is neither practical nor desirable.

As such each ORP as implemented in QMan can be configured with the range of conditions under which it is expected to fully restore order and with treatment of frames to be applied when those conditions are exceeded. The configuration takes the form of a set of non-overlapping windows within the sequence number space. The size of these windows is resident in the FQD and is configured by software; see [Section 3.3.9.5.1, “Initialize Frame Queues \(FQ\).”](#) The windows, once configured, are fixed in size but slide through the sequence number space as the value of NESN advances. The treatment of a frame enqueued via an ORP is determined by which window the sequence number of that frame falls within.

The windows are:

- Restoration Window

A frame enqueued with a sequence number within this window is held on the list as a deferred enqueue command until it is the next expected. The size of this window places a limit on the amount of mis-order skew that an ORP will attempt to correct (The skew is the difference between NESN and the sequence number of the deferred enqueue at the tail of the ORP ordered list). The size of this window indirectly places an upper bound on the number of entries that may be in the ordered list held at the ORP. The window is immediately to the right of (numerically after) NESN and can range in size from 32 to 4096 in power of 2 increments. These sizes include the NESN position which technically is not part of the Restoration Window so to be precise the window is 1 unit smaller than these sizes.

- Auto Advance NESN Window (Bump Window)

A frame enqueue with a sequence number within this window is held on the list as a deferred enqueue command similar to the case within the Restoration Window. In doing so it makes the skew greater than the Restoration Window so NESN is automatically advanced such that the Restoration Window includes this latest arrival. (the Restoration window slides to the right.) This constitutes an exceptional advance of NESN as described in [Section 3.3.13.5.2, “Using Sequence Numbers.”](#) and may result in one or more deferred enqueue commands being unblocked (bumped) and executed because they are no longer within the Restoration Window. This window is immediately to the right of (numerically after) the Restoration Window and can be equal in size to the Restoration Window or null (bump feature disabled).

- Acceptable Late Arrival Window

A frame enqueue with a sequence number within the Acceptable Late Arrival Window will be processed immediately. The state of the ORP is not affected at all. This window is immediately to the left of (numerically before) NESN. The size may be Null (all late arrivals rejected), 32 units, equal to the size of the Restoration Window, or 8K (accept all late arrivals).

- Late Arrival Rejection Window

A frame enqueue with a sequence number within the Late Arrival Rejection Window will produce an Enqueue Rejection Notification with an appropriate Rejection Code; see [Section 3.3.9.3, “ERN Message Response.”](#) This window is immediately to the left of (numerically before) the Acceptable Late Arrival Window. The size of this window is determined indirectly by the size of the Acceptable Late Arrival Window. It will be 0 when the Acceptable Late Arrival Window is configured to be 8K.

- Early Arrival Rejection Window

A frame enqueue with a sequence number within the Early Arrival Rejection Window will produce an Enqueue Rejection Notification with an appropriate Rejection Code; see [Section 3.3.9.3, “ERN Message Response.”](#) This window is immediately to the right of (numerically after) the Auto Advance NESN Window. The size is determined indirectly by the sizes of the Restoration Window and the Auto Advance NESN Window. It will be 0 when Restoration Window and Auto Advance NESN Window are configured to be 4K each.

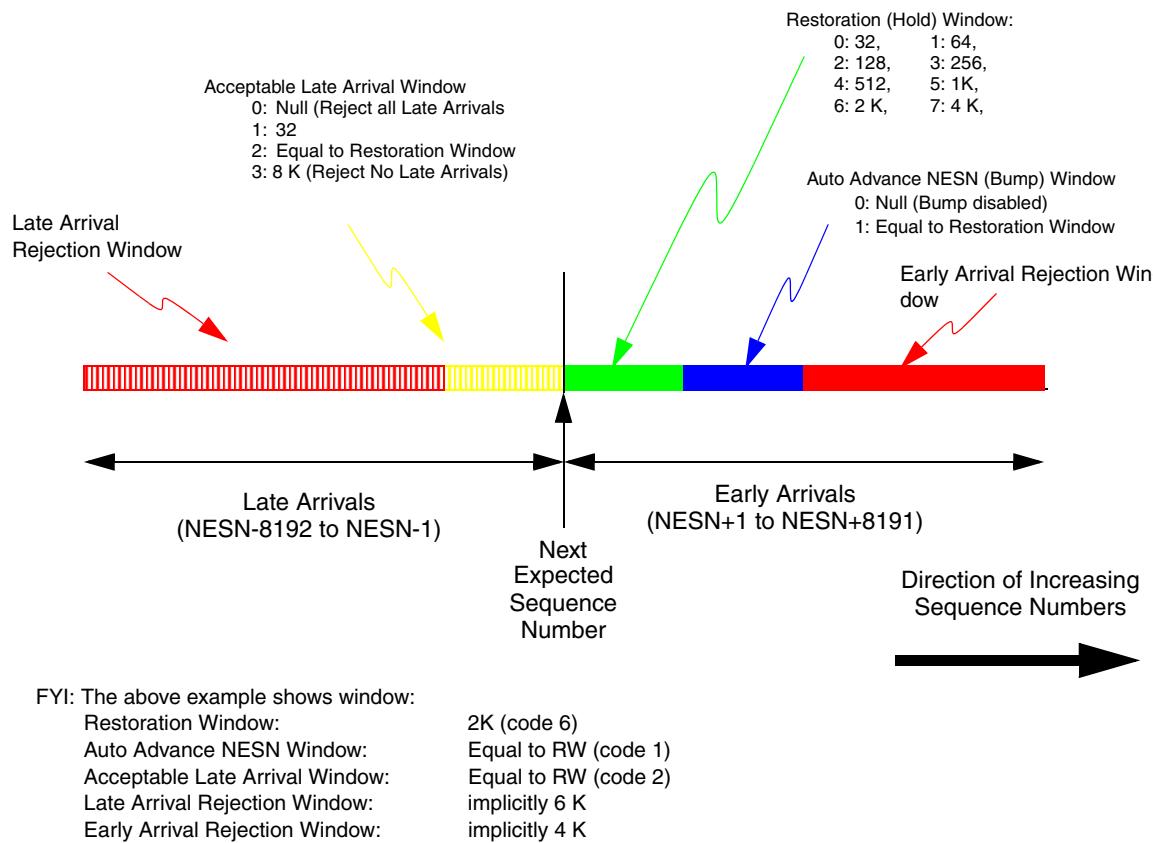


Figure 3-173. Order Restoration Sequence Number Windows

### 3.3.14 Congestion Management and Avoidance

Congestion management and avoidance in QMan supports three distinct mechanisms, the first is WRED and the second and third are two forms of tail drop, Congestion State (CS) Tail Drop and FQ Tail Drop.

WRED and CS tail drop are managed using a QMan private data structure called a Congestion Group Record (CGR; see [Section 3.3.14.7, “Congestion Group Record \(CGR\)”](#)). QMan supports a total of 256CGR which are stored in an internal SRAM, this is a performance benefit since operating with congestion management enabled requires a read-modify-write of the CGR for each frame enqueued and dequeued. FQs can be assigned to a congestion group when they are initialized. Each congestion group can be configured to track either byte counts or frame counts, that is, the number of bytes or frames in all FQ in the Congestion Group.

FQ tail drop is managed using a tail drop threshold stored in each FQD; see [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\)”](#). This threshold is configured by software when the FQ is initialized.

Each FQ can be configured by software to add or subtract a fixed number of bytes to/from the actual length of each frame enqueued and dequeued from that FQ, when calculating the instantaneous and average byte counts of the congestion group to which that FQ belongs. This can be used to account for a fixed per-frame overhead, such as media transmission overhead, when using WRED and/or CS tail drop for congestion

management and avoidance. The Overhead Accounting Control (OAC) and Overhead Accounting Length (OAL) fields in the FQD are used to configure this behavior.

Note that QMan does not perform the function of actually discarding a frame. QMan will accept a frame to be enqueued, determine whether it is going to be enqueued based on WRED or tail drop criteria and if it is not going to be enqueued it will be returned to the source via an enqueue rejection notification.

As a performance optimization, the congestion manager in the QMan can accept up to 3 enqueue and/or 3 dequeue frames in a single operation.

### 3.3.14.1 Random Early Discard (RED)

Random Early Discard (RED) is an active queue management algorithm which drops arriving packets with linearly increasing probability. The probability of drop increases as the average queue size grows. Note that RED responds to a time-averaged queue length, not instantaneous queue size. Since packets are dropped before the queues fill up, this gives flows such as TCP connections the opportunity to slow down the sending rate before the queues get full.

The main advantage of RED, over traditional techniques such as tail drop, is the avoidance of the global synchronization issue where multiple flows are throttling back followed by a sustained period of lowered link utilization, thus causing queue oscillation.

### 3.3.14.2 Weighted Random Early Discard (WRED)

Weighted Random Early Discard (WRED) fine-tunes the RED mechanism by specifying a RED policy for each packet drop priority or color. During congestion situations, lower priority traffic is slowed earlier and more aggressively, while high-priority traffic receives preferential treatment.

Functionally in the QMan, if a frame is to be added to a frame queue associated with a congestion group where WRED is enabled (via WR\_EN\_\* in the CGR), then before the frame is accepted, the average byte count (or average frame count if the congestion group has been configured in frame count mode) of the entire group is compared to two thresholds. If the average byte count is below minTH, no discards occur. Note that minTH is not a configured value. minTH is implied based on the configured values of MaxTH, Slope and MaxP stored in the WR\_PARM\_\* fields of the CGR. Between minTH and MaxTH there is an increasing chance of discard, up to a maximum discard probability of MaxP at MaxTH. Above MaxTH, all traffic is discarded. [Figure 3-174](#) shows this graphically. Each of the three frame colors are treated differently as far as WRED is concerned. Green, Yellow and Red frames each have a different WRED curve. The three curves can be configured to have different values of MaxTH, Slope and MaxP allowing for very complex congestion relationships for different types of policed traffic.

The first step in configuring a WRED curve, is choosing a MaxTH level. When the average byte count is above this level, all packets are discarded. MaxTH is comprised of MA and Mn, where  $\text{MaxTH} = \text{MA} * 2^{\text{Mn}}$ . The chosen value of MaxTH must be approximated by selecting different MA and Mn. For example, if MaxTH=10000 is desired; MA=78, Mn=7 results in MaxTH=9984. Alternatively, MA=39, Mn=8 also results in the same MaxTH. The next step is choosing a minTH level. When the average byte count is above this threshold, packets MAY be discarded. This value is not programmed, but is used in calculating the Slope. The third step, is choosing the value of MaxP. This value can be programmed to be 256 (representing 256/256 or 100% max probability) down to 4 (representing 4/256, or 1.56% max probability)

in steps of 4 (the MaxP value is set by programming the Pn value, where  $\text{MaxP}=[\text{Pn}+1]*4$ ). The MaxP value only affects the probability of discard when the average byte count is between minTH and MaxTH, and this probability decreases from MaxP down to 0 according to the chosen Slope value. The Slope represents the fractional drop in probability per byte down from MaxTh and 1/Slope represents the number of bytes per 1/256 probability level. For example, if the Slope value is configured to 0.1 (1/Slope=10) when MaxP=32 and MaxTH=9984, then from average byte count of 9975-9984, an incoming packet would have a 32/256% chance of being discarded, from 9965-9974 the probability would drop to 31/256%, from 9955-9964 the probability would be 30/256%, etc until the average byte count drops to 9664 or less when there is 0% chance of discard (the implied minTH value is 9664 in this case). The value of Slope must be calculated, not chosen, based on the three values; MaxTH, minTH and MaxP. Slope is calculated as follows;  $\text{MaxP}/(\text{MaxTH}-\text{minTH})$ . Now SA and Sn must be chosen to approximate this exact value with the condition that SA must be in the range of 64-127, and Sn must be in the range of 7-63. For example, if MaxTH=15872 (MA=62,Mn=8), minTH=7500, and MaxP=128 (50% max probability at MaxTH). The exact value of slope should be  $128/(15872-7500)=0.0152891$ . With the condition on SA, this fractional value can be approximated with SA=64, Sn=12 (to get  $64*2^{-12}=0.015625$ ) or for an even closer value SA=125, Sn=13 can be chosen (to get  $125*2^{-13}=0.015259$ ). With this approximated slope, working backwards, we can find the actual implied value of minTH is  $[\text{MaxTh}-\text{MaxP}/\text{Slope}]=7483.51$ .

The MaxTH value must always be less than  $2^{39}$ . Also, setting  $\text{Slope}>=1$  must be avoided. Undesirable behavior could result if any of these conditions occurs.

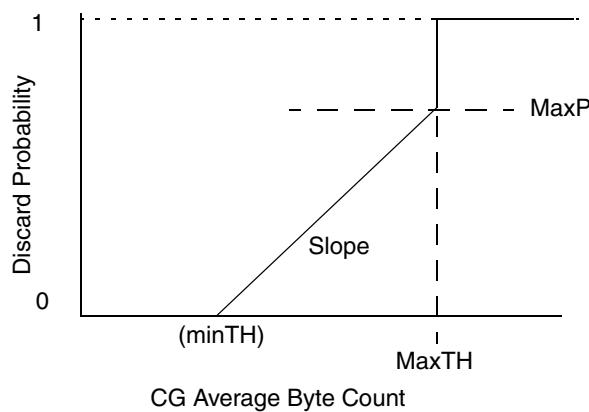


Figure 3-174. WRED Curve

### 3.3.14.3 Congestion Group Tail Drop

The Congestion Group Tail Drop algorithm is very simple when compared to the WRED algorithm. There are two different ways of causing a congestion group tail drop. If both the CS and the CSTD\_EN bits in the CGR are set, the incoming frame is marked for discard. If the CS bit is not set, and the CSTD\_EN bit is set, the frame is accepted. For details on how the CS bit is set and cleared; see [3.3.14.6, “Congestion State Change Notifications \(CSCN\)](#).” Additionally, the current instantaneous byte count (or frame count if the congestion group has been configured in frame count mode) is always compared to 0x7F80000000. If the I\_BCNT is less than or equal to this fixed value, the frame is accepted. If I\_BCNT is greater than this fixed value, the frame will be marked for discard (this discard occurs even for override-colored frames). This is done to avoid a byte count overflow in the congestion group. Note that the size of the incoming

frame is not included in the tail drop decision, only the CS and I\_BCNT values, before acceptance of the incoming frame, are used to make the tail drop decision.

While it is possible to have both WRED and CSTD\_EN enabled at the same time, it is unlikely that this would be useful since the WRED algorithm effectively implements a tail drop above MaxTH. CSTD\_EN is more likely to be used in situations where WRED would be considered as overkill, for management flows or non-traffic flows for example.

### 3.3.14.4 FQ Tail Drop

The Frame Queue Tail Drop (FQTD) algorithm is a second tail drop algorithm that is used on a per frame queue basis (independent of whether the frame queue has an associated congestion group). Much like the congestion group tail drop, there are two different ways of causing a frame queue tail drop. If Tail\_Drop\_Enable is set in the frame queue descriptor, then the algorithm compares the actual byte count of the frame queue to a threshold programmed within the frame queue itself ( $TD\_MANT * 2^{TD\_EXP}$ ). If this threshold is exceeded, then the frame is marked for discard. Otherwise, the frame is accepted. Additionally, the actual byte count is always compared to the fixed value 0xE0000000. If this value is exceeded, then the frame is marked for discard (this discard occurs even for override-colored frames). Otherwise, the frame is accepted. This is done to avoid a byte count overflow in the FQD. Note that the size of the incoming frame is not included in the tail drop decision, only the actual byte count of the FQ, before acceptance of the incoming frame, is used to make the tail drop decision. Therefore the maximum number of bytes that may be in the FQ before discards occur is threshold + maximum frame size.

As with CSTD, it is possible to have FQTD enabled at the same time as WRED, although the usefulness of this mode of operation could be debated. A properly configured WRED system will provide statistically driven early discards so that no one frame queue can monopolize the link.

If FQTD is used, it will likely be in conjunction with CSTD, such that no one frame queue can exceed a programmed threshold, and at the same time, groups of frame queues cannot monopolize the link either.

### 3.3.14.5 Enqueue Rejections

When an enqueue command received by QMan is to be rejected due to WRED or tail drop, a notification of the occurrence of this rejection must be sent. The receiver of this rejection notification is expected to discard the frame that was not enqueued and release its buffers.

For software portals, notifications of enqueue rejections are sent from QMan using a dedicated queue, which sits in parallel with the FQ WQs and contains enqueue rejection notifications (ERN) to be processed by the software servicing the portal. There is one ERN queue for each of the software portals. When the ERN queue is non-empty, ERN messages are removed from the queue and placed in the MR (see [Section 3.3.8.3, “Message Ring \(MR\)”](#)). Software can then retrieve the ERN messages from the MR and process them.

The ERN queues (1 per software portal) are maintained using a combination of internal resources (records stored in internal memory in the OLM) and external storage. The queues are built using internal records whenever possible, and will extend into main memory as needed. The internal storage records are shared for use as both Order Restoration List (ORL) records and as ERN list records. When extended to main memory, ERN list records are stored in the same area of memory reserved for PFDR (see [Section 3.2.4.43](#),

“[Data Structure Base Address Registers](#)”), PFDR are allocated from the PFDR manager as needed and then used for the purpose of storing the ERN records and building the ERN queues. Each ERN record is 64 bytes in size (see [Section 3.3.9.3, “ERN Message Response”](#)), and each ERN queue is maintained as linked list.

On a direct connect portal (DCP), notifications of enqueue rejections can be sent either to one of the software portals via its MR, or back to the hardware block using the DCP via a dedicated set of interface signals. The destination of ERN messages for each DCP is programmable; see [Section 3.2.4.11, “DCP Configuration Registers \(DCPi\\_CFG\)”](#). If sent to software, the ERN messages from a DCP use the same ERN queue and the same format as that used by the ERN messages generated by enqueue commands from the software portal itself. If sent back over the DCP, the ERN notifications sent on the dedicated link do not use an ERN queue as do the software portals. Instead, an ERN being sent on a direct connect portal is generated immediately when the enqueue command is processed and queued for delivery to the connected hardware block in a small FIFO at the interface.

Every enqueue command contains a 32 bit tag which is returned in the ERN for all rejected enqueues. The tag is intended to be used to associate the ERN with the enqueue command that contained the frame enqueue that was rejected. For example, the tag could contain the address of a callback routine in an upper layer of software.

### **3.3.14.6 Congestion State Change Notifications (CSCN)**

In addition to accepting or rejecting enqueues based on congestion, QMan is able to notify certain producers when a congestion group’s instantaneous byte count (or frame count if the congestion group has been configured in frame count mode) exceeds the CS threshold. When this threshold is exceeded, the CS bit is set in the CGR, and the congestion group is said to have entered congestion. When the group’s I\_BCNT returns below the threshold (minus approximately 1/8 of the threshold to provide hysteresis), the CS bit is cleared, and the congestion group’s state exits congestion.

On a direct connect portal, congestion state change notifications (CSCN) are sent from QMan using a dedicated set of interface signals.

Typically, only the original producer of data needs to receive congestion notifications, intermediate producers such as the Security Engine will not flow control or discard based on FQ congestion.

For software portals, congestion state change notifications (CSCN) are sent by asserting an interrupt on the interrupt signal associated with each software portal. Upon receiving an interrupt, and determining that the interrupt source is a CSCN by reading the ISR (CSCI bit asserted in the ISR), software must issue a command (see [Section 3.3.9.6.3, “Query Congestion State”](#)) to query the current congestion state of all groups from QMan. A snapshot of the current congestion state will be returned in the response to the command. Congestion group state contains a total of 256 bits (8 words), with each bit indicating the state (0 = not in congestion, 1 = in congestion) of one congestion group.

To ensure that no congestion state changes are missed, software must clear the CSCI bit in the ISR before issuing the query congestion state command. Depending on interrupt response latency in software, several state changes in different congestion groups may have occurred, therefore all congestion state changes that occurred before the snapshot was taken must be handled as part of the interrupt service.

To determine if a congestion state change has occurred, software must maintain a copy of the last known congestion state of every congestion group which it is managing, and compare (for example, perform a bitwise XOR) the last known congestion state with the new congestion state snapshot received in response to the Query Congestion Group State command. Note that it is possible for the state of a congestion group to change twice before the interrupt is processed, in which case the net result is that software sees no state change, and has no work to do in response to the interrupt.

It is likely that each processor will manage a certain subset of the congestion groups available, the entity that will receive a CSCN (one of the direct connect or software portals) is configurable for each of the congestion groups. The software receiving a CSCN interrupt need only read and compare the word or words of congestion group state (which contains a total of eight 32 bit words) in which it is interested.

### 3.3.14.7 Congestion Group Record (CGR)

Several fields in the CGR must be initialized by software, these are made available in a management command available to software portals; see [Section 3.3.9.6, “Congestion Group Management Commands.”](#) Table 3-165 describes the CGR fields.

**Table 3-165. Congestion Group Record (CGR)**

CGR Field	CGR Field Description
num_dcp_portals-1 +num_sw_portals +206 : num_sw_portals +206  CSCN_TARG_DCP	Congestion State Change Notification (CSCN) Target for DCP portals A per-portal bit mask where the following bits notify the corresponding portal. bit num_sw_portals+206 = Send CSCN notifications for this Congestion Group to DCP portal 0. ... bit num_sw_portals+206+n = Send CSCN notifications for this Congestion Group to DCP portal n.
num_sw_portals-1 +206 : 206  CSCN_TARG_SWP	Congestion State Change Notification (CSCN) Target for software portals A per-portal bit mask where the following bits notify the corresponding portal. bit 206 = Send CSCN notifications for this Congestion Group to software portal 0. ... bit 206+n = Send CSCN notifications for this Congestion Group to software portal n.
205 MODE	Congestion Group Mode, configured by software. 0 = Perform congestion avoidance based on byte count 1 = Perform congestion avoidance based on frame count
204 WR_EN_G	WRED Green Enable - Use WRED (random probability vs. Curve@ Average) for ‘Green’ colored enqueues
203:172 WR_PARM_G	WRED Green Parameters Bits 0-7: MA Bits 8-12: Mn MaxTH = MA * 2 <sup>Mn</sup> Bits 13-19: SA (value must be between 64-127) Bits 20-25: Sn (value must be between 7-63) Slope = SA * 1/2 <sup>Sn</sup> (can also be written as Slope = SA * 2 <sup>-Sn</sup> ) Bits 26-31: Pn MaxP = (Pn+1) * 4

**Table 3-165. Congestion Group Record (CGR) (continued)**

CGR Field	CGR Field Description
171 WR_EN_Y	WRED Yellow Enable - Use WRED (random probability vs. Curve@ Average) for 'Yellow' colored enqueues
170:139 WR_PARM_Y	WRED Yellow Parameters See WR_PARM_G
138 WR_EN_R	WRED Red Enable - Use WRED (random probability vs. Curve@ Average) for 'Red' colored enqueues
137:106 WR_PARM_R	WRED Red Parameters See WR_PARM_G
105 CSCN_EN	Congestion State Change Notification (CSCN) Enable - If this bit is asserted, CSCN will be sent to the portal indicated by CSCN_TARG_DCP and CSCN_TARG_SWP when this congestion group both enters and exits congestion. Congestion State threshold is used for the CSCN calculation.
104 CSTD_EN	Congestion State Tail Drop Enable - If an enqueue occurs while the CS bit is 1, the CM will tell the producer to drop the frame.
103:91 CS_THRES	Congestion State Threshold Bits 0:7: TA (value should be > 0. If 0 is used, the CS bit can never transition from 1 to 0) Bits 8-12: Tn CS Threshold = TA * 2^Tn
90 CS	Congestion State relative to Congestion State Threshold Used for the purpose of CSCN notifications and CS Tail Drop. See <a href="#">Section 3.3.14.6, "Congestion State Change Notifications (CSCN)"</a> . Determined by comparing the instantaneous byte (or frame) count to the Congestion State Threshold. This bit is set to 1 on a positive greater-than crossing of the threshold. This bit is set to 0 on a negative less-than crossing of approximately 7/8 of the threshold. 0 = This Congestion Group is currently not congested relative to Congestion State Threshold. 1 = This Congestion Group is currently congested relative to Congestion State Threshold.
89:51 I_BCNT	Congestion Group Instantaneous Byte (or frame) Count. Number of bytes (or frames, if MODE is set to 1) currently in use in all FQ that belong to this Congestion Group.
50:12 A_BCNT	Congestion Group Average Byte (or frame) Count. This is a running average of the number of bytes (or frames, if MODE is set to 1) used in all FQ that belong to this Congestion Group. This value is maintained by QMan and is updated on every enqueue and dequeue processed by QMan. Note that the averaging continues to operate normally even if the I_BCNT returns to 0 at any point in time. The running average is calculated as:  $\text{ABCNT}_{\text{delta}} = \text{ABCNT}_{\text{current}} - \text{IBCNT}_{\text{current}}$ $\text{TIMESTAMP}_{\text{delta}} = \text{CurrentTime} - \text{TIMESTAMP}$ $\text{ABCNT}_{\text{newdelta}} \sim= \text{ABCNT}_{\text{delta}} \times 2^{-\text{TIMESTAMP}_{\text{delta}}/128} \quad \text{--- approximating exponential decay}$ $\text{ABCNT}_{\text{new}} = \text{IBCNT}_{\text{current}} + \text{ABCNT}_{\text{newdelta}}$
11:0 TIMESTAMP	Time stamp - A time stamp of when the last update to this group happened. Used in calculating the average byte count for this group. The time stamp 'ticks' when the pre-scaler configured in CM_CFG rolls over.

### 3.3.15 Dynamic Debug (DD)

The QMan provides support for dynamic debug (DD), leveraging the debug infrastructure provided in the SoC. The following debug features are supported:

- Ability to generate trace events and information for marked frames
- Support for dynamic frame marking
- Support for debug halt on all portals when an internal trace event is generated
- Support for debug halt on all portals under command of the SoC debug logic
- Ability to generate performance monitor events.

#### 3.3.15.1 Data Path Frame Marking

Two bits have been set aside in the DD field of the FD to support marking of frames for dynamic debug purposes; see [Section 3.3.1.2, “Frame Descriptors \(FDs\).”](#) These bits are used to create a Dynamic Debug (DD) code:

- 00 = frame is not marked, no debug action taken
- non-zero = frame is marked, debug action may be taken

In general, since the DD code is located in the FD, marking is provided to QMan when a frame is enqueued via a software or direct connect portal, and QMan will pass the DD code marking along when frames are dequeued.

The meaning of a non-zero DD code is dependent on the configuration of each datapath block. QMan contains a number of configuration registers which are used to control how it will react to frames carrying a non-zero DD code. QMan can be configured to mark (or re-mark) a frame, generate various trace events, generate performance monitor events, and to halt a portal based on a frame’s DD code, these actions are described in more detail in the following sub sections.

#### 3.3.15.2 Debug Trace Points

QMan can be configured to generate trace events, performance monitor events, and internal halt requests for marked frames, that is, frames carrying a non-zero DD code, that pass through specific trace points. There are 3 different types of trace point defined within QMan:

- Enqueue Command Trace Points: This trace point is triggered when an enqueue command received via a portal and containing a marked frame is executed. One of these trace points exists for each software and direct connect portal in QMan.
- Deferred Enqueue Complete Trace Points: This trace point is triggered when the enqueue of a marked frame received via a software portal was deferred due to order restoration. The trace event is generated once the enqueue completes, that is, at the time when the frame is enqueued onto the FQ, after being delayed due to order restoration. One of these trace points exists for each software portal, but not for direct connect portals.
- Dequeue Trace Points: This trace point is triggered when a marked frame is dequeued from a channel. One of these trace points exists for each WQ channel in QMan.

When a trace event is generated as a result of a marked frame passing through a trace point, either terse or verbose trace information may be included with the trace event. A terse trace event occupies a single cycle on the dynamic debug interface, a verbose trace event carries more information and occupies 2 cycles on the interface. Multiple trace events can be generated in quick succession, and QMan will buffer these events as needed before sending them on to the external debug logic via the Data Path Debug Client (DPDC) block in the SoC. QMan’s behavior when its debug trace event output buffer becomes full can be configured (via a control register in the DPDC block) to operate in one of two modes:

- Lossless mode: In this mode, no trace events are lost, but stalls in QMan operation can occur if trace events are occurring faster than the external debug logic can consume them and the trace event output buffer becomes full.
- Lossy mode: In this mode, if a trace event is generated and the debug trace event output buffer is full, QMan may drop the trace event, and will send an indication to DPDC that a trace event was lost when such a trace event drop occurs. Note that if a user has configured a dynamic debug trace point with both trace and portal halt enabled, then the trace event generated at this trace point, which also caused a portal to halt, will never be dropped, even if lossy mode is used. Only trace events which were not accompanied by a portal halt may be dropped.

Enqueue trace points are configured individually for each portal, and dequeue trace points are configured individually for each channel. For each trace point, generation of trace events can be enabled or disabled individually for each DD code, and when enabled, terse or verbose tracing can be selected.

The configuration of enqueue trace points for software portals is described in [Section 3.2.4.3, “QCSP Dynamic Debug Configuration Registers \(QCSPi\\_DD\\_CFG\).”](#) The configuration of enqueue trace points for direct connect portals is described in [Section 3.2.4.12, “DCP Dynamic Debug Configuration Registers \(DCPi\\_DD\\_CFG\).”](#) The configuration of dequeue trace points is described in [Section 3.2.4.24, “WQ Channel Dynamic Debug Configuration Registers.”](#)

### **3.3.15.3 QMan Dynamic Frame Marking**

In addition to generating trace events, QMan can be configured to mark (or re-mark, if the frame already carries a non-zero DD code) frames which are enqueued through an order restoration point (ORP), and whose enqueue is deferred because a frame with a lower sequence number has not yet arrived at the same ORP.

Frames which are not enqueued with order restoration, or frames enqueued with order restoration but which are not deferred, are not marked by QMan. A deferred enqueue due to order restoration is the only condition in which QMan will dynamically mark a frame. Such marking can be used to identify frames which have been deferred (versus those that have not) when tracing those frames using trace events.

Dynamic frame marking is enabled on a per DD code basis, the configuration for each code and the new DD code value to be inserted in the frame are described in [Section 3.2.4.4, “QMan Dynamic Debug Configuration Register \(QMAn\\_DD\\_CFG\).”](#)

### **3.3.15.4 Debug Halt**

Each of the software and direct connect portals in QMan can be individually halted for debug purposes. The sources of debug halt request to each portal are as follows:

- Internal Debug Halt Request: This is a halt request generated by the arrival of a marked frame at one of QMan’s internal debug trace points. The generation of a halt request can be enabled for each DD code at each debug trace point within QMan, using the same registers as those used to configure the debug trace events; see [Section 3.3.15.2, “Debug Trace Points.”](#)
- External Halt Request: This is a halt request received from the SoC debug logic connected to QMan.
- Internal Error Halt Request: This is a halt request generated as the result of an internal error detected by QMan. Several types of internal error can be programmed to assert this halt request, and such an error will assert a halt request to all software and direct connect portals simultaneously. See [Section 3.2.4.55, “QMan Error Halt Enable Register \(QMAM\\_ERR\\_HER\).”](#)

When any halt request is received at a portal, that portal will stop dispatching commands (enqueue commands, dequeue commands, and management commands in the case of a software portal) to QMan’s Algorithmic Sequencers (AS). Any commands that have already been dispatched from that portal when the halt request is received will be completed and their responses (if any) returned. Once all in progress commands have completed, the portal enters halted state, and no new commands are dispatched until the halt request is removed.

The halt status of each portal (halted or not) can be read via a register (see [Section 3.2.4.9, “QCSP Dynamic Debug Halt Acknowledge Status Registers \(QCSP\\_DD\\_HASR\\_i\),”](#) and is also communicated to the SoC debug logic via a set of halt acknowledge signals. Also, for a portal halted by an internal debug halt request, if the trace point which caused the portal to halt is also configured to generate a trace event, that trace event will contain a bit indicating that the portal was halted by the frame which generated the trace event. This can be used to determine the exact frame and trace point which triggered an internal debug halt request.

When a software portal is in the halted state, it will not consume any entries from its enqueue command ring (EQCR). Software may still add entries to the ring (if it is not full) and advance the ring’s producer index, but these entries will not be consumed until the halt request is removed. The dequeue response ring (DQRR) may contain valid responses to previously dispatched commands when the portal is halted, and software may consume these entries and advance its consumer index, however QMan will add no new entries to the ring until the halt request is removed. The same applies to the MR and the management CR.

The same approach is applied to the direct connect portals. When a direct connect portal is in the halted state, it will not consume from its command FIFOs, however requesters may add commands as long as the FIFOs are not full. Similarly the response FIFOs may contain valid responses to previous commands, and requesters may drain these responses, but no new responses will be generated until the halt request is removed.

An External Halt Request is removed by the SoC debug logic when it negates the halt request signal for a particular portal. For an Internal Debug or Internal Error Halt Request, once asserted it remains latched and must be removed by writing to the appropriate bit in the appropriate Internal Halt Request Status Register; see [Section 3.2.4.6, “DCP Dynamic Debug Internal Halt Request Status Register \(DCP\\_DD\\_IHRSR\).”](#) When all internal and external halt requests to a portal are removed, the portal resumes normal operation. As such, each portal can be gracefully stopped and re-started. The entity (software or a direct connected hardware module) using a portal receives no special indication that a portal is halted, other than the fact that it stops consuming commands and generating responses.

### 3.3.16 Error Management and Recovery

QMan can detect a number of different internal error conditions. Each of these can be enabled to generate an interrupt on QMan’s error interrupt signal. QMan provides an interrupt signal for errors that is separate from the run time interrupt signals that are provided for each software portal. The various error sources detected by QMan are shown in [Section 3.2.4.51, “QMan Error Interrupt Status Register \(QMAM\\_ERR\\_ISR\).”](#)

In addition to generating an interrupt, each of the error sources can be configured to put QMan into a halted state by issuing a debug halt request to all of QMan’s software and direct connect portals. The configuration of which error sources can or cannot cause a QMan halt is programmable; see [Section 3.2.4.55, “QMan Error Halt Enable Register \(QMAM\\_ERR\\_HER\).”](#)

### 3.3.17 System Interfaces

QMan and BMan share a single interface to the system interconnect of the SoC, and connect as a single entity.

QMan provides two sets of system interfaces, one for initiating transactions onto the system interconnect, and one for receiving transactions from the system as a target.

### 3.3.18 System Target interface

The system target interface is used to receive read and write transactions and forward them to the software portals in either QMan or BMan.

The SoC implements Local Access Windows (LAW) used to define the system memory map. One of the LAW must be used for accessing QMan and BMan as a target, and must be programmed by software with the target ID of QMan’s target interface. QMan and BMan share the same target interface and thus the same target ID, and together they occupy a window in system memory space, with QMan’s software portals occupying the first (numerically smaller address) half and BMan’s software portals occupying the last (numerically larger address) half of the window. Within each half window, the lsbs of the address are decoded to determine which resource (which register within which portal) is the target of the transaction.

### 3.3.19 System Initiator Interface

The system initiator interface is used to issue transactions from QMan or BMan onto the system interconnect. QMan’s initiator interface carries two distinct types of transactions:

- Read and write transactions which target data in main memory.
- Stashing transactions which target the L2 frontside cache in the SoC.

Read and write transactions to main memory are used to access and maintain the private data structures used by QMan, and the memory regions accessed by these transactions are configured by software; see [Section 3.2.4.43, “Data Structure Base Address Registers.”](#)

Stashing transactions are used to position key data and context for dequeued frames into cache on behalf of software; see [Section 3.3.8.6, “DQRR Entry Stashing,”](#) [Section 3.3.8.7, “EQCR\\_CI Stashing,”](#)

### 3.3.19.1 SrcID and ICID

QMan contains a fixed, non-programmable 8 bit Source ID. This ID is readable in a register; see [Section 3.2.4.47, “QMan Source ID Register \(QMAM\\_SRCIDR\).”](#) This ID must be unique within a given SoC. This Source ID is attached to every transaction initiated by QMan.

All FDs include a ICID (see [Section 3.3.1.2, “Frame Descriptors \(FDs\)”](#)) to be used by the entity that will dequeue the frame. This ICID will apply to all addresses embedded in the frame including scatter/gather buffers. All ICID s must be provided by some hardware entity, software must not be trusted to directly specify its ICID in the FD that it provides. Therefore for frame enqueuees received from software (via a software portal), QMan contains an assigned ICID value for each software portal and inserts this value into the FD provided with the enqueue command, overwriting the ICID that was provided in the FD. The registers used to program the ICID value of each software portal are described in [Section 3.2.4.1, “QCSP ICID Configuration Registers \(QCSPi\\_ICID\\_CFG\).”](#)

For frames received from a direct connect portal, the hardware block issuing an enqueue command must insert a valid ICID into the FD that is provided with the enqueue command.

### 3.3.19.2 Initiator Scheduling and Priority

As mentioned earlier, QMan’s initiator interface carries the following distinct types of transactions:

- Read and write transactions that target data in main memory
- Stashing transactions that target a cache in the SoC.

For reads and writes to main memory, the initiator interface services transactions on behalf of both QMan and BMan. The read and write transactions from QMan’s internal sources and those from BMan are treated as two separate sources. The following sub-modules are sources of read and write transactions to the initiator interface.

- QMan internal sources of read and write transactions:
  - QMan Algorithmic Sequencers (AS).
  - QMan PFDR (Packed Frame Descriptor) Manager
  - QMan OLM (Order List Manager)
- BMan internal sources of read and write transactions:
  - BMan Fetch/Flush Sequencer

The initiator interface contains two separate read/write transaction queues, one for all reads and writes received from QMan’s internal sources, and the other for those received from BMan. A work conserving round robin arbiter is used to select transactions to be added to the QMan read/write transaction queue.

For stashing transactions, the initiator interface maintains a set of Stash Request Queues (SRQ); see [Section 3.3.8.9, “Stash Transaction Scheduling.”](#) The only source of stash transactions into the SRQs is the QMan AS sequencers which service the software portals. The arbitration between these AS for access to the tail of the SRQs is done on a round-robin basis with all AS treated as equal. At the head of the SRQs, a work conserving round robin arbiter is used to select transactions one at a time from one of the SRQs to be forwarded onto the system.

As a result, there are up to 3 separate transactions available at any time for forwarding onto the system interconnect, one from the QMan read/write transaction queue, one from the BMan read write transaction queue, and one from the SRQs.

The scheduler within the initiator sub-block services each of these 3 sources using a work-conserving Weighted Interleaved Round Robin (WIRR) algorithm. The relative weight (range 1 to 8) of each of the 3 sources is programmable (see [Section 3.2.4.46, “Initiator Scheduling Configuration \(CI\\_SCHED\\_CFG\)”](#)). The number of selections made from each source is tracked using a selection counter. The WIRR algorithm used is summarized below:

- All selection counters start at 0.
- A requester source is eligible for selection if it contains a pending request and its selection counter is not greater than its programmed weight value (0 to 7, corresponds to relative weight of 1 to 8).
- A work conserving round robin selection is made among all eligible sources (1 of 3). The selection counter for the selected requester is incremented.
- In any cycle in which all requesters are not eligible, all selection counters are reset to 0.

Once this scheduler has made a selection, the selected transaction is placed in a FIFO queue to be issued onto the system. The relative priority to be used for these transactions (Lower/Higher priority) is carried in-band with the transaction. For QMan’s read/write transactions the transaction priority of each of the different data structures accessed and managed by QMan is separately programmable (see [Section 3.2.4.44, “Data Structure Attributes Registers”](#)). For stash transactions the priority is programmable by transaction type and per software portal; see [Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\)”](#). For read/write transactions received from BMan, the transaction priority is signaled from BMan along with the transaction request. All transaction priorities are configured at initialization time, and should not be changed dynamically while traffic is flowing.

### **3.3.19.3 CPC Stashing of QMan Private Data Structures**

QMan uses two distinct regions of main memory for storing its private data structures, one for FQD and one for PFDR. These regions are allocated by software and programmed into QMan using a set of base address and attribute registers for each of the two regions; see [Section 3.2.4.43, “Data Structure Base Address Registers.”](#)

These private data structures are intended to be stored in external memory (DRAM), but as a performance improvement they may also be stored in a cache that can allocate writes and/or reads as they pass by on the system interconnect. From the point of view of QMan’s private data structures, such a cache is referred to as the Common Platform Cache (CPC). The act of storing a specific piece of data into the CPC when it is written to or read from main memory is known as stashing.

This is similar to the stashing function described in [Section 3.3.8.6, “DQRR Entry Stashing,”](#) except that for those stashing features QMan issues specific additional transactions to achieve the stashing, whereas CPC stashing of private data structures uses the writes and reads that are already occurring to those structures, with no additional transactions on the system interconnect.

Stashing of read and write transactions into the CPC can be done in two ways:

- Address based Implicit CPC stashing, in which the transaction issued by the initiator (QMan) signals a basic read or write transaction, and the CPC itself decides to stash the transaction based on a hit in a programmed address range.
- Attribute based Explicit CPC stashing, in which the transaction issued by the initiator (QMan) signals its desire to stash using attributes attached to the transaction.

Implicit CPC stashing can be done without QMan's involvement, it is wholly controlled by programming in the CPC.

Explicit CPC stashing involves programmed settings in QMan .

From the QMan side, explicit CPC stashing can be enabled or disabled separately for FQD and for PFDR; see [Section 3.2.4.44, “Data Structure Attributes Registers.”](#) For FQD, CPC stashing is also programmable on a per-FQ basis, using the CPC Stash Enable bit in the FQ\_CTRL field of the FQD (see [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\).”](#))

So to enable explicit CPC stashing of full 64 byte writes to a particular FQD (explicit stashing is never signaled on FQD reads, nor on partial 4 byte writes), both the global CPC stash enable bit in the FQD\_AR register and the CPC stash enable bit in the FQD must be asserted. Explicit CPC stashing of PFDR reads and writes is enabled/disabled only by its global enable bit in the PFDR\_AR register.

## 3.3.20 Customer Edge Egress Traffic Management (CEETM)

### 3.3.20.1 CEETM Overview

This section describes a version of egress traffic management support in QMan that provides hierarchical class based scheduling and traffic shaping. This version of egress traffic management is called Customer Edge Egress Traffic Management (CEETM).

CEETM is intended for use on links leading to a Wide Area network (WAN). The capabilities provided may be useful in other places but they are not the immediate design target of this feature. As a result the capabilities will be best suited to systems which lead into a WAN (essentially customer premises at the user rather than data center side of the network).

CEETM maintains the following functionality equivalent to that supported by FQ/WQ scheduling mode:

- Congestion management capabilities including WRED
- Dequeued frame context (Context\_A and Context\_B)
- Priority or traffic class flow control

Use of CEETM does not require a change to the frame sources (enqueuers). Enqueues to a queue which is subject to CEETM are done using the same interface and semantics as the current FQ/WQ based scheme.

CEETM is implemented as a mode of scheduling for sub-portals on specific QMan Direct Connect Portals (DCPs). Each sub-portal on a DCP which supports CEETM can be configured to use either the regular FQ/WQ scheduling mode or CEETM scheduling mode, see [Section 3.2.4.11, “DCP Configuration Registers \(DCPi\\_CFG\).”](#) It is possible to switch between FQ/WQ and CEETM scheduling mode on a given sub-portal, but not while traffic is flowing on that sub-portal. Also, not all sub-portals within a DCP can

be enabled to use CEETM at the same time, CEETM supports up to 8 logical network interfaces (LNI) that can each be mapped to a DCP sub-portal, whereas a DCP can support up to 16 sub-portals.

Although there is nothing within QMan which dictates how CEETM is used or connected within a SoC, it is intended to be used only with network interfaces and I/O interfaces. Specifically, on current SoC CEETM is intended to be used by FMan sub-portals used with Ethernet interfaces.

Generally, CEETM is supported on only a subset of the DCP portals available on a particular SoC, not on all DCP portals. A single instance of the CEETM logic is associated with a single DCP and therefore a single egress I/O module. If multiple modules in a SoC, for instance multiple FMan, need to support CEETM then multiple instances of the CEETM logic are present, one for each DCP which supports CEETM, and each containing its own set of separate resources described in the following section.

### **3.3.20.2 CEETM Features**

- In the LS1043A CEETM is supported on each DCP portal used by FMan, that is, one CEETM on DCP0. The CEETM in this SoC supports 8 channels.

Each instance of CEETM supports the following features:

- Supports hierarchical multi-level scheduling and shaping which
  - is performed in an atomic manner; all context at all levels is examined and updated synchronously.
  - employs no intermediate buffering between class queues and the direct connect portal to the FMan.
  - Algorithms for shaping and weighted fairness are based on actual packet size (in bytes) of selected frames - frame size may be up to 16KBytes
- Supports dual-rate shaping (paired committed rate (CR) shaper and excess rate (ER) shaper) at all shaping points.
  - Shapers are token bucket based with configurable rate and burst limit.
  - Paired CR/ER shapers may be configured as independent or coupled on a per pair basis; coupled means that credits to the CR shaper in excess of its token bucket limit is credited to the ER bucket
- Supports 8 logical network interfaces (LNI)
  - Each LNI:
    - aggregates frames from one or more channels.
    - priority schedules unshaped frames (aggregated from unshaped channels), CR frames, and ER frames (aggregated from shaped channels)
    - applies a dual-rate shaper to the aggregate of CR/ER frames from shaped channels
    - can be configured (or reconfigured for lossless interface failover) to deliver frames to any network interface.
  - Supports either 32 or 8 channels available for allocation across the 8 LNIs.
    - Each Channel (or the Class Scheduler of each channel):
      - can be configured to deliver frames to any one of the LNI.

- can be configured to be unshaped or shaped; when shaped, a dual rate shaper applies to the aggregate of CR/ER frames from the channel.
- contains a total of 16 class queues (CQ), with 8 independent classes and 8 grouped classes; grouped classes can be configured as 1 class group of 8 or as 2 class groups of 4.
- supports weighted bandwidth fairness within grouped class groups with weights configured on a channel and class basis.
- strict priority scheduling of the 8 independent classes and the aggregate(s) of the grouped classes; the priority of each of the 2 class groups can be independently configured to be immediately below any of the independent classes.
- is configurable such that each of the 8 independent classes and 2 class groups can supply CR frames, ER frames or both when channel is configured to be shaped.
- is configured independently of other channels.
- Each Class
  - has a dedicated class queue (CQ)
  - can have a dedicated or shared Class Congestion Group Record (CCGR). CEETM supports sufficient number of CCGRs for all CQs to have a dedicated CCGR if desired. CCGRs within CEETM provide equivalent or better functionality than CGRs provide to FQs.
  - can be flow-controlled by traffic-class flow control messages from the FMan; achieves backward compatibility with traffic class flow control in FQ/WQ scheduling mode by allowing each of the 16 CEETM classes to be configured (per LNI) to respect one or none of the 8 traffic classes which can be flow controlled.
  - is identified via a “logical frame queue identifier (LFQID)” to maintain semantic compatibility with enqueue commands to frame queues (non-CEETM queues). Semantic compatibility also permits the invocation of the order restoration function as a frame is enqueued to a class queue.
  - supports the identification of intra-class flows (logically equivalent to FQs but not queued separately) in order to apply static context (Context\_A and Context\_B) to frames as they are dequeued from CQs; this provides functionality equivalent to that available when a frame is dequeued from a frame queue (non-CEETM queues).
  - can queue up to 8 frames on its CQ in extended singular frame descriptor records (XSFDR) in dedicated internal memory for fast storage and retrieval, analogous to the way SFDRs are used in FQs in FQ/WQ scheduling mode. Storage for a total of 4096 (for 32 channel CEETM) or 1024 (for 8 channel CEETM) XSFDR is provided, shared among all CQs in all channels.
  - Uses packed frame descriptor records (PFDR) to queue a larger number of frames, analogous to the way PFDR are used in FQs in FQ/WQ scheduling mode. PFDR are stored in external memory (DRAM), and the same pool of PFDRs is used by all CEETM CQs, and all FQ/WQ scheduling mode FQs.

### 3.3.20.3 Example CEETM scheduling hierarchy for one logical network interface

Figure 3-175., “Example CEETM scheduling hierarchy for one logical network interface” shows an example of the shaping and scheduling possible for a single logical network interface (LNI) using CEETM. Each LNI is intended to provide the egress traffic for a single physical link, such as an Ethernet interface.

## **Queue Manager (QMan)**

This figure provides a brief preview of overall CEETM scheduling, and each of the CEETM components is described in more detail in the following sections.

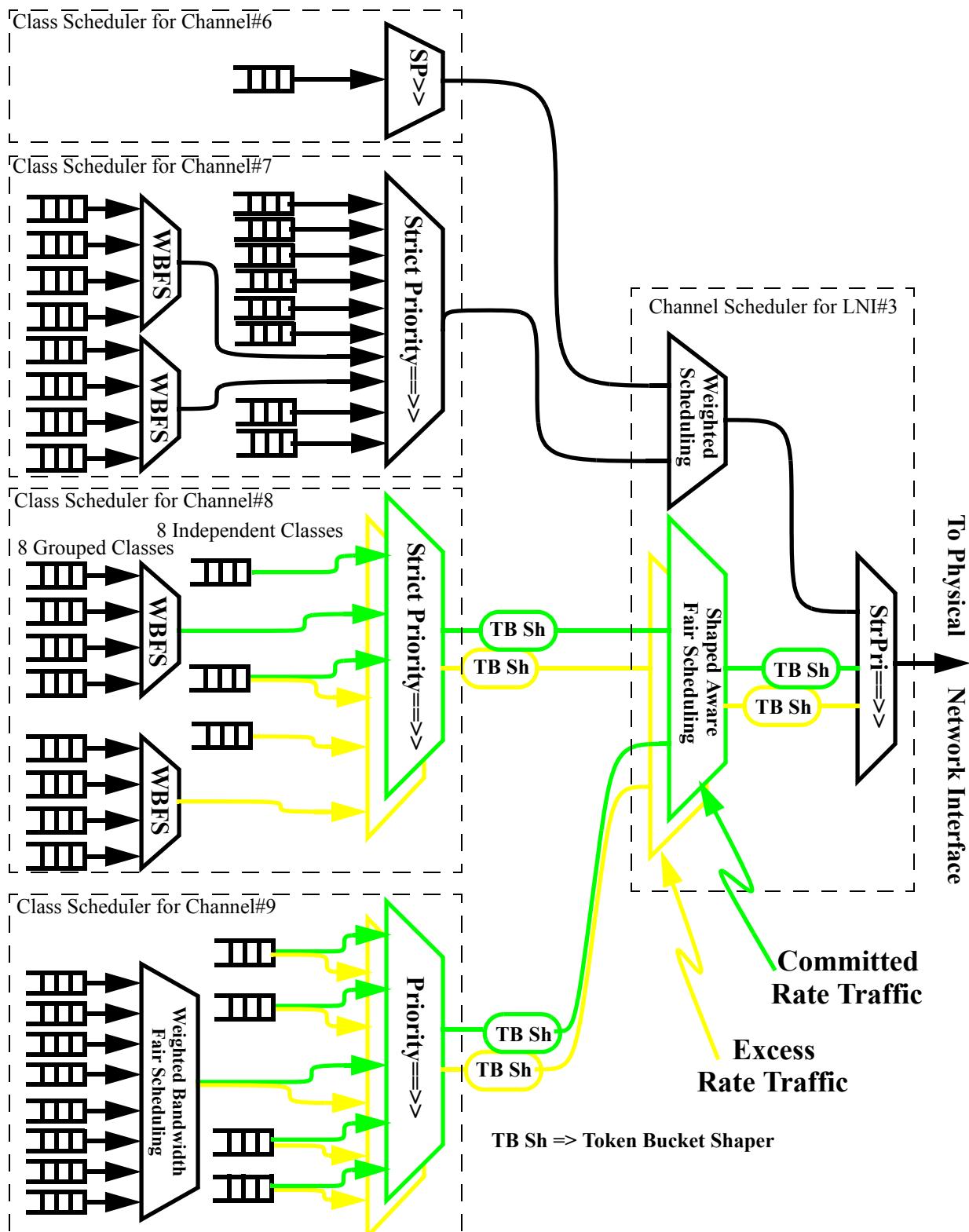


Figure 3-175. Example CEETM scheduling hierarchy for one logical network interface

As shown in [Figure 3-175](#) the scheduling hierarchy in CEETM consists of two major levels.

The lower level (left side) is the class scheduler level. At this level frames are selected from class queues to produce an aggregate that is referred to as a channel or, more specifically, as a “class queue channel” or CQ channel<sup>1</sup>. The class scheduler makes class selection decisions at two sub-levels. At the lower sub-level a grouped class queue is selected for among its peers (group members) based on an algorithm referred to as Weighted Bandwidth Fair Scheduling (WBFS). Grouped class queues can be treated as one group of eight or as two groups of four as configured. At the upper sub-level a independent class queue or class queue group is selected based on strict priority. The independent class queue have fixed priority relative to each other (variable priority would provide no value as the queues are functional equivalents). The priority of the groups can be independently configured to be below any of the independent queues. If the channel is configured to be shaped the strict priority selection is performed in a CR/ER aware manner which is to say it is aware whether independent class queues/class groups are configured to eligible for type of opportunity (CR/ER) being afforded to the channel. Implementation notwithstanding, the class schedulers can be viewed as tangible entity that exists for each channel and can be configured independently for each channel.

A channel may be configured to be shaped or unshaped. When configured to be shaped the class scheduler of the channel shall operate in a CR/ER aware manner and CR/ER opportunities. Implementation notwithstanding, a channel shaper can be viewed as tangible entity that exists for each channel and can be configured independently for each channel.

The upper level (right side) is the channel scheduler level. At this level channels are scheduled which means the class schedulers for particular are afforded opportunities to select frames the from their class queues. The aggregate that created by the channel scheduler is referred to as a Logical Network Interface. Each channel must be statically configured to be associated (feed) with the channel scheduler of one LNI. The channel schedulers can be viewed as tangible entity that exists for each LNI and can be configured independently for each channel. The class scheduler makes class selection decisions at two sub-levels. The upper level makes a strict priority selection between three sets of channels - unshaped channels with data, shaped channels with eligible CR data and shaped channels with eligible ER data. At the lower level a specific channel is selected using a fairness algorithm. For unshaped channels a simple algorithm attempts to equalize bandwidth. For shaped channels a proprietary algorithm, Shape Aware Fair Scheduling is used to select a channel such that channels achieve their shape to the degree possible. This algorithm has an awareness of the order in which channels became entitled to be selected even if there is no data available - this lets channels that are within their shape receive better latency.

The CEETM configuration illustrated is very asymmetrical and is intended to demonstrate the degrees of configurability rather than an envisioned use case. The color green denotes logic units and signal paths that relate to the request and fulfillment of committed rate (CR) packet transmission opportunities. The color yellow denotes the same for excess rate (ER). The color black denotes logic units and signal paths that are used for unshaped opportunities or that operate consistently whether used for CR or ER opportunities.

---

1. A channel within CEETM can be more specifically referred to a “class queue channel” or CQ channel to distinguish it from channels (WQ channels) that represent the aggregate of frames selected from a hierarchical structure of class work queues and frame queues as is the case for class scheduling outside of the CEETM. Note the consistency that in both cases a channel refers to result of an aggregation of frames from distinct classes produced by some form of class scheduler.

This scenario depicts the following:

- Channels #6,#7,#8 and #9 have been configured to be scheduled by the channel scheduler for LNI#3 (i.e. all packets from these channels are directed via LNI#3 to the physical network interface coupled by configuration to LNI#3).
- Channels #6 and #7 have been configured to be “unshaped”. Packets from these channels will not be subjected to shaping at the channel level and will feed the top priority level within the LNI which is also not subjected to shaping. Their class schedulers will not distinguish between CR and ER opportunities.
- Channels #8 and #9 have been configured to be “shaped”. Their class schedulers will distinguish between CR and ER opportunities. The CR/ER packets to be sent from each channel shall be subjected to a pair of CR/ER token bucket shapers specific to that channel. The aggregate of CR/ER packet from these channels shall be subject to a pair of CR/ER token bucket shapers specific to LNI#3.
- Channels #6 has only one class in use. That class queue will behave as if it were a channel queue and a peer to channel #7. Unused classes do not have to be configured as such - simply not used.
- Channel #7 has all 16 classes in use.
  - The group classes have been configured as 2 groups (A and B) of 4 classes.
  - The priority of the groups A and B have both been set to be immediately below independent class 5. In a case of similar configuration group A has higher priority than group B.
- Channels #8 has 3 independent classes and 2 groups of 4 grouped classes in use.
  - The priorities of the class groups A and B have been set to be immediately below independent class 0 and class 2 respectively.
  - Independent class 0 and class group A have been configured to request and fulfill only CR packet opportunities.
  - Independent class 1 has been configured to request and fulfill both CR and ER packet opportunities.
  - Independent class 2 and class group B have been configured to request and fulfill only ER packet opportunities.
- Channels #9 has 4 independent classes and 1 group of 8 grouped classes in use.
  - The group classes have been configured as 1 group (A) of 8 classes.
  - All independent classes and the class group (A) have been configured to request and fulfill both CR and ER packet opportunities.

### **3.3.20.4 CEETM Class Queue (CQ)**

CEETM uses a single level queueing structure, the class queue. Class queues are simply FIFO queues of frame descriptors. This differs significantly from the two level queuing structure used in the regular QMan work queues and frame queues.

A WQ is a FIFO queue of FQs, and FQs are FIFO queues of frame descriptors which can be enqueued onto WQs when they contain any frame descriptors. This structure guarantees ordering of frames within a FQ but not between frames on different FQs.

Class queues on the other hand provide ordering guarantees for all frames in the class queue based on the order in which they were enqueued on to the class queue.

### 3.3.20.4.1 CEETM Class Queue Descriptor

Class queue descriptors (CQDs) are internal data structures that are managed by the QMan only, and are not directly accessible by software or other hardware modules.

After reset all CQDs are initialized to a default value by hardware. The user can change this default initialization, and can examine CQD contents, by using specific software commands, see [Section 3.3.9.7.3, “CEETM Class Queue \(CQ\) Configure”](#) and [Section 3.3.9.7.4, “CEETM Class Queue \(CQ\) Query.”](#)

Unlike FQDs, CQDs are never stored in main memory, they are always stored in QMan internal memory, therefore no allocation of main memory is needed for CQDs.

The following table shows the contents of a CQD, and identifies those fields which can be initialized by software.

**Table 3-166. CEETM Class Queue Descriptor (CQD) Format Description**

Field	Software Config?	Description
11-0 RA2_XSFDR_PTR	—	XSFDR pointer for recently arrived frame number 2 Points to an XSFDR containing the FD of a frame recently arrived at the tail of the FQ. Only valid when NRA is 2 and IT is 0.
23-12 RA1_XSFDR_PTR	—	XSFDR pointer for recently arrived frame number 1 Points to an XSFDR containing the FD of a frame recently arrived at the tail of the CEETM CQ. Only valid when NRA is 1 or 2 and IT is 0.
35-24 OD6_XSFDR_PTR	—	XSFDR pointer for on-deck frame number 6 Points to an XSFDR containing the 6th on-deck FD at the head of the CEETM CQ. Only valid when NOD is equal to 6.
47-36 OD5_XSFDR_PTR	—	XSFDR pointer for on-deck frame number 5 Points to an XSFDR containing the 5th on-deck FD at the head of the CEETM CQ. Only valid when NOD is greater than or equal to 5.
59-48 OD4_XSFDR_PTR	—	XSFDR pointer for on-deck frame number 4 Points to an XSFDR containing the 4th on-deck FD at the head of the CEETM CQ. Only valid when NOD is greater than or equal to 4.
71-60 OD3_XSFDR_PTR	—	XSFDR pointer for on-deck frame number 3 Points to an XSFDR containing the 3rd on-deck FD at the head of the CEETM CQ. Only valid when NOD is greater than or equal to 3.
83-72 OD2_XSFDR_PTR	—	XSFDR pointer for on-deck frame number 2 Points to an XSFDR containing the 2nd on-deck FD at the head of the CEETM CQ. Only valid when NOD is greater than or equal to 2.
95-84 OD1_XSFDR_PTR	—	XSFDR pointer for on-deck frame number 1 Points to an XSFDR containing the first on-deck FD at the head of the CEETM CQ. Only valid when NOD is greater than or equal to 1.

**Table 3-166. CEETM Class Queue Descriptor (CQD) Format Description (continued)**

<b>Field</b>	<b>Software Config?</b>	<b>Description</b>
98-96 NOD	—	Number of on-deck frames in XSFDRs at the head of the CEETM CQ. Valid values are 0 to 6. The FD of these frames are stored in XSFDR indexed by OD1_XSFDR_PTR to OD6_XSFDR_PTR. If NOD = 0 but the CEETM CQ is not empty, then the frames at the head of the CEETM CQ are stored in the head PFDR rather than in XSFDRs.
100-99 NPR	—	Number of previously read frames in the head PFDR Valid values are 0 to 2. This value is independent of the value of NOD. This field supports consuming (fetching) or prefetching a subset of the FDs from within the head PFDR when the number of available XSFDRs has dropped below a critical threshold. Under ideal circumstances (where XSFDRs are abundant) this value is constantly zero because all 3 FDs from the head PFDR are moved into on-deck XSFDRs whenever the number of ODs drops to 3 or less, and the PFDR is deallocated.
101 FIP	—	Fetch In Progress. When set to 1, this value serves two purposes: - Don't initiate another fetch - Don't move recent arrivals (RA1/2_XSFDR_PTR) into on deck positions even when PFDR chain is null, because there is a fetch read in progress.
102 IT	—	Recently arrived frames In Tail PFDR 0 Recently arrived frames (NRA = 1 or 2) are stored in XSFDRs. 1 Recently arrived frames (NRA = 1 or 2) are stored in the tail PFDR.
104-103 NRA	—	Number of recently arrived frames at the tail of the CEETM CQ Valid values are 0 to 2. The FD of these frames are stored either in XSFDR indexed by RA1_XSFDR_PTR and RA2_XSFDR_PTR, or in the tail PFDR indexed by PFDR_TPTR. The IT bit tracks whether the 1 or 2 recently arrived FDs are currently stored in XSFDRs or in the tail PFDR.
128-105 PFDR_HPTR	—	Packed frame descriptor record (PFDR) head pointer Pointer to the first record on the linked list of PFDRs for this CEETM CQ. The PFDR at the head of the list contains 1-3 FDs, which normally follow the “on-deck” FDs within the CEETM CQ.
152-129 PFDR_TPTR	—	Packed frame descriptor record (PFDR) tail pointer Pointer to the last record on the linked list of PFDRs for this CEETM CQ. The tail PFDR may contain 0, 1, or 2 valid FDs. It is never filled (never contains 3 valid FDs) because a new empty PFDR is always attached at the same time as the last FD of a PFDR is written. Under ideal circumstances (where XSFDRs are abundant) there are always 0 valid FDs in the tail PFDR. In these circumstances, 2 recently arrived FDs (stored in XSFDRs and indexed by RA1_XSFDR_PTR and RA2_XSFDR_PTR), a currently arriving FD stored in a third XSFDR, and a link to the new empty tail PFDR are all written into the current tail PFDR in a single 64 byte write. In cases where an arriving FD was allocated into an XSFDR when the number of available XSFDR was below the critical threshold, a partial PFDR flush is performed, allowing the critical XSFDR that was allocated to be released immediately. In such a case, 1, 2, or 3 FDs may be released from XSFDRs and written into the tail PFDR. The IT bit is used to track whether the recently arrived FDs are currently stored in XSFDRs or in the tail PFDR.
156-153 CCGID	Yes	CEETM Class Congestion Group ID. See <a href="#">Section 3.3.20.11, “CEETM Class Congestion Management and Avoidance (CCM).”</a>
180-157 FRM_CNT	—	Frame count, not used by the logic but can be queried for debug purposes.

### 3.3.20.5 Enqueuing onto a CEETM Class Queue

As mentioned in the overview, enqueueing a frame onto a CEETM class queue uses the same interfaces and semantics as enqueueing onto a regular FQ. The exact class queue on to which a frame is to be enqueued is determined from the FQID used when enqueueing the frame.

The upper 1M FQIDs (out of a total of 16M available) are reserved for enqueues using CEETM. Enqueues using these FQID values as destination FQ will be directed to the CEETM logic used by one of the DCP portals instead of going to a regular FQ. As such these FQID values are considered logical FQIDs (LFQID), as opposed to regular FQIDs which lead directly to a real FQ.

The valid LFQIDs for use with CEETM are shown in the table below, CEETM supports 4K (for 32 channel CEETM) or 1K (for 8 channel CEETM) LFQIDs per DCP which supports CEETM. Any enqueue to a FQID of F0\_0000 and above that does not fall into the range of valid CEETM LFQIDs shown in the table will result in an enqueue rejection and an IEQI error indicated in the QMAN\_ERR\_ISR register ([Section 3.2.4.51, “QMan Error Interrupt Status Register \(QMAN\\_ERR\\_ISR\)”](#)).

Likewise, any regular FQ management command (see [Section 3.3.9.5, “Frame Queue Management Commands”](#)) attempted to a LFQID (i.e. FQID greater than or equal to F0\_0000) will result in an invalid FQID error in response to that command.

When an enqueue is destined for CEETM, a lookup in the CEETM Logical FQ Mapping Table (LFQMT) is done to convert the LFQID to a CEETM Class Queue ID (CQID), a CEETM Dequeue Context Table Index (DCTIDX), and a CEETM Class Congestion Group ID (CCGID) triplet. Each CEETM instance contains its own LFQMT. The software portal provides commands that allow software to configure and query the LFQMT, see [Section 3.3.9.7.1, “CEETM Logical FQ Mapping Table \(LFQMT\) Configure.”](#) Note that the CCGID value in the LFQMT is actually taken from the value configured in the class queue descriptor, see [Section 3.3.9.7.3, “CEETM Class Queue \(CQ\) Configure.”](#)

The LFQMT contains 4K or 1K entries, and CEETM supports 512 (for 32 channel CEETM) or 128 (for 8 channel CEETM) CQs. A LFQMT lookup uses the 12 or 10 lsbits of the LFQID as an index into the table, and the lookup provides 3 result items: a 9 or 7 bit CQID, a 12 or 10 bit DCTIDX, and a 4 bit CCGID. The CQID obtained from the table is then used to perform the enqueue onto the requested CEETM CQ.

The DCTIDX is stored in the CQ alongside the FD that was enqueued, and is used to provide the dequeue context associated with frames that are enqueued using this LFQID, see [Section 3.3.20.10, “CEETM dequeue context.”](#) Multiple entries in the LFQMT can be configured with the same CQID, but with different DCTIDX values, allowing a single class queue to carry frames with different dequeue contexts.

The CCGID may be used by the enqueue logic to make a congestion discard decision, see [Section 3.3.20.11, “CEETM Class Congestion Management and Avoidance \(CCM\).”](#)

### 3.3.20.6 CEETM Class Queue Channel and Class Scheduler

[Figure 3-176., “CEETM Class Queue Channel”](#) shows a more complete representation of an example class queue channel including the class scheduler and class queues. The configuration shown in the figure is an

example only since different configurations of the priority of the weighted scheduling (WBFS) groups versus the strict priority class queues are possible.

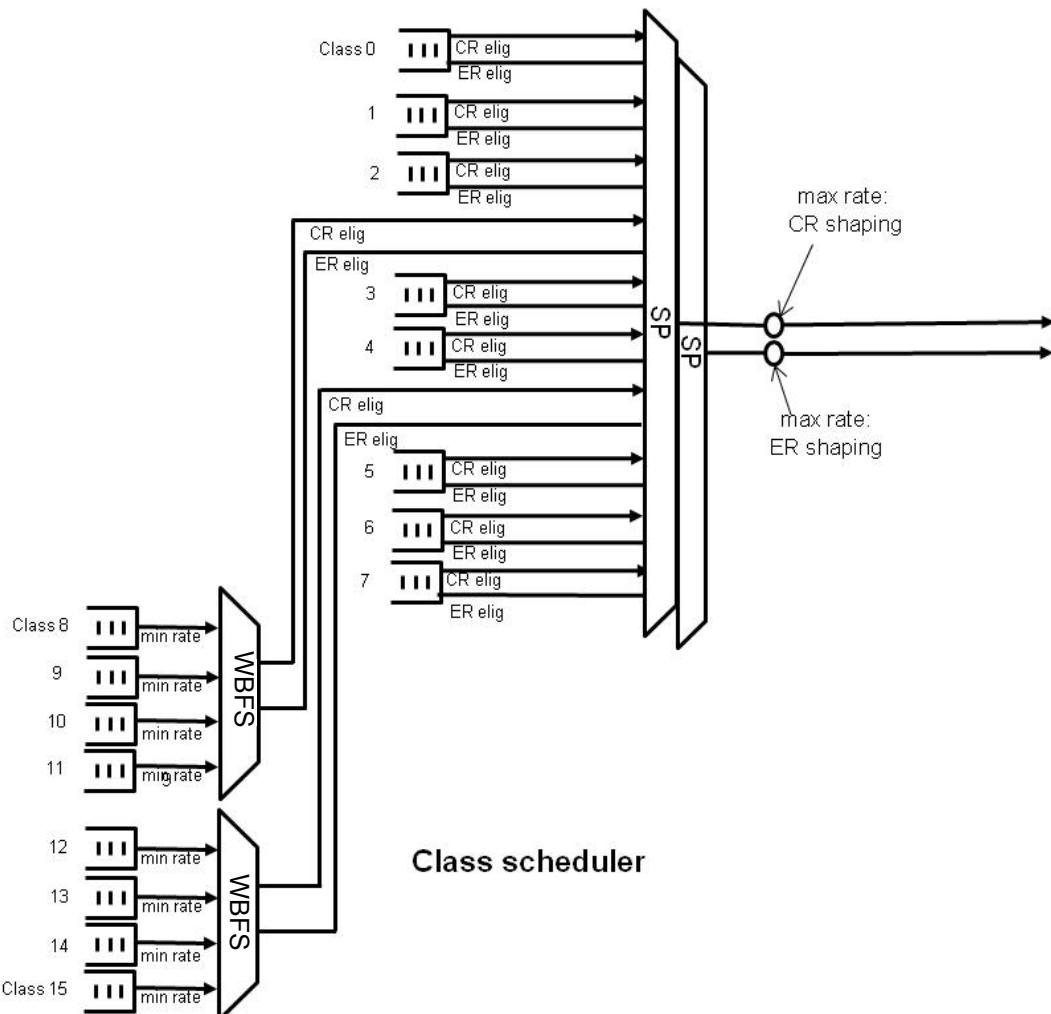


Figure 3-176. CEETM Class Queue Channel

The class queues (CQs) shown in this figure represent the only buffer points in the CEETM. Frame sources (such as a core via a software portal) enqueue frames to these CQs, and when scheduling decisions are made the frames are removed from these queues.

Sixteen class queues are shown with 8 classes scheduled using strict priority (labelled SP in the figure) and 2 groups of 4 classes scheduled using a weighted scheduling algorithm (WBFS). The eight strict priority CQs in a CQ channel are numbered CQ0 to CQ7, with CQ0 being the highest priority and CQ7 the lowest. The two WBFS groups are identified as group A and group B, group A contains four CQ numbered CQ8 to CQ11, and group B contains 4 CQ numbered CQ12 to CQ15.

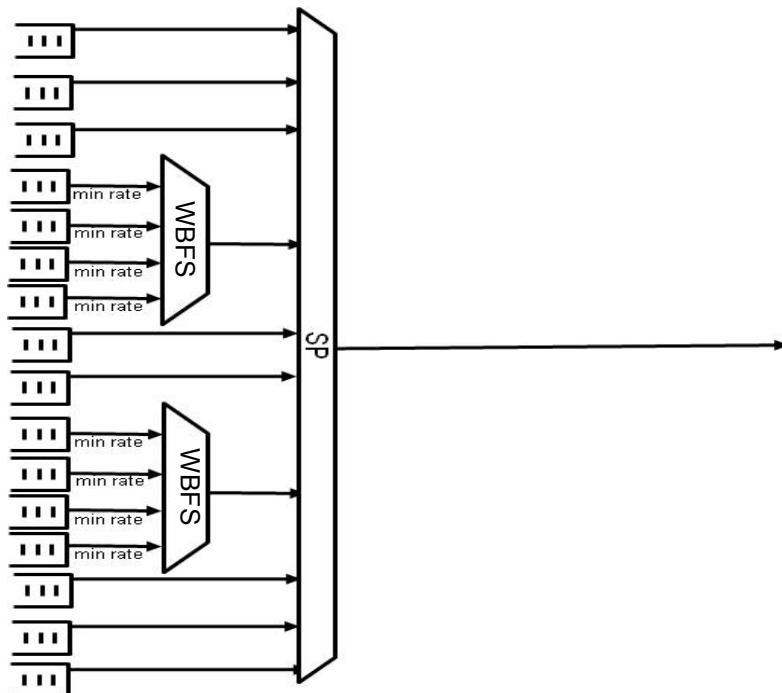
In the figure the class queues at the top of the diagram have the highest priority for scheduling. If there is any traffic on the top most class queue (CQ0) then that traffic is selected for transmission before any other traffic on this channel. Selection of frames to send within a group of 4 weighted class queues is performed using the weighted scheduling algorithm but only if there is no traffic available to send of a higher strict priority.

As noted the class queue configuration within the channel shown in this figure is just an example. The two WBFS groups can be inserted anywhere in the strict priority order except above CQ0, CQ0 is always the highest priority class queue. Also, the two groups of 4 WBFS queues can be combined into a single group of 8. Traffic at a lower priority in the strict priority order will only be transmitted when there is no traffic available at a higher priority. Class scheduler configuration is done via a software portal and is described in [Section 3.3.9.7.7, “CEETM Class Scheduler Configure.”](#)

The output of the class scheduler may be subjected to two shapers (rate limiters). One represents the Committed Rate (CR) for the class queue channel and the other Excess Rate (ER).

Each Strict Priority class and WBFS group can be configured to be eligible for either available bandwidth (rate) from the CR, the ER, or both, or neither. Functionally the strict priority scheduling in the shaped class scheduler can be thought of as two parallel schedulers, one for CR, the other for ER. Class queues or WBFS groups are only considered by the SP scheduler corresponding to the shapers which they are configured to be eligible for: CR, ER, or both. Each class scheduler can be thought of as having two outputs, one for CR and the other for ER. A frame is available at a given output only if the corresponding shaper has available bandwidth (its bucket has tokens) and at least one of the class queues eligible for that rate has a frame available.

It is also possible to configure a class queue scheduler to have no shaping or rate limiting. [Figure 3-177., “CEETM unshaped class queue channel”](#) shows an example of such a channel. Such a class queue channel has been configured to be unshaped, this setting is part of the channel scheduler mapping configuration, see [Section 3.3.9.7.9, “CEETM Channel Mapping Configure.”](#)



**Figure 3-177. CEETM unshaped class queue channel**

### 3.3.20.6.1 CEETM Weighted Scheduling among Grouped Classes

Grouped classes with data to send will receive a weighed portion of the bandwidth available to the group. The bandwidth will be divided according to a max-min. fairness principle. Details of the algorithm can be found in [Section 3.3.20.15.1, “Weighted Bandwidth Fair Scheduling”](#)

Weights may be specified for each class as a value from 1 to 248, in pseudo logarithmic steps of about 1.5%. The weights are specified as an 8 bit code during configuration (see [Section 3.3.9.7.7, “CEETM Class Scheduler Configure”](#)) with a 5 bit y value and a 3 bit x value, and the mapping from weight code to weight is shown in the following table.

**Table 3-168. WBFS weight codes to weight mapping**

Configuration Codes {y,x} : Effective Weight (for Grouped Classes Scheduled with WBFS)							
{0,0} : 1.00	{1,0} : 1.02	{2,0} : 1.03	{3,0} : 1.05	{4,0} : 1.07	{5,0} : 1.08	{6,0} : 1.10	{7,0} : 1.12
{8,0} : 1.14	{9,0} : 1.16	{10,0} : 1.19	{11,0} : 1.21	{12,0} : 1.23	{13,0} : 1.25	{14,0} : 1.28	{15,0} : 1.31
{16,0} : 1.33	{17,0} : 1.36	{18,0} : 1.39	{19,0} : 1.42	{20,0} : 1.45	{21,0} : 1.49	{22,0} : 1.52	{23,0} : 1.56
{24,0} : 1.60	{25,0} : 1.64	{26,0} : 1.68	{27,0} : 1.73	{28,0} : 1.78	{29,0} : 1.83	{30,0} : 1.88	{31,0} : 1.94
{0,1} : 2.00	{1,1} : 2.03	{2,1} : 2.06	{3,1} : 2.10	{4,1} : 2.13	{5,1} : 2.17	{6,1} : 2.21	{7,1} : 2.25
{8,1} : 2.29	{9,1} : 2.33	{10,1} : 2.37	{11,1} : 2.42	{12,1} : 2.46	{13,1} : 2.51	{14,1} : 2.56	{15,1} : 2.61
{16,1} : 2.67	{17,1} : 2.72	{18,1} : 2.78	{19,1} : 2.84	{20,1} : 2.91	{21,1} : 2.98	{22,1} : 3.05	{23,1} : 3.12
{24,1} : 3.20	{25,1} : 3.28	{26,1} : 3.37	{27,1} : 3.46	{28,1} : 3.56	{29,1} : 3.66	{30,1} : 3.76	{31,1} : 3.88
{0,2} : 4.00	{1,2} : 4.06	{2,2} : 4.13	{3,2} : 4.20	{4,2} : 4.27	{5,2} : 4.34	{6,2} : 4.41	{7,2} : 4.49
{8,2} : 4.57	{9,2} : 4.65	{10,2} : 4.74	{11,2} : 4.83	{12,2} : 4.92	{13,2} : 5.02	{14,2} : 5.12	{15,2} : 5.22
{16,2} : 5.33	{17,2} : 5.45	{18,2} : 5.57	{19,2} : 5.69	{20,2} : 5.82	{21,2} : 5.95	{22,2} : 6.10	{23,2} : 6.24
{24,2} : 6.40	{25,2} : 6.56	{26,2} : 6.74	{27,2} : 6.92	{28,2} : 7.11	{29,2} : 7.31	{30,2} : 7.53	{31,2} : 7.76
{0,3} : 8.00	{1,3} : 8.13	{2,3} : 8.26	{3,3} : 8.39	{4,3} : 8.53	{5,3} : 8.68	{6,3} : 8.83	{7,3} : 8.98
{8,3} : 9.14	{9,3} : 9.31	{10,3} : 9.48	{11,3} : 9.66	{12,3} : 9.85	{13,3} : 10.0	{14,3} : 10.2	{15,3} : 10.4
{16,3} : 10.7	{17,3} : 10.9	{18,3} : 11.1	{19,3} : 11.4	{20,3} : 11.6	{21,3} : 11.9	{22,3} : 12.	{23,3} : 12.5
{24,3} : 12.8	{25,3} : 13.1	{26,3} : 13.5	{27,3} : 13.8	{28,3} : 14.2	{29,3} : 14.6	{30,3} : 15.1	{31,3} : 15.5
{0,4} : 16.0	{1,4} : 16.3	{2,4} : 16.5	{3,4} : 17.8	{4,4} : 17.1	{5,4} : 17.4	{6,4} : 17.7	{7,4} : 18.0
{8,4} : 18.3	{9,4} : 18.6	{10,4} : 19.0	{11,4} : 19.3	{12,4} : 19.7	{13,4} : 20.1	{14,4} : 20.5	{15,4} : 20.9
{16,4} : 21.3	{17,4} : 21.8	{18,4} : 22.3	{19,4} : 22.8	{20,4} : 23.3	{21,4} : 23.8	{22,4} : 24.4	{23,4} : 25.0
{24,4} : 25.6	{25,4} : 26.3	{26,4} : 27.0	{27,4} : 27.7	{28,4} : 28.4	{29,4} : 29.3	{30,4} : 30.1	{31,4} : 31.0
{0,5} : 32.0	{1,5} : 32.5	{2,5} : 33.0	{3,5} : 33.6	{4,5} : 34.1	{5,5} : 34.7	{6,5} : 35.3	{7,5} : 35.9
{8,5} : 36.6	{9,5} : 37.2	{10,5} : 37.9	{11,5} : 38.6	{12,5} : 39.8	{13,5} : 40.2	{14,5} : 41.0	{15,5} : 41.8
{16,5} : 42.7	{17,5} : 43.6	{18,5} : 44.5	{19,5} : 45.5	{20,5} : 46.6	{21,5} : 47.6	{22,5} : 48.8	{23,5} : 50.0
{24,5} : 51.2	{25,5} : 52.5	{26,5} : 53.0	{27,5} : 55.4	{28,5} : 56.9	{29,5} : 58.5	{30,5} : 60.2	{31,5} : 62.1
{0,6} : 64.0	{1,6} : 65.0	{2,6} : 66.1	{3,6} : 67.2	{4,6} : 68.3	{5,6} : 69.4	{6,6} : 70.6	{7,6} : 71.9
{8,6} : 73.1	{9,6} : 74.5	{10,6} : 75.9	{11,6} : 77.3	{12,6} : 78.8	{13,6} : 80.3	{14,6} : 82.0	{15,6} : 83.6

**Table 3-168. WBFS weight codes to weight mapping**

Configuration Codes {y,x} : Effective Weight (for Grouped Classes Scheduled with WBFS)								
{16,6} : 85.3	{17,6} : 87.2	{18,6} : 89.0	{19,6} : 91.0	{20,6} : 93.1	{21,6} : 95.3	{22,6} : 97.5	{23,6} : 99.9	
{24,6} : 102	{25,6} : 105	{26,6} : 108	{27,6} : 111	{28,6} : 114	{29,6} : 117	{30,6} : 120	{31,6} : 124	
{0,7} : 128	{1,7} : 130	{2,7} : 132	{3,7} : 134	{4,7} : 137	{5,7} : 139	{6,7} : 141	{7,7} : 144	
{8,7} : 146	{9,7} : 149	{10,7} : 152	{11,7} : 155	{12,7} : 158	{13,7} : 161	{14,7} : 164	{15,7} : 167	
{16,7} : 171	{17,7} : 174	{18,7} : 178	{19,7} : 182	{20,7} : 186	{21,7} : 191	{22,7} : 195	{23,7} : 200	
{24,7} : 205	{25,7} : 210	{26,7} : 216	{27,7} : 221	{28,7} : 228	{29,7} : 234	{30,7} : 241	{31,7} : 248	

### 3.3.20.7 CEETM Channel Shapers

The CEETM shapers are token bucket based. Two shapers are provided per class queue channel to support Committed Rate and Excess Rate shaping.

Shaper configuration is done via a software portal and is described in [Section 3.3.9.7.13, “CEETM Shaper Configure.”](#)

Each shaper has an individually configured credit value (tokens) which is periodically added to the shaper’s bucket (accumulated credits). The credit is expressed as a number of bytes, with an 11 bit integer part and 13 bit fractional part. The update period for adding tokens to the shaper buckets is determined by a configured pre-scaler value, see [Section 3.2.4.27, “CEETM Configuration Shaper Pre-Scale Register \(CEETM\\_CFG\\_PRES\).](#)” This credit rate value and the period of the update determine the shaper’s rate.

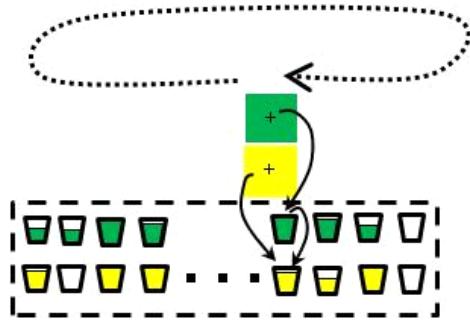
Also, each shaper has an individually configured maximum beyond which credits are not accumulated in it’s bucket. This value is the shaper’s token bucket limit, which is directly related to the maximum burst size<sup>1</sup> that will be allowed out of the shaper.

It should be noted that the channel will be allowed to transmit as much as one full frame plus the configured token credit rate (CRTCR or ERTCR) of the shaper more than the available tokens (see [Section 3.3.20.8.1, “CEETM channel scheduler Shaped Fair Queueing \(ShFQ\)”](#)), therefore it can exceed the configured token bucket limit for the shaper by that amount. Thus the token bucket limit should be configured to be an MTU plus the token credit rate less than the actual desired maximum burst size. It should be further noted that this additional MTU plus token credit rate of burst will not always be used since this is dependent on the size of the last frame sent which reduces the available tokens to zero or less.

The credits to be added to a shaper can be configured to be infinite, by setting its credit rate to all 1’s. In this case the shaper will always have accumulated credits and the shaper’s effective rate will be infinite. The token bucket limit for such a shaper must still be configured according to the maximum burst size desired at the output of the shaper, and the setting of an infinite credit rate results in setting the shaper’s token bucket accumulated credits to its maximum (i.e. token bucket limit) after every transmission from that shaper.

1. Burst size is the maximum amount of data (in bytes) sent as a consecutive burst of back to back frames on the network.

It is possible to configure the CR shapers such that their periodically added tokens, above their configured token bucket limit, are added instead to the ER shaper bucket for the same channel up to the ER shaper's token bucket limit. This overflow of CR tokens to the ER bucket is only possible between pairs of related CR/ER shaper buckets for a class queue channel.



**Figure 3-178. CEETM shaper token bucket update**

Figure 3-178., “CEETM shaper token bucket update” shows pictorially the shaper token buckets being updated. In this diagram the CR bucket (green) has reached its limit and the CR tokens are overflowing to the ER bucket (yellow).

It is possible to configure a class queue channel so that it is not shaped (unshaped). A class queue channel configured in this way is treated as an unshaped class queue channel as shown in Figure 3-177., “CEETM unshaped class queue channel,” and traffic from this channel is given higher priority than any traffic from a shaped channel as shown for the channel at the top of Figure 3-175., “Example CEETM scheduling hierarchy for one logical network interface.”

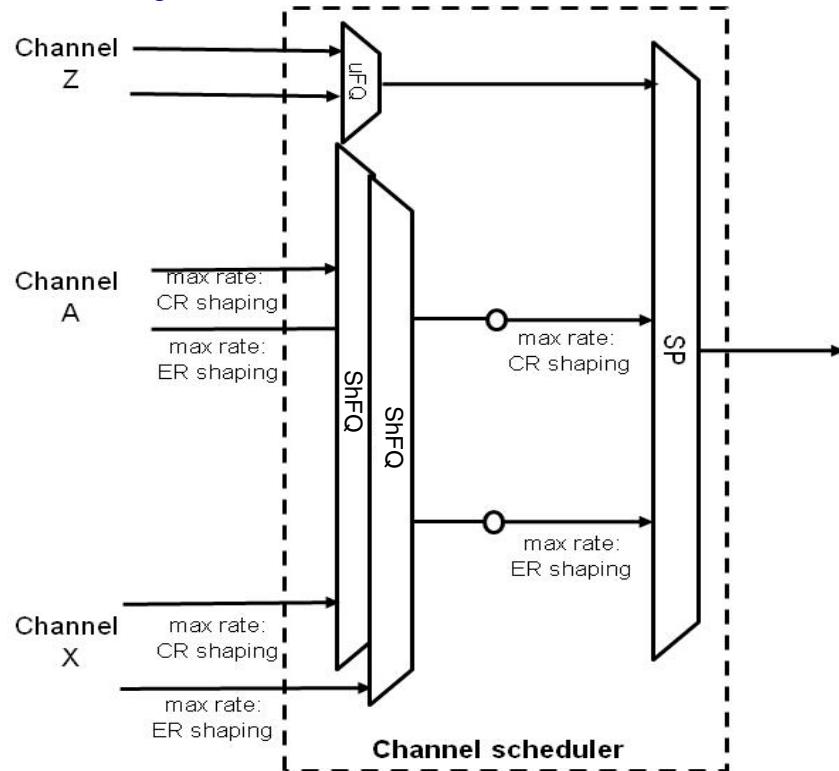
### 3.3.20.8 CEETM Channel Scheduler

The channel scheduler for a LNI selects between frames from one or more CQ channels. Frames may come from class queue channels which are not shaped, from class queue channels which have been shaped at CR, and from class queue channels which have been shaped at ER. The channel scheduler uses a strict priority selection giving highest priority to the unshaped channels, next highest to the CR shaped channels, and lowest priority to ER shaped channels. Figure 3-179., “CEETM Channel Scheduler” shows a channel scheduler which has been configured with 3 channels, two of which are shaped and one which is unshaped.

When selecting within class queue channels which have been shaped, the channel scheduler uses a fair queueing algorithm which interacts with the channel shapers, referred to as shaped fair queuing (ShFQ).

When selecting within the class queue channels which are unshaped the channel scheduler uses an algorithm referred to as unshaped fair queueing (uFQ), which is a simplified form of the ShFQ algorithm

in that unshaped channels are always considered time eligible. See [Section 3.3.20.8.1, “CEETM channel scheduler Shaped Fair Queueing \(ShFQ\).”](#)



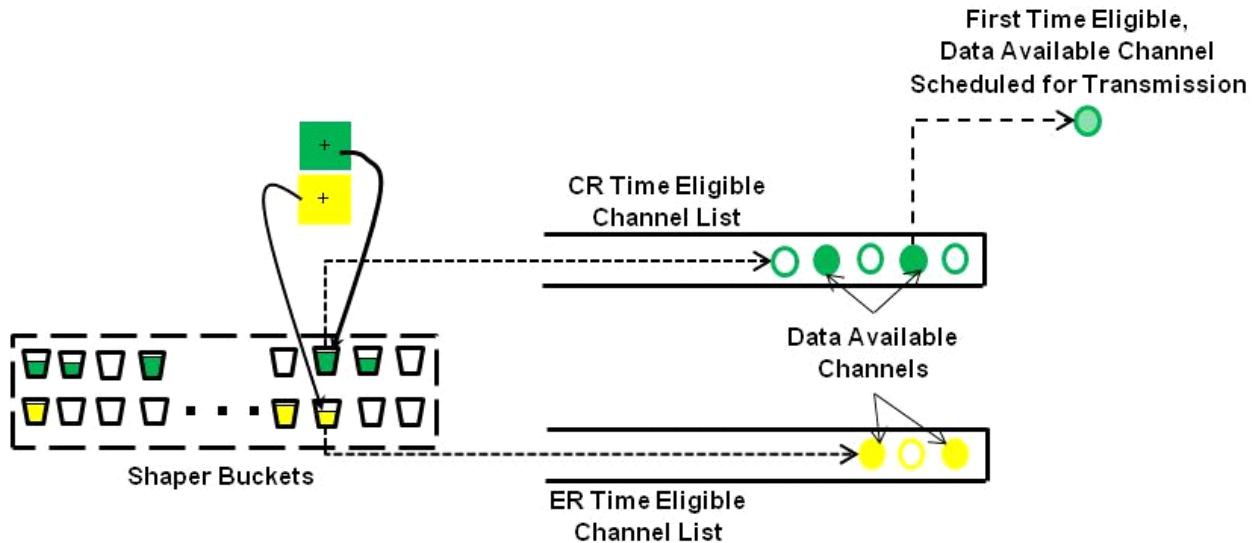
**Figure 3-179. CEETM Channel Scheduler**

Note that the LNI provides the capability to shape the output of the channel scheduler, which is the aggregate of the traffic from shaped class queue channels.

Traffic from class queue channels which are unshaped bypass the LNI shapers. That is this traffic is not subject to any shaping. Furthermore, it is given higher priority than traffic which undergoes shaping at either the channel shaper level or the LNI shaper level. Note that an unshaped CQ channel is not the same as a shaped channel whose credit rate is set to be infinite. Traffic that has been subject to a channel shaper with infinite rate is still considered to be shaped and is subject to the subsequent LNI shaper.

### 3.3.20.8.1 CEETM channel scheduler Shaped Fair Queueing (ShFQ)

When scheduling from shaped class queue channels the channel scheduler interacts with the channel shapers as well as the class schedulers.



**Figure 3-180. CEETM channel scheduler shaped fair queueing (ShFQ)**

The class schedulers provide an indication that there is data available on at least one of their queues. This is known as data availability. If a class queue channel is shaped then there are 2 different data availability indications: one for queues that are being considered for CR and the other for queues that are being considered for ER. Note that queues can be included for both data availability indications.

Shaping decisions in the channel shaper are made without knowledge of the length of the frame which will be provided by the class scheduler. Thus the shaper bucket (accumulated credits) may not be sufficient to send the frame. In this case the bucket may go into deficit (go negative). The channel shaper provides an indication to the channel scheduler when it has some available credits: i.e. when it is not in deficit and has 0 or more positive accumulated credits. This is known as time eligibility. During the time that it is in deficit credits continue to be added to the channel shaper's bucket. Until the channel shaper has some number of positive accumulated credits it is not "time eligible."

To be considered for inclusion in a scheduling round by the channel scheduler a channel must be both data available and time eligible.

The channel scheduler tracks the order in which channel shapers become time eligible. When selecting a class queue channel the channel scheduler selects the channel which has been time eligible the longest and which is also data available.

At the point that the class queue channel is selected the channel scheduler takes all the available tokens from the channel shaper's bucket, and uses only those tokens for making its transmit decisions. As frames are dequeued to be transmitted these tokens are reduced by the number of bytes in each frame being transmitted. As long as there are available tokens and available data the same class queue channel will be the source of the dequeued frames. Removing the tokens from the bucket rather than operating on the

bucket directly avoids the possibility of sending more than the shaper's token bucket limit due to the bucket being updated while the channel is being used to transmit.

When the channel no longer has data available or the tokens are all used the scheduler will select a new channel to dequeue from. At this point the remaining tokens are added back into the shaper bucket. This may involve adding a deficit back into the bucket.

If a channel is time eligible but not data available then it cannot be selected by the channel scheduler but it does not lose its place in the time eligible order. When it does become data eligible it will be selected for transmission before any channel which has more recently had an opportunity to transmit a frame.

Similarly if a channel is data available but not time eligible it can not be selected by the channel scheduler for transmission. When the channel does become time eligible it is added to the end of the list of time eligible channels and will be selected for transmission after any of the channels ahead of it in the list which have data available.

Channels whose shapers have an infinite credit rate configured, or channels which have been configured as unshaped, are treated as a special case within the ShFQ algorithm. Such channels are always time eligible, they are simply added to the end of the time eligible list when they have finished sending any available data, and their token bucket is re-filled to the configured token bucket limit (TBL). As such, the configured TBL in such channels becomes a fair queueing weight which controls how much each of the channels is allowed to send in turn. Unshaped channels are included in the CR time eligible list, and thus use the configured CRTBL value as their fair queueing weight. A TBL of 0 defaults to sending one frame at a time from each such channel. Note that this is not exactly the same as round-robin, since the scheduling does not follow a pre-determined order, rather it follows the order in which channels were placed in the time eligible list.

### **3.3.20.8.2 CEETM LNI shapers**

The LNI provides two rate shapers that can be used to shape the traffic from all class queue channels which are shaped on input to the channel scheduler. These shapers are identical in all respects to the channel shapers, see [Section 3.3.20.7, “CEETM Channel Shapers.”](#)

The LNI shapers are used to shape or rate limit the aggregate of the class queue channels which have each been shaped.

These two shapers provide CR and ER shaping. These rates are applied only to frames shaped by the corresponding shaper at the channel shaper level. For instance, if a frame was scheduled due to available Committed Rate at the channel shaper then it can only be scheduled due to available Committed Rate at the LNI shaper.

### **3.3.20.9 CEETM dequeue requests and scheduling decision flow**

As shown in [Figure 3-175., “Example CEETM scheduling hierarchy for one logical network interface”](#) the only data buffering in CEETM is in class queues.

Data availability indications are propagated from the class queues towards the channel scheduler in the LNI. Data availability indications are filtered by shapers. If a class queue channel has frames on one of its queues but is shaped and its shapers have no available credits, its data available signal will not be passed

on to the channel scheduler. Since a class queue can be eligible for both CR and ER data being available on a single queue can result in two data available indications being propagated forward.

Likewise, data availability indications from the channel schedulers of each LNI are passed on to the DCP sub-portal configured to use that LNI, and result in a data availability signal for each sub-portal (i.e. each LNI) being passed on to the hardware module using this DCP.

Scheduling decisions are driven by dequeue requests from the hardware module (such as FMan) using a DCP. If this DCP supports CEETM, and if the sub-portal on which the dequeue request is received has been configured in CEETM scheduling mode, then the CEETM scheduling and shaping hierarchy will be applied to generate the response to that dequeue request.

When a dequeue request is received the scheduling decision is made based on the state of the shapers and class queues at that time. Although the CEETM scheduling is shown as a progression of scheduling/shaping decisions logically it can be thought that all scheduling and shaping decisions are made simultaneously.

### **3.3.20.10 CEETM dequeue context**

When dequeues are performed from a regular FQ, dequeue responses delivered to a hardware module using a DCP contain, in addition to the FD and other information, two specific pieces of context information which are configured in the FQD. These are the Context\_A and Context\_B fields of the FQD, and are described in [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\).”](#)

This dequeue response context information can be used by the dequeuer to control how it processes the frame. This supports virtualization since it allows the dequeuer to support per FQ configuration either directly controlled by the Context\_A and Context\_B values or indirectly through a data structure referenced by using some portion of the dequeue context as an address.

The same dequeue context is provided when frames are dequeued from a DCP sub-portal in CEETM scheduling mode. In CEETM scheduling mode, the Context\_A and Context\_B values are not configured in the CQD, since we want the ability for a single class queue to carry a mix of frames with different dequeue contexts. Instead, each frame enqueued onto a class queue is appended with a DCT index (DCTIDX, see [Section 3.3.20.5, “Enqueuing onto a CEETM Class Queue”](#)), and this DCTIDX is used when the frame is dequeued to perform a lookup into the CEETM dequeue context table (DCT), which then provides the Context\_A and Context\_B values that will be issued in the dequeue response with that frame.

The DCT is indexed by DCTIDX and must contain a valid pair of Context\_A and Context\_B values for every DCTIDX that is configured in the LFQMT, and which is used during enqueuees to a CQ. The software portal provides commands to allow software to configure and query the DCT, see [Section 3.3.9.7.5, “CEETM Dequeue Context Table \(DCT\) Configure.”](#)

### **3.3.20.11 CEETM Class Congestion Management and Avoidance (CCM)**

When enqueueing to CEETM class queues, CEETM supports both Weighted Random Early Discard (WRED) and tail drop congestion management functionality which is equivalent in capability to that provided by FQ congestion groups and described in [Section 3.3.14, “Congestion Management and Avoidance.”](#) This feature of CEETM is referred to as CCM for short. Some of the features and capabilities

of CEETM CCM are different than those provided by FQ congestion groups, these differences are described below.

### CEETM CCM features:

- 512 (for 32 channel CEETM) or 128 (for 8 channel CEETM) Class Congestion Groups (CCG)
- CCG are divided into channels. Each channel contains 16 CCG, and each CCG channel is tied one-to-one with a CQ channel. As such, the CQ channel ID (upper 5 or 3 bits of the 9or 7 bit CQID) directly indexes the CCG channel.
- The CCG to be used for a particular CQ can be assigned to any of the 16 CCG within that channel. This allows each CQ to have its own CCG, or allows several (or all) CQ in the same channel to share the same CCG, whichever is required.
- The CCG within a channel to be used by each CQ is configured in the CQD of the CQ, the CCGID in the CQD indexes the CCG within the channel that is used by dequeues from the CQ. See [Section 3.3.9.7.3, “CEETM Class Queue \(CQ\) Configure.”](#)
- The CCG within a channel to be used by each CQ must also be present in the LFQMT, the CCGID in the LFQMT indexes the CCG within the channel that is used by enqueues to the CQ. When a LFQMT entry is configured, the CCGID value to be written to that entry is read from the CQD, to ensure that the two values match. See [Section 3.3.9.7.1, “CEETM Logical FQ Mapping Table \(LFQMT\) Configure.”](#)

As mentioned above, the CCG to be used by a CQ must be configured (unless the default configuration is sufficient) in the CQD, and if a CQ configure command is used it must be followed by the configuration of any LFQMT entries which use that CQID. Configuration of the CQ followed by LFQMT is required to prevent a mis-configuration of the CCGID values between the CQD and the LFQMT. When an enqueue is performed to a valid CEETM Lfqid, the CCGID to be used for the enqueue is pulled from the LFQMT, and used to perform the congestion avoidance check and update the CCG. If the enqueue is not rejected for congestion avoidance, it proceeds and causes the CQD to be read for the purpose of completing the enqueue. At this point the CEETM logic checks to make sure that the CCGID found in the CQD matches the CCGID that was read from the LFQMT for the enqueue. If the two CCGID values are not the same, a IECE configuration error has been detected and can cause an error interrupt to be asserted, see [Section 3.2.4.51, “QMan Error Interrupt Status Register \(QMAN\\_ERR\\_ISR\).”](#)

The WRED, tail drop, and enqueue rejection functionality provided by CCM is the same as that provided for FQ congestion groups, see [Section 3.3.14.2, “Weighted Random Early Discard \(WRED\),”](#) [Section 3.3.14.3, “Congestion Group Tail Drop,”](#) and [Section 3.3.14.5, “Enqueue Rejections”](#) for a description of these features.

### Differences between FQ congestion groups and CEETM CCM:

- Congestion notifications from CEETM CCM can be sent to both software and DCP portals, as with FQ congestion groups. There are minor differences in how these notifications are configured and used, see [Section 3.3.20.11.1, “CEETM Congestion State Change Notifications \(CEETM CSCN\).”](#)
- There is no equivalent to FQ tail drop provided for class queues, therefore [Section 3.3.14.4, “FQ Tail Drop”](#) does not apply for CEETM CCM.
- Overhead accounting length (OAL) for CCM is configured in the CCG itself rather than in CQ. In FQ/WQ scheduling mode the OAL is configured in each FQ.

- CEETM CCM provides separate thresholds for congestion entrance and exit, and an optional third threshold for tail drop. In FQ/WQ scheduling mode there is a single threshold for congestion notifications and for tail drop, with congestion exit fixed at 7/8 of the congestion entrance threshold.
- A Class Congestion Group Record (CCGR) used by CEETM CCM carries many of the same fields as a Congestion Group Record (CGR) used by FQ congestion groups, however the two records are different. See [Section 3.3.20.11.2, “CEETM Class Congestion Group Record \(CCGR\).”](#)

### **3.3.20.11.1 CEETM Congestion State Change Notifications (CEETM CSCN)**

CEETM CCM supports congestion state change notifications (CSCN) to software portals, and to DCP portals. The functionality of CEETM CSCN is the same as that used by FQ congestion groups and described in [Section 3.3.14.6, “Congestion State Change Notifications \(CSCN\).”](#) One key difference is that for CEETM CSCN both the congestion entrance and exit thresholds are programmable, in FQ congestion groups the entrance threshold is programmable and the exit threshold is fixed at 7/8 of the entrance threshold.

The differences between CEETM CSCN and FQ congestion group CSCN with respect to software portals are summarized as follows:

- A separate bit for CEETM CSCN interrupt is provided in the ISR of each software portal, see the CCSCI bit in [Section 3.2.3.21, “QCSP Interrupt Status Register \(QCSPi\\_ISR\).”](#)
- CEETM provides a different command for querying the congestion state of CCGs, see [Section 3.3.9.7.19, “CEETM Query Congestion State.”](#)

CEETM CSCN sent from QMan to a hardware block connected to a DCP portal are sent using the same dedicated interface signals that are used for sending CSCN from FQ congestion groups, see [Section 3.3.14.6, “Congestion State Change Notifications \(CSCN\).”](#) The configuration of which DCP portal(s) should receive CSCN notifications from a CCG is done in the same way as for FQ congestion groups, i.e. using the CSCN\_TARG\_DCP field of the CCGR, see [Section 3.3.20.11.2, “CEETM Class Congestion Group Record \(CCGR\).”](#) However when configuring CEETM CSCN in a CCGR, there is an additional field that needs to be configured, see the description of the CSCN\_DCP\_VCGID field in the CCGR.

### **3.3.20.11.2 CEETM Class Congestion Group Record (CCGR)**

The configuration and state information for each CCG is kept in a class congestion group record (CCGR) stored in internal memory. The table below identifies all of the fields that are present in a CCGR. The software portal provides commands to configure and query the CCGRs, see [Section 3.3.9.7.17, “CEETM Class Congestion Group Record \(CCGR\) CM Configure.”](#) After reset, all fields in all CCGRs are set to 0.

**Table 3-169. CEETM Class Congestion Group Record (CCGR)**

CGR Field	CGR Field Description
31:0 WR_PARM_G	WRED Green Parameters Bits 0-7: MA Bits 8-12: Mn $\text{MaxTH} = \text{MA} * 2^{\text{Mn}}$ Bits 13-19: SA (value must be between 64-127) Bits 20-25: Sn (value must be between 7-63) $\text{Slope} = \text{SA} * 1/2^{\text{Sn}}$ (can also be written as $\text{Slope} = \text{SA} * 2^{-\text{Sn}}$ ) Bits 26-31: Pn $\text{MaxP} = (\text{Pn}+1) * 4$
63:32 WR_PARM_Y	WRED Yellow Parameters See WR_PARM_G
95:64 WR_PARM_R	WRED Red Parameters See WR_PARM_G
96 WR_EN_G	WRED Green Enable - Use WRED (random probability vs. Curve@ Average) for 'Green' colored enqueues
97 WR_EN_Y	WRED Yellow Enable - Use WRED (random probability vs. Curve@ Average) for 'Yellow' colored enqueues
98 WR_EN_R	WRED Red Enable - Use WRED (random probability vs. Curve@ Average) for 'Red' colored enqueues
99 TD_EN	Tail Drop Enable - If an enqueue is attempted when this bit is set to 1, and the tail drop condition selected by TD_MODE is satisfied, the CCM will tell the producer to drop the frame.
112:100 CS_THRES	Congestion State Entrance Threshold. Bits 0-7: TA (value should be > 0. If 0 is used, the CS bit can never transition from 1 to 0) Bits 8-12: Tn $\text{CS Entrance Threshold} = \text{TA} * 2^{\text{Tn}}$
113 CSCN_EN	Congestion State Change Notification (CSCN) Enable - If this bit is asserted, CSCN will be sent to the portals indicated by CSCN_TARG_SWP and CSCN_TARG_DCP when this congestion group both enters and exits congestion. Congestion State entrance and exit thresholds are used for the CSCN calculation.
121:114 OAL	Overhead Accounting Length This is an 8-bit, 2's complement value (range -128 to +127) representing a fixed per-frame overhead to be added to the actual length of a frame when performing WRED and tail drop calculations and threshold comparisons. For each enqueue and dequeue operation, an overhead accounted frame length (OAFL) is calculated using the actual frame length (AFL) and the OAL value as follows: if (AFL > 16383) then OAFL = AFL, overhead accounting effect is negligible else if (AFL + OAL < 0) then OAFL = 0, negative OAFL is capped at 0 else OAFL = AFL + OAL.
122 MODE	Class Congestion Group Mode, configured by software. 0 = Perform congestion avoidance based on byte count 1 = Perform congestion avoidance based on frame count

**Table 3-169. CEETM Class Congestion Group Record (CCGR) (continued)**

CGR Field	CGR Field Description
123 CS	<p>Congestion State relative to Congestion State Thresholds  Used for the purpose of CSCN notifications, and for CS Tail Drop (if TD_MODE = 0). See <a href="#">Section 3.3.20.11.1, “CEETM Congestion State Change Notifications (CEETM CSCN).”</a> Determined by comparing the instantaneous byte (or frame) count to the Congestion State entrance and exit thresholds. This bit is set to 1 on a positive greater-than crossing of the entrance threshold. This bit is set to 0 on a negative less-than-or-equal-to crossing of the exit threshold.</p> <p>0 = This Congestion Group is currently not congested relative to Congestion State Thresholds.  1 = This Congestion Group is currently congested relative to Congestion State Thresholds.</p>
162:124 I_CNT	<p>Class Congestion Group Instantaneous Count.  Number of bytes (or frames, if MODE is set to 1) currently in use in all CQ that belong to this Class Congestion Group.</p>
201:163 A_CNT	<p>Class Congestion Group Average Count.  This is a running average of the number of bytes (or frames, if MODE is set to 1) used in all CQ that belong to this Class Congestion Group. This value is maintained by hardware and is updated on every enqueue and dequeue processed by CEETM. Note that the averaging continues to operate normally even if the I_CNT returns to 0 at any point in time.  The running average is calculated as:</p> $\begin{aligned} A_{\text{CNT}}_{\text{delta}} &= A_{\text{CNT}}_{\text{current}} - I_{\text{CNT}}_{\text{current}} \\ \text{TIMESTAMP}_{\text{delta}} &= \text{CurrentTime} - \text{TIMESTAMP} \\ A_{\text{CNT}}_{\text{newdelta}} &\approx A_{\text{CNT}}_{\text{delta}} \times 2^{-\text{TIMESTAMP}_{\text{delta}}/128} \quad \text{--- approximating exponential decay} \\ A_{\text{CNT}}_{\text{new}} &= I_{\text{CNT}}_{\text{current}} + A_{\text{CNT}}_{\text{newdelta}} \end{aligned}$
213:202 TIMESTAMP	<p>Time stamp  - A time stamp of when the last update to this group happened. Used in calculating the average count for this group. The time stamp ‘ticks’ when the pre-scaler configured in CM_CFG (see <a href="#">Section 3.2.4.25, “CM Configuration Register (CM_CFG)”</a>) rolls over.</p>
226:214 CS_THRES_X	<p>Congestion State Exit Threshold.  Bits 0-7: TA  Bits 8-12: Tn  CS Exit Threshold = TA * 2^Tn</p>
239:227 TD_THRES	<p>Tail Drop Threshold. Valid only if TD_MODE = 1.  Bits 0-7: TA  Bits 8-12: Tn  Tail Drop Threshold = TA * 2^Tn</p>
240 TD_MODE	<p>Tail Drop Mode. Valid only if TD_EN = 1.  0 = Tail drop occurs if an enqueue is attempted when the CS bit is 1. TD_THRES is not used.  1 = Tail drop occurs if an enqueue is attempted that would cause I_CNT to exceed the threshold configured in TD_THRES.</p>
241 CSCN_DCP_LSN	<p>Congestion State Change Notification (CSCN) for DCP portals, Last State Notified.  Used together with CSCN_TARG_DCP to decide when to send a CSCN to a DCP portal.  Whenever an operation (enqueue or dequeue) occurs on a class congestion group, QMan attempts to send a CSCN to one or more DCP portals if CS is not equal to CSCN_DCP_LSN and if one or more bits in CSCN_TARG_DCP are set. This includes an operation that changes the state of the CS bit.  If the CSCN is sent successfully, then CSCN_DCP_LSN is set equal to CS. If the CSCN cannot be sent (for example due to a full FIFO on the DCP interface) then CSCN_DCP_LSN is left unchanged, and QMan will re-try the CSCN send attempt when the next operation on this class congestion group occurs.</p>

**Table 3-169. CEETM Class Congestion Group Record (CCGR) (continued)**

CGR Field	CGR Field Description
249:242 CSCN_DCP_VCGID	<p>Congestion State Change Notification (CSCN) for DCP portals, Virtual CGID. The hardware block connected to a DCP portal can be notified of congestion on any of up to 256 congestion groups, and the CSCN sent on the DCP portal carries an 8 bit virtual congestion group identifier (VCGID).</p> <p>For a CSCN sent from a FQ congestion group, there is a 1:1 correspondence between the VCGID sent in the CSCN and the CGID of the group which originated the CSCN, see <a href="#">Section 3.3.14.6, “Congestion State Change Notifications (CSCN).”</a></p> <p>For a CSCN sent from a class congestion group, the VCGID sent to a DCP portal is taken from this field. This allows the hardware block connected to a DCP portal to receive CSCNs from both FQ congestion groups and class congestion groups. The software which configures these congestion groups should ensure that the VCGID is unique for CSCNs sent from different congestion groups, that is, if one or more FQ congestion groups are configured to send CSCNs to a particular DCP portal, then the CGIDs of those congestion groups should not be used as the VCGID value in any class congestion groups which are also configured to send CSCNs to that same DCP portal.</p>
297:250 REJ_BYT_CNT	<p>Rejected Byte Count.</p> <p>This is a cumulative count of all bytes in frames rejected in CQs belonging to this congestion group, due to either WRED or tail drop. This count can be queried and cleared (see <a href="#">Section 3.3.9.7.21, “CEETM Statistics Query/Write”</a>), and can also be written for verification purposes. If this count reaches its maximum value it will stick at that value until cleared or written.</p> <p>This count is valid independent of the MODE setting of this CCGR, and does not include the OAL.</p>
337:298 REJ_FRM_CNT	<p>Rejected Frame Count.</p> <p>This is a cumulative count of all frames rejected in CQs belonging to this congestion group, due to either WRED or tail drop. This count can be queried and cleared (see <a href="#">Section 3.3.9.7.21, “CEETM Statistics Query/Write”</a>), and can also be written for verification purposes. If this count reaches its maximum value it will stick at that value until cleared or written.</p> <p>This count is valid independent of the MODE setting of this CCGR, and does not include the OAL.</p>
num_dcp_portals_max+1+ 338 : 338 CSCN_TARG_DCP	<p>Congestion State Change Notification (CSCN) Target for DCP portals</p> <p>A per-portal bit mask where the following bits notify the corresponding portal.</p> <p>bit 338 = Send CSCN notifications for this Class Congestion Group to DCP portal 0.</p> <p>...</p> <p>bit 338+n = Send CSCN notifications for this Class Congestion Group to DCP portal n.</p>
num_sw_portals-1 +num_dcp_portals_max+338 : num_dcp_portals_max+338 CSCN_TARG_SWP	<p>Congestion State Change Notification (CSCN) Target for software portals</p> <p>A per-portal bit mask where the following bits notify the corresponding portal.</p> <p>bit num_dcp_portals_max+338= Send CSCN notifications for this CCG to software portal 0.</p> <p>...</p> <p>bit num_dcp_portals_max+338+n = Send CSCN notifications for this CCG to software portal n.</p>

### 3.3.20.12 CEETM Sub-Portal Mapping and Physical Interface Switchover

Each DCP sub-portal using CEETM is configured to be associated with a specific LNI. This configuration is done via a software portal, see [Section 3.3.9.7.11, “CEETM Sub-Portal Mapping Configure.”](#)

The intent is that each sub-portal on which CEETM is enabled should be connected to a different LNI, that is, only 1 to 1 connections from sub-portal to LNI should be used. If 2 or more sub-portals are configured

to draw frames from the same LNI, all frames will be dequeued successfully, however frames from the same CQ channel can and likely will emerge on different sub-portals (i.e. different Ethernet links), which is generally not a desirable outcome.

The DCP sub-portals in QMan contain specific optimizations that are intended for use with 10 GE links, in both CEETM scheduling mode and in FQ/WQ scheduling mode. In FQ/WQ scheduling mode, sub-portals 0 and 1 are optimized for 10 GE links, therefore FMan should be configured to use one of those two sub-portals for any 10 GE link on which FQ/WQ scheduling mode is used. In CEETM scheduling mode, LNI 0 and 1 are optimized for 10 GE links, therefore one of those two LNIs should be used for any 10 GE link on which CEETM scheduling mode is used. Note that in CEETM scheduling mode any sub-portal (SP) can be configured to use LNI 0 or LNI 1, therefore if both 10 GE links are using CEETM scheduling mode they can be configured on sub-portals other than 0 and 1, thereby enabling the optimizations in sub-portals 0 and 1 to be used by some other links that are using FQ/WQ scheduling mode and that may benefit from the optimizations.

To support switchover between physical Ethernet interfaces it is possible to switch the LNI from one DCP sub-portal to another, by re-configuring the two sub-portals, without disabling the channel scheduler or any other part of the CEETM LNI. Note that an LNI can only be used by sub-portals on the same DCP, so switchover between Ethernet interfaces on different DCP portals (i.e. different instances of FMan) is not possible.

The DCP sub-portal mapping (which LNI the sub-portal is connected to) should be configured before enabling CEETM scheduling mode on that sub-portal (see [Section 3.2.4.11, “DCP Configuration Registers \(DCPi\\_CFG\)”](#)). When a sub-portal is switched from FQ/WQ scheduling mode to CEETM scheduling mode, dequeues from the WQ channel attached to that sub-portal will immediately stop, therefore any FQs configured with that WQ channel as destination may need to be drained by software after the switchover.

### **3.3.20.13 CEETM traffic class flow control**

When a particular DCP sub-portal is configured for FQ/WQ scheduling mode, QMan provides the ability to flow control each of the 8 different WQs in the channel attached to that sub-portal, this is described in [Section 3.3.5, “Traffic Class \(TC\) Flow Control.”](#) In this case there is a fixed one-to-one mapping from a traffic class (TC) to each WQ in a WQ channel.

CEETM also supports traffic class flow control, but in CEETM scheduling mode each of the 16 class queues within a class queue channel can be mapped to one of the 8 traffic classes that can be flow controlled. This mapping is configured on a per LNI basis, see [Section 3.3.9.7.15, “CEETM Traffic Class Flow Control Configure.”](#) The flow control state is also maintained on a per LNI basis and applies to all class queue channels within that LNI.

For example, if for a given LNI class queues 3, 4, and 5 are mapped to traffic class 0 flow control, when traffic class 0 is flow controlled (set to XOFF) by a traffic class flow control command received by QMan, then all class queues numbered 3, 4, and 5 in all channels of that LNI will be ignored as part of any scheduling decision.

Note that flow control of individual class queues (equivalent to FQ flow control as described in [Section 3.3.6, “Frame Queue \(FQ\) Flow Control”](#)) is not supported in CEETM scheduling mode.

### 3.3.20.14 CEETM Configuration Summary

The commands that are available for configuring the features of CEETM are described in detail in [Section 3.3.9.7, “Customer Edge Egress Traffic Management \(CEETM\) Management Commands.”](#) This section provides a few figures intended to briefly illustrate how these management commands apply to the various elements of CEETM, and the default configuration that exists after reset.

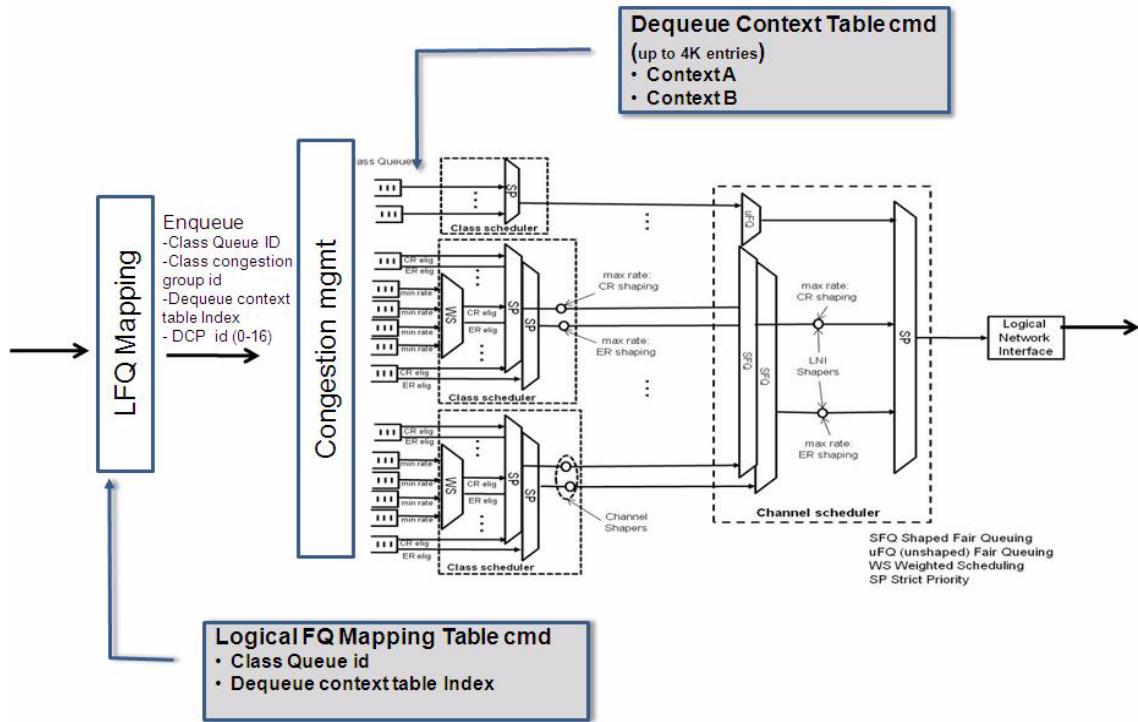


Figure 3-181. CEETM configuration summary 1

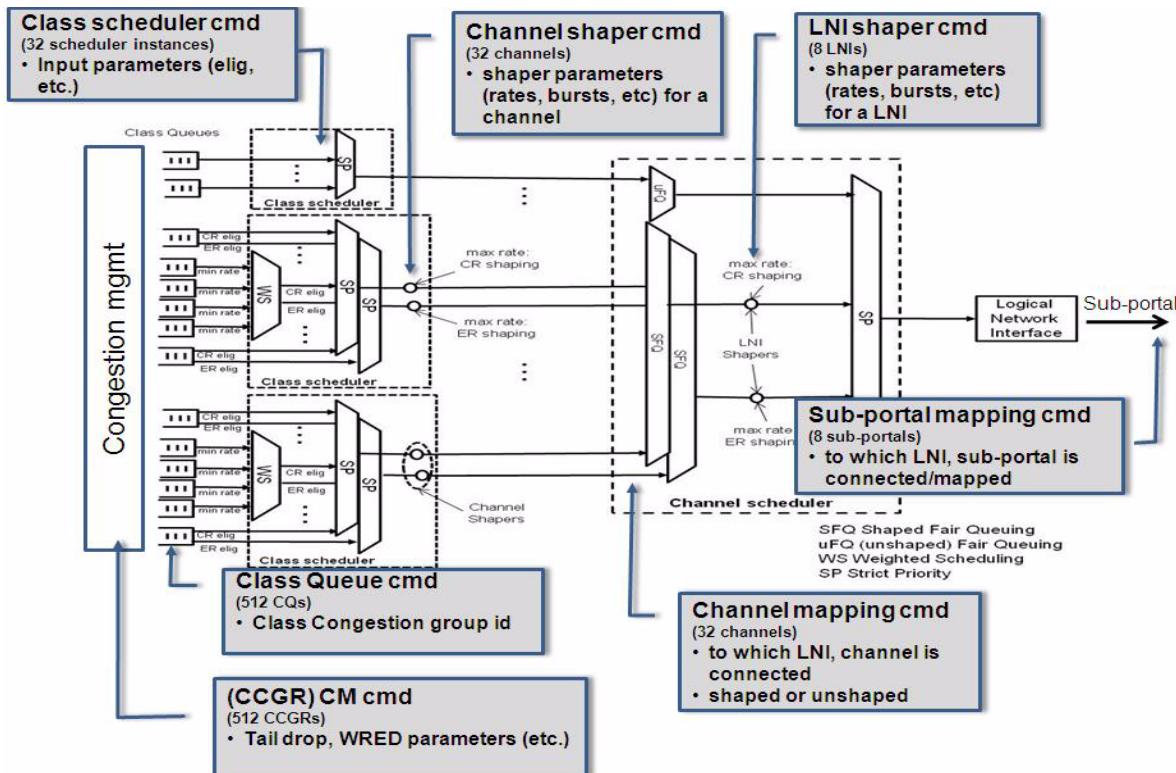


Figure 3-182. CEETM configuration summary 2

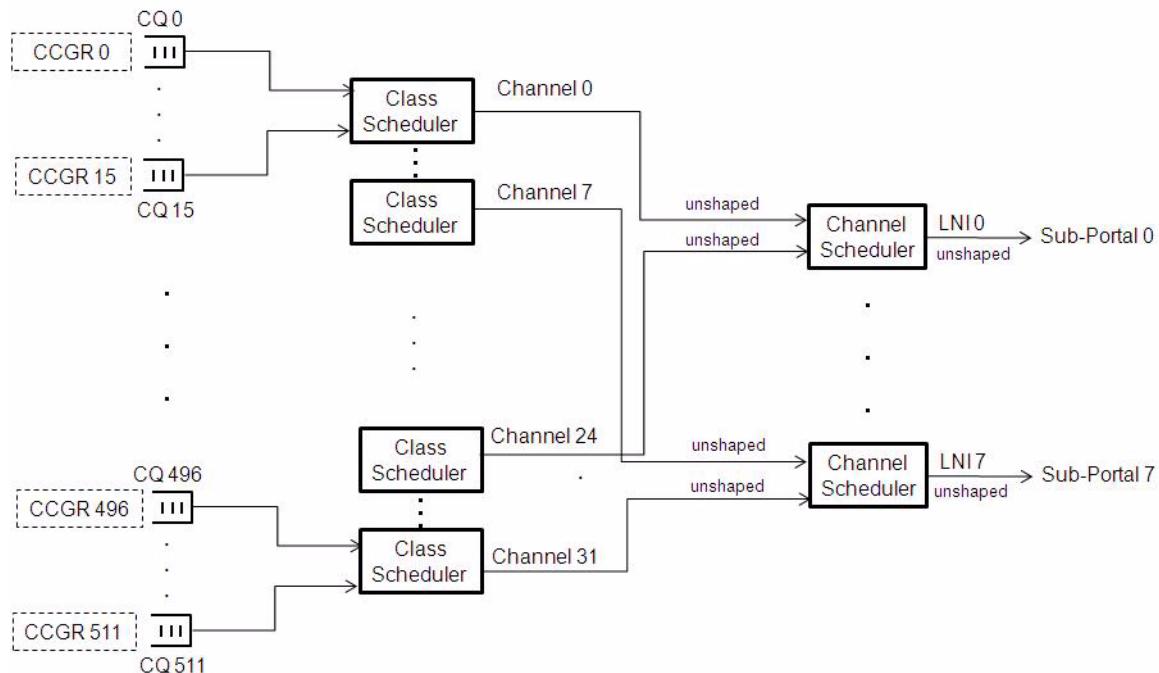


Figure 3-183. CEETM default configuration after reset

### 3.3.20.15 Detailed Description of TM Algorithms used in CEETM

#### 3.3.20.15.1 Weighted Bandwidth Fair Scheduling

Within the Class Scheduler of CEETM a Freescale proprietary algorithm, Weighted Bandwidth Fair Scheduling (WBFS), is used to schedule packets from queues within a priority group (A or B group) such that each gets a “fair” amount of bandwidth made available to that priority group.

Bandwidth (tx opportunities) that is made available to a priority group (A or B group) is fair shared among the class queues of that group in proportion to a “weight” value configured for that class. For example if class “X” has a weight of 5 and class “Y” has a weight of 1, class “X” has a fair share that is 5 times the fair share of class “Y”. The weight values for each class can range from 1 to 248 (in 255 pseudo logarithmic steps), which can be configured differently for each group and channel.

The theory, implementation and operation of the Bandwidth Fair Sharing algorithm (i.e. the unweighted or equal weighted version) is as follows - the relatively simple algorithmic enhancement with WBFS to support weighted fairness shall be describe subsequently.

The premises for fairness for algorithm, BFS is that:

1. available bandwidth is divided and offered equally to all classes.
2. offered bandwidth in excess of a class’s demand is to be re-offered equally to classes with unmet demand.

To understand bandwidth outcome (and arguably the merit of the fairness) of this scheduler it is instructive to apply the premises iteratively to scenarios, such as the example below, where the input demand for each class queue and output transmit bandwidth available to that priority group is steady state.

**Table 3-170. Example of Max-Min Fairness**

	Initial Distribution		First Redistribution		Second Redistribution		Total Bandwidth Attained	
Bandwidth available (all classes)	100 MB/sec		15 MB/sec		2 MB/sec			
#classes with unmet demand	5		3		2			
Bandwidth to be offer to each class	20 MB/sec		5 MB/sec		1 MB/sec			
	Demand (MB/sec)	Offered & Retained (MB/sec)	Unmet Demand (MB/sec)	Offered & Retained (MB/sec)	Unmet Demand (MB/sec)	Offered & Retained (MB/sec)		
<b>Class 0</b>	5	5	0	0	0	0	<b>5</b>	
<b>Class 1</b>	20	20	0	0	0	0	<b>20</b>	
<b>Class 2</b>	23	20	3	3	0	0	<b>23</b>	
<b>Class 3</b>	30	20	10	5	10	1	<b>26</b>	
<b>Class 4</b>	40	20	20	5	20	1	<b>26</b>	
<b>Total (all classes)</b>	118	.....					<b>100</b>	

Although BFS does not measures bit-rates over time, does not operate iteratively and does not rely on steady state conditions, the implementation of BFS operating under the above steady state conditions would produce steady state results consistent with the above.

It can be seen that overweight classes (classes that attain less than their demand) attain equal amounts of bandwidth. This result offers an insight into the simple principles for this algorithm which are:

- determine classes which are overweight and underweight at the time of a selection opportunity
- select packets from underweight classes before any packets from overweight classes
- select packets from overweight classes in a manner to equalize the bandwidth (bytes) attained by each overweight classes.

## WBFS Mechanism

WBFS is implemented by a scheduling mechanism distinct from the queueing mechanism - the mechanism that manages the enqueue and dequeue.

The scheduling mechanism has the means:

## Queue Manager (QMan)

- to know whether each class queue contains packets or not (non-empty or empty).
- to be informed when a class queue becomes non-empty.
- to maintain an ordered-list of underweight class queues with packets.
- to maintain an list of underweight class queues without packets.
- to maintain a list of overweight (or presumed overweight) classes and an associated numeric value of sufficient range to store a value equal to the number of bytes in the largest permitted packet. - the value (Debt) is a measure of bytes attained by the class relative to other overweight classes. (In a weighted implementation D is measure of bytes attained divided by the weight (W) of the class.)
- to instruct the queueing mechanism to dequeue a packet from a specified class queue.
- to be informed of the size of the dequeued packet.

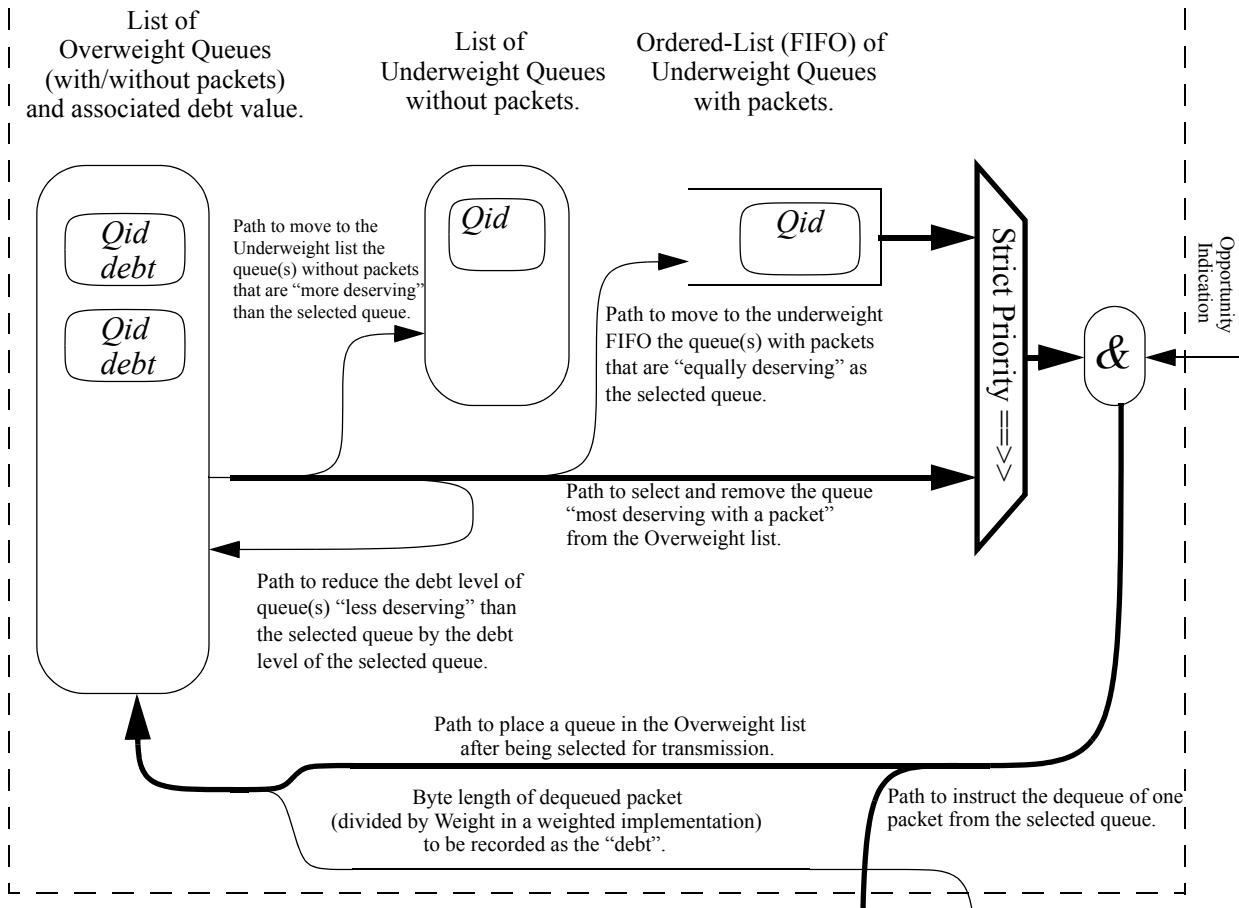
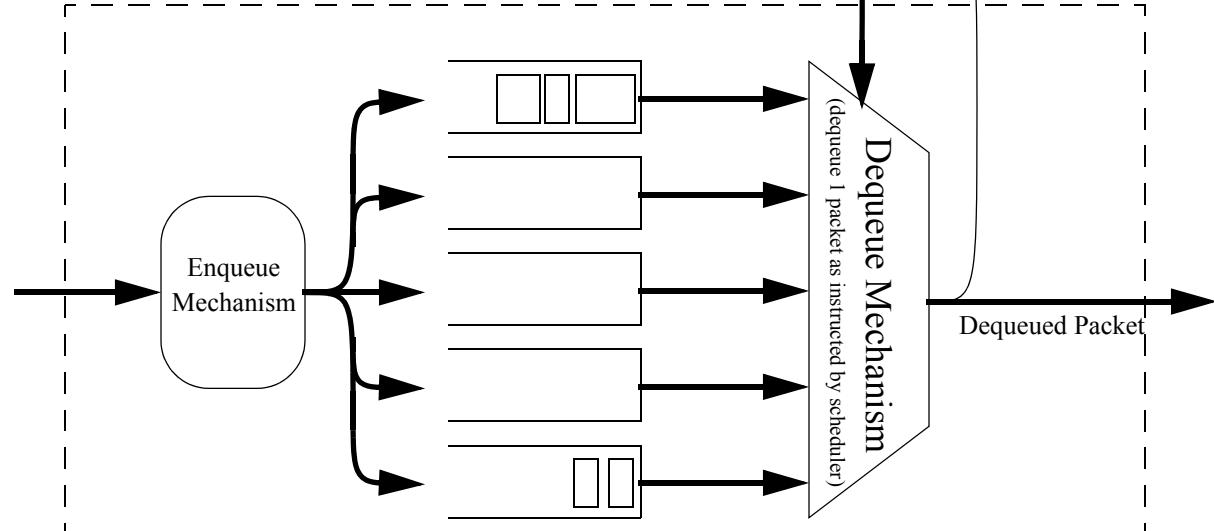
Scheduling MechanismQueueing Mechanism

Figure 3-184. WBFS: Class (Queue) Scheduling Mechanism &amp; Packet Queueing Mechanism

## WBFS Algorithm

The algorithm:

- makes and tracks presumptions about which classes are overweight and underweight
- maintains a debt value measured in bytes for overweight classes. (The difference in debt values between any two overweight classes represents the difference in the number of bytes attained by the classes while both have been overweight - greater debt represents greater attainment.)
- confirms all queues as underweight upon initialization or re-initialization.
- presumes a queue to be overweight after it is selected for a transmit opportunity and assigns an a value P to be the debt value of that class where P is equal to the number bytes in the packet transmitted. (In a weighted implementation  $P/W_{\text{class}}$  is assigned to debt.)
- holds in FIFO order any underweight queues that have had head-packets arrive since the last opportunity (i.e. multiple class queues have become non-empty) based on the arrival time of the head packet (i.e. the order in which the queue became non-empty).
- selects from the FIFO the next underweight queue to transmit it's head packet from the queue at the next transmit opportunity.
- selects the least overweight queue (lowest debt level) with an available packet for a opportunity when there is no underweight queues with a packet in the FIFO. If the selected overweight queue has a debt level of D then a credit of D is applied to erase its debt. To be fair D is credited to the debt level of each queue presumed to be overweight.
- confirms as underweight any class that was presumed overweight and had a debt level of less than D of the selected class but had no packets available. (It is considered “underweight” onward because it would have been selected if it had a packet. It will therefore attain somewhat less bandwidth than the overweight classes.)
- If all classes are without packets at any point in time the algorithm is re-initialized such that all classes are considered underweight.

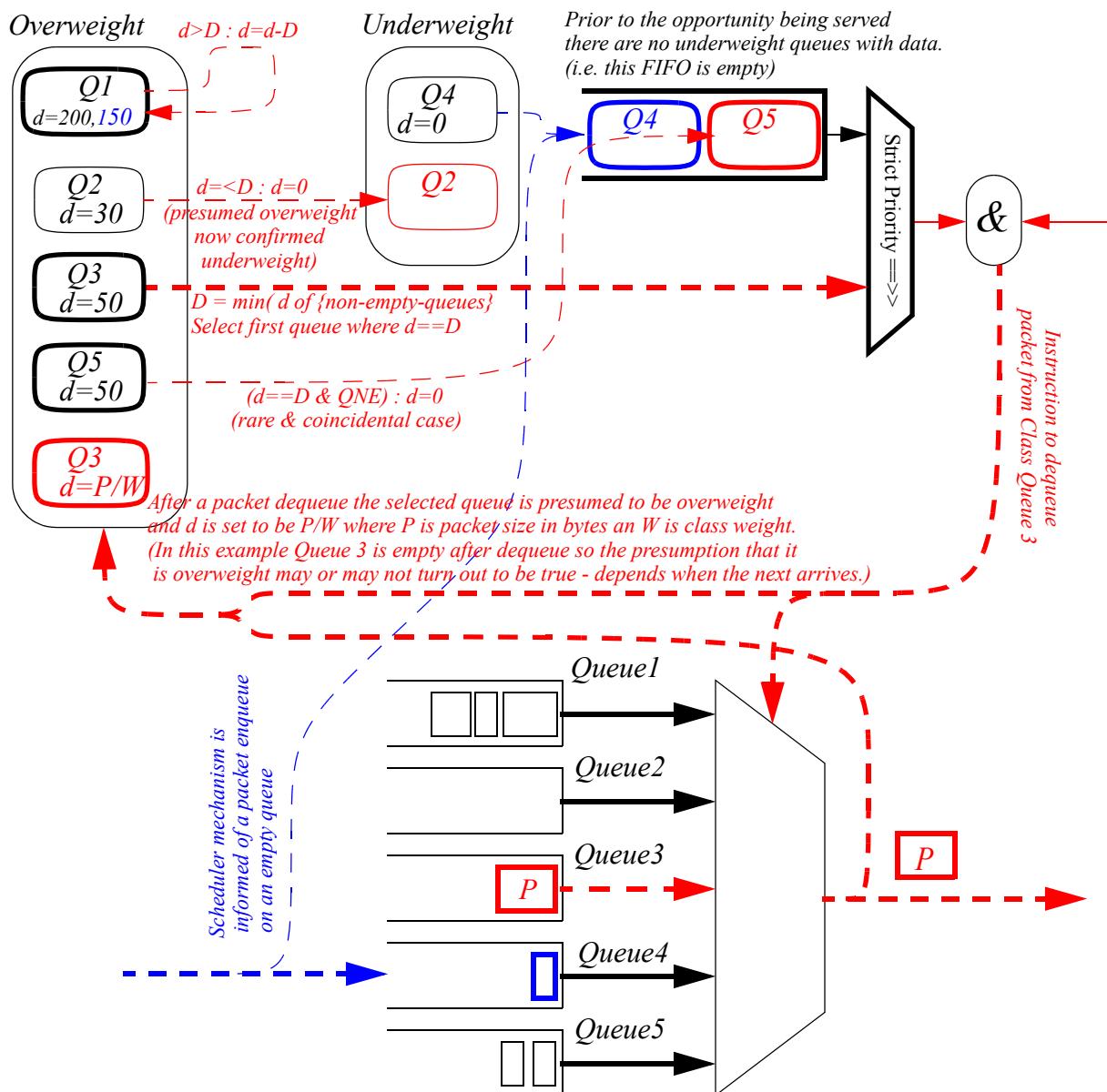
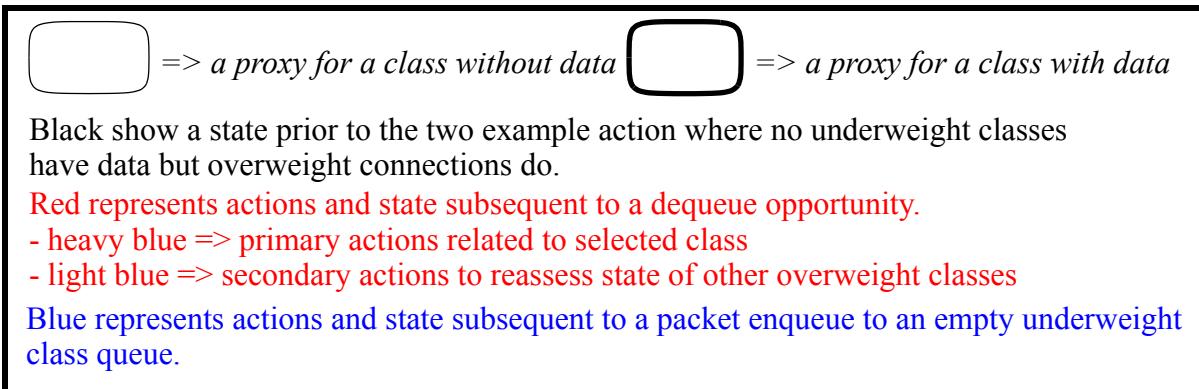


Figure 3-185. WBFS Mechanism Example: Overweight class dequeue & underweight class enqueue.

## Comparison to WFQ

WBFS will ensure that over any period in which any two classes are overweight, the cumulative bandwidth (bytes transmitted) attained (divided by class weight in a weighted implementation) by the each class shall not exceed the other by more than the size of one packet. WFQ offers a identical fairness behavior.

WBFS will prioritize the dequeues from underweight classes such that packets enqueued into an underweight queue obtain better latency. WFQ provides a similar behavior.

When multiple underweight (empty) classes receive single packets between dequeue opportunities the backlog of underweight class will be serviced in FIFO order - packets (one per class) will be dequeued in the order in which they arrived independent of packet size. By contrast WFQ orders them in the order in which packets would complete transmission in a GPS model. As such shorter packets may in some cases be dequeued ahead of longer packets that arrived first.

## 3.4 Initialize the QMan

To perform a basic initialization of the QMan, do the following required steps:

1. Set up frame queue descriptor memory (FQD).
  - a) Select a power of 2 memory size for FQDs.
  - b) Allocate the physical memory and zero the memory.
  - c) Write the base address into FQD\_BARE and FQD\_BAR registers.
  - d) Write the size and set FQD\_AR[EN] ([Section 3.2.4.44, “Data Structure Attributes Registers”](#)).
2. Set up the packed frame descriptor record (PFDR) memory.
  - a) Select a power of 2 memory size for PFDRs.
  - b) Allocate the physical memory.
  - c) Write the base address into PFDR\_BARE and PFDR\_BAR registers.
  - d) Write the size and set the EN bit in the PFDR\_AR register ([Section 3.2.4.44, “Data Structure Attributes Registers”](#)).
3. Initialize the PFDR free pool using the 0x01 command code in QMAN\_MCR ([Section 3.2.4.35, “QMan Management Command/Result Register \(QMAN\\_MCR\)”](#)).
4. Initialize the PFDR base constant ([Section 3.2.4.19, “PFDR Configuration \(PFDR\\_CFG\)”](#)).
5. Clear QMAN\_ERR\_ISR[PEBI] ([Section 3.2.4.51, “QMan Error Interrupt Status Register \(QMAN\\_ERR\\_ISR\)”](#)). This interrupt source is asserted after reset, and QMAN\_ERR\_ISR[PEBI] should be cleared after the PFDR free pool has been initialized.
6. Configure one or more software portals for use via the QCSPi\_CFG ([Section 3.2.3.17, “QMan Software Portal Configuration Register \(QCSPi\\_CFG\)”](#)).
7. Configure the destination for enqueue rejections that may occur on any direct connect portals (DCP) that will be used ([Section 3.2.4.11, “DCP Configuration Registers \(DCPi\\_CFG\)”](#)).

8. Initialize one or more frame queues (FQs) for use by using the initialize FQ command ([Section 3.3.9.5.1, “Initialize Frame Queues \(FQ\)”](#)), which is issued via the QCSPi\_CR ([Section 3.2.3.4, “QCSP Management Command Registers \(QCSPi\\_CR\)”](#)) in any of the software portals.
9. If any of the FQs initialized in the previous step are configured to use a congestion group for congestion management and avoidance ([Section 3.3.14, “Congestion Management and Avoidance”](#)), all congestion group records (CGRs) must be initialized using the initialize CGR command ([Section 3.3.9.6.1, “Initialize/Modify a CGR”](#)). Any unused CGR should be initialized to all zeros.

After completing these steps, the QMan is ready to accept enqueue and dequeue commands to any of the FQs that have been initialized.



# Chapter 4

## Buffer Manager (BMan)

### 4.1 BMan Overview

The Buffer Manager's (BMan) primary function is to reduce the overhead on software for managing free buffer pools for multiple hardware modules.

The BMan acts as a central resource in the multicore datapath infrastructure (DPI), managing pools of data storage buffers and the acquisition and release of these buffers on behalf of multiple processor cores, network interfaces, and hardware accelerators in a multicore SoC.

#### 4.1.1 BMan Features Overview

The BMan provides a centralized resource management function for memory buffers. Software allocates memory buffers that are suitable for use by the different hardware modules. These buffers are passed to the BMan, which maintains them as free lists. The hardware modules request buffers from the BMan when they need them to store data and can return the buffers when they are no longer required. The BMan also provides other functionality, such as resource depletion notification (see [Section 4.2.3.8, “BCSP Depletion State Change Interrupt Enable Registers \(BCSPn\\_SCNm\)”](#) for more information).

Whether a module releases buffers to the BMan or returns them to software is a matter of that module's configuration. It is the responsibility of the clients to ensure that the consistency of buffers on a free list is maintained.

The BMan's main functions are as follows:

- Maintains pools of buffers that have been provided by software. Other modules acquire these buffers by requesting them from the BMan. The BMan then removes buffers from one of its buffer pools and passes them to the module making the request. The requesting block must indicate which buffer pool it is requesting a buffer from. Likewise other modules can release buffers back to the BMan, providing the buffer and the pool in which to store it.
- Maintains the buffer pools that it manages as free lists using an area of cache-inhibited memory that is reserved by software for private use by the BMan. The Free Buffer Pointer Records (FBPRs) stored in this area are used to maintain the free lists of buffers and are the only external memory accessed by the BMan. The BMan does not read or write to the actual data buffers themselves.
- Keeps a small stockpile of buffers internally for each buffer pool that it is maintaining, which reduces latency when responding to a request for a buffer. The BMan keeps the number of buffers in the stockpile between acceptable limits by retrieving free buffers from the external list when a stockpile falls below a lower limit and adding buffers to the external list when the stockpile exceeds an upper limit.

## **Buffer Manager (BMan)**

The BMan assumes that the buffers in each pool:

- Have consistent, identical characteristics, such as size and memory partition/address ability, and that
- Are unique, such that no duplicate buffer releases occur.

It is up to software and the hardware clients to ensure that these assumptions are maintained. The BMan does not check that these assumptions are maintained and will still function correctly if they are violated.

This figure shows a simplified block diagram of the BMan's internal structure and interfaces.

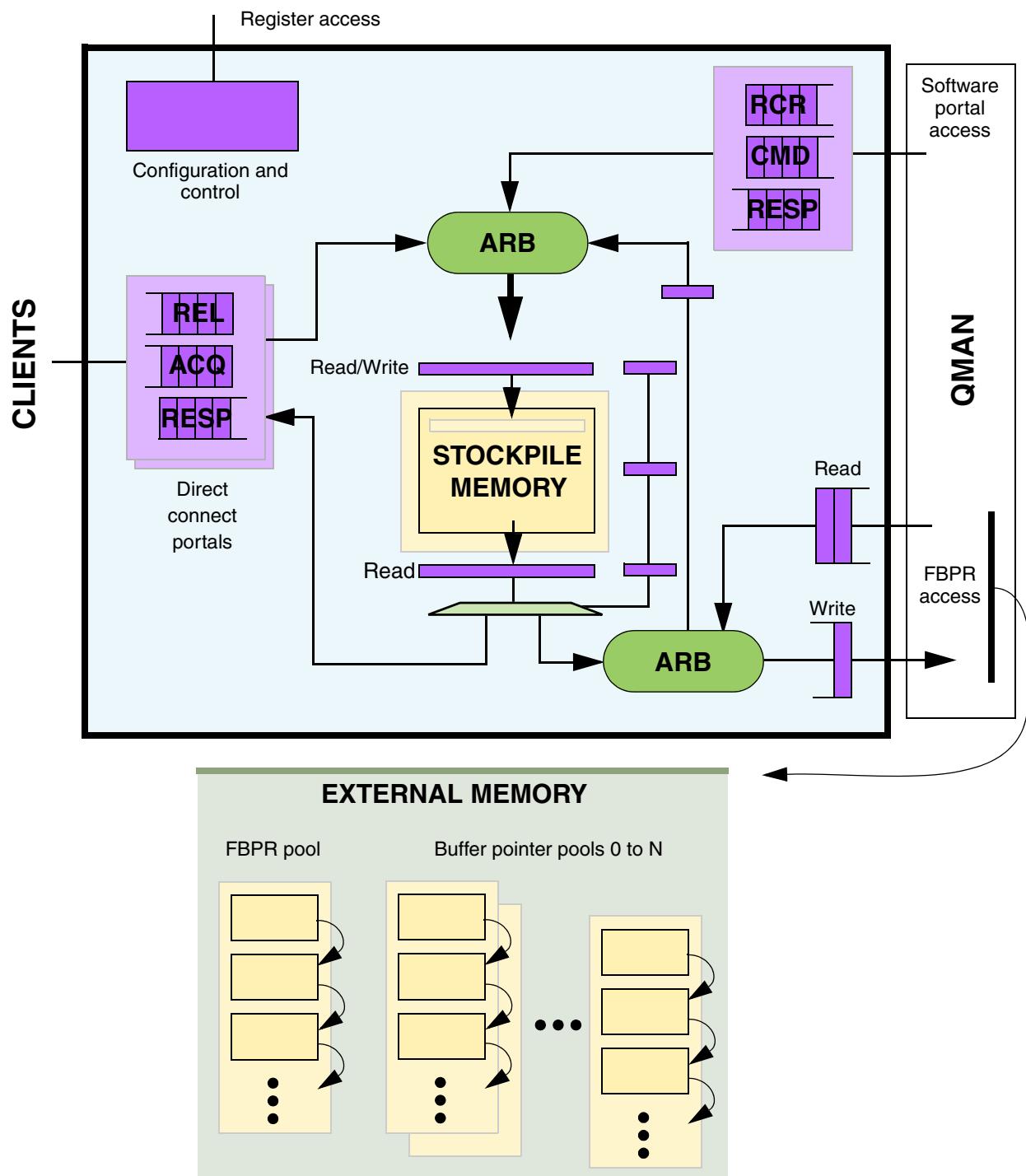


Figure 4-1. Buffer Manager (BMan) Block Diagram

## 4.1.2 BMan Features Summary

The BMan includes the following features:

- 10 Software portals (SWP) for software
- 2 Direct connect portals (DCP) for hardware
- Supports 64 separate pools of free buffers.
- On acquires, can return up to eight buffers per request.
- Keeps a stockpile of up to 64 buffers per pool in an internal memory.
  - The buffer pool stockpile:
    - reduces latency to deliver buffers, and
    - aids in handling bursts of acquires and releases.
  - If buffer acquires and releases are balanced for a pool, no external memory transactions are required.
- Per pool, maintains a linked-list of FBPRs in a private external memory area.
  - Buffer pointers are stored in FBPRs in LIFO order.
  - Accessing an FBPR requires a single external memory transaction.
- Stores a free list of unused FBPRs in the same manner as buffers belonging to regular pools.
- Data buffers are not accessed by BMan.
- Note that
  - Releases to pool id 255 (0xFF) are dropped.
  - Buffer pointers with upper bits beyond 40 bits are truncated to 40 bits.
  - For any pool that does not exist, for acquires (hardware and software portals) and queries (software portal), the response will be as if that pool were empty.
  - For any pool that does not exist, for releases (hardware and software portals), the released buffer pointer will be dropped and no exception will occur, the behavior will be as if the buffer pointer had been successfully accepted into the designated pool.

## 4.2 BMan Memory Map and Register Definition

This section includes the module memory map and detailed descriptions of all BMan registers and their offsets.

### 4.2.1 BMan Software Portal Memory Map

The BMan's software portals are mapped in an area of system memory space aligned on a 128 Mbyte boundary. The address offsets shown in this table refer to the addresses offset from this boundary.

**Table 4-1. BMan Software Portal Memory Map**

Offset	Register	Access	Reset Value	Section/Page
<b>BMan Software Portal 0, Command/Response and Release Command Ring Registers, Cache-Enabled Area</b>				
0x000_0000	BCSP0_CR—BMan software portal 0, command register	R/W	0x0000_0000	<a href="#">4.2.3.1/4-9</a>
0x000_0100	BCSP0_RR0—BMan software portal 0, response register 0	R	0x0000_0000	<a href="#">4.2.3.2/4-10</a>
0x000_0140	BCSP0_RR1—BMan software portal 0, response register 1	R	0x0000_0000	<a href="#">4.2.3.2/4-10</a>
0x000_1000– 0x000_11C0	BCSP0_RCR0—BMan software portal 0, RCR, Entry 0–7	R/W	0x0000_0000	<a href="#">4.2.3.3/4-11</a>
0x000_3000	BCSP0_RCR_PI_CENA—Software portal 0, release command ring, producer index cache-enabled register	R/W	0x0000_0008	<a href="#">4.2.3.4/4-12</a>
0x000_3100	BCSP0_RCR_CI_CENA—Software portal 0, release command ring, consumer index cache-enabled register	R	0x0000_8008	<a href="#">4.2.3.5/4-13</a>
0x000_3200– 0x000_FFFF	Reserved. BMan software portal 0, cache-enabled area	—	—	—
<b>BMan Software Portals 1 and Above, Cache-Enabled Area</b>				
0x001_0000– 0x001_FFFF	BMan Software Portal 1, cache-enabled area (64 KB)	—	—	—
0x002_0000 and above	BMan Software Portal cache-enabled area, 64 KB per portal In the LS1043A: 10 software portals occupy 0x000_0000 to 0x009_FFFF	—	—	—
0x002_0000– 0x3FF_FFFF	64 KB blocks corresponding to non-existent software portals are reserved.	—	—	—
<b>BMan Software Portal 0, RCR Index, Configuration, and Interrupt Control Registers, Cache-Inhibited Area</b>				
0x400_0000	BCSP0_CR—BMan software portal 0, command register	R/W	0x0000_0000	<a href="#">4.2.3.1/4-9</a>
0x400_0100	BCSP0_RR0—BMan software portal 0, response register 0	R	0x0000_0000	<a href="#">4.2.3.2/4-10</a>
0x400_0140	BCSP0_RR1—BMan software portal 0, response register 1	R	0x0000_0000	<a href="#">4.2.3.2/4-10</a>
0x400_1000– 0x400_11C0	BCSP0_RCR0—BMan software portal 0, RCR, Entry 0–7	R/W	0x0000_0000	<a href="#">4.2.3.3/4-11</a>
0x400_3000	BCSP0_RCR_PI_CINH—software portal 0, release command ring, producer index cache-inhibited register	R/W	0x0000_0008	<a href="#">4.2.3.4/4-12</a>
0x400_3100	BCSP0_RCR_CI_CINH—software portal 0, release command ring, consumer index cache-inhibited register	R	0x0000_8008	<a href="#">4.2.3.5/4-13</a>
0x400_3200	BCSP0_RCR_ITR—software portal 0, RCR interrupt threshold register	R/W	0x0000_0000	<a href="#">4.2.3.6/4-15</a>
0x400_3300	BCSP0_CFG—BMan software portal 0, configuration register	R/W	0x0000_0000	<a href="#">4.2.3.7/4-15</a>
0x400_3400	BCSP0_SCN0—software portal 0, depletion state change notification interrupt enable register, pools 0–31	R/W	0xFFFF_FFFF	<a href="#">4.2.3.8/4-16</a>
0x400_3440	BCSP0_SCN1—software portal 0, depletion state change notification interrupt enable register, pools 32–63	R/W	0xFFFF_FFFF	<a href="#">4.2.3.8/4-16</a>

**Table 4-1. BMan Software Portal Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x400_3E00	BCSP0_ISR—BMan software portal 0, interrupt status register	w1c	0x0000_0000	<a href="#">4.2.3.9/4-17</a>
0x400_3E40	BCSP0_IER—BMan software portal 0, interrupt enable register	R/W	0x0000_0000	
0x400_3E80	BCSP0_ISDR—BMan software portal 0, interrupt status disable register	R/W	0x0000_0000	
0x400_3EC0	BCSP0_IIR—BMan software portal 0, interrupt inhibit register	R/W	0x0000_0000	
0x400_3F00	BCSP0_IFR—BMan software portal 0, interrupt force register	W	0x0000_0000	
0x400_4000–0x400_FFFF	Reserved. BMan software portal 0, cache-enabled area	—	—	—
<b>BMan Software Portals 1 and above, Cache-Inhibited Area</b>				
0x401_0000–0x401_FFFF	BMan Software Portal 1, cache-inhibited area (64 KB)	—	—	—
0x402_0000 and above	BMan Software Portal cache-inhibited area, 64 KB per portal In the LS1043A: 10 software portals occupy 0x400_0000 to 0x409_FFFF	—	—	—
0x402_0000–0x7FF_FFFF	64 KB blocks corresponding to non-existent software portals are reserved.	—	—	—

## 4.2.2 BMan Configuration and Control Register Memory Map

This table shows the memory map of the BMan’s configuration and control registers. The BMan occupies a 4 KB address block (4 KB-aligned) in the configuration, control, and status register (CCSR) space of the system. The address offsets shown in this table refer to the address offset from the CCSR base address assigned for the BMan: (0x31A000)

**Table 4-2. BMan Configuration, Control, and Status Register Memory Map**

Offset	Register	Access	Reset Value	Section/ Page
<b>BMan Buffer Pool Configuration and Status Registers (Pool n)</b>				
0x000–0x0FC	BMAN_POOL0–63_SWDET—BMan software portal depletion entry threshold	R/W	0x0000_0000	<a href="#">4.2.4.3/4-23</a>
0x100–0x1FC	BMAN_POOL0–63_HWDET—BMan software portal depletion entry threshold	R/W	0x0000_0000	
0x200–0x2FC	BMAN_POOL0–63_SWDXT—BMan software portal depletion exit threshold	R/W	0x0000_0000	
0x300–0x3FC	BMAN_POOL0–63_HWDXT—BMan software portal depletion exit threshold	R/W	0x0000_0000	

**Table 4-2. BMan Configuration, Control, and Status Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/ Page
0x400– 0x4FC	BMAN_POOL0–63_SDCNT—BMan software portal depletion count	RR	0x0000_0000	<a href="#">4.2.4.5/4-25</a>
0x500– 0x5FC	BMAN_POOL0–63_HDCNT—BMan hardware portal depletion count	RR	0x0000_0000	
0x600– 0x6FC	BMAN_POOL0–63_CONTENT—Snapshot of buffer count in pool 0	R	0x0000_0000	<a href="#">4.2.4.6/4-26</a>
0x700– 0x7FC	BMAN_POOL0–63_HDPT—Head pointer for pool 0 FBPR list	R	0x0000_0000	<a href="#">4.2.4.7/4-27</a>
<b>Free Buffer Proxy Record (FBPR) Manager Query Registers</b>				
0x800	FBPR_FPC—FBPR free pool count register	R	0x0000_0000	<a href="#">4.2.4.6/4-26</a>
0x804	FBPR_FP_LWIT—FBPR free pool low watermark interrupt threshold register	R/W	0x0000_0000	<a href="#">4.2.4.4/4-25</a>
0x808	FBPR_HDPT—Head pointer for FBPR list register	R	0x0000_0000	<a href="#">4.2.4.7/4-27</a>
<b>Performance Monitor (PM) Configuration Registers</b>				
0x900– 0x91C	BMAN_CMD_PM0–7_CFG—BMan command performance monitor configuration register	R/W	0x0000_0000	<a href="#">4.2.4.8/4-27</a>
0x920– 0x93C	BMAN_FL_PM0–7_CFG—BMan free list performance monitor configuration register	R/W	0x0000_0000	<a href="#">4.2.4.9/4-29</a>
0x940– 0x95C	BMAN_CMD_PM0–7_CFG_CFIFO—BMan command performance monitor configuration register: command fifo select	R/W	0x0000_0000	<a href="#">4.2.4.8/4-27</a>
0x960	STATE_IDLE—idle state	R	0x0000_0003	<a href="#">4.2.4.10/4-29</a>
0x964	STATE_STOP—force idle	R/W	0x0000_0000	<a href="#">4.2.4.10/4-29</a>
<b>BMan Error Capture Registers</b>				
0xA04	BMAN_ECIR—BMan error capture information register	R	0x0000_0000	<a href="#">4.2.4.14/4-34</a>
0xA30	BMAN_SBET—BMan single-bit ECC error threshold register	R/W	0x0000_0000	<a href="#">4.2.4.12/4-33</a>
0xA34	BMAN_CECR—BMan corruption error capture register	R	0x0000_0000	<a href="#">4.2.4.15/4-35</a>
0xA38	BMAN_CEAR—BMan corruption error Address register	R	0x0000_0000	
0xA44	BMAN_AECR—BMan access error capture register	R	0x0000_0000	<a href="#">4.2.4.16/4-36</a>
0xA48	BMAN_AEAR—BMan access error address register	R	0x0000_0000	
0xA80	BMAN_SBEC0—BMan single-bit ECC error count 0 register	RR	0x0000_0000	<a href="#">4.2.4.13/4-33</a>

**Table 4-2. BMan Configuration, Control, and Status Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/ Page
<b>BMan ID/Revision Registers</b>				
0xBF8	BMAN_IP_REV_1—BMan IP Block Revision 1 register	R	0xA02_0201	4.2.4.1/4-20
0xBFC	BMAN_IP_REV_2—BMan IP Block Revision 2 register	R	0x0000_0103	
<b>External Memory Configuration Registers</b>				
0xC00	FBPR_BARE—Data structure extended base address register	R/W	0x0000_0000	4.2.4.2/4-21
0xC04	FBPR_BAR—Data structure base address register	R/W	0x0000_0000	
0xC10	FBPR_AR—Data structure attributes register	R/W	0x0000_0000	
0xD04	BMAN_SRCIDR—BMan source ID register	R	0x0000_0000	
0xD08	BMAN_ICIDR—BMan Isolation Context Identifier register	R/W	0x0000_0000	
<b>BMan Interrupt and Error Registers</b>				
0xE00	BMAN_ERR_ISR—BMan error interrupt status register	w1c	0x0000_0000	4.2.4.11/4-31
0xE04	BMAN_ERR_IER—BMan error interrupt enable register	R/W	0x0000_0000	
0xE08	BMAN_ERR_ISDR—BMan error interrupt status disable register	R/W	0x0000_0000	
0xE0C	BMAN_ERR_IIR—BMan error interrupt inhibit register	R/W	0x0000_0000	
0xE10	BMAN_ERR_IFR—BMan error interrupt force register	W	0x0000_0000	
<b>BMan Debug Control Registers</b>				
0xF00	DEBUG_CFIIFO—BMan debug: command fifo disable register	R/W	0x0000_0000	4.2.4.17/4-37
0xF04	DEBUG_FSM—BMan debug state machine disable register	R/W	0x0000_0000	4.2.4.18/4-38

### 4.2.3 BMan Software Portal (BCSP) Register Descriptions

The BMan software portals are designed for 64-byte cache lines. In processors with 32-byte cache lines and where 64-byte accesses are required, update the cache line containing the verb last. A write of the verb byte with a valid verb triggers software portal activity. All other data in the command must be valid and flushed to hardware by the time this final command-activating access arrives at the hardware.

All BMan software portal (BCSP) register arrays contain one register for each of the  $n$  software portals.

#### 4.2.3.1 BCSP Command Registers (BCSPn\_CR)

The BMan software portal command registers (BCSP $n$ \_CR) are described in more detail in Section 4.3.3.1, “Command Register (CR) Functionality.”

**Figure 4-2. BCSP Command Registers (BCSPn\_CR)**

**Table 4-3. BCSPn\_CR Field Descriptions**

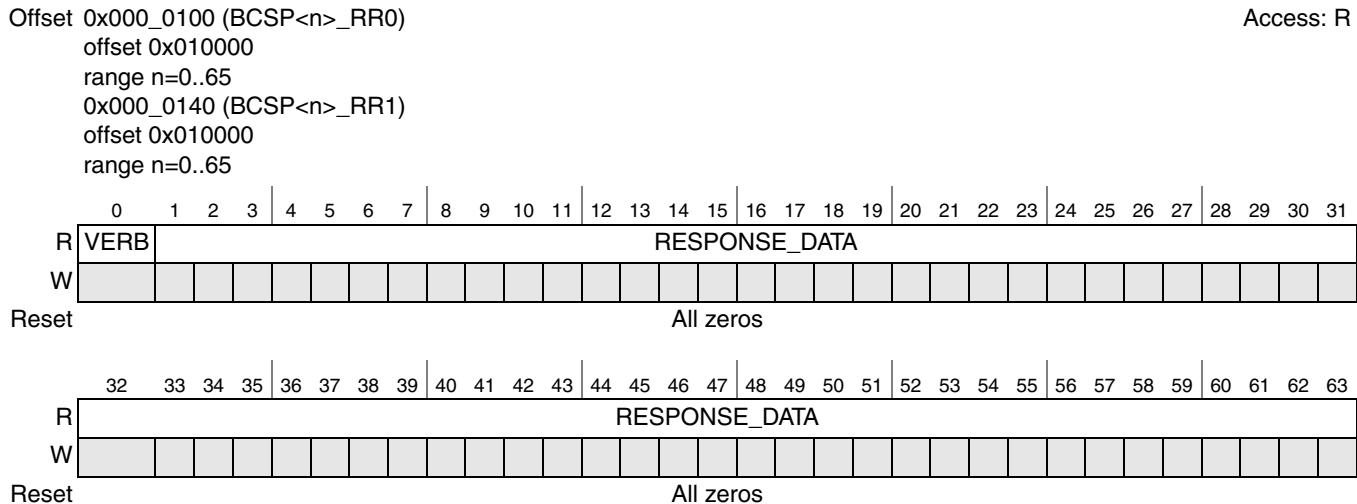
Field	Description
0 VERB	<p>The verb byte consists of:</p> <ul style="list-style-type: none"> <li>Bit 0 (msb): Valid bit This bit indicates to the BMan whether this CR command is valid or not. The expected polarity of this bit toggles each time a new command is written to the BCSPn_CR. This is an alternating polarity valid bit (see <a href="#">Section 4.3.3.1.2, “Benefits of Using Alternating Polarity”</a>). This bit also indicates which of the response registers (BCSPn_RRm) are used to return the response to this command. See <a href="#">Section 4.3.3.1, “Command Register (CR) Functionality”</a>, for more detail.</li> <li>Bits 1–7 (lsbs): Command verb Indicates the command being issued. See <a href="#">Section 4.3.3.1, “Command Register (CR) Functionality”</a>, for a description of the commands.</li> </ul>
1 CD	<p>Command data</p> <p>The information carried in this byte depends on the command. See <a href="#">Section 4.3.3.1, “Command Register (CR) Functionality”</a>, for a description of the commands.</p>
2-63	Reserved

### 4.2.3.2 BCSP Response Registers (BCSPn\_RRm)

The BMan software portal response registers (BCSPn\_RRm) are described in more detail in Section 4.3.3.2, “Response Register (RR) Functionality.”

#### NOTE

Disable the ECC function first by using BMAN\_SBET for BCSPn\_RRm reads using an access size of less than 64 bytes.



**Figure 4-3. BCSP Response Registers (BCSPn\_RRm)**

**Table 4-4. BCSPn\_RRm Field Descriptions**

Field	Description
0 VERB	<p>The verb byte consists of:</p> <ul style="list-style-type: none"> <li>Bit 0 (msb): Valid bit</li> </ul> <p>This bit is the same as the Valid bit in the BCSPn_CR for the command that initiated this response. As such, this bit is always 0 in BCSPn_RR0, and is always 1 when a valid response is present in BCSPn_RR1.</p> <ul style="list-style-type: none"> <li>Bit 1–7 (lsbs): Response verb</li> </ul> <p>Indicates the type of response. See Section 4.3.3.2, “Response Register (RR) Functionality” for a description of the responses.</p>
1-63 RESPONSE_DATA	<p>Response data</p> <p>The information carried in these bytes is dependent on the type of response. See Section 4.3.3.2, “Response Register (RR) Functionality,” for a description of the responses.</p>

#### 4.2.3.3 BCSP Release Command Ring Registers (BCSP*n*\_RCR*m*)

The BMan software portal release command ring registers (`BCSPn_RCRm`) issue buffer release commands from software to the BMan, and are described in more detail in [Section 4.3.3.3, “Release Command Ring \(RCR\) Functionality.”](#) The `BCSPn_RCRm` consist of eight, separate 64-byte entries that are organized as a circular FIFO.

## **NOTE**

Disable the ECC function first by using BMAN\_SBET for BCSPn\_RCRm when using access sizes of less than 64 bytes.

**Figure 4-4. BCSP Release Command Ring Registers (BCSP*n* RCR*m*)**

**Table 4-5. BCSP $n$ \_RCR $m$  Field Descriptions**

Field	Description
0 VERB	<p>The verb byte consists of:</p> <ul style="list-style-type: none"> <li>• Bit 0 (msb): Valid bit.</li> </ul> <p>The function of this bit is determined by the RCR production notification mode programmed by software for each portal, see <a href="#">Section 4.2.3.7, “BCSP Configuration Registers (BCSP<math>n</math>_CFG)</a>.”</p> <p>For the “valid bit mode” of RCR production notification, this bit must be set to the current valid bit polarity (1 or 0) in every command issued in an RCR entry, and the BMan uses this bit (along with the presence of a non-zero command verb) to determine when a valid entry has been placed in the ring. The expected polarity of this bit, indicating a valid entry, toggles each time that the ring wraps around from entry 7 to entry 0, and this is referred to as an alternating polarity valid bit. See <a href="#">Section 4.3.3.3, “Release Command Ring (RCR) Functionality</a>,” and <a href="#">Section 4.3.3.1.2, “Benefits of Using Alternating Polarity</a>,” for more detail.</p> <p>For either of the “PI write modes” of RCR production notification, this bit is ignored.</p> <ul style="list-style-type: none"> <li>• Bit 1-7 (lsbs): Release command verb</li> </ul> <p>Release to a single pool ID or multiple pool IDs. See <a href="#">Section 4.3.3.3, “Release Command Ring (RCR) Functionality</a>” for details.</p>
1-63 COMMAND DATA	Command Data. The information carried in these bytes is dependent on the type of command. See <a href="#">Section 4.3.3.3, “Release Command Ring (RCR) Functionality</a> ” for a description of the commands.

#### 4.2.3.4 BCSP Release Command Ring (RCR) Producer Index Registers (BCSP $n$ \_RCR\_PI\_CENA,BCSP $n$ \_RCR\_PI\_CINH)

For either of the “Producer Index (PI) write modes” of RCR production notification, software uses the RCR\_PI value to notify the BMan that one or more valid entries have been written into the RCR (see [Section 4.3.3.3, “Release Command Ring \(RCR\) Functionality](#)”).

RCR\_PI is a single value stored in the BMan, but is available in one of the following separate locations:

- Cache-enabled register (BCSP $n$ \_RCR\_PI\_CENA)
- Cache-inhibited register (BCSP $n$ \_RCR\_PI\_CINH)

Only one of these two registers is enabled to receive writes, and the other register is read only. The RCR production notification mode determines which of the two registers is write/read and which one is read only (see [Section 4.2.3.7, “BCSP Configuration Registers \(BCSP \$n\$ \\_CFG\)](#)”).

If the RCR production notification mode is programmed for valid bit mode rather than one of the two producer index write modes, then the RCR\_PI value is updated implicitly by BMan using the valid bits provided in the ring entries, and in this mode both of the RCR\_PI registers are read only.

BCSP $n$ \_RCR\_PI\_CENA and BCSP $n$ \_RCR\_PI\_CINH are located in word 0 (that is, the word at address offset 0) of the 64-byte aligned memory region.

Offset 0x000\_3000 (BCSP<n>\_RCR\_PI\_CENA)  
 offset 0x010000  
 range n=0..65  
 0x400\_3000 (BCSP<n>\_RCR\_PI\_CINH)  
 offset 0x010000  
 range n=0..65

Access: R/W or R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VP	PI		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

**Figure 4-5. BCSP RCR Producer Index Cache-Enabled and Cache-Inhibited Registers (BCSPn\_RCR\_PI\_CENA, BCSPn\_RCR\_PI\_CINH)**

**Table 4-6. BCSPn\_RCR\_PI\_CENA, BCSPn\_RCR\_PI\_CINH Field Descriptions**

Field	Description	
0-27	Reserved	
28	Current valid bit polarity at producer index	
VP	For the “valid bit mode” of RCR production notification, indicates the polarity that the BMan is expecting for the alternating polarity valid bit (see <a href="#">Section 4.3.3.1.2, “Benefits of Using Alternating Polarity”</a> ) in the RCR entry currently pointed to by RCR_PI. This bit toggles every time the BMan’s RCR_PI value wraps from 7 back to 0.	
	0 The BMan is expecting a 0 to indicate a valid entry, and 1 indicates an invalid entry.	
	1 The BMan is expecting a 1 to indicate a valid entry, and 0 indicates an invalid entry.	
	Reading this bit can be used as a debug assist or to re-synchronize software’s copy of this bit with the BMan’s. For either of the “PI write modes” of RCR production notification, this bit is reserved.	
29-31	Producer index	
PI	This field indicates the ring entry in which the next valid command will be written. When RCR_PI and RCR_CI are equal, the BMan considers the ring to be empty and waits for RCR_PI to be advanced. This field may be read/write or read-only depending on the programmed RCR production notification mode.	
	• If RCR production notification mode is 0: PI is read/write in RCR_PI_CINH and read only in RCR_PI_CENA.	
	• If RCR production notification mode is 1: PI is read/write in RCR_PI_CENA and read only in RCR_PI_CINH.	
	• If RCR production notification mode is 2 or 3: PI is read only in both RCR_PI_CINH and RCR_PI_CENA.	

#### 4.2.3.5 BCSP RCR Consumer Index Registers (BCSPn\_RCR\_CI\_CENA, BCSPn\_RCR\_CI\_CINH)

The Bman software portal RCR Consumer Index (RCR\_CI) registers indicate which entry in the ring is next to be consumed by the BMan. Software reads this value to determine when it is safe to add new entries to the ring.

The RCR\_CI is a single read-only value stored in the BMan, but is available in one of the following separate locations:

- Cache-enabled register (BCSPn\_RCR\_CI\_CENA)
- Cache-inhibited register (BCSPn\_RCR\_CI\_CINH)

BCSPn\_RCR\_CI\_CENA and BCSPn\_RCR\_CI\_CINH are located in word 0 (that is, the word at address offset 0) of the 64-byte aligned memory region.

## Buffer Manager (BMan)

Offset 0x000\_3100 (BCSP<n>\_RCR\_CI\_CENA)  
 offset 0x010000  
 range n=0..65  
 0x400\_3100(BCSP<n>\_RCR\_CI\_CINH)  
 offset 0x010000  
 range n=0..65

Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VP	0	0	0	0	0	0	0	0	0	0	0	0	0	VC	CI		
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

**Figure 4-6. BCSP RCR Consumer Index Cache-Enabled and Cache-Inhibited Registers (BCSPn\_RCR\_CI\_CENA, BCSPn\_RCR\_CI\_CINH)**

**Table 4-7. BCSPn\_RCR\_CI\_CENA, BCSPn\_RCR\_CI\_CENA Field Descriptions**

Field	Description
0-15	Reserved
16 VP	Current valid bit polarity at producer index For the “valid bit mode” of RCR production notification, this bit is a copy of QCSPn_RCR_PI[VP] (see <a href="#">Section 4.2.3.4, “BCSP Release Command Ring (RCR) Producer Index Registers (BCSPn_RCR_PI_CENA,BCSPn_RCR_PI_CINH)”</a> ). For either of the “PI write modes” of RCR production notification, this bit is reserved.
17-27	Reserved
28 VC	Current valid bit polarity at consumer index For the “valid bit mode” of RCR production notification, this bit indicates the polarity that BMan is currently expecting to see in the alternating polarity valid bit (see <a href="#">Section 4.3.3.1.2, “Benefits of Using Alternating Polarity”</a> ) of the RCR entry pointed to by RCR_CI. This bit toggles every time that BMan’s RCR_CI value wraps from 7 back to 0. 0 The BMan is expecting a 0 to indicate a valid entry, and 1 indicates an invalid entry. 1 The BMan is expecting a 1 to indicate a valid entry, and 0 indicates an invalid entry. Reading this bit can be used as a debug assist or to re-synchronize software’s copy of this bit with the BMan’s. For either of the “PI write modes” of RCR production notification, this bit is reserved.
29-31 CI	Consumer index This field indicates the ring entry that is next to be consumed by BMan. When the difference between RCR_PI and RCR_CI is greater than or equal to the maximum number of entries that the ring can hold, which for RCR is 7, then the ring is full and software must not add any new entries to the ring until the BMan has consumed an entry and advanced CI.

#### 4.2.3.6 BCSP RCR Interrupt Threshold Register (BCSP*n*\_RCR\_ITR)

The BMan software portal RCR interrupt threshold registers (BCSP*n*\_RCR\_ITR) should be mapped into a cache-inhibited page within each software portal.

Offset 0x400_3200(BCSP<n>_RCR_ITR)																															Access: R/W
offset 0x010000																															
range n=0..65																															
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RCR_IT	
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 4-7. BCSP RCR Interrupt Threshold Register (BCSP*n*\_RCR\_ITR)

Table 4-8. BCSP*n*\_RCR\_ITR Field Descriptions

Field	Description
0-28	Reserved
29-31 RCR_IT	RCR Interrupt Threshold. Sets the threshold for RCR ring interrupt generation (See BCSP <i>n</i> _ISR[RCRI] in Section 4.2.3.9, “BCSP Functional Interrupt Management Registers (BCSP <i>n</i> _ISR, BCSP <i>n</i> _IER, BCSP <i>n</i> _ISDR, BCSP <i>n</i> _IFR, BCSP <i>n</i> _IIR)”). The RCR ring interrupt asserts when the ring contains fewer than RCR_IT entries, see Section 4.3.4.3, “Functional Interrupt Source, per Software Portal.”

#### 4.2.3.7 BCSP Configuration Registers (BCSP*n*\_CFG)

The BMan software portal configuration registers (BCSP*n*\_CFG) should be mapped into a cache-inhibited page within each software portal.

Offset 0x400_3300 (BCSP<n>_CFG)																															Access: R/W		
offset 0x010000																																	
range n=0..65																																	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	WN	0	0	RPM	
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-8. BMan Software Portal Configuration Registers (BCSP*n*\_CFG)

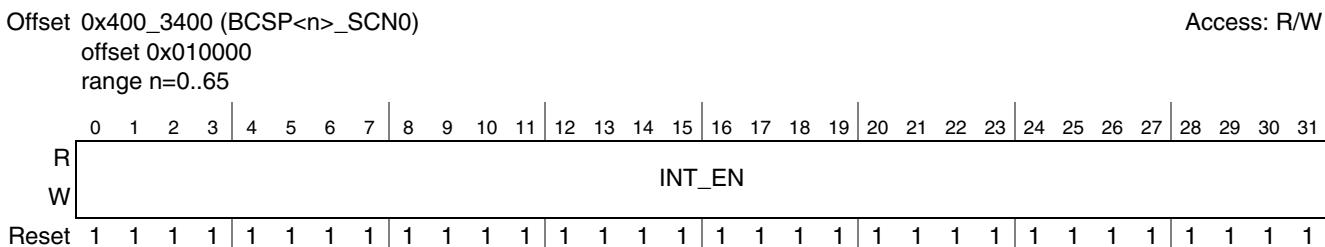
**Table 4-9. BCSPn\_CFG Field Descriptions**

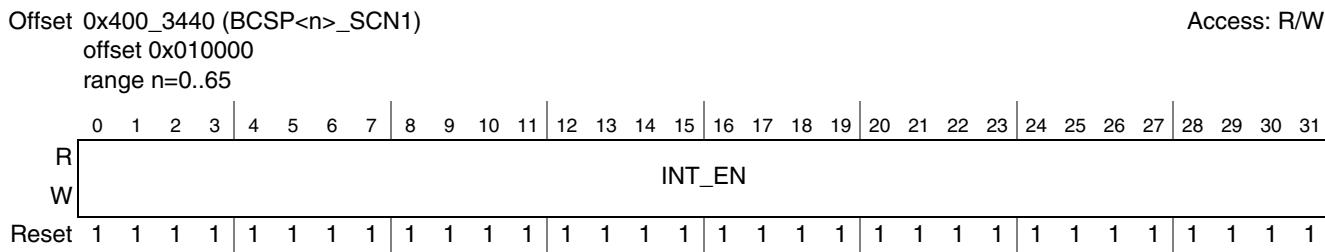
Field	Description
0-26, 28-29	Reserved
27 WN	<p>Writes Non-cacheable            The RCR and CR are 64 byte structures writeable by software which reside in both the cache-enabled and cache-inhibited areas of the portal.</p> <p>This bit specifies the area of the software portal memory map (see <a href="#">Section 4.2.1, “BMan Software Portal Memory Map”</a>) in which writes to these structures are enabled.</p> <ul style="list-style-type: none"> <li>0 RCR and CR writes are enabled in the cache-enabled portion of the memory map. Writes to the cache-inhibited portion will be ignored.</li> <li>1 RCR and CR writes are enabled in the cache-inhibited portion of the memory map. Writes to the cache-enabled portion will be ignored.</li> </ul>
30-31 RPM	<p>RCR production notification mode            (See <a href="#">Section 4.3.3.3, “Release Command Ring (RCR) Functionality.”</a>)</p> <ul style="list-style-type: none"> <li>0 PI write mode, cache-inhibited            RCR production notifications are issued by writing a producer index (PI) value in RCR_PI_CINH (see <a href="#">Section 4.2.3.4, “BCSP Release Command Ring (RCR) Producer Index Registers (BCSPn_RCR_PI_CENA,BCSPn_RCR_PI_CINH)”</a>).</li> <li>1 PI write mode, cache-enabled            RCR production notifications are issued by writing a producer index (PI) value in RCR_PI_CENA (see <a href="#">Section 4.2.3.4, “BCSP Release Command Ring (RCR) Producer Index Registers (BCSPn_RCR_PI_CENA,BCSPn_RCR_PI_CINH)”</a>).</li> <li>2 or 3 Valid bit mode            RCR production notifications are derived implicitly by the BMan using the alternating polarity valid bit in each RCR entry (see <a href="#">Section 4.3.3.1.2, “Benefits of Using Alternating Polarity.”</a>)</li> </ul>

#### 4.2.3.8 BCSP Depletion State Change Interrupt Enable Registers (BCSPn\_SCNm)

The BMan software portal State Change Notification (SCN) depletion interrupt enable registers (BCSPn\_SCNm) should be mapped into a cache-inhibited page within each software portal.

Bits that are set cause depletion interrupts for pools associated with those bits to the software portal associated with this register (BCSPn) (see [Section 4.3.5, “Buffer Pool State”](#)).

**Figure 4-9. BCSP SCN0 Depletion State Change Interrupt Enable Registers (BCSPn\_SCN0)**

**Figure 4-10. BCSP SCN1 Depletion State Change Interrupt Enable Registers (BCSPn\_SCN1)****Table 4-10. BCSPn\_SCNm Field Descriptions**

Field	Description
0-31 INT_EN	Depletion interrupt enable <ul style="list-style-type: none"> <li>For BCSP0-9_SCN0, each bit <i>m</i> corresponds to pool <i>m</i>.</li> <li>For BCSP0-9_SCN1, each bit <i>m</i> corresponds to pool 32 + <i>m</i>.</li> </ul> 1 Depletion notification enabled 0 Depletion notification disabled

#### 4.2.3.9 BCSP Functional Interrupt Management Registers (BCSPn\_ISR, BCSPn\_IER, BCSPn\_ISDR, BCSPn\_IFR, BCSPn\_IIR)

The BMan software portal functional interrupt management registers (BCSPn\_ISR, BCSPn\_IER, BCSPn\_ISDR, BCSPn\_IFR, BCSPn\_IIR) should be mapped into a cache-inhibited page within each software portal.

BCSPn interrupt management registers contain functional interrupts that pertain to software portals. A separate interrupt signal and register are provided for reporting error conditions, described in [Section 4.2.4.11, “Error Interrupt Management Registers \(BMAN\\_ERR\\_ISR, BMAN\\_ERR\\_IER, BMAN\\_ERR\\_ISDR, BMAN\\_ERR\\_IFR, BMAN\\_ERR\\_IIR\).”](#)

Any bit that is set in the interrupt status register (BCSPn\_ISR) register can assert the portal’s interrupt line if enabled to do so in the interrupt enable register (BCSPn\_IER) of the portal. When a bit in this register is set, it remains set until cleared by writing 1 to it.

Each of the interrupt enable register (BCSPn\_IER) provides the ability to individually enable each interrupt source to assert the interrupt signal associated with its portal. If a bit location is asserted in both the interrupt status register (BCSPn\_ISR) and the interrupt enable register (BCSPn\_IER), the interrupt is asserted. To clear an interrupt, clear the interrupt source (if applicable), and then clear the associated bit in the ISR.

The interrupt status disable register (BCSPn\_ISDR) controls reporting of interrupts in BCSPn\_ISR. A bit in BCSPn\_ISR is never set if the corresponding bit in BCSPn\_ISDR is set (that is, the interrupt status is disabled).

The interrupt inhibit register (BCSPn\_IIR) is a single control that allows a portal’s interrupt signal to be inhibited without modifying the state of the BCSPn’s remaining interrupt management registers.

## Buffer Manager (BMan)

The interrupt force register (BCSP $n$ \_IFR) allows software to simulate an interrupt event. Writing a 1 to any bit in this register sets the corresponding bit in BCSP $n$ \_ISR, unless the corresponding bit in BCSP $n$ \_ISDR is also set.

For examples of how these registers can be used, see [Section 4.3.4.1, “Interrupt Control Logic.”](#)

Offset 0x400_3E00 (BCSP<n>_ISR)																															Access: w1c					
offset 0x010000																																				
range n=0..65																																				
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RCDI	RCRI	BSCN			
W																																w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Figure 4-11. BCSP Interrupt Status Registers (BCSP $n$ \_ISR)

Offset 0x400_3E40 (BCSP<n>_IER)																															Access: R/W							
offset 0x010000																																						
range n=0..65																																						
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RCDI	RCRI	BSCN					
W																																						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Figure 4-12. BCSP Interrupt Enable and Interrupt Status Disable Registers (BCSP $n$ \_IER, BCSP $n$ \_ISDR)

Offset 0x400_3F00 (BCSP<n>_IFR)																															Access: W							
offset 0x010000																																						
range n=0..65																																						
R																																						
W																																						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

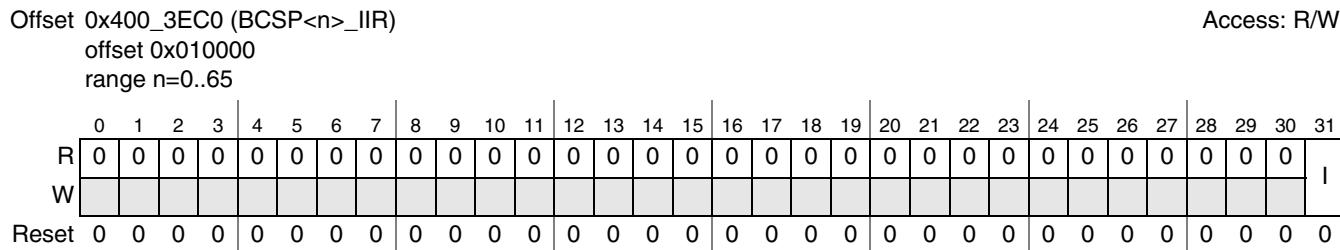
Figure 4-13. BCSP Interrupt Force Register (BCSP $n$ \_IFR)

Table 4-11. BCSP $n$ \_ISR, BCSP $n$ \_IER, BCSP $n$ \_IFR, BCSP $n$ \_ISDR Field Descriptions

Field	Description																																			
0–29	Reserved																																			
0–28	Reserved																																			
29 RCDI	Release command dispatched interrupt Asserted when a release command is dispatched for execution and an interrupt on command dispatch has been requested in that command																																			

**Table 4-11. BCSPn\_ISR, BCSPn\_IER, BCSPn\_IFR, BCSPn\_ISDR Field Descriptions (continued)**

Field	Description
30 RCRI	Release command ring threshold interrupt See <a href="#">Section 4.3.4.3, “Functional Interrupt Source, per Software Portal.”</a>
31 BSCN	Buffer pool state change notifications (BSCN) when the depletion state of a buffer pool changes See <a href="#">Section 4.3.5, “Buffer Pool State.”</a>

**Figure 4-14. BCSP Interrupt Inhibit Registers (BCSPn\_IIR)****Table 4-12. BCSPn\_IIR Field Descriptions**

Field	Description
0-30	Reserved
31 I	Interrupt inhibit 0 Interrupt not inhibited. 1 Interrupt inhibited.

## 4.2.4 BMan Configuration and Control Register Descriptions

This section describes the general configuration, control, and status registers used by software to initialize and monitor the BMan. These registers are located within the BMan and accessed as 32-bit words.

### 4.2.4.1 Block Revision Registers (BMAN\_IP\_REV\_1, BMAN\_IP\_REV\_2)

BMAN\_IP\_REV\_1 provides the unique IP block ID for the BMan block, as well as major and minor revision numbers. BMAN\_IP\_REV\_2 provides BMan block integration and configuration options.

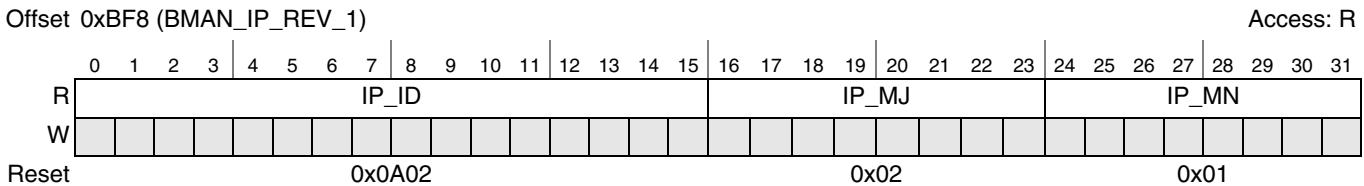


Figure 4-15. BMan IP Block Revision 1 Register (BMAN\_IP\_REV\_1)

Table 4-13. Register BMAN\_IP\_REV\_1 Field Descriptions

Field	Description
0–15 IP_ID	IP Block ID.
16–23 IP_MJ	Major revision.
24–31 IP_MN	Minor revision.

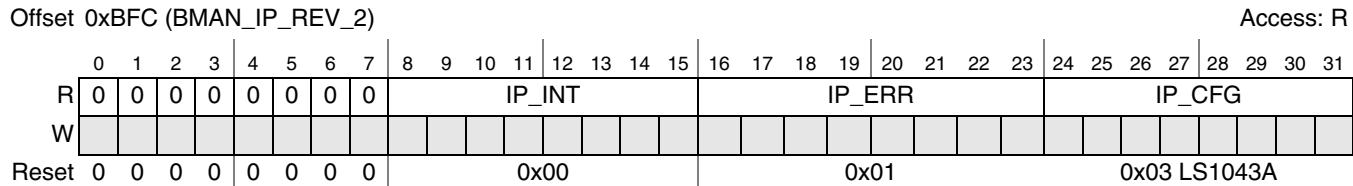


Figure 4-16. BMan IP Block Revision 2 Register (BMAN\_IP\_REV\_2)

Table 4-14. Register BMAN\_IP\_REV\_2 Field Descriptions

Field	Description
0–7	Reserved
8–15 IP_INT	Integration options
16–23 IP_ERR	Errata revision level
24–31 IP_CFG	Configuration options. 3: LS1043A - 64 pools, 2 direct connect portals, 10 software portals

#### 4.2.4.2 Memory Configuration Registers (FBPR\_BAR, FBPR\_BARE, FBPR\_AR, BMAN\_ICIDR, BMAN\_SRCID)

The BMan uses a private memory space to store free buffer proxy records (FBPRs) in external memory. This space is configured by setting the base address (FBPR\_BAR, FBPR\_BARE) and size (FBPR\_AR[SIZE]). In addition, the priority (FBPR\_AR[P]) and ICID (BMAN\_ICIDR[BMAN\_ICID]) must be configured for the memory transactions. The SRCID for memory transactions is also available to software (BMAN\_SRCID). Together the base address and size describe a region of main memory that is set aside for private use by BMan to store its FBPRs, and as such this is a non-coherent (M=0) memory region, and should be protected from access by anyone other than BMan.

Once BMan has been configured with a non-zero size (FBPR\_AR[SIZE]) it will immediately begin initializing its free list and storing to external memory. All memory configuration must be completed before initialization starts.

For configuring the base address, the FBPR\_BARE and FBPR\_BAR registers together identify the 36 most significant address bits of the 48-bit base address of the external memory window that has been allocated by software for storing data structures (FBPRs) in system memory. FBPR\_BARE contains the 16 most significant (that is, left-most) address bits, and FBPR\_BAR contains the next 20 most significant address bits.

$$\text{BASE\_ADDR} = (\text{FBPR_BARE} \ll 32 | \text{FBPR_BAR}) \ll 12 \quad \text{Eqn. 4-1}$$

$$\text{window size in bytes} = 2^{(\text{SIZE} + 1)} \quad \text{Eqn. 4-2}$$

It is possible to increase the size of the memory window after initialization but the base address **must not be changed**. For an increase in the size of the memory window to be possible, the memory space's base address must be aligned on a boundary that matches FBPR\_AR[SIZE]. For example, if the memory window is configured for 4 KB with the possibility of growing to 8 KB, then BAR[19] must be set at 0.

The maximum window size is  $2^{28+6}=16\text{GB}$ , holding  $2^{28}=256\text{M}$  FBPRs, with an FBPR holding 8 buffer pointers or  $2^{28+3}=2\text{G}$  buffer pointers in total, on average of  $2^{28+3-6}=32\text{M}$  buffers per pool.

The BMan never uses FBPR index zero.

All external memory transactions initiated by BMan will provide a 8 bit Isolation Context Identifier (ICID) along with the address of the transaction. Note that the ICID should be unique and consistent across the system. The BMan transactions target one system memory region allocated for BMan's private use. The ICID is required to allow each transaction to be authenticated and translated.

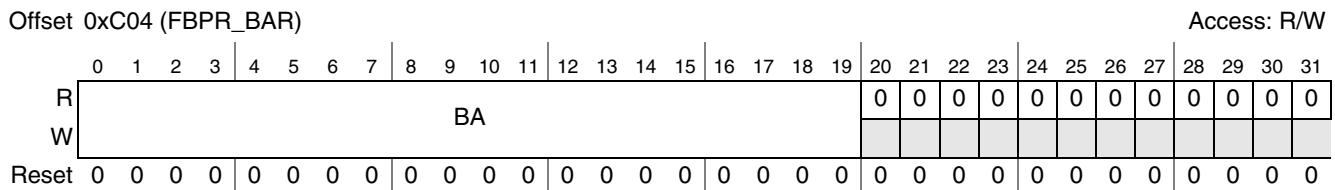
The BMan contains a fixed, non-configurable, 8-bit Source ID (BMAN\_SRCID). This ID is unique within the SoC and is attached to every external memory transaction initiated by the BMan.

Offset 0xC00 (FBPR_BARE)																	Access: R/W														
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

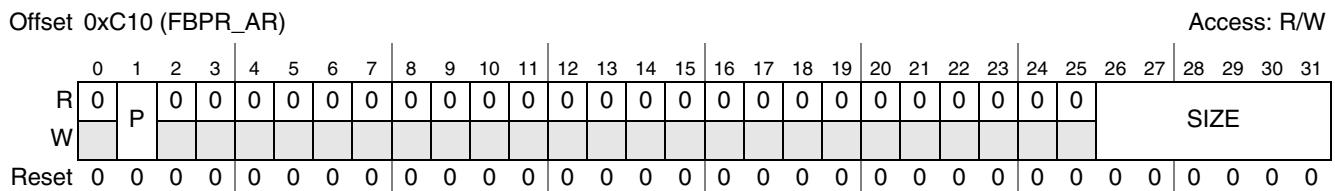
Figure 4-17. Data Structure Extended Base Address Registers (FBPR\_BARE)

**Table 4-15. Register FBPR\_BARE Field Descriptions**

Field	Description
0-15	Reserved
16-31 EBA	Extended Base Address. BARE contains the 16 most significant (that is, left-most) bits of the base address.

**Figure 4-18. Data Structure Base Address Registers (FBPR\_BAR)****Table 4-16. Register FBPR\_BAR Field Descriptions**

Field	Description
0-19 BA	Base Address. FBPR_BARE contains the most significant bits of the base address, BAR contains the next 20 most significant bits.
20-31	Reserved

**Figure 4-19. Data Structure Attribute Registers (FBPR\_AR)****Table 4-17. Register FBPR\_AR Field Descriptions**

Field	Description
0	Reserved
1 P	FBPR Transaction Priority. 0 - transactions for this data structure will be signalled with lower priority. 1 - transactions for this data structure will be signalled with higher priority.
2-25	Reserved
26-31 SIZE	Identifies the size of the window from the base address. Window size is $2^{(SIZE+1)}$ bytes. 0-10 = Reserved 11 = 4 KB 12 = 8 KB .... = $2^{(SIZE+1)}$ bytes 33 = 16 GByte 34-63 = Reserved

Offset 0xD08 (BMAN_ICIDR)																				Access: R/W												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 4-20. BMan Isolation Context Identifier Register (BMAN\_ICIDR)

Table 4-18. BMAN\_ICIDR Field Descriptions

Field	Description
0–23	Reserved
24–31 BMAN_ICID	The BMan’s Isolation Context Identifier

Offset 0xD04 (BMAN_SRCIDR)																				Access: R													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Figure 4-21. BMan Source ID Register (BMAN\_SRCIDR)

Table 4-19. BMAN\_SRCIDR Field Descriptions

Field	Description
0–23	Reserved
24–31 SRCID	BMan’s 8-bit Source ID This value is not programmable.

#### 4.2.4.3 Hardware and Software Portal Depletion Threshold Registers (BMAN\_POOLn\_SWDET, BMAN\_POOLn\_SWDXT)

The BMan can indicate to the software portals when a pool  $n$  has entered or left the depletion state (including some hysteresis) by setting the software portal depletion entry and exit thresholds (BMAN\_POOL $n$ \_SWDET and BMAN\_POOL $n$ \_SWDXT, respectively). The BMan can also indicate a depletion state to hardware clients of BMan which may enable some client dependent reaction (BMAN\_POOL $n$ \_HWDDET and BMAN\_POOL $n$ \_HWDXT for entry and exit thresholds respectively). The hardware and software threshold configurations are separate and per-pool.

For software portals, the depletion state is conveyed as a response to a software portal query (see [Section 4.3.3.2, “Response Register \(RR\) Functionality”](#)) or as a software portal functional interrupt when the depletion state transitions from “depleted” to “not depleted” or vice-versa and the interrupt is enabled for a given software portal (see [Section 4.2.3.8, “BCSP Depletion State Change Interrupt Enable Registers \(BCSPn\\_SCNm\)”](#)). There is a single software threshold configuration (entry and exit thresholds) shared amongst all software portals, per-pool.

## Buffer Manager (BMan)

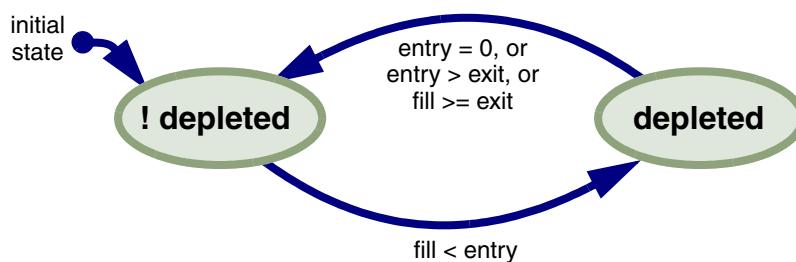
For hardware clients, depletion state is reported directly to those clients. There is a single hardware threshold configuration (entry and exit thresholds) shared amongst all hardware clients of the BMan, per-pool.

The fields in the threshold registers are combined to obtain a threshold

$$\text{THRESHOLD} = \text{COEF} * 2^{\text{EXP}}$$

**Eqn. 4-3**

which is used to determine the depletion state of the pool. The range of the threshold value is between  $2^{23} - 2^{15} = \sim 8M$  buffers per-pool and a minimum of 0. The pool enters the depleted state with respect to the software threshold when the pool fill (POOLn\_CONTENT) transitions from greater-than or equal to the entry threshold (DET), to less than the entry threshold. The pool leaves the depletion state with respect to the software threshold when the pool fill transitions from less-than the exit threshold (DXT), to greater than the exit thresholds or if the entry threshold becomes zero (disabled) or if the entry threshold is set greater than the exit threshold (disabled). (See [Figure 4-22](#).)



**Figure 4-22. Buffer Manager (BMan) threshold depletion state changes**

Offset 0x000 (BMAN\_POOL<n>\_SWDET)

Access: R/W

offset 4

range n=0..63

0x200 (BMAN\_POOL<n>\_SWDXT)

offset 4

range n=0..63

0x100 (BMAN\_POOL<n>\_HWDET)

offset 4

range n=0..63

0x300 (BMAN\_POOL<n>\_HWDXT)

offset 4

range n=0..63

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

**Figure 4-23. BMan Software Portal Depletion Threshold Register (BMAN\_POOLn\_SWDET, BMAN\_POOLn\_SWDXT, BMAN\_POOLn\_HWDET, BMAN\_POOLn\_HWDXT)**

**Table 4-20. BMAN\_POOL $n$ \_SWDET, BMAN\_POOL $n$ \_SWDXT, BMAN\_POOL $n$ \_HWDDET, BMAN\_POOL $n$ \_HWDXT Field Descriptions**

Field	Description
0–19	Reserved
20–23 EXP	Software depletion threshold exponent
24–31 COEF	Software depletion threshold coefficient Threshold is COEF * 2 <sup>EXP</sup> buffer pointers

#### **4.2.4.4 FBPR Pool Low Watermark Interrupt Threshold Register (FBPR\_FP\_LWIT)**

If the number of unused free buffer proxy records (FBPRs) as reported in FBPR\_FPC is below a threshold configured using the FBPR pool low watermark interrupt threshold (FBPR\_FP\_LWIT), the BMan will assert an error interrupt, as described in [Section 4.2.4.11, “Error Interrupt Management Registers \(BMAN\\_ERR\\_ISR, BMAN\\_ERR\\_IER, BMAN\\_ERR\\_ISDR, BMAN\\_ERR\\_IFR, BMAN\\_ERR\\_IIR\),”](#) if the interrupt is enabled. The interrupt source is automatically disabled until initialization of the FBPR free-list has completed. (See [Section 4.2.4.2, “Memory Configuration Registers \(FBPR\\_BAR, FBPR\\_BARE, FBPR\\_AR, BMAN\\_ICIDR, BMAN\\_SRCID\),”](#))

**Figure 4-24. FBPR Free Pool Low Watermark Interrupt Threshold Register (FBPR\_FP\_LWIT)**

**Table 4-21. FBPR\_FP\_LWIT Field Descriptions**

Field	Description
0-3	Reserved
4-31 TH	FBPR low watermark interrupt threshold An interrupt can be asserted (if enabled in BMAN_ERR_IER) when FBPR_FPC is below this threshold value, and is disabled when 0.

#### **4.2.4.5 Hardware and Software Portal Depletion Count Register (BMAN\_POOLn\_SDCNT, BMAN\_POOLn\_HDCNT)**

The BMan tracks the number of times any one of its  $n$  pools changes from the “not depleted” to the “depleted” state with respect to hardware or software depletion thresholds in the BMAN\_POOL $n$ \_HDCNT and BMAN\_POOL $n$ \_SDCNT registers, per-pool (see [Section 4.2.4.3](#),

## Buffer Manager (BMan)

“Hardware and Software Portal Depletion Threshold Registers (BMAN\_POOLn\_SWDET, BMAN\_POOLn\_SWDXT)” for details on depletion state transitions and setting depletion thresholds).

Offset 0x400 (BMAN_POOL<n>_SDCNT)	Access: RR																																																																																																																																												
offset 4																																																																																																																																													
range n=0..63																																																																																																																																													
0x500 (BMAN_POOL<n>_HDCNT)																																																																																																																																													
offset 4																																																																																																																																													
range n=0..63																																																																																																																																													
<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td> </tr> <tr> <td>R</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>CNT</td><td></td><td></td><td></td><td></td> </tr> <tr> <td>W</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CNT					W																																			Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																																																																																																														
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CNT																																																																																																															
W																																																																																																																																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																								

**Figure 4-25. BMan Software Portal Depletion Count Registers (BMAN\_POOLn\_SDCNT, BMAN\_POOLn\_HDCNT)**

**Table 4-22. BMAN\_POOLn\_SDCNT, BMAN\_POOLn\_HDCNT Field Descriptions**

Field	Description
0–23	Reserved
24–31 CNT	8-bit linear count indicating number of times pool entered depletion Terminal count of 8'd255 until cleared by a read (read-reset).

### 4.2.4.6 Pool Content Registers (BMAN\_POOLn\_CONTENT, FBPR\_FPC)

The BMan reports a snapshot of the number of free buffers available in pool *n* (BMAN\_POOL*n*\_CONTENT). Note that this is a snapshot only. It is inherently out-of-date by the time software receives the information, unless there is no activity through BMan at the time.

The free buffer proxy record (FBPR) free pool count register (FBPR\_FPC) is a snapshot of the number of FBPRs that are available to hold buffer pointers.

Offset 0x600 (BMAN_POOL<n>_CONTENT)	Access: R																																																																																																																																												
offset 4																																																																																																																																													
range n=0..63																																																																																																																																													
0x800 (FBPR_FPC)																																																																																																																																													
offset 4																																																																																																																																													
range n=0..0																																																																																																																																													
<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td> </tr> <tr> <td>R</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>W</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>Reset</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> </table>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	R																																			W																																			Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																																																																																																														
R																																																																																																																																													
W																																																																																																																																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																								

**Figure 4-26. BMan Pool Content Register (BMAN\_POOLn\_CONTENT, FBPR\_FPC)**

**Table 4-23. BMAN\_POOLn\_CONTENT Field Descriptions**

Field	Description
0–31 FB	Number of buffers

#### **4.2.4.7 Free List Head Pointer Registers (BMAN\_POOLn\_HDPTR, FBPR\_HDPTR)**

The BMan free list head pointer registers (BMAN\_POOL $n$ \_HDPTR) are available to assist in debugging. The value read from the FBPR\_HDPTR and BMAN\_POOL $n$ \_HDPTR is an index that points to a Free Buffer Proxy Record (FBPR) that is at the head of the singly-linked list (the free list) of buffer pointers in the BMan’s private memory space. (See [Section 4.3.6, “Free Buffer Proxy Records \(FBPRs\).”](#)) The location of byte zero of the FBPR at the head of the free list in BMan’s private memory space is

$$\text{ADDR} = (\text{LAST\_IDX} * 64) + \text{BASE\_ADDR} \quad \text{Eqn. 4-4}$$

where **BASE\_ADDR** is calculated as [Equation 4-1](#).

Index zero is not used as an index for storing FBPRs and indicates an empty free list. This does not indicate that the pool is empty because there may still be buffers in the BMan's stockpile.

The FBPR free list head pointer (FBPR\_HDPTR) behaves in exactly the same way as the regular pool head pointers (BMAN\_POOLn\_HDPTR), except that the FBPRs in this free list contain FBPR pointers.

**Figure 4-27. BMan Pool Content Registers (BMAN\_POOLn, HDPTR, FBPR, HDPTR)**

**Table 4-24. BMAN POOL $n$  HDPTR, FBPR HDPTR Field Descriptions**

Field	Description
0–3	Reserved
4–31 LAST_IDX	28-bit index into the BMan system memory space Offset from 0 of last stored FBPR

#### **4.2.4.8 Command Performance Monitor Configuration Registers (CMD\_PMn\_CFG)**

The BMan can monitor performance of buffer acquire and release requests. The command performance monitor configuration registers (CMD\_PM $n$ \_CFG) set the configuration of eight monitors that allow the scope of the monitoring to be adjusted based on service delays, number of buffers in a request, request source, source FIFO full state when serviced, and requested pool. The behavior of these monitors is described in [Section 4.3.7, “Performance Monitor.”](#) The service delay condition is configured such that if a command is not serviced for longer than **DELAY** cycles, a performance event is signalled.

$$\text{DELAY} = \text{DCOEF} * 2^{\text{DEXP}} \quad \text{Eqn. 4-5}$$

## Buffer Manager (BMan)

Offset 0x900 (BMAN\_CMD\_PM<n>\_CFG) Access: R/W  
offset 4  
range n=0..7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCOEFF	DEXP			0	0	0	0	CBTH		R	C					PRS															
W					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Reset 0

Figure 4-28. BMAN Performance Monitor Configuration Registers (CMD\_PMn\_CFG)

Table 4-25. CMD\_PMn\_CFG Field Descriptions

Field	Description
0-3 DCOEF	Command wait time threshold, coefficient A clock count value of DELAY = DCOEF x $2^{DEXP}$ is used as a threshold for reporting a command service interval.
4-6 DEXP	Command wait time threshold, exponent See DCOEF field.
7-10	Reserved
11-13 CBTH	Acquire command buffer threshold Sets the minimum threshold as CBTH + 1 for number of buffers in an acquire command for it to be monitored
14 RF	Response service FIFO full 1: the event will only be signalled if the FIFO selected was full when the command was removed, indicating back pressure was occurring from the BMan's client. 0: the response FIFO full state does not affect the performance monitor outcome
15 CF	Command service FIFO full 1: the event will only be signalled if the FIFO selected was full when the command was removed, indicating back pressure was being exerted by the BMan on the client. 0: the command FIFO full state does not affect the performance monitor outcome
16-23 PRS	Pool ID at start of selected contiguous range Selects the first pool in a contiguous range of pools for command performance monitoring.
24-31 PRE	Pool ID at end of selected contiguous range Selects the last pool in a contiguous range of pools for command performance monitoring.

Offset 0x940 (BMAN\_CMD\_PM<n>\_CFG\_CFIFO) Access: R/W  
offset 4  
range n=0..7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																

Reset 0

Figure 4-29. BMAN Performance Monitor Configuration Registers (CMD\_PMn\_CFG\_CFIFO)

**Table 4-26. CMD\_PM<sub>n</sub>\_CFG\_CFIQ Field Descriptions**

Field	Description
0-31 CFIFO	Command service FIFO Selects one or more portal FIFOs to be monitored. Bit [31-n] enables hardware command from portal <i>n</i> . For example, bit [31] enables hardware portal 0 for monitoring. Bit [0] selects all software portals for monitoring.

#### 4.2.4.9 Free List Performance Monitor Configuration Register (BMAN\_FL\_PM<sub>n</sub>\_CFG)

The BMan can monitor performance of free list operations. The free list performance monitor configuration registers (FL\_PM<sub>n</sub>\_CFG) set the configuration of eight monitors that allow the scope of the monitoring to be adjusted based on the pool. The behavior of these monitors is described in [Section 4.3.7, “Performance Monitor.”](#)

Offset 0x920 (BMAN\_FL\_PM<n>\_CFG) Access: R/W  
offset 4  
range n=0..7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 4-30. BMAN Performance Monitor Configuration Register (BMAN\_FL\_PM\_CFG)****Table 4-27. BMAN\_FL\_PM\_CFG Field Descriptions**

Field	Description
0 FP	Monitor free pool of FBPRs When set, the monitor associated with this register will monitor activity for the list of free FBPR's (that is, FBPR's not belonging to any buffer pools)
1-15	Reserved
16-23 PRS	Pool ID at start of selected contiguous range. Selects the first pool in a contiguous range of buffer pools for monitoring, unless field FP is set.
24-31 PRE	Pool ID at end of selected contiguous range. Selects the last pool in a contiguous range of buffer pools for monitoring, unless field FP is set.

#### 4.2.4.10 Idle State and Stop Registers (STATE\_IDLE, STATE\_STOP)

The BMan provides registers for monitoring the idle state of the BMan and forcing the BMan into the idle state gracefully.

Offset 0x960 (STATE\_IDLE) Access: R

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A	S	E	I	
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

**Figure 4-31. BMAN Idle Status Register (STATE\_IDLE)**

**Table 4-28. STATE\_IDLE Field Descriptions**

Field	Description
0-27	Reserved
28 A	Stop Acknowledged A stop request (bit [29] S) has been acknowledged when set. The BMan's FIFOs are empty and the BMan is idle (bit [30] E and bit [31] I are set). Will remain set until the stop request is cleared.
29 S	Stop Request A stop has been requested when set. Can be asserted as a result of a request from STATE_STOP or a request from the system.
30 E	Empty All request and response FIFOs within the BMan are empty when set.
31 I	Idle The BMan is currently idle and this bit is set when: - there are no outstanding external memory accesses, and - the BMan state machines are in their idle state.

Offset 0x964 (STATE\_STOP)

Access: R/W

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	D	P	F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	S	0	0	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

**Figure 4-32. BMAN Stop Register (STATE\_STOP)****Table 4-29. STATE\_STOP Field Descriptions**

Field	Description
0 D	Stop servicing hardware portal requests. Gracefully ceases servicing hardware portals.
1 P	Stop servicing software portal requests. Gracefully ceases servicing software portals. This bit will only take effect when BMan and its software portals go idle. Once this bit becomes set, BMan will service all in-progress software portal commands, and will continue to accept new commands so long as it has at least one software portal command still in progress. When all in-progress work is completed, and no further software portal commands have arrived while the in-progress work is underway, BMan will cease accepting new software portal work and will indicate its idle status via the STATE_IDLE register.
2 F	Stop servicing fetch/flush requests from the stockpile. Gracefully cease external memory access.
3-28	Reserved
29 S	Stop Request A graceful stop has been requested.
30-31	Reserved

#### 4.2.4.11 Error Interrupt Management Registers (BMAN\_ERR\_ISR, BMAN\_ERR\_IER, BMAN\_ERR\_ISDR, BMAN\_ERR\_IFR, BMAN\_ERR\_IIR)

The BMan error interrupt management registers (BMAN\_ERR\_ISR, BMAN\_ERR\_IER, BMAN\_ERR\_ISDR, BMAN\_ERR\_IFR, BMAN\_ERR\_IIR) contain error interrupts that pertain to situations that affect all pools. There is a single dedicated error interrupt signal from the BMan. Separate interrupt signals and registers are provided for reporting functional conditions to individual software portals, described in [Section 4.2.3.9, “BCSP Functional Interrupt Management Registers \(BCSPn\\_ISR, BCSPn\\_IER, BCSPn\\_ISDR, BCSPn\\_IFR, BCSPn\\_IIR\).”](#)

Any bit that is set in the interrupt status register (BMAN\_ERR\_ISR) register can assert the error interrupt line if enabled to do so in the interrupt enable register (BMAN\_ERR\_IER). When a bit in this register is set, it remains set until cleared by writing 1 to it.

The interrupt enable register (BMAN\_ERR\_IER) provides the ability to individually enable each interrupt source to assert the error interrupt signal. If a bit location is asserted in both the interrupt status register (BMAN\_ERR\_ISR) and the interrupt enable register (BMAN\_ERR\_IER), the interrupt is asserted. To clear an interrupt, clear the interrupt source (if applicable), and then clear the associated bit in the ISR.

The interrupt status disable register (BMAN\_ERR\_ISDR) controls reporting of interrupts in BMAN\_ERR\_ISR. A bit in BMAN\_ERR\_ISR is never set if the corresponding bit in BMAN\_ERR\_ISDR is set (that is, the interrupt status is disabled).

The interrupt inhibit register (BMAN\_ERR\_IIR) is a single control that allows a portal’s interrupt signal to be inhibited without modifying the state of the remaining interrupt management registers.

The interrupt force register (BMAN\_ERR\_IFR) allows software to simulate an interrupt event. Writing a 1 to any bit in this register sets the corresponding bit in BMAN\_ERR\_ISR, unless the corresponding bit in BMAN\_ERR\_ISDR is also set.

For examples of how these registers can be used, see [Section 4.3.4.1, “Interrupt Control Logic.”](#)

Offset 0xE00 (BMAN_ERR_ISR)																																	Access: w1c								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31										
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
W																																	w1c	w1c	w1c	w1c	w1c				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-33. BMan Error Interrupt Status Register (BMAN\_ERR\_ISR)

Offset 0xE04 (BMAN_ERR_IER) 0xE08 (BMAN_ERR_ISDR)																																		Access: R/W								
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31											
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
W																																										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-34. BMan Error Interrupt Enable Register (BMAN\_ERR\_IER)

Offset 0xE10 (BMAN_ERR_IFR)																															Access: W	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 4-35. BMan Error Interrupt Force Register (BMAN\_ERR\_IFR)

Table 4-30. BMAN\_ERR\_ISR Field Descriptions

Field	Description
0–24	Reserved
25 EMAI	External memory access interrupt Asserted when a corrupted FBPR is read back from system memory, some related error information are captured when the error occurs, see <a href="#">Section 4.2.4.16, “Access Error Capture and Address Registers (BMAN_AECR, BMAN_AEAR).”</a> Some related error information are captured.
26 EMCI	External memory corruption interrupt Asserted when a corrupted FBPR is read back from system memory and detected by the BMan, as described in <a href="#">Section 4.2.4.15, “Corruption Error Capture and Address Registers (BMAN_CECR, BMAN_CEAR).”</a> Some related error information are captured.
27 IVCI	Invalid verb command interrupt Asserted when an unrecognized command verb is received in any software portal. The portal in which the error was detected, and some related error information, are captured when the error occurs, see <a href="#">Section 4.2.4.14, “Error Capture Information Register (BMAN_ECIR).”</a>
28 FLWI	FBPR low watermark interrupt Asserted when the FBPR free pool occupancy is below the configured threshold, see <a href="#">Section 4.2.4.4, “FBPR Pool Low Watermark Interrupt Threshold Register (FBPR_FP_LWIT).”</a>
29 MBEI	Multi-bit ECC error interrupt Asserted when a multi bit ECC error is detected in the BMan internal memories.
30 SBEI	Single-bit ECC error interrupt Asserted when the number of single bit ECC errors detected in the BMan internal memories exceeds the configured threshold described in <a href="#">Section 4.2.4.12, “Single Bit ECC Error Threshold Register (BMAN_SBET).”</a>
31 BSCN	Buffer pool availability state change interrupt Asserts when a pool enters or leaves the empty state as described in <a href="#">Section 4.3.5, “Buffer Pool State.”</a>

Offset 0xE0C (BMAN_ERR_IIR)																															Access: R/W	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	I		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

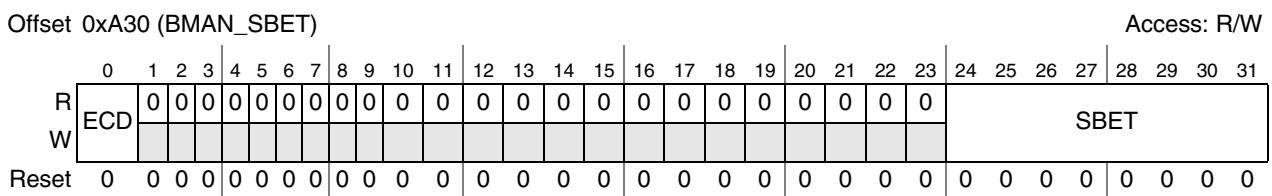
Figure 4-36. BMan Error Interrupt Inhibit Register (BMAN\_ERR\_IIR)

**Table 4-31. BMAN\_ERR\_IIR Field Descriptions**

Field	Description
0-30	Reserved.
31 I	Interrupt Inhibit 0 Interrupt not inhibited. 1 Interrupt inhibited.

#### **4.2.4.12 Single Bit ECC Error Threshold Register (BMAN\_SBET)**

The BMan internal SRAMs used to implement the stockpile support ECC. When the single-bit error threshold (SBET) is exceeded a single bit error interrupt (SBEI) will be asserted.



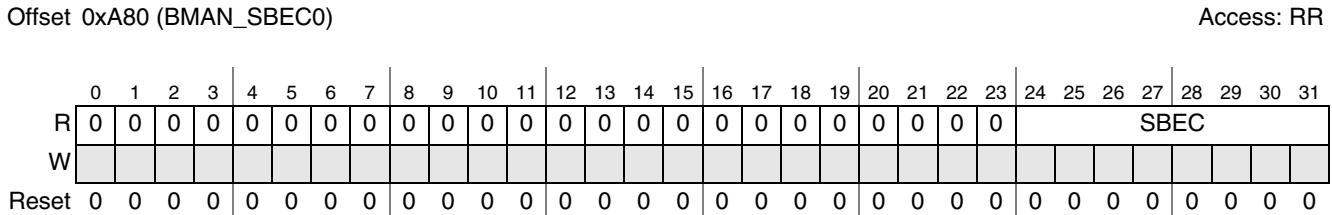
**Figure 4-37. BMan Single Bit Error Threshold Register (BMAN\_SBET)**

**Table 4-32. BMAN\_SBET Field Descriptions**

Field	Description
0 ECD	Error correction and detection disable This field allows ECC error correction and detection to be disabled for the BMan's internal stockpile memories. 0 ECC correction and ECC error reporting are enabled 1 ECC correction and ECC error reporting are disabled
1-23	Reserved
24-31 SBET	Single-bit error threshold Threshold value for the number of single bit ECC errors in the BMan internal memories before reporting an error.

#### **4.2.4.13 Single Bit ECC Error Count Register (BMAN\_SBEC0)**

Provides a count of the number of single-bit ECC errors. The count does not roll-over, so reaches a maximum of 8'd255 and holds until cleared on a read access.



**Figure 4-38. BMan Single Bit Error Count Registers (BMAN\_SBEC0-1)**

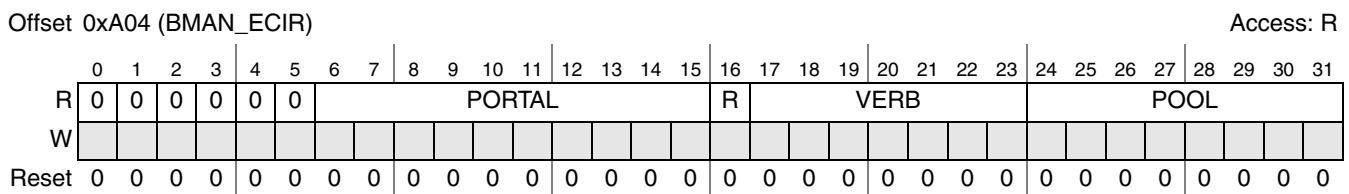
**Table 4-33. BMAN\_SBEC0-1 Field Descriptions**

Field	Description
0-23	Reserved
24-31 SBEC	Single-bit error count BMAN_SBEC0: Stockpile memory This register is cleared on read. When the count reaches its max value (0xFF), it will stick at that value without rollover until this register is read.

#### 4.2.4.14 Error Capture Information Register (BMAN\_ECIR)

An unrecognized command verb issued to a software portal will result in an error status interrupt (bit IVCI described in [Section 4.2.4.11, “Error Interrupt Management Registers \(BMAN\\_ERR\\_ISR, BMAN\\_ERR\\_IER, BMAN\\_ERR\\_ISDR, BMAN\\_ERR\\_IFR, BMAN\\_ERR\\_IIR\)](#)) if enabled to do so. Information related to the exception, such as the software portal and ring that issued the command, pool ID, and number of buffers requested will be saved in the BMAN\_ECIR register.

This register is assigned when ERR\_ISR.IVCI is set and cleared when ERR\_ISR.IVCI is cleared.

**Figure 4-39. Free Buffer Proxy Record Free Pool Count Register (BMAN\_ECIR)****Table 4-34. BMAN\_ECIR Field Descriptions**

Field	Description
0-5	Reserved
6-15 PORTAL	Software portal associated with the invalid command.
16 R	Software ring associated with the invalid command 1: RCR 0: Command
17-23 VERB	Verb associated with the invalid command, excluding alternating valid bit Includes the command and the number of buffers requested, if appropriate. See <a href="#">Section 4.3.3.1.3, “Acquire and Query Commands”</a> and <a href="#">Section 4.3.3.3.2, “Release Commands”</a> for verb definitions.
24-31 POOL	Pool ID associated with the invalid command.

#### 4.2.4.15 Corruption Error Capture and Address Registers (BMAN\_CECR, BMAN\_CEAR)

The BMan memory region where FBPRs are stored is private to the BMan. The BMan has rudimentary checks to determine if an FBPR has been scribbled upon. Information about where the issue occurred is captured in the BMAN\_CECR and BMAN\_CEAR when the EMCI error interrupt bit is set (BMAN\_ERR\_ISR[EMCI]). The free list for the pool on which the problem was detected on read is treated as corrupted.

When a free list is detected as corrupted, all buffer pointers and FBPRs in that free-list are lost. This can occur on a regular pool's free list or on the FBPR free list. In either case, there are still some pointers left in the stockpile and buffer pointers and FBPRs that are currently in use may be released back to the pool. (BMAN\_POOLn\_HDPT will be cleared and BMAN\_POOLn\_CONTENT will be set to the current stockpile fill or FBPR\_HDPT will be cleared and FBPR\_FPC will be set to the current stockpile fill according to which free list the error was detected on.) A new free list will be created and the BMan will continue to operate correctly though system performance may be significantly affected by the loss of buffers. There is no mechanism to recover the lost buffers.

The checks on FBPRs read from external memory are: (a) that the address that is read matches the address stored in the FPA field of the FBPR, (b) that the buffer pool id stored in the BPID field of the FBPR matches the expected pool, and (c) (for the FBPR free list only) that the first FBPR pointer stored in the FBPR matches the FPA field of the FBPR. This provides a basic sanity check against the entire FBPR being overwritten, the memory subsystem being misconfigured or reconfigured during operation or the BMan's base address being changed (FBPR\_BAR/FBPR\_BARE). These registers are assigned when ERR\_ISR.EMCI is set and cleared when ERR\_ISR.EMCI is cleared.

Offset 0xA34 (BMAN_CECR)																															Access: R	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LID	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 4-40. BMan Error Fetch Capture Register (BMAN\_CECR)

Table 4-35. BMAN\_CECR Field Descriptions

Field	Description
0-23	Reserved
24-31 LID	List ID The BMan pool- or free-list data structure (FBPRs) affected by the EMCI interrupt. LID = 0 to 63; List for Pool 0, or Pool 1, .. , or Pool 63 LID = 255 (0xFF); List of free FBPRs

## Buffer Manager (BMan)

Offset 0xA38 (BMAN_CEAR)																														Access: R			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	ADDR																												
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-41. BMan Corruption Error Address Register (BMAN\_CEAR)

Table 4-36. BMAN\_CEAR Field Descriptions

Field	Description
0-3	Reserved
4-31 ADDR	Address The 28-bit FBPR offset that caused the EMCI interrupt.

### 4.2.4.16 Access Error Capture and Address Registers (BMAN\_AECR, BMAN\_AEAR)

When the BMan accesses external memory to read FBPRs, external read errors can be returned by the memory sub-system. Information about where the error occurred are captured in BMAN\_AECR and BMAN\_AEAR when the EMAI error interrupt bit is set (BMAN\_ERR\_ISR[EMAI]). An interrupt may also be asserted by the QMan in QMAN\_ERR\_ISR. The free list for the pool on which the read error occurred is treated as corrupted.

When a free list is detected as corrupted, all buffer pointers and FBPRs in that free-list are lost. This can occur on a regular pool's free list or on the FBPR free list. In either case, there are still some pointers left in the stockpile and buffer pointers and FBPRs that are currently in use may be released back to the pool. (BMAN\_POOLn\_HDPT will be cleared and BMAN\_POOLn\_CONTENT will be set to the current stockpile fill or FBPR\_HDPT will be cleared and FBPR\_FPC will be set to the current stockpile fill according to which free list the error was detected on.) A new free list will be created and the BMan will continue to operate correctly though system performance may be significantly affected by the loss of buffers. There is no mechanism to recover the lost buffers.

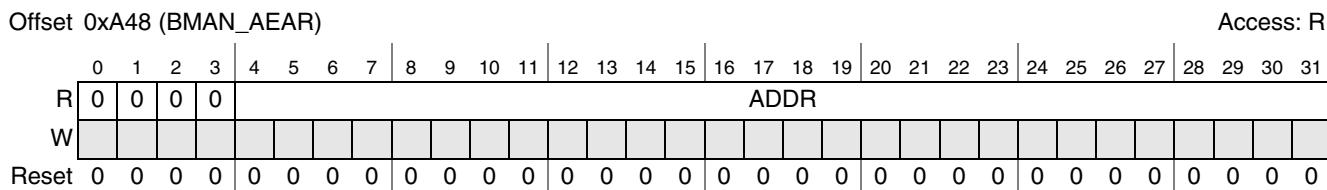
These registers are assigned when ERR\_ISR.EMAI is set and cleared when ERR\_ISR.EMAI is cleared.

Offset 0xA44 (BMAN_AECR)																														Access: R			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LID			
W																																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4-42. BMan Error Fetch Capture Register (BMAN\_AECR)

**Table 4-37. BMAN\_AECR Field Descriptions**

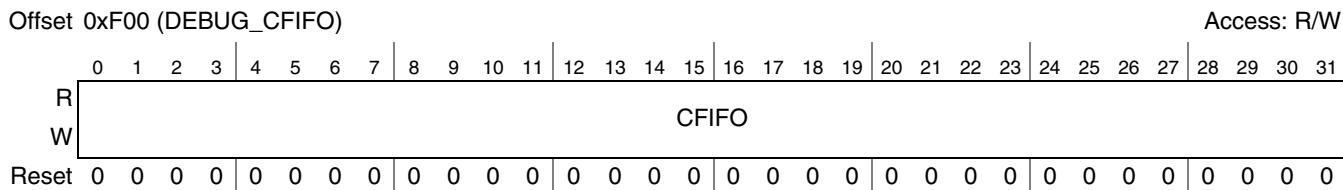
Field	Description
0-23	Reserved
24-31 LID	List ID The BMan pool- or free-list data structure (FBPRs) affected by the EMAI interrupt. LID = 0 to 63; List for Pool 0, or Pool 1, .. , or Pool 63 LID = 255 (0xFF); List of free FBPRs

**Figure 4-43. BMan Error Fetch Address Register (BMAN\_AEAR)****Table 4-38. BMAN\_AEAR Field Descriptions**

Field	Description
0-3	Reserved
4-31 ADDR	Address The 28-bit FBPR offset that caused the EMAI interrupt.

#### 4.2.4.17 BMan Debug: Command FIFO Disable Register (DEBUG\_CFIIFO)

This register is for debug purposes. Asserting one of the bits will disable the corresponding command FIFO so that requests from that source will no longer be serviced.

**Figure 4-44. BMAN Command FIFO Disable Registers (DEBUG\_CFIIFO)****Table 4-39. DEBUG\_CFIIFO Field Descriptions**

Field	Description
0-31 CFIFO	Command service FIFO disable Selects one or more portal FIFOs to be disabled. Bit [n] disables hardware commands from portal n. For example, bit [0] disables hardware portal 0 so it will not be serviced. Bit [31] selects all software portals to be disabled. 0: portal enabled, 1: portal disabled

#### 4.2.4.18 BMan Debug: State Machine Disable (DEBUG\_FSM)

This register is for debug purposes. Asserting a bit will disable the state machine. The service state machine services the command FIFOs and depends on the state of the stockpile. The fetch/flush state machine monitors the state of the stockpile and performs fetches and flushes for the stockpile to external memory lists. Disabling the fetch/flush state machine will prevent all external memory accesses.

Offset 0xF04 (DEBUG_FSM)																																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F	S	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 4-45. BMAN State Machine Disable Registers (DEBUG\_FSM)

Table 4-40. DEBUG\_FSM Field Descriptions

Field	Description
0-29	Reserved
30 F	Fetch/Flush state machine disable 0: state machine enabled, 1: state machine disabled
31 S	FIFO service state machine disable 0: state machine enabled, 1: state machine disabled

## 4.3 BMan Functional Description

### 4.3.1 Direct Connect Portals (DCPs)

Direct connect portals (DCPs) connect one or more hardware modules within the SoC directly to the BMan. Examples of these hardware modules include the FMan and SEC.

### 4.3.2 External Memory Interfaces

The BMan shares the QMan's interface to external memory. The BMan does not communicate directly with the external memory, but is mediated by the QMan.

### 4.3.3 Software Portal Components

The software portals are used by software running on processor cores to communicate with the BMan. Software can interact through the software portals with the BMan in the following ways:

- Query the status of the pools and acquire buffer pointers through the software portal command registers (BCSPn\_CR) and response registers (BCSPn\_RR0 and BCSPn\_RR1),
- Release buffers to pools through the software portal release command rings (RCR),
- Configure the software portal buffer depletion interrupt behavior (BCSPn\_SCNm), and
- Configure the software portal itself and how software will interact with it.

Matching this functionality, the memory area occupied by each software portal is divided into the following separate regions:

- a 64 KB region, aligned on a 64 KB boundary, that contains registers and structures which are 64 bytes in size. This area of the portal should be configured as cache-enabled in the processor core using the portal. The cache-enabled area of the portal contains the following major structures:
  - Command register ([Section 4.3.3.1, “Command Register \(CR\) Functionality”](#))
  - Response registers ([Section 4.3.3.2, “Response Register \(RR\) Functionality”](#))
  - Release command ring ([Section 4.3.3.3, “Release Command Ring \(RCR\) Functionality”](#))
- a 64 KB region, aligned on a 64 KB boundary, that contains only 32-bit wide registers that are accessed using 4-byte transactions. Configure this area as cache-inhibited and guarded in the processor core using the software portal.

[Section 4.2.1, “BMan Software Portal Memory Map”](#) contains a detailed description of all of the software portal’s resources, and [Section 4.2.3, “BMan Software Portal \(BCSP\) Register Descriptions,”](#) contains a detailed description of these resources. The following sub-sections contain a detailed description of the functionality provided by the software portals.

### 4.3.3.1 Command Register (CR) Functionality

The command register (CR) allows software to issue buffer acquire and buffer pool query commands to the BMan.

The CR is 64 bytes in the cache-enabled area of BMan’s software portal. Its format is described in [Section 4.2.3.1, “BCSP Command Registers \(BCSPn\\_CR\).”](#)

Only one command can be issued to the CR at a time. After a command is issued, software must wait for the CR to become available again before issuing a new command to it. When the response to the previous command becomes valid in the response register (see [Section 4.3.3.2, “Response Register \(RR\) Functionality”](#)) associated with that command the CR is available.

#### 4.3.3.1.1 Writing a Command into the CR

To write a command into the CR, software must perform the following tasks:

1. Zero out the command cache line (Use DC ZVA for example)
2. Write all words of the command other than word 0.
3. Issue a synchronizing instruction (Use DMB st for example)
4. Write word 0 (containing the command verb and alternating valid bit).
5. Flush the command from cache to BMan (Use DC CVAC for example).

#### NOTE

The exact order of the above sequence is required to ensure that a castout of a partially completed command from the processor’s cache always writes a 0 (null) command verb, preventing the BMan from attempting to execute this partial command.

Using the alternating polarity valid bit in the CR is required to protect against the possibility of a castout followed by a speculative read from the processor's cache when software has not completed writing a command into the CR. The BMan does not modify CR data. See [Section 4.3.3.1.2, “Benefits of Using Alternating Polarity,”](#) for more details on alternating polarity.

The BMan expects the first command written after reset to have a 1 in its alternating valid bit. Software must alternate the current valid bit polarity each time a new command is written to the CR. The BMan will not execute a command in the CR until the alternating valid bit is the expected polarity and the command verb is non-zero.

If a command verb is invalid the BMan issues a response with the same invalid command verb. An error interrupt is signalled using the interrupt status register IVCI bit (see [Section 4.2.4.11, “Error Interrupt Management Registers \(BMAN\\_ERR\\_ISR, BMAN\\_ERR\\_IER, BMAN\\_ERR\\_ISDR, BMAN\\_ERR\\_IFR, BMAN\\_ERR\\_IIR\)”](#)) and debug information is captured in the BMAN\_ECIR registers (see [Section 4.2.4.14, “Error Capture Information Register \(BMAN\\_ECIR\)”](#)). Subsequent commands from this client continue to be processed normally.

#### 4.3.3.1.2 Benefits of Using Alternating Polarity

Using alternating polarity valid bits eliminates the need for a memory synchronization instruction between the flush instruction after writing the command verb to the command register (CR) and a subsequent read of the response register (RR0 or RR1). Alternating the response register location used by each command allows software to tell the difference between a response to the previously completed command and the most recent command.

#### 4.3.3.1.3 Acquire and Query Commands

The acquire command is able to request 1–8 buffers from the BMan, and is written by software into the command register (see [Section 4.2.3.1, “BCSP Command Registers \(BCSPn\\_CR\)”](#)). The associated response to this command is described in [Section 4.3.3.2.2, “Acquire Response.”](#)

The query command requests a snapshot of the availability and depletion state of each of the BMan's buffer pools. The associated response to this command is described in [Section 4.3.3.2.3, “Query Response.”](#)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	BPID														
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

Figure 4-46. Acquire Command Format

**Table 4-41. Acquire Command Format Field Descriptions**

Field	Description
0 VERB	Bit 0 (msb): Alternating valid bit. See <a href="#">Section 4.2.3.1, “BCSP Command Registers (BCSPn_CR).”</a> Bit 1-7 (lsbs): Command verb. Specifies the type of command to be executed. Unused values reserved. Bit 1-3: Command type 1 = Acquire buffers. 4 = Query buffer pool state, depletion and availability. Bit 4-7: Number of buffers to acquire. (Must be 0 for query command.) 0 = Zero buffers 1 = One buffer .... 8 = Eight buffers
1 BPID	Buffer Pool ID Buffer pool from which the buffers should be acquired. (Must be 0 for query command.)
2-63	Reserved

### 4.3.3.2 Response Register (RR) Functionality

Every command issued in the CR results in a response, which is always returned in the RR $n$  (either RR0 or RR1).

The response registers (RR $n$ ) functions and features are as follows:

- Software uses the response registers (RR0 and RR1) to read the response to commands issued from software to the BMan in the command register (CR).
- 64 bytes in size
- Contained in the cache-enabled area of BMan’s software portal
- Format described in [Section 4.2.3.2, “BCSP Response Registers \(BCSPn\\_RRm\).”](#)

#### 4.3.3.2.1 Determining the Appropriate Response Register

The RR used for the response is indicated by the polarity of the valid bit in that command. For example, if the valid bit in the command is 0, then RR0 is used to return the response to that command. In this way, the RR used to return a response alternates with every command issued, starting with RR1 after reset (because the polarity of the valid bit is 1 after reset).

The BMan contains a single internal storage register for command responses. When this internal register contains a valid response, it is visible in either the RR0 or RR1 location (as indicated by the valid bit polarity in the command that initiated the response). [Table 4-42](#) shows the different internal storage register conditions and results.

**Table 4-42. Internal Storage Register Conditions/Results**

Condition	Result
RR0 contains a valid response.	A read of RR1 returns all zeros data.
RR1 contains a valid response.	A read of RR0 returns all zeros data.

**Table 4-42. Internal Storage Register Conditions/Results (continued)**

<b>Condition</b>	<b>Result</b>
After reset, and whenever a write of a valid command to the CR is detected	The response held in the internal register is cleared, and reads from both RR locations return all zeros data.
A command completes and its response is placed in the internal response register.	The response: <ul style="list-style-type: none"> <li>Is available for reads in the RR location indicated by the command's valid bit polarity</li> <li>May be read multiple times, and remains available for reads until a new command with a valid bit of the expected polarity and a non-zero command verb are written to the CR</li> </ul>

#### 4.3.3.2.2 Acquire Response

The acquire response returns 1–8 buffers from the BMan to software in response to an Acquire command, and is returned in the response register (see [Section 4.2.3.2, “BCSP Response Registers \(BCSPn\\_RRm\)”](#)).

Consider the following:

- If the buffer pool specified in the acquire command contains enough buffers to satisfy the number of buffers requested in the command, the acquire response contains exactly the number of buffers requested (1–8).
- If the specified buffer pool does not contain enough buffers to satisfy the request immediately, the response returns no buffers (that is, response VERB = 0x10).
- If the acquire command is invalid, or there is an error processing the command, a different response for each event results (see [Table 4-43](#)).
- An acquire from non-existent pool, or pool 0xff, returns the same result as an empty pool.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	BPID	BUF0								BUF1					
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
		BUF2								BUF3					
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
		BUF4								BUF5					
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
		BUF6								BUF7					

**Figure 4-47. Acquire Response Format**

**Table 4-43. Acquire Response Format Field Descriptions**

<b>Field</b>	<b>Description</b>
0 VERB	Bit 0 (msb): Alternating valid bit. See <a href="#">Section 4.2.3.2, “BCSP Response Registers (BCSPn_RRm).”</a> Bit 1-7 (lsbs): Response verb. Specifies the type of command that has been executed. Unused values are reserved. Bit 1-3: Response type 1 = Acquire buffers 6 = Invalid command 7 = Stockpile ECC error Bit 4-7: Number of buffers, maximum 8 0 = Zero buffers 1 = One buffer .... 8 = Eight buffers
1 BPID	Buffer Pool ID Buffer pool from which the returned buffers were acquired.
2-7 BUF0	Buffer pointer for the 1st of up to 8 buffers returned in this response.
8-9	Reserved
10-15 BUF1	Buffer pointer for the 2nd of up to 8 buffers returned in this response.
16-17	Reserved
18-23 BUF2	Buffer pointer for the 3rd of up to 8 buffers returned in this response.
24-25	Reserved
26-31 BUF3	Buffer pointer for the 4th of up to 8 buffers returned in this response.
32-33	Reserved
34-39 BUF4	Buffer pointer for the 5th of up to 8 buffers returned in this response.
40-45	Reserved
46-47 BUF5	Buffer pointer for the 6th of up to 8 buffers returned in this response.
48-49	Reserved
50-55 BUF6	Buffer pointer for the 7th of up to 8 buffers returned in this response.
56-57	Reserved
58-63 BUF7	Buffer pointer for the 8th of up to 8 buffers returned in this response.

#### 4.3.3.2.3 Query Response

The query response responds to the query command (see [Section 4.3.3.2, “Response Register \(RR\) Functionality”](#)) and returns a snapshot of the availability and depletion state of each of the BMan’s buffer

pools (see [Section 4.3.5, “Buffer Pool State”](#).) The query response is returned in the response register ([Section 4.2.3.2, “BCSP Response Registers \(BCSPn\\_RRm\)”](#)).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
															BP_DS[0:63]

Figure 4-48. Query Response Format

Table 4-44. Query Response Format

Byte Field	Description
0 VERB	Bit 0 (msb): Alternating valid bit. See <a href="#">Section 4.2.3.2, “BCSP Response Registers (BCSPn_RRm)”</a> . Bit 1-7 (lsbs): Response verb. Specifies the type of command that has been executed. Unused values are reserved. Bit 1-3: Response type 4 = Buffer pool state change query, a request for a depletion and availability status snapshot. Bit 4-7: Reserved
1-39	Reserved
40-47 BP_AS[0:63]	Buffer pool availability state. This 64-bit field contains 1 bit that indicates the availability state of each of the 64 buffer pools in the BMan. Bit 0 (msb): State of buffer pool 0: 0 = free buffers are available, 1 = no free buffers available .... Bit 63 (lsb): State of buffer pool 63: 0 = free buffers are available, 1 = no free buffers available
48-54	Reserved
55-63 BP_DS[0:63]	Buffer pool depletion state. This 64-bit field contains 1 bit that indicates the depletion state of each of the 64 buffer pools in the BMan. The state is derived by comparing buffer pool occupancy with the programmed software depletion threshold for each buffer pool. Bit 0 (msb): State of buffer pool 0: 0 = not depleted, 1 = depleted .... Bit 63 (lsb): State of buffer pool 63: 0 = not depleted, 1 = depleted

#### 4.3.3.2.4 Determining the Expected Polarity of the Valid Bit

If software loses synchronization with the BMan, it needs to determine the expected polarity of the valid bit for the next CR command. Read both RR0 and RR1 and determine if either response is valid by

examining the returned command verb (see [Section 4.3.3.2.2, “Acquire Response,”](#) and [Section 4.3.3.2.3, “Query Response”](#)), then see [Table 4-45](#), which lists the possible explanations of the read results.

**Table 4-45. Determining the Expected Polarity of the Valid Bit**

Result	Explanation
RR1 contains a valid response.	The expected alternating valid bit polarity for the next command is 0.
RR0 contains a valid response.	The expected alternating valid bit polarity for the next command is 1.
Both RR0 and RR1 contain all zeros.	<ul style="list-style-type: none"> <li>• No command has been executed since reset and the expected alternating valid bit polarity is 1.</li> <li>Or</li> <li>• A command is currently in progress.</li> </ul>

### 4.3.3.3 Release Command Ring (RCR) Functionality

The release command ring (RCR) allows software to release buffer pointers to the buffer pools in the BMan. The RCR is an 8-entry, circular FIFO contained in the cache-enabled area of BMan’s software portal (see [Section 4.2.3.3, “BCSP Release Command Ring Registers \(BCSPn\\_RCRm\)”](#)). Each entry is 64 bytes in size and carries a command as described in [Section 4.3.3.3.2, “Release Commands.”](#) The BMan does not modify the data in the RCR entries.

#### 4.3.3.3.1 Issuing a Command Using the RCR

Software produces entries into the RCR by writing commands to the ring entries. Software can issue multiple commands to the ring, but must flow-control its production into the RCR to prevent the ring from overflowing. For the BMan to act on a command, software must notify the BMan of new commands through one of the production notification methods. The BMan will process the RCR command and inform software. The flow control of the ring is achieved using a producer index (PI) controlled by software and a consumer index (CI) controlled by the BMan or by using alternating valid bits.

Software must do the following to issue a command:

1. Determine that there is room in the RCR. (RCR PI-CI fill is less than 8.)
2. Zero and allocate the entry in the processor’s cache without reading it. (Use DC ZVA for example.)
3. Write all words of the RCR entry other than word 0.
4. If using "alternating valid bit PI mode," issue a synchronizing instruction (Use DMB st for example).
5. Write word 0 of the RCR entry.
6. Flush the entry from cache to the BMan. (Use DC CVAC for example)
7. Update the RCR PI, if not using “alternating valid bit mode”.

The RCR PI is initially 0, and is incremented by software after commands are written into the ring. The PI wraps from 7 back to 0 when the end of the ring is reached. As each command is consumed by software the RCR CI is updated by the BMan. The software’s view of the RCR fill level can be determined by taking the RCR PI and subtracting the current RCR CI, modulo 8.

$$\text{FILL} = (\text{PI} - \text{CI}) \% 8$$

*Eqn. 4-6*

There are three RCR PI write modes available to indicate when a command has been produced:

- Cache-Inhibited PI Write Mode
- Cache-Enabled PI Write Mode
- Alternating Valid Bit PI Write Mode

The mode is configured in a control register within each portal (see [Section 4.2.3.7, “BCSP Configuration Registers \(BCSPn\\_CFG\)”](#)) while the RCR is empty. Software may maintain a private RCR PI and only update the BMan’s RCR PI intermittently if using the cache-inhibited or cache-enabled PI write modes. In this case, the software private RCR PI must be used for calculating RCR fill level.

Consumption notification from the BMan to software is tracked using a read-only consumer index value (RCR CI) stored in the BMan. The RCR CI is a 3-bit value that is initially 0 and indicates the ring entry that is next to be consumed by the BMan. The RCR CI value can be read from the following registers in the portal:

- Word-aligned register located in the cache-inhibited region
- 64-byte-aligned register located in the cache-enabled region of the portal.

See [Section 4.2.3.5, “BCSP RCR Consumer Index Registers \(BCSPn\\_RCR\\_CI\\_CENA, BCSPn\\_RCR\\_CI\\_CINH\).”](#)

### *Cache-Inhibited PI Write Mode*

In cache-inhibited PI write mode, the RCR PI is a 3-bit value in a cache-inhibited, read/write register. This register is mapped on a 4-byte word boundary, and software issues a cache-inhibited store to this word to advance the RCR PI. (See [Section 4.2.3.4, “BCSP Release Command Ring \(RCR\) Producer Index Registers \(BCSPn\\_RCR\\_PI\\_CENA,BCSPn\\_RCR\\_PI\\_CINH\).”](#))

Software must issue an appropriate synchronizing instruction between the instruction that flushes the RCR command and the cache-inhibited write that updates the RCR PI. The synchronizing instruction used must ensure ordering between cache-enabled and cache-inhibited stores so that the PI update arrives at the BMan after the complete RCR entry.

To issue a command, software must perform the following steps:

1. Issue an appropriate synchronizing instruction after RCR command flush.
2. Zero and allocate the entry in the processor’s cache without reading it. (Use DC ZVA for example)
3. Write the updated PI.
4. Flush the PI from cache to the BMan. (Use DC CVAC for example)

The valid bit in the RCR entries is ignored by the BMan. If a null command or an undefined command is written to an RCR entry, and the producer index is updated to include this entry, then an invalid command interrupt will be asserted if enabled (see [Section 4.3.4.3, “Functional Interrupt Source, per Software Portal”](#)).

### *Cache-Enabled PI Write Mode*

In cache-enabled PI write mode, the RCR PI is a 3-bit value in a cache-enabled, read/write register. This register is mapped on a 64-byte cache-line boundary, and software issues cache-enabled store followed by the flush instruction to this cache-line to advance the RCR PI.

Software must issue an appropriate synchronizing instruction between the instruction that flushes the RCR command and the cache-enabled write that updates the RCR PI. The synchronizing instruction used must ensure ordering between cache-enabled stores so that the PI update arrives at the BMan after the complete RCR entry.

To issue a command, software must perform the following steps:

1. Issue an appropriate synchronizing instruction after RCR command flush.
2. Zero and allocate the entry in the processor's cache without reading it. (Use DC ZVA for example)
3. Write the updated PI.
4. Flush the entry from cache to the BMan. (Use DC CVAC for example)

The valid bit in the RCR entries is ignored by the BMan. If a null command or an undefined command is written to an RCR entry, and the producer index is updated to include this entry, then an invalid command interrupt will be asserted if enabled (see [Section 4.3.4.3, “Functional Interrupt Source, per Software Portal”](#)).

### *Alternating Valid Bit PI Write Mode*

Alternating valid bit mode removes the need for software to explicitly update RCR PI. An alternating polarity valid bit and the presence of a non-zero command verb are used in each RCR entry to indicate a valid ring entry.

When software writes a command verb into an RCR entry, it also sets the valid bit (the msb of the command verb) to the current valid bit polarity. The valid bit polarity is initially 1. Software must track the current valid bit polarity and toggle the polarity it is using to issue RCR commands when the ring wraps from entry 7 to entry 0.

Note that a command verb (excluding the valid polarity bit) of 0x00 is interpreted as an entry that has not yet been produced. The exact order of the RCR entry writes is required to ensure that a castout of a partially completed command from the processor's cache always writes a 0 (null) command verb.

#### **4.3.3.2 Release Commands**

Each release command ring (RCR) entry comprises a release command verb and buffers for release to the BMan. Each command may release from 1 to 8 buffers. Separate command verbs are available to release buffers belonging to the same pool, and for to release to a variety of pools.

#### **NOTE**

A release to a non-existent pool, or pool 255 (0xFF), results in the buffer pointer being dropped. For forward compatibility, the discard pool must be set as 255 (0xFF), using the full 8 bits of the BPID field.

Buffer pointers have the upper bits beyond 40 bits truncated.

## Buffer Manager (BMan)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VERB	BPID0	BUF0						IOCD	BPID1	BUF1					
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	BPID2	BUF2						BPID3	BUF3						
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
	BPID4	BUF4						BPID5	BUF5						
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	BPID6	BUF6						BPID7	BUF7						

Figure 4-49. Release Command Format

Table 4-46. Release Command Format

Byte Field	Description
0 VERB	Bit 0 (msb): Valid bit. See “ <a href="#">Alternating Valid Bit PI Write Mode</a> .” Bit 1-7 (lsbs): Command Verb. Specifies the type of command to be executed. Unused values are reserved. Bit 1-3: Command Type. 0h = Invalid command. Note that in valid bit mode, a write with all bits of the command verb (bits 1-7) equal to 0 is treated as a castout of a partially completed command (rather than an invalid command), and the ring entry is considered not yet produced (RCR_PI is not advanced). See <a href="#">Section 4.3.3.1.1, “Writing a Command into the CR.”</a> 2h = Release buffers to the pool identified in byte field 1 (BPID0). 3h = Release each buffer to the pool identified in byte field immediately preceding its buffer field. Bit 4-7: Number of buffers associated with command type, maximum 8. 0h = Zero buffers 1h = One buffer ... 8h = Eight buffers
1 BPID0	Buffer pool ID For command verb 21h to 28h, specifies the buffer pool to which all buffers should be released For command verb 31h to 38h, specifies the buffer pool to which BUF0 should be released
2-7 BUFO	Buffer address for the 1st of up to 8 buffers to be released
8 IOCD	Interrupt on command dispatch Bit 0 (msb, left-most bit): Setting this bit in a release command will cause the RCDI interrupt to be asserted (if enabled to do so) when this command is dispatched for execution. Bit 1-7: Reserved
9 BPID1	Buffer pool ID For command verb 32h to 38h, specifies the buffer pool to which BUF1 should be released
10-15 BUF1	Buffer address for the 2nd of up to 8 buffers to be released
16	Reserved
17 BPID2	Buffer pool ID For command verb 33h to 38h, specifies the buffer pool to which BUF2 should be released
18-23 BUF2	Buffer address for the 3rd of up to 8 buffers to be released.

**Table 4-46. Release Command Format (continued)**

<b>Byte Field</b>	<b>Description</b>
24	Reserved
25 BPID3	Buffer pool ID For command verb 34h to 38h, specifies the buffer pool to which BUF3 should be released
26-31 BUF3	Buffer address for the 4th of up to 8 buffers to be released
32	Reserved
33 BPID4	Buffer pool ID For command verb 35h to 38h, specifies the buffer pool to which BUF4 should be released
34-39 BUF4	Buffer address for the 5th of up to 8 buffers to be released
40	Reserved
41 BPID5	Buffer pool ID For command verb 36h to 38h, specifies the buffer pool to which BUF5 should be released
42-47 BUF5	Buffer address for the 6th of up to 8 buffers to be released.
48	Reserved
49 BPID6	Buffer pool ID For command verb 37h to 38h, specifies the buffer pool to which BUF6 should be released
50-55 BUF6	Buffer address for the 7th of up to 8 buffers to be released
56	Reserved
57 BPID7	Buffer pool ID For command verb 38h, specifies the buffer pool to which BUF7 should be released
58-63 BUF7	Buffer address for the 8th of up to 8 buffers to be released

### 4.3.4 Interrupts

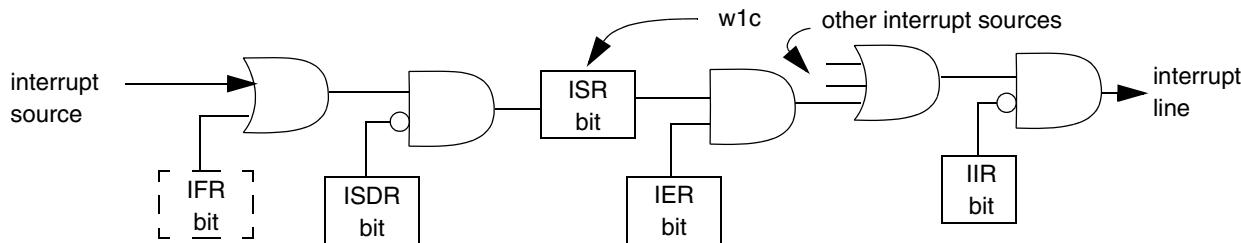
This section summarizes the available interrupt control logic and various interrupt scenarios for both software portal functional interrupts and the shared error interrupt. These interrupts allow software to act on changes in depletion state and various error conditions. When an interrupt is received by software possible responses include the following:

- Releasing more buffers into the BMan buffer pools
- Performing system level flow control
- Performing system level error recovery activities

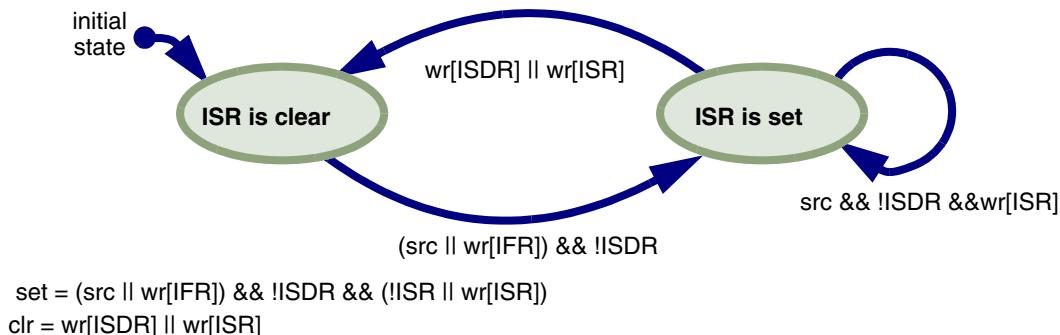
#### 4.3.4.1 Interrupt Control Logic

Both the error and functional interrupt lines are controlled by a similar set of logic. The interrupt status disable register (ISDR) bit masks the interrupt source event from asserting the interrupt status register (ISR) bit. The interrupt enable register (IER) bit controls the assertion of the interrupt line based on the ISR bit. The ISR bit is cleared by writing 1 to it. An interrupt force register (IFR) bit is provided so that software can simulate an interrupt source. A single interrupt inhibit register (IIR) bit is provided per interrupt line to prevent all interrupt sources from asserting the interrupt line. (See [Section 4.2.3.9, “BCSP Functional Interrupt Management Registers \(BCSPn\\_ISR, BCSPn\\_IER, BCSPn\\_ISDR, BCSPn\\_IFR, BCSPn\\_IIR\)”](#) and [Section 4.2.4.11, “Error Interrupt Management Registers \(BMAN\\_ERR\\_ISR, BMAN\\_ERR\\_IER, BMAN\\_ERR\\_ISDR, BMAN\\_ERR\\_IFR, BMAN\\_ERR\\_IIR\).”](#))

All interrupt bits for the interrupt line are OR-ed together to form the single line so that there is a single error interrupt line and a functional interrupt line per software portal.



**Figure 4-50. Interrupt Control Logic Flow**



**Figure 4-51. Buffer Manager (BMan) interrupt state changes**

#### 4.3.4.2 Error Interrupt Sources

Additional registers that capture information related to the error that occurred or some interrupts that occur on the error interrupt line are listed and described in [Table 4-47](#). The interrupts themselves are described

in Section 4.2.4.11, “Error Interrupt Management Registers (BMAN\_ERR\_ISR, BMAN\_ERR\_IER, BMAN\_ERR\_ISDR, BMAN\_ERR\_IFR, BMAN\_ERR\_IIR).”

**Table 4-47. Additional BMan Error Event Registers**

Interrupt	Condition, Further Information	Response
EMAI	External memory error reported a FBPR read error. Address and pool ID are captured in the AECR and AEAR registers. (See <a href="#">Section 4.2.4.16, “Access Error Capture and Address Registers (BMAN_AECR, BMAN_AEAR).”</a> )	<i>None required.</i> FBPRs and buffer pointers will have been lost, leading to early pool depletion. Look for memory misconfiguration.
EMCI	Detected bad data in a FBPR read. Address and pool ID are captured in the CECR and CEAR registers. (See <a href="#">Section 4.2.4.15, “Corruption Error Capture and Address Registers (BMAN_CECR, BMAN_CEAR).”</a> )	<i>None required.</i> FBPRs and buffer pointers will have been lost, leading to early pool depletion. Look for memory misconfiguration or software writing in the BMan’s private FBPR memory space.
IVCI	A software portal wrote an invalid command. Information on the invalid command and the portal are captured in the ECIR register. (See <a href="#">Section 4.2.4.14, “Error Capture Information Register (BMAN_ECIR).”</a> )	<i>None required.</i> Command was not executed, possible loss of a small number of buffers or unexpected response at software portal. Look at software command execution sequence. Is the command sequence being respected? Are commands being written too soon? Is software out of sync with the BMan software portal state? Is the processor cache causing an unexpected write to the software portal?
FLWI	Available FBPRs has dropped below the threshold. (See <a href="#">Section 4.2.4.4, “FBPR Pool Low Watermark Interrupt Threshold Register (FBPR_FP_LWIT)</a> and <a href="#">FBPR_FPC</a> in <a href="#">Section 4.2.4.6, “Pool Content Registers (BMAN_POOLn_CONTENT, FBPR_FPC).”</a> )	Acquire buffers so that FBPRs become available to other pools, or increase the number of FBPRs at initialization. It is possible that this is a normal condition that can occur when few buffers are being used by the system: if the number of FBPRs at initialization is near x8 the total number of buffers being held by the BMan in all pools. This condition may also occur as a result of a loss of buffers due to a FBPR read error. (See <a href="#">Section 4.3.4.5, “Effect of FBPR Depletion on Release Commands.”</a> )
MBEI	A multi-bit ECC error occurred on internal stockpile memory.	See <a href="#">Section 4.3.4.4, “Reaction to Multi-Bit ECC Interrupts (MBEI).”</a>

**Table 4-47. Additional BMan Error Event Registers (continued)**

Interrupt	Condition, Further Information	Response
SBEI	Single bit ECC errors exceeded the threshold on internal stockpile memory. (See Section 4.2.4.12, “Single Bit ECC Error Threshold Register (BMAN_SBET)” and Section 4.2.4.13, “Single Bit ECC Error Count Register (BMAN_SBEC0).”) Captures the number of ECC events occurring on the internal stockpile SRAMs. Note that stockpile locations are only read once before being over-written. Therefore, single-bit ECC errors do not require corrective action.	<i>None required.</i> Stockpile locations are only read once before being over-written so are not affected by the possibility of single bit ECC errors accumulating into a multi-bit ECC error.
BSCN	Indicates that a pool has entered or left the empty pool state. (See Section 4.2.4.6, “Pool Content Registers (BMAN_POOLn_CONTENT, FBPR_FPC)” and Section 4.3.5.1, “Buffer Availability State.”)	The availability state of all pools can be observed by issuing a query to the software portal, see <a href="#">Section 4.3.3.1.3, “Acquire and Query Commands.”</a> If entering the “empty” state, software can rectify the issue by releasing more buffers or by modifying system behavior to reduce the number of buffers in use (that is: policing, flow control, etc). More buffer pointers can also be released into the system via the software portals.

#### 4.3.4.3 Functional Interrupt Source, per Software Portal

Interrupts that occur on the functional interrupt lines (per software portal) are listed and described in [Table 4-47](#). The interrupts themselves are described in [Section 4.2.3.9, “BCSP Functional Interrupt Management Registers \(BCSPn\\_ISR, BCSPn\\_IER, BCSPn\\_ISDR, BCSPn\\_IFR, BCSPn\\_IIR.\)”](#)

**Table 4-48. Additional BMan Error Event Registers**

Interrupt	Condition, Further Information	Response
RCDI	Software portal RCR release command dispatched with interrupt requested. (See IOCD field, <a href="#">Section 4.3.3.3.2, “Release Commands.”</a> )	Software portal RCR is available to accept further buffer releases.
RCRI	Software portal RCR fill dropped below a threshold. (See <a href="#">Section 4.2.3.6, “BCSP RCR Interrupt Threshold Register (BCSPi_RCR_ITR).”</a> )	Software portal RCR is available to accept further buffer releases.
BSCN	Indicates that a pool has entered or left the depleted state. (See <a href="#">Section 4.2.4.6, “Pool Content Registers (BMAN_POOLn_CONTENT, FBPR_FPC),</a> <a href="#">Section 4.2.4.3, “Hardware and Software Portal Depletion Threshold Registers (BMAN_POOLn_SWDET,</a> <a href="#">BMAN_POOLn_SWDXT),</a> <a href="#">Section 4.2.3.8, “BCSP Depletion State Change Interrupt Enable Registers (BCSPn_SCNm)”</a> and <a href="#">Section 4.3.5.2, “Buffer Depletion State.”</a> )	The depletion state of all pools can be observed by issuing a query to the software portal, see <a href="#">Section 4.3.3.1.3, “Acquire and Query Commands.”</a> If entering the “depleted” state, software can rectify the issue by releasing more buffers or by modifying system behavior to reduce the number of buffers in use (that is: policing, flow control, etc). More buffer pointers can also be released into the system via the software portals.

#### 4.3.4.4 Reaction to Multi-Bit ECC Interrupts (MBEI)

A system-wide reset is required to correct multi-bit ECC interrupts. There is no way to identify whether the multi-bit ECC event occurred on an FBPR flush, a software portal or a DCP acquire.

Multi-bit ECC errors can occur on reads from the internal BMan stockpile in the following ways:

- For a DCP acquire with a multi-bit ECC error, the corrupted pointer is returned as “invalid” so it does not corrupt the pool, nor does it “escape” beyond the BMan.
- For a software portal acquire with a multi-bit ECC error, the corrupted pointer is written to the software portal acquire ring with an indication that the response does not contain valid buffer pointers so it does not corrupt the pool, nor does it “escape” beyond the BMan.
- For an FBPR flush with a multi-bit ECC error, the pool count is updated to indicate the loss of eight pointers.

#### 4.3.4.5 Effect of FBPR Depletion on Release Commands

Releases are never rejected for both software portal (RCR) and hardware (DCP) release commands.

If sufficient FBPRs are unavailable to complete a release, then the BMan stops consuming entries from the RCR or DCP until space becomes available. A global interrupt for the FBPR depleted condition (FLWI) is described in [Section 4.2.4.11, “Error Interrupt Management Registers \(BMAN\\_ERR\\_ISR, BMAN\\_ERR\\_IER, BMAN\\_ERR\\_ISDR, BMAN\\_ERR\\_IFR, BMAN\\_ERR\\_IIR\).”](#)

### 4.3.5 Buffer Pool State

The BMan is able to notify software and hardware when a buffer pool’s occupancy exceeds or falls below a threshold; it can also indicate whether a buffer pool is empty or not.

There are two buffer pool states, as follows:

- Buffer availability state—A buffer pool’s state is set to “no buffers available” when the pool is completely empty of buffers ([Section 4.3.5.1, “Buffer Availability State”](#)).
- Depletion state—When the number of buffers in a pool falls below a programmable entry threshold, the pool is “depleted.” When the buffer pool’s occupancy rises above a programmable exit threshold, the buffer pool exits depletion (see [Section 4.3.5.2, “Buffer Depletion State”](#)).

To determine if a buffer pool’s state has changed, software can maintain a copy of the last known state of buffer pools it is managing, and compare the last known state with the new buffer pool state received after a query command. (See [Section 4.3.3.1.3, “Acquire and Query Commands.”](#))

#### 4.3.5.1 Buffer Availability State

Buffer availability is the condition of having any buffers at all in a buffer pool. A buffer availability state change notification (BASCN) occurs when a pool enters or leaves the empty state.

For the DCPS, buffer pool state (rather than state changes) are provided using a dedicated set of interface signals. Using these signals hardware clients can automatically flow-control or provide other actions when resources become unavailable.

For software, availability state changes are communicated by asserting the single error interrupt line. (See the BSCN bit in [Section 4.2.4.11, “Error Interrupt Management Registers \(BMAN\\_ERR\\_ISR, BMAN\\_ERR\\_IER, BMAN\\_ERR\\_ISDR, BMAN\\_ERR\\_IFR, BMAN\\_ERR\\_IIR\).”](#)) The availability state change interrupts are not managed in the software portal.

#### **4.3.5.2 Buffer Depletion State**

Buffer depletion is the condition of the buffer count for a pool having fallen below some programmed threshold. A buffer pool leaves the depleted state when the buffer count for a pool rises above a separate programmed threshold. Having an entry and exit threshold allows for hysteresis in the depletion state. A buffer depletion state change notification (BDSCN) occurs when a pool enters or leaves the depleted state. There are two pairs of thresholds for each pool; one for software depletion state change notifications via the software portal functional interrupt (if enabled to do so) and one for hardware modules through the direct connect portals (DCPs).

For the DCPs, depletion state (rather than state changes) are provided using a dedicated set of interface signals. Using these signals hardware clients can automatically flow-control or provide other actions when resources are becoming depleted.

For software, depletion state changes are communicated by asserting a functional interrupt for each software portal, provided that one or more pools that change depletion state are enabled for a depletion state change notifications as described in [Section 4.2.3.8, “BCSP Depletion State Change Interrupt Enable Registers \(BCSPn\\_SCNm\)”](#) and the BSCN bit in [Section 4.2.3.9, “BCSP Functional Interrupt Management Registers \(BCSPn\\_ISR, BCSPn\\_IER, BCSPn\\_ISDR, BCSPn\\_IFR, BCSPn\\_IIR\).”](#)

The number of depletion state changes is counted for both hardware and software thresholds. (See [Section 4.2.4.5, “Hardware and Software Portal Depletion Count Register \(BMAN\\_POOLn\\_SDCNT, BMAN\\_POOLn\\_HDCNT\).”](#))

#### **4.3.5.3 Changing Depletion Thresholds**

##### **NOTE**

In general, it is not necessary to change the depletion thresholds while the BMan is active.

If changing depletion thresholds (see [Section 4.2.4.3, “Hardware and Software Portal Depletion Threshold Registers \(BMAN\\_POOLn\\_SWDET, BMAN\\_POOLn\\_SWDXT\)”](#)) after buffers have been released to the BMan, the entry and exit level must be modified so that they do not change the depletion state of the pool until the modification to the final register. This is only possible if the module is idle (that is, the buffer pool contents are not changing).

First software must query the current depletion state through the software portal. (See [Section 4.3.3.1.3, “Acquire and Query Commands.”](#))

If not depleted, set the following:

1. Entry threshold = 0 (disabled)
2. Exit threshold to new value
3. Entry threshold to new value

If depleted, set the following:

1. Exit threshold = MAX
2. Entry threshold to new value
3. Exit threshold to new value

If interrupts are the only concern (as opposed to toggling the hardware interface depletion state on the DCP interface), an additional option exists, as follows:

1. Disable the depletion state change functional interrupt for all software portals. (See BSCN bit in [Section 4.2.3.9, “BCSP Functional Interrupt Management Registers \(BCSPn\\_ISR, BCSPn\\_IER, BCSPn\\_ISDR, BCSPn\\_IFR, BCSPn\\_IIR\).”](#))
2. Adjust the software thresholds.
3. Clear the ISR bit after modifying the thresholds.

The last step can be avoided if the interrupt is disabled using the ISDR register instead of the IER register.

### 4.3.6 Free Buffer Proxy Records (FBPRs)

#### NOTE

Software should never need to access FBPRs.

A free buffer proxy record (FBPR) is a 64-byte data structure internal to the BMan that stores buffer pointers in the BMan’s external memory space in a singly-linked list. The BMan holds the head of this singly-linked list for each pool and the FBPR free-list in the HDPTR registers. (See [Section 4.2.4.7, “Free List Head Pointer Registers \(BMAN\\_POOLn\\_HDPTR, FBPR\\_HDPTR\).”](#))

An empty list is indicated when the head of the list is zero. This can be due to the following:

- No buffers have been released to a pool.
- Only a few buffers have been released which are all in the stockpile.
- Buffers have been released and then acquired such that all buffers are in the stockpile.

#### 4.3.6.1 FBPR Format

An individual FBPR only contains buffer pointers for a single pool and a singly-linked list of FBPRs is always associated with the same pool.

Contains the following fields (all right justified with upper bits zero):

- Field BPR\_BPA contains the FBPR index where this FBPR is stored.
- Field BP0-7 are the buffer pointers stored in this FBPR.
- Field BPID is the buffer pool to which these buffer pointers belong.
- Field FBPR\_IDX is the FBPR index to the next FBPR in the singly-linked list. All-zero indicates the end of the list.

The same format is used for storing FBPRs in the FBPR free-list and buffer pointers in pools. The buffer pointers are stored right-justified, such that the upper bits are zero, when they are less than 48 bits (BMan stores 40 bit buffer pointers and 28 bit FBPR pointers.)

## Buffer Manager (BMan)

FBPR\_BPA and BPID are intended as a crude debug check to test for memory scribbling by other entities or FBPR pool corruption but do not provide any guarantees that this behavior will be caught. When an FBPR belongs to the FBPR pool, the buffer pointer ID (BPID) is set to 0xff, and FBPR\_BPA matches BP0. An EMCI error interrupt is raised and the linked list of the pool is dropped if, the FBPR\_BPA and BPID on an FBPR read do not match their expected values. If BPID = 0xff (FBPR pool), then BP0 is also checked to confirm that it matches FBPR\_BPA.

The FBPR format holding buffer pointers is shown in this figure.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BP0				BP1				FBPR_BPA							
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BP2				BP3											
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
BP4				BP5											
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
BP6				BP7				FBPR_IDX							

Figure 4-52. Free Buffer Proxy Record (FBPR) Using Buffer Pointers (BPs)

### 4.3.7 Performance Monitor

The BMan implements 32 performance monitor output signals.

The BMan can report one of the following types of events:

- An acquire or release request was serviced (a “command” event), or
- An FBPR data structures was fetched or flushed (an external memory access).

#### 4.3.7.1 Command Performance Monitoring

The command performance monitor can indicate on the performance monitor output when an acquire or release matching the following criteria has been serviced:

- Matches a contiguous range of pools
- Requests a minimum number of buffers (acquire only)
- Has waited longer than a threshold time before being serviced, measured in BMan clock cycles, from the time a request is stamped at the BMan interface to the time it receives service
- The request came from a selected set of interfaces (DCP and software portal)
- The request FIFO was full when serviced

A DCP acquire or release request is timestamped when it enters the BMan.

For the software portal, all portals share a common interface and are treated as a single source by the performance monitor logic. Software portal timestamps do not account for the time a request has waited in the RCR or CR before being transferred to the BMan’s internal FIFOs.

See [Section 4.2.4.8, “Command Performance Monitor Configuration Registers \(CMD\\_PMn\\_CFG\),”](#) to configure the performance monitor registers

## NOTE

The timestamp used by the BMan is a 12-bit counter incremented at the BMan’s clock. If a command takes more than 4096 cycles ( $2^{12}$  cycles, 12.8 $\mu$ s at 320 MHz, 10.2 $\mu$ s at 400 MHz) before being serviced, roll-over of the timer mechanism for that event occurs and the service delay as compared in the performance monitor requirement is invalid. This results in bounded, undefined performance monitor output behavior for that particular transaction. However, these events are expected to be infrequent.

### 4.3.7.2 Free List Fetch/Flush Performance Monitoring

The BMan has eight monitors that can monitor the FBPR free list (fetch/flush only) *OR* contiguous range of pools. Each fetch or flush is compared to the configuration of the eight performance monitors and, if all conditions are met, generates a pulse on the appropriate performance monitor pin.

See [Section 4.2.4.9, “Free List Performance Monitor Configuration Register \(BMAN\\_FL\\_PMn\\_CFG\),”](#) to configure the performance monitor registers.

## 4.4 BMan Initialization Tips

Some general tips for better BMan initialization are as follows:

- The BMan must be configured with a valid external memory size window using FBPR\_AR[SIZE], where upon it immediately begins initializing its list of free data structures (FBPRs) and writing them out to external memory.
- The following registers must be configured first and should not be changed after a value is written to FBPR\_AR[SIZE], which causes initialization to occur (See [Section 4.2.4.2, “Memory Configuration Registers \(FBPR\\_BAR, FBPR\\_BARE, FBPR\\_AR, BMAN\\_ICIDR, BMAN\\_SRCID\).”](#)):
  - The base address and extended base address registers, FBPR\_BAR, FBPR\_BARE



# Chapter 5

## Frame Manager (FMan)

### 5.1 Frame Manager Overview

The Frame Manager (FMan) is a functional unit that combines the Ethernet network interfaces with packet distribution logic to provide intelligent distribution and queuing decisions for incoming traffic at line rate. This integration allows the FMan to perform configurable parsing and classification of the incoming frame with the purpose of selecting the appropriate input frame queue for expedited processing by a CPU or pool of CPU cores.

#### **NOTE: SoC-Specific Implementation**

Because FMan implementation can vary between DPAA-enabled SoCs, see the applicable SoC reference manual or [Chapter 1, “Data Path Acceleration Architecture \(DPAA\) Overview”](#) for other device-specific information related to the FMan.

#### **NOTE: Differences between devices**

The FMan is used in many devices. This reference manual documents the FMan\_v3 the version of the FMan found on the LS1043A. Most of the features are identical among the devices, but there are slight differences between the various FMans. The features apply to the various devices as follows:

**Table 5-1. Mapping features to devices**

Name in this reference guide	List of devices to which the feature applies
FMan_v3	LS1043A

#### **NOTE: Interchangeable Terms**

The words ‘frame’ and ‘packet’ are used interchangeably throughout this reference manual. The term ‘FMan internal memory’ is sometimes referred to as ‘FMan memory.’

### 5.1.1 FMan Terms, Acronyms, and Abbreviations

This table describes common FMan terms, acronyms, and abbreviations.

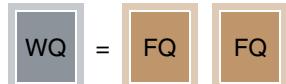
**Table 5-2. FMan Terms, Acronyms, and Abbreviations**

Term	Definition	Graphical Representation
AD	Action descriptor	—
BMan	Buffer Manager	—
BMI	Buffer Manager interface	—

**Table 5-2. FMan Terms, Acronyms, and Abbreviations (continued)**

Term	Definition	Graphical Representation
Buffer	Region of contiguous memory, allocated by software, and managed by the DPAA BMan	
BD	Buffer descriptor	—
Buffer pool	Set of buffers with common characteristics (mainly size, alignment, access control)	
Channel	Set of eight WQs with hardware provided prioritized access. <b>Note:</b> The terms ‘work queue channel’ and ‘WQ channel’ are sometimes used instead of ‘channel’.	
CAPWAP	Control and provisioning of wireless access points	—
Dedicated channel	A channel statically assigned to a particular end point, from which that end point can dequeue frames. End point may be a CPU, FMan, or SEC.	—
DoS	Denial of Service	—
DPAA	Data Path Acceleration Architecture	—
DSCP	Differentiated services code point	—
DTLS	Datagram transport layer security	—
FD	Frame descriptor	—
FMan	Frame Manager	—
Frame	Single buffer or list of buffers that hold data (for example, packet payload, header, and other control information)	
Frame queue (FQ)	FIFO of frames	
FQD	Frame queue descriptor	—
FQID	Frame queue ID	—
IC	Internal context	—
ICAD	Internal context action descriptor	—
MAC	Media access control	—
NIA	Next invoked action	—

**Table 5-2. FMan Terms, Acronyms, and Abbreviations (continued)**

Term	Definition	Graphical Representation
PCD	Parse, classify, and distribute	—
Pool channel	A channel statically assigned to a group of end points, from which any of the end points may dequeue frames.	—
O/H	Offline port/Host command	—
QMan	Queue Manager	—
QoS	Quality of service	—
Rx	Receive	—
SEC	Security Engine	—
SNAP	Subnetwork access protocol (detailed in IETF RFC 1042)	—
SoC	System on a chip	—
Task	Handles one frame	—
TNUM	The ID of a task	—
UD	User-defined	—
UDP	User datagram protocol	—
Work queue (WQ)	FIFO of FQs	 A diagram illustrating the relationship between Work Queue (WQ) and Frame Queues (FQ). It shows a large gray rectangle labeled 'WQ' followed by an equals sign (=). To the right of the equals sign are two smaller brown rectangles, each labeled 'FQ', representing a stack of two Frame Queues.

## 5.1.2 FMan Features Summary

The FMan includes the following features:

- Supports Ethernet network interfaces (see the SerDes chapter of the *QorIQ LS1043A Reference Manual* for available configurations)
  - 5 x 1-Gbps Ethernet interfaces
  - 1 x 1/2.5-Gbps Ethernet interfaces
  - 1 x 1/2.5/10-Gbps Ethernet interface
- MDIO for each Ethernet controller
- MDIO for PHY management:
  - Some of the Ethernet controller MDIO are connected to the SoC interface for PHY management
  - In FMan\_v3, Dedicated MDIO control for PHY management
- Interfaces with the Queue Manager (QMan)
  - Dequeues/Enqueues frame descriptors (FD) from/to the QMan
  - Supports drop on tail-drop/wred per the QMan response
  - Supports transmit pause frame due to queue congestion state
  - Supports priority based flow control message pass from Ethernet MAC to QMan.

## Frame Manager (FMan)

- Interfaces with the Buffer Manager (BMan)
  - Allocates/Deallocates frames buffers
  - Supports transmit pause-frame due to buffer pools depletions
- Packet parsing at wire speed
  - Parsing of standard headers using hardware
  - Parsing of non-standard headers using soft examination sequences
- Classification
  - Custom classifier using internal tables
- Policing
  - Policing functionality based on classification result
  - RFC4115 and RFC2968
  - See the applicable device reference manual for each SoC for exact number of policer profiles
- Distribution
  - Distributes to frame queues (FQs) according to custom classifier
  - Statistical distribution to FQs based on extracted key
- Storage Profile selection
  - Storage profile selection per physical port
  - Storage profile selection after distribution function evaluation or after custom classifier.
- Offline port
  - Supports Parse classify distribute (PCD) function on frames extracted frame descriptor (FD) from the QMan
  - Supports frame copy or move from a storage profile to an other.
- FMan internal memory (see the applicable device reference manual for the size)
  - Packet buffering (Tx/Rx FIFOs)
  - Custom classifier table
  - Frames internal context
- Statistics
- Port virtualization
  - Separate programming model for each port in a different 4 KB region
  - Functional separation of resources for each port by management programming
  - Virtual Storage profile selection after classification or distribution function evaluation.
  - Different isolation context identifier (ICID) (in FMan\_v3) per virtual storage profile
- Supports the following host commands through the QMan:
  - Debug
  - Statistics read/write
  - Configuration
- Error reporting through the following:

- The QMan using the FD
- Interrupts
- Independent Ethernet mode
  - Does not use QMan and BMan for enqueueing and buffer allocation (i.e. ‘independent’)
  - Use of Ethernet native interface
  - Buffer descriptor (BD) ring programming model
  - Up to 100-Mbps rate per port, if more than one port is running (in independent or normal mode)
  - Up to 1 Gbps, provided that no other ports are active, either Ethernet or offline (applicable for boot loader), and that the FMan frequency is set to 500 MHz or higher
- Configurable pipeline architecture
  - Allows for the configuration of packet flow through the FMan on a per port basis
  - Classification results can modify the packet flow
  - Operational mode bits, on a per queue basis, can modify the flow

## 5.2 FMan aggregate rate

The following table describes the FMan aggregate rate. The aggregate is the total rate supported by the FMan. The total rate of the Ethernet ports (full duplex) and the O/H ports must be considered for the aggregate rate. The O/H ports, if enabled, consume part of the aggregate rate, therefore in some systems the usage of O/H ports can be at the expense of the Ethernet ports. Note that the FMan performance is linear with the frequency (subject to min/max constraints).

**Table 5-3. FMan aggregate rate**

Device	FMan frequency MHz	FMan aggregate rate Mpps @packet size = 64 bytes	FMan aggregate rate Gbps @ packet size = 1518 bytes
LS1043A	500	6.02	13.06

## 5.3 Frame Manager High-Level Functional Description

The following sections describe the FMan architectural principles at a high-level. Important aspects of the FMan are addressed as an introductory information.

### 5.3.1 FMan Hardware Ports Types

The Frame Manager (FMan) supports several types of hardware ports, listed in this table.

**Table 5-4. Supported FMan Hardware Ports**

Port Type	Speed	Function(s)
Ethernet receive (Rx)	1 Gbps or 2.5Gbps or 10 Gbps	Network interface for the Ethernet receiver that receives frames from the Ethernet interface, applies a parse, classify, distribute (PCD) flow, and enqueues them in a QMan queue.
Ethernet transmitter (Tx)	1 Gbps or 2.5Gbps or 10 Gbps	Network interface Ethernet transmitter that dequeues frames from a QMan queue and transmits them to the Ethernet interface.
Offline (O/H)	FMan_v3: 3.75Mpps	Able to dequeue and enqueue from/to a QMan queue. The FMan applies a Parse Classify Distribute (PCD) flow and (if configured to do so) enqueues the frame back in a QMan queue. In FMan_v3, the FMan is able to copy the frame into new buffers and enqueue back to the QMan. <b>Note:</b> The registers for Offline and Host commands are named O/H port registers.
Host command	—	Able to dequeue host commands from a QMan queue. The FMan executes the host command (such as a table update) and enqueues a response to the QMan. The Host commands, require a dedicated PortID (one of the O/H ports). The registers for Offline and Host commands are named O/H port registers.

See the applicable SoC reference manual for SoC-specific information related to actual hardware ports and rates supported in a specific SoC. For more details on hardware ports, refer to [Section 5.4.1, “FMan Hardware Ports.”](#)

### 5.3.2 FMan Network Interfaces

The FMan\_v3 integrates the following Ethernet controllers:

- 100Mbps/1Gbps/2.5Gbps/10Gbps multirate Ethernet MAC (mEMAC) controller

See the applicable device reference manual SerDes chapter for the exact number of interfaces and available configurations.

The Ethernet controllers support the following:

- Programmable CRC generation and checking
- RMON statistics
- Jumbo frames of up to 9600 bytes
- Designed to comply with IEEE Std 802.3, IEEE 802.3u, IEEE 802.3x, IEEE 802.3z, IEEE 802.3ac, IEEE 802.3ab, and IEEE-1588 v2 (clock synchronization over Ethernet). In FMan\_v3: IEEE 803.3az and IEEE 802.1Qbb.

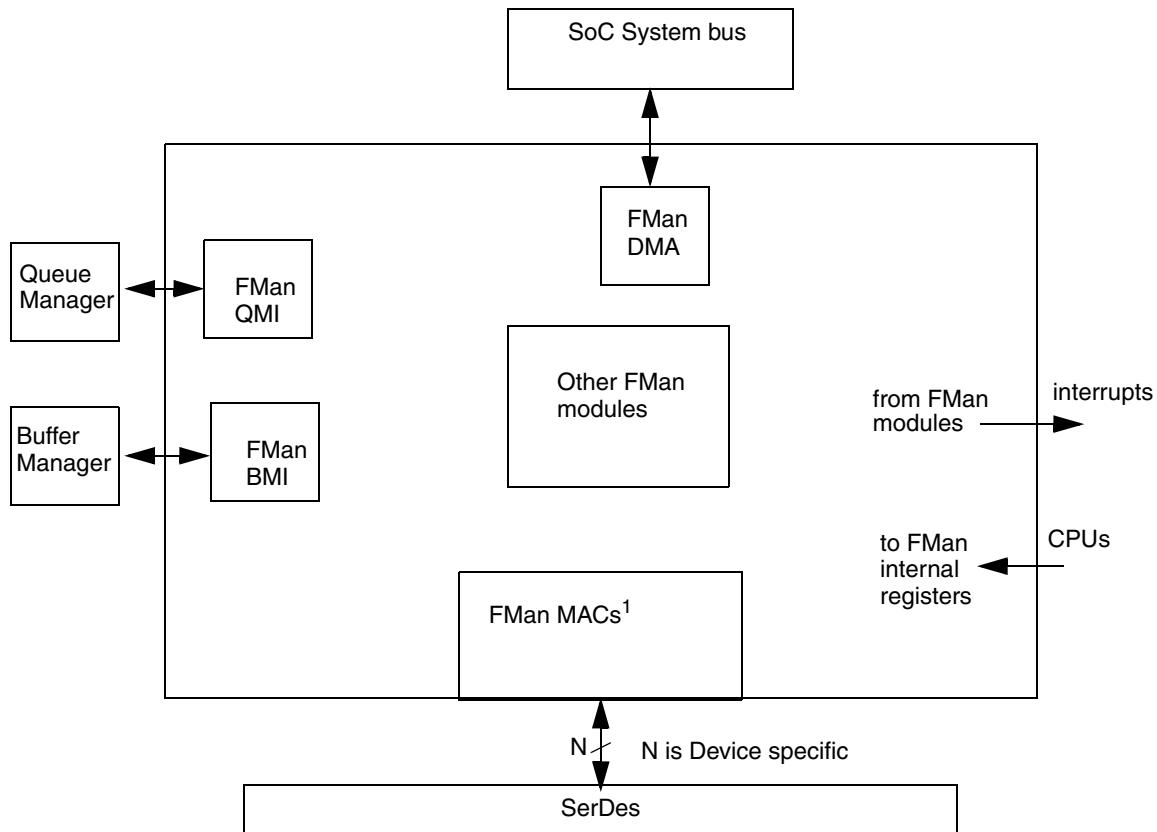
### 5.3.3 FMan-to-SoC Interfaces

The FMan interfaces to SoC modules in the following ways:

- QMan via the Queue Manager interface (QMI)

- BMan via the Buffer Manager interface (BMI)
- SoC system bus via the DMA.

In addition, the FMan asserts interrupts to the generic interrupt controller (GIC), and is connected to the SoC memory space for internal register programming. The FMan Ethernet interfaces are connected to the Serdes and/or the 10G interfaces. For details on these interfaces, see the applicable device reference manual.



**Figure 5-1. FMan Interfaces Block Diagram**

### 5.3.4 FMan Module Architecture

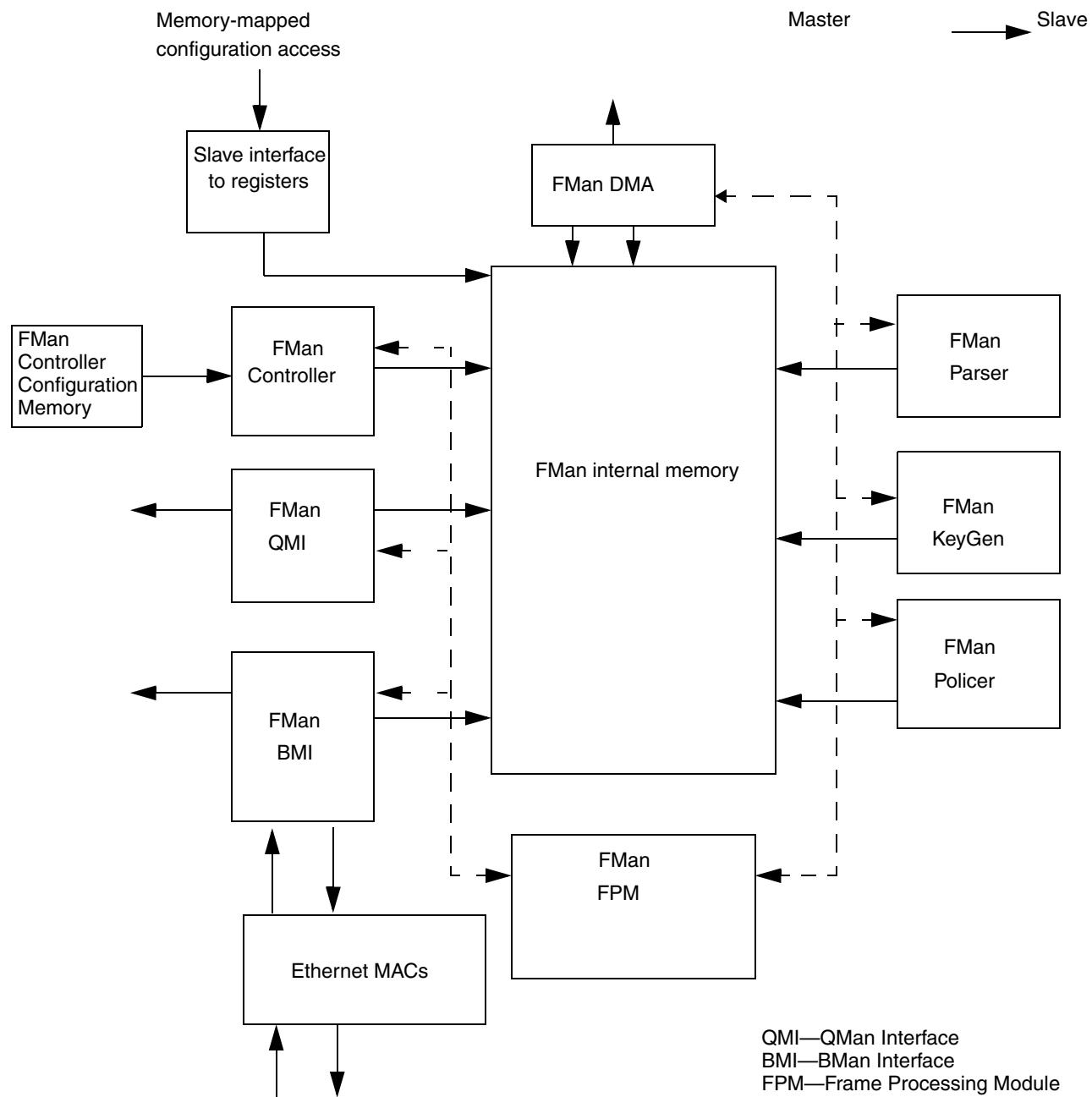
The FMan is composed of modules, each of which accelerates different functions in the parse, classify, distribute (PCD) flow. The FMan internal memory, which is shared between the FMan modules, holds a common data structure per-packet called the internal context (IC). The IC is read and can be modified by the different modules. It also holds the FIFOs needed for the various ports. See [Section 5.4.3, “FMan Frame Internal Context \(IC\),”](#) for more information.

There is no fixed processing order between the modules. The processing order is PCD flow-dependent based on user configurations. Each step is dependent on previous state results that are stored in the IC. This structure enables flexibility, which efficiently supports many flows.

1. See SoC-specific information regarding the number of MACs in [Chapter 1, “Data Path Acceleration Architecture \(DPAA Overview.”](#)

## Frame Manager (FMan)

This figure shows the high-level FMan block diagram.



**Figure 5-2. FMan Block Diagram**

The modules shown in Figure 5-2 are described as follows:

- FMan DMA—moves data between FMan internal memory and external memory (see [Section 5.8, “Frame Manager—DMA”](#))
- FMan Internal memory—shared between all the FMan modules. It contains data structures that are common and written to or read by the modules. The FMan internal memory is split into the following parts:

- Transmit, receive and O/H FIFOs, which consist of linked lists of 256-byte buffers (the FIFO includes the IC per frame)
- Custom classifier data structures (see [Section 5.3.13, “FMan Internal Memory,”](#) for more information)
- Queue Manager interface (QMI)—interface to the QMan
  - FMan module that provides an interface to the QMan for enqueueing and dequeuing new frames to/from the multicore system.
  - Responsible for transferring packet-based work assignments between the QMan and the FMan.
  - Enqueues and de-queues frame descriptors (FDs)
    - Enqueue of classified FDs (Rx)
    - Dequeue of FDs for Tx, Host command and offline port
    - Enqueue of FDs for offline/host commands and for Tx confirmation (optional) (see section [Section 5.6, “Frame Manager—Queue Manager Interface \(QMI\)”](#))
- Buffer Manager interface (BMI)—interface to the BMan
  - Manages internal and external buffer allocate/deallocate and transfers MACs data between the external memory and the native interfaces (MACs) through the FMan internal memory (see [Section 5.5, “Frame Manager—Buffer Manager Interface \(BMI\)”](#)).
- Frame processing manager (FPM)—manages the frame processing by transferring the processing tasks (TNUMs) from one block to another (see [See Section 5.7, “Frame Manager—Frame Processing Manager”](#))
  - Qualifies task execution by their open DMA transaction
  - Provides task ordering mechanism
  - Provides task synchronization mechanism
- Parser—parses incoming frames and generates parser results that contain header types and indexes to the headers (see [Section 5.9, “Frame Manager—Parser”](#))
- KeyGen—key generator that distributes frames to a range of FQs (see [Section 5.10, “Frame Manager—Key Generator”](#))
  - Applies a deterministic hash function on an extracted key to calculate the FQID and the policer profile
  - Ensures that the same network flow always follows the same path
- FMan controller (see [Section 5.12, “Frame Manager—FMan Controller”](#))
  - Custom classifier by searching through a tree of exact match classification tables.
  - Ethernet independent mode (IM)
  - Host commands.
    - Sync—wait until all old processes are finished.
    - Indirect atomic setup writes and reads.
- Policer—colors the frame (“green,” “yellow,” “red”) (see [Section 5.11, “Frame Manager—Policer”](#))
  - The frame is associated with a profile

- The profile parameters and the frame length are used to determine the frame color
- Internal timing mechanism
- Supports RFC 4115 or RFC 2698
- Ethernet MACs—
  - FMan\_v3: [Chapter 6, “Multirate Ethernet Media Access Controller \(mEMAC\),”](#) describes the 100Mbps/1Gbps/2.5Gbps/10Gbps Ethernet MAC used for all the Ethernet interfaces in FMan\_v3. The rate for a specific port is configured by the user (Note: only some ports can be configured to 10Gbps rate). This chapter also describes the MDIO interface.

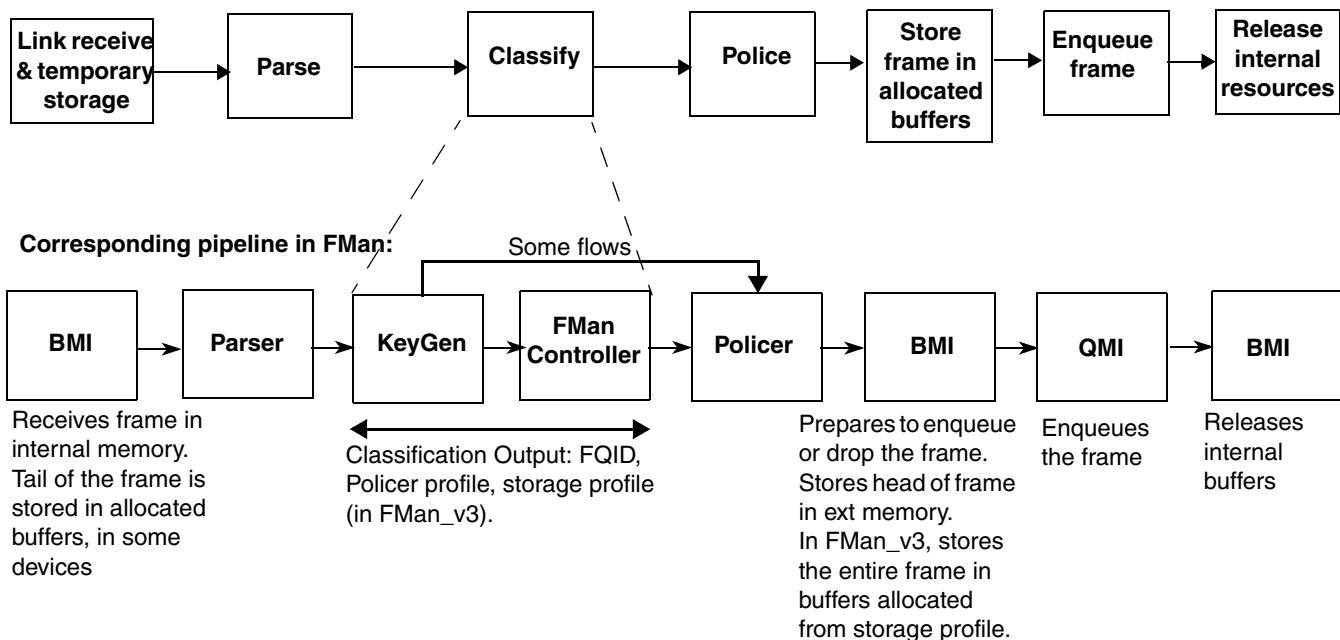
## 5.3.5 FMan User-Configurable Pipeline Architecture—Introducing the NIA

This section introduces the concept of the next invoked action (NIA).

### 5.3.5.1 Packet Flow in the FMan Configurable Pipeline Architecture

When a packet is received from the network, it is processed by the FMan modules in several stages. This figure shows an example of how the FMan modules might act as stages in a specific order to achieve a goal. In each stage a function is performed on the packet. These stages form a pipeline that a given packet traverses in series. The stages of the pipeline operate in parallel, allowing the processing of multiple packets at the same time. In a given application, as a result of classification, different packets may traverse a different stages in the pipeline. In addition, the FMan allows the user to configure the FMan pipeline stages, thus allowing each application to perform different functions on the packets. This is called a “configurable pipeline architecture.”

In a similar fashion, the user may configure the pipeline of all its hardware ports (Rx, Tx, O/H).

**Example of Functional pipeline for Rx:****Figure 5-3. FMan User-Configurable Pipeline Example**

The purpose of this flow example is to describe the function of the NIA. For additional examples of flows see [Section 5.4.4.1, “NIA Register Configuration—Rx Flows.”](#)

### 5.3.5.2 Role of the NIA in the FMan Configurable Pipeline Architecture

At the end of its operation, each module in the FMan “dispatches” a user-programmable “next invoked action” (NIA) code, which is used by the FMan hardware (in the FPM) to determine the next processing module (in other words, the next stage in the pipeline). This is the mechanism that allows the user to configure the pipeline.

In some cases, the NIA also determines a specific action taken by the module. For a list of supported NIAs, see [Section 5.4.4, “Next Invoked Action \(NIA\).”](#)

#### NOTE

The following sections use the terms ‘receive (Rx) flow’ and ‘transmit (Tx) flow’ to describe the path (that is, the pipeline stages) that frames traverse through the FMan.

### 5.3.5.3 Operational mode bits

There are a number of user-configurable “operational mode bits” that affect the operation of the BMI. Some of these bits (for example, NL or NENQ) affect the pipeline flow, while others affect the access to buffers (for example, EBD or EBAD). These bits are initialized when a new frame is processed and may be altered during the processing of the frame by various modules.

- On Rx ports, the operational mode bits are initialized by the BMI, and may be altered during the classification process (KeyGen and/or custom classifier).
- On Tx ports, the operational mode bits are initialized by the BMI, and may be altered during the dequeue of frame by value programmed in the frame queue descriptor (context A).
- On Offline ports, the operational mode bits are initialized by the BMI, and may be altered during the dequeue of frame by value programmed in the frame queue descriptor (context A). In addition, these mode bits may be further altered during the classification process (KeyGen and/or custom classifier).

### **5.3.6 FMan Multitasking—Introducing the Task and TNUM**

The FMan is a multitasking machine. Each frame is handled by one task, and each task handles one frame. A task's IDs is called a TNUM. The TNUMs are dynamically allocated by the FMan hardware to the frames when the processing starts, and are automatically returned to a “free pool” of available TNUMs when the processing of a frame is terminated. The FPM module handles the TNUM allocation. The FMan is able to handle multiple tasks (that is, multiple frames) concurrently. Each FMan variation supports a different number of tasks (TNUMs).

### **5.3.7 Virtual Storage Profiles**

The virtual storage profile mechanism allows the virtualization of the buffer pool selection for frame storage (and other parameters related to storage in external memory) from the physical hardware ports. The virtual storage profile ID (SPID) is selected as a result of the classification on the frame headers. The same Storage Profile ID (SPID) values from the classification on different physical ports, may yield to different storage profile selection. Up to 64 storage profiles per port are supported.

### **5.3.8 ICID**

The isolation context identifier (ICID) maps an incoming transaction from an IO device to one of the contexts and maps to StreamID as described in ARM documentation. By using different ICIDs, a single hardware module can perform memory accesses on behalf of different requestors with the mapping and access controls appropriate to that requestor.

### **5.3.9 Rx frame replicator**

It is possible to configure the FMan to duplicate incoming packets and place them in separate queues. Each replication has its own virtual storage profile (i.e. allocation of buffers from different buffer pools) and its own Frame Descriptor (FD). This mode of operation is supported with the FMan controller and appropriate configuration of operational mode bits. See [Section 5.3.19.1.2, “Receive \(Rx\) Flow Example with Custom Classifier”](#) (next to “BMI Release Internal resources“ stage) for more details.

### 5.3.10 FMan Hardware Ports, QMan sub-portals

One of the QMan direct connect portals (DCPn) is used to connect the QMan to FMan (for devices with more than one FMan, each FMan is connected to a separate direct connect portal).

A DCP contains one or more sub portals (SP) to which FMan dequeue commands can be issued.

Each QMan SP is connected to a QMan ‘channel’ (see [Section 3.3.2, “Work Queues \(WQs\) and Channels”](#)).

Each SP in the QMan DCP (and thus each channel) is associated with a separate FMan Tx or O/H hardware port. The mapping of SPs to FMan hardware ports is configured in FMan QMI FMQM\_PnDC[SP] field.

All QMan FQs associated with FMan, are configured (in the FQD) to be enqueued by the QMan onto one of these work queues for subsequent dequeue by the FMan. This is done by programming FQD[DEST\_WQ] with the WQ ID corresponding to the SP which is associated to the desired FMan hardware port (Tx or O/H). See [Section 5.6.4.3, “FMan Hardware Ports, QMan sub-portals - details”](#) for a more detailed description of these concepts.

### 5.3.11 FMan Timestamp

A high precision time measurement is provided by the FPM as a global utility to FMan modules which need a timestamp. The configurable pre-scale logic in the FPM, enables high precision count rates by fixed-point accumulation in addition to the 32-bit integer counter.

A number of FMan modules use the global time measurement. For example: The BMI uses this value for rate limiting the Offline port dequeue rate; the BMI also uses this value to rate limit the transmission rate; the Policer uses the timestamp for rate calculations for token bucket updates. The IP Reassembly function, uses this value for reassembly timeouts, and so on.

Each module using the time measurement, may be configured to scale down the accuracy of the 32 bit global value which is provided by the FPM.

### 5.3.12 Partitioning

The programming model of the FMan allows for partitioning of the resources into separate and independent entities. Some of the configurable registers in the FMan are common to all partitions, and are programmed in common area of memory while other belong to the partition. In FMan\_v3 the reset values of most of the registers are set in hardware to values that implement the most commonly used application and partitioning.

The following section describes the configurable parts of FMan where the partitioning is applied.

#### 5.3.12.1 FMan Memory Map and partitioning

The FMan consumes 1 MB of address space. The FMan address space is divided into regions.

One of the regions is the hardware port memory region. This region is divided into 4 KB pages, one for each hardware port (Rx, Tx, O/H) on FMan. The first 4 KB page is used for what are called “common” registers (see [Table 5-18](#)). The subsequent 63 4 KB pages are allocated to hardware ports 0x1–0x63. See

Section 5.4.2, “FMan Detailed Memory Map,” for more details. The modules containing registers in the 4K page are BMI, QMI and parser. In addition, each Ethernet MAC (mEMAC) is mapped into a separate 4K page.

This memory map concept allows each hardware port (Rx, Tx and O/H) to be mapped into a different partition.

### 5.3.12.2 FIFO partitioning

The FIFO for the mEMACs are located in a region of the FMan internal memory. In FMan\_v3 the HW reset values of the registers are set to allocate FIFO sizes for all the ports. During initialization, management SW may change the values in very special applications (for example, in systems where jumbo frames are smaller than 9K, it is possible to allocate more memory for lookup tables, less for FIFOs). Once the initialization is complete, there is no need to change the values dynamically.

### 5.3.12.3 KeyGen, Policer, storage profile scheme partitioning

KeyGen schemes and Classification plans, Policer profiles and storage profiles may be divided into partitions by programming global configuration registers.

Each storage profile contains up to four buffer pools, therefore through the storage profiles buffer pool partitioning is provided.

The HW reset values allocated all the resources to all the ports. During initialization, SW configuration may split the resources between the ports, where each port may access only its resources. Each port must initialize its values and may modify its partition attributes with FMan controller host commands.

### 5.3.12.4 Custom classifier partitioning

It is possible to configure the custom classifier to configure a separate lookup table for each partition. The SW needs to support partitioning of the FMan internal memory region dedicated for the custom classifier at the virtual address and MMU levels.

### 5.3.12.5 Congestion group partitioning

The QMI congestion groups may be partitioned between the ports. Each port holds N bits to configure the congestion group that corresponds to the port. The HW reset value does not allocate any congestion group.

In FMan\_v3, the Congestion Group Priority Mapping table is updated with an FMan controller host command.

### 5.3.13 FMan Internal Memory

The FMan internal memory is a shared resource that is accessed by all the FMan modules. Each FMan variation has a different size of FMan internal memory. See the applicable DPAA-enabled SoC reference manual for the size of the FMan internal memory.

**Table 5-5. Mapping Different FMAns to Devices**

FMan Type Devices	Size of FMan Internal Memory
LS1043A	384 Kbytes

The FMan internal memory is split into the following regions:

- The transmit (Tx), receive (Rx) FIFOs, offline/host command O/H Port FIFOs, and the internal context (IC) of the frame
- Custom classifier data structures

The size of each of these regions is user-configurable in FMBM\_CFG1 register.

#### 5.3.13.1 FMan Internal Memory for Tx, Rx, and O/H FIFOs

The FIFO region on the FMan internal memory is organized in linked lists of 256-byte buffers. The management of the linked list is done in hardware; therefore, the user does not handle the linked list directly. Each hardware port in the FMan has its own FIFO, and each packet consumes part of the FIFO. The FIFO buffers for the packets are dynamically allocated at the beginning of the processing flow (either Rx, Tx, or O/H), and are automatically released at the end of the processing flow by the BMI (see [Section 5.5.3, “BMI Input NIA”](#)). The FIFO contains buffers to store the data of the frame, and buffers to manage the frame, such as the internal context (IC).

The user must configure the following:

- The base address and size of the FIFO region in the FMan internal memory, which is done in the FMBM\_CFG1 register. This register resides in the common area in the FMan memory map.
- The size of the FIFO allocated for each hardware port (Rx, Tx and O/H), which is done in the FMBM\_PFS\_n registers. These registers reside in the port-specific area of the FMan memory map.
- Additional registers in the BMI are configured with various FIFO thresholds

See [Section 5.5.6.18, “Internal FIFO Configuration Requirements,”](#) and the associated registers for more details on how to configure the FMan FIFO space in the FMan internal memory.

#### 5.3.13.2 FMan Internal Memory for Custom Classifier

The Custom Classifier region of the FMan internal memory is allocated by the user. Each hardware port may be configured to access a separate region for its Custom Classifier data structure. The base address of custom classifier data structures is programmed by the FMBM\_RCCB (for Rx ports) FMBM\_OCCB (for O/H ports) registers. These registers reside in the port-specific area of the FMan memory map. The total size allocated for the Custom Classifier is determined by the configuration of all the Custom Classifier data structures.

### 5.3.14 FMan Parser

The primary function of the packet parse module is to identify the incoming frame for the purpose of determining the desired treatment to apply. This parse function can parse many standard protocols, including options and tunnels, and supports a generic configurable capability to allow proprietary or future protocols to be parsed.

There are several types of parser headers, shown in this table.

**Table 5-6. Parser Header Types**

Header Type	Definition
Self-describing	Announced by proprietary values of Ethertype, Protocol Identifier, Next Header, and other standard fields. They are self-describing in that the frame contains information that describes the presence of the proprietary header.
Non-self-describing	Does not contain any information that indicates the presence of the header. For example, a frame that always contains a proprietary header before the Ethernet header would be non-self-describing.
Proprietary	Can be defined as being self-describing or non-self-describing

**Note:** Both self-describing and non-self-describing headers are supported by means of parsing rules in the FMan.

The underlying notion is that different frames may require different treatment (see [Table 5-7, “Classification types”](#)), and only through detailed parsing of the frame can proper treatment be determined.

The parser output, is a data structure named ‘parse result’. This data structure, which is stored in the internal context (IC) of the frame, is used by the classification modules in the FMan. The FMan may be configured to place it in the margin at the beginning of the first buffer of the frame, and thus pass it to the CPU. The ‘parse array’ is a data structure internal to the parser; the user needs to be exposed to it only if the soft parser is used.

### 5.3.15 FMan Classification, Distribution

After parsing is complete, the user can identify the proper action to be taken. These actions are described in [Table 5-7](#) and are divided into two categories:

- *Hash based: Spreading and distribution* of frames according to hash performed on selected fields of the frame.
- *Table Lookup: Action to take on frame is determined by table lookup on selected fields of the frame.*

**Table 5-7. Classification types**

Treatment	Function	Benefits
Hash	<ul style="list-style-type: none"> <li>Performed by the KeyGen module</li> <li>Allows a hash key to be built from many different fields in the frame to provide differentiation between flows</li> <li>A key can be built exclusively from fields present in the frame based on what the parse function has detected. Alternatively, default values can be used if fields are not present in the frame.</li> <li>Hashes selected fields in the frame as part of a spreading mechanism</li> <li>The result is a specific frame queue identifier (FQID). To support added control, this FQID can be indexed by values found in the frame, such as TOS or p-bits, or any other desired field(s).</li> </ul>	<ul style="list-style-type: none"> <li>Useful when spreading traffic while obeying QoS constraints is required</li> </ul>
Classification look-up	<ul style="list-style-type: none"> <li>Performed by the FMan Controller module</li> <li>Looks up certain fields in the frame to determine subsequent action to take, including policing. The FMan contains internal memory that holds small tables for this purpose.</li> <li>Classification look-ups are performed based on the combination of user configuration and what fields the Parser actually encountered. That is, the user configures the sets of look-ups to perform, and the parse results dictate which one of those sets to use.</li> <li>Look-ups can be chained together such that a successful look-up can provide key information for a subsequent look-up. After all the look-ups are complete, the final classification result provides either a hash key to use for spreading, or a FQID directly.</li> </ul>	<ul style="list-style-type: none"> <li>Useful when hash distribution is insufficient and a more detailed examination of the frame is required</li> <li>Can determine whether policing is required and the policing context to use</li> </ul>

The subsequent choice of FQ could depend on various conditions, including the following:

- The flow being either directed to a particular CPU
- Quality of service consideration
- Control plane/data plane traffic

For example, the most obvious scenario is the distribution of flows on FQs based on their DSCP or IP precedence bits. Thus, up to eight different FQs would be created. CPUs that service those FQs could then preferentially schedule these queues or let the QMan perform preferential scheduling for the CPUs.

### 5.3.16 FMan Policer

Key features of the FMan Policer module are as follows:

- Because the FMan has many policing profiles (this number varies in different FMan implementations), any FQ or group of FQs can be policed to either drop or mark packets if the flow exceeds a preconfigured rate.
- Policing and classification can be used in conjunction for mitigating Distributed Denial of Service Attack (DDOS).
- The policing is based on two-rate, three-color marking algorithm (RFC2698). The sustained and peak rates as well as the burst sizes are user-configurable, and RFC 4115 (sustained and excess rates) is also supported. Hence, the policing function can rate-limit traffic to conform to the rate the flow is mapped to at flow set-up time.

- By prioritizing and policing traffic prior to software processing, CPU cycles can be focused on the important and urgent traffic ahead of other traffic.

## 5.3.17 DPAA congestion management

### 5.3.17.1 Congestion management on Rx

Congestion management on Rx occurs when the QMan detect a congestion in a queue serving for FMan enqueue on Rx. The QMan defines ‘congestion groups’. FQs can be assigned to a congestion group when they are initialized. See section ‘Congestion Management and Avoidance’ in QMan chapter. The number of congestion groups supported varies in each device. When a congestion is detected in one of the congestion groups, the QMan notifies the FMan. The FMan allows mapping of the congestion groups into the physical ports [Section 5.5.4.5.23, “Receive Congestion Group Map Register \(FMBM\\_RCGM\).”](#) Upon goings-on notification, the FMan transmits a pause frame on the ports mapped to the congestion group.

In FMan\_v3, the pause frame complies with ‘priority based flow control’ 802.1Qbb, and contains an 8 bit ‘class enable vector’. Each congestion group has a class enable vector associated with it. This vector is programmed in the Congestion groups priority mapping. table. See [Section 5.5.4.2, “Congestion Group Priority Mapping table.”](#)

### 5.3.18 Congestion management on Tx

Congestion management on Tx occurs when the FMan receives a pause frame from the other end of the Ethernet connection.

In this case, the Ethernet MAC automatically stops transmission over the port that received the pause frame.

In FMan\_v3, if a 802.1Qbb pause frame is received, the QMan traffic class corresponding to the bit in the ‘class enable vector’ of the pause frame is not scheduled to transmit packets as long as the flow control condition is valid. See QMan chapter for more details. See [Section 3.3.5, “Traffic Class \(TC\) Flow Control.”](#)

In FMan\_v3, it is possible to configure the mapping of PFC class (in incoming PFC class enable vector) to QMan traffic class. See [Section 5.5.4.3, “PFC priority mapping to QMan traffic classes.”](#)

### 5.3.19 FMan Functional Flows

The following sections depict examples of functional flows, which illustrate the capabilities of the FMan. The FMan is very flexible and can be configured to operate in many different ways to implement different applications. The figures may introduce some terminology that has not yet been defined in the specification and is clarified in subsequent sections.

## 5.3.19.1 Receive (Rx) Flows

### 5.3.19.1.1 Configurable Receive (Rx) Flows

Figure 5-4 depicts multiple receive flows that are supported by the FMan. The flows are configurable by programming the NIA value in the different FMan modules.

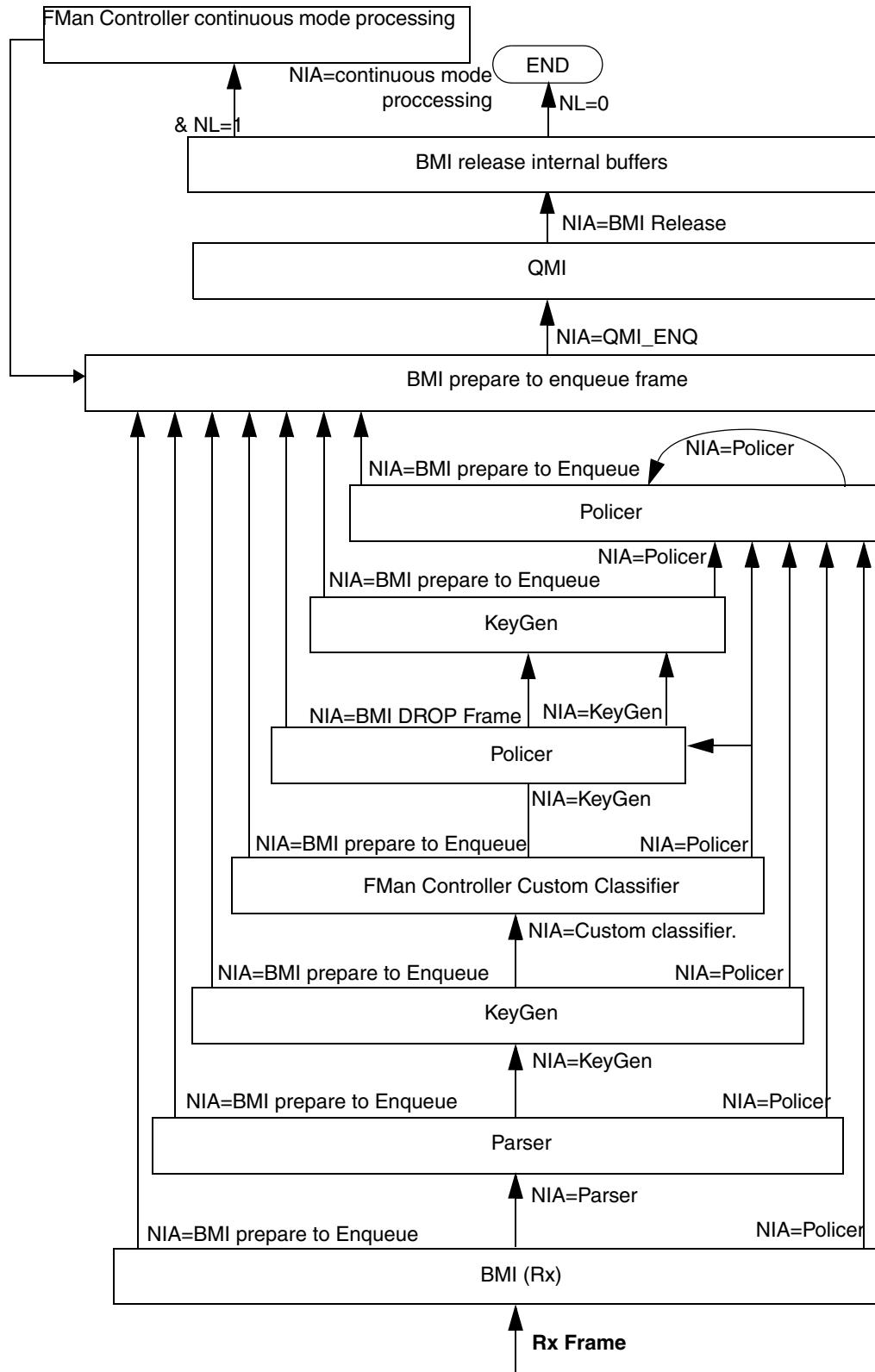
For example, Figure 5-4 shows that after the BMI (Rx), it is possible to invoke either the Parser, the BMI (prepare to enqueue), or the Policer.

Important things to consider about configurable Rx flows are as follows:

- The simplest flow uses the default Frame queue ID (FQID) programmed in FMBM\_RFQID, and it enqueues all frames in one queue without any processing. For this flow, the NIA for the BMI (Rx) is configured to “BMI (prepare to enqueue).” In this configuration, packets are processed in four stages:
  - BMI (Rx)
  - BMI (prepare to enqueue)
  - QMI (enqueue)
  - BMI (release internal buffers)
- If policing is needed to limit the incoming rate, the NIA for the BMI (Rx) is configured to the Policer, and then the NIA for the Policer is configured to “BMI (prepare to enqueue).” In this configuration, packets are processed in five stages.
- If full classification and distribution to multiple queues is needed, a longer flow is configured. Usually, the main Rx flow (simplified) follows these steps:
  - a) Packets are received from one of the Ethernet MACs, are temporarily stored in the FMan internal memory, then written to the SoC memory via the FMan DMA.
  - b) The packet header (maximum size 256 bytes) is stored in the FMan internal memory and the module’s IC structure is allocated.
  - c) The packet is parsed by the Parser.
  - d) According to parsing results, the packet is then classified by the FMan controller, KeyGen, or both, and then policed.
  - e) At the end of this process, packets are enqueued to a frame queue or dropped. An example of such flow is described in [Section 5.3.19.1.2, “Receive \(Rx\) Flow Example with Custom Classifier.”](#)

Because the NIAs are evaluated for each packet in a given system, different packets may go through different flows.

The NIAs are user-programmable; therefore, it is conceivable that more flows are possible, although not likely to be needed, in most applications. The following figure shows the flows supported in FMan. A more detailed version of this figure is shown in [Section 5.4.4.1, “NIA Register Configuration—Rx Flows.”](#)



**Figure 5-4. Configurable Receive (Rx) Flows<sup>1</sup>**

### 5.3.19.1.2 Receive (Rx) Flow Example with Custom Classifier

This section describes an example of a receive flow. This example shows some of the FMan capabilities, and is used as a base for the functional description of the FMan. Note that this example does not describe a real application.

In this example, the KeyGen is configured for two schemes (a scheme defines the behavior of the Keygen for packets containing a specific protocol stack, which is defined by the user):

- *Scheme1* identifies TCP/UDP packets and extracts a quintuple look-up key. This scheme generates the CCBASE for the FMan Controller, default FQID, PNUM, and the NIA for the next module (in this example, the NIA is for the FMan Controller).
- *Scheme2* identifies IP packets (that are not TCP/UDP) and generates an FQID with the hash value (statistical distribution). The NIA in this scheme is configured for the Policier. Packets that do not match any scheme (non-IP packets) are directed by the KeyGen to the Policier and are enqueued to a default queue with the default NIA programmed in FMKG\_GCR[DEFNIA].

TCP/UDP packets go to the FMan Controller Custom Classifier module to perform exact match to identify certain TCP/UDP flows. On these flows (match in the look-up table), the Custom Classifier replaces the FQID, and the NIA to proceed to the BMI. Flows that do not match in the Custom Classifier use the default FQID from the KeyGen, and the NIA is the Policier.

In some applications (not depicted in the example), it may be necessary to direct all packets to the FMan Controller Custom Classifier after the first KeyGen invocation. The KeyGen is able to modify the CCBASE, which is the entry point for the Custom Classifier action descriptor (AD) on a per-frame basis. With this mechanism, it is possible to choose the first Custom Classifier AD based on the protocol stack of each frame, thus minimizing the number of KeyGen schemes. The Custom Classifier may determine the FQID of a few flows (such as “control plane” flows), while the FQID for the remaining flows (such as “data plane” flows) is determined with statistical distribution (hash-based) by the KeyGen (in the second invocation). To implement this flow, the Custom Classifier is configured to compare with exact match only the control plane packets. The packets that do not have an exact match in the Custom classifier (that is, data plane flows), are directed to the KeyGen module for a second time. It is also possible to architect a system in which most frames are processed by the KeyGen only once by comparing with exact match the data plane frames.

The example contains detailed information about registers that are programmed to configure the NIAs in the flow. Some of the NIAs contain an “xx..” value. These values are user-defined, and are normally values specific to the processing module being invoked (for example, for the Policier, the “xxxx” is the Policier profile ID).

---

1. For a more detailed figure of the Rx flows, see [Section 5.3.19.1, “Receive \(Rx\) Flows.”](#)

**QorIQ LS1043A Data Path Acceleration Architecture (DPAA) Reference Manual, Rev. 0**

## Frame Manager (FMan)

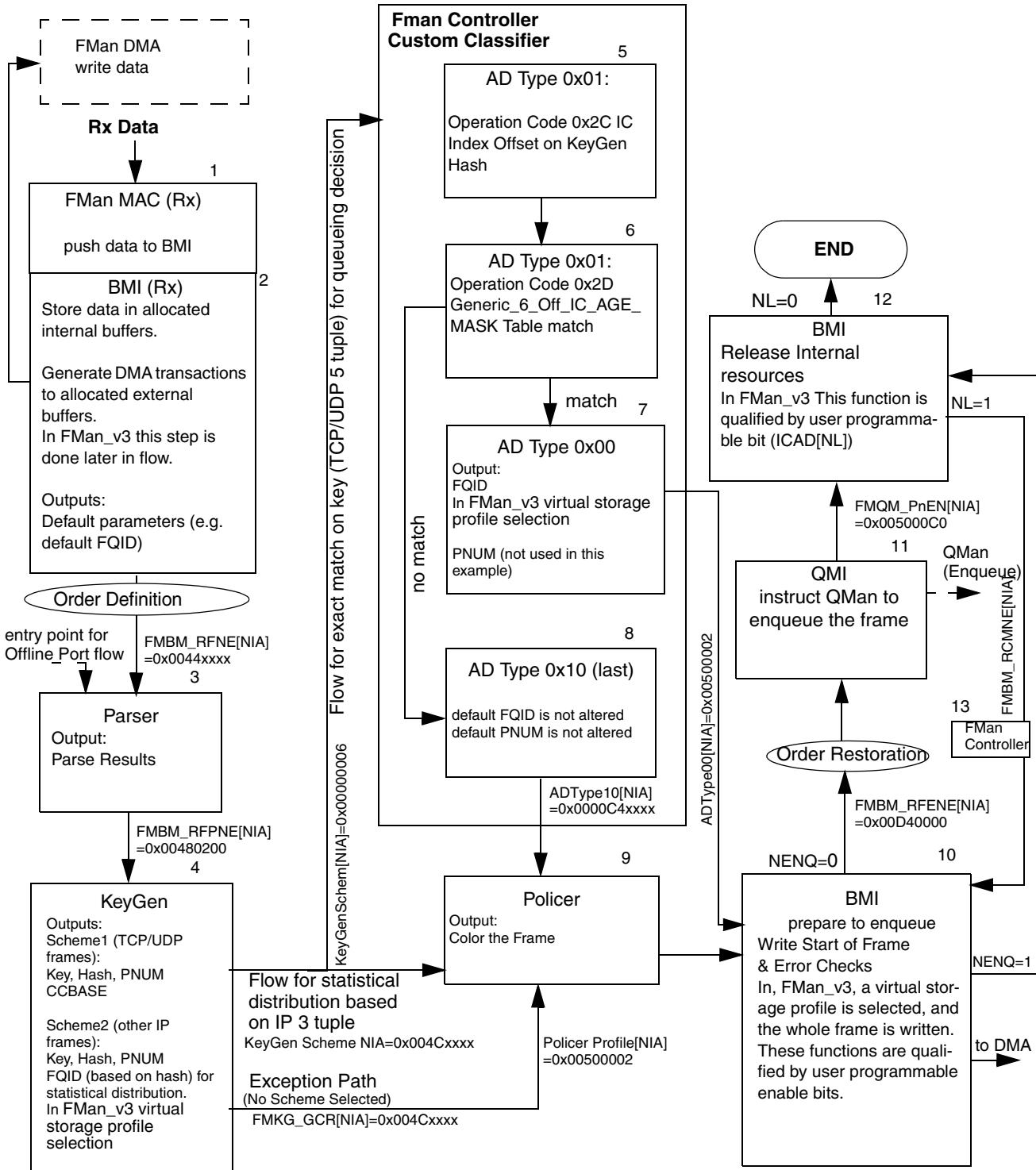


Figure 5-5. FMan Receive (Rx) Functional Flow (Example)<sup>1</sup>

1. This figure uses detailed terminology from subsequent chapters of the FMan specification. First-time readers should skip over the details.

**Table 5-8. FMan Receive (Rx) Functional Flow (Example)**

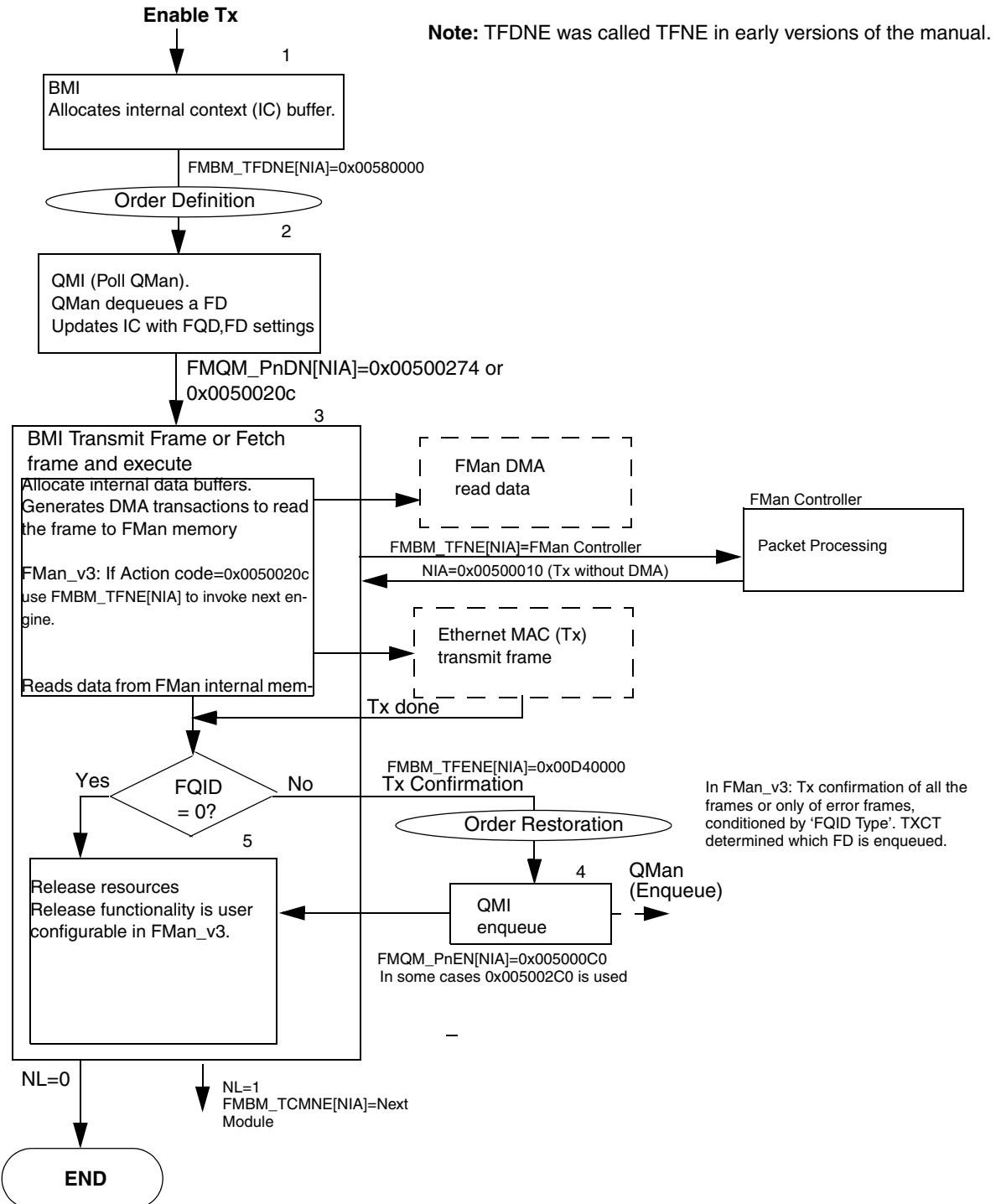
<b>Stage</b>	<b>Module</b>	<b>Description</b>
1	FMan MAC	<ul style="list-style-type: none"> <li>Gets frame data from the Ethernet physical interface.</li> <li>Pushes data into the FMan BMI.</li> </ul>
2	FMan BMI	<ul style="list-style-type: none"> <li>Stores incoming data in FMan internal memory (as a temporary buffer).</li> <li>Acquires external buffers from BMan pools.</li> <li>In FMan_v3 this step is done later in flow. Generates Direct Memory Access (DMA) transactions to store the tail of the frame in allocated buffers. The tail of the frame is the part of the frame which is not stored in the first internal buffer.</li> <li>Frame Descriptor (FD) status and parser results are initialized.</li> <li>User configured NIA directs processing to the parser; the NIA is configured in the FMBM_RFNE register of the appropriate Rx port page.</li> <li>Enqueued Frame Queue ID's (FQID) initial value is based on FMBM_RFQID of the appropriate Rx port page.</li> <li>The order of the incoming frames is preserved with the order definition mechanism.</li> <li>See section <a href="#">Section 5.5.5.3.2, “BMI ‘Rx Frame’</a>, for more details</li> </ul>
3	FMan Parser	<ul style="list-style-type: none"> <li>Parses the frame and identifies network layers</li> <li>Parser generates a parser result used by subsequent FMan processing elements.</li> <li>User configured NIA directs processing to the KeyGen; the NIA is configured in the FMBM_RFPNE register of the appropriate Rx port page.</li> <li>This step can be the entry point for the Offline Port flow. See <a href="#">Figure 5-7</a>.</li> </ul>
4	FMan KeyGen	<ul style="list-style-type: none"> <li>Generates a key (based on user configurations), parser results, and frame data.</li> <li>Hash value is calculated based on the key. This value is used to determine the FQID for statistical distribution (where needed).</li> <li>KeyGen can steer the processing flow for different protocols stacks. As needed, user initializes KeyGen schemes as per incoming traffic.</li> <li>Processing flow may be steered in three user-programmable ways: <ul style="list-style-type: none"> <li>KeyGen scheme1, (IP-TCP/UDP frames) steers the processing flow to the FMan Controller for 5 tuple exact match. The KeyGen modifies the FMan Controller CCSBASE to point to an Custom Classifier Action Descriptor (AD).</li> <li>KeyGen scheme2, (other IP frames) steers the processing flow to the Policer, and the hash result is used to form an FQID for statistical distribution.</li> <li>KeyGen scheme (no scheme selection), which handles the exception path, has an NIA for the next functional block. <a href="#">Figure 5-7</a>, for example, shows this as a Policer for avoiding DoS attacks.</li> </ul> </li> </ul>
5	FMan Controller AD Type 0x01: Operation Code 0x2C	<ul style="list-style-type: none"> <li>Uses the KeyGen calculated hash value to choose a bucket.</li> <li>First stage of an exact match, hash table lookup.</li> <li>A table—containing a key list (for exact matches) and bucket-associated actions—is attached to each bucket.</li> </ul>
6	FMan Controller AD Type 0x01: Operation Code 0x2D	<ul style="list-style-type: none"> <li>The bucket, with key list, is searched in order to find an entry with the same key (exact match) as that generated by KeyGen.</li> <li>If there is a match then the Custom Classifier Action Descriptor (AD), associated with that key, is used. If there isn't a match then the last AD is used.</li> <li><a href="#">Figure 5-7</a> shows that the Policer is invoked when there is a lookup miss (exception path).</li> </ul>
7	FMan Controller AD Type 0x00	<ul style="list-style-type: none"> <li>The next processing stage after the FMan Controller is determined by the NIA configured in this AD.</li> <li>FQID— where the frame is queued at the end of processing—is also configured in this AD.</li> <li>FMan Controller execution can end in this AD or continue to an extension AD.</li> <li>This AD updates the Internal Context Action Descriptor (ICAD) field (FQID, etc.)</li> </ul>

**Table 5-8. FMan Receive (Rx) Functional Flow (Example) (continued)**

<b>Stage</b>	<b>Module</b>	<b>Description</b>
8	FMan Controller AD Type 0x10	<ul style="list-style-type: none"> <li>This AD is invoked when no exact match entry is found in the table in stage 7. The next processing stage after the FMan Controller is determined by the NIA configured in this AD.</li> <li>This AD does not affect the FQID. The default FQID programmed in the FMBM_RFQID register is used to enqueue the frame; the NIA—configured in the previous AD—is used.</li> </ul>
9	FMan Policer	<ul style="list-style-type: none"> <li>The frame is colored as per the user-configured profile.</li> <li>Profiles can be selected relative to the port, or in an absolute way.</li> <li>Profiles, for the Policer, are coded in the NIA.</li> </ul>
10	FMan BMI 'Prepare to Enqueue'	<ul style="list-style-type: none"> <li>Check for errors/indications in the Frame Descriptor (FD) status.</li> <li>Error frames are discarded/moved to default or error queues based on FD status bits (and user configurable masks). Error FQID is configured in the FMBM_REFQID register on the appropriate Rx port page.</li> <li>In FMan_v3 - If ICAD[EBAD] = 0 (from classification stage), allocate buffers from the storage profile (hardware or virtual, if enabled) for the entire frame from BMan pools. Storage profile virtualization per port is evaluated with FMBM_SPICID. For Rx ports ICAD[EBAD] must be reset. If ICAD[FWD]=0, the frame is written in external memory. If ICAD[CWD]=0, part of the Internal Context (IC) is written in external memory.</li> <li>User configured NIA directs processing to the QMI; the NIA is configured in the FMBM_RFENE register of the appropriate Rx port page. In FMan_v3, this step is conditioned by ICAD[NENQ]=0. If ICAD[NENQ]=1 the enqueueing step is skipped, and processing is continued directly in BMI 'Release Internal Resources'.</li> <li>Order restoration is required before enqueue.</li> <li>See <a href="#">Section 5.5.5.3.3, "Rx BMI 'Discard Frame' or 'Prepare to Enqueue Frame'"</a> for more details.</li> </ul>
11	FMan QMI	<ul style="list-style-type: none"> <li>Generates a frame enqueue request to the QMan with the FQID value.</li> <li>User configured NIA directs processing to the BMI; the NIA is configured in the FMQM_PnEN register of the appropriate Rx port page. n is the number associated with the port.</li> <li>QMan enqueues the frame</li> </ul>
12	BMI 'Release Internal Resources'	<ul style="list-style-type: none"> <li>If ICAD[NL] = 0 (from classification stage), Release all internal resources consumed by the task.</li> <li>If ICAD[NL] = 1 (from classification stage), internal resources are not released and FMBM_CMNE[NIA] is used to invoke the next stage. This is useful for virtualization, where Rx frames are enqueued to more than one queue destined to different CPUs (i.e. Rx frame replicator).</li> </ul>
13	FMan Controller 'Continuous Mode' processing	This stage is needed in applications where the same packet needs to be processed more than once. For example, the packet header may be modified, the whole packet may be copied to an other buffer, and enqueued to an other queue(s). It is possible to circle this loop numerous times. The next stage is BMI 'Prepare to enqueue' (stage 10).

### **5.3.19.2 Transmit (Tx) Flow Example**

This figure depicts an example of the transmit flow. The functionality depicted is of typical paths that packets traverse when processed by the FMan. On Tx, the frames are transmitted via the desired MAC with optional IP/TCP/UDP checksum generation.



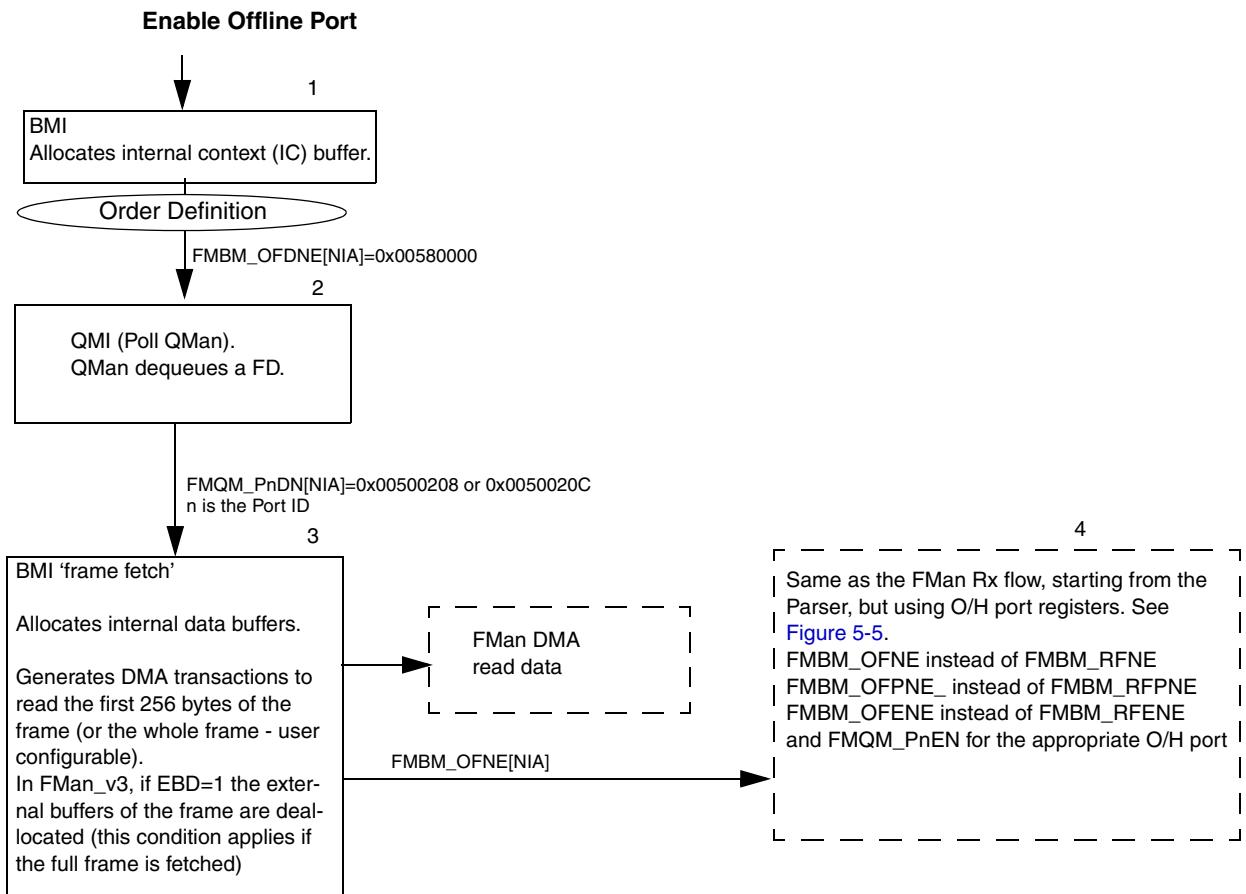
**Figure 5-6. FMan Transmit (Tx) Functional Flow (Example)**

**Table 5-9. FMan Transmit (Tx) Functional Flow (Example)**

<b>Stage</b>	<b>Module</b>	<b>Description</b>
1	FMAN BMI	<ul style="list-style-type: none"> <li>Allocates an internal context (IC) buffer for the frame. Writes default values to this buffer.</li> <li>The next processing module is the QMI (to dequeue an FD from the QMan). The NIA is configured in FMBM_TFDNE of the appropriate Tx port.</li> <li>The default FMBM_TFCA[OR]=1.</li> </ul>
2	FMan QMI	<ul style="list-style-type: none"> <li>Generates a dequeue request to the QMan.</li> <li>After dequeue, user configured NIA directs processing to the BMI; the NIA is configured in the FMQM_PnDN register of the appropriate Tx port page. n is the number associated with the port.</li> <li>Updates internal context with the FQID and MAC commands extracted from the dequeued FQID context. See section <a href="#">Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision.”</a></li> <li>The QMan performs the dequeue procedure</li> </ul>
3	FMan BMI	<ul style="list-style-type: none"> <li>Generates DMA transactions that read the frame into FMan internal memory.</li> <li>If FD[RPD] is set, generates DMA transaction that reads portion of the IC into FMan internal memory. FMBM_TICP or FMBM_VICP is used to configure the IC copy parameters.</li> <li>In FMan_v3: If the input action code is ‘fetch frame and execute’ the BMI invokes the next module according to FMBM_TFNE. If FD[RPD] is set, generates DMA transaction that reads portion of the IC into FMan internal memory. FMBM_TICP or FMBM_VICP (from storage profile) are used to configure the IC copy parameters. In FMan_v3, absolute SPID (ASPID) is used to fetch the storage profile (if enabled). See <a href="#">Section 5.5.4.1, “Storage Profiles.”</a></li> <li>Reads frame data from FMan internal memory and pushes it to the Ethernet MAC.</li> <li>After the Ethernet MAC is done transmitting, the BMI checks if Tx confirmation is configured for this frame: If the FQID=0 no confirmation is needed; otherwise, the frame is enqueued back to the QMan (see next steps).</li> <li>Error frames are enqueued to an error queue.</li> <li>If confirmation is needed (FQID!=0): <ul style="list-style-type: none"> <li>In FMan_v3: FQID Type is checked. Either all the frames are enqueued, or only error frames are enqueued in the confirmation queue. TXCT determines which FD is enqueued in the confirmation.</li> <li>The QMI is invoked.</li> </ul> </li> </ul>
4	FMan QMI	<ul style="list-style-type: none"> <li>Enqueues the frame.</li> <li>The QMan performs enqueue procedure.</li> </ul>
5	FMan BMI	<ul style="list-style-type: none"> <li>Release resources:</li> <li>In FMan_v3: External buffers are deallocated if ICAD[EBD]=1 AND there is no error. Internal resources are released if ICAD[NL]=0. If ICAD[NL]=1 internal resource are not released and the FMan invokes the next module as programmed FMBM_TCMNE.</li> </ul>

### 5.3.19.3 FMan Offline Port Flow Example

This figure depicts an example of the Offline Port flow. The functionality depicted is of typical paths that packets traverse when processed by the FMan.



**Figure 5-7. FMan Offline Port Functional Flow (Example for Offline Parsing)**

**Table 5-10. FMan Offline Port Functional Flow (Example for Offline Parsing)**

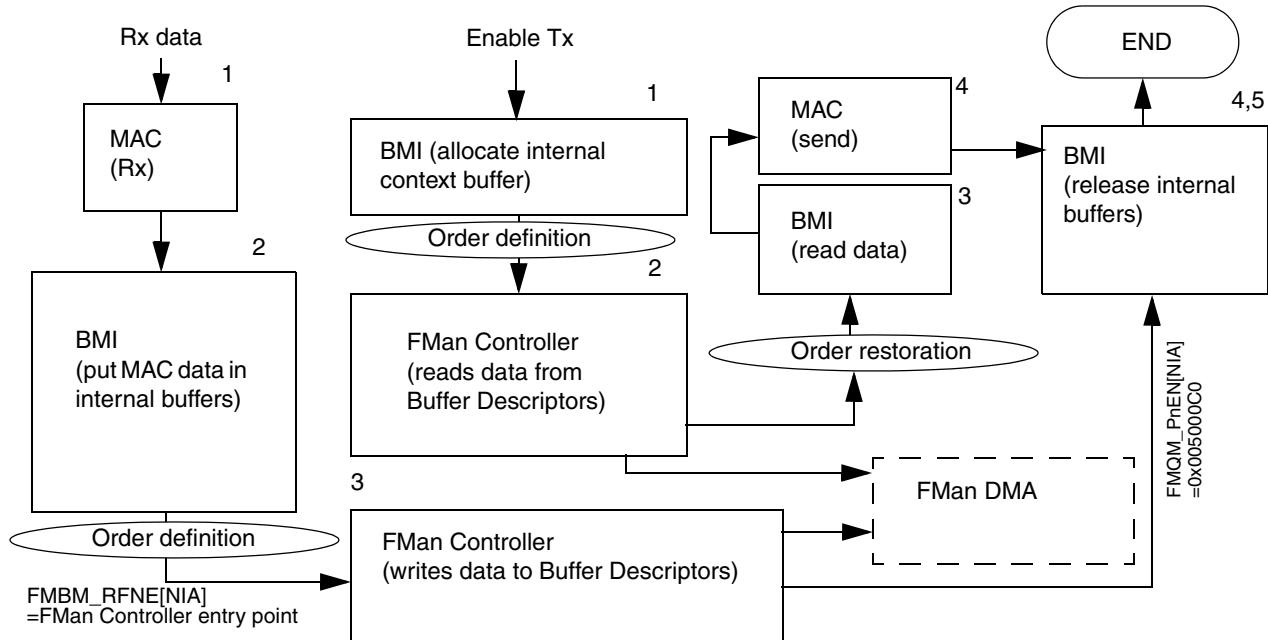
Stage	Module	Description
1	FMAN BMI	<ul style="list-style-type: none"> <li>Allocates an internal context (IC) buffer for the frame. Write default ICAD, CCBASE and HPNIA to the IC.</li> <li>User configured NIA directs processing to the QMI.</li> </ul>
2	FMan QMI	<ul style="list-style-type: none"> <li>Generates a dequeue request.</li> <li>Initializes internal context with the FQID extracted from the dequeued FQID context.</li> <li>After dequeue, user configured NIA directs processing to the BMI.</li> </ul>

**Table 5-10. FMan Offline Port Functional Flow (Example for Offline Parsing) (continued)**

Stage	Module	Description
3	FMan BMI	<ul style="list-style-type: none"> <li>Generates DMA transactions that read the first 256 bytes of the frame into FMan internal memory. If FD[RPD] is set, generates DMA transaction that reads portion of the IC into FMan internal memory. FMBM_OICP or FMBM_VICP (from storage profile) are used to configure the IC copy parameters. In FMan_v3, absolute SPID (ASPID) is used to fetch the storage profile (if enabled). See <a href="#">Section 5.5.4.1, “Storage Profiles”</a>.</li> <li>In FMan_v3, Absolute value for storage profile is used for fetching IC (if enabled). See <a href="#">Section 5.5.4.1, “Storage Profiles”</a>.</li> <li>It is also possible to invoke the BMI with an action code to read (fetch) the whole frame from external memory, if necessary.</li> <li>Write to the IC default values of PR lines that were not copied in DMA operation.</li> <li>If ICAD[EBD]=1, and the full frame is fetched, the BMI deallocates the external buffers, and returns them to the BMan. This bit should be set (and ICAD[EBAD] should be reset), for frames that are copied to other buffers.</li> </ul>
4	FMan Parser and subsequent steps	This example depicts the most common flow where the Offline Ports used to parse and classify frames from the CPU. From this point, the flow is identical to the flow described in <a href="#">Figure 5-5</a> . It is also possible to perform other functions with the Offline Port.

### 5.3.19.4 Independent Mode (IM) Flow

This figure shows the flow for Rx and Tx independent mode flow. See [Section 5.12.15, “Ethernet Independent Mode \(IM\)”](#), for more information.

**Figure 5-8. Independent Mode Flow (Rx and Tx)**

This table describes the independent Rx flow.

**Table 5-11. Independent Rx Flow**

Stage	Module	Operation
1	MAC	Receives data from the communication interface
2	BMI	Allocate internal buffers and transfer data coming from MAC into FMan internal memory.
3	FMan Controller	Write data to BDs by generating DMA transactions.
4	BMI	Releases internal buffers that contain the frame and the internal context of the frame

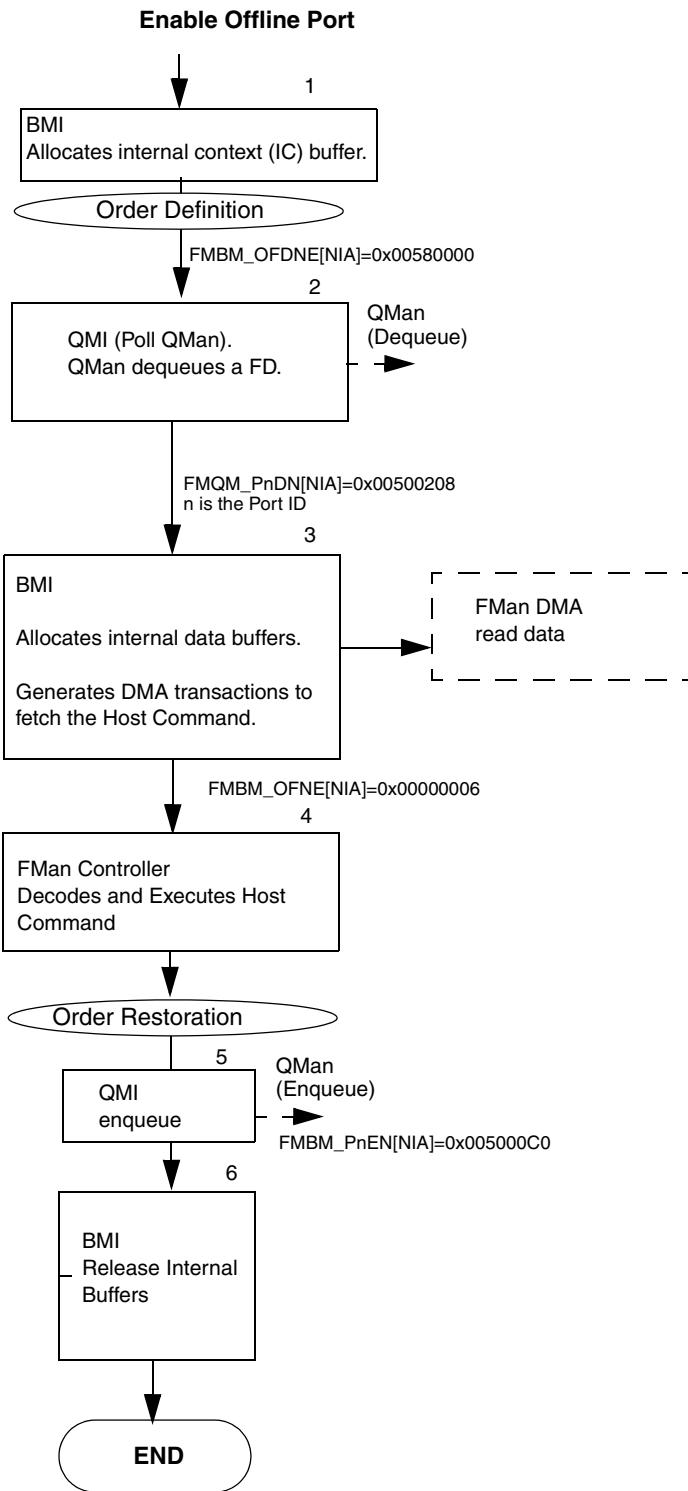
This table describes the independent Tx flow.

**Table 5-12. Independent Tx Flow**

Stage	Module	Operation
1	BMI	Allocates internal frame context buffer
2	FMan Controller	Allocates internal data buffers, Generates DMA transaction from BDs in order to read the data
3	BMI	Transfers the data from the internal memory to the MAC
4	MAC	Sends the data on the native communication interface
5	BMI	Releases internal buffers

### 5.3.19.5 Host Command Flow

Host commands are dequeued from the QMan. After the commands are completed, the result of the HOST command or its confirmation is transferred to the software through the QMan. [Figure 5-9](#) shows a general flow for HOST commands. See [Section 5.12.16, “Host Commands,”](#) for more information.

**Figure 5-9. FMan Host Command Flow**

**Table 5-13. HOST Command Flow**

<b>Stage</b>	<b>Module</b>	<b>Operation</b>
1	BMI	Allocates internal frame context buffer
2	QMI	Polls for frame descriptor (FD) from QMan QMan dequeues frame descriptor (FD) from a frame queue (FQ).
3	BMI	Generates DMA to read the command
4	FMan Controller	Processes the host command. The FMan controller generates a command success/fail indication, which is enqueued to the QMI
5	QMI	Enqueues the HOST command result/confirmation on a queue through QMan. QMan enqueues the host command result/confirmation.
6	BMI	Releases internal buffers and internal context of the command

## 5.4 Frame Manager Detailed Functional Description

### 5.4.1 FMan Hardware Ports

The FMan supports a few types of hardware ports and a few ports of each type. The FMan hardware ports are referred to as “sub-portals” in some instances.

Each hardware port on the FMan has its own hardware PortID. This table specifies the super-set of all hardware PortIDs. See [Table 5-15](#) and the applicable DPAA-enabled SoC reference manual for device specific information related to availability of FMan ports on each device.

**Table 5-14. Hardware PortIDs**

Type	PortID	
Offline/Host Command 1	0x01 <sup>1</sup>	
Offline/Host Command 2	0x02	
Offline/Host Command 3	0x03	
Offline/Host Command 4	0x04	
Offline/Host Command 5	0x05	
Offline/Host Command 6	0x06	
Offline/Host Command 7	0x07	
	Tx	Rx
1/ 2.5 Gbps Eth 1 <sup>2</sup>	0x28	0x08
1/ 2.5 Gbps Eth 2	0x29	0x09
1/ 2.5 Gbps Eth 3	0x2A	0x0A
1/ 2.5 Gbps Eth 4	0x2B	0x0B
1/ 2.5 Gbps Eth5	0x2C	0x0C

**Table 5-14. Hardware PortIDs (continued)**

Type	PortID	
1/ 2.5 Gbps Eth 6	0x2D	0x0D
1/ 2.5 Gbps Eth 7	0x2E	0x0E
1/ 2.5 Gbps Eth 8	0x2F	0x0F
10 Gbps Eth 1 or 2.5 / 1 Gbps Eth 9	0x30	0x10
10 Gbps Eth 2 or 2.5 / 1 Gbps Eth 10	0x31	0x11

<sup>1</sup> Not supported in FMan\_v3<sup>2</sup> In some SoCs only 1GbE is supported

The following table shows which PortIDs are supported in current devices. See the applicable DPAA-enabled SoC reference manual for devices which are not depicted in this table and for other device specific information related to the DPAA.

**Table 5-15. FMan Hardware Ports in Devices**

Type	Offline/Host Command Ports (O/H n)							1 / 2.5 Gbps Ethernet								1 / 2.5 / 10 Gbps Eth	1 / 2.5 / 10 Gbps Eth	
	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6	n = 7		n = 1	n = 2	n = 3	n = 4	n = 5	n = 6	n = 7	n = 8	n = 9	n = 10
Hardware PortID	0x01	0x02	0x03	0x04	0x05	0x06	0x07	Rx	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	0x10	0x11
								Tx	0x28	0x29	0x2A	0x2B	0x2C	0x2D	0x2E	0x2F	0x30	0x31
LS1043A	NO	YES	YES	YES	YES	NO	NO		YES <sup>1</sup>	YES	YES <sup>1</sup>	YES <sup>1</sup>	YES <sup>1</sup>	YES <sup>1</sup>	NO	NO	YES	NO

<sup>1</sup> Only 1 Gbps supported. See the SerDes chapter of the SoC manual for configuration options and exclusions.

### 5.4.1.1 FMan PortIDs

There are two types of PortIDs in the FMan:

- Hardware PortIDs are related to physical ports and are fixed in hardware. In devices with multiple instantiations of the FMan, the same hardware PortIDs in both FMans are used in a symmetrical fashion. The hardware PortIDs are used by FMan modules to allocate certain resources to specific hardware ports for the purpose of partitioning or virtualization. The hardware port is sometimes referenced as ‘<n>’ in the FMan chapters.
- Logical PortID is used to distinguish physical ports with the same hardware PortID in devices with multiple FMans. This is done by initializing different values in the logical PortID field in each FMan. A logical PortID is a field in the ‘parse result’ configurable in the BMI Rx and O/H hardware ports initialization registers (See [Section 5.5.4.5.14, “Rx Parse Result Initialization](#)

Register (FMBM\_RPRI),” Section 5.5.4.7.15, “O/H Parse Result Initialization Register (FMBM\_OPRI)”). This field may be modified by the SW parser (Section 5.9.4.10, “Parse Result”).

## 5.4.2 FMan Detailed Memory Map

The FMan consumes 1 MB of address space. Table 5-17 describes the regions of the programming model occupied by elements of the FMan. This table shows the superset of all the possible ports on the FMan.

The addresses in Table 5-17 are relative to the FMan base address in the SoC. If more than one FMan is instantiated in the SoC, a base address is allocated for each instantiation of the FMan.

**Table 5-16. FMan base address in SoC Memory map**

Type	FMan base address in SoC
LS1043A	FMan: 1A0_0000h

See the CCSR Block Base Address Map in SoC reference manual for more details.

See the Section 1.9, “Specific DPAA Implementation Details” or the applicable SoC reference manual for details on the availability of ports and MDIO interfaces.

Important things to consider when accessing the FMan registers are as follows:

- All accesses to and from the FMan registers must be made as 32-bit accesses.
- There is no support for accesses of sizes other than 32 bits.
- Writing to reserved register bits must always preserve the previous value, because changing the value of reserved bits may have unintended side effects.
- Unless otherwise specified, the read value of unmapped registers or of reserved bits in mapped registers is not defined, and must not be assumed to be 0.

### 5.4.2.1 Hardware Port Pages in the FMan Memory Map

Table 5-17. FMan Memory Map Regions

FMan Controller—SoC Base Address (see device specific memory map)			
Offset	Size	Name	Section/Page
0x0_0000–0x7_FFFF	512 KB	FMan memory. Each variation of FMan contains a different size of FMan internal memory. See SoC FMan introduction chapter for details.	—
0x8_0000–0xB_FFFF	256 KB	FMan hardware ports memory region. The memory region is divided into 4 KB pages, arranged according to their port ID where the first 4 KB page is common.	5.4.2.1/5-34
0xC_0000–0xC_0FFF	4 KB	Policer	5.11.4/5-476
0xC_1000–0xC_1FFF	4 KB	Key generator (KeyGen)	5.10.2/5-414
0xC_2000–0xC_2FFF	4 KB	FMan DMA	5.8.4/5-320
0xC_3000–0xC_3FFF	4 KB	Frame processing manager (FPM)	5.7.3/5-280
0xC_4000–0xC_4FFF	4 KB	FMan controller configuration data	5.12.4/5-550
0xC_5000–0xC_6FFF	8 KB	Reserved	
0xC_7000–0xC_7FFF	4 KB	Soft Parser	—
0xC_8000–0xD_7FFF	64 KB	Reserved	
0xD_9000–0xD_BFFF	12 KB	Reserved	
0xD_B000–0xD_BFFF	4 KB	FMan_v3: Congestion groups priority mapping table	
0xD_C000–0xD_CFFF	4 KB	FMan_v3: Virtual Storage Profiles	
0xD_D000–0xD_FFFF	12 KB	Reserved	
0xE_0000–0xE_07FF	2 KB	EMAC1 (2.5/1Gbps) (mEMAC <sup>1</sup> )	6.4.2/65-5
0xE_1000–0xE_1FFF	4 KB	MDIO-1 for EMAC1 <sup>2</sup>	6.4.2/65-5
0xE_2000–0xE_27FF	2 KB	EMAC2 (2.5/1Gbps) (mEMAC)	6.4.2/65-5
0xE_3000–0xE_3FFF	4 KB	MDIO-2 for EMAC2	6.4.2/65-5
0xE_4000–0xE_47FF	2 KB	EMAC3 (2.5/1Gbps) (mEMAC)	6.4.2/65-5
0xE_5000–0xE_5FFF	4 KB	MDIO-3 for EMAC3	6.4.2/65-5
0xE_6000–0xE_67FF	2 KB	EMAC4 (2.5/1Gbps) (mEMAC)	6.4.2/65-5
0xE_7000–0xE_7FFF	4 KB	MDIO-4 for EMAC4	6.4.2/65-5
0xE_8000–0xE_87FF	2KB	EMAC5 (2.5/1Gbps) (mEMAC)	6.4.2/65-5
0xE_9000–0xE_9FFF	4 KB	MDIO-5 for EMAC5	6.4.2/65-5
0xE_A000–0xE_A7FF	2 KB	EMAC6 (2.5/1Gbps) (mEMAC)	6.4.2/65-5
0xE_B000–0xE_BFFF	4 KB	MDIO-6 for EMAC6	6.4.2/65-5
0xF_0000–0xF_07FF	2 KB	EMAC9(10/2.5/1Gbps) (mEMAC 9)	6.4.2/65-5
0xF_1000–0xF_1FFF	4 KB	MDIO9 for EMAC9	6.4.2/65-5

**Table 5-17. FMan Memory Map Regions (continued)**

FMan Controller—SoC Base Address (see device specific memory map)			
Offset	Size	Name	Section/Page
0xF_4000–0xF_BFFF	48 KB	Reserved	
0xF_C000–0xF_CFFF	4 Kbytes	FMan_v3: Dedicated MDIO1 <sup>3</sup> • External MDIO for Clause 22 devices (EMI1) <sup>4</sup>	<a href="#">6.4.3.4/65-46</a>
0xF_D000–0xF_DFFF	4 Kbytes	FMan_v3: Dedicated MDIO2 • External MDIO for Clause 45 devices (EMI) <sup>5</sup>	<a href="#">6.4.3.4/65-46</a>
0xF_E000–0xF_EFFF	4 KB	IEEE 1588	<a href="#">7.4.1/75-2</a>
0xF_F000–0xF_FFFF	4 KB	Reserved	

<sup>1</sup> mEMAC is ‘Multirate Ethernet Media Access Controller (MAC)’ used in FMan\_v3.

<sup>2</sup> Some of the MDIO interfaces may not be available at the device external signals. Those management interfaces not configured to use external signals, are still available for use with the corresponding SGMII interface. Please see the device external signals chapter).

<sup>3</sup> The FMan includes two sets of dedicated MDIOs, in addition to an MDIO for each Ethernet. The dedicated MDIOs are connected to the IO interface (see SoC reference guide ) Two Interrupts from the FMan MDIO are connected to the GIC(Generic interrupt controller) (see SoC reference manual and the Ethernet management interface (EMI) in the data sheet.)

<sup>4</sup> MDIO1 is EMI1 for Clause 22 physical interfaces (1G, 2.5G, overclocked SGMII and below).

<sup>5</sup> MDIO2 is EMI2 for Clause 45 physical interfaces (10G).

Each hardware port is assigned a 4 KB, “port-specific” page in the FMan hardware port memory region (which is part of the FMan memory map). The first 4 KB in the FMan hardware ports memory region is used for what are called “common” registers (see [Table 5-18](#)). The subsequent 63 4 KB pages are allocated to hardware ports 0x1–0x63. Each 4 KB page is divided in 4 \* 1 KB sections that are allocated to FMan modules BMI, QMI, and Parser respectively (see the table below).

**Table 5-18. FMan Hardware Port Page Memory Map**

Hardware Port Base Address <sup>1</sup>	Offset	Size	FMan Module
0x8_0000	0x0000–0x03FF	4 KB	BMI common
	0x0400–0x07FF		QMI common
	0x0800–0x0BFF		Reserved common
	0x0C00–0x0FFF		Reserved common
	0x< n >000–0x< n >3FF	63* 4 KB	BMI (< n >=1..63)
	0x< n >400–0x< n >7FF		QMI (< n >=1..63)
	0x< n >800–0x< n >BFF		Parser (< n >=1..63)
	0x< n >C00–0x< n >FFF		Reserved (< n >=1..63)

<sup>1</sup> The hardware port base address is added to the offset. Note that the offset is 6 bits long. Note that < n > is 6 bits long.

In the following chapters, port-specific registers are numbered as follows:

- “Offset” is the offset from the FMan base address (check in BMI, Parser, and QMI if the register offsets are from the FMan base or from the block base address).
- “*<n>*” is the hardware port base address (same as PortID).
- “xxx” is the offset from the port base address (for the internal address space of the hardware port).
- The FMan architecture supports up to 63 hardware ports, therefore the *<n>* values run from 1..63.
- Each variation of the FMan supports a different subset of the possible hardware ports (and PortIDs).

### 5.4.2.2 Examples for the Evaluation of Addresses of FMan Registers

The following examples show how to calculate the absolute address for a module (DMA) and for a hardware port page in the FMan.

The examples show how to calculate the offset of the registers from the memory-mapped configuration, control, and status registers (CCSRs) base address in the SoC.

At the SoC level, the memory-mapped configuration, control, and status registers (CCSRs) location is programmable using CCSR base address register high and CCSR base address register low (CCSRBARTH and CCSRBARL); together, these registers are referred to as CCSRBAR. Refer to device specific SoC reference guide for base address in each device. (See Accessing Configuration, Control, and Status Registers under Memory map/Register section of the SoC reference guide).

#### 5.4.2.2.1 Calculate the SoC Absolute Address for an FMan Register

This figure shows an example of how to calculate the address of the KeyGen global status register (FMKG\_GSR).

SoC										FMan module Internal address space									
SoC (8 bits, or more in some SoCs)										0000_0010_0100 <sup>3</sup>									
SoC address space <sup>4</sup>										FMan address space									

Figure 5-10. FMan Hardware Port Pages Address Space

<sup>1</sup> FMan base address in SoC (example): 0x0040\_0000. See the applicable SoC reference manual.

<sup>2</sup> FMan KeyGen module offset address. See [Table 5-17, “FMan Memory Map Regions.”](#)

<sup>3</sup> FMQM\_GSR register. See [Table 5-359, “KeyGen Memory Map.”](#)

<sup>4</sup> In some SoCs, the SoC address space may have more bits. The most significant byte is configurable at the SoC level.

As a result, the offset for FMKG\_GSR register in the SoC CCSR space is 0x004C\_1024.

#### **5.4.2.2.2 Calculate the SoC Absolute Address for FMan Hardware Port Page Register**

This figure shows an example of how to calculate the address of PortID  $n$  dequeue NIA register for hardware port number 0x04 (FMQM\_P4DN for offline port) (FMQM\_PnDN).

**Figure 5-11. FMan Hardware Port Pages Address Space**

- <sup>1</sup> FMan hardware ports pages offset. See [Table 5-17, “FMan Memory Map Regions.”](#)
  - <sup>2</sup> FMan modules are BMI, QMI, Parser. See [Table 5-18, “FMan Hardware Port Page Memory Map.”](#) For this example, the QMI is chosen.
  - <sup>3</sup> See memory map table in each module chapter.
  - <sup>4</sup> FMQM\_PnDN register. See [Table 5-195, “QMI Memory Map”](#)
  - <sup>5</sup> In some SoCs, the SoC address space may have more bits.

As a result, the address for FMQM\_P4DN register in the SoC CCSR space is 0x0048\_442C.

### **5.4.3 FMan Frame Internal Context (IC)**

The frame internal context (IC) is a data structure associated to every frame being processed. For every new frame, the IC is automatically allocated in the FMan internal memory and is initialized with user-configurable, initial values.

The IC is shared by all the FMan modules and contains status information about the frame and actions that need to be performed by the FMan modules. Each module that processes a frame reads and updates certain fields of the IC. At the end of the processing of each frame, its IC is automatically released.

**Table 5-19. Internal Context (IC)**

Offset	Field Size (Bytes)	Field Name	Description
0x00	16	FD	<p>Frame descriptor (FD)</p> <p>The FD contains pointers to the external buffer data offset and length of the frame. It also contains a status, which is defined below. See <a href="#">Section 5.4.3.2, “Frame Descriptor (FD).”</a></p>
0x10	8	ICAD	<p>Internal Context Action descriptor</p> <p>Describes the necessary action taken by the FMan at the end of the frame processing. Fields in the ICAD can be over written by a processing module to change the behavior.</p> <p>See <a href="#">Section 5.4.3.3, “Internal Context Action Descriptor (ICAD),”</a> for details and initial values.</p>
0x18	4	CCBASE	<p>Custom classifier base</p> <p>Custom classifier base initial value is programmed in the FMBM_RCCB register. The low byte may be overwritten by the KeyGen to direct the custom classifier to the right search tree (see <a href="#">Section 5.10.3.12.12, “KeyGen Scheme Entry Custom Classifier Bit Select Register (AWR20_RFMODE_FMKG_SE_CCBS).”</a> Only the three least significant bytes of this field are used by the FMan Controller.</p>

Table 5-19. Internal Context (IC) (continued)

Offset	Field Size (Bytes)	Field Name	Description
0x1c	1	KS	Key size Size of the key calculated by the KeyGen
0x1d	3	HPNIA	Hardware parser NIA Specifies which module processes the frame after the FMan Parser. Initial value of this field is programmed in FMBM_RFPNE. This value may be over written by the soft parser.
0x20	32	PR	Parser result  Rx: A 32 byte region that contains the result of the parsing of the frame. Initial value is programmed in FMBM_RPRI. It is possible to program FMBM_RPRI with a Logical port ID that can be used for classification to distinguish between two FMans in the same device.  O/H: Similar to Rx. Initial value is programmed in FMBM_OPRI or can be initialized for each frame at the beginning of the buffer. These values can be copied to the IC by enabling in FD Command [RPD].  Tx: User supplied parser result for TCP checksum calculation. The parser result indicates to the transmitter where to find the IP/TCP stack in the packet.
0x40	8	Time Stamp	Rx This is the time stamp captured by the Ethernet MAC (1G and 10G) when the frame is received. If the timestamp feature is disabled in the Ethernet MAC, this field is zeroed.  Tx: this is the time stamp captured by the physical PHY when the frame is transmitted. If the timestamp feature is disabled in the Ethernet MAC, this field is zeroed.  O/H: This field is not changed by the FMan. The value in this field may be valid depending on the origin of the frame (e.g. if the frame's origin is the receiver of an other FMan in the device).
0x48	8	HASH	Rx and O/H Hash—64-bits of the Hash Value calculated by the KeyGen module on the extracted key. This value is qualified with the IC[KS] value, meaning if the KS value is 0 than the HASH value has not been over written by the KeyGen.  Tx: This field is not changed
0x50	56	KEY	For Rx and O/H Key that is extracted from the frame headers by the KeyGen module. This field contains valid information if KeyGen module is enabled. If IC[KS]=0 this field is not valid.  Tx: This field is not changed. See KeyGen chapter
0x88	8	—	Reserved
0x90 <sup>1</sup>	112	Debug	When working in debug mode, every module working on the frame can dump debug information in this region. The debug information can be prepended to the frame's data. Debug trace information per module is defined in each of the module sections.  The debug offset in the IC is programmable. FMBM_RDCFG[DTO], FMBM_TDCFG[DTO], FMBM_ODCFG[DTO] configures the starting address for debug information in the IC.

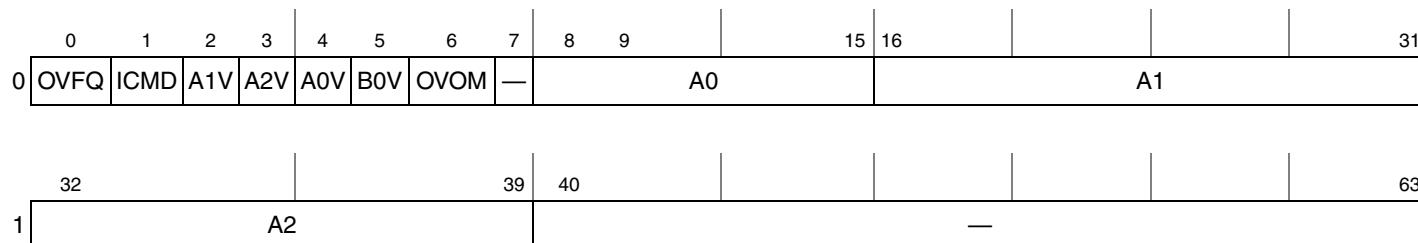
<sup>1</sup> The offset in the IC for debug information is programmable.

#### **5.4.3.1    Override IC from FQD or Data Buffer**

During the frame dequeue operation (on Tx or O/H ports), the FMan can be configured to extract some fields from the frame buffer or from the FQD (see [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\)”](#)) and override some fields in the internal context (IC). These fields are used by the FMan modules. The exact usage is described in the relevant sections as needed.

#### **5.4.3.1.1 Frame Queue Descriptor (FQD) Context A**

Context A field in the FQD can be configured to override some fields in the IC.



**Figure 5-12. FQD[Context A] for FMan dequeue**

This table describes the Context A fields as they are used by the FMan when a frame is dequeued (Tx or O/H).

**Table 5-20. FQD[Context A] for FMan dequeue description**

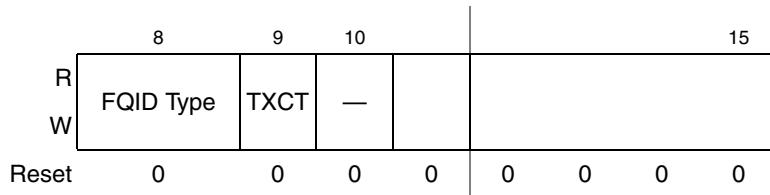
Bits	Name	Description
0	OVFQ	<p>Override Frame Queue</p> <p>This bit affects the enqueue decision tree for Tx (confirmation) or O/H ports.</p> <p>0 - FQID in Context B is not used for enqueueing the frame</p> <p>1 - FQID in Context B may be used for enqueueing the frame (subject to other conditions)</p> <p>This bit is supported in all versions of the FMan.</p>
1	ICMD	<p>Invalidate Tx or O/H FD[Command].</p> <p>0 - Do not ignore FD[Command] bits. FMan acts according to the description in “<a href="#">Tx Frame Descriptor Command/Status Word</a>” or “<a href="#">Offline Port Frame Descriptor Command/Status</a>.”</p> <p>1 - Ignore the FD[Command] bits which are set. Action for all the bits should be as if they are zero.</p> <p>This bit is normally reset. It is set in applications where the bits in the FD[Command] bits are set by other modules in the SoC in a way which cannot be interpreted by the FMan. See specific application note for the need of setting this bit.</p>
2	A1V	<p>A1 field is valid</p> <p>0 - A1 field in Context A of the FQD is not valid</p> <p>1 - A1 field in Context A of the FQD is written in the frame Internal Context Action Descriptor (ICAD[A1]) (ICAD bits 48-63) overriding any default value.</p> <p>See <a href="#">Section 5.4.3.3, “Internal Context Action Descriptor (ICAD)”</a>.</p>

**Table 5-20. FQD[Context A] for FMan dequeue description (continued)**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
3	A2V	A2 field is valid 0 - A2 field in Context A of the FQD is not valid 1 - A2 field in Context A of the FQD (except VSPE bit) is written in the frame Internal Context Action Descriptor (ICAD[A2]) (ICAD bits 32-39) overriding any default value. See <a href="#">Section 5.4.3.3, “Internal Context Action Descriptor (ICAD).”</a>
4	A0V	A0 field is valid 0 - A0 field in Context A is not valid. 1 - A0 field in Context A of the FQD is written in the frame Internal Context Action Descriptor (ICAD[A0]) (ICAD bits 40-47) overriding any default value. See <a href="#">Section 5.4.3.3, “Internal Context Action Descriptor (ICAD).”</a>
5	B0V	B0 field is valid 0 - B0 field in Context B is not valid. 1 - B0 field in Context B of the FQD and VSPE bit in Context A A2 field are written in the frame Internal Context Action Descriptor (ICAD[B0]-ICAD bits 0-7 and ICAD bit 39) overriding any default value. See <a href="#">Section 5.4.3.3, “Internal Context Action Descriptor (ICAD).”</a>
6	OVOM	OVerride Operational Mode bits. 0 - ‘Setting one’ mode: Do not override operational mode bits in Internal Context Action Descriptor (ICAD). Set to ‘one’ bits which are set in Context A A2 field. 1 - ‘Override mode’: Override operation mode bits with values programmed in Context A A2 field. See <a href="#">Section 5.5.6.1, “Operational Mode bits.”</a> <b>Note:</b> VSPE is not affected by OVOM since it is not an operational mode bit.
7	—	Reserved
8–15	A0	This field is split into subfields. Interpretation of this field is described in <a href="#">“Context A—A0 Field Description for Tx Port”</a> and <a href="#">“Context A—A0 Field Description for Offline Port.”</a>
16–31	A1	See A1V bit. Interpretation of this field is described in <a href="#">Section 5.12, “Frame Manager—FMan Controller,”</a> and is application-specific.
32–39	A2	See A2V bit. Interpretation of this field is described in <a href="#">“Context A—A2 Field Description for Tx Port”</a> and <a href="#">“Context A—A2 Field Description for Offline Port.”</a> This field contains seven operational mode bits and the VSPE bit.
40–63	—	Reserved

## Context A—A0 Field Description for Tx Port

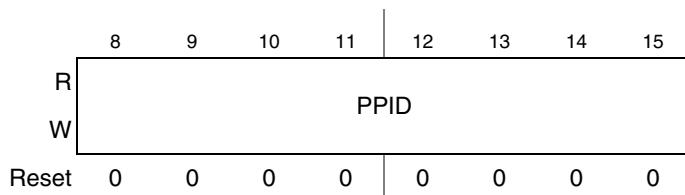
This section describes the detailed bits in Context A field A0 as they are interpreted by Tx ports when a frame is dequeued from the QMan. Note that the index of the bits is taken from [Figure 5-12, “FQD\[Context A\] for FMan dequeue.”](#) The valid bit (A0V) is also depicted here.

**Figure 5-13. Context A—A0 Field Description for Tx Port****Table 5-21. Context A—A0 Field Descriptions for Tx Port**

Bits	Name	Description
8	FQID Type	<p>FQID Type This bit is used in Tx ports. 0 - If Tx confirmation is enabled (FQID!=0) all frames are enqueued in the confirmation queue 1 - If Tx confirmation is enabled (FQID!=0) only error frames are enqueued in the confirmation queue. See <a href="#">Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision.”</a></p> <p><b>Note:</b> In FMan_v3 Context A field A0 is validated by A0V and is placed in ICAD[40-47]. See <a href="#">Table 5-33</a>.</p>
9	TXCT	<p>Tx Confirmation Type 0 - FD which was dequeued for transmission is used for confirmation 1 - FD for confirmation is taken from first 16 bytes of the frame payload, pointed by FD. This bit is needed if the user needs to receive Tx confirmation in order to release a buffer which is not the buffer pointed by the FD which is dequeued. In this case the FMan enqueues in the confirmation queues, an FD taken from the first 16 bytes of the frame payload, which contains the pointer to the buffer that needs to be released. The transmitter omits the first 16 bytes of the payload (which are not transmitted). See <a href="#">Section 5.5.6.12, “DMA resource Load balancing”</a> for more details.</p>
10	—	Reserved

### Context A—A0 Field Description for Offline Port

This section describes the detailed bits in Context A field A0 as they are interpreted by Offline ports when a frame is dequeued from the QMan. Note that the index of the bits is taken from [Figure 5-12, “FQD\[Context A\] for FMan dequeue.”](#)

**Figure 5-14. Context A—A0 Field Description for Offline Port**

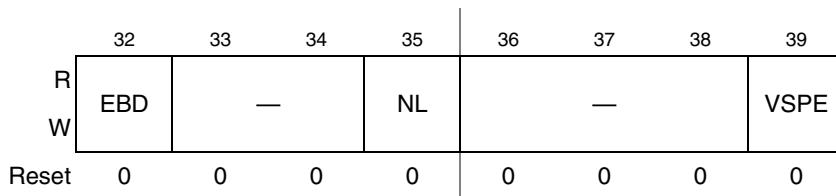
**Table 5-22. Context A—A0 Field Descriptions for Offline Port**

Bits	Name	Description
8-15	PPID	Policer profile ID.

**Context A—A2 Field Description for Tx Port**

This section describes the detailed bits in Context A field A2 as they are interpreted by the FMan when a frame is dequeued by the FMan on Tx port. The FMan interprets the EBD and NL bits as the ‘operational mode bits’. See [Section 5.3.5.3, “Operational mode bits.”](#) The VSPE bit is part of the virtual storage profile.

Note that the index of the bits is taken from [Figure 5-12](#).

**Figure 5-15. Context A—A2 Field Description for Tx Port****Table 5-23. Context A—A2 Field Descriptions for Tx Port**

Bits	Name	Description
32	EBD	External Buffer Deallocation 0 - External buffers are not deallocated at the end of the FMan flow. 1 - External buffers are deallocated at the end of the FMan flow.
33–34	—	Reserved. Set to zero.
35	NL	Not Last (continuous mode) 0 - The BMI releases all the internal buffers. 1 - The BMI does not release all the internal buffers.
36–38	—	Reserved. Set to zero.
39	VSPE	Virtual Storage Profile Enable 0 - Virtual storage profile is disabled. IC parameters are taken from FMBM_TICP 1 - Virtual storage profile is enabled. IC parameters are taken from FMBM_VICP according to ASPID (will be described later). <a href="#">See Section 5.3.7, “Virtual Storage Profiles.”</a> <b>Note:</b> VSPE bit is valid if Context A[B0V] is set.

**Context A—A2 Field Description for Offline Port**

This section describes the detailed bits in Context A field A2 as they are interpreted by the FMan when a frame is dequeued by the FMan on an Offline port. The FMan interprets seven bits [32-38] as the ‘operational mode bits’. See [Section 5.3.5.3, “Operational mode bits.”](#) The VSPE bit is part of the storage profile.

## NOTE

The bit descriptions in this section, are also applicable to Rx port ‘operational mode bits’. After being initialized by BMI registers, on Rx ports, the bits are (optionally) updated by the classification stages, while on offline ports these bits are (optionally) updated by Context A during dequeue, and (optionally) updated by the classification stages.

Note that the index of the bits is taken from [Figure 5-12](#).

	32	33	34	35	36	37	38	39
R	EBD	EBAD	FWD	NL	CWD	NENQ	—	VSPE
W	0	0	0	0	0	0	0	0
Reset								

**Figure 5-16. Context A—A2 Field Description for Offline Port**

**Table 5-24. Context A—A2 Field Descriptions for Offline Port**

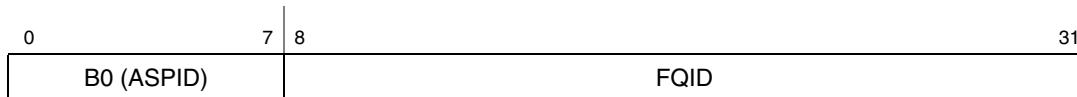
Bits	Name	Description
32	EBD	External Buffer Deallocation 0 - External buffers are not deallocated 1 - External buffers from which the frame is read are deallocated back to BMan if the frame read operation completed without errors. Example: This bit is set if the Offline port is used to move a frame to an other buffer. The new buffer is allocated from a buffer pool selected by the virtual storage profile. By setting this bit the FMan is instructed to release the buffers of the original frame.
33	EBAD	External Buffer Allocation Disable 0 - External buffers are allocated by the FMan to store data. 1 - Disable External allocation to store data. <b>Note: For Offline ports</b> - If ICAD[VSPE]=0 (Virtual Storage Profiles are disabled), the FMan ignores this bit, and does not allocate buffers. Example: This bit determines if the Offline port is used to move or copy a frame to an other buffer. The new buffer is allocated from a buffer pool selected by the virtual storage profile. When this bit is cleared the FMan is instructed to allocate the new buffers.
34	FWD	Frame Write Disable 0 - FMan writes frame to external memory 1 - FMan does not write frame to external memory <b>Note: For Offline ports</b> - Only the portion of the frame that is fetched, is written to external memory (see BMI action codes 0x20C, 0x208, <a href="#">Table 5-45</a> .). This bit does not affect writing the IC to the first buffer of the frame.Example: This bit is cleared if the Offline port is used to move or copy a frame to an other buffer. This bit is set if the Offline port is used move the frame from to queue to queue, and there is no need to change the frame payload, or change its storage profile.
35	NL	Not Last (continuous mode) 0 - The BMI releases all the internal buffers at the end of the flow. The flow is ended. 1 - The BMI does not release all the internal buffers at the end of the flow. The flow is continued to the next engine. Example: This bit is used internally by the FMan and should be cleared in the context.

**Table 5-24. Context A—A2 Field Descriptions (continued)for Offline Port (continued)**

Bits	Name	Description
36	CWD	Context Write Disable 0 - Copy (part of) the Internal Context into the margin at the beginning of the frame. 1 - Do not copy the (part of) the Internal Context into the margin at the beginning of the frame.
37	NENQ	No ENQueue 0 - The BMI when in 'Prepare to Enqueue' stage, uses FMBM_OFENE to invoke the next module (normally the QMI). 1 - The BMI when in 'Prepare to Enqueue' stage, automatically invokes BMI 'Release Internal Resources' stage. See <a href="#">Figure 5-5</a> . Example: This bit is useful, for a reassembly function. Incoming fragments are reassembled, and written to a buffer in the BMI 'prepare to enqueue' stage. As long as the entire frame is not reassembled, it is not enqueued.
38	—	Reserved. Set to zero.
39	VSPE	Virtual Storage Profile Enable 0 - Virtual storage profile is disabled. Hardware port registers are used to configure parameters related to DMA transactions. Allocation of new buffers is disabled. 1 - Virtual storage profile is enabled. Virtual storage profile configuration is used to configure parameters related to dma transactions. Allocation new buffers is possible (if enabled in EBAD) using information in virtual storage profile configuration. See <a href="#">Section 5.3.7, "Virtual Storage Profiles."</a> <b>Note:</b> VSPE bit is valid if Context A[B0V] is set.

#### 5.4.3.1.2 Frame Queue Descriptor Context B

The FQID and Virtual Storage Profile ID field in the ICAD may be overwritten by the FQID field in Context B. See [Section 5.5.6.9, "Tx confirmation enqueue and buffer deallocation decision,"](#) and [Section 5.5.5.8, "BMI Host Command Flow,"](#) for details on the conditions that qualify this action.

**Figure 5-17. Context B Structure****Table 5-25. Context B - field bit Descriptions**

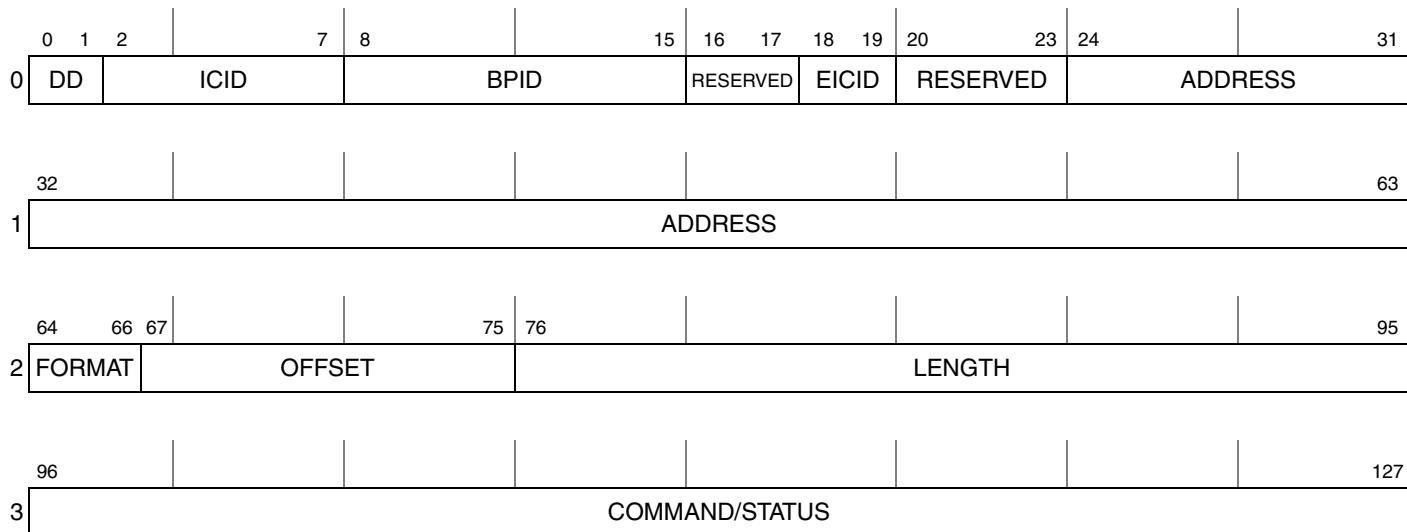
Bits	Name	Description
0-7	ASPID	Absolute virtual Storage Profile ID. See <a href="#">Section 5.5.4.1, "Storage Profiles"</a> for details on the usage of this field. <b>Note:</b> ASPID field is valid if Context A[B0V] is set.
8-31	FQID	Frame Queue ID

In the Tx and Offline flows, the user may overwrite the initial values of a user programmable part of the IC, with values stored in the first buffer in a user programmable margin before the beginning of the frame. This allows generating a default IC for each frame. This mode is enabled by setting the RPD bit in the Frame Descriptor (FD) Command/Status word, and is configured in FMBM\_TICP and FMBM\_OICP for

Tx and Offline ports respectively. This feature is useful in order to generate TCP checksum. The user puts the Parse Results for the frame being transmitted in the first buffer of the frame. The FMan uses some of the offsets to locate the L3/L4 parts of the frame for checksum calculation.

### 5.4.3.2 Frame Descriptor (FD)

The Frame Descriptor (FD) is a data structure which is associated with each frame in the DPAA. All DPAA components recognize this data structure. Some of the fields in the FD are interpreted differently in each DPAA component.



**Figure 5-18. Frame Descriptor**

The frame descriptor (FD) is described in [Chapter 1, “Data Path Acceleration Architecture \(DPAA\) Overview.”](#) The following table describes FMan specific details of the FD:

**Table 5-26. Frame Descriptor Fields**

Bits	Name	Description
0–1	DD	Dynamic Debug marking code point See <a href="#">Chapter 1, “Data Path Acceleration Architecture (DPAA) Overview.”</a>
2–7	ICID	Isolation context identifier (6 lsbits of complete ICID) See <a href="#">Chapter 1, “Data Path Acceleration Architecture (DPAA) Overview.”</a> and <a href="#">Section 5.5.6.14, “ICID (Isolation Context Identifier).”</a>
8–15	BPID	Buffer Pool ID See <a href="#">Chapter 1, “Data Path Acceleration Architecture (DPAA) Overview.”</a>
16–17	Reserved	—
18–19	EICID	Extended ICID (2 msbits of complete ICID) See <a href="#">Chapter 1, “Data Path Acceleration Architecture (DPAA) Overview.”</a> and <a href="#">Section 5.5.6.14, “ICID (Isolation Context Identifier).”</a>
20–23	Reserved	—

**Table 5-26. Frame Descriptor Fields**

Bits	Name	Description
24–63	ADDRESS	Address The memory address of the start of the buffer holding the frame data or the buffer containing the scatter/gather list describing the frame. This field must be aligned to 16 bytes; the suggested alignment is 64 for optimal memory usage with the caches on the device. Appropriate programming of BMan and QMan data structures are needed. See <a href="#">Chapter 1, “Data Path Acceleration Architecture (DPAA) Overview.</a>
64–66	FORMAT	Format of Frame Descriptor (FD) The FMan supports two values in this field: 000 - Short single buffer simple frame 100 - Short multi buffer simple frame Other values are illegal. See <a href="#">Chapter 1, “Data Path Acceleration Architecture (DPAA) Overview.”</a>
67–75	OFFSET	See <a href="#">Chapter 1, “Data Path Acceleration Architecture (DPAA) Overview.”</a>
76–95	LENGTH	Length Total number of valid bytes of data in the frame. The FMan supports the following maximum values in this field: Receive (Rx) ports - up to 16,358 bytes Transmit (Tx) ports - the minimum between 65,535 bytes and the Tx FIFO size Offline ports - The minimum between 65,535 bytes and the port fifo size
96–127	COMMAND/STATUS	See <a href="#">Section 5.4.3.2.1, “FD Command/Status Word,”</a> for details. For Rx: FD-bits 0:7 initial value is programmed in by FMBM_RFNE[FDGS].

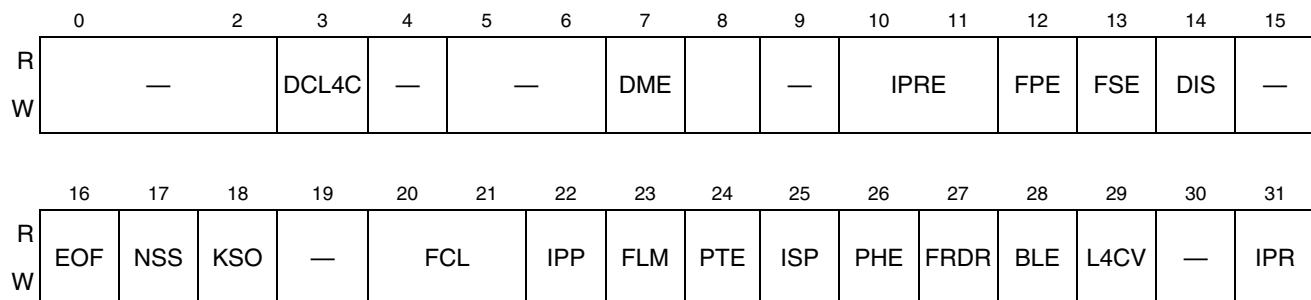
#### 5.4.3.2.1 FD Command/Status Word

The following sections describe the command/status word of the frame descriptor in the following flows: Rx, Tx, Offline Port, Host command.

#### Rx Frame Descriptor Status Word

This figure describes the frame descriptor status bits for Rx. The first byte, except bit 7, initial values can be configured in the BMI.

Bits 0–7 in the Rx FD Status are initialized by the user configurable FMBM\_RFNE[FDGS]. A bit which is set by the user in FMBM\_RFNE[FDGS] is reflected in the corresponding bit, and is not reset by the FMan hardware.

**Figure 5-19. Rx FD Status**

**Table 5-27. Rx FD Status Field Description**

<b>Bits</b>	<b>Name</b>	<b>Command/ Status</b>	<b>Per-Packet Default Value</b>	<b>Description</b>
0–2	—	—	FMBM_RFNE[FD CS]	Reserved. Corresponding bits in FMBM_RFNE[FD CS] must be cleared.
3	DCL4C	Status	FMBM_RFNE[FD CS]	L4 (IP/TCP/UDP) Checksum validation 0 - L4 (IP/TCP/UDP) checksum is validation is enabled 1 - L4 (IP/TCP/UDP) checksum is validation is disabled The IP/TCP/UDP checksum validation is enabled/disabled by programming the corresponding bit in FMBM_RFNE[FD CS]. This bit reflects the value programmed in FMBM_RFNE[FD CS].
4–6	—	—	FMBM_RFNE[FD CS]	Reserved. Corresponding bits in FMBM_RFNE[FD CS] must be cleared.
7	DME	Status	—	DMA error. 0 - No DMA error occurred on this frame 1 - DMA error occurred when accessing external memory. See <a href="#">Section 5.5.6.20.7, “DMA Error”</a> When DME is set in DMA write transactions, no counting takes place.
8–9	—	—	—	Reserved
10–11	IPRE	—	—	Same as offline description
12	FPE	Status	—	Frame Physical Error 0 - No physical error associated with this frame 1 - A physical error is associated with this frame One of the following events is associated with this frame: Rx FIFO overflow, FCS error, code error, running disparity error (SGMII mode), FIFO parity error. PHY Sequence error, PHY error control character detected. For statistics on the specific errors please see the appropriate MAC (10G or 1G) specification. Counter FMBM_RBFC is incremented when this bit is set.
13	FSE	Status	—	Frame Size Error 0 - No size error is detected on this frame 1 - A size error is detected on this frame  In FMan_v3, this bit is set if one of the following conditions occur: <ul style="list-style-type: none"><li>• Frame truncation as a result of 1GEC or 10GEC HW FIFO overflow</li><li>• Frame truncation as a result of FMan internal FIFO overflow (IFSZ)</li><li>• Frame truncation as a result of violation of Maximum Frame Length Register (MAXFRM)</li></ul> <b>Note:</b> See <a href="#">Section 5.5.6.18.1, “Internal FIFO for Rx Ports”</a> for more details on violation of IFSZ rule.

**Table 5-27. Rx FD Status Field Description (continued)**

<b>Bits</b>	<b>Name</b>	<b>Command/ Status</b>	<b>Per-Packet Default Value</b>	<b>Description</b>
14	DIS	Status	—	<p><b>Discard</b>  This bit is set only for frames that are supposed to be discarded, but are enqueued in an error queue for debug purposes. For correct operation of this bit, FMBM_RFSDM[14] must be set.  Typically this bit is used during debug to identify frames which are supposed to be discarded during the Classification.  0 - The BMI Input Action Code was of type ‘prepare to enqueue’ (no discard).  1 - The BMI Input Action Code (per this specific frame) was of type ‘discard’. Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set.</p>
15	—	—	—	Reserved.
16	EOF	Status	—	<p><b>Extract Out of Frame Error</b>  0 - No key extraction error  1 - Key extraction goes out of the frames data.  This bit is set if the KeyGen attempts to extract a field which does not exist in the frame. Note: If KeyGen is not invoked this bit is never set.  Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set.</p>
17	NSS	Status	—	<p><b>No Scheme Selection</b>  0 - KeyGen attempt to find a scheme for this frame succeeded.  1 - KeyGen attempt to find a scheme for this frame failed.  Note: If KeyGen is not invoked this bit is never set.  Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set</p>
18	KSO	Status	—	<p><b>Key Size Over flow Error</b>  0 - No key size error  1 - Key size is too large (overflow)  This bit is set if a software configuration error is detected. If set, the classification result is not reliable.  Note: If KeyGen is not invoked this bit is never set.  Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set</p>
19	—	—	—	Reserved

**Table 5-27. Rx FD Status Field Description (continued)**

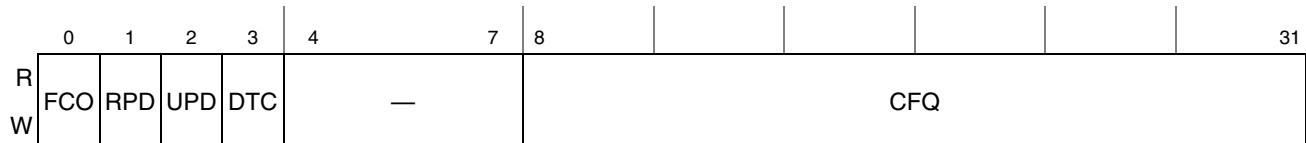
<b>Bits</b>	<b>Name</b>	<b>Command/ Status</b>	<b>Per-Packet Default Value</b>	<b>Description</b>
20–21	FCL	Status	—	<p>Frame color as determined by the Policer</p> <p>00 green 01 yellow 10 red 11 no reject</p> <p>This field is used the QMan WRED. The QMan WRED operation uses this field to determine rejection of the frame. If FCL=11 the QMan does not reject the frame.</p> <p>The initial value is programmed in FMBM_RFCA[COLOR].</p> <p>Note: If Policer is not invoked these bits are not set.</p> <p>Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set</p>
22	IPP	Status	—	<p>Illegal Policer Profile error0 - No illegal policer profile error 1 - Illegal policer profile is selected</p> <p>This bit is set if a software configuration error is detected. It is set, if the policer profile selected is not initialized.</p> <p>Note: If Policer is not invoked this bit is never set.</p> <p>Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set</p>
23	FLM	Status	—	<p>Frame Length Mismatch—the offset to the policer is larger than the frame size.</p> <p>0 - Frame length calculated by the policer is not negative 1 - Frame length calculated by the policer is negative</p> <p>This bit is set if a software configuration error is detected.</p> <p>Note: If Policer is not invoked this bit is never set.</p> <p>Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set</p>
24	PTE	Status	—	<p>Parser Time-out (Cycle Limit Exceed) Error</p> <p>0 - No parser time-out error 1 - Parser time-out error</p> <p>See <a href="#">Section 5.9.4.12.2, “Parse Cycle Limit”</a></p> <p>Note: If Parser is not invoked this bit is never set.</p> <p>Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set</p>
25	ISP	Status	—	<p>Invalid Soft Parser instruction Error</p> <p>0 - No Invalid Soft Parser instruction Error 1 - Invalid Soft Parser instruction Error</p> <p>See <a href="#">Section 5.9.4.12.1, “Invalid Soft Parser Instruction.”</a></p> <p>This bit is set if a software configuration error is detected.</p> <p>Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set</p>
26	PHE	Status	—	<p>Header Error</p> <p>0 - No header error detected 1 - Header error is detected during parsing</p> <p>See parser result for details on the error.</p> <p><b>Note:</b> If Parser is not invoked this bit is never set.</p>

**Table 5-27. Rx FD Status Field Description (continued)**

Bits	Name	Command/ Status	Per-Packet Default Value	Description
27	FRDR	Status	—	<p>Frame Drop</p> <p>0 - The corresponding port is enabled in the parser</p> <p>1 - The corresponding port is disabled in the parser.</p> <p><b>Note:</b> If parser is not invoked this bit is never set</p> <p><b>Note:</b> If Parser is not invoked this bit is never set.</p> <p>Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set</p>
28	BLE	Status	—	<p>Block limit is exceeded</p> <p>0 - Block limit is not exceeded</p> <p>1 - Block limit is exceeded: The end of the first internal buffer (256 bytes) is reached by the parser before completing the parsing. The parser does not cross the first internal buffer for parsing. The first internal buffer may contain some margin before the actual data starts. It is up to the user to configure the FMBM_RIM[FOF] value to leave enough space in the first buffer for the frame headers to be parsed.</p> <p>Note: If parser is not invoked this bit is never set Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set</p>
29	L4CV	Status	—	<p>L4 Checksum Validation</p> <p>0 - No L4 checksum validation is performed on this frame</p> <p>1 - L4 checksum validation is performed on this frame. The result (pass fail is found in the parse result)</p> <p>If a frame does not fit in the first internal buffer in the FMan internal memory, and the frame is padded, the FMan is not able to validate L4 checksum.</p> <p>Note: If Parser is not invoked this bit is never set.</p>
30-31	—	—	—	Reserved
31	IPR	Status	—	if IP acceleration is enabled see description in bits 10-11

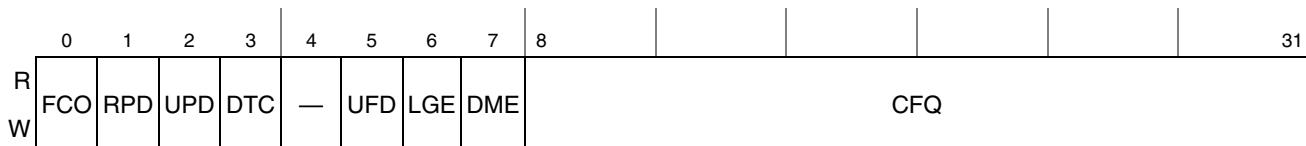
## Tx Frame Descriptor Command/Status Word

This figure describes the command bits in the frame descriptor in case of Tx. Reserved bits should be zero..



**Figure 5-20. Tx FD Command**

This figure describes the FD status bits when the frame is enqueued back to the user after transmit. This happens in Tx confirmation mode.



**Figure 5-21. Tx FD Status (Confirmation)**

**Table 5-28. Tx FD Command/Status Field Description**

<b>Bits</b>	<b>Name</b>	<b>Command/ Status</b>	<b>Description</b>
0	FCO	Command	Frame queue Context Override. This bit impacts the Tx Confirmation queuing decision. For detailed description see <a href="#">Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision.”</a>
		Status	Sticky from command
1	RPD	Command	Read prepended data. This bit enables reading a block of data from the first buffer of the frame to the Internal Context (IC) of the frame. These bytes are not part of the frame itself; they are prepended to the actual frame by the user for processing by the FMan. See <a href="#">Section 5.5.4.6.6, “Tx Internal Context Parameters Register (FMBM_TICP)”</a> for more details. 0 - Disable copying prepended block of data to the frame IC 1 - Enable copying prepended block of data to the frame IC <b>Note:</b> RPD must be cleared if FMBM_RIM[FOF] or FMBM_OIM[FOF] !=0.
		Status	Sticky from command
		Command	Update prepended data. This bit enables writing (updating) a block of data from the Internal Context (IC) to the first buffer of the frame in external memory. These bytes are not part of the frame itself; they are prepended to the actual frame by the FMan. See <a href="#">Section 5.5.4.6.6, “Tx Internal Context Parameters Register (FMBM_TICP)”</a> for more details. 0 - Disable copying IC to external buffer 1 - Enable copying IC of data to external buffer
2	UPD	Status	Sticky from command
		Command	Do IP/TCP/UDP Checksum This bit enables the calculation of TCP/UDP checksum on frames before transmission. The engine calculating the IP/TCP/UDP checksum uses the offset to the frame headers which are placed by the user in the IC prepended to the frame; therefore TX_FD[RPD] must be set for TCP/UDP checksum to operate. See <a href="#">Section 5.9.4.10, “Parse Result”</a> for details on the location of the offsets. 0 - Disable TCP/UDP checksum calculation 1 - Enable TCP/UDP checksum calculation <b>Note:</b> FMan can't support checksum update on transmit when L4 offset >= 0xf0.
		Status	Sticky from command
4	—		Reserved
5	UFD	Command	This bit must be reset by the user
		Status	Unsupported Frame Descriptor format 0 - FD format is supported by the FMan 1 - FD format bits carry unsupported values This bit is set by the BMI. See <a href="#">Section 5.5.6.20.5, “Unsupported Frames.”</a>
6	LGE	Command	This bit must be reset by the user
		Status	Length error: The frame is too long (Tx, Offline port). 0 - The frame size is not too long 1 - The frame size is larger than the maximum supported frame size (see LENGTH field in <a href="#">Table 5-26</a> )

**Table 5-28. Tx FD Command/Status Field Description (continued)**

Bits	Name	Command/ Status	Description
7	DME	Command	This bit must be reset by the user
		Status	DMA error. 0 - No DMA error occurred on this frame 1 - DMA error occurred when accessing external memory. See <a href="#">Section 5.5.6.20.7, “DMA Error”</a> Counter FMBM_TFDC is incremented when this bit is set on a DMA read transaction. In write transactions, if DME is set, there is not count.
8–31	CFQ		Confirmation Frame Queue. This FQID may be used to enqueue the frame for Tx confirmation. For detailed description see <a href="#">Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision.”</a>

**Offline Port Frame Descriptor Command/Status**

This figure specifies the bits in the FD when the FD is being dequeued for offline port. Bit 0 value should be cleared. Bits 5–31 apply default starting value to the offline parsing status. Reserved bits should be zero.

0	1	2	3	4	5	6	7	11	12	13	14	15
R	—	RPD	—	DCL4C	—	—	—	—	DIS	—		
W												
16	17	18	19	20	21	22	23	24	25	26	27	28
R	EOF	NSS	KSO	—	FCL	IPP	FLM	PTE	ISP	PHE	FRDR	BLE
W												L4CV
29	30	31										

**Figure 5-22. Offline Port FD Command**

This figure specifies the FD status after offline port processing. Reserved bits have the same value as they have in [Figure 5-22](#).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	—	RPD	—	DCL4C	—	UFD	LGE	DME	NFE	IPRE	—	DIS	—		
W															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOF	NSS	KSO	—	FCL	IPP	FLM	PTE	ISP	PHE	FRDR	BLE	L4CV	—	IPR
W															

**Figure 5-23. Offline Port FD Status**

**Table 5-29. Offline Port FD Command/Status Field Description**

<b>Bits</b>	<b>Name</b>	<b>Command/ Status</b>	<b>Per packet Default Value</b>	<b>Description</b>
0	—	—	—	This bit must be reset by the user.
1	RPD	Command	—	Read prepended data. The description is in <a href="#">Table 5-28</a> . The reference register is FMBM_OICP. See <a href="#">Section 5.5.4.7.4, “O/H Internal Context Parameters Register (FMBM_OICP)”</a>
		Status	—	Sticky from command
2	—	—	—	This bit must be reset by the user
		Status	—	Sticky from command
3	DCL4C	Command	—	Enable validation of L4 (IP/UDP/TCP) Checksum 0 - Enable L4 (IP/UDP/TCP) checksum validation 1 - Disable L4 (IP/UDP/TCP) checksum validation This bit is normally not modified by the parser. It is possible to modify it with the soft parser. This bit is used by the FMan when the FMan’s offline port dequeues the frame. If using the “fetch frame header and execute” method, described in <a href="#">Section 5.5.6.6, “Conditions that enable Offline checksum validation,”</a> then the engine calculating the IP/TCP/UDP checksum uses the running sum which is placed by the user in the IC prepended to the frame; therefore FD[RPD] must be set for TCP/UDP checksum to operate. See <a href="#">Section 5.9.4.10, “Parse Result”</a> for details on the location of the running sum.
		Status	—	Sticky from command
		—	—	Reserved
5	UFD	Command	—	This bit must be reset by the user
		Status	—	Unsupported Frame Descriptor format The description is in <a href="#">Table 5-28</a>
6	LGE	Command	—	This bit must be reset by the user
		Status	—	Length error: The frame is too long 0 - The frame size is not too long 1 - The frame is larger than or larger than 65535 bytes or larger than the FIFO (see <a href="#">Table 5-45</a> , <a href="#">Section 5.5.6.20.5, “Unsupported Frames”</a> )
7	DME	Command	—	This bit must be reset by the user
		Status	—	DMA error. 0 - No DMA error occurred on this frame 1 - DMA error occurred when accessing external memory. See <a href="#">Section 5.5.6.20.7, “DMA Error”</a> . FMBM_OFDC is incremented when this bit is set on a DMA read transaction. On O/H in write transactions, if DME is set, there is not count.

**Table 5-29. Offline Port FD Command/Status Field Description (continued)**

Bits	Name	Command/ Status	Per packet Default Value	Description																			
9	—	Command	—	This bit must be reset by the user																			
	NFE	Status	—	Non FMan error. (Error which is asserted by a non-FMan DPAA module) 0 - No error 1 -Error This bit is used to report an error from the SEC, which is reported to the CPU through the FMan.																			
8	—	Status	—	Reserved																			
10-11	IPRE or Reserved	Command	—	Reserved																			
		Status	—	<p><b>Table 5-30. If IP acceleration is enabled: {IPR,IPRE} coding</b></p> <table border="1"> <thead> <tr> <th>value</th><th>Description</th><th></th></tr> </thead> <tbody> <tr> <td>000</td><td>No error</td><td>Frame is not a result of reassembly</td></tr> <tr> <td>001-011</td><td></td><td>Reserved.</td></tr> <tr> <td>100</td><td>No error</td><td>Frame is a result of IP reassembly</td></tr> <tr> <td>101</td><td>NCSP</td><td>In FMan_v3: Non Consistent SP 1 - opening fragment on the line has a different SPID than the first fragment (the fragment with offset=0).</td></tr> <tr> <td>110</td><td>Reassembly error</td><td>IP reassembly error</td></tr> <tr> <td>111</td><td>Timeout error</td><td>IP Reassembly timeout error.</td></tr> </tbody> </table>	value	Description		000	No error	Frame is not a result of reassembly	001-011		Reserved.	100	No error	Frame is a result of IP reassembly	101	NCSP	In FMan_v3: Non Consistent SP 1 - opening fragment on the line has a different SPID than the first fragment (the fragment with offset=0).	110	Reassembly error	IP reassembly error	111
value	Description																						
000	No error	Frame is not a result of reassembly																					
001-011		Reserved.																					
100	No error	Frame is a result of IP reassembly																					
101	NCSP	In FMan_v3: Non Consistent SP 1 - opening fragment on the line has a different SPID than the first fragment (the fragment with offset=0).																					
110	Reassembly error	IP reassembly error																					
111	Timeout error	IP Reassembly timeout error.																					
12-13	—	—	—	Reserved																			

**Table 5-29. Offline Port FD Command/Status Field Description (continued)**

<b>Bits</b>	<b>Name</b>	<b>Command/ Status</b>	<b>Per packet Default Value</b>	<b>Description</b>
14	DIS	Command	—	This bit must be reset by the user
		Status	—	Discard The description is in <a href="#">Table 5-27</a> Counter FMBM_OFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set.
15	—	—	—	Reserved.
16	EOF	Command	—	This bit must be reset by the user
		Status	—	Extract Out of Frame Error The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set
17	NSS	Command	—	This bit must be reset by the user
		Status	—	No Scheme Selection The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set
18	KSO	Command	—	This bit must be reset by the user
		Status	—	Key Size Over flow Error The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set
19	—	—	—	Reserved
20–21	FCL	Command	—	This bit must be reset by the user
		Status	FMBM_RFC A[COLOR]	Frame color as determined by the Policer The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set
22	IPP	Command	—	This bit must be reset by the user
		Status	—	Illegal Policer Profile error. The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set
23	FLM	Command	—	This bit must be reset by the user
		Status	—	Frame Length Mismatch The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set

**Table 5-29. Offline Port FD Command/Status Field Description (continued)**

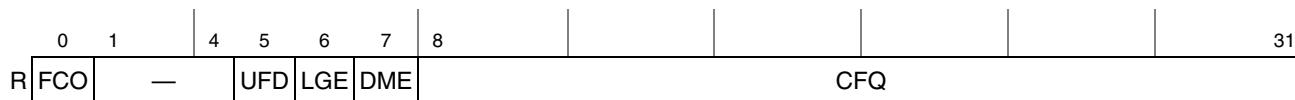
Bits	Name	Command/ Status	Per packet Default Value	Description
24	PTE	Command	—	This bit must be reset by the user
		Status	—	Parser Time out Exceed Error The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set
25	ISP	Command	—	This bit must be reset by the user
		Status	—	Invalid Soft Parser instruction Error The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set
26	PHE	Command	—	This bit must be reset by the user
		Status	—	Header Error The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set
27	FRDR	Command	—	This bit must be reset by the user
		Status	—	Frame Drop The description is in <a href="#">Table 5-27</a> Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set
28	BLE	Command	—	This bit must be reset by the user
		Status	—	Block limit is exceeded Counter FMBM_RFFC is incremented when this bit is set and the corresponding bit in FMBM_RSDM is also set The description is in <a href="#">Table 5-27</a> . The register is FMBM_OIM[FOF].
29	L4CV	Command	—	This bit must be reset by the user
		Status	—	L4 Checksum Validation The description is in <a href="#">Table 5-27</a>
30	—	Status	—	Reserved
31	IPR	—	—	Reserved if IP acceleration is enabled see description in bits 10-11

**Host Command Frame Descriptor Command/Status Word**

This figure describes the bit in the command of the FD of the host command. Reserved bits should be zero.

**Figure 5-24. Host Command FD Command**

This figure specifies the bits of the FD status of the host command after the host command has been executed.



**Figure 5-25. Host Command FD Status**

**Table 5-31. Host Command FD Command/Status Field Description**

Bits	Name	Command/ Status	Initialized by	Description
0	FCO	Command	—	Frame queue Context Override. This bit impacts the Host Command Confirmation queuing decision. See <a href="#">Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision”</a>
		Status	—	—
1-4	—	—	—	Reserved
5	UFD	Command	—	This bit must be reset by the user
		Status	—	Unsupported Frame Descriptor format The description is in <a href="#">Table 5-28</a>
6	LGE	Command	—	This bit must be reset by the user
		Status	—	Length error: The frame is too long. 0 - The frame size is not too long 1 - The BMI action code is ‘Fetch frame and execute’ and the frame is larger than 65535 bytes or larger than the FIFO (see <a href="#">Table 5-45, Section 5.5.6.20.5, “Unsupported Frames”</a> )
7	DME	Command	—	This bit must be reset by the user
		Status	—	DMA error. 0 - No DMA error occurred on this frame 1 - DMA error occurred when accessing external memory. See <a href="#">Section 5.5.6.20.7, “DMA Error.”</a> Counter FMBM_OFDC is incremented when this bit is set.
8-31	CFQ		—	Confirmation Frame Queue. This FQID may be used to enqueue the frame for Host Command confirmation.

### **5.4.3.3 Internal Context Action Descriptor (ICAD)**

The Internal Context Action Descriptor (ICAD) describes the action needed to be taken at the end of execution. [Table 5-32](#) describes the structure of the ICAD.

In the receive (Rx) and O/H flows, fields in the ICAD are updated by the various FMan modules as the packet traverses them. The user configures the modules (such as the KeyGen and Custom Classifier) to update the values that determine the queue (FQID) and the Policer profile (PNUM) to be used for this packet. Some of the fields in the ICAD are updated by the QMI during the frame dequeue operation (Tx or O/H ports). These fields are extracted from the Frame Queue Descriptor (FQD) Context A and Context

B fields (see [Section 3.3.1.4, “Frame Queue Descriptors \(FQDs\)”](#)). The FMan modules interpret the values and act accordingly. The specific interpretation is described in the relevant module sections (for example, the FMan Controller).

In the transmit (Tx) direction, the QMI may update the FQID field with FD[CFQ] or with the value extracted from Context B of the Frame Queue Descriptor (FQD) (see [Section 3.3.12, “Frame Queue Descriptor Cache”](#)). See complete description in [Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision.”](#)

#### 5.4.3.3.1 Internal Context Action Descriptor (ICAD) for Rx

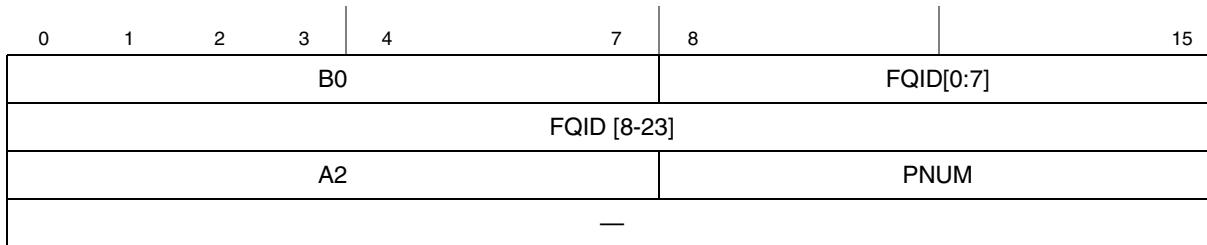


Figure 5-26. Internal Context Action Descriptor (ICAD) for Rx

Table 5-32. ICAD for Rx Fields

Bits	Name	Description	Initial value	Supported in FMan version
0-7	B0	Updated by classification stages. Bit description are the same as in <a href="#">Section 5.5.4.5.15, “Rx Frame Queue ID Register (FMBM_RFQID)”</a> .	See <a href="#">Section 5.5.4.5.15, “Rx Frame Queue ID Register (FMBM_RFQID)”</a> .	Supported on FMan_v3
8-31	FQID	FQID for enqueue at the end of processing. Updated by classification stages.	FMBM_RFQID register	Supported on all FMan versions
32-39	A2	Updated by classification stages. Bit description for this field (in Rx) are the same as “ <a href="#">Context A—A2 Field Description for Offline Port</a> ”	See <a href="#">Section 5.5.4.5.13, “Rx Policer Profile Register (FMBM_RPP)”</a>	Supported on FMan_v3
40-47	PNUM	Policer Profile Number. This is the ID of the policer profile to be used for policing this frame.	For Rx: FMBM_RPP register	Supported on all FMan versions.
48-63	—	Reserved	—	—

#### 5.4.3.3.2 Internal Context Action Descriptor (ICAD) for Tx or Offline Port

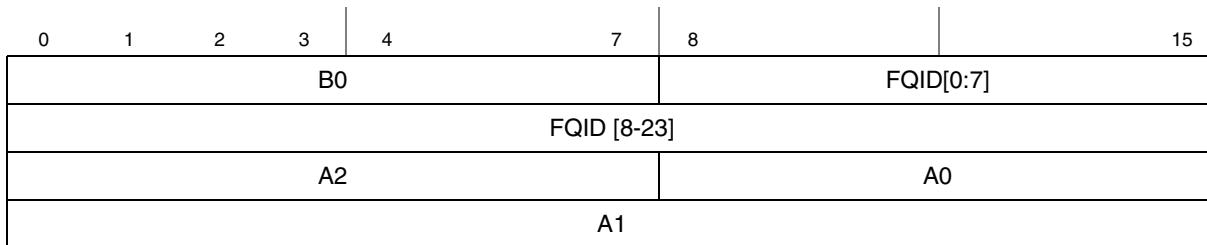


Figure 5-27. Internal Context Action Descriptor (ICAD) for Tx or Offline

**Table 5-33. ICAD for Tx or Offline Fields**

<b>Bits</b>	<b>Name</b>	<b>Description</b>	<b>Initial value</b>	<b>Supported in FMan version</b>
0-7	B0	Updated during frame dequeue with Context B field B0. For Tx see <a href="#">Section 5.4.3.1.2, “Frame Queue Descriptor Context B”</a>	FMBM_TCFQID[0-7] or FMBM_OFQID[0-7] register	In FMan_v3 Context A field A0 is placed in bits 40-47.
8-31	FQID	FQID for enqueue at the end of processing (all port types). See <a href="#">Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision”</a> or <a href="#">Section 5.5.5.8, “BMI Host Command Flow.”</a>	FMBM_TCFQID or FMBM_OFQID register	Supported on all FMan versions
32-39	A2	If Context A [A2V] is set, the value is copied during frame dequeue from Context A, A2 field. For Tx see <a href="#">5.4.3.1.1, “Frame Queue Descriptor (FQD) Context A”</a> For Offline see “ <a href="#">Context A—A2 Field Description for Offline Port</a> .”	FMBM_TFNE[0-7] or FMBM_OPP[0-7] register	Supported on FMan_v3
40-47	A0	If Context A [A0V] is set, the value is copied from Context A, A0 field. See <a href="#">Section 5.4.3.1, “Override IC from FQD or Data Buffer.”</a> See <a href="#">Section 5.4.3.1.1, “Frame Queue Descriptor (FQD) Context A”</a>	FMBM_TFCA[16-23] FMBM_OPP[8-15]	Supported on FMan_v3
48-63	A1	If Context A [A1V] is set, the value is copied during frame dequeue from Context A, A1 field.	Zero	Supported on FMan_v3

#### 5.4.4 Next Invoked Action (NIA)

Processing flow in the FMan often requires different processing modules to perform different tasks. Whichever module finishes its task on a certain flow should decide which module continues with the task and what the next step is. Modules communicate via the next invoked action (NIA); when one module finishes its task, it uses the NIA to determine what the next module should do. The NIA is 24 bits wide. See each module’s section for the usage of the most significant byte of the associated registers (note that most modules do not use this byte).

Bits that are not described in this section are described in [Table 5-35, “NIA Codes Cross References.”](#)

## Frame Manager (FMan)

Module Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																		
General NIA Format	Not part of NIA	ORR	ENG				Action Code																																											
FMan Controller		ORR	0	0	0	0	0	FMan Controller Entry Point																																										
Parser		ORR	1	0	0	0	1	Reserved						HXS ID																																				
KeyGen		ORR	1	0	0	1	0	Reserved						CCEN	SS	Reserved	SCHEME																																	
Policer		ORR	1	0	0	1	1	Res	PMO	Reserved						PNUM																																		
BMI		ORR	1	0	1	0	0	Action Codes																																										
QMI Enqueue		ORR	1	0	1	0	1	All zero																																										
QMI Dequeue		ORR	1	0	1	1	0	All zero																																										

**Figure 5-28. FMan Next Invoked Action (NIA) Structure**

**Table 5-34. NIA Field Descriptions**

Bits	Name	Description
0–7	—	These bits are not part of the NIA. In some NIA registers, these bits have other functionality.
8	ORR	Order restoration required 0 No order restoration is required for next dispatch of this task to next module. 1 Order restoration is required for next dispatch of this task to the next module.
9–13	ENG	Module (engine) code 00000 FMan controller 10001 Parser 10010 KeyGen 10011 Policer 10100 BMI 10101 QMI enqueue 10110 QMI dequeue All others reserved
14–31	AC	Action code This field defines the operation that module should perform. The decoding of action is module-specific. For detailed description of this field, see corresponding module. See <a href="#">Table 5-35</a> for cross references.

This table contains links to the AC encoder for FMan module.

**Table 5-35. NIA Codes Cross References**

Module	Cross Reference
FMan Controller	<a href="#">Section 5.12.3, “FMan Controller NIA—Action Codes”</a>
Parser	<a href="#">Section 5.9.4.4.1, “Possible Parser NIA Action Codes”</a>
KeyGen	<a href="#">Section 5.10.4.9, “Next Invoke Action (NIA) Generation”</a>
Policer	<a href="#">Section 5.11.6.4.1, “Next Invoked Action (NIA)”</a>

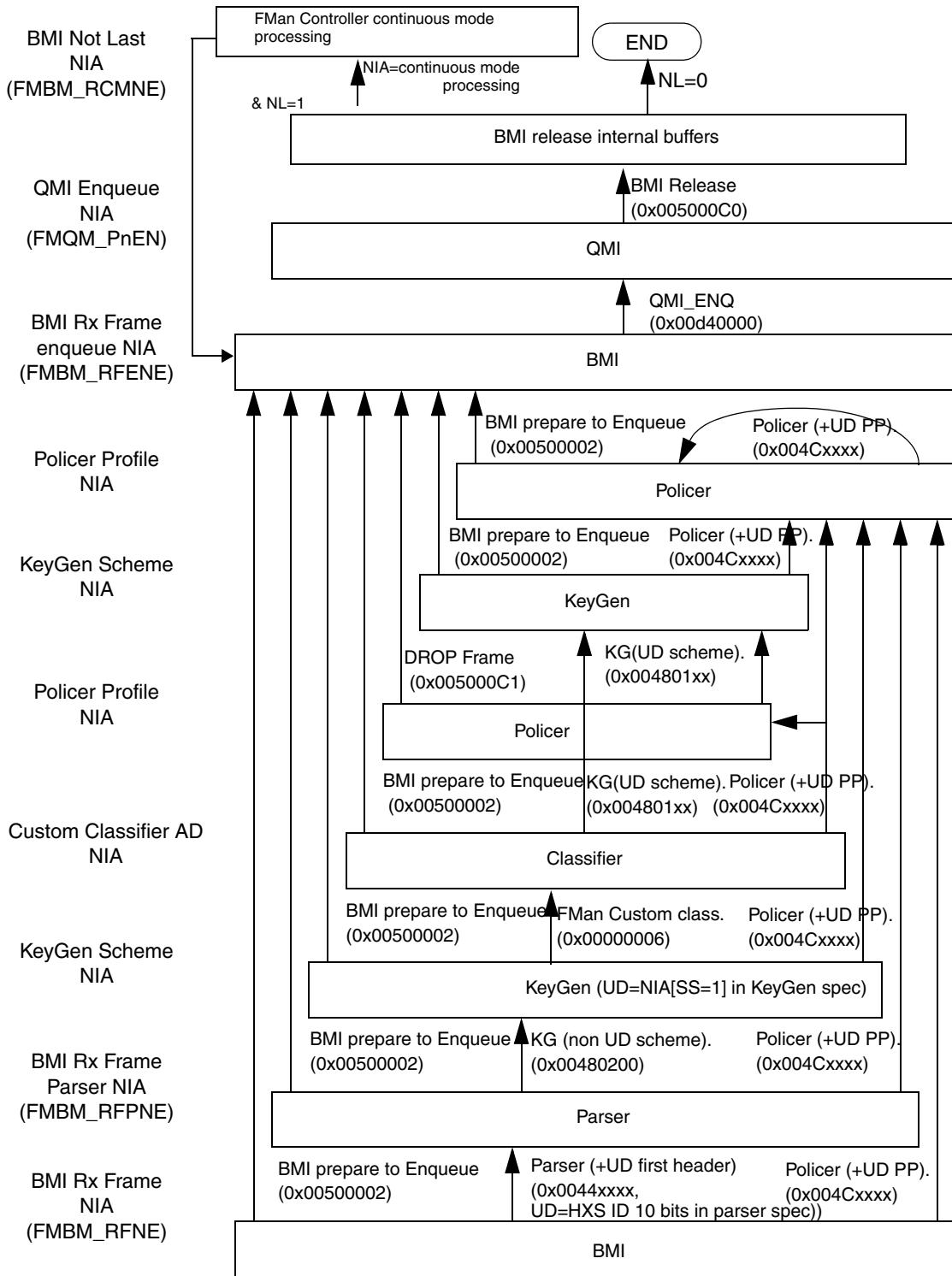
**Table 5-35. NIA Codes Cross References (continued)**

Module	Cross Reference
BMI	<a href="#">Section 5.5.3, “BMI Input NIA”</a>
QMI	<a href="#">Section 5.6.2, “QMI Input NIA”</a>

#### 5.4.4.1 NIA Register Configuration—Rx Flows

Figure 5-29 describes the FMan NIA registers and their possible configurations according to the selected Rx flow (below each register).

## Frame Manager (FMan)



**Note:** “UD” means user-defined

**Figure 5-29. Rx Flows**

## 5.4.5 FMan Debug

### 5.4.5.1 FMan Debug Overview

The actual FMan debug logic is implemented within each specific FMan module and is more fully described in the debug section of each module's section.

This section provides a high-level view for each of the different debug options and suggested debug flows.

Although the goal of the FMan's trace option is to allow for minimally intrusive trace capability, performance may degrade noticeably if verbose tracing is enabled for a high number of debug flows. Each FMan module typically offers three levels of verbosity so that the best option may be chosen depending on the desired balance of application performance to trace data detail.

### 5.4.5.2 FMan Debug Features Summary

The FMan contains an internal hardware infrastructure to support a frame-based debug capability that supports the following:

- External frame marking options for QMan
- FMan port flow capture based on marked/trapped frames for deep packet flow inspection
- Internal trace dump to frame buffer (external system memory) or/and the SoC's Nexus facilities
- Each FMan module supports three debug register sets to track three independent flows
- Debug-time window options can be controlled by an external signal connected to the FMan
- Flow capture notification for each FMan module
- External trigger for resumable FMan Halt
- Halt acknowledge
- Key indication exposure for debug
- Trap counter for each of three flows in each FMan module
- Performance monitoring

### 5.4.5.3 FMan Debug Glossary

This table describes important FMan debug terms.

**Table 5-36. FMan Debug Terms**

Term	Definition
Debug flow	A collection of properties that are used to identify flows of interest for the purposes of debugging or performance monitoring. Depending on the implementation, the FMan may support one or multiple debug flows, which are labeled "A", "B", "C", and so on.
Flow	Defined as a set of IP packets passing an observation point in a certain time interval that all have a set of common properties, such as header fields or other packet characteristics.
Trap collaboration	In the case where an implementation of FMan supports more than one debug flow, trap collaboration allows one debug flow to donate its traps to another debug flow in a chained fashion, allowing for more trap conditions to be available to a given debug flow.

**Table 5-36. FMan Debug Terms (continued)**

Term	Definition
Debug Trap	A single criterion that contributes to the definition of a debug flow. Multiple traps can be combined in Boolean combinations to form complex flow patterns. Each debug trap consists of a comparison operator, a Boolean operator (AND/OR), and a field selector (FSEL). The FSEL value is an encoding that allows the programmer to select which properties of a packet that will be used in the comparison. The options indicated by the FSEL value are specific to each FMan module. The comparison values may also be masked, thus allowing for further modification of the desired trap criteria.
Debug flow bypass	A debug flow may be put into a bypass state, meaning that a packet can be ignored for debug processing by the current FMan module, but it remains in a debug state for all downstream FMan modules.

### 5.4.5.4 FMan Packet Debug

During packet processing, FMan can trace packet processing flow through each of the FMan modules and trap a packet. Each of the FMan's execution modules that take part in packet processing can append debug information to the frame's IC. This information may be written to the external memory (via DMA) if programmed to do so, or it can be written out through the Nexus trace interface. See the FMan module sections for debug data collection options and trace format definitions.

#### 5.4.5.4.1 Flow Debug

All FMan flows are initialized at the BMI, so any corresponding debug flows (A, B, and C) are initiated there as well. Each FMan port can be associated with up to three debug flows (A, B, and C) independent of the other ports. The description below applies to any one of the three debug flows.

When a port has been associated with a debug flow, the BMI signals the next processing module during task dispatch, which does one of the following based on its internal debug configuration:

- Attempts to trap the frame by evaluating the configured criteria. A successful trap causes it to write its debug context and propagate the debug signal (continue debug for this flow) to the next processing module during task dispatch. If a frame is detected by traps related to different debug flows (A, B, or C) within the same module, the most verbose trace is applied. For example, if debug flow A is configured to dump debug data in terse mode and debug flow B is configured for verbose mode, and the criteria for both flows are met, then the verbose trace format is chosen. Failing to trap causes the tracing of that packet within all subsequent modules to stop.
- Traps in bypass state, which acts as a successful trap. This allows the packet to be ignored by the current module, but remain a packet of interest for all downstream FMan modules.

When frame processing reaches the BMI, any remaining debug flow indications cause debug data to be written to the frame's IC. The following actions could be selected for each flow when its criteria is met:

- Write debug trace to the following:
  - External memory
  - SoC's Nexus facilities
- Halt the trapped task in its current state of processing within the module chain.
- Halt the frame port associated with the trapped flow.
- Halt all FMan ports.

- If the flow of interest matches more than one debug flow (any combination of A, B, or C), the BMI would act according to the strongest necessary action (the strongest action halts all FMan ports, the intermediate action halts just the frame port associated with the trapped flow, and the weakest action halts the trapped task).

When a trapped frame is enqueued to the QMan, it is marked to one of the QMan codes according to what was programmed at QMI. If this frame was trapped by multiple debug flows, it will be marked according to the priority defined at QMI.

To activate the FMan debug system, users can choose between setting the global enable bit in BMI or using the global debug enable signals connected to the SoC's Nexus facilities. The debugger may experience additional debug traces after closing the time window for the remaining frames being processed.

For SoC-level performance monitoring and event cross-triggering purposes, a statistics counter is associated with each debug flow in each module, which counts how many times the flow debug indications were not negated while the frames are processed by that module. This indicates the number of times there was successful trap event for that debug flow, including bypass.

#### 5.4.5.4.2 Trace

An FMan module appends debug data to the debug section of the IC only if debug was requested during task dispatch and if the frame has been trapped (for this action, trap bypass is considered a successful trap). The FMan writes trace information that is relevant to the active FMan module into the debug section of the internal context that is stored (see [Table 5-19, “Internal Context \(IC\)”](#)). The trace is controlled by TL\_X (trace level for flow X, where X = A, B, or C) dedicated for each of the three independent debug flows. This value selects the level of detail of trace information that is written to the debug section of the frame's IC when trace information is generated. If the frame is trapped by more than one debug flow, the most verbose trace of the trapped debug flows is applied. During frame dispatch, the current debug offset is provided. This value is updated according to the amount of trace that was written by the current module, rounded up to 4 bytes, to prepare for possible trace data being added by the next module in the frame flow.

Each module that adds trace data uses the first 4 bytes as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
VL	SIZE																												NIA		

**Figure 5-30. Trace First 4 Bytes**

**Table 5-37. Trace NIA Field Descriptions**

Bits	Name	Description
0–1	VL	Verbosity level See <a href="#">Section 5.4.5.5.1, “Generic Debug Control Register Model.”</a>

**Table 5-37. Trace NIA Field Descriptions (continued)**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
2–7	SIZE	Size of the trace dump for that module The next dump starts after the end of the current dump at the minimal 4-byte aligned offset. Size is specified in bytes except the FMan controller where it is specified in words, which are 4 bytes.
8–31	NIA	Action code received by the dumping module This NIA specifies the dumping module and the action that was done on the frame. <ul style="list-style-type: none"> <li>• On Rx: The first NIA is BMI generic NIA (0x500000). The last NIA is the BMI prepare to enqueue NIA.</li> <li>• On Tx: The first NIA is the dequeue NIA. The last NIA is the BMI transmit NIA.</li> <li>• On offline: The first NIA is the dequeue NIA. The last NIA is the BMI prepare to enqueue NIA.</li> </ul>

**NOTE**

After these first 4 bytes, it is up to each module to append trace data that is relevant to the module's specific function.

**5.4.5.4.3 Debug Traps**

Each module has three programmable sets of debug traps (A, B, and C), each of which is dedicated for one of three possible debug flows. The debug traps are discrete criteria for a specific debug flow are related to each other through boolean operations, thereby forming a debug event for the debug flow. If a debug event occurs, the next module is signaled during task dispatch that the current frame is still in a debug state. Users may choose to configure a trap to be in bypass mode, meaning that the specific trap operation would be considered successful for any frame. A bypass acts as a logical true in an OR operation, and allows traps to be defined, yet logically ignored when necessary. Each module implements as many traps as deemed sufficient for that module.

Each FMan module notifies the SoC's Nexus facilities about trap events for each of the three debug flows with dedicated output signals. Upon a trap event, a pulse for three clock cycles is fired towards the SoC's Nexus facilities. Each FMan module counts trap events for each of the three debug flows.

**5.4.5.5 Debug Programming Model****NOTE**

The register descriptions in this section describe the register architecture used by each FMan module, and do not represent actual registers. See the debug section of each module for the definition of the actual registers and their respective offsets in the memory map.

**5.4.5.5.1 Generic Debug Control Register Model**

The debug control register defines the verbosity level of each of the three debug flow patterns as well as the interactions amongst those patterns (A, B, and C) for the relevant module. See the debug trace

configuration registers in each module for the instantiations of this type of register. Each module may implement a subset of the four levels of verbosity.

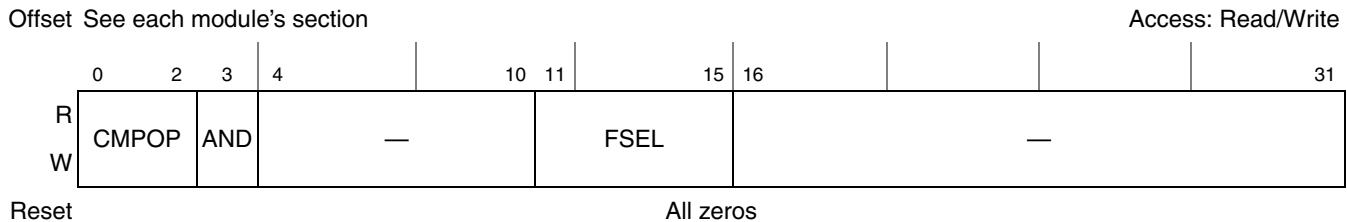
**Figure 5-31. Generic Debug Trace Configuration Register Model**

**Table 5-38. Generic Debug Trace Configuration Model Field Descriptions**

Bits	Name	Description
0–1	—	Reserved.
2–3	TL_A	<p>Trace level flow A.</p> <p>Selects the level of trace information that is dumped to the debug area</p> <ul style="list-style-type: none"> <li>00 Trace disable</li> <li>01 Minimum trace</li> <li>10 Verbose trace</li> <li>11 Very verbose trace</li> </ul>
4–5	TL_B	<p>Trace level flow B</p> <p>Selects the level of trace information that is dumped to the debug area</p> <ul style="list-style-type: none"> <li>00 Trace disable</li> <li>01 Minimum trace</li> <li>10 Verbose trace</li> <li>11 Very verbose trace</li> </ul>
6–7	TL_C	<p>Trace level flow C</p> <p>Selects the level of trace information that is dumped to the debug area</p> <ul style="list-style-type: none"> <li>00 Trace disable</li> <li>01 Minimum trace</li> <li>10 Verbose trace</li> <li>11 Very verbose trace</li> </ul>
8–11	TR_CO	<p>To provide a larger number of traps for debug flow A, the traps that belong to debug flows B and/or C can be chained as a continuation of the traps for debug flow A. This lowers the number of active debug flows, but allows the criteria for debug flow A to be more extensive. The chained trap set would act as one trap entity with a single boolean result. The debug flows (B and/or C) that donated their traps to debug flow A are put into bypass since they would no longer have the resources to define match criteria. If debug flows B or C are not participants in the collaboration chain, then their trapping behavior will remain active and operate independently as configured.</p> <ul style="list-style-type: none"> <li>0000 No collaboration between trap registers</li> <li>0001 Debug flow A's traps are chained to debug flow B's traps. Debug flow B is in bypass.</li> <li>0010 Debug flow A's traps are chained to debug flow C's traps. Debug flow C is in bypass</li> <li>0011 Debug flow A's traps are chained to debug flow B's traps, which are then chained to debug flow C's traps. Debug flows B and C are in bypass.</li> </ul>
12–31	—	Reserved

### 5.4.5.5.2 Generic Debug Trap Configuration Register (GDTCR) Model

The GDTCR defines how a selected value is going to be compared to a reference value. It also defines the logical operation which combines the trap result with the next debug trap if it exists. The result of each trap's comparison can be combined with its siblings to form a more complex condition, which results in the definition of a single debug event. The set of registers defining an event are instantiated three times, one for each debug flow (A, B, or C). See the debug trap collaboration registers in each module for the instantiations of this type of register.



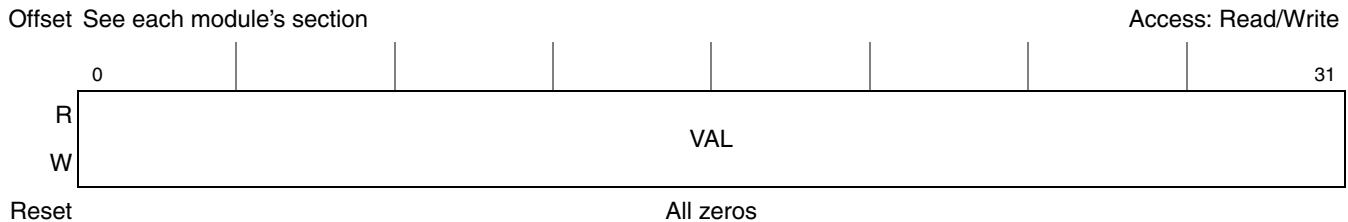
**Figure 5-32. Generic Debug Trap Configuration Register (GDTCR) Model**

**Table 5-39. GDTCR Field Descriptions**

Bits	Name	Description
0–2	CMPOP	Compare Operator 000 Always match (trap bypass) 001 Trap Disabled (never match) 010 Match if (selected field & MASK) equals Value 011 Match if (selected field & MASK) does not equal Value 100 Match if (selected field & MASK) is greater than Value 101 Match if (selected field & MASK) is less or equal to Value 110 Match if (selected field & MASK) is less than Value 111 Match if (selected field & MASK) is greater or equal to Value
3	AND	Combines trap results into a trap equation By default it is an OR; therefore, if all subsequent traps in the debug flow's trap sequence are disabled (or this is the last trap), it will have no effect on the combined match result. The combined match evaluation is done in ascending order from trap number $n$ to trap number $n + 1$ , and without precedence of AND over OR. For example: TRAP1 or TRAP2 and TRAP3 will result in equation equivalent to (TRAP1    TRAP2) && TRAP3. The sequence of traps is therefore significant when attempting to construct the proper logical equation. 0 OR between this trap result and the next trap result 1 AND between this trap result and the next trap result
4–10	—	Reserved
11–15	FSEL	Field selection Each encoding of FSEL identifies the data sources for the values that will be used in this trap's comparison operation, and each such field option may consist of a single value or a concatenation of multiple whole or partial values. The selected field data from its original sources is then compared to the value in the corresponding value register combined with the mask defined in a corresponding mask register. Fields with fewer than 32 bits are usually left padded with zeros to form a 32-bit value.
16–31	—	Reserved

### 5.4.5.5.3 Generic Debug Value Register (DVR) Model

The debug value register contains the value against which the selected field at the debug trap configuration register is compared. See the debug value registers in each module for the instantiations of this type of register.



**Figure 5-33. Generic Debug Trap Value Register (DVR) Model**

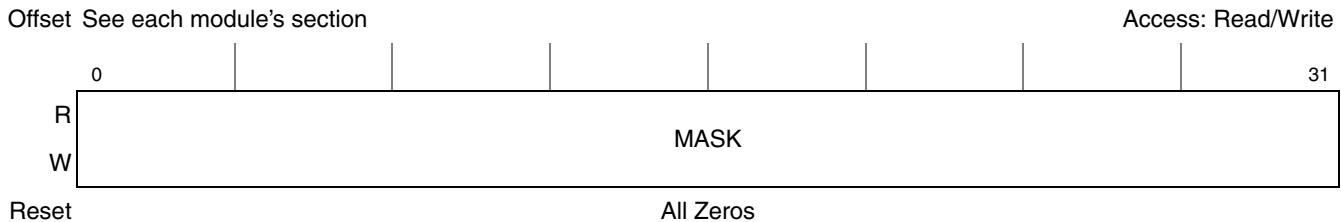
**Table 5-40. Generic Debug Trap Value Register (DVR) Model Field Descriptions**

Bits	Name	Description
0–31	VAL	Value.

### 5.4.5.4 Generic Debug Trap Mask Register Model

This mask value which has been specified is ANDed bit-wise with the selected value for comparison, and the result is compared to the debug value register. See the debug trap mask registers in each module for the instantiations of this type of register.

### 5.4.5.6 DMA Trace Write



**Figure 5-34. Generic Debug Trap Mask Register Model**

**Table 5-41. Generic Debug Trap Mask Register Model Field Descriptions**

Bits	Name	Description
0–31	MASK	MASK

The FMan DMA has a special bus connected to the SoC’s Nexus facilities. When the DMA writes the debug portion of the IC of the frame, the following debug write options can be programmed in the BMI:

- All of the data is written to the following:
  - Memory
  - Debug interface only (SoC’s Nexus facilities)

## 5.4.6 Implementing Statistics Counters

Statistics are treated consistently in all the FMan modules. The following rules describe the implementation of the statistics counters.

- Statistics counters are initialized to 0 after a reset.
- Software can write to a statistics counter to initialize its value.
- To calculate the increment of a statistics counter, software must poll the counters and calculates the increment using the previous value of the counter.
- Statistics counters count continuously and wrap around.
- Statistics counters are either 32 or 64 bits, whichever is necessary to ensure that the frequency needed by the software to poll the counter is no more than 1 second (that is, if the counter has not counted its full range within 1 second).
- For 64-bit counters, use following procedure:
  - The 64-bit counter is represented by two 32-bit registers; one representing the lower 32-bits and the other representing the upper 32 bits. Software uses the “time base” trick which requires three read accesses, which is summarized in the following pseudocode:

```
do {
    U = get_upper();
    L = get_lower();
    T = get_upper();
} while (U != T);
```

- No interrupt is required.

## 5.4.7 Error Handling

The FMan errors are divided into several classes, listed in this table.

**Table 5-42. Types of FMan Errors**

Type	Definition
Catastrophic Errors	An interrupt is raised to the SoC so that it is aware of the error and may issue a global reset
“Local Catastrophic Errors	Causes a stall in the port associated with this error. This behavior is mainly for debug recovery from local catastrophic error, requiring a reset of the FMan
“Serious Errors	May cause a stall in the port associated with this error. A recovery from such an error can occur without a reset. However, a memory leak can happen in certain error cases
“Operational Errors	Reported to the user and does not stall any interface

### 5.4.7.1 Local Catastrophic Errors

Local catastrophic errors in the FMan can occur in the following scenarios:

- FPM time-out—The frame task reached a time-out during its processing.
- Bad NIA—The NIA given to the FPM does not point to a valid module.
- Unrecoverable ECC error—An interrupt is generated; the FMan halts if configured to do so.

When a local catastrophic error is detected the FPM stalls and marks the task that caused the local catastrophic error. The port that tasks are related to are stalled. All of the tasks that belongs to the same port are prevented from entering any of the dispatch queues. The only way to recover from this situation is by FMan reset.

### 5.4.7.2 Serious Errors

Serious errors in the FMan can occur in two scenarios, which are the same as for local catastrophic errors except for the FPM error configuration, as follows:

- FPM time out—The frame task reached a time-out during its processing (can be configured).
- BadNIA—A NIA given to the frame processor manager (FPM) does not point to a valid module.

When a serious error is detected, the FPM stalls and marks the task that caused this error. The port that a task is related to may be stalled due to order restoration. The user is able to look at the information related to that task and release task processing after debug finishes. After the release of the task, everything continues as usual. Other ports are not affected in order to maintain the connection to the processor and to allow debugging to be done, even remotely.

### 5.4.7.3 Operational Errors

Operational errors include the following:

- Platform bus error: During write or read to platform bus, transaction has a data error indication.

When a frame that had a data error is enqueued to the QM, error indication is set in the Frame Descriptor (FD) status bits (set by the QMI). Each port has an error FQID (programmed in the BMI) so that there is the possibility to deliver the frame with the error to the user. Data error can behave as a local catastrophic error according to FPM configuration (this is the default state when going out of reset).

The various operational data errors are described as follows:

- Rx—On receive, the data errors can be only on writes, FQID is not changed. An error indication is set in FD[DME].
- Tx—On transmit, the data errors are on read. This prevents the transmission of the frame. Frame is forwarded to confirmation FQID with an error indication. If confirmation FQID is not specified, frame is delivered to error FQID.
- Offline port—Offline ports behave the same as Tx.
- Independent mode—In independent mode FMan should be configured so that a data error is regarded as a local catastrophic error.

### 5.4.8 Configuration Register Access

Each register is defined in a different 4-byte aligned memory region. The access size to each register should only be according to its size. Partial size accesses are not supported. Access to unimplemented/reserved do not cause data error. The value returned by reading an undefined register is undefined. When reading reserved bits it cannot be assumed that they read zero. When writing to register reserved bits must keep their default value.

## 5.4.9 Graceful Stop

Graceful stop is defined as the flow in which the hardware does not accept new frames, but finishes all frames or tasks that had already begun.

### 5.4.9.1 Tx Port Graceful Stop

To gracefully stop the Tx path the enable dequeue bit in the QMI for the port in question must be negated. The software must wait until the busy bit for that port in the QMI is negated. This is to insure that there are no FDs stored in the QMI buffer. The software then must go to the BMI and disable the transmitter. In order to verify that the transmissions have stopped, the software must wait until the busy bit in the BMI for that port is negated.

- a. The QMI graceful stop procedure is executed for each Tx:
  - Clear the enable Port ID bit in the FMMQ\_PnC register for each Tx port.
  - Wait until the QMI is not busy with FDs. This is done by polling on the PBSY\_DF bit in the FMMQ\_PnS register for each Tx port.
- b. If all the Tx and O/H port are stopped
  - Wait until BSY\_DF bit in the FMMQ\_GS register is cleared
- c. BMI graceful stop procedure:
  - Clear the FMBM\_TCFG(EN)
  - The process completes when all tasks associated with the port are terminated. This is done by polling for the BSY bit =0 in the FMBM\_TST registers.

### 5.4.9.2 Rx Port Graceful Stop

In order to gracefully stop the receiver, the Rx enable bit for the relevant port must be cleared. Then the software must wait until the busy bit for that port is negated.

## 5.4.10 Dynamic Changes

Dynamic configuration changes are allowed in general, while obeying to limitations that are specified in the specific configuration register where applicable. Specifically, the parser has a hardware protection that does not allow dynamic changes while the port is active. This is described in [Section 5.9, “Frame Manager—Parser.”](#) A special atomic method of updating is supported in the following blocks:

- The custom classifier tables can be updated dynamically with the help of host commands.
- The Policer has support for atomic change of the policer profiles.
- The KeyGen has support for atomic changes in the schemes (indirect memory space).

## 5.5 Frame Manager—Buffer Manager Interface (BMI)

### 5.5.1 Frame Manager BMI Introduction

The Buffer Manager interface (BMI) communicates with BMan, Network and other FMan modules. Its main function is to move frames from/to external memory to/from network through the FMan internal memory with possible intervention of other modules for further processing of frames.

#### 5.5.1.1 Acronyms

This table is a glossary of acronyms and abbreviations.

**Table 5-43. Glossary of Acronyms and Abbreviations**

Acronym/ Abbreviation	Meaning
AD	Action Descriptor
BMan	Buffer Manager
BMI	Buffer Manager Interface
DPAA	Data Path Acceleration Architecture
DPSS	Data Path Sub System
FD	Frame Descriptor
FMan	Frame Manager
FQ	Frame Queue
FQD	Frame Queue Descriptor
FQID	Frame Queue ID
IC	Internal Context
ICAD	Internal Context Action Descriptor
NIA	Next Invoked Action
O/H	Offline Port / Host Command
QMan	Queue Manager
SEC	Security Engine
SoC	System on Chip

### 5.5.1.2 Terminology

This table lists terms and descriptions.

**Table 5-44. Terms and Descriptions**

Term	Description
BMan	<ul style="list-style-type: none"> <li>• Buffer Manager (BMan) manages a set of buffer pools.</li> <li>• Different pools manage buffers of varying sizes and/or from different memory (internal and external) regions.</li> <li>• Data Patch Acceleration Architecture (DPAA) modules can allocate/release buffers from any pool.</li> <li>• Buffer users must have size information as it is unknown to the BMan.</li> <li>• Optimize performance by allocating at least one pool in the internal Datapath Static Random Access Memory (SRAM) if available.</li> </ul>
Buffer	<ul style="list-style-type: none"> <li>• Unit of software-allocated contiguous memory.</li> <li>• Buffer(s) hold a data element (generally a frame).</li> <li>• Frames may be stored in single or multiple buffers (scatter/gather lists).</li> <li>• A “simple” frame has one delimited data element while “compound” frames have more than one.</li> </ul>
Buffer Pool	<ul style="list-style-type: none"> <li>• List of available buffers that have the same characteristics (size, address-ability, accessibility).</li> </ul>
DPAA Infrastructure	<ul style="list-style-type: none"> <li>• Queue Manager (QMan), Buffer Manager (BMan), Frame Manager (FMan), together comprise the DPAA infrastructure. There are other accelerators components in the DPAA (such as Security Engine).</li> </ul>
External Memory	<ul style="list-style-type: none"> <li>• Memory which is external to the FMan</li> </ul>
External Buffers	<ul style="list-style-type: none"> <li>• Buffers which are allocated in External Memory</li> </ul>
FMan	<ul style="list-style-type: none"> <li>• FMan manages frames that are supplied to the FMan via two mechanisms: <ul style="list-style-type: none"> <li>– Ethernet interface, the Receiver (Rx) or,</li> <li>– Frame Queue (FQ) consumer (either for the Ethernet Tx interface or the offline port).</li> </ul> </li> <li>• Each FMan frame is handled by a single task that flows from module to module.</li> <li>• The flow is configured using Next Invoked Action(s) (NIA).</li> <li>• The internal context: <ul style="list-style-type: none"> <li>– Is used to store information associated with one frame being processed.</li> <li>– Is updated by every module processing the frame</li> <li>– Subsequent modules use the updated information when processing the frame</li> </ul> </li> </ul>
FMan Memory	<ul style="list-style-type: none"> <li>• Memory which is internal to the FMan. This memory is also called ‘internal memory’ or ‘FMan internal memory’</li> </ul>

**Table 5-44. Terms and Descriptions (continued)**

Term	Description
Frame/Packet	<ul style="list-style-type: none"><li>“Frame” and “packet” are used interchangeably in this document.</li></ul>
QMan	<ul style="list-style-type: none"><li>QMan provides queueing functionality that allows packet transfers between varying device accelerators and cores (e.g., cores, DPAA modules). QMan manages a set of FQs.<ul style="list-style-type: none"><li>“Producer” is an entity that enqueues frames into an FQ.</li><li>“Consumer” is an entity that dequeues frames from an FQ.</li><li>Each FQ can have multiple producers, but only a single consumer.</li></ul></li><li>Software and DPAA modules enqueue and dequeue using portals.</li><li>Each FQ has a set of associated context values configured by the software. They are handed to the FQ consumer as part of the dequeue process.<ul style="list-style-type: none"><li>An FQ has only a single producer enqueueing a frame to an FQ. The producer implicitly dictates a frame’s processing.</li></ul></li></ul>
SEC	<ul style="list-style-type: none"><li>SEC is a cryptographic accelerator and assurance module that implements several encryption and hashing functions, a runtime integrity checker, and a random number generator.</li><li>A set of descriptors defines the SEC work performed on a frame.</li><li>‘PreHeader’ and ‘Shared Descriptors’ are referenced by the FQ context from which the SEC module dequeues the frame. Thus, all frames going via the same FQ experience the same security processing operation.</li><li>As an output, the SEC module generates a new frame containing output data; the frame is enqueued back to an FQ.</li><li>The PreHeader contains the BMan Pool ID information and the shared descriptors determine the type of security operation.</li><li>The buffer holding the input frame is released to the BMan.</li></ul>

## 5.5.2 Frame Manager BMI Overview

The BMI module has a number of functions

- Receive (Rx) port: Transfers data between the Ethernet receive network interface and internal FMan memory
- Transmit (Tx) port: Transfers data between the internal FMan memory and the Ethernet transmit port
- Offline Port: Processes data placed by other components on the device (such as the host CPUs or the security block) on the offline port.
- Host commands: Processes commands received from the host CPUs.
- FMan data structure management: Generates frame descriptors (FDs), initializes the internal context (IC), manages the internal buffers, allocates/deallocates external buffers with the help of BMan and activates the DMA to transfer data between internal and external RAMs.

### 5.5.2.1 Frame Manager BMI Features

The BMI module supports the following features:

- General features

- Support for N \* 10Gbps MAC. See DPAA overview chapter for port definition in various devices.
- Support N\* 1Gbps MACs See DPAA overview chapter for port definition in various devices.
- Supports four types of flows: Rx, Tx, Offline Port and Host commands
- Two operational modes are supported, normal and independent.
- Supporting dynamic FIFO buffers (256 data bytes per buffer) allocation in internal FMan memory
- Total internal free buffers pool is divided between all ports
  - Configurable FIFO size per port with programmable excessive buffer size shared between all Rx ports
- Allocate/deallocate internal buffers for frame processing and temporary storage
- Manages internal data structures (free buffer pool, processed frames, and so on) in a linked list manner by using internal RAM
- Allocate/Deallocate external buffers in conjunction with BMan via hardware portal
- User defined conditions to generate pause frame on depletion status of BMan pools
- Generates statistic information about external buffers
  - Number of allocate/deallocate operations per pool per port
- Configurable priority per each MAC.
- Congestion control and rejection messages via dedicated interface with the QMan
- Hardware assist for IEEE 1588-compliant timestamping
  - Pass actual timestamp to host for received frames.
  - Configurable pass of actual timestamp of transmitted frames to host.
- Statistics counters and Performance counters per Rx, Tx and Offline Port/host command port.
- Rx features
  - Transfer received data from MACs to FMan memory
  - To avoid FIFO overflow due to lack of free internal buffers, the BMI issues pause request according to a configurable threshold per MAC.
  - Storage Profile
    - Storage profile (including buffer pool allocation) for each received frame according to Rx port and frame length.
    - Storage profile (including buffer allocation) for each received frame according to the results of classification (and frame length).
  - Creation and initialization of IC with programmable content of the first 72 bytes
  - Configurable limits for number of open DMA tasks (to avoid DMA overload from single port)
    - Committed
  - Configurable number of BMan pools with different external buffer length per port
  - Generate DMA to move the IC and data from FMan memory to external memory
  - Programmable margin for free space in external buffer before start of frame and after end of the frame

- Programmable offset and size for copying IC to external buffer
- Programmable offset in external buffer for writing the IC
- Calculate running sum (one's complement sum) for the whole frame except configurable size at the end of frame (up to 16 bytes) and pass it to parser for checksum validation in normal mode
- Programmable number of bytes chopped from the end of frame (up to 16 bytes)
- Scatter/Gather or single FD in normal mode
- Configurable NIA to be used to start frame processing
- Configurable NIA to be used when frame processing is finished
- Configurable hardware parser NIA to be written in IC
- User-defined behavior for frames with bad CRC: enqueue to error FQID or discard
- Programmable discard mask to be compared against FD status word when frame processing stage completes, and result in frame discard if match
- Programmable error mask to be compared against FD status word when frame processing stage completes, and result in frame redirection to error FQID if match
- Programmable parsing start offset from beginning of frame
- Programmable default FQID
- Congestion notification
  - Automatic internal pause upon receiving congestion notification from QMan on Rx.
  - Transmitting Priority based flow control (FPC) upon QMan congestion notification
  - Mapping of PFC class (in incoming PFC class enable vector) to QMan traffic class
- Tx features
  - Configurable update of TCP/UDP checksum and IPv4 header checksum
  - Rate limiter per MAC
    - Maximum number bits/sec per port
  - Confirmation
    - Configurable default Tx confirmation FQID
    - Confirmation on transmitted frame (enqueue FD to return FQID once frame was transmitted)
    - Notification on Error Frame to user defined FQID.
  - Configurable Tx pipeline (number of outstanding frames waiting for dequeue) up to 8 frames
  - Configurable limit for number of open DMA commands (to avoid DMA overload from single port)
  - Configurable NIA in Tx flow
  - Configurable FIFO size
  - Configurable FIFO low comfort level per port
  - Programmable offset in external buffer for copying the IC
  - Option to invoke other module after frame fetch

- Continuous mode: Option to disable release of buffers and invoke other module at the end of the flow
- Offline Port/Host Commands features
  - Option to copy frame to external memory location. Storage profile (including buffer allocation) for each frame according to the results of classification (and frame length).
  - Up to N assigned channels for Offline Port and host commands. See SoC DPAA introduction chapter for more details.
  - Rate limiter
    - Maximum number frames/sec per each Offline Port/host command channel
    - Configurable low/high rate limit when Rx or Tx ports are DMA overloaded/not busy
  - Scatter/Gather FD supported in Offline Port channels
  - Congestion notification
    - Automatic internal pause upon receiving congestion notification from QMan on Offline Port/Host Command (O/H) port dequeue
    - Priority based pause upon congestion notification
  - Configurable NIA per each channel
  - Configurable hardware parser NIA to be written in IC
  - Programmable parsing start offset from beginning of frame
  - Programmable default FQID
  - Programmable content of parser results to be written to IC for Offline Port
- Continuous mode: Option to disable release of buffers and invoke other module at the end of the flow (Rx, Tx Offline flows)

### 5.5.2.2 Frame Manager BMI Modes of Operation

The BMI supports the following modes of operation:

- Normal mode—In this mode several FMan blocks take part in the flow and frame is processed by these agents (like hardware parser, DMA, etc.). The BMI is responsible to allocate/deallocate internal and external buffers. See [Section 5.5.5, “Frame Manager BMI Functional Description,”](#) for details.
- Independent mode—In this mode, the BMI does not initiate any external buffer allocation and it does not initiate DMA operations. In Rx flow, the BMI saves incoming frame in the FMan memory and then the FMan controller is responsible to initiate requests to DMA in order to move data to external memory. In Tx flow FMan controller is responsible to fetch the frame to FMan memory and request from the BMI to send this frame to network. See [Section 5.5.5, “Frame Manager BMI Functional Description,”](#) for details.

### 5.5.3 BMI Input NIA

The BMI receives Action Codes through the Next Invoked Action (NIA) from other modules within the FMan. The user programs the NIA for the BMI in the module whose execution precedes the BMI.

The format of the NIA for the BMI is:

Module Name	0 1 2 3   4 5 6 7	8 9 10 11   12 13 14 15	16 17 18 19   20 21 22 23	24 25 26 27   28 29 30 31
General NIA Format	Not part of NIA	ORR	ENG	Action Code
BMI		ORR	1 0 1 0 0	Action Codes

See [Section 5.4.4, “Next Invoked Action \(NIA\),”](#) for the description of the NIA concept.

This table provides detailed description of the NIA action codes supported by the BMI.

**Table 5-45. BMI Input Actions Codes (NIAs)**

Action Code	Action Name	Affected Flows	Usage	BMI Actions
0x0C0	Release all internal resources	Rx, Tx in normal mode. (see note) Offline/Host Command Rx, Tx in independent mode	This action releases all the internal resources allocated for this frame.  If ICAD[NL]=1, the internal resources are not released apart from the internal buffers that are attached to the ICP.  <b>Note:</b> This action code should be used on Tx normal mode	Release all internal buffers associated with this frame. Release tnum to FPM.
0x2C0	Release all internal resources	Tx in normal mode	This action releases all the internal resources (including internal buffers) allocated for this frame.  <b>Note:</b> action code 0x0C0 should be used to release internal resources.	Release IC associated with this frame. Release tnum to FPM.
0x274	Transmit frame and release external and internal buffers or enqueue to return FQID.	Tx in normal mode	The BMI receives acknowledgment that dequeuing of frame has been completed. FD associated with this frame has been copied to FMan memory for the BMI's use.	Copy the frame to FMan memory (DMA). If the return FQID is not zero, then use TFENE for the next execution. Otherwise, release the external buffers. Internal buffers can be released anyway except IC.
0x010	Transmit frame without DMA (for FMan_v3)	Tx in normal mode	Data already exist in FMan memory and the BMI is requested to transmit data from FMan memory without releasing internal buffers and return task back at the end.	Send data from FMan memory to the MAC; when finished, if NL=1 internal buffers are not released. If FQID!=0, confirmation mode flow is followed. See <a href="#">Figure 5-6</a> for details.
0x002	Prepare to Enqueue Frame	Rx, Offline Port	The BMI receives acknowledgment that parsing and/or classification completed. Frame header and IC must be written out, FD can be enqueued.	Request DMA to copy frame's header and IC to external memory. When DMA is finished, continue execution using NIA from RFENE or OFENE.
0x0C1	Discard Frame, Termination of the task	Rx, Offline Port	The BMI receives action to discard the frame.	PatentFrame is discarded, subject to conditions described in <a href="#">Section 5.5.6.8, “Rx Normal mode and Offline - Enqueue or discard decision logic”</a> )

**Table 5-45. BMI Input Actions Codes (NIAs) (continued)**

Action Code	Action Name	Affected Flows	Usage	BMI Actions
0x20C	Fetch frame and execute	Offline Port, Host command Tx (for FMan_v3)	Fetch the frame and send to next module.	Read the frame from ext memory to FMan internal memory and continue execution using NIA from OFNE. In FMan_v3, it is possible to copy the entire frame into a new storage profile by configuring the operational mode bits accordingly. FMan_v3, used in Tx when TFNE is used to invoke an other module (typically the FMan controller) before transmission.
0x208	Fetch frame header and execute	Offline Port, Host command	Fetch the frame header and send to next module	Read the frame header (up to 256 bytes - FMBM_OIM[FOF]) from ext memory to FMan internal memory and continue execution using NIA from OFNE. It is not possible to copy the frame header to external memory and enqueue the buffer. In FMan_v3, it is not possible to copy the frame header into a new storage profile; therefore it is illegal to configure the operational mode bits to this mode.
others	Reserved	—	Illegal	None

## 5.5.4 Frame Manager BMI Memory Map and Register Definition

The FMan registers are divided into 4-Kbyte groups. The BMI has 1Kbyte in each group. [Table 5-46](#) specifies the groups of registers used to configure the BMI. The storage profile section and the congestion group section are located in a separate 4-Kbyte regions in FMan Memory space. (See [Section 5.3.12.1, “FMan Memory Map and partitioning,”](#) for details of BMI memory space within FMan memory space.)

[Table 5-47](#) lists the offset, name, and a cross-reference to the complete description of each register. The offsets to the memory map (the 12 least significant bits of the absolute address of the registers), are relative to the beginning of the 4-Kbyte, “port-specific” page in the FMan memory map. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map”](#) for detailed explanation:

0	3	4	7	8	11	12	13	14	15	16	19	20	21	22	31
SoC	Fman Base	FMan <sup>1</sup>	HW Port ID (PortID) <n>		FMan module <sup>2</sup>	FMan module HW port address space <sup>3</sup>									
SoC specific	SoC specific	10	PortID (6 bits)		BMI=00	register offset from BMI base									
SoC address space <sup>4</sup>			FMan address space			Register offsets in this chapter									

**Figure 5-35. FMan Hardware Port Pages Address Space**

<sup>1</sup> FMan hardware ports pages offset. See [Table 5-17, “FMan Memory Map Regions.”](#)

<sup>2</sup> See [Table 5-18, “FMan Hardware Port Page Memory Map.”](#)

<sup>3</sup> See [Table 5-46, “BMI Memory Map.”](#)

<sup>4</sup> In some SoCs, the SoC address space has more bits.

## NOTE

Table 5-46 contains a super-set of FMan ports. See [Section 5.4.1, “FMan Hardware Ports,”](#) and the applicable SoC reference manual for SoC-specific information related to availability of FMan hardware ports (PortIDs) on each SoC.

**Table 5-46. BMI Memory Map**

Port ID	Address Range	Register
<b>General Registers Groups</b>		
—	0x0_0000–0x0_03FF	BMI Common Registers
0x01 <sup>1</sup>	0x0_1000–0x0_13FF	O/H1 (Offline Port/Host Command) Port Registers
0x02	0x0_2000–0x0_23FF	O/H2 Port Registers
0x03	0x0_3000–0x0_33FF	O/H3 Port Registers
0x04	0x0_4000–0x0_43FF	O/H4 Port Registers
0x05	0x0_5000–0x0_53FF	O/H5 Port Registers
0x06	0x0_6000–0x0_63FF	O/H6 Port Registers
0x07	0x0_7000–0x0_73FF	O/H7 Port Registers
0x08	0x0_8000–0x0_83FF	1/2.5Gbps Rx1 <sup>2</sup>
0x09	0x0_9000–0x0_93FF	1/2.5Gbps Rx2
0x0A	0x0_A000–0x0_A3FF	1/2.5Gbps Rx3
0x0B	0x0_B000–0x0_B3FF	1/2.5Gbps Rx4
0x0C	0x0_C000–0x0_C3FF	1/2.5Gbps Rx5
0x0D	0x0_D000–0x0_D3FF	1/2.5Gbps Rx6
0x0E	0x0_E000–0x0_E3FF	1/2.5Gbps Rx7
0x0F	0x0_F000–0x0_F3FF	1/2.5Gbps Rx8
0x10	0x1_0000–0x1_03FF	10G Rx1 / 10/1/2.5Gbps Rx9 <sup>3</sup>
0x11	0x1_1000–0x1_13FF	10G Rx2 / 10/1/2.5Gbps Rx10
0x12–0x27	0x1_1000–0x2_7FFF	Reserved
0x28	0x2_8000–0x2_83FF	1/2.5Gbps Tx1
0x29	0x2_9000–0x2_93FF	1/2.5Gbps Tx2
0x2A	0x2_A000–0x2_A3FF	1/2.5Gbps Tx3
0x2B	0x2_B000–0x2_B3FF	1/2.5Gbps Tx4
0x2C	0x2_C000–0x2_C3FF	1/2.5Gbps Tx5
0x2D	0x2_D000–0x2_D3FF	1/2.5Gbps Tx6

**Table 5-46. BMI Memory Map (continued)**

Port ID	Address Range	Register
0x2E	0x2_E000–0x2_E3FF	1/2.5Gbps Tx7
0x2F	0x2_F000–0x2_F3FF	1/2.5Gbps Tx8
0x30	0x3_0000–0x3_03FF	10G Tx1 / 1/2.5Gbps Tx9
0x31	0x3_1000–0x3_13FF	10G Tx2 / 1/2.5Gbps Tx10
0x32–0x3F	0x3_1000–0x3_FFFF	Reserved

<sup>1</sup> O/H1 and 10G Rx2/Tx2 cannot operate concurrently.

<sup>2</sup> Some ports in some devices do not support 2.5Gbps as well. BMI Rx1 processes frames from EMAC1 similarly for all ports BMI Rxn processes frames from EMACn. BMI Tx1 processes frames to EMAC1 etc. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map”](#).

<sup>3</sup> In FMan\_v3 this port is numbered ‘9’ and the second 10G port is numbered ‘10’.

[Table 5-47](#) lists the BMI memory-mapped registers

**Table 5-47. BMI Detailed Memory Map**

Offset	Register	Access	Reset Value <sup>1</sup>	Section/Page
<b>BMI Congestion Group Priority</b>				
0x000-0xFFFF	Congestion Group Priority	R/W	All zeros	<a href="#">5.5.4.2/5-93</a>
<b>BMI Virtual Storage Profile</b>				
0x000-0xFFFF	Virtual Storage Profile	R/W	All zeros	<a href="#">5.5.4.1.1/5-88</a>
<b>BMI Common Registers</b>				
0x000	FMBM_INIT—BMI Initialization	W	All zeros	<a href="#">5.5.4.4.1/5-93</a>
0x004	FMBM_CFG1—BMI Configuration 1	R/W	0x0nnnn_0000	<a href="#">5.5.4.4.2/5-94</a>
0x008	FMBM_CFG2—BMI Configuration 2	R/W	0x00nn_00nn	<a href="#">5.5.4.4.3/5-96</a>
0x00C–0x01F	Reserved.	—	—	—
0x020	FMBM_IER—Interrupt Event Register	R/W	All zeros	<a href="#">5.5.4.4.4/5-96</a>
0x024	FMBM_IER—Interrupt Enable Register	R/W	All zeros	<a href="#">5.5.4.4.5/5-97</a>
0x028	FMBM_IFR—Interrupt Force Register	R/W	All zeros	<a href="#">5.5.4.4.6/5-98</a>
0x02C–0x05F	Reserved	—	—	—
0x060–0x08F	Reserved	—	—	—
0x090–0x098	FMBM_DTC<1-3>—Debug Trap Counter.	R/W	All zeros	<a href="#">5.5.4.4.7/5-99</a>
0x09C	Reserved	—	—	—
0x0A0–0x0CC	FMBM_DCV_<1-4>_<1-3>—Debug Compare Value.	R/W	All zeros	<a href="#">5.5.4.4.8/5-99</a>
0x0D0–0x0FC	FMBM_DCM_<1-4>_<1-3>—Debug Compare Mask.	R/W	All zeros	<a href="#">5.5.4.4.9/5-100</a>
0x100	FMBM_GDE—Global Debug Enable	R/W	All zeros	<a href="#">5.5.4.4.10/5-100</a>

**Table 5-47. BMI Detailed Memory Map**

Offset	Register	Access	Reset Value <sup>1</sup>	Section/Page
0x104–0x1FC	FMBM_PP_1-63—BMI Port Parameters	R/W	0xnnn00_0n00	<a href="#">5.5.4.4.11/5-101</a>
0x200	Reserved	—	—	—
0x204–0x2FC	FMBM_PFS_1-63—BMI Port FIFO Size	R/W	0x0000_0nnn	<a href="#">5.5.4.4.12/5-103</a>
0x300	Reserved	—	—	—
0x304–0x3FC	FMBM_SPICID_1-63—Port Partition ID	R/W	All zeros	<a href="#">5.5.4.4.13/5-106</a>
<b>BMI O/H Port Registers</b>				
0x000	FMBM_OCFG—O/H Configuration	R/W	All zeros	<a href="#">5.5.4.7.1/5-174</a>
0x004	FMBM_OST—O/H Status	R only	0x00nn_0000	<a href="#">5.5.4.7.2/5-175</a>
0x008	FMBM_ODA—O/H DMA attributes	R/W	All zeros	<a href="#">5.5.4.7.3/5-176</a>
0x00C	FMBM_OICP—O/H Internal Context Parameters	R/W	All zeros	<a href="#">5.5.4.7.4/5-177</a>
0x010	FMBM_OFDNE—O/H Frame Dequeue Next Engine	R/W	0x0058_0000	<a href="#">5.5.4.7.5/5-179</a>
0x014	FMBM_OFNE—O/H Frame Next Engine	R/W	0x0044_0000	<a href="#">5.5.4.7.6/5-179</a>
0x018	FMBM_OFCA—O/H Frame Attributes.	R/W	0x803C_0000	<a href="#">5.5.4.7.7/5-180</a>
0x01C	FMBM_OFPNE—O/H Frame Parser Next Engine	R/W	0x0048_0000	<a href="#">5.5.4.7.8/5-181</a>
0x020	FMBM_OPSO—O/H Parse Start Offset	R/W	All zeros	<a href="#">5.5.4.7.9/5-182</a>
0x024	FMBM_OPP—O/H Policier Profile	R/W	All zeros	<a href="#">5.5.4.7.10/5-182</a>
0x028	FMBM_OCCB—O/H custom classifier Base	R/W	All zeros	<a href="#">5.5.4.7.11/5-183</a>
0x02C	FMBM_OIM—O/H Internal Margins	R/W	All zeros	<a href="#">5.5.4.7.12/5-183</a>
0x030	FMBM_OFP —O/H FIFO Parameters.	R/W	0x0000_n000	<a href="#">5.5.4.7.13/5-184</a>
0x034	FMBM_OFED—O/H Frame End Data.	R/W	All zeros	<a href="#">5.5.4.7.14/5-185</a>
0x038–0x03F	Reserved	—	—	—
0x040–0x05C	FMBM_OPRI—O/H Parse Result Initialization	R/W	0xnnnn_nnnn	<a href="#">5.5.4.7.15/5-186</a>
0x060	FMBM_OFQID—O/H Frame Queue ID	R/W	All zeros	<a href="#">5.5.4.7.16/5-187</a>
0x064	FMBM_OEFQID—O/H Error Frame Queue ID	R/W	All zeros	<a href="#">5.5.4.7.17/5-188</a>
0x068	FMBM_OFSDM—O/H Frame Status Discard Mask	R/W	All zeros	<a href="#">5.5.4.7.18/5-189</a>
0x06C	FMBM_OFSEM—O/H Frame Status Error Mask	R/W	All zeros	<a href="#">5.5.4.7.19/5-189</a>
0x070	FMBM_OFENE—O/H Frame Enqueue Next Engine	R/W	0x00D4_0000	<a href="#">5.5.4.7.20/5-190</a>
0x074	FMBM_ORLMTS—O/H Rate Limiter Scale	R/W	All zeros	<a href="#">5.5.4.7.21/5-191</a>
0x078	FMBM_ORLMT—O/H Rate Limiter	R/W	All zeros	<a href="#">5.5.4.7.22/5-192</a>
0x07C	FMBM_OCMNE—O/H Continuous Mode Next Engine.	R/W	All zeros	<a href="#">5.5.4.7.23/5-193</a>
0x080–0xFF	Reserved	—	—	—
0x160	FMBM_OCGM - Observed Congestion Group Map.	R/W	All zeros	<a href="#">5.5.4.7.24/5-194</a>

**Table 5-47. BMI Detailed Memory Map**

Offset	Register	Access	Reset Value <sup>1</sup>	Section/Page
0x184 - 0x1FF	Reserved	—	—	—
0x200	FMBM_OSTC—O/H Statistics Counters	R/W	All zeros	5.5.4.7.25/5-195
0x204	FMBM_OFRC—O/H Frame Counter	R/W	All zeros	5.5.4.7.26/5-196
0x208	FMBM_OFDC—O/H Frames Discard Counter	R/W	All zeros	5.5.4.7.27/5-196
0x20C	FMBM_OFLEDC—O/H Frames Length Error Discard Counter	R/W	All zeros	5.5.4.7.28/5-197
0x210	FMBM_OFUFDC—O/H Frames Unsupported Format Discard Counter	R/W	All zeros	5.5.4.7.29/5-198
0x214	FMBM_OFFC—O/H Filter Frames Counter	R/W	All zeros	5.5.4.7.30/5-198
0x218	FMBM_OFWDC—O/H Frames WRED Discard Counter	R/W	All zeros	5.5.4.7.31/5-199
0x21C	FMBM_OFLDEC—O/H Frames List DMA Error Counter	R/W	All zeros	5.5.4.7.32/5-200
0x220	FMBM_OBDC—O/H Buffers Deallocate Counter	R/W	All zeros	5.5.4.7.33/5-201
0x224	FMBM_OODC—O/H Out of Buffers Discard Counter.	R/W	All zeros	5.5.4.7.34/5-201
0x228	FMBM_OPEC—O/H Prepare to enqueue Counter.	R/W	All zeros	5.5.4.7.35/5-202
0x22C–0x27F	Reserved	—	—	—
0x280	FMBM_OPC—O/H Performance Counters	R/W	All zeros	5.5.4.7.36/5-203
0x284	FMBM_OPCP—O/H Performance Count Parameters	R/W	All zeros	5.5.4.7.37/5-203
0x288	FMBM_OCCN—O/H Cycle Counter	R/W	All zeros	5.5.4.7.38/5-205
0x28C	FMBM_OTUC—O/H Tasks Utilization Counter	R/W	All zeros	5.5.4.7.39/5-205
0x290	FMBM_ODUC—O/H DMA Utilization Counter	R/W	All zeros	5.5.4.7.40/5-206
0x294	FMBM_OFUC—O/H FIFO Utilization Counter	R/W	All zeros	5.5.4.7.41/5-207
0x298–0x2FF	Reserved	—	—	—
0x300–0x308	FMBM_ODCFG—O/H Debug Configuration.	R/W	0x0000_0090	5.5.4.7.42/5-207
0x30C	FMBM_OGPR—O/H General Purpose Register.	R/W	All zeros	5.5.4.7.43/5-209
0x310–0x3FF	Reserved	—	—	—

**BMI Rx Port Registers**

0x000	FMBM_RCFG—Rx Configuration	R/W	All zeros	5.5.4.5.1/5-108
0x004	FMBM_RST—Rx Status	R	0x00nn_0000	5.5.4.5.2/5-109
0x008	FMBM_RDA—Rx DMA attributes	R/W	All zeros	5.5.4.5.3/5-110
0x00C	FMBM_RFP—Rx FIFO Parameters	R/W	0x03FF_03FF	5.5.4.5.4/5-112
0x010	FMBM_RFED—Rx Frame End Data	R/W	All zeros	5.5.4.5.5/5-113
0x014	FMBM_RICP—Rx Internal Context Parameters	R/W	0x0000_0002	5.5.4.5.6/5-115

**Table 5-47. BMI Detailed Memory Map**

Offset	Register	Access	Reset Value <sup>1</sup>	Section/Page
0x018	FMBM_RIM—Rx Internal Margins	R/W	All zeros	<a href="#">5.5.4.5.7/5-116</a>
0x01C	FMBM_REBM—Rx External Buffer Margins	R/W	0x0002_0000	<a href="#">5.5.4.5.8/5-117</a>
0x020	FMBM_RFNE—Rx Frame Next Engine	R/W	0x0044_0000	<a href="#">5.5.4.5.9/5-118</a>
0x024	FMBM_RFCA—Rx Frame Attributes.	R/W	0x803C_0000	<a href="#">5.5.4.5.10/5-119</a>
0x028	FMBM_RFPNE—Rx Frame Parser Next Engine	R/W	0x0048_0000	<a href="#">5.5.4.5.11/5-120</a>
0x02C	FMBM_RPSO—Rx Parse Start Offset	R/W	All zeros	<a href="#">5.5.4.5.12/5-121</a>
0x030	FMBM_RPP—Rx Policer Profile	R/W	All zeros	<a href="#">5.5.4.5.13/5-121</a>
0x03C	Reserved	—	—	—
0x040–0x05C	FMBM_RPRI—Rx Parse Result Initialization	R/W	0xnnnn_nnnn	<a href="#">5.5.4.5.14/5-122</a>
0x060	FMBM_RFQID—Rx Frame Queue ID	R/W	All zeros	<a href="#">5.5.4.5.15/5-123</a>
0x064	FMBM_REFQID—Rx Error Frame Queue ID	R/W	All zeros	<a href="#">5.5.4.5.16/5-124</a>
0x068	FMBM_RFSDM—Rx Frame Status Discard Mask	R/W	All zeros	<a href="#">5.5.4.5.17/5-125</a>
0x06C	FMBM_RFSEM—Rx Frame Status Error Mask	R/W	All zeros	<a href="#">5.5.4.5.18/5-125</a>
0x070	FMBM_RFENE—Rx Frame Enqueue Next Engine	R/W	0x00D4_0000	<a href="#">5.5.4.5.19/5-126</a>
0x070–0x078	Reserved	—	—	—
		—	—	—
0x07C	FMBM_RCMNE—Rx Frame Continuous Mode Next Engine.	R/W	All zeros	<a href="#">5.5.4.5.20/5-127</a>
0x080–0x0FF	Reserved	—	—	—
0x100–0x10C	FMBM_REBMPI—Buffer Manager pool Information	R/W	All zeros	<a href="#">5.5.4.5.21/5-127</a>
0x110–0x11C	FMBM_REBMPI—Buffer Manager pool Information.	R/W	All zeros	<a href="#">5.5.4.5.21/5-127</a>
0x120–0x13C	FMBM_RACNT—Allocate Counter	R/W	All zeros	<a href="#">5.5.4.5.22/5-129</a>
0x140–0x15F	Reserved	—	—	—
0x160–0x17C	FMBM_RCGM—Congestion Group Map	R/W	All zeros	<a href="#">5.5.4.5.23/5-130</a>
0x180	FMBM_RMPD—BMan Pool Depletion	R/W	All zeros	<a href="#">5.5.4.5.24/5-131</a>
0x184–0x1FF	Reserved	—	—	—
0x200	FMBM_RSTC—Rx Statistics Counters	R/W	All zeros	<a href="#">5.5.4.5.25/5-132</a>
0x204	FMBM_RFRC—Rx Frame Counter	R/W	All zeros	<a href="#">5.5.4.5.26/5-133</a>
0x208	FMBM_RBFC—Rx Bad Frames Counter	R/W	All zeros	<a href="#">5.5.4.5.27/5-133</a>
0x20C	FMBM_RLFC—Rx Large Frames Counter	R/W	All zeros	<a href="#">5.5.4.5.28/5-134</a>
0x210	FMBM_RFFC—Rx Filter Frames Counter	R/W	All zeros	<a href="#">5.5.4.5.29/5-135</a>
0x214	FMBM_RFDC—Rx Frame Discard Counter	R/W	All zeros	<a href="#">5.5.4.5.30/5-135</a>
0x218	FMBM_RFLDEC—Rx Frames List DMA Error Counter	R/W	All zeros	<a href="#">5.5.4.5.31/5-136</a>

**Table 5-47. BMI Detailed Memory Map**

Offset	Register	Access	Reset Value <sup>1</sup>	Section/Page
0x21C	FMBM_RODC—Rx Out of Buffers Discard Counter	R/W	All zeros	<a href="#">5.5.4.5.32/5-137</a>
0x220	FMBM_RBDC—Rx Buffers Deallocate Counter	R/W	All zeros	<a href="#">5.5.4.5.33/5-138</a>
0x224	FMBM_RPEC—RX Prepare to enqueue Counter.	R/W	All zeros	<a href="#">5.5.4.5.34/5-138</a>
0x228–0x27F	Reserved	—	—	—
0x280	FMBM_RPC—Rx Performance Counters	R/W	All zeros	<a href="#">5.5.4.5.35/5-139</a>
0x284	FMBM_RPCP—Rx Performance Count Parameters	R/W	All zeros	<a href="#">5.5.4.5.36/5-139</a>
0x288	FMBM_RCCN—Rx Cycle Counter	R/W	All zeros	<a href="#">5.5.4.5.37/5-141</a>
0x28C	FMBM_RTUC—Rx Tasks Utilization Counter	R/W	All zeros	<a href="#">5.5.4.5.38/5-142</a>
0x290	FMBM_RRQUC—Rx Receive Queue Utilization Counter	R/W	All zeros	<a href="#">5.5.4.5.39/5-142</a>
0x294	FMBM_RDUC—Rx DMA Utilization Counter	R/W	All zeros	<a href="#">5.5.4.5.40/5-143</a>
0x298	FMBM_RFUC—Rx FIFO Utilization Counter	R/W	All zeros	<a href="#">5.5.4.5.41/5-144</a>
0x29C	FMBM_RPAC—Rx Pause Activation Counter	R/W	All zeros	<a href="#">5.5.4.5.42/5-144</a>
0x2A0–0x2FF	Reserved	—	—	—
0x300–0x308	FMBM_RDCFG—Rx Debug Configuration	R/W	0x0000_0090	<a href="#">5.5.4.5.43/5-145</a>
0x30C	FMBM_RGPR—Rx General Purpose Register.	R/W	All zeros	<a href="#">5.5.4.5.44/5-147</a>
0x310–0x3FF	Reserved	—	—	—

**BMI Tx Port Registers**

0x000	FMBM_TCFG—Tx Configuration	R/W	All zeros	<a href="#">5.5.4.6.1/5-147</a>
0x004	FMBM_TST—Tx Status	R only	0x00nn_0000	<a href="#">5.5.4.6.2/5-148</a>
0x008	FMBM_TDA—Tx DMA attributes	R/W	All zeros	<a href="#">5.5.4.6.3/5-149</a>
0x00C	FMBM_TFP—Tx FIFO Parameters	R/W	0x0000_n013	<a href="#">5.5.4.6.4/5-149</a>
0x010	FMBM_TFED—Tx Frame End Data	R/W	All zeros	<a href="#">5.5.4.6.5/5-151</a>
0x014	FMBM_TICP—Tx Internal Context Parameters	R/W	All zeros	<a href="#">5.5.4.6.6/5-152</a>
0x018	FMBM_TFDNE—Tx Frame Dequeue Next Engine	R/W	0x0058_0000	<a href="#">5.5.4.6.7/5-153</a>
0x01C	FMBM_TFCA—Tx Frame Attributes	R/W	0x8C00_0000	<a href="#">5.5.4.6.8/5-154</a>
0x020	FMBM_TCFQID—Tx Confirmation Frame Queue ID.	R/W	All zeros	<a href="#">5.5.4.6.9/5-155</a>
0x024	FMBM_TEFQID—Tx Error Frame Queue ID	R/W	All zeros	<a href="#">5.5.4.6.10/5-156</a>
0x028	FMBM_TFENE—Tx Frame Enqueue Next Engine	R/W	0x00D4_0000	<a href="#">5.5.4.6.11/5-156</a>
0x02C	FMBM_TRLMTS—Tx Rate Limiter Scale	R/W	All zeros	<a href="#">5.5.4.6.12/5-157</a>
0x030	FMBM_TRLMT—Tx Rate Limiter	R/W	All zeros	<a href="#">5.5.4.6.13/5-158</a>
0x038 - 0x06C	Reserved	—	—	—
0x034	FMBM_TCCB-Tx Custom Classifier Base.	R/W	All zeros	<a href="#">5.5.4.6.14/5-159</a>

**Table 5-47. BMI Detailed Memory Map**

Offset	Register	Access	Reset Value <sup>1</sup>	Section/Page
0x070	FMBM_TFNE—Tx Frame Next Engine.	R/W	0x8000_0000	<a href="#">5.5.4.6.15/5-160</a>
0x074	FMBM_TPFCM0—Tx PFC Mapping 0	R/W	0x01234567	<a href="#">5.5.4.6.17/5-161</a>
0x078	Reserved	R/W	—	—
0x07C	FMBM_TCMNE-Tx Frame Continuous Mode Next Engine.	R/W	0x0000_0000	<a href="#">5.5.4.6.17/5-161</a>
0x080 - 0x1FF	Reserved	—	—	—
0x200	FMBM_TSTC—Tx Statistics Counters	R/W	All zeros	<a href="#">5.5.4.6.18/5-162</a>
0x204	FMBM_TFRC—Tx Frame Counter	R/W	All zeros	<a href="#">5.5.4.6.19/5-163</a>
0x208	FMBM_TFDC—Tx Frames Discard Counter	R/W	All zeros	<a href="#">5.5.4.6.20/5-163</a>
0x20C	FMBM_TFLEDC—Tx Frames Length Error Discard Counter	R/W	All zeros	<a href="#">5.5.4.6.21/5-164</a>
0x210	FMBM_TFUFD—Tx Frames Unsupported Format Discard Counter	R/W	All zeros	<a href="#">5.5.4.6.22/5-165</a>
0x214	FMBM_TBDC—Tx Buffers Deallocate Counter	R/W	All zeros	<a href="#">5.5.4.6.23/5-165</a>
0x218–0x27F	Reserved	—	—	—
0x280	FMBM_TPC—Tx Performance Counters	R/W	All zeros	<a href="#">5.5.4.6.24/5-166</a>
0x284	FMBM_TPCP—Tx Performance Count Parameters	R/W	All zeros	<a href="#">5.5.4.6.25/5-166</a>
0x288	FMBM_TCCN—Tx Cycle Counter	R/W	All zeros	<a href="#">5.5.4.6.26/5-168</a>
0x28C	FMBM_TTUC—Tx Tasks Utilization Counter	R/W	All zeros	<a href="#">5.5.4.6.27/5-169</a>
0x290	FMBM_TTCQUC—Tx Transmit Confirm Queue Utilization Counter	R/W	All zeros	<a href="#">5.5.4.6.28/5-169</a>
0x294	FMBM_TDUC—Tx DMA Utilization Counter	R/W	All zeros	<a href="#">5.5.4.6.29/5-170</a>
0x298	FMBM_TFUC—Tx FIFO Utilization Counter	R/W	All zeros	<a href="#">5.5.4.6.30/5-171</a>
0x29C–0x2FF	Reserved	—	—	—
0x300–0x308	FMBM_TDCFG—Tx Debug Configuration	R/W	0x0000_0090	<a href="#">5.5.4.6.31/5-171</a>
0x30C	FMBM_TGPR—Tx General Purpose Register.	R/W	All zeros	<a href="#">5.5.4.6.32/5-173</a>
0x310–0x3FF	Reserved	—	—	—

<sup>1</sup> Some reset values differ between version of the FMan. See register description for details.

### 5.5.4.1 Storage Profiles

The storage profiles contain parameters (most importantly the buffer pool ID) that are used by the FMan in order to store frames being received on the Rx ports, or to copy frames being fetched from the O/H ports or to determine the parameters that affect writing the Internal Context in the frame margin on Tx. There are two mechanisms for selecting storage profiles:

- Hardware port (physical) storage profile

- Virtual storage profile

The hardware port storage profile mechanism selects the storage profile according to the hardware port the frame is received on (this is supported on Rx port only). The virtual storage profile mechanism allows the virtualization of the storage profile selection (which includes buffer pool selection) from the physical hardware ports. Both storage profile mechanisms can be used concurrently.

On Tx ports, the storage profile ID is an absolute offset that points to the storage profile used for the frame. Up to 256 storage profiles are supported. See SoC introduction for number of storage profiles supported in a given device. The Tx ports uses the storage profile to determine how to handle the reading/writing of the Internal Context (IC) in the margin of the frame. Virtual storage profiles are enabled in FMBM\_TFNE[VSPE] or in FQD Context A; the storage profile ID is initialized in FMBM\_TCFQID and optionally modified by FQD Context B. If FMBM\_TFNE[VSPE]=0 FMBM\_TICP values are used.

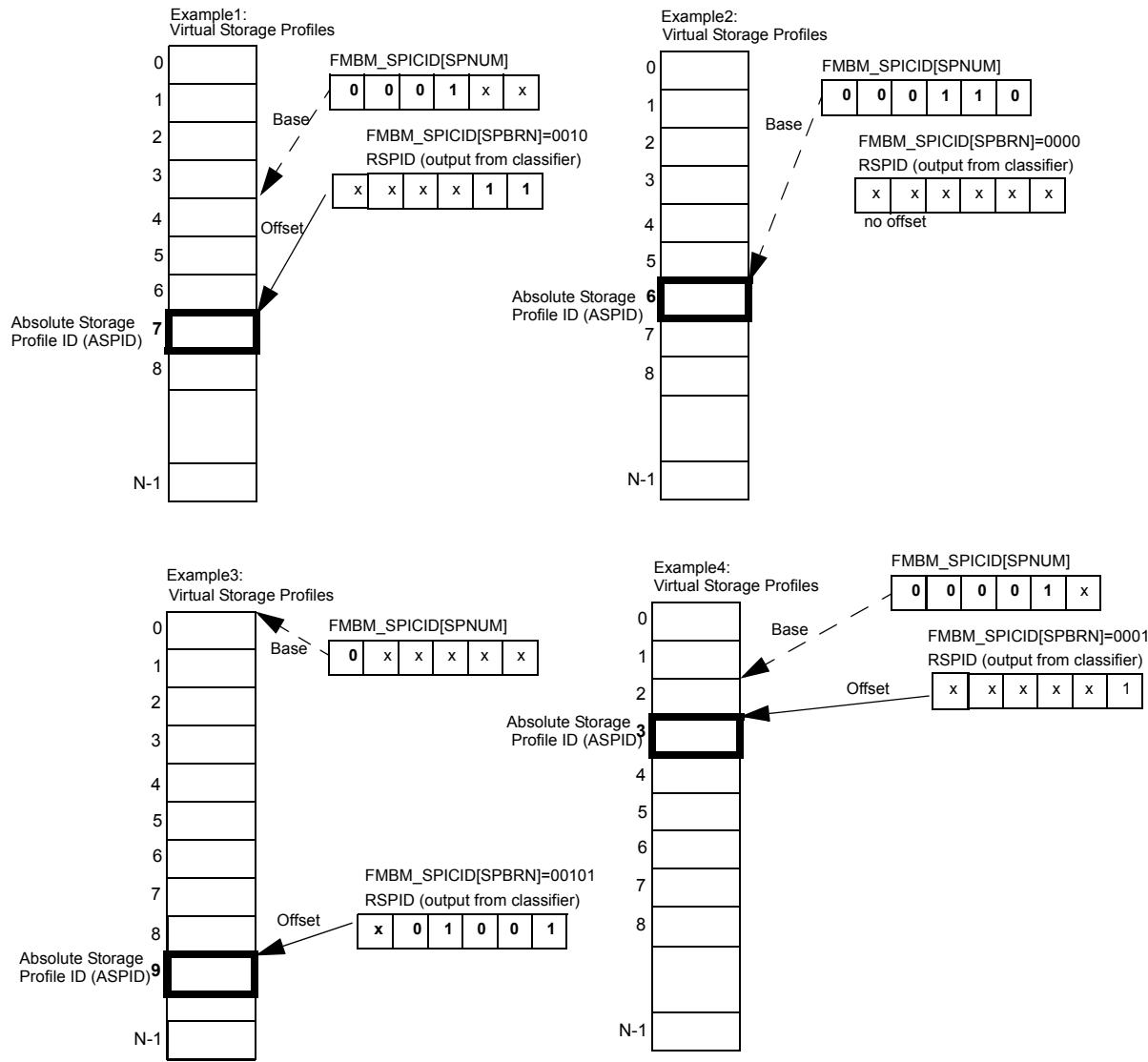
#### **5.5.4.1.1 Storage Profile Virtualization per port (for Rx and Offline ports)**

There is a user configurable mechanism that allows for the virtualization of the storage profiles. With this mechanism some Relative Storage Profile ID (RSPID) values in the classification on different physical ports may yield to different absolute storage profile IDs (ASPID). On Rx and offline ports, the RSPID values are relative offsets, and are evaluated with fields in the FMBM\_SPICID register to format an absolute storage profile ID. The number of virtual storage profiles supported varies on each device.

**Table 5-48. Number of virtual storage profiles**

FMan Type Devices	Total Number of virtual storage profiles	Number of virtual storage profiles per port
LS1043A (FMan_v3)	64	64

See [Figure 5-36](#) for examples of settings for storage profile selection.



**Figure 5-36. Examples of virtual profile selection**

#### 5.5.4.1.2 Hardware port storage profiles

With this mechanism, the storage profile is defined on a per port basis, and is selected dynamically based on the hardware port that frame is received on (Rx port). The storage profile registers are mapped in the port specific section of the FMan memory map. This mechanism is supported on Rx ports in all FMan versions.

#### NOTE

Hardware port storage profiles are supported on Rx ports only.

The Hardware storage profiles are selected when the Virtual Storage Profile Enable (VSPE) value equals to zero. See [Section 5.5.4.1.3, “Virtual storage profiles”](#) for details on SPID. [Table 5-49](#) describes the storage profile. Each storage profile contains up to four buffer pools which are selected by the BMI based on the packet size and mode (FMBM\_REBMPI[BP]).

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	19	20	21	22	23	24	25	26	27	31	BMI Register Name <sup>1</sup>
As defined in <a href="#">Table 5-47</a>	VAL	ACE	BP	—									BPID				PBS										<a href="#">Section 5.5.4.5.21, “Rx External Buffers Manager Pool Information Register (FMBM_REBMPI)”</a>	
													CNT															<a href="#">Section 5.5.4.5.22, “Rx Allocate Counter Register (FMBM_RACNT)”</a>
	Up to four (or eight) pairs of FMBM_REBMPI and FMBM_RACNT per hardware port <sup>2</sup>																											
	SGD	—							BSM				—				BEM										<a href="#">Section 5.5.4.5.8, “Rx External Buffer Margins Register (FMBM_REBM)”</a>	
	0	1	2	3	4	5	6	7	8	9	10	11	12	15	16		22	23									31	
	SWAP	ICC	FHC	SGC	—	WOP							—															<a href="#">Section 5.5.4.5.3, “Rx DMA Attributes Register (FMBM_RDA)”</a>
	0	1	2	3	4	5	6	7	8	9	10	11	12	15	16	19	20	23	24	26	27						31	
		—							ICEOF				—	ICIOF		—	ICSZ											<a href="#">Section 5.5.4.5.6, “Rx Internal Context Parameters (FMBM_RICP)”</a>
	NBPDE				—				NBPD				PFCPEV				SBPD											<a href="#">Section 5.5.4.5.24, “BMan Pool Depletion Register (FMBM_RMPD)”</a>
	0		3	4	7	8	10	11	12	13	15	16				23	24										31	
	—		SPBRN	—		SPNUM			—				—				ICID											<a href="#">Section 5.5.4.4.13, “SPCID Register (FMBM_SPCID_1–63)”<sup>3</sup></a>

<sup>1</sup> The description of the bits are found in these sections

<sup>2</sup> The number of FMBM\_REBMPI, FMBM\_RACNT registers per hardware port, varies from device to device (up to eight in some devices).

<sup>3</sup> SPBRN and SPNUM bits are only supported in FMan\_v3 and only in the hardware port storage profile.

**Table 5-49. Hardware port storage profile**

### 5.5.4.1.3 Virtual storage profiles

Each virtual storage profile consists of a 64 byte data structure. The FMan supports multiple virtual storage profile (see SoC DPAA introduction for details on how many virtual storage profiles are supported by the FMan in the various devices). The virtual storage profiles are stored in a 4KByte region of the FMan memory map. The FMan selects a virtual storage profile dynamically on a frame-per-frame basis, during the classification process. The relative Storage Profile ID (RSPID) is programmed in custom classifier data structures, and/or in the KeyGen schemes (see [5.10.3.12.15, “KeyGen Scheme Entry Virtual Storage Profile Register \(AWR8\\_RFMODE\\_FMKG\\_SE\\_VSP\)”](#)). The default value is programmed in

Section 5.5.4.5.15, “Rx Frame Queue ID Register (FMBM\_RFQID)” or Section 5.5.4.7.16, “O/H Frame Queue ID Register (FMBM\_OFQID).” Note that VSPE=0 selects the hardware port storage profile.

The storage profiles can be initialized by accessing the memory structure defined below directly. For dynamic changes of the parameters of the storage profile, a ‘Storage Profile Update’ Host command is issued to the FMan controller.

Table 5-50 describes the storage profile. Each storage profile contains up to four buffer pools which are selected by the BMI based on the packet size and mode (FMBM\_REBMPI[BP]).

Offset <sup>1</sup>	Register name Name in virtual storage profile <sup>2</sup>	0	1	2	3	9	10	15	16	31	Description is found in <sup>3</sup>								
0x0	FMBM_VEBMPI_0	VAL	ACE	BP	—	BPID		PBS			Section 5.5.4.5.21, “Rx External Buffers Manager Pool Information Register (FMBM_REBMPI)” <sup>4</sup>								
0x4	FMBM_VEBMPI_1	VAL	ACE	BP	—	BPID		PBS											
0x8	FMBM_VEBMPI_2	VAL	ACE	BP	—	BPID		PBS											
0xC	FMBM_VEBMPI_3	VAL	ACE	BP	—	BPID		PBS											
0x10-0x1F		Reserved																	
0x20	FMBM_VACNT	CNT																	
		0	1		6	7		15	16	22	31								
0x24	FMBM_VEBM	SGD	—		BSM			—	BEM		Section 5.5.4.5.8, “Rx External Buffer Margins Register (FMBM_REBM)” <sup>4</sup>								
		0	1	2	3	4	5	6	7	8	31								
0x28	FMBM_VDA	SWAP	ICC	FHC	SGC	—	WOPT	—	—			Section 5.5.4.5.3, “Rx DMA Attributes Register (FMBM_RDA)” <sup>4</sup>							
0x2C	FMBM_VICP	—			ICEOF			—	ICIOF	—	ICSZ	Section 5.5.4.5.6, “Rx Internal Context Parameters (FMBM_RICP)” <sup>4</sup>							
		0			7	8	10	11	12	13	15	16	19	20	23	24	26	27	31
0x30	FMBM_VMPD	NBPDE				—	NBPD	PFCPEV			SBPD		Section 5.5.4.5.24, “BMan Pool Depletion Register (FMBM_RMPD)” <sup>4</sup>						
0x34-0x38		Reserved																	
		0								23	24		31						
0x3C	FMBM_VSPICID	—								ICID		Section 5.5.4.4.13, “SPICID Register (FMBM_SPICID_1-63)” <sup>5</sup>							

<sup>1</sup> The Offset from the beginning of this storage profile

<sup>2</sup> The reset value of the virtual storage profile registers these registers is the same as the reset value for the corresponding HW storage profile registers.

<sup>3</sup> The description of the bits are found in these sections. The ‘R’ in the register name is replaced by ‘V’. The port numbers in these sections are not applicable to the Virtual Storage Profile.

<sup>4</sup> There is one counter per storage profile

<sup>5</sup> The rest of the fields in this registers are reserved in the virtual storage profile, the fields are used in the hardware port register, to select the virtual storage profile.

**Table 5-50. Virtual Storage Profile**

### NOTE

ICID bits are protected, therefore they are accessed only by direct read/write by management. The ‘Storage Profile Update’ host command does not write this entry.

#### **5.5.4.2 Congestion Group Priority Mapping table**

When congestion is detected in one of the QMan congestion groups, a priority based flow control (PFC) packet is transmitted to the appropriate port as configured in FMBM\_RCGM. Each congestion group has a congestion group priority (eight bits) associated with it. There are up to 256 congestion groups, therefore there are up to 64 registers, each one contains four congestion group priorities.

The congestion group priorities are programmed in a 4K memory space in the FMan memory map (see [Section 5.4.2, “FMan Detailed Memory Map”](#): ‘Congestion groups priority mapping’).

Offset	1	0	7	8	15	16	23	24	31
0x0	congestion group 255 priority (CGP255)		CGP254		CGP253		CGP252		
0x4	CGP251		CGP250		CGP249		CGP248		
.....	Continue in a similar way								
0xFC	CGP3		CGP1		CGP1		CGP0		

<sup>1</sup> The Offset from the base of congestion group priority. The base address is documented in FMan memory map.

**Table 5-51. Congestion group priority mapping table**

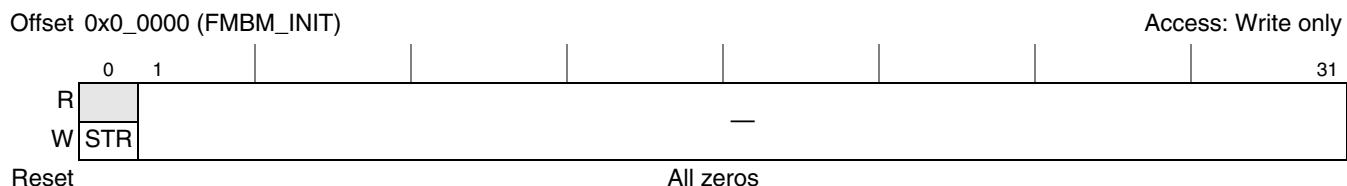
#### **5.5.4.3 PFC priority mapping to QMan traffic classes**

When a PFC flow control packet is received, the corresponding Traffic Class in QMan is paused. The feature is described in the QMan chapter section “Traffic Class (TC) Flow Control” and ”Traffic Class Flow Control Configure” section. The FMBM\_TPFCM0,1 registers are used to configure the mapping of PFC class (in incoming PFC class enable vector) to QMan traffic class. See [Section 5.5.4.6.16, “Tx Priority based Flow Control \(PFC\) Mapping Registers \(FMBM\\_TPFCM0\).”](#)

#### **5.5.4.4 FMan BMI Common Registers Description**

#### **5.5.4.4.1 BMI Initialization Register (FMBM\_INIT)**

The BMI initialization register (FMBM\_INIT), shown in Figure 5-37, controls initialization of the BMI link list and internal buffers.



**Figure 5-37. BMI Initialization Register (FMBM\_INIT)**

Table 5-52 describes the FMBM INIT fields.

**Table 5-52. FMBM\_INIT Field Descriptions**

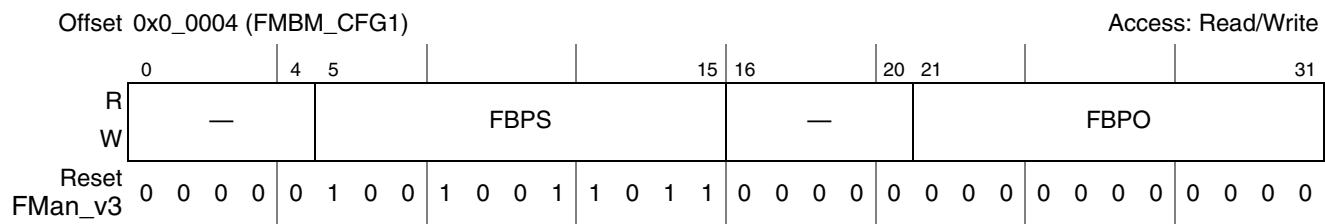
Bits	Name	Description
0	STR	Start When set, the BMI initializes internal buffers data base. Any data on internal buffers is lost when this bit is set. BMI STR should be asserted before enabling any communication port.
1–31	—	Reserved

#### 5.5.4.4.2 BMI Configuration 1 Register (FMBM\_CFG1)

The BMI configuration 1 register (FMBM\_CFG1), shown in [Figure 5-38](#), is used to configure the BMI global settings. The fields in this register, define the FMan internal memory for Tx, Rx, and O/H FIFOs. Any remaining FMan internal memory may be used by the FMan controller (for custom classifier tables and other protocols).

The value of FBPS field in this register must be larger the sum of the values programmed in the FMBM\_PFSn for all ports. In FMan\_v3, the default reset value of FMBM\_CFG1 and FMBM\_PFSn comply with this rule. Note that in some devices it is not possible to enable all ports at once. Line rate and memory limitations must be taken in consideration.

For FMan\_v3 it is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug. Suggest value per port.



**Figure 5-38. BMI Configuration 1 Register (FMBM\_CFG1)**

[Table 5-53](#) describes the FMBM\_CFG1 fields.

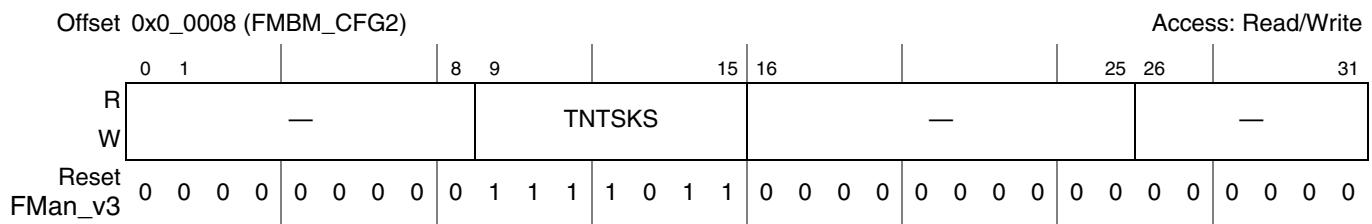
**Table 5-53. FMBM\_CFG1 Field Descriptions**

Bits	Name	Description
0–4	—	Reserved.
5–15	FBPS	<p>Free buffer pool size.          The total size of the free buffer pool used by network ports in FMan memory.          The specified size should correspond to the actual BMI allocated size in FMan memory.          Note that the sum of the free buffer pool size (FBPS) and offset (FBPO) cannot exceed the FMan memory total size. See <a href="#">Section 5.3.13, “FMan Internal Memory”</a>.</p> <p>The size is specified in 256 bytes granularity.          0x000 The size of free buffer pool is <math>1 \times 256</math>bytes          0x001 The size of free buffer pool is <math>2 \times 256</math>bytes          ...          0xFF The size of free buffer pool is <math>256 \times 256</math>bytes.          ...          0x3FF The size of free buffer pool is <math>1024 \times 256</math>bytes.          0x400 The size of free buffer pool is <math>1025 \times 256</math>bytes          ...  <b>Note:</b> Values that are bigger than the FMan memory are reserved and should not be used. See SoC reference guide for the size of the FMan memory in each SoC.</p> <p>The BMI internal buffers consist of blocks of 256 bytes in the FMan memory which are organized as a linked list. This linked list of buffers is handled by the BMI hardware. With this register, the user allocates the memory space for these buffers. Initially all the buffers are ‘free’ therefore this is called the ‘BMI free buffer pool’. The free buffer pool allocated by this register, is the overall internal memory space used by the BMI, for temporary storage associated with data for the various ports (Rx, Tx, Offline FIFOs), and the relevant parameters (such as the Internal Context (IC)). The FMan Controller custom classifier lookup tables are not included in this space; they are placed in the address space which is not allocated by the combination of FBPS and FPBO. See <a href="#">Section 5.3.13, “FMan Internal Memory”</a> for more details regarding on the configuration of the internal memory.</p>
16–20	—	Reserved.
21–31	FBPO	<p>Free Buffer Pool Offset.          The offset in FMan memory to the start address of Free Buffer Pool.          The offset is specified in 256 bytes granularity.          Note that the sum of the free buffer pool size (FBPS) and offset (FBPO) cannot exceed the FMan memory total size. The user may configure FBPO = N, and place the custom classifier tables in the address space 0:N-1. See <a href="#">Section 5.3.13, “FMan Internal Memory”</a>.</p> <p>0x000 The start address of free buffer pool is 0x0.          0x001 The start address of free buffer pool is 0x100.          0x002 The Start address of free buffer pool is 0x200.          ...          0x0FE The start address of free buffer pool is 0xFE00.          ...          0x3FE The start address of free buffer pool is 0x3FE00.          ...          0x7FE The start address of free buffer pool is 0x7FE00.          0x7FF Reserved  <b>Note:</b> Values that are bigger than the FMan memory are reserved and should not be used.</p>

#### 5.5.4.4.3 BMI Configuration 2 Register (FMBM\_CFG2)

The BMI configuration 2 register (FMBM\_CFG2), shown in [Figure 5-39](#), is used to configure the BMI global settings.

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug.



**Figure 5-39. BMI Configuration 2 Register (FMBM\_CFG2)**

**Table 5-54. FMBM\_CFG2 Field Descriptions**

Bits	Name	Description
0-8	—	Reserved.
9-15	TNTSKS	Total number of tasks. Specify the maximum number of concurrent tasks (received frames, transmit frames, offline parse action, host command action) that the BMI can create. A task is created when the specific port needs to allocate a TNUM. The task is finished when the resources that are associated with it are freed (DMA, internal buffers) and the processing which is associated with it is finished. The total number of BMI initiated concurrent tasks is shared between the BMI ports. This number cannot exceed the total number of tasks supported by the FMan.0x00 Reserved 0x01 Reserved 0x02 Reserved 0x03 BMI max opened outstanding tasks is 4. 0x04 BMI max opened outstanding tasks is 5. ... 0x3F BMI max opened outstanding tasks is 64. ... 0x7F BMI max opened outstanding tasks is 128.
16-25	—	Reserved
26-31	—	Reserved

#### 5.5.4.4.4 Interrupt Event Register (FMBM\_IEVR)

The FMBM\_IEVR, shown in [Figure 5-40](#), specifies BMI events that may have caused an interrupt. If a BMI interrupt is asserted, this register is read in order to detect the interrupt initiator. The interrupt bits reflect the events regardless of the state of the corresponding FMBM\_IER bits. The interrupt signal asserts if at least one enabled interrupt event is active. The interrupt destination is the system PIC.

Note that the BMI interrupt is used for error reporting. There are no functional interrupt reported by the BMI.

Offset 0x0\_0020 (FMBM\_IEVR)

	0	1	2	3	4																														Access: w1c
R	SPEC	LEC	STEC	DEC		—																		31											
W	w1c	w1c	w1c	w1c		All zeros																													

Reset

**Figure 5-40. Interrupt Event Register (FMBM\_IEVR)**

**Table 5-55. FMBM\_IEVR Field Descriptions**

Bits	Name	Description
0	SPEC	Storage Profile ECC Error detected 0 - No uncorrectable ECC errors were detected in pipeline internal RAM transaction 1 - Uncorrectable ECC error was detected in pipeline internal RAM transaction The BMI asserts interrupt if uncorrectable ECC error was detected in pipeline internal RAM transaction and the corresponding bit in FMBM_IER is set. Write 1 clears the interrupt.
1	LEC	Linked List RAM ECC Error. 0 - No uncorrectable ECC errors were detected in 'linked list internal RAM' transaction 1 - Uncorrectable ECC error was detected in 'linked list internal RAM' transaction The BMI asserts interrupt if uncorrectable ECC error was detected in 'linked list internal RAM' transaction and the corresponding bit in FMBM_IER is set. Write 1 clears the interrupt.
2	STEC	Statistics Count RAM ECC Error. 0 - No uncorrectable ECC errors were detected in 'statistics count internal RAM' transaction 1 - Uncorrectable ECC error was detected in 'statistics count internal RAM' transaction The BMI asserts interrupt if uncorrectable ECC error was detected in 'statistics count internal RAM' transaction and the corresponding bit in FMBM_IER is set. Write 1 clears the interrupt.
3	DEC	Dispatch RAM ECC Error. 0 - No uncorrectable ECC errors were detected in 'Dispatch internal RAM' transaction 1 - Uncorrectable ECC error was detected in 'Dispatch internal RAM' transaction BMI asserts an interrupt if uncorrectable ECC error is detected in 'Dispatch internal RAM' transaction and the corresponding bit in FMBM_IER is set. Write 1 clears the interrupt.
4-31	—	Reserved.

#### 5.5.4.4.5 Interrupt Enable Register (FMBM\_IER)

The interrupt enable register (FMBM\_IER) specifies enable bits to the BMI interrupt events.

Offset 0x0\_0024 (FMBM\_IER)

	0	1	2	3	4																														Access: Read/Write
R	SPECE	LECE	SECE	DECE		—																		31											
W						All zeros																													

Reset

**Figure 5-41. Interrupt Enable Register (FMBM\_IER)**

**Table 5-56. FMBM\_IER Field Descriptions**

Bits	Name	Description
0	SPECE	Storage Profile ECC Error Enable 0 - PEC Interrupt is disabled. 1 - PEC Interrupt is enabled.
1	LECE	Linked List RAM ECC Error Enable. 0 - LEC Interrupt is disabled. 1 - LEC Interrupt is enabled.
2	SECE	Statistics Count RAM ECC Error Enable. 0 - Statistics Count RAM ECC Error Interrupt is disabled. 1 - Statistics Count RAM ECC Error Interrupt is enabled.
3	DECE	Dispatch RAM ECC Error Enable. 0 - DEC Interrupt is disabled. 1 - DEC Interrupt is enabled.
4-31	—	Reserved.

#### 5.5.4.4.6 Interrupt Force Register (FMBM\_IFR)

The FMBM\_IFR can be used to force an interrupt event, regardless of the logic conditions associated with the event. This is a debug/verification assist register.

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires debug.

**Figure 5-42. Interrupt Force Register (FMBM\_IFR)**

Table 5-57 describes the FMBM\_IFR fields.

**Table 5-57. FMBM\_IFR Field Descriptions**

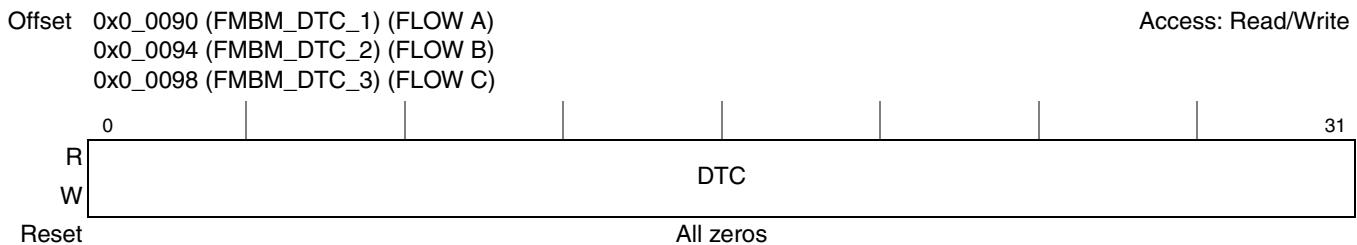
Bits	Name	Description
0	SPECF	Storage Profile ECC Error Force 0 - PEC Interrupt is not forced. 1 - PEC Interrupt is forced.
1	LECF	Linked List RAM ECC Error Force. 0 - LEC Interrupt is not forced. 1 - LEC Interrupt is forced.
2	SECF	Statistics Count RAM ECC Error Force. 0 - Statistics Count RAM ECC Error Interrupt is not forced. 1 - Statistics Count RAM ECC Error Interrupt is forced.

**Table 5-57. FMBM\_IFR Field Descriptions (continued)**

Bits	Name	Description
3	DECForce	Dispatch RAM ECC Error Force. 0 - DEC Interrupt is not forced. 1 - DEC Interrupt is forced.
4-31	—	Reserved.

#### 5.5.4.4.7 Debug Trap Counter Register (FMBM\_DTC)

The FMBM\_DTC register counts the number of successful trap events, including bypass. There are three counters, one per debug flow.



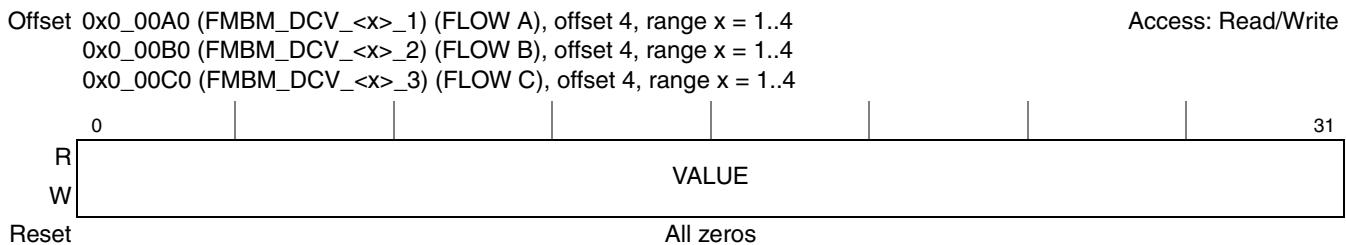
**Figure 5-43. Debug Trap Counter Register (FMBM\_DTC)**

**Table 5-58. FMBM\_DTC Field Descriptions**

Bits	Name	Description
0-31	DTC	Debug Trap Event Count. Counts the number of successful trap events, including bypass. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

#### 5.5.4.4.8 Debug Compare Value Register (FMBM\_DCV)

The FMBM\_DCV register contains the value to be compared against the frame FD. Because the FD size is 16 bytes, there are 4 compare registers and 4 compare masks registers per debug flow. The registers are designated by two dimensional indices where x stands for the part of the 64 bytes to be compared and y stands for the flow. Note that x ranges from 1 to 4 meaning that the first register compares against the first 4 bytes of the FD, the second compare against the seconds group of 4 bytes of the FD and so on. See [Section 5.5.6.22, “Debug Capabilities,”](#) for more details about debug capabilities. Note that debug trap compare value and mask registers are common to all ports, while the debug configuration registers are defined per port.



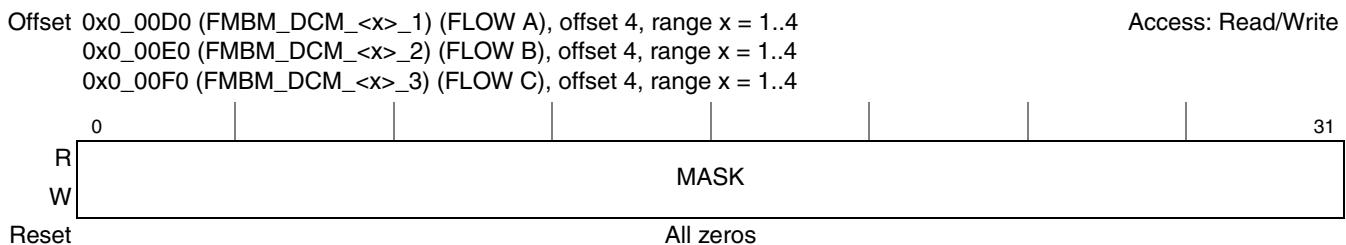
**Figure 5-44. Debug Compare Value Register (FMBM\_DCV)**

**Table 5-59. FMBM\_DCV Field Descriptions**

Bits	Name	Description
0-31	VALUE	Compare value Compare is considered matched if (trap comp value & trap mask) equals to the (frame FD and trap mask).

#### 5.5.4.4.9 Debug Compare Mask (FMBM\_DCM)

The FMBM\_DCM register contains the masks to be used when FMBM\_DCV is compared against the frame FD. Because the FD size is 16 bytes, there are four compare registers and four compare masks registers per debug flow. The registers are designated by two dimensional indexes where x stands for the part of the 64 bytes to be compared and y stands for the debug flow (A,B,C). Note that x ranges from 1 to 4 meaning that the first register compares against the first 4 bytes of the FD, the second compare against the second group of 4 bytes of the FD and so on. See [Section 5.5.6.22, “Debug Capabilities,”](#) for more details about debug capabilities. Note that debug trap compare value and mask registers are common to all ports, while the debug configuration registers are defined per port.



**Figure 5-45. Debug Compare Mask (FMBM\_DCM)**

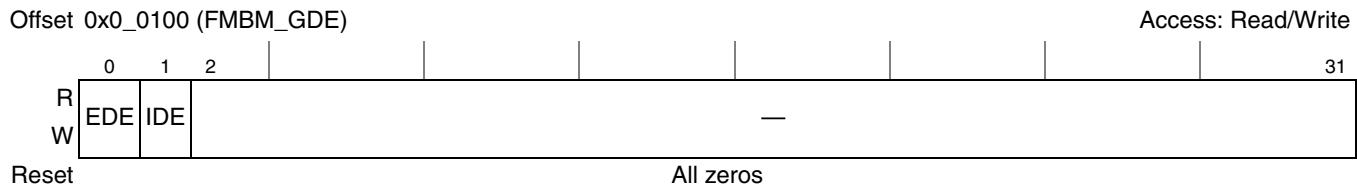
**Table 5-60. FMBM\_DCM Field Descriptions**

Bits	Name	Description
0-31	MASK	Compare Mask. Compare is considered matched if (trap comp value & trap mask) is equal to the (frame FD & trap mask).

#### 5.5.4.4.10 Global Debug Enable (FMBM\_GDE)

This register (and all the debug registers) is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires debug.

The FMBM\_GDE register, controls global debug enable. This register bits controls the debug enable or disable for all BMI ports and the entire FMan. Note that debug can be enabled internally by writing ‘1’ to FMBM\_GDE[IDE] or externally by setting an external debug indication from the SoC provided that it is not masked by FMBM\_GDE[EDE].



**Figure 5-46. Global Debug Enable (FMBM\_GDE)**

Table 5-61 describes the FMBM\_GDE fields.

**Table 5-61. FMBM\_GDE Field Descriptions**

Bits	Name	Description
0	EDE	External Debug Enable. When set, the BMI enables or disables debug logic based on the level of the debug_en input indication. 0 - Debug_en indication is not valid and cannot turn on debug logic. 1 - Debug_en indication is valid, debug logic is enabled if debug_en signal is ‘1’.
1	IDE	Internal Debug Enable. 0 - Debug logic is disabled. 1 - Debug logic is enabled.
2-31	—	Reserved

#### 5.5.4.4.11 Port Parameters Register (FMBM\_PP\_1–63)

The FMBM\_PP register, shown in Figure 5-47, is used to set Rx, Tx and O/H port parameters. There are up to 63 registers, one per port (FMBM\_PP\_1 at 0x0\_0104, FMBM\_PP\_2 at 0x0\_0108.. FMBM\_PP\_63 at 0x0\_01FC).

##### NOTE

EXT field is not valid (treat as reserved) for ports other than Rx ports.

##### NOTE

In FMan\_v3 it is recommended not to modify this register, since the reset values address all configurations.

Inactive ports (meaning that the port is not enabled and not busy) are not accounted in the summary of committed resources, thus enlarging the shared pool for the benefit of the active ports.

The configuration of the parameters in the FMBM\_PP registers can be altered per need during the operation of the system, even while the ports are enabled and moving data. An exception to this is a change of MXT for O/H ports, in which the new written value is lower than the previous. Such change requires that the port is disabled and not busy (FMBM\_OST[BSY] is read ‘0’) before the configuration change operation.

## NOTE

Reserved ports' registers are not implemented. See [Table 5-46](#).

Offset <sup>1</sup>	0x0_0104 (FMBM_PP_<n>)	Access: Read/Write
offset <4*(n-1)>		
range		
n=0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x10,0x11,0x30,0x31;		
R	—	MXT
W	—	EXT
Reset	0 0 n n   n n n n <sup>2</sup>	0 0 0 0   0 0 0 0   0 0 0 0   0   n n n n <sup>3</sup>   0 0 0 0   0 0 0 0   0 0 0 0
		0 1 2   7 8 11 12   15 16 19   20 23 24   27 28   31

**Figure 5-47. Port Parameters Register (FMBM\_PP\_1–63)**

<sup>1</sup> There is one register for each port. n is the PortID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details.

<sup>2</sup> MXT default depend on the port:

In FMan\_v3:

RX 1G – 000011 (4 max open tasks)

RX 10G – 001101 (14 max open tasks)

TX 1G – 000011 (4 max open tasks)

TX 10G – 001101 (14 max open tasks)

O - 000101 (6 max open tasks)

H - (PortID=0x02) - 00000 (1 task)

<sup>3</sup> MXD default depends on the port:

In FMan\_v3:

RX 1G – 0001 (2 max DMAs)

RX 10G – 0111 (8 max DMAs)

TX 1G – 0010 (3 max DMAs)

TX 10G – 1011 (12 max DMAs)

O - 0011 (4 max DMAs)

H (PortID=0x02) - 0001 (2 max DMAs)

See “FMan Hardware Ports in Devices” for exact list of which ports are supported.

**Table 5-62. FMBM\_PP Field Descriptions**

Bits	Name	Description
0-1	—	Reserved.
2-7	MXT	MaXimum number of Tasks. The maximum number of concurrent tasks that the port can run. Each task had a TNUM associated with it. The FMan automatically creates tasks (by allocating TNUMs) at the beginning of flows, and terminates tasks (by releasing TNUMs, after having released other resources) at the end of flows. The sum of the tasks on all ports should be <u>less or equal</u> to the total number of BMI initiated concurrent tasks as defined in FMBM_CFG2[TNTSKS]. In addition, the sum tasks on all O/H ports should be <u>less</u> than the total number of tasks as defined in FMBM_CFG2[TNTSKS]. For FMan_v3 the recommended values are the default reset values listed above.
8-11	—	Reserved

**Table 5-62. FMBM\_PP Field Descriptions (continued)**

Bits	Name	Description
12-15	EXT	<p>Extra number of tasks.</p> <p>The BMI allocates the extra tasks to the port (beyond the number configured in MXT) if the FIFO threshold configured in FMBM_RFP[FTH] is exceeded.</p> <p>Note that extra tasks are not committed; there are no extra tasks allocation if the total number of BMI initiated concurrent tasks has reached its limit, as defined in FMBM_CFG2[TNTSKS]. The excessive allocation also depends on the availability of tasks in the shared pool, meaning that the allocation is not on the expense of the committed resources of the other ports.</p> <p>0000 - No Extra tasks allocated.      0001 - 1 Extra tasks may be allocated.      ...      1000 - 8 Extra tasks may be allocated.      1001 - 1111 - Reserved.      Recommended value is 0x0.</p>
16-19	—	Reserved.
20-23	MXD	<p>Max DMA.</p> <p>Number of maximum outstanding external bus access (DMA) requests allowed to the port.</p> <p>0000 - 1 Max DMAs. Recommended value for 1 Gbps ports (Rx and Tx).      0001 - 2 Max DMAs. Recommended value for 2.5 Gbps Rx ports. Also recommended value for O/H ports in FMan_v3.      ...      0010 - 3 Max DMAs. Recommended value for 2.5 Gbps Tx ports      ...      0111 - 8 Max DMAs. Recommended value for 10 Gbps Rx ports      ...      1011 - 12 Max DMAs. Recommended value for 10 Gbps Tx ports      ...      1111 - 16 Max DMAs</p> <p><b>Note:</b> In FMan_v3: The sum of MXD on all ports should not exceed the size of the DMA command queue (84 entries).</p> <p><b>Note:</b> In FMan_v3: For Offline ports the minimum value of this field is 0001 (2 max DMAs).</p>
24-27	—	Reserved.
28-31	—	Reserved

#### 5.5.4.12 Port FIFO Size Register (FMBM\_PFS\_1–63)

The FMBM\_PFS register, shown in [Figure 5-48](#), is used to set Rx, Tx and O/H port FIFO parameters. There are 63 registers, one per port (FMBM\_PFS\_1 at 0x0\_0204, FMBM\_PFS\_2 at 0x0\_0208, FMBM\_PFS\_63 at 0x0\_02FC). Note that EXBS is not valid (treat as reserved) for ports other than Rx ports.

Inactive ports (meaning that the port is disabled and no frames are being received or transmitted) are not accounted in the summary of committed resources, thus enlarging the shared pool for the benefit of the active ports.

The configuration of the parameters in the FMBM\_PFS registers can be altered per need during the operation of the system, even while the ports are enabled and moving data. An exception to this is a change of IFSZ in which the new written value is lower than the previous, at Tx and O/H ports. Such change

requires that the port is disabled and not busy (FMBM\_TST[BSY] or FMBM\_OST[BSY] is read ‘0’) before the configuration change operation.

Note that reading from reserved ports’ registers does not result in consistent data. See [Table 5-46](#).

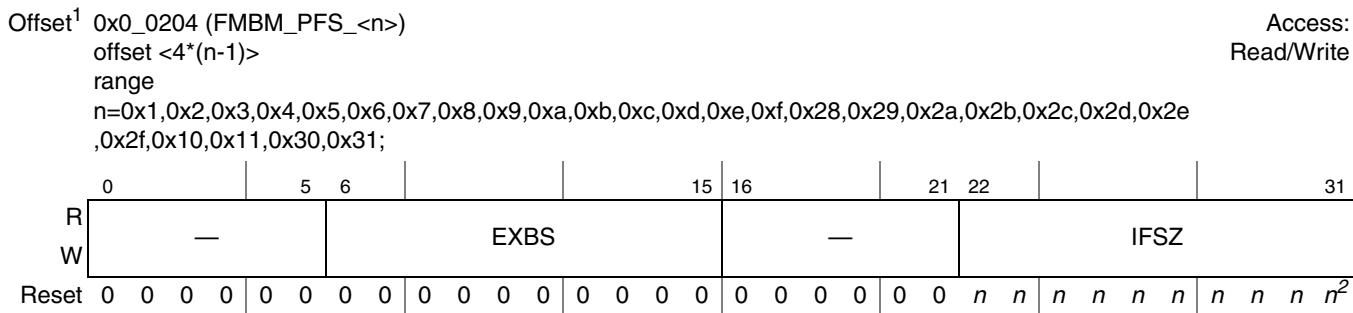
Note that the programming of this register must meet certain rules, in order to avoid deadlock conditions and frames dropping. See details in [Section 5.5.6.18, “Internal FIFO Configuration Requirements.”](#)

#### **NOTE**

In FMan\_v3 this register is for internal use. There is no need to program this register, since the reset values address all configurations.

Note that it is recommended to allocate the Host command port to PortID=0x2, with smaller internal FIFO allocation.

[Table 5-64](#) describes the FMBM\_PFS fields.



**Figure 5-48. Port FIFO Size Register (FMBM\_PFS\_1-63)**

<sup>1</sup> There is one register for each port. n is the Rx PortID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details.

<sup>2</sup> Internal FIFO size default value

In FMan\_v3:

Rx 1/2.5G - 00\_0011\_0001

Rx 1G - 00\_0010\_1011

Rx 10G - 00\_0101\_1111

Tx 1G - 00\_0011\_0001

Tx 10G - 00\_0011\_1111

O/H - 00\_0011\_0001 except for PortID=0x2 which is initialized with 00\_0000\_1001

See “[Table 5-15](#)” for exact list of which ports are supported.

**Table 5-63. FMBM\_PFS Reset Values in FMan\_v3**

Reg Name	Reset Value
FMBM_PFS_3, FMBM_PFS_4, FMBM_PFS_5, FMBM_PFS_6, FMBM_PFS_7	0000_0000_0000_0000_0000_0011_0001
FMBM_PFS_2	0000_0000_0000_0000_0000_0000_1001
FMBM_PFS_8, FMBM_PFS_9, FMBM_PFS_10, FMBM_PFS_11, FMBM_PFS_12, FMBM_PFS_13, FMBM_PFS_14, FMBM_PFS_15	0000_0000_0000_0000_0000_0011_0001
FMBM_PFS_40, FMBM_PFS_41, FMBM_PFS_42, FMBM_PFS_43, FMBM_PFS_44, FMBM_PFS_45, FMBM_PFS_46, FMBM_PFS_47	0000_0000_0000_0000_0000_0011_0001
FMBM_PFS_16, FMBM_PFS_17	0000_0000_0000_0000_0000_0101_1111
FMBM_PFS_48, FMBM_PFS_49	0000_0000_0000_0000_0000_0011_1111
else	0000_0000_0000_0000_0000_0000_0000

**Table 5-64. FMBM\_PFS Field Descriptions**

Bits	Name	Description
0-5	—	Reserved.
6-15	EXBS	<p>Excessive Buffer Size.</p> <p>When Rx port FIFO size exceeds its max value configured by IFSZ, the BMI may allocate additional internal buffers configured by EXBS to avoid frames discard. The sum of EXBS + IFSZ, on all ports, may exceed the total buffer pool size configured by FMBM_CFG1[FBPS].</p> <p>The excessive buffer size is not guaranteed and is not allocated if the total buffers in use exceed the total buffer pool size configured by FMBM_CFG1[FBPS]. The excessive allocation also depends on the availability of buffers in the shared pool, meaning that the allocation is not on the expense of the committed resources of the other ports.</p> <p>The size is in 256 bytes granularity.</p> <ul style="list-style-type: none"> <li>0x000 - Do not allocate excessive buffers.</li> <li>0x001 - Allocate 1 excessive buffer.</li> <li>...</li> <li>0x0FF - Allocate 256 excessive buffers.</li> <li>...</li> <li>0x3FF - Allocate 1023 excessive buffers.</li> </ul> <p>Note: Value of excessive buffer size is limited by FMan memory size left. Recommended value is 0x0.</p>
16-21	—	Reserved.
22-31	IFSZ	<p>Internal FIFO Size.</p> <p>The size of port's FIFO committed by the BMI to allocate from the internal buffer pool.</p> <p>The number of buffers (each buffer 256 bytes) allocated to the port FIFO, i.e the size is IFSZ*256 bytes.</p> <p>The sum of IFSZ on all ports should not exceed the total buffer pool size configured by FMBM_CFG1[FBPS].</p> <ul style="list-style-type: none"> <li>0x000 - The port internal FIFO size is 1 buffer.</li> <li>...</li> <li>0x007 - The port internal FIFO size is 8 buffers.</li> <li>...</li> <li>0x02B - The port internal FIFO size is 44 buffers.</li> <li>...</li> <li>0x02F - The port internal FIFO size is 48 buffers.</li> <li>...</li> <li>0x0FF - The port internal FIFO size is 256 buffers.</li> <li>...</li> <li>0x3FF - The port internal FIFO size is 1024 buffers.</li> </ul> <p><b>Note:</b> Values that are bigger than the FMan memory are reserved and should not be used. Refer to <a href="#">Section 5.5.6.18, “Internal FIFO Configuration Requirements”</a> for more details.</p>

#### 5.5.4.4.13 SPICID Register (FMBM\_SPICID\_1–63)

This register is used to initialize the ICID, and to provide port separation of the virtual storage profiles. On Rx and Offline ports, the virtual storage profile offset is calculated as a combination of RSPID from the classification and the base number (SPNUM) field in FMBM\_SPICID register. See [Section 5.5.6.14, “ICID \(Isolation Context Identifier\)”](#) and [Section 5.5.4.1.1, “Storage Profile Virtualization per port \(for Rx and Offline ports\).”](#)

Offset<sup>1</sup> 0x0\_0304 (FMBM\_SPICID\_<n>)  
 offset <4\*(n-1)>  
 range  
 n=0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x28,0x  
 29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x10,0x11,0x30,0x31;

	0	3	4	7	8	9	10	15	16	23	24	31
R	SPBRN	—	SPNUM				—	ICID				
W												
Reset				All zeros								

**Figure 5-49. Storage Profile and Isolation Context Identifier Register (FMBM\_SPICID)**

<sup>1</sup> There is one register for each port. n is the Rx PortID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details.

[Table 5-65](#) describes the FMBM\_SPICID fields.

**Table 5-65. FMBM\_SPICID Field Descriptions**

Bits	Name	Description
0-3	SPBRN	<p>Storage Profile ID Bit Replacement Number.            This field defines the storage group size by selecting the number of most significant bits from the absolute storage profile ID (ASPID) that are replaced by their corresponding SPNUM bits. RSPID is a six bits value which is initialized in FMBM_RFQID, and optionally replaced by Keygen or Custom Classifier.</p> <p>0000 - Virtual storage is {SPNUM[0:7]}; the group contains 1 storage profile.            0001 - Virtual storage is {SPNUM[0:6],RSPID[5]}; the group contains 2 storage profiles.            0010 - Virtual storage is {SPNUM[0:5],RSPID[4:5]}; the group contains 4 storage profiles.            0011 - Virtual storage is {SPNUM[0:4],RSPID[3:5]}; the group contains 8 storage profiles.            0100 - Virtual storage is {SPNUM[0:3],RSPID[2:5]}; the group contains 16 storage profiles.            0101 - Virtual storage is {SPNUM[0:2],RSPID[1:5]}; the group contains 32 storage profiles.            0110 - Virtual storage is {SPNUM[0:1],RSPID[0:5]}; the group contains 64 storage profiles.            0111 – 1111 – reserved</p> <p><b>Note:</b> In some versions of FMan the MSB bits of SPNUM are zero (this depends on the total number of storage profiles supported).</p> <p><b>Note:</b> This field is programmed in Hardware Port Storage profile register (FMBM_SPICID) and is used only when virtual storage profile is enabled. In virtual storage register profile (FMBM_VSPICID) this field is reserved and must be written with zero.</p>
4-9	—	Reserved. write with zeros.
8-15	SPNUM	<p>virtual Storage Profile base NUMber.            This field maps the base profile number where the group of mapped profiles is located. In case of a match the number of SPNUM most significant bits selected by SPBRN is concatenated with the remaining least significant bits from the profile selection (RSPID). The number of storage profile entries in the group is always a power of two.</p> <p><b>Note:</b> The number of bits in this field depends on the total number of storage profiles supported in a given FMan. For example: For 64 virtual storage profiles, the 2 MSB bits are always zero.</p> <p><b>Note:</b> This field is programmed in Hardware Port Storage profile register (FMBM_SPICID) and is used only when virtual storage profile is enabled. In virtual storage register profile (FMBM_VSPICID) this field is reserved and must be written with zero.</p>

**Table 5-65. FMBM\_SPICID Field Descriptions (continued)**

Bits	Name	Description
16-23	—	Reserved.
24-31	ICID	ICID Storage profile isolation context identifier <b>Note:</b> On Offline port and Tx port (normal mode) this field is always overwritten by the value configured in the FD. In Tx port independent mode, this field is used.

### 5.5.4.5 Rx Port Register Descriptions

The following registers are the registers for the Ethernet receive (Rx) port. There is one register (or range of registers) of each type for each Rx port.

#### 5.5.4.5.1 Rx Configuration Register (FMBM\_RCFG)

The FMBM\_RCFG register, shown in [Figure 5-50](#), is used to enable and configure Ethernet Rx port. There is one register per Rx port, numbered 1 to ‘the number of Ethernet Rx Ports’.

Offset<sup>1</sup> 0x0000 (FMBM\_RCFG\_<n>)  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;  
Access: Read/Write

0	EN	—	FDOVR	IM	—	—	—	—	—	—	AM	31
R												W
Reset												
All zeros												

**Figure 5-50. Rx Configuration Register (FMBM\_RCFG)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PVID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map”](#) for details.

[Table 5-66](#) describes the FMBM\_RCFG fields.

**Table 5-66. FMBM\_RCFG Field Descriptions**

Bits	Name	Description
0	EN	Enable. 0 - Port is disabled. If reset during data transfer, the BMI does not receive new frames from the MAC and completes the ongoing data transfer. 1 - When set the port is enabled to receive data. <b>Note:</b> The user may enable and disable the port dynamically. If disabled during receive or transmit, the FMan performs a graceful stop (Rx or Tx), so frames are fully received or transmitted (no truncation occurs). The port is ‘inactive’ once the entire frame has been received or transmitted. <b>Note:</b> If the port is disabled dynamically (EN=0), the user must ensure that other fields in this register maintain the same value as long as FMBM_RST[BSY]=1. Only after FMBM_RST[BSY]=0, the user may change the values of the other fields in this register.
1-5	—	Reserved

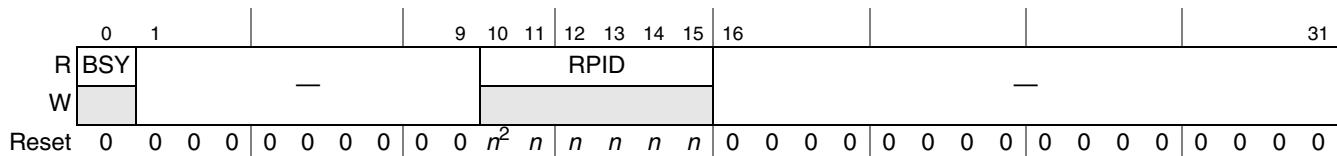
**Table 5-66. FMBM\_RCFG Field Descriptions (continued)**

Bits	Name	Description
6	FDOVR	Frame Discard Override. Set the BMI treatment when frame should be discarded due to erroneous condition detected in the process of the frame reception or processing. Normal mode (IM=0): 0 - Discard the frame. 1 - Enqueue the frame to error queue configured in FMBM_REFQID[EFQID]. Independent mode (IM=1): 0 - Discard the frame, 1 - Ignore errors, frame treatment continue by FMan controller (set error code in FD status word). This bit is useful for debug sessions, where discard decision can be overwritten by programming this bit.
7	IM	Independent Mode. When this bit is set the BMI skips external buffers allocation and DMA transfers as done in normal mode. 0 - Normal mode. Frame is received from MAC. The BMI allocates external buffers and initiate DMA transfers. 1 - Independent Mode. Frame is received from MAC, the BMI does not allocate external buffers and does not issue any DMA transfers.
8-15	—	Reserved
16-24	—	Reserved.
25	AM	Accumulate mode 0 - Normal mode 1 - Accumulate mode. In this mode the FMan does not allocate buffers from the BMan, does not write frames to external memory, and does not enqueue the FD into the QMan. In this mode the frame frames are accumulated in the FMan internal memory (FIFO). This mode is used in devices that support low power mode, when the device is in low power mode (only the FMan is powered).
26-31	—	Reserved.

### 5.5.4.5.2 Rx Status Register (FMBM\_RST)

The FMBM\_RST register, shown in [Figure 5-51](#), is used to read status of Rx port.

Offset<sup>1</sup> 0x0004 (FMBM\_RST\_<n>)  
Access: Read only  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;



**Figure 5-51. Rx Status Register (FMBM\_RST)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

<sup>2</sup> RPID reset value is set to the Rx port ID (PortID).

**Table 5-67** lists the FMBM\_RST reset values. Note that in this table the register index is marked in decimal format

**Table 5-67. FMBM\_RST Reset Values**

Reg Name	Reset Value
FMBM_RST_8	0000_0000_0000_1000_0000_0000_0000_0000
FMBM_RST_9	0000_0000_0000_1001_0000_0000_0000_0000
FMBM_RST_10	0000_0000_0000_1010_0000_0000_0000_0000
FMBM_RST_11	0000_0000_0000_1011_0000_0000_0000_0000
FMBM_RST_12	0000_0000_0000_1100_0000_0000_0000_0000
FMBM_RST_13	0000_0000_0000_1101_0000_0000_0000_0000
FMBM_RST_14	0000_0000_0000_1110_0000_0000_0000_0000
FMBM_RST_15	0000_0000_0000_1111_0000_0000_0000_0000
FMBM_RST_16	0000_0000_0001_0000_0000_0000_0000_0000
FMBM_RST_17	0000_0000_0001_0001_0000_0000_0000_0000

**Table 5-68** describes the FMBM\_RST fields.

**Table 5-68. FMBM\_RST Field Descriptions**

Bits	Name	Description
0	BSY	Busy. BSY is asserted while frames are processed in the port. When Enable is cleared, BSY should be read in order to detect that port is not busy. 0 - Rx port is not busy. 1 - Rx port is busy.
1-9	—	Reserved
10-15	RPID	Rx Port ID. This is the hardware port ID associated with this register.
16-31	—	Reserved

#### 5.5.4.5.3 Rx DMA Attributes Register (FMBM\_RDA)

The FMBM\_RDA register, shown in [Figure 5-52](#), is used to set DMA attributes to be used for the port. Note that read operations always generate cacheable/no allocate, and write of payload besides the frame header always generates cacheable/no stash attributes to the internal SoC interconnect.

Offset<sup>1</sup> 0x0008 (FMBM\_RDA\_<n>)  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;  
Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12																	31
R	SWAP	ICC	FHC	SGC	—	WOPT											—													
W	All zeros																													
Reset																														

**Figure 5-52. Rx DMA Attributes Register (FMBM\_RDA)**

<sup>1</sup> There is one register for each Rx port. n is the Rx Poid. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-69](#) describes the FMBM\_RDA fields.

**Table 5-69. FMBM\_RDA Field Descriptions**

Bits	Name	Description
0-1	SWAP	Swap payload data. 00 - No swap, transfer data as is. 01-11 Reserved
2-3	ICC	IC write cache attributes. Used when DMA copies IC from FMan memory to external memory 00 - Cacheable, no Allocate (No Stashing) 01 - Cacheable and Allocate (Stashing on) 10, 11 - Reserved.
4-5	FHC	Frame Header write cache attributes. Used when DMA copies frame header from FMan memory to external memory 00 - Cacheable, no Allocate (No Stashing). 01 - Cacheable and Allocate (Stashing on). 10, 11 - Reserved.
6-7	SGC	Scatter gather write cache attributes. Used when DMA copies the scatter/gather list from FMan memory to external memory 00 - Cacheable, no Allocate (No Stashing). 01 - Cacheable and Allocate (Stashing on). 10, 11 - Reserved.
8-9	—	Reserved

**Table 5-69. FMBM\_RDA Field Descriptions (continued)**

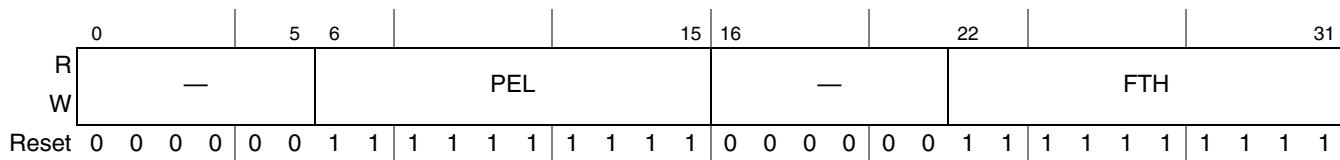
Bits	Name	Description
10-11	WOPT	<p>Optimize on write 00 - No optimization is done.</p> <p>01 - Write optimization is enabled, Write optimization is performed by extending the write transaction of the frame payload at the head and/or tail as needed to achieve optimal bus transfers. Extra bytes are not added to IC or the scatter/gather list.</p> <p>Optimization done on the head of the payload: The first write is pre-extended to be on 16 bytes aligned block. In FMan_v3, the 'extra' bytes have undefined values;</p> <p>Optimization done on the tail of the payload: The last write is extended to be on 64 bytes aligned block. 'extra' bytes are undefined.</p> <p>10, 11 - Reserved.</p> <p><b>Note:</b> In FMan_v3 write optimization on the tail of the frame is aligned to 64 bytes. See DMA chapter for additional information regarding this feature.</p>
12-31	—	Reserved

#### 5.5.4.5.4 Rx FIFO Parameters Register (FMBM\_RFP)

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug.

The FMBM\_RFP register, shown in [Figure 5-53](#), is used to set Rx port FIFO parameters.

Offset<sup>1</sup> 0x000c (FMBM\_RFP\_<n>)  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;  
Access: Read/Write



**Figure 5-53. Rx FIFO Parameters Register (FMBM\_RFP)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-70 describes the FMBM\_RFP fields.

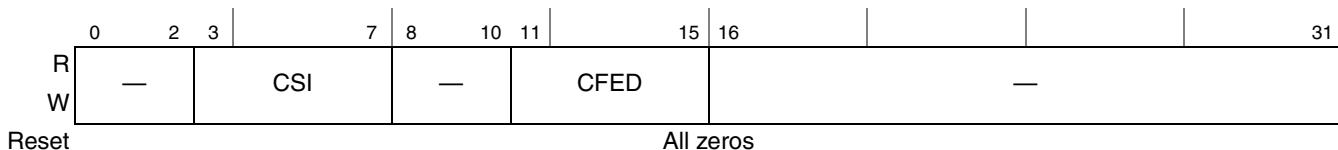
**Table 5-70. FMBM\_RFP Field Descriptions**

Bits	Name	Description
0-5	—	Reserved.
6-15	PEL	<p>Priority Elevation Level. If the total number of buffers which are currently in use and associated with the specific Rx port exceed the amount specified in PEL, the BMI signals the DMA module to elevate its priority. The size is in 256 bytes granularity. 0x000 - Priority Elevation Level is 1*256bytes 0x001 - Priority Elevation Level is 2*256bytes ... 0x0FF - Priority Elevation Level is 256*256bytes. ... 0x3FF- Priority Elevation Level is 1024*256bytes</p>
16-21	—	Reserved.
22-31	FTH	<p>FIFO threshold. Set the threshold of the FIFO fill level. If the total number of buffers which are currently in use and associated with the specific Rx port exceed this threshold, the BMI signals the MAC to send a pause frame over the link. Pause time is programmed in the MAC registers. In addition, exceeding the threshold allows the port to use excess dma budget and excess task numbers budget, as defined in FMBM_PP[EXD] and FMBM_PP[EXT]. In FMan_v3, this restriction does not apply. In FMan_v3 excessive tasks may be used regardless of this threshold Note that buffers in use include buffers for frames received but not copied to external memory, as well as two buffers per frame for frames waiting to be processed or waiting to be enqueued to a frame queue. The size is in 256 bytes granularity. 0x000 - FIFO threshold is 1*256bytes 0x001 - FIFO threshold is 2*256bytes ... 0x0FF - FIFO threshold is 256*256bytes. ... 0x3FF- FIFO threshold is 1024*256bytes</p>

#### 5.5.4.5.5 Rx Frame End Data Register (FMBM\_RFED)

The FMBM\_RFED register, shown in Figure 5-54, is used to define Rx port action on frame's last bytes.

Offset<sup>1</sup> 0x0010 (FMBM\_RFED\_<n>) Access: Read/Write  
 offset <0x1000\*n>  
 range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;



**Figure 5-54. Rx Frame End Data Register (FMBM\_RFED)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-71 describes the FMBM\_RFED fields.

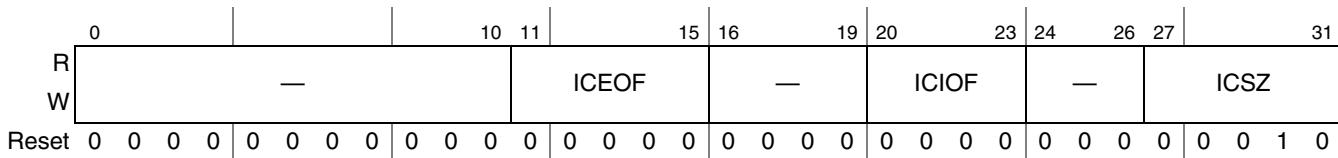
**Table 5-71. FMBM\_RFED Field Descriptions**

Bits	Name	Description
0-2	—	Reserved
3-7	CSI	<p>Checksum Ignore. Set the number of bytes from end of frame ignored in frame one's complement running sum calculation (for TCP/UDP checksum). Used to eliminate data located in end of frame from sum calculation.</p> <p>00000 - Calculate running sum for the whole frame. 00001 - Eliminate last 1bytes from frame running sum. 00010 - Eliminate last 2bytes from frame running sum. ... 10000 - Eliminate last 16 bytes from frame running sum. 10001 - 11111 - Reserved.</p> <p><b>Note:</b> Bytes chopped from the end of the frame (as defined in CFED field in this register) are not included in the running sum calculation, meaning that the number of bytes to ignore is in addition to the number of bytes to chop, if such exists.</p>
8-10	—	Reserved
11-15	CFED	<p>Chop Frame's End Data. Set the number of bytes chopped from end of received frame. Used to remove data located in end of frame.</p> <p>00000 - Do not remove data from end of frame. 00001 - Remove last 1byte from end of frame. 00010 - Remove last 2bytes from end of frame. ... 00100 - Remove last 4bytes from end of frame. Recommended value. ... 10000 - Remove last 16 bytes from end of frame. 10001 - 11111 - Reserved.</p> <p><b>Note:</b> if the result of (frame length before chop - CFED) is less than 14 bytes, the chop operation is not executed.</p> <p><b>Note:</b> With the recommended value, make sure that the corresponding Ethernet MAC is configured to leave the FCS in the frame.</p> <p><b>Note:</b> The sum of CSI and CFED must be limited to 16 bytes. If their configured sum is more than 16 bytes the effective CSI value is reduced such that their sum is 16 bytes, meaning: Effective-CSI = min (CSI, 16-CFED).</p>
16-31	—	Reserved

### 5.5.4.5.6 Rx Internal Context Parameters (FMBM\_RICP)

The FMBM\_RICP register, shown in [Figure 5-55](#), is used to configure internal context parameters. Those parameters supply the mechanism to indicate to the BMI which portion of the IC block should be copied to external memory once the processing of the frame was finished.

Offset<sup>1</sup> 0x0014 (FMBM\_RICP\_<n>)  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;  
Access: Read/Write



**Figure 5-55. Rx Internal Context Parameters (FMBM\_RICP)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-72](#) describes the FMBM\_RICP fields.

**Table 5-72. FMBM\_RICP Field Descriptions**

Bits	Name	Description
0-10	—	Reserved
11-15	ICEOF	Internal Context External Offset. Specifies the offset in the external buffer to which the internal context is transferred, 16-byte granularity. The offset is added to the address that is written in the frame descriptor. 00000 - IC is transferred to offset 0x0 in external buffer. 00001 - IC is transferred to offset 0x10 in external buffer. ... 11111 - IC is transferred to offset 0x1F0 in external buffer.
16-19	—	Reserved
20-23	ICIOF	Internal Context Internal Offset. Specifies the offset within the internal context from which data is transferred to external buffer, 16-byte granularity. Together with the ICSZ, the ICIOF is used to specify the internal context segment transferred to the external buffer. The sum of ICIOF and ICSZ should not cross the IC block boundary (256 bytes from the beginning of the block). 0000 - IC data to be transferred is from the beginning of the IC block, no offset. 0001 - IC data to be transferred is from offset 0x10 from the beginning of the IC block. ... 1111 - IC data to be transferred is from offset 0xF0 from the beginning of the IC block.

**Table 5-72. FMBM\_RICP Field Descriptions (continued)**

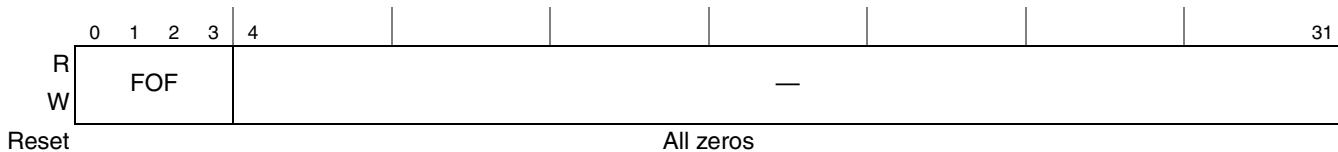
Bits	Name	Description
24-26	—	Reserved
27-31	ICSZ	Internal Context copy Size. Specifies the size of the internal context transferred to the external buffer. Together with ICIOF used to specify the internal context segment transferred to the external buffer. 16 bytes granularity. (up to 256 bytes). 00000 - IC segment is not transferred. 00001 - IC segment to be transferred size is 16 bytes. 00010 - IC segment to be transferred size is 32 bytes. ... 10000 - IC segment to be transferred size is 256 bytes. 10001 - 11111 - reserved.

#### 5.5.4.5.7 Rx Internal Margins Register (FMBM\_RIM)

The FMBM\_RIM register, shown in [Figure 5-56](#), is used to configure internal margins. See [Section 5.5.6.4, “Internal and External Margins.”](#) In most systems, this functionality is not needed, therefore this register contains a value of zero.

This register needs to be programmed with a value different than zero, if the FMan controller is required to add or remove content to the received frame header. Note that if content is added or removed the frame offset in FD is updated by the FMan controller. Note that if frame header manipulation is desired the user must take this into account in the programming of FMBM\_REBM[BSM] and leave room to frame offset to grow, and not exceed the maximum of 511. The need for this register being different than zero is documented in the FMan controller chapter.

Offset<sup>1</sup> 0x0018 (FMBM\_RIM\_<n>)  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;  
Access: Read/Write



**Figure 5-56. Rx Internal Margins Register (FMBM\_RIM)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-73 describes the FMBM\_RIM fields.

### **Table 5-73. FMBM\_RIM Field Descriptions**

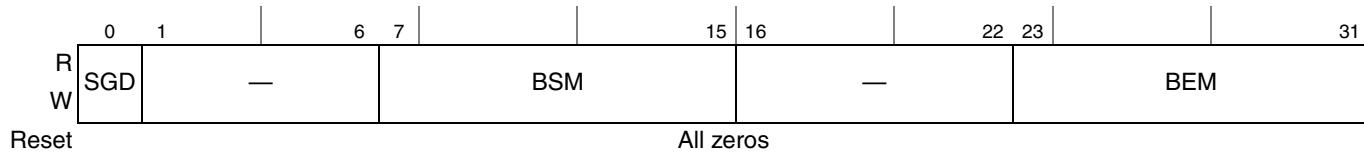
Bits	Name	Description
0-3	FOF	<p>Frame Offset. Specify the offset in the first buffer in which frame is written. 16 bytes granularity.</p> <ul style="list-style-type: none"> <li>0000 - Frame is written to first buffer start address.</li> <li>0001 - Frame is written to first buffer offset 0x10 (16 bytes).</li> <li>0010 - Frame is written to first buffer offset 0x20 (32 bytes).</li> <li>0011 - Frame is written to first buffer offset 0x30 (48 bytes).</li> <li>0100 - Frame is written to first buffer offset 0x40 (64 bytes).</li> <li>0101 - Frame is written to first buffer offset 0x50 (80 bytes).</li> <li>0110 - Frame is written to first buffer offset 0x60 (96 bytes).</li> <li>0111 - Frame is written to first buffer offset 0x70 (112 bytes).</li> <li>1000 - Frame is written to first buffer offset 0x80 (128 bytes).</li> <li>1001 - Frame is written to first buffer offset 0x90 (144 bytes).</li> <li>1010 - Frame is written to first buffer offset 0xa0 (160 bytes).</li> <li>1011 - Frame is written to first buffer offset 0xb0 (176 bytes).</li> <li>1100 - Frame is written to first buffer offset 0xc0 (192 bytes).</li> <li>1101 - 1111 - Reserved.</li> </ul> <p><b>Note:</b> Including the FOF, all frame headers parsed by the FMan parser, must reside in the first internal buffer (256B). For example, if the FMan parser examines headers in the first 90 bytes of the frame, the maximum value of FOF is 1010 (for 160 bytes) since <math>256-90=166</math>.</p> <p>See <a href="#">Section 5.5.6.4.2, “External Margins”</a> and <a href="#">Section 5.5.6.4.3, “External memory accesses optimization”</a> for more details.</p>
4-31	—	Reserved

#### 5.5.4.5.8 Rx External Buffer Margins Register (FMBM\_REBM)

The FMBM\_REBM register, shown in Figure 5-57, configures the additional size (to frame size) allocated at start and end of external buffer. See Section 5.5.6.4, “Internal and External Margins,” for a detailed description.

Offset<sup>1</sup> 0x001c (FMBM\_REBM\_<n>)  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc

Access: Read/Write



**Figure 5-57. Rx External Buffer Margins Register (FMBM\_REBM)**

<sup>1</sup> There is one register for each Rx port.  $n$  is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

[Table 5-74](#) describes the FMBM\_REBM fields.

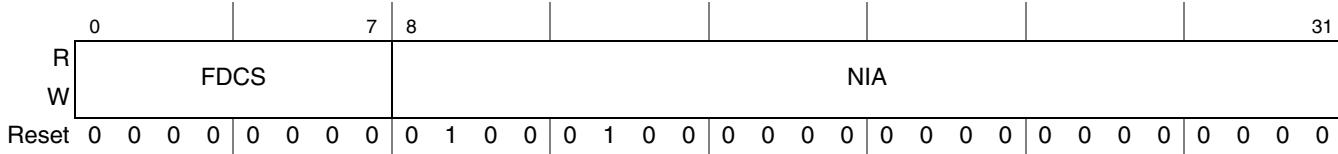
**Table 5-74. FMBM\_REBM Field Descriptions**

Bits	Name	Description
0	SGD	Scatter Gather Disable. 0 - If BMI cannot store the frame in a single buffer it may select a buffer of smaller size and store the frame in scatter gather (S/G) buffers (i.e. S/G is enabled). 1 – Scatter gather format is not enabled for frame storage. If BMI cannot store the frame in a single buffer, the frame is discarded.
1-6	—	Reserved.
7-15	BSM	Buffer Start Margin. Specify the size, in the allocated external buffer, added to the frame start location. 1 byte granularity up to 511 bytes. 0x000 - No extra size, in addition to frame size, is allocated at beginning of external buffer. 0x001 - 1 byte extra size, in addition to frame size, is allocated at beginning of external buffer. ... 0x1FF - 511 bytes extra size, in addition to frame size, is allocated at beginning of external buffer. See <a href="#">Section 5.5.6.4.2, “External Margins”</a> and <a href="#">Section 5.5.6.4.3, “External memory accesses optimization”</a> for more details.
16-22	—	Reserved.
23-31	BEM	Buffer End Margin. Specify the size, in the allocated external buffer, added to the frame end location. 1 byte granularity up to 511 bytes. 0x000 - No extra size, in addition to frame size, is allocated at the end of external buffer. 0x001 - 1 byte extra size, in addition to frame size, is allocated at the end of external buffer. ... 0x1FF- 511 bytes extra size, in addition to frame size, is allocated at the end of external buffer.

#### **5.5.4.5.9 Rx Frame Next Engine Register (FMBM\_RFNE)**

The FMBM\_RFNE register, shown in Figure 5-58, configures NIA issued by the BMI once the frame has been processed by the BMI. It is also used for setting bits in FD command/status field, if required.

Offset<sup>1</sup> 0x0020 (FMBM\_RFNE\_<n>) Access: Read/Write  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;



**Figure 5-58. Rx Frame Next Engine Register (FMBM\_RFNE)**

<sup>1</sup> There is one register for each Rx port.  $n$  is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-75 describes the FMBM\_RFNE fields.

**Table 5-75. FMBM\_RFNE Field Descriptions**

Bits	Name	Description
0-7	FDCS	<p>Rx FD Status bits 0-7 Set.</p> <p>Setting a bit in FDCS sets the respective bit in the Rx FD Status. Bits which are set by this field are not reset by the FMan hardware, and are reflected in the FD when the frame goes to processing and also when the frame goes to QMan for enqueue.</p> <p>For detailed description refer to <a href="#">Section 5.4.3.2.1, “FD Command/Status Word.”</a></p>
8-31	NIA	<p>Next Invoked Action.</p> <p>When frame is received with no error, the BMI issues this NIA. Default value is set to 0x44_0000, which means that the next module is the hardware parser, and the protocol is Ethernet.</p> <p><b>Note:</b> If the soft parser is invoked, the NIA must be set to 0x44_0000 plus the soft parser address shifted right by one bit (The soft parser address is 2X of the low part 16 bits of the NIA). For example, when FMBM_RFNE is set to 0x44_0020 it will invoke the soft parser at offset 0x40.</p> <p>For detailed description refer to <a href="#">Section 5.3.5, “FMan User-Configurable Pipeline Architecture—Introducing the NIA.”</a></p>

#### **5.5.4.5.10 Rx Frame Attributes Register (FMBM\_RFCA)**

The FMBM\_RFCA register configures initial frame processing attributes. COLOR attribute value is used at FD Status color field initialization in the IC of the frame.

**Figure 5-59. Rx Frame Attributes Register (FMBM\_RFCA)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

**Table 5-76. FMBM RFCA Field Descriptions**

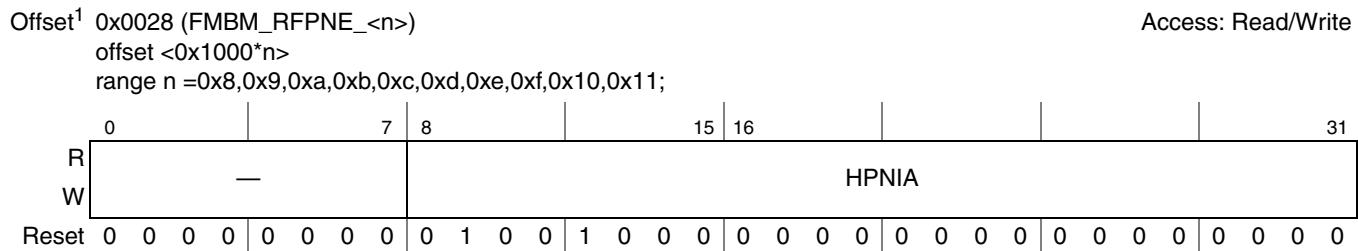
Bits	Name	Description
0	OR	ORder definition. Determine the Order Definition. If OR is set, the FPM attaches the ordering attributes to the task. Later the order restoration may be requested for this task before the task enters order sensitive state (for example, enqueue). 0 - No order definition is needed. 1 - Order definition is needed.
1-3	—	Reserved

**Table 5-76. FMBM\_RFCA Field Descriptions (continued)**

Bits	Name	Description
4-5	COLOR	Default color. Determine the default color of the frame. Color can be altered by frame processing modules if it is needed. 00 - Green 01 - Yellow 10 - Red 11 - Override. Enqueue regardless of RED/WRED/tail drop thresholds.
6-7	SYNC	Synchronization attributes. 00 - Inactive. The synchronization attributes of the task are unchanged. 01 - Reserved. 10 - Synchronization request. The current task is attached to the tail of FPM synchronization queue. 11 - Reserved. <b>Note:</b> The value of 10 is allowed only for ports which are doing dynamic table updates (see <a href="#">Section 5.12, “Frame Manager—FMan Controller”</a> ).
8-9	—	Reserved
10-15	MR	Mode Attributes. These bits define internal modes of operation in the FMan controller and update the MR[0:5] when the first task of the frame is executed on the FMan controller.
16-31	—	Reserved

#### 5.5.4.5.11 Rx Frame Parser Next Engine Register (FMBM\_RFPNE)

The FMBM\_RFPNE register is used to configure NIA used by Hardware Parser when dispatching task to the next module. HPNIA is written to Internal Context.



**Figure 5-60. Rx Frame Parser Next Engine Register (FMBM\_RFPNE)**

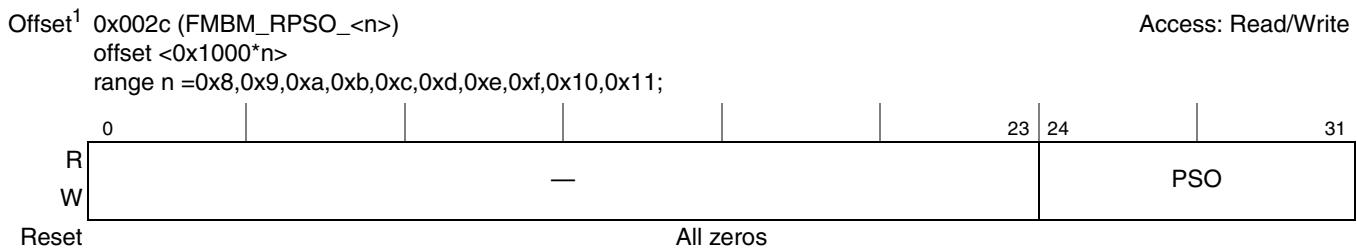
<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map”](#).

**Table 5-77. FMBM\_RFPNE Field Descriptions**

Bits	Name	Description
0-7	—	Reserved.
8-31	HPNIA	Hardware Parser NIA (Next Invoked Action). Set the NIA used by Hardware Parser to jump to the next module (followed by Hardware Parser). HPNIA is written by the BMI to Internal Context. Default value is set to 0x480000, which means that the next module is KeyGen. For detailed description see <a href="#">Section 5.4.4, “Next Invoked Action (NIA)”</a> .

### 5.5.4.5.12 Rx Parsing Start Offset Register (FMBM\_RPSO)

The FMBM\_RPSO register, shown in [Figure 5-61](#), defines an offset in the frame's first buffer. This offset indicates to Hardware Parser the parse start point.



**Figure 5-61. Rx Parsing Start Offset Register (FMBM\_RPSO)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map”](#).

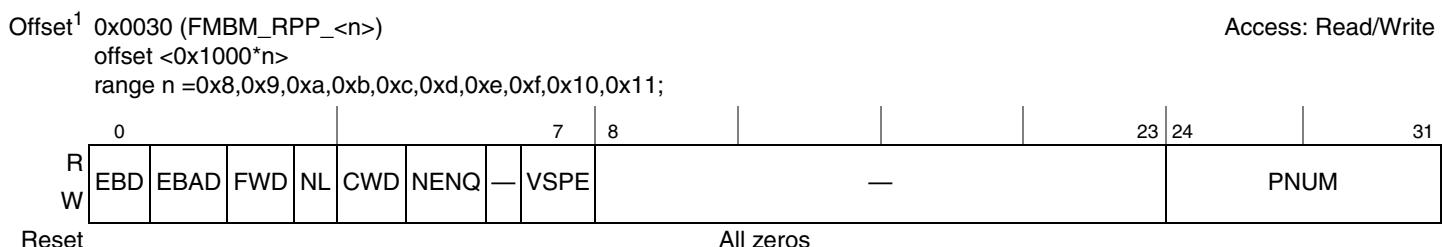
[Table 5-78](#) describes the FMBM\_RPSO fields.

**Table 5-78. FMBM\_RPSO Field Descriptions**

Bits	Name	Description
0-23	—	Reserved.
24-31	PSO	Parsing Start Offset. Specify to Hardware Parser the offset, in first frame's buffer, to parsing start point. 0x00 - Parsing start offset in frame's first buffer is 0. 0x01 - Parsing start offset in frame's first buffer is 1 byte. 0x02 - Parsing start offset in frame's first buffer is 2 bytes. ... 0xFF - Parsing start offset in frame's first buffer is 255 bytes.

### 5.5.4.5.13 Rx Policer Profile Register (FMBM\_RPP)

The FMBM\_RPP register, shown in [Figure 5-62](#), configures the default Policer profile issued by the BMI. The value is written to the Internal Context Action Descriptor ICAD[PNUM] field in the as part of the IC initialization. In addition, this register contains initial values for operational mode bits.



**Figure 5-62. Rx Policer Profile Register (FMBM\_RPP)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map”](#).

[Table 5-79](#) describes the FMBM\_RPP fields.

**Table 5-79. FMBM\_RPP Field Descriptions**

Bits	Name	Description
0	EBD	Bit description are the same as <a href="#">Section 5.4.3.1.1, “Frame Queue Descriptor (FQD) Context A”</a> (using appropriate registers for Rx port).
1	EBAD	
2	FWD	
3	NL	
4	CWD	
5	NENQ	
6	—	
7	VSPE	
8-23	—	Reserved.
24-31	PNUM	Policer Profile. Initiate Policier Profile in Internal Context. For further information refer to <a href="#">Section 5.11, “Frame Manager—Policer.”</a>

#### 5.5.4.5.14 Rx Parse Result Initialization Register (FMBM\_RPRI)

The FMBM\_RPRI register is used to set initial value of the parse result transferred to Hardware Parser when parse request is issued. There are 8 registers per Rx port.

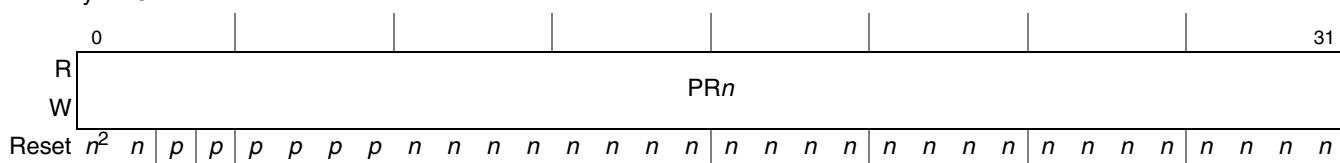
Offset<sup>1</sup> 0x0040 (FMBM\_RPRI\_<n>\_<y>)

## Access: Read/Write

offset <4096\*n+4\*(y-1)>

range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;

$v=1.8$



**Figure 5-63. Rx Parse Result Initialization Register (FMBM\_RPRI)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports,” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

<sup>2</sup> FMBM\_RPRI\_x\_1[2:7] reset value is set in hardware to the PortID. Note the reset values in Table 5-160, “FMBM\_OPRI Reset Values.”

This table lists the FMBM\_RPRI reset values. Note that in this table the register index is marked in decimal format.

**Table 5-80. FMBM\_RPRI Reset Values**

Reg Name	Reset Value
FMBM_RPRI_8_1	0000_1000_0000_0000_0000_0000_0000_0000
FMBM_RPRI_9_1	0000_1001_0000_0000_0000_0000_0000_0000
FMBM_RPRI_10_1	0000_1010_0000_0000_0000_0000_0000_0000
FMBM_RPRI_11_1	0000_1011_0000_0000_0000_0000_0000_0000
FMBM_RPRI_12_1	0000_1100_0000_0000_0000_0000_0000_0000
FMBM_RPRI_13_1	0000_1101_0000_0000_0000_0000_0000_0000
FMBM_RPRI_14_1	0000_1110_0000_0000_0000_0000_0000_0000
FMBM_RPRI_15_1	0000_1111_0000_0000_0000_0000_0000_0000
FMBM_RPRI_16_1	0001_0000_0000_0000_0000_0000_0000_0000
FMBM_RPRI_17_1	0001_0001_0000_0000_0000_0000_0000_0000
FMBM_RPRI_*_2	0000_0000_0000_0000_0000_0000_0000_0000
FMBM_RPRI_*_3	0000_0000_0000_0000_0000_0000_0000_0000
FMBM_RPRI_*_4	0000_0000_0000_0000_0000_0000_0000_0000
FMBM_RPRI_*_5	1111_1111_1111_1111_1111_1111_1111_1111
FMBM_RPRI_*_6	1111_1111_1111_1111_1111_1111_1111_1111
FMBM_RPRI_*_7	1111_1111_1111_1111_1111_1111_1111_1111
FMBM_RPRI_*_8	1111_1111_1111_1111_1111_1111_1111_1111

Table 5-81 describes the FMBM\_RPRI fields.

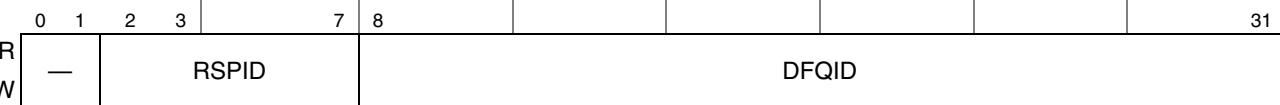
**Table 5-81. FMBM\_RPRI Field Descriptions**

Bits	Name	Description
0–31	PRn	Parse Result. Set the parse result initiated by the BMI when hardware parser services are requested.

#### 5.5.4.5.15 Rx Frame Queue ID Register (FMBM\_RFQID)

The FMBM\_RFQID register, shown in Figure 5-64, configures the default frame queue ID (FQID), and RSPID, in the Internal Context Action Descriptor (ICAD).

Offset<sup>1</sup> 0x0060 (FMBM\_RFQID\_<n>)  
 offset <0x1000\*n>  
 range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;



	0	1	2	3	7	8																												31
R	—	RSPID		DFQID																														
W																					All zeros													
Reset																																		

**Figure 5-64. Rx Frame Queue ID Register (FMBM\_RFQID)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-82](#) describes the FMBM\_RFQID fields.

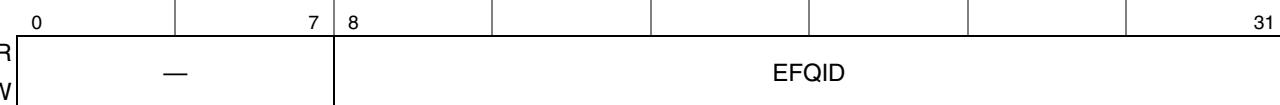
**Table 5-82. FMBM\_RFQID Field Descriptions**

Bits	Name	Description
0-1	—	Reserved.
2-7	RSPID	Default Relative Storage Profile ID The BMI initializes the default storage profile ID to the value of RSPID. This is the relative value of the virtual storage profile offset. The absolute offset is evaluated with fields in the FMBM_SPICID register.
8-31	DFQID	Default Frame Queue ID. The BMI initializes the default frame queue ID to the value set in DFQID. This value is used for all frames, if the parse and classifier are not activated. Note that 0x00_0000 is not a legal value for an FQID.

#### 5.5.4.5.16 Rx Error Frame Queue ID Register (FMBM\_REFQID)

The FMBM\_REFQID register, shown in [Figure 5-65](#), configures the error frame queue ID issued by the BMI.

Offset<sup>1</sup> 0x0064 (FMBM\_REFQID\_<n>)  
 offset <0x1000\*n>  
 range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;



	0		7	8																														31
R	—	EFQID																																
W																					All zeros													
Reset																																		

**Figure 5-65. Rx Error Frame Queue ID Register (FMBM\_REFQID)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

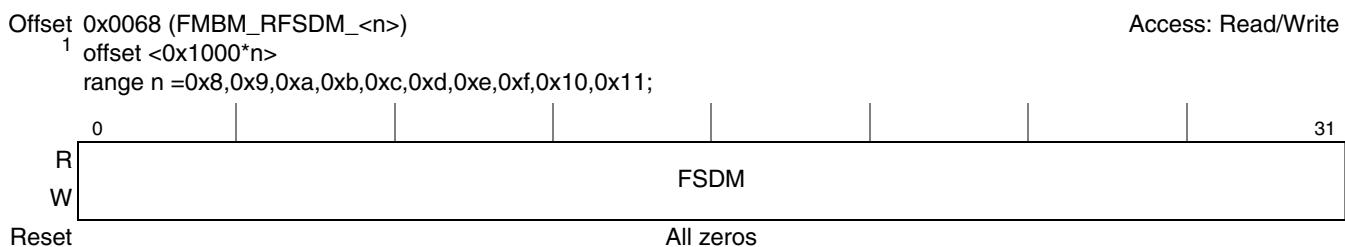
Table 5-83 describes the FMBM\_REFQID fields.

**Table 5-83. FMBM\_REFQID Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	EFQID	<p>Error Frame Queue ID.</p> <p>EFQID is the FQID used when FMan processing concluded that the frame needs to be enqueued to error queue, or when the FMan processing concluded that the frame should be discarded but frame discard override (FMBM_RCFG[FDOVR]) bit is ‘1’.</p> <p>Note that 0x00_0000 is not a legal value for an FQID.</p>

#### **5.5.4.5.17 Rx Frame Status Discard Mask Register (FMBM\_RFSDM)**

The FMBM\_RFSDM register, shown in Figure 5-66, configures the discard mask that applies before enqueue operation is done. For any bit set to ‘1’ in the frame status word, if the corresponding bit in FMBM\_RFSDM is set, the frame is discarded. In addition, FMBM\_RFFC counter is incremented. Note that if FMBM\_RCFG[FDOVR] is set and the corresponding bit in the FMBM\_RFSEM is set, the frame is enqueued to EFQID. If any of the events described above does not occur, the frame can continue processing and thus can be enqueued to FQID as selected by the classification process.



**Figure 5-66. Rx Frame Status Discard Mask Register (FMBM\_RFSDM)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-84 describes the FMBM RFSDM fields.

**Table 5-84. FMBM\_RFSDM Field Descriptions**

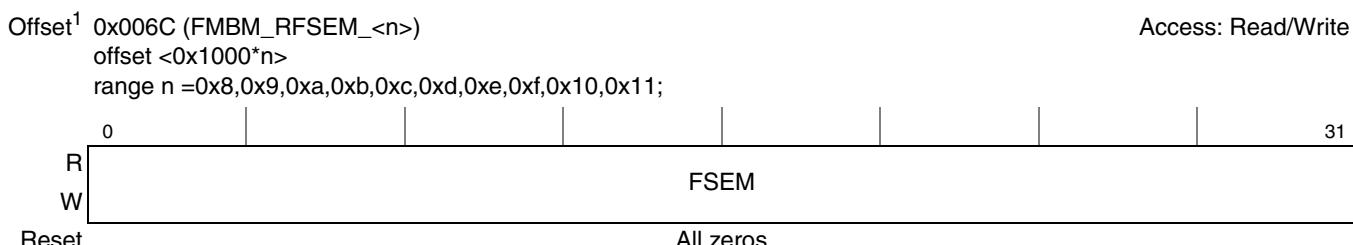
Bits	Name	Description
0-31	FSDM	<p>Frame Status Discard Mask.</p> <p>Each bit of FSDM is checked against the corresponding bit in the frame status word. For any bit set to '1' in the frame status word, if the corresponding bit in FSDM is set, the frame is discarded.</p>

#### 5.5.4.5.18 Rx Frame Status Error Mask Register (FMBM\_RFSEM)

The FMBM\_RFSEM register, shown in Figure 5-67, configures the error mask that applies before enqueue operation is done. For any bit set to ‘1’ in the frame status word, if the corresponding bit in FMBM\_RFSEM is set, the frame is enqueued to the error QID as defined in FMBM\_REFQID[EFQID]. Otherwise, the frame can continue processing and thus can be enqueued to the FQID selected by the

classification process. Frames with FD FPE bit set are not processed in the regular classification, so FQID is selected by the RX port defaults.

If a certain bit is set in both FMBM\_RFSDM and FMBM\_RFSEM, the latter has priority, and the frame discard is not considered.



**Figure 5-67. Rx Frame Status Error Mask Register (FMBM\_RFSEM)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

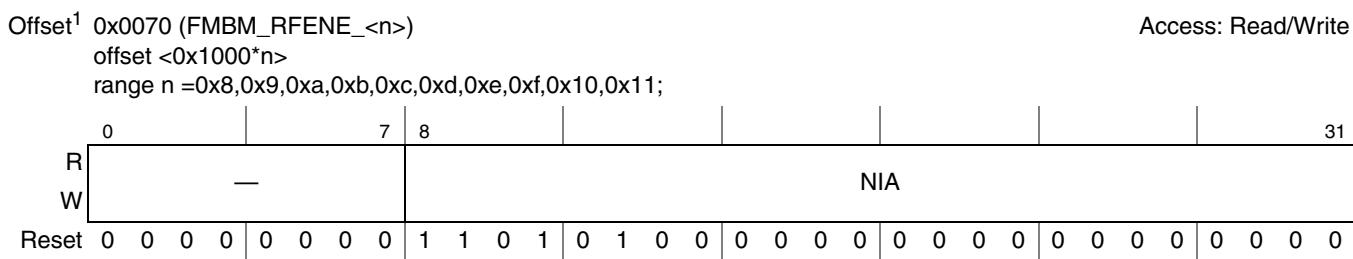
Table 5-85 describes the FMBM RFSEM fields.

**Table 5-85. FMBM\_RFSEM Field Descriptions**

Bits	Name	Description
0-31	FSEM	<p>Frame Status Error Mask.</p> <p>Each bit of FSEM is checked against the corresponding bit in the frame status word. For any bit set to '1' in the frame status word, if the corresponding bit in FSEM is set, the frame queue ID is overwritten by EFQID.</p>

#### **5.5.4.5.19 Rx Frame Enqueue Next Engine Register (FMBM RFENE)**

The FMBM\_RFENE register, shown in Figure 5-68, configures NIA issued by the BMI when frame is ready to be enqueued.



**Figure 5-68. Rx Frame Enqueue Next Engine Register (FMBM\_RFENE)**

<sup>1</sup> There is one register for each Rx port.  $n$  is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-86 describes the FMBM\_RFENE fields.

**Table 5-86. FMBM\_RFENE Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	NIA	<p>Next Invoked Action.</p> <p>When frame is ready to be enqueued, the BMI issues the specified NIA. Default value is set to 0xD40000, which means that the next module is QMI enqueue, and ORR (order restoration required) bit is set.</p> <p>For detailed description refer to <a href="#">Section 5.4.4, “Next Invoked Action (NIA).”</a></p>

#### **5.5.4.5.20 Rx Continuous Mode Next Enqueue Register (FMBM\_RCMNE)**

The FMBM\_RCMNE register, shown in Figure 5-69, configures the Rx port, for a flow in which the same frame is enqueued to multiple queues. This register is used when ICAD[NL]=1 (i.e. this is Not the Last stage in the flow). The NIA is used by the BMI in the ‘Release Internal Resources’ stage, if ICAD[NL]=1. Note that if ICAD[NL]=1 the BMI does not release its internal resources. See Figure 5-5.

**Figure 5-69. Rx Continuous Mode Next Engine (FMBM RCMNE)**

<sup>1</sup> There is one register for each Rx port. n is the Rx Port ID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-87 describes the FMBM RCMNE fields.

**Table 5-87. FMBM RCMNE Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0-7	—	Reserved
8-31	NIA	If ICAD[NL]=0 the BMI end the flow at the end of this stage (no NIA is needed) If ICAD[NL]=1 this NIA is used for the next stage.

## 5.5.4.5.21 Rx External Buffers Manager Pool Information Register (FMBM\_REBMPI)

The FMBM\_REBMPI register, shown in Figure 5-70, allows configuration of a maximum of four external buffers pool IDs per port. Each pool ID manages a pool of external buffers with the same size as initialized in BMan. The size of the pool needs to be configured by software.

## NOTE

It is required that software sets the order of FMBM\_REBMPI<sub><n></sub>\_i according to the external buffers size in ascending order, that is, small buffers go to small i. FMBM\_REBMPI<sub><n></sub>\_i=1 reflects the smallest buffer size related to the Rx port.

Note that the programming of the buffers to be used must contain buffers which are big enough to include the scatter/gather list (256 bytes) and the buffer start margin (as defined in FMBM\_REBM[BSM]) in a single external buffer.

Offset<sup>1</sup> 0x00100 (FMBM\_REBMPI<sub><n></sub>\_<y>)  
 offset <4096\*n+4\*(y-1)>  
 range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;  
 y=1..4

	0	1	2	—	9	10	15	16	—	—	—	—	31
R	VAL	ACE	BP	—	BPID		PBS						
W											All zeros		
Reset													

**Figure 5-70. External Buffers Manager Pool Information Register (FMBM\_REBMPI)**

- <sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 4\*(y-1) + 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-88](#) describes the FMBM\_REBMPI fields.

**Table 5-88. FMBM\_REBMPI<sub><nn></sub>\_y Field Descriptions**

Bits	Name	Description
0	VAL	Valid. When set, the pool specified in PID is valid to allocate external buffers on this port.
1	ACE	Allocate Counter Enable. 0 - RACNT <sub>&lt;n&gt;</sub> _y counter is disabled. 1 - RACNT <sub>&lt;n&gt;</sub> _y counter is enabled.
2	BP	Backup Pool This bit defines the pool as a backup pool when VAL=1. A backup pool is considered for buffer allocation only if all other enabled non-backup enabled pools (i.e. pools with EN=1 and BP=0) are out of buffers. 0 - Define this pool as a regular pool (conditioned by VAL=1). this pool may be used for allocating external buffers. 1 - Define the pool as a backup pool (conditioned by VAL=1). Only when all other enabled and usable non-backup pools (i.e. pools configured with VAL=1 and BP=0) are out of buffers allow external buffer allocation from this pool. Otherwise this pool is not used for external buffer allocations. Note that if SGD is set, some enabled pools may not be usable. <a href="#">Section 5.5.6.2, “Buffer Pool Selection for Rx and O/H.”</a>
3-9	—	Reserved.

**Table 5-88. FMBM\_REBMPI<nn>\_y Field Descriptions (continued)**

Bits	Name	Description
10-15	BPID	<p>Buffer Pool ID.</p> <p>Specify the BM's pool ID used to allocate external buffer to this port.</p> <p>BMan manges 64 pools in which each pool holds buffers of an initialized size.</p> <p>Software configures the BMI according to the initialization of BM.</p> <p>0x00 - Pool ID 0 in BMan is associated with this register.</p> <p>0x01 - Pool ID 1 in BMan is associated with this register.</p> <p>...</p> <p>0x1F - Pool ID 31 in BMan is associated with this register.</p> <p>...</p> <p>0x3F - Pool ID 63 in BMan is associated with this register.</p>
16-31	PBS	<p>Pool Buffer Size.</p> <p>Specify the size of the buffers, in bytes, allocated by BMan at this pool.</p> <p>Software configures the BMI according to the initialization of BM.</p>

#### **5.5.4.5.22 Rx Allocate Counter Register (FMBM\_RACNT)**

For hardware port storage profiles: When enabled in the corresponding FMBM\_REBMPI register, the BMI counts the number of total allocated external buffers by the BMI. The counters gives the possibility to track the load on BM's pools and indicate on external buffers balance issues. There are 'y' counters per Rx port, one for each hardware port storage profile buffer pool.

For virtual storage profiles: The BMI counts the number of total allocated external buffers for this storage profile.

## NOTE

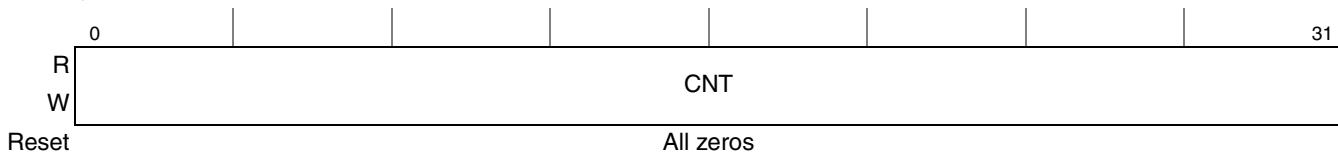
In FMan\_v3 ‘y’=4.

Figure 5-71 shows the FMBM\_RACNT register.

Offset<sup>1</sup> 0x0120 (FMBM\_RACNT\_<n>\_<y>)

## Access: Read/Write

```
offset <4096*n+4*(y-1)>
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;
y=1..4
```



**Figure 5-71. Allocate Counter Register (FMBM\_RACNT)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $4*(y-1) + 0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-89 describes the FMBM\_RACNT fields.

**Table 5-89. FMBM\_RACNT Field Descriptions**

Bits	Name	Description
0-31	CNT	Counter Value. When FMBM_REBMPI< n >_y[ACE] is '1' the count is enabled. The counter value increments whenever the specific BPID which is associated with the Rx port is used for buffer allocation. Write to FMBM_RACNT causes the counter to be initialized to the written value. Read from FMBM_RACNT returns the current counter value.

#### 5.5.4.5.23 Receive Congestion Group Map Register (FMBM\_RCGM)

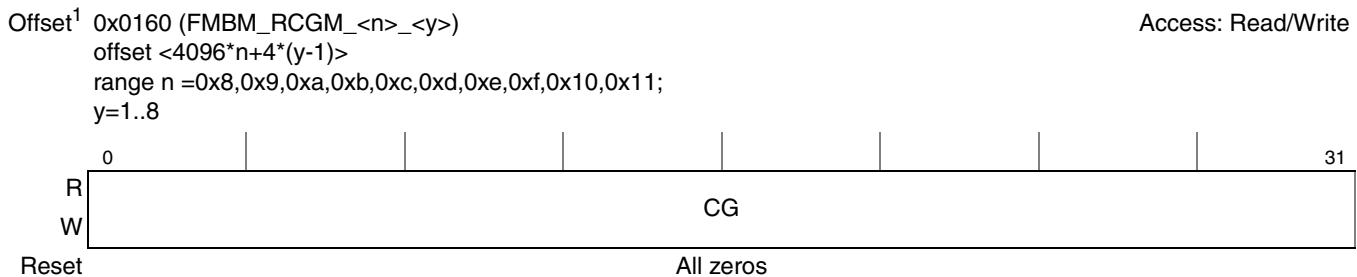
The BMI supports up to N congestion group notifications from the QMan. Each port holds N bits to configure the congestion group that corresponds to the port. The BMI issues pause frames when port congestion is detected. See [Section 5.5.6.15, “Congestion or Rejection Handling”](#) for more details.

FMBM\_RCGM\_n\_1[0] corresponds to the congestion group N-1 of port n and FMBM\_RCGM\_n\_8[31] corresponds congestion group 0 of port n.

##### NOTE

In FMan\_v3 N=256. Therefore in these devices there are 8 FMBM\_RCGM registers for each port. In FMan\_v3 a congestion group priority is associated with each congestion group. This priority is used to generate a priority based flow control packet. The congestion group priority is programmed in Congestion group Priority[CGPx].

Hysteresis is provided by the QMan upon congestion notification state transition. [Figure 5-72](#) shows the FMBM\_RCGM register.



**Figure 5-72. Receive Congestion Group Map Register (FMBM\_RCGM)**

<sup>1</sup> There are up to eight registers per Rx port. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 4\*(y-1) + 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-90 describes the FMBM\_RCGM fields.

**Table 5-90. FMBM\_RCGM Field Descriptions**

Bits	Name	Description
0-31	RCG	Receive Congestion Group Used to map 256 congestion groups from QMan. When a congestion group mapped to the Rx port is detected, the BMI signals the MAC to send pause frame.

#### 5.5.4.5.24 BMan Pool Depletion Register (FMBM\_RMPD)

The FMBM\_RMPD register, shown in Figure 5-73, defines the conditions to generate pause frames subject to the depletion status of BMan pools. The pause frame is generated if at least one of the conditions is valid.

Hysteresis is provided by BMan upon external buffer depletion state transition.

Offset<sup>1</sup> 0x0180 (FMBM\_RMPD\_<n>)  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;

Access: Read/Write

0	3	4	7	8	12	13	15	16	23	24	27	28	31
R	NBPDE	—	—	—	NBPD	—	PFCPEV	—	SBPD	—	—	—	—
W	Reset	—	—	—	All zeros	—	—	—	—	—	—	—	—

**Figure 5-73. BMan Pool Depletion Register (FMBM\_RMPD)**

<sup>1</sup> There is one register for each Rx port. n is the Rx POLD. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-91 describes the FMBM\_RMPD fields.

**Table 5-91. FMBM\_RMPD Field Descriptions**

Bits	Name	Description
0-3	NBPDE	Buffer Pools Depleted Enable. Specify which pool id depletion state should be counted by NBPD criteria. Each bit enables a specific pool. Bit 0 enables pool ID configured in REBMPI<n>_1, Bit 1 enables pool ID configured in REBMPI<n>_2, Bit 2 enables pool ID configured in REBMPI<n>_3, Bit 3 enables pool ID configured in REBMPI<n>_4,  Examples: 0x0 - No buffer pools enabled for counting using NBPD criteria. 0x8 - Pool ID configured in REBMPI<n>_1 is valid for counting using NBPD criteria. 0x4 - Pool ID configured in REBMPI<n>_2 is valid for counting using NBPD criteria. 0xC - Pool ID configured in REBMPI<n>_1 and in REBMPI<n>_2 are valid for counting using NBPD criteria. 0xF - Pools configured in REBMPI<n>_1-4 are valid for counting using NBPD criteria.
4-12	—	Reserved.

**Table 5-91. FMBM\_RMPD Field Descriptions (continued)**

Bits	Name	Description
13-15	NBPD	<p>Number of Buffer Pools Depleted.</p> <p>Set the number of depleted pools, enabled by NBPDE, that if reached, the BMI indicates the MAC to send a pause frame.</p> <p>000 - Send pause frame if 1 pool is depleted.</p> <p>001 - Send pause frame if 2 pools are depleted.</p> <p>...</p> <p>011 - Send pause frame if 4 pools are depleted.</p>
16-23	PFCPEV	<p>802.1Qbb Priority based Flow Control - Priority Enable Vector</p> <p>This field is used by the MAC as the Priority Enable Vector in the PFC frame which is transmitted.</p> <p>For each bit in the priority enable vector that is set to 1, the corresponding timer value in the PFC frame sent by the MAC is valid.</p> <p>The receiver of the PFC frame operates as follows: Upon receipt of a valid PFC PAUSE frame, the MAC control sublayer programs 0 to 8 separate timers (depending on the priority enable vector). For each bit in the priority enable vector that is set to 1, the corresponding timer value is set to the corresponding time value in the PFC frame. A time value of zero has the effect of stopping the timer and placing the corresponding priority in the “not_paused” state. For each bit in the priority enable vector that is set to zero, the corresponding timers are unaffected.</p>
24-27	SBPD	<p>Single Buffer Pool Depletion.</p> <p>SBPD is an 4 bits vector in which each bit corresponds to BM's pool depletion indication configured in FMBM_REBMPI&lt;n&gt;_y[BPID].</p> <p>If a bit is set, the BMI indicates the MAC to send pause frame when the corresponding pool is depleted.</p> <p>Bit 24 enables pause frame signal if pool ID configured in REBMPI&lt;n&gt;_1 is depleted.</p> <p>Bit 25 enables pause frame signal if pool ID configured in REBMPI&lt;n&gt;_2 is depleted.</p> <p>Bit 26 enables pause frame signal if pool ID configured in REBMPI&lt;n&gt;_3 is depleted.</p> <p>Bit 27 enables pause frame signal if pool ID configured in REBMPI&lt;n&gt;_4 is depleted.</p> <p>Examples:</p> <p>0x8 - Initiate pause frame if pool ID configured in REBMPI&lt;n&gt;_1 is depleted.</p> <p>0x4 - Initiate pause frame if pool ID configured in REBMPI&lt;n&gt;_2 is depleted.</p> <p>0xC - Initiate pause frame if pool ID configured in REBMPI&lt;n&gt;_1 or REBMPI&lt;n&gt;_2 is depleted.</p> <p>...</p> <p>0xF - Initiate pause frame if any of the pools specified in REBMPI&lt;n&gt;_1-4 registers are depleted.</p>
28-31	—	Reserved

#### **5.5.4.5.25 Statistics Counters Register (FMBM\_RSTC)**

The FMBM\_RSTC register, shown in Figure 5-74, is used to control and enable Rx port statistics counters. Note that statistics counters are not affected by the assertion of soft reset, but they are affected by the assertion of hard reset.

Offset <sup>1</sup>	0x0200 (FMBM_RSTC_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;	
0	1	
R	EN	
W		
Reset	All zeros	

**Figure 5-74. Rx Statistics Counters Register (FMBM\_RSTC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-92 describes the FMBM\_RSTC fields.

**Table 5-92. FMBM\_RSTC Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0	EN	Enable Statistics Counters. 0 - Statistics Counters are disabled. 1 - Statistics Counters are enabled.
1-31	—	Reserved.

#### **5.5.4.5.26 Rx Frame Counter Register (FMBM\_RFRC)**

The FMBM\_RFRC register, shown in Figure 5-75, counts the total number of frames received on the Rx port. Note that in some cases, the number

Offset <sup>1</sup>	0x0204 (FMBM_RFRC_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;	
0	1	
R		
W		
Reset	All zeros	31

**Figure 5-75. Rx Frame Counter Register (FMBM\_RFRC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-93 describes the FMBM\_RFRC fields.

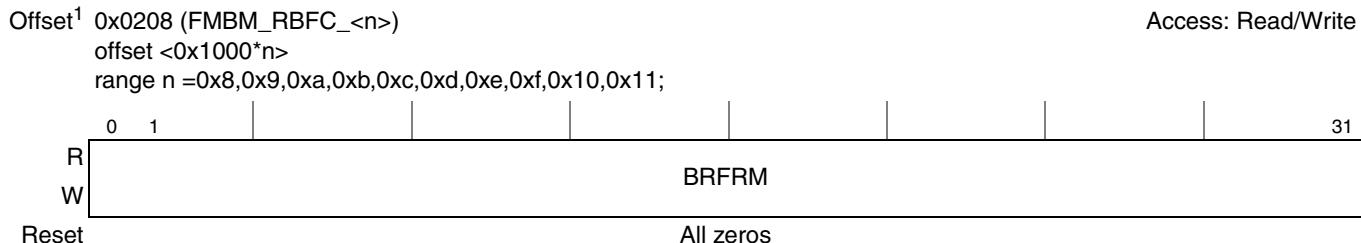
**Table 5-93. FMBM\_RFRC Field Descriptions**

Bits	Name	Description
0-31	RFRM	<p>Received Frames.</p> <p>Counts the number of received frames.</p> <p>The counter is enabled to count if FMBM_RSTC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### 5.5.4.5.27 Rx Bad Frames Counter Register (FMBM\_RBFC)

The FMBM\_RBFC register, shown in [Figure 5-76](#), counts the number of frames received on the Rx port with an error indication. Error cause could be bad CRC, MAC FIFO overflow, coding error, etc.

Bad frames are discarded and not shown to receive queues, unless FMBM\_RCFG[FDOVR] is set, in which case the bad frames are enqueued on the queue configured in FMBM\_REFQID[EFQID].



**Figure 5-76. Rx Bad Frames Counter Register (FMBM\_RBFC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-94](#) describes the FMBM\_RBFC fields.

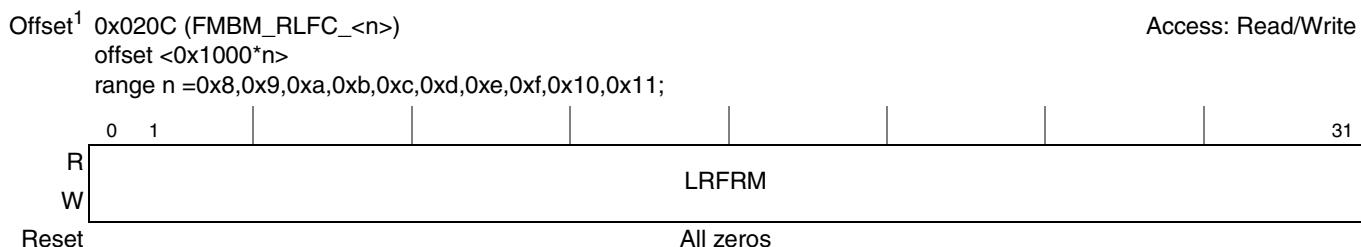
**Table 5-94. FMBM\_RBFC Field Descriptions**

Bits	Name	Description
0-31	BRFRM	Bad Received Frames. Counts the number of bad received frames. The counter is enabled to count if FMBM_RSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

#### 5.5.4.5.28 Rx Large Frames Counter Register (FMBM\_RLFC)

The FMBM\_RLFC register, shown in [Figure 5-77](#), counts the number of frames received on the Rx port with an over size indication. Over size indication is marked when frame size exceeds the maximum configured in the corresponding MAC configuration register (see [Section 6.4.3.1.4, “Maximum Frame Length Register \(MAXFRM\)”](#)). In addition this counter is incremented also when the frame is larger than the FIFO; it is incremented if the frame is larger than FMBM\_PFS\_n[IFSZ]-3\*256

Large frames are truncated to not exceed the maximum configured frame size. In addition, an indication bit is set in the status field in the frame descriptor. Other than this, the frame is processed and enqueued as all other frames.



**Figure 5-77. Rx Large Frames Counter Register (FMBM\_RLFC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-95 describes the FMBM\_RLFC fields.

**Table 5-95. FMBM\_RLFC Field Descriptions**

Bits	Name	Description
0-31	LRFRM	<p>Large Received Frames. Counts the number of large received frames. The counter is enabled to count if FMBM_RSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### 5.5.4.5.29 Rx Filter Frames Counter Register (FMBM\_RFFC)

The FMBM\_RFFC register, shown in Figure 5-78, counts the number of frames received on the Rx port that were filtered out by the parse and classify modules of the FMan.

Those frames are discarded and not shown to receive queues, unless FMBM\_RCFG[FDOVR] is set, in which case the frames are enqueued on the queue configured in FMBM\_REFQID[EFQID].

Offset <sup>1</sup>	0x0210 (FMBM_RFFC_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;	
0	1	31
R	FRFRM	
W		
Reset	All zeros	

**Figure 5-78. Rx Filter Frames Counter Register (FMBM\_RFFC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-96 describes the FMBM\_RFFC fields.

**Table 5-96. FMBM\_RFFC Field Descriptions**

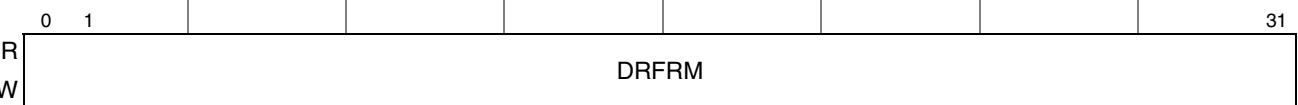
Bits	Name	Description
0-31	FRFRM	<p>Filtered Received Frames. Counts the number of filtered received frames. The counter is enabled to count if FMBM_RSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### 5.5.4.5.30 Rx Frames Discard Counter Register (FMBM\_RFDC)

The FMBM\_RFDC register, shown in Figure 5-79, counts the number of frames received on the Rx port that were not able to enter the receive queue system due to WRED algorithm. Other reasons for enqueue reject may be tail drop, out of service FQ, etc. See QMan chapter for more details.

Those frames are discarded and not shown to receive queues, regardless of the state of FMBM\_RCFG[FDOVR].

Offset<sup>1</sup> 0x0214 (FMBM\_RFDC\_<n>)  
 offset <0x1000\*n>  
 range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;



0	1																														31
R		DRFRM																													
W																															
Reset		All zeros																													

**Figure 5-79. Rx Frames Discard Counter Register (FMBM\_RFDC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-97](#) describes the FMBM\_RFDC fields.

**Table 5-97. FMBM\_RFDC Field Descriptions**

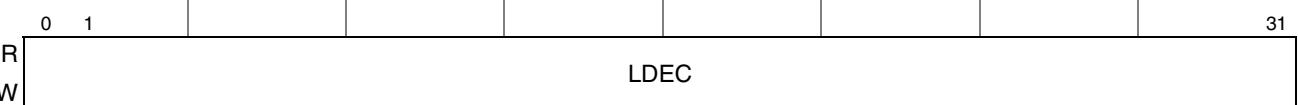
Bits	Name	Description
0-31	DRFRM	<p>Discarded Received Frames.            Counts the number of received frames that were discarded due to WRED algorithm or other QMan rejection reasons.            The counter is enabled to count if FMBM_RSTC[EN] is set. Read operation returns the current count value.            Write operation sets the counter value and the count increments from the written value.            Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### 5.5.4.5.31 Rx Frames List DMA Error Counter Register (FMBM\_RFLDEC)

The FMBM\_RFLDEC register, shown in [Figure 5-80](#), counts the number of frames received on the Rx port that were not able to enter the receive queue system due to QMan reject (mostly due to WRED algorithm), and not able to release their buffers due to DMA error on the scatter/gather list read.

Those frames are discarded and not shown to receive queues, regardless of the state of FMBM\_RCFG[FDOVR], however the buffers that are allocated to the frame are not released.

Offset<sup>1</sup> 0x0218 (FMBM\_RFLDEC\_<n>)  
 offset <0x1000\*n>  
 range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;



0	1																														31
R		LDEC																													
W																															
Reset		All zeros																													

**Figure 5-80. Rx Frames List DMA Error Counter Register (FMBM\_RFLDEC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-98 describes the FMBM\_RFLDEC fields.

**Table 5-98. FMBM\_RLFDEC Field Descriptions**

Bits	Name	Description
0-31	LDEC	<p>List DMA Error Count.</p> <p>Counts the number of received frames that were discarded due to WRED algorithm, and not able to release their buffers due to DMA error on the scatter/gather list read</p> <p>The counter is enabled to count if FMBM_RSTC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### **5.5.4.5.32 Rx Out of Buffers Discard Counter Register (FMBM\_RODC)**

The FMBM\_RODC register, shown in [Figure 5-81](#), counts the number of frames on the Rx port that were not able to enter the receive queue, since buffer allocation has failed. See [Section 5.5.6.20.3, “Buffer Depletion,”](#) for more details.

Those frames are discarded and not shown to receive queues, regardless of the state of FMBM\_RCFG[FDOVR].

Offset<sup>1</sup> 0x021C (FMBM\_RODC\_<n>)  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;

Access: Read/Write

0	1									31
R		ODRFRM								
W										
Reset	All zeros									

**Figure 5-81. Rx Out of Buffers Discard Counter Register (FMBM\_RODC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

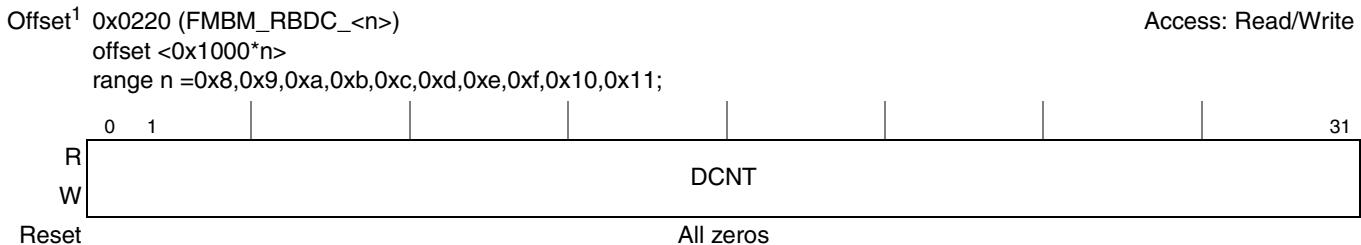
Table 5-99 describes the FMBM RODC fields.

**Table 5-99. FMBM\_RODC Field Descriptions**

Bits	Name	Description
0-31	ODRFRM	<p>Discarded Received Frames.</p> <p>Counts the number of received frames that were discarded due to lack of external buffers.</p> <p>The counter is enabled to count if FMBM_RSTC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

### 5.5.4.5.33 Rx Buffers Deallocate Counter Register (FMBM\_RBDC)

The FMBM\_RBDC register, shown in [Figure 5-82](#), counts the number of buffer deallocate operations.



**Figure 5-82. Rx Buffers Deallocate Counter Register (FMBM\_RBDC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

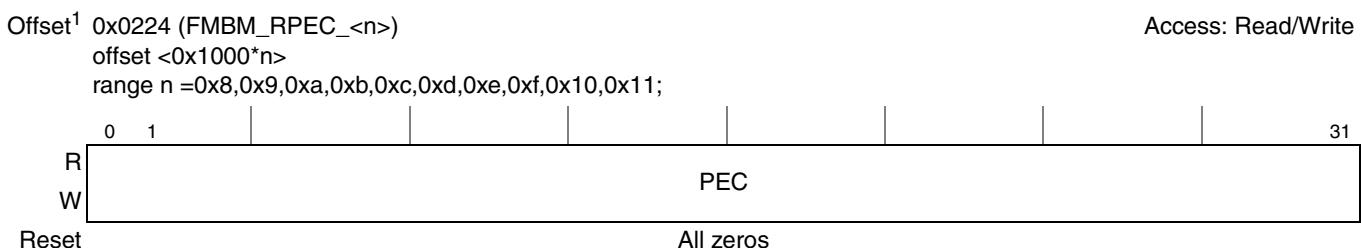
[Table 5-100](#) describes the FMBM\_RBDC fields.

**Table 5-100. FMBM\_RBDC Field Descriptions**

Bits	Name	Description
0-31	DCNT	Buffers Deallocate Counter. Counts the number of buffer deallocate operations, that the port initiated. The counter increments whenever a buffer is returned to the BMan pools, regardless of the BPID. The counter is enabled to count if FMBM_RSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

### 5.5.4.5.34 RX Prepare to Enqueue Counter (FMBM\_RPEC)

The FMBM\_RPEC register, shown in [Figure 5-82](#), counts the total number of time the BMI is invoked with ‘prepare to enqueue’ NIA (per port).



**Figure 5-83. Rx Prepare to Enqueue (FMBM\_RPEC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

**Table 5-101. FMBM\_RPEC Field Descriptions**

Bits	Name	Description
0-31	PEC	<p>Prepare to enqueue counter.</p> <p>The FMBM_RPEC register, shown in <a href="#">Figure 5-82</a>, counts the total number of time the BMI is invoked with ‘prepare to enqueue’ or ‘discard frame’ NIA (per port). In most FMan Rx flows the value of this counter is the same as the value of FMBM_RFRC (frames received on the port), since all the frames received traverse the ‘prepare to enqueue’ stage in the flow (the frames may be dropped in this stage). It is possible to configure a flow (operational mode bit NENQ=1) where the frame ‘loops around’ the ‘prepare to enqueue’ stage numerous times for reassembly of fragments. In this case the FMBM_RPEC counter has a larger value than FMBM_RFRC. An other example, is a flow where operational mode bit NL=1. In this flow, the BMI loops back to ‘prepare to enqueue’ stage. This is useful for implementing multicast of Rx packets. In this cases the FMBM_RPEC counter also has a larger value than FMBM_RFRC.</p> <p>The counter is enabled to count if FMBM_RSTC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### **5.5.4.5.35 Rx Performance Counters Register (FMBM\_RPC)**

The FMBM\_RPC register, shown in Figure 5-84, is used to control and enable Rx port performance counters.

Offset <sup>1</sup>	0x0280 (FMBM_RPC_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;	
0	1	
R	EN	
W		
Reset	All zeros	

**Figure 5-84. Rx Performance Counters Register (FMBM RPC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-102 describes the FMBM RPC fields.

**Table 5-102. FMBM RPC Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0	EN	Enable Performance Counters. 0 - Performance Counters are disabled. 1 - Performance Counters are enabled.
1-31	—	Reserved.

#### **5.5.4.5.36 Rx Performance Count Parameters Register (FMBM\_RPCP)**

This register is for internal use for debug.

The FMBM\_RPCP register, shown in [Figure 5-85](#), is used to define the parameters of the Rx port performance counters.

Offset <sup>1</sup>	0x0284 (FMBM_RPCP_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;	
	0 1 2   7 8 9 10   15 16 19   20 21 22       31	
R	—   TCV   —   RCV   DCV   —   FUCV	
W		All zeros
Reset		

**Figure 5-85. Rx Performance Count Parameters Register (FMBM\_RPCP)**

- <sup>1</sup> There is one register for each Rx port. n is the Rx PVID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-103](#) describes the FMBM\_RPCP fields.

**Table 5-103. FMBM\_RPCP Field Descriptions**

Bits	Name	Description
0-1	—	Reserved.
2-7	TCV	Tasks compare value. Used to set the criteria for concurrent tasks pipeline counter. The counter counts the number of cycles where the criteria is met. 0x00 - Number of tasks allocated for the port is equal or greater than 1. 0x01 - Number of tasks allocated for the port is equal or greater than 2. ... 0x3F- Number of tasks allocated for the port is equal or greater than 64.
8-9	—	Reserved.
10-15	RCV	Receive Compare Value. Used to set the criteria for the received frames queue counter. Receive frame queue holds the frames in the process of receive and their content is in the FMan memory but a task number was not yet allocated for them. The counter counts the number of cycles where the criteria is met. 0x00 - Number of frames in receive queue is equal or greater than 1. 0x01 - Number of frames in receive queue is equal or greater than 2. ... 0x3F- Number of frames in receive queue is equal or greater than 64.
16-19	DCV	DMA Compare Value. Used to set the criteria for open DMA counter. The counter counts the number of cycles where the criteria is met. 0000 - Number of open DMA is equal or greater than 1. 0001 - Number of open DMA is equal or greater than 2. ... 1111- Number of open DMA is 16.

**Table 5-103. FMBM\_RPCP Field Descriptions (continued)**

Bits	Name	Description
20-21	—	Reserved.
22-31	FUCV	<p>FIFO Utilization Compare Value.            Used to set the criteria for FIFO size counter.            The counter counts the number of cycles where the FIFO size is equal or greater than FUCV.            The FIFO size compare value is in 256 bytes granularity (Internal buffer size).            0x000 - Number of utilized buffers in port's FIFO is equal or greater than 1 buffer (256 bytes).            0x001 - Number of utilized buffers in port's FIFO is equal or greater than 2 buffers (512 bytes).            ...            0x3FF - Number of utilized buffers in port's FIFO is equal or greater than 1024 buffers (256k bytes).</p>

#### **5.5.4.5.37 Rx Cycle Counter Register (FMBM\_RCCN)**

The FMBM\_RCCN register, shown in Figure 5-86, is used to count clock cycles.

Offset <sup>1</sup>	0x0288 (FMBM_RCCN_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;	
0		31
R		
W		
Reset	All zeros	

**Figure 5-86. Rx Cycle Counter Register (FMBM\_RCCN)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

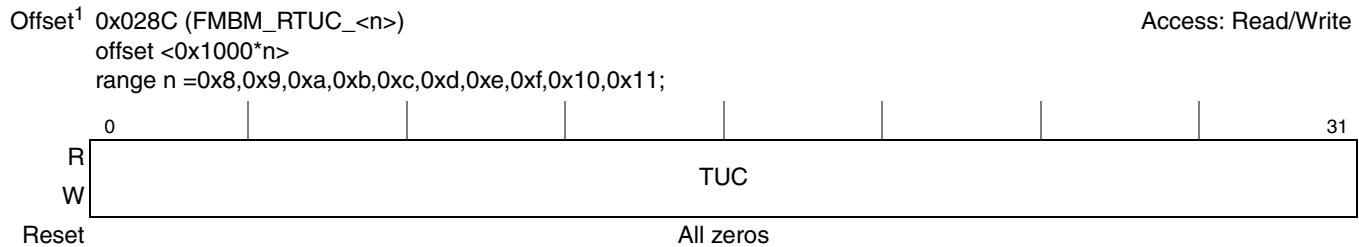
Table 5-104 describes the FMBM\_RCCN fields.

**Table 5-104. FMBM\_RCCN Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0-31	CNUM	<p>Cycle Number.</p> <p>Counts the number of the BMI clock cycles.</p> <p>The counter is enabled to count if FMBM_RPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

### 5.5.4.5.38 Rx Tasks Utilization Counter Register (FMBM\_RTUC)

The FMBM\_RTUC register, shown in [Figure 5-87](#), is used to monitor tasks utilization.



**Figure 5-87. Rx Tasks Utilization Counter Register (FMBM\_RTUC)**

- <sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

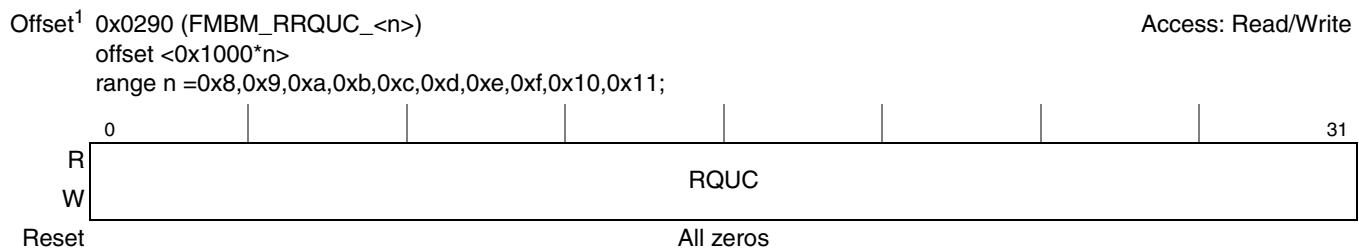
[Table 5-105](#) describes the FMBM\_RTUC fields.

**Table 5-105. FMBM\_RTUC Field Descriptions**

Bits	Name	Description
0-31	TUC	Tasks Utilization Counter. Counts the number of cycles in which the number of tasks allocated for the port is equal or greater than the number specified in FMBM_RPCP[TCV]. The counter is enabled to count if FMBM_RPC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

### 5.5.4.5.39 Rx Receive Queue Utilization Counter Register (FMBM\_RRQUC)

The FMBM\_RRQUC register, shown in [Figure 5-88](#), is used to monitor the receive queue utilization.



**Figure 5-88. Rx Receive Queue Utilization Counter Register (FMBM\_RRQUC)**

- <sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-106](#) describes the FMBM\_RRQUC fields.

**Table 5-106. FMBM\_RRQUC Field Descriptions**

Bits	Name	Description
0-31	RQUC	<p>Receive Queue Utilization Counter.</p> <p>Counts the number of cycles in which the number of frames in receive queue is equal or greater than the number specified in FMBM_RPCP[RCV].</p> <p>The counter is enabled to count if FMBM_RPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### 5.5.4.5.40 Rx DMA Utilization Counter Register (FMBM\_RDUC)

The FMBM\_RDUC register, shown in Figure 5-89, is used to monitor the DMA utilization.

Offset<sup>1</sup> 0x0294 (FMBM\_RDUC\_<n>)  
offset <0x1000\*n>  
range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;

Access: Read/Write

0 31

R W DUC

All zeros

**Figure 5-89. Rx DMA Utilization Counter Register (FMBM\_RDUC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

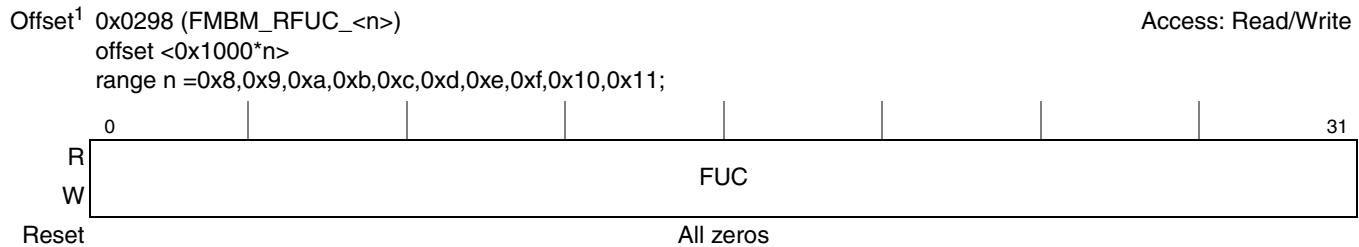
[Table 5-107](#) describes the FMBM\_RDUC fields.

**Table 5-107. FMBM\_RDUC Field Descriptions**

Bits	Name	Description
0-31	DUC	<p>DMA Utilization Counter.</p> <p>Counts the number of cycles in which the number of open DMA is equal or greater than the number specified in FMBM_RPCP[DCV].</p> <p>The counter is enabled to count if FMBM_RPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

### 5.5.4.5.41 Rx FIFO Utilization Counter Register (FMBM\_RFUC)

The FMBM\_RFUC register, shown in [Figure 5-90](#), is used to monitor FIFO utilization.



**Figure 5-90. Rx FIFO Utilization Counter Register (FMBM\_RFUC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

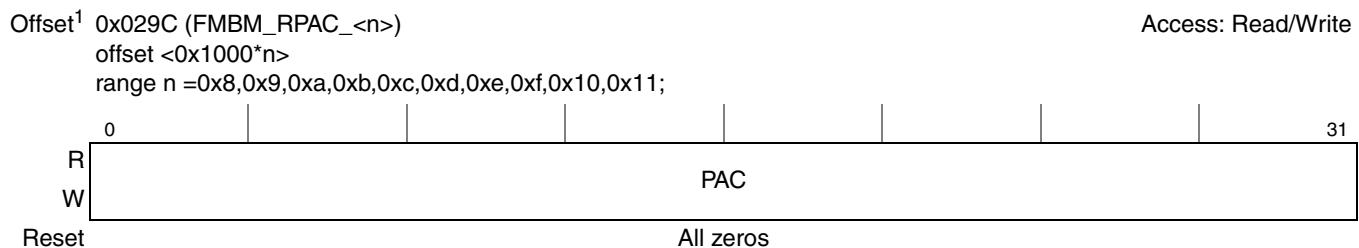
[Table 5-108](#) describes the FMBM\_RFUC fields.

**Table 5-108. FMBM\_RFUC Field Descriptions**

Bits	Name	Description
0-31	FUC	FIFO Utilization Counter. Counts the number of cycles in which actual used FIFO size is equal or greater than the size specified in FMBM_RPCP[FUCV]. The counter is enabled to count if FMBM_RPC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

### 5.5.4.5.42 Rx Pause Activation Counter Register (FMBM\_RPAC)

The FMBM\_RPAC register, shown in [Figure 5-91](#), is used to monitor pause activation.



**Figure 5-91. Rx Pause Activation Counter Register (FMBM\_RPAC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-109 describes the FMBM\_RPAC fields.

**Table 5-109. FMBM\_RFUC Field Descriptions**

Bits	Name	Description
0-31	PAC	Pause Activation Counter. Counts the number of cycles in which the pause activation control signal state is on. The counter is enabled to count if FMBM_RPC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

#### 5.5.4.5.43 Rx Debug Configuration Register (FMBM\_RDCFG)

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires debug.

The FMBM\_RDCFG register, shown in Figure 5-92, sets the condition in which a BMI trap is considered a match, the level of trace that the BMI should add to the debug trace portion of the IC, and the action to be taken if a debug mark is set when a frame is ready to be enqueued. There is one register per debug flow. See Section 5.5.6.22, “Debug Capabilities,” for more details about BMI debug options. Note that debug trap compare value and mask registers apply to all ports and they are defined in the common registers group.

#### NOTE

DTO field is not relevant and not used in flows B and C.

Offset <sup>1</sup>	0x0300 (FMBM_RDCFG_<n>_<y>)	Access: Read/Write																																																																					
	offset <4096*n+4*(y-1)>																																																																						
	range n =0x8,0x9,0xa,0xb,0xc,0xd,0xe,0xf,0x10,0x11;																																																																						
	y=1..3																																																																						
Reset	<table border="1"> <tr> <td>0</td><td>1</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td></td><td></td><td>23</td><td>24</td><td>27</td><td>28</td><td>31</td> </tr> <tr> <td>R</td><td>—</td><td>CMPOP</td><td>—</td><td>TL</td><td>—</td><td>TR_DST</td><td>—</td><td>HALT</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td> </tr> <tr> <td>W</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td> </tr> </table>	0	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16			23	24	27	28	31	R	—	CMPOP	—	TL	—	TR_DST	—	HALT	—	—	—	—	—	—	—	—	—	0	0	0	0	0	W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
0	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16			23	24	27	28	31																																																	
R	—	CMPOP	—	TL	—	TR_DST	—	HALT	—	—	—	—	—	—	—	—	—	0	0	0	0	0																																																	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0																																																	

**Figure 5-92. Rx Debug Configuration Register (FMBM\_RDCFG)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is 4\*(y-1) + 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.” ‘y’=1,2,3 corresponds to Flow A, B,C respectively.

Table 5-110 describes the FMBM\_RDCFG fields.

**Table 5-110. FMBM\_RDCFG Field Descriptions**

Bits	Name	Description
0	—	Reserved
1-3	CMPOP	Compare Operator. Selects the trap condition. 000 - Trap Disabled (never match) 001 - Always match (referred to as trap bypass) 010 - Match if (trap comp & trap mask) equals the (frame FD & trap mask). 011 - 111 - reserved
4-5	—	Reserved.
6-7	TL	Trace Level. Selects the amount of trace data to be written in debug portion if a trap match was found. 00 - Trace disable 01 - Minimum trace 10 - verbose trace 11 - very verbose trace
8-9	—	Reserved.
10-11	TR_DST	Trace Destination. If the debug mark remains set when the BMI is in preparation for enqueue, the BMI signals DMA to copy the debug portion in IC. The destination of this copy depends on the programming below. Note that if system port is selected, it is the user's responsibility to keep enough margin in the beginning of the frame's buffer, to avoid debug data from writing over the frame data. 00 - Debug trace is written to external memory. 01 - Debug trace routed to debug port (Nexus). 10 - Reserved 11 - Reserved
12-13	—	Reserved.
14-15	HALT	Halt execution. If the debug mark remains set when the BMI is before enqueue and NIA is about to be issued, the BMI may indicate to FPM that halt is desired, according to the programming below. 00 - No halt. normal operation continues. 01 - stop this task if debug mark remained set. 10 - stop this port if debug mark remained set. 11 - stop all ports if debug mark remained set.
16-23	—	Reserved.
24-27	DTO	Debug Trace Offset. Sets the offset of the debug trace from the beginning of the IC buffer. 0000 - 0111 - Reserved. 1000 - Debug trace starts at offset 0x80 from the beginning of the IC buffer. 1001 - Debug trace starts at offset 0x90 from the beginning of the IC buffer. 1010 - Debug trace starts at offset 0xa0 from the beginning of the IC buffer. 1011 - Debug trace starts at offset 0xb0 from the beginning of the IC buffer. 1100 - Debug trace starts at offset 0xc0 from the beginning of the IC buffer. 1101 - Debug trace starts at offset 0xd0 from the beginning of the IC buffer. 1110 - Debug trace starts at offset 0xe0 from the beginning of the IC buffer. 1111 - Debug trace starts at offset 0xf0 from the beginning of the IC buffer.
28-31	—	Reserved.

#### 5.5.4.5.44 Rx General Purpose Register (FMBM\_RGPR)

The FMBM\_RGPR register, shown in [Figure 5-90](#), is used as a general purpose register. This register does not serve a specific purpose. It may be used for application specific purposes.



**Figure 5-93. Rx General Purpose Register (FMBM\_RGPR)**

- <sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

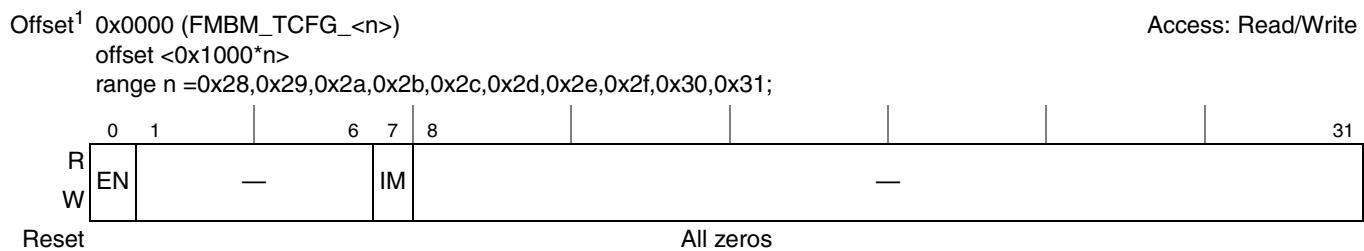
**Table 5-111. FMBM\_RGPR Field Descriptions**

Bits	Name	Description
0-31	GPR	General Purpose Register value.

#### 5.5.4.6 Tx Port Register Descriptions

##### 5.5.4.6.1 Tx Configuration Register (FMBM\_TCFG)

The FMBM\_TCFG register, shown in [Figure 5-94](#), is used to enable and configure Tx port.



**Figure 5-94. Tx Configuration Register (FMBM\_TCFG)**

- <sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-112 describes the FMBM\_TCFG fields.

**Table 5-112. FMBM\_TCFG Field Descriptions**

Bits	Name	Description
0	EN	<p>Enable 0 Port is disabled. If set to 0 during data transfer, the BMI completes the data transfer of the frames already started, including outstanding dequeue, outstanding DMA, and frames waiting for confirmation, and does not initiate new tasks for Tx. 1 When set the port is enabled to transmit data.</p> <p><b>Note:</b> If the port is disabled dynamically (EN=0), the user must ensure that other fields in this register maintain the same value as long as FMBM_TST[BSY]=1. Only after FMBM_TST[BSY]=0, the user may change the values of the other fields in this register.</p>
1-6	—	Reserved
7	IM	<p>Independent mode 0 - Normal mode 1 - Independent mode</p>
8-31	—	Reserved. (bits 24:31 are implemented)

#### 5.5.4.6.2 Tx Status Register (FMBM\_TST)

The FMBM\_TST register, shown in Figure 5-95, is used to read status of Tx port. This register is used for graceful stop.



**Figure 5-95. Tx Status Register (FMBM\_TST)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

<sup>2</sup> TPID reset value is set to Tx PortID.

Table 5-113 describes the FMBM\_TST fields.

**Table 5-113. FMBM\_TST Field Descriptions**

Bits	Name	Description
0	BSY	<p>Busy. BSY is asserted while frames are processed in the port. When Enable is cleared, BSY should be read in order to detect that the port is not busy. 0 - Tx port is not busy. 1 - Tx port is busy.</p>
1-9	—	Reserved

**Table 5-113. FMBM\_TST Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
10-15	TPID	Tx Port ID. This is the hardware port ID (PortID) associated with the register.
16-31	—	Reserved

#### 5.5.4.6.3 Tx DMA Attributes Register (FMBM\_TDA)

The FMBM\_TDA register, shown in [Figure 5-96](#), is used to set DMA attributes to be used for the port. Note that read operations always generate cacheable/no allocate and write of payload besides the frame header always generates cacheable/no stash attributes.

**Figure 5-96. Tx DMA Attributes Register (FMBM\_TDA)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000 \cdot n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

**Table 5-114** describes the FMBM\_TDA fields.

**Table 5-114. FMBM\_TDA Field Descriptions**

Bits	Name	Description
0–1	SWAP	Swap payload data. 00 - No swap, transfer data as is. 01–11 Reserved
2–3	ICC	IC write cache attributes. Used when DMA copies IC from FMan memory to external memory 00 - Cacheable, no Allocate (No Stashing). 01 - Cacheable and Allocate (Stashing on). 10, 11 - Reserved.
4–31	—	Reserved

#### 5.5.4.6.4 Tx FIFO Parameters Register (FMBM\_TFP)

The FMBM\_TFP register, shown in Figure 5-97, is used to set Tx port fifo parameters. The configuration of the parameters in the FMBM\_TFP registers can be altered per need during the operation of the system, even while the ports are enabled and moving data. An exception to this is a change of DPDE in which the new written value is lower than the previous. Such change requires that the port is disabled and not busy (FMBM\_TST[BSY] is read ‘0’) before the configuration change operation. In case DPDE value is increased the port FMBM\_PFS configured IFSZ may also have to be increased to satisfy the internal FIFO

configuration requirements. For details refer to [Section 5.5.6.18, “Internal FIFO Configuration Requirements.”](#) See [Section 5.5.6.13, “FMan DMA Priority Elevation”](#) for more details.

This register is for internal use. It is recommended not to modify its reset value (except for DPDE see the table below). This value can be modified in a system which requires fine tuning or debug.

Offset<sup>1</sup> 0x000C (FMBM\_TFP\_<n>)  
offset <0x1000\*n>  
range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;

0	—	5 6	MFL	DPDE	—	FLCL	31
R	—				—		
Reset	0 0 0 0   0 0 0 0   0 0 0 0   0 0 0 0   n n n n <sup>2</sup>   0 0 0 0   0 0 0 1   0 0 1 1						

**Figure 5-97. Tx FIFO Parameters Register (FMBM\_TFP)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

<sup>2</sup> DPDE default depends on the port:

1G - 0001

10G - 0011

**Table 5-115. FMBM\_TFP Reset Values**

Reg Name	Reset Value
FMBM_TFP_40, FMBM_TFP_41, FMBM_TFP_42, FMBM_TFP_43, FMBM_TFP_44, FMBM_TFP_45, FMBM_TFP_46, FMBM_TFP_47	0000_0000_0000_0000_0001_0000_0001_0011
FMBM_TFP_48, FMBM_TFP_49	0000_0000_0000_0000_0011_0000_0001_0011

[Table 5-116](#) describes the FMBM\_TFP fields.

**Table 5-116. FMBM\_TFP Field Descriptions**

Bits	Name	Description
0-5	—	Reserved.
6-15	MFL	Minimum Fill Level. If the amount of valid data on transmit buffer is smaller than the amount specified in MFL, the BMI signals the main FM’s DMA to elevate the FMan priority on the SoC main bus The size is in 256 bytes granularity. 0x000 - Minimum Fill Level is 0*256bytes (no priority elevation ever) 0x001 - Minimum Fill Level is 1*256bytes ... 0x0FF - Minimum Fill Level is 255*256bytes. ... 0x3FF- Minimum Fill Level is 1023*256bytes

**Table 5-116. FMBM\_TFP Field Descriptions**

Bits	Name	Description
16-19	DPDE	<p>Dequeue Pipeline Depth.</p> <p>Set the number of outstanding dequeue requests.</p> <p>0000 - Pipeline depth is 1 Recommended for port bandwidth of 1Gbps</p> <p>0001 - Pipeline depth is 2 Recommended for port bandwidth of 2.5Gbps.</p> <p>0010 - Pipeline depth is 3</p> <p>0011 - Pipeline depth is 4 Recommended for port bandwidth of 10Gbps.</p> <p>...</p> <p>0111 - Pipeline depth is 8</p> <p>1000 - 1111 - Reserved</p> <p><b>Note:</b> If the number is bigger than 1, it is possible to exceed FIFO Low Comfort Level by the accumulative size of the frames that arrive from the queue. On the other hand, setting DPDE too small affects the transmit bandwidth by introducing inter-frame gap longer than the minimum required.</p> <p><b>Note:</b> There is no limitation on this field. Please follow recommendations.</p> <p><b>Note:</b> There is a restriction between this field and FMMQ_GC[ENQ_THR] and FMMQ_GC[DEQ_THR]. See <a href="#">Section 5.6.3.2.1, “QMI General Configuration Register (FMMQ_GC).”</a></p> <p><b>Note:</b> See <a href="#">Section 5.5.6.18.2, “Internal FIFO for Tx Ports”</a> for a description on the connection between DPDE and the FIFO size.</p>
20-21	—	Reserved.
22-31	FLCL	<p>FIFO Low Comfort Level.</p> <p>Set a byte count limit for transmit queue size. The limit should be small enough to prevent high latencies of high priority frames, while on the other hand it should be big enough to allow reasonable transmit data streaming.</p> <p>The BMI initiates next frame dequeue only if the current total size of frames serviced on the port is not greater than FLCL and the number of pending dequeue tasks do not exceeds dequeue pipe line depth as configured on DPDE.</p> <p>The size is in 256 bytes granularity.</p> <p>0x000 - FIFO Low Comfort Level is 1*256bytes</p> <p>0x001 - FIFO Low Comfort Level is 2*256bytes</p> <p>...</p> <p>0x0FF - FIFO Low Comfort Level is 256*256bytes.</p> <p>...</p> <p>0x3FF- FIFO Low Comfort Level is 1024*256bytes</p>

#### **5.5.4.6.5 Tx Frame End Data Register (FMBM\_TFED)**

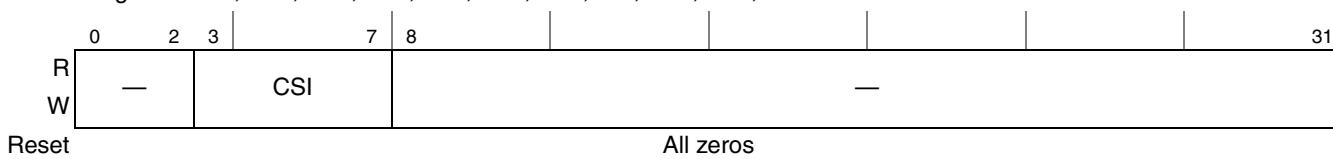
The FMBM\_TFED register, shown in Figure 5-98, is used to define Tx port action on frame's last bytes.

Offset<sup>1</sup> 0x0010 (FMBM\_TFED\_<n>)

## Access: Read/Write

offset <0x1000\*n>

range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;



**Figure 5-98. Tx Frame End Data Register (FMBM\_TFED)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000 \cdot n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-117 describes the FMBM\_TFED fields.

**Table 5-117. FMBM\_TFED Field Descriptions**

Bits	Name	Description
0-2	—	Reserved
3-7	CSI	Checksum Ignore. Set the number of bytes from end of frame ignored in frame checksum calculation. Used to eliminate data located in end of frame from TCP/UDP checksum calculation. 00000 - Calculate all frame checksum. 00001 - Eliminate last 1 byte from frame checksum. 00010 - Eliminate last 2 bytes from frame checksum. ... 10000 - Eliminate last 16 bytes from frame checksum. 10001 - 11111 - Reserved.
8-31	—	Reserved

#### 5.5.4.6.6 Tx Internal Context Parameters Register (FMBM\_TICP)

The FMBM\_TICP register, shown in Figure 5-99, is used to configure Internal Context (IC) parameters. These parameters supply the mechanism to tell the BMI which portion of the IC block should be copied from external memory prior to the frame fetch. The same portion is written back to external memory after the transmission of the frame is done and the transmission's timestamp has been captured and written into the IC. Note that the IC read is enabled by FD Command [RPD] bit, and the IC write is enabled by FD Command [UPD] bit in the FD command word. Section 5.5.6.4, “Internal and External Margins.”

##### NOTE

The user must use caution when configuring the values in this register, so the FD field in the IC is NOT overwritten by this action.

##### NOTE

In FMan\_v3 this register is used when VSPE=0 (virtual storage profile is disabled).

Offset <sup>1</sup>	0x0014 (FMBM_TICP_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;	
R	0 — ICEOF — ICIQF — ICSZ	31
W		
Reset	All zeros	

**Figure 5-99. Tx Internal Context Parameters Register (FMBM\_TICP)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-118 describes the FMBM\_TICP fields.

**Table 5-118. FMBM\_TICP Field Descriptions**

Bits	Name	Description
0-10	—	Reserved
11-15	ICEOF	<p>Internal Context External Offset.</p> <p>Specifies the offset in external buffer in which internal context is transferred from, 16 bytes granularity.</p> <p>The offset is added to the address that is written in the frame descriptor.</p> <p>00000 - IC is transferred from offset 0x0 in external buffer.</p> <p>00001 - IC is transferred from offset 0x010 in external buffer.</p> <p>...</p> <p>11111 - IC is transferred from offset 0x1F0 in external buffer.</p>
16-19	—	Reserved
20-23	ICIOF	<p>Internal Context Internal Offset.</p> <p>Specifies the offset in internal context to which data is transferred from the external buffer to internal context buffer, 16 bytes granularity.</p> <p>Together with ICSZ used to specify the internal context segment transferred to internal buffer. The sum of ICIOF and ICSZ should not cross the IC block boundary (256 bytes from the beginning of the block).</p> <p>0000 - IC data to be transferred to the beginning of the IC block, no offset.</p> <p>0001 - IC data to be transferred to offset 0x10 from the beginning of the IC block.</p> <p>...</p> <p>1111 - IC data to be transferred to offset 0xF0 from the beginning of the IC block.</p>
24-26	—	Reserved
27-31	ICSZ	<p>Internal Context copy Size.</p> <p>Specifies the size of internal context transferred to internal context internal buffer.</p> <p>Together with ICIOF used to specify the internal context segment transferred to internal buffer.</p> <p>16 bytes granularity. (up to 256 bytes).</p> <p>00000 - IC segment is not transferred.</p> <p>00001 - IC segment to be transferred size is 16 bytes.</p> <p>00010 - IC segment to be transferred size is 32 bytes.</p> <p>...</p> <p>10000 - IC segment to be transferred size is 256 bytes.</p> <p>10001 - 11111 - reserved.</p>

#### **5.5.4.6.7 Tx Frame Dequeue Next Engine Register (FMBM\_TFDNE)**

The FMBM\_TFDNE register, shown in Figure 5-100, configures NIA issued by the BMI. Free tnum allocated by the BMI is sent to the next module according to specified NIA. When NIA is set to FMan controller, Tx port is operating in independent mode. In normal mode NIA should be set to QMI.

**Figure 5-100. Tx Frame Dequeue Next Engine Register (FMBM\_TFDNE)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000 \cdot n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-119 describes the FMBM\_TFDNE fields.

**Table 5-119. FMBM\_TFDNE Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	NIA	Next Invoked Action. When allocating free nnum, the BMI issues the specified NIA. Default value is set to 0x580000, which means that the next module is QMI dequeue. For detailed description refer to <a href="#">Section 5.4.4, “Next Invoked Action (NIA).”</a>

#### 5.5.4.6.8 Tx Frame Attributes Register (FMBM\_TFCA)

The FMBM\_TFCA register, shown in [Figure 5-101](#), configures frame attributes. The attributes are used when the frame task is moved to the next module for the first time.

Offset<sup>1</sup> 0x001C (FMBM\_TFCA\_<n>) Access: Read/Write  
offset <0x1000\*n>  
range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;

0	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OR	—	COLOR	—	—	—	—	MR	—	—	—	—	A0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
Reset	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 5-101. Tx Frame Attributes Register (FMBM\_TFCA)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx Poid. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-120 describes the FMBM\_TFCA fields.

**Table 5-120. FMBM\_TFCA Field Descriptions**

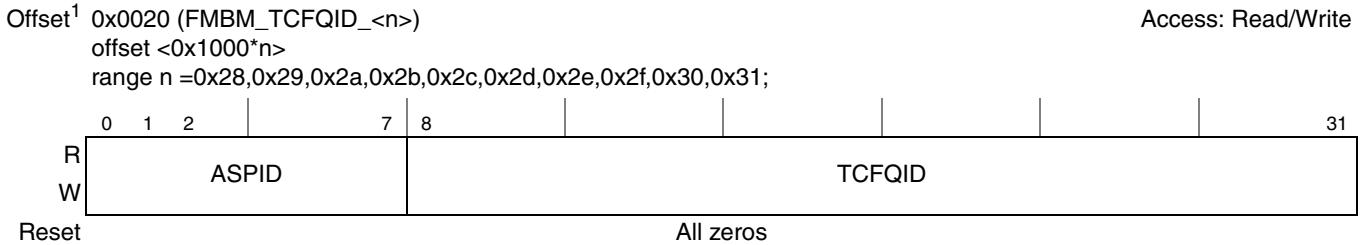
Bits	Name	Description
0	OR	Order definition. Will determine the OD. If OR is set, the FPM attaches the ordering attributes to the task. Later the order restoration may be requested for this task before the task enters order-sensitive state (for example, enqueue). 0 - No order definition is needed. 1 - Order definition is needed.
1-3	—	Reserved
4-5	COLOR	Default color. Will determine the default color of the frame. Color can be altered by frame processing modules as needed. 00 - Green 01 - Yellow 10 - Red 11 - Override. Enqueue regardless of RED/WRED/tail drop thresholds.
6-7	—	Reserved
8-9	—	Reserved

**Table 5-120. FMBM\_TFCA Field Descriptions (continued)**

Bits	Name	Description
10-15	MR	Mode Attributes. These bits define internal modes of operation in the FMan controller and update the MR[0:5] when this task is executed on the FMan controller.
16-23	A0	In FMan_v3: Initial value for ICAD[A0] field. See <a href="#">Section 5.4.3.3, “Internal Context Action Descriptor (ICAD)</a> See FMan “Context A—A0 Field Description for Tx Port” and <a href="#">Section 5.4.3.3, “Internal Context Action Descriptor (ICAD)</a> ’ for detailed description of this field.
24-31	—	Reserved

#### 5.5.4.6.9 Tx Confirmation Frame Queue ID Register (FMBM\_TCFQID)

The FMBM\_TCFQID register, shown in [Figure 5-102](#), configures the default frame queue ID to enqueue Tx confirm frame.



**Figure 5-102. Tx Confirmation Frame Queue ID Register (FMBM\_TCFQID)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

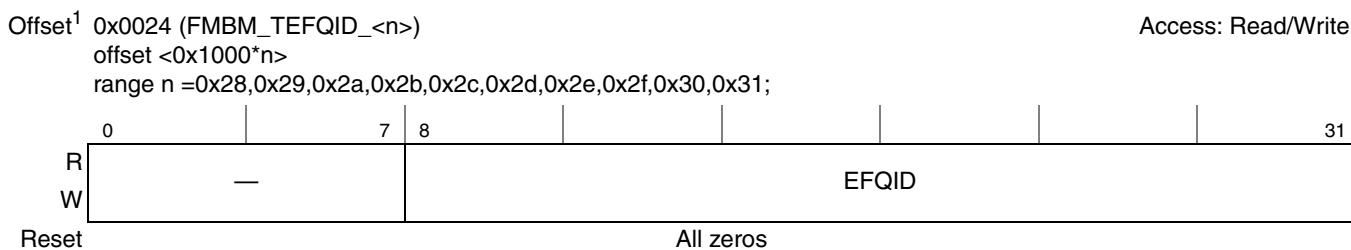
[Table 5-121](#) describes the FMBM\_TCFQID fields.

**Table 5-121. FMBM\_TCFQID Field Descriptions**

Bits	Name	Description
0-7	ASPID	Absolute Storage Profile ID The BMI initializes the default storage profile ID to the value of ASPID. This is the absolute value of the virtual storage profile offset. This value is used for all frames, unless overwritten by the value programmed in Context B. This storage profile is used for Tx confirmation (if enabled).
8-31	TCFQID	Default Confirmation Frame Queue ID. The BMI initializes the confirmation queue ID to the value set in DCFQID. In case of Tx confirm frame, the task is enqueued to the CFQID upon the completion of the frame transmission. If CFQID equal 0x00_0000, it means that no confirmation is required.

#### 5.5.4.6.10 Tx Error Frame Queue ID Register (FMBM\_TEFQID)

The FMBM\_TEFQID register, shown in Figure 5-103, configures the error frame queue ID issued by the BMI.



**Figure 5-103. Tx Error Frame Queue ID Register (FMBM\_TEFQID)**

<sup>1</sup> There is one register for each Tx port.  $<n>$  is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

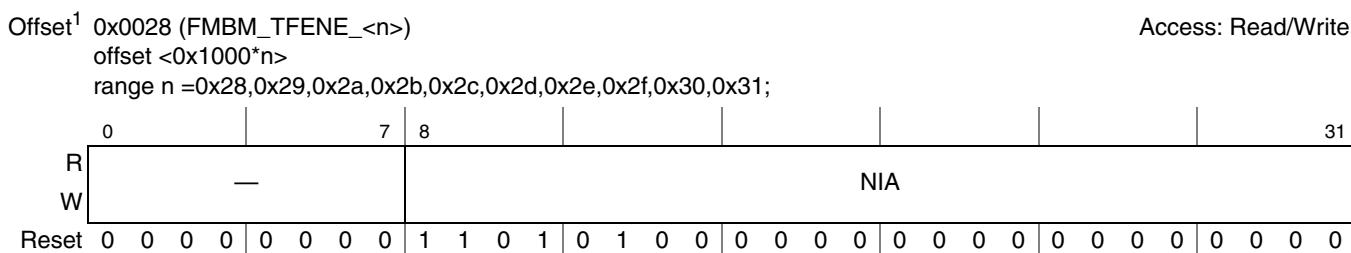
Table 5-122 describes the FMBM\_TEFQID fields.

**Table 5-122. FMBM\_TEFQID Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	EFQID	Error Frame Queue ID.
		EFQID is the FQID used when a DMA error detected during DMA transfer of the frame or its scatter/gather list, or if UFD or LGE conditions was met (described in <a href="#">Section 5.5.6.20.5, “Unsupported Frames”</a> ), and the frame confirmation queue (CFQID) is not legal (0x00_0000). Note that 0x00_0000 is not a legal value for an FQID.

#### **5.5.4.6.11 Tx Frame Enqueue Next Engine Register (FMBM\_TFENE)**

The FMBM\_TFENE register, shown in Figure 5-104, configures NIA issued by the BMI, when frame is ready to be enqueued on Tx confirm QID, if needed.



**Figure 5-104. Tx Frame Enqueue Next Engine Register (FMBM\_TFENE)**

<sup>1</sup> There is one register for each Tx port.  $<n>$  is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-123 describes the FMBM\_TFENE fields.

**Table 5-123. FMBM\_TFENE Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	NIA	Next Invoked Action. When frame is ready to be enqueued, the BMI issues the specified NIA. Default value is set to 0xD40000, which means that the next module is QMI enqueue, and ORR (order restoration required) bit is set. For detailed description refer to <a href="#">Section 5.4.4, “Next Invoked Action (NIA).”</a>

#### 5.5.4.6.12 Tx Rate Limiter Scale Register (FMBM\_TRLMTS)

The FMBM\_TRLMTS register, shown in [Figure 5-105](#), controls Tx port rate limiter and configures its time unit scale.

Offset<sup>1</sup> 0x002C (FMBM\_TRLMTS\_<n>)  
offset <0x1000\*n>  
range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;  
Access: Read/Write

0	1	—	10	11	15	16	—	—	—	—	—	31
R	EN	—	—	TSBS	—	—	—	—	—	—	—	—

Reset: All zeros

**Figure 5-105. Tx Rate Limiter Scale Register (FMBM\_TRLMTS)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx Poid. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-124 describes the FMBM\_TRLMTS fields.

**Table 5-124. FMBM\_TRLMTS Field Descriptions**

Bits	Name	Description
0	EN	Enable. 0 - Rate limiter disabled. The port is allowed to transmit as fast as the hardware is able and as long as data is available. 1 - Rate limiter enabled. The port transmit rate is limited and cannot exceed the programmed rate.
1-10	—	Reserved.

**Table 5-124. FMBM\_TRLMTS Field Descriptions (continued)**

Bits	Name	Description
11-15	TSBS	<p>Time Stamp Bus Scale.</p> <p>TSB is the FPM central Timestamp Bus (TSB) that provides 32-bit integer count with high precision rate independent of the FMan clock frequency (see <a href="#">Section 5.7, “Frame Manager—Frame Processing Manager,”</a> for more details).</p> <p>TSBS specifies the bit of TSB that is used as the reference unit count for the Tx rate limiter calculations. For example, If TSB count is incremented per 1/1024 microseconds, and the TSBS is set to 21 (31–10), the Tx rate limiter time unit is counting per 1 microseconds.</p> <p>Note that TSB is numbered as TSB[0:31], such that 31 is the LSB.</p> <p>00000 - Rate Counter time unit equals to TSB count period / <math>2^{31}</math> (toggle period of TSB[0]).</p> <p>00001 - Rate Counter time unit equals to TSB count period / <math>2^{30}</math> (toggle period of TSB[1]).</p> <p>...</p> <p>11111 - Rate Counter time unit equals to TSB count period (toggle period of TSB[31]). (toggle period is the time elapsed between changes of state of the referenced bit).</p>
16-31	—	Reserved.

#### 5.5.4.6.13 Tx Rate Limiter Register (FMBM\_TRLMT)

The FMBM\_TRLMT register, shown in [Figure 5-106](#), configures Tx port rate limiter used to control the average transmission rate and burst rate. The algorithm used to implement the rate limiter is Token Bucket. Detailed explanation can be found in [Section 5.5.6.17, “Tx and O/H Rate Limiter.”](#)

##### NOTE

If PFC is enabled in mEMAC (COMMAND\_CONFIG[PFC\_MODE]) the rate limiter should not be used. If Tx rate is limited, FPC stop transmit timing may be violated.

Offset<sup>1</sup> 0x0030 (FMBM\_TRLMT\_<n>)  
offset <0x1000\*n>  
range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;  
Access: Read/Write

0	5 6	MBS	15 16	21 22	31
R	—	MBS	—	RLM	
Reset	All zeros				

**Figure 5-106. Tx Rate Limiter Register (FMBM\_TRLMT)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

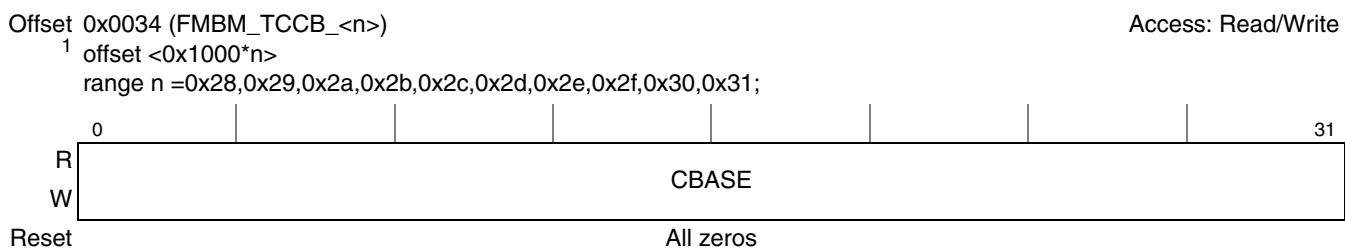
Table 5-125 describes the FMBMTRLMT fields.

**Table 5-125. FMBMTRLMT Field Descriptions**

Bits	Name	Description
0-5	—	Reserved.
6-15	MBS	<p>Maximum Burst Size. Specify the Maximum burst size supported on Tx port. MBS reflects the maximum transmission size when starting to transmit frames after a period of time in which the line was not busy. Units are in <math>10^3</math> bytes granularity. 0x000 - Limit the burst size to <math>10^3</math> bytes. 0x001 - Limit the burst size to <math>2 \cdot 10^3</math> bytes. ... 0x3FF - Limit the burst size to <math>1024 \cdot 10^3</math> bytes. <b>Note:</b> MBS must be programmed to a value which is bigger than the longest frame that the port may transmit.</p>
16-21	—	Reserved.
22-31	RLM	<p>Rate Limit. RLM is configured to the desired rate on Tx port. Specify the average transmission rate on Tx port. Units in <math>(16/F) \cdot 10^6</math> bits/sec granularity. F is the factor for compensation of the rate counter time unit. If TSB registers and FMBMTRLMTS[TSBS] programming are such that the Rate Counter time unit is 1 microsecond, then F = 1. Else, F = (Rate Counter time unit) / 1 microsecond. 0x000 - Limit the transmission rate to <math>(16/F) \cdot 10^6</math> bits/sec. 0x001 - Limit the transmission rate to <math>(32/F) \cdot 10^6</math> bits/sec. ... 0x3FF - Limit the transmission rate to <math>(16384/F) \cdot 10^6</math> bits/sec.</p>

#### 5.5.4.6.14 Tx Custom Classifier Base Register (FMBM\_TCCB)

The FMBM\_TCCB register configures the custom classifier base address. The value is written to CCBASE field in the IC as part of the IC initialization.



**Figure 5-107. Tx Custom Classifier Base Register (FMBM\_TCCB)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is  $0x1000 \cdot n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-126 describes the FMBM\_TCCB fields.

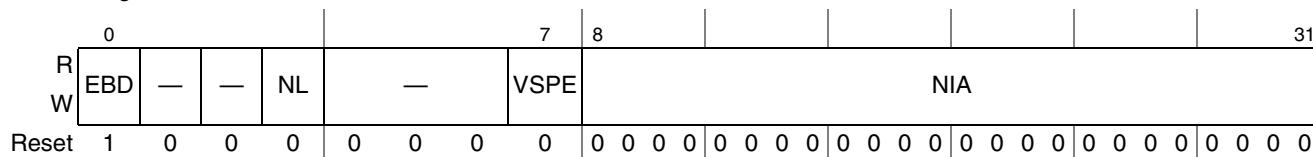
**Table 5-126. FMBM\_TCCB Field Descriptions**

Bits	Name	Description
0-31	CBASE	Custom Classifier Base Address. CBASE is written to CBASE field in the IC as part of the IC initialization process.

#### **5.5.4.6.15 Tx Frame Next Engine Register (FMBM\_TFNE)**

The FMBM\_TFNE register, shown in Figure 5-108, configures NIA issued by the BMI on FPM command bus once the frame has been fetched by the BMI from external memory. This allows for the flexibility of processing the frame before transmission. In addition, this register contains initial values for operational mode bits.

Offset<sup>1</sup> 0x0070 (FMBM\_TFNE\_<n>) Access: Read/Write  
offset <0x1000\*n>  
range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;



**Figure 5-108. Tx Frame Next Engine Register (FMBM\_TFNE)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000 \cdot n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

[Table 5-127](#) describes the FMBM\_TFNE fields.

**Table 5-127. FMBM\_TFNE Field Descriptions**

Bits	Name	Description
0	EBD	Bit description are the same as <a href="#">Context A—A0 Field Description for Tx Port</a> .
1-2	—	Reserved
3	NL	Bit description are the same as <a href="#">Context A—A2 Field Description for Tx Port</a> .
4-6	—	Reserved.
7	VSPE	Bit description are the same as <a href="#">Context A—A2 Field Description for Tx Port</a> .
8-31	NIA	<p>Next Invoked Action.</p> <p>This NIA is used by the BMI after frame fetch. In some cases the application needs to invoke the FMan Controller for manipulations before the frame is transmitted.</p> <p>For detailed description refer to <a href="#">Section 5.4.4, “Next Invoked Action (NIA)</a>.</p>

### **5.5.4.6.16 Tx Priority based Flow Control (PFC) Mapping Registers (FMBM\_TPFCM0)**

When a PFC flow control packet is received, the corresponding Traffic Class in QMan is paused. The feature is described in the QMan chapter [Section 3.3.5, “Traffic Class \(TC\) Flow Control”](#) and [“Traffic](#)

Class Flow Control Configure.” The FMBM\_TPFCM0,1 registers are used to configure the mapping of PFC class (in incoming PFC class enable vector) to QMan traffic class.

### NOTE

It is possible to map more than one CEVMx field to the same QMan traffic class (TC). In this case the TC is paused until all the respective pause conditions expire.

Offset <sup>1</sup>	0x0074 (FMBM_TPFCM0_<n>)	Access: Read/Write					
	offset <0x1000*n>						
	range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;						
R W	CEVM0	CEVM1					
Reset	0 0 0 0 0	0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 0 1 1 0 0 1 1 1					
	CEVM2	CEVM3	CEVM4	CEVM5	CEVM6	CEVM7	31

**Figure 5-109. Tx PFC Mapping Register 0 (FMBM\_TPFCM0)**

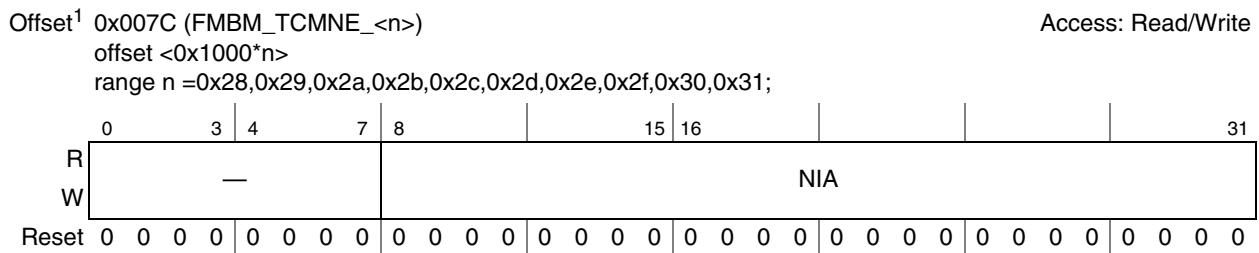
- <sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

**Table 5-128. FMBM\_TPFCM0 Descriptions**

Bits	Name	Description
0-3	CEVM0	PFC ‘Class Enable Vector (CEV’ bit 0 to QMan Traffic Class (TC) Mapping. Used to determine which QMan Traffic Class (TC) is paused. When set to n, and CEV bit 0 is set in the incoming PFC frame, the corresponding Traffic Class (TCn) in QMan is paused. <b>Note:</b> See QMan chapter for the number of TCs supported.
4-7	CEVM1	Same description as above for CEV bit 1.
8-11	CEVM2	Same description as above for CEV bit 2.
12-15	CEVM3	Same description as above for CEV bit 3.
16-19	CEVM4	Same description as above for CEV bit 4.
20-23	CEVM5	Same description as above for CEV bit 5.
24-27	CEVM6	Same description as above for CEV bit 6.
28-31	CEVM7	Same description as above for CEV bit 7.

### 5.5.4.6.17 Tx Continuous Mode Next Enqueue Register (FMBM\_TCMNE)

The FMBM\_TCMNE register, shown in [Figure 5-69](#), configures the Tx port, for a flow in which the same frame is enqueued to multiple queues. This register is used when ICAD[NL]=1 (i.e. this is Not the Last stage in the flow). The NIA is used by the BMI in the ‘Release Internal Resources’ stage Tx in normal mode, if ICAD[NL]=1.



**Figure 5-110. Tx Continuous Mode Next Engine (FMBM\_TCMNE)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx POID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

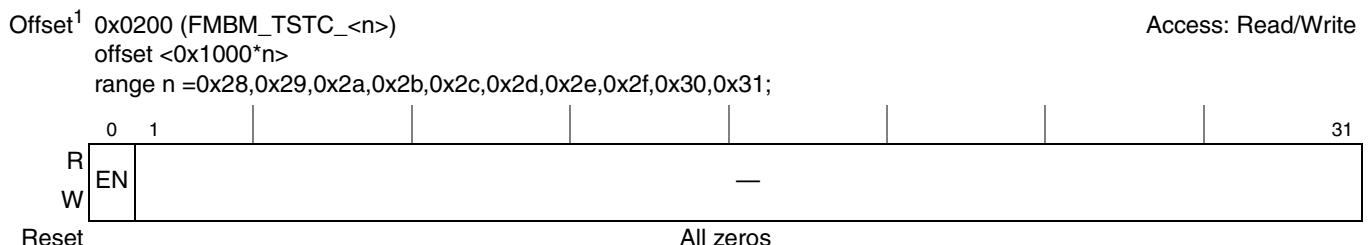
[Table 5-128](#) describes the FMBM\_TCMNE fields.

**Table 5-129. FMBM\_TCMNE Descriptions**

Bits	Name	Description
1-7	—	Reserved
8-31	NIA	If ICAD[NL]=0 the BMI end the flow at the end of this stage (no NIA is needed) If ICAD[NL]=1 this NIA is used for the next stage. The reset value is set for the FMan Controller.

#### 5.5.4.6.18 Tx Statistics Counters Register (FMBM\_TSTC)

The FMBM\_TSTC register, shown in [Figure 5-111](#), is used to control and enable Tx port statistics counters. Note that statistics counters are not affected by the assertion of soft reset, but they are affected by the assertion of hard reset.



**Figure 5-111. Tx Statistics Counters Register (FMBM\_TSTC)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx POID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

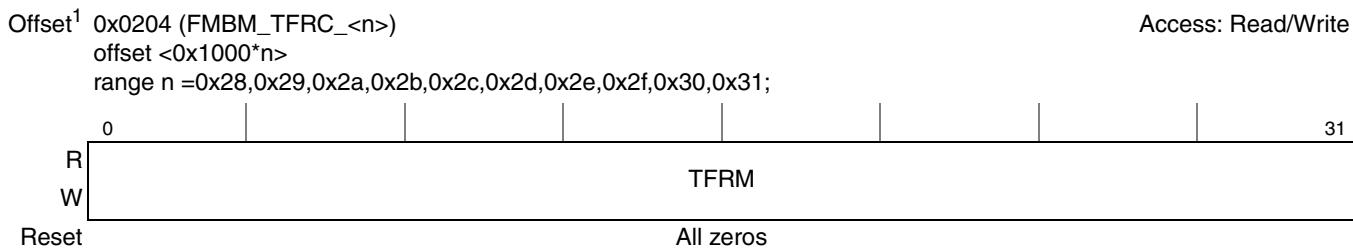
[Table 5-130](#) describes the FMBM\_TSTC fields.

**Table 5-130. FMBM\_TSTC Field Descriptions**

Bits	Name	Description
0	EN	Enable Statistics Counters. 0 - Statistics Counters are disabled. 1 - Statistics Counters are enabled.
1-31	—	Reserved.

#### **5.5.4.6.19 Tx Frame Counter Register (FMBM\_TFRC)**

The FMBM\_TFRC register, shown in Figure 5-112, counts the total number of frames flowing on the Tx port, regardless of whether they got transmitted or discarded.



**Figure 5-112. Tx Frame Counter Register (FMBM\_TFRC)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-131 describes the FMBM\_TFRC fields.

**Table 5-131. FMBM\_TFRC Field Descriptions**

Bits	Name	Description
0-31	TFRM	<p>Transmitted Frames.</p> <p>Counts the number of transmitted frames.</p> <p>The counter is enabled to count if FMBM_TSTC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### **5.5.4.6.20 Tx Frames Discard Counter Register (FMBM\_TFDC)**

The FMBM\_TFDC register, shown in Figure 5-113, counts the number of frames that are discarded due to dma error indication that is sensed during the process of frame payload or frame context loading.

Those frames are discarded and not sent to the MAC transmit interface. The frame appears on the confirmation queue if CFQID does not equal zero, or in error queue if CFQID equals zero. DME bit is set on the FD status word.



**Figure 5-113. Tx Frames Discard Counter Register (FMBM\_TFDC)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-132 describes the FMBM\_TFDC fields.

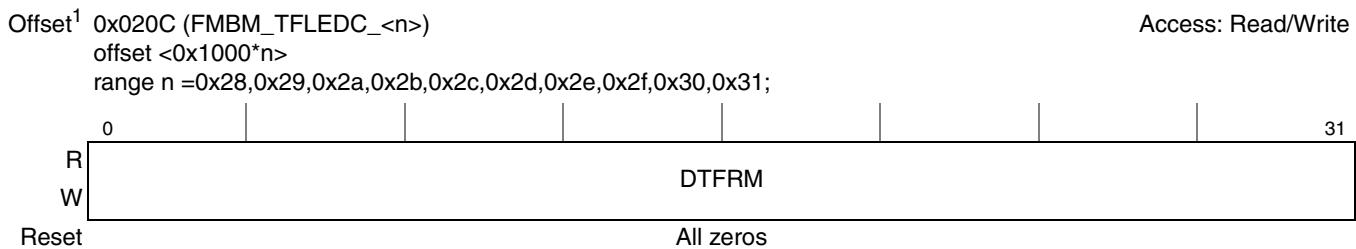
**Table 5-132. FMBM\_TFDC Field Descriptions**

Bits	Name	Description
0-31	DTFRM	<p>Discarded Transmit Frames. Counts the number of transmit frames that are discarded due to dma error. The counter is enabled to count if FMBM_TSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### 5.5.4.6.21 Tx Frames Length Error Discard Counter Register (FMBM\_TFLED)C

The FMBM\_TFLED register, shown in Figure 5-114, counts the number of frames that are discarded due to frame length error. Terms that define length error are described in Section 5.5.6.20.5, “Unsupported Frames.”

Those frames are discarded and not sent to the MAC transmit interface. The frame appears on the confirmation queue if CFQID does not equal zero, or in error queue if CFQID equals zero. The LGE bit is set on the FD status word.



**Figure 5-114. Tx Frames Length Error Discard Counter Register (FMBM\_TFLED)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx POID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-133 describes the FMBM\_TFLED fields.

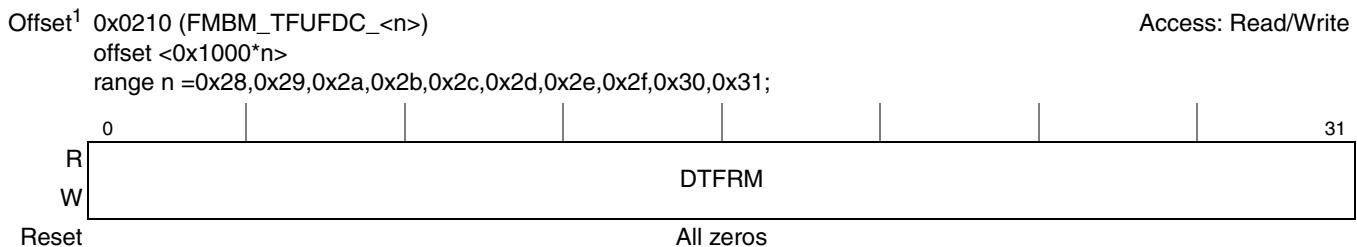
**Table 5-133. FMBM\_TFLED Field Descriptions**

Bits	Name	Description
0-31	DTFRM	<p>Discarded Transmit Frames. Counts the number of transmit frames that were discarded due to frame length error. The counter is enabled to count if FMBM_TSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

### 5.5.4.6.22 Tx Frames Unsupported Format Discard Counter Register (FMBM\_TFUFD)C

The FMBM\_TFUFD register, shown in Figure 5-115, counts the number of frames that were discarded due to frame format error - frame descriptor contains unsupported format.

Those frames are discarded and not sent to the MAC transmit interface. The frame appears on the confirmation queue if CFQID does not equal zero, or in error queue if CFQID equals zero. UFD bit is set on the FD status word.



**Figure 5-115. Tx Frames Unsupported Format Discard Counter Register (FMBM\_TFUFD)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

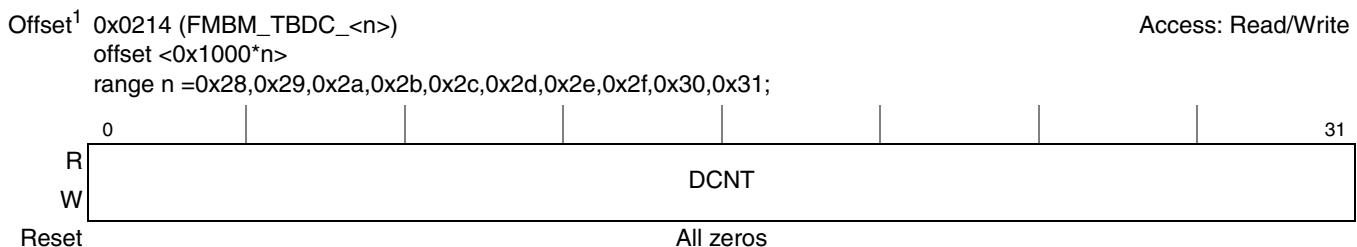
Table 5-134 describes the FMBM\_TFUFD fields.

**Table 5-134. FMBM\_TFUFD Field Descriptions**

Bits	Name	Description
0-31	DTFRM	<p>Discarded Transmit Frames. Counts the number of transmit frames that were discarded due to frame format error. The counter is enabled to count if FMBM_TSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

### 5.5.4.6.23 Tx Buffers Deallocate Counter Register (FMBM\_TBDC)

The FMBM\_TBDC register, shown in Figure 5-116, counts the number of buffer deallocate operations.



**Figure 5-116. Tx Buffers Deallocate Counter Register (FMBM\_TBDC)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-135 describes the FMBM\_TBDC fields.

**Table 5-135. FMBM\_TBDC Field Descriptions**

Bits	Name	Description
0-31	DCNT	<p>Buffers Deallocation Counter.</p> <p>Counts the number of buffer deallocate operations, that the port initiated. The counter increments whenever a buffer is returned to the BMAn pools, regardless of the BPID.</p> <p>The counter is enabled to count if FMBM_TSTC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### **5.5.4.6.24 Tx Performance Counters Register (FMBM\_TPC)**

The FMBM\_TPC register, shown in Figure 5-117, is used to control and enable Tx port performance counters.

Offset <sup>1</sup>	0x0280 (FMBM_TPC_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;	
0	1	
R	EN	
W		—
Reset	All zeros	

**Figure 5-117. Tx Performance Counters Register (FMBM\_TPC)**

<sup>1</sup> There is one register for each Tx port.  $<n>$  is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

[Table 5-136](#) describes the FMBM TPC fields.

**Table 5-136. FMBM\_TPC Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0	EN	Enable Performance Counters. 0 - Performance Counters are disabled. 1 - Performance Counters are enabled.
1-31	—	Reserved.

#### 5.5.4.6.25 Tx Performance Count Parameters Register (FMBM\_TPCP)

This register is for internal use for debug.

The FMBM\_TPCP register, shown in [Figure 5-118](#), is used to define the parameters of the Tx port performance counters.

Offset <sup>1</sup>	0x0284 (FMBM_TPCP_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;	
	0 1 2   7 8   12 13 15   16   19   20 21 22         31	
R	—   TCV   —   TCCV   DCV   —   FUCV	
W		All zeros
Reset		

**Figure 5-118. Tx Performance Count Parameters Register (FMBM\_TPCP)**

- <sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map”](#).

[Table 5-137](#) describes the FMBM\_TPCP fields.

**Table 5-137. FMBM\_TPCP Field Descriptions**

Bits	Name	Description
0-1	—	Reserved.
2-7	TCV	Tasks compare value. Used to set the criteria for concurrent tasks pipeline counter. The counter counts the number of cycles where the criteria is met. 0x00 - Number of tasks allocated for the port is equal or greater than 1. 0x01 - Number of tasks allocated for the port is equal or greater than 2. ... 0x3F- Number of tasks allocated for the port is equal or greater than 64.
8-12	—	Reserved.
13-15	TCCV	Transmit Confirm Compare Value. Used to set the criteria for the transmit confirm queue counter. The transmit confirm queue holds descriptors of frames which were already sent to the MAC and wait for transmit time stamp to be returned. The counter counts the number of cycles where the criteria is met. 000 - Number of frames in transmit confirm queue is equal or greater than 1. 001 - Number of frames in transmit confirm queue is equal or greater than 2. ... 111 - Number of frames in transmit queue is equal or greater than 8.
16-19	DCV	DMA Compare Value. Used to set the criteria for open DMA counter. The counter counts the number of cycles where the criteria is met. 0000 - Number of open DMA is equal or greater than 1. 0001 - Number of open DMA is equal or greater than 2. ... 1111- Number of open DMA is 16.

**Table 5-137. FMBM\_TPCP Field Descriptions (continued)**

Bits	Name	Description
20-21	—	Reserved.
22-31	FUCV	<p>FIFO Utilization Compare Value.            Used to set the criteria for FIFO size counter.            The counter counts the number of cycles where the FIFO size is equal or greater than FUCV.            The FIFO size compare value is in 256 bytes granularity (Internal buffer size).            0x000 - Number of utilized buffers in port's FIFO is equal or greater than 1 buffer (256 bytes).            0x001 - Number of utilized buffers in port's FIFO is equal or greater than 2 buffers (512 bytes).            ...            0x3FF - Number of utilized buffers in port's FIFO is equal or greater than 1024 buffers (256k bytes).</p>

### **5.5.4.6.26 Tx Cycle Counter Register (FMBM\_TCCN)**

The FMBM\_TCCN register, shown in Figure 5-119, is used to count clock cycles.

Offset<sup>1</sup> 0x0288 (FMBM\_TCCN\_<n>) Access: Read/Write  
 offset <0x1000\*n>  
 range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;

0 31  
 R W CNUM  
 Reset All zeros

**Figure 5-119. Tx Cycle Counter Register (FMBM\_TCCN)**

<sup>1</sup> There is one register for each Tx port.  $<n>$  is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map”](#).

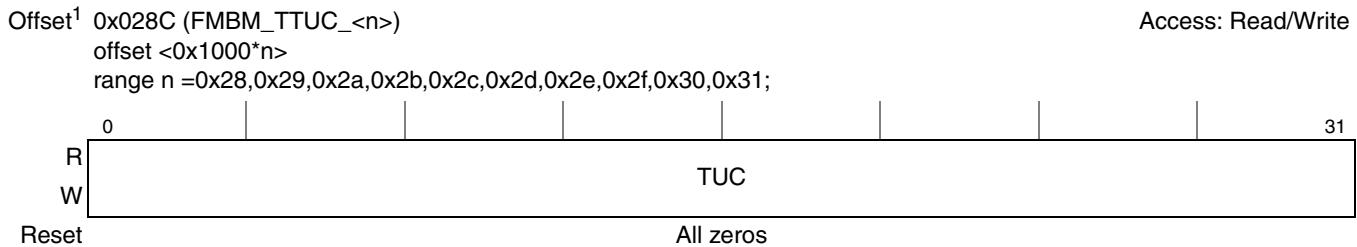
Table 5-138 describes the FMBM\_TCCN fields.

**Table 5-138. FMBM\_TCCN Field Descriptions**

Bits	Name	Description
0-31	CNUM	<p>Cycle Number.</p> <p>Counts the number of BMI clock cycles.</p> <p>The counter is enabled to count if FMBM_TPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

### 5.5.4.6.27 Tx Tasks Utilization Counter Register (FMBM\_TTUC)

The FMBM\_TTUC register, shown in [Figure 5-120](#), is used to monitor tasks utilization.



**Figure 5-120. Tx Tasks Utilization Counter Register (FMBM\_TTUC)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

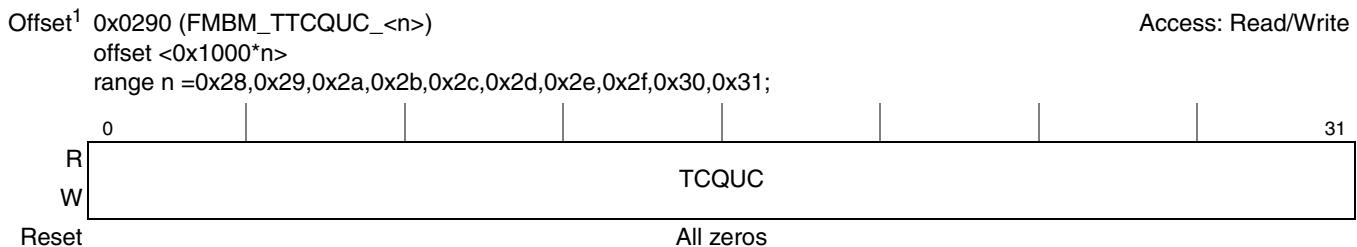
[Table 5-139](#) describes the FMBM\_TTUC fields.

**Table 5-139. FMBM\_TTUC Field Descriptions**

Bits	Name	Description
0-31	TUC	Tasks Utilization Counter. Counts the number of cycles in which the number of tasks allocated for the port is equal or greater than the number specified in FMBM_TPCP[TCV]. The counter is enabled to count if FMBM_TPC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

### 5.5.4.6.28 Tx Transmit Confirm Queue Utilization Counter Register (FMBM\_TTCQUC)

The FMBM\_TTCQUC register, shown in [Figure 5-121](#), is used to monitor the transmit confirm queue utilization.



**Figure 5-121. Tx Transmit Confirm Queue Utilization Counter Register (FMBM\_TTCQUC)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-140](#) describes the FMBM\_TTCQUC fields.

**Table 5-140. FMBM\_TTCQUC Field Descriptions**

Bits	Name	Description
0-31	TCQUC	<p>Transmit Confirm Queue Utilization Counter.</p> <p>Counts the number of cycles in which the number of frames in transmit confirm queue is equal or greater than the number specified in FMBM_TPCP[TCCV].</p> <p>The transmit confirm queue holds descriptors of frames which were already sent to the MAC and wait for transmit time stamp to be returned.</p> <p>The counter is enabled to count if FMBM_TPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

### **5.5.4.6.29 Tx DMA Utilization Counter Register (FMBM\_TDUC)**

The FMBM\_TDUC register, shown in Figure 5-122, is used to monitor the DMA utilization.

Offset <sup>1</sup>	0x0294 (FMBM_TDUC_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;	
0		31
R	DUC	
W		
Reset	All zeros	

**Figure 5-122. Tx DMA Utilization Counter Register (FMBM\_TDUC)**

<sup>1</sup> There is one register for each Tx port.  $<n>$  is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

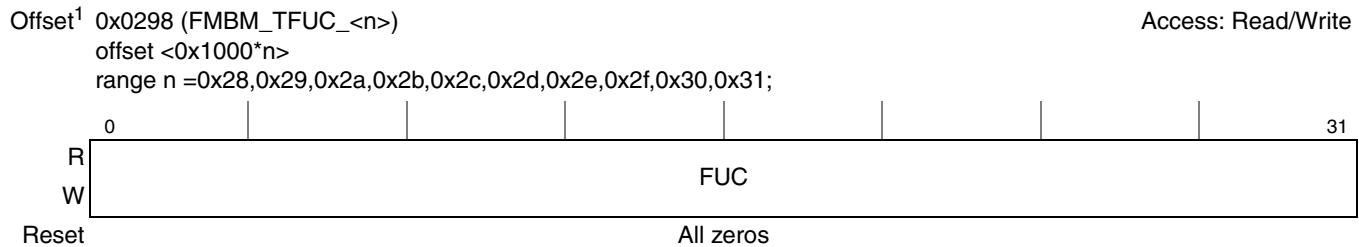
Table 5-141 describes the FMBM\_TDUC fields.

**Table 5-141. FMBM\_TDUC Field Descriptions**

Bits	Name	Description
0-31	DUC	<p>DMA Utilization Counter.</p> <p>Counts the number of cycles in which the number of open DMA is equal or greater than the number specified in FMBM_TPCP[DCV].</p> <p>The counter is enabled to count if FMBM_TPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

### 5.5.4.6.30 Tx FIFO Utilization Counter Register (FMBM\_TFUC)

The FMBM\_TFUC register, shown in [Figure 5-123](#), is used to monitor FIFO utilization.



**Figure 5-123. Tx FIFO Utilization Counter Register (FMBM\_TFUC)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-142](#) describes the FMBM\_TFUC fields.

**Table 5-142. FMBM\_TFUC Field Descriptions**

Bits	Name	Description
0-31	FUC	FIFO Utilization Counter. Counts the number of cycles in which actual used FIFO size is equal or greater than the size specified in FMBM_TPCP[FUCV]. The counter is enabled to count if FMBM_TPC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

### 5.5.4.6.31 Tx Debug Configuration Register (FMBM\_TDCFG)

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires debug.

The FMBM\_TDCFG register, shown in [Figure 5-124](#), sets the condition in which BMI trap is considered a match, the level of trace that the BMI should add in debug trace portion of the IC, and the action to be taken if a debug mark is set when a frame is ready to be enqueued. There is one register per debug flow. per debug flow. See [Section 5.5.6.22, “Debug Capabilities,”](#) for more details about debug capabilities.

Note that debug trap compare value and mask registers are common to all ports and they are defined in the common registers group.

#### NOTE

DTO field is not relevant and not used in flows B and C.

Offset<sup>1</sup> 0x0300 (FMBM\_TDCFG\_<n>\_<y>)  
 offset <4096\*n+4\*(y-1)>  
 range n  
 =0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31;  
 y=1..3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16			23	24	27	28	31
R	—	CMPOP	—	TL	—	TR_DST	—	HALT	—	—	—	—	—	—	—	—	—	—	DTO	—	—	—	—	—
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

**Figure 5-124. Tx Debug Configuration Register (FMBM\_TDCFG)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 4\*(y-1) + 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#) ‘y’=1,2,3 corresponds to Flow A, B,C respectively.

[Table 5-143](#) describes the FMBM\_TDCFG fields.

**Table 5-143. FMBM\_TDCFG Field Descriptions**

Bits	Name	Description
0	—	Reserved
1-3	CMPOP	Compare Operator. Selects the trap condition. 000 - Trap Disabled (never match) 001 - Always match (referred as trap bypass) 010 - Match if (trap comp & trap mask) equals to the (frame FD & trap mask). 011 - 111 - reserved
4-5	—	Reserved.
6-7	TL	Trace Level. Selects the amount of trace data to be written in debug portion if a trap match was found. 00 - Trace disable 01 - Minimum trace 10 - verbose trace 11 - very verbose trace
8-9	—	Reserved.
10-11	TR_DST	Trace Destination. If debug mark remained set when the BMI is in preparation for enqueue action, the BMI signals DMA to copy the debug portion in IC. The destination of this copy depends on the programming below. Note that if system port is selected, it is the user responsibility to keep enough margin in the beginning of the frame’s buffer, to avoid debug data write over the frame data. 00 - Debug trace written to memory. 01 - Debug trace routed to debug port (Nexus). 10 - Reserved 11 - Reserved
12-13	—	Reserved.
14-15	HALT	Halt execution. If debug mark remained set when the BMI is before enqueue NIA is about to be issued, the BMI may indicate to FPM that halt is desired, according to the programming below. 00 - No halt. normal operation continues. 01 - stop this task if debug mark remained set. 10 - stop this port if debug mark remained set. 11 - stop all ports if debug mark remained set.

**Table 5-143. FMBM\_TDCFG Field Descriptions (continued)**

Bits	Name	Description
16-23	—	Reserved.
24-27	DTO	Debug Trace Offset. Sets the offset of the debug trace from the beginning of the IC buffer. 0000 - 0111 - Reserved. 1000 - Debug trace starts at offset 0x80 from the beginning of the IC buffer. 1001 - Debug trace starts at offset 0x90 from the beginning of the IC buffer. 1010 - Debug trace starts at offset 0xa0 from the beginning of the IC buffer. 1011 - Debug trace starts at offset 0xb0 from the beginning of the IC buffer. 1100 - Debug trace starts at offset 0xc0 from the beginning of the IC buffer. 1101 - Debug trace starts at offset 0xd0 from the beginning of the IC buffer. 1110 - Debug trace starts at offset 0xe0 from the beginning of the IC buffer. 1111 - Debug trace starts at offset 0xf0 from the beginning of the IC buffer.
28-31	—	Reserved.

#### **5.5.4.6.32 Tx General Purpose Register (FMBM\_TGPR)**

The FMBM\_TGPR register, shown in Figure 5-90, is used as a general purpose register. This register does not serve a specific purpose. It may be used for application specific purposes.

**Figure 5-125. Tx General Purpose Register (FMBM\_TGPR)**

<sup>1</sup> There is one register for each Tx port. <n> is the Tx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is  $0x1000 \cdot n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

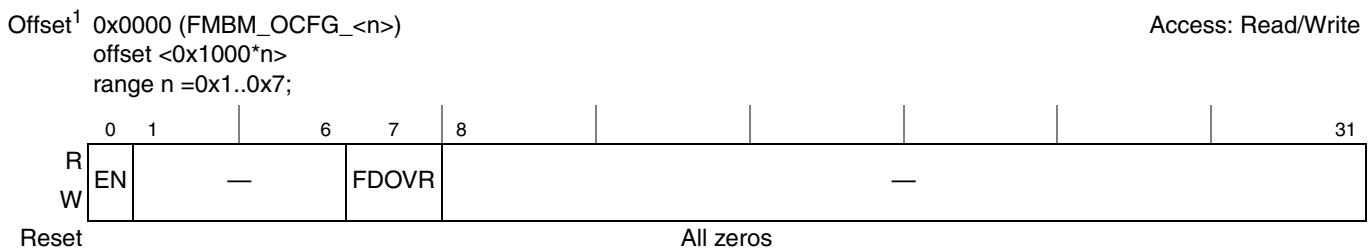
**Table 5-144. FMBM TGPR Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0-31	GPR	General Purpose Register value.

## 5.5.4.7 Offline Port/Host Command Port Registers Description

### 5.5.4.7.1 Offline Port/Host Command (O/H) Configuration Register (FMBM\_OCFG)

The FMBM\_OCFG register, shown in [Figure 5-126](#), is used to enable and configure Offline Port/Host command port.



**Figure 5-126. Offline Port/Host Command (O/H) Configuration Register (FMBM\_OCFG)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-145](#) describes the FMBM\_OCFG fields.

**Table 5-145. FMBM\_OCFG Field Descriptions**

Bits	Name	Description
0	EN	Enable. 0 – Port is disabled. If changed to 0 during task processing, the BMI does not dequeue new frames or commands from the QMan, and completes the ongoing tasks. 1 - Port is enabled <b>Note:</b> If the port is disabled dynamically (EN=0), the user must ensure that other fields in this register maintain the same value as long as FMBM_OST[BSY]=1. Only after FMBM_OST[BSY]=0, the user may change the values of the other fields in this register.
2-6	—	Reserved
7	FDOVR	Frame Discard Override. Set the BMI treatment when frame should be discarded due to erroneous condition detected in the process of the frame reception or processing. 0 - Discard the frame. 1 - Enqueue the frame to error queue configured in FMBM_OEFQID[EFQID].
8-31	—	Reserved.

#### **5.5.4.7.2 O/H Status Register (FMBM\_OST)**

The FMBM\_OST register, shown in Figure 5-127, is used to read status of Offline Port/Host command port.

Offset<sup>1</sup> 0x0004 (FMBM\_OST\_<n>)

## Access: Read/Write

offset <0x1000\*n>

range n =0x1..0x7;

	0							9	10		15	16					31
R	BSY							OPID									
W															—		
Reset	0	0	0	0	0	0	0	0	0	$n^2$	$n$	$n$	$n$	$n$	0	0	0

**Figure 5-127. O/H Status Register (FMBM OST)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H Port ID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

<sup>2</sup> OPID reset value is set to hard wired port ID (PortID).

**Table 5-146. FMBM\_OST Reset Values**

<b>Reg Name</b>	<b>Reset Value</b>
FMBM_OST_1	0000_0000_0000_0001_0000_0000_0000_0000
FMBM_OST_2	0000_0000_0000_0010_0000_0000_0000_0000
FMBM_OST_3	0000_0000_0000_0011_0000_0000_0000_0000
FMBM_OST_4	0000_0000_0000_0100_0000_0000_0000_0000
FMBM_OST_5	0000_0000_0000_0101_0000_0000_0000_0000
FMBM_OST_6	0000_0000_0000_0110_0000_0000_0000_0000
FMBM_OST_7	0000_0000_0000_0111_0000_0000_0000_0000

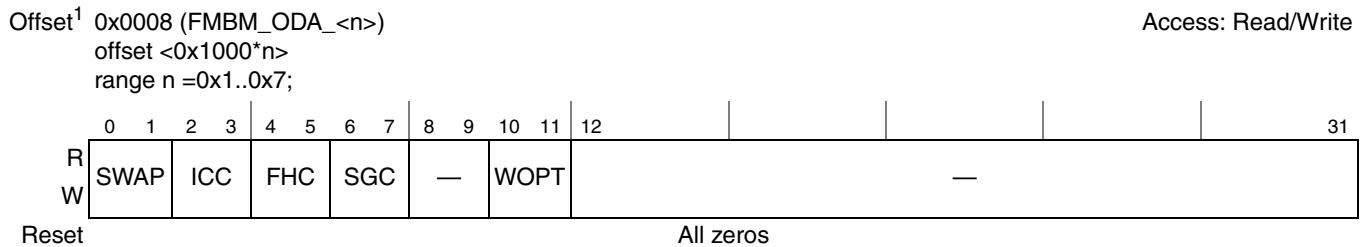
[Table 5-147](#) describes the FMBM\_OST fields.

**Table 5-147. FMBM\_OST Field Descriptions**

Bits	Name	Description
0	BSY	<p>Busy.</p> <p>BSY is asserted while frames are processed in the port.</p> <p>When Enable is cleared, BSY should be read in order to detect that port is not busy.</p> <p>0 - O/H port is not busy.</p> <p>1 - O/H port is busy.</p>
1-9	—	Reserved
10-15	OPID	<p>O/H Port ID.</p> <p>This is the hardware port ID associated with the register.</p>
16-31	—	Reserved

### 5.5.4.7.3 O/H DMA Attributes Register (FMBM\_ODA)

The FMBM\_ODA register, shown in [Figure 5-128](#), is used to set DMA attributes to be used for the port. Note that read operations always generate cacheable/no allocate and write of payload. Also, the frame header always generates cacheable/no stash attributes.



**Figure 5-128. O/H DMA Attributes Register (FMBM\_ODA)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H Poid. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-148](#) describes the FMBM\_ODA fields.

**Table 5-148. FMBM\_ODA Field Descriptions**

Bits	Name	Description
0-1	SWAP	Swap payload data. 00 - No swap, transfer data as is. 01 - 11 Reserved
2-3	ICC	IC write cache attributes. Used when DMA copies IC from FMan memory to external memory 00 - Cacheable, no Allocate (No Stashing). 01 - Cacheable and Allocate (Stashing on). 10, 11 - Reserved.
4-5	FHC	Frame Header write cache attributes. Used when DMA copies frame header from FMan memory to external memory 00 - Cacheable, no Allocate (No Stashing). 01 - Cacheable and Allocate (Stashing on). 10, 11 - Reserved.
6-7	SGC	Scatter/Gather write cache attributes. Used when DMA copies the scatter/gather list from FMan memory to external memory 00 - Cacheable, no Allocate (No Stashing). 01 - Cacheable and Allocate (Stashing on). 10, 11 - Reserved.
8-9	—	Reserved

**Table 5-148. FMBM\_ODA Field Descriptions (continued)**

Bits	Name	Description
10-11	WOPT	Optimize on write (of the tail of the payload) 00 - No optimization is done. 01 - Write optimization is enabled. See more details in <a href="#">Table 5-69., “FMBM_RDA Field Descriptions”</a> .
12-31	—	Reserved

#### 5.5.4.7.4 O/H Internal Context Parameters Register (FMBM\_OICP)

The FMBM\_OICP register, shown in [Figure 5-129](#), used to configure internal context parameters. These parameters supply the mechanism to tell the BMI which portion of the IC block should be copied from external memory prior to the frame data copy and which portion is written back to external memory once the processing of the frame was finished. Note that the IC read is enabled by FD Command [RPD] bit.

##### NOTE

The user must use caution when configuring the values in this register, so the FD field in the IC is NOT overwritten by this action.

##### NOTE

If FMBM\_OIM[FOF]!=0 the parser result offsets do not point to the correct headers once the frame is in the FMan internal memory. Therefore the parser results cannot be used, and should not be read from the frame margin.

##### NOTE

In FMan\_v3, this register is used if VSPE=0. If VSPE = 1 the FMBM\_VICP from the selected storage profile is used.

##### NOTE

In FMan\_v3, it is possible that the portion of the IC which is written (during enqueue) is different than the portion that is read (during dequeue). This is because the storage profile being used for enqueue and dequeue may be different. On read, the storage profile ID is absolute (ASPID), while during write the storage profile ID is relative (RSPID). During write the absolute SPID is evaluated by using FMBM\_SPICID[SPBRN] and FMBM\_SPICID[SPNUM].

Offset <sup>1</sup>	0x000C (FMBM_OICP_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x1..0x7;	
0	—	
R	—	ICEOF
W	—	—
Reset	All zeros	ICIOF
		ICSZ
		31

**Figure 5-129. O/H Internal Context Parameters Register (FMBM\_OICP)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-149 describes the FMBM\_OICP fields.

**Table 5-149. FMBM\_OICP Field Descriptions**

Bits	Name	Description
0-10	—	Reserved
11-15	ICEOF	<p>Internal Context External Offset. Specifies the offset in external buffer in which internal context is transferred to or from, 16 bytes granularity. The offset is added to the address that is written in the frame descriptor. 00000 - IC is transferred to offset 0x0 in external buffer. 00001 - IC is transferred to offset 0x10 in external buffer. ... 11111 - IC is transferred to offset 0x1F0 in external buffer.</p>
16-19	—	Reserved
20-23	ICIOF	<p>Internal Context Internal Offset. Specifies the offset in internal context from which data is transferred from the internal context to external buffer, 16 bytes granularity. Together with ICSZ used to specify the internal context segment transferred to external buffer. The sum of ICIOF and ICSZ should not cross the IC block boundary (256 bytes from the beginning of the block). 0000 - IC data to be transferred is from the beginning of the IC block, no offset. 0001 - IC data to be transferred is from offset 0x10 from the beginning of the IC block. ... 1111 - IC data to be transferred is from offset 0xF0 from the beginning of the IC block.</p>
24-26	—	Reserved
27-31	ICSZ	<p>Internal Context copy Size. Specifies the size of internal context transferred to or from external buffer. Together with ICIOF used to specify the internal context segment transferred to external buffer. 16 bytes granularity. (up to 256 bytes). 00000 - IC segment is not transferred. 00001 - IC segment to be transferred size is 16 bytes. 00010 - IC segment to be transferred size is 32 bytes. ... 10000 - IC segment to be transferred size is 256 bytes. 10001 - 11111 - reserved.</p>

### **5.5.4.7.5 O/H Frame Dequeue Next Engine Register (FMBM\_OFDNE)**

The FMBM\_OFDNE register, shown in Figure 5-130, configures NIA issued by the BMI when the task is initialized. Free TNUMs allocated by the BMI is sent to the next module according to specified NIA. In normal mode NIA should be set to QMI.

**Figure 5-130. O/H Frame Dequeue Next Engine Register (FMBM\_OFDNE)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-150 describes the FMBM\_OFDNE fields.

**Table 5-150. FMBM\_OFDNE Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	NIA	Next Invoked Action. When the task is initialized, the BMI issues the specified NIA. Default value is set to 0x580000, which means that the next module is QMI dequeue. For detailed description refer to <a href="#">Section 5.4.4, “Next Invoked Action (NIA)”</a> .

#### **5.5.4.7.6 O/H Frame Next Engine Register (FMBM\_OFNE)**

The FMBM\_OFNE register, shown in Figure 5-131, configures NIA issued by the BMI when the frame is ready to be processed by the next module.

**Figure 5-131. O/H Frame Dequeue Next Engine Register (FMBM\_OFNE)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H Port ID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-151 describes the FMBM\_OFNE fields.

**Table 5-151. FMBM\_OFNE Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	NIA	Next Invoked Action. When the frame is ready to be processed, the BMI issues the specified NIA. Default value is set to 0x440000, which means that the next module is hardware parser, and the protocol is Ethernet. For detailed description refer to <a href="#">Section 5.4.4, “Next Invoked Action (NIA).”</a>

#### 5.5.4.7.7 O/H Frame Attributes Register (FMBM\_OFCA)

The FMBM\_OFCA register, shown in [Figure 5-132](#), configures frame attributes. The attributes are used when the frame task is moved to the next module for the first time. The COLOR attribute value is used at the FD status color field initialization in the IC of the frame.

Offset<sup>1</sup> 0x0018 (FMBM\_OFCA\_<n>)  
offset <0x1000\*n>  
range n =0x1..0x7;

Access: Read/Write

		0	3	4	5	6	7	8	9	10	15	16	23	24	31
R		OR	—	COLOR	SYNC	—	—	MR			—				
Reset	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0

**Figure 5-132. O/H Frame Attributes Register (FMBM\_OFCA)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H POLID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-152 describes the FMBM\_OFCA fields.

**Table 5-152. FMBM\_OFCA Field Descriptions**

Bits	Name	Description
0	OR	Order definition (OD). Determines the OD. If OR is set, the FPM attaches the ordering attributes to the task. Later, the order restoration may be requested for this task before the task enters an order-sensitive state (for example, enqueue). 0 - No order definition is needed. 1 - Order definition is needed.
1-3	—	Reserved
4-5	COLOR	Default color. Will determine the default color of the frame. Color can be altered by frame processing modules as needed. 00 - Green 01 - Yellow 10 - Red 11 - Override. Enqueue regardless of RED/WRED/tail drop thresholds.

**Table 5-152. FMBM\_OFCA Field Descriptions (continued)**

Bits	Name	Description
6-7	SYNC	<p>Synchronization attributes</p> <p>00 - Inactive. The synchronization attributes of the task is unchanged.</p> <p>01 - Reserved.</p> <p>10 - Synchronability request. The current task is attached to the tail of FPM synchronization queue.</p> <p>11 - Reserved.</p> <p><b>Note:</b> The value of 10 is allowed only for ports which are doing dynamic table updates (see <a href="#">Section 5.12, “Frame Manager—FMan Controller”</a>).</p>
8-9	—	Reserved
10-15	MR	<p>Mode Attributes.</p> <p>These bits define internal modes of operation in the FMan controller and updates the MR[0:5] when this task is executed on the FMan controller.</p>
16-31	—	Reserved

### **5.5.4.7.8 O/H Frame Parser Next Engine Register (FMBM\_OFPNE)**

The FMBM\_OFPNE register, shown in Figure 5-133, is used to configure NIA used by Hardware Parser when dispatching task to the next module. HPNIA is written to Internal Context.

**Figure 5-133. O/H Frame Dequeue Next Engine Register (FMBM\_OFPNE)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

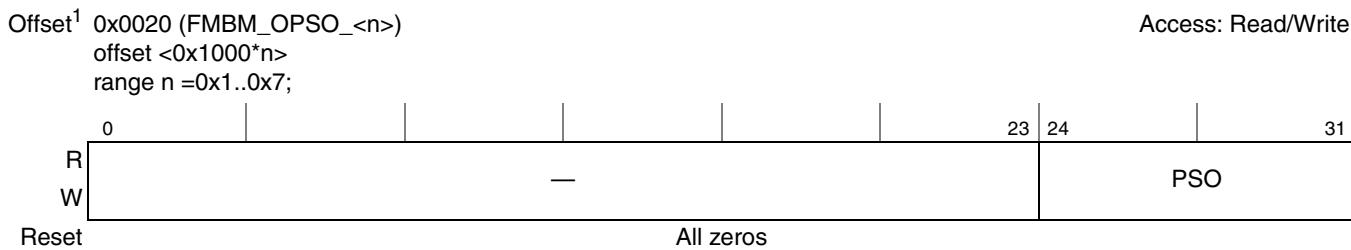
Table 5-153 describes the FMBM\_OFPNE fields.

**Table 5-153. FMBM\_OFPNE Field Descriptions**

Bits	Name	Description
0-7	—	Reserved.
8-31	HPNIA	<p>Hardware Parser NIA (Next Invoked Action). Set the NIA used by Hardware Parser to jump to the next module (followed by Hardware Parser). HPNIA is written by the BMI to Internal Context. Default value is set to 0x480000, which means that the next module is KeyGen.</p> <p>For detailed description refer to <a href="#">Section 5.3.5, “FMan User-Configurable Pipeline Architecture—Introducing the NIA.”</a></p>

### **5.5.4.7.9 O/H Parsing Start Offset Register (FMBM\_OPSo)**

The FMBM\_OPPO register, shown in Figure 5-134, defines an offset to the first buffer of a frame. This offset indicates to Hardware Parser the parse start point.



**Figure 5-134. O/H Parsing Start Offset Register (FMBM\_OPSO)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

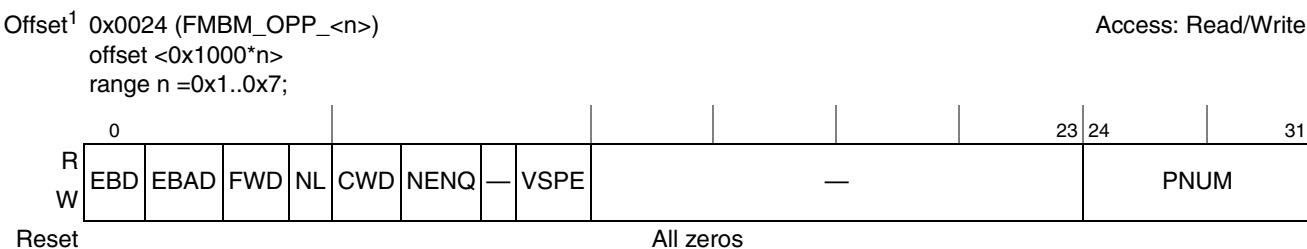
Table 5-154 describes the FMBM\_OPSO fields.

**Table 5-154. FMBM\_OPPO Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0-23	—	Reserved.
24-31	PSO	<p>Parsing Start Offset.</p> <p>Specify to Hardware Parser the offset, in first frame's buffer, to parsing start point.</p> <p>0x00 - Parsing start offset in frame's first buffer is 0.</p> <p>0x01 - Parsing start offset in frame's first buffer is 1 byte.</p> <p>0x02 - Parsing start offset in frame's first buffer is 2 bytes.</p> <p>...</p> <p>0xFF - Parsing start offset in frame's first buffer is 255 bytes.</p>

#### **5.5.4.7.10 O/H Policer Profile Register (FMBM\_OPP)**

The FMBM\_FROPP register, shown in [Figure 5-135](#), configures the default Policer profile issued by the BMI. In addition, this register contains initial values for the operational mode bits. The configuration bits in the ICAD may be overwritten by values in Context A of the FQD when the frame is dequeued.



**Figure 5-135. O/H Policer Profile Register (FMBM\_OPP)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 \times n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-155 describes the FMBM\_FROPP fields.

**Table 5-155. FMBM\_OPP Field Descriptions**

Bits	Name	Description
0	EBD	Bit description are the same as <a href="#">5.4.3.1.1, “Frame Queue Descriptor (FQD) Context A”</a> (using appropriate registers for Rx port).
1	EBAD	
2	FWD	
3	NL	
4	CWD	
5	NENQ	
6	—	
7	VSPE	
8-23	—	Reserved.
24-31	PNUM	Policer Profile. Initiate Policer Profile in Internal Context A0 field. For further information refer to <a href="#">Section 5.11, “Frame Manager—Policer.”</a>

#### **5.5.4.7.11 O/H Custom Classifier Base Register (FMBM\_OCCB)**

The FMBM\_OCCB register, shown in Figure 5-136, configures the custom classifier base address. The value is written to CBASE field in the IC as part of the IC initialization.

**Figure 5-136. O/H Custom Classifier Base Register (FMBM\_OCCB)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-156 describes the FMBM\_OCCB fields.

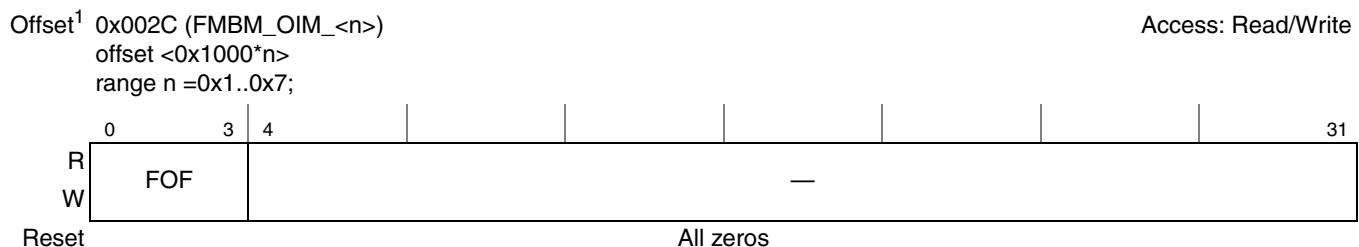
**Table 5-156. FMBM\_OCCB Field Descriptions**

Bits	Name	Description
0-31	CBASE	Custom Classifier Base Address. Cbase is written to Cbase field in the IC as part of the IC initialization process.

#### **5.5.4.7.12 O/H Internal Margins Register (FMBM\_OIM)**

The FMBM\_OIM register, shown in Figure 5-137, used to configure internal margins, which are useful if the internal FMan controller is required to add or remove content to the received frame header. Note that

if content is added or removed the frame offset in the FD is updated by the FMan controller. See [Section 5.5.6.4, “Internal and External Margins.”](#)



**Figure 5-137. O/H Internal Margins Register (FMBM\_OIM)**

- <sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-157](#) describes the FMBM\_OIM fields.

**Table 5-157. FMBM\_OIM Field Descriptions**

Bits	Name	Description
0-3	FOF	<p>Frame Offset. Specify the offset in the internal buffer that is used when copying the frame header into FMan memory. 16 bytes granularity (up to 64 bytes).</p> <ul style="list-style-type: none"> <li>0000 - Frame is written to first buffer start address.</li> <li>0001 - Frame is written to first buffer offset 0x10 (16 bytes).</li> <li>0010 - Frame is written to first buffer offset 0x20 (32 bytes).</li> <li>0011 - Frame is written to first buffer offset 0x30 (48 bytes).</li> <li>0100 - Frame is written to first buffer offset 0x40 (64 bytes).</li> <li>0101 - Frame is written to first buffer offset 0x50 (80 bytes).</li> <li>0110 - Frame is written to first buffer offset 0x60 (96 bytes).</li> <li>0111 - Frame is written to first buffer offset 0x70 (112 bytes).</li> <li>1000 - Frame is written to first buffer offset 0x80 (128 bytes).</li> <li>1001 - Frame is written to first buffer offset 0x90 (144 bytes).</li> <li>1010 - Frame is written to first buffer offset 0xa0 (160 bytes).</li> <li>1011 - Frame is written to first buffer offset 0xb0 (176 bytes).</li> <li>1100 - Frame is written to first buffer offset 0xc0 (192 bytes).</li> <li>1101 - 1111 - Reserved.</li> </ul> <p><b>Note:</b> Including the FOF, all frame headers parsed by the FMan parser, must reside in the first internal buffer (256B). For example, if the FMan parser examines headers in the first 90 bytes of the frame, the maximum value of FOF is 1010 (for 160 bytes) since 256-90=166.</p> <p>See <a href="#">Section 5.5.6.4.2, “External Margins”</a> and <a href="#">Section 5.5.6.4.3, “External memory accesses optimization”</a> for more details.</p>
4-31	—	Reserved

### 5.5.4.7.13 O/H FIFO Parameters Register (FMBM\_OFP)

The FMBM\_OFP Register sets O/H port fifo dequeue pipeline depth.

This register is for internal use. It is recommended not to modify its reset value (except for DPDE see the table below). This value can be modified in a system which requires fine tuning or debug.

The configuration of FMBM\_OFP can be altered per need during system operation, even while the ports are enabled and moving data. An exception to this is a change of DPDE in which the new written value is lower than the previous. Such change requires the port to be disabled and not busy (FMBM\_OST[BSY] is read ‘0’) before the configuration change operation. In case DPDE value is increased the port FMBM\_PFS configured IFSZ may also have to be increased to satisfy the internal FIFO configuration requirements. For details refer to [Section 5.5.6.18, “Internal FIFO Configuration Requirements.”](#)

**Figure 5-138. O/H FIFO Parameters Register (FMBM\_OFP)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. See Section 5.4.1, “FMan Hardware Ports,” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

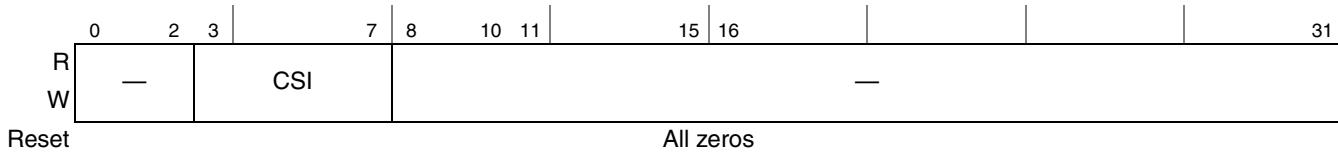
**Table 5-158. FMBM OFP Field Descriptions**

Bits	Name	Description
0-15	—	Reserved
16-19	DPDE	<p>Dequeue Pipeline Depth.</p> <p>Set the number of outstanding dequeue requests.</p> <p>0000 - Pipeline depth is 1 Recommended for port bandwidth of 1Gbps</p> <p>0001 - Pipeline depth is 2 Recommended for port bandwidth of 2.5Gbps.</p> <p>0010 - Pipeline depth is 3.</p> <p>0011 - Pipeline depth is 4.</p> <p>...</p> <p>0101 - Pipeline depth is 6 frames.</p> <p>0110 - 1111 - Reserved</p> <p><b>Note:</b> There is a restriction between this field and FMMQ_GC[ENQ_THR] and FMMQ_GC[DEQ_THR].</p> <p><b>Note:</b> See <a href="#">Section 5.5.6.18.3, “Internal FIFO for O/H Ports”</a> for a description on the connection between DPDE and the FIFO size.</p>
20-31	—	Reserved

#### 5.5.4.7.14 O/H Frame End Data Register (FMBM\_OFED)

The FMBM\_OFED register, shown in Figure 5-139, is used to define Offline port action on frame's last bytes.

Offset<sup>1</sup> 0x0034 (FMBM\_OFED\_<n>)  
offset <0x1000\*n>  
range n =0x1..0x7;  
Access: Read/Write



**Figure 5-139. O/H Frame End Data Register (FMBM\_OFED)**

- <sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is  $0x1000*n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-159 describes the FMBM OFED fields.

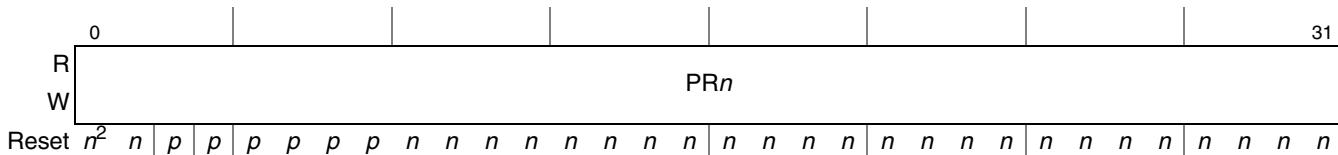
**Table 5-159. FMBM\_OFED Field Descriptions**

Bits	Name	Description
0-2	—	Reserved
3-7	CSI	<p>Checksum Ignore.  Set the number of bytes from end of frame ignored in frame one's complement running sum calculation  (for TCP/UDP checksum).</p> <p>Used to eliminate data located in end of frame from sum calculation.</p> <p>00000 - Calculate running sum for the whole frame.  00001 - Eliminate last 1bytes from frame running sum.  00010 - Eliminate last 2bytes from frame running sum.  ...  10000 - Eliminate last 16 bytes from frame running sum.  10001 - 11111 - Reserved.</p>
8-31	—	Reserved

#### 5.5.4.7.15 O/H Parse Result Initialization Register (FMBM\_OPRI)

The FMBM\_OPRI register is used to set initial value of the parse result transferred to Hardware Parser when parse request is issued. There are eight registers per O/H port.

Offset<sup>1</sup> 0x0040 (FMBM\_OPRI\_<n>\_<y>) Access: Read/Write  
offset <4096\*n+4\*(y-1)>  
range n =0x1..0x7; y=1..8



**Figure 5-140. O/H Parse Result Initialization Register (FMBM\_OPRI)**

- <sup>1</sup> There are eight registers for each O/H port. <n> is the O/H PortID. See Section 5.4.1, “FMan Hardware Ports,” for further information. The actual address of the register is  $4*(y-1) + 0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

- <sup>2</sup> FMBM\_OPRI\_x\_1[2:7] reset value is set to hard wired port ID. Note the reset values in [Table 5-160](#), “FMBM\_OPRI Reset Values.”

This table lists the FMBM OPRI reset values.

**Table 5-160. FMBM\_OPRI Reset Values**

Reg Name	Reset Value
FMBM_OPRI_1_1	0000_0001_0000_0000_0000_0000_0000_0000
FMBM_OPRI_2_1	0000_0010_0000_0000_0000_0000_0000_0000
FMBM_OPRI_3_1	0000_0011_0000_0000_0000_0000_0000_0000
FMBM_OPRI_4_1	0000_0100_0000_0000_0000_0000_0000_0000
FMBM_OPRI_5_1	0000_0101_0000_0000_0000_0000_0000_0000
FMBM_OPRI_6_1	0000_0110_0000_0000_0000_0000_0000_0000
FMBM_OPRI_7_1	0000_0111_0000_0000_0000_0000_0000_0000
FMBM_OPRI_*_2	0000_0000_0000_0000_0000_0000_0000_0000
FMBM_OPRI_*_3	0000_0000_0000_0000_0000_0000_0000_0000
FMBM_OPRI_*_4	0000_0000_0000_0000_0000_0000_0000_0000
FMBM_OPRI_*_5	1111_1111_1111_1111_1111_1111_1111_1111
FMBM_OPRI_*_6	1111_1111_1111_1111_1111_1111_1111_1111
FMBM_OPRI_*_7	1111_1111_1111_1111_1111_1111_1111_1111
FMBM_OPRI_*_8	1111_1111_1111_1111_1111_1111_1111_1111

Table 5-161 describes the FMBM\_OPRI fields.

**Table 5-161. FMBM\_OPRI Field Descriptions**

Bits	Name	Description
0-31	PRn	Parse Result. Set the parse result initiated by the BMI when hardware parser services are requested. For details refer to <a href="#">Section 5.9, “Frame Manager—Parser.”</a>

#### 5.5.4.7.16 O/H Frame Queue ID Register (FMBM\_OFQID)

The FMBM\_OFQID register, shown in Figure 5-141, configures the default frame queue ID to enqueue O/H frame. The ASPID field is used to fetch the storage profile for the IC copy parameters (if enabled).

Offset<sup>1</sup> 0x0060 (FMBM\_OFQID\_<n>) Access: Read/Write  
offset <0x1000\*n>  
range n =0x1..0x7;



**Figure 5-141. O/H Frame Queue ID Register (FMBM\_OFQID)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H Poid. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

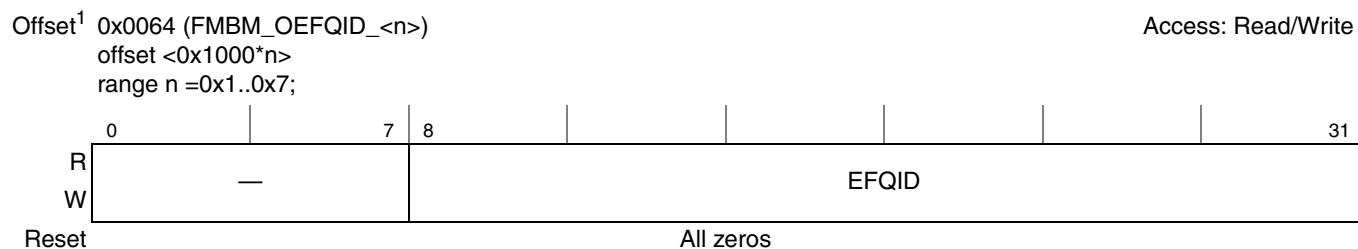
Table 5-162 describes the FMBM\_OFQID fields.

**Table 5-162. FMBM\_OFQID Field Descriptions**

Bits	Name	Description
0-7	ASPID	Absolute Storage Profile ID The BMI initializes the default storage profile ID to the value of ASPID. This is the absolute value of the virtual storage profile offset. This value is used for all frames, unless overwritten by the value programmed in Context B. This value is used for evaluating SPID for IC copy parameters (part of the IC is copied from the frame buffer). Note that default RSPID for enqueue, is also taken from this field, and may be overwritten by the value programmed in Context B, and then it may be overwritten as programmed in the KeyGen or coarse classification.
8-31	DFQID	Default Frame Queue ID. The BMI initializes the default frame queue ID to the value set in DFQID. value is used for all frames, if the parse and classifier are not activated. Note that 0x00_0000 is not a legal value for an FQID.

#### 5.5.4.7.17 O/H Error Frame Queue ID Register (FMBM\_OEFQID)

The FMBM\_OEFQID register, shown in Figure 5-142, configures the error frame queue ID to enqueue O/H frame.



**Figure 5-142. O/H Error Frame Queue ID Register (FMBM\_OEFQID)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H Poid. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

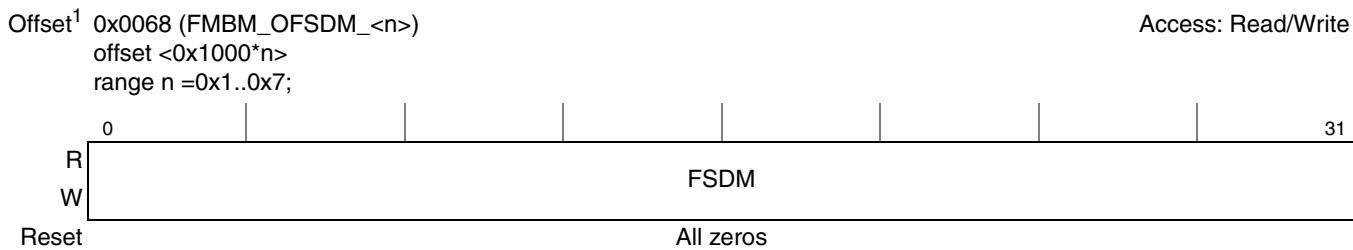
Table 5-163 describes the FMBM\_OEFQID fields.

**Table 5-163. FMBM\_OEFQID Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	EFQID	Error Frame Queue ID. EFQID is the FQID used when FMan processing concluded that the frame needs to be enqueued to error queue, or when the FMan processing concluded that the frame should be discarded but frame discard override (FMBM_OCFG[FDOVR]) bit is ‘1’. Note that 0x00_0000 is not a legal value for an FQID.

### 5.5.4.7.18 O/H Frame Status Discard Mask Register (FMBM\_OFSDM)

The FMBM\_OFSDM register, shown in [Figure 5-143](#), configures the discard mask that applies when frame is ready for enqueue operation. For any bit set to ‘1’ in the frame status word, if the corresponding bit in FMBM\_OFSDM is set, the frame is discarded. In addition, FMBM\_OFFC counter is incremented. Note that if FMBM\_OCFG[FDOVR] is set, the frame is enqueued to EFQID. If any of the events described above do not occur, the frame can be enqueued to FQID as determined from the classification process.



**Figure 5-143. O/H Frame Status Discard Mask Register (FMBM\_OFSDM)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H Poid. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-164](#) describes the FMBM\_OFSDM fields.

#### NOTE

FMBM\_OFSDM[UFD] (bit 5), should never set, because with this type of error (unknown format), the BMI cannot discard a frame and release its external buffers.

**Table 5-164. FMBM\_OFSDM Field Descriptions**

Bits	Name	Description
0-31	FSDM	Frame Status Discard Mask. Each bit of FSDM is checked against the corresponding bit in the frame status word. For any bit set to ‘1’ in the frame status word, if the corresponding bit in FSDM is set, the frame is discarded.

### 5.5.4.7.19 O/H Frame Status Error Mask Register (FMBM\_OFSEM)

The FMBM\_OFSEM register, shown in [Figure 5-144](#), configures the error mask that applies when the frame is ready for enqueue operation. For any bit set to ‘1’ in the frame status word, if the corresponding bit in FMBM\_OFSEM is set, the frame is enqueued to the error QID as defined in FMBM\_OEFQID[EFQID]. Otherwise, the frame is enqueued to FQID as determined from the classification process.

If a certain bit is set in both FMBM\_OFSDM and FMBM\_OFSEM, the latter has priority, and the frame discard is not considered.

Offset<sup>1</sup> 0x006C (FMBM\_OFSEM\_<n>)  
           offset <0x1000\*n>  
           range n =0x1..0x7;

Access: Read/Write

FSEM

Reset All zeros

**Figure 5-144. O/H Frame Status Error Mask Register (FMBM\_OFSEM)**

- <sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-165 describes the FMBM\_OFSEM fields.

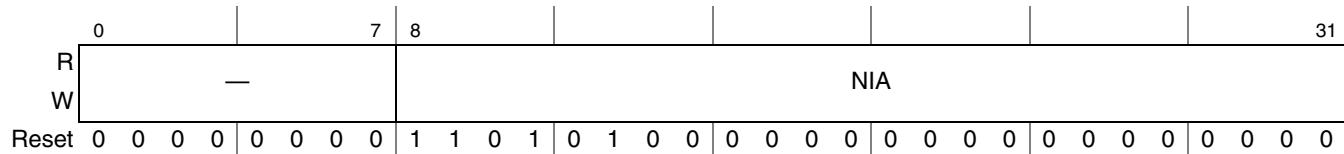
**Table 5-165. FMBM OFSEM Field Descriptions**

Bits	Name	Description
0-31	FSEM	<p>Frame Status Error Mask.</p> <p>Each bit of FSEM is checked against the corresponding bit in the frame status word. For any bit set to '1' in the frame status word, if the corresponding bit in FSEM is set, the frame queue ID is overwritten by EFQID.</p>

#### **5.5.4.7.20 O/H Frame Enqueue Next Engine Register (FMBM\_OFENE)**

The FMBM\_OFENE register, shown in Figure 5-145, configures NIA issued by the BMI, when frame is ready to be enqueued.

Offset<sup>1</sup> 0x0070 (FMBM\_OFENE\_<n>) Access: Read/Write  
offset <0x1000\*n>  
range n =0x1..0x7;



**Figure 5-145. O/H Frame Enqueue Next Engine Register (FMBM\_OFENE)**

- <sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-166 describes the FMBM\_OFENE fields.

**Table 5-166. FMBM\_OFENE Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	NIA	<p>Next Invoked Action.</p> <p>When frame is ready to be enqueued, the BMI issues the specified NIA. Default value is set to 0xD40000, which means that the next module is QMI enqueue, and ORR (order restoration required) bit is set.</p> <p>For detailed description refer to <a href="#">Section 5.3.5, “FMan User-Configurable Pipeline Architecture—Introducing the NIA.”</a></p>

#### **5.5.4.7.21 O/H Rate Limiter Scale Register (FMBM\_ORLMTS)**

The FMBM\_ORLMTS register, shown in Figure 5-146, controls O/H port rate limiter and configures its time unit scale. This register is configured in conjunction of the FMBM\_ORLMT register.

It is recommended not to modify its reset value. In some (rare) cases, this register is modified to limit the rate of operation of the offline port (packets-per-sec), and may improve overall performance.

Offset <sup>1</sup>	0x0074 (FMBM_ORLMTS_<n>)	Access: Read/Write			
	offset <0x1000*n>				
	range n =0x1..0x7;				
	0 1     10 11   15 16       29 30 31				
R	EN	—	TSBS	—	LRLS
W					
Reset	All zeros				

**Figure 5-146. O/H Rate Limiter Scale Register (FMBM\_ORLMTS)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-167 describes the FMBM\_ORLMTS fields.

**Table 5-167. FMBM\_ORLMTS Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0	EN	Enable. 0 - Rate limiter disabled. The port is allowed to operate as fast as the hardware is able and as long as data is available. 1 - Rate limiter enabled. The port frame rate is limited and cannot exceed the programmed rate.
1-10	—	Reserved.

**Table 5-167. FMBM\_ORLMTS Field Descriptions (continued)**

Bits	Name	Description
11-15	TSBS	<p>Time Stamp Bus Scale.</p> <p>TSB is the FPM central Timestamp Bus (TSB) that provides 32-bit integer count with high precision rate independent on the FMan clock frequency (see FPM chapter for more details).</p> <p>TSBS specifies the bit of TSB which is used as the reference unit count for the Tx rate limiter calculations. For example, If TSB count is incremented per 1/1024 microsecends, and TSBS is set to 21 (31–10), the Tx rate limiter time unit is counting per 1 microseconds.</p> <p>Note that TSB is numbered as TSB[0:31], such that 31 is the LSB.</p> <ul style="list-style-type: none"> <li>00000 - Rate Counter time unit equals to TSB count period / <math>2^{31}</math> (toggle period of TSB[0]).</li> <li>00001 - Rate Counter time unit equals to TSB count period / <math>2^{30}</math> (toggle period of TSB[1]).</li> <li>...</li> <li>11111 - Rate Counter time unit equals to TSB count period (toggle period of TSB[31]). (toggle period is the time elapsed between changes of state of the referenced bit).</li> </ul>
16-29	—	Reserved.
30-31	LRLS	<p>Low Rate Limit Scale.</p> <p>Lower O/H port rate limit by a selected scale.</p> <p>When the system detects high load on internal resources, The scale period selected by ORLMT[RLM] gets scaled down according to the LRLS configuration</p> <ul style="list-style-type: none"> <li>00 - ORLMT[RLM] is not scaled down.</li> <li>01 - Actual Rate Limit is set to INT{ORLMT[RLM] / 2}</li> <li>10 - Actual Rate Limit is set to INT{ORLMT[RLM] / 4}</li> <li>11 - Actual Rate Limit is set to INT{ORLMT[RLM] / 8}</li> </ul>

#### **5.5.4.7.22 O/H Rate Limiter Register (FMBM\_ORLMT)**

The FMBM\_ORLMT register, shown in Figure 5-147, configures O/H port rate limiter used to control the average rate and burst rate for O/H processing. The algorithm used to implement the rate limiter in Token Bucket. See Section 5.5.6.17, “Tx and O/H Rate Limiter,” for more information.

This register is for internal use. It is recommended not to modify its reset value. In some (rare) cases, this register is modified to balance FMan controller performance.

Offset <sup>1</sup>	0x0078 (FMBM_ORLMT_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x1..0x7;	
R	0	31
W	—	—
	MBS	RLM
Reset	All zeros	

**Figure 5-147. O/H Rate Limiter Register (FMBM\_ORLMT)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-168 describes the FMBM\_ORLMT fields.

**Table 5-168. FMBM\_ORLMT Field Descriptions**

Bits	Name	Description
0	BSG	Burst Size Granularity 0 - MBS low granularity. Each MBS increment step adds $10^3$ frame to the max burst size. 1 - MBS high granularity. Each MBS increment step adds a single frame to the max burst size.
1-5	—	Reserved.
6-15	MBS	Maximum Burst Size. Specify the Maximum burst size supported on O/H port. MBS reflects the maximum dequeue burst size when starting to dequeue frames after a period of time in which the frame queues assigned to the port are empty. MBS is selected by BSG. If BSG is cleared MBS is represented with low granularity of $10^3$ frames, otherwise it uses high granularity of single frames. If BSG=0: Units in $10^3$ frames granularity. 0x000 - Limit the burst rate to $10^3$ frames 0x001 - Limit the burst rate to $2 \cdot 10^3$ frames ... 0x3FF - Limit the burst rate to $1024 \cdot 10^3$ frames  If BSG=1 Units in $10^3$ frames granularity. 0x000 - Limit the burst rate to one frame. 0x001 - Limit the burst rate to two frames. ... 0x3FF - Limit the burst rate to 1024 frames.
16-21	—	Reserved.
22-31	RLM	Rate Limit. Specify the average frame dequeue rate on O/H port. RLM is configured to the desired rate on O/H port. Units in $(10/F) \cdot 10^3$ frames/sec granularity. F is the factor for compensation of the rate counter time unit. If TSB registers and FMBM_ORLMTS[TSBS] programming are such that the Rate Counter time unit is 1 microsecond, then F = 1. Else, F = (Rate Counter time unit) / 1 microsecond. 0x000 - Limit the frame dequeue rate to $(10/F) \cdot 10^3$ frames/sec. 0x001 - Limit the frame dequeue rate to $(20/F) \cdot 10^3$ frames/sec. ... 0x3FF - Limit the frame dequeue rate to $(10240/F) \cdot 10^3$ frames/sec.

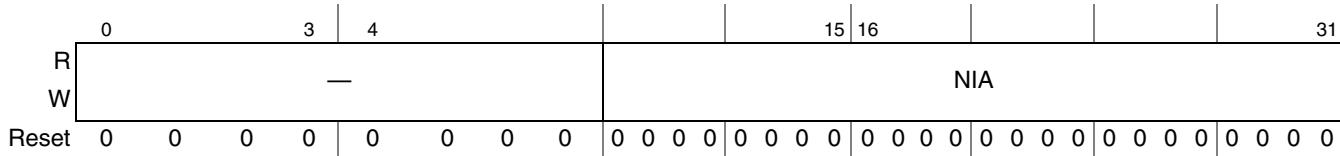
#### 5.5.4.7.23 O/H Continuous Mode Next Enqueue Register (FMBM\_OCMNE)

The FMBM\_OCMNE register, shown in Figure 5-148, configures the O/H port, for a flow in which the same frame is enqueued to multiple queues. This register is used when ICAD[NL]=1 (i.e. this is Not the

Last stage in the flow). The NIA is used by the BMI in the ‘Release Internal Resources’ stage, if ICAD[NL]=1.

Offset<sup>1</sup> 0x007C (FMBM\_OCMNE\_<n>)  
offset <0x1000\*n>  
range n =0x1..0x7;

Access: Read/Write



**Figure 5-148. O/H Continuous Mode Next Engine (FMBM\_OCMNE)**

- <sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

Table 5-169 describes the FMBM\_OCMNE fields.

**Table 5-169. FMBM\_OCMNE Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0-7	—	Reserved
8-31	NIA	If ICAD[NL]=0 the BMI end the flow at the end of this stage (no NIA is needed) If ICAD[NL]=1 this NIA is used for the next stage. The reset value is set for the FMan Controller.

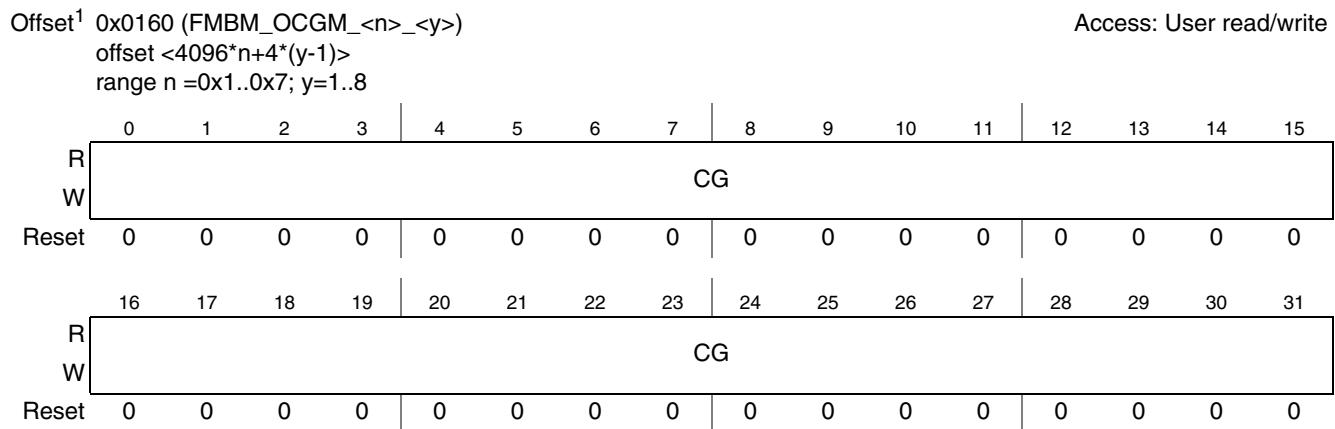
### **5.5.4.7.24 FMBM\_OCGM - Observed Congestion Group Map**

The O/H congestion group mapping is used by the BMI to observe congestion state on queue assigned to selected data consumer. The BMI gets N congestion group notifications from the QMan. Each port holds N bits to configure the congestion group that corresponds to it. BMI stops initiating tasks and effectively stops dequeue of new frames of the O/H port when congestion is detected on its associated groups. These registers are used for internal SoC flow control between the BMI O/H ports enqueued frames and their target consumer.

Hysteresis is provided by QMan upon congestion notification state transition.

## NOTE

In FMan\_v3 N=256.



**Figure 5-149. FMBM\_OCGM - Observed Congestion Group Map**

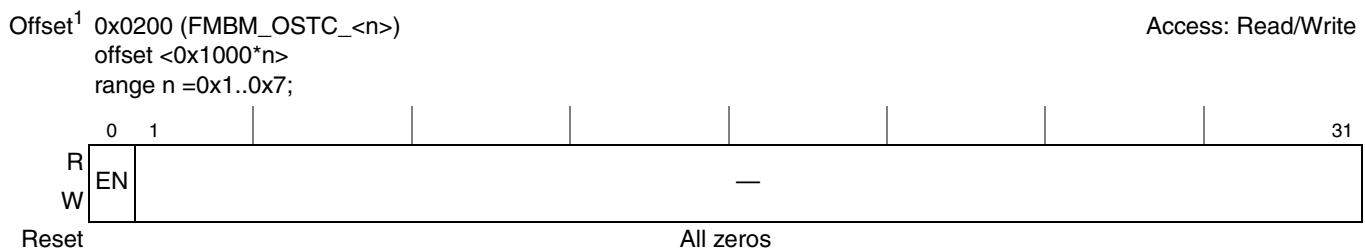
<sup>1</sup> There are up to eight registers per O/H port. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is  $4*(y-1) + 0x1000*n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

**Table 5-170. FMBM\_OCGM Field Descriptions**

Bits	Name	Description
0-31	CG	Congestion Group. Used to map 32 congestion groups from QMan. When a congestion group mapped to the O/H port is detected, the O/H port enters internal PAUSE state and stops sending dequeue requests to QMI. When the congestion condition is over the internal PAUSE state is exited and O/H traffic can resume.

#### 5.5.4.7.25 O/H Statistics Counters Register (FMBM\_OSTC)

The FMBM\_OSTC register, shown in [Figure 5-150](#), is used to control and enable O/H port statistics counters. Note that statistics counters are not affected by the assertion of soft reset, but they are affected by the assertion of hard reset.



**Figure 5-150. O/H Statistics Counters Register (FMBM\_OSTC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

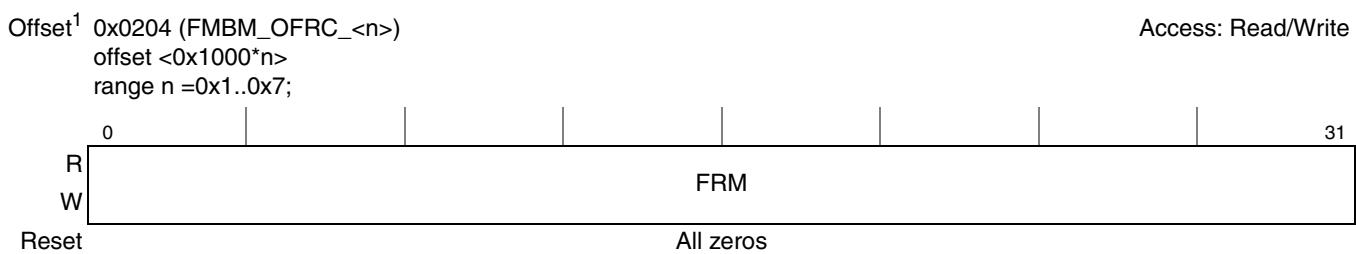
Table 5-171 describes the FMBM\_OSTC fields.

**Table 5-171. FMBM\_OSTC Field Descriptions**

Bits	Name	Description
0	EN	Enable Statistics Counters. 0 - Statistics Counters are disabled. 1 - Statistics Counters are enabled.
1-31	—	Reserved.

#### 5.5.4.7.26 O/H Frame Counter Register (FMBM\_OFRC)

The FMBM\_OFRC register, shown in Figure 5-151, counts the total number of frames/commands entered and processed via the O/H port.



**Figure 5-151. O/H Frame Counter Register (FMBM\_OFRC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-172 describes the FMBM\_OFRC fields.

**Table 5-172. FMBM\_OFRC Field Descriptions**

Bits	Name	Description
0-31	FRM	Number of Frames/Commands. Counts the number of processed frames or commands. The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

#### 5.5.4.7.27 O/H Frames Discard Counter Register (FMBM\_OFDC)

The FMBM\_OFDC register, shown in Figure 5-152, counts the number of frames or commands that are discarded due to dma error indication that was sensed during the process of loading frame header or command header or frame context (or any part of the frame if the BMI was instructed to bring the entire frame).

Those frames are discarded and not sent to the processing module. The frame shows on error queue and the DME bit is set in the FD status word.

**Figure 5-152. O/H Frames Discard Counter Register (FMBM\_OFDC)**

<sup>1</sup> There is one register for each O/H port.  $< n >$  is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

[Table 5-173](#) describes the FMBM\_OFDC fields.

**Table 5-173. FMBM OFDC Field Descriptions**

Bits	Name	Description
0-31	DTFRM	<p>Discarded Offline Port Frames or Host Commands.</p> <p>Counts the number of O/H frames/commands that were discarded due to dma error.</p> <p>The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### 5.5.4.7.28 O/H Frames Length Error Discard Counter Register (FMBM\_OFLEDC)

The FMBM\_OFLEDC register, shown in Figure 5-153, counts the number of frames or commands that are discarded due to frame size error. Terms that define length error are described in Section 5.5.6.20.5, “Unsupported Frames.”

Those frames are discarded and not sent to the processing module. The frame shows on error queue and the LGE bit is set in the FD status word.

Offset<sup>1</sup> 0x020C (FMBM\_OFLED\_C\_<n>)  
 offset <0x1000\*n>  
 range n =0x1..0x7;

Access: Read/Write

0 | | | | | | | | 31

R | DTFRM

W

Reset All zeros

**Figure 5-153. O/H Frames Length Error Discard Counter Register (FMBM\_OFLEDC)**

<sup>1</sup> There is one register for each O/H port.  $<n>$  is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-174 describes the FMBM\_OFLED C fields.

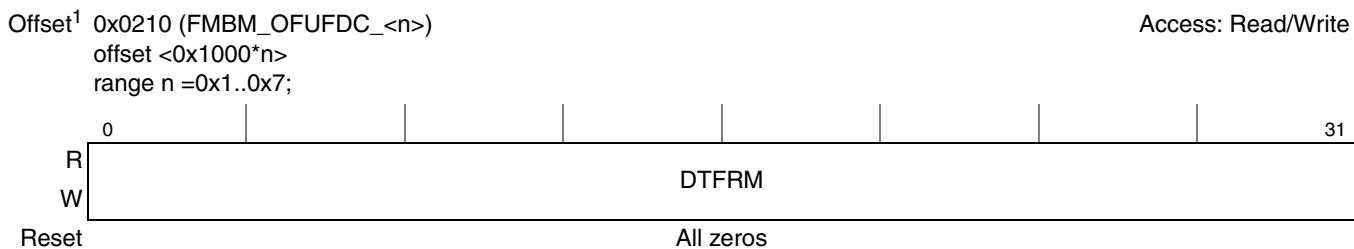
**Table 5-174. FMBM\_OFLED C Field Descriptions**

Bits	Name	Description
0-31	DTFRM	Discarded Offline Port Frames or Host Commands. Counts the number of O/H frames/commands that were discarded due to frame size error. The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

#### 5.5.4.7.29 O/H Frames Unsupported Format Discard Counter Register (FMBM\_OFUFDC)

The FMBM\_OFUFDC register, shown in Figure 5-154, counts the number of frames or commands that were discarded due to frame format error - frame descriptor contains unsupported format.

Those frames are discarded and not sent to the processing module. The frame shows on error queue and the UFD bit is set in the FD status word.



**Figure 5-154. O/H Frames Unsupported Format Discard Counter Register (FMBM\_OFUFDC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H Poid. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-175 describes the FMBM\_OFUFDC fields.

**Table 5-175. FMBM\_OFUFDC Field Descriptions**

Bits	Name	Description
0-31	DTFRM	Discarded Offline Port Frames or Host Commands. Counts the number of O/H frames/commands that were discarded due to unsupported format error. The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

#### 5.5.4.7.30 O/H Filter Frames Counter Register (FMBM\_OFFC)

The FMBM\_OFFC register, shown in Figure 5-155, counts the number of frames flowing through the O/H port that were filtered out by the parse and classify modules of the frame manager.

Those frames are discarded and not shown to return queues, unless FMBM\_OCFG[FDOVR] is set, in which case the frames are enqueued on the queue configured in FMBM\_OEFQID[EFQID].



**Figure 5-155. O/H Filter Frames Counter Register (FMBM\_OFFC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-176 describes the FMBM\_OFFC fields.

**Table 5-176. FMBM\_OFFC Field Descriptions**

Bits	Name	Description
0-31	FRFRM	Filtered Received Frames. Counts the number of filtered received frames. The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

#### 5.5.4.7.31 O/H Frames WRED Discard Counter Register (FMBM\_OFWDC)

The FMBM\_OFWDC register, shown in Figure 5-156, counts the number of frames flowing through the O/H port that were not able to enter the return queue system due to WRED algorithm. Other reasons for enqueue reject may be tail drop, out of service FQ, etc. See QMan chapter for more details.

Those frames are discarded and not shown to return queues, regardless of the state of FMBM\_OCFG[FDOVR].



**Figure 5-156. O/H Frames WRED Discard Counter Register (FMBM\_OFWDC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-177 describes the FMBM\_OFWDC fields.

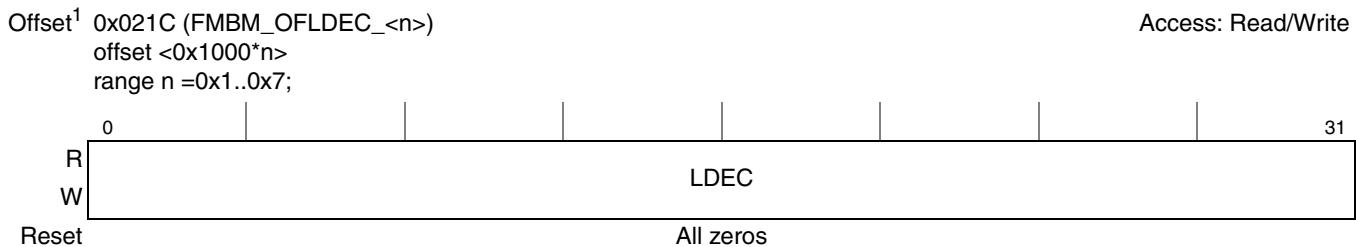
**Table 5-177. FMBM\_RFWDC Field Descriptions**

Bits	Name	Description
0-31	DRFRM	<p>Discarded Received Frames. Counts the number of received frames that were discarded due to WRED algorithm or other QMan rejection reasons. The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### 5.5.4.7.32 O/H Frames List DMA Error Counter Register (FMBM\_OFLDEC)

The FMBM\_OFLDEC register, shown in Figure 5-157, counts the number of frames flowing through the port that were not able to enter the receive queue system due to QMan reject (mostly due to WRED algorithm), and not able to release their buffers due to DMA error on the scatter/gather list read.

Those frames are discarded and not shown to receive queues, regardless of the state of FMBM\_OCFG[FDOVRR], however the buffers that are allocated to the frame are not released.



**Figure 5-157. O/H Frames List DMA Error Counter Register (FMBM\_OFLDEC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

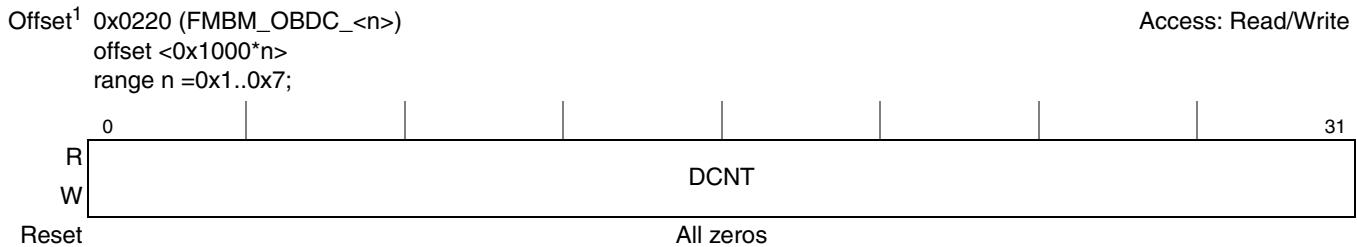
Table 5-178 describes the FMBM\_OFLDEC fields.

**Table 5-178. FMBM\_OFLDEC Field Descriptions**

Bits	Name	Description
0-31	LDEC	<p>List DMA Error Count. Counts the number of received frames that were discarded due to WRED algorithm, and not able to release their buffers due to DMA error on the scatter/gather list read The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

### 5.5.4.7.33 O/H Buffers Deallocate Counter Register (FMBM\_OBDC)

The FMBM\_OBDC register, shown in [Figure 5-158](#), counts the number of buffer deallocate operations.



**Figure 5-158. O/H Buffers Deallocate Counter Register (FMBM\_OBDC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PVID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-179](#) describes the FMBM\_OBDC fields.

**Table 5-179. FMBM\_OBDC Field Descriptions**

Bits	Name	Description
0-31	DCNT	Buffers Deallocate Counter. Counts the number of buffer deallocate operations, that the port initiated. The counter increments whenever a buffer is returned to the BMan pools, regardless of the BPID. The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

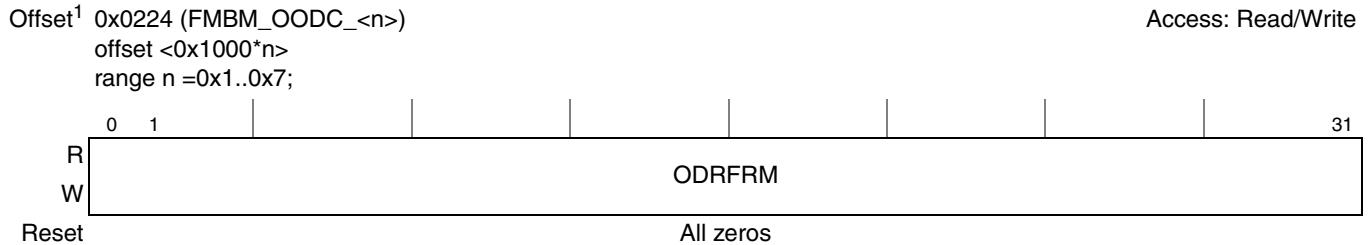
### 5.5.4.7.34 O/H Out of Buffers Discard Counter Register (FMBM\_OODC)

The FMBM\_OODC register, shown in [Figure 5-159](#), counts the number of frames that O/H port is not able to copy to a new buffer, since buffer allocation has failed. This may be caused by the following reasons:

- lack of external buffers
- lack of suitable buffers that cause the scatter/gather list to grow beyond 16 entries
- lack of suitable buffer to hold the scatter/gather list (including start margin)
- lack of suitable buffer to hold the frame header (including start margin).
- FMBM\_VEBM[SGD] bit is set and the frame does not fit in one available buffer

See [Section 5.5.6.20.3, “Buffer Depletion,”](#) for more details.

Those frames are discarded and not shown to receive queues, regardless of the state of FMBM\_OCFG[FDOVR].



**Figure 5-159. O/H Out of Buffers Discard Counter Register (FMBM\_OODC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

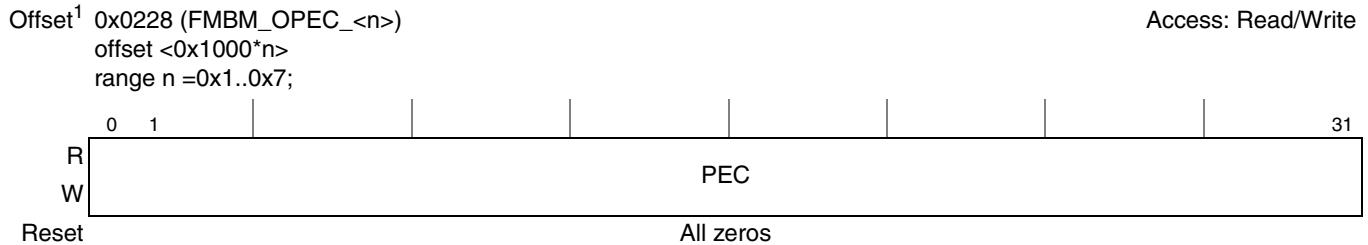
Table 5-180 describes the FMBM\_OODC fields.

**Table 5-180. FMBM\_OODC Field Descriptions**

Bits	Name	Description
0-31	ODRFRM	Discarded Frames. Counts the number of frames that were discarded due to lack of external buffers. The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value. Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.

#### 5.5.4.7.35 O/H Prepare to Enqueue Counter (FMBM\_OPEC)

The FMBM\_OPEC register, shown in Figure 5-82, counts the total number of time the BMI is invoked with ‘prepare to enqueue’ NIA (per port).



**Figure 5-160. O/H prepare to enqueue Counter Register (FMBM\_OPEC)**

<sup>1</sup> There is one register for each Rx port. n is the Rx PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

**Table 5-181. FMBM\_OPEC Field Descriptions**

Bits	Name	Description
0-31	PEC	<p>Prepare to enqueue counter.</p> <p>The FMBM_OPEC register, shown in <a href="#">Figure 5-82</a>, counts the total number of time the BMI is invoked with ‘prepare to enqueue’ or ‘discard frame’ NIA (per port) or with Similar explanation as in <a href="#">Table 5-101</a>. The counter is enabled to count if FMBM_OSTC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value. Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### **5.5.4.7.36 O/H Performance Counters Register (FMBM OPC)**

The FMBM\_OPC register, shown in Figure 5-161, is used to control and enable O/H port performance counters.

Offset<sup>1</sup> 0x0280 (FMBM\_OPC\_<n>)  
offset <0x1000\*n>  
range n =0x1..0x7;

Access: Read/Write

0	1										31
R	EN	All zeros									
W											

Reset

**Figure 5-161. O/H Performance Counters Register (FMBM\_OPC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000 * n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-182 describes the FMBM OPC fields.

**Table 5-182. FMBM OPC Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0	EN	Enable Performance Counters. 0 - Performance Counters are disabled. 1 - Performance Counters are enabled.
1-31	—	Reserved.

#### 5.5.4.7.37 O/H Performance Count Parameters Register (FMBM\_OPCP)

This register is for internal use for debug.

The FMBM\_OPCP register, shown in [Figure 5-162](#), is used to define the parameters of the O/H port performance counters.

Offset <sup>1</sup>	0x0284 (FMBM_OPCP_<n>)	Access: Read/Write
	offset <0x1000*n>	
	range n =0x1..0x7;	
	0 1 2   7 8   15 16 19   20 21 22   31	
R	—   TCV   —   DCV   —   FUCV	
W		
Reset	All zeros	

**Figure 5-162. O/H Performance Count Parameters Register (FMBM\_OPCP)**

- <sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for more details. The actual address of the register is 0x1000\*n + Offset. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

[Table 5-183](#) describes the FMBM\_OPCP fields.

**Table 5-183. FMBM\_OPCP Field Descriptions**

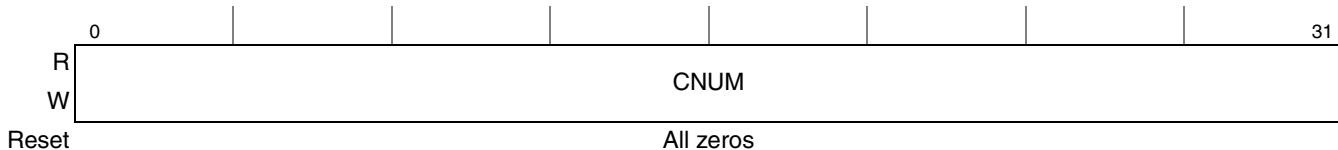
Bits	Name	Description
0-1	—	Reserved.
2-7	TCV	Tasks compare value. Used to set the criteria for concurrent tasks pipeline counter. The counter counts the number of cycles where the criteria is met. 0x00 - Number of tasks allocated for the port is equal or greater than 1. 0x01 - Number of tasks allocated for the port is equal or greater than 2. ... 0x3F- Number of tasks allocated for the port is equal or greater than 64.
8-15	—	Reserved.
16-19	DCV	DMA Compare Value. Used to set the criteria for open DMA counter. The counter counts the number of cycles where the criteria is met. 0000 - Number of open DMA is equal or greater than 1. 0001 - Number of open DMA is equal or greater than 2. ... 1111- Number of open DMA is 16.
20-21	—	Reserved.
22-31	FUCV	FIFO Utilization Compare Value. Used to set the criteria for FIFO size counter. The counter counts the number of cycles where the FIFO size is equal or greater than FUCV. The FIFO size compare value is in 256 bytes granularity (Internal buffer size). 0x000 - Number of utilized buffers in port's FIFO is equal or greater than 1 buffer (256 bytes). 0x001 - Number of utilized buffers in port's FIFO is equal or greater than 2 buffers (512 bytes). ... 0x0FF - Number of utilized buffers in port's FIFO is equal or greater than 256buffers (64k bytes). ... 0x3FF - Number of utilized buffers in port's FIFO is equal or greater than 1024 buffers (256k bytes).

#### **5.5.4.7.38 O/H Cycle Counter Register (FMBM\_OCCN)**

The FMBM\_OCCN register, shown in Figure 5-163, is used to count clock cycles.

Offset<sup>1</sup> 0x0288 (FMBM\_OCCN\_<n>)  
offset <0x1000\*n>  
range n =0x1..0x7;

Access: Read/Write



**Figure 5-163. O/H Cycle Counter Register (FMBM OCCN)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-184 describes the FMBM OCCN fields.

**Table 5-184. FMBM OCCN Field Descriptions**

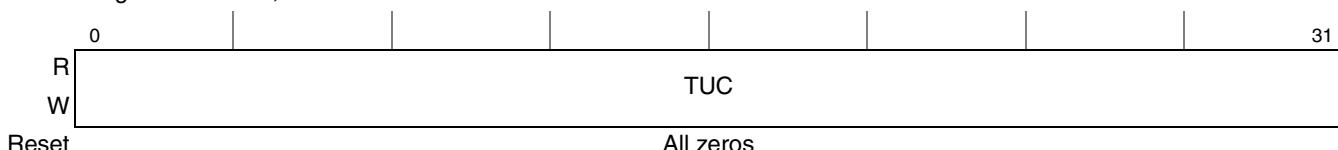
Bits	Name	Description
0-31	CNUM	<p>Cycle Number.</p> <p>Counts the number of BMI clock cycles.</p> <p>The counter is enabled to count if FMBM_OPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### **5.5.4.7.39 O/H Tasks Utilization Counter Register (FMBM\_OTUC)**

The FMBM\_OTUC register, shown in Figure 5-164, is used to monitor tasks utilization.

Offset<sup>1</sup> 0x028C (FMBM\_OTUC\_<n>)  
offset <0x1000\*n>  
range n =0x1..0x7;

### Access: Read/Write



**Figure 5-164. O/H Tasks Utilization Counter Register (FMBM OTUC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

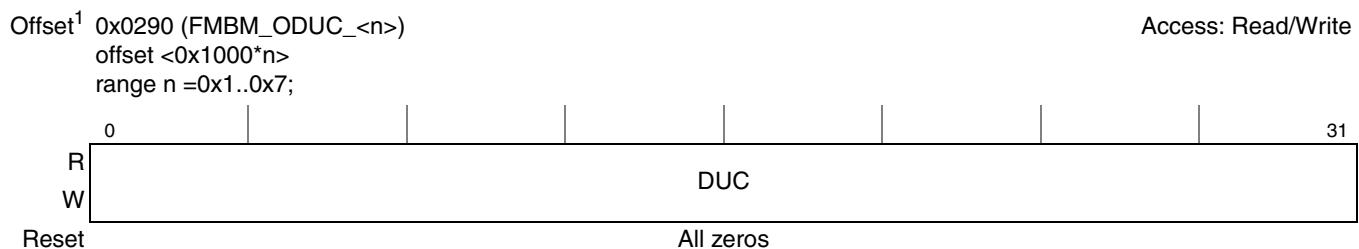
Table 5-185 describes the FMBM\_OTUC fields.

**Table 5-185. FMBM\_OTUC Field Descriptions**

Bits	Name	Description
0-31	TUC	<p>Tasks Utilization Counter.</p> <p>Counts the number of cycles in which the number of tasks allocated for the port is equal or greater than the number specified in FMBM_OPCP[TCV].</p> <p>The counter is enabled to count if FMBM_OPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### 5.5.4.7.40 O/H DMA Utilization Counter Register (FMBM\_ODUC)

The FMBM\_ODUC register, shown in Figure 5-165, is used to monitor the DMA utilization.



**Figure 5-165. O/H DMA Utilization Counter Register (FMBM\_ODUC)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-186 describes the FMBM\_ODUC fields.

**Table 5-186. FMBM\_ODUC Field Descriptions**

Bits	Name	Description
0-31	DUC	<p>DMA Utilization Counter.</p> <p>Counts the number of cycles in which the number of open DMA is equal or greater than the number specified in FMBM_OPCP[DCV].</p> <p>The counter is enabled to count if FMBM_OPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### **5.5.4.7.41 O/H FIFO Utilization Counter Register (FMBM\_OFUC)**

The FMBM\_OFUC register, shown in Figure 5-166, is used to monitor FIFO utilization.

Offset<sup>1</sup> 0x0294 (FMBM\_OFUC\_<n>)  
offset <0x1000\*n>  
range n =0x1..0x7;

## Access: Read/Write



**Figure 5-166. O/H FIFO Utilization Counter Register (FMBM\_OFUC)**

- <sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for more details. The actual address of the register is  $0x1000*n + \text{Offset}$ . See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”

Table 5-187 describes the FMBM OFUC fields.

**Table 5-187. FMBM\_OFUC Field Descriptions**

Bits	Name	Description
0-31	FUC	<p>FIFO Utilization Counter.</p> <p>Counts the number of cycles in which actual used FIFO size is equal or greater than the size specified in FMBM_OPCP[FUCV].</p> <p>The counter is enabled to count if FMBM_OPC[EN] is set. Read operation returns the current count value.</p> <p>Write operation sets the counter value and the count increments from the written value.</p> <p>Note that the counter wraps around when the count value reaches 0xFFFF_FFFF and is incremented.</p>

#### **5.5.4.7.42 O/H Debug Configuration Register (FMBM\_ODCFG)**

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires debug.

The FMBM\_ODCFG register, shown in [Figure 5-167](#), sets the condition in which BMI trap is considered a match, the level of trace that the BMI should add in debug trace portion of the IC, and the action to be taken if a debug mark is set when a frame is ready to be enqueued. This register is replicated three times, one per debug flow. See [Section 5.5.6.22, “Debug Capabilities,”](#) for more details about debug capabilities. Note that debug trap compare value and mask registers are common to all ports and they are defined in the common registers group.

## **NOTE**

DTO field is not relevant and not used in flows B and C.

Offset 0x0300 (FMBM\_ODCFG\_<n>\_<y>)  
<sup>1</sup> offset <4096\*n+4\*(y-1)>  
range n =0x1..0x7; y=1..3

	0	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16		23	24	27	28	31	
R	—	CMPOP	—	TL	—	TR_DST	—	HALT	—	—	—	—	—	—	—	—	DTO	—	—	—	—	—	
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

**Figure 5-167. O/H Debug Configuration Register (FMBM\_ODCFG)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to Section 5.4.1, “FMan Hardware Ports” for further information. The actual address of the register is 4\*(y-1) + 0x1000\*n + Offset. See Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.” ‘y’=1,2,3 corresponds to Flow A, B,C respectively.

Table 5-188 describes the FMBM\_ODCFG fields.

**Table 5-188. FMBM\_ODCFG Field Descriptions**

Bits	Name	Description
0	—	Reserved
1-3	CMPOP	Compare Operator. Selects the trap condition. 000 - Trap Disabled (never match) 001 - Always match (referred as trap bypass) 010 - Match if (trap comp & trap mask) equals to the (frame FD & trap mask). 011 - 111 - reserved
4-5	—	Reserved.
6-7	TL	Trace Level. Selects the amount of trace data to be written in debug portion if a trap match was found. 00 - Trace disable 01 - Minimum trace 10 - verbose trace 11 - very verbose trace
8-9	—	Reserved.
10-11	TR_DST	Trace Destination. If debug mark remained set when the frame is ready for enqueue, the BMI signals DMA to copy the debug portion in IC. The destination of this copy depends on the programming below. Note that if system port is selected, the user must retain enough margin in the beginning of the frame’s buffer, to avoid debug data write over the frame data. 00 - Debug trace written to memory. 01 - Debug trace written to debug port (Nexus). 10 - Reserved 11 - Reserved
12-13	—	Reserved.
14-15	HALT	Halt execution. If debug mark remained set when the BMI is before enqueue NIA is about to be issued, the BMI may indicate to FPM that halt is desired, according to the programming below. 00 - No halt. normal operation continues. 01 - stop this task if debug mark remained set. 10 - stop this port if debug mark remained set. 11 - stop all ports if debug mark remained set.
16-23	—	Reserved.

**Table 5-188. FMBM\_ODCFG Field Descriptions (continued)**

Bits	Name	Description
24-27	DTO	Debug Trace Offset. Sets the offset of the debug trace from the beginning of the IC buffer. 0000 - 0111 - Reserved. 1000 - Debug trace starts at offset 0x80 from the beginning of the IC buffer. 1001 - Debug trace starts at offset 0x90 from the beginning of the IC buffer. 1010 - Debug trace starts at offset 0xa0 from the beginning of the IC buffer. 1011 - Debug trace starts at offset 0xb0 from the beginning of the IC buffer. 1100 - Debug trace starts at offset 0xc0 from the beginning of the IC buffer. 1101 - Debug trace starts at offset 0xd0 from the beginning of the IC buffer. 1110 - Debug trace starts at offset 0xe0 from the beginning of the IC buffer. 1111 - Debug trace starts at offset 0xf0 from the beginning of the IC buffer.
28-31	—	Reserved.

#### **5.5.4.7.43 O/H General Purpose Register (FMBM\_OGPR)**

The FMBM\_OGPR register, shown in Figure 5-90, is used as a general purpose register. This register does not serve a specific purpose. It may be used for application specific purposes.

**Figure 5-168. O/H General Purpose Register (FMBM OGPR)**

<sup>1</sup> There is one register for each O/H port. <n> is the O/H PoID. Refer to [Section 5.4.1, “FMan Hardware Ports”](#) for further information. The actual address of the register is  $0x1000 \cdot n + \text{Offset}$ . See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map.”](#)

**Table 5-189. FMBM OGPR Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0-31	GPR	General Purpose Register value.

### **5.5.5 Frame Manager BMI Functional Description**

### **5.5.5.1 Introduction to BMI Functional Description**

The FMan BMI module manages four main types of flows concurrently: Ethernet receive flow (Rx), Ethernet transmit flow (Tx), Offline Port flow, Host Commands flow.

- Ethernet (Rx and Tx) is supported for 10/100/1000/2500 and 10G rates. In the Rx direction, the BMI is responsible for allocating buffers from the BMan, and place the frame in external memory. If configured to do so, the BMI may invoke other modules (e.g. Parser, Classifier, Policer) to

process the frame. In the Tx direction, the BMI is responsible for transmitting the frame, and (optionally) releasing its buffers to the BMan.

- The Offline Port is used to provide acceleration for various functions (as parsing). Frames are enqueued in dedicated Offline Port queues in the QMan. Normally the frames are enqueued by the CPU for acceleration, although it is possible to configure the system to have other engines in the device to enqueue frames to the Offline Port (such as the Security engine). Host Commands and Offline Port flows use a very similar mechanism therefore (where appropriate), they are referred in this specification as ‘O/H’.

The supported rate on the offline port is up to 1.5Mpps dequeue and enqueue from/to a QMan queue on all devices except for FMan\_v3 where the supported rate is up to 3.75Mpps.

- The aggregated bandwidth of all the ports which are enabled cannot surpass the aggregated bandwidth supported by the FMan. Therefore in some devices it is not possible to enable all the ports at their maximum line rates at once.
- Host Commands are issued by the CPU for a number of purposes: Configuration, statistics gathering or debug. The CPU enqueues the host commands in dedicated queues in the QMan.

Each port in the BMI is configured (in FMBI\_RCFG and FMBM\_TCFG registers) to operate in ‘normal mode’ or in ‘independent mode’. In normal mode, the BMI works in conjunction with the QMan and BMan to enqueue, dequeue frames and to allocate, deallocate buffers for Ethernet Rx and Tx. In independent mode, the FMan does not use any hardware resources external to FMan to enqueue, dequeue frames and to allocate/deallocate buffers for Ethernet Rx and Tx. Independent mode, may be used in the initial stages of system bring-up, before all the SoC resources are initialized.

Prior to enabling the BMI operation, the software is responsible to program the BMI’s registers and allocate some space in the FMan internal memory for the BMI internal buffers. This is done by initializing the FMBM\_CFG1 register.

The BMI internal buffers consist of blocks of 256 bytes in the FMan internal memory which are organized as a linked list. This linked list of buffers is handled by the BMI hardware. The user allocates the memory space for these buffers. Initially all the buffers are ‘free’ therefore, some times, this memory space is called the ‘BMI free buffer pool’. For initialization steps, see [Section 5.5.7, “Frame Manager BMI Initialization/Application Information.”](#)

The BMI also allocates and deallocates buffers in external memory. External memory, is memory external to the FMan. Buffers in external memory are referred to as ‘external buffers’. The exact location of the external memory is user configurable at the System on a Chip (SoC) level. Normally, external memory refers to external DDR, but in some cases the user may configure some other SoC level internal memory to function as an SRAM (for e.g. part of the L3 cache). The BMI uses a hardware portal to the BMan to request new external buffers or to release already used external buffers.

### 5.5.5.2 Introduction to BMI Data flows

The BMI supports the following data flows:

- Rx flow, normal mode. The physical MAC is involved in this flow. Frames are received from the network, saved in FMan memory, processed by other modules within the FMan and moved to external memory using the DMA. Finally, the Frame Descriptor (FD) is added to the tail of the

relevant queue in the QMan (enqueue operation). See [Section 5.5.5.3, “BMI Rx Flow—Normal Mode,”](#) for details.

- Tx flow, normal mode. The physical MAC is involved in this flow. The Frame Descriptor (FD) is moved from the head of relevant queue in QMan to FMan memory (dequeue operation), frame is copied from external buffers to FMan memory’s buffers using DMA module and then data is transferred to MACs for transmission. Optionally, the Internal Context (IC) containing actual transmit timestamp may be copied back to external memory and then the frame is enqueued to Tx confirmation queue. See [Section 5.5.5.5, “BMI Tx Flow—Normal Mode,”](#) for details.
- Rx flow, independent mode. The flow is similar to the flow in normal mode, except that the FMan performs the Ethernet Rx operations without the ‘help’ of the BMan and QMan (therefore it is called ‘independent mode’). In independent mode, the next module initialized in FMBM\_RFNE[NIA] should be configured as FMan controller. See [Section 5.5.5.4, “BMI Rx Flow—Independent Mode,”](#) for details. In this mode, the buffers are pre-allocated by the SW, and the frames are placed in ring-type structure.
- Tx flow, independent mode. The flow is similar to flow in normal mode except that the FMan performs the Ethernet Tx operations without the ‘help’ of the BMan and QMan (therefore it is called ‘independent mode’). In independent mode, the next module initialized in FMBM\_TFDNE[NIA] should be configured as FMan controller. See [Section 5.5.5.6, “BMI Tx Flow—Independent Mode,”](#) for details.
- Offline Port flow. No physical MAC is involved in this flow. A FD is dequeued from relevant queue within QMan, header of frame is moved from external buffer to FMan memory’s buffer and then the BMI issues parsing request to FPM. Once processing is completed, the header (or the entire frame, in some versions of the FMan) is moved back from FMan memory to external memory using DMA module. The user may configure the FMan to write part of the Internal Context (IC) in the margin of the first buffer of the frame. Then FD is enqueued again and FD is added to relevant frame queue. See [Section 5.5.5.7, “BMI Offline Port Flow,”](#) for details.
- Host commands flow. No physical MAC is involved in this flow. Frame queues that containing commands for FMan controller are dequeued, moved from external buffers to FMan memory’s buffers using DMA module and then dispatched to FMan controller for further processing. See [Section 5.5.5.8, “BMI Host Command Flow,”](#) for details.

### 5.5.5.3 BMI Rx Flow—Normal Mode

In normal mode, the FMan operates in conjunction with the QMan and BMan, to allocate buffers and enqueue frames. This mode is enabled by initializing FMBM\_RCFG[IM] = 0.

#### 5.5.5.3.1 Initialization

Prior to enabling the port (FMBM\_RCFG[EN] = 1), the BMI has to pass through initialization sequence as described in [Section 5.5.7.2, “Rx Port Initialization Steps in Normal Mode.”](#) It is recommended to enable BMI Rx port before enabling the same port in the Ethernet MAC in order to match the values of statistic counters. The BMI ignores the frames received before it is enabled.

### 5.5.5.3.2 BMI ‘Rx Frame’

This section describes the operation of the BMI Rx part of the Rx flow. See [Figure 5-5](#).

Each incoming frame has an Internal Context (IC) data structure associated with it. For a description of the IC refer to [Section 5.4.3, “FMan Frame Internal Context \(IC\)”](#). Before the reception of each frame, the FMan BMI initializes the Internal Context (IC) with default values which are programmed by the user at initialization in the FMBM\_RPP, FMBM\_RCCB, FMBM\_RFQID, FMBM\_RFPNE, and FMBM\_RPRI registers. Each Rx port has its own set of initial values. In FMan\_v3 FMBM\_RPP register initializes the operational mode bits in the ICAD A2 field. See [Section 5.5.6.1, “Operational Mode bits”](#) for more details. In addition, FMBM\_RFNE[FDCS] field initializes IC FD Status[0:7]. Bits which are set by this field are not reset by the FMan hardware, and are reflected in the FD when the frame goes to processing and also when the frame goes to QMan for enqueue. See “[Rx Frame Descriptor Status Word](#)” for details on the update of FD Status word. In addition, Key Size (KS) and reserved bytes in the first 64 bytes of the IC are initialized to zero. The IEEE1588 TimeStamp (8 bytes) is written with the time stamp entry in the IC of the frame (if this feature is disabled in the MAC, the BMI writes a zero in this field). The BMI stores parts of the frame in internal memory (in some devices, the whole frame is stored in internal memory). The user needs to allocate enough FIFO space for this purpose. See [Section 5.5.6.18.1, “Internal FIFO for Rx Ports”](#) for more details. The user may configure the BMI to leave an ‘internal margin’ at the beginning of the first internal buffer (as configured in FMBM\_RIM). This margin is needed if subsequent functions (FMan Controller) increase the size of the frame by doing some header manipulation. See [Section 5.5.6.4, “Internal and External Margins”](#) for more details.

In FMan\_v3, the BMI allocates the buffers after classification. This is done in the BMI ‘Prepare to Enqueue Frame’ stage. See [Section 5.5.3.3, “Rx BMI ‘Discard Frame’ or ‘Prepare to Enqueue Frame’”](#). The storage profile selection is done after classification. In this case, the whole frame is stored in internal memory, before it is written to external memory. It is possible to disable the allocation of buffers by setting EBAD operational mode bit during classification.

At the end of reception, the BMI updates the ADDR field in the Frame Descriptor (FD) section of the Internal Context (IC) (see [Table 5-19](#)). For a single buffer frame the FD[ADDR] field contains the pointer to the external buffer where the frame is stored. For a multi buffer frame the FD[ADDR] field points to the S/G list for the frame. The IC[FD] is copied into the frame QMan FD, when the QMI enqueues the frame to the QMan.

This completes the operation of the BMI Rx part of the Rx flow. The next processing module (normally the ‘Parser’) is user programmable during initialization in FMBM\_RFNE[NIA]. See [Figure 5-5](#). The NIA used after the parser is programmed in [Section 5.5.4.5.11, “Rx Frame Parser Next Engine Register \(FMBM\\_RFPNE\)”](#). Normally this NIA invokes the ‘KeyGen’ processing module. Note that as a result of the classification stages, some fields of the Internal Context (IC) are updated with per-frame-results of the classification. See [Section 5.4.3, “FMan Frame Internal Context \(IC\)”](#) for details of fields being updated.

### 5.5.5.3.3 Rx BMI ‘Discard Frame’ or ‘Prepare to Enqueue Frame’

This section describes the operation of the ‘BMI Prepare to Enqueue’ or ‘Discard Frame’ of the Rx flow. See [Figure 5-5](#).

The default NIA after classification is programmed in the Keygen Module. After parsing and classification are completed, the BMI receives an Next Invoked Action (NIA) with an action code. The NIA is

programmed in the KeyGen Scheme or in the Custom Classifier Action Descriptor (this depends on the configuration of the Rx flow, see [Section 5.3.19.1.1, “Configurable Receive \(Rx\) Flows”](#) for a high level description). The action code can be either ‘Discard Frame’ or ‘Prepare to Enqueue Frame’ (see [Section 5.5.3, “BMI Input NIA”](#)). The action code is evaluated together with the FD status word, the FMBM\_RFSDM, FMBM\_RFSEM and FMBM\_RCFG[FDOVR] to determine if the frame is discarded, enqueued to the specified target queue, or enqueued to an error queue. For details on the operation of the discard logic refer to [Section 5.5.6.8, “Rx Normal mode and Offline - Enqueue or discard decision logic”](#).

In FMan\_v3, if the result is ‘prepare to enqueue’, the BMI selects a storage profile. Either a hardware port storage profile is selected (if VSPE=0), or a virtual storage profile is selected (if VSPE=1). As a result of the storage profile selection, some values initiated by port registers, are overwritten with values extracted from the virtual storage profile. See [Section 5.5.4.1, “Storage Profiles”](#) for more details. Then the BMI allocates the buffers at this stage (after classification), from buffers pools belonging to the selected storage profile. The allocation of buffers is enabled if:

- ICAD[EBAD]=0 (External Buffer Allocation is not disabled)

See [Section 5.5.6.2, “Buffer Pool Selection for Rx and O/H”](#) for explanation on buffer pool allocation.

See “[Context A—A2 Field Description for Offline Port](#)” for detailed description of the bits (the same description applies for Rx and Offline). If buffer allocation is enabled and frame write is not disabled (ICAD[FWD]=0), the whole frame is copied to the allocated buffers in external memory. In addition, if ICAD[CWD]=0, a user configurable part of the Internal Context (IC) is written to the margin in the first buffer at the beginning of the frame in external memory. ICAD[FWD,CWD] (operation mode bits) are initialized in FMBM\_RPP, and optionally updated by the classification results. At the end of this operation, the next NIA is invoked. In most applications this is the QMI which enqueues the frame in the QMan. See [Section 5.5.6.1, “Operational Mode bits”](#) for general description about these mode bits.

#### NOTE

The conditions for buffer allocation differ from Rx port and Offline port.

The user may configure the BMI to leave an ‘external margin’ in the external buffer before and after the frame. This is configured in FMBM\_REBM. This margin may be used for application specific information, which is beyond the scope of this manual. See [Section 5.5.6.4, “Internal and External Margins”](#) for details about this feature.

The QMan may reject the enqueue operation and not place in a queue; in this case a rejection condition occurs. If QMan detects congestion in the target queue, the FMan may transmit a pause frame to the other end. See [Section 5.5.6.15, “Congestion or Rejection Handling”](#) for details on this aspect.

During Rx flow, BMI may receive a frame with illegal CRC or other error indications. In that case BMI sets the FPE, FSE bit in FD status word as appropriate. For a description of these bits, see “[Rx Frame Descriptor Status Word](#).”

The FD status word and other mode bits, affect the decision logic for enqueueing or discarding the frame. Refer to [Section 5.5.6.8, “Rx Normal mode and Offline - Enqueue or discard decision logic”](#) for details.

#### 5.5.5.3.4 BMI ‘Release Internal Buffers’

This section describes the operation of the ‘BMI Release internal buffers’ of the Rx flow. See [Figure 5-5](#).

At the end of the receive flow, the BMI releases all internal resources (internal FMan memory buffers, Internal Context and TNUMs).

In FMan\_v3, the release of internal resources is conditional on ICAD[NL]=1. See “[Context A—A2 Field Description for Offline Port](#)” for detailed description.

#### **5.5.5.4 BMI Rx Flow—Independent Mode**

To work in independent mode, software initializes FMBM\_RCFG[IM] = 1. The flow is similar to the flow in normal mode, except that the BMan and QMan are not used to allocate buffers or to enqueue data. The FMan controller performs these tasks, as well as writing the frame in external memory buffers. When working in independent mode, FMBM\_RFNE[NIA] is initialized to an entry point of the FMan controller.

At the end of the flow, the BMI releases the internal resources which are not released by the FMan controller.

#### **5.5.5.5 BMI Tx Flow—Normal Mode**

To choose the normal mode, software initializes FMBM\_TCFG[IM] = 0 and FMBM\_TFDNE[NIA] to set next module as QMI dequeue.

##### **5.5.5.5.1 Tx BMI ‘Allocate internal IC’ and QMI dequeue**

This section describes the operation of the ‘BMI Allocate internal IC’ and ‘QMI dequeue’ of the Tx flow. See [Figure 5-6](#).

Once the port Tx is enabled, (FMBM\_TCFG[EN] = 1), the BMI allocates internal resources, and updates the Internal context with per-frame default values (ICAD[FQID] is initialized with value programmed in FMBM\_TCFQID). Each Tx port has a configurable pipeline depth that defines how many frames can be dequeued concurrently by the QMI (see [Section 5.5.4.6.4, “Tx FIFO Parameters Register \(FMBM\\_TFP\)”](#)). If the number of frames in process by the current port is below the pipeline depth, the BMI invokes the next module (normally the QMI to dequeue the next frame).

The QMI dequeues the frame from the QMan queue, and updates some frame Internal Context (IC) fields with information which is extracted from the Frame Queue (FQ), Frame Queue Descriptor (FQD). This information is user configurable, and is used by the FMan as instructions which are given on a per frame queue basis. The following data structures are updated (if the update is enabled by the user):

- IC[FD Command] is updated from FD[Status/Command] field. See [Section 5.4.3.2, “Frame Descriptor \(FD\),”](#) specifically “[Tx Frame Descriptor Command/Status Word](#).” When FD[ICMD] bit is set, the BMI ignores all Tx Frame Descriptor command bits, and considers them to be all zeros.
- ICAD fields are updated with FQD[Context A] and FQD[Context B] fields. See [Section 5.4.3.3, “Internal Context Action Descriptor \(ICAD\)”](#) and [Section 5.4.3.1, “Override IC from FQD or Data Buffer.”](#) This update is enabled by various bits in Context A.
- ICAD[FQID] field is updated by FQD[Context B] FQID field or with FD[CFQ] field. This update is conditional, subject to [Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision.”](#)

Any field in the IC which is not initialized by the above procedure, is unitized with the port based registers.

### 5.5.5.2 BMI ‘Transmit Frame’ or ‘Process and Transmit Frame’

This section describes the operation of the ‘BMI Transmit Frame’ of the Tx flow. See [Figure 5-6](#).

The BMI invokes the DMA to fetch the frame from external memory. See [Section 5.5.6.12, “DMA resource Load balancing”](#) for details about DMA resource load balancing. If the frame is a scatter gather (S/G) frame, the BMI automatically scans the S/G list and fetches the whole frame from external memory. External buffers are automatically deallocated if the following condition is met:

- In FMan\_v3: If ICAD[EBD] = 1 and there is no error, external buffers are deallocated after the frame fetch. ICAD[EBD] is initialized by FMBM\_TFNE register and optionally updated by the value programmed in FQD Context A (See [Section 5.4.3.1.1, “Frame Queue Descriptor \(FQD\) Context A”](#)). See [Section 5.5.6.1, “Operational Mode bits,”](#) for general description about this mode bit.

For more details on Tx confirmation and buffer deallocation see [Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision.”](#)

For restriction on S/G list see [Section 5.5.6.3, “Restrictions on a scatter/gather list.”](#)

Any part of the IC (e.g. parse results) can be overwritten with user programmable ‘IC’ which is placed in the margin before the payload in the first frame buffer. In this way, the user may add frame specific information, that is used by the FMan (e.g. for UDP/TCP checksum calculation). This update is enabled with FD[RPD] bit. The BMI uses the size and offset as configured in FMBM\_TICP register. In FMan\_v3, if VSPE=1 (virtual storage profile is enabled), the BMI fetches the size and offset from the virtual storage profile as initialized by the user in FMBM\_VICP. If FD[UDP]=1 the BMI copies the same portion of the IC back to the margin of the frame before it is enqueued (confirmation). For a detailed description of conditions necessary for checksum calculation see [Section 5.5.6.5, “Conditions that enable Tx checksum generation.”](#)

Once the entire frame is fetched in the FMan internal FIFO (See [Section 5.5.6.18, “Internal FIFO Configuration Requirements”](#) for details on FIFO configuration.) the BMI acts in the following way:

- If the action code is ‘Transmit Frame’, the BMI starts sending the frame to the Ethernet interface through the MAC.
- If the action code is ‘fetch frame then execute’, the BMI invokes the next engine with FMBM\_TFNE[NIA]. The next engine can be the FMan Controller. At the end of processing the BMI is invoked again for the actual transmission of the frame.

For more details on checksum during Tx, see [Section 5.5.6.5, “Conditions that enable Tx checksum generation.”](#)

During transmission, the BMI applies port based rate limiting policy, as configured the by the user. See [Section 5.5.6.17, “Tx and O/H Rate Limiter.”](#)

The timestamp of the frame transmission is captured by the MAC, and copied to the appropriate location in the IC. If FD Command[UPD]=1 portion of the IC (as programmed in FMBM\_TICP) is copied back to the margin at the beginning of the first frame buffer. See [Section 5.5.6.19, “Hardware Assist for IEEE 1588-Compliant Timestamping”](#) for more details.

If Tx confirmation is enabled, the BMI invokes the next engine using FMBM\_TFENE. Normally the QMI is invoked, to enqueue the frame. In FMan\_v3, ICAD[FQID Type] and ICAD[TXCT] affect the confirmation flow. See [Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision”](#) for more details. As an example, Tx confirmation is useful for time synchronization frames.

At the end of processing the BMI operates as follows:

- If ICAD[NL]=0, the BMI releases the internal buffers holding the frame and ends the Tx flow.
- If ICAD[NL]=1 the BMI does not release the internal buffers, and invokes the next processing module as programmed in FMBM\_TCMNE register. This mode is needed if further processing is necessary for the frame.

See [Section 5.5.6.1, “Operational Mode bits”](#) for general description about this mode bit.

#### NOTE

BMI discards frames with length = 0, with no confirmation and no buffer release.

### 5.5.5.6 BMI Tx Flow—Independent Mode

To choose independent mode, software initializes FMBM\_TFDNE[NIA] to invoke the FMan controller. The flow is similar to the flow in normal mode except that instead of issuing dequeue NIA request for QMI, the BMI issues NIA request for FMan controller to prepare the next frame. FMan controller is responsible to generate DMA transactions in order to copy the frame from external memory to internal. Upon DMA completion, the FMan controller dispatches a NIA request to the BMI to send the frame.

Once the frame is sent, the BMI releases all internal buffers occupied by the frame to the free buffer pool and releases the tnum to FPM.

### 5.5.5.7 BMI Offline Port Flow

#### 5.5.5.7.1 Initialization

Prior to enabling the port (FMBM\_OCFG[EN] = 1), the BMI has to pass through initialization sequence as described in [Section 5.5.7.6, “Initialization Steps for Offline Ports.”](#)

#### 5.5.5.7.2 Offline BMI ‘Allocate internal IC’ and QMI dequeue

This section describes the operation of the ‘BMI Allocate internal IC’ and ‘QMI dequeue’ of the Offline flow. See [Figure 5-7](#).

Once the port Offline port is enabled, (FMBM\_OCFG[EN] = 1), the BMI allocates internal resources, and updates the Internal context with per-frame default values (ICAD[FQID] is initialized with value programmed in FMBM\_OCFQID).

In FMan\_v3, each O/H port has a configurable pipeline depth that defines how many frames can be dequeued concurrently by the QMI (see [Section 5.5.4.7.13, “O/H FIFO Parameters Register \(FMBM\\_OFP\)”](#)).

The BMI invokes the next module (normally the QMI to dequeue the next frame).

The Offline Port has a configurable rate limiter (see [Section 5.5.4.7.22, “O/H Rate Limiter Register \(FMBM\\_ORLMT\)](#)) that defines how many frames can be processed per second. The FMan QMI dequeues frames from the QMan at the rate programmed in FMBM\_ORLMT.

The QMI dequeues the frame from the QMan queue, and updates some frame Internal Context (IC) fields with information which is extracted from the Frame Queue (FQ), Frame Queue Descriptor (FQD). This information is user configurable, and is used by the FMan as instructions which are given on a per frame queue basis. The following data structures are updated (if the update is enabled by the user):

- IC[FD Command] is updated from FD[Status/Command] field. See [5.4.3.2, “Frame Descriptor \(FD\)](#)”, specifically “[Offline Port Frame Descriptor Command/Status](#). When FQD Context A [ICMD] bit is set, the BMI ignores all Offline Frame Descriptor command bits, and considers them to be all zeros.
- ICAD fields are updated with FQD[Context A] and FQD[Context B] fields. See [Section 5.4.3.3, “Internal Context Action Descriptor \(ICAD\)](#)” and [5.4.3.1, “Override IC from FQD or Data Buffer](#). This update is enabled by various bits in Context A.
- ICAD[FQID] field is updated by FQD[Context B] FQID field or with FD[CFQ] field. This update is conditional, subject to [5.5.6.10, “Offline Port Enqueue Decision Flow](#).

The QMI invokes the next engine with FMQM\_PnDN register.

### 5.5.5.7.3 BMI Offline ‘frame fetch’

This section describes the operation of the ‘BMI Frame Fetch’ of the Offline flow. See [Section 5.3.19.3, “FMan Offline Port Flow Example](#).

The action code invoking the BMI can be ‘fetch frame header and execute’ or ‘fetch frame and execute’. The BMI acts in the following way depending on the action code:

- ‘fetch frame header and execute’: The BMI fetches data from the first buffer in external memory. In this mode, the BMI fills the first internal buffer up to 256 bytes including FMBM\_OIM[FOF] (i.e. frame header is defined as 256 - FMBM\_OIM[FOF] or frame size, whichever is smaller). External buffers are not deallocated.
- ‘fetch frame and execute’: The BMI fetches the entire frame from external memory. If the frame is a scatter gather (S/G) frame, the BMI automatically scans the S/G list and fetches the whole frame from external memory. If ICAD[EBD] = 1, external buffers are deallocated after the frame fetch. ICAD[EBD] is initialized by FMBM\_OPP register and optionally updated by the value programmed in FQD Context A (See [5.4.3.1.1, “Frame Queue Descriptor \(FQD\) Context A](#)”).

For restriction on S/G list see [5.5.6.3, “Restrictions on a scatter/gather list](#).

For checksum validation see [5.5.6.6, “Conditions that enable Offline checksum validation](#).

Fields in the IC are unitized with the port based registers (FMBM\_OPP, FMBM\_OCCB, FMBM\_OFQID, FMBM\_OFPNE and FMBM\_OPRAI). In addition, Key Size (KS) and reserved bytes in the first 64 bytes of the IC are initialized to zero.

Any part of the IC (e.g. parse results) can be overwritten with user programmable ‘IC’ which is placed in the margin before the payload in the first frame buffer. In this way it is possible to carry information between Rx port to Offline. This information (which may have been generated by the original receive port),

may be used by the Offline port (for example, this is useful for carrying the running sum from receive port while processing checksum in the current FMan’s parser. Another example is for carrying the timestamp information from the original receive port). The BMI uses the size and offset as configured in FMBM\_OICP register (or FMBM\_VICP, if VSPE=1). This function is enabled with FD[RPD] bit. If FD[UDP]=1 the BMI copies the same portion of the IC back to the margin of the frame before it is enqueued. This feature is most useful in devices which contain more than one instance of the FMan.

The user may configure the BMI to leave an ‘internal margin’ at the beginning of the first internal buffer (as configured in FMBM\_OIM[FOF]). This margin is needed if subsequent functions increase the size of the frame by doing some header manipulation. See [Section 5.5.6.4, “Internal and External Margins”](#) for more details.

Once the entire frame or the frame header is fetched in the FMan internal FIFO (See [5.5.6.18, “Internal FIFO Configuration Requirements”](#) for details on FIFO configuration.) the frame fetch operation of the BMI Offline flow is completed. The BMI then invokes the next module as programmed during initialization in FMBM\_OFNE[NIA] (see [Section 5.5.4.7.6, “O/H Frame Next Engine Register \(FMBM\\_OFNE\)”](#)).

The next processing module is normally the ‘Parser’. The NIA used after the parser is programmed in FMBM\_OFPNE. Normally this NIA invokes the ‘KeyGen’ processing module. Note that as a result of the classification stages, some fields of the Internal Context (IC) are updated with per-frame-results of the classification. See [Section 5.4.3, “FMan Frame Internal Context \(IC\)”](#) for details of fields being updated.

#### **5.5.5.7.4 BMI Offline ‘Discard Frame’ or ‘Prepare to Enqueue Frame’**

The continuation of the flow is user configurable. Normally in offline port, the FMan classifies the frame and distributes it to queues (similar to an Rx flow). The classification may result in ‘enqueue’ or ‘discard’ decision. See [Section 5.5.6.8, “Rx Normal mode and Offline - Enqueue or discard decision logic”](#) and section [Section 5.5.6.10, “Offline Port Enqueue Decision Flow”](#) for details on these aspects.

Although the flow is very similar to the Rx flow, there are some subtle differences, which are described in the sections below. The description focuses on the differences between the two flows. For completeness, the user is advised to read the corresponding section in the Rx flow.

The default NIA after classification is programmed in the Keygen Module. After parsing and classification are completed, the BMI receives an Next Invoked Action (NIA) with an action code. The NIA is programmed in the KeyGen Scheme or in the Custom Classifier Action Descriptor (this depends on the configuration of the Offline flow).

#### **Offline Enqueue operation in FMan\_v3**

If the result is ‘prepare to enqueue’ and if VSPE=1, the BMI selects a virtual storage profile. See [Section 5.5.4.1, “Storage Profiles”](#) for more details. Then the BMI allocates the buffers (after classification), from buffers pools belonging to the selected storage profile. The allocation of buffers is enabled if the following conditions are met:

- ICAD[EBAD]=0 (External Buffer Allocation is not disabled, see [“Context A—A2 Field Description for Offline Port”](#)) AND
- ICAD[VSPE]=1 (Virtual Storage Profile is Enabled)

See [Section 5.5.6.2, ‘Buffer Pool Selection for Rx and O/H’](#) for explanation on buffer pool allocation. See [Section 5.5.6.1, ‘Operational Mode bits’](#) for general description about these mode bits.

#### NOTE

The conditions for buffer allocation differ from Rx port and Offline port. Buffer allocation should not be enabled if input action code is ‘fetch frame header and execute’.

The user may configure the BMI to leave an ‘external margin’ in the external buffer before and after the frame. This is configured in the virtual storage profile register FMBM\_VEBM. This margin may be used for application specific information, which is beyond the scope of this manual. See [5.5.6.4, ‘Internal and External Margins’](#) for details about this feature. See [Section 5.5.6.4, ‘Internal and External Margins’](#).

If buffer allocation not disabled (ICAD[EBAD]=0) and frame write is not disabled (ICAD[FWD]=0), the whole frame or part of it, is copied to the allocated buffers in external memory. The BMI copies into external memory, the exact number of buffers it has fetched in ‘fetch frame header and execute’ or ‘fetch frame and execute’ stage. In addition, if ICAD[CWD]=0, a user configurable part of the Internal Context (IC) is written to the margin in the first buffer at the beginning of the frame in external memory.

ICAD[FWD,CWD] (operation mode bits) are initialized in FMBM\_OPP, and optionally updated by FQD Context A - A2 field, and/or classification results.

Then, the BMI updates the ADDR field in the Frame Descriptor (FD) section of the Internal Context (IC) (see [Section Table 5-19., ‘Internal Context \(IC\)’](#)). For a single buffer frame the FD[ADDR] field contains the pointer to the external buffer where the frame is stored. For a multi buffer frame the FD[ADDR] field points to the S/G list for the frame. The IC[FD] is copied into the frame QMan FD, when the QMI enqueues the frame to the QMan.

At the end of this operation, the next NIA is invoked. In most applications this is the QMI which enqueues the frame in the QMan. The FD status word and other mode bits, affect the decision logic for enqueueing or discarding the frame. Refer to [Section 5.5.6.8, ‘Rx Normal mode and Offline - Enqueue or discard decision logic’](#) for details. If the QMan is not able to enqueue the frame, a congestion/rejection condition occurs. See [5.5.6.15, ‘Congestion or Rejection Handling’](#) for details on this aspect.

#### 5.5.5.7.5 BMI Offline ‘Release Internal Buffers’

For description of this aspect see [Section 5.5.5.3.4, ‘BMI ‘Release Internal Buffers’’](#). Same behavior applies to Offline port, using offline port registers when appropriate.

#### 5.5.5.7.6 Illegal combinations

In FMan\_v3 there are many ways of configuring the various modes of operations. The following section specifies combinations that are legal and which ones are illegal. Illegal combinations result in undefined behavior and must not be used.

external buffers deallocate <sup>1</sup>	new buffer allocate enabled <sup>2</sup>	enqueue(1)/discard(0) <sup>3</sup>	result
0	0	0	buffer leak

0	0	1	ok
0	1	0	buffer leak
0	1	1	buffer leak
1	0	0	ok
1	0	1	data lost
1	1	0	ok
1	1	1	ok

<sup>1</sup> Final EBD

<sup>2</sup> Final VSPE=1 AND Final EBAD=0

<sup>3</sup> O/H enqueue decision. See [Section 5.5.6.10, “Offline Port Enqueue Decision Flow”](#)

### 5.5.5.8 BMI Host Command Flow

One of the O/H ports is dedicated to host commands to the FMan. On this port software configures FMBM\_OFNE[NIA] to invoke the FMan Controller as the next module. Host commands are issued by the host CPU in order to send commands to the FMan. This mechanism is used to assist with port virtualization (in Keygen and Policier), dynamic updates for custom classifier tables etc. The host CPU places in the ‘host command buffer’ pointed by the FD (normally used for the frame buffer) and enqueues an FD pointing to this buffer in a dedicated host command queue. The FMan controller is responsible for executing the associated host command. The host command response is enqueued in a QMan queue. See the host command enqueue decision flow in [Section 5.5.6.11, “Host command enqueue decision flow.”](#) See FMan Controller chapter for details on host commands.

## 5.5.6 BMI Internal Operation Details

### 5.5.6.1 Operational Mode bits

There are a number of user configurable ‘operational mode bits’ which affect the operation of the BMI. Some of these bits (e.g. NL, NENQ) affect the pipeline flow, while others affect the access to buffers (e.g. EBD, EBAD, FWD, CWD). The exact type of bits available vary from Rx, Tx and offline ports. These bits are initialized in the Internal Context Action Descriptor (ICAD) when a new frame is processed (see FMBM\_RPP, FMBM\_OPP and FMBM\_TFNE registers) and may be altered during the processing of the frame by various modules. For Rx ports, the bits may be altered by the Keygen result and/or by the custom classifier result. For Tx the bits may be altered by the values programmed in FQD Context A. For Offline ports, all three alterations are supported in order of the invocation of the module. Note that there are two modes of altering the bits: ‘Setting one’ and ‘override’. In ‘setting one’ mode, the altering module is able to set some of the bits, but does not reset values which are already set (accruing all the ‘sets’ from the previous modules). In ‘override’ mode, the altering module changes the value of the bits to zero or to one overriding any previous value. This mode is controlled by ‘OVOM’ mode bit. See [Section 5.4.3.1.1, “Frame Queue Descriptor \(FQD\) Context A”](#) for detailed description of the function of these bits.

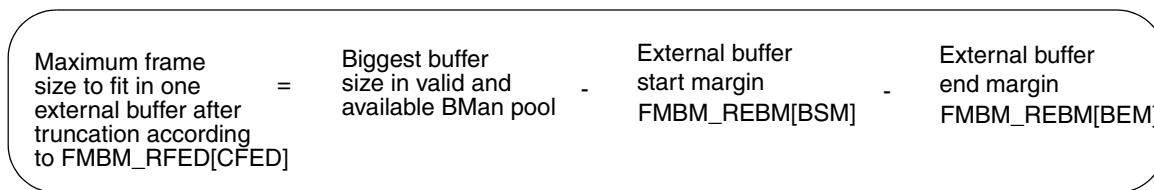
### 5.5.6.2 Buffer Pool Selection for Rx and O/H

Buffers for storage in external memory are allocated from buffer pools in the BMan. The buffer pools are organized into storage profiles. See [Section 5.5.4.1, “Storage Profiles.”](#) Each storage profile contains up to four buffer pools (for different size of buffers). The size of the buffer is calculated based on the frame size. The BMI optimizes the selection of the buffer, to minimize the memory footprint. The BMI tries to obtain the BMan pool with smallest available buffers to fit the whole frame. Then the BMI tries to allocate one external buffer within the selected pool from BMan. If requested pool is depleted, the BMI requests a buffer from the next larger size pool. If that pool is also depleted, it tries another pool with bigger buffers and so on. This process continues until either a buffer is found or no more pools with larger buffers are available.

In the latter case, and if S/G is not disabled (SGD=0), the BMI checks if there are available pools of smaller buffers, and requests a buffer in order to build an S/G list, the frame is stored in multiple buffers in a Scatter Gather (S/G) list (On Rx SGD is initialized by FMBM\_REBM; on Rx and Offline SGD is updated by the virtual profile, if enabled, The BMI updates the FD external pointer to point to the beginning of the S/G list).

The BMI discards the frame, releases all external and internal buffers, and increments the relevant discard counter if a buffer depletion condition occurs. See [Section 5.5.6.20.3, “Buffer Depletion”](#) for more details.:.

The following figure shows what is the maximum frame size that fits in one external buffer. Note that in the first and last buffer, the user may leave some space for margin. See [Section 5.5.6.4, “Internal and External Margins.”](#)



**Figure 5-169. Maximum Frame Size to Fit in One External Buffer**

### 5.5.6.2.1 Backup Pools

Some of the buffer pools can be programmed to serve as backup pools. When all enabled non-backup pools are out of buffers, the backup pools become available to the BMI for buffer selection. The BMI attempts to allocate a buffer from the backup pool with the same criteria used on the normal pools. If both type of pools are depleted, the frame is discarded. Note that if SGD bit is set (Scatter Gather is disabled), the BMI enforces this rule, therefore some enabled non-backup pools may not be used. This mechanism is useful in systems with two types of external memory. Backup pools are enabled in the FMBM\_REBMPI register for Rx or FMBM\_VEBMPI register for Rx or O/H (in virtual profile).

### 5.5.6.3 Restrictions on a scatter/gather list

The restrictions on a scatter/gather list:

1. Scatter/gather list cannot have more than 16 entries.
2. If a scatter/gather list is shorter than 16 entries, reading 256 consecutive bytes from the start of the list does not produce memory or system protection errors.

### 5.5.6.4 Internal and External Margins

Internal and external margins are defined to leave space in the data buffers of the frame for various purposes. Here are a few examples for usage of margins:

- External margin at the beginning of the buffer may be used to store a (user configurable) part of the Internal Context (IC). The IC contains the parser results, timestamp and other information. On Rx, this information may be useful to the host processor. On Tx the FMan uses some of the information for checksum generation. On O/H, the information is used for checksum validation See [Section 5.4.3, “FMan Frame Internal Context \(IC\)”](#) for more details.
- External margin at the end of the frame of the last buffer of the frame, may be used to add a trailer to the frame. This type of usage is application specific, and is not used in mainstream flows of the FMan.
- Internal margin (FOF) is a margin left in the first internal buffer holding the frame. This margin may be used for packet manipulations. Internal margin is also used to configure the part of the IC that is placed at the beginning of the first buffer of the frame.

#### 5.5.6.4.1 Internal Margins

The internal margins are listed as follows:

- FMBM\_RIM[FOF] defines the offset (from start of buffer) in the first internal buffer to which the received frame is to be written.
- FMBM\_RICP[ICIOF] defines the offset in the IC buffer from which part of IC is copied to external memory.
- FMBM\_RICP[ICSZ] defines the size of IC to be copied to external memory.
- FMBM\_RICP[ICEOF] defines the offset in the external buffer from the start of the buffer to which the IC has to be written. When there are no scatter/gather buffers, the IC is copied to the external buffer where the frame is written (see [Figure 5-171](#)). However, when a scatter/gather list is created and the external buffer is allocated for scatter/gather list storage, the IC is copied to the external buffer where the scatter/gather list is stored. In such case, the IC is written within the external start margin boundary (for details, see [Section 5.5.6.4.2, “External Margins”](#)).

#### 5.5.6.4.2 External Margins

The external margins are listed as follows:

BSM field from FMBM\_REBM or FMBM\_VEBM - if virtual storage profile is selected, defines the margin that the BMI creates in the frame’s first buffer before a frame content start. In case of a scatter/gather frame format (the frame occupies more than one buffer), the margin is also inserted in the buffer which is allocated for the scatter/gather list. Note that the scatter/gather list is always aligned to 16 bytes boundary - the BMI ignores the 4 lsbs in BSM field when building the margin in the scatter/gather buffer. On the other hand, the payload margin takes the full BSM field and may be unaligned.

- In order to avoid the IC data write over the frame area, the BSM size should be as follows:

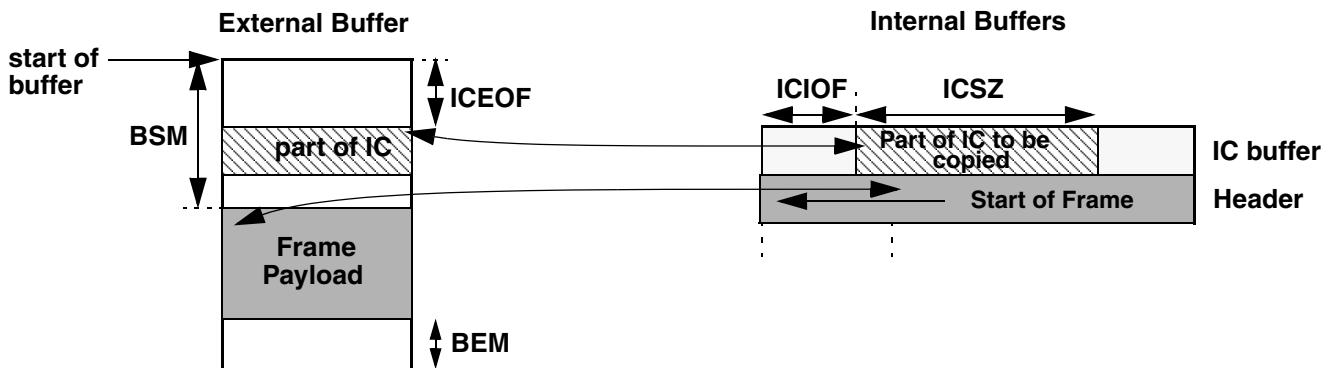
$$\text{BSM} \geq \text{ICEOF} + \text{ICSZ}$$

**Figure 5-170. External Buffer Start Margin Calculation**

- Note that if BSM is not aligned, optimization of the first write transaction of the frame's payload is possible (with some restrictions).
- FMBM\_REBM[BEM] - defines the margin from end of the frame to be added to the allocated buffer or buffers.

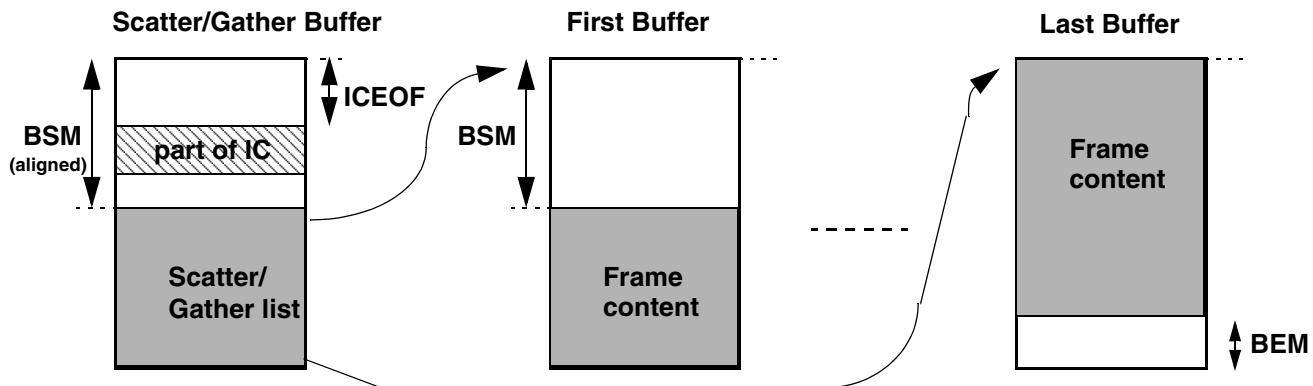
Refer to FMBM\_RDA[WOPT] for more details on write optimization ([Table 5-69](#)).

[Figure 5-171](#) summarizes the internal and external margins in case when frame fits in one external buffer.



**Figure 5-171. Internal and External Margins, Single Buffer Frame Format**

[Figure 5-172](#) summarizes the internal and external margins in case when frame splits into several buffers and is described in a scatter/gather frame format. Note that the tail margin can be spread over more than one buffer, if the last buffer that contains content cannot suffice the remainder of the frame content and the full tail margin.



**Figure 5-172. Internal and External Margins, Scatter/Gather Frame Format**

For proper BMI operation, software is responsible to configure BMan pools and their sizes reasonably and according to recommendations.

#### 5.5.6.4.3 External memory accesses optimization

In the following section FOF is used instead of FMBM\_RIM[FOF] or FMBM\_OIM[FOF].

The configuration of the internal and external margins has an effect on external memory access optimization. For clarification see the following scenarios (assuming cache line size of 64 bytes):

- If BSM is configured to 64bytes (or a multiple of 64bytes) then FOF should be ‘0’ or any multiple of cache line size (64 bytes).
- If BSM is not a multiple of 64bytes, then FOF should be configured to satisfy the equation: BSM - FOF = multiple of cache line (64 bytes). For example: FOF = 0x30 and BSM = 0x70 will allow for optimal external memory accesses.
- If BSM is not a multiple of 16 bytes, it is not possible to find a value for FOF which optimizes external memory accesses (this is due to the fact that BSM has 1 byte granularity while FOF has 16 byte granularity).

Failing to follow these suggestions results in partial cache-line-size accesses are performed by FMan when writing to external memory. In some systems this may have a significant effect on overall system performance.

#### 5.5.6.5 Conditions that enable Tx checksum generation

The FMan is able to generate a checksum on Tx, if the parse result information (e.g. offsets to L4 header) is placed at the frame start margin. See [Section 5.5.4.6.6, “Tx Internal Context Parameters Register \(FMBM\\_TICP\).”](#) In addition, reading of the IC must be enabled in the Tx FD Command field. See [Section 5.4.3.2.1, “FD Command/Status Word.”](#)

The following conditions for Tx checksum update operation:

- L4 checksum is updated when all following conditions are valid:
  - RPD (to read parse results from margin before the frame) and DTC (enable) bits in FD command are set (in FMan\_v3 this is conditioned by ICMD=0). In FMan\_v3, it is possible to calculate checksum without reading the parse results from the frame (thus removing the requirement for setting RPD). For this mode an FMan Controller routine needs to be activated. See [Section 5.12, “Frame Manager—FMan Controller”](#) for availability of this mode.
  - L4 type is UDP or TCP.
  - L3 type is IPv4 or IPv6, one IP header only. Only First IP bit of parse results is asserted. If Last IP header or GRE or MINENC are present, L4 Checksum is not calculated. (Tunneled IP).
  - If IPv4, HL is legal ( $0x5 \leq HL \leq 0xF$ ).
  - ( $L3Offset < (FD[LENGTH] - 20\text{bytes})$ ).
  - ( $L4\ checksum\ field\ offset + 1 < FD[LENGTH]$ , two cases exist:
    - If L4 protocol is TCP:  $L4\ offset + 17 < FD[length]$
    - If L4 protocol is UDP:  $L4\ offset + 7 < FD[length]$

- L4Offset < 0xF0.
- (L4Offset - L3Offset) is not an odd number.
- L3 checksum is updated when all following conditions are valid:
  - RPD and DTC bits in FD command are set (in FMan\_v3 this is conditioned by ICMD=0). In FMan\_v3, it is possible to calculate checksum without reading the parse results from the frame (thus removing the requirement for setting RPD). For this mode an FMan Controller routine needs to be activated. See [Section 5.12, “Frame Manager—FMan Controller”](#) for availability of this mode.
  - L3 type is IPv4 only, one IP header only. Only First IPv4 bit of parse results is asserted. If not, L3 checksum is not calculated.
  - (L3Offset < (FD[LENGTH] - 20bytes)).
  - L3 header must be entirely included in the first 256 bytes of the frame.

### 5.5.6.6 Conditions that enable Offline checksum validation

The action code invoking the BMI can be ‘fetch frame header and execute’ or ‘fetch frame and execute’. The BMI acts in the following way depending on the action code:

- ‘fetch frame header and execute’: The FMan validates the TCP/UDP checksum using the gross checksum placed by the user in the appropriate location in the parse results (in the margin holding the IC at the beginning of the first buffer of the frame). The user must set FD[RPD] bit to instruct the FMan to read the gross checksum.
- In FMan\_v3: ‘fetch frame and execute’: If enabled in FD[DCL4C], the FMan validates the TCP/UDP checksum without any gross checksum assistance. Note that the BMI calculates the gross checksum, thus the user does not need to place the gross checksum (running sum) at the beginning of the frame. The user may configure the BMI (FMBM\_OFED[CSI]), to ignore part of the trailer of the frame for checksum validation.

### 5.5.6.7 Transmit Confirmation Type (TXCT)

The user may set the TXCT bit in FQD Context A A0 field. This section explains the usage of this bit.

It is possible, that the frame which is dequeued by the FMan for transmission does not reside in the original buffer allocated by the CPU (and pointed by the original FD which is enqueued by the CPU), since during processing, the frame may copied to an intermediate buffer (for e.g. by the security block). TXCT bit may be set in applications where the CPU needs to ‘receive’ back the original buffer of frame it has enqueued, while FMan deallocates the intermediate buffer. The CPU must place the original FD in the first 16 bytes of the frame payload.

The sequence of events is as follows:

- FMan dequeues a frame pointed by FD2, from a queue which is pre-configured with Context A A0[TXCT]=1 and Context A A2[EBD]=1. A copy of FD1 (the original FD) resides in the first 16 bytes of the frame data.
- The FMan transmits the frame pointed by FD2, excluding the first 16 bytes (which contain FD1)

- The FMan deallocates (releases) the buffers pointed by FD2 (if EBD=1)
- If frame error (UFD or LGE or DME) is encountered, the frame is not transmitted and the error status bit is set in FD2, which is enqueued to confirmation queue.

If Tx confirmation is enabled, the FMan operates as follows:

- The FMan enqueues FD1 in the confirmation queue

For detailed flow of this functionality, see [Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision”](#) (junction A and B).

### **5.5.6.8 Rx Normal mode and Offline - Enqueue or discard decision logic**

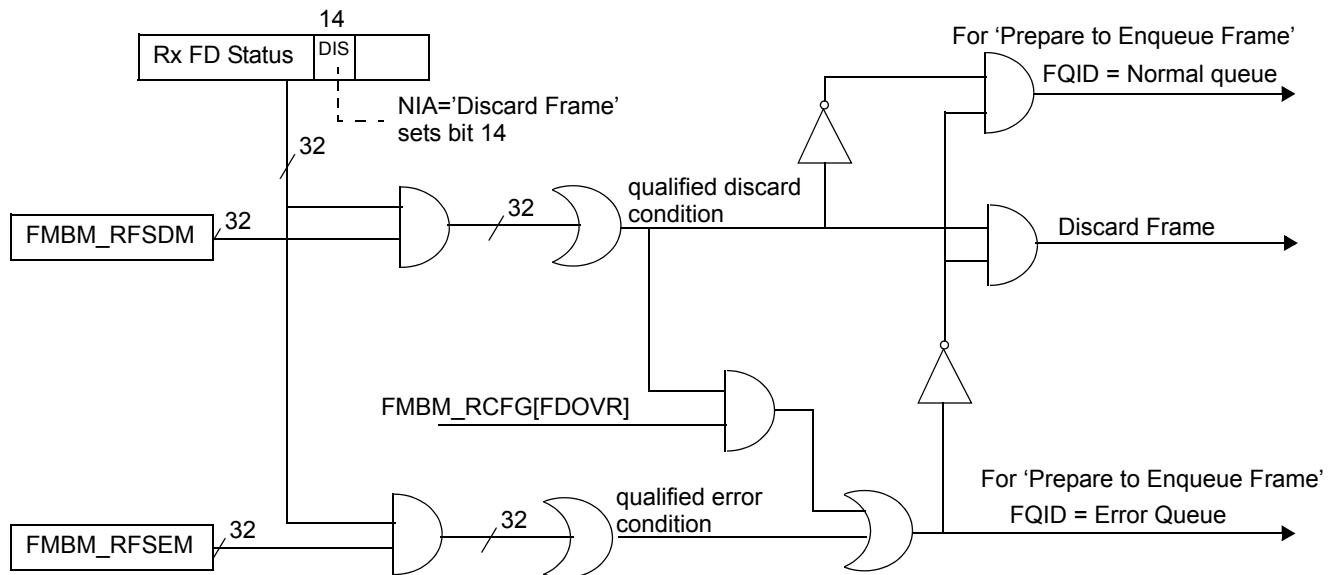
At the end of the receive flow or Offline flow, the BMI receives one of two Action Codes in the NIA: ‘Discard Frame’ or ‘Prepare to Enqueue Frame’ (see [Section 5.5.3, “BMI Input NIA”](#)). These action codes are evaluated together with the Rx FD status word (see “[Rx Frame Descriptor Status Word](#)”), the FMBM\_RFSDM, FMBM\_RFSEM and FMBM\_RCFG[FDOVR] (or FMBM\_OFSDM, FMBM\_OFSEM and FMBM\_OCFG[FDOVR] for Offline ports) to determine if the frame is discarded or enqueued. If the frame is enqueued, either an error queue or a normal queue is selected. This section describes what affects the decision to discard, enqueue to a normal queue or enqueue to an error queue. Refer to [Section 5.5.4.5.17, “Rx Frame Status Discard Mask Register \(FMBM\\_RFSDM\),”](#) [Section 5.5.4.5.18, “Rx Frame Status Error Mask Register \(FMBM\\_RFSEM\),”](#) [Section 5.5.4.5.1, “Rx Configuration Register \(FMBM\\_RCFG\)”](#) for details about the registers.

#### **NOTE**

This decision logic is not used for Host Commands.

The following diagram depicts the decision logic:

**Note:** For Offline ports, respective registers are used.



**Table 5-190. BMI Rx/O/H Flow - Enqueue or discard decision logic**

The enqueue or discard decision logic operates as follows:

- Error destination queue has highest priority, followed by discard, followed by normal destination queue.
- The NIA action code ‘Discard Frame’ results in setting Rx FD Status[14]. This bit trickles through the decision logic. Note that Rx FD Status[14] is also used as a status bit (named ‘DIS’), as described in “[Rx Frame Descriptor Status Word](#).”
- A qualified error condition exists if at least one bit in the Rx FD Status and its corresponding bit in FMBM\_RFSEM register are set.
- A qualified discard condition exists if at least one bit in the Rx FD Status and its corresponding bit in FMBM\_RFSDM register are set.
- The frame destination queue is an error queue (its FQID is configured in FMBM\_REFQID) if one of two conditions occur:
  - A qualified error condition exists
  - OR
  - A qualified discard condition exists and the discard override bit (FMBM\_RCFG[FDOVR]) is set.
- The frame is discarded if the conditions for enqueue to error queue are not valid and a qualified discard condition exists.
- If the frame it is not discarded and its destination queue is not an error queue, its destination queue is a normal queue. The FQID initial value is configured in FMBM\_RFQID, and optionally altered by the results of the Classification.

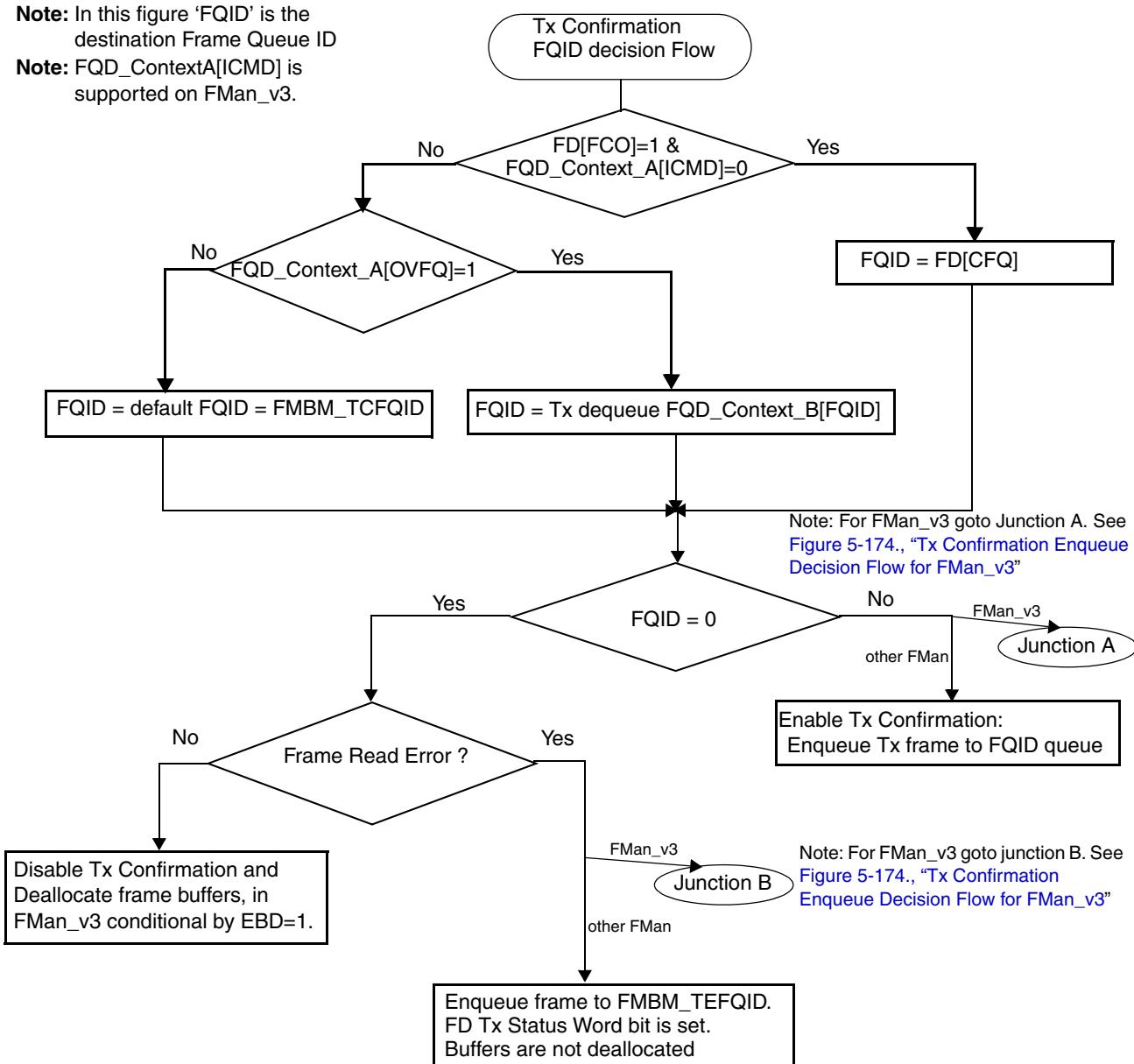
- If the frame is not discarded the BMI completes the ‘Prepare to Enqueue Frame’ action. Then the FMan continues execution with the NIA from FMBM\_RFENE. Normally this NIA invokes the QMI to enqueue the frame.

### 5.5.6.9 Tx confirmation enqueue and buffer deallocation decision

The user may configure the FMan to enqueue the frame after transmission into a queue. If Tx confirmation is enabled, the FMan does not release the external buffers associated with the frame. In this way, the CPU is able to monitor errors on transmitted frames, release the buffers when appropriate or use the frame for other purposes. This mechanism may be useful for the implementation of multicast to multiple physical ports, allowing the CPU to release the external buffers when all physical ports have transmitted the frame.

The following chart depicts the decision flow to enable/disable Tx confirmation and how the destination FQID is determined:

**Note:** In this figure ‘FQID’ is the destination Frame Queue ID  
**Note:** FQD\_ContextA[ICMD] is supported on FMan\_v3.

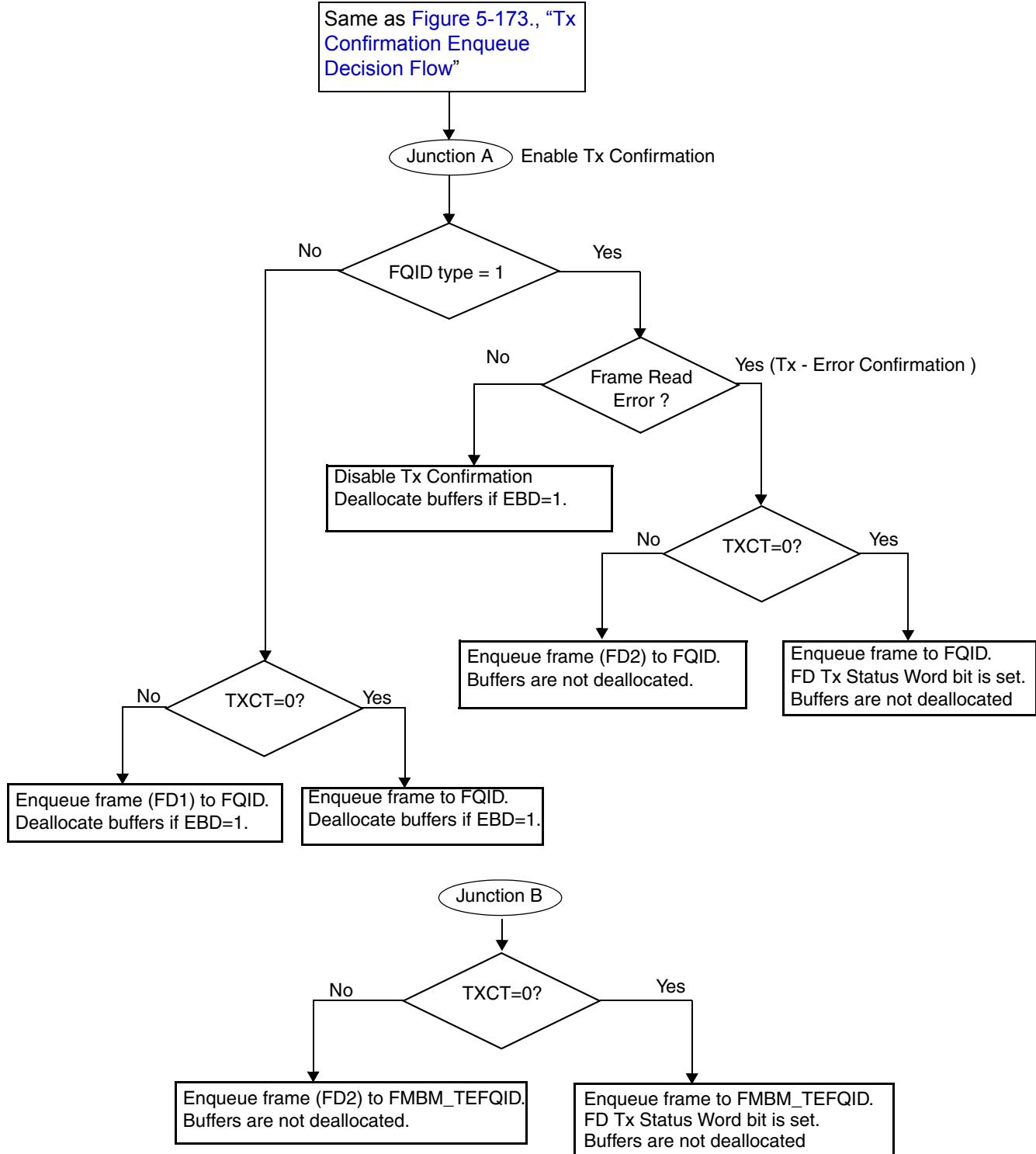


**Figure 5-173. Tx Confirmation Enqueue Decision Flow**

For details on Frame Read Error see [Section 5.5.6.20.5, “Unsupported Frames”](#) and [Section 5.5.6.20.7, “DMA Error.”](#)

Additional fields in Context A affect the operation of the FMan. Refer to [Section 5.4.3.1.1, “Frame Queue Descriptor \(FQD\) Context A”](#) for more details.

In FMan\_v3 (and following versions of FMan) the Tx Confirmation Enqueue decision flow is described in [Figure 5-174, “Tx Confirmation Enqueue Decision Flow for FMan\\_v3.”](#)

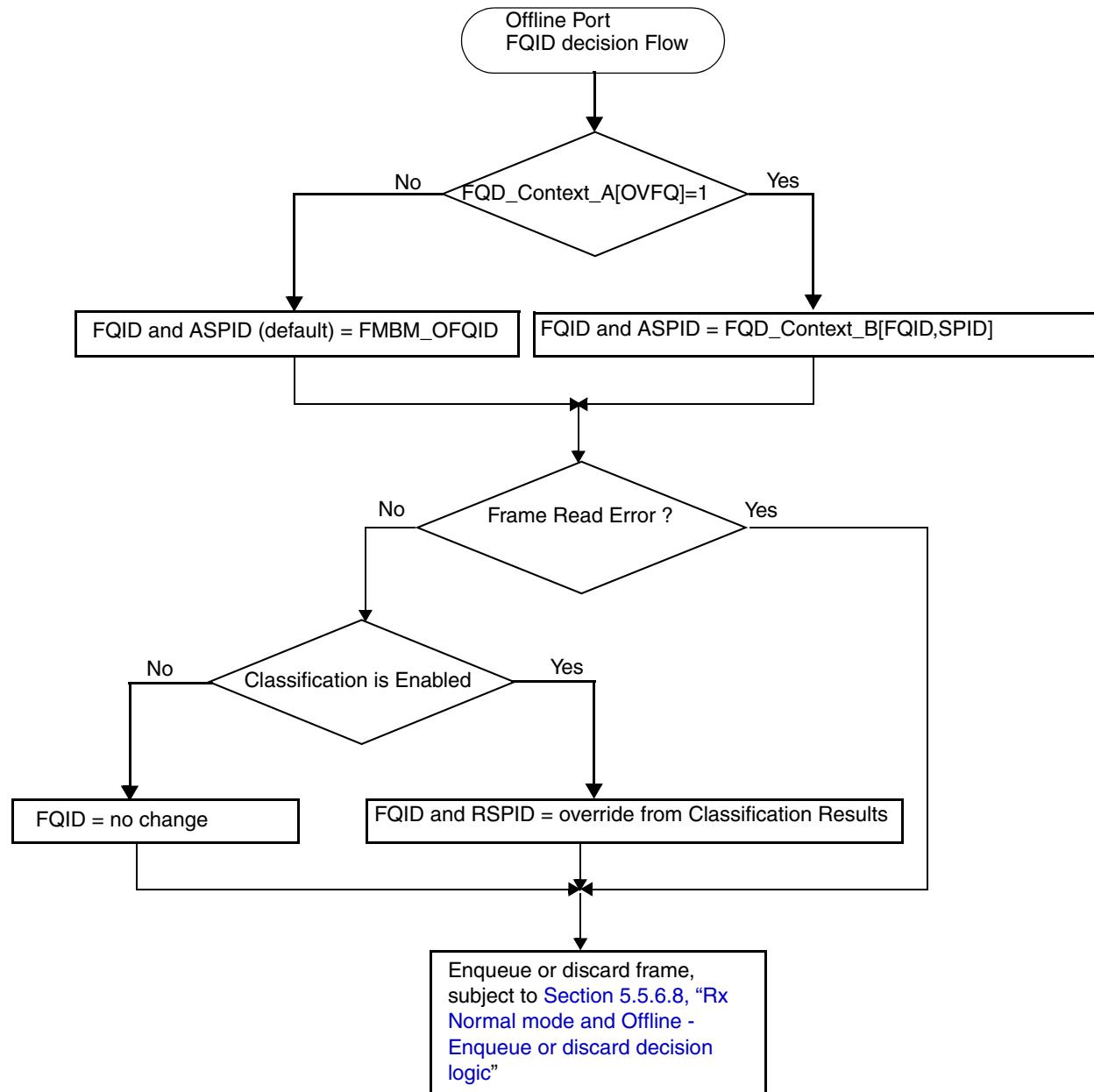


**Figure 5-174. Tx Confirmation Enqueue Decision Flow for FMan\_v3**

### 5.5.6.10 Offline Port Enqueue Decision Flow

At the end of the Offline Port flow, the FMan enqueues the frame into a queue.

The following chart depicts the decision flow to determine the destination FQID for Offline Port enqueue:



**Note:** In this figure ‘FQID’ is the destination Frame Queue ID, and ‘RSPID’ is the storage profile ID.

**Note:** FD Command[FCO] bit must be zero, since the CFQ field in the FD is used for parse results.

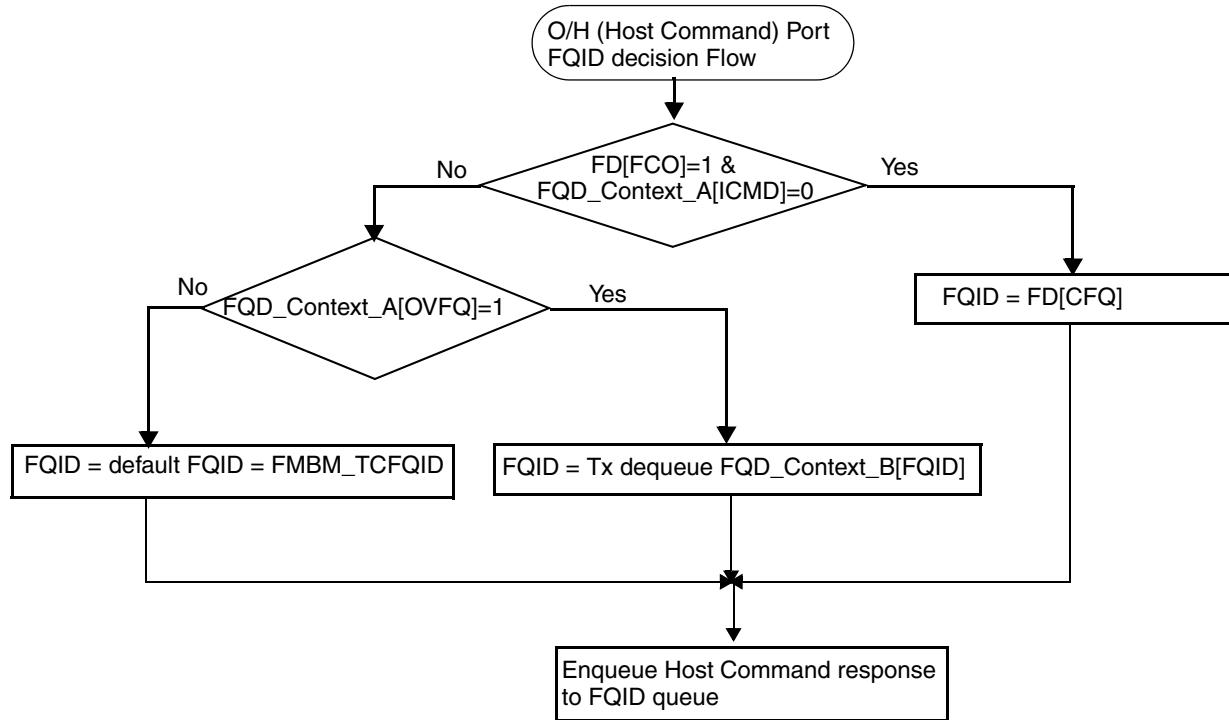
**Note:** FQID=0 does NOT disable enqueue as in Tx confirmation. **In Offline ports, the value FQID=0 is not allowed.**

**Table 5-191. Offline Port Enqueue Decision Flow**

### 5.5.6.11 Host command enqueue decision flow

At the end of the Host command flow (on the O/H port), the FMan enqueues the response into a queue.

The following chart depicts the decision flow to determine the destination FQID for host Command enqueue on the O/H port:



**Note:** In this figure 'FQID' is the destination Frame Queue ID

**Note:** The user must ensure that the FQID resulting from this flow is NOT zero (FQID=0 is an illegal value).

**Figure 5-175. Host Command Enqueue Decision Flow**

### 5.5.6.12 DMA resource Load balancing

To avoid one hardware port taking over the DMA resources, a limitation for open outstanding DMA requests is applied in the BMI. By writing to register FMBM\_PP, software can program limits that configure the maximum number of open DMA requests per port. In this way load balancing of the DMA resource is achieved. Each port is allowed to submit up to FMBM\_PP[MXD] outstanding DMA requests according to its configuration.

Also see [Section 5.5.6.13, “FMan DMA Priority Elevation.”](#)

### 5.5.6.13 FMan DMA Priority Elevation

The FMan is able to dynamically elevate its priority in the arbitration over the SoC system bus. This mechanism minimizes Ethernet FIFO overrun and underrun events when the link is overloaded, while the Host CPUs benefits from high priority on the SoC bus, when the Ethernet links operate in normal load.

Any of the following conditions causes the elevation of priority over the SoC bus:

- For any of the Rx port: The total number of buffers that are currently in use and associated with this specific Rx port exceed the amount specified in FMBM\_RFP[PEL].

For any of the Tx port: The Tx port has the amount of valid data on transmit buffer smaller than the amount specified in FMBM\_TFP[MFL].

### 5.5.6.14 ICID (Isolation Context Identifier)

The FMBM\_SPICID register shown in [Section 5.5.4.4.13, “SPICID Register \(FMBM\\_SPICID\\_1–63\)”](#) is used to select a virtual storage profile and to specify the ICID. This section focuses on the ICID.

ICID is the isolation context identifier, which on each transaction is provided to the system bus. The ICID allows transactions to be authenticated and translated as needed before entering the SMMU. The complete ICID code is produced in the FMan DMA. There are up to 63 registers, one per port (FMBM\_SPICID\_1 at 0x0\_0304, FMBM\_SPICID\_2 at 0x0\_0308.. FMBM\_SPICID\_63 at 0x0\_03FC). The suffix of the register represents the port ID associated with the configured ICID.

The 8 bit ICID provided to the system is generated as follows:

$$\text{ICID} = \{2'b0, \text{ICID}\}$$

With this mechanism it is possible to have a different ICID for each hardware port.

See [Section 5.8.4.13, “FMan DMA PortID Registers \(FMDM\\_PortIDn\).”](#)

The following bullets describe the source of ICID in various cases:

- If Tx port is configured in independent mode, the 8 bit ICID is configured in FMBM\_SPICIDn and copied into the designated field in Frame Descriptor (FD). The FMan uses this information when DMA generates a transaction on the system bus.
- When reading a frame from external memory (Tx port is operating in normal mode and O/H port), the ICID is taken from the FD as configured by the user.
- When writing a frame to external memory (Rx ports in independent or normal mode, or in FMan\_v3, O/H ports when configured to copy the frame into a new buffer) ICID is configured for each hardware port, from the FMBM\_SPICID register into the new FD built by the FMan. If virtual storage profile is enabled, the ICID is taken from the profile.
- Offline ports use the ICID from the FD when reading from external memory, and use ICID from the virtual storage profile (if enabled) when writing to external memory.

Note that the 8-bit ICID in the FD is split into two fields {2'bEICID, 6'bICID}.

ICID bits are configured by direct read and write by management. The host command does not write these bits.

### 5.5.6.15 Congestion or Rejection Handling

The BMI is connected to the QMan via a special hardware portal. Through this portal, the BMI may receive the following messages:

- Congestion
  - Congestion on Rx port: Congestion group has entered/exited congestion state. This is the earliest notification about congestion in one of the congestion groups within the QMan. Upon receiving this message, the BMI decodes the congestion group to detect which Rx port is associated with this group. The decoding is based on FMBM\_RCGM $< n >$ \_y ( $y = 1..8$ ) registers which define which groups are related to which physical port. Rx congestion leads to transmission of pause frame to the Tx port paired with the receiver. The Tx port continues sending pause frames (each time one pause time is expired, the next pause frame is sent) until the congestion in the QMan has resolved. In FMan\_v3, the priority vector of the flow control frame is determined by the congestion group priority[CGPx].
  - Congestion on O/H port: When an O/H port is congested, the BMI stops issuing new frame dequeue commands on the congested port, until it receives a message that the congestion group has exited the congestion state. For details see [5.5.4.7.24, “FMBM\\_OCGM - Observed Congestion Group Map.”](#) In FMan\_v3 congestion state leads to send priority enable vector message to QMan, that may disable dequeue from certain priority levels in the sub portal work queue, as configured in congestion group priority[CGPx].

Also see [Section 5.5.6.16, “Pause Frames for Flow Control.](#)

- Frame rejection, due to following reasons:
  - Tail drop management
  - RED/WRED congestion avoidance
  - Retired or out of service FQ
  - Invalid FQID (FQID=0)
  - Catastrophic error (for example, multi ECC error in the QMan)

See QMan specification for more details on frame rejection.

This message indicates that an enqueue request initiated for a particular frame is rejected due to one of the reasons described above. Frames discarded due to QMan enqueue rejection are counted in the relevant statistics counter (FMBM\_RFDC, FMBM\_OFDC).

Note that tail drop management and RED/WRED are valid reasons for frame discard. Other reasons should not happen and may indicate a possible programming error. In such case, the QMan error capture registers should be checked.

Note also that enqueue of Tx confirmation frames use color override, and it is not possible for QMan to reject those kind of frames due to tail drop or RED/WRED reasons.

### 5.5.6.16 Pause Frames for Flow Control

To avoid overflow conditions in any of the Rx ports, the BMI can request to send the pause frame from any MAC. Note that the BMI does not define the pause quanta to be sent; the number is defined in Ethernet MAC registers.

Flow control frames are sent if at least one of the following conditions is met:

- When the Rx port consumes too many internal buffers and crosses the configurable FIFO threshold (defined as number of FMan memory buffers). The BMI continues to assert a pause frame request to the MAC until the total number of busy buffers occupied by this port declines under the threshold. This threshold is programmed by writing to FMBM\_RFP[FTH].
- When external buffers are depleted in one or more BMan pools. Software can program the combinations of depleted BMan pools in which it wants the pause frame to be sent. For more information, see [Section 5.5.4.5.24, “BMan Pool Depletion Register \(FMBM\\_RMPD\).”](#)
- When the BMI receives a congestion message from the QMan. The congestion may occur in any of N congestion groups. N varies from device to device. The BMI decodes this group to the relevant MAC according to the congestion group map as configured in FMBM\_RCGM\_0–7. See [Section 5.5.6.15, “Congestion or Rejection Handling,”](#) for more information.

The Ethernet MAC sends flow control frames as long as the condition persists.

### 5.5.6.17 Tx and O/H Rate Limiter

Software can limit the rate in the BMI for Tx, Offline Port, and host commands flows. The rate limit can be expressed in terms of bits/sec or frames/sec, depending on flow to which the rate limiter is applied. The rate limiter is enabled when FMBM\_TRLMTS[EN] = 1 for Tx ports and FMBM\_ORLMTS[EN] for Offline Port/Host commands ports. The algorithm of the rate limiter is based on the token bucket mechanism. For programming details, see [Section 5.5.4.6.12, “Tx Rate Limiter Scale Register \(FMBM\\_TRLMTS\),”](#) and [Section 5.5.4.7.21, “O/H Rate Limiter Scale Register \(FMBM\\_ORLMTS\).”](#)

The token bucket is a control mechanism that dictates when traffic can be transmitted, based on the presence of tokens in the bucket. The token bucket contains tokens and has programmable size in terms of bits/frames. Each token has a programmable size in terms of bits/frames. The bucket is getting filled at filling rate by adding new tokens to the bucket. This rate is predefined by FPM configuration. Each time the BMI is about to transmit a frame through a Tx port, it checks whether frame size is below the total size of all tokens in the bucket. When the tokens are present and above the frame size, a flow is allowed to transmit traffic and the bucket is reduced by the frame size.

If the frame is transferred via Offline Port/Host command port the behavior differs between the devices: The bucket size is represented by multiples of 1000 tokens.

It is possible to configure the granularity to single token by setting FMBM\_ORLMT[BSG]. Before each dequeue operation, the BMI checks if the bucket has at least one token and the bucket is reduced by one token (frame).

In O/H we count frames in Tx we count bytes. Single bucket rate limiter. In Tx burst size is in KBytes. In O/H the burst size work in Kpackets.

If there are no tokens in the bucket, a flow cannot transmit its packets. Therefore, a flow can transmit traffic up to its max burst size if there are adequate tokens in the bucket before the rate limiter starts regulating the traffic. The bucket accumulates tokens at a programmed rate for the next frame transmission.

For details on Tx port rate limiter programming refer to [Section 5.5.4.6.12, “Tx Rate Limiter Scale Register \(FMBM\\_TRLMTS\),”](#) and [Section 5.5.4.6.13, “Tx Rate Limiter Register \(FMBM\\_TRLMT\).”](#)

Figure 5-176 shows the token bucket mechanism.

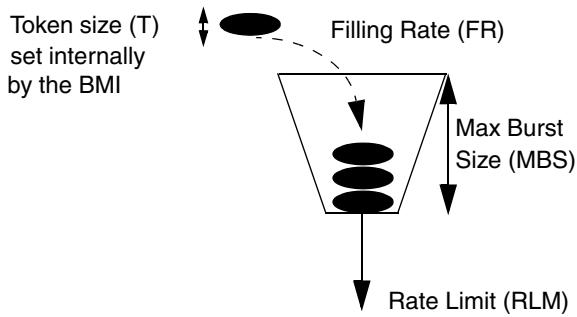


Figure 5-176. BMI Token Bucket Mechanism

### 5.5.6.18 Internal FIFO Configuration Requirements

#### NOTE

In the following paragraphs, ‘IFSZ’ is used instead of ‘FMBM\_PFS[IFSZ]’ and ‘DPDE’ is used instead of FMBM\_TFP[DPDE] or FMBM\_OFP[DPDE].

In the following paragraphs, when referring to the parameters IFSZ, MXT, and DPDE, the reference means the actual configuration selection and not the value which is written into the register field. For example, if a certain register is programmed such that  $FMBM\_PP\_n[MXT] = 0$ , it means that the “maximum number of outstanding tasks” is set to 1, and that is the number to be used in the equation for the restriction. Same rule applies to DPDE and IFSZ.

#### 5.5.6.18.1 Internal FIFO for Rx Ports

The relationship between the size of the internal FIFO (configured in FMBM\_PFS[IFSZ]) and the external buffer pool sizes (configured in FMPB\_EBMPI[PBS]) affects the way the BMI allocates external buffers for the incoming frames. Erroneous configuration could cause a deadlock, or unwanted dropped frames or less than optimal buffer allocation. Please use the recommended value as described in [Section 5.5.4.4.12, “Port FIFO Size Register \(FMBM\\_PFS\\_1–63\)](#).” The constraints described in this section are minimal values and not optimal values. They should be used by SW error checking routine.

#### NOTE

The IFSZ field is programmed with the number of buffers (each buffer 256 bytes) allocated to the port FIFO, i.e the size of the FIFO is  $IFSZ * 256$  bytes.

The following rule must be applied when programming the registers:

- In FMan\_v3

The  $IFSZ \geq (\text{max frame size rounded up to 256-byte boundary}) + 3 * 256$  bytes.

If an internal frame offset exists ( $FMBM\_RIM[FOF] > 0$ ) it should be added to the frame size. If this rule is violated, the frame is truncated to the size fitting the rule, and FD[FSE] bit is set.

- The available external buffers must be large enough such that 16 buffers hold the largest incoming frame (organized in a S/G list). Frames that do not fit in 16 buffers are dropped.
- If the Rx port is configured for Independent mode, IFSZ >= frame size rounded up to  $256 + 4 * 256$  bytes. If an internal frame offset exists (FMBM\_RIM[FOF] > 0) it should be added to the frame size.
- If excessive buffering is defined (FMBM\_PFS<n>\_y[EXBS] > 0), the user must preserve buffers in the excessive pool at least as the number of Rx ports which are enabled in which EXBS > 0. The excessive pool size is defined as “total free pool buffer size - total ports’ committed internal FIFO size”. “total free pool buffer size” is defined in FMBM\_CFG1[FBPS], and “total ports’ committed internal FIFO size” is the sum of the values defined in FMBM\_PFS[IFSZ] for all the enabled ports.

#### **5.5.6.18.2 Internal FIFO for Tx Ports**

If Tx port is configured to normal mode:

- In FMan\_v3: IFSZ must be at least ((frame size rounded up to 256) +  $4 \times 256 + DPDE \times 256$ ) bytes. If the system attempts to transmit a frame which is bigger than this rule, it is not transmitted.

If Tx port is configured for Independent mode, IFSZ must be at least (frame size rounded up to  $256 + 3 \times 256$ ) bytes.

#### **5.5.6.18.3 Internal FIFO for O/H Ports**

- In FMan\_v3: IFSZ must be at least ((frame size rounded up to 256)+  $5 * 256 + DPDE * 256$ ) bytes, if BMI must bring entire frame, or at least ( $6 * 256 + DPDE * 256$ ) bytes if BMI brings in only the header of the frame. If an internal frame offset exists (FMBM\_OIM[FOF] > 0) it should be added to in the frame size.

### **5.5.6.19 Hardware Assist for IEEE 1588-Compliant Timestamping**

There is a push in industrial control applications to use Ethernet as the principal link layer for communications. This requires Ethernet to be used for both data transfer and real-time control. For real-time systems, each node is required to be synchronized to a master clock. The precision of this clock is dictated by the application, but generally needs to be of the order of <1uSec for high-speed machinery (for example, printing presses). Because Ethernet was never designed to have real-time support in the link layer, upper layer protocols (specifically TCP/IP) are used to enable this. IEEE 1588 specifies a mechanism for synchronizing multiple nodes to the same master clock.

Support for IEEE 1588 can be done entirely in software running on a host CPU, but applications that require sub-10μSec accuracy need hardware support for accurate timestamping of incoming packets. This section describes the flows for incoming/outgoing SYNC frames.

The BMI supports the IEEE 1588 protocol for synchronizing multiple clocks over networks.

On Rx flow, the Ethernet MAC samples the 8 byte timestamp which is placed in the appropriate location in the Internal Context (IC). The user may configure the BMI to copy parts of the IC into a margin at the beginning of the first buffer of the frame. This is done by programming the FMBM\_RICP register. In this way, the timestamp is passed to the host CPU.

On Tx flow, the user may enable confirmation of frames that need to return the timestamp on transmission. See [Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision”](#) for details on this aspect. In this case, the user also configures the BMI to copy parts of the IC into a margin at the beginning of the first buffer of the frame. This is done by programming the FMBM\_TICP register. In this way, the timestamp is passed to the host CPU.

#### NOTE

If the timestamp mechanism is not enabled in the MAC, the BMI writes zero in the timestamp location and proceeds as usual. It is not allowed to change the state of TS\_EN in the MAC configuration while the BMI Tx port is enabled or busy.

### 5.5.6.20 Error Handling

This section describes the types of errors that may be encountered.

#### 5.5.6.20.1 Non-Recoverable Errors

Non-recoverable errors are reported via interrupt and reflected in FMBM\_IER. When such an error is detected, the operation of the BMI is undefined because it may lose critical data. It is expected that the BMI should be reset and re-initialized when such an error occurs. Non-recoverable errors are as follows:

1. Pipeline Table RAM multiple ECC Error. The RAM has built-in error checking and correction (ECC) mechanism. Whenever BMI detects a multi-bit error during a read transactions, the FMBM\_IEVR[SPEC] event is set. If FMBM\_IER[SPECE] is set, then an interrupt to host will be activated. FMan\_v3, This error condition is replaced by Storage profile RAM ECC Error.
2. Linked list RAM ECC Error. This internal RAM has built-in error checking and correction (ECC) mechanism. Whenever BMI detects multi-bit error during a read transaction, the FMBM\_IEVR[LEC] event will be set. If FMBM\_IER[LECE] is set, then an interrupt to host will be activated.
3. Statistics RAM ECC Error. The statistic counters RAM has built-in error checking and correction (ECC) mechanism. Whenever BMI detects multi-bit error during a read transaction, the FMBM\_IEVR[STEC] event will be set. If FMBM\_IER[SECE] is set, then an interrupt to host will be activated.

In FMan\_v3 additional non-recoverable error is supported:

4. Dispatch RAM ECC Error. This internal RAM has built-in error checking and correction (ECC) mechanism. Whenever BMI detects multi-bit error during a read transaction, the FMBM\_IEVR[DEC] event will be set. If FMBM\_IER[DECE] is set, then an interrupt to host is activated.

#### 5.5.6.20.2 Network Errors

Frame reception from the network may encounter certain error conditions. Such errors are reported by the Ethernet MAC when the frame is transferred to the BMI. The action taken per error case is described below. Besides the interrupts, the BMI is capable of recognizing several conditions and setting a corresponding flag in FD status field for Host usage. These conditions are as follows:

- Physical Error. One of the following events were detected by the Ethernet MAC: Rx FIFO overflow, FCS error, code error, running disparity error (in applicable modes), FIFO parity error, PHY Sequence error, PHY error control character detected, CRC error. The BMI discards the frame, or enqueue directly to EFQID if FMBM\_RCFG[FDOVR] is set. Note that FMBM\_RFSDM and FMBM\_RFSEM programming may effect the flow according to the state of the corresponding bit in either one of them (refer to register description). When working in independent mode, FMBM\_RCFG[FDOVR] state determines if the frame is discarded or continues to be processed by the FMan controller. FPE bit is set in the FD status.
- Frame size error. The Ethernet MAC detected a frame that its length exceeds the maximum allowed as configured in the MAC registers. The frame is truncated by the MAC to the maximum allowed, and it is marked as truncated. The BMI sets FSE in the FD status and forwards the frame to next module in the FMan as usual.
- Some other network error may result in the frame being discarded by the MAC and not shown to the BMI. However, the MAC is responsible for counting such errors in its own statistics counters.

#### **5.5.6.20.3    Buffer Depletion**

Buffer depletion can cause a received frame to be discarded. If one of the following condition occurs the BMI discards the incoming packet:

- The incoming frame does not fit in 16 buffer of the scatter/gather list (i.e. max of 16 scatter/gather buffers are supported).
- No buffer to store the entire scatter/gather list (including margin). The minimal buffer size required is 256bytes+start margin configured in the BSM field (in FMBM\_REBM or FMBM\_VEBM - if virtual storage profile is selected).
- No buffer to store the entire frame header. The minimal buffer size required is 256bytes+start margin configured in the BSM field (in FMBM\_REBM or FMBM\_VEBM - if virtual storage profile is selected).
- If SGD = 1 and not able to store the entire frame (including start and end margins) in a single buffer.

When this situation occurs, the frame is discarded and the appropriate discard counter is incremented. Buffer depletion indications from Bman can be mapped by FMBM\_RMPD configuration to pause traffic on the associated ports. When associated by configuration of an Rx port it is used to generate PAUSE frame request to the appropriate MAC. In FMan\_v3, when associated by configuration of an O/H port it is used to stop dequeue operation of new frames. Once the configured buffer depletion state ends its associated Rx pause request or O/H traffic internal pause is cancelled and traffic is allowed to resume. In FMan\_v3, priority based flow control is supported. This mechanism allows for the generation of PAUSE frames with specific priority levels. The priority is selected in the virtual storage profile FMBM\_RMPD[PFCPEV].

#### **5.5.6.20.4    Queue Error**

If the enqueue operation is using an illegal FQID, the QMan rejects the operation and logs the error. If this occurs, it may indicate incorrect programming of FQID registers.

#### **5.5.6.20.5    Unsupported Frames**

These error conditions are checked when the BMI reads the frame during Tx or O/H ports dequeue.

UFD: The FMan supports two types of Frame Descriptors (FDs): Short single buffer simple frame and Short multi buffer simple frame. See [Section 5.4.3.2, “Frame Descriptor \(FD\).”](#) In addition to this, the FMan does not support scatter/gather frames in which one of the entries has the extension bit set. The FD[Status] UFD bit is set on the error queue or confirmation queue (or default FQID for O/H ports).

#### NOTE

In the O/H ports, the UFD bit should never be set in FMBM\_OFSDM, because the BMI cannot discard a frame and release its external buffers when the format is unknown.

LGE: The FMan is not able to read frames with a length larger than the FIFO size allocated for a specific Tx or O/H port. If the frame length is bigger than 65535, or if the frame length contradicts the requirements that were listed in [Section 5.5.6.18.2, “Internal FIFO for Tx Ports”](#) or in [Section 5.5.6.18.3, “Internal FIFO for O/H Ports,”](#) such frames are discarded, meaning they are not transmitted or processed. The LGE bit is set on the error queue or confirmation queue. In FMan\_v3 additional reasons for Length Error are:

- The frame is stored in a scatter/gather format and the sum of entries' length is smaller than the total length as it appears in the FD.
- The frame is stored in a scatter/gather format and the length in an entry is bigger than 65535.
- The frame is stored in a scatter/gather format and no "Final" bit was detected after reading 16 entries.

The enqueue/discard decision is taken based on the FD[Status] and mask registers (FMBM\_OFSEM, FMBM\_OFSDM (for O/H)).

#### 5.5.6.20.6 Timestamp Errors

Timestamp errors occur when the BMI is expected to deliver a timestamp but the MAC is programmed to not do so. This is also true for Rx and Tx. In this case, the BMI recognizes the fault and delivers a zero timestamp.

#### 5.5.6.20.7 DMA Error

DMA errors can occur during DMA write or read operations due to random ECC errors detected on the target memory, or due to an illegal address that may be the result of a programming error. The FMan and the BMI react to DMA error detection according to the following cases:

- DMA error detected during receive flow—The frame is enqueued to FQID as usual. However, the DME bit is set in the FD status word.
- DMA error detected during offline parse/host command when the frame/command is read from main memory to FMan memory—in this case, the frame/command is not processed. The BMI enqueues the frame directly to EFQID and the DME bit is set in the FD status word.
- DMA error detected during offline parse/host command when the frame header is written back to main memory—in this case, the frame/command is already processed. The BMI enqueues the frame to the FQID as requested by the FMan module. However, the DME bit is set in FD status word.

- DMA error detected during transmit flow when the frame is read from main memory to FMan memory—In this case, the frame is not transmitted and DME bit is set in the FD status word. See [Section 5.5.6.9, “Tx confirmation enqueue and buffer deallocation decision”](#) for more details.
- DMA error detected on DMA read of the scatter/gather list—When the BMI detects WRED rejection of a certain frame, it acts to release the external buffers that are allocated to this frame. If this is a scatter/gather type of frame, the BMI acts to copy the scatter/gather list into FMan memory, to obtain the buffers pointers and BPIDs. If a DMA error is detected while trying to read the scatter/gather list, the buffers cannot be released and the BMI skips the attempt. Software should read the DMA error capture registers and handle the situation. Note that scatter/gather read type can be observed by reading the DMA tag bits in the error capture registers.

### **5.5.6.21 Graceful Stop**

Graceful stop is defined as the flow in which the hardware does not accept new frames, but finishes all frames or tasks that had already begun.

BMI graceful stop is initiated when FMBM\_RCFG/FMBM\_TCFG/FMBM\_OCFG[EN] is cleared. The process is complete when all tasks associated with the port were terminated (transmitted, enqueued or discarded, according to their normal flow). At this time the relevant bit FMBM\_RST/FMBM\_TST/FMBM\_OST[BSY] shows ‘0,’ which means “not busy.”

For Tx and O/H ports, it is required to stop the port in QMI, wait for stop complete acknowledge, and only then stop the port in the BMI.

### **5.5.6.22 Debug Capabilities**

The FMan contains internal hardware infrastructure to support frame-based debug processing. Three debug flows (A, B, C) can be programmed, meaning that three types of packets can be debugged concurrently.

A debug flow includes flow initialization, flow trap, and trap action. The debug flow details are described in [Section 5.4.5, “FMan Debug.”](#) Here we shall focus on BMI-specific details.

In addition to this, global debug capability is controlled by FMBM\_GDE register, including global enable and external “global debug enable” signal,

#### **5.5.6.22.1 Flow Initialization**

A debug flow is initialized by the BMI, when initializing a task and calling the next module. The debug marks are set per flow unless the match operation fields are configured as “never match”.

#### **5.5.6.22.2 Flow Trap**

When the task returns to the BMI, for example, in Tx normal flow when QMI returns the task after dequeue, the BMI reads the FD from the IC and compare the unmasked bits against the values in the trap match registers. If a match is found, the trap is considered successful. The trap can be bypassed if the match operation is configured as “always match”; this is also called a successful trap. Each debug flow has its own trap registers.

Note that in Rx flow, because there is no dequeue operation, debug flow initialization and flow trap happen together.

Debug trap compares are done per debug flow, if the input debug marks of that flow are on (indicating that all previous modules kept them on), and the compare operation is “match if” (FMBM\_RDCFG/FMBM\_TDCFG/FMBM\_ODCFG[CMPOP] = “010”). The output debug marks are kept on if the match had been successful or if the compare operation is “always match”.

The trap operation at Rx flow is done twice, before PCD and before enqueue. The first trap compare masks (ignores) the last 16 bits of the FD, regardless of the programming of FMBM\_DCM registers, in order to be able to match those bits when the FD returns after the PCD process, before enqueue.

Similarly, the trap operation at O/H flow is done twice, before PCD and before enqueue. If FD[FCO] is ‘0’, the first trap compare masks (ignores) the last 16 bits of the FD, regardless of the programming of FMBM\_DCM registers, in order to be able to match those bits when the FD returns after the PCD process, before enqueue.

In FMan\_v3, the first trap (Rx and O.H flows) behaves like an “always match”, no compare done, unless trap is disabled (CMPOP = 000), in which the debug marks are cleared.

The trap operation at Tx flow is done only once, after dequeue operation.

### 5.5.6.22.3 Flow Trace

A successful trap causes debug trace information to be written into the debug portion of the frame’s internal context (IC). The amount of trace depends on the verbosity level which is programmed in the flow’s debug configuration register. If more than one flow is trapped, the BMI uses the highest, most verbose level among all trapped flows.

The debug trace information written into the frame’s IC is described in [Table 5-192](#).

**Table 5-192. BMI Trace Information**

Offset	Minimal Trace	Verbose Trace	Very Verbose Trace
0x0	BMI Prefix (4 bytes)	BMI Prefix (4 bytes)	BMI Prefix (4 bytes)
0x4	BMI port ID (4 bytes)	BMI port ID (4 bytes)	BMI port ID (4 bytes)
0x8	—	FPM time stamp (4 bytes)	FPM time stamp (4 bytes)
0xC	—	FD[32-63] (4 bytes)	Reserved (4 bytes)
0x10 - 0x1F	—	—	FD (16 bytes)

The BMI prefix structure is described in [Table 5-193](#). The BMI port ID structure is described in [Table 5-194](#).

**Table 5-193. BMI Prefix Description**

Bit	0–1	2–7	8–31
Description	TL (Trace level)	BMI trace size (8, 16 or 32 bytes, according to verbosity chosen)	BMI NIA <sup>1</sup>

**Note:**

- <sup>1</sup> BMI NIA is the BMI engine code and action code that had been received from previous engine prior to the BMI in the flow of the frame/command. When the BMI is the first engine of a frame processing (Rx flow before PCD), Action Code is set to 18b00, and Engine Code is set to 5b10100.

**Table 5-194. BMI Port ID Description**

Bit	0–15	16–20	21–23	24–31
Description	TL = 1,3 - Reserved (all zero) TL = 2 - FD[16–31] (high address bits)	Reserved (all zero)	Debug trap state {A,B,C}	port ID

#### 5.5.6.22.4 Trap Action

When the frame task reaches other modules, each module can clear the debug mark if programmed to disable the trap or if programmed to find a match and the match was not found. When the frame task returns to the BMI after classification is completed, and its debug mark in a specific debug flow is still set, and the BMI second trap is successful, it means that the frame trap is considered successful. In this case, the BMI tries to write the debug trace portion of the frame's IC to debug destination port, as programmed in the flow's configuration register. The amount of data to be written depends on the trace data that had been collected thus far, and the start address depends on the frame's IC parameters so that the debug trace data is appended to the normal IC block in external memory.

Trace destination port is selected out of the following: memory, or debug port (Nexus). If more than one flow is trapped, and different destinations are programmed, the behavior will be undefined. Therefore debug trace destination must be equally set in all flows. In addition to this, the BMI can instruct the FPM to halt the specific task (lowest halt level), halt the specific port (medium halt level) or halt all of the frame manager's ports (highest halt level), if programmed to do so in the debug flow's configuration registers. Again, if more than one flow is trapped and different halt levels are configured, the BMI follows the highest level of halt among all active traps.

Note that in Tx flow, debug trap actions only happen if transmit confirmation is applied (CFQID not equal zero), or if an unexpected event caused the frame to be enqueued to an error queue.

### 5.5.7 Frame Manager BMI Initialization/Application Information

#### 5.5.7.1 Initialization of Common Registers

Initialization of the common registers must occur prior to initialization of any specific port. Because common registers affect all ports, it must be initialized by the software entity that is aware of the activity

and demands of all of the BMI ports, including different software partitions. This responsibility is usually relegated to a layer of software often called the Hypervisor.

The following initialization steps should be followed:

1. Set the total size of the free buffer pool and its offset by writing to FMBM\_CFG1.
2. Set the total allowed tasks and DMA slots to be used by the BMI in FMBM\_CFG2.
3. Set the priorities (weights) per port in FMBM\_ARB.
4. Set the allocation of tasks and DMA slots per port (committed and excess) in FMBM\_PP.
5. Set the allocation of FIFOs per port (committed and excess) in FMBM\_PFS.
6. Set the ICID per port in FMBM\_SPICID.
7. In FMan\_v3, if necessary, initialize the virtual storage profiles.
8. Set the interrupt enable bits to the desired mode in FMBM\_IER.
9. Initialize the BMI linked list by writing 1 to FMBM\_INIT[STR].
10. Signal the specific port drivers that the common parameters are initialized, and that they can go ahead to initialize the port.

### 5.5.7.2 Rx Port Initialization Steps in Normal Mode

The initialization steps for Rx port of one MAC in normal mode are described as follows:

1. Set the DMA attributes to be used in FMBM\_RDA.
2. Set the FIFO parameters in FMBM\_RFP.
3. Set the desired frame margins parameters in FMBM\_RFED, FMBM\_RIM and FMBM\_REBM.
4. Set the Internal Context parameters in FMBM\_RICP.
5. Set the Hard Parser as next module in FMBM\_RFNE.
6. Set the attributes in FMBM\_RFCA.
7. Set the parser's NIA by writing to register FMBM\_RFPNE, start offset in register FMBM\_RPSO, initialize the parse results in FMBM\_RPRI registers and policer profile by writing to FMBM\_RPP.
8. Set FQID and EFQID by writing to FMBM\_RFQID and FMBM\_REFQID.
9. Configure the mask vectors in FMBM\_RFSDM and FMBM\_RFSEM according to the desired action of the BMI.
10. Configure the QMI enqueue as the next module in FMBM\_RFENE.
11. Configure FMBM\_REBMPI according to the external BMan pools allocated to the port. The valid pools should be programmed in ascending order. Information of the BMan pool with smallest buffers should be written into FMBM\_REBMPI\_0. In FMan\_v3, if required, define one or more of the buffer pools as backup pools, making it available when all enabled non-backup pools are out of buffers. In FMan\_v3, if necessary, initialize the virtual storage profiles.
12. Configure the FMBM\_CGM and FMBM\_RMPD according to the desired affect of Buffers depletion and Queue congestion on the specific port pause state.
13. In FMan\_v3, if necessary, initialize FMBM\_RCMNE.
14. Enable the statistics counters, if desired.

15. Enable the Rx port by setting FMBM\_RCFG[EN]. FMBM\_RCFG[IM] bit should be cleared.

### 5.5.7.3 Rx Port Initialization Steps in Independent Mode

The initialization steps for Rx port of one MAC in independent mode are described as follows:

1. Set the FIFO parameters in FMBM\_RFP.
2. Set the FMan controller as the next module in FMBM\_RFNE.
3. Set the attributes in FMBM\_RFCA.
4. Enable the Rx port by setting FMBM\_RCFG[EN]. FMBM\_RCFG[IM] bit should be set.

### 5.5.7.4 Tx Port Initialization Steps in Normal Mode (Example)

The initialization steps for Tx port of one MAC in normal mode are described as follows:

1. Prior to BMI Tx port initialization, it is required that the MAC registers are initialized and Tx operation in the MAC is enabled.
2. Set DMA attributes to be used in FMBM\_TDA.
3. Set FIFO parameters in FMBM\_TFP.
4. Set desired frame margins parameters in FMBM\_TFED.
5. Set Internal Context parameters in FMBM\_TICP.
6. Set QMI dequeue as next module in FMBM\_TFDNE.
7. Set CFQID and EFQID by writing to FMBM\_TCFQID and FMBM\_TEFQID.
8. Set QMI enqueue as next module in FMBM\_TFENE.
9. Configure rate limiter parameters in FMBM\_TRLMTS and FMBM\_TRLMT, if desired.
10. Enable statistics counters if desired.
11. Enable Tx port by setting FMBM\_TCFG[EN]. FMBM\_TCFG[IM] bit should be cleared.

### 5.5.7.5 Tx Port Initialization Steps in Independent Mode

The initialization steps for Tx port of one MAC in independent mode are as follows:

1. Prior to BMI Tx port initialization, the MAC registers are required to be initialized, and Tx operation in the MAC is enabled.
2. Set the FIFO parameters in FMBM\_TFP.
3. Set the desired frame margin parameters in FMBM\_TFED.
4. Set the FMan controller as next the module in FMBM\_TFDNE.
5. Set the FMan controller as next the module in FMBM\_TFENE.
6. Configure the rate limiter parameters in FMBM\_TRLMTS and FMBM\_TRLMT, if desired.
7. Enable the Tx port by setting FMBM\_TCFG[EN]. FMBM\_TCFG[IM] should be set.

### **5.5.7.6 Initialization Steps for Offline Ports**

The initialization steps for Offline Port/Host Command port in Offline Port mode are as follows:

1. Set DMA attributes to be used in FMBM\_ODA.
2. Set Internal Context parameters in FMBM\_OICP.
3. Set QMI dequeue as the next module in FMBM\_OFDNE.
4. Set hard parser as the next module in register FMBM\_OFNE. Set the attributes in FMBM\_OFCA.
5. Set the parser's NIA by writing to register FMBM\_OFPNE, start the offset in register FMBM\_OPSON, and initialize the parse results in FMBM\_OPRI and the policer profile by writing to FMBM\_OPP.
6. Set FQID and EFQID by writing to FMBM\_OFQID and FMBM\_OEFQID.
7. Configure the mask vectors in FMBM\_OFSDM and FMBM\_OFSEM according to the desired action of the BMI.
8. Configure the QMI enqueue as the next module in FMBM\_OFENE.
9. Configure the rate limiter parameters in FMBM\_ORLMTS and FMBM\_ORLMT, if desired.
10. In FMan\_v3, configure the FMBM\_OCGM according to the desired effect of queue congestion on the specific port internal pause requirement.
11. In FMan\_v3, if necessary, initialize FMBM\_OCMNE.
12. Enable the statistics counters, if desired.
13. Enable the port by setting FMBM\_OCFG[EN].

### **5.5.7.7 Initialization Steps of Host Command Ports**

The initialization steps for Offline Port/Host Command port in host command mode are as follows:

1. Set the DMA attributes to be used in FMBM\_ODA.
2. Set Internal Context parameters in FMBM\_OICP.
3. Set the QMI dequeue as the next module in FMBM\_OFDNE.
4. Set the FMan controller as next the module in FMBM\_OFNE. Set the attributes in FMBM\_OFCA.
5. Set FQID and EFQID by writing to FMBM\_OFQID and FMBM\_OEFQID.
6. Configure the mask vectors in FMBM\_OFSDM and FMBM\_OFSEM according to the desired action of the BMI.
7. Configure the QMI enqueue as the next module in FMBM\_OFENE.
8. Enable the statistics counters, if desired.
9. Enable the port by setting FMBM\_OCFG[EN].

## 5.6 Frame Manager—Queue Manager Interface (QMI)

### 5.6.1 QMI Overview

The FMan Queue Manager interface (QMI) module provides an interface to the QMan for enqueueing and dequeuing frames to/from the queues in the Data Path Acceleration Architecture (DPAA). The QMI supports the following types of FMan hardware ports:

- Receive (Rx)
- Transmit (Tx)
- Offline/Host Command (O/H)

See [Section 5.3.1, “FMan Hardware Ports Types,”](#) for detailed descriptions of these ports.

The QMI also serves as an error-reporting mechanism from the FMan to the system.

This figure illustrates the QMI connection block diagram.

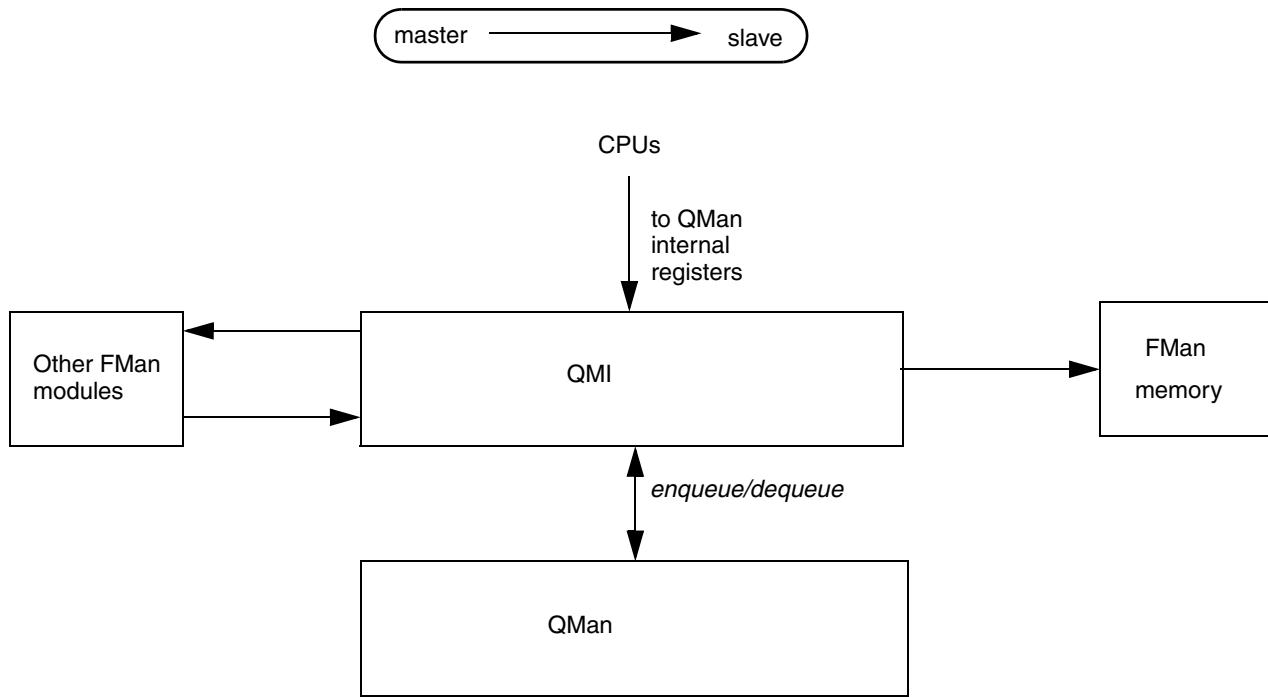


Figure 5-177. QMI Connection Block Diagram

#### 5.6.1.1 QMI Features Summary

The FMan QMI includes the following features:

- General
  - Supports graceful stop operation
- Enqueue
  - Supports enqueue to QMan for:

- Ethernet Transmitter (Tx) confirmation
- Ethernet Receiver (Rx)
- Host command
- Offline port
- Pipeline depth for processing multiple enqueue tasks concurrently
- Dequeue
  - QMan sub-portals (SPs)
  - Supports dequeuing for all of the supported TX and O/H portIDs
  - Each portID can be configured as follows:
    - Mapping to a specific Sub-Portal (channel)
    - Byte count per request
  - Pipeline depth for processing multiple enqueue tasks concurrently
  - Arbitration between allocated TNUMs of portIDs that are ready to be dequeued
    - Available per Sub-Portal (channel) reported by the QMan
    - Round Robin arbitration between each portID
- Debug mode
  - Support for runtime debug mode

## 5.6.2 QMI Input NIA

The QMI dequeue and enqueue operations are activated using a 24-bit input Next Invoked Action (NIA). The QMI receives the NIA from other modules within the FMan. The user programs the NIA for the QMI in the module whose execution proceeds the QMI. See [Section 5.4.4.1, “NIA Register Configuration—Rx Flows”](#) for example of flows which use this NIA.

The format of the NIA and the corresponding QMI action is:

NIA Hex value	QMI action	ORR <sup>1</sup>	ENG	Action Code	31
0x54_0000	QMI Enqueue	Not part of NIA	0 1 0 1 0 1	All zero	
0x58_0000	QMI Dequeue		0 1 0 1 1 0	All zero	

<sup>1</sup> The QMI does not use the ORR and Action code fields in the NIA.

See [Section 5.4.4, “Next Invoked Action \(NIA\)”](#) for the description of the NIA concept.

## 5.6.3 QMI Memory Map and Register Definition

[Table 5-195](#) lists the offset, name, and a cross-reference to the complete description of each register.

### 5.6.3.1 Register offsets

The offsets to the memory map (the 12 least significant bits of the absolute address of the registers), are relative to the beginning of the 4 KB, “port-specific” page in the FMan memory map. See [Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map,”](#) for detailed explanation:

**Figure 5-178. FMan Hardware Port Pages Address Space**

<sup>1</sup> FMan hardware ports pages offset. See Table 5-17, “FMan Memory Map Regions.”

<sup>2</sup> See Table 5-18, “FMan Hardware Port Page Memory Map.”

<sup>3</sup> See Table 5-195, "QMI Memory Map."

<sup>4</sup> In some SoCs, the SoC address space has more bits.

[Section 5.4.2.1, “Hardware Port Pages in the FMan Memory Map,”](#) explains how the FMan memory map is organized to accommodate port partitioning.

**Table 5-195. QMI Memory Map**

Register Offset	Name	Access	Reset	Section/Page	Supported in devices
<b>QMI Common Registers</b> <b>General Configuration, Status, and Counter Registers</b>					
0x400	FMQM_GC—General Configuration Register	R/W	0xC000_	<a href="#">5.6.3.2.1/5-252</a>	All
0x404	Reserved	W	0x0000_0000		
0x408	FMQM_EIE—Error Interrupt Event Register	R/W1C	0x0000_0000	<a href="#">5.6.3.2.2/5-252</a>	All
0x40C	FMQM_EIEN—Error Interrupt Enable Register	R/W	0x0000_0000	<a href="#">5.6.3.2.3/5-253</a>	All
0x410	FMQM{EIF—Error Interrupt Force Register	W	0x0000_0000	<a href="#">5.6.3.2.4/5-254</a>	All
0x420	FMQM_GS—Global Status Register	R	0x0000_0000	<a href="#">5.6.3.2.5/5-255</a>	All
0x424	Reserved	R	0x0000_0000		
0x428	FMQM_ETFC—Enqueue Total Frame Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.6/5-256</a>	All
0x42C	FMQM_DTF—Dequeue Total Frame Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.7/5-256</a>	All
0x430	FMQM_DC0—Dequeue Counter 0	R/W	0x0000_0000	<a href="#">5.6.3.2.8/5-257</a>	All
0x440–0x468	Reserved	R/W	0x0000_0000	—	
0x480	FMQM_DTRC—Debug Trace Configuration Register	R/W	0x0000_0000	<a href="#">5.6.3.2.9/5-257</a>	

**Table 5-195. QMI Memory Map (continued)**

Register Offset <sup>1</sup>	Name	Access	Reset	Section/Page	Supported in devices
0x484	FMQM_EFD—Enqueue Frame Descriptor Dynamic	R/W	0x0000_0000	<a href="#">5.6.3.2.10/5-258</a>	
0x488–0x48F	Reserved	—	—	—	
	Debug Registers				
0x490	FMQM_DTCA1—Debug Trap Configuration A1 Register	R/W	0x0000_0000	<a href="#">5.6.3.2.11/5-259</a>	
0x494	FMQM_DTVVA1—Debug Trap Value A1 Register	R/W	0x0000_0000	<a href="#">5.6.3.2.12/5-261</a>	
0x498	FMQM_DTMA1—Debug Trap Mask A1 Register	R/W	0x0000_0000	<a href="#">5.6.3.2.13/5-261</a>	
0x49C	FMQM_DTCA—Debug Trap Counter A Register	R/W	0x0000_0000	<a href="#">5.6.3.2.14/5-261</a>	
0x4A0	FMQM_DTCA2—Debug Trap Configuration A2 Register	R/W	0x0000_0000	<a href="#">5.6.3.2.11/5-259</a>	
0x4A4	FMQM_DTVVA2—Debug Trap Value A2 Register	R/W	0x0000_0000	<a href="#">5.6.3.2.12/5-261</a>	
0x4A8	FMQM_DTMA2—Debug Trap Mask A2 Register	R/W	0x0000_0000	<a href="#">5.6.3.2.13/5-261</a>	
0x4EC–0x5FF	Reserved	—	—	—	

### QMI O/H Port Registers

0x400	FMQM_PnC—PortID <i>n</i> Configuration Register	R/W	0x0000_0000	<a href="#">5.6.3.2.15/5-262</a>	
0x404	FMQM_PnS—PortID <i>n</i> Status Register	R	0x0000_0000	<a href="#">5.6.3.2.16/5-263</a>	
0x408	FMQM_PnTS—PortID <i>n</i> Task Status Register	R	0x0000_0000	<a href="#">5.6.3.2.17/5-263</a>	
0x40C–0x41B	Reserved	—	—	—	
0x41C	FMQM_PnEN—PortID <i>n</i> Enqueue NIA Register	R/W	0x0050_00C0	<a href="#">5.6.3.2.18/5-264</a>	
0x420	FMQM_PnETFC—PortID <i>n</i> Enqueue Total Frame Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.20/5-266</a>	
0x424–0x42B	Reserved	—	—	—	
0x42C	FMQM_PnDN—PortID <i>n</i> Dequeue NIA Register	R/W	0x0050_0208	<a href="#">5.6.3.2.21/5-266</a>	
0x430	FMQM_PnDC—PortID <i>n</i> Dequeue Config Register	R/W	0x1000_FFFF	<a href="#">5.6.3.2.23/5-267</a>	
0x434	FMQM_PnDTFC—PortID <i>n</i> Dequeue Total Frame Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.24/5-269</a>	
0x438	FMQM_PnDFNOC—PortID <i>n</i> Dequeue FQID Not Override Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.25/5-270</a>	
0x43C	FMQM_PnDCC—PortID <i>n</i> Dequeue Confirm Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.26/5-270</a>	

**Table 5-195. QMI Memory Map (continued)**

Register Offset <sup>1</sup>	Name	Access	Reset	Section/Page	Supported in devices
0x440–0x7FF	Reserved	—	—	—	
<b>QMI Tx Port Registers</b>					
0x400	FMQM_PnC—PortID <i>n</i> Configuration Register	R/W	0x0000_0000	<a href="#">5.6.3.2.15/5-262</a>	
0x404	FMQM_PnS—PortID <i>n</i> Status Register	R	0x0000_0000	<a href="#">5.6.3.2.16/5-263</a>	
0x408	FMQM_PnTS—PortID <i>n</i> Task Status Register	R	0x0000_0000	<a href="#">5.6.3.2.17/5-263</a>	
0x40C–0x41B	Reserved	—	—	—	
0x41C	FMQM_PnEN—PortID <i>n</i> Enqueue NIA Register	R/W	0x0050_0nC0	<a href="#">5.6.3.2.19/5-265</a>	
0x420	FMQM_PnETFC—PortID <i>n</i> Enqueue Total Frame Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.20/5-266</a>	
0x424–0x42B	Reserved	—	—	—	
0x42C	FMQM_PnDN—PortID <i>n</i> Dequeue NIA Register	R/W	0x0050_0274	<a href="#">5.6.3.2.22/5-267</a>	
0x430	FMQM_PnDC—PortID <i>n</i> Dequeue Config Register	R/W	0x1000_FFFF	<a href="#">5.6.3.2.23/5-267</a>	
0x434	FMQM_PnDTFC—PortID <i>n</i> Dequeue Total Frame Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.24/5-269</a>	
0x438	FMQM_PnDFNOC—PortID <i>n</i> Dequeue FQID Not Override Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.25/5-270</a>	
0x43C	FMQM_PnDCC—PortID <i>n</i> Dequeue Confirm Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.26/5-270</a>	
0x458–0x7FF	Reserved	—	—	—	
<b>QMI Rx Port Registers</b>					
0x400	FMQM_PnC—PortID <i>n</i> Configuration Register.	R/W	0x0000_0000	<a href="#">5.6.3.2.15/5-262</a>	
0x404	FMQM_PnS—PortID <i>n</i> Status Register <sup>2</sup>	R	0x0000_0000	<a href="#">5.6.3.2.16/5-263</a>	
0x408	FMQM_PnTS—PortID <i>n</i> Task Status Register	R	0x0000_0000	<a href="#">5.6.3.2.17/5-263</a>	
0x40C–0x41B	Reserved	—	—	—	
0x41C	FMQM_PnEN—PortID <i>n</i> Enqueue NIA Register	R/W	0x0050_00C0	<a href="#">5.6.3.2.18/5-264</a>	
0x420	FMQM_PnETFC—PortID <i>n</i> Enqueue Total Frame Counter	R/W	0x0000_0000	<a href="#">5.6.3.2.20/5-266</a>	
0x424–0x7FF	Reserved	—	—	—	

<sup>1</sup> See [Section 5.6.3.1, “Register offsets”](#) for explanation of offset.

<sup>2</sup> In case of portID of type RxMac, only the enqueue status bits are valid.

### 5.6.3.2 QMI Register Descriptions

This section provides a detailed description of all the FMan QMI registers. Because all of the QMI registers are 32-bits-wide, only 32-bit register accesses are supported.

#### 5.6.3.2.1 QMI General Configuration Register (FMQM\_GC)

FMQM\_GC describes the general configuration of the QMI module.

This register is for internal use. In FMan\_v3, it is recommended not to modify its reset value.

Offset 0x400																Access: Read/Write				
R	0	1	2	3	4					17	18		23	24	25	26		31		
W	—	—	—	STEN	—	—	—	—	—	ENQ_THR	—	DEQ_THR	—	—	—	—	—	—		
Reset FMan_v3	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0		

Figure 5-179. QMI General Configuration Register (FMQM\_GC)

Table 5-196. Register FMQM\_GC Field Descriptions

Bits	Field	Description
0–1	—	Reserved.
2	—	Reserved
3	STEN	QMI global statistic counters enabled (not for portID counters). 0 Global statistic counters are not enabled. 1 Enables global statistic counters to be updated.
4–17	—	Reserved
18–23	ENQ_THR	Enqueue threshold. The maximum number of allowed enqueue TNUMs inside the QMI. <b>Note:</b> To avoid dead lock situation, this field must be smaller than the 64 subtracted by the sum of the BMI dequeue pipeline depth on all the hardware ports (O/H and Tx) that are enabled. $\text{ENQ\_THR} < 64 - \text{SUM}(\text{FMBM\_OFPn}[DPDE]) - \text{SUM}(\text{FMBM\_TFPn}[DPDE])$
24–25	—	Reserved
26–31	DEQ_THR	Dequeue threshold The maximum number of allowed dequeue TNUMs inside the QMI. <b>Note:</b> To avoid dead lock situation, this field must be greater than the sum of the BMI dequeue pipeline depth on all the hardware ports (O/H and Tx) that are enabled. $\text{DEQ\_THR} > \text{SUM}(\text{FMBM\_OFPn}[DPDE]) + \text{SUM}(\text{FMBM\_TFPn}[DPDE])$

#### 5.6.3.2.2 Error Interrupt Event Register (FMQM\_EIE)

The FMQM\_EIE holds run time error events of the QMI that are associated with the QMI error interrupt. Each error event status bit has a corresponding mask bit in Section 5.6.3.2.3, “Error Interrupt Enable Register (FMQM\_EIEN).

**Figure 5-180. Error Interrupt Event Register (FMQM\_EIE)**

**Table 5-197. Register FMQM EIE Field Descriptions**

Bits	Name	Description
0	DEE	<p>Double-bit ECC error has been detected on the QMI internal RAM read access</p> <p>Write '1' to clear. Writing '0' has no effect.</p> <p>0 No double-bit ECC error has been detected on the QMI internal RAM access since last clearing of this bit.</p> <p>1 Double-bit ECC error has been detected on the QMI internal RAM at least on one access. If FMQM_EIEN[DEEN] is set an error interrupt is asserted.</p> <p>This is an unrecoverable error.</p>
1	DFUPE	<p>Dequeue from unknown PortID error.</p> <p>This event occurs for a SW configuratoin error.</p> <p>Write '1' to clear. Writing '0' has no effect.</p> <p>0 The QMI does not have a TNUM that is associated with a unknown PortID.</p> <p>1 The QMI has a TNUM that is associated with unknown PortID. In case of dequeue operation this TNUM forwards by the QMI to the default queue meaning that this TNUM will bypass the dequeue flow and send an error size FD to the next module. In case of enqueue operation this TNUM will operate in normal enqueue flow.</p>
2-31	—	Reserved

### **5.6.3.2.3 Error Interrupt Enable Register (FMQM\_EIEN)**

**FMQM\_EIEN** is used to provide the software the ability to mask error events that do not require issuing an error interrupt.

Offset 0x40C Access: Read/Write

The diagram shows a memory map for offset 0x40C. The bits are numbered from 0 to 31. Bits 0, 1, and 2 are labeled as 'R' (Read) and contain the values 'DEE', 'DFUPE', and 'All zeros' respectively. Bit 31 is labeled as 'W' (Write). There is a blank space between bit 2 and bit 31.

Bit	Value
0	DEE
1	DFUPE
2	All zeros
31	—

**Figure 5-181. Error Interrupt Enable Register (FMQM\_EIEN)**

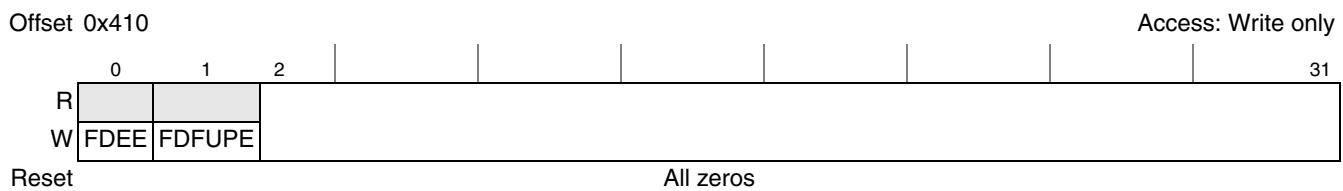
**Table 5-198. Register FMMQ\_EIEN Field Descriptions**

Bits	Name	Description
0	DEEN	Double-bit ECC error interrupt enable (for the internal RAM) 0 This double-bit ECC error interrupt is masked. 1 This double-bit ECC error interrupt is enabled. The error interrupt line is asserted when FMMQ_EIE[DEE] is '1'.
1	DFUPEN	Dequeue from unknown PortID event interrupt enable 0 The interrupt for this event is masked. 1 The interrupt for this event is enabled. The error interrupt line is asserted when FMMQ_IE[DFUPE] is '1'. This bit may be useful to detect of configuration error.
2-31	—	Reserved

#### 5.6.3.2.4 Error Interrupt Force Register (FMMQ{EIF})

This register is for internal use, or for debug purposes.

This register is write-only and writing 0 to it does not affect it. Writing 1 to this register sets the corresponding bit of the error interrupt event register. Reading from this register always provides 0x0.



**Figure 5-182. Error Interrupt Force Register (FMMQ{EIF})**

**Table 5-199. FMMQ{EIF} Field Descriptions**

Bits	Name	Description
0	FDEE	Force double ECC error Force to set the DEE bit of the FMMQ_EIE register 0 Any reading from this field returns zero. Writing zero to this field has no effect. 1 Force to set the DEE bit of the FMMQ_EIE register
1	FDFUPE	Force dequeue from unknown PortID error Force to set the DFUPE bit of the FMMQ_IE register 0 Any reading from this field returns zero. Writing zero to this field has no effect. 1 Force to set the DFUPE bit of the FMMQ_IE register.
2-31	—	Reserved

### 5.6.3.2.5 Global Status Register (FMQM\_GS)

FMQM\_GS holds the status bits of the QMI module that includes four busy types.

Offset 0x420

Access: Read only



**Figure 5-183. QMI Global Status Register (FMQM\_GS)**

**Table 5-200. Register FMQM\_GS Field Descriptions**

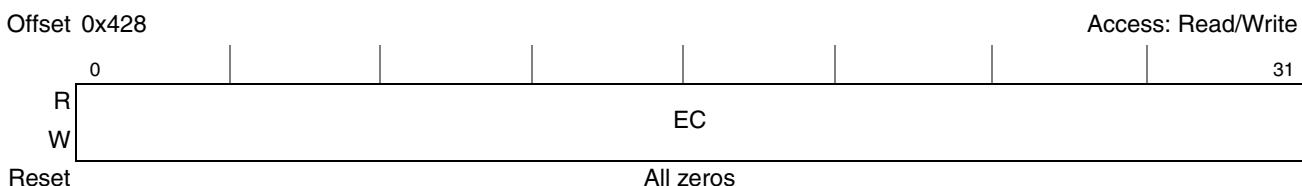
Bits	Name	Description
0	BSY_DT	QMI Busy Dequeue TNUM Indication This bit continuously reflects the QMI module dequeue TNUM processing state. 0 QMI is not busy with dequeue TNUMs. No dequeue TNUMs are being processed inside the QMI. 1 QMI is busy with dequeue TNUMs. There are dequeue TNUMs that are being processed inside the QMI. This is a global busy indication, which means at least one of the Port IDs is busy with dequeue TNUM. The software can find out which one of the Port IDs is busy using the FMQM_PnS[PBSY_DT] fields, for more details see <a href="#">Section 5.6.3.2.16, “PortID n Status Register (FMQM_PnS).”</a>
1	BSY_ET	QMI Busy Enqueue TNUM Indication This bit continuously reflects the QMI module enqueue TNUM processing state. 0 QMI is not busy with enqueue TNUMs. No enqueue TNUMs are being processed inside the QMI. 1 QMI is busy with enqueue TNUMs. There are enqueue TNUMs that are being processed inside the QMI. This is a global busy indication, which means at least one of the Port IDs is busy with enqueue TNUM. The software can find out which one of the Port IDs is busy using the FMQM_PnS[PBSY_ET] fields, for more details see <a href="#">Section 5.6.3.2.16, “PortID n Status Register (FMQM_PnS).”</a>
2	BSY_DF	QMI Busy Dequeue FD Indication. 0 QMI is not busy with dequeue FDs and not waiting for a response from the QMan. 1 QMI is busy with dequeue FDs or is waiting for a response from the QMan. This bit is set to one when a request was lunched and the corresponding response didn't arrive. This is a global dequeue FD busy indication, which means at least one of the Port IDs is busy with dequeue FD or waiting for a response from the QMan. The software can find out which one of the Port IDs is busy with dequeue FD using the FMQM_PnS[PBSY_DF] fields, for more details see <a href="#">Section 5.6.3.2.16, “PortID n Status Register (FMQM_PnS).”</a>
3	BSY_EF	QMI Busy Enqueue FD Indication. This bit continuously reflects the FDs in the enqueue part that are waiting for the QMan to pull them. 0 QMI is not busy with enqueue FDs that are waiting for the QMan to pull them. 1 QMI is busy with enqueue FDs that are waiting for the QMan to pull them. This is a global enqueue FD busy indication, which means at least one of the Port IDs is busy with enqueue FD.
4–14	—	Reserved
15	SRS	Soft Reset Sequence. For more details see <a href="#">Section 5.6.4.5, “FMan Software Reset Operation.”</a> 0 The QMI is not in soft reset sequence. 1 The QMI is in soft reset sequence.
16–29	—	Reserved

**Table 5-200. Register FQM\_GS Field Descriptions (continued)**

Bits	Name	Description
30	HNB	Halt Not Busy. 0 QMI is not in halt not busy state. 1 QMI is in halt not busy state.
31	—	Reserved

### **5.6.3.2.6 Enqueue Total Frame Counter Register (FMQM\_ETFC)**

FMQM\_ETFC counts the total number of enqueue operations the QMI performed.



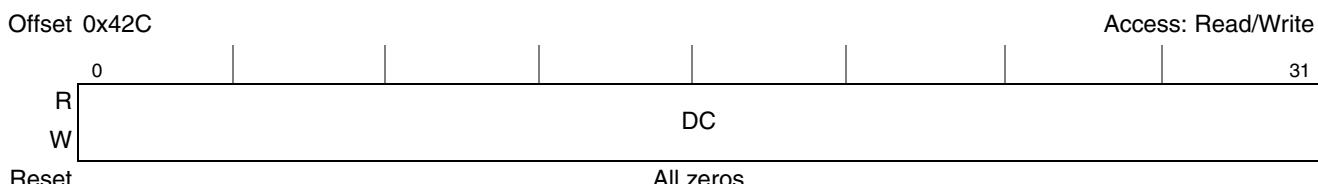
**Figure 5-184. Enqueue Total Frame Counter Register (FMQM\_ETFC)**

**Table 5-201. Register FMMQ\_M\_EFC Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	EC	Enqueue counter The total number of enqueue operations the QMI performed to the QMan

### **5.6.3.2.7 Dequeue Total Frame Counter Register (FMQM\_DTFC)**

**FMQM\_DTFC** counts the total number of FDs that dequeued from the QMan.



**Figure 5-185. Dequeue Total Frame Counter Register (FQM-DTFC)**

**Table 5-202. Register FMQM DTFC Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	DC	Dequeue counter The total number of FDs that dequeued from the QMan

### 5.6.3.2.8 Dequeue Counter 0 Register (FMQM\_DC0)



**Figure 5-186. Dequeue Counter 0 Register (FMQM\_DC0)**

**Table 5-203. Register FMQM\_DC0 Field Descriptions**

Bits	Name	Description
0–31	DC0	Dequeue counter 0  The number of times that the QMI received a NULL FD from the QMan as a response to a dequeue request (command). The NULL is and FD with address (pointer to buffer) and size (of frame) equals zero. These FDs are ignored by the FMan.

### 5.6.3.2.9 Debug Trace Configuration Register (FMQM\_DTRC)

FMQM\_DTRC configures the verbosity of the trace information provided by the QMI.



**Figure 5-187. Debug Trace Configuration Register (FMQM\_DTRC)**

**Table 5-204. Register FMQM\_DTRC Field Descriptions**

Bits	Name	Description
0–1	—	Reserved
2–3	TL_A	Trace level flow A TL_A selects the trace information that will be dumped to the debug area. 00 Trace disable 01 Minimum trace 10 Verbose trace 11 Very verbose trace
4–5	TL_B	Trace level flow B TL_B selects the trace information that will be dumped to the debug area. 00 Trace disable 01 Minimum trace 10 Verbose trace 11 Very verbose trace

**Table 5-204. Register FMMQ\_M\_DTRC Field Descriptions (continued)**

Bits	Name	Description
6–7	TL_C	Trace level flow C  TL_C selects the trace information that will be dumped to the debug area. 00 Trace disable 01 Minimum trace 10 Verbose trace 11 Very verbose trace
8–9	—	Reserved
10–11	TR_CO	Trap collaboration  To provide a larger number of traps for debug flow A, the traps that belong to debug flows B and/or C can be chained as a continuation of the traps for debug flow A. This lowers the number of active debug flows, but allows the criteria for debug flow A to be more extensive. The chained trap set would act as one trap entity with a single boolean result. The debug flows (B and/or C) that donated their traps to debug flow A are put into bypass since they would no longer have the resources to define match criteria. If debug flows B or C are not participants in the collaboration chain, then their trapping behavior will remain active and operate independently as configured. 00 No collaboration between trap registers 01 Debug flow A's traps are chained to debug flow B's traps. Debug flow B is in bypass. 10 Debug flow A's traps are chained to debug flow C's traps. Debug flow C is in bypass 11 Debug flow A's traps are chained to debug flow B's traps, which are then chained to debug flow C's traps. Debug flows B and C are in bypass.
12–31	—	Reserved

#### 5.6.3.2.10 Enqueue Frame Descriptor Dynamic Debug Register (FMMQ\_M\_EFDDD)

FMMQ\_M\_EFDDD controls the value of the dynamic debug field (DD) in the FD that is sent to the QMan on an enqueue operation. The value of the DD field depends on the debug trace level flow (A, B and C) that was received via the dispatch bus.

FMan\_v3: The debug trace level is a 3-bit field that represent three debug flows in the following order: {FlowA, FlowB, FlowC}. Every one of the eight combinations is mapped to one of the four DD options.

FMan\_v3: The options are written in the DA[0:7]. The DD field is overwritten with the DV[0:7] value only if the appropriate DOE[0:7] bit is set.

Offset 0x484 (FMQM\_EFDDD)

Access: Read/Write

R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	—	DOE0	DV0	—	DOE1	DV1	—	DOE2	DV2	—	DOE3	DV3	—	—	—	—
Reset	All zeros								All zeros							
R	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	—	DOE4	DV4	—	DOE5	DV5	—	DOE6	DV6	—	DOE7	DV7	—	—	—	—
Reset	All zeros								All zeros							

**Figure 5-188. Enqueue Frame Descriptor Dynamic Debug Register (FMQM\_EFDDD)****Table 5-205. Register FMQM\_EFDDD Field Descriptions**

Bits	Name	Description
0,4,8,12, 16,20,24 ,28	—	Reserved
1,5,9,13, 17,21,25 ,29	DOEn	Dynamic debug field override enable n FMan_v3: n=0..7). Flows A,B,C are supported.  0 When an enqueue task that has debug trace level flow - A = 0 the FD[DD] field is not overridden by the DVn field. 1 When an enqueue task that has debug trace level flow - A = 0 the FD[DD] field is overridden by the DVn field.
2–3, 6–7,10–1 1,14–15, 18–19,22 –23,26–2 7,30–31	DVn	Dynamic debug field value n FMan_v3: n=0..7). Flows A,B,C are supported.  When an enqueue task that has debug trace level flow - A = 0 and in addition DOEn bit set to one, the QMI enqueues the associated FD with a DV0 value on FD[DD] field.

### 5.6.3.2.11 Debug Trap Configuration *n* Register (FMQM\_DTC*n*)

FMQM\_DTC*n* define the event type for trigger, how it is going to be compared to the reference value, and the logic operation that combines this event trap result with the next one if it exists. There are six different trap conditions organized in pairs: DTCA1 and DCTA2 relate to flow A and DCTB1 and DCTB2 see flow B, etc.

Offset 0x490 (FMQM\_DTCA1)  
0x4A0 (FMQM\_DTC2)

Access: Read/Write

R	0	1	2	3	4	—	13	14	15	16	—	—	—	—	—	31
W	CMPOP	AND	—	—	—	FSEL	—	—	—	—	—	—	—	—	—	—
Reset	All zeros								All zeros							

**Figure 5-189. Debug Trap Configuration *n* Register (FMQM\_DTC*n*)**

**Table 5-206. FMQM\_DTC $n$  Field Descriptions**

Bits	Name	Description
0–2	CMPOP	Compare operator 000 Trap Disabled (never match) 001 Always match 010 Match if (selected field AND FMQM_DTMn[MASK]) equals FMQM_DTVn[VAL] 011 Match if (selected field AND FMQM_DTMn[MASK]) does not equal FMQM_DTVn[VAL] 100 Match if (selected field AND FMQM_DTMn[MASK]) is greater than FMQM_DTVn[VAL] 101 Match if (selected field AND FMQM_DTMn[MASK]) is less or equal to FMQM_DTVn[VAL] 110 Match if (selected field AND FMQM_DTMn[MASK]) is less than FMQM_DTVn[VAL] 111 Match if (selected field AND FMQM_DTMn[MASK]) is greater or equal to FMQM_DTVn[VAL]
3	AND	AND This field is combining trap results into a trap equation. By default it is an OR, if all next traps are disabled (or this is the last trap), they have no effect on the combined match result. The combined match evaluation is done in ascending order from one trap number n to trap number n+1, and without precedence of AND over OR. For example: TRAP1 or TRAP2 and TRAP3 will result in equation equivalent to (TRAP1    TRAP2) && TRAP3. 0 OR between this trap result and the next trap result. 1 AND between this trap result and the next trap result.
4–13	—	Reserved
14–15	FSEL	Field selection The selected field will be compared to the value at the FMQM_DTVn[VAL] combined with the mask for this value defined at the PLDBTMRx[MASK] register. Fields with fewer bits are left padded with zeros to form a 32-bit value. 00 FSEL1 - Frame Descriptor Command Field. (32-bit value, matched against FMQM_DTVn[0:31]). 01 FSEL2 - 32-bit value, matched against FMQM_DTVn[0:31] as followed: FSEL2[0:1] = Dynamic Debug field in the dequeued FD. (2-bit value, matched against FMQM_DTVn[0:1]). FSEL2[2:3] = Source of the Returned FQID. (2-bit value, matched against FMQM_DTVn[2:3]) 00 Default FQID. 01 FQ Context B. 10 Command field in the FD. 11 The associated TNUM has bypassed the dequeue block. FSEL2[4:7] = Dequeue Response Sub-Portal ID. (4-bit value, matched against FMQM_DTVn[4:7]) FSEL2[8:31] = Returned FQID (24-bit value <sup>1</sup> , matched against FMQM_DTVn[8:31]) 10 FSEL3 - 32-bit value, matched against FMQM_DTVn[0:31] as follows: FSEL3[0:1] = Dynamic Debug field in the dequeued FD. (2-bit value, matched against FMQM_DTVn[0:1]). FSEL3[2] = Dequeue Confirm. (1-bit value, matched against FMQM_DTVn[2]) 0 No Confirmation is needed. 1 Confirmation is needed. In FMan_v3: FSEL3[3:4] = Dequeue Response Frame Count - The number of frames delivered in the dequeue response. (2-bit value, matched against FMQM_DTVn[3:4]). It can be 0,1,2 or 3. FSEL3[5:7] = Reserved, tied to zero. FSEL3[8:15] = Dequeue Response Tag <sup>2</sup> . (8-bit value, matched against FMQM_DTVn[8:15]) FSEL3[16:31] = Dequeue Response Byte Count. The total number of bytes dequeued frames delivered by the QMan. (16-bit value <sup>3</sup> , matched against FMQM_DTVn[16:31]) 11 Reserved
16–31	—	Reserved

**Note:**

<sup>1</sup> The FQID is padded with zeros (in the MSB) when its width is less than 24 bits.

<sup>2</sup> Using this field, the user can map the response tag to a specific port.

<sup>3</sup> When the byte count value is greater than 0xFFFF, the QMI truncates it and writes 0xFFFF to this field.

### 5.6.3.2.12 Debug Trap Value *n* Register (FMQM\_DTV*n*)

FMQM\_DTV*n* contain the value against which the selected field, indicated by the FMQM\_DTC*n*[FSEL], is compared.

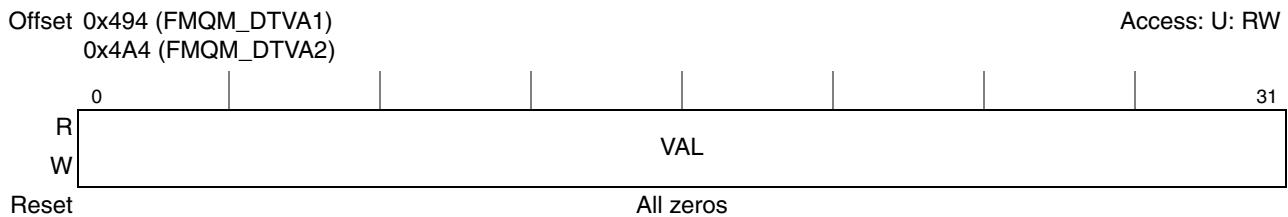


Figure 5-190. Debug Trap Value *n* Register (FMQM\_DTV*n*)

Table 5-207. FMQM\_DTV*n* Field Descriptions

Bits	Name	Description
0–31	VAL	<p>Value This value is used for comparison against the selected field at the FMQM_DTC<i>n</i> with the mask specified in FMQM_DTM<i>n</i> applied to it.</p>

### 5.6.3.2.13 Debug Trap Mask *n* Register (FMQM\_DTM*n*)

The mask value specified at the FMQM\_DTM*n* is ANDed bit-wise with the selected value for comparison, and the result is compared to the FMQM\_DTV*n*[VAL].

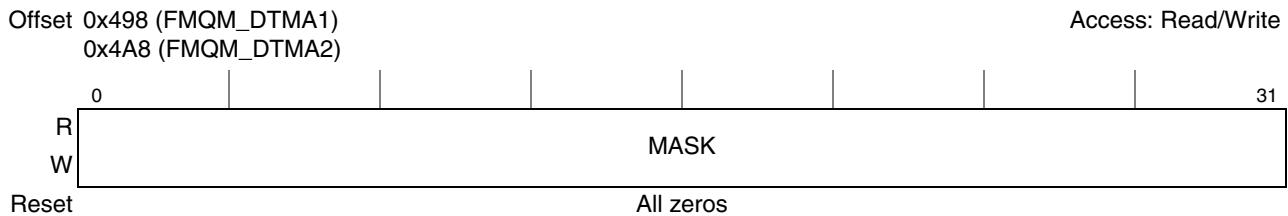


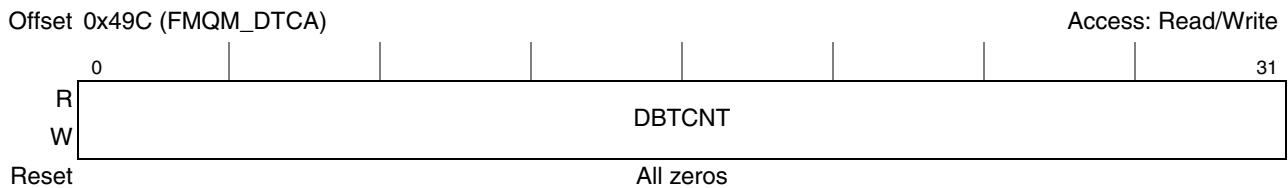
Figure 5-191. Debug Trap Mask *n* Register (FMQM\_DTM*n*)

Table 5-208. FMQM\_DTM*n* Field Descriptions

Bits	Name	Description
0–31	MASK	<p>MASK This field gets ANDed bit-wise with the selected field indicated by FSEL, then the result is compared to FMQM_DTV<i>n</i>[VAL]. Only cared bits from the selected field pass the AND function.</p>

### 5.6.3.2.14 Debug Trap Counter *n* Register (FMQM\_DTC*n*)

FMQM\_DTC*n* count the packets that filter the debug trap criteria.



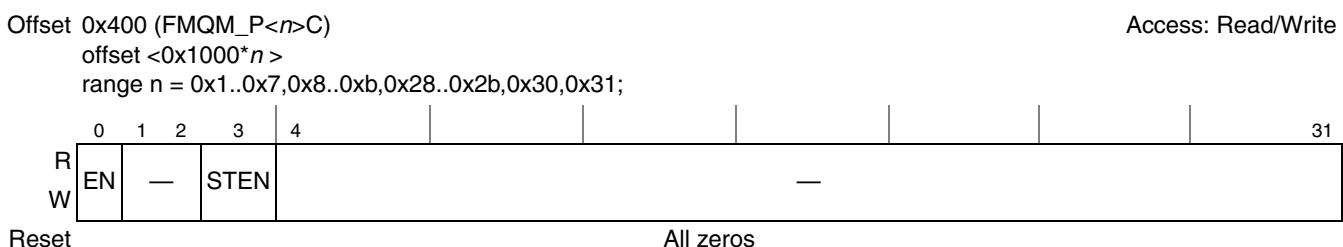
**Figure 5-192. Debug Trap Counter  $n$  Register (FMQM\_DTCn)**

**Table 5-209. FMMQ\_DTC*n* Field Descriptions**

Bits	Name	Description
0–31	DBTCNT	Debug trap event counter This is the total count of packets that filter the debug trap criteria defined by FMQM_DTCFGn, FMQM_DTVn, FMQM_DTMn registers.

### **5.6.3.2.15 PortID *n* Configuration Register (FMQM\_P*n*C)**

**FMQM\_PnC** describes the HW ports configuration of the QMI module which controls the enable state of a HW port as well as the counters associated with a HW port. There is one register for each HW port in FMan.



**Figure 5-193. PortID *n* Configuration Register (FMQM\_PnC)**

**Table 5-210. FFMQ<sub>M</sub> PnC Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0	EN	<p>Enable portID</p> <p>There is no meaning to this bit when the associated portID is a Rx port. The Rx portIDs are always enabled.</p> <p>0 The dequeue portID is not enabled. If the QMI is in the middle of operation and the EN bit is reset by the user, a graceful stop operation is started.</p> <p>1 The portID is enabled.</p>
1–2	—	Reserved
3	STEN	<p>PortID counters enabled</p> <p>0 Statistics not enabled.</p> <p>1 Enables internal counters to be updated.</p>
4–31	—	Reserved

### 5.6.3.2.16 PortID *n* Status Register (FMQM\_P*n*S)

FMQM\_P*n*S holds the status bits of the HW port that includes three busy types. There is one register for each HW port in FMan.

Offset 0x404 (FMQM_P< <i>n</i> >S) offset <0x1000*n> range n = 0x1..0x7,0x8..0xb,0x28..0x2b,0x30,0x31;				Access: Read only														
R PBSY_DT PBSY_ET PBSY_DF				—														
W				—														
Reset				All zeros														

Figure 5-194. PortID *n* Status Register (FMQM\_P*n*S)

Table 5-211. FMQM\_P*n*S Field Descriptions

Bits	Name	Description
0	PBSY_DT	PortID busy dequeue TNUM indication This bit continuously reflects the dequeue TNUM processing state for this portID. 0 The portID is not busy with dequeue TNUMs. No dequeue TNUMs are being processed with this portID. 1 The portID is busy with dequeue TNUMs. There are dequeue TNUMs that are being processed with this portID.
1	PBSY_ET	PortID busy enqueue TNUM indication This bit continuously reflects the enqueue TNUM processing state for this portID. 0 The portID is not busy with enqueue TNUMs. No enqueue TNUMs are being processed with this portID. 1 The portID is busy with enqueue TNUMs. There are enqueue TNUMs that are being processed with this portID.
2	PBSY_DF	PortID busy dequeue FD indication FMan_v3: This bit continuously reflects the FDs of the portID in the dequeue part that are prefetched and are waiting to be associated with an allocated TNUM. 0 The PortID is not waiting for a response from the QMan. 1 The portID is waiting for a response from the QMan. This bit is set to 1 when a request is lunched and the corresponding response did not arrive. FMan_v3: This bit is set also when the portID is busy with dequeue FDs.
3–31	—	Reserved

### 5.6.3.2.17 PortID *n* Task Status Register (FMQM\_P*n*TS)

FMQM\_P*n*TS provides (online) the number of TNUMs that are processed for this PortID In FMan\_v3 it also provides the number of FDs that are prefetched from the QMan and still do not have any associated

TNUM. Not all of the fields of this register are valid when the associated HW port is a receiver (Rx). There is one of register for each HW port in FMan.

This register is for internal use during debug.

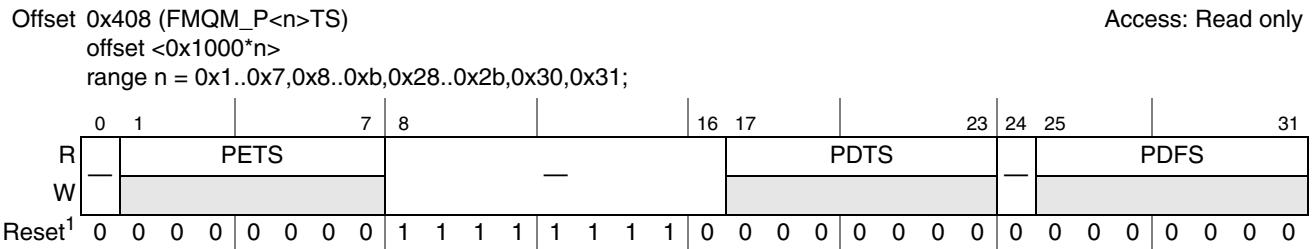


Figure 5-195. PortID n Task Status Register (FMQM\_PnTS)

<sup>1</sup> In FMan\_v3 the reset value is 0x00FF\_0000.

Table 5-212. FMQM\_PnTS Field Descriptions

Bits	Name	Description
0	—	Reserved.
1–7	PETS	PortID enqueue TNUM status.  The number of TNUMs that are inside the QMI for the enqueue part.
8–16	—	Reserved.
17–23	PDTS	PortID dequeue TNUM status  The number of TNUMs that are inside the QMI for the dequeue part. This field is not valid when the associated portID is Rx MAC, it is tied to zero.
24	—	Reserved
25–31	PDFS	PortID dequeue FD status ) The number of FD that are inside the QMI for the dequeue part. This field is not valid when the associated portID is Rx MAC, it is tied to zero.

### 5.6.3.2.18 PortID n Enqueue NIA Register (FMQM\_PnEN)

FMQM\_PnEN holds the next invoked action (see [Section 5.4.4, “Next Invoked Action \(NIA\)”](#)) to be performed next after the enqueue operation. This register is valid only when the associated HW port is a RX or O/H port. There is one of register for each HW port in FMan.

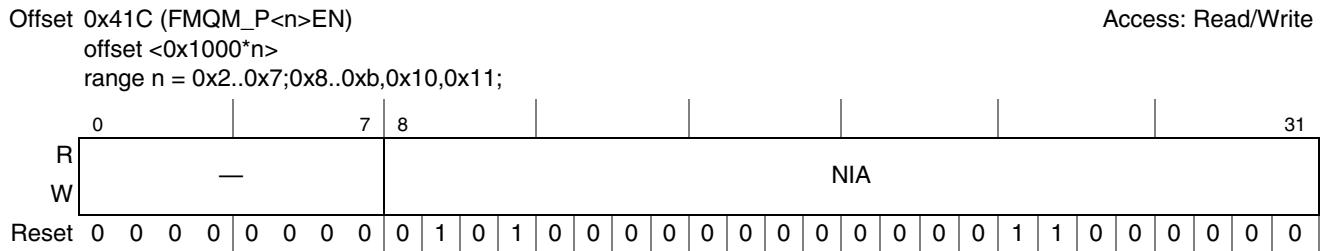


Figure 5-196. PortID n Enqueue NIA Register (FMQM\_PnEN)

**Table 5-213. FQM\_PnEN Field Descriptions**

Bits	Field	Description
0–7	—	Reserved
8–31	NIA	<p>NIA</p> <p>The next module and Action Code combination to be performed next after the enqueue operation. The default value for that NIA is as follows: the next module is the BMI and the meaning of the Action Code is “Release Internal resources (buffers and TNUM) for the termination of the task. An example of a flow using this NIA is found in “Receive (Rx) Functional Flow (Example)” in the FMan opening chapter and see <a href="#">Section 5.5.3, “BMI Input NIA”</a> for list of BMI NIAs.</p>

### **5.6.3.2.19 PortID *n* Enqueue NIA Register (FMQM\_PnEN)**

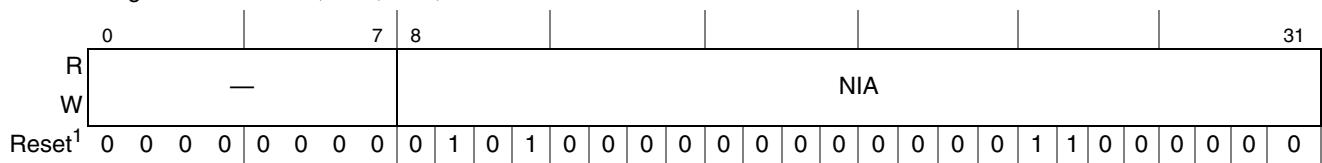
FMQM\_PnEN holds the next invoked action (see [Section 5.4.4, “Next Invoked Action \(NIA\)”,](#)) to be performed next after the enqueue operation. This register is only valid when the associated HW port is a Transmitter (Tx).

Offset 0x41C (FMQM P<n>EN)

## Access: Read/Write

offset <0x1000\*n>

range n = 0x28..0x2b,0x30,0x31;



**Figure 5-197. PortID  $n$  Enqueue NIA Register (FMQM\_PnEN)**

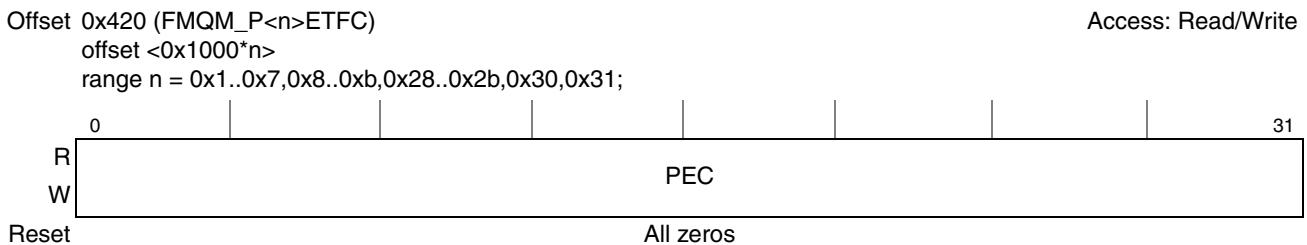
<sup>1</sup> This reset value applies to FMan\_v3.

**Table 5-214. FFMQM PnEN Field Descriptions**

Bits	Field	Description
0–7	—	Reserved
8–31	NIA	<p>NIA</p> <p>The next module and Action Code combination to be performed next after the enqueue operation. The default value for that NIA is as follows: the next module is the BMI and the meaning of the Action Code is “Release Internal resources (buffers and TNUM) for the termination of the task. An example of a flow using this NIA is found in <a href="#">Figure 5-6, “FMan Transmit (Tx) Functional Flow (Example),”</a> and see <a href="#">Section 5.5.3, “BMI Input NIA”</a> for list of BMI NIAs.</p>

### 5.6.3.2.20 PortID $n$ Enqueue Total Frame Counter Register (FMQM\_PnETFC)

**FMQM\_PnETFC** counts the number of enqueue operations performed for this portID. There is one of register for each HW port in FMan.



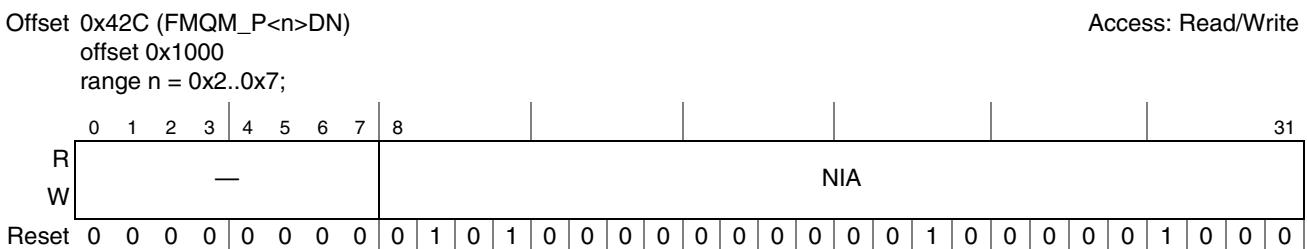
**Figure 5-198. PortID  $n$  Enqueue Total Frame Counter Register (FMQM\_PnETFC)**

**Table 5-215. FMMQ PnETFC Field Descriptions**

Bits	Name	Description
0–31	PEC	PortID enqueue counter The total number of enqueue operations performed by this portID.

### **5.6.3.2.21 PortID *n* Dequeue NIA Register (FMQM\_PnDN)**

FMQM\_PnDN holds the next invoked action (see [Section 5.4.4, “Next Invoked Action \(NIA\)”,](#)) to be performed next after the dequeue operation. This register is only valid when the associated HW port is a O/H port command.



**Figure 5-199. PortID  $n$  Dequeue NIA Register (FMQM\_PnDN)**

**Table 5-216. FFMQ M PnDN Field Descriptions**

<b>Bits</b>	<b>Field</b>	<b>Description</b>
0–7	—	Reserved
8–31	NIA	<p>NIA</p> <p>The next module and Action Code combination to be performed next after the dequeue operation. The default value for that NIA is as follows: The next module is the BMI and the meaning of the AC is “Fetch the frame header and send to the next module. See <a href="#">Section 5.5.3, “BMI Input NIA”</a> for list of BMI NIAs.</p>

### 5.6.3.2.22 PortID *n* Dequeue NIA Register (FMQM\_P*n*DN)

FMQM\_P*n*DN holds the next invoked action (see Section 5.4.4, “Next Invoked Action (NIA)”) to be performed after the dequeue operation. This register is only valid when the associated HW port is a Transmitter (Tx).

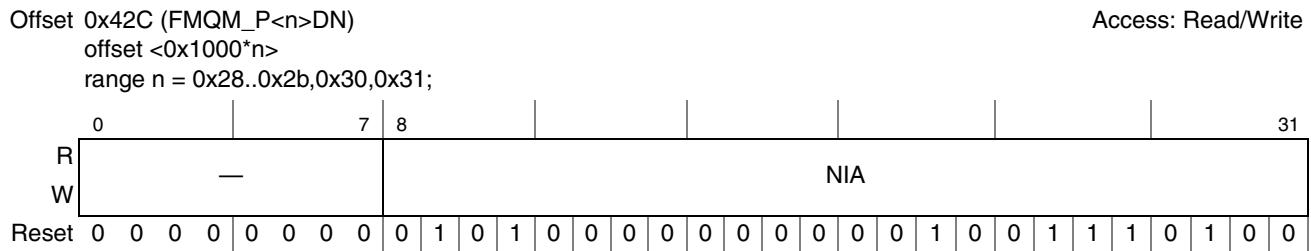


Figure 5-200. PortID *n* Dequeue NIA Register (FMQM\_P*n*DN)

Table 5-217. FMQM\_P*n*DN Field Descriptions

Bits	Field	Description
0-7	—	Reserved
8-31	NIA	<p>NIA</p> <p>The next module and Action Code combination to be performed next after the dequeue operation. The default value for that NIA is as follows: The next module is the BMI and the meaning of the AC is “Transmit frame and release external and internal buffers or enqueue to return FQID. See Section 5.5.3, “BMI Input NIA” for list of BMI NIAs.</p>

### 5.6.3.2.23 PortID *n* Dequeue Config Registers (FMQM\_P*n*DC)

FMQM\_P*n*DC determines the configuration of the dequeue action. This register also contains the priority level of this portID, and its content can be changed by software at run time. This register is not valid when the associated portID is Rx MAC, in which case any read from it returns zeros.

The recommended values allow reaching optimized performance in typical systems. Some fine tuning may be necessary in certain applications.

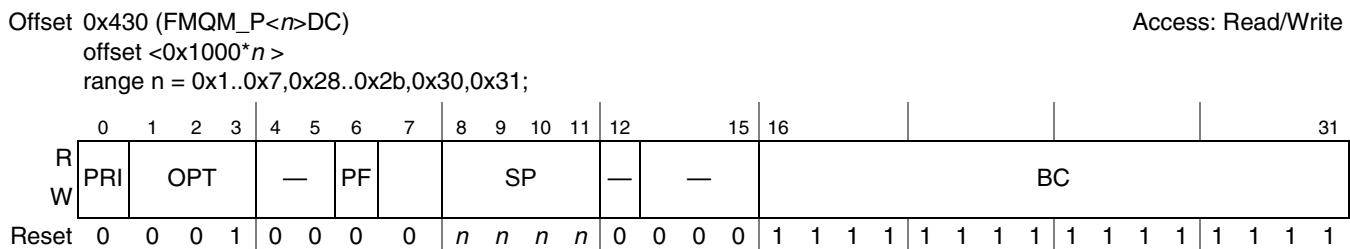


Figure 5-201. PortID *n* Dequeue Config Register (FMQM\_P*n*DC)

**Table 5-218. FMQM\_PnDC Reset Values in FMan\_v3**

Reg Name	Reset Value
FMQM_PnDC_2	0001_0000_0010_0000_1111_1111_1111_1111
FMQM_PnDC_3	0001_0000_0011_0000_1111_1111_1111_1111
FMQM_PnDC_4	0001_0000_0100_0000_1111_1111_1111_1111
FMQM_PnDC_5	0001_0000_0101_0000_1111_1111_1111_1111
FMQM_PnDC_6	0001_0000_0110_0000_1111_1111_1111_1111
FMQM_PnDC_7	0001_0000_0111_0000_1111_1111_1111_1111
FMQM_PnDC_40	0001_0000_1000_0000_1111_1111_1111_1111
FMQM_PnDC_41	0001_0000_1001_0000_1111_1111_1111_1111
FMQM_PnDC_42	0001_0000_1010_0000_1111_1111_1111_1111
FMQM_PnDC_43	0001_0000_1011_0000_1111_1111_1111_1111
FMQM_PnDC_44	0001_0000_1100_0000_1111_1111_1111_1111
FMQM_PnDC_45	0001_0000_1101_0000_1111_1111_1111_1111
FMQM_PnDC_48	0001_0000_0000_0000_1111_1111_1111_1111
FMQM_PnDC_49	0001_0000_0001_0000_1111_1111_1111_1111
else	0000_0000_0000_0000_0000_0000_0000_0000

**Table 5-219. FMQM\_PnDC Field Descriptions**

Bits	Name	Description
0	PRI	<p>Priority option select dequeue arbitration priority. This dequeue priority elevates the portID arbitration priority when multiple sub-portals are available and therefore multiple QMI ports are arbitrating for sending dequeue command to the QMan. For details see <a href="#">Section 5.6.4.2, “Dequeue Operation”</a>.</p> <p>0 Normal priority. The channel gets normal arbitration priority when requesting to send dequeue command to the QMan.</p> <p>1 High priority. The channel gets high arbitration priority when requesting to send dequeue command to the QMan.</p> <p>It is recommended to set this bit for 10Gbps ports.</p>
1-3	OPT	<p>Options for dequeue source and type.</p> <p>000 Reserved</p> <p>001 Dequeue from the Sub-Portal channel - with priority precedence, and Intra-Class Scheduling respected (type1). Recommended value.</p> <p>010 Dequeue from the Sub-Portal channel - with active FQ precedence, and Intra-Class Scheduling respected (type2).</p> <p>011 Dequeue from the Sub-Portal channel - with active FQ precedence, and override Intra-Class Scheduling (type3).</p> <p>1xx Reserved</p> <p>See <a href="#">Chapter 3, “Queue Manager (QMan)”</a> for details.</p>
4-5	—	Reserved.

Bits	Name	Description
6	PF	Pre-Fetch control. This field selects the frame pre-fetch policy. 0 No full prefetch mode. 1Full prefetch mode. This is the most efficient dequeue mode. <b>Note:</b> This bit must be cleared for Host Command port. It is recommended to set this bit.
7	—	Reserved.
8–11	SP	Sub-Portal (channel) number. This number is associated with the sub-portal available bit coming from the QMan. Every sub-portal is associated with a specific QMan channel. See <a href="#">Section 5.6.4.3, “FMan Hardware Ports, QMan sub-portals - details”</a> for a description of how the FMan connects to QMan.
12–15	—	Reserved
16–31	BC	Byte count level control  A value of all 1's (0xFFFF) in this field indicates that the byte count limit is not used, equivalent to specifying an infinite byte count. Recommended value for 1Gbps ports is 0x0400; for 10Gbps ports 0x1400. For details, see <a href="#">Section 5.6.4.2, “Dequeue Operation.”</a>

### 5.6.3.2.24 PortID *n* Dequeue Total Frame Counter Register (FMQM\_P*n*DTFC)

FMQM\_P*n*DTFC counts the total number of FDs that dequeued from the QMan for this portID. This register is not valid when the associated portID is Rx MAC, in which case any read from it returns zeros.

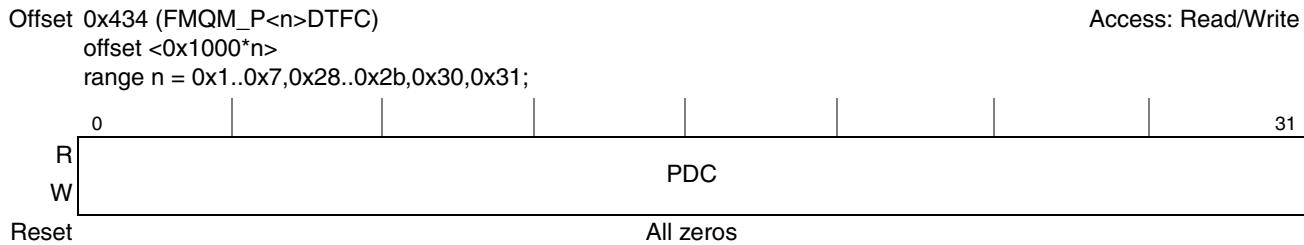


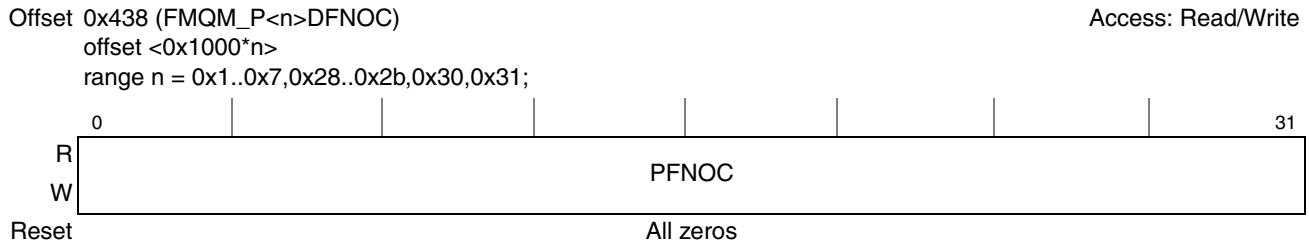
Figure 5-202. PortID *n* Dequeue Total Frame Counter Register (FMQM\_P*n*DTFC)

Table 5-220. FMQM\_P*n*DTFC Field Descriptions

Bits	Name	Description
0–31	PDC	PortID Dequeue Counter The total number of FDs that dequeued from the QMan for this portID.

### 5.6.3.2.25 PortID *n* Dequeue FQID Not Override Counter Register (FMQM\_P*n*DFNOC)

FMQM\_P*n*DFNOC counts the number of time the portID used the returned FQID that was not override by the QMan. This register is not valid when the associated portID is Rx MAC, any read from it returns zeros.



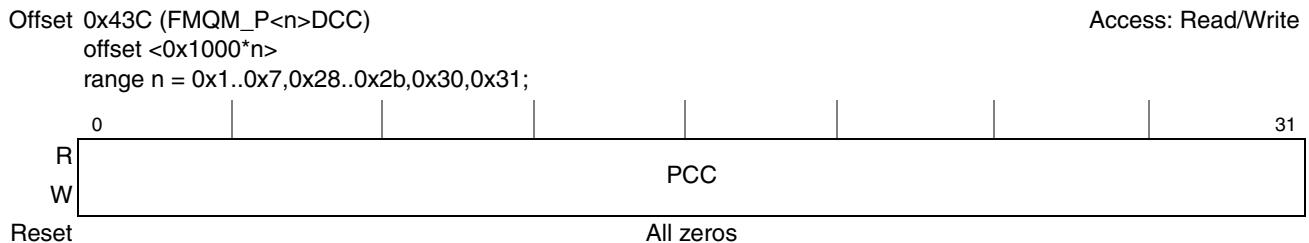
**Figure 5-203. PortID *n* Dequeue FQID Not Override Counter Register (FMQM\_P*n*DFNOC)**

**Table 5-221. FMQM\_P*n*DFNOC Field Descriptions**

Bits	Name	Description
0–31	PFNOC	PortID FQID not override counter The number of times that the portID used the default FQID and it was not override by the QMan.

### 5.6.3.2.26 PortID *n* Dequeue Confirm Counter Register (FMQM\_P*n*DCC)

FMQM\_P*n*DCC counts the number of times the PortID received a FD from the dequeue response that require confirmation. This register is not valid when the associated portID is Rx MAC, and any read from it returns zeros.



**Figure 5-204. PortID *n* Dequeue Confirm Counter Register (FMQM\_P*n*DCC)**

**Table 5-222. FMQM\_P*n*DCC Field Descriptions**

Bits	Name	Description
0–31	PCC	Confirm Counter. The number of times that the PortID received a FD from the dequeue response that require confirmation.

## 5.6.4 QMI Functional Description

### 5.6.4.1 Enqueue Operation

The QMI enqueues a frame by passing the following to the QMan:

- Frame descriptor

- Frame color
- ID of the frame queue (FQID) on which the frame should be placed

The QMI provides status bit that is set to one when there is at least one frame descriptor in the QMan clock domain that is waiting for service from the QMan. for more details see [Section 5.6.3.2.5, “Global Status Register \(FMQM\\_GS\).”](#)

#### **5.6.4.2 Dequeue Operation**

The QMI pulls frame descriptors (FDs) from queues by issuing dequeue commands to the QMan.

The QMI supports the following types of frame dequeue commands:

- Dequeue from the Sub-Portal channel—In the dequeue command, the QMI specifies which Sub-Portal channel it wishes to service. In this case, after a particular channel is chosen for service, the QMan selects a WQ within that channel, identifies the FQ at the head of that WQ, and dequeues the frame(s) at the head of that FQ.

For fine tuning purposes, it is possible to configure the following dequeue parameters:

- The maximum number of bytes which are dequeued (byte count).
- The maximum number of frame descriptors (FDs) which are dequeued (prefetch). Three modes are supported:
  - No prefetch
  - Full prefetch
  - Partial prefetch

This configuration is programmed for each portID (Tx or O/H) in FMQM\_PnDC register. For more details, see [Section 5.6.3.2.23, “PortID n Dequeue Config Registers \(FMQM\\_PnDC\).”](#) and [Section 5.6.4.4, “QMI parameters which affect performance.”](#)

#### **5.6.4.3 FMan Hardware Ports, QMan sub-portals - details**

This section describes the connection between the QMan and the FMan. It mainly clarifies some terms which are used in the QMan and FMan chapters of the reference manual and explains how to map QMan sub-portals to FMan hardware ports.

##### **NOTE**

The terms ‘channel’, ‘work queue (WQ) channel’, ‘dedicated channel’ and ‘dedicated work queue channel’ are used interchangeably across the DPAA reference manual (including the QMan chapter).

One of the QMan direct connect portals (DCPn) is used to connect the QMan to FMan (for devices with more than one FMan, each FMan is connected to a separate direct connect portal).

A DCP contains one or more sub portals (SP) to which FMan dequeue commands can be issued.

Each QMan SP is connected to a QMan ‘channel’ (see [Section 3.3.2, “Work Queues \(WQs\) and Channels”](#) within QMan).

Each SP in the QMan DCP (and thus each channel) is associated with a separate FMan Tx or O/H hardware port. The mapping of SPs to FMan hardware ports is configured in FMan QMI FMQM\_PnDC[SP] field. The SP value programmed in this field is taken from “Work Queue Channel assignments in QMan”, Channel Number column (LSBits).

QMan channels have work queues associated with them. Each channel contains up to 8 work queues.

All QMan FQs associated with FMan, are configured (in the FQD) to be enqueued by the QMan onto one of these work queues for subsequent dequeue by the FMan. This is done by programming FQD[DEST\_WQ] with the WQ ID corresponding to the SP which is associated to the desired FMan hardware port (Tx or O/H). [Section 3.3.2.4, “Work Queue Channel Assignments”](#) specifies the mapping between channels and the entities which service them. In this table the WQ IDs for SP associated with FMan are defined.

The following picture depicts the concepts described in sections above.

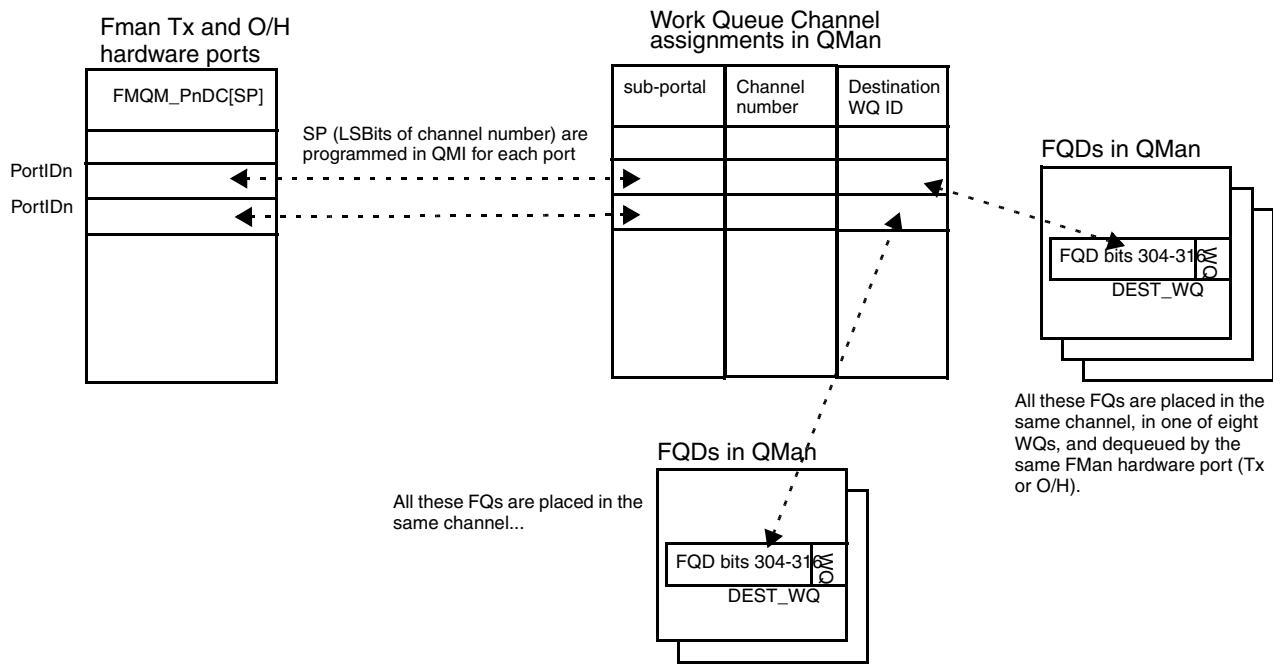


Figure 5-205. FMan hardware ports and QMan sub-portals

#### 5.6.4.4 QMI parameters which affect performance

Register FMQM\_PnDC contains a few fields which affect the dequeue performance of the QMan (thus affecting the whole FMan).

The following parameters are configurable and affect the FMan QMI operation:

- PRI - Priority. The FMan automatically monitors its state, and generates high priority requests when appropriate. Setting this bit, forces high priority dequeue requests to QMan.
- BC - Byte count. This field limits the total number of bytes the is dequeued by the QMI. This number limits the overall latency of processing, since the larger it is, the longer the latency, the shorter it is, the performance may be limited. Fine tuning may be necessary.

- PF - Prefetch control. This field configures the type prefetch mechanism in the QMI.

In addition, the OPT parameter is configurable and affects the QMan operation. See QMan chapter for details.

The operation of these parameters is described in [Section 5.6.3.2.23, “PortID n Dequeue Config Registers \(FMQM\\_PnDC\).](#)” The recommended values allow reaching optimized performance in typical systems. Some fine tuning may be necessary in certain applications.

### **5.6.4.5 FMan Software Reset Operation**

When the FMan gets a soft reset from the CPU (by setting FMFP\_RSTC[RFM]), all the QMI internal state machines and all the TNUMs and FDs that are currently inside the QMI are flushed. All the QMI registers are reset except of the statistic counters. If the user does not want to lose any FD or TNUM, the software reset operation can be initiated after the graceful stop operation, see [Section 5.6.4.6, “QMI Graceful Stop Operation.”](#) When the QMI gets a soft reset request, the SRS bit in the FMQM\_GS register is set to one, until the QMI finishes its internal soft reset sequence (until the QMan will send all the dequeue responses). For more details see [Section 5.6.3.2.5, “Global Status Register \(FMQM\\_GS\).](#)”

### **5.6.4.6 QMI Graceful Stop Operation**

If the software wants to perform graceful stop operation to a specific portID, it should activate the graceful stop sequence. The graceful stop sequence is described in [Section 5.4.9, “Graceful Stop.”](#)

### **5.6.4.7 QMI Debug Functionality**

The QMI contains an internal hardware infrastructure to support several debug options. There are two debug activities supported by the QMI, during which the QMI continues to function normally until stopped, subject to programming:

- Packet processing flow debug (the user may encounter performance degradation)
- Performance debug

#### **5.6.4.7.1 Packet Processing Flow Debug**

During packet processing, the QMI can trace a packet by writing debug information to FMan memory. This information could be written to external memory, if so configured. To trace a packet, the packet must first be marked as a packet of interest by the inclusion of non-zero values in the 2-bit DD field of the frame descriptor (FD).

#### **QMI Flow Debug**

Each TNUM can be associated with one or more debug flow options, regardless of other TNUMs. The following describes one flow but applies to all. The QMI does one of the following according to its internal debug configuration:

- Attempts to trap the frame. A successful trap leads to writing its debug context, and signals the next processing module that a packet is in a debug state during the next task dispatch. If a frame was detected by traps related to different flows in the QMI, the most verbose trace is applied. Failing to

trap leads to the termination of tracing by subsequent modules. In this case, the QMI forwards the TNUM to the next module without associating it to that flow.

- Traps in bypass state, which acts as a successful trap

When a debugged frame is enqueued to QMan it is marked with a QMan code according to the FMQM\_EFDDD register. For more details see [Section 5.6.3.2.10, “Enqueue Frame Descriptor Dynamic Debug Register \(FMQM\\_EFDDD\).”](#)

The subsequent sections describe the various debug actions that can be triggered once a packet of interest meets the stated criteria.

## **QMI Trace**

QMI trace takes place only if debug was requested during task dispatch and if the frame had been trapped (for this action trap bypass is considered as a successful trap). If these conditions are met, the QMI writes trace information into the debug section of the internal context. The trace is controlled by TL\_X (trace level for flow  $x$ ) bits in FMQM\_DTRC register. This selects the amount of trace detail that is written when trace is performed. If the frame is identified by more than one debug flow, the most verbose trace is applied.

During frame dispatch, the current debug offset is given. This value is updated according to the amount of trace data that was written to the next in the frame flow.

During debug, the QMI writes trace size and received NIA and, later, specific trace information. The final debug offset points to the next available debug place in the internal context for the next module.

[Table 5-223](#) describes the QMI trace size according to the selected verbosity level.

**Table 5-223. QMI Trace Size**

<b>Verbosity Level</b>	<b>Trace Size (Bytes)</b>
Trace disabled	0
Minimum trace	4
Verbose trace	8
Very verbose trace	20

Table 5-224 defines the information written to FMan memory by the QMI trace.

**Table 5-224. FMan QMI Trace Debug Format**

Verbosity Level		Size	Field Descriptions
Very verbose trace	Verbose trace	Minimum trace	1 byte Trace config <ul style="list-style-type: none"> <li>• The selected verbosity level (2 bits)</li> <li>• Trace size in 4-byte granularity (6 bits)</li> </ul>
			3 bytes Dispatched NIA command 0x58_0000 - Indicates the QMI received the <i>Dequeue_cmd</i> . 0xFFFFFFF - Indicates the QMI received an illegal command and it was accepted by the QMI as <i>Dequeue_cmd</i> command.
	—	4 bytes	FPM timestamp
			4 bytes portID dequeue config, from <a href="#">Section 5.6.3.2.15, “PortID n Configuration Register (FMQM_PnC)”</a> and <a href="#">Section 5.6.3.2.23, “PortID n Dequeue Config Registers (FMQM_PnDC)”</a> : <ul style="list-style-type: none"> <li>• Port priority - 1 bit.</li> <li>• Option for dequeue - 3 bit.</li> <li>• Enable port - 1bit.</li> <li>• Reserved, should be cleared - 1 bit.</li> <li>• Prefetch option - 1 bit.</li> <li>• Frame count - 1 bit.</li> <li>• Sub-Portal number - 4 bits.</li> <li>• Reserved - 1 bits.</li> <li>• Byte Count - 16 bits.</li> </ul>
		4 bytes	FD command field.
		4 bytes	Dequeue response: <ul style="list-style-type: none"> <li>• SP - 4 bits.</li> <li>• FQID source - 2 bits:<ul style="list-style-type: none"> <li>00 - The default FQID.</li> <li>01 - Context B.</li> <li>10 - FD command field.</li> <li>11 - The associated TNUM has bypassed the dequeue block.</li> </ul></li> <li>• Byte count - 16 bits. When the byte count value is more than 0xFFFF, the QMI cuts it and writes 0xFFFF to this field.</li> <li>• Frame count - 2 bits.</li> <li>• TAG (physical port) - 8 bits.</li> </ul>

## Debug Traps

The QMI has two programmable sets of debug traps for each debug flow. The debug traps for a specific flow are related to each other through boolean operations, thereby defining a combinatorial debug event. If a debug event occurs, the next module is signaled that the frame is in a debug state during task dispatch. If a trap is bypassed, the trap in question is regarded as successful.

The QMI has two programmable debug traps per flow. The total size of the trap is 8 bytes per flow and the user can select it from 12 bytes at 4byte granularity.

[Table 5-225](#) describes the trapped data according to the FSEL field. For more details see [Section 5.6.3.2.11, “Debug Trap Configuration n Register \(FMQM\\_DTCn\).”](#)

**Table 5-225. Trapped Data**

FSEL Field	Trapped Data
00	Command field of dequeued Frame Descriptor (from QMan).
01	Debug field in the dequeued FD (QMan Mark) (2 bits), Source of the Returned FQID (2 bits), • Dequeue Response Sub-Portal ID (4 bits) Returned FQID (24 bits padded with zeroes (in the MSB) when the width of the FQID is less than 24).
10	Debug field in the dequeued FD (QMan Mark) (2 bits), Dequeue Confirm (1 bit), • Dequeue Response Frame Count (2 bits), Reserved (bits). Dequeue Response Tag <sup>1</sup> - physical port (8 bits), Dequeue Response Byte Count (16 bits).
11	Reserved

<sup>1</sup> Using this field, the user can map the response tag to a specific port.

#### 5.6.4.7.2 QMI Performance Monitoring

QMI has internal counters, global counters, and portID counters. These counters are accessible via QMI memory space, and can be measured periodically by the software for QMI performance analysis.

#### QMI Global Counters

The QMI has global statistics counters that can measure the overall QMI performance. These counters are enabled by setting FMQM\_GC[STEN] (see [Section 5.6.3.2.1, “QMI General Configuration Register \(FMQM\\_GC\).”](#))

The global counters supported by the QMI are described in [Section 5.6.3.2.6, “Enqueue Total Frame Counter Register \(FMQM\\_ETFC\).”](#)

#### QMI PortID Counters

The QMI has statistics counters for each portID that can measure the QMI performance for each portID. Each specific portID counter can be enabled or disabled using FMQM\_PnC[STEN] (see [Section 5.6.3.2.15, “PortID n Configuration Register \(FMQM\\_PnC\).”](#))

## 5.6.5 QMI Initialization Sequence

Table 5-226. QMI PortID Counters

Counter Register	Function
PortID n Enqueue Total Frame Counter Register (FMQM_PnETFC)	Counts the total number of the enqueue operation which the QMI performed for a specific portID. See <a href="#">Section 5.6.3.2.20, “PortID n Enqueue Total Frame Counter Register (FMQM_PnETFC).”</a>
PortID n Dequeue Total Frame Counter Register (FMQM_PnDTFC)	Counts the total number of the FDs dequeued from the QMan for a specific portID. See <a href="#">Section 5.6.3.2.24, “PortID n Dequeue Total Frame Counter Register (FMQM_PnDTFC).”</a>
PortID n Dequeue FQID Not Override Counter Register (FMQM_PnDFNOC)	Counts the number of times the QMI used the non-overridden returned FQID for a specific portID. See <a href="#">Section 5.6.3.2.25, “PortID n Dequeue FQID Not Override Counter Register (FMQM_PnDFNOC).”</a>
PortID n Dequeue Confirm Counter Register (FMQM_PnDCC)	Counts the number of times the QMI received a FD from the dequeue response sub-link that requires confirmation for a specific portID. See <a href="#">Section 5.6.3.2.26, “PortID n Dequeue Confirm Counter Register (FMQM_PnDCC).”</a>

The QMI initialization sequence is divided into two sections: initialization of the common registers, which affects all the hardware ports (portIDs), and initialization of a specific hardware port (portID).

### 5.6.5.1 Initializing the Common QMI Registers

The QMI is ready for operation after reset and there is no need for special configuration (only for the common part).

If not using the default configuration, follow these initialization steps:

1. Enable the assertion forth error interrupt lines for the wanted events, using [Error Interrupt Enable Register \(FMQM\\_EIEN\)](#).

For FMan\_v3:

2. Configure the maximum number of allowed enqueue/dequeue TNUM inside the QMI using the [QMI General Configuration Register \(FMQM\\_GC\)](#).

### 5.6.5.2 Initializing a Specific PortID

#### 5.6.5.2.1 Initializing an Rx Port

The QMI is ready for operation after reset with no need for special configuration.

Optionally, in Rx register space, change the NIA that the QMI sends to the FPM after the enqueue operation, which is done using the [PortID n Enqueue NIA Register \(FMQM\\_PnEN\)](#).

### **5.6.5.2.2 Initializing an Tx/Host Command/Offline Parsing Port**

Perform the following initialization steps in Tx/OH register space:

1. If the FMan transmit flow requires, change the NIA that the QMI sends to the FPM after the enqueue operation, which is done by the [PortID n Enqueue NIA Register \(FMQM\\_PnEN\)](#).
2. If the FMan transmit flow requires, change the NIA that the QMI sends to the FPM after the dequeue operation, which is done using the [PortID n Enqueue NIA Register \(FMQM\\_PnEN\)](#).
3. Configure [PortID n Dequeue Config Registers \(FMQM\\_PnDC\)](#):
  - a) Mapping this port to a sub-portal
  - b) Performance optimizations options
  - c) In FMan\_v3, the default values in this register should be adequate for most systems.
4. Finally, enable this port by writing 1 to FMQM\_PnC[EN].

## 5.7 Frame Manager—Frame Processing Manager

### 5.7.1 FPM Overview

The Frame Processing Manager (FPM) distributes frame processing tasks amongst the various FMan modules (BMI, QMI, Parser, KeyGen, FMan Controller, Policer). The FPM also does the following:

- Distributes timestamp counts to the various FMan modules. This timestamp is used as the basis for various timing functions in the FMan (Policer rate, BMI rate limiting features etc.)
- Gathers all the FMan internal interrupts and drives two single interrupts (normal and error) to the SoC interrupt controller.

The purpose of the FPM module is to dispatch frame-specific tasks to the various FMan modules that are necessary for the successful processing of a frame. Each frame processing task is identified by and assigned a number, called a TNUM, that is unique within the scope of an individual frame, but which may be returned to a pool and allocated by other frames.

#### 5.7.1.1 Frames and the FPM

Frames can do the following:

- Arrive from different ports (interfaces), which are each identified by a PortID.
- Have a defined order, per port. The order is determined at an order *definition* point. This defined order is not enforced during the entire flow of processing, but it can be reestablished at an order *restoration* point. Frames then pass through this restoration point in the prescribed order.

### 5.7.2 FPM Features

- Handles multiple hardware tasks
- Distributes tasks to the FMan processing modules.
  - Buffer Manager Interface (BMI)
  - Queue Manager Interface (QMI)
  - FMan controllers
  - Parser
  - Policer
  - Keygen
- Supports order restoration per PortID
- FPM Sync command: execution of some task is stalled until all of the tasks before it have been finished (ordering mechanism).
- Provides a free task number (TNUM) pool. The tasks in the pool are not associated with any processing until they are needed.
- Generates high-precision timestamp count bus (TSB) distributed to the FMan modules
- Generates FMan soft reset request upon command
- Halt FMan for any double ECC error in FMan

- Debug features have the ability to do the following:
  - Read and Write the internal RAM
  - Read task status

### 5.7.3 Frame Manager—FPM Memory Map and Register Definition

All accesses to and from the registers must be made as 4-byte aligned, 32-bit accesses. There is no support for accesses of sizes other than 32 bits. Writes to reserved register bits must always preserve the previous value, because changing the value of reserved bits may have unintended side effects. Unless otherwise specified, the read value of unmapped registers or of reserved bits in mapped registers is not defined and must not be assumed to be 0.

The FPM memory-mapped registers are accessed by reading and writing to an address consisting of the base address (specified in CCSRBAR, as defined in Chapter 2, “Memory Map” in the applicable processor reference manual) plus the block base address, plus the offset of the specific register to be accessed. Note that all memory-mapped registers must only be accessed as 32-bit quantities. See [Section 5.3.13, “FMan Internal Memory,”](#) for details on the FMan memory map.

This table lists the offset, name, and cross-reference to the complete description of each register. The offsets to the memory map table are defined for the FPM starting at 0x000 address offset and continue to 0xFFFF (4 KB region represented by 1024 registers of 4 bytes each).

**Table 5-227. FPM Memory Map<sup>1</sup>**

Offset	Register	Access	Reset Value	Section/Page
0x004	FMFP_PRC - FPM PortID Control	W	All zeros	<a href="#">5.7.3.1/5-281</a>
0x00C	FMFP_MXD - FPM Maximum Dispatches	R/W	Device-specific	<a href="#">5.7.3.2/5-282</a>
0x010	FMFP_DIST1 - FPM Dispatch Thresholds1	R/W	Device-specific	<a href="#">5.7.3.3/5-283</a>
0x014	FMFP_DIST2 - FPM Dispatch Thresholds2	R/W	Device-specific	<a href="#">5.7.3.4/5-284</a>
0x018	FM_EPI - FMan Error Pending Interrupts	R	All zeros	<a href="#">5.7.3.5/5-284</a>
0x01C	FM_RIE - FMan Rams Interrupt Enable	R/W	All zeros	<a href="#">5.7.3.6/5-288</a>
0x020–0x02C	FMFP_FCEV0-3 - FPM FMan Controller Event 0-3	R/W	All zeros	<a href="#">5.7.3.7/5-288</a>
0x040–0x04C	FMFP_CEE0-3 - FPM FMan Controller Event Enable 0-3	R/W	All zeros	<a href="#">5.7.3.8/5-289</a>
0x060	FMFP_TSC1 - FPM Timestamp Control1	R/W	All zeros	<a href="#">5.7.3.9/5-290</a>
0x064	FMFP_TSC2 - FPM Timestamp Control2	R/W	32'h0001_0000	<a href="#">5.7.3.10/5-290</a>
0x068	FMFP_TSP - FPM Time Stamp	R	All zeros	<a href="#">5.7.3.11/5-291</a>
0x06C	FMFP_TS - FPM Time Stamp Fraction	R	All zeros	<a href="#">5.7.3.12/5-291</a>
0x070	FM_RCR - FMan Rams Control and Event	R/W	All zeros	<a href="#">5.7.3.13/5-292</a>
0x074	FMFP_EXTC - FPM External Requests Control	R/W	All zeros	<a href="#">5.7.3.14/5-293</a>
0x080–0x08C	FMFP_DRD0-3 - FPM Data_Ram Data 0-3	R/W	All zeros	<a href="#">5.7.3.15/5-293</a>
0x0BC	FM_DECCES - FM Double ECC Error Source	R/W	All zeros	<a href="#">5.7.3.16/5-294</a>

**Table 5-227. FPM Memory Map<sup>1</sup>**

Offset	Register	Access	Reset Value	Section/Page
0x0C0	FMFP_DRA - FPM Data Ram Access	R/W	All zeros	<a href="#">5.7.3.17/5-296</a>
0x0C4	FM_IP_REV_1 - FMan IP Block Revision 1	R	Device-specific	<a href="#">5.7.3.18/5-296</a>
0x0C8	FM_IP_REV_2 - FMan IP Block Revision 2	R	All zeros	<a href="#">5.7.3.19/5-297</a>
0x0CC	FM_RSTC - FMan Reset Command	W	All zeros	<a href="#">5.7.3.20/5-298</a>
0x0D0	FMFP_CLDC - FMan Controller Debug Control	R/W	All zeros	<a href="#">5.7.3.21/5-298</a>
0x0D4	FM_NPI - FMan Normal Pending Interrupts	R	All zeros	<a href="#">5.7.3.22/5-301</a>
0x0DC	FMFP_EE - FPM Event and Enable	R/W	All zeros	<a href="#">5.7.3.23/5-304</a>
0x0E0–0x0EC	FMFP_CEVO-3 - FPM CPU Event 0-3	R/W	All zeros	<a href="#">5.7.3.24/5-305</a>
0x100–0x1C4	FMFP_PS0-49 - FPM PortID Status	R	Device-specific	<a href="#">5.7.3.25/5-306</a>
0x200	FMFP_CLFABC - FMan Controller Flow AB Control	R/W	All zeros	<a href="#">5.7.3.26/5-307</a>
0x204	FMFP_CLFCC - FMan Controller Flow C Control	R/W	All zeros	<a href="#">5.7.3.27/5-308</a>
0x208	FMFP_CLFAVAL - FMan Controller Flow A Value	R/W	All zeros	<a href="#">5.7.3.28/5-309</a>
0x20C	FMFP_CLFBVAL - FMan Controller Flow B Value	R/W	All zeros	<a href="#">5.7.3.29/5-309</a>
0x210	FMFP_CLFCVAL - FMan Controller Flow C Value	R/W	All zeros	<a href="#">5.7.3.30/5-310</a>
0x214	FMFP_CLFAMSK - FMan Controller Flow A Mask	R/W	All zeros	<a href="#">5.7.3.31/5-310</a>
0x218	FMFP_CLFBMSK - FMan Controller Flow B Mask	R/W	All zeros	<a href="#">5.7.3.32/5-311</a>
0x21C	FMFP_CLFCMSK - FMan Controller Flow C Mask	R/W	All zeros	<a href="#">5.7.3.33/5-311</a>
0x220	FMFP_CLFAMC - FMan Controller Flow A Match Count	R	All zeros	<a href="#">5.7.3.34/5-312</a>
0x224	FMFP_CLFBMC - FMan Controller Flow B Match Count	R	All zeros	<a href="#">5.7.3.35/5-312</a>
0x228	FMFP_CLFCMC - FMan Controller Flow C Match Count	R	All zeros	<a href="#">5.7.3.36/5-313</a>
0x22C	FM_DECCEH - FM Double ECC Error Halt	R/W	All zeros	<a href="#">5.7.3.37/5-313</a>
0x400–0x5FC	FMFP_TS0-127 - FPM Task Status	R	All zeros	<a href="#">5.7.3.38/5-315</a>

<sup>1</sup> All missing addresses are reserved.

### 5.7.3.1 FPM PortID Control Register (FMFP\_PRC)

This register is for internal use. It should not be modified by software, unless specifically instructed as in [Section 5.12, “Frame Manager—FMan Controller.”](#)

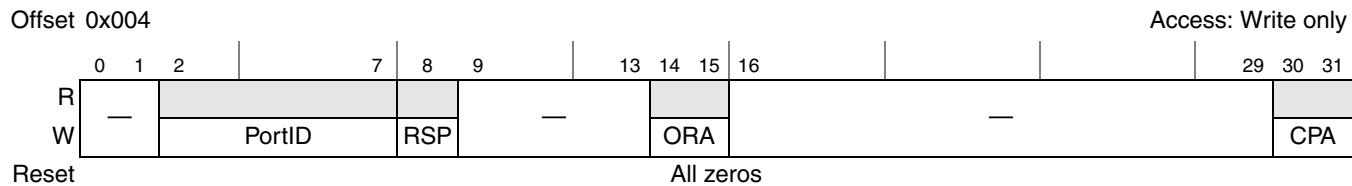
The properties of a given PortID can be changed when writing to the FPM PortID control register (FMFP\_PRC). This register is used to control which FMan controller(s) are designated to handle a given PortID.

For FMan\_v3:

For a task (not reaching an order restoration point) unless programmed by FMFP\_TNC, the PortID bit-field designates the task FMan controller association.

The reset values are as follows:

- All FMan controllers can handle all PortIDs.
- Order restoration task is treated as constrained.



**Figure 5-206. FPM PortID Control Register (FMFP\_PRC)**

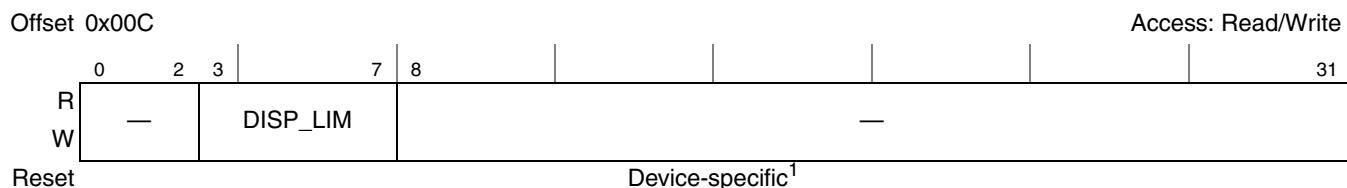
**Table 5-228. FMFP\_PRC Field Description**

Bits	Name	Description
0–1	—	Reserved
2–7	PortID	Selected PortID
8	RSP	Release stalled PortID 0 No change 1 Release stalled PortID. When a stall occurs due to a data error, one must set FMFP_TNC[CLD] for the proper task. When the stall occurs due to an incorrect NIA, the NIA must be corrected for the proper task before release.
9–13	—	Reserved
14–15	ORA	FMan controllers association for order restoration 00 No change 01 FMan controller 1 handles this PortID for order restoration. 10 FMan controller 2 handles this PortID for order restoration. 11 Reserved The SW driver defines which RISCs handles which port. This is needed for fragmentation. The FMan controller needs to keep the order of the packets. This field is not supported in FMan_v3
16–29	—	Reserved
30–31	CPA	FMan controllers PortID association  For FMan_v3: 00 No change 01 this PortID is treated as constrained 10 this PortID is treated as constrained 11 this PortID is treated as unconstrained - can be handled by all FMan controllers.

### 5.7.3.2 FPM Maximum Dispatches Register (FMFP\_MXD)

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system that requires fine tuning or debug.

This number is a global configuration for all the interfaces.



**Figure 5-207. FPM Maximum Dispatches Register (FMFP\_MXD)**

<sup>1</sup> Reset value:  
32'h0000\_0000

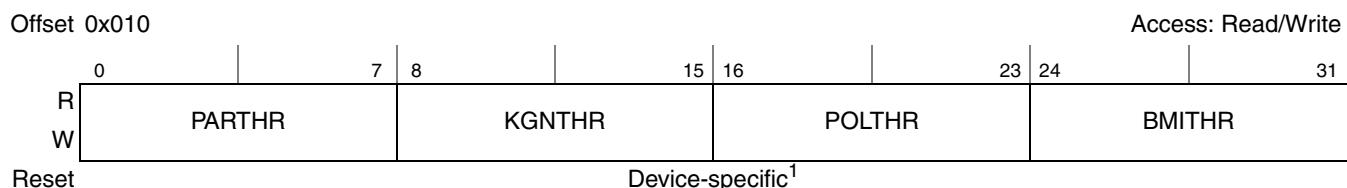
**Table 5-229. FMFP\_MXD Field Descriptions**

Bits	Name	Description
0–2	—	Reserved
3–7	DISP_LIM	Dispatch limit is the number of maximum dispatches allowed per TNUM. (A value of 0 represents no limit to the number of allowed dispatches.)
8–31	—	Reserved

### 5.7.3.3 FPM Dispatch Thresholds1 Register (FMFP\_DIST1)

The FPM dispatch thresholds 1 register (FMFP\_DIST1) determines thresholds for several dispatch queues. The threshold crossing indications might be used by an engine to elevate its priority.

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug.



**Figure 5-208. FPM Dispatch Thresholds 1 Register (FMFP\_DIST1)**

<sup>1</sup> Reset value:  
32'h10101010

**Table 5-230. FMFP\_DIST1 Field Descriptions**

Bits	Name	Description
0–7	PARTHR	Parser dispatch Threshold -FPM issues threshold crossing indication if Parser dispatch queue is above this value
8–15	KGNTTHR	KeyGen dispatch Threshold -FPM issues threshold crossing indication if KeyGen dispatch queue is above this value

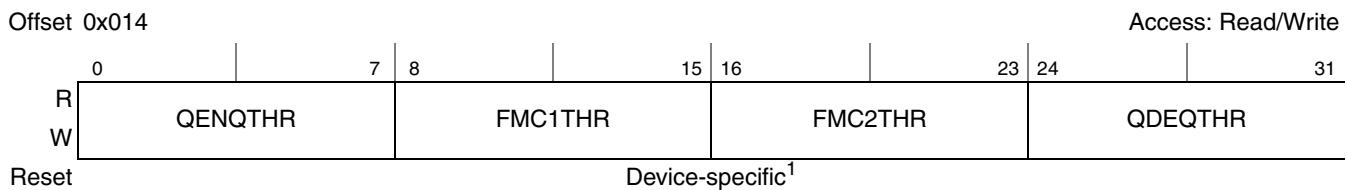
**Table 5-230. FMFP\_DIST1 Field Descriptions (continued)**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
16–23	POLTHR	Policer dispatch Threshold -FPM issues threshold crossing indication if Policer dispatch queue is above this value
24–31	BMITHR	BMI dispatch Threshold -FPM issues threshold crossing indication if BMI dispatch queue is above this value

#### 5.7.3.4 FPM Dispatch Thresholds2 Register (FMFP\_DIST2)

The FPM dispatch thresholds 2 register (FMFP\_DIST2) determines thresholds for several dispatch queues. The threshold crossing indications might be used by an engine to elevate its priority.

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug.



**Figure 5-209. FPM Dispatch Thresholds 2 Register (FMFP\_DIST2)**

<sup>1</sup> Reset value:  
32'h10101010

**Table 5-231. FMFP\_DIST2 Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–7	QENQTHR	QMI_ENQ dispatch threshold The FPM issues threshold crossing indication if QMI_ENQ dispatch queue is above this value.
8–15	FMC1THR	FMan controller 1 dispatch threshold The FPM issues threshold crossing indication if FMan controller 1 dispatch queue is above this value.
16–23	FMC2THR	FMan controller 2 dispatch threshold The FPM issues threshold crossing indication if FMan controller 2 dispatch queue is above this value.
24–31	QDEQTHR	QMI_DEQ dispatch threshold The FPM issues threshold crossing indication if QMI_DEQ dispatch queue is above this value.

### 5.7.3.5 FMan Error Pending Interrupt Register (FM\_EPI)

Each bit in the FMan error pending interrupt register (FM\_EPI) corresponds to an error interrupt source. When an error interrupt is received, a corresponding bit in FM\_EPI is set.

## **NOTE**

FM\_EPI is a read-only register, so when a pending error interrupt is handled, the user must clear the related event register bit.

Offset 0x018 Access: Read only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FPM_E RR	—	PMU_ ERR		—		BMI_ ERR	QMI_ ERR	PR SR E RR	KG N E RR	PLCR ERR	MUR_ ERR	IRM_E RR	DMA_ ERR		
W																
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	P10G1 _ERR	P1G0_ ERR	P1G1_ ERR	P1G2_ ERR	P1G3_ ERR	P1G4 _ERR	P1G5 _ERR	P1G6 _ERR	P1G7 _ERR	P10G 2_ER R	—	MC SC 5_E RR	MCSC 4_ERR	MCSC 3_ER R	MCSC 2_ERR	—
W																
Reset	All zeros															

Figure 5-210. FMan Error Pending Interrupt Register (FM\_EPI)

Table 5-232. FM\_EPI Field Descriptions

Bits	Name	Description
0	FPM_ERR	Pending FPM error interrupt 1 FPM error interrupt is asserted. 0 FPM error interrupt is negated. This bit is asserted when at least one of the events in FMFP_EE[0:2] is set and is enabled in FMFP_EE[16:18] respectively. See <a href="#">Section 5.7.3.23, “FPM Event and Enable Register (FMFP_EE)”</a> and other relevant registers.
1-2	—	Reserved
3	PMU_ERR	Pending PMU error interrupt 1 PMU error interrupt is asserted. 0 PMU error interrupt is negated.
4-7	—	Reserved
8	BMI_ERR	Pending BMI error interrupt 1 BMI error interrupt is asserted. 0 BMI error interrupt is negated. This bit is asserted when at least one of the events in FMBM_IEVR is set and is enabled in FMBM_IER. For system debug purposes it is possible to force the event with FMBM_IFR. See <a href="#">Section 5.5.4.4.4, “Interrupt Event Register (FMBM_IEVR)”</a> and other relevant registers.
9	QMI_ERR	Pending QMI error interrupt 1 QMI error interrupt is asserted. 0 QMI error interrupt is negated. This bit is asserted when at least one event in FMQM_EIE is set and enabled in FMQM_EIEN. For system debug purposes it is possible to force the event with FMQM{EIF}. See <a href="#">Section 5.6.3.2.2, “Error Interrupt Event Register (FMQM_EIE)”</a> and other relevant registers.
10	PRSR_ERR	Pending Parser error interrupt 1 Parser error interrupt is asserted. 0 Parser error interrupt is negated. This bit is asserted when the event in FMPR_PERR is set and enabled in FMPR_PERER. See <a href="#">Parser Error Register (FMPR_PERR)”</a> and other relevant registers.

**Table 5-232. FM\_EPI Field Descriptions (continued)**

Bits	Name	Description
11	KGN_ERR	<p>Pending KeyGen error interrupt            1 KeyGen error interrupt is asserted.            0 KeyGen error interrupt is negated.</p> <p>This bit is asserted when at least one event in FMKG_EER is set and enabled in FMKG_EEER. For system debug purposes it is possible to force the event with FMKG_FEER. See <a href="#">Section 5.10.3.2, “KeyGen Error Event Register (FMKG_EER)”</a> and other relevant registers. Note that FMKG_EER[KSO] bit is set if at least one bit in FMKG_SEER is set.</p>
12	PLCR_ERR	<p>Pending Policer error interrupt            1 Policer error interrupt is asserted.            0 Policer error interrupt is negated.</p> <p>This bit is asserted when at least one event in FMPL_EEVR is set and enabled in FMPL_EIER. See <a href="#">Section 5.11.4.6, “FMan Policer Error Event Register (FMPL_EEVR)”</a> and other relevant registers.</p>
13	MUR_ERR	<p>Pending FMan internal memory error interrupt            1 Multi-user RAM double ECC error interrupt is asserted.            0 Multi-user RAM double ECC error interrupt is negated.</p> <p>This bit is asserted when event in FM_RCR[MDEC] is set and enabled in FMPL_RIE. See <a href="#">Section 5.7.3.13, “FMan Rams Control and Event Register (FM_RCR)”</a>. Refer to block diagram in <a href="#">Section 5.3.4, “FMan Module Architecture.”</a></p>
14	IRM_ERR	<p>Pending FMan Controller configuration memory error interrupt            1 Instruction RAM double ECC error interrupt is asserted in FM_RCR and enabled in FM_RIE.            0 Instruction RAM double ECC error interrupt is negated.</p> <p>This bit is asserted when event in FM_RCR[IDECK] is set and enabled in FMPL_RIE. See <a href="#">Section 5.7.3.13, “FMan Rams Control and Event Register (FM_RCR)”</a>. Refer to block diagram in <a href="#">Section 5.3.4, “FMan Module Architecture.”</a></p>
15	DMA_ERR	<p>Pending DMA error interrupt            1 DMA error interrupt is asserted.            0 DMA error interrupt is negated.</p> <p>This bit is asserted when at least one of the error events in FMDM_SR is set and enabled in FMDM_MR. See <a href="#">Section 5.8.4.1, “FMan DMA Status Register (FMDM_SR)”</a> and other relevant registers.</p>
16	P10G1_ERR	<p>Pending EMAC9 error interrupt.            1 Error interrupt is asserted.            0 Error interrupt is negated.</p> <p>In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>
17	P1G0_ERR	<p>Pending EMAC1 error interrupt            1 Error interrupt is asserted.            0 Error interrupt is negated.</p> <p>In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>
18	P1G1_ERR	<p>Pending EMAC2 error interrupt            1 Error interrupt is asserted.            0 Error interrupt is negated.</p> <p>In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>

**Table 5-232. FM\_EPI Field Descriptions (continued)**

Bits	Name	Description
19	P1G2_ERR	<p>Pending EMAC3 error interrupt</p> <p>1 Error interrupt is asserted. 0 Error interrupt is negated.</p> <p>In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>
20	P1G3_ERR	<p>Pending EMAC4 error interrupt</p> <p>1 Error interrupt is asserted. 0 Error interrupt is negated.</p> <p>In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>
21	P1G4_ERR	<p>Pending EMAC5 error interrupt</p> <p>1 Error interrupt is asserted. 0 Error interrupt is negated.</p> <p>In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>
22	P1G5_ERR	<p>FMan_v3: Pending EMAC6 error interrupt</p> <p>1 Error interrupt is asserted. 0 Error interrupt is negated.</p> <p><b>Note:</b> In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>
23	P1G6_ERR <sup>1</sup>	<p>FMan_v3: Pending EMAC7 error interrupt</p> <p>1 Error interrupt is asserted. 0 Error interrupt is negated.</p> <p><b>Note:</b> In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>
24	P1G7_ERR <sup>1</sup>	<p>FMan_v3: Pending EMAC8 error interrupt</p> <p>1 Error interrupt is asserted. 0 Error interrupt is negated</p> <p><b>Note:</b> In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>
25	P10G2_ERR	<p>FMan_v3: Pending EMAC10 error interrupt</p> <p>1 Error interrupt is asserted. 0 Error interrupt is negated.</p> <p><b>Note:</b> In FMan_v3 this bit is asserted when at least one of IEVENT[TECC_ERR or RECC_ERR] (bits 22,23) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other error bits in mEMAC IEVENT register do not cause an interrupt.</p>
27	MCSC5_ER R	<p>Pending MACSEC5 error interrupt</p> <p>1 MACSEC5 error interrupt is asserted. 0 MACSEC5 error interrupt is negated</p>

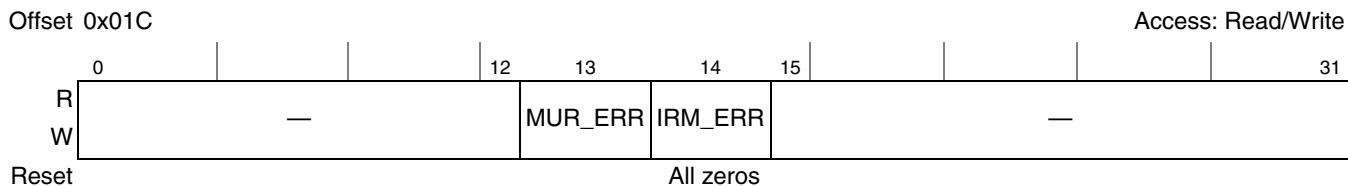
**Table 5-232. FM\_EPI Field Descriptions (continued)**

Bits	Name	Description
28	MCSC4_ER R	Pending MACSEC4 error interrupt 1 MACSEC4 error interrupt is asserted. 0 MACSEC4 error interrupt is negated
29	MCSC3_ER R	Pending MACSEC3 error interrupt 1 MACSEC3 error interrupt is asserted. 0 MACSEC3 error interrupt is negated
30	MCSC2_ER R	Pending MACSEC2 error interrupt 1 MACSEC2 error interrupt is asserted. 0 MACSEC2 error interrupt is negated
31	—	Reserved

<sup>1</sup> Refer to Table 5-15 for details on availability of specific ports on each device.

### **5.7.3.6 FMan Rams Interrupt Enable Register (FM\_RIE)**

The FMan RAMs interrupt enable register (FM\_RIE) determines masks for pending error interrupt bits defined by FM\_RCR. A non-masked bit causes an interrupt.



**Figure 5-211. FMan RAMs Interrupt Enable Register (FM\_RIE)**

**Table 5-233. FM\_RIE Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–12	—	Reserved
13	MUR_ERR	FMan internal memory error interrupt enable 1 Multi-user RAM double ECC error interrupt is enabled. 0 Multi-user RAM double ECC error interrupt is masked.
14	IRM_ERR	FMan Controller configuration memory error interrupt enable 1 Instruction RAM double ECC error interrupt is enabled. 0 Instruction RAM double ECC error interrupt is masked.
15–31	—	Reserved

### 5.7.3.7 FPM FMan Controller Event 0-3 Registers (FMFP\_FCEV)

These are FMan controller event registers. FMan controller may use them in order to assert an interrupt. See Fman Controller chapter for details on how to interpret this bit.

The CPU can clear events by writing ‘1’ to the proper bits of FMFP\_CEV0-3 registers.

A non-masked event bit, programmed by FMFP\_CEE $n$ , causes an interrupt.



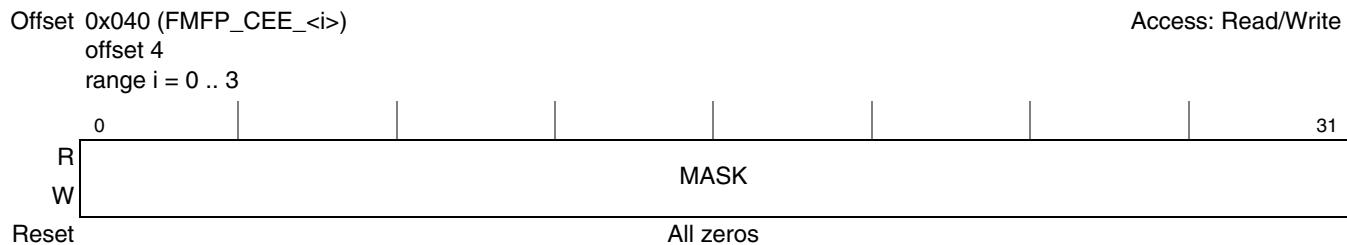
**Figure 5-212. FPM FMan Controller Event 0–3 Registers (FMFP\_FCEV $n$ )**

**Table 5-234. FMFP\_FCEV $n$  Field Descriptions**

Bits	Name	Description
0–31	EVENT	Event bits An event is set by an FMan controller when writing ‘1’. 0 No event has been detected. 1 An event has been detected.

### 5.7.3.8 FPM FMan Controller Event Enable0–3 Registers (FMFP\_CEE)

The FPM FMan controller event enable 0–3 registers (FMFP\_CEE $n$ ) determine masks for event bits defined by FMFP\_FCEV $n$ . A non-masked event bit causes an interrupt.



**Figure 5-213. FPM FMan Controller Event Enable 0–3 Registers (FMFP\_CEE $n$ )**

**Table 5-235. ReFMFP\_CEE $n$  Field Descriptions**

Bits	Name	Description
0–31	MASK	Mask bits Bit per event as defined by FMFP_FCEV $n$ . 0 Interrupt is masked. 1 Interrupt is enabled.

### 5.7.3.9 FPM TimeStamp Control1 Register (FMFP\_TSC1)

The FPM time stamp control 1 register (FMFP\_TSC1) is used to scale the FMan clock frequency to timestamp counter update rate.

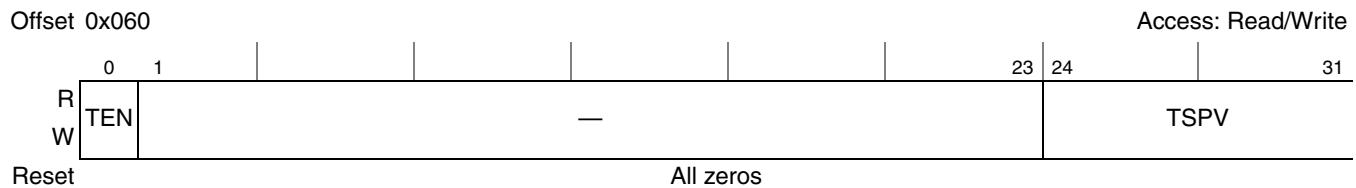


Figure 5-214. FPM TimeStamp Control 1 Register (FMFP\_TSC1)

Table 5-236. FMFP\_TSC1 Descriptions

Bits	Name	Description
0	TEN	Timestamp enable 1 Timestamp logic is enabled. 0 Timestamp logic is disabled.
1-23	—	Reserved
24-31	TSPV	Timestamp pre-scale value This field is used to pre-scale the FMan clock and set the minimum count resolution of the timestamp value. Use it in case the clock frequency pre-dividing can increase accuracy in combination with FMFP_TSC2[TSIV_FRAC] value. Clock is pre-scaled by (FMFP_TSC1[TSPV] + 1).

### 5.7.3.10 FPM TimeStamp Control2 Register (FMFP\_TSC2)

The FPM time stamp control 2 register (FMFP\_TSC2) defines the integer and fraction increments of the timestamp counter on each count. Software can use this register to achieve precision rate of the timestamp value.

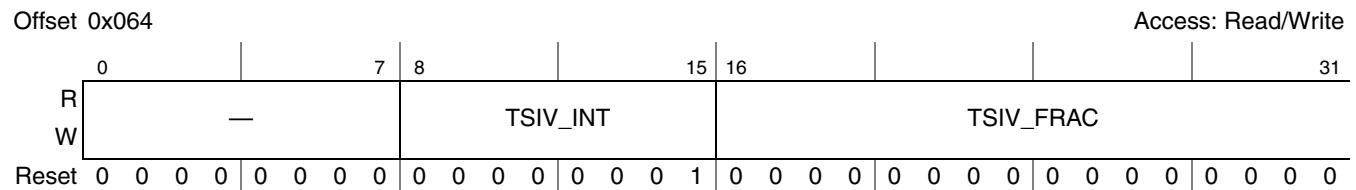


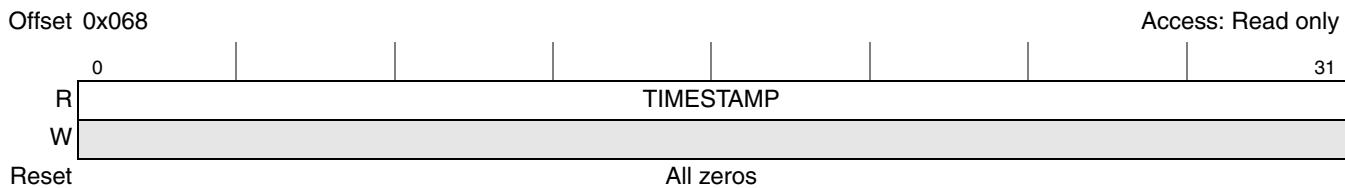
Figure 5-215. FPM TimeStamp Control 2 Register (FMFP\_TSC2)

**Table 5-237. FMFP\_TSC2 Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-15	TSIV_INT	Timestamp increment value integer bits This field holds the integer part of the number added to FMFP_TSP[TIMESTAMP] value on every pre-scaled clock. Together with FMFP_TSC2[TSIV_FRAC], it provides a 24-bit, fixed-point number to be added to the timestamp counter to achieve precision count rates. The value stored on FMFP_TSP[TIMESTAMP] represents the integer part of timestamp counter. The 16-fraction bits are accumulated separately in FMFP_TS[TS_FRACT]. Whenever fraction bits of the next count accumulate to an integer value, the FMFP_TSP[TIMESTAMP] value increments by the additional count and the fraction bits store the remaining non-integer part. Out of reset, FMFP_TSC2[TSIV_INT] = 1 and FMFP_TSC2[TSIV_FRAC] = 0, indicating increment by 1.
16-31	TSIV_FRAC	Timestamp increment value fraction bits This field holds the fraction part of the number added to FMFP_TS[TS_FRACT] value on every pre-scaled clock. Together with FMFP_TSC2[TSIV_INT], it provides precision count rate for FMFP_TSP[TIMESTAMP].

### 5.7.3.11 FPM Time Stamp Register (FMFP\_TSP)

The FPM time stamp register (FMFP\_TSP) holds the timestamp integer value.



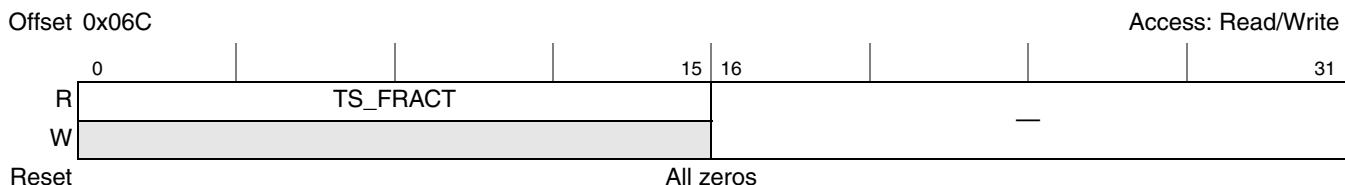
**Figure 5-216. FPM Time Stamp Register (FMFP\_TSP)**

**Table 5-238. FMFP\_TSP Field Descriptions**

Bits	Name	Description
0-31	TIMESTAMP	Time stamp counter value Holds a 32-bit read-only counter value. The counter is initialized to zero at reset or when the timestamp is disabled (FMFP_TSC1[TEN] = 0). This field represents the integer value of the timestamp.

### 5.7.3.12 FPM Time Stamp Fraction Register (FMFP\_TSF)

The FPM time stamp fraction register (FMFP\_TSF) holds the timestamp fraction value.



**Figure 5-217. FPM TimeStamp Fraction Register (FMFP\_TSF)**

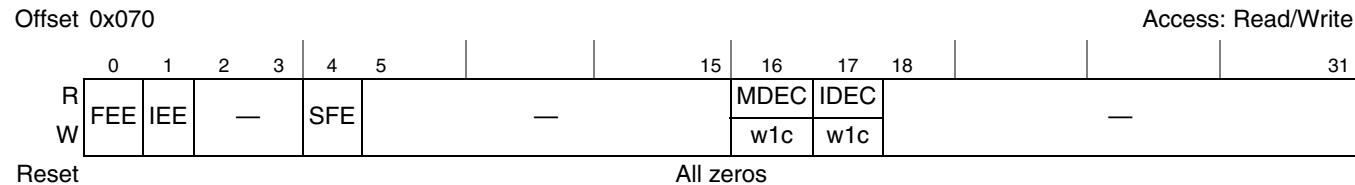
**Table 5-239. FMFP\_TS Field Descriptions**

Bits	Name	Description
0–15	TS_FRACT	Time stamp counter fraction bits value The time stamp fraction bits hold the accumulated sub-integer timestamp value. This field is initialized to zero at reset, or when the timestamp is disabled (FMFP_TSC1[TEN] = 0).
16–31	—	Reserved

### 5.7.3.13 FMan Rams Control and Event Register (FM\_RCR)

The FMan RAMs control and event register (FM\_RCR) controls the mode of operation of the multi-user RAM and the instruction RAM. It also contains these RAMs event bits.

After reset, RAMs operate in ECC disabled mode, that is, ECC is generated for writes but is not checked for reads. To switch to ECC enabled mode, all RAMs must first be initialized by writing all memory locations to put them in a known, cleared state. After the initialization, ECC mode can be enabled.



**Figure 5-218. FMan RAMs Control and Event Register (FM\_RCR)**

**Table 5-240. FM\_RCR Field Descriptions**

Bits	Name	Description
0	FEE	FMan RAMs ECC enable (except for Instruction RAM) when SFE = 0 0 ECC disabled in FMan RAMs (except for Instruction RAM) when SFE = 0 1 ECC enabled in FMan RAMs (except for Instruction RAM) when SFE = 0
1	IEE	Instruction RAM ECC enable 0 ECC disabled in the instruction RAM 1 ECC enabled in the instruction RAM
2–3	—	Reserved
4	SFE	Select FMan RAMs ECC enable source 0 FMan RAMs ECC Enable is programmed by FEE (bit 0) 1 FMan RAMs ECC Enable is controlled by COP DCFG ECCCR register
5–15	—	Reserved
16	MDEC	Double ECC error on FMan internal memory access FMan_v3: Bit is cleared by writing ‘1’ 1 Double ECC error has been detected on multi-user RAM access. 0 Double ECC error has not been detected on multi-user RAM access.

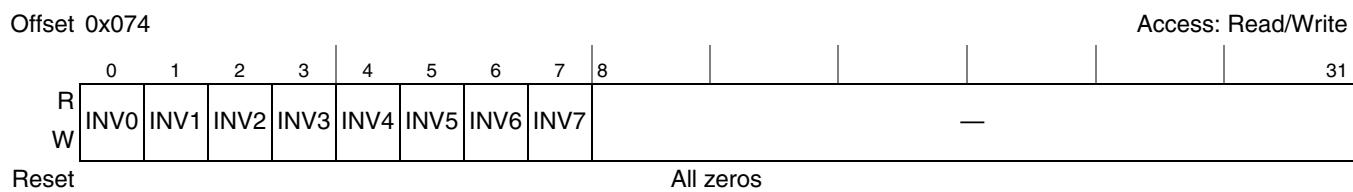
**Table 5-240. FM\_RCR Field Descriptions (continued)**

Bits	Name	Description
17	IDEC	Double ECC error on FMan Controller configuration memory access FMan_v3: Bit is cleared by writing '1' 1 Double ECC error has been detected on instruction RAM access. 0 Double ECC error has not been detected on instruction RAM access.
18–31	—	Reserved

### 5.7.3.14 FPM External Requests Control Register (FMFP\_EXTC)

This register is for internal use. It should not be programmed unless specifically instructed in the FMan controller chapter.

Software can invoke TNUMs by setting FMFP\_EXTC[INVn], and the FMan clears the bit after execution. This indicates to software that the FMan is ready to accept a new INVn command. Subsequent FMFP\_EXTC[INVn] commands can be given only after FMFP\_EXTC[INVn] is cleared.



**Figure 5-219. FPM External Requests Control Register (FMFP\_EXTC)**

**Table 5-241. FMFP\_EXTC Field Descriptions**

Bits	Name	Description
0–7	INVn	Invoke External Requestn TNUM 0 FMan is ready to receive a new invoke command 1 External Requestn is invoked - set by SW
8–31	—	Reserved

### 5.7.3.15 FPM Data Ram Data0-3 Register (FMFP\_DRDn)

The FPM data RAM data 0–3 registers (FMFP\_DRDn) are for debug purposes only.

Upon reading from this register, the appropriate TNUM data is read. When writing to these registers, data is written to the appropriate TNUM. The TNUM for which data is written/read is determined by the FMFP\_DRA value.

After each four consecutive reads/writes, the FMFP\_DRA is incremented, so subsequent reads/writes have an updated TNUM that has automatically been advanced.

Offset 0x080 (FMFP\_DRD\_<i>)  
offset 4  
range i = 0 .. 3

0																																			31
R																																			
W																																			

Reset All zeros

**Figure 5-220. FPM Data RAM Data 0–3 Registers (FMFP\_DRDn)**

**Table 5-242. FMFP\_DRDn Field Descriptions**

Bits	Name	Description
0–31	DATA	On reads, reads TNUM data On writes, writes TNUM data

### 5.7.3.16 FM Double ECC Error Source Register (FM\_DECSES)

The FM double ECC error source register (FM\_DECSES) holds the sources of non-masked (by FM\_DECCEH) double ECC errors in FM. A bit is cleared upon writing ‘1’ to it.

Offset 0x0BC (FM\_DECSES) Access: User Read/Write

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R	BMI	QMI	DMA	POL	PAR	KGN	MUR	IR	X1GR	X1GT	X2GR	X2GT	0	FPM	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
R	TS1R	TS1T	TS2R	TS2T	TS3R	TS3T	TS4R	TS4T	TS5R	TS5T	TS6R	TS6T	TS7R	TS7T	TS8R	TS8T
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 5-221. FM\_DECSES Configuration**

**Table 5-243. Register FM\_DECSES Bits Description**

Field	Description
0 BMI	non-masked BMI double ECC error
1 QMI	non-masked QMI double ECC error
2 DMA	non-masked DMA double ECC error
3 POL	non-masked Policer double ECC error
4 PAR	non-masked Parser double ECC error

**Table 5-243. Register FM\_DECCES Bits Description (continued)**

Field	Description
5 KGN	non-masked Keygen double ECC error
6 MUR	non-masked Multi-User RAM double ECC error
7 IR	non-masked Instruction RAM double ECC error
8 X1GR	non-masked EMAC9 <sup>1</sup> Rx double ECC error.
9 X1GT	non-masked EMAC9 Tx double ECC error
10 X2GR	non-masked EMAC10 Rx double ECC error
11 X2GT	non-masked EMAC10 Tx double ECC error
12 —	Reserved
13 FPM	non-masked FPM double ECC error
14–15 —	Reserved
16 TS1R	non-masked EMAC1 <sup>2</sup> Rx double ECC error
17 TS1T	non-masked EMAC1 Tx double ECC error
18 TS2R	non-masked EMAC2 Rx double ECC error
19 TS2T	non-masked EMAC2 Tx double ECC error
20 TS3R	non-masked EMAC3 Rx double ECC error
21 TS3T	non-masked EMAC3 Tx double ECC error
22 TS4R	non-masked EMAC4 Rx double ECC error
23 TS4T	non-masked EMAC4 Tx double ECC error
24 TS5R	non-masked EMAC5 Rx double ECC error
25 TS5T	non-masked EMAC5 Tx double ECC error

**Table 5-243. Register FM\_DECCES Bits Description (continued)**

Field	Description
26 TS6R	non-masked EMAC6 Rx double ECC error
27 TS6T	non-masked EMAC6 Tx double ECC error
28 TS7R	non-masked EMAC7 Rx double ECC error
29 TS7T	non-masked EMAC7 Tx double ECC error
30 TS8R	non-masked EMAC8 Rx double ECC error
31 TS8T	non-masked EMAC8 Tx double ECC error

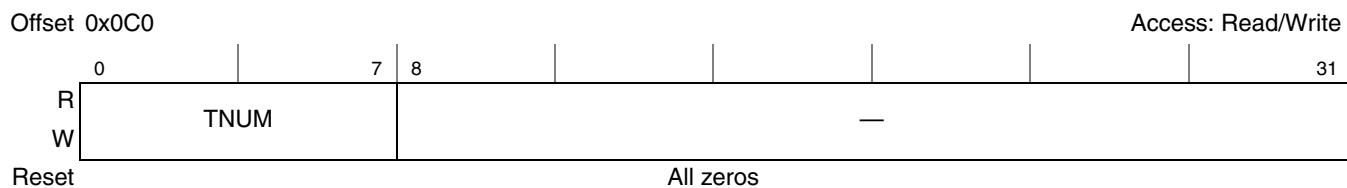
<sup>1</sup> EMAC9..10 (10/2.5/1Gbps): For FMan\_v3 refer to Chapter 6, “Multirate Ethernet Media Access Controller (mEMAC).”

<sup>2</sup> EMAC1..8 (2.5/1Gbps): For FMan\_v3 refer to Chapter 6, “Multirate Ethernet Media Access Controller (mEMAC).”

### 5.7.3.17 FPM Data RAM Access Register (FMFP\_DRA)

The FPM data RAM access register (FMFP\_DRA) enables access to data RAM for debug purposes only.

This register, when written, determines the first TNUM for which its data is read or written using FMFP\_DRDn. One TNUM data is read or written by accessing FMFP\_DRDn consecutively.



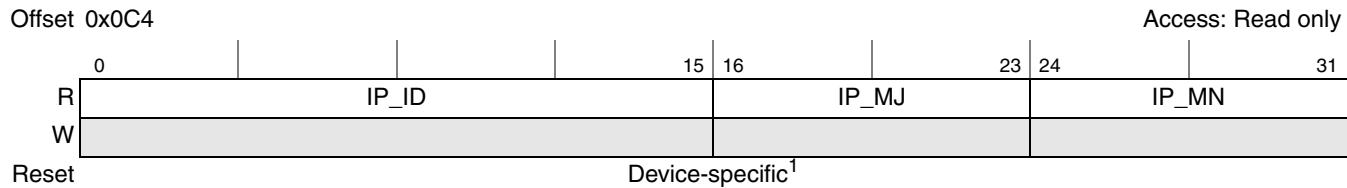
**Figure 5-222. FPM Data RAM Access Register (FMFP\_DRA)**

**Table 5-244. FMFP\_DRA Field Descriptions**

Bits	Name	Description
0–7	TNUM	<p>This is the first TNUM for which the user wants to read or write its data (it is auto-incremented) when writing this field.</p> <p>This is the current TNUM for which data is read or written when reading this field (it is auto-incremented).</p>
8–31	—	Reserved

### 5.7.3.18 FMan IP Block Revision 1 Register (FM\_IP\_REV\_1)

The FMan IP block revision 1 register (FM\_IP\_REV\_1) provides the unique IP block ID for the FMan block, as well as major and minor revision numbers.



**Figure 5-223. FMan IP Block Revision 1 Register (FM\_IP\_REV\_1)**

<sup>1</sup> Reset value:

FMan\_V3: 32'h0A070603

**Table 5-245. Register FM\_IP\_REV\_1 Bits Description**

Bits	Name	Description
0-15	IP_ID	IP Block ID
16-23	IP_MJ	Major revision.
24-31	IP_MN	Minor revision.

### 5.7.3.19 FMan IP Block Revision 2 Register (FM\_IP\_REV\_2)

The FMan IP block revision 2 register (FM\_IP\_REV\_2) provides FMan block integration and configuration options.



**Figure 5-224. FMan IP Block Revision 2 Register (FM\_IP\_REV\_2)**

**Table 5-246. FM\_IP\_REV\_2 Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-15	IP_INT	Integration options. This is set to 0x00.
16-23	IP_ERR	Errata revision level. This is set to 0x00.
24-31	IP_CFG	Configuration options. This is set to 0x00.

### 5.7.3.20 FMan Reset Command Register (FM\_RSTC)

The FMan reset command register (FM\_RSTC) is used by the CPU to issue reset commands to the FMan. Bits 1–10 are for internal use or for debug purposes and should not be set by the user.

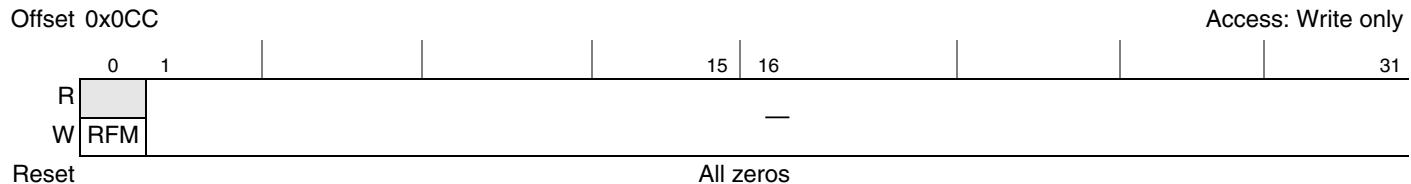


Figure 5-225. FMan Reset Command Register (FM\_RSTC)

Table 5-247. FM\_RSTC Field Descriptions

Bits	Name	Description
0	RFM	FMan soft reset command: Entire FMan is reset (including MACs) Set by CPU and cleared by the FMan. When this bit is set by CPU, the other bits should not be set. 0 No FMan reset 1 Reset FMan <b>Note:</b> The user must disable ECC detection in FMan by setting FM_RCR[FEE]=0 and FM_RCR[SFE]=0 before setting this bit. <b>Note:</b> The user must ensure that the FMan is not processing frames when soft reset is done. Graceful stop procedure is needed in order to stop the ethernet ports. <b>Note:</b> After soft reset, the user must re-configure the FMan. <b>Note:</b> The user must ensure that MAC clocks are enabled before soft reset is issued (also to MACs which are not in use). This is done at the SoC level.
1–31	—	Reserved

### 5.7.3.21 FMan Controller Debug Control Register (FMFP\_CLDC)

The FMan controller debug control register (FMFP\_CLDC) configures the FMan controller debug for the 3 debug flows (A,B,C) by defining the level of detail to be inserted into the debug trace stream. It also configures TNUM prefetch. See Table 5-249 for a description of the contents of trace messages initiated from FMan Controller.



Figure 5-226. FPM Controller Debug Control Register (FMFP\_CLDC)

**Table 5-248. FMFP\_CLDC Field Descriptions**

Bits	Name	Description
0–1	—	Reserved
2–3	TL_A	<p>Trace level flow A Selects the level of detail to be inserted into the debug area of the frame's internal context for debug flow A. See <a href="#">Table 5-249</a> for trace record description.</p> <ul style="list-style-type: none"> <li>00 Trace disable</li> <li>01 Minimum trace—Only the first 8 bytes of the trace record are saved.</li> <li>10 Verbose trace—The trace size is dependent on the number of table lookups. The 4-byte entry “table base – lookup result” is repeated according to the number of lookup tables.</li> <li>11 Reserved</li> </ul>
4–5	TL_B	<p>Trace level flow B Selects the level of detail to be inserted into the debug area of the frame's internal context for debug flow B. See <a href="#">Table 5-249</a> for trace record description.</p> <ul style="list-style-type: none"> <li>00 Trace disable</li> <li>01 Minimum trace—Only the first 8 bytes of the trace record are saved.</li> <li>10 Verbose trace—The trace size is dependent on the number of table lookups. The 4-byte entry “table base – lookup result” is repeated according to the number of lookup tables.</li> <li>11 Reserved</li> </ul>
6–7	TL_C	<p>Trace level flow C Selects the level of detail to be inserted into the debug area of the frame's internal context for debug flow C. See <a href="#">Table 5-249</a> for trace record description.</p> <ul style="list-style-type: none"> <li>00 Trace disable</li> <li>01 Minimum trace—Only the first 8 bytes of the trace record are saved.</li> <li>10 Verbose trace—The trace size is dependent on the number of table lookups. The 4-byte entry “table base – lookup result” is repeated according to the number of lookup tables.</li> <li>11 Reserved</li> </ul>
8–11	TR_CO	<p>Trap collaboration</p> <p>To provide a larger number of traps for debug flow A, the traps that belong to debug flows B and/or C can be chained as a continuation of the traps for debug flow A. This lowers the number of active debug flows, but allows the criteria for debug flow A to be more extensive. The chained trap set would act as one trap entity with a single boolean result. The debug flows (B and/or C) that donated their traps to debug flow A are put into bypass since they would no longer have the resources to define match criteria. If debug flows B or C are not participants in the collaboration chain, then their trapping behavior will remain active and operate independently as configured.</p> <ul style="list-style-type: none"> <li>0000 No collaboration between trap registers</li> <li>0001 Debug flow A's traps chained to debug flow B's traps. Debug flow B is in bypass.</li> <li>0010 Debug flow A's traps chained to debug flow C's traps. Debug flow C is in bypass</li> <li>0011 Debug flow A's traps chained to debug flow B's traps, which are then chained to debug flow C's traps. Debug flows B and C are in bypass</li> </ul>
12–15	—	Reserved
16	TPFE	<p>TNUM prefetch enable This field enables prefetch operation for allocation of the new TNUM from the FPM free pool by the FMan Controller.</p> <ul style="list-style-type: none"> <li>0 TNUM prefetch is disabled. TNUM allocation by the FMan Controller is done only when needed, but is time consuming.</li> <li>1 TNUM prefetch is enabled. A free TNUM is always allocated for FMan Controller usage.</li> </ul>
17–19	—	Reserved

**Table 5-248. FMFP\_CLDC Field Descriptions (continued)**

Bits	Name	Description
20	CDD	1 - enable dynamic clock disable for dma (applicable for FMan_v3)
21	CDK	1 - enable dynamic clock disable for keygen (applicable for FMan_v3)
22	CDP	1 - enable dynamic clock disable for parser (applicable for FMan_v3)
23–27	—	Reserved
28	FD3	FMan Controller 3 Disable (applicable for FMan_v3) 0 FMan Controller 3 is enabled. 1 FMan Controller 3 is disabled.
29	FD2	FMan Controller 2 Disable (applicable for FMan_v3) 0 FMan Controller 2 is enabled. 1 FMan Controller 2 is disabled.
30	FD1	FMan Controller 1 Disable (applicable for FMan_v3) 0 FMan Controller 1 is enabled. 1 FMan Controller 1 is disabled.
31	FD0	FMan Controller 0 Disable (applicable for FMan_v3) 0 FMan Controller 0 is enabled. 1 FMan Controller 0 is disabled.

This table describes the FMan Controller debug trace format.

**Table 5-249. FMan FPM Debug Trace Format**

Verbosity Level		Size		Field Description
Verbose trace	Minimum trace	1 byte		Trace configuration <ul style="list-style-type: none"><li>The selected verbosity level (2 bits)</li><li>Trace size in 4-byte granularity (6 bits)</li></ul>
		3 bytes		Dispatched NIA
		1 byte		TNUM
		1 bytes		FMan controller number
		2 bytes		Reserved
	4 <sup>1</sup> bytes	4 bytes		Timestamp
		3 bytes		Table base pointer
		1 byte		Lookup result The number of the first table entry that satisfies the lookup criteria

**Note:**

<sup>1</sup> The 4-byte table lookup trace entry is repeated according to the number of table lookups.

### 5.7.3.22 FMan Normal Pending Interrupt Register (FM\_NPI)

Each bit in the FMan normal pending interrupt register (FM\_NPI) corresponds to an interrupt source. When an interrupt is received, a corresponding bit in FM\_NPI is set. FM\_NPI is a read-only register, so when a pending interrupt is handled, the user must clear the related event register bit.

Offset 0x0D4																Access: Read only				
R		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
W		—	—	PRSR	PMU	PLCR	M C4	MC5	TMR	—	—	P10G2	P10G1	P1G0	P1G1	P1G2	P1G3			
Reset																				
R		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			
W		FCEVT0	FCEVT1	FCEVT2	FCEVT3	MC3	—	—	—	—	P1G4	P1G5	—	P1G6	—	P1G7	—			
Reset																				
All zeros																				

Figure 5-227. FMan Normal Pending Interrupt Register (FM\_NPI)

Table 5-250. FM\_NPI Field Descriptions

Bits	Name	Description
0	—	Reserved
1	—	Reserved
2	PRSR	Pending parser interrupt. 1 Parser interrupt is asserted 0 Parser interrupt is negated. This bit is asserted when at least one event in FMPR_PEVR is set and enabled in FMPR_PEVER. See <a href="#">Section 5.9.3.1.3, “Interrupt Registers”</a> and other relevant registers.
3	PMU	Pending PMU interrupt. 1 PMU interrupt is asserted 0 PMU interrupt is negated.
4	PLCR	Pending policer interrupt 1 Policier interrupt is asserted. 0 Policier interrupt is negated. This bit is asserted when at least one event in FMPL_EVR is set and enabled in FMPL_IER. For system debug purposes it is possible to force the event with FMPL_IFR. See <a href="#">Section 5.11.4.3, “FMan Policier Event Register (FMPL_EVR)”</a> and other relevant registers.
5	MC4	Pending MACSEC4 interrupt 1 MACSEC4 interrupt is asserted 0 MACSEC4 interrupt is negated
6	MC5	Pending MACSEC5 interrupt 1 MACSEC5 interrupt is asserted 0 MACSEC5 interrupt is negated

**Table 5-250. FM\_NPI Field Descriptions (continued)**

Bits	Name	Description
7	TMR	Pending 1588 timer interrupt 1 1588 timer interrupt is asserted. 0 1588 timer interrupt is negated. This bit is asserted when at least one bit in the IEEE1588 Timer module TMR_TEVENT register is set and enabled in TMR_TEMASK.
8–9	—	Reserved
10	P10G2	Pending EMAC10 normal interrupt 1 Interrupt is asserted. 0 Interrupt is negated. <b>Note:</b> This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.
11	P10G1	Pending EMAC9 normal interrupt <b>Note:</b> This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.
12	P1G0	In FMan_v3: Pending EMAC1 interrupt 1 Interrupt is asserted. 0 Interrupt is negated. This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.
13	P1G1	In FMan_v3: Pending EMAC2 interrupt 1 Interrupt is asserted. 0 Interrupt is negated. This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.
14	P1G2	In FMan_v3: Pending EMAC3 interrupt 1 Interrupt is asserted. 0 Interrupt is negated This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.
15	P1G3	In FMan_v3: Pending EMAC4 interrupt 1 Interrupt is asserted. 0 Interrupt is negated This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.

**Table 5-250. FM\_NPI Field Descriptions (continued)**

Bits	Name	Description
16	FCEVT0	Pending FMan controller event0 interrupt 1 FMan controller event0 interrupt is asserted. 0 FMan controller event0 interrupt is negated. See <a href="#">Section 5.12, “Frame Manager—FMan Controller”</a> for more details.
17	FCEVT1	Pending FMan controller event1 interrupt 1 FMan controller event1 interrupt is asserted. 0 FMan controller event1 interrupt is negated. See <a href="#">Section 5.12, “Frame Manager—FMan Controller”</a> for more details.
18	FCEVT2	Pending FMan controller event2 interrupt 1 FMan controller event2 interrupt is asserted. 0 FMan controller event2 interrupt is negated. See <a href="#">Section 5.12, “Frame Manager—FMan Controller”</a> for more details.
19	FCEVT3	Pending FMan controller event3 interrupt 1 FMan controller event3 interrupt is asserted. 0 FMan controller event3 interrupt is negated. See <a href="#">Section 5.12, “Frame Manager—FMan Controller”</a> for more details.
20	MC3	Pending MACSEC3 interrupt 1 MACSEC3 interrupt is asserted 0 MACSEC3 interrupt is negated
21	MC2	Pending MACSEC2 interrupt 1 MACSEC2 interrupt is asserted 0 MACSEC2 interrupt is negated
22	MC1	Pending MACSEC1 interrupt 1 MACSEC1 interrupt is asserted 0 MACSEC1 interrupt is negated. This bit is asserted if at least one bit in MACSEC_EVR register is set and enabled in MACSEC_EVER register.
23–24	—	Reserved
25	P1G4T	In FMan_v3: Pending EMAC5 interrupt 1 Interrupt is asserted. 0 Interrupt is negated. <b>Note:</b> This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.
26	P1G5	In FMan_v3: Pending EMAC6 interrupt 1 Interrupt is asserted. 0 Interrupt is negated. <b>Note:</b> This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.
27	—	Reserved

**Table 5-250. FM\_NPI Field Descriptions (continued)**

Bits	Name	Description
28	P1G6 <sup>1</sup>	In FMan_v3: Pending EMAC7 interrupt 1 Interrupt is asserted. 0 Interrupt is negated. <b>Note:</b> This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.
29	—	Reserved
30	P1G7 <sup>1</sup>	In FMan_v3: Pending EMAC8 interrupt 1 Interrupt is asserted. 0 Interrupt is negated. <b>Note:</b> This bit is asserted when event in mEMAC IEVENT[MGI] (bit 17) is set and enabled in the corresponding bit in mEMAC IMASK register. Note that other normal events in the IEVENT register are always masked, and do not generate an interrupt.

<sup>1</sup> Refer to [Table 5-15](#) for details on availability of specific ports on each device.

### 5.7.3.23 FPM Event and Enable Register (FMFP\_EE)

The FPM event and enable register (FMFP\_EE) holds FPM events and masks that are associated with FPM error interrupt. A non-masked bit causes an interrupt.

FMFP\_EE also holds FPM configuration for various error cases.

Offset 0x0DC																		Access: Mixed																		
R	0	1	2	3	7	8	9	10	11	14	15	16	17	18	19	—	27	28	29	30	31	DECC	STL	SECC	—	—	—	0	DECC_EN	STL_EN	SECC_EN	—	EHM	—	CER	DER
W	w1c	w1c	w1c	—	—	—	—	—	—	RFM	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—						

Reset

All zeros

**Figure 5-228. FPM Event and Enable Register (FMFP\_EE)**

**Table 5-251. FMFP\_EE Field Descriptions**

Bits	Name	Description
0	DECC	Double ECC error on FPM RAM access. Write ‘1’ to clear the event. 1 Double ECC error has been detected on FPM RAM access. 0 Double ECC error has not been detected on FPM RAM access.
1	STL	Stall of task(s)/PortID(s) on FPM. Write ‘1’ to clear the event. 1 Stall of task(s)/PortID(s) on FPM because of errors 0 No stall of task(s)/PortID(s) on FPM
2	SECC	Single ECC error on FPM RAM access. Write ‘1’ to clear the event. 1 Single ECC error has been detected on FPM RAM access. 0 Single ECC error has not been detected on FPM RAM access.
3–14	—	Reserved
15	RFM	Release FMan after halt FMan command or after unrecoverable ECC error (bit is self-cleared) 0 Normal operation 1 Release FMan

**Table 5-251. FMFP\_EE Field Descriptions (continued)**

Bits	Name	Description
16	DECC_EN	Double ECC error (on FPM RAM access) interrupt mask. 1 Double ECC error interrupt is enabled. 0 Double ECC error interrupt is masked.
17	STL_EN	Stall of task(s)/PortID(s) on FPM interrupt mask 1 Stall of task(s)/PortID(s) on FPM interrupt is enabled. 0 Stall of task(s)/PortID(s) on FPM interrupt is masked.
18	SECC_EN	Single ECC error (on FPM RAM access) interrupt mask 1 Single ECC error interrupt is enabled. 0 Single ECC error interrupt is masked.
19–27	—	Reserved
28	EHM	External halt mask 0 FMan is halted upon external halt activation. 1 FMan is not halted upon external halt activation.
29	—	Reserved
30	CER	Catastrophic error behavior 0 PortID is stalled (only reset can release it). 1 Only erroneous task is stalled.
31	DER	DMA error behavior 0 DMA error is treated as a catastrophic error. 1 DMA error is just reported.

### 5.7.3.24 FPM CPU Event 0–3 Registers (FMFP\_CEvn)

This register contains interrupt events that are asserted by the FMan controller. See [Section 5.12, “Frame Manager—FMan Controller,”](#) for more details on each bit.

FMan controllers assert an interrupt by writing to FMFP\_FCEVn registers. CPU can clear events by writing ‘1’ to the proper bits of FMFP\_CEvn registers. A non-masked event bit programmed by FMFP\_CEEn causes an interrupt.



**Figure 5-229. FPM CPU Event 0–3 Registers (FMFP\_CEvn)**

**Table 5-252. FMFP\_CEVn Field Descriptions**

Bits	Name	Description
0–31	EVENT	Event bits. An event is cleared by CPU when writing ‘1’. 0 No event has been detected. 1 An event has been detected.

### 5.7.3.25 FPM PortID Status 0–49 Registers (FMFP\_PSn)

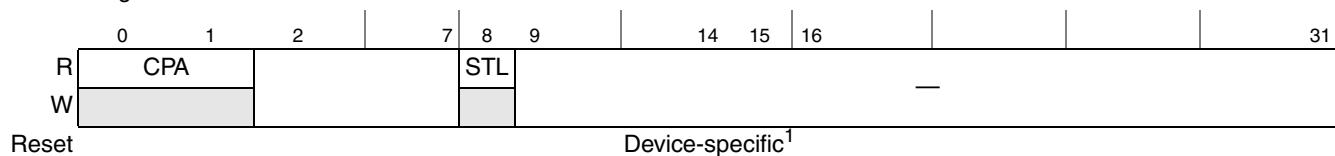
The FPM PortID status 0–49 registers (`FMFP_PSn`) hold PortIDs statuses. A specific PortID status is determined by the address. PortID  $x$  status is found at offset  $0x100 + x*4$ .

Offset 0x100 (FMFP\_PS\_<i>)

Access: Read only

offset 4

```
range i = 0 .. 49
```



**Figure 5-230. FPM PortID Status 0–49 Registers (FMFP\_PSn)**

## 1 Reset value:

FMan\_v3: 32'h8000\_0000

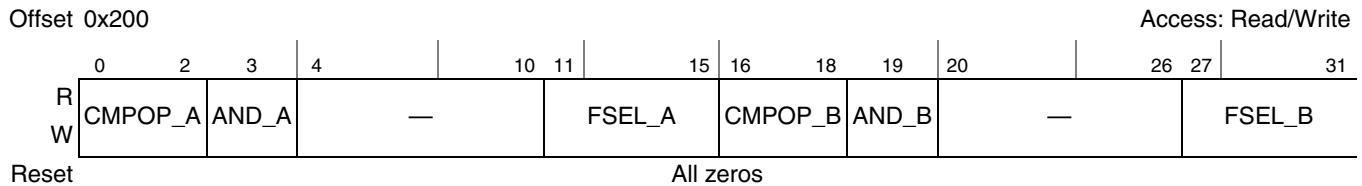
**Table 5-253. FMFP\_PSn Field Descriptions**

Bits	Name	Description
0–1	CPA	FMan Controller PortID association. In FMan_v3: 00 this PortID is treated as constrained. 10 this PortID is treated as unconstrained.
2–7	—	Reserved
8	STL	Stalled PortID (the stall indication is not set as an effect of breakpoint; it is also not set as an effect of whole FMan halt) 0 PortID works normally. 1 PortID is stalled.
9–31	—	Reserved

### 5.7.3.26 FMan Controller Flow AB Control Register (FMFP\_CLFABC)

This register is used for debug purposes.

The FMan controller flow AB control register (FMFP\_CLFABC) defines how the selected value is compared to the reference value. It also defines the logic operation that combines the trap result with the next debug trap, if it exists.



**Figure 5-231. FMan Controller Flow AB Control Register (FMFP\_CLFABC)**

**Table 5-254. FMFP\_CLFABC Field Descriptions**

Bits	Name	Description
0–2	CMPOP_A	Compare operator flow A 000 Trap disabled (never match) 001 Always match (referred as trap bypass) 010 Match if (selected field & MASK) equals value 011 Match if (selected field & MASK) does not equal value 100 Match if (selected field & MASK) is greater than value 101 Match if (selected field & MASK) is less or equal to value 110 Match if (selected field & MASK) is less than value 111 Match if (selected field & MASK) is greater or equal to value
3	AND_A	AND flow A This field combines trap results into a trap equation. By default, it is an OR. If all subsequent traps are disabled (or if this is the last trap), they have no effect on the combined match result. The combined match evaluation is done in ascending order from trap number $n$ to trap number $n + 1$ , and without precedence of AND over OR. For example: TRAP1 or TRAP2 and TRAP3 result in an equation equivalent to (TRAP1    TRAP2) && TRAP3. 0 OR between this trap result and the next trap result 1 AND between this trap result and the next trap result
11–15	FSEL_A	Field selection for flow A The selected field combination is compared to the value from appropriate FMFP_CLFxVAL register. 00000 Start color[0:1], start Port ID[0:5], start NIA[0:23] 00001 Start color[0:1], start Port ID[0:5], start FQID[0:23] 00010 Reserved[0:7], start Tx Command[0:7], start Operation Mode Bits[0:6], start VSPE, start ASPID[0:7] 00100 End color[0:1], end Port ID[0:5], end NIA[0:23] 00101 End color[0:1], end Port ID[0:5], end FQID[0:23] 00110 Reserved[0:6], coarse classification termination action descriptor debug mark, end Tx Command[0:7], end Operation Mode Bits[0:6], end VSPE, end ASPID[0:7] rest Reserved The start value means value of the field before the FMan Controller started the classification. The end value means value of the field after the FMan Controller finished the classification. Note that for reserved fields the appropriate bits in FMFP_CLFxMSK should be zero.

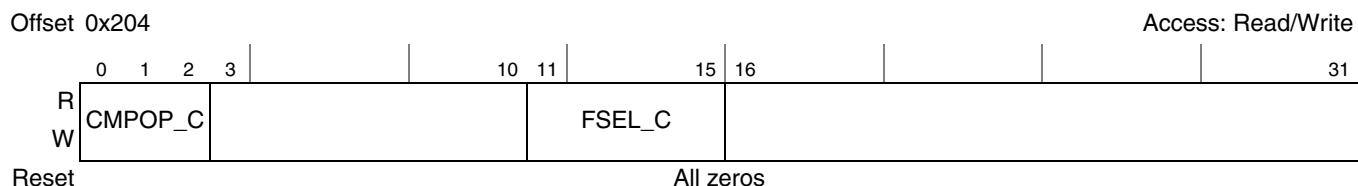
**Table 5-254. FMFP\_CLFABC Field Descriptions (continued)**

Bits	Name	Description
16–18	CMPOP_B	Compare operator flow B 000 Trap disabled (never match) 001 Always match (referred as trap bypass) 010 Match if (selected field & MASK) equals value 011 Match if (selected field & MASK) does not equal value 100 Match if (selected field & MASK) is greater than value 101 Match if (selected field & MASK) is less or equal to value 110 Match if (selected field & MASK) is less than value 111 Match if (selected field & MASK) is greater or equal to value
19	AND_B	AND flow B This field combines trap results into a trap equation. By default, it is an OR. If all subsequent traps are disabled (or if this is the last trap), they have no effect on the combined match result. The combined match evaluation is done in ascending order from trap number $n$ to trap number $n + 1$ , and without precedence of AND over OR. For example: TRAP1 or TRAP2 and TRAP3 result in an equation equivalent to (TRAP1    TRAP2) && TRAP3. 0 OR between this trap result and the next trap result 1 AND between this trap result and the next trap result
27–31	FSEL_B	Field selection flow B The selected field is compared to the value; the decoding is the same as FMFP_CLFABC[FSEL_A].

### **5.7.3.27 FMan Controller Flow C Control Register (FMFP\_CLFCC)**

This register is used for debug purposes.

The FMan controller flow C control register (FMFP\_CLFCC) defines how the selected value is compared to the reference value. It also defines the logic operation that combines the trap result with the next debug trap, if it exists.



**Figure 5-232. FMan Controller Flow C Control Register (FMFP\_CLFCC)**

**Table 5-255. FMFP\_CLFCC Field Descriptions**

Bits	Name	Description
0–2	CMPOP_C	<p>Compare operator flow C</p> <ul style="list-style-type: none"> <li>000 Trap Disabled (never match)</li> <li>001 Always match (referred as trap bypass)</li> <li>010 Match if (selected field &amp; MASK) equals value</li> <li>011 Match if (selected field &amp; MASK) does not equal value</li> <li>100 Match if (selected field &amp; MASK) is greater than value</li> <li>101 Match if (selected field &amp; MASK) is less or equal to value</li> <li>110 Match if (selected field &amp; MASK) is less than value</li> <li>111 Match if (selected field &amp; MASK) is greater or equal to value</li> </ul>
11–15	FSEL_C	<p>Field selection flow C</p> <p>The selected field is compared to the value; the decoding is same as FMFP_CLFABC[FSEL_A].</p>

### 5.7.3.28 FMan Controller Flow A Value Register (FMFP\_CLFAVAL)

This register is used for debug purposes.

The FMan controller flow A value register (FMFP\_CLFAVAL) contains the value against which the selected field at the FMFP\_CLFABC is compared to.

Offset 0x208 Access: Read/Write

The diagram shows a memory map for offset 0x208. It features a horizontal line representing memory space from bit 0 to bit 31. On the left, there is a box labeled 'R' above 'W' (Read/Write) and 'Reset'. To the right of this box is the label 'VAL'. Above the line, bit 0 is labeled '0' and bit 31 is labeled '31'. Vertical tick marks are present at bits 1, 5, 9, 13, 17, 21, 25, and 29.

**Figure 5-233. FMan Controller Flow A Value Register (FMFP\_CLFAVAL)**

**Table 5-256. FMFP\_CLFAVAL Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	VAL	Comparison value

### 5.7.3.29 FMan Controller Flow B Value Register (FMFP\_CLFBVAL)

This register is used for debug purposes.

The FMan controller flow B value register (FMFP\_CLFBVAL) contains the values against which the selected fields identified in FMFP\_CLFABC are compared to.

Offset 0x20C Access: Read/Write

The diagram shows a memory map for offset 0x20C. It features a horizontal line representing memory space from address 0 to 31. On the left, the label "Offset 0x20C" is at the top, and "Access: Read/Write" is at the bottom. In the center, there is a box labeled "VAL". To the left of "VAL", the letters "R" and "W" are stacked vertically. Above the line, the number "0" is at the start and "31" is at the end. There are seven vertical tick marks along the line, each aligned with a small vertical line segment above it.

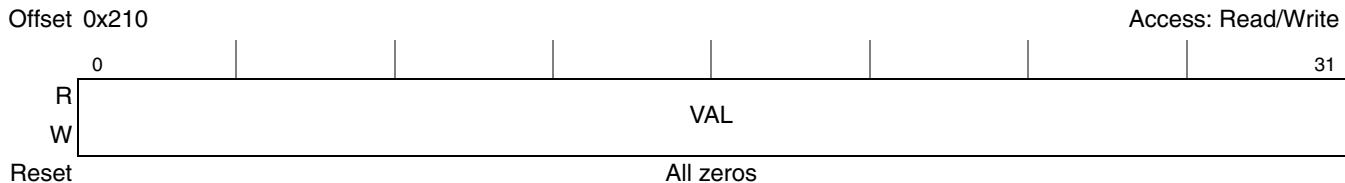
**Figure 5-234. FMan Controller Flow B Value Register (FMFP\_CLFBVAL)**

**Table 5-257. FMFP\_CLFBVAL Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	VAL	Comparison value

### **5.7.3.30 FMan Controller Flow C Value Register (FMFP\_CLFCVAL)**

The FMan controller flow C value register (FMFP\_CLFCVAL) contains the value against which the selected field at the FMFP\_CLFCC is compared to.



**Figure 5-235. FMan Controller Flow C Value Register (FMFP\_CLFCVAL)**

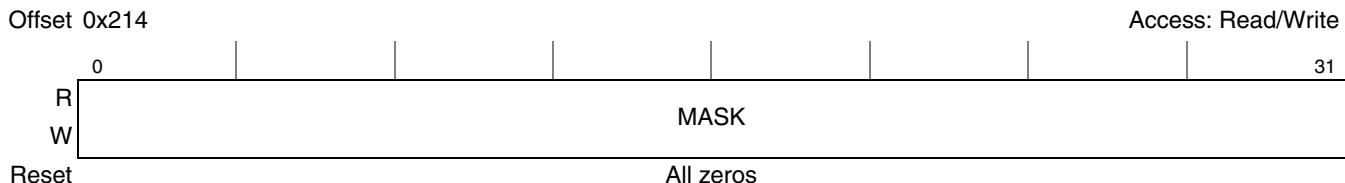
**Table 5-258. FMFP\_CLFCVAL Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	VAL	Comparison value

### 5.7.3.31 FMan Controller Flow A Mask Register (FMFP\_CLFAMSK)

This register is used for debug purposes.

The FMan controller flow A mask register (FMFP\_CLFAMSK) contains the mask for debug flow A. This mask is ANDed bit-wise with the selected value for comparison, and the result is compared to the debug value register, which contains the comparison value.



**Figure 5-236. FMan Controller Flow A Mask Register (FMFP\_CLFAMSK)**

### Table 5-259. FMFP CLFAMSK Field Descriptions

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	MASK	Mask for debug flow A

### **5.7.3.32 FMan Controller Flow B Mask Register (FMFP\_CLFBMSK)**

This register is used for debug purposes.

The FMan controller flow B mask register (FMFP\_CLFBMSK) contains the mask for debug flow B. This mask is ANDed bit-wise with the selected value for comparison, and the result is compared to the debug value register, which contains the comparison value.



**Figure 5-237. FMan Controller Flow B Mask Register (FMFP\_CLFBMSK)**

**Table 5-260. FMFP\_CLFBMSK Field Descriptions**

Bits	Name	Description
0–31	MASK	Mask for debug flow B

### **5.7.3.33 FMan Controller Flow C Mask Register (FMFP\_CLFCMSK)**

This register is used for debug purposes.

The FMan controller flow C mask register (FMFP\_CLFCMSK) contains the mask for debug flow B. This mask is ANDed bit-wise with the selected value for comparison, and the result is compared to the debug value register, which contains the comparison value.



**Figure 5-238. FMan Controller Flow C Mask Register (FMFP\_CLFCMSK)**

## **Table 5-261. FMFP\_CLFCMSK Field Descriptions**

Bits	Name	Description
0–31	MASK	Mask for debug flow C

### 5.7.3.34 FMan Controller Flow A Match Count Register (FMFP\_CLFAMC)

This register is used for debug purposes.

The FMan controller flow A match count register (FMFP\_CLFAMC) counts the number of match events on flow A debug traps.

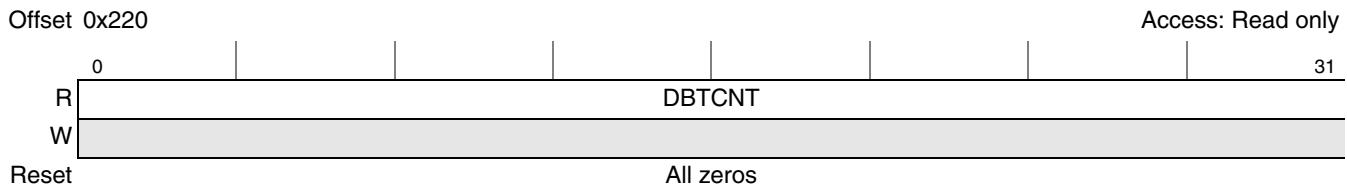


Figure 5-239. FMan Controller Flow A Match Count Register (FMFP\_CLFAMC)

Table 5-262. FMFP\_CLFAMC Field Descriptions

Bits	Name	Description
0–31	DBTCNT	Debug trap event match counter for flow A

### 5.7.3.35 FMan Controller Flow B Match Count Register (FMFP\_CLFBMC)

This register is used for debug purposes.

The FMan controller flow B match count register (FMFP\_CLFBMC) counts the number of match events on flow B debug traps.

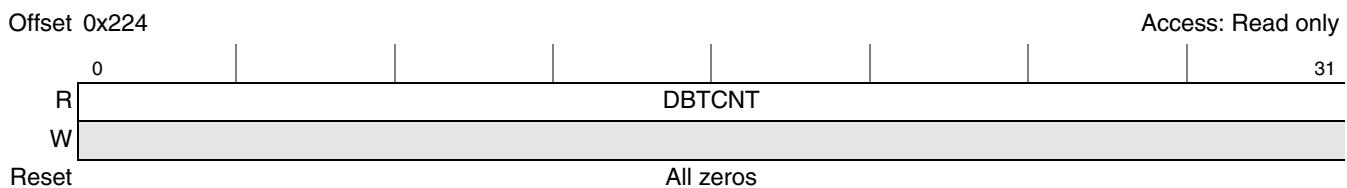


Figure 5-240. FMan Controller Flow B Match Count Register (FMFP\_CLFBMC)

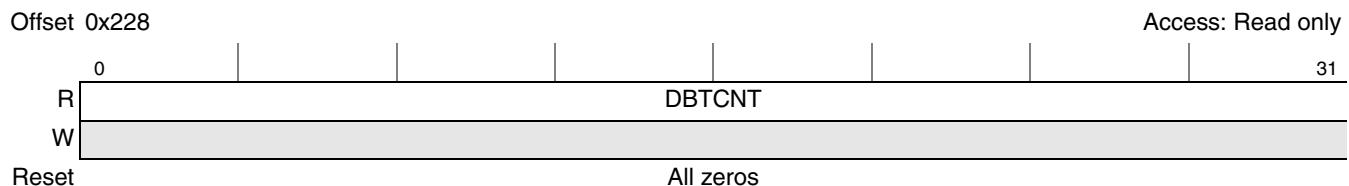
Table 5-263. FMFP\_CLFBMC Field Descriptions

Bits	Name	Description
0–31	DBTCNT	Debug trap event match counter for flow B

### 5.7.3.36 FMan Controller Flow C Match Count Register (FMFP\_CLFCMC)

This register is used for debug purposes.

The FMan controller flow C match count register (FMFP\_CLFCMC) counts the number of match events on flow C debug traps.



**Figure 5-241. FMan Controller Flow C Match Count Register (FMFP\_CLFCMC)**

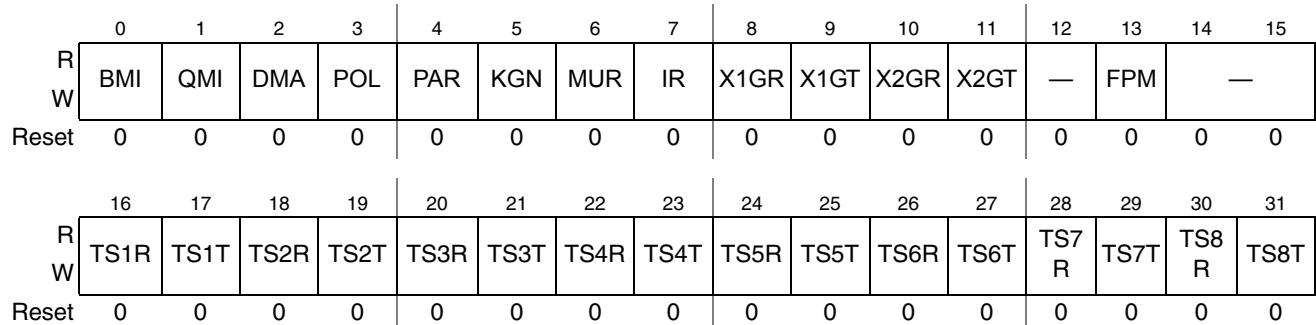
**Table 5-264. FMFP\_CLFCMC Field Descriptions**

Bits	Name	Description
0–31	DBTCNT	Debug trap event match counter for flow C

### 5.7.3.37 FM\_DECCEH - FM Double ECC Error Halt Register

The FM double ECC error halt register (FM\_DECCEH) configures masks for double ECC errors in FM. An enabled double ECC error halts the FM and causes Ethernet MACs to corrupt CRC on transmit.

Offset 0x22C (FM\_DECCEH) Access: User read/write



**Figure 5-242. FM\_DECCEH Configuration**

**Table 5-265. Register FM\_DECCEH Bits Description**

Field	Description
0 BMI	0 - masked BMI double ECC error 1 - enabled BMI double ECC error
1 QMI	QMI double ECC error
2 DMA	DMA double ECC error

**Table 5-265. Register FM\_DECCEH Bits Description (continued)**

Field	Description
3 POL	Policer double ECC error
4 PAR	Parser double ECC error
5 KGN	Keygen double ECC error
6 MUR	Multi-User RAM double ECC error
7 IR	Instruction RAM double ECC error
8 X1GR	EMAC9 <sup>1</sup> ,(10/2.5/1Gbps) Rx double ECC error
9 X1GT	EMAC9 (10/2.5/1Gbps) Tx double ECC error
10 X2GR	FMan_v3: EMAC10 (10/2.5/1Gbps) Rx double ECC error.
11 X2GT	FMan_v3: EMAC10 (10/2.5/1Gbps) Tx double ECC error.
12 —	Reserved
13 FPM	FPM double ECC error
14–15 —	Reserved
16 TS1R	EMAC1 <sup>2</sup> (2.5/1Gbps) <sup>3</sup> Rx double ECC error
17 TS1T	EMAC1 (2.5/1Gbps) Tx double ECC error
18 TS2R	EMAC2 (2.5/1Gbps) Rx double ECC error
19 TS2T	EMAC2 (2.5/1Gbps) Tx double ECC error
20 TS3R	EMAC3 (2.5/1Gbps) Rx double ECC error
21 TS3T	EMAC3 (2.5/1Gbps) Tx double ECC error
22 TS4R	EMAC4 (2.5/1Gbps) Rx double ECC error
23 TS4T	EMAC4 (2.5/1Gbps) Tx double ECC error

**Table 5-265. Register FM\_DECCEH Bits Description (continued)**

Field	Description
24 TS5R	EMAC5 (2.5/1Gbps) Rx double ECC error
25 TS5T	EMAC5 (2.5/1Gbps) Tx double ECC error
26 TS6R	EMAC6 (2.5/1Gbps) Rx double ECC error
27 TS6T	EMAC6 (2.5/1Gbps) Tx double ECC error
28 TS7R	EMAC7 (2.5/1Gbps) Rx double ECC error
29 TS7T	EMAC7 (2.5/1Gbps) Tx double ECC error
30 TS8R	EMAC8 (2.5/1Gbps) Rx double ECC error
31 TS8T	EMAC8 (2.5/1Gbps) Tx double ECC error

---

<sup>1</sup> EMAC9..10 (10/2.5/1Gbps): For FMan\_v3 refer to mEMAC chapter.

<sup>2</sup> EMAC1..8 (2.5/1Gbps): For FMan\_v3 refer to mEMAC chapter Note that this table depicts the superset of MACs. Some bits in this table may be reserved in a given SoC, if the corresponding EMAC does not exist. See SoC reference manual for a specific device.

<sup>3</sup> Some ports on some device do not support 2.5Gbps. See SoC reference manual for a specific device.

### 5.7.3.38 FPM TNUM Status 0–127 Registers (FMFP\_TS $n$ )

The FPM TNUM status 0–127 registers (FMFP\_TS $n$ ) hold TNUM statuses. A specific TNUM status is determined by the address. TNUM  $x$  status address is found at Offset 0x400 +  $x$ \*4.

Offset 0x400 (FMFP\_TS\_<i>)

Access: Read only

offset 4

range i = 0 .. 127

**Figure 5-243. FPM TNUM Status 0–127 Registers (FMFP\_TS<sub>n</sub>)**

**Table 5-266. FMFP\_TS<sub>n</sub> Field Descriptions**

Bits	Name	Description
0–1	CTA	FMan controller task association For FMan_v3: 00 FMan controller association is according to PortID FMan controller association 01 task is treated as constrained 10 task is treated as unconstrained 11 Reserved
2–5	—	Reserved
6	STL	Stalled task 0 Tasks works normally 1 Task is stalled
7	PRK	Parked task 0 Task is not parked 1 Task is parked
8	INEX	Task in execution 0 Task is not in an execution stage in an engine 1 Task is in an execution stage in an engine
9	EXEC	Task should be executed 0 Normal operation 1 Task should run in an engine
10	EOT	Task should end 0 Normal operation 1 Task should end
11	SYNC	Syncable task 0 Task is not syncable 1 Task is syncable
12	ORD	Ordered task 0 Task is not ordered 1 Task is in defined order
13	NIG	Task should be triggered 0 Normal operation 1 Task should be triggered after termination (for FMan controllers pending requests)
14	DER	Task with error 0 No data error 1 Data error occurred
15	ORR	Task with order restoration 0 No order restoration 1 Order restoration was requested
16	—	Reserved
17–23	ODC	Open DMA count Incremented for DMA transaction and decremented when DMA transaction is done
24–26	—	Reserved
27–31	DISPC	Dispatches Count

## 5.7.4 Functional Description

The frame processing manager (FPM) manages the processing of frames. Each frame processing is associated with an FPM hardware task number (tnum). Each FMan module signals the FPM when it is ready to accept tasks, and the FPM dispatches a task to the module only if it is ready to accept it.

### 5.7.4.1 Order Definition and Restoration

TNUMs (like Rx frames) might need to be ordered.

Order definition point - a point where TNUMs get their order - it is done per PortID.

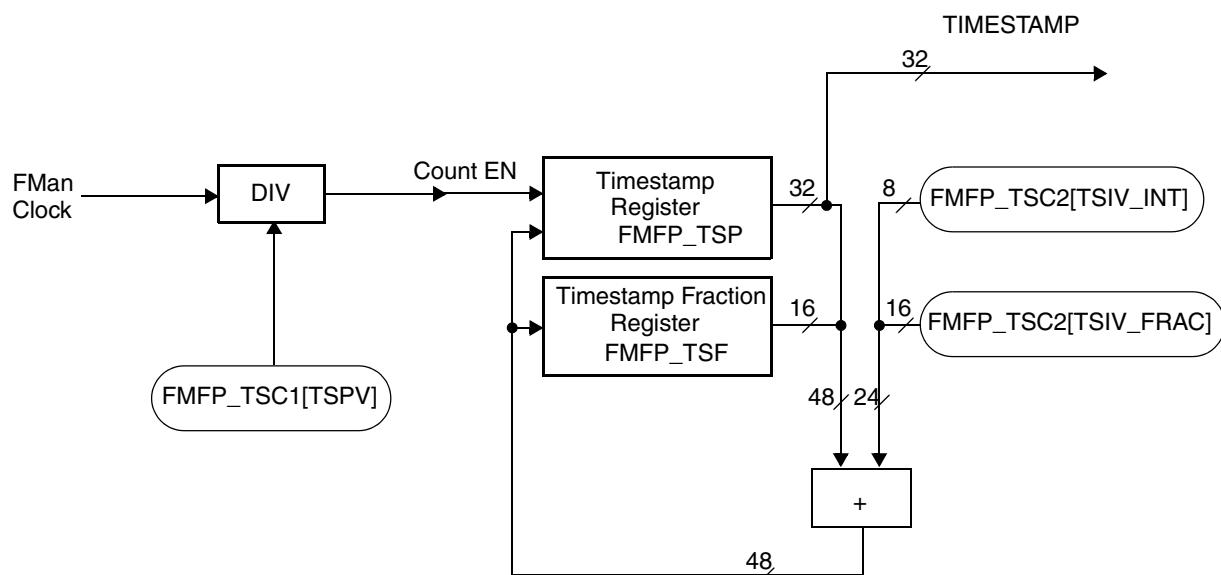
Order restoration point - a point where TNUMs that have order should pass in the defined order before being dispatched to the next engine.

### 5.7.4.2 Timestamp and Prescale (TSP)

The timestamp logic provides high precision time measurement. The prescale logic enables high precision count rates by fixed-point accumulation in addition to the 32-bit integer counter.

The timestamp and prescale logic is described in [Figure 5-244](#). The FMan clock is passed through a pre-divider controlled by FMFP\_TSC1[TSPV] that enables frequency division (see [Section 5.7.3.9, “FPM TimeStamp Control1 Register \(FMFP\\_TSC1\)”](#)).

The timestamp value comprises 32-bit integer part in FMFP\_TSP and 16-bit fraction part in FMFP\_TS<sub>F</sub> (see [Section 5.7.3.11, “FPM Time Stamp Register \(FMFP\\_TSP\),”](#) and [Section 5.7.3.12, “FPM Time Stamp Fraction Register \(FMFP\\_TS<sub>F</sub>\)”](#)). The TIMESTAMP value is taken from FMFP\_TSP. On each timestamp count a 48-bit fix point addition is done to accumulate fraction and integer increment values that are configured in FMFP\_TSC2[TSIV\_FRAC] and FMFP\_TSC2[TSIV\_INT] respectively. This mechanism enables accurate TIMESTAMP average rate.



**Figure 5-244. Timestamp & Prescale Logic**

#### 5.7.4.2.1 TSP Example

Assuming the clock is 600MHz and requirement is to achieve 1MHz count rate on a TIMESTAMP representation of 16-bit integer and 16-bit fraction, the following setup would achieve an accurate count rate:

- Find effective count rate at TIMESTAMP least significant bits: Effective\_Count\_Rate = 1MHz x  $2^{16} = 65,536\text{MHz}$
- Find frequency ratio between effective count rate and the clock:  
Effective\_Count\_Rate / CLK =  $65,536/600 = 109.2266666\dots$   
This is the accurate TIMESTAMP integer increment value.
  - The integer part 109 (0x6D) should be programmed to FMFP\_TSC2[TSIV\_INT].
  - The fraction part should be converted to the closest hex fraction representation as follows:  
Multiply of the accurate division fraction result by  $2^{16}$  and round the result to closest decimal integer. This gives 14855. Converting this value to hex gives 0x3A07. Program this value to FMFP\_TSC2[TSIV\_FRAC].
- Verify the time base accuracy by converting back the result as follows:
  - Look at the TIMESTAMP and fraction part as a 48-bit integer in hex representation counting at  $2^{16}$  times the effective frequency. Convert the value to decimal. This gives 7,158,279.
  - Multiply by 600MHz and divide by  $2^{16}$  to get the effective count rate in the least significant TIMESTAMP bit. Divide again by  $2^{16}$  to get the count rate of TIMESTAMP 16 MS bits. The result is:  $7,158,279 \times 600 / (65536^2) = 1.0000000242\text{ MHz}$ .

#### 5.7.5 Initialization Information

The FPM allocates and de-allocates TNUMs automatically. Therefore, the only registers that are required to be configured are the following:

- FM\_RIE ([Section 5.7.3.6, “FMan Rams Interrupt Enable Register \(FM\\_RIE\)”](#))
- FMFP\_TSC1 ([Section 5.7.3.9, “FPM TimeStamp Control1 Register \(FMFP\\_TSC1\)”](#))
- FMFP\_TSC2 ([Section 5.7.3.10, “FPM TimeStamp Control2 Register \(FMFP\\_TSC2\)”](#))
- FM\_RCR ([Section 5.7.3.13, “FMan Rams Control and Event Register \(FM\\_RCR\)”](#))
- FMFP\_EE ([Section 5.7.3.23, “FPM Event and Enable Register \(FMFP\\_EE\)”](#))

Other registers are either used for debug or by the FMan controllers.

## 5.8 Frame Manager—DMA

### 5.8.1 FMan DMA Overview

The FMan DMA transfers blocks of data from the FMan memory to the external memory and vice-versa.

This figure shows a block diagram of the DMA within the FMan.

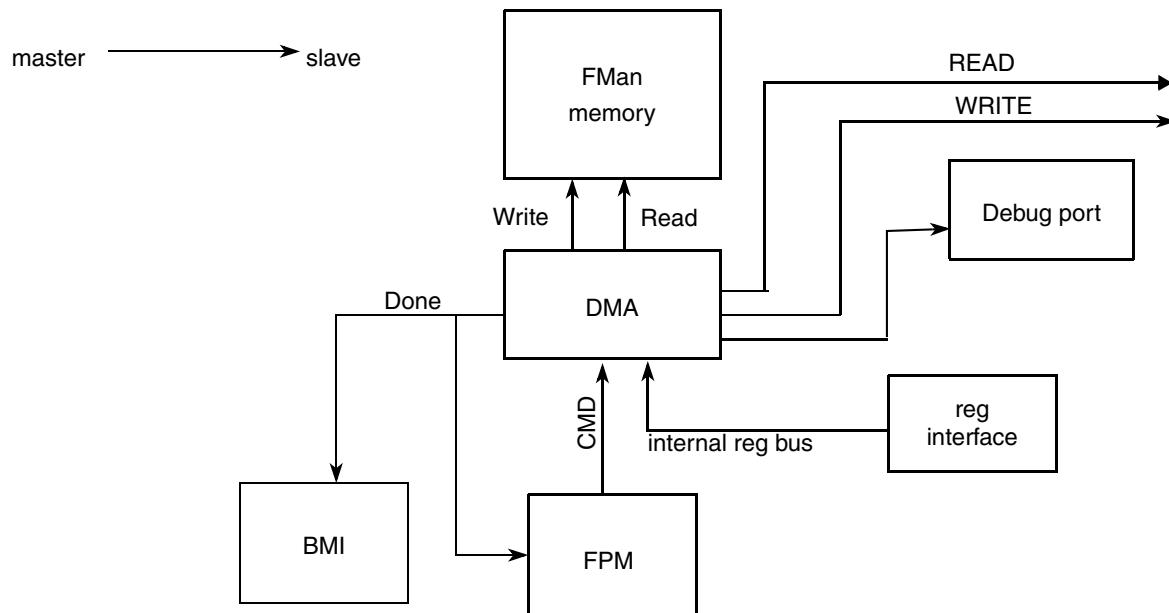


Figure 5-245. DMA in the FMan Block Diagram

### 5.8.2 FMan DMA Feature Summary

Key features of the FMan DMA include the following:

- DMA command queue (for pending transactions)
  - FMan\_v3: support for a 84-command DMA command queue
- Support for system bus emergency states
  - Support for emergency smoother, so transition to emergency has minimum amount of data beats transferred
- Support for changing priorities on the FMan memory and on the external bus, due to internal events (native network connection emergency, queues full/empty) and user settings (mode register)
- Support external bus/FMan asynchronous clock ratios, all clock ratios are valid
- Support for transfer of aligned/unaligned data
- Output of current transaction information on the external bus interface (portID bits represent which hardware port requested the transaction)

### 5.8.3 Introduction about the FMan DMA

The FMan DMA module is triggered automatically by the BMI or the FMan controller when the FMan needs to access to external memory.

### 5.8.4 FMan DMA Memory Map/Register Definition

FMan DMA registers reside in a 4 KB space in the FMan memory map. See [Section 5.4.2, “FMan Detailed Memory Map,”](#) for details.

Important things to consider when accessing the FMan DMA registers are as follows:

- All the FMan DMA registers (unless specifically noted) are accessible for read/write.
- FMDM\_TAx and FMDM\_TCID are read-only registers.
- FMDM\_SR bits are cleared by writing ones to them; writing zeros has no effect. After initialization of the system, FMDM\_SR[BER] must be cleared (by writing ‘1’) before unmasking the interrupt controller mask bits.
- During system reset, all FMan DMA registers are reset except for FMDM\_SR[BER], FMDM\_TAx, and FMDM\_TCID.

This table lists the FMan DMA register descriptions and offset addresses.

**Table 5-267. FMan DMA Memory Map**

Offset	Register	Access	Reset Value	Section/Page
<b>General Registers</b>				
0x00	FMan DMA status register (FMDM_SR)	R/W	—	<a href="#">5.8.4.1/5-321</a>
0x04	FMan DMA mode register (FMDM_MR)	R/W	FMan_v3: 0x2000_E800	<a href="#">5.8.4.2/5-322</a>
0x08	FMan DMA bus threshold register (FMDM_TR)	R/W	FMan_v3: 0x3F00_3F00	<a href="#">5.8.4.3/5-324</a>
0x0c	FMan DMA bus hysteresis register (FMDM_HY)	R/W	FMan_v3: 0x2A00_0000	<a href="#">5.8.4.4/5-326</a>
0x10	FMan DMA SOS emergency Threshold Register (FMDM_SETR)	R/W	All zeros	<a href="#">5.8.4.5/5-327</a>
0x14	FMan DMA transfer bus address high register (FMDM_TAH)	R	—	<a href="#">5.8.4.6/5-327</a>
0x18	FMan DMA transfer bus address low register (FMDM_TAL)	R	—	<a href="#">5.8.4.7/5-328</a>
0x1C	FMan DMA transfer bus communication ID register (FMDM_TCID)	R	—	<a href="#">5.8.4.8/5-328</a>
0x28	FMan DMA buffer watchdog counter value (FMDM_WCR).	R/W	All zeros	<a href="#">5.8.4.9/5-329</a>
0x2C	Fman DMA buffer base in Fman memory register (FMDM_EBCR).	R/W	All zeros	<a href="#">5.8.4.10/5-330</a>
0x30-0x50	Reserved	—	—	—
0x54	FMan DMA Debug Counter (FMDM_DCR)	R/W	All zeros	<a href="#">5.8.4.11/5-330</a>
0x58	FMan DMA EMergency Smoother Register (FMDM_ESMR)	R/W	All zeros	<a href="#">5.8.4.12/5-330</a>

**Table 5-267. FMan DMA Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x5C	Reserved	—	—	—
0x60-0xDC	FMan DMA portID #0..31 register (FMDM_PortID0..31) Actual numbers depend on PortIDs implemented in each device	R/W	All zeros	<a href="#">5.8.4.13/5-331</a>
0xFC	Reserved	—	—	—
0x100-0x10C	Reserved	—	—	—

### 5.8.4.1 FMan DMA Status Register (FMDM\_SR)

FMDM\_SR reports bus error events that are recognized by the FMan DMA module on all of the FMan DMA channels. On recognition of a bus error or watch-point event, the FMan DMA sets the corresponding bit in FMDM\_SR. In addition, FMDM\_SR holds a status bit to indicate that there are pending transactions in the DMA that are not completed. FMDM\_SR is a memory-mapped register that can be read at any time.

All bits (except bit 0) are cleared by writing ones to them. Writing zeros has no effect. All unmasked bits must be cleared before the CPU clears the interrupt request.

Offset 0x00															Access: Mixed				
R	0	2	3	4	5	6	7	8	9	10	11	12	13	—	—	—	—	—	31
	—	CMDQNE	BER	RDB_ECC	WRB_SECC	WRB_FECC	DPEXT_SECC	DPEXT_FECC	DPDAT_SECC	DPDAT_FECC	SPDAT_FECC	—	—	—	—	—	—	—	—
W	—	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Figure 5-246. FMan DMA Status Register (FMDM\_SR)**

**Table 5-268. Register FMDM\_SR Bits Description**

Bits	Name	Description
0–2	—	Reserved
3	CMDQNE	Indication that there are uncompleted pending transactions in the DMA. (i.e. the command queue is not empty) 0 Command queue is empty. 1 Command queue not empty. <b>Note:</b> CMDQNE is read-only.
4	BER	Bus error event Used to indicate that the FMan DMA channel terminated with an error during a read or write transaction. The FMan DMA bus 1 error address is read from the FMDM_TAH/L register. The portID, TNUM, Done Tag, and ICID values are read from the FMDM_TCID register. 0 No bus error event occurred on bus 1 A bus error event occurred on bus <b>Note:</b> BER bit is not reset. The value is maintained after reset. On power on reset the value is undefined, therefore it must be cleared by writing ‘1’ before unmasking.

**Table 5-268. Register FMDM\_SR Bits Description (continued)**

Bits	Name	Description
5	RDB_ECC	Read buffer ECC error on the FMan side 0 No ECC error 1 ECC Error occurred on the Read Buffer This field is not supported in FMan_v3
6	WRB_SECC	Write buffer ECC error on the system side 0 No ECC error 1 ECC Error occurred on the Write Buffer System side This field is not supported in FMan_v3.
7	WRB_FECC	Write buffer ECC error on the FMan side 0 No ECC error 1 ECC Error occurred on the Write Buffer FMan side This field is not supported in FMan_v3.
8	DPEXT_SECC	Dual-port external address ECC error on the system side 0 No ECC error 1 ECC Error occurred on the dual-port external address memory on the system side This field is not supported in FMan_v3
9	DPEXT_FECC	Dual-port external address ECC error on the FMan side 0 No ECC error 1 ECC Error occurred on the dual-port external address memory on the FMan side This field is not supported in FMan_v3
10	DPDAT_SECC	Dual-port data ECC error on the system side 0 No ECC error 1 ECC Error occurred on dual-port data memory on the system side This field is not supported in FMan_v3.
11	DPDAT_FECC MEM0_ECC	Dual-port data ECC error on the FMan side 0 No ECC error 1 ECC error occurred on dual-port data memory on the FMan side This bit is not supported in FMan_v3 In FMan_v3 the memory is single port.
12	SPDAT_FECC MEM1_ECC	Single-port data ECC error on the FMan side 0 No ECC error 1 ECC Error occurred on single-port data memory on the FMan side In FMan_v3 the memory is single port.
13–31	—	Reserved

### 5.8.4.2 FMan DMA Mode Register (FMDM\_MR)

FMDM\_MR enables the user to mask the FMan DMA interrupts and emergency mode and to set them manually. If an FMDM\_MR bit is set, the corresponding interrupt in FMDM\_SR is enabled. If the bit is cleared, the corresponding interrupt in FMDM\_SR is masked.

#### NOTE

In FMan\_v3 do not modify the reset value unless events are unmasked (BER\_MSK, ECC\_MSK).

In other devices, the recommended value to be programmed is 0x00006800.

Offset 0x04 Access: Read/Write

R	0	1	2	3	4	7	8	9	10	11	12	13	14	15	16	18	19	20	21	22	24	25	26	27	28	31
W	Cache_ovrd	AID_ovrd	—	—	—	—	BER_MSK	EB_MSK	—	—	EB_EME	CEN	—	—	DBG_CNT	BMI_EMR	ECC_MSK	—	—	—	—	—	—	—	—	
Reset FMan_v	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	

Figure 5-247. FMan DMA Mode Register (FMDM\_MR)

Table 5-269. Register FMDM\_MR Bits Description

Bits	Name	Description
0–1	Cache_ovrd	Override the cache field 00 No override of the cache field 01 Data must not be stashed in system level cache. 10 Data can be stashed in system level cache. 11 Data must be stashed in system level cache.  Cache Attributes This field is used for debug purposes and should be configured to '00'. The functional mode of operation is configured in FMBM_xDA (for all types of hardware ports. x='R' or 'T' or 'O').
2	AID_ovrd	AID override Determines the ordering of transactions on the system bus 0 Normal; work according to AID mode (bit 27). 1 Override; AID is '0000' If this bit is set, all transactions to the system bus are performed in order.
3–9	—	Reserved.
10	BER_MSK	Mask external bus error events 0 Mask external bus error events 1 Enable external bus error events
11	EB_MSK	Mask emergency on external bus. 0 Mask emergency towards External bus 1 Enable emergency towards External bus
12–13	—	Reserved
14–15	EB_EME	Set external bus emergency mode. 00 normal (no emergency) 01 Elevated Bandwidth State (EBS) priority 10 SOS priority 11 EBS + SOS priority It is recommended to clear these bits.
16–18	CEN (DMA buffer)	DMA buffer size The FMan DMA requires a buffer to be allocated in the FMan internal memory. This requires the allocation of $(N+1) * 512$ bytes in FMan internal memory. See <a href="#">Section 5.8.4.10, "FMan DMA buffer Base in FMan Memory Value Register (FMDM_EBCR)." </a> Smaller values may degrade performance in some systems.
19	—	Reserved The bit reset value ('0') should be preserved
20	—	Reserved, The bit reset value ('1') should be preserved

**Table 5-269. Register FMDM\_MR Bits Description (continued)**

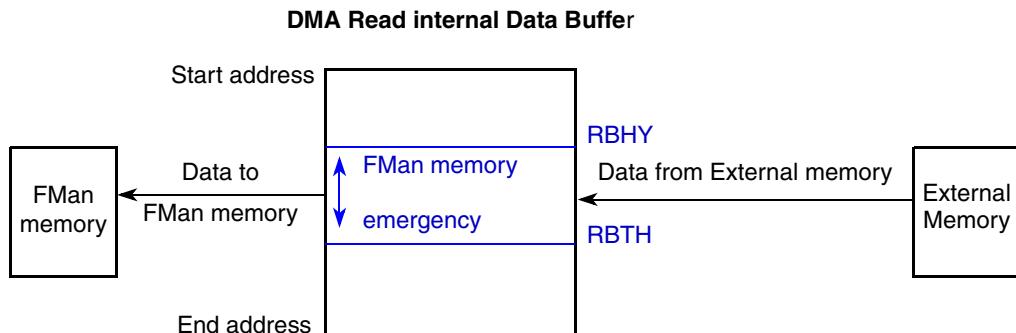
Bits	Name	Description
21	—	Reserved The bit reset value ('0') should be preserved
22–24	DBG_CNT	Set the debug counter to count the following signal: 000 No counting 001 Count transactions that are completed (DONE commands). 010 Count Command Queue emergency signal 101 Count FPM WAIT signal 110 Single bit ECC errors other Reserved
25	BMI_EMR	Emergency level that is set by the BMI emergency signal 0 EBS emergency 1 SOS emergency It is recommended to clear this bit.
26	ECC_MSK	Mask ECC error events (FMDM_SR) 0 Mask ECC error events 1 Enable ECC error events
27–31	—	Reserved

### 5.8.4.3 FMan DMA Threshold Register (FMDM\_TR)

FMDM\_TR defines the emergency levels towards FMan memory and the external bus in combination with FMDM\_HY. An emergency is activated upon reaching the threshold value. The emergency level is held until the data buffer level reaches the hysteresis value.

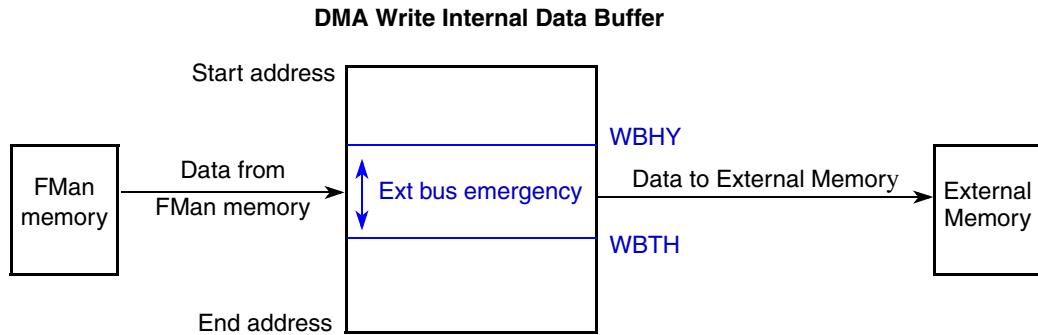
In FMan\_v3, this register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug.

This threshold is not supported in FMan\_v3. When the internal DMA read data buffer reaches its threshold (RBTH), high priority towards FMan memory port is asserted. It is held asserted until it reaches the hysteresis value (RBHY). This figure shows the DMA read data path.



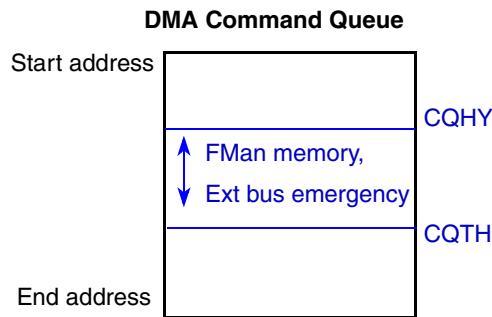
**Figure 5-248. DMA Read Data Path**

This threshold is not supported in FMan\_v3. When the internal DMA write data buffer reaches its threshold (WBTH), an emergency towards the external bus is asserted. It is held asserted until it reaches the hysteresis value (WBHY). This figure shows the DMA write data path.



**Figure 5-249. DMA Write Data Path**

The DMA contains a command queue. When the DMA command queue reaches its threshold (CQTH), high priority towards FMan memory port and emergency level towards the external bus is asserted. It is held asserted until it reaches the hysteresis value (CQHY). This figure shows the command queue path.



**Figure 5-250. DMA Command Queue**

Note, write to the register must preserve the default value of reserved bits. This figure shows the FMan DMA threshold register (FMDM\_TR).

Offset 0x08																Access: Read/Write																
R				CQTH				RBTH				—				WBTH																
W																																
Reset	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

**Figure 5-251. FMan DMA Threshold Register (FMDM\_TR)**

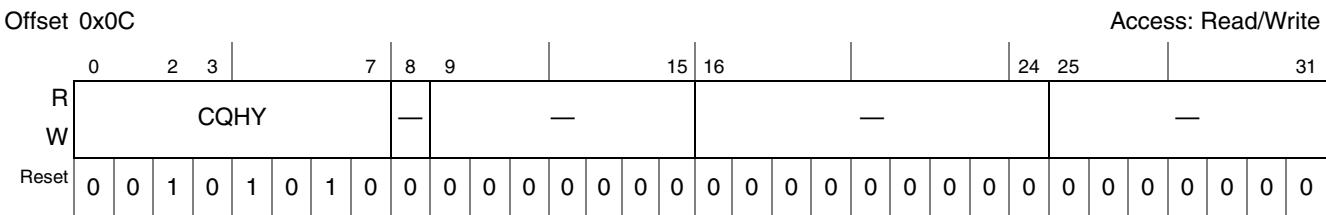
**Table 5-270. Register FMDM\_TR Bits Description**

Field	Name	Description
0–7	CQTH	Command queue threshold When the command queue reaches its threshold value, high priority towards FMan memory port and external address bus is asserted.
8	—	Reserved
9–15	RBTH	Read internal data buffer threshold When the read internal data buffer reaches its threshold value, high priority towards FMan memory port is asserted. This field is not supported in FMan_v3.
16–24	—	Reserved (do not change reset values).
25–31	WBTH	Write internal data buffer threshold When the write internal data buffer reaches its threshold value, high priority towards external bus is asserted. This field is not supported in FMan_v3.

#### 5.8.4.4 FMan DMA Hysteresis Registers (FMDM\_HY)

FMDM\_HY defines the emergency levels towards FMan memory and the external bus in combination with FMDM\_TR. An emergency level is activated upon reaching the threshold value in the internal buffer. The emergency level is held until the internal buffer is reduced to the hysteresis value. For more details, see [Section 5.8.4.3, “FMan DMA Threshold Register \(FMDM\\_TR\).”](#)

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug.



**Figure 5-252. FMan DMA Hysteresis Register (FMDM\_HY)**

**Table 5-271. Register FMDM\_HY Bits Description**

Field	Name	Description
0–7	CQHY	Command queue hysteresis When the command queue reaches its threshold value, high priority towards FMan memory port and external address bus is asserted. It is held until reaching the hysteresis value.
8	—	Reserved
9–15	RBHY	Read internal data buffer hysteresis When the read internal data buffer reaches its threshold value, high priority towards FMan memory port is asserted. It is held until reaching the hysteresis value. This field is not supported in FMan_v3.

**Table 5-271. Register FMDM\_HY Bits Description (continued)**

Field	Name	Description
16–24	—	Reserved
25–31	WBHY	Write internal data buffer hysteresis When the Write internal data buffer reaches its threshold value, high priority towards external bus is asserted. It is held until reaching the hysteresis value. This field is not supported in FMan_v3.

### 5.8.4.5 FMan DMA SOS Emergency Threshold Register (FMDM\_SETR)

FMDM\_SETR sets the amount of clocks that can pass before the FMan DMA raises the emergency level to the system bus.

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug.



**Figure 5-253. FMan DMA SOS Emergency Threshold Register (FMDM\_SETR)**

**Table 5-272. Register FMDM\_SETR Bits Description**

Bits	Name	Description
0–31	SOSTR	Pre-programmed value for the emergency counter

### 5.8.4.6 FMan DMA Transfer Address High Register (FMDM\_TAH)

FMDM\_TAH holds the higher 16 bits of the address accessed during the current bus transaction. In case of a bus error or watch-point event, the address is captured in the register until the corresponding status bit in the FMDM\_SR register is reset.



**Figure 5-254. FMan DMA Transfer Address High Register (FMDM\_TAH)**

**Table 5-273. Register FMDM\_TAH Bits Description**

Bits	Name	Description
0–15	—	Reserved
16–31	transfer_address_high	The 16 msbs of the address accessed during current bus transaction This value is updated at every transfer end. The value is captured in case of bus error or watch-point event and released when the FMDM_SR[BER] or FMDM_SR[WATCHP_EV] bit is reset by the user (writing “1” to it). The value of this field is undefined immediately following reset.

#### 5.8.4.7 FMan DMA Transfer Address Low Register (FMDM\_TAL)

FMDM\_TAL holds the lower 32 bits of the address accessed during the current bus transaction. In case of a bus error or watch-point event, the address is captured in the register until the corresponding status bit in the FMDM\_SR register is reset.



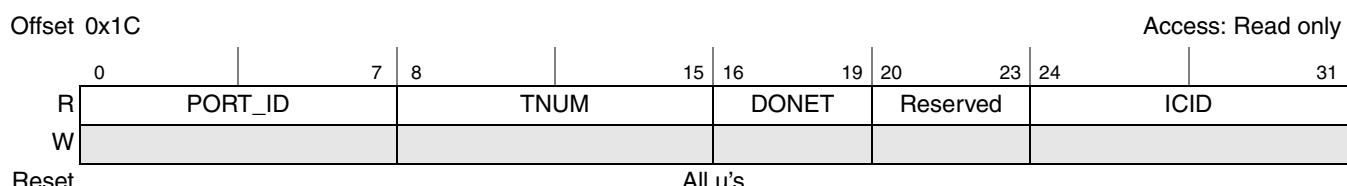
**Figure 5-255. FMan DMA Transfer Address Low Register (FMDM\_TAL)**

**Table 5-274. Register FMDM\_TAL Bits Description**

Bits	Name	Description
0–31	transfer_address_low	32 lsbs of the address accessed during current bus transaction This value is updated at every transfer end. The value is captured in case of bus error and released when the FMDM_SR[BER] or FMDM_SR[WATCHP_EV] bit is reset by the user (writing “1” to it). The value of this field is undefined immediately following reset.

#### 5.8.4.8 FMan DMA Transfer Communication ID Register (FMDM\_TCID)

FMDM\_TCID holds the serial number of the peripheral that was served during the current bus transaction. In case of a bus error or watch-point event, the portID, TNUM, Done Tag, and ICID values are captured in the register until the corresponding status bit in the FMDM\_SR register is reset.



**Figure 5-256. FMan DMA Transfer Communication ID Register (FMDM\_TCID)**

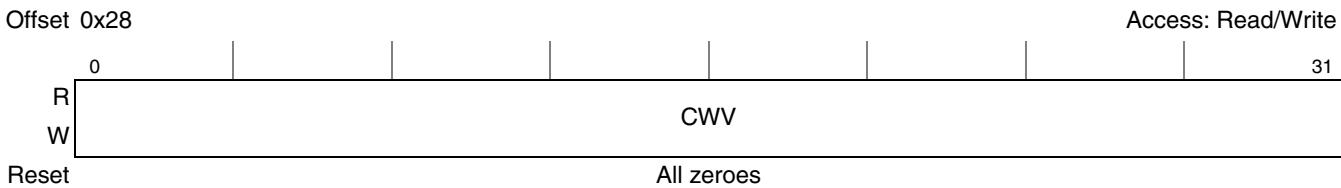
**Table 5-275. Register FMDM\_TCID Bits Description**

Bits	Name	Description
0–7	PORT_ID	PortID served during current bus transaction This value is updated at every transfer end. The value is captured in case of bus error and released when the FMDM_SR[BER] or in FMan_v3, FMDM_SR[WATCHP_EV] bit is reset by the user (writing ‘1’ to it). The value of this field is undefined immediately following reset.
8–15	TNUM	TNUM served during current bus transaction This value is updated at every transfer end. The value is captured in case of bus error and released when the FMDM_SR[BER] or in FMan_v3, FMDM_SR[WATCHP_EV] bit is reset by the user (writing ‘1’ to it). The value of this field is undefined immediately following reset.
16–19	DONET	DONE TAGserved during current bus transaction This value is updated at every transfer end. The value is captured in case of bus error and released when the FMDM_SR[BER] or in FMan_v3, FMDM_SR[WATCHP_EV] bit is reset by the user (writing ‘1’ to it).
20–23	—	Reserved
24–31	ICID	ICID served during current bus transaction This value is updated at every transfer end. The value is captured in case of bus error and released when the FMDM_SR[BER] or in FMan_v3, FMDM_SR[WATCHP_EV] bit is reset by the user (writing ‘1’ to it). The value of this field is undefined immediately following reset.

#### 5.8.4.9 FMan DMA buffer Watchdog Counter Value (FMDM\_WCR)

This register is for debug purposes. It should be cleared.

FMDM\_WCR holds the value of the DMA buffer watchdog counter. The DMA buffer may cause a stall condition, in a non-operational system. This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug.



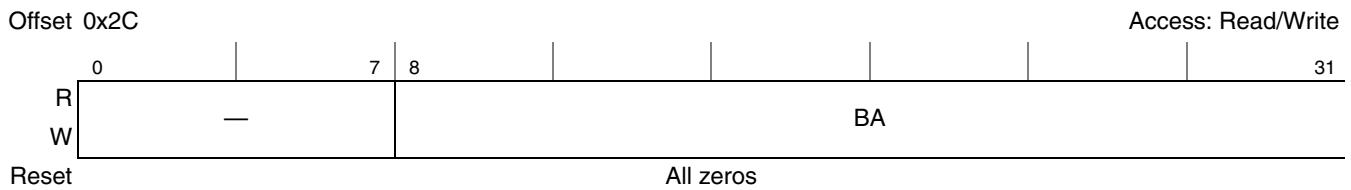
**Figure 5-257. FMan DMA buffer Watchdog Counter Value (FMDM\_WCR)**

**Table 5-276. Register FMDM\_WCR Bits Description**

Bits	Name	Description
0–31	CWV	DMA buffer watchdog counter initial value

### 5.8.4.10 FMan DMA buffer Base in FMan Memory Value Register (FMDM\_EBCR)

FMDM\_EBCR holds the DMA buffer base address in the FMan memory. The base address must be aligned to 64 bytes.



**Figure 5-258. FMan DMA buffer Base in FMan Memory Value Register (FMDM\_EBCR)**

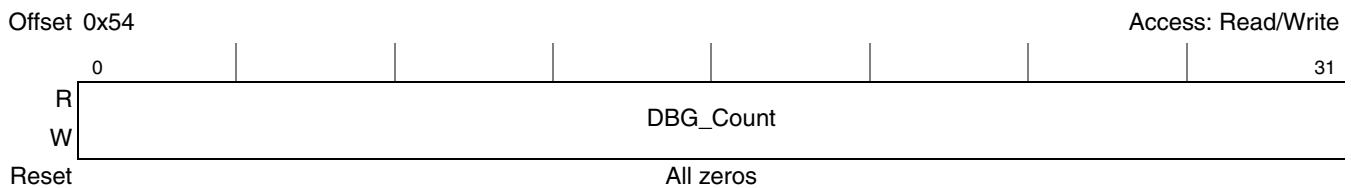
**Table 5-277. Register FMDM\_EBCR Bits Description**

Bits	Name	Description
0–7	—	Reserved
8–31	BA	DMA buffer base address in Fman memory. Must be aligned to 64 bytes. The user allocates a buffer from this address. See <a href="#">Section 5.8.4.2, “FMan DMA Mode Register (FMDM_MR)”</a> CEN field for buffer size.

### 5.8.4.11 FMan DMA Debug Counter (FMDM\_DCR)

FMDM\_DCR holds the value of the debug counter events. The event to be counted is set according to FMDM\_MR[DBG\_CNT]. The value written to the register is the initial value of the counter.

This register is for internal use. It is recommended not to modify its reset value.



**Figure 5-259. FMan DMA Debug Counter (FMDM\_DCR)**

**Table 5-279. Register FMDM\_DCR Bits Description**

Bits	Name	Description
0–31	DBG_Count	Number of debug events

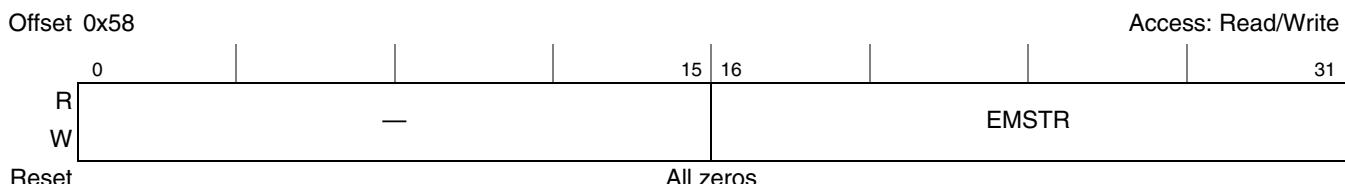
### 5.8.4.12 FMan DMA Emergency Smoother Register (FMDM\_EMCSR)

FMDM\_EMCSR sets the minimum amount of data beats transferred on the AXI read and write ports before lowering the emergency level after the internal emergency condition is no longer met.

The counter counts the amount of data beats from the deassertion condition, and if there is no longer a need for the emergency mode when the counter reaches zero, the emergency level is lowered. Every time the internal emergency condition is met, the counter of the DATA beats transferred is reloaded.

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires fine tuning or debug.

A value of all zeros disables the counter.



**Figure 5-260. FMan DMA Emergency Smoother Register (FMDM\_EMSR)**

**Table 5-280. Register FMDM EMSR Bits Description**

Bits	Name	Description
0–15	—	Reserved
16–31	EMSTR	Preprogrammed value for the emergency switching counter Represents the minimum amount of data beats on system bus read and write ports. After emergency from the BMI is deasserted or a condition for SOS emergency is deasserted (long period in EBS emergency mode), transferred before emergency level is deasserted.

#### 5.8.4.13 FMan DMA PortID Registers (FMDM\_PortIDn)

The FMDM\_PortID $n$  registers map the hardware port (portID) to the required PortID\_BASE. Only valid portIDs have a valid register entry (see [Table 5-14](#), “Hardware PortIDs”). The rest of the entries are reserved.

Each register holds PortID base entries for two PortIDs, as follows:

- FMDM\_PortID0 holds PortID entry 1 (PortID 0 does not exist).
  - FMDM\_PortID1 holds PortID entries 2–3.
  - ....
  - ....
  - FMDM\_PortID31 holds PortID entries 62–63.

PortID\_BASE for odd PortIDs are written to register bits [20:31], for even PortIDs are written to bits [4:15]. In total, 32 FMDM PortID registers hold 63 entries.

## **NOTE**

Only PortIDs which are implemented in the device have an associated FMDM\_PortID $n$  register. Refer for DPAA introduction chapter in the SoC manual for specifics on each device.

Offset 0x60 (FMDM_PortID<i>)										Access: Read/Write	
offset 4											
range 0.. 31											
	0	3	4			15	16	19	20		31
R	—	PortID_BASE<2*i>				—	PortID_BASE<2*i+1>				
W	—					—					
Reset						All zeros					

**Figure 5-261. Hardware Port (PortID) Registers (FMDM\_PortID)**

**Table 5-281. Register FMDM\_PortID<i> (0.. 31) Bits Description**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–3	—	Reserved
4–15	PortID_BASE<2*i>	PortID_BASE—for PortID 2*i
16–19	—	Reserved
20–31	PortID_BASE<2*i+1>	PortID_BASE—for PortID 2*i+1

### **5.8.5 FMan DMA Functional Description**

The DMA is used to transfer data between external memory and the FMan memory. The DMA transactions are triggered by the BMI or the FMan controller when external memory needs to be accessed. The DMA performs read or write transactions and performs some alignment or padding to minimize bus utilization.

### **5.8.5.1 Bus Error**

Upon encountering a bus error, status bits are set in the status register (See [Section 5.8.4.1, “FMan DMA Status Register \(FMDM\\_SR\)”](#)). In addition, other parameters pertaining the bus transaction when the error condition is encountered, are captured (See [Section 5.8.4.6, “FMan DMA Transfer Address High Register \(FMDM\\_TAH\),”](#) [Section 5.8.4.7, “FMan DMA Transfer Address Low Register \(FMDM\\_TAL\),”](#) and [Section 5.8.4.8, “FMan DMA Transfer Communication ID Register \(FMDM\\_TCID\)”](#)). Releasing the captured parameters is done by writing 1 to the corresponding bus error bit in the status register. Bus error events can be masked, using the DMA mode register. (See [Section 5.8.4.2, “FMan DMA Mode Register \(FMDM\\_MR\),”](#))

There are two modes in the DMA handling of bus errors configured in FMDM\_MR[SBER]:

- Stop on bus error—If a bus error is reported to the FMan DMA, the FMan DMA freezes its activity (after ending all open FMan DMA requests) and no longer performs any new read or write transactions until the FMan DMA is reset.
  - Continue on bus error—The FMan DMA continues its activity after a bus error is reported. The FMan DMA freezes the FMDM\_TAx and FMDM\_TCID registers holding the last address, TNUM, portID, DONE TAG, and ICID until FMDM\_SR[BER] is reset.

Because the status bit is not reset with the system reset and can wake up in any state, the user must clear this bit (by writing “1” to it) before unmasking the interrupt (FMDM\_MR[BER\_MSK] or in FMan\_v3, FMDM\_SMR[BER\_MSK]) and activating the FMan DMA. Otherwise, a false interrupt may be generated.

### 5.8.5.2 ECC Error

In case of an ECC multi-bit error while reading from the DMA internal memory, a status bit is asserted to indicate which memory had the fail and an interrupt is generated (unless masked by FMDM\_MR[ECC\_MSK]).

Because the status bit is not reset in the system reset and can wake up in any state, the user must clear this bit by writing 1 to it before unmasking the interrupt (FMDM\_MR[ECC\_MSK]) and activating the FMan DMA. Otherwise, a false interrupt may be generated.

Single-bit ECC errors can be counted in the debug counter (FMDM\_DCR) by setting FMDM\_MR[DBG\_CNT] to “110” (see [Table 5-269](#), “Register FMDM\_MR Bits Description”).

### 5.8.5.3 Halt

When the DMA receives the HALT state indication, it does not generate new external transactions.

The DMA completes all transfers for opened transactions on the external bus and then halts its work until the HALT state signal is deasserted.

When HALT state indication is deasserted, the DMA resumes its work.

### 5.8.5.4 Reset

When a system reset is issued, all DMA registers and state machines are reset except the registers related to bus error (FMDM\_SR[BER], or in FMan\_v3, FMDM\_SR[WATCHP\_EV], FMDM\_TAx, FMDM\_TCID) whose values are maintained so the user can read them.

When the user issues a soft reset to FMan by setting FM\_RSTC[RFM], the DMA gracefully finished all pending transactions. The data transferred in these transaction is not valid. The DMAs registers are not reset, therefore the DMA does not need to be re-initialized after this type of reset.

## 5.8.6 FMan DMA Initialization

The following initialization steps must be completed before starting to work with the FMan DMA:

1. Clear the FMan DMA Status Register (FMDM\_SR) bits by writing 1 to them.
2. Initialize the FMan DMA buffer Base in FMan Memory Value Register (FMDM\_EBCR). The base address must be aligned to 64 bytes. See FMDM\_MR[CEN]) for explanation about the size. Program the relevant FMDM\_PortIDn registers for the PortID\_BASE value according to the portIDs that are used.

## 5.8.7 FMan DMA Debug Functionality

Key features of the FMan DMA debug functionality include the following:

- DEBUG port— Transferring debug trace data to the NEXUS port.
- Source ID—PORT ID field is output to interconnect.

- Bus Error or parameter capture —Once there is an indication of a bus error, the address, portID, TNUM, DONE TAG and ICID of the last served transaction are captured in three special registers.

## 5.9 Frame Manager—Parser

### 5.9.1 Parser Overview

The Parser performs protocol header parsing and validation for a wide range of frame formats with varying protocols and encapsulation. A hard-coded parser function is used for the known and stable protocols. The hardwired parsing capabilities can be supplemented with user-programmed parse functions to support protocols not supported by the hardwired functionality, including proprietary protocols and shim headers inserted between otherwise well-known protocols.

The Parser also does the following:

- Is notified of complete received frames to process.
- Parses the frame according to the per-port configuration
- Reads the frame header from the FMan Memory and writes the frame parse results to the FMan Memory

### 5.9.2 Parser Features Summary

- Performance
  - Supports wirespeed rates
- Configurable and very flexible. Enables a wide range of frame formats with varying protocols and encapsulation to be supported
- Parsing and validation of frame and packet headers with hard-coded parser functions for the known and stable protocols
- Hardwired parsing capabilities can be supplemented with user programmed parse functions to support protocols not supported by the hardwired functionality including proprietary protocols and shim headers inserted between otherwise well known protocols.
- Up to 256 bytes of the incoming frame can be parsed.
- Can start parsing from any header/layer, in any offset, from the beginning of the frame
- Supports of input ports
  - Supports programmable parse tree per port
  - Each port configuration is in a separate 4 KB partition.
- Indication for parse exceeding 256 bytes
- Parser next invoked action (NIA) output:
  - At start, gets default hard parser NIA (HPNIA)
  - Can change the Parser NIA during the parse stages by soft parser
- Can update the Internal Frame Context (first 64 bytes) located in FMan Memory
  - Frame descriptor (FD) (16 bytes)
  - Action descriptor (AD) (8 bytes)
  - Hard Parser NIA (4 bytes)
  - Parse result (32 bytes)

- Time Stamp (4 bytes)
- The 32-byte Parse Result includes the following:
  - Headers results
  - Headers offsets
  - Classification Plan Id
  - 32-bit Line-up Confirmation Vector
  - Logical Port ID
  - Gross frame checksum (input), TCP like checksum result
  - Next Header
- Supports short packet padding removal for the purpose of L4 checksum validation.
- Parser cycle limit—Stop frame parsing after configurable # of parsing cycles
- Hard parser supported header formats
  - L2 headers
    - Ethernet II
    - IEEE 802.3/SNAP
    - VLAN
      - Saves 2 header offsets to each VLAN Tag Control Information (TCI) and updates Layer 2 result vector. Can support more than 2 VLANs then only the first VLAN TCI and the last VLAN TCI offsets are saved.
    - PPPoE+PPP
    - MPLS
      - Saves 2 header offsets to each MPLS label and updates Layer 2 result vector. Can support more than 2 MPLS labels then only the first label and the last label offsets are saved.
  - L3 headers
    - IPv4
    - IPv6
    - Tunnelled IPv4/IPv6
    - GRE version 0
    - Min Encap
    - IPSec
  - Supports parsing L4 headers: TCP/UDP/SCTP/DCCP
  - User configurable via soft parser.

### 5.9.3 Parser Memory Map/Register Definition

The memory map is divided into two parts:

- The first part is composed of 4 KB regions arranged according to their the port ID numbering where the region is valid only for valid RX/Offline parsing port IDs.

- The second part is a 4 KB parser global configuration region. This region uses the first 2 KB for global parser memory configuration with the remainder of that space (2 KB) available for register configuration.

[Table 5-282](#) provides the overall memory map and [Table 5-283](#) provides a specific break down of the regions. See [Section 5.4.2, “FMan Detailed Memory Map,”](#) for details of Parser memory space within FMan memory space.

**Table 5-282. Parser Overall Memory Map**

Offset	Registers
0xn000–0xn3FF	Port $n^1$ configuration $n = 1–63$
Offset <sup>2</sup>	—
0x000–0xFFFF	Parser global configuration

<sup>1</sup> Valid only if  $n$  equals Rx and Offline parsing/Host commands port IDs.

<sup>2</sup> The base for this region is defined in [Section 5.4.2, “FMan Detailed Memory Map,”](#) soft parser for details, offset is added to that base

This table provides the memory map for each register within the parser.

**Table 5-283. Parser Individual Register Memory Map**

Offset	Register	Access	Reset Value	Section/Page
<b>Port <math>n^1</math> configuration</b>				
0xn000–0xn07C	Port 1 Parse Memory direct access (128-byte)	R/W	—	<a href="#">5.9.3.1.1/5-341</a>
0xn080–0xn03F7	—	—	—	—
0xn3F8	FMPR_PxCAC—Port x Configuration Access Control	Mixed	0x0000_0000	<a href="#">/5-342</a>
0xn3FC	FMPR_PxCTPID—Port x Configured TPID	R/W	0x9100_9100	<a href="#">/5-343</a>
<b>Parser Global Configuration Registers (see soft parser base in FM memory map, offset added to the base)</b>				
0x000–0x7FF	Parse Memory Read/Write direct access (2 Kbytes)	R/W	—	<a href="#">5.9.3.1.2/5-343</a>
0x800	FMPR_SXPAW0—Soft Examination Parameter Array W0	R/W	0x0000_0000	<a href="#">/5-343</a>
0x804	FMPR_SXPAW1—Soft Examination Parameter Array W1	R/W	0x0000_0000	<a href="#">/5-343</a>
0x808	FMPR_SXPAW2—Soft Examination Parameter Array W2	R/W	0x0000_0000	<a href="#">/5-343</a>
0x80C	FMPR_SXPAW3—Soft Examination Parameter Array W3	R/W	0x0000_0000	<a href="#">/5-343</a>
0x810	FMPR_SXPAW4—Soft Examination Parameter Array W4	R/W	0x0000_0000	<a href="#">/5-343</a>
0x814	FMPR_SXPAW5—Soft Examination Parameter Array W5	R/W	0x0000_0000	<a href="#">/5-343</a>
0x818	FMPR_SXPAW6—Soft Examination Parameter Array W6	R/W	0x0000_0000	<a href="#">/5-343</a>
0x81C	FMPR_SXPAW7—Soft Examination Parameter Array W7	R/W	0x0000_0000	<a href="#">/5-343</a>

**Table 5-283. Parser Individual Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x820	FMPR_SXPAW8—Soft Examination Parameter Array W8	R/W	0x0000_0000	<a href="#">/5-343</a>
0x824	FMPR_SXPAW9—Soft Examination Parameter Array W9	R/W	0x0000_0000	<a href="#">/5-343</a>
0x828	FMPR_SXPAW10—Soft Examination Parameter Array W10	R/W	0x0000_0000	<a href="#">/5-343</a>
0x82C	FMPR_SXPAW11—Soft Examination Parameter Array W11	R/W	0x0000_0000	<a href="#">/5-343</a>
0x830	FMPR_SXPAW12—Soft Examination Parameter Array W12	R/W	0x0000_0000	<a href="#">/5-343</a>
0x834	FMPR_SXPAW13—Soft Examination Parameter Array W13	R/W	0x0000_0000	<a href="#">/5-343</a>
0x838	FMPR_SXPAW14—Soft Examination Parameter Array W14	R/W	0x0000_0000	<a href="#">/5-343</a>
0x83C	FMPR_SXPAW15—Soft Examination Parameter Array W15	R/W	0x0000_0000	<a href="#">/5-343</a>
0x840	FMPR_RPCLIM—Rx Parsing Cycle Limit	R/W	0x0000_0000	<a href="#">/5-344</a>
0x844	FMPR_RPIMAC—Parse Internal memory access Control	Mixed	0x0000_0000	<a href="#">/5-344</a>
0x848	FMPR_PMEEC—Parse Memory ECC Error Capture Register	RR	0x0000_0000	<a href="#">/5-345</a>
0x84C–0x85F	Reserved	—	—	—
0x860	FMPR_PEVR—Parser Event Register	w1c	0x0000_0000	<a href="#">/5-346</a>
0x864	FMPR_PEVER—Parser Event Enable Register	R/W	0x0000_0000	<a href="#">/5-347</a>
0x86C	FMPR_PERR—Parser Error Register	w1c	0x0000_0000	<a href="#">/5-348</a>
0x870	FMPR_PERER—Parser Error Enable Register	R/W	0x0000_0000	<a href="#">/5-348</a>
0x878–0x89F	Reserved	—	—	—
0x8A0	FMPR_PPSC—per Port Parse Statistic Control Register	R/W	All zeros	<a href="#">/5-349</a>
0x8A4	Reserved	—	—	—
0x8A8	FMPR_PDS—Parse Dispatch Statistic Register	R/W	0x0000_0000	<a href="#">/5-350</a>
0x8AC	FMPR_L2RRS—L2 Result Returned Statistic Register	R/W	0x0000_0000	<a href="#">/5-350</a>
0x8B0	FMPR_L3RRS—L3 Result Returned Statistic Register	R/W	0x0000_0000	<a href="#">/5-351</a>
0x8B4	FMPR_L4RRS—L4 Result Returned Statistic Register	R/W	0x0000_0000	<a href="#">/5-351</a>
0x8B8	FMPR_SRRES—Shim Result Returned Statistic Register	R/W	0x0000_0000	<a href="#">/5-351</a>
0x8BC	FMPR_L2RRES—L2 Result Returned with Error Statistic Register	R/W	0x0000_0000	<a href="#">/5-352</a>
0x8C0	FMPR_L3RRES—L3 Result Returned with Error Statistic Register	R/W	0x0000_0000	<a href="#">/5-352</a>
0x8C4	FMPR_L4RRES—L4 Result Returned with Error Statistic Register	R/W	0x0000_0000	<a href="#">/5-352</a>
0x8C8	FMPR_SRRES—Shim Result Returned with Error Statistic Register	R/W	0x0000_0000	<a href="#">/5-353</a>
0x8CC	FMPR_SPCS—Soft Parser Cycle Statistic Register	R/W	0x0000_0000	<a href="#">/5-353</a>
0x8D0	FMPR_SPSCS—Soft Parser Stall Cycle Statistic Register	R/W	0x0000_0000	<a href="#">/5-353</a>
0x8D4	FMPR_HXSCS—HXS Cycle Statistic Register	R/W	0x0000_0000	<a href="#">/5-354</a>

**Table 5-283. Parser Individual Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x8D8	FMPR_MRCS—FMan Memory Read Cycle Statistic Register	R/W	0x0000_0000	<a href="#">/5-354</a>
0x8DC	FMPR_MWCS—FMan Memory Write Cycle Statistic Register	R/W	0x0000_0000	<a href="#">/5-354</a>
0x8E0	FMPR_MRSCS—FMan Memory Read Stall Cycle Statistic Register	R/W	0x0000_0000	<a href="#">/5-355</a>
0x8E4	FMPR_MWSCS—FMan Memory Write Stall Cycle Statistic Register	R/W	0x0000_0000	<a href="#">/5-355</a>
0x8E8	FMPR_FCSCS—FPM Command Stall Cycle Statistic Register	R/W	0x0000_0000	<a href="#">/5-355</a>
0x8EC	FMPR_PDATES—Parser Debug Flow A Trap Event Statistic Register	R/W	0x0000_0000	<a href="#">/5-356</a>
0x8F0	FMPR_PDBTES—Parser Debug Flow B Trap Event Statistic Register	R/W	0x0000_0000	<a href="#">/5-356</a>
0x8F4	FMPR_PDCTES—Parser Debug Flow C Trap Event Statistic Register	R/W	0x0000_0000	<a href="#">/5-356</a>
0x8F8–0x8FF	Reserved	—	—	—
0x900	FMPR_PDC—Parser Debug Control Register	R/W	All zeros	<a href="#">/5-356</a>
0x904	FMPR_PDAT0C—Parser Debug Flow A Trap 0 Configuration Register	R/W	All zeros	<a href="#">/5-357</a>
0x908	FMPR_PDAT0V—Parser Debug Flow A Trap 0 Value Register	R/W	All zeros	<a href="#">/5-358</a>
0x90C	FMPR_PDAT0M—Parser Debug Flow A Trap 0 Mask Register	R/W	All zeros	<a href="#">/5-359</a>
0x910	FMPR_PDAT1C—Parser Debug Flow A Trap 1 Configuration Register	R/W	All zeros	<a href="#">/5-357</a>
0x914	FMPR_PDAT1V—Parser Debug Flow A Trap 1 Value Register	R/W	All zeros	<a href="#">/5-358</a>
0x918	FMPR_PDAT1M—Parser Debug Flow A Trap 1 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x91C	FMPR_PDAT2C—Parser Debug Flow A Trap 2 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>
0x920	FMPR_PDAT2V—Parser Debug Flow A Trap 2 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x924	FMPR_PDAT2M—Parser Debug Flow A Trap 2 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x928	FMPR_PDAT3C—Parser Debug Flow A Trap 3 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>
0x92C	FMPR_PDAT3V—Parser Debug Flow A Trap 3 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x930	FMPR_PDAT3M—Parser Debug Flow A Trap 3 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x934	FMPR_PDBT0C—Parser Debug Flow B Trap 0 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>
0x938	FMPR_PDBT0V—Parser Debug Flow B Trap 0 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x93C	FMPR_PDBT0M—Parser Debug Flow B Trap 0 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x940	FMPR_PDBT1C—Parser Debug Flow B Trap 1 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>
0x944	FMPR_PDBT1V—Parser Debug Flow B Trap 1 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x948	FMPR_PDBT1M—Parser Debug Flow B Trap 1 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x94C	FMPR_PDBT2C—Parser Debug Flow B Trap 2 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>
0x950	FMPR_PDBT2V—Parser Debug Flow B Trap 2 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x954	FMPR_PDBT2M—Parser Debug Flow B Trap 2 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x958	FMPR_PDBT3C—Parser Debug Flow B Trap 3 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>

**Table 5-283. Parser Individual Register Memory Map (continued)**

Offset	Register	Access	Reset Value	Section/Page
0x95C	FMPR_PDBT3V—Parser Debug Flow B Trap 3 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x960	FMPR_PDBT3M—Parser Debug Flow B Trap 3 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x964	FMPR_PDCT0C—Parser Debug Flow C Trap 0 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>
0x968	FMPR_PDCT0V—Parser Debug Flow C Trap 0 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x96C	FMPR_PDCT0M—Parser Debug Flow C Trap 0 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x970	FMPR_PDCT1C—Parser Debug Flow C Trap 1 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>
0x974	FMPR_PDCT1V—Parser Debug Flow C Trap 1 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x978	FMPR_PDCT1M—Parser Debug Flow C Trap 1 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x97C	FMPR_PDCT2C—Parser Debug Flow C Trap 2 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>
0x980	FMPR_PDCT2V—Parser Debug Flow C Trap 2 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x984	FMPR_PDCT2M—Parser Debug Flow C Trap 2 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x988	FMPR_PDCT3C—Parser Debug Flow C Trap 3 Configuration Register	R/W	0x0000_0000	<a href="#">/5-357</a>
0x98C	FMPR_PDCT3V—Parser Debug Flow C Trap 3 Value Register	R/W	0x0000_0000	<a href="#">/5-358</a>
0x990	FMPR_PDCT3M—Parser Debug Flow C Trap 3 Mask Register	R/W	0x0000_0000	<a href="#">/5-359</a>
0x994–0xFFFF	Reserved	—	—	—

**Note:**

<sup>1</sup> Valid only if  $n$  equals RX and Offline parsing/Host commands port IDs

[Table 5-284](#) describes the memory map of the parse internal memory

**Table 5-284. Parse Internal Memory Map**

Regions	Offset (16-Bit Words)	Offset (Bytes)	0–15	16–31	32–47	48–63
<sup>1</sup> HXS address (64 bytes)	0x0	0x0	Ethernet HXS	LLC+SNAP HXS	VLAN HXS	PPPoE+PPP HXS
	0x4	0x8	MPLS HXS	IPv4 HXS	IPv6 HXS	GRE HXS
	0x8	0x10	MinEncap HXS	Other L3 Shell HXS	TCP HXS	UDP HXS
	0xC	0x18	IPSec HXS	SCTP HXS	DCCP HXS	Other L4 Shell HXS
	0x10–0x1C	0x20–0x38	Reserved			
SoftParse Instructions	0x20	0x40	Parse program space (1984 bytes)			
	...	...	<b>Note:</b> Addrs 0x3fe and 0x3ff are not to be used as normal instructions, and must be zeroed for proper operation			
	0x3FC	0x7F8				

**Table 5-284. Parse Internal Memory Map (continued)**

Regions	Offset (16-Bit Words)	Offset (Bytes)	0–15	16–31	32–47	48–63
Port 1–16 Header Examination Configuration (2048 bytes)	0x400–0x43C	0x800–0x878	Port 16 Header Examination Config; see <a href="#">Table 5-285</a> for details.			
	0x440–0x47C	0x880–0x8F8	Port 1 Header Examination Config			
	...	...		—		
	0x7C0–0x7FC	0xF80–0xFF8	Port 15 Header Examination Config			

**Note:**

<sup>1</sup> The actual 64 bytes of physical memory is unused.

### 5.9.3.1 Parser Register Descriptions

The following sections provide detailed descriptions of the parser internal registers.

The parser can be configured on a per-port basis. The valid configuration is only for valid Rx and Offline parsing port IDs. For non-valid regions, writes have no effect and reads return zeros.

#### 5.9.3.1.1 Port x Parse Memory Direct Access Registers

This 128 bytes of the 1-Kbyte CCSR space provides read/write access to the “Port x Header Examination Config” in the parse internal memory (see [Table 5-285](#)) for the associated port (128 bytes).

**Table 5-285. Port x Header Examination Configuration**

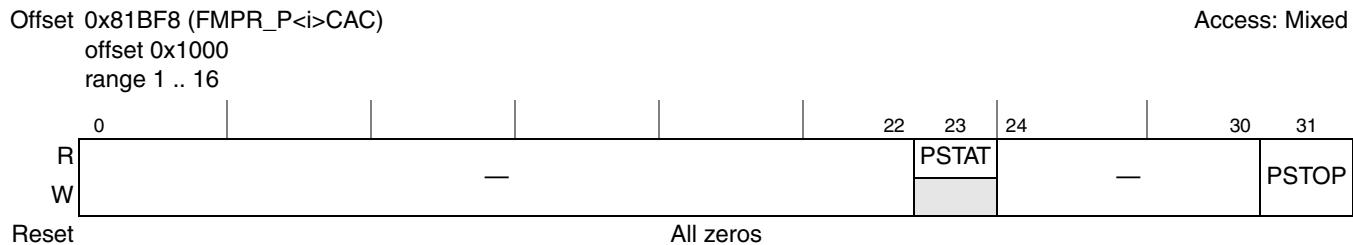
Offset (Bytes)	0–15	16–31	32–47	48–63
0x0	Ethernet HXS—Soft Sequence Attachment (see <a href="#">Table 5-323</a> )		Ethernet HXS—Line-up Enable Confirmation Mask (see <a href="#">Table 5-332</a> )	
0x8	LLC+SNAP HXS—Soft Sequence Attachment		LLC+SNAP HXS—Line-up Enable Confirmation Mask	
0x10	VLAN HXS—Soft Sequence Attachment		VLAN HXS—Line-up Enable Confirmation Mask	
0x18	PPPoE+PPP HXS—Soft Sequence Attachment		PPPoE+PPP HXS—Line-up Enable Confirmation Mask	
0x20	MPLS HXS—Soft Sequence Attachment		MPLS HXS—Line-up Enable Confirmation Mask	
0x28	IPv4 HXS—Soft Sequence Attachment		IPv4 HXS—Line-up Enable Confirmation Mask	
0x30	IPv6 HXS—Soft Sequence Attachment		IPv6 HXS—Line-up Enable Confirmation Mask	
0x38	GRE HXS—Soft Sequence Attachment		GRE HXS—Line-up Enable Confirmation Mask	
0x40	MinEncap HXS—Soft Sequence Attachment		MinEncap HXS—Line-up Enable Confirmation Mask	
0x48	Other L3 Shell HXS—Soft Sequence Attachment		Other L3 shell HXS—Line-up Enable Confirmation Mask	
0x50	TCP HXS—Soft Sequence Attachment		TCP HXS—Line-up Enable Confirmation Mask	
0x58	UDP HXS—Soft Sequence Attachment		UDP HXS—Line-up Enable Confirmation Mask	

**Table 5-285. Port x Header Examination Configuration (continued)**

Offset (Bytes)	0–15	16–31	32–47	48–63
0x60	IPSec HXS—Soft Sequence Attachment		IPSec HXS—Line-up Enable Confirmation Mask	
0x68	SCTP HXS—Soft Sequence Attachment		SCTP HXS—Line-up Enable Confirmation Mask	
0x70	DCCP HXS—Soft Sequence Attachment		DCCP HXS—Line-up Enable Confirmation Mask	
0x78	Other L4 Shell HXS—Soft Sequence Attachment		Other L4 Shell HXS—Line-up Enable Confirmation Mask	

### Port x Configuration Access Control Register (FMPR\_PxCAC)

The FMPR\_PxCAC controls access to the parse internal memory for the purpose of updating the per port configuration data (“Port x Header Examination Config”). To update the per-port portion of the parse internal memory, the parser for that port should be stopped (or put into the Idle state). This is achieved by setting the FMPR\_PxCAC[PSTOP] bit to 1. Once in the Idle state, which is indicated by the FMPR\_PxCAC[PSTAT] bit being set to zero (Idle), the portion of the parse internal memory specific to that port can be modified. The port must also be put in Idle state (achieved as described above) if its Statistic Enable bit in the Per Port Parser Statistic Control Register (FMPR\_PPSC) is being changed.



**Figure 5-262. Port x Configuration Access Control Register (FMPR\_PxCAC)**

**Table 5-286. FMPR\_PxCAC Field Descriptions**

Bits	Name	Description
0–22	—	Reserved
12	PSTAT	Parser status port x 0 Parser is Idle for port x 1 Parser is Active for port x
24–30	—	Reserved
31	PSTOP	Parser stop port x When asserted, the parser stops parsing and bypasses new work from FPM dispatch if a frame from port x is received.

## Port x Configured TPID Register (FMPR\_PxCTPID)

The FMPR\_PxCTPID configures one or two distinct Ethertype values (or TPID values) to indicate a VLAN tag in addition to the TPID values 0x8100 and 0x88A8.

**Figure 5-263. Port x Configured TPID Register (FMPR\_PxCTPID)**

**Table 5-287. FMPR\_PxCTPID Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–15	ConfigTPID1	Configured TPID 1
16–31	ConfigTPID2	Configured TPID 2

### **5.9.3.1.2 Parse Memory Read/Write Direct Access Registers—Global Configuration**

The first 2 Kbytes of the parser's 4-Kbyte CCSR space provides read/write access to the first 2 Kbytes of the parse internal memory (see [Table 5-284](#)). The memory write accesses should be performed only while FMPR\_RPIMAC[PSTAT] is idle. When memory is accessed via this 4-Kbyte region, there is no aliasing of the “Port x Header Examination Config.” Access to “Port x Header Examination Config” must be done via the Port x Parse Memory Direct Access Registers registers.

## Soft Examination Parameter Array Wx Register (FMPR\_SXPAWx)

The FMPR\_SXPaw<sub>x</sub> provides the ability to configure parameters used by the Soft Examinations.

Offset 0xC7800 (FMPR_SXPAW<i>)								Access: Read/Write	
offset 0x4									
range 0 .. 15									
	0		7	8		15	16		31
R	B0		B1		B2		B3		
W									
Reset	All zeros								

**Figure 5-264. Soft Examination Parameter Array Wx Register (FMPR\_SXPAWx)**

**Table 5-288. FMPR\_SXPAWx Field Descriptions**

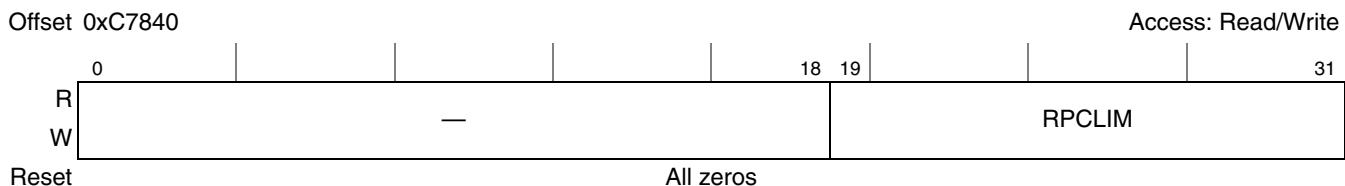
<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–7	B0	Array Byte 0 of Word x
8–15	B1	Array Byte 1 of Word x

**Table 5-288. FMPR\_SXPAWx Field Descriptions (continued)**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
16–23	B2	Array Byte 2 of Word x
24–31	B3	Array Byte 3 of Word x

## Rx Parsing Cycle Limit Register (FMPR\_RPCLIM)

The FMPR\_RPCLIM sets the maximum parse cycle limit and is used to prevent the parser getting into an endless loop when using hard and soft Header Examination Sequences (HXSs). A limit value of 0 disables this mechanism. A cycle count is initiated at the onset of parsing and counts every parse cycle until parsing for this frame completes or the count exceeds the limit set in the RPCLIM. If this count exceeds the RPCLIM then the parser ceases parsing the frame and reports the error as an exception notification. The parser does not halt processing of frames when this error condition is detected; it proceeds normally to process other frames.



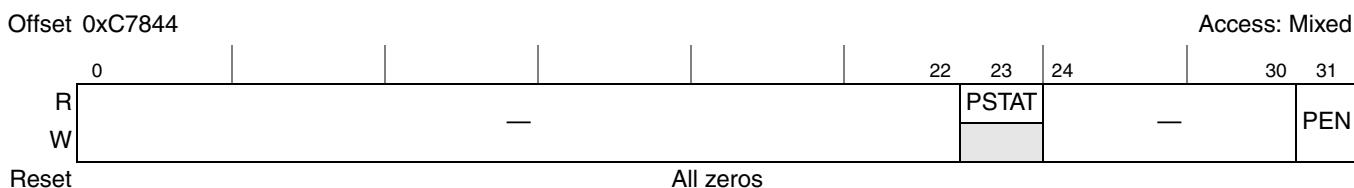
**Figure 5-265. Rx Parsing Cycle Limit Register (FMPR\_RPCLIM)**

**Table 5-289. FMPR\_RPCLIM Field Descriptions**

Bits	Name	Description
0–18	—	Reserved
19–31	RPCLIM	Maximum parse cycle limit When this value is exceeded, the parser ceases parsing on the frame and report the error as an exception notification.

## Rx Parse Internal Memory Access Control Register (FMPR\_RPIMAC)

The FMPR\_RPIMAC is used to control access to the parse internal memory for the purpose of updating any portion of this memory. To update any portion of the parse internal memory, the parser should be stopped (or put into the idle state). This is achieved by setting the FMPR\_RPIMAC[PEN] bit to zero.



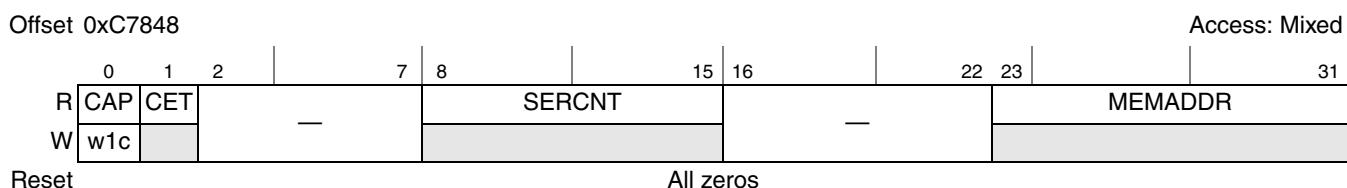
**Figure 5-266. Rx Parse Internal Memory Access Control Register (FMPR\_RPIMAC)**

**Table 5-290. FMPR\_RPIMAC Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–22	—	Reserved
23	PSTAT	Parser status 0 Parser is Idle 1 Parser is Active
24–30	—	Reserved
31	PEN	Parser enable When de-asserted, the parser stops acquiring new work from FPM dispatch. When asserted, the parser is enabled.

## **Parse Memory ECC Error Capture Register (FMPR\_PMEEC)**

The FMPR\_PMEEC counts single-bit ECC errors on the Parse internal memory, and capture the address in the Parse internal memory at which the ECC error (single-bit or multi-bit) occurred. The first ECC error (single-bit or multi-bit) that occurs locks the captured memory address of the error from any additional single-bit ECC errors. The first multi-bit ECC error that occurs overwrites any captured memory address from a single-bit ECC error and locks the captured memory address from any additional ECC errors (single-bit or multi-bit). The captured lock can be cleared to enable the capturing of memory address from additional single-bit or multi-bit ECC errors.



**Figure 5-267. Parse Memory ECC Error Capture Register (FMPR\_PMEEC)**

**Table 5-291. FMPR\_PMEEC Field Descriptions**

Bits	Name	Description
0	CAP	<p>Captured Error Indication</p> <p>0 No error captured in MEMADDR field.</p> <p>1 ECC error is captured. Any single-bit ECC errors set the CAP bit, and the errored memory location is captured in the MEMADDR field and the MEMADDR is locked from capturing a new single-bit ECC error. For multi-bit ECC errors, the CAP bit is set, the MEMADDR field holds the memory address of the errored location, and MEMADDR is locked from re-capturing any other ECC errors (single-bit or multi-bit).</p>
1	CET	<p>Captured Error Type</p> <p>0 Single-Bit ECC Error is captured if CAP = 1. No ECC error captured if CAP = 0.</p> <p>1 Multi-Bit ECC error is captured. CET is never asserted if CAP = 0.</p>
2–7	—	Reserved

**Table 5-291. FMPR\_PMEEC Field Descriptions (continued)**

Bits	Name	Description
8–15	SERCNT	<p>Soft Error Counter</p> <p>This counter accumulates single-bit ECC errors that have been detected and corrected by the ECC logic. Multi-bit ECC errors are not counted and directly set the FMPR_PERR[ECCE] error status bit. If FMPR_PERER[ECCE] = 1, then a multi-bit ECC error also asserts the parser error interrupt.</p> <p>This field is cleared on read. When the count reaches its maximum value (0xFF), it sticks at that value without rollover until this register is read.</p>
16–22	—	Reserved
23–31	MEMADDR	<p>Captured memory address in the Parse internal memory that caused the ECC error</p> <p>Valid when CAP is set. Cleared when CAP is written with a one (that is, clears the captured lock).</p>

### **5.9.3.1.3 Interrupt Registers**

## Parser Event Register (FMPR PEVR)

The parser contains one dedicated interrupt line (that is, parser interrupt) for signaling event conditions to software. The FMPR\_PEVR, shown in Figure 5-268, identifies the source of the interrupt. If an event or condition occurs that sets a bit in the FMPR\_PEVR register the parser interrupt is generated if the corresponding bit in the FMPR\_PEVER register is also set. Once a bit in the FMPR\_PEVR register is asserted, it remains asserted until the original event is cleared and the bit in the FMPR\_PEVR register is cleared. The bit in FMPR\_PEVR is cleared by writing a 1 to that bit position. Clearing the FMPR\_PEVR bit clears the parser interrupt indication as long as the original event is no longer present (the interrupt source has been cleared) and no other FMPR\_PEVR/FMPR\_PEVER bit pairs are set. A write of 0 has no effect.

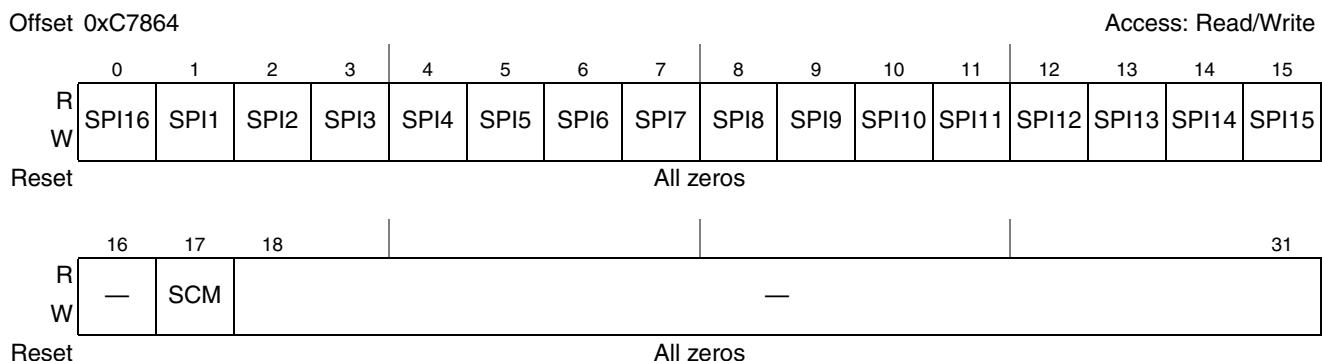
**Figure 5-268. Parser Event Register (FMPR PEVR)**

**Table 5-292. FMPR\_PEVR Field Descriptions**

Bits	Field	Description
0–11	SPI <sub>n</sub>	Stopped port <i>n</i> is now Idle A stop is requested via FMPR_PnCAC[PSTOP] and the FMPR_PnCAC[PSTAT] is now Idle. This status bit is asserted if the FMPR_PnCAC[PSTOP] and the FMPR_PnCAC[PSTAT] conditions are met (stopped and idle) and it remains latched asserted until the original event conditions are no longer present and this bit is writing to 1.
17	SCM	Single-bit ECC error counter FMPR_PMEEC[SERCNT] is at max This status bit stays asserted until the counter is read and this bit is writing to 1
18–31	—	Reserved

## Parser Event Enable Register (FMPR\_PEVER)

The FMPR\_PEVER provides control over which possible interrupt events are allowed to generate the actual Parser interrupt. If the corresponding bits in both FMPR\_PEVR and FMPR\_PEVER are set, then the parser interrupt is generated. The state of an FMPR\_PEVER bit does not affect setting of the corresponding FMPR\_PEVR bit; it controls whether or not setting of the FMPR\_PEVR bit causes an interrupt. Setting a FMPR\_PEVER bit to 0 causes the interrupt indication to be cleared if no other FMPR\_PEVR/FMPR\_PEVER bit pairs are set. All implemented bits in this register are read/write. This register is cleared upon hardware reset



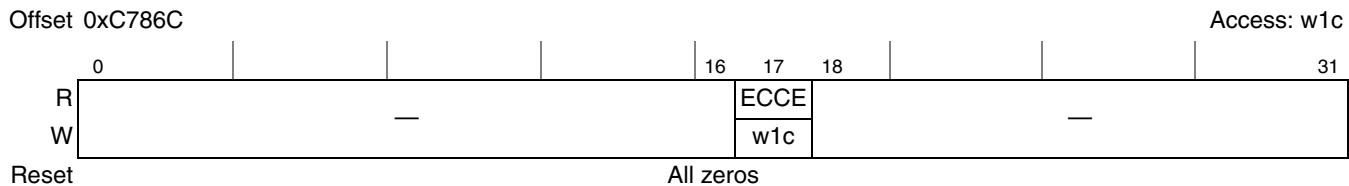
**Figure 5-269. Parser Event Enable Register (FMPR\_PEVER)**

**Table 5-293. FMPR\_PEVER Field Descriptions**

Bits	Name	Description
0–11	SPI $n$	Stopped port $n$ is now idle interrupt enable 0 Disable interrupt 1 Enable interrupt
17	SCM	Single-bit ECC error counter FMPR_PMEEC[SERCNT] is at max Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
18–31	—	Reserved

## Parser Error Register (FMPR\_PERR)

The parser contains one dedicated interrupt indication (that is, Parser error interrupt) for signaling error conditions to software. The FMPR\_PERR identifies the source of the interrupt. If an error occurs that sets a bit in the FMPR\_PERR register the Parser error interrupt is generated if the corresponding bit in the FMPR\_PERER register is also set. When a bit in the FMPR\_PERR is asserted, it remains asserted until the bit in the FMPR\_PERR is cleared. The bit in the FMPR\_PERR is cleared by writing a 1 to that bit position. Clearing the FMPR\_PERR bit clears the parser error interrupt indication as long as no other FMPR\_PERR/FMPR\_PERER bit pairs are set. A write of 0 has no effect.



**Figure 5-270. Parser Error Register (FMPR\_PERR)**

**Table 5-294. FMPR\_PERR Field Descriptions**

Bits	Name	Description
0–16	—	Reserved
17	ECCE	Parser memory ECC multiple-bit error (un-correctable) detected. The Parser continues operation but it should be reset since memory integrity is compromised (that is, it does not halt processing of frames) when this condition is detected.
18–31	—	Reserved

## Parser Error Enable Register (FMPR\_PERER)

The FMPR\_PERER provides control over which possible interrupt errors are allowed to generate the actual parser error interrupt. If the corresponding bits in both FMPR\_PERR and FMPR\_PERER are set, then the parser error interrupt is generated. The state of an FMPR\_PERER bit does not affect setting of the corresponding FMPR\_PERR bit, it controls whether or not setting of the FMPR\_PERR bit causes an interrupt. Setting a FMPR\_PERER bit to 0 causes the interrupt indication to be cleared if no other FMPR\_PERR/FMPR\_PERER bit pairs are set. All implemented bits in this register are read/write. This register is cleared upon hardware reset.



**Figure 5-271. Parser Error Enable Register (FMPR\_PERER)**

**Table 5-295. FMPR\_PERER Field Descriptions**

Bits	Name	Description
0–16	—	Reserved
17	ECCE	Parser memory ECC multiple-bit error Interrupt Enable. 0 Disable interrupt 1 Enable interrupt
18–31	—	Reserved

#### 5.9.3.1.4 Parse Statistic Registers

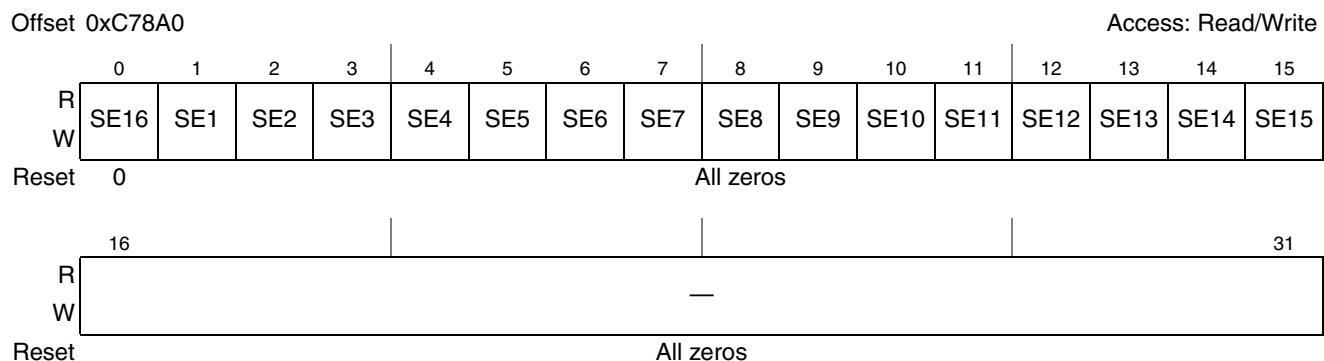
The statistic registers measure and analyze performance of the parser, with the following additional characteristics:

- Statistics counters are initialized to 0 after a reset.
- Software can write to the statistics counters in order to initialize their value.
- Software has to poll the counters and calculates the increment using the previous value of the counter.
- Statistics counters count continuously and wrap around.
- Statistics counters are 32-bit such that the frequency in which the software needs to poll the counters is no more than 1 second. That is the counter has not counted its full range within 1 second.
- No interrupt is required.

#### Per Port Parser Statistic Control Register (FMPR\_PPSC)

The FMPR\_PPSC provides control over which ports contribute to the parse statistic registers. When a frame is processed by the parser for a given port, the parser statistic increments if the port has its Statistic Enable (SE) bit set. When none of the bits are set, no statistics are incremented. One, many, or all bits can be set.

Because the collection points for the statistics are distributed along the various pipeline stages of the parser, the SE bit of a port should only be changed while the port status is idle (must stop the port if not in the Idle state, achieved). This way it provides coherency between the statistics.



**Figure 5-272. Per Port Parser Statistic Control Register (FMPR\_PPSC)**

**Table 5-296. FMPR\_PPSC Fields Description**

Bits	Name	Description
0–11	SEn	Port $n$ Statistic Enable 0 Disable Statistic 1 Enable Statistic
16–31	—	Reserved

### Parse Dispatch Statistic Register (FMPR\_PDS)

The FMPR\_PDS counts the number of times the parser block is dispatched by FPM.



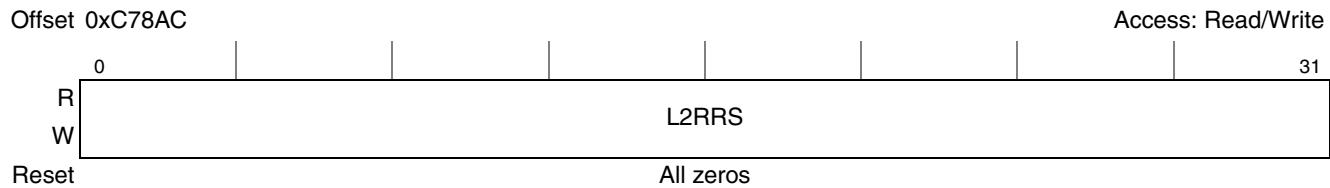
**Figure 5-273. Parse Dispatch Statistic Register (FMPR\_PDS)**

**Table 5-297. Register FMPR\_PDS Bits Description**

Bits	Name	Description
0–31	PDS	Number of times parser block is dispatched by FPM

### L2 Result Returned Statistic Register (FMPR\_L2RRS)

The FMPR\_L2RRS counts the number of times L2 parse result is returned (including with errors).



**Figure 5-274. L2 Result Returned Statistic Register (FMPR\_L2RRS)**

**Table 5-298. FMPR\_L2RRS Field Descriptions**

Bits	Name	Description
0–31	L2RRS	Number of times L2 parse result is returned

## L3 Result Returned Statistic Register (FMPR\_L3RRS)

The FMPR\_L3RRS counts the number of times L3 parse result is returned (including with errors).



Figure 5-275. L3 Result Returned Statistic Register (FMPR\_L3RRS)

## L4 Result Returned Statistic Register (FMPR\_L4RRS)

Table 5-299. FMPR\_L3RRS Field Descriptions

Bits	Name	Description
0–31	L3RRS	Number of times L3 parse result is returned

The FMPR\_L4RRS counts the number of times L4 parse result is returned (including with errors).

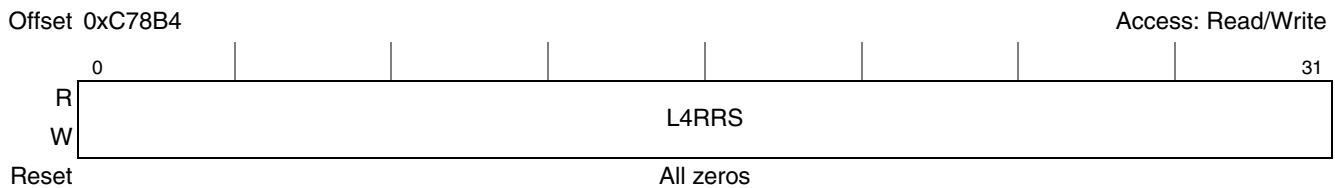


Figure 5-276. L4 Result Returned Statistic Register (FMPR\_L4RRS)

Table 5-300. FMPR\_L4RRS Field Descriptions

Bits	Name	Description
0–31	L4RRS	Number of times L4 parse result is returned

## Shim Result Returned Statistic Register (FMPR\_SRRS)

The FMPR\_SRRS counts the number of times Shim parse result is returned (including with errors).

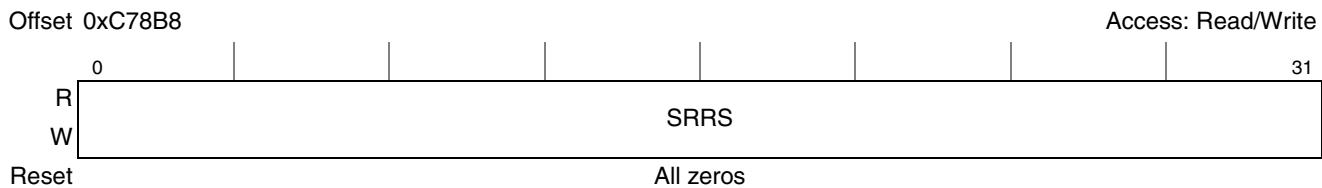


Figure 5-277. Shim Result Returned Statistic Register (FMPR\_SRRS)

Table 5-301. FMPR\_SRRS Field Descriptions

Bits	Name	Description
0–31	SRRS	Number of times Shim parse result is returned

## L2 Result Returned with Error Statistic Register (FMPR\_L2RRES)

The FMPR\_L2RRES counts the number of times L2 parse result is returned with errors.



Figure 5-278. L2 Result Returned with Error Statistic Register (FMPR\_L2RRES)

Table 5-302. FMPR\_L2RRES Field Descriptions

Bits	Name	Description
0–31	L2RRES	Number of times L2 parse result is returned with error

## L3 Result Returned with Error Statistic Register (FMPR\_L3RRES)

The FMPR\_L3RRES counts the number of times L3 parse result is returned with errors.

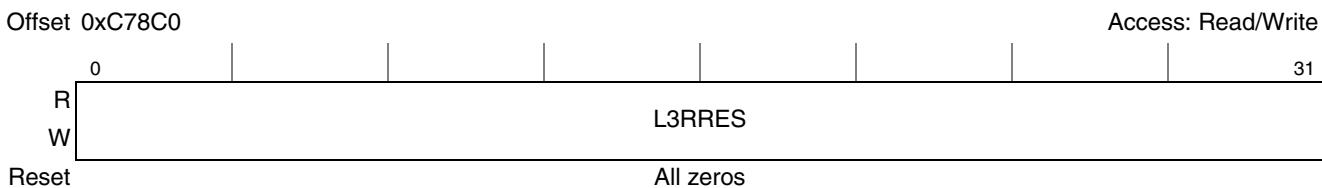


Figure 5-279. L3 Result Returned with Error Statistic Register (FMPR\_L3RRES)

Table 5-303. FMPR\_L3RRES Field Descriptions

Bits	Name	Description
0–31	L3RRES	Number of times L3 parse result is returned with error

## L4 Result Returned with Error Statistic Register (FMPR\_L4RRES)

The FMPR\_L4RRES counts the number of times L4 parse result is returned with errors.

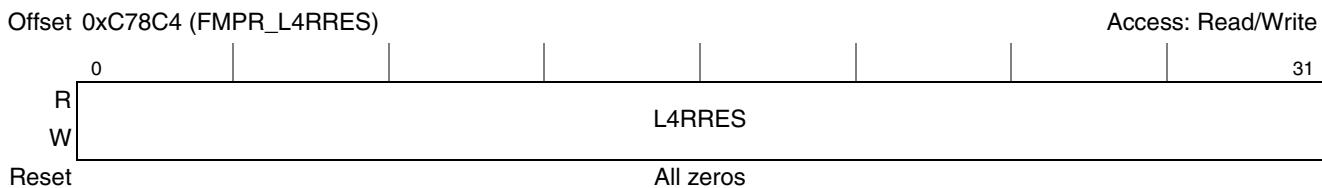


Figure 5-280. L4 Result Returned with Error Statistic Register (FMPR\_L4RRES)

Table 5-304. FMPR\_L4RRES Field Descriptions

Bits	Name	Description
0–31	L4RRES	Number of times L4 parse result is returned with error

## Shim Result Returned with Error Statistic Register (FMPR\_SRRES)

The FMPR\_SRRES counts the number of times Shim parse result is returned with errors.



Figure 5-281. Shim Result Returned with Error Statistic Register (FMPR\_SRRES)

Table 5-305. FMPR\_SRRES Field Descriptions

Bits	Name	Description
0–31	SRRES	Number of times Shim parse result is returned with error

## Soft Parser Cycle Statistic Register (FMPR\_SPCS)

The FMPR\_SPCS counts the number of cycles spent executing soft parser instruction (including stall cycles).

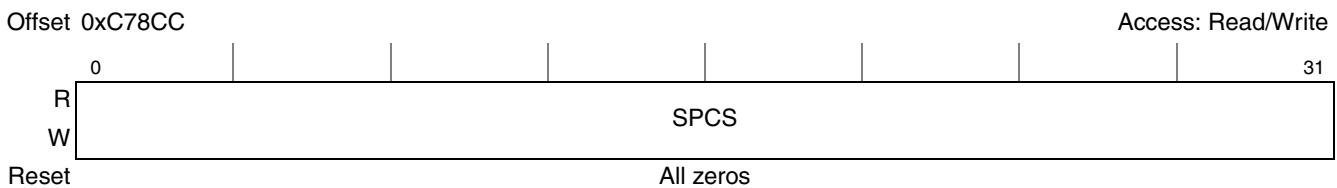


Figure 5-282. Soft Parser Cycle Statistic Register (FMPR\_SPCS)

Table 5-306. FMPR\_SPCS Field Descriptions

Bits	Name	Description
0–31	SPCS	Number of cycles spent executing soft parser instruction

## Soft Parser Stall Cycle Statistic Register (FMPR\_SPSCS)

The FMPR\_SPSCS counts the number of cycles stalled waiting for parser internal memory reads while executing soft parser instruction.

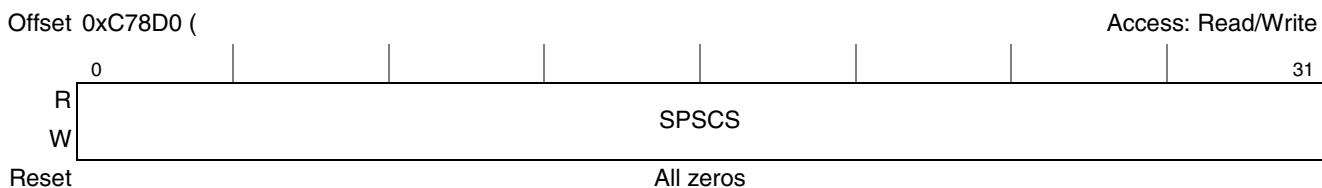


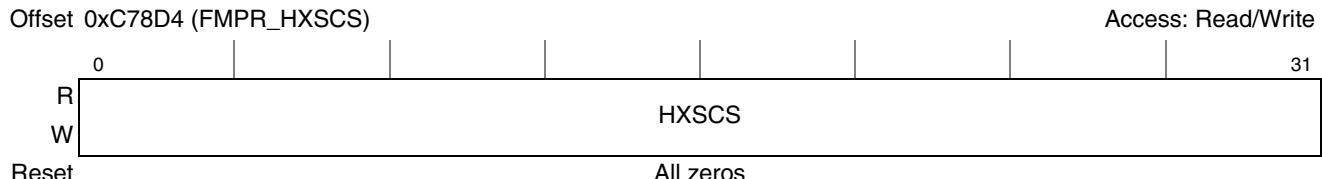
Figure 5-283. Soft Parser Stall Cycle Statistic Register (FMPR\_SPSCS)

**Table 5-307. FMPR\_SPSCS Field Descriptions**

Bits	Name	Description
0–31	SPSCS	Number of cycles stalled while executing soft parser instruction

### HXS Cycle Statistic Register (FMPR\_HXSCS)

The FMPR\_HXSCS counts the number of cycles spent executing hard parser (including stall cycles).



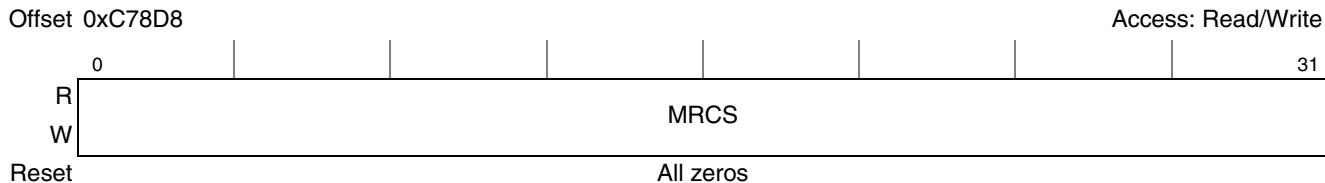
**Figure 5-284. HXS Cycle Statistic Register (FMPR\_HXSCS)**

**Table 5-308. FMPR\_HXSCS Descriptions**

Bits	Name	Description
0–31	HXSCS	Number of cycles spent executing hard parser

### FMan Memory Read Cycle Statistic Register (FMPR\_MRCS)

The FMPR\_MRCS counts the number of cycles while performing FMan Memory read.



**Figure 5-285. FMan Memory Read Cycle Statistic Register (FMPR\_MRCS)**

**Table 5-309. FMPR\_MRCS Field Descriptions**

Bits	Name	Description
0–31	MRCS	Number of FMan Memory read address cycles (including rwait)

### FMan Memory Write Cycle Statistic Register (FMPR\_MWCS)

The FMPR\_MWCS counts the number of cycles while performing FMan Memory write.



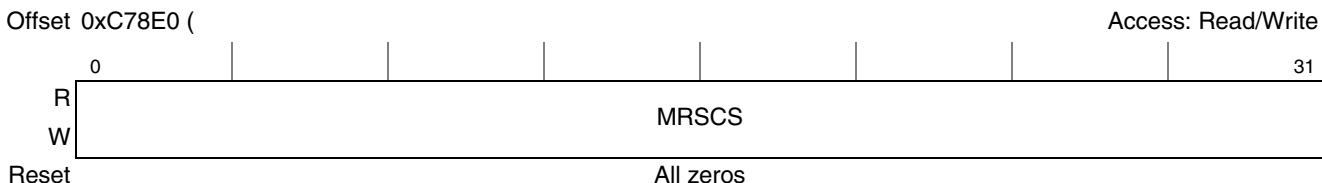
**Figure 5-286. FMan Memory Write Cycle Statistic Register (FMPR\_MWCS)**

**Table 5-310. FMPR\_MWCS Field Descriptions**

Bits	Name	Description
0–31	MWCS	Number of FMan Memory write address cycles (including rwait)

## FMan Memory Read Stall Cycle Statistic Register (FMPR\_MRSCS)

The FMPR\_MRSCS counts the number of cycles stalled while performing FMan Memory read.



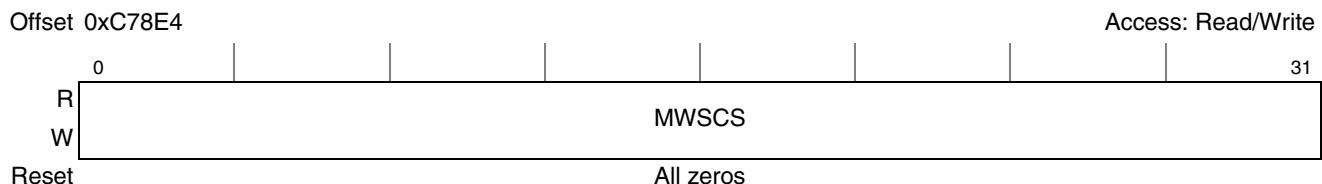
**Figure 5-287. FMan Memory Read Stall Cycle Statistic Register Format**

**Table 5-311. FMPR\_MRSCS Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	MRSCS	Number of cycles stalled while performing FMan Memory read (dwait)

## FMan Memory Write Stalled Cycle Statistic Register (FMPR\_MWSCS)

The FMPR\_MRSCS counts the number of cycles stalled while performing FMan Memory write.



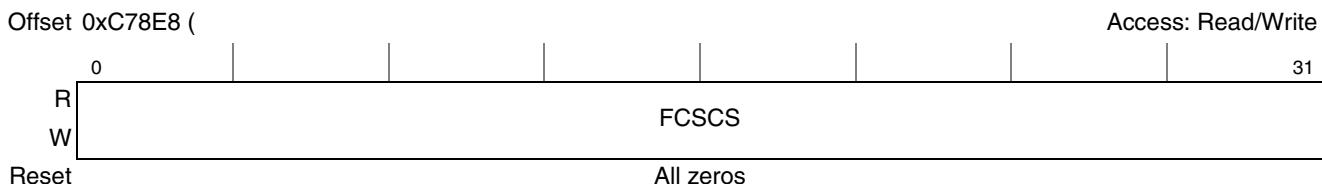
**Figure 5-288. FMan Memory Write Stalled Cycle Statistic Register (FMPR\_MWSCS)**

**Table 5-312. FMPR\_MWSCS Field Descriptions**

Bits	Name	Description
0–31	MWSCS	Number of cycles stalled while performing FMan Memory write (dwait)

## FPM Command Stall Cycle Statistic Register (FMPR\_FCSCS)

The FMPR\_FCSGS counts the number of cycles stalled while performing a FPM command.



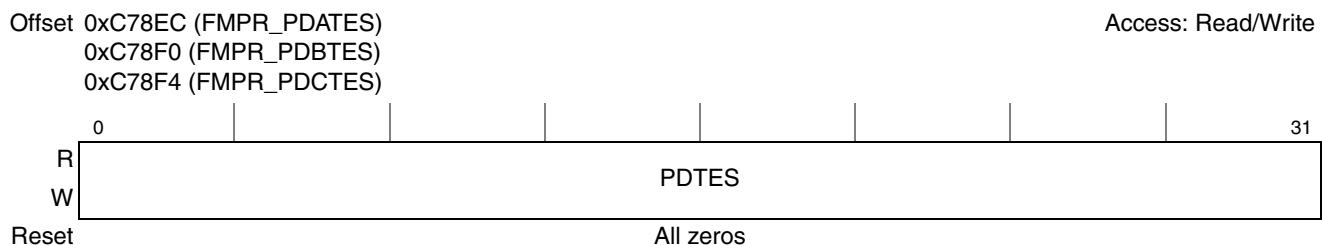
**Figure 5-289. FPM Command Stall Cycle Statistic Register (FMPR\_FCSCS)**

**Table 5-313. FMPR\_FCSCS Field Descriptions**

Bits	Name	Description
0–31	FCSCS	Number of cycles stalled while performing a FPM command

### Parser Debug Flow x Trap Event Statistic Registers (FMPR\_PDxTES)

The FMPR\_PDxTES count the number of frames with trap event(s) on debug flow  $x$ . The counter increments by one when a frame causes one or multiple trap event(s) for the given flow within the Parser.



**Figure 5-290. Parser Debug Flow x Trap Event Statistic Registers (FMPR\_PDxTES)**

**Table 5-314. FMPR\_PDxTES Field Descriptions**

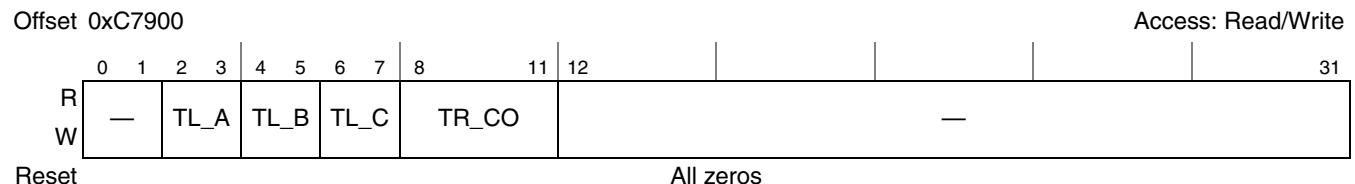
Bits	Name	Description
0–31	PDTES	Number of times a frame caused one or multiple trap event on debug flow

#### 5.9.3.1.5 Parser Debug Registers

The parser debug registers provide global control over the debug and per-flow trap registers. There are three flows: A, B and C. Each flow has four trap registers in a set, and each of these sets has a configuration, value, and a mask.

### Parser Debug Control Register (FMPR\_PDC)

The FMPR\_PDC configures the per-flow trace level and the linkage between the trap registers of the three debug flows.



**Figure 5-291. Parser Debug Control Register (FMPR\_PDC)**

**Table 5-315. FMPR\_PDC Field Descriptions**

Bits	Name	Description
0–1	—	Reserved
2–3	TL_A	Trace level for flow A TL_A selects the trace information that is dumped to the debug area of the frame's internal context. 00 Trace disabled 01 Minimum trace 10 Verbose trace 11 Very verbose trace
4–5	TL_B	Trace level for flow B TL_B selects the trace information that is dumped to the debug area of the frame's internal context. 00 Trace disabled 01 Minimum trace 10 Verbose trace 11 Very verbose trace
6–7	TL_C	Trace level for flow C TL_C selects the trace information that is dumped to the debug area of the frame's internal context. 00 Trace disabled 01 Minimum trace 10 Verbose trace 11 Very verbose trace
8–11	TR_CO	Trap collaboration When fewer debug flows are needed, trap registers for unused debug flows can be chained to flow A. Chained traps would then act as one trap entity. 0000 No collaboration between trap registers 0001 Flow A chained to Flow B trap registers, flow B is in bypass 0010 Flow A chained to Flow C trap registers, flow C is in bypass 0011 Flow A chained to Flow B chained to Flow C, flow B and C are in bypass
12–31	—	Reserved

### Parser Debug Flow x Trap y Configuration Registers (FMPR\_PDxTyC)

The FMPR\_PDxTyC configure trap y for flow x.

Offset 0xC7904 (FMPR_PDAT0C)	0xC794C (FMPR_PDBT2C)	Access: Read/Write
0xC7910 (FMPR_PDAT1C)	0xC7958 (FMPR_PDBT3C)	
0xC791C (FMPR_PDAT2C)	0xC7964 (FMPR_PDCT0C)	
0xC7928 (FMPR_PDAT3C)	0xC7970 (FMPR_PDCT1C)	
0xC7934 (FMPR_PDBT0C)	0xC797C (FMPR_PDCT2C)	
0xC7940 (FMPR_PDBT1C)	0xC7988 (FMPR_PDCT3C)	

0	2	3	4	9	10	15	16	31
R	CMPOP	AND	—	FSEL	—	—	—	
W	All zeros							

Reset

**Figure 5-292. Parser Debug Flow x Trap y Configuration Registers (FMPR\_PDxTyC)**

**Table 5-316. FMPR\_PDxTyC Field Descriptions**

Bits	Name	Description
0–2	CMPOP	Compare operator 000 Trap disabled (never matches) 001 Always match (referred as trap bypass) 010 Match if (selected field & MASK) equals value 011 Match if (selected field & MASK) does not equal value 100 Match if (selected field & MASK) is greater than value 101 Match if (selected field & MASK) is less than or equal to value 110 Match if (selected field & MASK) is less than value 111 Match if (selected field & MASK) is greater than or equal to value
3	AND	AND AND combines trap results into a trap equation. By default, it is an OR; if all next traps are disabled (or this is the last trap), they have no effect on the combined match result. The combined match evaluation is done in ascending order from trap number $n$ to trap number $n + 1$ , and without precedence of AND over OR. For example, TRAP1 or TRAP2 and TRAP3 result in an equation equivalent to (TRAP1    TRAP2) && TRAP3. 0 OR between this trap result and the next trap result 1 AND between this trap result and the next trap result
4–9	—	Reserved
10–15	FSEL	Field selection The selected field is compared to the value. 0: Parse Array bytes 0–3 1: Parse Array bytes 4–7 2: Parse Array bytes 8–11 3: Parse Array bytes 12–15 4–30: ... 31: Parse Array bytes 124–127 32: Port ID[0:5] concatenated with program counter {0x0000, port_id, program_counter} 33: LCV 34–63: Reserved
16–31	—	Reserved

### Parser Debug Flow x Trap y Value Registers (FMPR\_PDxTyV)

The FMPR\_PDxTyV contain the value against which the selected field in the debug trap configuration register is compared to.

Offset 0xC7908 (FMPR_PDAT0V)	0xC7950 (FMPR_PDBT2V)	Access: Read/Write
0xC7914 (FMPR_PDAT1V)	0xC795C (FMPR_PDBT3V)	
0xC7920 (FMPR_PDAT2V)	0xC7968 (FMPR_PDCT0V)	
0xC792C (FMPR_PDAT3V)	0xC7974 (FMPR_PDCT1V)	
0xC7938 (FMPR_PDBT0V)	0xC7980 (FMPR_PDCT2V)	
0xC7944 (FMPR_PDBT1V)	0xC798C (FMPR_PDCT3V)	
0		
R	VAL	31
W	All zeros	
Reset		

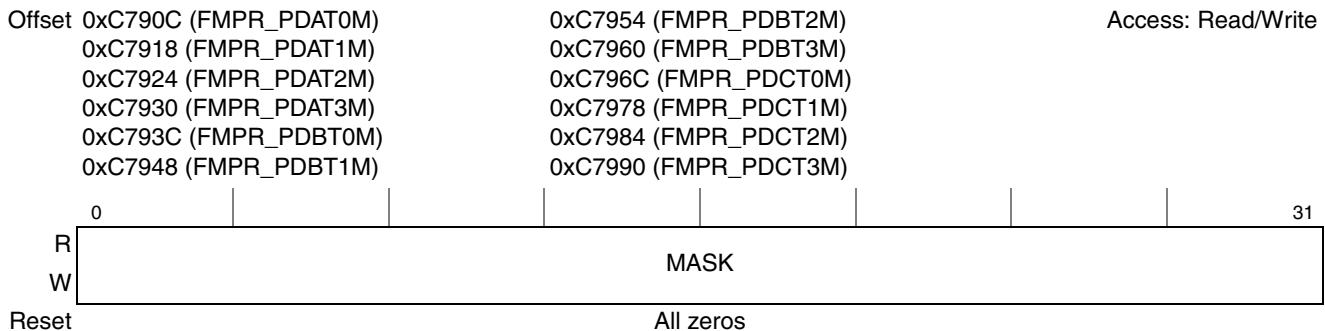
**Figure 5-293. Parser Debug Flow x Trap y Value Registers (FMPR\_PDxTyV)**

## Table 5-317. FMPR\_PDxTyV Field Descriptions

Bits	Name	Description
0–31	VAL	Comparison value

## Parser Debug Flow x Trap y Mask Registers (FMPR\_PDxTyM)

The FMPR\_PDxTyM contain the mask that is ANDed bit-wise with the selected value for comparison, the result of which is compared to the debug value register.



**Figure 5-294. Parser Debug Flow x Trap y Mask Registers (FMPR\_PDxTyM)**

### Table 5-318. FMPR\_PDxTyM Field Descriptions

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	Mask	Mask

## 5.9.4 Parser Functional Description

The parsing function enables a wide range of frame formats with varying protocols and encapsulation to be supported, including the capability to include headers (for example, system header/user-defined shim) that are proprietary. A hard-coded parser function is used for the known and stable protocols. The hardwired parsing capabilities can be supplemented with user programmed parse functions (soft parser) to support protocols not supported by the hardwired functionality including proprietary protocols and shim headers inserted between otherwise well known protocols.

#### **5.9.4.1 Benefits of Using Hard Header Examination Sequences (HXSSs)**

The hard-coded parser function is implemented through hard header examination sequences (HXSs).

Each hard HXS does the following:

- Works within a protocol layer presuming a certain type of header (variants within the header may be supported)
  - Confirms the validity of the header (to degree possible using information such as checksums from header)
  - Extracts values that may be used by the soft examination sequences (soft parser)
  - Identifies boundaries of the header

- Determines the next hard HXS to examine beyond the current header

The set of hard HXSs and their ability to invoke each other effectively implements a parse tree that supports a multitude of networks scenarios involving common Layer 2-4 protocols.

See [Section 5.9.4.7, “Hard Header Examination Sequences \(HXSs\),”](#) for a detailed description.

### **5.9.4.2 Benefits of Using Soft Examination Sequences**

The user-programmed parse functions are implemented through soft examination sequences built from examine instructions pre-positioned in a dedicated internal RAM (that is, Parse internal memory) to the parser.

One option for soft examination sequences is that they can be used to integrate additional protocol headers types (that perhaps are newly defined, obscure, or proprietary) into the parse tree. In that situation, these soft examination sequences are referred as soft header examination sequences (HXSs).

Soft examination sequences can also be used to replace or augment specific hard HXSs to support proprietary/new variants. These soft examination sequences are referred as “soft extensions”.

### **5.9.4.3 Parse Array Functionality Overview**

Parsing of a frame produces an array of extracted values and derived information from each frame that is the input to the downstream stage (for example, frame classification). While the parser is executing, it operates on a 128-byte internal array (to the parser) called the Parse Array. See [Section 5.9.4.6, “Parse Array,”](#) for a detailed description of the Parse Array. When parsing is completed, a 32-byte Parse Result is produced from a subset of the Parse Array. The Parse Result is located in the FMan Memory. See [Section 5.9.4.10, “Parse Result,”](#) for more information.

### **5.9.4.4 Parser Inputs/Outputs**

The parser is notified of complete received frames to process from the FPM. Most of the information required to parse the frame is retrieved from the FMan Memory. Once parsing is completed, the frame parse results are written to the FMan Memory and an NIA is issued to the FPM to dispatch the next processing module.

#### **5.9.4.4.1 Possible Parser NIA Action Codes**

The NIA action code provided to the parser is used to determine the starting HXS.

**Table 5-319. Possible NIA Action Codes**

Value	Code
0x00	Ethernet HXS
0x01	LLC+SNAP HXS
0x02	VLAN HXS
0x03	PPPoE+PPPHXS
0x04	MPLS HXS

**Table 5-319. Possible NIA Action Codes (continued)**

Value	Code
0x05	IPv4 HXS
0x06	IPv6 HXS
0x07	GRE HXS
0x08	MinEncap HXS
0x09	Other L3 Shell
0x0A	TCPHXS
0x0B	UDP HXS
0x0C	IPSec HXS
0x0D	SCTP HXS
0x0E	DCCP HXS
0x0F	Other L4 Shell
0x20–0x3FD	Soft part program
0x3FF	Null parsing

#### 5.9.4.4.2 Parser Inputs/Outputs from/to FMan Memory

Using the Internal Context Pointer the parser reads the following Internal Frame Context fields from the FMan Memory and stores them in a 128-byte internal array (to the parser) called the Parse Array (see [Section 5.9.4.6, “Parse Array,”](#) for a detailed description of the Parse Array). The fields in the Parse Array are not modified unless the parser has parsing results to store. The soft parser can also modify any of the fields stored in the Parse Array:

- Frame descriptor (FD)
- Action descriptor (AD)
- Coarse classification base (CCBASE)
- Key size (KS)
- Hard parser NIA (HPNIA)
- Parse result (PR)

The parser then reads the frame data (up to 256 bytes) using the Internal Frame Pointer and the length of the frame contained in the frame descriptor (FD).

When parsing is completed, the following Internal Frame Context fields in the FMan Memory are updated:

- PR from a subset of the fields in the Parse Array if modified by the parser or the soft parser
- FD, AD, CCBASE, KS and HPNIA from the corresponding fields in the Parse Array if these fields have been modified internally by the soft parser
- FD status bits according to the error events detected
- Debug trace. If debug is enabled and the parser traps, a debug trace is written starting at debug offset, up to the trace size or the end of the debug area of the internal context.

This table lists the parser inputs/outputs from and to FMan memory.

**Table 5-320. Parser Inputs/Outputs from/to FMan Memory**

Field	Size [Bytes]	Who Normally Sets this Value	Description
FD	16	BMI	Frame Descriptor <ul style="list-style-type: none"> <li>Frame length: The parser reads the frame data according to the frame length (up to 256 bytes).</li> <li>Frame status: The parser updates some error/functional bits 24–29 in the frame status.</li> <li>All the fields of the FD can be changed by the soft parser</li> </ul>
AD	8	BMI	Action Descriptor. The soft parser can change this field.
CCBASE	4	BMI	CCBASE. The soft parser can change this field.
KS	1	BMI	Key Size. The soft parser can change this field.
HPNIA	3	BMI Changed by the Parser	Hard Parser NIA. Identifies the next module/process after parsing ends. See <a href="#">Section 5.9.4.4.4, “Hard Parser Next Instruction Address (HPNIA),”</a> for a more detailed description. The soft parser can change this field.
PR	32	Initiate by BMI Changed by Parser	Parser Result. The BMI initiates the parser result according to the port configuration. The parser updates the PR. See <a href="#">Section 5.9.4.10, “Parse Result,”</a> for a more detailed description.
Debug	Up to 112	All FMan modules	Debug. The parser writes its debug trace to this area
Frame	Up to 256	Written by BMI	Frame data. The parser reads the first buffer of the frame (up to 256 bytes)

#### 5.9.4.4.3 Purpose of the Status Region of the Frame Descriptor (FD[STATUS])

The status region of the frame descriptor (FD[STATUS]) is used to receive and provide signaling information. [Table 5-321](#) lists the break-out of the bits used and set by the parser. See [Section 5.4.3.2.1, “FD Command/Status Word,”](#) for a complete description of FD[STATUS].

**Table 5-321. FD[STATUS] Bits**

Field	Bit	In/Out	Description
DCL4C	3	I	Signals the parser to not perform L4 checksum validation.
PTE	24	I/O	Indicates that the Cycle limit is exceeded and the parser is forced to terminate early. See <a href="#">Section 5.9.4.12.2, “Parse Cycle Limit.”</a>
ISP	25	I/O	Indicates an invalid softparse instruction is encountered. See <a href="#">Section 5.9.4.12.1, “Invalid Soft Parser Instruction.”</a>
PHE	26	I/O	Indicates there is an error during parsing. L*R results contain the details. See <a href="#">Section 5.9.4.12.3, “Parse Error Status.”</a>
FRDP	27	I/O	Indicates a frame is received on a disabled port. See <a href="#">Section 5.9.4.12.5, “Frame Received on a Disabled Port or Unsupported Port.”</a>

**Table 5-321. FD[STATUS] Bits (continued)**

Field	Bit	In/Out	Description
BLE	28	I/O	Indicates 256 block limit is exceeded. See <a href="#">Section 5.9.4.8, “256 Limit Event Handling.”</a>
L4CV	29	O	When set, indicates Layer 4 checksum validation is performed.

#### 5.9.4.4.4 Hard Parser Next Instruction Address (HPNIA)

The parser reads the Hard Parser NIA (HPNIA) from the Internal Frame Context in the FMan Memory and stores it internally in the Parse Array. The HPNIA can be updated by the parser (specifically the soft parser). The HPNIA is used to identify the next processing stage after the parser ends. Possible next processing modules are defined in this table.

**Table 5-322. HPNIA Next Processing Modules**

Module	Description
KeyGen	The parser can jump to the KeyGen for pre-classification and frame queue spreading.
Policer	The parser can jump directly to the policer
BMI	The parser can jump to the BMI to do any of the following: <ul style="list-style-type: none"><li>• Enqueue Frame without doing classification and policing</li></ul>
FMan Controller	The parser can jump to the FMan Controller to do any of the following: <ul style="list-style-type: none"><li>• -Coarse Classification</li><li>• Error handling</li><li>• Any special routine that the user wants (for example policing.)</li></ul>

The manipulation of the HPNIA is illustrated in this figure.

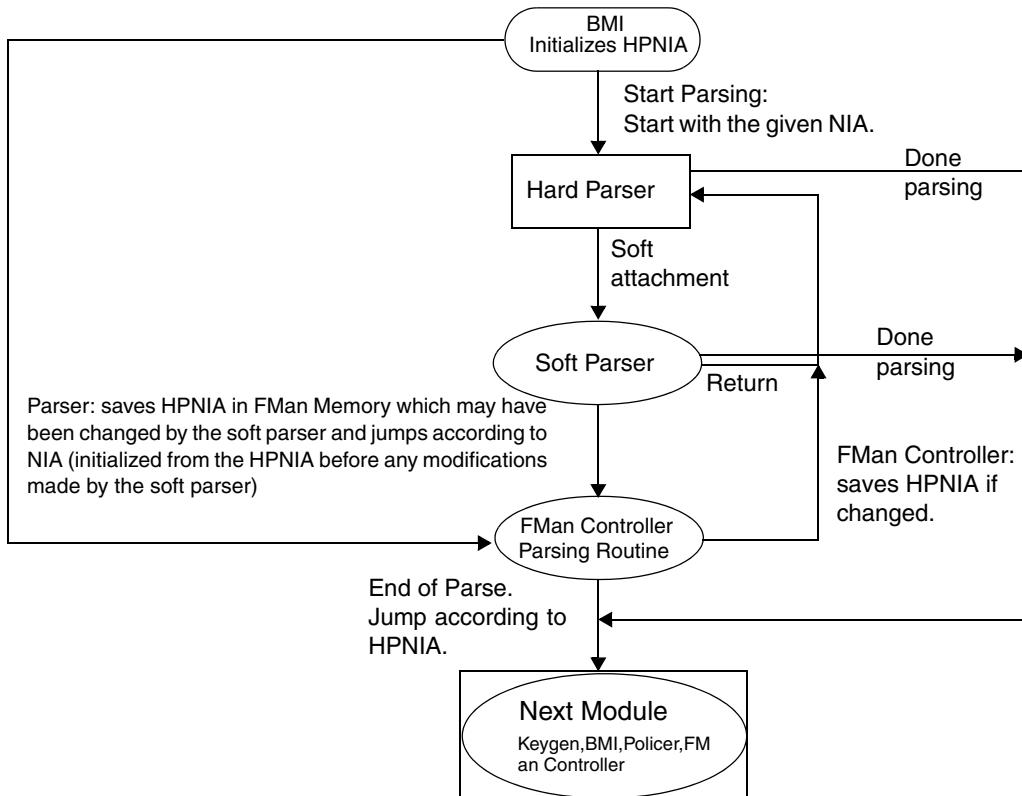


Figure 5-295. HPNIA Flow

#### 5.9.4.4.5 FMan Controller Activation for Parsing

The parser can activate the FMan Controller for special parsing routine, which can be done only by the soft parser. When the FMan Controller completes its parsing routine, it uses the HPNIA to determine the next module to dispatch.

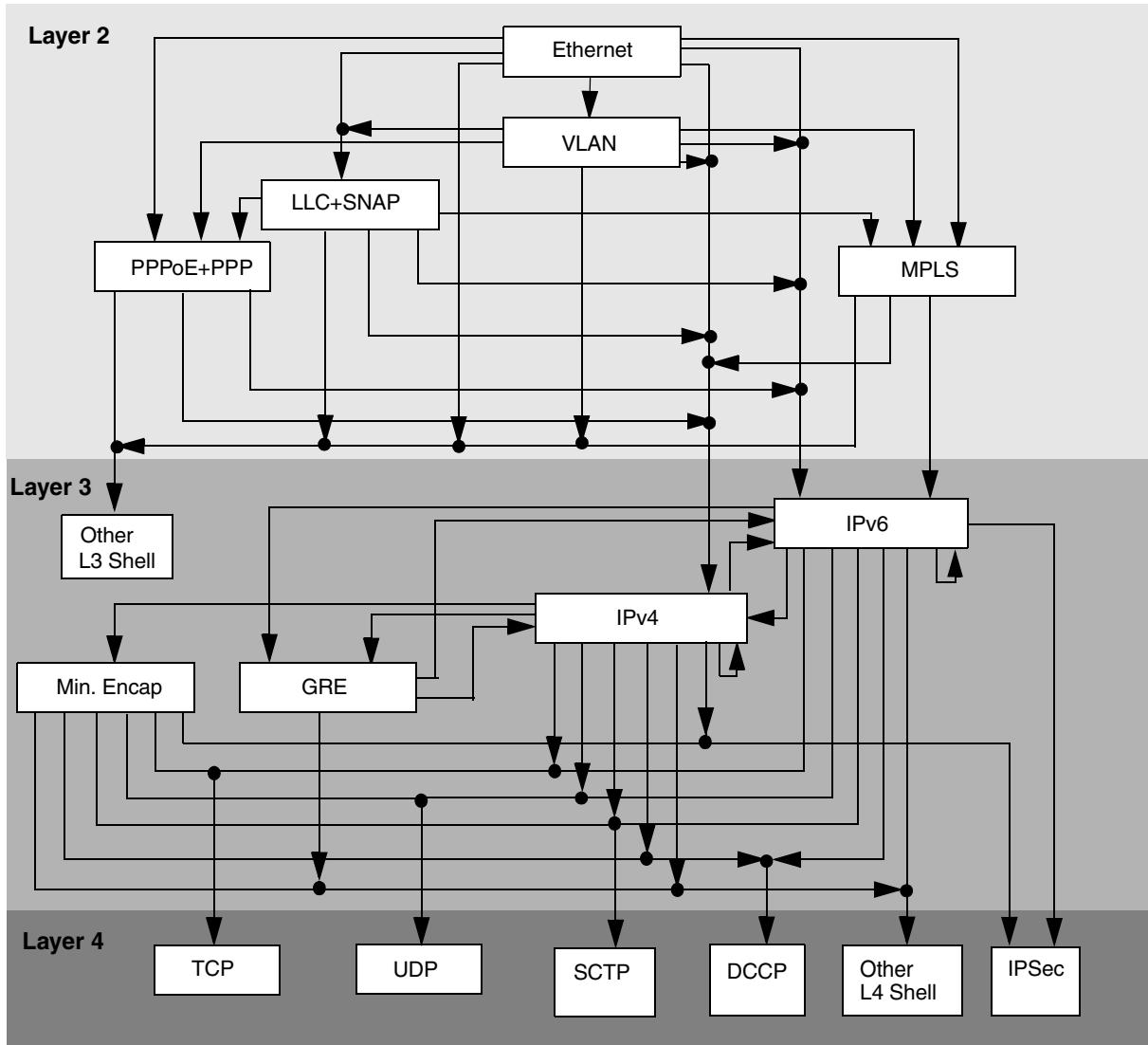
#### 5.9.4.5 Parse Tree and Hard Header Examination Sequences (HXSs)

As shown in [Figure 5-296](#), the hard HXSs form a standard parse tree that covers many common networks protocol headers without requiring the creation of any soft HXSs. Parsing normally progresses from the lower layer to the upper layers. In the event that a completed layer has to be re-entered, the previous parse data from that layer is overwritten. This applies to both fields that are set (for example, packet header offset) and modified (for example, Classification Plan ID).

The starting point (hard or soft HXS) within the parse tree and the offset within the frame where parsing should start are provided via the FPM. This provides, on a per frame basis, the ability to begin the examination of a frame at a different offset within the frame with a different presumption of the first header type. The parse tree is defined by a set of Line-up Enable Confirmation Masks and soft attachments located in the Parse internal memory. Line-up Enable Confirmation Masks and soft attachments are described respectively in [Section 5.9.4.5.3, “Line-up Enable Confirmation Mask,”](#) and [Section 5.9.4.5.2, “Soft Sequence Attachment,”](#) respectively. The Parse internal memory is defined in [Table 5-284](#).

The end point of parsing is either the end of a branch in the parse tree, when reaching the end of the first block of frame data (see [Section 5.9.4.8, “256 Limit Event Handling”](#)), or when a parsing error is encountered,. Furthermore, a hard HXS soft extension can also terminate parsing

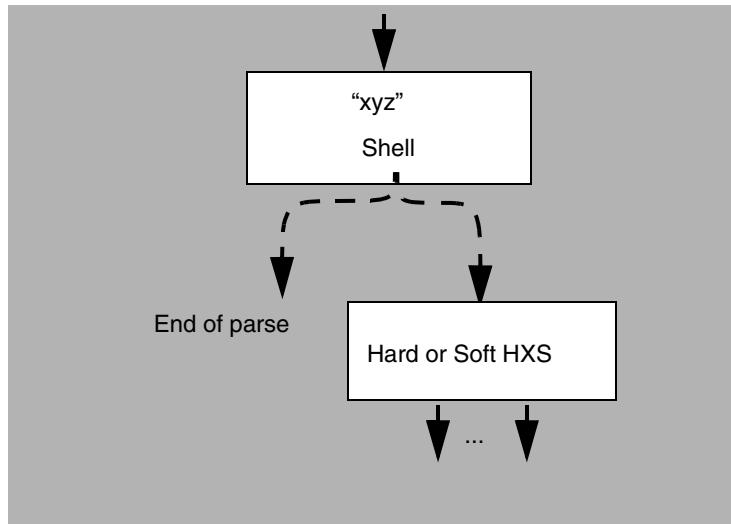
The parse tree is configured on a per port basis. The port is used to select the parse tree configuration.



**Figure 5-296. Hard Parse Tree**

#### 5.9.4.5.1 What is a Parse Shell?

The tree includes some soft HXSs called shells (for examples, see [Table 5-319](#)), each of which can act as a termination point for the parsing or entry point to a specific hard or soft HXS as illustrated in [Figure 5-297](#). Parsing can begin at any hard HXSs or at a shell. A soft HXS can also jump to any hard HXS.

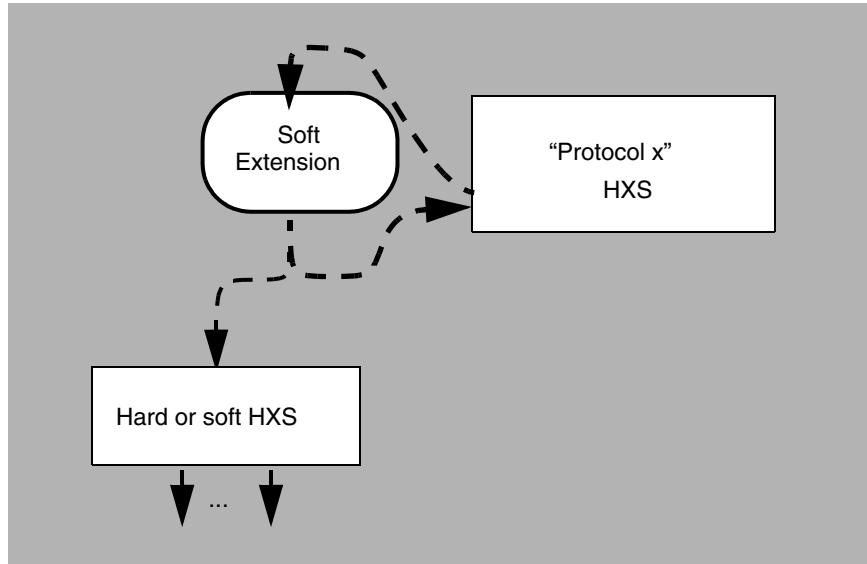


**Figure 5-297. Shell**

#### 5.9.4.5.2 Soft Sequence Attachment

Each hard HXS (including “Shell” HXSs) has a Soft Sequence Attachment record that is located in the Parse internal memory (see [Table 5-285](#)). The structure of the Soft Sequence Attachment record is shown in [Table 5-323](#).

The Soft Sequence Attachment is used to specify a soft extension of examination instructions for a hard HXS. As shown in [Figure 5-298](#) is the fact that each hard HXS can be augmented with a soft extension of examination instructions. The soft extension is considered to be part of the HXS because it operates on the same header. The soft extension could for instance extract additional values from a header or in a more extreme case effectively replace the hard HXS by overwriting any assignment made to the Parse Array. When a soft extension is added, the sequence examination instructions specified by the soft extension begins execution after the hard HXS has completed its parsing and extracted values into the Parse Array before it advances the Header Base register (pointer to the start of the current header). While the Header Base register is not altered, the window offset register (offset within the header being parsed) is reset to the beginning of the current layer. Execution is carried out regardless of errors found in the current layer. The soft sequence instructions may augment or supersede (replace) the work of the hard HXS based on the instructions it executes because all areas of the Parse Array can be assigned by soft instructions (including re-assign values just extracted). Based on the instructions within it, the soft sequence may re-enter the hard HXS to follow its defined advancements (made before transitioning to the to soft extension) to other HXSs or not re-enter and perform its own advancements to any of the hard or soft HXS.



**Figure 5-298. Soft Extension**

Note that upon returning to the current hard HXS, the following actions are performed:

- All changes to the Parse Array are kept, but not re-evaluated, except errors as noted below.
- All changes to the Line-up Confirmation Vector (internal register) are kept. See [Section 5.9.4.5.4, “Line-Up Confirmation Vector,”](#) for a detailed description of the Line-up Confirmation Vector.
- The current hard HXS applies (that is, ORed) its Line-up Enable Confirmation Mask (located in the Parse internal memory, see [Table 5-284](#)) to the Line-up Confirmation Vector register if the updated result (L2R, L3R, L4R) indicates no error. The Line-up Enable Confirmation Mask bits correspond to classification searches and when set indicates the classification searches to be enabled when the hard HXS has parsed its header successfully.
- The hard HXS restores the values of Window Offset and Header Base to where the hard HXS left off when initially transitioning to the soft extension.
- The hard HXS continues to the next layer as determined before it transitioned to the soft extension if no error had been encountered by the hard HXS and no error is set by the soft extension.
- If an error is detected by the hard HXS (this is made before it transitioned to the soft extension), upon returning, the parsing ends regardless of any modifications made to the result by the soft extension. If the soft extension wishes to change that outcome, it must either terminate parsing itself or branch to the next soft or hard HXS without doing a return to the hard HXS.

Presence of a soft extension of examination instructions in a Soft Sequence Attachment is specified by setting the EN field in the Soft Sequence Attachment record. When enabled, the soft sequence attachment is executed at the end of the hard HXS execution, just before the hard HXS branches out to the next HXS.

The Soft Sequence Attachment also includes fields that can direct a hard HXS to alter the Classification Plan ID in the Parse Array when certain types of frames are detected (for example, broadcast frame/packet). This provides the ability to change the Classification Plan ID without having to specify a soft extension for some specific types of frames that would require a different treatment. The Classification Plan ID can be changed in power of 2 only, this way a binary tree of plans can be built. The Classification

Plan ID offset can be programmed to the following values: 0 means no change, 1-8 means offset of 1, 2, 4, 8, 16, 32, 64, 128, 9-15 also means no change. These fields are always acted on whether or not the soft extension is enabled (EN field in the Soft Sequence Attachment record).

The Soft Sequence Attachment also includes Hard HXS configuration fields. These fields are always acted on whether or not the soft extension is enabled (EN field in the Soft Sequence Attachment record).

**Table 5-323. Hard HXS—Soft Sequence Attachment**

Bit Offset	Field	Description
0–19	HXS configuration	Different configuration options supported for the following hard HXSs: <ul style="list-style-type: none"> <li>• Ethernet HXS (see <a href="#">Table 5-324</a>)</li> <li>• VLAN HXS (see <a href="#">Table 5-326</a>)</li> <li>• PPPoE+PPP (see <a href="#">Table 5-325</a>)</li> <li>• MPLS HXS (see <a href="#">Table 5-327</a>)</li> <li>• IPv4 HXS (see <a href="#">Table 5-328</a>)</li> <li>• IPv6 HXS (see <a href="#">Table 5-329</a>)</li> </ul> Reserved, set to 0 for all other HXS
20	Error Reporting Mask	Per HXS error reporting into FD[STATUS] PHE (bit 26) 0 disable, report error from this HXS to the PHE bit of FD[STATUS] 1 enabled, do not report error An error is reported into the status field at the end of the execution of the HXS or, if soft attachment is enabled, applied upon completion of the SSA whether by instruction or termination event. When enabled this mask prevents this error from being reported. The mask has no effect when pure soft parsing.
21	EN	Enable. When set to 1, it enables soft extension of examination instructions beginning at Soft Sequence Start
22–31	Soft Sequence Start	A 10-bit index into the Parse internal memory (16-bit offset) indicating the soft sequence start address. Index value must be equal or higher than 0x20 (that is, start of soft parse instructions in the Parse internal memory).

**Table 5-324. Ethernet HXS Configuration**

Bit Offset	Field	Description
0–3	BC Plan Offset	Broadcast Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a broadcast frame is decoded.
4–7	MC Plan Offset	Multicast Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a multicast frame is decoded.
8–19	Reserved	Set to 0

**Table 5-325. PPPoE+PPP HXS Configuration**

Bit Offset	Field	Description
0	Enable MTU checking	Ethernet has a maximum payload size of 1500 octets. PPPoE header is 6 octets and the PPP Protocol ID is 2 octets thus the PPP MTU can not exceed 1492 <sup>1</sup> . (RFC2516 section 7)
1–19	Reserved	Set to 0

<sup>1</sup> The 2 byte ppp protocol id field is part of the payload thus the max value of the length field is 1494 as per RFC2516

**Table 5-326. VLAN HXS Configuration**

Bit Offset	Field	Description
0–11	Reserved	Set to 0
12–15	SVLAN Plan Offset	Stacked VLAN Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a second VLAN tag is decoded (is not added again if subsequent if more VLAN(s) found).
16–19	Reserved	Set to 0

**Table 5-327. MPLS HXS Configuration**

Bit Offset	Field	Description
0–9	MPLS Default Next Parse	A 10-bit index into the Parse internal memory (16-bit offset) indicating the HXS or soft sequence start address to advance after the MPLS HXS when the MPLS Label Interpretation is disabled or MPLS label is >15. Index value must be equal or higher than 0x5 (IPv4 and beyond, can not go back).
10	MPLS Label Interpretation enable	When this bit is set, the last MPLS label is interpreted as per <a href="#">Table 5-340</a> . When the bit is cleared, the parser advances to MPLS Default Next Parse
11	Reserved	Set to 0
12–15	SMPLS Plan Offset	Stacked MPLS Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a second MPLS label is decoded (is not added again if subsequent if more MPLS(s) found).
16–19	Reserved	Set to 0

**Table 5-328. IPv4 HXS Configuration**

Bit Offset	Field	Description
0–3	IPV4_1 BC Plan Offset	Broadcast Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a broadcast frame is decoded in the outer header.
4–7	IPV4_1 MC Plan Offset	Multicast Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a multicast frame is decoded in the outer header.
8–11	IP2 UC Plan Offset	IP2 Unicast Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a unicast frame is decoded in the inner header.
12–15	IP2 MC/BC Plan Offset	IP2 Multicast/Broadcast Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a multicast or broadcast frame is decoded in the inner header.
16–19	Reserved	Set to 0

**Table 5-329. IPv6 HXS Configuration**

Bit Offset	Field	Description
0–3	Reserved	Set to 0
4–7	IPV6_1 MC Plan Offset	Multicast Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a multicast frame is decoded.
8–11	IP2 UC Plan Offset	IP2 Unicast Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a unicast frame is decoded in the inner header.
12–15	IP2 MC Plan Offset	IP2 Multicast Classification Plan ID Offset. Offset in power of 2 ( $2^{\text{Offset}-1}$ ) added to the Classification Plan ID located in the Parse Array when a multicast frame is decoded in the inner header.
16	Routing Header Enable	When not set (by default), the routing header is ignored and the destination address from the Main header is used instead. The presence of the routing header is still reported in the Parse Array but is not handed off with the Parse Result.
17–19	Reserved	Set to 0

**Table 5-330. TCP HXS Configuration**

Bit Offset	Field	Description
0	Short Packet Padding Removal From Checksum Calculation	When set to 1, the contribution of the padded region at the end of a frame which is padded to 60 <sup>1</sup> bytes are removed from the checksum calculation. Required for layer 4 checksum validation on short frames
1–19	Reserved	Set to 0

**Note:**

- <sup>1</sup> From the Ethernet standard, short packets are padded to form 64-byte frames. This stipulation includes the 4-byte FCS. The presence of the FCS must be indicated in the FD[STATUS] field.

**Table 5-331. UDP HXS Configuration**

Bit Offset	Field	Description
0	Short Packet Padding Removal From Checksum Calculation	When set to 1, the contribution of the padded region at the end of a frame which is padded to 60 <sup>1</sup> bytes is removed from the checksum calculation. Required for layer 4 checksum validation on short frames
1–19	Reserved	Set to 0

**Note:**

- <sup>1</sup> From the Ethernet standard, short packets are padded to form 64-byte frames. This stipulation includes the 4-byte FCS. The presence of the FCS must be indicated in the FD[STATUS] field.

#### 5.9.4.5.3 Line-up Enable Confirmation Mask

The Line-up Enable Confirmation Mask is a 32-bit value that is applied by a HXS or the Soft Parser to the line-up confirmation vector. The Line-up Enable Confirmation Mask values for the hard Header Examination Sequences (HXSs) are stored in pre-defined locations in the Parse internal memory as detailed in [Table 5-284](#).

**Table 5-332. Line-up Enable Confirmation Mask**

Bit Offset	Field	Description
0–31	Line-up Enable Confirmation Mask	Mask applied by HXS to the line-up confirmation vector when HXS completes parsing without error.

#### 5.9.4.5.4 Line-Up Confirmation Vector

The 32-bit Line-up Confirmation Vector is initialized from the Parse Result at the start of parsing each frame. Bits in the register correspond to classification searches and are intended to confirm that appropriate headers/fields have been found/verified for the searches to be executed if requested. As each HXS (this includes shells) is successfully executed (no error reported), the associated Line-up Enable Confirmation Mask is ORed with the Line-up Confirmation Vector. This process continues until all parsing is complete at which point this register is captured as context that travels with the associated frame. See [Figure 5-299](#) for details.

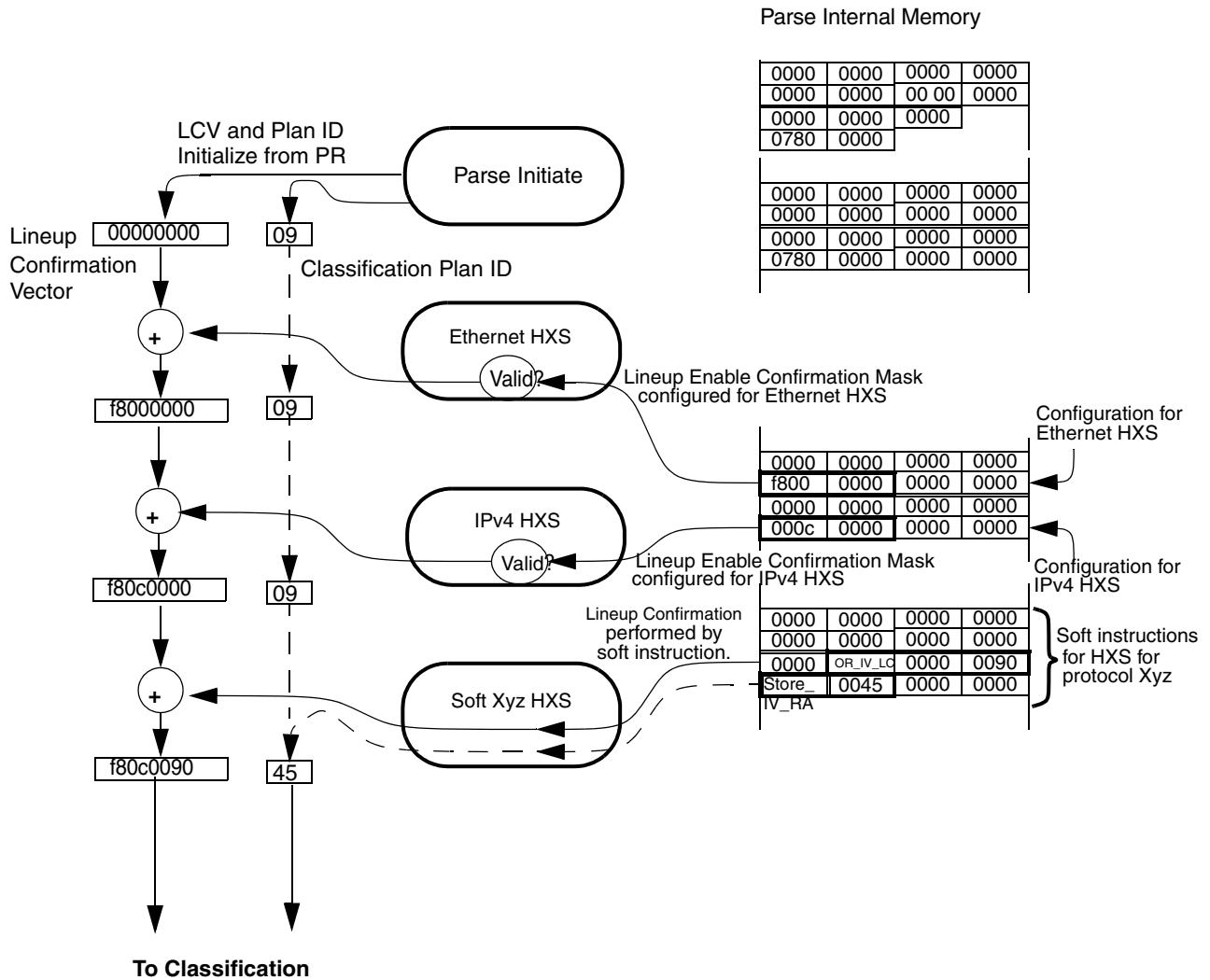


Figure 5-299. Line-Up Confirmation Example

#### 5.9.4.6 Parse Array

The parser stores all parsing results in a 128-byte Parse Array located inside the parser. As parsing begins for a frame, the values in this array are set to zero except for initial default values which are assigned based on the values read from Internal Frame Context in the FMan Memory. Both the hard and soft Header Examination Sequences (HXSs) modify and assign values in this array. This array provides the means for an HXS to incorporate results from a previously executed HXS (for example TCP checksum requires the IP addresses). A portion (Parsing Result) of this array is passed along with the frame to the next stage. The results of each layer and offsets are parse history values for the frame. Their intent is to record where each layer of header began within the frame head (offset) and some basic result of the examination (for example, header type, validity, variant). The remaining values in the array are values extracted from the frame head by hard or soft HXS. The names of the locations in the array are based on the values placed there by hard HXS, however, a soft HXS can reassign other values to these locations. To allow soft and hard HXSs to coexist, 16 bytes GPRV0..15 (General Purpose Registered Values) are available for exclusive use of the

soft HXSs. Also it must be noted that when an error is encountered, any fields populated by the errored HXS can no longer be considered valid. This includes not just HXS specific fields but also some common fields like: Next Header, Next Header offset, Running Sum and possibly even Classification Plan ID (if the current layer legally could update the plan). [Table 5-333](#) shows the parse array.

**Table 5-333. Parse Array**

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF								
Byte: 0x	General Purpose Registered Values 0–15																							
Byte: 1x	Logical Port ID [0:7]	ShimR	L2R		L3R		L4R	Classification Plan ID	NxtHdr		Running Sum		Flags [0:2], Frag Offset [3:15]		Route Type	RHP[0 ] IPValid [1]								
Byte: 2x	ShimOff set_1	ShimOff set_2	IP Pid Offset	EthOff set	LLC+S NAOff set	VLANT CIOffset _1	VLANT CIOffs et_n	LastETy peOffset	PPPoE Offset	MPLSOf fset_1	MPLSOf fset_n	IPOffs et_1	IPOffs et_n or MinEn capO	GREOff set	L4Off set	NxtHdr Offset								
Byte: 3x	Frame Descriptor																							
Byte: 4x	Action Descriptor							CCBASE				KS	HPNIA											
Byte: 5x	SPERC [0:3]	Reserved			IP Ver[4:7]	IP Length		Reserved		ICP		ATTR [2:7]	NIA											
Byte: 6x	IPv4 SA Bytes 0–3 or IPv6 SA Bytes 0–3			IPv4 DA Bytes 0–3 or IPv6 SA Bytes 4–7				IPv6 SA Bytes 8–15																
Byte: 7x	IPv6 DA Bytes 0–15																							

[Table 5-334](#) describes the parse array fields.

**Table 5-334. Parse Array Field Descriptions**

Parse Array Field	Description
General Purpose Registered Values (GPRV) 0–15 registers	16-byte wide general purpose register (GPRV0..15) area for exclusive use of the soft examination sequences for targeting their results
Logical Port ID	An 8-bit identifier that can be used to represent the logical port origin of a frame. This field is initialized to the Logical Port ID value stored in the Parse Result. The Logical Port ID field can only be modified through soft examination sequences. For example, a soft examination sequence could be defined to set this field to a value extracted from a system header/user-defined shim if present.
ShimR (Shim Result)	A coded result assigned by a soft examination sequence considered to be for a shim Layer. The default value, zero, is intended to be reserved to indicate no Shim Layer examination method is executed. This field is initialized to the ShimR value stored in the Parse Result.

**Table 5-334. Parse Array Field Descriptions (continued)**

Parse Array Field	Description
L2R (Layer 2 Result)	A coded result assigned by a HXS considered to be for Layer 2. The default value, zero, is intended to be reserved to indicate no explicit Layer 2 HXS is executed. This field is initialized to the L2R value stored in the Parse Result. The Ethernet, LLC+SNAP, VLAN, PPPoE+PPP and MPLS HXSs assign to L2R one of the reserved values (see section <a href="#">Section 5.9.4.7.6, “L2 HXS—L2 Results”</a> ). One of the non-reserved values could be assigned by a soft HXS if considered to be Layer 2. Once an error is detected and reported in this result any field extracted at this layer can no longer be trusted or considered valid. No further parsing is done regardless of any modifications made to the result by the soft extension if present. If the soft extension wish to change that outcome, it must either terminate parsing itself or branch to the next soft or hard HXS (not doing a return to the hard HXS)
L3R (Layer 3 Result)	A coded result assigned by a HXS considered to be for Layer 3. The default value, zero, is intended to be reserved to indicate no explicit Layer 3 HXS is executed. This field is initialized to the L3R value stored in the Parse Result. The IPv4, IPv6, GRE and Min Encap HXSs assign to L3R one of the unique reserved values (see section <a href="#">Section 5.9.4.7.13, “L3 HXS—L3 Results”</a> , for details). One of the non- reserved values could be assigned by a soft HXS if considered to be Layer 3. Once an error is detected and reported in this result any field extracted at this layer can no longer be trusted or considered valid. No further parsing are done regardless of any modifications made to the result by the soft extension if present. If the soft extension wish to change that outcome, it must either terminate parsing itself or branch to the next soft or hard HXS (not doing a return to the hard HXS).
L4R (Layer 4 Result)	A coded result assigned by a HXS considered to be for Layer 4. The default value, zero, is intended to be reserved to indicate no explicit Layer 4 HXS is executed. This field is initialized to the L4R value stored in the Parse Result. The TCP, UDP, SCTP, DCCP and IPSec HXSs assign to L4R one of the unique reserved values (see section <a href="#">Section 5.9.4.7.20, “L4 HXS—L4 Results”</a> ). One of the non- reserved values could be assigned by a soft HXS if considered to be Layer 4. Once an error is detected and reported in this result any field extracted at this layer can no longer be trusted or considered valid. No further parsing are done regardless of any modifications made to the result by the soft extension if present. If the soft extension wish to change that outcome, it must either terminate parsing itself or branch to the next soft or hard HXS (not doing a return to the hard HXS).
Classification Plan ID	The Classification Plan ID is an 8-bit index that points to a classification plan in the FMan Controller or KeyGen module. Each entry in the Classification Plan table consists of a 32-bit vector where each bit corresponds to a classification search. When a bit is set, it indicates that a search is to be performed if the headers/fields for the search have been found/validated. This field is initialized to the Classification Plan ID value stored in the Parse Result. The Classification Plan ID can be altered by a soft examination sequence and by a Ethernet, VLAN, MPLS, IPv4 or IPv6 HXS for specific scenarios. Specifically, the Classification Plan ID can be altered by the Ethernet HXS or IPv4 HXS when the parser decodes a multicast or broadcast frame/packet. The Classification Plan ID can also be altered by the IPv6 HXS when the parser decodes a multicast packet. The Classification Plan ID can also be altered by the VLAN, MPLS, IPv4 and IPv6 HXS when stacked header are found.
NxtHdr (Next Header)	Holds the last result of the parsed header of the next header type. This field can be used by a soft examination sequence to determine the next HXS when parsing a shim header that does not have the definition of the next header type and depends on the previous header. Assigned by every hard HXS that completes without error a layer and found a next header descriptor (Ethertype, Protocol ID, etc.). Can also be assigned by a soft examination sequence. This field is initialized to the NxtHdr value stored in the Parse Result.
Running Sum	At the start this is the bulk ones complement sum of all the data in the frame excluding FCS. As each layer is parsed that layer's contribution to the bulk sum is removed from this sum except for layer 4. In the case of TCP or UDP the ones complement sum of the pseudo header is added to this remaining sum. Thus a result of 0xffff indicates a valid checksum. If the TCP/UDP header checksum field had been 0 then the inverted version of this final sum is the checksum for that header. In all other layer 4 headers only the portion parsed are removed from the running sum. See the individual HXS sections for more information.

**Table 5-334. Parse Array Field Descriptions (continued)**

Parse Array Field	Description
Flags [0:2]	Flags field in a IPv4/IPv6 packet. For IPv4, this field is set by the IPv4 HXS to the Flags value (rsv,D,M) extracted from the IPv4 packet. For IPv6, this field is set by the IPv6 HXS to the Flag value (rsv,rsv,M) of the Fragment Header (identified by a Next Header value of 44). This field is initialized to zero. Indication that this field has been updated by the IPv4/IPv6 HXS is provided by having the L3R indicating the presence of IP & Result Type in L3R set to "0".
Frag Offset [3:15]	Fragment Offset field in a IPv4/IPv6 packet. This field is set by the IPv4/IPv6 HXS to the Fragment Offset value extracted from the IPv4/IPv6 packet. In the case of IPv6, this field is extracted from the Fragment Header (identified by a Next Header value of 44). This field is initialized to zero. Indication that this field has been updated by the IPv4/IPv6 HXS is provided by having the L3R indicating the presence of IP & Result Type in L3R set to "0" (and L3R[5] set to "1" for IPv6).
Rout Type (Routing Type)	Routing type field of a IPv6 routing extension header. The field is set by the IPv6 HXS to the routing type extracted from the IPv6 routing extension header of an IPv6 packet. This field is initialized to zero. Indication that this field has been updated by the IPv6 HXS is provided by having the RHP bit set in the Parse Array.
RHP (Routing Extension Header Present)	Routing Extension Header Present. This field is set by the IPv6 HXS when a routing extension header is present in the IPv6 packet. This field is initialized to zero. Indication that this field has been updated by the IPv6 HXS is provided by L3R indicating the presence of an IPv6 header.
IPvalid [1]	IP captured fields are valid. This bit is initialized to zero when parsing is started. When an IPv4/IPv6 HXS is executed, it sets the bit. If this bit is cleared, the L4 checksum is not validated.
ShimOffset_1/2 (Shim 1/2 Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be Shim 1/2. These fields are initialized to the ShimOffset_1/2 values stored in the Parse Result. They remain unchanged if no Shim soft HXS is executed. There is no hard HXS that assigns these values, but a soft examination sequence can assign these values.
IP PID Offset	Byte position (0–255) within frame head where parsing reached the point considered to be last IP Protocol identifier. This field is initialized to the IP PID Offset value stored in the Parse Result. The IPv4 and IPv6 HXSs assign this field. This field can also be altered by a soft examination sequence.
EthOffset (Ethernet Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be the start of the Ethernet header. This field is initialized to the EthOffset value stored in the Parse Result. The Ethernet HXS assigns this field. This field can also be altered by a soft examination sequence.
LLC+SNAPOffset (LLC+SNAP Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be the start of the LLC+SNAP header. This field is initialized to the LLC+SNAPOffset value stored in the Parse Result. The LLC+SNAP HXS assign this field. This field can also be altered by a soft examination sequence.
VLANTCIOffset_1 (VLAN TCI 1 Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be the first VLAN (TCI). This field is initialized to the VLANTCIOffset_1 value stored in the Parse Result. The VLAN HXS assigns this field. This field can also be altered by a soft examination sequence.
VLANTCIOffset_n (VLAN TCI n Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be the last VLAN (TCI). This field is initialized to the VLANTCIOffset_n value stored in the Parse Result. The VLAN HXS assigns this field. This field can also be altered by a soft examination sequence.
LastETypeOffset (Last EtherType Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be the last EtherType. This field is initialized to the LastETypeOffset value stored in the Parse Result. The Ethernet HXS, the VLAN HXS, and the LLC+SNAP HXS assign this field. This field can also be altered by a soft examination sequence.
PPPoEOffset (PPPoE Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be PPPoE. This field is initialized to the PPPoEOffset value stored in the Parse Result. The PPPoE+PPP HXS assign this field. This field can also be altered by a soft examination sequence.

**Table 5-334. Parse Array Field Descriptions (continued)**

Parse Array Field	Description
MPLSOffset_1 (MPLS 1 Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be the first MPLS header. This field is initialized to the MPLSOffset_1 value stored in the Parse Result. The MPLS HXS assigns this field. This field can also be altered by a soft examination sequence.
MPLSOffset_n (MPLS n Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be the last MPLS header. This field is initialized to the MPLSOffset_n value stored in the Parse Result. The MPLS HXS assigns this field. This field can also be altered by a soft examination sequence.
IPOffset_1 (IP_1 Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be first IP header (outer). This field is initialized to the IPOffset_1 value stored in the Parse Result. The IPv4 and IPv6 HXSs assign this field. This field can also be altered by a soft examination sequence.
IPOffset_n or MinEncapO (IP_n or Minimum Encapsulation Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be last IP header (inner) or MinEncap header (mutually exclusive, qualified by L3R). This field is initialized to the IPOffset_n or MinEncapO value stored in the Parse Result. The IPv4, IPv6 and MinEncap HXSs assign this field. This field can also be altered by a soft examination sequence.
GREOffset (GRE Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be GRE. This field is initialized to the GREOffset value stored in the Parse Result. The GRE and Other L3 Shell HXSs assign this field. This field can also be altered by a soft examination sequence.
L4Offset (Layer 4 Offset)	Byte position (0–255) within frame head where parsing reached the point considered to be Layer 4. This field is initialized to the L4Offset value stored in the Parse Result. The TCP, UDP, IPSec, SCTP and DCCP HXSs assign this field. This field can also be altered by a soft examination sequence.
NxtHdrOffset (Next Header Offset)	Byte position (0–255) within frame head pointing to the payload of the last parsed header. Can also be defined as the offset to the last header that has not been parsed. This field is initialized to the NxtHdrOffset value stored in the Parse Result which must be non zero. It remains unchanged if no previous HXS is executed. The hard HXSs assign this value otherwise it stays unchanged. In the final stage of parsing, if it is found to be >255 then it is capped at 0xff.
Frame Descriptor	The Frame Descriptor (FD) is initialized to the FD value read from the Internal Frame Context located in the FMan Memory. The FD can be altered by a soft examination sequence. If modified, this value is written back to the FD field of the Internal Frame Context located in the FMan Memory when parsing ends.
DCL4C (bit 3 of FD[Status])	Do not calculate layer 4 checksum. When set this disables the checking of the L4 checksum. This field is initialized to the DCL4C value stored in the FD[Status] bit 3.
Action Descriptor	The Action Descriptor (AD) is initialized to the AD value read from the Internal Frame Context located in the FMan Memory. The AD can be altered by a soft examination sequence. If modified, this value is written back to the AD field of the Internal Frame Context located in the FMan Memory when parsing ends.
CCBASE	CCBASE initialized to the CCBASE value read from the Internal Frame Context located in the FMan Memory. The CCBASE can be altered by a soft examination sequence. If modified, this value is written back to the CCBASE field of the Internal Frame Context located in the FMan Memory when parsing ends.
KS	Key Size (KS) initialized to the KS value read from the Internal Frame Context located in the FMan Memory. The KS can be altered by a soft examination sequence. If modified, this value is written back to the KS field of the Internal Frame Context located in the FMan Memory when parsing ends.
HPNIA	The HPNIA is initialized to the HPNIA value read from the Internal Frame Context located in the FMan Memory. The HPNIA can be altered by a soft examination sequence. If modified, this value is written back to the HPNIA field of the Internal Frame Context located in the FMan Memory when parsing ends. For NIA format details, see <a href="#">Section 5.4.4, “Next Invoked Action (NIA)”</a> .

**Table 5-334. Parse Array Field Descriptions (continued)**

Parse Array Field	Description
SPERC (Soft Parser Error Report Control)	<p>The Soft Parser Error Report Control is used by the parser to determine where the soft parser errors are to be reported and allow the masking of the error from the FDS. This field is initialized to zero. This field is set by a soft examination sequence. If the soft parse sequence was invoked by a soft extension, the parser reports soft parse errors in the *R that is being extended, unless this field has been written while in the soft extension sequence. In the latter, the parser reports the soft parser error in the *R specified in SPERC. If the soft parser sequence is not invoked by a soft extension, the parser reports the soft parser error in the *R specified in SPERC field. This provides the ability for a soft examination to specify the *R as it knows what layer is being worked on.</p> <p>When field[0:2] is written, errors are reported with type 0 in the *R.</p> <ul style="list-style-type: none"> <li>000 Error reported in ShimR</li> <li>001 Error reported in L2R</li> <li>010 Error reported in L3R first</li> <li>011 Error reported in L3R last</li> <li>100 Error reported in L4R</li> <li>101–111 Reserved (note: error not reported)</li> </ul> <p>Field[3] controls the error reporting into FD[STATUS] PHE (bit 26).</p> <ul style="list-style-type: none"> <li>0 disable, report error from this HXS to the PHE bit of FD[STATUS]</li> <li>1 enabled, do not report error</li> </ul> <p>See <a href="#">Section 5.9.4.12.4, “Soft Parser Parsing Events,”</a> for details.</p>
IP Version	This is the version of the last IP. Required for pseudo header calculation
IP Length	The IP Length is set to indicate the length of the last IP payload for the purpose of determining the length of the upper-layer (for example, TCP) header and data when creating the pseudo header for upper-layer protocol checksum calculation. The IPv4, IPv6, and Minimal Encapsulation HXSs assign this field. For IPv4, the length is determined by subtracting the Header Length field in the IPv4 packet from Total Length field in the IPv4 packet. For IPv6, the IP length is obtained by subtracting the lengths of all extension headers present between the IPv6 main header and the last header from the Payload Length field in the IPv6 header. For Minimal Encapsulation, the length of the header is subtracted from IP Length. This field can also be altered by a soft examination sequence.
ICP	ICP (Internal context pointer) from FPM Dispatch. This field is initialized to the ICP provided on the FPM.
ATTR	ATTR (Mode Attribute) from FPM Dispatch. This field is initialized to the ATTR provided on the FPM Dispatch. This field is used to pass the ATTR value on the FPM Command interface.
NIA	<p>Next Instruction Address is a 24-bit field passed to the next processing stage (ex: the FMan Controller or KeyGen module).</p> <p>This field is initialized to the HPNIA[0:23] value read from the Internal Frame Context located in the FMan Memory. The NIA can be altered by a soft examination sequence.</p> <p>For NIA format details, see <a href="#">Section 5.4.4, “Next Invoked Action (NIA),”</a></p>
IPv4 SA Bytes 0-3	Source Address field in a IPv4 packet for the purpose of creating the pseudo header for upper-layer protocol checksum calculation. This field is set by the IPv4 and Minimal Encapsulation HXS to the Source Address value extracted from the IPv4 packet. This field is initialized to zero. Indication that this field has been updated by the IPv4 HXS is provided by having the L3R indicating the presence of IPv4 & Result Type in L3R set to “0”.
IPv4 DA Bytes 0-3	Destination Address field in a IPv4 packet for the purpose of creating the pseudo header for upper-layer protocol checksum calculation. This field is set by the IPv4 and Minimal Encapsulation HXS to the Destination Address value extracted from the IPv4 packet. This field is initialized to zero. Indication that this field has been updated by the IPv4 HXS is provided by having the L3R indicating the presence of IPv4 & Result Type in L3R set to “0”.

**Table 5-334. Parse Array Field Descriptions (continued)**

Parse Array Field	Description
IPv6 SA Bytes 0-15	Source Address field in a IPv6 packet for the purpose of creating the pseudo header for upper-layer protocol checksum calculation. This field is set by the IPv6 HXS to the Source Address value extracted from the IPv6 packet. This field is initialized to zero. Indication that this field has been updated by the IPv6 HXS is provided by having the L3R indicating the presence of IPv6 & Result Type in L3R set to "0".
IPv6 DA Bytes 0-15	Destination Address field in a IPv6 packet for the purpose of creating the pseudo header for upper-layer protocol checksum calculation. This field is set by the IPv6 HXS to the Destination Address value extracted from the IPv6 packet. This field is initialized to zero. Indication that this field has been updated by the IPv6 HXS is provided by having the L3R indicating the presence of IPv6 & Result Type in L3R set to "0".

### 5.9.4.7 Hard Header Examination Sequences (HXSs)

Parsing can start at any hard Header Examination Sequence (HXS). The only requirement is that the Parse Array fields required by a hard HXS to successfully parse a frame be populated appropriately.

For example IP Source and Destination fields must be setup if Layer 4 checksum validation is to be performed. Or when entering at the VLAN HXS, the Next Header field must contain an EtherType value known by a Layer 2 HXS (for example, Ethernet) as indicating that the VLAN HXS is to be executed for the next HXS.

#### 5.9.4.7.1 L2 HXS - Ethernet

The Ethernet HXS has the ability to parse the following Ethernet frame formats (shown in [Figure 5-300](#)):

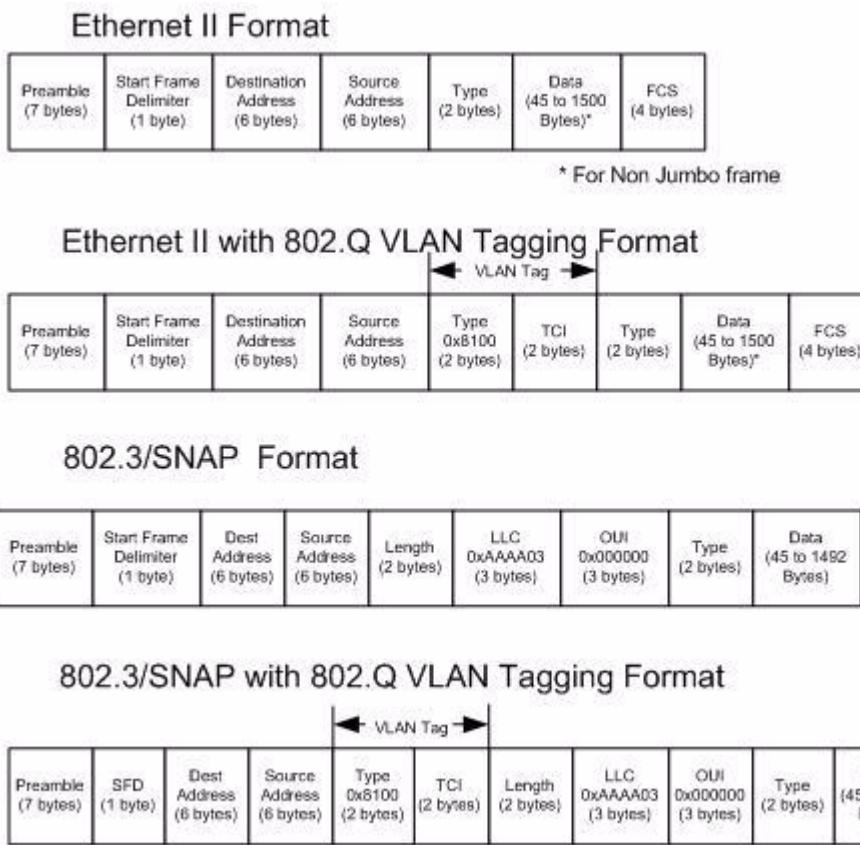
- Ethernet II
- Ethernet II with IEEE 802.1Q VLAN tagging (up to 2 VLAN tags stacked can be extracted)
- IEEE 802.3/SNAP
- IEEE 802.3/SNAP with IEEE 802.1Q VLAN tagging (up to 2 VLAN tags stacked can be extracted)

Ethernet II is version 2 of the original Ethernet standard. It includes a 2-byte Protocol ID field known as the EtherType field. This field follows the Source MAC address and is used to specify the next protocol in the frame. As part of the Full Duplex Ethernet standardization project, the IEEE incorporated the use of the Type field (Ethernet II format) into the 802.3 standard (designated as the EtherType/Length field).

The 802.3/SNAP format is defined by the IEEE. In the 802.3/SNAP format, the 2-byte field after the source MAC address (if VLAN not present) is a length specifying the number of bytes in the LLC and data fields. This length field does not include DA, SA, Length, VLAN tag if present, FCS or padding bytes. Note that the length value remains unchanged through the use of VLAN tagging but gets offset from its original position, due to the VLAN tag(s).

Both Ethernet II and 802.3/SNAP frames may be tagged with the IEEE 802.1Q VLAN tagging approach. The 4-byte IEEE 802.1Q tag which is inserted right after Source MAC Address, has two 2-byte components: the Tag Protocol Identifier (TPID, residing within the type/length field) and the Tag Control Information (TCI).

The Ethernet frame format is determined by examining the EtherType/Length field. If the EtherType/Length field is 0x0600 (1536 decimal) or greater, it is a Type format (that is, Ethernet II or VLAN tagged). If the EtherType/Length field is 0x05DC (1500 decimal) or less, the frame is a Length format (802.3/SNAP). Values between 1501 and 1535 inclusively are undefined.



**Figure 5-300. Ethernet Frame Formats**

The Ethernet HXS updates the Parse Array with the following information:

- Next Header is set to the extracted EtherType/Length
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the EtherType field.
- Ethernet Offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the Ethernet header.
- Last EtherType Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte of the EtherType field.
- Classification Plan ID is updated with information specified in the Hard HXS configuration field of the Soft Sequence Attachment when certain types of frames (for example, multicast, broadcast) are detected. See [Table 5-324](#) for more details. While specialized MAC addresses exist for multicast IPv4 and IPv6 they are not treated with any special regard. Multicast is detected in the

more generic case by examining the multicast bit (least significant bit of the most significant byte of the destination MAC address set to 1 if multicast). An address of all 1's indicates broadcast.

- Running Sum; Ethernet header (up to and including the EtherType) contribution to the bulk sum is removed
- L2 Result (L2R)

The Ethernet HXS performs the following validation on the Ethernet header:

- There are no validation checks performed on the Source and Destination MAC addresses, and EtherType field.

The Ethernet HXS uses the information stored in the EtherType field to determine the next HXS to be executed. The EtherType field is located following the Source MAC address field. The next HXS is determined according to the “Ethernet—EtherType to Next HXS” mapping described in [Table 5-335](#). If the EtherType field is set to a value other than the ones included in [Table 5-335](#), the next header is considered an unknown type and the next HXS to be executed is the Other L3 shell.

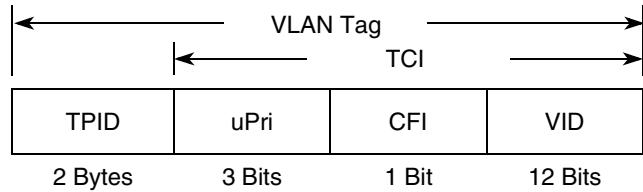
If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the Ethernet HXS updates is overwritten. This applies to both the Parse Array fields that are set (for example, packet header offset) and modified (for example, Classification Plan ID).

**Table 5-335. Ethernet—EtherType to Next HXS Mapping**

EtherType	Next HXS
x05DC or less	LLC+SNAP
x0800	IPv4
x86dd	IPv6
x8847	MPLS
x8848	MPLS
x8100	VLAN
x88A8	VLAN
ConfigTPID1	VLAN
ConfigTPID2	VLAN
x8864	PPPoE+PPP

#### 5.9.4.7.2 L2 HXS—VLAN

Ethernet frames may optionally contain a 4-byte IEEE 802.1Q tag to identify what VLAN it belongs to and its IEEE 802.1p priority (quality of service) as well as 4-byte IEEE 802.1ad stacked VLAN tags.



**Figure 5-301. VLAN Tag**

As illustrated [Figure 5-301](#), the VLAN tag contains two 2-byte components: the Tag Protocol Identifier (TPID), residing within the type/length field, and the Tag Control Information (TCI). The following TPID values are recognized as VLAN tag:

- 0x8100
- 0x88A8
- ConfigTPID1
- ConfigTPID2

The TCI includes the following fields:

- User Priority—The first three bits of the TCI define user priority.
- CFI—Canonical Format Indicator (single-bit field)
- VID—The 12-bit VLAN ID (VID) field identifies the VLAN the frame belongs to. The 12 bits allow for 4096 different VLANs. Out of the 4096 VLANs, the VID of zero is used to identify priority frames and the value of 4095 is reserved.

The VLAN HXS captures up to two VLAN offsets (TCI field of first and last VLAN tags encountered) where applicable. Note that frames with more than two VLANs are parsed, but these additional intermediate VLAN offsets are not captured, just the first and last one. The VLAN HXS starts at the TCI field of the first encountered VLAN tag. The VLAN HXS updates the Parse Array with the following information:

- VLANTCIOffset\_1—byte position (0–255) within frame head where parsing reached the point considered to be the start of the TCI field in the VLAN tag at the top of the VLAN stack (appears earliest in the packet)
- VLANTCIOffset\_n—byte position (0–255) within frame head where parsing reached the point considered to be the start of the TCI field in the VLAN tag at the bottom of the VLAN stack of depth n (appears latest in the packet). If there is only one VLAN tag then VLANTCIOffset\_1 and VLANTCIOffset\_n are set to the same offset value (TCI field of the single VLAN tag)
- Last EtherType Offset—byte position (0–255) within frame head where parsing reached the point considered to be the first byte of the EtherType field following the last VLAN tag.
- Next Header is set to the extracted EtherType following the last VLAN tag.
- Next Header Offset—byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the EtherType field that follows the last VLAN tag
- Classification Plan ID is updated with information specified in the Hard HXS configuration field of the Soft Sequence Attachment when certain types of frames are detected. See [Table 5-326](#) for more details.

- Running Sum; VLAN header (from TCI field of first VLAN tag to EtherType following the last VLAN tag inclusively) contribution to the bulk sum is removed
- L2 Result (L2R)

The VLAN HXS performs the following validation:

- An info is reported if Canonical Format Indicator (CFI) bit in an “8100” VLAN tag is set. VLAN parsing stops and the next HXS to be executed is the Other L3 shell. Header base is updated to point to the TCI field of this “8100” VLAN tag and the Window offset is set to 0. The Line-up Confirmation Vector is not updated. Parse Array is updated as if this “8100” VLAN tag is successfully parsed with the following exceptions:
  - Next Header, Next Header Offset, and Last Ethertype Offset are not moved after this “8100” VLAN tag. If this is the first VLAN tag encountered, then Next Header, Next Header Offset, and Last Ethertype Offset are not updated. If this is not the first VLAN tag then Next Header is set to the TPID value of this “8100” VLAN tag, Next Header Offset is set to the start of the TCI field in this “8100” VLAN tag and Last Ethertype Offset is set to the start of the TPID field in this “8100” VLAN tag.
  - Checksum portion of this “8100” VLAN tag (from TCI field of this VLAN tag to EtherType following this VLAN tag) is not removed from the Running Sum
  - L2R; CFI bit and Unknown Protocol bit are set.

The VLAN HXS locates the 2-byte field (that is, EtherType field) after the TCI and uses the information stored in that field to determine the next HXS. The next HXS is determined according to the “VLAN - EtherType to Next HXS” mapping described in [Table 5-336](#). If the EtherType is set to x88A8, x8100, ConfigTPID1 or ConfigTPID2 then the VLAN HXS continues by parsing the next VLAN tag.

If the EtherType field is set to a value other than those included in [Table 5-336](#), the next header is considered an unknown type and the next HXS to be executed is the Other L3 shell.

If this HXS is re-entered after it completes, existing data in the Parse Array fields that the VLAN HXS updates is overwritten. This applies to both the Parse Array fields that are set (for example, packet header offset) and modified (for example, Classification Plan ID).

**Table 5-336. VLAN—EtherType to Next HXS Mapping**

EtherType	Next HXS
x05DC or less	LLC + SNAP
x0800	IPv4
x86dd	IPv6
x8847	MPLS
x8848	MPLS
x8100	Remains in VLAN HXS
x8864	PPPoE+PPP
x88A8	Remains in VLAN HXS

**Table 5-336. VLAN—EtherType to Next HXS Mapping (continued)**

EtherType	Next HXS
ConfigTPID1	Remains in VLAN HXS
ConfigTPID2	Remains in VLAN HXS

#### 5.9.4.7.3 L2 HXS—LLC + SNAP

When SNAP is present the DSAP and SSAP fields within the LLC header (3-byte) contain the value x'AA' each and the Control field is set to 3. The SNAP header follows the LLC header and contains a 3-byte Organizational Unit Identifier (or OUI) identifying the authority assigning the protocol ID followed by two bytes specifying the protocol ID according to that authority. When the three byte OUI is set to zero (IEEE), the last two bytes are considered to be an EtherType code.

The LLC+SNAP HXS locates the EtherType field by checking that the first 3 bytes (LLC header) is set to 0xAAAA03 and the next 3 bytes (OUI) are set to 0. It uses the information stored in the EtherType field to determine the next HXS to be executed. The next HXS is determined according to the “802.3/SNAP - EtherType to Next HXS” mapping described in [Table 5-337](#). If the EtherType field is set to a value other than the ones included in [Table 5-337](#), the next header is considered an unknown type and the next HXS to be executed is the Other L3 shell.

The LLC+SNAP HXS updates the Parse Array with the following information when a SNAP header is found with the OUI field is set to zero:

- The Next Header field in the Parse Array is updated with the EtherType value extracted from the SNAP header.
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the SNAP header.
- LLC+SNAP Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte of the LLC+SNAP header (first byte of LLC).
- Last EtherType Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte of the EtherType field.
- Running Sum; LLC+SNAP header contribution to the bulk sum is removed
- L2 Result (L2R)

The LLC+SNAP HXS performs the following validation on the header:

- 802.3 Length does not exceed the frame length

If the first 3 bytes (LLC header) is not set to 0xAAAA03 or the next 3 bytes (OUI) are not set to 0, LLC+SNAP or there are less than 8 bytes reported in the 802.3 length field, parsing stops and the next HXS to be executed is the Other L3 shell. Header base is not updated and the Window offset is set to 0. The Line-up Confirmation Vector is not updated. Parse Array is updated as follows:

- LLC+SNAP Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte of the LLC+SNAP header (first byte of LLC).
- Next Header, Next Header Offset and Last EtherType Offset are not updated
- Checksum portion of the LLC+SNAP header is not removed from the Running Sum

- L2 Result (L2R); LLC + SNAP present bit, Unknown LLC/OUI bit and Unknown Protocol bit is set.

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the LLC + SNAP HXS updates is overwritten. This applies to both the Parse Array fields that are set (for example, packet header offset) and modified (for example, Running Sum).

**Table 5-337. 802.3/SNAP—EtherType to Next HXS Mapping**

EtherType	Next HXS
x0800	IPv4
x86dd	IPv6
x8847	MPLS
x8848	MPLS
x8864	PPPoE+PPP

#### 5.9.4.7.4 L2 HXS—PPPoE+PPP

PPP over Ethernet (PPPoE+PPP) is used to connect a group of hosts in a network to a remote access concentrator allowing each host to maintain their own PPP stack and familiar user interface. This in turn allows type of service, access management and billing to be performed on a per user instead of per site basis.

To set-up PPPoE session a unique Session ID and the Ethernet address of the remote peer must be determined. The stage that performs this set-up is referred to as the Discovery Stage. During the Discovery Stage several frames are transferred back and forth between the host and the various access concentrators. The Ethernet type during this stage is 0x8863. This negotiation allows the host to learn of all the access concentrators and the services they provide. At the conclusion of this process the host selects one and the determined Session ID uniquely binds the two together. Data transfer can now commence and this stage is referred to as the PPP Session Stage. The Ethernet type during this stage is 0x8864.

A LCP negotiation begins to establish communications over a PPP link. Each link sends LCP frames (indicated by the PPP protocol field set to 0xC021) to configure and test the data link connection. When communications are established, the PPP Protocol field is set to 0x0021 for IPv4 and 0x0057 for IPv6.

The hard parser does not parse PPPoE-PPP Discovery Stage frames beyond the Ethernet header (that is, does not parse beyond the Ethernet Type field) and as a result would advance to the “Other L3 shell. However, the hard parser parses PPPoE-PPP Session Stage frames beyond the Ethernet header (including VLAN tag(s)) and the later is achieved through the PPPoE+PPP HXS.

The PPPoE+PPP HXS updates the Parse Array with the following information:

- PPPoE Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte of the PPPoE header.
- The Next Header is loaded with the PPP Protocol extracted from the header.
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the PPPoE+PPP header.
- Running Sum; PPPoE+PPP header contribution to the bulk sum is removed

- L2 Result (L2R)

The PPPoE+PPP HXS performs the following validation on the header:

- PPPoE MTU can not exceed 1492(Not active by default. See [Table 5-325](#))
- The VER field must be set to 0x1
- The TYPE field must be set to 0x1
- The Code field must be 0
- Session ID set to 0xffff which is reserved and thus must not be used.
- PPPoE Length does not exceed frame length

When communications is established the PPP Protocol field is set to 0x0021 for IPv4 and 0x0057 for IPv6. The parser then proceeds to IPv4 HXS or IPv6 HXS respectively. If a different protocol is selected the parser advances to Other L3 shell (where parsing may end or direct to a soft HXS). The Other L3 Shell does not provide any header parsing and validation results. It can act as a termination point for the parsing or entry point to a L3 soft HXS. [Table 5-338](#) shows the Ethernet with PPPoE frame formats.

**Table 5-338. Ethernet with PPPoE Frame Format**

B0	B1	B2	B3	B4	B5	B6	B7
Access Concentrator MAC address						Host MAC Address	
Host MAC address Cont.			Ethernet Type = 0x8864		Ver=1,Type=1	Code=0	
Session ID		Length		PPP Protocol		—	—

The PPPoE+PPP HXS uses the information stored in the PPP Protocol field to determine the next HXS to be executed. The next HXS is determined according to the “PPPoE+PPP - Protocol to Next HXS” mapping described in [Table 5-339](#). If the PPP Protocol field is set to a value other than the ones included in [Table 5-339](#), the next header is considered an unknown type and the next HXS to be executed is the Other L3 shell.

**Table 5-339. PPPoE+PPP—Protocol to Next HXS Mapping**

EtherType	Next HXS
x0021	IPv4
x0057	IPv6

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the PPPoE+PPP HXS updates is overwritten. This applies to both the Parse Array fields that are set (for example, packet header offset) and modified (for example, Running Sum).

#### 5.9.4.7.5 L2 HXS—MPLS

The MPLS HXS captures the offsets of the first label in the MPLS header (MPLSOffset\_1) and the offset of the last label in the MPLS header (MPLSOffset\_n) into the Parse Array. Frames with more than 2 labels are parsed but the offsets to the other labels are not captured. The MPLS HXS checks the bottom of the stack bit (S bit) in the label to identify the last label. If there is only 1 label, then the offset of that label is placed in MPLSOffset\_1 and MPLSOffset\_n. If there are 2 labels, then the offsets of the two labels are

placed in MPLSOffset\_1 and MPLSOffset\_n respectively. If there are more than 2 labels, then the offset of the first label is placed in MPLSOffset\_1 and the offset of the last label is placed in MPLSOffset\_n.

The MPLS HXS updates the Parse Array with the following information:

- MPLSOffset\_1; byte position (0–255) within frame head where parsing reached the point considered to be the start of the first label (appears earliest in the packet)
- MPLSOffset\_n; byte position (0–255) within frame head where parsing reached the point considered to be the start of the last label (appears latest in the packet). If there is only one MPLS label then MPLSOffset\_1 and MPLSOffset\_n are set to the same offset value.
- Next Header is set to the last MPLS label. Since the last label is 20 bits, bits 0 to 4 are ORed and stored in Next Header bit 0, label bit 5 to 19 are stored in Next Header bit 1 to 15
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the MPLS header(s).
- Classification Plan ID is updated with information specified in the Hard HXS configuration field of the Soft Sequence Attachment when certain types of frames are detected. See [Table 5-327](#) for more details.
- Running Sum; The MPLS header contribution to the bulk sum is removed from the Running Sum
- L2 Result (L2R)

When the last label has been reached, the next HXS or soft sequence to execute can either be the default next parse sequence (MPLS Default Next Parse field of the MPLS HXS Configuration (see [Table 5-327](#))) or be based on the interpretation of the last label. The above configuration option is specified in the MPLS Label Interpretation field of the MPLS HXS Configuration (see [Table 5-327](#)).

The default next parse sequence is an index indicating which HXS to advance to (IP or beyond) or an address to a soft parse program.

If the parser is configured to interpret the last label, then the next HXS is determined according to the “MPLS Label to Next HXS” mapping described in [Table 5-340](#).

When the parser is configured to interpret the last label, and the label is in the 0-15 range, and an unknown label is detected (not 0: IPv4 or 2: IPv6) the “Unknown Protocol” flag is set.

**Table 5-340. MPLS Label to Next HXS Mapping**

Label (Decimal)	Next HXS
0	IPv4
1	Other L3 Shell
2	IPv6
3-15	Other L3 Shell
> 15	Default next parse sequence



**Figure 5-302. MPLS (RFC3032)**

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the MPLS HXS updates is overwritten. This applies to the Parse Array fields that are set (for example, packet header offset) but not to the Classification Plan ID which can only be updated on the first occurrence of a stacked MPLS.

#### 5.9.4.7.6 L2 HXS—L2 Results

This table lists the L2R decoded information.

**Table 5-341. L2R Decoded**

L2 Type Bit 0:4	Info Bit 5:10	Error Bit 11:15
00000: No parsing executed or soft parsing	user defined	See <a href="#">Table 5-342</a>
Bit 0: Ethernet present Bit 1: VLAN present Bit 2: LLC+SNAP present Bit 3: MPLS present Bit 4: PPPoE+PPP present	Bit 5: CFI (Canonical Format Indicator) bit in a “8100” VLAN tag is set. Bit 6: Unknown LLC/OUI (LLC is not AAAA03 or OUI is not zero or Ethernet Length is <= 8) Bit 7: Stacked VLAN Bit 8: Unknown Protocol (set when next HXS to be executed is the Other L3 shell) Bit 9:10: Frame type -00: Unicast -10: Multicast -11: Broadcast	See <a href="#">Table 5-342</a>

This table lists the L2R result codes.

**Table 5-342. L2R Result Codes**

Type	Code	Result	Description
Common	0	No Error	—
Ethernet	1	Reserved	—
	2	Frame Truncation	Parser reached end of frame while parsing the header
VLAN	3	Reserved	—
	4	Frame Truncation	Parser reached end of frame while parsing the header
LLC+SNAP	5	Reserved	—
	6	Frame Truncation	Parser reached end of frame while parsing the header
	7	802.3 Truncation	802.3 length is larger than the frame received.
MPLS	8	Reserved	—
	9	Frame Truncation	Parser reached end of frame while parsing the header

**Table 5-342. L2R Result Codes (continued)**

Type	Code	Result	Description
PPPoE+ PPP	10	Reserved	—
	11	Frame Truncation	Parser reached end of frame while parsing the header
	12	PPPoE Truncation	PPPoE length is larger than the frame received.
	13	MTU violated	Ethernet has a maximum payload size of 1500 octets. PPPoE header is 6 octets and the PPP Protocol ID is 2 octets thus the PPP MTU cannot exceed 1492 <sup>1</sup> . (RFC2516 section 7)
	14	Version Invalid	The VER field must be set to 0x1 (RFC2516 section 4)
	15	Type Invalid	The TYPE field must be set to 0x1 (RFC2516 section 4)
	16	Code Invalid	On PPP Session Stage frames, the Code field must be 0 (RFC2516 section 4)
	17	Session ID Invalid	Session ID set to 0xffff which is reserved and thus must not be used. (RFC2516 section 4)
—	18–29	Reserved	—
Soft parsing	30	Reserved	—
	31	Frame Truncation	Soft Parser reached end of frame while parsing the header

**Note:**

<sup>1</sup> The 2-byte ppp protocol id field is part of the payload thus the max value of the length field is 1494 as per RFC2516.

If an L2 error is encountered, the parsing is stopped for this frame and the soft extension attached to the HXS is invoked if enabled. In the case where the soft extension re-enters the HXS or the soft extension is not enabled, parsing is terminated without having the Line-up Confirmation Vector updated and the frame is passed to the next stage. Since the Line-up Confirmation Vector is not updated for this hard HXS, no operations using fields parsed by Layer 2 should be performed. However other operations using fields that have been successfully validated by other HXSs (for example, Soft HXS) can be performed.

#### 5.9.4.7.7 L3 HXS

Only 2 layers of IP (where Minimal Encapsulation is considered IP) are supported. When parsing the second IP header in the case of an IP over IP tunnel, the IP header fields in the Internal Parse Array are replaced by the new captured fields and result. The one exception, in the case of different IP versions, is the lineup enable confirmation mask, which is cumulative; that is, the 2nd mask is applied and added to the existing vector instead of removing the effect of the previous mask. (Note that the Classification Plan ID can be used to determine the order in which the IPv6 and IPv4 header were encountered.)

#### 5.9.4.7.8 L3 HXS—IPv4

The IPv4 HXS updates the Parse Array with the following information:

- Version extracted from the header
- Fragment Offset extracted from the header
- Flags extracted from the header

- Source Address, Destination Address extracted from the header
- IPValid is set to 1
- Offset to IP Protocol ID field
- Next Header is loaded with the Protocol (in bit location [8-15], bits [0-7] are zeroed)
- Next Header Offset; byte position (0-255) within frame head where parsing reached the point considered to be the first byte following the IPv4 header (including options)
- The IP\_1 Offset; byte position (0-255) within frame head where parsing reached the point considered to be the start of the first IP header (outer). If L3R “First IP Present IPv4” bit or “First IP Present IPv6” bit is set then this field is not updated since this is not the first IP header encountered.
- The IP\_n Offset; byte position (0-255) within frame head where parsing reached the point considered to be the start of the last IP header (inner) in the case of an IP over IP tunnel. This field is always updated by the IPv4 HXS to the current IP header offset. If there is only one IP header parsed in the packet, then IP\_1 Offset and IP\_n Offset are set to the same offset value.
- Classification Plan ID is updated with information specified in the Hard HXS configuration field of the Soft Sequence Attachment when certain types of frames are detected. See [Table 5-328](#) for more details. The IPv4 HXS identifies the packet as multicast if the destination IP address is set to a Class D address (224.0.0.0 to 239.255.255.255). No special handling of 224/24 addresses is performed and thus treated as generic multicast. If the destination IP address is set to 255.255.255.255, then packet is identified as a broadcast packet. Otherwise, the packet is identified as a unicast packet. If L3R “First IP Present IPv4” bit or “First IP Present IPv6” bit is set then the IP2\* Plan offset would be applied instead of the IPV4\_1\* Plan offset.
- Running Sum; IPv4 header contribution to the bulk sum is removed
- IP Length; set by subtracting the Header Length field in the IPv4 packet from Total Length field in the IPv4 packet. This field is to be used for determining the length of the upper-layer (for example, TCP) header and data when creating the pseudo header for upper-layer protocol checksum calculation.
- L3 Result (L3R); “First IP Present IPv4” bit is set to 1 if this is the first IP header encountered (First IP Present IPv4” bit and “First IP Present IPv6” bit are set to 0) otherwise the “Last IP Present IPv4” bit is set to 1 and the “Last IP Present IPv6” bit is set to 0.

The IPv4 HXS performs the following validation on the header:

- Version number in IPv4 packet is not equal to 4
- IPv4 checksum is incorrect. Checksum is performed on the region defined by a minimum of 20 bytes to a maximum defined by the header length.
- The IPv4 header length field is less than 20 bytes or exceeds the length (excluding any header preceding the IPv4 header) of the packet received
- The IPv4 header length field exceeds the IPv4 total length. Not performed if this is a later IP where the primary IP was marked as a fragment.
- The IPv4 total length field exceeds the length (excludes any header preceding the IPv4 header) of the packet received.

- The minimum size of an IP fragment is the minimum size of an IP header plus eight data bytes. Unless this is the last fragment

Except for the calculation/validation of Header Checksum, the IPv4 HXS does not examine the option fields (a soft extension could be attached to do so).

If it is a non-fragmented packet (Fragment Offset is set to zero and the More Fragments flag is set to zero) or it is an initial fragment packet ((Fragment Offset is set to zero and the More Fragments flag is set to one), then the IPv4 HXS uses the information stored in the Protocol field to determine the next HXS to be executed. The next HXS is determined according to the “IPv4 - Protocol to Next HXS” mapping described in [Table 5-343](#). If the protocol field is set to a value other than the ones included in [Table 5-343](#), the next header is considered an unknown type and the next HXS to be executed is the Other L4 shell.

If it is a non-initial fragment (Fragment Offset is non zero), then it advances to the Other L4 shell. L4 offset is not advanced because no L4 has been encountered.

If the parser has already encountered at least two IP headers (indicated by L3R “Last IP Present IPv4” bit being set) and the Protocol field in the current IPv4 header is set to IPv4, IPv6, GRE, or Minimal Encapsulation then it advances to the Other L4 shell.

If this HXS is re-entered after it has completed, it behaves like it is parsing the last IP header and existing data in the Parse Array fields that the IPv4 HXS update is overwritten. This applies to both the Parse Array fields that are set (for example, packet header offset) and modified (for example, Classification Plan ID).

**Table 5-343. IPv4—Protocol to Next HXS Mapping**

Protocol (Decimal)	Next HXS
4	IPv4 if first IP header encountered otherwise Other L4 Shell
6	TCP
17	UDP
33	DCCP
41	IPv6 if first IP header encountered otherwise Other L4 Shell
50	IPSec
51	IPSec
47	GRE if first IP header encountered otherwise Other L4 Shell
55	Minimal Encapsulation if first IP header encountered otherwise Other L4 Shell
132	SCTP

#### 5.9.4.7.9 L3 HXS—IPv6

The IPv6 HXS updates the Parse Array with the following information:

- Version extracted from the main header
- Source Address and Destination Address extracted from the header
- Flags; Flags[0:2] are set to the {rsv,rsv,M} fields of the Fragment Extension Header if present
- Fragment Offset is set to the Fragment Offset field of the Fragment Extension Header if present

- Routing Extension Header Present is set when a Routing Extension Header is present in the IPv6 packet
- Routing Type is set to the Routing Type field of the Routing Extension Header if present
- IPValid is set to 1
- Offset to IP Protocol ID field
- The IP\_1 Offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the first IPv6 header (outer). If L3R First IP Present IPv4 bit or First IP Present IPv6 bit is set then this field is not updated since this is not the first IP header encountered.
- The IP\_n Offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the last IP header (inner) in the case of an IP over IP tunnel. This field is always updated by the IPv6 HXS to the current IP header offset. If there is only one IP header parsed in the packet, then IP\_1 Offset and IP\_n Offset are set to the same offset value.
- Next Header is loaded with the IPv6 Next Header (in bit location [8–15], bits [0–7] are zeroed)
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte of the upper-layer protocol carrying the packet's payload (that is, first byte following the IPv6 main header and extension headers if any present)
- Classification Plan ID is updated with information specified in the Hard HXS configuration field of the Soft Sequence Attachment when certain types of frames are detected. See [Table 5-329](#) for more details. The IPv6 HXS identifies the packet as multicast if the destination IP address upper bits 8 bits are set to FF (ff00::/8). Otherwise, the packet is identified as a unicast packet. Otherwise, the packet is identified as a unicast packet. If L3R “First IP Present IPv4” bit or “First IP Present IPv6” bit is set then the IP2\* Plan offset would be applied instead of the IPV6\_1\* Plan offset.
- Running Sum; IPv6 header contribution to the bulk sum is removed
- IP Length; set by subtracting the lengths of all extension headers present between the IPv6 main header and the last header from the Payload Length field in the IPv6 header. This field is to be used for determining the length of the upper-layer (for example, TCP) header and data when creating the pseudo header for upper-layer protocol checksum calculation.
- L3 Result (L3R); “First IP Present IPv6” bit is set to 1 if this is the first IP header encountered (First IP Present IPv4” bit and “First IP Present IPv6” bit are set to 0) otherwise the “Last IP Present IPv6” bit is set to 1 and the “Last IP Present IPv4” bit is set to 0.

The IPv6 HXS performs the following validation on the header:

- Version number in IPv6 packet is not equal to 6
- The IPv6 Payload length + 40 bytes exceeds the length (excludes any header preceding the IPv6 header) of the received packet. Not performed if this is a later IP where the primary IP was marked as a fragment. If Payload length is equal to 0 (indicates hop-by-hop Jumbo payload), then this causes this error code to be generated as this option is not supported.
- An error is reported if the Hop by Hop header did not immediately follow the IPv6 main header or there is more than one Hop by Hop header.
- Routing headers encountered with type set to 0 and Header Length not an even value, or not equal to or greater than twice the number of Left Segments, are reported as errored.

The IPv6 main header may be followed by an arbitrary number (0, 1 or multiple) of extension headers chained together through the Next Header field. The extension header chain typically ends with the last header corresponding to the upper-layer protocol carrying the packet's payload (for example, TCP, UDP). In the presence of extension headers, the IPv6 HXS traverses the extension header chain for the main purpose of finding the upper-layer protocol header (for example, TCP or UDP). Parsing of the IPv6 extension headers is described in [Table 5-344](#).

**Table 5-344. IPv6 Extension Header Parsing**

Extension Header (Decimal)	Processing
0 (Hop-by-Hop Options)	Header is parsed without being examined (values not checked nor captured). However if it is present, it must immediately proceed the IPv6 main header and there can only be one of these in a packet.
43 (Routing)	If Routing Type is set to 0 and Segments Left is non-zero, and Routing Header Enable in the IPv6 HXS Configuration is set to 1 then the last destination address stored in the Routing Extension Header is extracted for the use in generating a pseudo header for a potential Layer 4 checksum as well as determining unicast or multicast addressing. Routing Type is also stored in the Parse Array.
44 (Fragment)	Extracts the Fragment Offset and the More Flag fields from the Fragment Extension Header and stores into them the Parse Array. If it is a non-initial fragment, then it does not process the next header (if any) and advances to the Other L4 shell.
59 (No Next Hdr)	Indicates no more data to follow. Will transition to Other L4 shell.
60 (Destination Options)	Header is parsed without being examined (values not checked nor captured).

The IPv6 HXS uses the information stored in the IPv6 Next Header field to determine the Next HXS to be executed. The next HXS is determined according to the “IPv6 - Next Header to Next HXS” mapping described in [Table 5-345](#). If the Next Header field is set to a value other than the ones included in [Table 5-344](#) or [Table 5-345](#), the next header is considered an unknown type and the next HXS to be executed is the Other L4 shell.

If the parser has already encountered at least two IP headers (indicated by L3R “Last IP Present IPv6” bit being set) and the last header corresponding to the upper-layer protocol in the current IPv6 header is set to IPv4, IPv6 or GRE then it advances to the Other L4 shell.

If this HXS is re-entered after it has completed, it behaves like it is parsing the last IP header, and existing data in the Parse Array fields that the IPv6 HXS updates is overwritten. This applies to both the Parse Array fields that are set (for example, packet header offset) and modified (for example, Classification Plan ID).

**Table 5-345. IPv6—Next Header to Next HXS Mapping**

Protocol (Decimal)	Next HXS
4	IPv4 if first IP header encountered otherwise Other L4 Shell
6	TCP
17	UDP
33	DCCP
41	IPv6 if first IP header encountered otherwise Other L4 Shell
50	IPSec (ESP)

**Table 5-345. IPv6—Next Header to Next HXS Mapping (continued)**

Protocol (Decimal)	Next HXS
51	IPSec (AH)
47	GRE if first IP header encountered otherwise Other L4 Shell
132	SCTP

#### 5.9.4.7.10 L3 HXS - GRE version 0

The GRE HXS supports the GRE header as defined in RFC2784 and RFC 2890 (see [Figure 5-303](#)). The GRE header definition in RFC 1701 is not supported in the GRE HXS. That is the checksum, key and sequence fields are parsed but the routing fields (RFC 1701) are not.

The GRE HXS updates the Parse Array with the following information if the Routing Present bit is not set:

- The GRE Offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the GRE header.
- Next Header is loaded with Protocol Type
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the GRE header.
- Running Sum; GRE header contribution to the bulk sum is removed
- L3 Result (L3R)

The GRE HXS performs the following validation on the header:

- The version field is verified and must be zero, if not an error is generated
- If the checksum is present, the GRE HXS does not validate the checksum as the checksum includes the GRE header and payload.

Since RFC1701 is not supported in the GRE HXS, if the Routing Present bit is set, then GRE HXS branches to the Other L4 Shell. Header base is not updated and the Window offset is set to 0. The Line-up Confirmation Vector is not updated. Parse Array is updated as follows:

- The GRE Offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the GRE header.
- Next Header and Next Header Offset are not updated
- Checksum portion of the GRE header is not removed from the Running Sum
- L3 Result (L3R); GRE Present bit is set and RFC1701 R bit set

0				7   8	13    15   16		23   24		31						
C	R	K	S	Reserved	Ver	Protocol Type									
Checksum(Optional)				Reserved(Optional)											
Key (Optional)															
Sequence Number (Optional)															

**Figure 5-303. GRE (RFC2890)**

For supported GRE headers, the protocol type field in the header is used to determine where the parser advances next. (that is, Next HXS to be executed). The next HXS is determined according to the “GRE - Protocol Type to Next HXS” mapping described in [Table 5-346](#). If the Protocol Type field is set to a value other than the ones included in [Table 5-346](#) the next header is considered an unknown type and the next HXS to be executed is the Other L4 shell.

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the GRE HXS updates is overwritten. This applies to both the Parse Array fields that are set (for example, packet header offset) and modified (for example, Running Sum).

**Table 5-346. GRE—Protocol Type to Next HXS Mapping**

Protocol Type	Next HXS
x0800	IPv4
x86dd	IPv6

#### 5.9.4.7.11 L3 HXS—Minimal Encapsulation

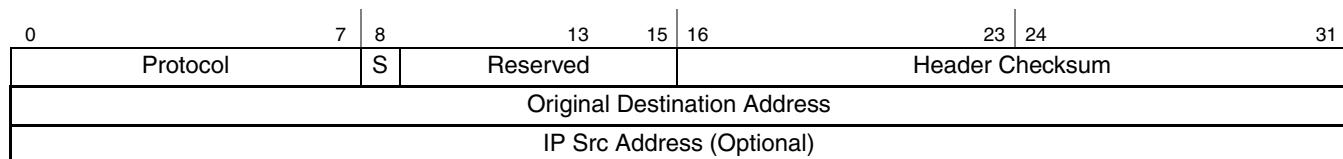
The Minimal Encapsulation HXS parses through the minimal encapsulation header as defined by RFC2004 (see [Figure 5-304](#)). The S bit in the header is used to determine whether the header length is 8 or 12 bytes.

The Minimal Encapsulation HXS updates the Parse Array with the following information:

- Destination Address and optionally (if the s flag is set) Source Address extracted from the header. These values replace the IPv4 Source and Destination Addresses.
- The Minimum Encapsulation Offset; byte position (0-255) within frame head where parsing reached the point considered to be the start of the Minimum Encapsulation header.
- Next Header is loaded with Protocol (in bit location [8-15], bits [0-7] are zeroed)
- Next Header Offset; byte position (0-255) within frame head where parsing reached the point considered to be the first byte following the MinEncap header.
- Running Sum; Minimal Encapsulation header contribution to the bulk sum is removed
- IP Length; adjusted by subtracting 8 or 12 bytes (based on S bit) from the current IP Length field. This field is to be used for determining the length of the upper-layer (for example, TCP) header and data when creating the pseudo header for upper-layer protocol checksum calculation.
- L3 Result (L3R)

The Minimum Encapsulation HXS performs the following validation on the header:

- The header checksum is calculated and validated against the value in the minimal encapsulation header.



**Figure 5-304. Minimal Encapsulation for IP**

The protocol field in the header is used to determine where the parser advances next. (that is, Next HXS to be executed). The next HXS is determined according to the “Min. Encap. - Protocol to Next HXS” mapping described in [Table 5-347](#). If the Protocol field is set to a value other than the ones included in [Table 5-347](#), the next header is considered an unknown type and the next HXS to be executed is the Other L4 shell. The extracted destination and source addresses as well as the protocol field are used in TCP/UDP pseudo header generation.

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the Minimal Encapsulation HXS updates is overwritten. This applies to both the Parse Array fields that are set (for example, packet header offset) and modified (for example, Running Sum).

**Table 5-347. Min. Encap.—Protocol to Next HXS Mapping**

Protocol (Decimal)	Next HXS
6	TCP
17	UDP
33	DCCP
50	IPSec
51	IPSec
132	SCTP

#### 5.9.4.7.12 L3 HXS—Other L3 Shell

The Other L3 Shell does not provide any header parsing and validation results. It can act as a termination point for the parsing or entry point to a soft HXS.

The Other L3 Shell HXS does not update the Parse Array. Since Next Header Offset has been set by the HXS who jumped to Other L3 Shell, it represents the byte position (0–255) within frame head where parsing reached the point considered to be the start of the unknown L3 header.

If a soft parser error is found while executing the Other L3 Shell soft attachment, it is reported in First Result column of the L3R. If there is already a result in First Result column, then the error is reported in Last Result column of the L3R.

Parsing ends when execution of the Other L3 Shell HXS completes.

#### 5.9.4.7.13 L3 HXS—L3 Results

[Table 5-348](#) lists the results gathered for Layer 3. Of the two sets of identical columns, the first represents the primary Layer 3 header while the second represents any tunnel header that may be present. For GRE and MinEncap, results are captured in the Last Result column. It should be noted that if GRE is followed by an IP header then the GRE results are overwritten with the IP header results.

**Table 5-348. L3R Decoded**

L3 Type Bit 0:5	First Result Type Bit 6	First Result Bit 7:11	Last Result Type Bit 12	Last Result <sup>1</sup> Bit 13:15
000000: No parsing executed or soft parsing	0: info	User defined	0: info	User defined
	1: error	See <a href="#">Table 5-351</a>	1: error	See <a href="#">Table 5-351</a>
Bit 0: First IP Present IPv4 Bit 1: First IP Present IPv6 Bit 2: GRE Present Bit 3: MinEnc Present Bit 4: Last IP Present IPv4 Bit 5: Last IP Present IPv6	0: info	See <a href="#">Table 5-349</a>	0: info	See <a href="#">Table 5-350</a>
	1: error	See <a href="#">Table 5-351</a>	1: error	See <a href="#">Table 5-351</a>

**Note:**

<sup>1</sup> If an error is reported in the Result bit 7:11, then this field is invalid.

**Table 5-349. First L3R Info Results Decoded**

L3 Type	Info Results Bits 0:4 map to L3R bits 7:11
IPv4	Bit 0: IP option present Bit 1: Unknown protocol (not IP/GRE/MINENC/TCP/UDP/SCTP/DCCP/IPSec) Bit 2: Packet is a fragment (“more fragments” flag is set or the “fragment offset” field is non-zero) Bit 3:4 Packet type -00: unicast -10: multicast -11: Broadcast <sup>1</sup>
IPv6	Bit 0: Hop-by-Hop Options (0), Routing (43) and/or Destination Options (60) headers present Bit 1: Next Header value in the last header processed is not IP/GRE/TCP/UDP/SCTP/DCCP/IPSec (could be a No Next header) Bit 2: Packet is a fragment (IPv6 Fragment Extension Header present) Bit 3: Packet type -0: unicast -1: multicast Bit 4: Hop-by-Hop Options (0) header present

**Note:**

<sup>1</sup> Hard parser can only detect local network broadcasts but soft parser can further refine broadcast detection by applying a subnet mask

Table 5-350 lists the last L3R decoded results.

**Table 5-350. Last L3R Info Results Decoded**

L3 Type	Info Results Bits 0:2 map to L3R bits 13:15
IPv4	Bit 0: Packet is a fragment (“more fragments” flag is set or the “fragment offset” field is non-zero) Bit 1:2 Packet type -00: unicast -10: multicast -11: Broadcast <sup>1</sup>
IPv6	Bit 0: Packet is a fragment (IPv6 Fragment Extension Header present) Bit 1: Packet type -0: unicast -1: multicast Bit 2: Hop-by-Hop Options (0) header present
GRE	Bit 0: RFC1701 R bit set Bit 1: Unknown protocol (not IPv4 or IPv6) Bit 2: reserved
MinEncap	Bit 0: Original Source Address Present Bit 1: Unknown protocol (not TCP/UDP/SCTP/DCCP/IPSec) Bit 2: reserved

**Note:**

<sup>1</sup> Hard parser can only detect local network broadcasts but soft parser can further refine broadcast detection by applying a subnet mask

Table 5-351 lists the L3R error result codes.

**Table 5-351. L3R Error Result Codes**

Type	Code	Result	Description
Common	0	No Error	—
	1	Reserved	—
	2	Frame Truncation	Parser reached end of frame while parsing the header
Soft Parsing	3–7	User defined	—
IPv4	3	IP Packet Truncation	The IPv4 total length field exceeds the received packet length (excludes L2 header).
	4	IP Checksum error	IPv4 checksum is incorrect
	5	IP Version error	Version number in IPv4 packet is not equal to 4
	6	Minimum Fragment Size error	An IP fragment does not contain at least eight data bytes and this is not the last fragment.
	7	Header Length error	IPv4 header length is less than 20 bytes, or exceeds the received packet length (excludes L2 header), or exceeds the IPv4 header Total Packet Length.

**Table 5-351. L3R Error Result Codes (continued)**

Type	Code	Result	Description
IPv6	3	IP Truncation	The IPv6 Payload length + 40 bytes exceeds the received packet length (excludes L2 header). If Payload length is equal to 0 (indicates hop-by-hop Jumbo payload), then this causes this error code to be generated as this option is not supported.
	4	Extension Header Violation	Indicates extension headers applied that are in violation of the IPv6 specification. Currently indicates the Hop by Hop header did not immediately follow the IPv6 main header or there is more than one Hop by Hop header.
	5	IP Version error	Version number in packet is not equal to 6
	6	Routing Header Error	A routing header of type 0 encountered with header length not an even value or larger than twice the segs left.
GRE	3	GRE version error	GRE version is non zero
	4–7	Reserved	—
MinEnc	3	Reserved	—
	4	MinEnc Checksum error	Minimum encapsulation checksum is incorrect
	5–7	Reserved	—

If an L3 error is encountered, the parsing is stopped for this frame and the soft extension attached to the HXS is invoked if enabled. In the case where the soft extension re-enters the HXS or the soft extension is not enabled, parsing is terminated without having the Line-up Confirmation Vector updated and the frame is passed to the next stage. Since the Line-up Confirmation Vector is not updated for this hard HXS, no operations using fields parsed by Layer 3 should be performed. However other operations using fields that have been successfully validated by other HXSs (for example, Ethernet HXS) can be performed.

#### 5.9.4.7.14 L4 HXS—TCP

The TCP HXS updates the Parse Array with the following information:

- The L4 Offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the TCP header.
- Next Header is loaded with DP
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the TCP header (L5 offset)
- L4 Result (L4R)

The following error checks are performed on the TCP header:

- Checksum error. The checksum of the TCP datagram is validated. This requires access to values extracted by IPv4/IPv6/Minimal Encapsulation HXS to form a pseudo header. The length value in the pseudo header is taken from the Parse Array IP Length field which is updated by the IPv4/IPv6/Minimal Encapsulation HXS. The protocol value used in the pseudo header is taken from the lower byte of the Parse Array Next Header field. In order to validate the checksum, the Ethernet FCS must be removed from the frame Running Sum by BMI.

If the checksum is validated, a notification is set in the L4 Result as well as the L4CV bit (bit 29)

in FD[STATUS].

Checksum is not validated on:

- packets in which the Layer 3 is exclusively parsed by the soft parser,<sup>1</sup>
- packets which entered the parser at the TCP HXS
- on IP fragment packets,
- on packets with padding whose frame size is not equal to 60<sup>2</sup> bytes (as defined by Ethernet short packet padding)
- DCL4C bit in the Parse Array is set to 1; this disables the checking of the TCP checksum
- invalid offsets or truncation errors as defined below
- Invalid offset; set to a value smaller than 5
- TCP truncation: data offset greater than Parse Array IP Length (can only be validated if IP is parsed by L3 HXS)
- Invalid combination of flags; combination of SYN, FIN and/or RESET, no flag set

It indicates in the result field if options are present but it does not examine the option fields (a soft extension could be attached to do so). [Figure 5-305](#) shows the TCP (RFC793).

0	7   8	15   16	24   25	31
	Ports <sub>src</sub>		Port <sub>dst</sub>	
		sequence number		
		ack number		
data offset	reserved	flags	window	
			urgent pointer	
		options	padding	

**Figure 5-305. TCP (RFC793)**

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the TCP HXS updates is overwritten.

Parsing ends when execution of the TCP HXS completes.

#### 5.9.4.7.15 L4 HXS—UDP

The UDP HXS updates the Parse Array with the following information:

- The L4 header offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the UDP header.
- Next Header is loaded with DP
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the UDP header (L5 offset)
- L4 Result (L4R)

1. If the soft parser populates the IP SA, DA, version and length fields of the Parse Array then the checksum can be enabled by setting the IPValid field in the parse result portion of the Parse Array.

2. From the Ethernet standard, short packets are padded to form 64-byte frames. This stipulation includes the 4-byte FCS. The presence of the FCS must be indicated in the FD[STATUS] field.

The following error checks are performed on the UDP header:

- Checksum error. The checksum of the UDP datagram is validated. Checksum calculation for UDP over IPv4 packets is optional, and if a frame arrives with a UDP checksum of zero, no validation is carried out. IPv6, on the other hand, requires that a checksum always be calculated, and if a packet arrives with a 0 checksum, it resolves to an error. Checksum calculation requires access to values extracted by IPv4/IPv6/Minimal Encapsulation HXS to form pseudo header. The length value in the pseudo header is derived from the UDP header regardless of IP version. The protocol value used in the pseudo header is taken from the lower byte of the Parse Array Next Header field. If the UDP checksum is non zero, the result of calculating the checksum over UDP plus the UDP checksum must net a result of 0xffff to be valid. In order to validate the checksum, the Ethernet FCS must be removed from the frame Running Sum by BMI.

If the checksum is validated, a notification is set in the L4 Result as well as the L4CV bit (bit 29) in FD[STATUS].

Checksums are not validated on:

- packets in which the Layer 3 is exclusively parsed by the soft parser,<sup>1</sup>
- packets which entered the parser at the UDP HXS,
- on IP fragment packets,
- on packets with padding whose frame size is not equal to 60<sup>2</sup> bytes (as defined by Ethernet short packet padding), or
- on packets with UDP padding (UDP length < IP length), or length errors
- DCL4C bit in the Parse Array is set to 1; this disables the checking of the UDP checksum
- Length error; UDP length must be equal or greater than 8. As well the UDP length must fit inside the Parse Array IP Length when not an IP fragment. If Layer 3 is parsed by the soft HXS or the parser started at L4, then it is impossible for the hard HXS to detect truncation errors.

[Figure 5-306](#) shows the UDP (RFC768).

0	7   8	15   16	24   25	31
	Portsrc		Portdst	
	Length		Checksum	

**Figure 5-306. UDP (RFC768)**

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the UDP HXS updates is overwritten.

Parsing ends when execution of the UDP HXS completes.

1. If the soft parser populates the IP SA, DA version and length fields of the Parse Array then the checksum can be enabled by setting the IPValid field in the parse result portion of the Parse Array.

2. From the Ethernet standard, short packets are padded to form 64-byte frames. This stipulation includes the 4-byte FCS. The presence of the FCS must be indicated in the FD[STATUS] field.

#### 5.9.4.7.16 L4 HXS—IPSec

The IPSec HXS updates the Parse Array with the following information:

- The L4 header offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the IPSec header.
- Next Header is loaded with IPSec AH next header field for AH(51) header, and loaded with zeros for IPSec ESP.
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the sequence number.
- L4 Result with (L4R)

Other than block limit and frame truncation, no error checks are performed on the IPSec header. Parsing does not proceed past the sequence number. The header, up to and including the sequence number is removed from the running sum.

Figure 5-307 shows the AH (RFC2402).

0	7   8	15   16	24   25	31
Next Header	Payload Length		Reserved	
Security Parameters Index				
Sequence Number				
Authentication Header data (variable)				

Figure 5-307. AH (RFC2402)

Figure 5-308 shows the ESP (RFC2406).

0	7   8	15   16	24   25	31
Security Parameters Index				
Sequence Number				
ESP Payload data (variable)				
Padding (Variable)		Pad Length		Next Header
ESP Authentication data (variable)				

Figure 5-308. ESP (RFC2406)

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the IPSec HXS updates is overwritten.

Parsing ends when execution of the IPSec HXS completes.

#### 5.9.4.7.17 L4 HXS—SCTP

The SCTP HXS updates the Parse Array with the following information:

- The L4 header offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the SCTP header.
- Next Header is loaded with destination port
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the destination port.

- L4 Result (L4R)

Other than block limit and frame truncation, no error checks are performed on the SCTP header. Parsing does not proceed past the destination port. The header, up to and including the destination port is removed from the running sum.

[Figure 5-309](#) shows the SCTP (RFC2960).

0	7   8	15   16	24   25	31
	Portsrc		Portdst	
		Verification tag		
		Checksum		

**Figure 5-309. SCTP (RFC2960)**

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the SCTP HXS updates is overwritten.

Parsing ends when execution of the SCTP HXS completes.

#### 5.9.4.7.18 L4 HXS—DCCP

The DCCP HXS updates the Parse Array with the following information:

- The L4 header offset; byte position (0–255) within frame head where parsing reached the point considered to be the start of the DCCP header.
- Next Header is loaded with destination port
- Next Header Offset; byte position (0–255) within frame head where parsing reached the point considered to be the first byte following the destination port.
- L4 Result (L4R)

Other than block limit and frame truncation, no error checks are performed on the DCCP header. Parsing does not proceed past the destination port. The header, up to and including the destination port is removed from the running sum.

[Figure 5-310](#) shows the DCCP (RFC4340).

0	7   8	15   16	24   25	31
	Portsrc		Portdst	
	Data Offset	CCVal	CsCov	Checksum
Reserved	Type	X		Sequence Number
				Sequence Number cont

**Figure 5-310. DCCP (RFC4340)**

If this HXS is re-entered after it has completed, existing data in the Parse Array fields that the DCCP HXS updates is overwritten.

Parsing ends when execution of the DCCP HXS completes.

#### 5.9.4.7.19 L4 HXS—Other L4 Shell

The Other L4 Shell does not provide any header parsing and validation results. It can act as a termination point for the parsing or entry point to a soft HXS.

The Other L4 Shell HXS does not update the Parse Array. Since Next Header Offset has been set by the HXS that jumped to Other L4 Shell, it represents the byte position (0-255) within frame head where parsing reached the point considered to be the start of the unknown L4 header.

Parsing ends when execution of the Other L4 Shell HXS completes.

#### 5.9.4.7.20 L4 HXS—L4 Results

[Table 5-352](#) shows the L4R decoded information.

**Table 5-352. L4R Decoded**

L4 Type Bit 0:2	Result Type Bit 3	Result Bit 4:7
0: no parsing executed or soft parsing executed	0: info	User-defined
	1: error	See <a href="#">Table 5-353</a> type 0
1: TCP	0: info	Bit 4: Option present (that is, offset value higher than 5) Bit 5: Control bits 6-11 set (one or many of URG, ACK, PSH, RST, SYN, FIN bits are set) Bit 6: Control bits 3-5 set (one or many of NS, CWR, ECE bits are set) Bit 7: Checksum validation is performed
	1: error	See <a href="#">Table 5-353</a> type 1
2: UDP	0: info	Bit 4: Checksum non zero Bit 5-6: reserved Bit 7: Checksum validation is performed
	1: error	See <a href="#">Table 5-353</a> type 2
3: IPSec	0: info	Bit 4: ESP found (50) Bit 5: AH found (51) Bit 6-7: unused
	1: error	See <a href="#">Table 5-353</a> type 3
4: SCTP	0: info	User defined
	1: error	See <a href="#">Table 5-353</a> type 4
5: DCCP	0: info	User defined
	1: error	See <a href="#">Table 5-353</a> type 5
6-7: Reserved	0	0

[Table 5-352](#) shows the L4R result codes.

**Table 5-353. L4R Result Codes**

Type	Code	Result	Description
Common	0	Reserved	—
	1	Frame Truncation	Parser reached end of frame while parsing the header.
0: soft parsing	2–15	User defined	—
1: TCP	2	Invalid Offset	Set to a value smaller than 5
	3	TCP truncation	Data offset greater than IP len
	4	Checksum error	The TCP checksum is incorrect
	5	Bad flags	Following combinations of SYN, FIN and/or RESET being set: SYN/FIN SYN/RST FIN/RST SYN/FIN/RST None of the flags being set is considered “bad.”
	6–15	Reserved	—
	2	Length error	UDP length must be equal to or greater than 8. In addition, the UDP length must fit inside the IP length when it is not a fragment.
2: UDP	3	Reserved	—
	4	Checksum error	The UDP checksum is incorrect. Zero checksum is considered disabled and is not tested for IPv4.
	5–15	Reserved	—
	2–15	Reserved	—
3: IPSec	2–15	Reserved	—
4: SCTP	2–15	Reserved	—
5: DCCP	2–15	Reserved	—

If an L4 error is encountered, the parsing is stopped for this frame and the soft extension attached to the HXS is invoked if enabled. In the case where the soft extension re-enters the HXS or the soft extension is not enabled, parsing is terminated without having the Line-up Confirmation Vector updated and the frame is passed to the classification stage. Since the Line-up Confirmation Vector is not updated for this hard HXS, no operations using fields parsed by Layer 4 should be performed. However other operations using fields that have been successfully validated by other HXSs (for example, Ethernet HXS) can be performed.

#### 5.9.4.7.21 Shim Result

The ShimR can be used to report results and errors encountered while soft parsing. If the Status bit of the ShimR is set (error), then further parsing of the frame by the hard HXS is skipped but classification can be

attempted. The ShimR is primarily intended for use by the soft HXS. [Table 5-354](#) shows ShimR decoded information.

**Table 5-354. ShimR Decoded**

ShimR Type Bit 0:1	Status Bit 2	Result Bit 3:7
00: No parsing executed 01: Shim 1 result 10: Shim 2 result 11: Shim 3 result	0: Info	Software Defined
	1: Error	0: no error 1: Reserved 2: Frame truncation: Soft Parser reached end of frame while soft parsing the header 3–31: Software Defined

### 5.9.4.8 256 Limit Event Handling

The maximum data per frame that can be read in and parsed is 256 bytes. This is the block limit. If the parser reaches this limit before completing all parsing, it sets the BLE bit (bit 28) of FD[STATUS], halt all further processing of the current frame and back out the parse result to the previous successful parsed HXS. No soft attachment is run. Any changes in the rest of the parse array remain as they are. Upon exiting the parser, it should appear as if the limit encountering HXS was never parsed. If there is no previous HXS then the PR will be reset to initial values. If in the middle of a soft sequence attachment when the 256 limit is encountered, everything a hard HXS would have changed is backed out to the previous HXS. This does not include any other changes the soft parsing may have done to the rest of the parse array or even other HXS results.

The following fields are always backed out: Running sum, Next Header, Next Header Offset, LCV, Class Plan and layer2–4/shim statistics.

The following fields do not change:

- Port ID
- Shim result
- Shim offsets

The following are HXS-specific:

Ethernet:

- Ethernet offset is set to initial value.
- Last Ether type offset is set to initial value.
- L2Results are returned to previous HXS state

VLAN offset:

- At first vlan, VLAN Offset 1, VLAN Offset N and Last Etype Offset revert to initial state. L2Results are returned to previous HXS state
- At second vlan, VLAN Offset 1 is kept and VLAN Offset N & Last Etype Offset revert to offset1. L2Results are returned to previous VLAN state
- At nth vlan, VLAN Offset 1 is kept and VLAN Offset N & Last Etype Offset revert to nth-1 offset. L2Results are returned to previous VLAN state

#### LLC/SNAP:

- LLC/SNAP offset is set to initial value.
- Last Ether type offset is set to previous HXS offset value.
- L2Results are returned to previous HXS state

#### PPPoE:

- PPPoE offset is set to initial value.
- Last Ether type offset is set to previous HXS offset value.
- L2Results are returned to previous HXS state

#### MPLS:

- At first label, MPLS Offset 1, MPLS Offset N and Last Etype Offset revert to initial state.  
L2Results are returned to previous HXS state
- At second label, MPLS Offset 1 is kept and MPLS Offset N & Last Etype Offset revert to offset 1.  
L2Results are returned to previous MPLS state
- At nth label, MPLS Offset 1 is kept and MPLS Offset N & Last Etype Offset revert to nth-1 offset.  
L2Results are returned to previous MPLS state

#### IPv4:

- At first IP header, IP Offset 1, IP pid Offset and IP Offset 2 revert to initial state.  
All L3Results are returned to previous HXS
- At second IP header, IP Offset 1& IP pid Offset is kept and IP Offset 2 reverts to offset 1.  
First L3 results are kept. Second L3Results are returned to previous HXS

#### IPv6:

- At first IP header, IP Offset 1, IP pid Offset and IP Offset 2 revert to initial state.  
All L3Results are returned to previous HXS
- At second IP header, IP Offset 1& IP pid Offset is kept and IP Offset 2 reverts to offset 1.  
First L3 results are kept. Second L3Results are returned to previous HXS

#### GRE:

- GRE offset is set to initial value.
- L3Results are returned to previous HXS

#### Minimal Encapsulation:

- MinEnc offset is set to initial value.
- L3Results are returned to previous HXS

#### TCP,UDP,IPsec,SCTP,DCCP:

- L4 offset is set to initial value.
- L4 Results are returned to previous HXS

#### Soft parse only:

- no additional back out

### 5.9.4.9 Soft HXS Parse Array updates

When a soft sequence comes to an end there are a variety of fields in the Parse Array that must be populated in order for later stages to correctly interpret the parse results. These fields are set automatically in the hard HXS but some of these fields may need to be explicitly set by the soft sequence upon returning to the hard HXS, branching to the next soft or hard HXS or terminating. The fields in question are detailed in [Table 5-355](#).

**Table 5-355. Parse Array Required Field Population**

Parse Array Fields	Returning to the hard HXS	Branching to the Next Soft or Hard HXS or Terminating
Classification Plan ID	Done by Hard HXS	Only if equivalent Hard HXS required
Next Header	Done by Hard HXS	Must update
Running Sum	Must update if additional parsing beyond the window offset of the hard HXS <sup>1</sup>	Must update <sup>1</sup>
Lineup Confirmation Vector	Done by Hard HXS	Must update <sup>1</sup>
HXS Offsets	Done by Hard HXS	Must update
Last E type Offset	Done by Hard HXS	Update if Ethernet, VLAN & LLC+SNAP HXS equivalent is parsed
Next Header Offset	Done by Hard HXS	Must update
<b>Note:</b>		

<sup>1</sup> Soft Parser has special instructions for this update

### 5.9.4.10 Parse Result

The Parse Result, shown in [Table 5-356](#), is the parse output of the parser. It is written to the FMan memory internal context (IC). The Parse Result contains information such as the headers offsets, the Logical Port ID and the parse result vector for each layer.

The parser stores all parsing results in a 32-byte Parse Result. As a frame is received, BMI sets the values in this array to zero, except for initial default values that are assigned based on a per Rx port configuration. Both the parser and FMan Controller firmware can modify and assign values in this array. This array provides the means for the parser stages to incorporate results from a previously executed stages. The contents of this array are passed along with the frame to other modules (for example, KeyGen module). Values within the array provide the default outcomes of classification, partially determine the searches performed and provide header offsets used in the construction of search keys. The headers offsets and the layer results are parse history values for the frame. Their intent is to record where each layer of header began within the frame head (offset) and some basic result of the examination (for example, header type, validity, variant).

**Table 5-356. Parse Result**

	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x0	Logical Port ID [0:7]	ShimR	L2R		L3R		L4R	C.Plan	NxtHdr		Running Sum		Line-up Confirmation Vector			
0x10	ShimOffset_1	ShimOffset_2	IP PID Offset	EthOffset	LLC+SNAPOffset	VLANTCI_Offset_1	VLANTCI_Offset_n	LastEType	PPPoE_Offset	MPLSO_Offset_1	MPLSO_Offset_n	IPOffset_1	IPOffset_n or MInEncapO	GREOffset	L4Offset	NxtHdr Offset

Table 5-357 describes the parse result fields.

**Table 5-357. Parse Result Field Description**

Parse Result Field	Description
Logical Port ID [0:7]	Indicates the Logical Port number of the received frame. This field is set to the Logical Port ID value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the Logical Port ID field.
ShimR [0:7]	Shim header coded result. This field is set to the ShimR value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the ShimR field.
L2R [0:15]	Layer 2 coded result. This field is set to the L2R value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the L2R field.
L3R [0:15]	Layer 3 coded result. This field is set to the L3R value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the L3R field.
L4R [0:7]	Layer 4 coded result. This field is set to the L4R value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the L2R field.
Classification Plan ID [0:7]	This field is set to the Classification Plan ID value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the Classification Plan ID field.
NxtHdr [0:15]	Next Header type. This field is set to the NxtHdr value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the NxtHdr field.
Running Sum	This field is set to the Running Sum value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the Running Sum field.
Line-up Confirmation Vector [0:31]	This field is set to the value stored in the Line-up Confirmation Vector register. See section <a href="#">Section 5.9.4.5.4, “Line-Up Confirmation Vector,”</a> for a detailed description of the Line-up Confirmation Vector register.
ShimOffset_1/2 (Shim Offset 1/2) [0:7]	Shim header offset relative to the beginning of the frame. This field is set to the ShimOffset_1/2 values stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the ShimOffset_1/2 fields.
IP PID Offset [0:7]	IPv4/6 Protocol Identifier offset relative to the beginning of the frame. This field is set to the IP PID Offset value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the IP PID Offset field.
EthOffset [0:7]	Ethernet header offset relative to the beginning of the frame. This field is set to the EthOffset value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the EthOffset field.
LLC+SNAPOffset [0:7]	LLC+SNAP header offset relative to the beginning of the frame. This field is set to the LLC+SNAPOffset value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the LLC+SNAPOffset field.

**Table 5-357. Parse Result Field Description (continued)**

Parse Result Field	Description
VLANTCIOffset_1 [0:7]	The first Vlan (TCI) offset relative to the beginning of the frame. This field is set to the VLANTCIOffset_1 value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the VLANTCIOffset_1 field.
VLANTCIOffset_n [0:7]	The last Vlan (TCI) offset relative to the beginning of the frame. This field is set to the VLANTCIOffset_n value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the VLANTCIOffset_n field.
LastETypeOffset	The last EtherType offset relative to the beginning of the frame. This field is set to the LastETypeOffset value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the LastETypeOffset field.
PPPoEOffset [0:7]	PPPoE header offset relative to the beginning of the frame. This field is set to the PPPoEOffset value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the PPPoEOffset field.
MPLSOffset_1 [0:7]	MPLS1 header offset relative to the beginning of the frame. This field is set to the MPLSOffset_1 value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the MPLSOffset_1 field.
MPLSOffset_n [0:7]	MPLSn header offset relative to the beginning of the frame. This field is set to the MPLSOffset_n value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the MPLSOffset_n field.
IPOffset_1 [0:7]	IPv4/6 header 1 offset relative to the beginning of the frame. This field is set to the IPOffset_1 value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the IPOffset_1 field.
IPOffset_n or MinEncapO [0:7]	IPv4/6 header n offset or Minimum Encapsulation header offset relative to the beginning of the frame. This field is set to the IPOffset_n or Minimum Encapsulation value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the IPOffset_n or Minimum Encapsulation field.
GREOffset	GRE header offset relative to the beginning of the frame. This field is set to the GREOffset value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the GREOffset field.
L4Offset [0:7]	Layer 4 header offset relative to the beginning of the frame. This field is set to the L4Offset value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the L4Offset field.
NxtHdrOffset [0:7]	Next Header type offset. This field is set to the NxtHdrOffset value stored in the Parse Array. See <a href="#">Table 5-334</a> for a detailed description of the NxtHdrOffset field.

### 5.9.4.11 Soft Parser

Soft parser programs are produced through the use of configuration tools.

Note: when writing to bytes 0x38 to 0x3b (FD[format,offset,length]) of the Parse Array, the soft parse program must write a 32 bit value, individual byte write to that range is not supported.

### 5.9.4.12 Parsing Exception handling

#### 5.9.4.12.1 Invalid Soft Parser Instruction

If an invalid (undefined) soft parser instruction is detected, the hardware immediately sets the Program Counter (PC) to 0x3FF (signalling the End of all Parsing) and sets the ISP bit (bit 25) of FD[STATUS].

#### **5.9.4.12.2 Parse Cycle Limit**

In the event that the FMPR\_RPCLIM is exceeded, the parsing on the frame ceases and the FD[STATUS] has the PTE bit set (bit 24). This is a catastrophic event and can terminate at any point in the parse sequence. Any valid results obtained up to this point are reported with the exception of the line up confirmation. A writeback of results, for all information found that would normally require a writeback, is done. Because of the very abrupt termination it is unlikely that any reliable information can be obtained from the results but the frame is not lost and the parser immediately recovers and begin parsing the next frame.

#### **5.9.4.12.3 Parse Error Status**

If an error is reported in any of the result fields (ShimR, L2R, L3R, L4R) of the Parse Result, the FD[STATUS] has the PHE bit set (bit 26) as an indication that an error is found.

#### **5.9.4.12.4 Soft Parser Parsing Events**

The soft parser hardware can detect 2 types of parsing events: Frame Truncation errors and Exceeding 256 bytes of frame block.

If the soft parser ever detects that the Load\_Bits\_FW\_to\_WR instruction is trying to load a byte past the end of the frame (HB+WO+m[0:3]), the parsing will end and Frame Truncation will be reported in ShimR, L2R, L3R or L4R.

Where the error is reported is determined by the following:

- If the soft parse sequence is invoked as a soft extension (SSA), it reports in the L\*R that is being extended. The PHE bit in FD[STATUS] is set if the error reporting is not disabled (configured in Error Disable Mask field of the Soft Sequence Attachment).
- If the soft parser sequence is not invoked as a soft extension (SP) the error is reported in the L\*R specified in the SPERC field in the Parse Array. The PHE bit in FD[STATUS] is not set if the error reporting disable is set (configured in bit 4 of the SPERC field of the Parse Array).

It should be noted that if the SPERC field in the Parse Array is ever written by a store instruction then the remaining sequence must now be considered as SP and not SSA.

If the soft parser ever detects that the Load\_Bits\_FW\_to\_WR instruction is trying to load a byte past the 255 byte (the maximum size of the Frame Head) point (HB + WO + m[0:3]), the parsing ends and everything a hard HXS could have changed is backed out to the previous HXS (see [Section 5.9.4.8, “256 Limit Event Handling”](#)). Note: the header base and window offset combination must be less than 512 bytes for this condition to be detected.

#### **5.9.4.12.5 Frame Received on a Disabled Port or Unsupported Port**

If a frame is received on a port that is disabled or unsupported (only 1–16 are supported), the frame is not parsed according to the received NIA. Instead, the PC is forced to 0x3ff (Null Parse) and the FD[STATUS] FRDP error bit is set (bit 27). This effectively causes a bypass of that frame through the parser.

#### **5.9.4.12.6 Frame Received with 256 Limit or Error(s) in Parse Result/FD[STATUS]**

If a frame is dispatched to the parser with any of the LxR in Parse Result indicating an error, or with any of the 4 parser error bits or 256 limit exceed bit set in the FD[STATUS], then the frame is not parsed and treated as bypass.

#### **5.9.4.13 Parser Debug Functionality**

During packet processing, the FMan can trace packet processing flow and trap a packet. Each FMan execution module that took part in this packet processing can write debug information to the FMan Memory.

The debug sampling points occur:

- At the end of each HXS, just before processing the Soft Sequence Attachment
- After each soft parse instruction while running the soft parser

All FMan flows are initialized at the BMI, so any debug flow must also be initiated at the BMI. Each port can be associated with any of the three debug flow options, one or more, regardless of the other ports. The following pertains one flow but applies to all.

When a port has been associated with a debug flow (A, B, or C), the BMI signals the parser during task dispatch accordingly. The parser does one of the following according to its internal debug configuration:

- Attempt to trap the frame, successful trap would lead to write its debug context and signal (continue debug for this flow) to the next processing module during task dispatch. If a frame is detected by traps related to multiple flows the most verbose trace is applied. Failing to trap leads to the stop of the tracing on subsequent modules.
- Trap in bypass state, act as successful trap.

By definition, if there is no successful trap, the FPM debug attributes do not carry through to the next stage. Frames dispatched to the parser with any of the LxR in Parse Result indicating an error, or with any of the 4 parser error bits or 256 limit exceed bits set in the FD[STATUS], are not parsed and treated as a pass-through. A debug sample is performed during this pass-through to allow debug attributes to propagate.

##### **5.9.4.13.1 Parser Trace**

Parser trace takes place only if debug is enabled during task dispatch and if the frame has been trapped (for this action, trap bypass is considered a successful trap). The Parser writes trace information into the debug section of the internal context (IC). The trace is controlled by the FMPR\_PDC dedicated for each of the three independent debug flows. This register is used to specify the amount of trace information that should be stored when a trace is performed. If a frame is trapped by more than one debug flow, the most verbose trace is applied. During frame dispatch, the current debug offset (4 bytes of granularity) is given. This value is updated according to the amount of trace that is written by the Parser so the updated offset is available to the next module in the frame flow.

The Parser writes the following information into the debug trace:

**Table 5-358. FMan Parser Debug Trace Format**

Verbosity Level		Size	Field Description
Very verbose trace	Verbose trace	Minimum trace	1 byte • The selected verbosity level (2 bits) • Trace size in 4-byte granularity (6 bits)
			1 byte TNUM
			1 byte FMan Controller number
			2 bytes Reserved
			3 bytes Received NIA
	—	—	4 bytes Timestamp
			1 byte Parse start point
			4 bytes Program counter(s)

After the Parser has written its debug information, the final debug offset should point to the next available debug location in the internal context for the next module. The trace size is limited to 16 program counters. If the trace is longer, then it is truncated. The total maximum trace size from the Parser module is 41 bytes.

#### 5.9.4.13.2 Debug Traps

The parser has three programmable sets of debug traps, each of which is dedicated to one of the debug flows. The debug traps for a specific flow are related to each other through boolean operations by forming a combined debug event. There are four debug trap registers per debug flow. If a debug event occurs, the next module is signalled during task dispatch that the frame is in a debug state. The user may choose to bypass the trap, meaning that the trap operation is considered successful for any frame.

### 5.9.5 Parser Initialization Information

#### 5.9.5.1 Start-Up or Restart

After a hard reset on the FMan parser, the parser is disabled.

The parse internal memory is initialized to all zeros after a hard async reset. If an access to the parser is attempted while the initialization is taking place, the access does not complete until the memory initialization is completed. The initialization time is 512 input clocks.

The parser memory must be configured via the CCSR space. For each port planned to be used, the “Port *x* Header Examination Config” must be loaded. If soft parsing is to be used, the program(s) must be loaded into the “Soft Parse Instructions” space.

After the configuration is completed, the parser can be enabled by setting the FMPR\_RPIMAC[PEN] bit.

## 5.10 Frame Manager—Key Generator

### 5.10.1 KeyGen Overview

The FMan key generator, referred to in this document as the KeyGen. The KeyGen module is invoked after the Parser module, and uses the parser module results (PR) which are stored in the Internal Context (IC). See [Section 5.4.3, “FMan Frame Internal Context \(IC\).”](#)

The primary functions of the KeyGen are as follows:

- Holds 32 key generation schemes in internal memory. Each scheme can generate:
  - Frame Queue IDs (FQIDs)
    - The FQID may be used to enqueue the frame (distribution as a result of hashing the key)
    - The FQID is written in the Internal Context Action Descriptor (ICAD)
  - Policer profile Number (PNUM)
  - Custom Classifier Base Address (CCBASE)
    - The CCBASE is used as a starting point for the FMan Controller Custom Classifier (CC).
    - The CC may generate a new FQID which overrides the value generated by the KeyGen. This is used for control frames which do not need to be distributed, but need to be enqueued to specific queues.
- For each incoming frame a key generation scheme is selected based on the parser results.
- Generates a key according to the Parser result and calculates the FQID and PNUM without looking in tables

#### 5.10.1.1 KeyGen Features Summary

KeyGen features include the following:

- Key generation up to a 56-byte key
- Key field extractions are based on the Parser result (PR).
  - Classification plan ID and line-up confirmation vector (LCV) gives the scheme number.
  - Headers/Layers results
  - Headers offsets
- Hash function: 64-bit CRC
  - Can shift the hash result
  - Supports symmetric hash for source and destination pairs:
    - MAC source and destination addresses
    - IP source and destination addresses
    - Layer 4 source and destination ports
- Supports port-based partition
  - Scheme port partition
  - Classification plan group port partition

- Up to N classification plans (CPs) (see the applicable device reference manual for the exact number of classification plans)
  - Each 8 classification plans can be in a different port partition.
- Up to N KeyGen schemes (see the applicable device reference manual for the exact number of schemes)
  - Each scheme can be in different port partition.
  - Specify key generation
  - Specify hash usage
  - Specify queuing function (hash, mask/rotate, base addition, logical “OR” extracted field)
  - Specify Policer profile function (mask/shift, base addition)
  - Statistic counter
  - Specify the next invoked action (NIA)
  - Custom classifier root offset generation based on selected bits from the LCV
- Key generation codes
  - Known headers/fields extract bits
  - Generic extract commands
    - Header type—Index number in parser result and type of header
    - Offset—relative to the header offset
    - Extract size in bytes
    - 8-bit mask of the last extracted byte
    - Default value selection if header does not exist
  - Constant value if header is not valid—four configurable options
  - Can extract data of the key from the following:
    - First received frame buffer up to 256 bytes
    - Two global default values
    - Two scheme default values
    - Parse result

## 5.10.2 KeyGen Memory Map/Register Definition

The memory-mapped registers are divided in terms of direct and indirect access. Memory-mapped general configuration and global registers are accessed directly. The rest of the memory map represents the following functions:

- Schemes
- Classification plans
- Port configuration

These are accessed indirectly using FMKG\_AR (Section 5.10.3.11, “KeyGen Action Register (FMKG\_AR)”).

### 5.10.2.1 KeyGen Indirect Memory Space

The KeyGen contains an indirect memory space. This allows for atomic updates to data structures which contain multiple fields. The indirect memory space is used to program three types of data structures:

- KeyGen schemes
- Classification plan
- Port partition

These three data structures are programmed separately. In order to access the indirect space (either for read or for write), the user programs all the relevant fields in a list of registers, and then activates the atomic access with the KeyGen Command Register (by writing to FMKF\_AR register).

### 5.10.2.2 KeyGen Memory Map

This section provides a summary of the KeyGen memory-mapped registers and offsets. The KeyGen base address is 0x0C\_1000. See [Section 5.3.12.1, “FMan Memory Map and partitioning,”](#) for details of KeyGen memory space within FMan memory space.

**Table 5-359. KeyGen Memory Map**

Offset <sup>1</sup>	Register	Access	Reset Value	Section/Page
<b>General Configuration and Status Registers</b>				
0x000	FMKG_GCR—KeyGen General Configuration Register	R/W	0x0000_0000	<a href="#">5.10.3.1/5-418</a>
0x004–0x008	Reserved	—	—	—
0x00C	FMKG_EER—KeyGen Error Event Register	rwm	0x0000_0000	<a href="#">5.10.3.2/5-418</a>
0x010	FMKG_EEER—KeyGen Error Event Enable Register	R/W	0x0000_0000	<a href="#">5.10.3.3/5-419</a>
0x014–0x018	Reserved	—	—	—
0x01C	FMKG_SEER—KeyGen Scheme Error Event Register	rwm	0x0000_0000	<a href="#">5.10.3.4/5-420</a>
0x020	Reserved	—	—	—
0x024	FMKG_GSR—KeyGen Global Status Register	R	0x0000_0000	<a href="#">5.10.3.5/5-420</a>
<b>Global Statistic Counters</b>				
0x28	FMKG_TPC—KeyGen Total Packet Counter	R	0x0000_0000	<a href="#">5.10.3.6/5-421</a>
0x2c	FMKG_SERC—KeyGen Soft Error Capture	Mixed	0x0000_0000	<a href="#">5.10.3.7/5-421</a>
0x30–0x3F	Reserved	—	—	—
<b>KeyGen Global Registers</b>				
0x40	FMKG_FDOR—KeyGen Frame Data Offset Register	R/W	0x0000_0000	<a href="#">5.10.3.8/5-422</a>
0x44	FMKG_GDV0R—KeyGen Global Default Value 0 Register	R/W	0x0000_0000	<a href="#">5.10.3.9/5-422</a>
0x48	FMKG_GDV1R—KeyGen Global Default Value 1 Register	R/W	0x0000_0000	<a href="#">5.10.3.9/5-422</a>
0x4C–0x5F	Reserved	—	—	—
<b>Internal Debug Registers</b>				

**Table 5-359. KeyGen Memory Map (continued)**

Offset <sup>1</sup>	Register	Access	Reset Value	Section/Page
0x060	Reserved	—	—	—
0x064	FMKG_FEER—KeyGen Force Error Event Register	R/W	0x0000_0000	<a href="#">5.10.3.10/5-423</a>
0x068–0x07F	Reserved	—	—	—
<b>Scheme Configuration RAM Registers</b>				
0x100	FMKG_SE_MODE—MODE	R/W	0x0000_0000	<a href="#">5.10.3.12.1/5-426</a>
0x104	FMKG_SE_EKFC—Extract Known Fields Command	R/W	0x0000_0000	<a href="#">5.10.3.12.2/5-427</a>
0x108	FMKG_SE_EKDV—Extract Known Default Value	R/W	0x0000_0000	<a href="#">5.10.3.12.3/5-431</a>
0x10C	FMKG_SE_BMCH—Bit Mask Command High	R/W	0x0000_0000	<a href="#">5.10.3.12.4/5-433</a>
0x110	FMKG_SE_BMCL—Bit Mask Command Low	R/W	0xFFFF_FFFF	<a href="#">5.10.3.12.5/5-435</a>
0x114	FMKG_SE_FQB—Frame Queue Base	R/W	0x0000_0000	<a href="#">5.10.3.12.6/5-436</a>
0x118	FMKG_SE_HC—Hash Command	R/W	0x0000_0000	<a href="#">5.10.3.12.7/5-437</a>
0x11C	FMKG_SE_PPC—Policer Profile Command	R/W	0x0000_0000	<a href="#">5.10.3.12.8/5-438</a>
0x120	FMKG_SE_GEC0—Generic Extract Command 0	R/W	0x00FF_0000	<a href="#">5.10.3.12.9/5-439</a>
0x124	FMKG_SE_GEC1—Generic Extract Command 1	R/W	0x00FF_0000	<a href="#">5.10.3.12.9/5-439</a>
0x128	FMKG_SE_GEC2—Generic Extract Command 2	R/W	0x00FF_0000	<a href="#">5.10.3.12.9/5-439</a>
0x12C	FMKG_SE_GEC3—Generic Extract Command 3	R/W	0x00FF_0000	<a href="#">5.10.3.12.9/5-439</a>
0x130	FMKG_SE_GEC4—Generic Extract Command 4	R/W	0x00FF_0000	<a href="#">5.10.3.12.9/5-439</a>
0x134	FMKG_SE_GEC5—Generic Extract Command 5	R/W	0x00FF_0000	<a href="#">5.10.3.12.9/5-439</a>
0x138	FMKG_SE_GEC6—Generic Extract Command 6	R/W	0x00FF_0000	<a href="#">5.10.3.12.9/5-439</a>
0x13C	FMKG_SE_GEC7—Generic Extract Command 7	R/W	0x00FF_0000	<a href="#">5.10.3.12.9/5-439</a>
0x140	FMKG_SE_SPC—KeyGen Scheme Entry Statistic Packet Counter	R/W	0x0000_0000	<a href="#">5.10.3.12.10/5-443</a>
0x144	FMKG_SE_DV0—KeyGen Scheme Entry Default Value 0.	R/W	0x0000_0000	<a href="#">5.10.3.12.11/5-443</a>
0x148	FMKG_SE_DV1—KeyGen Scheme Entry Default Value 1	R/W	0x0000_0000	<a href="#">5.10.3.12.11/5-443</a>
0x14C	FMKG_SE_CCBS—KeyGen Scheme Entry Custom Classifier Bit Select	R/W	0x0000_0000	<a href="#">5.10.3.12.12/5-444</a>
0x150	FMKG_SE_MV—KeyGen Scheme Entry Match vector	R/W	0x0000_0000	<a href="#">5.10.3.12.13/5-444</a>
0x154	FMKG_SE_OM—KeyGen Scheme Entry Operation Mode bits.	R/W	0x0000_0000	<a href="#">5.10.3.12.14/5-445</a>
0x158	FMKG_SE_VSP—KeyGen Scheme Entry Virtual Storage Profile.	R/W	0x0000_0000	<a href="#">5.10.3.12.15/5-446</a>
<b>Classification Plan Table Access</b>				
<i>(The following eight registers share offsets with FMKG_SE_* registers listed at corresponding offsets above; the registers are used for two purposes in two contexts.)</i>				
0x100	FMKG_CPE0—KeyGen Classification Plan Entry 0	R/W	0xFFFF_FFFF	<a href="#">5.10.3.13.1/5-448</a>
0x104	FMKG_CPE1—KeyGen Classification Plan Entry 1	R/W	0xFFFF_FFFF	<a href="#">5.10.3.13.1/5-448</a>
0x108	FMKG_CPE2—KeyGen Classification Plan Entry 2	R/W	0xFFFF_FFFF	<a href="#">5.10.3.13.1/5-448</a>

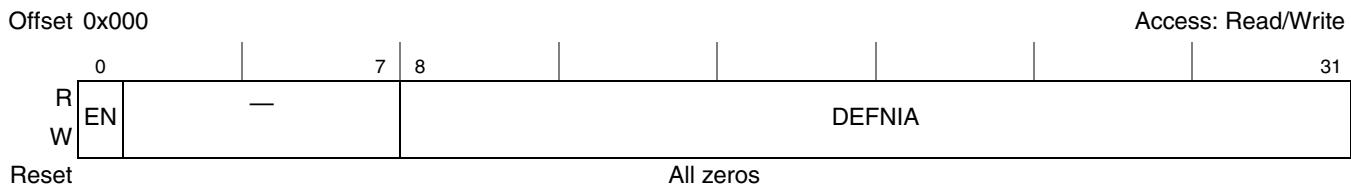
**Table 5-359. KeyGen Memory Map (continued)**

Offset <sup>1</sup>	Register	Access	Reset Value	Section/Page
0x10C	FMKG_CPE3—KeyGen Classification Plan Entry 3	R/W	0xFFFF_FFFF	<a href="#">5.10.3.13.1/5-448</a>
0x110	FMKG_CPE4—KeyGen Classification Plan Entry 4	R/W	0xFFFF_FFFF	<a href="#">5.10.3.13.1/5-448</a>
0x114	FMKG_CPE5—KeyGen Classification Plan Entry 5	R/W	0xFFFF_FFFF	<a href="#">5.10.3.13.1/5-448</a>
0x118	FMKG_CPE6—KeyGen Classification Plan Entry 6	R/W	0xFFFF_FFFF	<a href="#">5.10.3.13.1/5-448</a>
0x11C	FMKG_CPE7—KeyGen Classification Plan Entry 7	R/W	0xFFFF_FFFF	<a href="#">5.10.3.13.1/5-448</a>
<b>Port Partition Configuration</b>				
0x100	FMKG_PE_SP—KeyGen Port entry Scheme Partition	R/W	0xFFFF_FFFF	<a href="#">5.10.3.14.1/5-448</a>
0x104	FMKG_PE_CPP—KeyGen Port Entry Classification Plan Partition	R/W	0x0000_0000	<a href="#">5.10.3.14.2/5-449</a>
<b>Indirect Access Register</b>				
0x1FC	FMKG_AR—KeyGen Action Register	R/W	0x0000_0000	<a href="#">5.10.3.11/5-424</a>
<b>Debug Registers</b>				
0x200	FMKG_DCR—KeyGen Debug Control Register	R/W	0x0000_0000	<a href="#">5.10.3.15.1/5-450</a>
0x204 offset 4 range j=<1 .. 3>	FMKG_D<j>TC—KeyGen Debug <j>Trap Counter	R	0x0000_0000	<a href="#">5.10.3.15.2/5-451</a>
0x210 offset 16 range j=<1 .. 3> K=<1 .. 5>	FMKG_D<j>T<k>CR—KeyGen Debug flow <j>Trap <k> Configuration Register	R/W	0x0000_0000	<a href="#">5.10.3.15.3/5-451</a>
0x214 offset 16 range j=<1 .. 3> K=<1 .. 5>	FMKG_D<j>T<k>VR—KeyGen Debug flow <j> Trap <k> Value Register	R/W	0x0000_0000	<a href="#">5.10.3.15.4/5-454</a>
0x218 offset 16 range j=<1 .. 3> K=<1 .. 5>	FMKG_D<j>T<k>MR—KeyGen Debug flow <j> Trap <k> Mask Register	R/W	0x0000_0000	<a href="#">5.10.3.15.5/5-454</a>

<sup>1</sup> The offset is relative to the KeyGen base = 0xC1000.

### 5.10.3 KeyGen Detailed Register Definitions

### 5.10.3.1 KeyGen General Configuration Register (FMKG\_GCR)

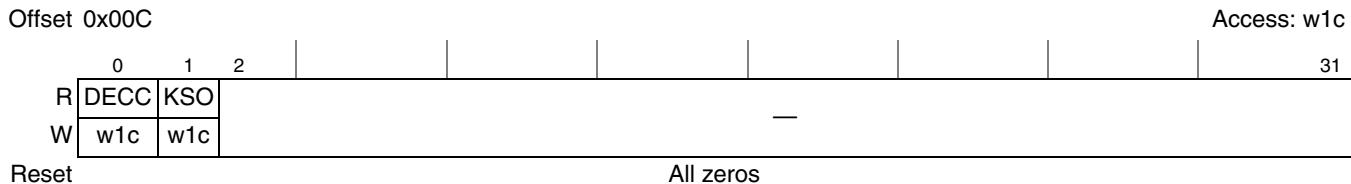


**Figure 5-311. KeyGen General Configuration Register (FMKG\_GCR)**

**Table 5-360. FMKG GCR Field Descriptions**

Bits	Name	Description
0	EN	<p>Enable KeyGen</p> <p>0 KeyGen is disabled and does not accept new jobs. Packets in progress run to completion and are indicated by FMKG_GSR[BSY].</p> <p>1 KeyGen is enabled</p> <p>When EN = 1, the KeyGen is active. It gets jobs, processes packets, and generates results. When EN is cleared, the KeyGen performs a graceful disable sequence. It stops accepting new packets for processing the bus and runs existing packets to completion. During that sequence, the BSY bit can still be asserted. When there are no more packets in progress, the BSY bit is cleared.</p>
1–7	—	Reserved
8–31	DEFNIA	<p>Default next invoked action (NIA)</p> <p>If the selected scheme is disabled, the KeyGen moves the frame task to the next module according to DEFNIA. For NIA format details, see <a href="#">Section 5.10.4.10.1, “KeyGen NIA Action Codes”</a> and <a href="#">Section 5.3.5, “FMan User-Configurable Pipeline Architecture—Introducing the NIA.”</a></p>

### 5.10.3.2 KeyGen Error Event Register (FMKG\_EER)



**Figure 5-312. KeyGen Error Event Register (FMKG\_EER)**

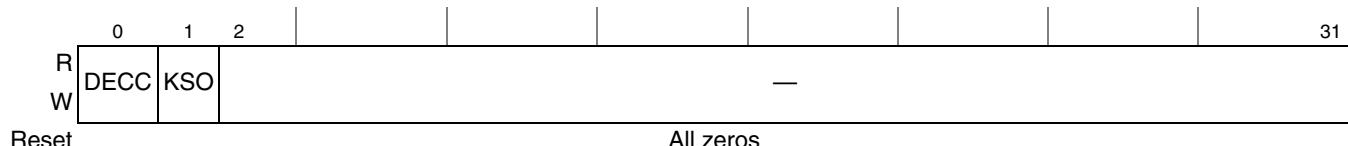
**Table 5-361. FMKG\_EER Field Descriptions**

Bits	Name	Description
0	DECC	<p>Double-bit ECC error is detected on KeyGen internal memory read access. Write ‘1’ to clear. Writing ‘0’ has no effect.</p> <p>0 No double-bit ECC error is detected on any KeyGen internal memory access since the last clearing of this bit.</p> <p>1 Double-bit ECC error is detected on at least one KeyGen internal memory access. If FMKG_EEER[DECC] is set, the KeyGen error interrupt is asserted.</p>
1	KSO	<p>Key size overflow</p> <p>This error bit indicates a scheme error configuration that configures a key size of more than 56 bytes. Corresponding to this bit, a FMKG_SEER[SCx] should be asserted.</p> <p>0 No key size overflow occurs</p> <p>1 Key size overflow occurs on the selected scheme. The selected scheme is mentioned in FMKG_SEER[SCx].</p>
2–31	—	Reserved

### 5.10.3.3 KeyGen Error Event Enable Register (FMKG\_EEER)

Offset 0x010

Access: w1c



**Figure 5-313. KeyGen Error Event Enable Register (FMKG\_EER)**

**Table 5-362. FMKG EEER Field Descriptions**

<b>Bits</b>	<b>Field</b>	<b>Description</b>
0	DECC	<p>Double-bit ECC error interrupt mask</p> <p>0 KeyGen “Double-bit ECC Error” interrupt is masked.</p> <p>1 KeyGen “Double-bit ECC Error” interrupt is enabled. The KeyGen error interrupt line is asserted when FMKG_EER[DECC] is set.</p>
1	KSO	<p>Key size overflow mask</p> <p>0 KeyGen “Key Size Overflow” interrupt is masked.</p> <p>1 KeyGen “Key Size Overflow” interrupt is enabled. The KeyGen error interrupt line is asserted when FMKG_EER[KSO] is set.</p>
2–31	—	Reserved

### 5.10.3.4 KeyGen Scheme Error Event Register (FMKG\_SEER)

Offset 0x01C

Access: w1c

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SC0	SC1	SC2	SC3	SC4	SC5	SC6	SC7	SC8	SC9	SC10	SC11	SC12	SC13	SC14	SC15
W	w1c	w1c	w1c	w1c	w1c	w1c										

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SC16	SC17	SC18	SC19	SC20	SC21	SC22	SC23	SC24	SC25	SC26	SC27	SC28	SC29	SC30	SC31
W	w1c															

Reset

All zeros

Figure 5-314. KeyGen Scheme Error Event Register (FMKG\_SEER)

Table 5-363. FMKG\_SEER Field Descriptions

Bits	Name	Description
0	SC0	Scheme 0 interrupt is detected. Write '1' to clear. Writing '0' has no effect. 0 No event is detected on scheme number N since last clearing of this bit. 1 An event is detected on scheme number N. The event that can assert this bit is a Key Size Overflow.
1-31	SCn	Scheme <i>n</i> interrupt is detected.

### 5.10.3.5 KeyGen Global Status Register (FMKG\_GSR)

Offset 0x024

Access: Read only

	0	1	—	—	—	—	—	—	—	—	—	—	—	—	—	31
R	BSY		—	—	—	—	—	—	—	—	—	—	—	—	—	
W		—	—	—	—	—	—	—	—	—	—	—	—	—	—	

Reset

All zeros

Figure 5-315. KeyGen Global Status Register (FMKG\_GSR)

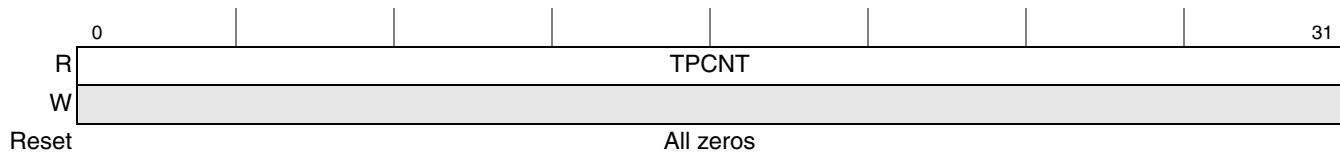
Table 5-364. FMKG\_GSR Field Descriptions

Bits	Name	Description
0	BSY	KeyGen busy indication This bit is continuously reflecting the KeyGen module packet processing state. When disabling the KeyGen (by clearing FMKG_GCR[EN]), use this bit to qualify the completion of a KeyGen graceful stop. 0 KeyGen is not busy. No packets are processed by the KeyGen. If KeyGen is disabled by clearing FMKG_GCR[EN], this bit indicates that KeyGen graceful stop has completed. 1 KeyGen is busy. There are packets in process by the KeyGen. If KeyGen has been disabled by clearing FMKG_GCR[EN], this bit indicates that a KeyGen graceful stop is in progress.
1-31	—	Reserved

### 5.10.3.6 KeyGen Total Packet Counter Register (FMKG\_TPC)

Offset 0x028

Access: Read only



**Figure 5-316. KeyGen Total Packet Counter Register (FMKG\_TPC)**

**Table 5-365. FMKG\_TPC Field Descriptions**

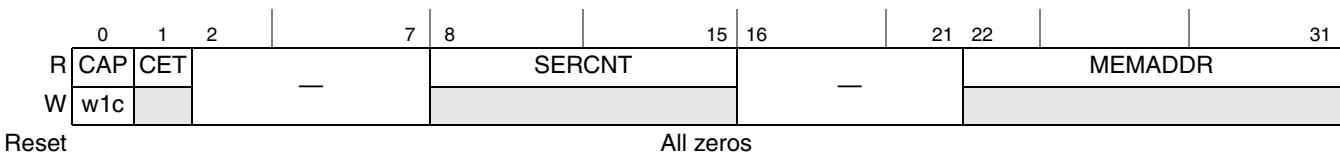
Bits	Name	Description
0–31	TPCNT	Total packet counter This is the total count of packets passed in the KeyGen on all schemes. Writes to this counter are ignored.

### 5.10.3.7 KeyGen Soft Error Capture Register (FMKG\_SERC)

The FMKG\_SERC counts single-bit ECC errors and captures a KeyGen internal memory address of either a single-bit or double-bit ECC error. The first single-bit error detection locks the captured fields from additional single-bit error detections. The first double-bit error detection locks the captured fields from any additional error detection. Software can clear the capture lock to enable additional soft error detections.

Offset 0x02C

Access: Mixed



**Figure 5-317. KeyGen Soft Error Capture Register (FMKG\_SERC)**

**Table 5-366. FMKG\_SERC Field Descriptions**

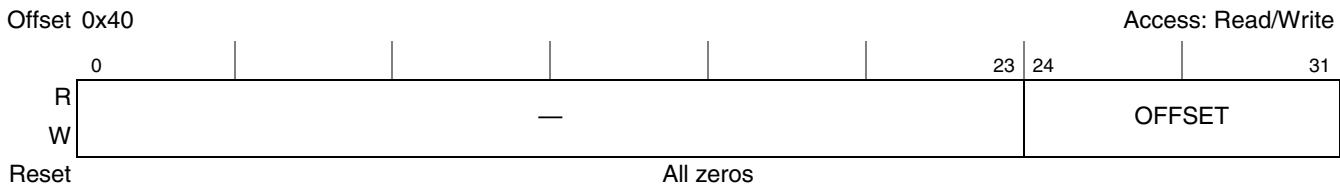
Bits	Name	Description
0	CAP	Captured error indication 0 No error is captured in the MEMADDR field. 1 An ECC error is captured. Any single ECC error sets the CAP bit and the error memory location is captured in the MEMADDR field, and the MEMADDR is locked from capturing a new single ECC error. For a multi-bit ECC error the MEMADDR field holds the memory address of the error location and is locked from re-capturing any other ECC errors.
1	CET	Captured error type 0 Single-bit ECC error is captured if CAP = 1. No ECC error is captured if CAP = 0. 1 Double ECC error is captured. CET is never asserted if CAP = 0.
2–7	—	Reserved

**Table 5-366. FMKG\_SERC Field Descriptions (continued)**

Bits	Name	Description
8–15	SERCNT	Soft error counter This counter accumulates single-bit ECC errors that have been detected and corrected by the KeyGen internal memory ECC logic. Double-bit ECC errors are not counted and directly set the FMKG_EER[DECC] error event bit. If FMKG_EEER[DECC] = 1, double-bit ECC would also assert the KeyGen error interrupt. This field is cleared on read. When the count reaches its maximum value of 0xFF, it does not rollover until this register is read.
16–21	—	Reserved
22–31	MEMADDR	Captured memory address of access that caused ECC error When CAP = 1 and CET = 0, this field is locked from additional single ECC errors but could capture double-bit ECC errors. When CAP = 1 and CET = 1, the field is locked from capturing either single-bit or double-bit ECC errors. Valid when CAP is set. Cleared when CAP is written with a ‘1.’ The ECC is checked for 4 bytes therefore the address is given in 4-byte units.

### 5.10.3.8 KeyGen Frame Data Offset Register (FMKG\_FDOR)

The KeyGen can extract data from the start of the frame up to the last header that was parsed by the Parser. At the end of parsing, the frame offset points to the payload of the last parsed header. To allow the KeyGen to extract also from the non-parsed payload, configure FMKG\_FDOR[OFFSET].



**Figure 5-318. KeyGen Frame Data Offset Register (FMKG\_FDOR)**

**Table 5-367. FMKG\_FDOR Field Descriptions**

Bits	Name	Description
0–23	—	Reserved
24–31	OFFSET	Offset to the end of the parsing point 00 Extract data only from the parsed headers. 01 Enable extract from the frame up to 1 byte from the non parsed payload. 10 Enable extract from the frame up to 2 bytes from the non parsed payload. ... FF Enable extract from the frame up to 255 bytes from the non-parsed payload. The KeyGen extract limit is the lower value between the two: frame length or the 256 bytes buffer size.

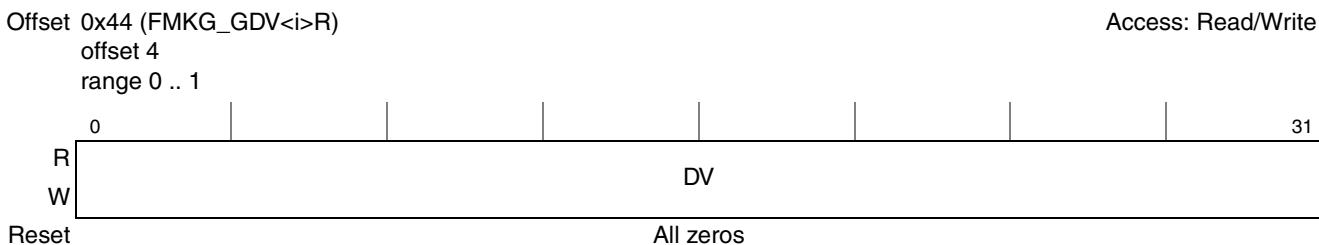
### 5.10.3.9 KeyGen Global Default Value Register (FMKG\_GDV*<i>*R)

The FMKG\_GDV*<i>*R is used when an extracted field does not exist in the frame. If a header does not exist in the processed frame and there is an extract command for this field, a default value is used. The user

can select from four different values. The default values are configured by the user in the FMKG\_GDV0–1R and two per scheme in FMKG\_SE\_DV0–1.

This is a debug/verification assist register.

This register is for internal use. It is recommended not to modify its reset value. This value can be modified in a system which requires debug.



**Figure 5-319. KeyGen Global Default Value Register (FMKG\_GDV<i>R)**

**Table 5-368. FMKG\_GDV*<i>R* Field Descriptions**

Bits	Name	Description
0–31	DV	Default value

#### 5.10.3.10 KeyGen Force Error Event Register (FMKG\_FEER)

The FMKG\_FEER is not a physically-implemented register. Rather, it is an address to which the FMKG\_EER can be written. Therefore, this register is write-only and writing 0 to it has no affect. Writing 1 to this register sets the corresponding bit of the error interrupt event register. Reading from this register always provides 0x0.



**Figure 5-320. KeyGen Force Error Event Register (FMKG\_FEER)**

**Table 5-369. FMKG FEER Field Descriptions**

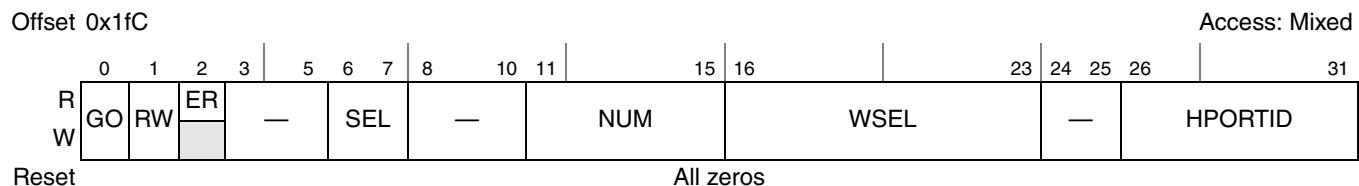
Bits	Name	Description
0	FEDECC	Force event double ECC error Force to set the DECC bit of the FMKG_EER register 0 Any reading from this field returns zero. Writing zero to this field has no effect. 1 Force to set the DECC bit of the FMKG_EER register

**Table 5-369. FMKG\_FEER Field Descriptions (continued)**

Bits	Name	Description
1	FEKSO	Force event key size overflow mask 0 Any reading from this field returns zero. Writing zero to this field has no effect. 1 Force to set the KSO bit of the FMKG_EER register
2–31	—	Reserved

### 5.10.3.11 KeyGen Action Register (FMKG\_AR)

Schemes, classification plan, and port configuration registers are accessed indirectly by FMKG\_AR. By selecting the FMKG\_AR[SEL] function, the application may choose to configure one of these spaces. Offset 0x100 and onward in the memory map is used alternately as the data memory for configuring schemes, classification plan, and port configuration. For example, when FMKG\_AR[SEL] = 0, address 0x100 represents FMKG\_SE\_MODE, but if FMKG\_AR[SEL] = 1, address 0x100 represents FMKG\_CPE0.



**Figure 5-321. KeyGen Action Register (FMKG\_AR)**

**Table 5-370. FMKG\_AR Field Descriptions**

Bits	Name	Description
0	GO	GO GO activates the atomic scheme entry access Writing ‘1’ when GO = 0 starts an atomic read or atomic write access to the selected scheme entry, according to the setting of RW, NUM, PORTID, SEL and WSEL. The GO bit is self-clearing at the end of the atomic access. When GO = 1 all the accesses to the indirect registers and to the FMKG_AR are held until the GO bit is cleared. 0 KeyGen atomic access is idle. 1 KeyGen atomic access is in progress. The access types, entry access selection, and field-select bits are configured RW by NUM, PORTID, and WSEL.
1	RW	Read/Write access type 0 Atomic access is a write. At the end of the atomic access, the contents of all AWR* configuration registers associated with set WSEL mask bits are copied into the entry selected by NUM. The write access is atomic and does not interfere with the selected entry updates due to packet processing. 1 Atomic access is a read. At the end of the atomic access, the contents of the entry selected by NUM are copied to their associated AWR* configuration registers. The read access is atomic and does not get interference by the selected entry updates due to packet processing.
2	ER	Error bit 0 No error occurred 1 Error occurred; access to a wrong scheme port partition
3–5	—	Reserved

**Table 5-370. FMKG\_AR Field Descriptions (continued)**

Bits	Name	Description																
6–7	SEL	Select read/write entry 0 Scheme entry 1 Classification plan entry 2 Port partition entry 3 Reserved																
8–10	—	Reserved																
11–15	NUM	This field depends on FMKG_AR[SEL]. <ul style="list-style-type: none"><li>• If SEL = 0: Scheme Number This field selects the scheme number to be accessed by the atomic read/write access.</li><li>• If SEL = 1: Classification Plan Group This field selects the Classification plan group to be accessed by the atomic read/write access.</li><li>• If SEL = 2: This field is ignored.</li><li>• If SEL = 3: This field is ignored.</li></ul>																
16–23	WSEL Where SEL = 0	WSEL[0:7]—word select This field provides a bit-wise selection mask of a specific scheme 32-bit entries for the atomic access. A field associated with set WSEL bit is written by an atomic write from its associated *FMKG_SE_* register. A field associated with cleared WSEL bit is not written by atomic write access. Atomic reads would always access the full scheme words to their associated *FMKG_SE_* registers. When writing a scheme entry, the user should write the entire entry. The only register that the user can choose not to update is the scheme packet counter (*FMKG_SE_SPC). This can be accomplished by not selecting this register by writing 0 in WSEL[0] bit. Following are the WSEL bits associations to configuration registers and scheme entry fields when: SEL = 0—the user should write to all the associated registers.  <b>WSEL Bit Associations SEL = 0</b> <table border="1"><thead><tr><th>WSEL BIT</th><th>Associated Reg.</th><th>Associated KeyGen indirect memory Entry Field</th><th>Section/Page</th></tr></thead><tbody><tr><td>0</td><td>FMKG_SE_SPC</td><td>Statistic Packet Counter</td><td><a href="#">5.10.3.12.10/5-443</a></td></tr><tr><td>1–7</td><td>Reserved</td><td>—</td><td>—</td></tr></tbody></table>	WSEL BIT	Associated Reg.	Associated KeyGen indirect memory Entry Field	Section/Page	0	FMKG_SE_SPC	Statistic Packet Counter	<a href="#">5.10.3.12.10/5-443</a>	1–7	Reserved	—	—				
WSEL BIT	Associated Reg.	Associated KeyGen indirect memory Entry Field	Section/Page															
0	FMKG_SE_SPC	Statistic Packet Counter	<a href="#">5.10.3.12.10/5-443</a>															
1–7	Reserved	—	—															
	WSEL Where SEL = 1	The WSEL bit associations to configuration registers and classification plan entry fields when SEL = 1 are as follows.  <b>WSEL Bit Associations SEL = 1</b> <table border="1"><thead><tr><th>WSEL BIT</th><th>Associated Reg.</th><th>Associated KeyGen indirect memory Entry Field</th><th>Section/Page</th></tr></thead><tbody><tr><td>0–7</td><td>FMKG_CPE0–7</td><td>KeyGen Classification Plan Entry 0–7</td><td><a href="#">5.10.3.13.1/5-448</a></td></tr></tbody></table>	WSEL BIT	Associated Reg.	Associated KeyGen indirect memory Entry Field	Section/Page	0–7	FMKG_CPE0–7	KeyGen Classification Plan Entry 0–7	<a href="#">5.10.3.13.1/5-448</a>								
WSEL BIT	Associated Reg.	Associated KeyGen indirect memory Entry Field	Section/Page															
0–7	FMKG_CPE0–7	KeyGen Classification Plan Entry 0–7	<a href="#">5.10.3.13.1/5-448</a>															
	WSEL Where SEL = 2	The WSEL bit associations to the port partition configuration entry fields when SEL = 2 are as follows  <b>WSEL Bit Associations SEL = 2</b> <table border="1"><thead><tr><th>WSEL BIT</th><th>Associated Reg.</th><th>Associated KeyGen indirect memory Entry Field</th><th>Section/Page</th></tr></thead><tbody><tr><td>0</td><td>FMKG_PE_SP</td><td>KeyGen Port Entry Scheme Partition</td><td><a href="#">5.10.3.14.1/5-448</a></td></tr><tr><td>1</td><td>FMKG_PE_CPP</td><td>KeyGen Port Entry Classification Plan Partition</td><td><a href="#">5.10.3.14.2/5-449</a></td></tr><tr><td>2–7</td><td>Reserved</td><td>—</td><td>—</td></tr></tbody></table>	WSEL BIT	Associated Reg.	Associated KeyGen indirect memory Entry Field	Section/Page	0	FMKG_PE_SP	KeyGen Port Entry Scheme Partition	<a href="#">5.10.3.14.1/5-448</a>	1	FMKG_PE_CPP	KeyGen Port Entry Classification Plan Partition	<a href="#">5.10.3.14.2/5-449</a>	2–7	Reserved	—	—
WSEL BIT	Associated Reg.	Associated KeyGen indirect memory Entry Field	Section/Page															
0	FMKG_PE_SP	KeyGen Port Entry Scheme Partition	<a href="#">5.10.3.14.1/5-448</a>															
1	FMKG_PE_CPP	KeyGen Port Entry Classification Plan Partition	<a href="#">5.10.3.14.2/5-449</a>															
2–7	Reserved	—	—															

**Table 5-370. FMKG\_AR Field Descriptions (continued)**

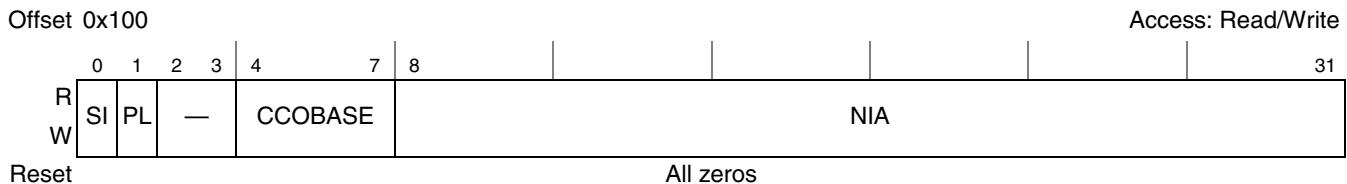
Bits	Name	Description
24–25	—	Reserved
26–31	HPORTID	<p>Hardware portID</p> <p>The hardware portID is required for checking that the partition is not corrupted. It updates the scheme only if the scheme number is part of the hardware portID partition. The reserved portID 0x0 can access and configure all scheme entries and classification plans ignoring the port partition limit (FMKG_PE_SP and FMKG_PE_CPP).</p> <p>The hardware portID is also required for the port partition configuration.</p>

### 5.10.3.12 KeyGen Scheme Entry Memory Map

This section describes the internal memory organization of a single-scheme memory residing in KeyGen indirect memory space. The user access to memory is accomplished via indirect mapping using the atomic access logic. For details on atomic KeyGen indirect memory entry access, see [Section 5.10.3.11, “KeyGen Action Register \(FMKKG\\_AR\).”](#)

The offsets of the scheme entry fields are given relative to the entry base location.

#### 5.10.3.12.1 KeyGen Scheme Entry MODE Register (AWR1\_RFMODE\_FMKG\_SE\_MODE)



**Figure 5-322. KeyGen Scheme Entry MODE Register (AWR1\_RFMODE\_FMKG\_SE\_MODE)**

**Table 5-371. AWR1\_RFMODE\_FMKG\_SE\_MODE Field Descriptions**

Bits	Name	Description
0	SI	<p>Scheme initialization bit</p> <p>0 Scheme has not been explicitly initialized by user. No access to this scheme occurs.</p> <p>1 Scheme has been initialized.</p>
1	PL	<p>Policer type</p> <p>0 The next module is not the Policer. The KeyGen does not add the calculated Policer profile to the scheme NIA (AWR1_RFMODE_FMKG_SE_MODE[NIA]).</p> <p>1 The next module is the Policer. The KeyGen adds the calculated policer profile to the scheme NIA (AWR1_RFMODE_FMKG_SE_MODE[NIA]).</p> <p>See <a href="#">Section 5.10.4.9, “Next Invoke Action (NIA) Generation.”</a></p>
2–3	—	Reserved
4–7	CCOBASE	<p>Custom classifier offset base</p> <p>This field is part of the configuration of the custom classifier base last byte.</p> <p>See <a href="#">Section 5.10.4.8, “Custom Classifier Base (CCBASE) Update.”</a></p>
8–31	NIA	<p>Scheme next invoked action (NIA)</p> <p>For NIA format details, see <a href="#">Section 5.3.5, “FMan User-Configurable Pipeline Architecture—Introducing the NIA.”</a></p>

### 5.10.3.12.2 KeyGen Scheme Entry Extract Known Fields Command Register (AWR2\_RFMODE\_FMKG\_SE\_EKFC)

The user can extract known fields from known headers by selecting the wanted fields in the AWR2\_RFMODE\_FMKG\_SE\_EKFC. The KeyGen extracts the field from the frame based on the Parser result (PR) header offset and layer results. If the header does not exist or is not valid, the KeyGen extracts default value based on FMKG\_SE\_EKDV configuration.

Offset 0x104

Access: Read/Write<sup>1</sup>

R W	0 PORTID	1 MACDST	2 MACSRC	3 VLANTCI_1	4 VLANTCI_N	5 ETYP	6 PPPSID	7 PPPPID
Reset	All zeros							
R W	8 MPLSL_1	9 MPLSL_2	10 MPLSL_N	11 IPSRC_1	12 IPDST_1	13 PTYP	14 IPTOS_TC_1	15 IPV6FL_1
Reset	All zeros							
R W	16 IPSRC_N	17 IPDST_N	18 PTYP	19 IPTOS_TC_N	20 IPV6FL_N	21 GREPTYP	22 IPSECSP	23 IPSECNH
Reset	All zeros							
R W	24 IPPID	—			28 L4PSRC	29 L4PDST	30 TFLG	
Reset	All zeros							

**Figure 5-323. KeyGen Scheme Entry Extract Known Fields Command Register (AWR2\_RFMODE\_FMKG\_SE\_EKFC)**

<sup>1</sup> Indirect by access logic

**Table 5-372. AWR2\_RFMODE\_FMKG\_SE\_EKFC Field Descriptions**

Bits	Name	Description
0	PORTID	Extract 1 byte of portID from the Parse result first byte and add it to the key. 0 Disable 1 Enable The parser results reside in the Internal Context (IC).
1	MACDST	Extract a 48-bit (6-byte) MAC destination address for the key. 0 Disable extraction of this field. 1 Enable extraction of this field.
2	MACSRC	Extract a 48-bit (6-byte) MAC source address for the key. 0 Disable extraction of this field. 1 Enable extraction of this field.

**Table 5-372. AWR2\_RFMODE\_FMKG\_SE\_EKFC Field Descriptions (continued)**

Bits	Name	Description
3	VLANTCI_1	<p>Extract a 16-bit (2-byte) VLAN TCI from the first Q-Tag in the frame.</p> <p>0 Disable extraction of this field. 1 Enable extraction of this field. If a Q-TAG is not present in the frame or is not valid, the default value is used.</p> <p>The default value is chosen by VLANTCIDV.</p> <p>The user may mask some of these bits using either FMKG_SE_BMCH or FMKG_SE_BMCL.</p>
4	VLANTCI_N	<p>Extract a 16-bit (2-byte) VLAN TCI from the last Q-Tag in the frame.</p> <p>0 Disable extraction of this field. 1 Enable extraction of this field. If a Q-TAG is not present in the frame or is not valid, the default value is used.</p> <p>The default value is chosen by VLANTCIDV.</p> <p>The user may mask some of these bits using either FMKG_SE_BMCH or FMKG_SE_BMCL.</p>
5	ETYPE	<p>Extract a 16-bit (2-byte) Ethernet Type field from frame.</p> <p>Extracts the last EType that has been seen. For Ethernet II frames, this corresponds to the 16-bit field following the last encountered VLAN tag extracted from the frame. For IEEE 802.3/SNAP, the 16 bits following the SNAP OUI are extracted. This field is extracted according to PR[ETYPEO].</p> <p>0 Disable extraction of this field. 1 Enable extraction of this field. If Etype is not present in the frame or is not valid, the default value is used. The default value us chosen by ETYPEDV.</p>
6	PPPSID	<p>Extract a 16-bit (2-byte) PPPoE Session ID Field.</p> <p>0 Disable extraction of this field. 1 Enable extraction of this field.If PPP is not present in the frame or is not valid, the default value is used.</p> <p>The default value is chosen by PPPSIDDV.</p>
7	PPPPIID	<p>Extract a 16-bit (2-byte) PPP Protocol ID Field.</p> <p>0 Disable extraction of this field. 1 Enable extraction of this field.If PPP is not present in the frame or is not valid, the default value is used. The default value is chosen by PPPPIDDV.</p>
8	MPLSL_1	<p>Extract 32 bits (4 bytes) of MPLS first label.</p> <p>0 Disable extraction of this field. 1 Enable extraction of this field.If MPLS is not present in the frame or is not valid, the default value is used. The default value is chosen by MPLSLDV.</p>
9	MPLSL_2	<p>Extract 32 bits (4 bytes) of MPLS second label.</p> <p>0 Disable extraction of this field. 1 Enable extraction of this field.If second MPLS is not present in the frame or is not valid, the default value is used. The default value is chosen by MPLSLDV.</p>
10	MPLSL_N	<p>Extract 32 bits (4 bytes) of MPLS last label.</p> <p>0 Disable extraction of this field. 1 Enable extraction of this field.If MPLS is not present in the frame or is not valid, the default value is used. The default value is chosen by MPLSLDV.</p>
11	IPSRC_1	<p>Extract 32-/128-bit (4-/16-byte) IPv4/6 Source Address Field of the first (outer) IP header.</p> <p>0 Disable extraction of this field. 1 Enable extraction of this field.</p> <p>If IPv4/6 headers do not present in the frame the extract size is 4 bytes of the selected default value. If IP header does not valid, the default value is used, with its relative size (IPv4 - 4 bytes, IPv6 16 bytes). The default value is chosen by IPADV.</p>

**Table 5-372. AWR2\_RFMODE\_FMKG\_SE\_EKFC Field Descriptions (continued)**

Bits	Name	Description
12	IPDST_1	Extract 32-/128-bit (4-/16-byte) IPv4/6 Destination Address Field of the first (outer) IP header. 0 Disable extraction of this field. 1 Enable extraction of this field. If IPv4/6 headers are not present in the frame the extract size is 4 bytes of the selected default value. If IP header is not valid, the default value is used, with its relative size (IPv4 - 4 bytes, IPv6 16 bytes). The default value is chosen by IPADV.
13	PTYPE_1	Extract 8-bit (1-byte) IPv4 Protocol Type Field or IPv6 the next header type of the first (outer) IP header. 0 Disable extraction of this field. 1 Enable extraction of this field. If IP header is not present in the frame or is not valid, the default value is used. The default value is chosen by PTYPEDV.
14	IPTOS_TC_1	Extract TOS (IPv4) or Traffic Class(IPv6) of the first (outer) IP header. If IPv4 exist the KeyGen will add to the key 1 byte of TOS field. If IPv6 exist the KeyGen will add 2bytes to the key which hold: IPv6 version (4bits), TC (8bits) and 4bit zero. 0 Disable extraction of this field. 1 Enable extraction of this field. If IPv4/6 headers do not present in the frame the extract size is 1byte of the selected default value. If IP header does not valid, the default value is used, with its relative size (IPv4 - 1byte, IPv6 2bytes). The default value is chosen by IPTOSDV.
15	IPV6FL_1	Extract IPv6 Flow Label of the first (outer) IP header. The KeyGen adds 3 bytes to the key where the 4 most significant bits are zero and the lower 20 bits hold the IPV6 flow label field. 0 Disable extraction of this field. 1 Enable extraction of this field. If IPv6 header is not present in the frame or is not valid, the default value is used. The default value is chosen by IP6FLDV.
16	IPSRC_N	Extract 32-/128-bit (4-/16-byte) IPv4/6 Source Address Field of the last (inner) IP header, or Min Encap Source Address Field. 0 Disable extraction of this field. 1 Enable extraction of this field. If inner IPv4/6 or Min Encap headers are not present in the frame the extract size is 4 bytes of the selected default value. If inner IP header is not valid, the default value is used, with its relative size (IPv4/Min Encap - 4 bytes, IPv6 16 bytes). The default value is chosen by IPADV.
17	IPDST_N	Extract 32-/128-bit (4-/16-byte) IPv4/6 of the last (inner) IP header, or Min Encap Destination Address field. 0 Disable extraction of this field. 1 Enable extraction of this field. If inner IPv4/6 or Min Encap headers are not present in the frame the extract size is 4 bytes of the selected default value. If inner IP header is not valid, the default value is used, with its relative size (IPv4/Min Encap - 4 bytes, IPv6 16 bytes). The default value is chosen by IPADV.
18	PTYPE_N	Extract 8-bit (1-byte) IPv4 Protocol Type Field or IPv6 next header type of the last (inner) IP header, or Min Encap protocol type. 0 Disable extraction of this field. 1 Enable extraction of this field. If inner IP header or Min Encap header are not present in the frame or are not valid, the default value is used. The default value is chosen by PTYPEDV.

**Table 5-372. AWR2\_RFMODE\_FMKG\_SE\_EKFC Field Descriptions (continued)**

Bits	Name	Description
19	IPTOS_TC_N	<p>Extract TOS (IPv4) or Traffic Class(IPv6) of the last (inner) IP header.            If IPv4 exist the KeyGen will add to the key 1 byte of TOS field.            If IPv6 exist the KeyGen will add 2bytes to the key which hold: IPv6 version (4bits), TC (8bits) and 4bit zero.</p> <p>0 Disable extraction of this field.            1 Enable extraction of this field.            If inner IPv4/6 headers do not present in the frame the extract size is 1byte of the selected default value.            If inner IP header does not valid, the default value is used, with its relative size (IPv4 - 1byte, IPv6 2 bytes).            The default value is chosen by IPTOSDV.</p>
20	IPV6FL_N	<p>Extract IPv6 Flow Label of the last (inner) IP header.            The KeyGen will add 3bytes to the key where the most significant 4 bits are zero and the lower 20bits hold the IPv6 flow label field.</p> <p>0 Disable extraction of this field.            1 Enable extraction of this field. If inner IPv6 header is not present in the frame or is not valid, the default value is used. The default value is chosen by IP6FLDV.</p>
21	GREPTYPE	<p>Extract a 16-bit (2-byte) GRE Protocol Type field.</p> <p>0 Disable extraction of this field.            1 Enable extraction of this field. If GRE header is not present in the frame or is not valid, the default value is used. The default value is chosen by ETYPEDV.</p>
22	IPSECSPI	<p>Extract 32 bits (4 bytes) of IPsec SPI field.            Extract this field according to the parser L4R vector.</p> <p>0 Disable extraction of this field.            1 Enable extraction of this field. If IPsec header is not present in the frame or is not valid, the default value is used. The default value is chosen by IPSECSPIDV.</p>
23	IPSECNH	<p>Extract 8 bits (1 byte) of IPsec (AH only) Next Header field.            Extract this field according to the parser L4R vector.</p> <p>0 Disable extraction of this field.            1 Enable extraction of this field. If IPsec header AH is not present in the frame or is not valid, the default value is used. The default value is chosen by PTYPEDV.</p>
24	IPPID	<p>Extract an 8-bit (1-byte) IP Protocol identifier (PID) at PR[IP PID Offset].            If two IP headers are present (tunneling) the PID is taken from the inner IP header.</p> <p>0 Disable extraction of this field.            1 Enable extraction of this field. If IP header is not present in the frame or is not valid, default value is used. The default value is chosen by PTYPEDV.</p>
25-28	—	Reserved
29	L4PSRC	<p>Extract a 16-bit (2-byte) TCP or UDP or SCTP or DCCP source Port Field            Extract this field according to the parser L4R vector.</p> <p>0 Disable extraction of this field.            1 Enable extraction of this field.</p>

**Table 5-372. AWR2\_RFMODE\_FMKG\_SE\_EKFC Field Descriptions (continued)**

Bits	Name	Description
30	L4PDST	Extract a 16-bit (2-byte) TCP or UDP or SCTP or DCCP destination Port Field Extract this field according to the parser L4R vector. 0 Disable extraction of this field. 1 Enable extraction of this field.
31	TFLG	Extract the 14th byte of the TCP header, which contains TCP flags. The user may mask some of these bits, by using AWR2_RFMODE_FMKG_SE_BMCH/L. Extract this field according to the parser L4R vector. 0 Disable extraction of this field. 1 Enable extraction of this field.

### 5.10.3.12.3 KeyGen Scheme Entry Extract Known Default Value Register (AWR3\_RFMODE\_FMKG\_SE\_EKDV)

Offset 0x108

Access: User read/write<sup>1</sup>

R	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	MACADV	VLANTCIDV	ETYPEDV	PPPSIDDV	PPPIDDV	MPLSLDV	IPADV	PTYPEDV								
Reset															All zeros	
R	16	17	18	19	20	21	22	23	24	25	26				31	
W	IPTOSDV	IP6FLDV	IPSECSPIDV	L4PDV	TFLGDV							—				
Reset															All zeros	

**Figure 5-324. KeyGen Scheme Entry Extract Known Default Value Register (AWR3\_RFMODE\_FMKG\_SE\_EKDV)**

<sup>1</sup> Indirect by access logic

**Table 5-373. AWR3\_RFMODE\_FMKG\_SE\_EKDV Field Descriptions**

Bits	Name	Description
0-1	MACADV	Mac Address Default Value Selection. <ul style="list-style-type: none"><li>• If Ethernet header does not exist in the frame replace Mac destination/source address in the Key to the selected default value.</li><li>• Instead of Mac address, put in key {*FMKG_*DV*[0:31],*FMKG_*DV*[0:15]}</li></ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
2-3	VLANTCIDV	VLAN TCI Default Value Selection. <ul style="list-style-type: none"><li>• If first Vlan or last Vlan headers do not exist in the frame replace TCI in the Key to the selected default value.</li><li>• Instead of Vlan TCI, put in key {*FMKG_*DV*[0:16]}</li></ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1

**Table 5-373. AWR3\_RFMODE\_FMKG\_SE\_EKDV Field Descriptions (continued)**

Bits	Name	Description
4–5	ETYPEDV	Etype Default Value Selection. <ul style="list-style-type: none"> <li>If Ethernet header does not exist in the frame, replace Etype in the Key to the selected default value.</li> <li>Instead of Etype, put in key {*FMKG_*DV*[0:16]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
6–7	PPPSIDDV	• PPPSID Default Value Selection. <ul style="list-style-type: none"> <li>If PPP header does not exist in the frame, replace PPPSID in the Key to the selected default value.</li> <li>Instead of PPPSID, put in key {*FMKG_*DV*[0:16]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
8–9	PPPPIDDV	PPPPID Default Value Selection. <ul style="list-style-type: none"> <li>If PPP header does not exist in the frame, replace PPPPID in the Key to the selected default value.</li> <li>Instead of PPPPID, put in key {*FMKG_*DV*[0:16]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
10–11	MPLSLDV	MPLS Label Default Value Selection. <ul style="list-style-type: none"> <li>If MPLS 1, 2, or 3 header does not exist in the frame, replace MPLS 1, 2, or 3 in the Key to the selected default value.</li> <li>Instead of MPLS, label put in key {*FMKG_*DV*[0:31]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
12–13	IPADV	IP Address Default Value Selection. <ul style="list-style-type: none"> <li>If IP header (IPv4 or IPv6) does not exist in the frame, replace the IP address in the Key to the selected default value.</li> <li>Instead of IPv4 destination/source address put in key {*FMKG_*DV*[0:31]}</li> <li>Instead of IPv6 destination/source address put in key {*FMKG_*DV*[0:31],*FMKG_*DV*[0:31],*FMKG_*DV*[0:31],*FMKG_*DV*[0:31]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
14–15	PTYPEDV	IP Protocol Type Default Value Selection.(In IPv6 is Next Header (NH) field) <ul style="list-style-type: none"> <li>If IP header (IPv4 or IPv6) isn't exist in the frame replace IP protocol type / next header in the Key to the selected default value.</li> <li>Instead of IPv4 PTYPE or IPv6 NH fields put in key {*FMKG_*DV*[0:7]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1

**Table 5-373. AWR3\_RFMODE\_FMKG\_SE\_EKDV Field Descriptions (continued)**

Bits	Name	Description
16–17	IPTOSDV	IP Type of Service (TOS) default Value Selection.(In IPv6 is Traffic Class(TC) field) <ul style="list-style-type: none"> <li>If IP header (IPv4 or IPv6) isn't exist in the frame replace IPv4 TOS / IPv6 TC field in the Key to the selected default value.</li> <li>Instead of IPv4 TOS or IPv6 TC fields put in key {*FMKG_*DV*[0:7]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
18–19	IP6FLDV	IPv6 Flow Label default Value Selection. <ul style="list-style-type: none"> <li>If IPv6 header isn't exist in the frame replace IPv6 FL field in the Key to the selected default value.</li> <li>Instead of IPv6 FL field put in key {4'b0000,*FMKG_*DV*[0:19]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
20–21	IPSECSPIDV	IPsec SPI default Value Selection. <ul style="list-style-type: none"> <li>If IPsec header isn't exist in the frame replace IPsec SPI field in the Key to the selected default value.</li> <li>Instead of IPsec SPI field put in key {*FMKG_*DV*[0:31]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
22–23	L4PDV	Layer 4 Port default Value Selection. <ul style="list-style-type: none"> <li>If L4 header isn't exist in the frame replace L4 port source/destination field in the Key to the selected default value.</li> <li>Instead of L4 Port number field put in key {*FMKG_*DV*[0:15]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
24–25	TFLGDV	TCP Flag default Value Selection. <ul style="list-style-type: none"> <li>If TCP header isn't exist in the frame replace TCP Flag field in the Key to the selected default value.</li> <li>Instead of TCP Flag field put in key {*FMKG_*DV*[0:7]}</li> </ul> 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use AWR_RFMODE_FMKG_SE_DV0 11 Use AWR_RFMODE_FMKG_SE_DV1
26–31	—	Reserved

#### 5.10.3.12.4 KeyGen Scheme Bit Mask Command High Register (AWR4\_RFMODE\_FMKG\_SE\_BMCH)

In the key, the user can mask 1 byte in bit resolution. This command can be used for masking the known extracted fields and the generic extracted fields. There are four mask commands. Each mask command contains three fields: Mask command select (MCS), mask offset (MO) and byte mask (BM). The MCS,

MO and BM programmability is spread on FMKG\_SE\_BMCH, FMKG\_SE\_BMCL and FMKG\_SE\_FQB registers.

Offset 0x10C

Access: Read/Write<sup>1</sup>

	0	5 6	11	12	15	16	21 22	27	28	31
R W	MCS0	MCS1	MO0	MCS2	MCS3	MO1				
Reset	All zeros									

**Figure 5-325. KeyGen Scheme Bit Mask Command High Register (AWR4\_RFMODE\_FMKG\_SE\_BMCH)**

<sup>1</sup> Indirect by access logic

**Table 5-374. AWR4\_RFMODE\_FMKG\_SE\_BMCH Field Descriptions**

Bits	Name	Description
0–5	MCS0	Mask Command Select 0 Select for which extract command to do the mask command. 0x0–0x1f The mask command mask a byte of the extract data that indicated by FMKG_SE_EKFC[0–31] 0x20–0x27 The mask command mask a byte of the extract data that indicated by FMKG_SE_GEC0–7. Other Reserved
6–11	MCS1	Mask Command Select 1 Select for which extract command to do the mask command. 0x0–0x1f The mask command mask a byte of the extract data that indicated by FMKG_SE_EKFC[0–31] 0x20–0x27 The mask command mask a byte of the extract data that indicated by FMKG_SE_GEC0–7. Other Reserved
12–15	MO0	Mask Offset 0. Offset in bytes from the start of the extracted data of the extracted command - MCS0.
16–21	MCS2	Mask Command Select 2 Select for which extract command to do the mask command. 0x0–0x1f The mask command mask a byte of the extract data that indicated by FMKG_SE_EKFC[0–31] 0x20–0x27 The mask command mask a byte of the extract data that indicated by FMKG_SE_GEC0–7. Other Reserved
22–27	MCS3	Mask Command Select 3 Select for which extract command to do the mask command. 0x0–0x1f The mask command mask a byte of the extract data that indicated by FMKG_SE_EKFC[0–31] 0x20–0x27 The mask command mask a byte of the extract data that indicated by FMKG_SE_GEC0–7. Other Reserved
28–31	MO1	Mask Offset 1. Offset in bytes from the start of the extracted data of the extracted command - MCS1.

### 5.10.3.12.5 KeyGen Scheme Bit Mask Command Low Register (AWR5\_RFMODE\_FMKG\_SE\_BMCL)

The AWR5\_RFMODE\_FMKG\_SE\_BMCL is part of the bit mask commands configuration. It is continuous to the FMKG\_SE\_BMCH register (see [Section 5.10.3.12.4, “KeyGen Scheme Bit Mask Command High Register \(AWR4\\_RFMODE\\_FMKG\\_SE\\_BMCH\)”](#)).

Offset 0x110								Access: Read/Write <sup>1</sup>
0		7	8		15	16		23 24
R W	BM0 BM1 BM2 BM3							

Reset All ones

**Figure 5-326. KeyGen Scheme Bit Mask Command Low Register (AWR5\_RFMODE\_FMKG\_SE\_BMCL)**

<sup>1</sup> Indirect by access logic

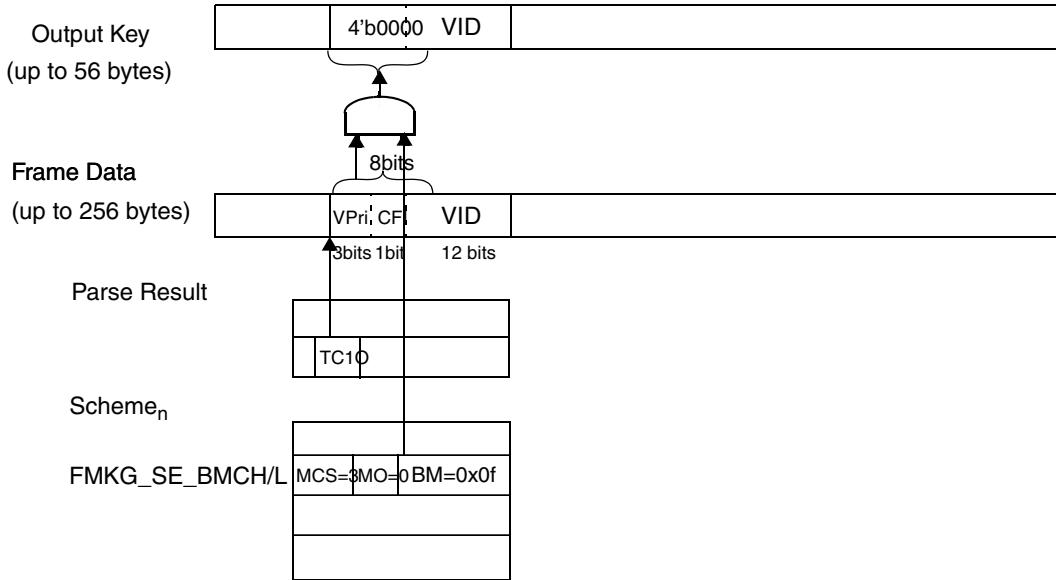
**Table 5-375. AWR5\_RFMODE\_FMKG\_SE\_BMCL Field Descriptions**

Bits	Name	Description
0–7	BM0	Bit Mask 0 Mask the unwanted bit by writing 0. For masking the selected byte according to MCS0 and MO0
8–15	BM1	Bit Mask 1 Mask the unwanted bit by writing 0. For masking the selected byte according to MCS1 and MO1
16–23	BM2	Bit Mask 2 Mask the unwanted bit by writing 0. For masking the selected byte according to MCS2 and MO2
24–31	BM3	Bit Mask 3 Mask the unwanted bit by writing 0. For masking the selected byte according to MCS3 and MO3

#### Example 5-1. Bit Mask Command Example

The user would perform the following steps if they want to extract only VLAN ID (12 bits) from the first VLAN tag (TCI). See also [Figure 5-327](#):

1. Configure FMKG\_SE\_EKFC[VLANTCI\_1] = 1 for extracting the VLAN TCI field.
2. Configure FMKG\_SE\_BMCH[MCS0] = 0x3 to mask a byte of the extract data that indicated by FMKG\_SE\_EKFC[VLANTCI\_1]
3. Configure FMKG\_SE\_BMCH[MO0] = 0x0 to point to the start of the extracted VLAN TCI field.
4. Configure FMKG\_SE\_BMCL[BM0] = 0x0f to mask (zero) the first nibble of VLAN TCI field (VLAN PRIO[0:2] and TCI bit).



**Figure 5-327. Bit Mask Command Example**

### 5.10.3.12.6 KeyGen Scheme Entry Frame Queue Base Register (AWR6\_RFMODE\_FMKG\_SE\_FQB)

The AWR6\_RFMODE\_FMKG\_SE\_FQB has two functions:

- Part of the bit mask command configuration. It is a continuation of the FMKG\_SE\_BMCH/L registers (see [Section 5.10.3.12.4, “KeyGen Scheme Bit Mask Command High Register \(AWR4\\_RFMODE\\_FMKG\\_SE\\_BMCH\)”\).](#)
- FQ base configuration (see [Section 5.10.4.5, “Frame Queue ID \(FQID\) Generation”](#)).

Offset 0x114										Access: Read/Write <sup>1</sup>
FQBASE										31
All zeros										

Reset

R  
W

MO2      MO3

**Figure 5-328. KeyGen Scheme Entry Frame Queue Base Register (AWR6\_RFMODE\_FMKG\_SE\_FQB)**

<sup>1</sup> Indirect by access logic

**Table 5-376. AWR6\_RFMODE\_FMKG\_SE\_FQB Field Descriptions**

Bits	Name	Description
0–3	MO2	Mask Offset 2 Offset in bytes from the start of the extracted data of the extracted command - MCS2. (This field belongs to the previous registers to the Bit Mask Command (FMKG_SE_BMCH/L).)
4–7	MO3	Mask Offset 3 Offset in bytes from the start of the extracted data of the extracted command - MCS3. (This field belongs to the previous registers to the Bit Mask Command (FMKG_SE_BMCH/L).)
8–31	FQB BASE	Frame Queue Base (See <a href="#">Section 5.10.4.5, “Frame Queue ID (FQID) Generation.”</a> )

### 5.10.3.12.7 KeyGen Scheme Entry Hash Configuration Register (AWR7\_RFMODE\_FMKG\_SE\_HC)

Offset 0x118

## Access: Read/Write<sup>1</sup>



Reset

All zeros

**Figure 5-329. KeyGen Scheme Entry Hash Configuration Register (AWR7\_RFMODE\_FMKG\_SE\_HC)**

<sup>1</sup> Indirect by access logic

**Table 5-377. AWR7\_RFMODE\_FMKG\_SE\_HC Field Descriptions**

Bits	Name	Description
0–1	—	Reserved
0	NO_FQI_D_GEN	<p>Do not generate FQID            0 - FQID is generated and updated in the Internal context.            1 - FQID isn't generated.</p> <p>If this bit is the FQID is not overwritten by the KeyGen. This feature is needed when there is a need to generate a hash key for the extracted headers, but the destination queue is left as programmed in the BMI for each port.</p>
1	SYM	<p>Symmetric Hash            0 - Symmetric Hash disabled            1 - Symmetric Hash Enable</p> <p>See <a href="#">Section 5.10.4.3.1, “Symmetric Hash Function.”</a></p>

**Table 5-377. AWR7\_RFMODE\_FMKG\_SE\_HC Field Descriptions (continued)**

Bits	Name	Description
2–7	HSHIFT	Hash Shift Right. Used to select a portion of the 64-bit key hash result The hash result is used according to HSHIFT configuration: 0x0 - Hash[40:63] 0x1 - Hash[39:62] 0x2 - Hash[38:61] ... 0x28 - Hash[0:23] 0x29-0x3F - Reserved
8–31	HMASK	Hash Mask. Used to mask 24 bits from the hash result according to HSHIFT

### 5.10.3.12.8 KeyGen Scheme Entry Policer Profile Command Register (AWR8\_RFMODE\_FMKG\_SE\_PPC)

The AWR8\_RFMODE\_FMKG\_SE\_PPC describes how to calculate the policer profile. See [Section 5.10.4.5, “Frame Queue ID \(FQID\) Generation.”](#)

Offset 0x11C Access: Read/Write<sup>1</sup>

	0	1	2	3	4	7	8		15	16	19	20	23	24		31
R W	PPSH	—	NO_P NUM _GEN	—	—	PPMASK	—	PPSL	—	—	—	—	PPBASE	—	—	—

Reset All zeros

**Figure 5-330. KeyGen Scheme Entry Policer Profile Command Register (AWR8\_RFMODE\_FMKG\_SE\_PPC)**

<sup>1</sup> Indirect by access logic

**Table 5-378. Register AWR8\_RFMODE\_FMKG\_SE\_PPC Bits Description**

Bits	Name	Description
0	PPSH	Policer Profile Shift High bit PPSH and PPSL define the Policer Profile Shift bits (PPS[0]=PPSH, PPS[1:4]=PPSL). The PPS is a shift right indication. The policer profile is calculated based on the Keygen Distribution Function Value (KDFV). It uses the lower 8 bits of the shifted KDFV. 0x0 - No Shift. (KDFV[16:23]) 0x1 - Shift 1 bits. (KDFV[15:22]) ... 0xc - Shift 12 bits. (KDFV[4:11]) ... 0x17 - Shift 23 bits - {7'h0,KDFV[0]} (use only the MSbit of KDFV). 0x18-0x1f -Reserved See <a href="#">Section 5.10.4.4, “Keygen Distribution Function Value (KDFV).”</a>
1–2	—	Reserved

**Table 5-378. Register AWR8\_RFMODE\_FMKG\_SE\_PPC Bits Description (continued)**

Bits	Name	Description
3	NO_PNU M_GEN	Don't generate Policer Profile 0 - PNUM is generated and updated in the internal context (IC) by the Keygen module. 1 - PNUM is not updated in the Internal context by the Keygen module
4–7	—	Reserved
8–15	PPMASK	Policer Profile Mask. Also see <a href="#">Section 5.10.4.5, “Frame Queue ID (FQID) Generation.”</a>
16–19	PPSL	Policer Profile Shift Low bits.
20–23	—	Reserved
24–31	PPBASE	Policer Profile Base. Also see <a href="#">Section 5.10.4.5, “Frame Queue ID (FQID) Generation.”</a>

#### **5.10.3.12.9 KeyGen Scheme Entry Generic Extract Command Register (AWR<9+i>\_RFMODE\_FMKG\_SE\_GEC<i>)**

The generic extract command supports two types of commands:

- Extract data to key where TYPE = 0—Extracts up to 16 bytes from the first 256 bytes of the frame to build a key.
- Extract data to FQID/Policer Profile or command where TYPE=1—Extracts 1 byte from the first 256 bytes of the frame and influence directly on the FQID/Policer profile value. The extracted byte can be masked and bit wise ORed with the calculated FQID/Policer profile.

Offset 0x120  
offset 4  
range 0 .. 7

Access: Read/Write<sup>1</sup>

	0	1	2	3	7	8	15	16	17	23	24		31
R	V	DV	SIZE		MASK			TYPE	HT		EO		
W	0	0	0	0	0	0	1	1	1	1	1	0	0
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0

**Figure 5-331. KeyGen Scheme Entry Generic Extract Command Register (AWR<9+i>\_RFMODE\_FMKG\_SE\_GEC<i>)**

<sup>1</sup> Indirect by access logic

**Table 5-379. AWR<9+i>\_RFMODE\_FMKG\_SE\_GEC<i> (0 .. 7) Field Descriptions**

Bits	Name	Description
0	V	Valid bit 0 not valid skip 1 key generation entry
1–2	DV	Default Value If field does not exist put default value to the key. The value is extracted from four user-configurable options. If SIZE is bigger than the default value, the default value is duplicated. 00 Use FMKG_GDV0R 01 Use FMKG_GDV1R 10 Use FMKG_SE_DV0 11 Use FMKG_SE_DV1 The meaning of the term ‘does not exist’ depends on the value programmed in the HT field in this register. If extract with validation is chosen in the HT field, the header “exists” if it is found in the frame by the parser (that is, the Parse result header offset is not 0xFF) and no error is found in the corresponding header. If extracted without validation being chosen in the HT field, parser errors are not checked to determine if the header exists.
3–7	SIZE	This field depends on TYPE bit. If TYPE is 1: ROTATE Right The value in this field specifies the number of bits to rotate the result right (after extraction and masking). The extracted byte is masked by MASK, right-padded with 24 zeros and then rotated right by SIZE. The rotated result is used for calculating the FQID and the Policer profile, described in <a href="#">Section 5.10.4.5, “Frame Queue ID (FQID) Generation.”</a> If TYPE is 0: SIZE Byte size of extracted field added to the key. 0 1 byte 1 2 bytes ... 15 16 bytes (maximum number of bytes that can be extracted in one command). 16-31 Not valid
8–15	MASK	Bit Mask. Mask the last extracted byte by writing 0.
16	TYPE	Generic Extract Command Type 0 Hash extract command. The extracted field is concatenated to the Key that was extracted so far. A hash function is applied to the whole key and a hash key is generated as a result. 1 Or extract command. The extracted field is ORed with the hash key (the output of the hash function).

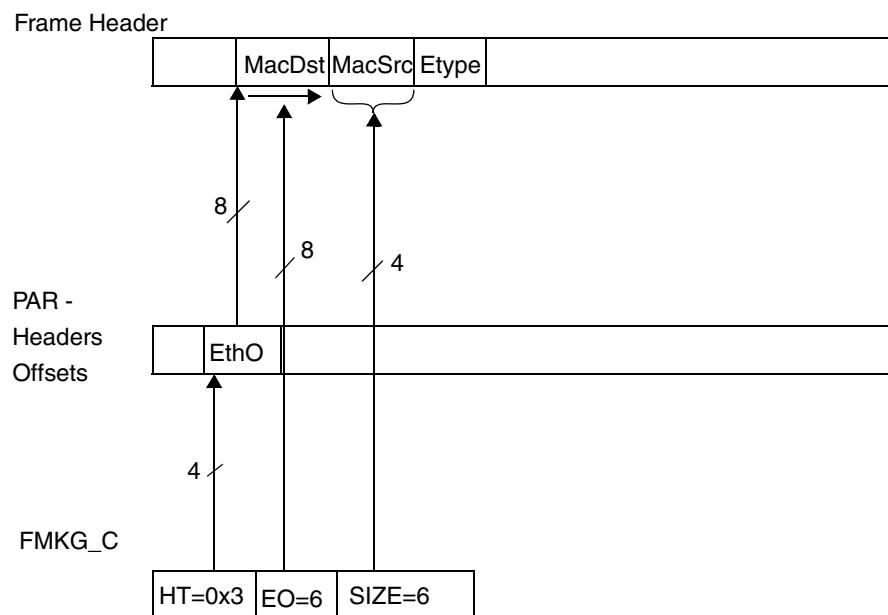
**Table 5-379. AWR<9+i>\_RFMODE\_FMKG\_SE\_GEC<i> (0 ..7) Field Descriptions (continued)**

Bits	Name	Description
17–23	HT	<p>Header Type The last 4 bits is the index of the header offsets in the Parse result (PR) header offset vector (second 16 bytes starting at address 0x10 of the parse result. For example, HT[20:23] = 0x8 the offset to the PPPoE header is used for extraction.</p> <p>The first 3 bits are used for different types of header or for different extract functionality:</p> <ul style="list-style-type: none"><li>• Extract from default values, parse result, AD[FQID], start of frame (see details below).</li><li>• Extract with validation: The KeyGen checks if the extracted header exists in the packet (that is, the Parse result header offset is not 0xFF) and that it does not contain errors (that is, it is based on the PR[LxR, x = 2, 3, 4])</li><li>• Extract without validation: The KeyGen checks if the extracted header exists in the packet (that is, the Parse result header offset is not 0xFF).</li></ul> <p>To extract from the last header's payload, also configure FMKG_FDOR (see <a href="#">Section 5.10.3.8, “KeyGen Frame Data Offset Register (FMKG_FDOR)”</a>).</p> <p>7'h70 - Shim1 7'h10 - Extract from one of four default values according to the DV field in this register. 7'h20 - Extract additional fields, using the EO[2:7] to select the field:     EO[2:7]=0x00-0x0F: Parse Result first 1-16 bytes     In FMan_v3 the following fields may also be extracted:     EO[2:7]=0x10-0x1F: Parse Result Second 16 bytes,     EO[2:7]=0x20-0x22: The FQID in the IC[AD] - 3 bytes,     EO[2:7]=0x23-0x3F: Reserved 7'h30 - Extract additional fields, using the EO[2:7] to select the field:     EO[2:7]=0x00-0x0F: Parse Result second 16 bytes 7'h40 - Extract bytes from the start of the frame. 7'h71 - Shim2 7'h72 - IP PID w/o validation 7'h03 - Ethernet with validation 7'h73 - Other - Ethernet w/o validation 7'h04 - SNAP with validation 7'h74 - Other - SNAP w/o validation 7'h05 - Vlan TCI 1 with validation 7'h75 - Other - VlanTCI 1 w/o validation 7'h06 - Vlan TCI last (VLANTCI_N) with validation 7'h76 - Other - Vlan TCI last (VLANTCI_N) w/o validation 7'h07 - Etype with validation 7'h77 - Other - Etype w/o validation 7'h08 - PPPoE with validation 7'h78 - Other - PPPoE w/o validation 7'h09 - MPLS1 with validation 7'h79 - Other - MPLS1 w/o validation 7'h19 - MPLS2 - based on (MPLS1O+4bytes) with validation that MPLS2 exist (difference between MPLS2O and MPLS1O) 7'h29 - MPLS3 - based on (MPLS1O+8bytes) with validation that MPLS3 exists (difference between MPLS2O and MPLS1O)</p>

**Table 5-379. AWR<9+i>\_RFMODE\_FMKG\_SE\_GEC<i> (0 ..7) Field Descriptions (continued)**

Bits	Name	Description
17–23	HT	7'h0a - Last MPLS with validation 7'h7a - Other - MPLS last w/o validation 7'h0b - IPv4 with validation 7'h1b - IPv6 with validation 7'h7b - Other w/o validation 7'h0c - Second IPv4 with validation 7'h1c - Second IPv6 with validation 7'h2c - Min Encapsulation with validation 7'h7c - Other - Using IP2O w/o validation 7'h0d - GRE with validation 7'h7d - Other - Using GREO w/o validation 7'h0e - TCP with validation 7'h1e - UDP with validation 7'h2e - IPSec AH w/ validation 7'h3e - SCTP with validation 7'h4e - DCCP with validation 7'h6e - IPSec ESP with validation 7'h7e - Other - Using L4O w/o validation 7'h7f - Other - Using NXTHDRO w/o validation
24–31	EO	Extract Offset Offset relative to the parser header type (HT) offset

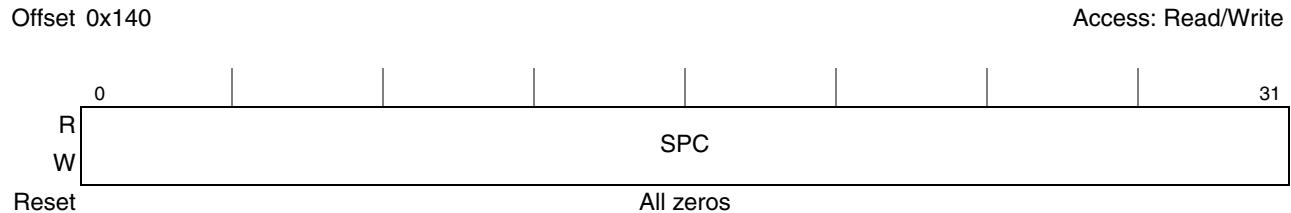
Figure 5-332 describes the key generation command.



**Figure 5-332. Key Generation Command Description**

### 5.10.3.12.10 KeyGen Scheme Entry Statistic Packet Counter Register (AWR17\_RFMODE\_FMKG\_SE\_SPC)

The KeyGen statistic counters hold a counter per scheme that calculates the number of accesses to a specific scheme.



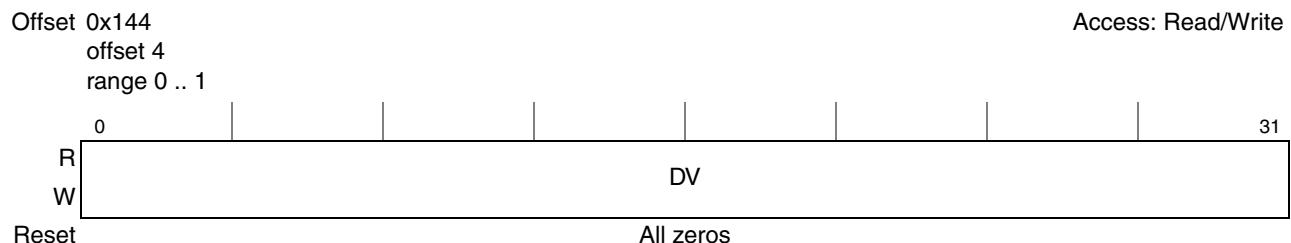
**Figure 5-333. KeyGen Scheme Entry Statistic Packet Counter Register (AWR17\_RFMODE\_FMKG\_SE\_SPC)**

**Table 5-380. AWR17\_RFMODE\_FMKG\_SE\_SPC Field Descriptions**

Bits	Name	Description
0–31	SPC	Scheme Packet Counter Counts the number of access for this scheme.

### 5.10.3.12.11 KeyGen Scheme Entry Default Value Register (AWR<18+i>\_RFMODE\_FMKG\_SE\_DV<i>)

The KeyGen default value entries are used when an extracted field does not exist in the frame. When the header does not exist in the processed frame and there is an extract command for this field a default value is used. These entries are two of four optional DVs, the others are global. See [Section 5.10.3.9, “KeyGen Global Default Value Register \(FMKG\\_GDV<i>R\).”](#)



**Figure 5-334. KeyGen Scheme Entry Default Value Register (AWR<18+i>\_RFMODE\_FMKG\_SE\_DV<i>)**

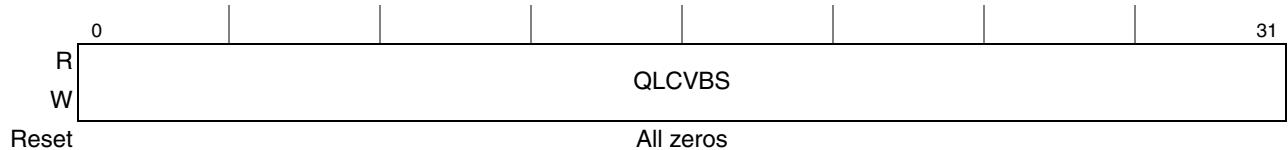
**Table 5-381. AWR<18+i>\_RFMODE\_FMKG\_SE\_DV<i> Field Descriptions**

Bits	Name	Description
0–31	DV	Default Value.

### 5.10.3.12.12 KeyGen Scheme Entry Custom Classifier Bit Select Register (AWR20\_RFMODE\_FMKG\_SE\_CCBS)

Offset 0x14c

## Access: Read/Write



**Figure 5-335. KeyGen Scheme Entry Custom Classifier Bit Select Register (AWR20 RFMODE FMKG SE CCBS)**

**Table 5-382. AWR20 RFMODE FMKG SE CCBS Field Descriptions**

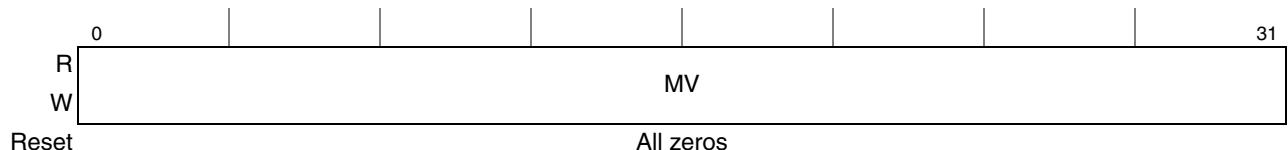
Bits	Name	Description
0–31	QLCVBS	<p>Qualified LCV Bit Select.</p> <p>Selects which of the bits from the qualified LCV (QLCV) will be used to define the custom classifier offset (CCO[0:3]).</p> <p>The user can configure up to 4 bits to be selected.</p> <ul style="list-style-type: none"> <li>• Bit 0 ==1 means that bit 0 of the QLCV will be selected.</li> <li>• ...</li> <li>• Bit 31 = 1 means that bit 31 of the QLCV will be selected.</li> </ul> <p>Among the selected bits only the 4 least significant bits are used according to their order in the QLCVBS, meaning:</p> <ul style="list-style-type: none"> <li>• The first least significant bit that is asserted will influence CCO[3] bit.</li> <li>• The second least significant bit that is asserted will influence CCO[2] bit.</li> <li>• The third least significant bit that is asserted will influence CCO[1] bit.</li> <li>• The firth least significant bit that is asserted will influence CCO[0] bit.</li> </ul>

### **5.10.3.12.13 KeyGen Scheme Entry Match Vector Register (AWR21\_RFMODE\_FMKG\_SE\_MV)**

In an indirect scheme selection the lineup confirmation vector is masked with the selected classification plan and then it is matched with the match vectors configured for this port, where each match vector represents a scheme. The first match that is found points to the selected scheme. The match vector is one per scheme and is part of the scheme configuration. Figure 5-336 shows the AWR21\_RFMODE\_FMKG\_SE\_MV.

Offset 0x150

## Access: Read/Write



**Figure 5-336. KeyGen Scheme Entry Match Vector Register (AWR21\_RFMODE\_FMGK\_SE\_MV)**

**Table 5-383. AWR21\_RFMODE\_FMKG\_SE\_MV Field Descriptions**

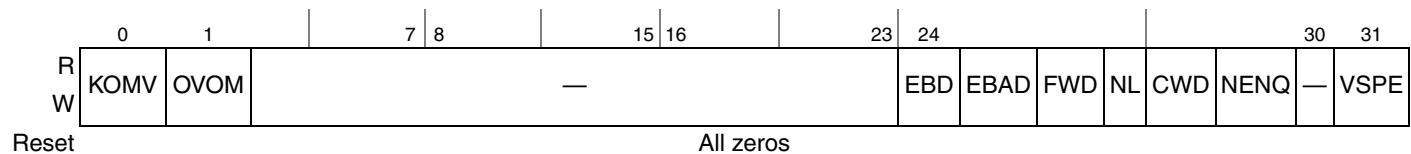
Bits	Name	Description
0-31	MV	Match Vector

#### 5.10.3.12.14 KeyGen Scheme Entry Operational Mode bits Register (AWR21\_RFMODE\_FMKG\_SE\_OM)

This entry contains the operational mode bits that affect the operation of the BMI. Some of these bits (for example, NL and NENQ) affect the pipeline flow, while others affect the access to buffers (for example, EBD, EBAD, FWD, CWD). [Section 5.5.6.1, “Operational Mode bits,”](#) describes the functionality of these bits.

Offset 0x154

Access: Read/Write



**Figure 5-337. KeyGen Scheme Entry Match Vector Register (AWR21\_RFMODE\_FMKG\_SE\_OM)**

**Table 5-384. AWR21\_RFMODE\_FMKG\_SE\_OM Field Descriptions**

Bits	Name	Description
0	KOMV	KeyGen Operational Mode bits Valid 0 - Operational mode bits (bits 24-30) are not valid. 1 - Operational mode bits (bits 24-30) are written in the frame Internal Context Action Descriptor (ICAD[A2]) (ICAD bits 32-38) overriding any default value. <b>Note:</b> VSPE bit is not affected by KOMV. See <a href="#">Section 5.4.3.3, “Internal Context Action Descriptor (ICAD).”</a>
1	OVOM	Override Operational Mode bits. 0 - ‘Setting one’ mode: Do not override operational mode bits in Internal Context Action Descriptor (ICAD). Set to ‘one’ bits which are set in this register (bits 24-30) 1 - ‘Override mode’: Override operation mode bits in ICAD with values programmed in this register (bits 24-30). See <a href="#">Section 5.5.6.1, “Operational Mode bits.”</a>
2-23	—	Reserved
24	EBD	Bit description are the same as <a href="#">5.4.3.1.1, “Frame Queue Descriptor (FQD) Context A, “Context A—A2 Field Description for Offline Port”.</a>
25	EBAD	
26	FWD	
27	NL	
28	CWD	
29	NENQ	

**Table 5-384. AWR21\_RFMODE\_FMKG\_SE\_OM Field Descriptions**

Bits	Name	Description
30	—	Reserved
31	VSPE	Bit description are the same as <a href="#">5.4.3.1.1, “Frame Queue Descriptor (FQD) Context A, “Context A—A2 Field Description for Offline Port”.</a>

### 5.10.3.12.15 KeyGen Scheme Entry Virtual Storage Profile Register (AWR8\_RFMODE\_FMKG\_SE\_VSP)

The AWR8\_RFMODE\_FMKG\_SE\_VSP describes how to calculate the virtual storage profile. Also see [Section 5.10.4.7, “KeyGen Storage Profile ID Generation.”](#)

Offset 0x158 Access: Read/Write<sup>1</sup>

AWR8_RFMODE_FMKG_SE_VSP Register Map									
Field Descriptions									
Bit 31:0									
Access: Read/Write <sup>1</sup>									
Field Descriptions									
NO_KSPEN (R/W)									
KSPS (R/W)									
KSPMASK (R/W)									
KSPID (R/W)									
Reset									
All zeros									

**Figure 5-338. KeyGen Scheme Entry Virtual Storage Profile Command Register (AWR8\_RFMODE\_FMKG\_SE\_VSP)**

<sup>1</sup> Indirect by access logic

**Table 5-385. Register AWR8\_RFMODE\_FMKG\_SE\_VSP Bits Description**

Bits	Name	Description
0	NO_KSPEN	No Keygen virtual Storage Profile Enable 0 - Keygen generates a storage profile ID, and writes in the frame Internal Context Action Descriptor (ICAD[B0], ICAD bits 0-7) overriding any previous value. In addition, the value programmed in FMKG_SE_OM[VSPE] is written in ICAD bit 39. 1 - Keygen does not change the value of the storage profile in the Internal Context Action Descriptor (ICAD). FMKG_SE_OM[VSPE] is (ICAD bit 39) is not changed. <b>Note:</b> The KeyGen generates a 6 bit RSPID (relative SPID), thus ICAD bits 0,1 are always set to zero by KeyGen. See <a href="#">5.5.4.1.1, “Storage Profile Virtualization per port (for Rx and Offline ports)”</a> for more details. Also see <a href="#">5.10.4.7, “KeyGen Storage Profile ID Generation.”</a>
2–6	—	Reserved
3–7	KSPS	Keygen virtual Storage Profile Shift The VSPS is a shift right indication. The Virtual Storage profile is calculated based on the Keygen Distribution Function Value (KDFV). It uses the lower six bits of the shifted KDFV. The most significant bits are padded with zeroes. 0x00 - No Shift. (KDFV[18:23]) 0x01 - Shift right 1 bit. (KDFV[17:22]) ... 0x0C - Shift right 12 bits. (KDFV[6:11]) ... 0x17 - Shift right 23 bits - {5'h0,KDFV[0]} (use only the MSbit of KDFV). 0x18-0x1F-Reserved See <a href="#">Section 5.10.4.4, “Keygen Distribution Function Value (KDFV).”</a>
8–13	—	Reserved

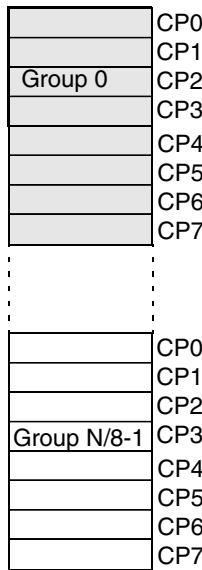
**Table 5-385. Register AWR8\_RFMODE\_FMKG\_SE\_VSP Bits Description (continued)**

Bits	Name	Description
14–19	KSPMASK	Keygen virtual Storage Profile Mask. This field is ANDed with the shifted (after the operation of VSPS) KDFV.
20–25	—	Reserved
26–31	KSPID	Keygen virtual Storage Profile ID. See <a href="#">Section 5.10.4.7, “KeyGen Storage Profile ID Generation”</a> for details.

### 5.10.3.13 KeyGen Classification Plan Entry Memory Map

The classification plan table has N classification plans (CPs). Refer to device specific information for exact number of CPs in each SoC. Each CP entry consumes 4 bytes. The classification plan table is divided into groups of eight CPs each. The user can map each CP group to each hardware portID. The application access to the classification plan table is accomplished by indirect mapping, using the atomic access logic. For details on atomic KeyGen indirect memory entry access, see [Section 5.10.3.11, “KeyGen Action Register \(FMKG\\_AR\).”](#) Figure 5-339 shows classification plan tables.

**Classification Plan Tables**

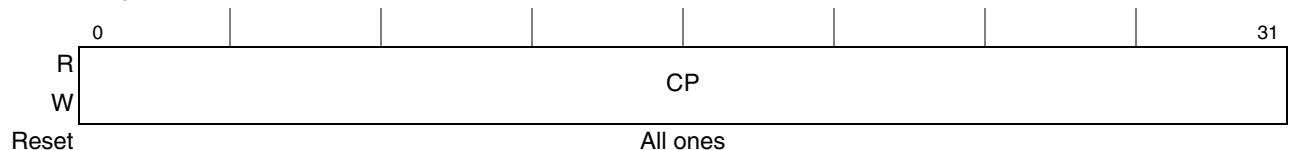


**Figure 5-339. Classification Plan Tables**

### **5.10.3.13.1 KeyGen Classification Plan Entry Register (AWR<1+i>\_RFMODE\_FMKG\_CPE<i>)**

Offset 0x100  
offset 4  
range 0 .. 7

## Access: Read/Write



**Figure 5-340. KeyGen Classification Plan Entry Register (AWR<1+i>\_RFMODE\_FMKG\_CPE<i>)**

**Table 5-386. AWR<1+i>\_RFMODE\_FMKG\_CPE<i> Field Descriptions**

Bits	Name	Description
0–31	CP	Classification Plan. A 32-bit vector that mask the lineup confirmation vector (LCV) corresponding bits by writing 0.

#### **5.10.3.14 KeyGen Port Partition Configuration**

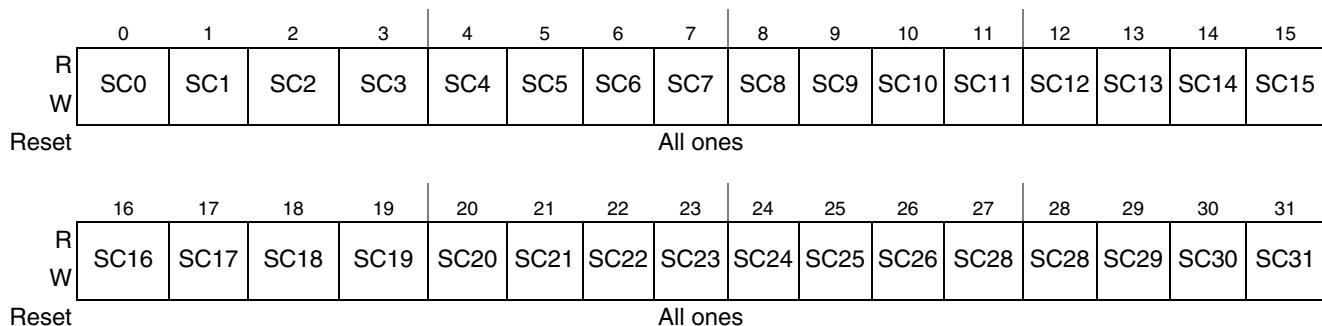
A partition can be configured for each port scheme and classification plan group. The user can map each CP group and each scheme to each partition. The application access to the memory is provided via indirect mapping, using the atomic access logic. For details on atomic KeyGen indirect memory entry access, see Section 5.10.3.11, “KeyGen Action Register (FMKG\_AR).”

### **5.10.3.14.1 KeyGen Port Entry Scheme Partition Register (AWR1 RFMODE FMKG PE SP)**

There is one register for each physical Rx and O/H port. The KeyGen scheme partition entry is used to configure the scheme partition per port. The port scheme partition configuration is indirect using FMKG AR.

Offset 0x100

## Access: Read/Write



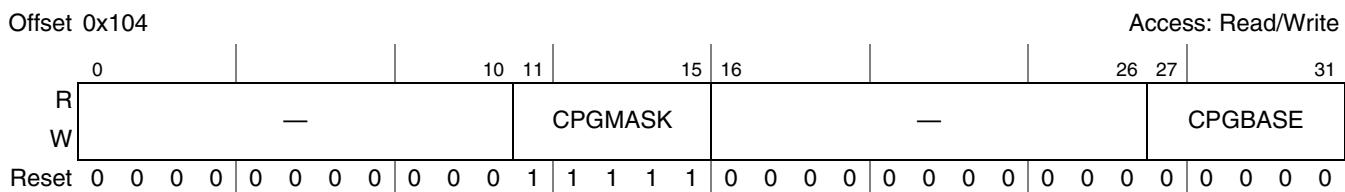
**Figure 5-341. KeyGen Port Entry Scheme Partition Register (AWR1\_RFMODE\_FMKG\_PE\_SP)**

**Table 5-387. AWR1\_RFMODE\_FMKG\_PE\_SP Field Descriptions**

Bits	Name	Description
0–31	SC $n$	Scheme $n$ belong to the partition of the selected port in FMKG_AR[PORTID].

#### **5.10.3.14.2 KeyGen Port Entry Classification Plan Partition Register (AWR2\_RFMODE\_FMKG\_PE\_CPP)**

There is one register for each physical Rx and O/H port. The KeyGen Classification Plan group partition entry is used to configure the partitioning per port. The user can configure per port the classification plan groups that belong to the port. The port partition configuration is indirect using FMKG\_AR.



**Figure 5-342. KeyGen Port Entry Classification Plan Partition Register (AWR2\_RFMODE\_FMKG\_PE\_CPP)**

**Table 5-388. AWR2\_RFMODE\_FMKG\_PE\_CPP Field Descriptions**

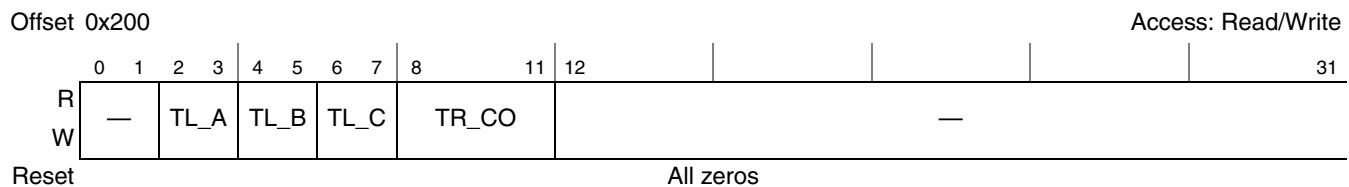
Bits	Name	Description
0–10	—	Reserved
11–15	CPGMASK	<p>Classification Plan Group Mask</p> <p>Mask the Classification Plan ID taken from the Parse Result (PR[CPLANID]).</p> <p>Define the number of classification plan groups per port.</p> <p>0 - mask the CPLANID value.</p> <p>1 - pass the CPLANID value.</p> <p>The CPGMASK masks the 5 MSBs.</p> <p>5'h00000 - Means that 1 CPlan group is used per port.</p> <p>5'h00001 - Means that 2 CPlan groups are used per port.</p> <p>5'h00010 - Means that 2 CPlan groups are used per port, but they are not contiguous.</p> <p>5'h00011 - Means that 4 CPlan groups are used per port.</p> <p>...</p> <p>5'h11111 - Means that All 32 CPlan groups can be used per port.</p>
16–26	—	Reserved
27–31	CPGBASE	<p>Classification Plan Group Base</p> <p>The classification Plan table is divided to 32 groups. Each port could look at the classification Plan table as only it exists in the system - meaning start from 0. As such, each port can have its own virtual classification plan table. The base resolution is in 8 plan IDs.</p> <p>The selection of the classification plan group for given hardware portID is:</p> <p>CPlan group number = (CPLANID[0:4] &amp; CPGMASK)   CPGBASE</p> <p>The Final CPlan ID is CPlan group number[0:4] concatenated with the CPLANID[5:7].</p>

### 5.10.3.15 KeyGen Debug Registers

The KeyGen debug registers provide global control over the debug and per-flow trap registers. There are three debug flows: A, B, and C. Each flow has four trap registers that act as a set. Each set has a configuration, value, and a mask.

#### 5.10.3.15.1 KeyGen Debug Control Register (FMKG\_DCR)

The FMKG\_DCR configures the per-flow trace level and the linkage between the trap registers of the debug flows.



**Figure 5-343. KeyGen Debug Control Register (FMKG\_DCR)**

**Table 5-389. FMKG\_DCR Field Descriptions**

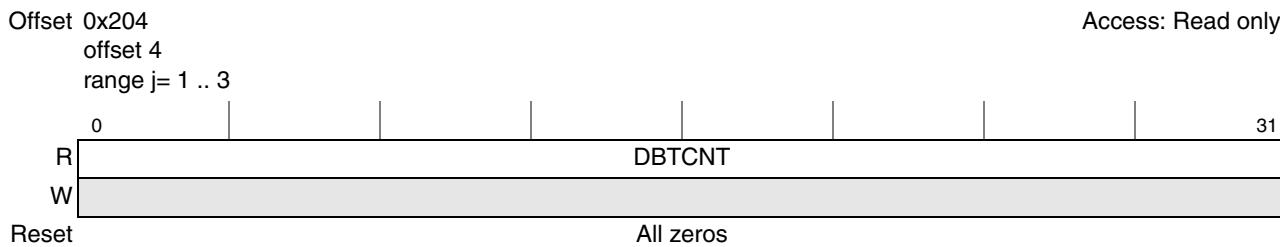
Bits	Name	Description
0–1	—	Reserved
2–3	TL_A	Trace level for debug flow A Selects the trace information that will be dumped to the debug area of a frame's internal context (IC). 00 Trace disable 01 Minimum trace 10 Verbose trace 11 Very verbose trace
4–5	TL_B	Trace level for debug flow B Selects the trace information that will be dumped to the debug area of a frame's internal context (IC). 00 Trace disable 01 Minimum trace 10 Verbose trace 11 Very verbose trace
6–7	TL_C	Trace level for debug flow C Selects the trace information that will be dumped to the debug area of a frame's internal context (IC). 00 Trace disable 01 Minimum trace 10 Verbose trace 11 Very verbose trace

**Table 5-389. FMKG\_DCR Field Descriptions (continued)**

Bits	Name	Description
8–11	TR_CO	<p>Trap collaboration</p> <p>To provide a larger number of traps for debug flow A, the traps that belong to debug flows B and/or C can be chained as a continuation of the traps for debug flow A. This lowers the number of active debug flows, but allows the criteria for debug flow A to be more extensive. The chained trap set would act as one trap entity with a single boolean result. The debug flows (B and/or C) that donated their traps to debug flow A are put into bypass since they would no longer have the resources to define match criteria. If debug flows B or C are not participants in the collaboration chain, then their trapping behavior will remain active and operate independently as configured.</p> <ul style="list-style-type: none"> <li>00 No collaboration between trap registers</li> <li>01 Debug flow A's traps are chained to debug flow B's traps. Debug flow B is in bypass.</li> <li>10 Debug flow A's traps are chained to debug flow C's traps. Debug flow C is in bypass</li> <li>11 Debug flow A's traps are chained to debug flow B's traps, which are then chained to debug flow C's traps. Debug flows B and C are in bypass.</li> </ul>
12–31	—	Reserved

### 5.10.3.15.2 KeyGen Debug Flow <j> Trap Counter Register (FMKG\_D<j>TC)

The FMKG\_D<j>TC counts the packets that match the debug trap criteria.



**Figure 5-344. KeyGen Debug Flow <j> Trap Counter Register (FMKG\_D<j>TC)**

**Table 5-390. FMKG\_D<j>TC Field Descriptions**

Bits	Name	Description
0–31	DBTCNT	Debug Trap event Counter. This is the total count of packets that match the debug trap criteria defined by FMKG_DTCRn, FMKG_D<j>T<k>VR, FMKG_D<j>T<k>MR registers. Writes to this counter are ignored.

### 5.10.3.15.3 KeyGen Debug Flow <j> Trap <k> Configuration Register (FMKG\_D<j>T<k>CR)

The FMKG\_D<j>T<k>CR defines the event type for the trigger, how it is going to be compared to the reference value and the logic operation which combines this event trap result with the next one if it exists.

## **NOTE**

FMKG\_D<j>T<k>CR, FMKG\_D<j>T<k>VR, FMKG\_D<j>T<k>MR are grouped together in subsequent addresses as follows:

0x210 FMKG\_D1T1CR  
0x214 FMKG\_D1T1VR  
0x218 FMKG\_D1T1MR

	0x220	FMKG_D1T2CR	
	...		
	0x260	FMKG_D2T1CR	
	0x264	FMKG_D2T1VR	
	0x268	FMKG_D2T1MR	
	...		
	0x2B0	FMKG_D3T1CR	
	...		
Offset	0x210		Access: Read/Write
	offset 16		
	range j= 1 .. 3, k= 1 .. 5		
R	0 1 2 3   4	9 10   15 16	25 26   31
W	CMPOP AND   —	FSEL	—   SHIFT
Reset		All zeros	

**Figure 5-345. KeyGen Debug Flow <j> Trap <k> Configuration Register (FMKG\_D<j>T<k>CR)**

**Table 5-391. FMKG\_D<j>T<k>CR Field Descriptions**

Bits	Name	Description
0–2	CMPOP	<p>Compare Operator</p> <ul style="list-style-type: none"> <li>000 Trap disabled (never matches)</li> <li>001 Always match</li> <li>010 Match if (selected field AND FMKG_D&lt;j&gt;T&lt;k&gt;MR[MASK]) equals FMKG_D&lt;j&gt;T&lt;k&gt;VR[VAL]</li> <li>011 Match if (selected field AND FMKG_D&lt;j&gt;T&lt;k&gt;MR[MASK]) does not equal FMKG_D&lt;j&gt;T&lt;k&gt;VR[VAL]</li> <li>100 Match if (selected field AND FMKG_D&lt;j&gt;T&lt;k&gt;MR[MASK]) is greater than FMKG_D&lt;j&gt;T&lt;k&gt;VR[VAL]</li> <li>101 Match if (selected field AND FMKG_D&lt;j&gt;T&lt;k&gt;MR[MASK]) is less or equal to FMKG_D&lt;j&gt;T&lt;k&gt;VR[VAL]</li> <li>110 Match if (selected field AND FMKG_D&lt;j&gt;T&lt;k&gt;MR[MASK]) is less than FMKG_D&lt;j&gt;T&lt;k&gt;VR[VAL]</li> <li>111 Match if (selected field AND FMKG_D&lt;j&gt;T&lt;k&gt;MR[MASK]) is greater or equal to FMKG_D&lt;j&gt;T&lt;k&gt;VR[VAL]</li> </ul>
3	AND	<p>AND.</p> <p>This field combines trap results into a trap equation. By default, it is an OR. If all next traps are disabled (or this is the last trap), they have no effect on the combined match result. The combined match evaluation is performed in ascending order from trap number <math>n</math> to trap number <math>n + 1</math>, and without precedence of AND over OR. For example: TRAP1 or TRAP2 and TRAP3 will result in equation equivalent to (TRAP1    TRAP2) &amp;&amp; TRAP3.</p>
4–9	—	<p>0 OR between this trap result and the next trap result.</p> <p>1 AND between this trap result and the next trap result.</p>
—	Reserved	

**Table 5-391. FMKG\_D<j>T<k>CR Field Descriptions (continued)**

Bits	Name	Description
10–15	FSEL	<p>Field selection.</p> <p>Select for which extract command to do the debug trap.</p> <p>The Match will be to the extracted data of the selected command.</p> <p>The extracted data size is up to 16 bytes there for the SHIFT field would be use to select which bytes to match.</p> <p>The debug trap is on the extract data that indicated by FMKG_SE_EKFC[0-31].</p> <p>0x0–0x1f - This value selects one of the commands according to the bits in FMKG_SE_EKFC[0-31].</p> <p>0x20–0x27 - The debug trap is on the extract data that indicated by FMKG_SE_GEC0-7.</p> <p>0x28–0x2f - Reserved.</p> <p>0x30 - {3'h0, Scheme Number[0:4], 2'h0, Key Size[0:5], FD[EOF], FD[NSS], 6'h0, PNUM[0:7]}. Where FD[NSS] is asserted scheme number and policer profile are non valid.</p> <p>0x31 - {8'h0, FQID[0:23]}</p> <p>0x32 - {8'h0, Or value [0:23]}</p> <p>0x33 - Trap Hash (look at SHIFT field for selecting Hash bits)</p> <p>0x34 - {2'h0, portid[0:5], KeyGen output NIA [0:23]}</p> <p>0x35 - key[0:447]</p>
16–25	—	Reserved
26–31	SHIFT	<p>Extracted data SHIFT test.[</p> <p>When the FSEL selects an extract command (0x0-0x26) then:</p> <p>Extract command can extract up to 16 bytes.</p> <p>This field defines which 4 bytes are trapped from the 16 extracted bytes.</p> <ul style="list-style-type: none"> <li>0 The trap compares against the 4 MSBs of Extract data [0:31].</li> <li>1 The trap compares against the Extract data [8:39]</li> <li>2 The trap compares against the Extract data [16:47]</li> <li>...</li> <li>15 The trap compares against the {Extract data [120:127],24'h0}</li> </ul> <p>When the FSEL selects Key (0x35) than:</p> <ul style="list-style-type: none"> <li>0 The trap compares against the 4 MSBs of the key[0:31].</li> <li>1 The trap compares against the key[8:39]</li> <li>2 The trap compares against the Key[16:47]</li> <li>...</li> <li>55 The trap compares against the {key[440:447],24'h0}</li> </ul> <p>When the FSEL selects trap Hash (0x33), then:</p> <ul style="list-style-type: none"> <li>0 The trap compares against the Hash[0:31]</li> <li>1 The trap compares against the Hash[32:63]</li> </ul> <p>Other values Reserved</p>

#### 5.10.3.15.4 KeyGen Debug Flow <j> Trap <k> Value Register (FMKG\_D<j>T<k>VR)

The FMKG\_D<j>T<k>VR contain the value against which the selected field at the FMKG\_D<j>T<k>CR is compared.

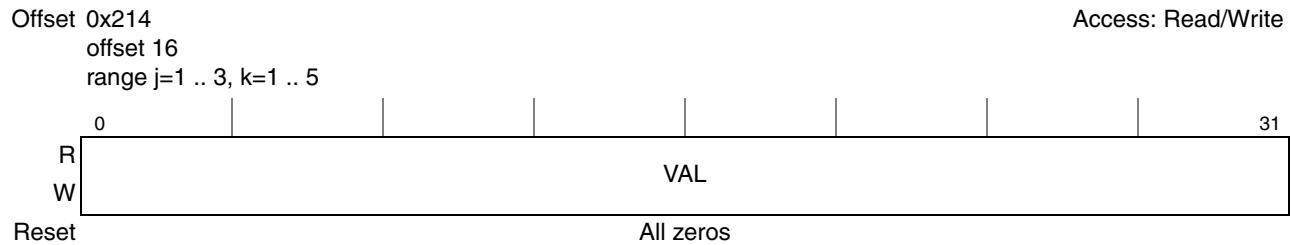


Figure 5-346. KeyGen Debug Flow <j> Trap <k> Value Register (FMKG\_D<j>T<k>VR)

Table 5-392. FMKG\_D<j>T<k>VR Field Descriptions

Bits	Name	Description
0–31	VAL	<p>Value.</p> <p>This value compares against the selected field specified in the FMKG_D&lt;j&gt;T&lt;k&gt;CR with the mask specified at FMKG_D&lt;j&gt;T&lt;k&gt;MR applied to it.</p>

#### 5.10.3.15.5 KeyGen Debug Flow <j> Trap <k> Mask Register (FMKG\_D<j>T<k>MR)

The mask value specified in the FMKG\_D<j>T<k>MR is ANDed bit-wise with the selected value for comparison value and the result is compared to the FMKG\_D<j>T<k>VR[VAL].

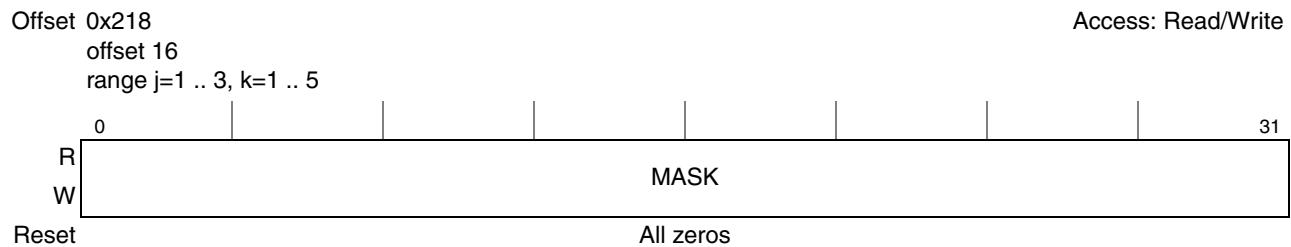


Figure 5-347. KeyGen Debug Flow <j> Trap <k> Mask Register (FMKG\_D<j>T<k>MR)

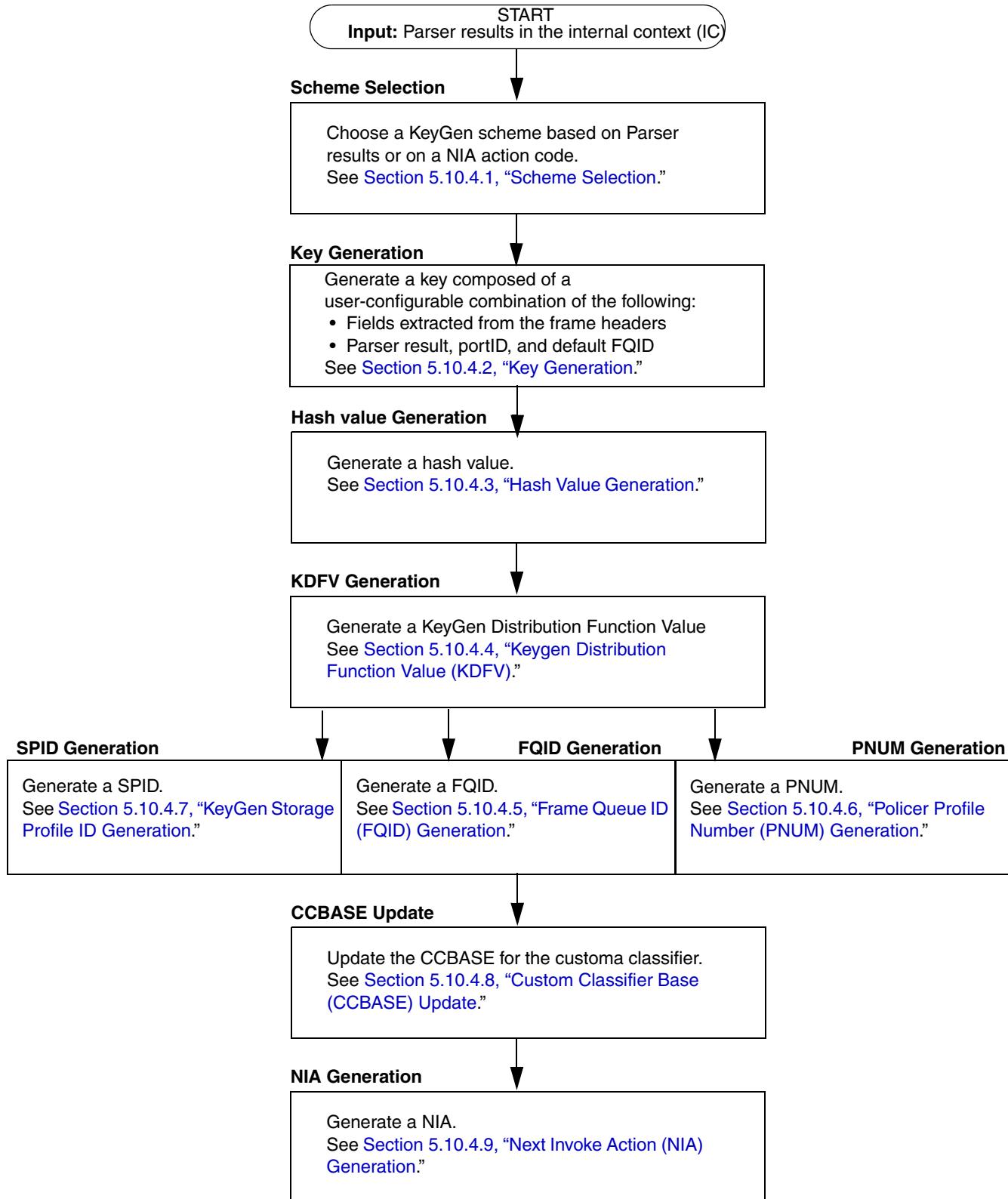
Table 5-393. FMKG\_D<j>T<k>MR Field Descriptions

Bits	Name	Description
0–31	MASK	<p>MASK.</p> <p>This field serves as the mask that is applied with a bit-wise AND to the incoming data identified by FMKG_D&lt;j&gt;T&lt;k&gt;CR[FSEL], and the result is compared to the value in FMKG_D&lt;j&gt;T&lt;k&gt;VR[VAL].</p>

### 5.10.4 KeyGen Functional Description

The KeyGen starts the classification process by distributing the received frame-to-frame queues and generating a Policer profile (PNUM). The KeyGen reads frame data and parse results and then determines what kind of scheme needs to be executed. It generates an appropriate key for the requested scheme and

calculates the FQID and the PNUM according to a hash function, base addition, and mask with extracted bits. See [Figure 5-348](#) for a depiction of the KeyGen functional flow.



**Figure 5-348. KeyGen Functional Flow**

## 5.10.4.1 Scheme Selection

The KeyGen holds one centralized scheme table in the FMan that is shared among all Rx and O/H physical ports. Each scheme has a number associated with it that identifies the scheme in ascending order. During initialization, software identifies the scheme being initialized with FMKG\_AR[NUM] (see [Section 5.10.3.11, “KeyGen Action Register \(FMKG\\_AR\)”](#)). The user configures how the key, FQID, and Policer profile number (PNUM) are generated. It is possible to allocate separate schemes for each physical port (see [Section 5.10.3.14.1, “KeyGen Port Entry Scheme Partition Register \(AWR1\\_RFMODE\\_FMKG\\_PE\\_SP\)”](#)).

The scheme selection can be accomplished using one of the following methods:

- Direct Scheme Selection
- Indirect Scheme Selection

### 5.10.4.1.1 Direct Scheme Selection

The KeyGen gets the scheme number from the NIA field (see [Section 5.10.4.10.1, “KeyGen NIA Action Codes”](#)).

This option can be used, for example, by the FMan Controller after custom classifier if the scheme programmed is not allocated to the physical port the NIA is taken from. See [Section 5.10.3.1, “KeyGen General Configuration Register \(FMKG\\_GCR\)”](#).

### 5.10.4.1.2 Indirect Scheme Selection

The user chooses different classification schemes for different protocol stacks, and a scheme is selected based on the existence of a specific protocol stack in the frame.

The scheme selection is dependent on the parse result. The Parser generates a 32-bit line-up confirmation vector (LCV), which is masked with the classification plan pointed by the classification plan ID (CPID). These values are located in the internal context (IC). The LCV and the CPID are part of the parse result and are outputs from the parser (see [Section 5.9.4.10, “Parse Result”](#)).

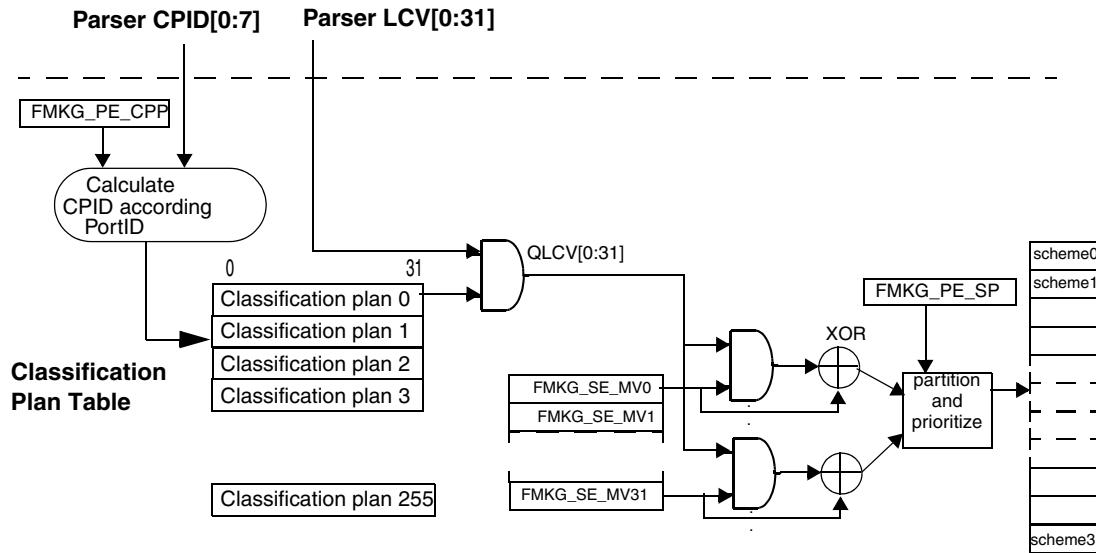
The CPID is an 8-bit index that points to the classification plan table. Each entry in the classification plan table consists of a 32-bit vector. See [Section 5.10.3.13.1, “KeyGen Classification Plan Entry Register \(AWR<1+i>\\_RFMODE\\_FMKG\\_CPE<i>,”](#) for a description of the table entry.

The classification plan table is split into groups of eight classification plans. It is possible to allocate the groups to different physical ports. The actual index to the classification plan table is computed by performing a logical OR between the CPID (from the Parser) and the value programmed for each port in FMKG\_PE\_CPP (see [Section 5.10.3.14.2, “KeyGen Port Entry Classification Plan Partition Register \(AWR2\\_RFMODE\\_FMKG\\_PE\\_CPP\)”](#)). If no partitioning is required, the FMG\_PE\_CPP should be reset.

The CPID vector is used to mask the line-up confirmation vector (LCV) of the received frame. See [Section 5.9, “Frame Manager—Parser,”](#) for more information. The masked LCV is called a qualified LCV (QLCV).

Each scheme has a match vector associated with it (see [Section 5.10.3.12.13, “KeyGen Scheme Entry Match Vector Register \(AWR21\\_RFMODE\\_FMKG\\_SE\\_MV\)”](#)). The QLCV is compared with all the

match vectors starting from scheme with the smallest NUM, for all the schemes allocated to a given physical port (see [Section 5.10.4.1, “Scheme Selection”](#)). If no match is found, no scheme is selected, and the NIA of the FMan is taken from FMKG\_GCR (see [Section 5.10.3.1, “KeyGen General Configuration Register \(FMKG\\_GCR\)”](#)). In this case, the KeyGen does not perform additional actions.



**Figure 5-349. Scheme Selection Flow**

#### **Example 5-2. Indirect Scheme Selection—Example 1**

To implement a key generation and classification per protocol stack, program each hard header examination sequence (HXS) in the parser to assert 1 bit of the LCV. This means that each bit represents a header type.

Configure the line-up enable confirmation mask (LECM) in the Parser (see [Section 5.9.3.1.1, “Port x Parse Memory Direct Access Registers”](#)).

**Table 5-394. LECM Configuration**

	Bit <sup>1</sup>	LECM =
Ethernet	0	0x80000000
VLAN	1	0x40000000
IPv4	2	0x20000000
TCP	3	0x10000000

<sup>1</sup> The position of the bits is user-programmable.

Each scheme match vector is programmed with a different protocol stack.

**Table 5-395. Scheme Match Vector Configuration**

Scheme	Register	Value	Selected If:
Scheme 0 match vector	FMKG_SE_MV0	0xC000_0000	The frame has Ethernet and VLAN headers
Scheme 1 match vector	FMKG_SE_MV1	0xA000_0000	The frame has Ethernet and IPv4 headers and no previous scheme is selected
Scheme 2 match vector	FMKG_SE_MV2	0x3000_0000	The frame has IPv4 and TCP headers and no previous scheme is selected
Scheme 3 match vector	FMKG_SE_MV3	0x0000_0000	The previous schemes are not selected, because it masks all the bits

---

**Example 5-3. Indirect Scheme Selection—Example 2**

---

Input frames are of one of following types:

- IPv4
- IPv6
- TCP
- UDP.

The user must perform different Custom Classifier actions for each one of these input frame types.

The goal is to minimize the usage of schemes, which is a scarce resource. To accomplish this, there should be an LCV, which allows multiple CCBASE for the FMan Controller by using one KeyGen scheme.

The Parser generates the LCV as it proceeds with the parsing of the frame headers from the outer header to the inner headers. Each HXS applies a user-defined line-up enable confirmation mask, which is ORed with the current LCV, so the final LCV is the result of all the ORs done on the parsing path. See [Section 5.9.4.5.3, “Line-up Enable Confirmation Mask”](#) for more details.

This example focuses on the 4 msbs of the LCV; other bits are not relevant. The user may configure which bits in the LCV are set at each stage of the parsing by configuring the LECM for that stage.

The Parser line-up enable confirmation mask of the HXSs are configured to set the 4 bits as show in [Table 5-396](#).

**Table 5-396. HXS Configuration**

HXS	Line-up Enable Confirmation Mask[0..3] =
IPv4	0001
IPv6	1001
UDP	0010
TCP	0110

Therefore, the final LCV for an IPv6 UDP frame is 1011 (1001 OR 0010); similarly the final LCV for an IPv4 TCP frame is 0111.

For all the frames, the 2 lsbs of the final LCV are ‘11’; these bits are used to select the KeyGen scheme (that is, the same scheme is selected for all the frames). For this example, the scheme match vector is FMKG\_SE\_MV0 = 0x3000\_0000.

The 2 msbs differ for each one of the four combinations of frames; these bits are used to select the CCBASE, yielding to a different CCBASE for the FMan Controller Custom Classifier, thus processing each frame with a different Custom Classifier Action Descriptor. For this example, the Custom Classifier Bit Select Register is FMKG\_SE\_CCBS = 0xC000000.

---

### 5.10.4.2 Key Generation

The Key is composed of a user-configurable combination of fields extracted from the frame header’s Parser result, portID, and FQID (from the AD). The length of the key is user-configurable up to a maximum of 56 bytes.

In each scheme, the user configures the key generation in two types of extract commands:

- Extract Known Field Commands
- Generic Known Field Commands

The KeyGen scheme has a place-holder for nine extract commands. The commands are enabled with user-configurable bits in the commands themselves.

It is assumed that the parse results are available in the internal context (IC) before the KeyGen extracts the key, because the KeyGen uses fields from the parse results (for example, the offsets for the key extraction). It is also possible to generate a key or part of a key from attributes that are not extracted from the frame headers, such as the portID (Rx or O/H), or FQID, which is placed in the action descriptor (AD).

#### NOTE

If the key is composed of these fields, it is not necessary to run the parser before the KeyGen.

#### 5.10.4.2.1 Extract Known Field Commands

The Extract Known Field command specifies from what fields from known protocols (for example, IP, TCP, and so on.) of the frame headers the key is composed. Known protocols are protocols that are recognized by the Parser and marked in the Parser results (see [Section 5.9.4.10, “Parse Result”](#)). Key features of the Extract Known Field commands are as follows:

- The user sets the bits in the Extract Known Fields command corresponding to the fields that need to be extracted.
- The size of each field is predefined.
- The KeyGen always validates the field before it is extracted (see [Section 5.10.4.2.3, “Field Validation”](#)).

See [Section 5.10.3.12.2, “KeyGen Scheme Entry Extract Known Fields Command Register \(AWR2\\_RFMODE\\_FMKG\\_SE\\_EKFC\)”](#).

#### 5.10.4.2.2 Generic Known Field Commands

The Generic Extract commands are used to extract fields from headers that are not recognized by the Parser, or to extract some other information about the frame that is not present in the headers. With this mechanism, it is possible to form a key composed of fields beyond L4 headers or from proprietary headers. Key features of the Generic Known Field commands are as follows:

- It is possible to configure the size of the extracted field (up to 16 bytes) and its offset from a known header; in this case, the parse results are used for the offset to the header.
- The user may configure the KeyGen to validate or not validate an extracted field.
- The user can configure a mask to the last (least significant) byte of the field being extracted. See [Section 5.10.3.12.9, “KeyGen Scheme Entry Generic Extract Command Register \(AWR<9+i>\\_RFMODE\\_FMKG\\_SE\\_GEC<i>\)”](#).
- The user can apply a bit mask to some of the extracted fields (generic or known). Up to four masks are supported. See [Section 5.10.3.12.4, “KeyGen Scheme Bit Mask Command High Register \(AWR4\\_RFMODE\\_FMKG\\_SE\\_BMCH\)”](#).
- The size of the field is not changed; the masked bits in the field are reset.

The order of extraction of the fields is as follows:

1. The Known Fields command is scanned first from bit 0–31.
2. The Generic Extract commands are scanned from command 0–7.

#### 5.10.4.2.3 Field Validation

The KeyGen is able to validate the fields being extracted from the headers before they are concatenated to the key. When validation is done, the KeyGen checks if the extracted header exists in the packet (that is, the parse result header offset is not 0xFF) and that it does not contain errors (in the corresponding parse result [LxR, x=2,3,4]). If the validation fails, the KeyGen places a default value in place of the extracted header. The default value is configurable for each extracted field in the FMKG\_SE\_EKDV register (see [Section 5.10.3.12.3, “KeyGen Scheme Entry Extract Known Default Value Register \(AWR3\\_RFMODE\\_FMKG\\_SE\\_EKDV\)”](#)). One of four values may be chosen for the default value.

#### 5.10.4.3 Hash Value Generation

KeyGen performs a hash function on the extracted key. The hash function generates a uniform distribution of the keys to a finite (relatively small) set of values. This mechanism is used to do one of the following:

- Distribute frames to FQs
- Perform an exact match table lookup with the Custom Classifier module

Calculating the hash function over the flow ensures that the same flow is always enqueued to the same FQ. The CRC-64-ECMA-182 is a 64-bit CRC polynomial used in the KeyGen for the hash function calculation. This hash function has good distribution properties in order to evenly distribute the flows among the frame queues.

The polynomial is CRC-64-ECMA-182:  $x^{64} + x^{62} + x^{57} + x^{55} + x^{54} + x^{53} + x^{52} + x^{47} + x^{46} + x^{45} + x^{40} + x^{39} + x^{38} + x^{37} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{27} + x^{24} + x^{23} + x^{22} + x^{21} + x^{19} + x^{17} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^4 + x + 1$ . The Hash Init value is 0xFFFF\_FFFF\_FFFF\_FFFF\_FFFF

The hash is calculated on the extracted key, and an eight byte hash result is generated. The hash result is shifted then masked. It is possible to configure separate mask a shift values for each scheme, and for the FQID and PNUM separately (see [Section 5.10.3.12.7, “KeyGen Scheme Entry Hash Configuration Register \(AWR7\\_RFMODE\\_FMKG\\_SE\\_HC\)](#),” and [Section 5.10.3.12.8, “KeyGen Scheme Entry Policer Profile Command Register \(AWR8\\_RFMODE\\_FMKG\\_SE\\_PPC\)](#)”).

The user needs to calculate the hash function in SW in order to add a new entry into the Custom Classifier lookup tables.

#### 5.10.4.3.1 Symmetric Hash Function

Support symmetric hash for source and destination pairs:

- Mac source and destination addresses (known extracts)
- IP source and destination addresses (known extracts)
- Layer 4 source and destination ports (known extracts)

The calculation of the hash on the key is done as if the source Mac/IP/L4 port and the destination Mac/IP/L4 are XORed to each other respectively. This assures that the hash result stays the same when the source and destination fields are exchanged respectively.

The key written to the internal context of the frame shows the original fields extracted from the frame headers (i.e. different source and destination field) and does not reflect the XOR operation performed for the symmetric hash.

For hash calculation the fields which are XORed appear only once in the key. Therefore for hash, they key is shorter.

The symmetric hash field is done only for the following known extracted headers:

- FMKG\_SE\_EKFC[MACDST], FMKG\_SE\_EKFC[MACSRC]
- FMKG\_SE\_EKFC[IPSRC\_1], FMKG\_SE\_EKFC[IPDST\_1]
- FMKG\_SE\_EKFC[IPSRC\_N], FMKG\_SE\_EKFC[IPDST\_N]
- FMKG\_SE\_EKFC[L4SRC], FMKG\_SE\_EKFC[L4DST]

Notice, when symmetric mode is active, both source & destination known extract command must be asserted.

For inner header IP extract, where Min Encap header exists, if source address does not exist the destination will be XORed with the chosen default value.

See [Section 5.10.3.12.2, “KeyGen Scheme Entry Extract Known Fields Command Register \(AWR2\\_RFMODE\\_FMKG\\_SE\\_EKFC\)](#)

#### 5.10.4.4 Keygen Distribution Function Value (KDFV)

The Keygen distribution function value (KDFV) generates the final FQID, and Policer profile (PNUM), and Storage profile ID. The KDFV, a 24-bit value, is calculated based on the values programmed in the Keygen scheme:

$$KDFV[0:23] = (24 \text{ lsbs of}(Hash result[0:63] >> (FMKG_SE_HC[SHIFT]))) \&$$

Hash Mask (FMKG\_SE\_HC[MASK[0:23]])) |  
 Or Data<sub>0</sub>[0:23] (FMKG\_SE\_GEC0) | ...Or Data<sub>7</sub>[0:23] (FMKG\_SE\_GEC7)

The last line in the question is the ‘OR Data Vector’.

#### 5.10.4.4.1 OR Data Vector Generation

The Or Data Vector allows the user to use some bits directly from the frame headers (without hash function) to form the FQID. For example, it is possible to use the VLAN priority bits or the IP TOS bits, as part of the FQID, thus distributing frames to different queues according to the QoS attributes of the frame. For maximum flexibility, the FMan allows placing these bits anywhere in the FQID.

Part of the generic extract commands can be used as OR commands and directly influence the FQID and Policer profile (PNUM) values by configuring FMKG\_SE\_GEC[TYPE] = 1. When TYPE = 1, the Extract command extracts 1 byte from the frame according to the validate Header type (FMKG\_SE\_GEC[HT]) and the additional extract offset (FMKG\_SE\_GEC[EO]). The 1 extracted byte is masked by FMKG\_SE\_GEC[MASK], is right-padded with 24 zeros, and is rotated right according to FMKG\_SE\_GEC[SIZE]. This generates an OR data 32-bit vector for the specific generic command. Up to eight OR data vectors can be generated according to the number of generic commands (FMKG\_SE\_GEC0–7). All eight OR data vectors (one for each generic extract command) are bit-wise logical ORed with the masked hash result and the FQID base (FMKG\_SE\_FQB). The 8 lsbs of the OR data are used to calculate directly the Policer profile.

Or Data[0:31] = Rotate Right(Right Padded (Extracted byte & MASK))

#### 5.10.4.5 Frame Queue ID (FQID) Generation

FQID generation is enabled if FMKG\_SE\_HC[NO\_FQID\_GEN] = 0, otherwise it is disabled. See Section 5.10.3.12.7, “KeyGen Scheme Entry Hash Configuration Register (AWR7\_RFMODE\_FMKG\_SE\_HC).”

The FQID is generated according to the following:

FQID[0:23] = KDFV[0:23] | FQID Base (FMKG\_SE\_FQB[FQB BASE])

FQID Base is used to map the final FQID in a user configurable area of the device FQID space.

Figure 5-350 shows the FQID generation.

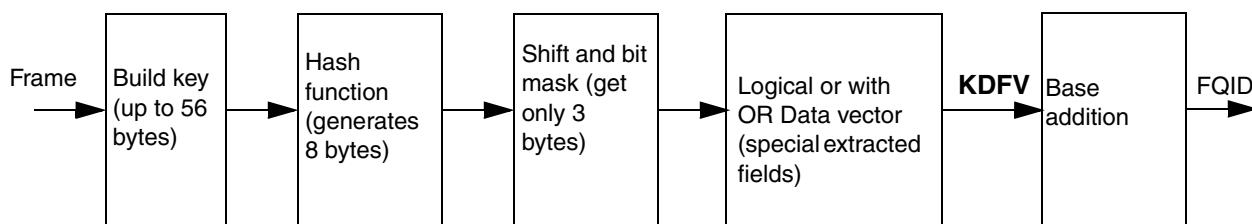


Figure 5-350. FQID Generation Flow

For example, after the hash function, given the following conditions:

- The result is 0x6512445425436354
- The mask is 0xffC
- The base is 0x4000
- The priority bit value is 0x2

After the bit mask, the result is 0x354, after the base addition the result is 0x4354, and after the logical OR the result is 0x4356, which is the requested FQID.

#### 5.10.4.6 Policer Profile Number (PNUM) Generation

The Policer profile (PNUM) is generated according to the following:

$$\begin{aligned} \text{PNUM[0:7]} &= (\text{8LSBits of(KDFV[0:23] >> SHIFT (FMKG_SE_PPC[PPSH,PPSL] ))} \& \\ \text{PNUM Mask (FMKG_SE_PPC[PPMASK] ))} \mid \\ \text{PNUM Base (FMKG_SE_PPC[PPBASE])} \mid \\ \text{Or Data}_0[24:31] \dots \text{Or Data}_7[24:31] \text{ (Optional, according to validity of generic extract command).} \end{aligned}$$

Policer profile generation is enabled if FMKG\_SE\_PP[NO\_PP\_GEN]= 0, otherwise it is disabled. See [Section 5.10.3.12.8, “KeyGen Scheme Entry Policer Profile Command Register \(AWR8\\_RFMODE\\_FMKG\\_SE\\_PPC\).”](#)

#### 5.10.4.7 KeyGen Storage Profile ID Generation

For details on storage profile see [Section 5.3.7, “Virtual Storage Profiles”](#) and [Section 5.5.4.1, “Storage Profiles.”](#) The KeyGen generates a 6 bit relative storage profile ID (RSPID) and writes it to the Internal Context Action descriptor (ICAD). ICAD bits 0,1 are always set to zero by KeyGen:

$$\begin{aligned} \text{SPID[0:7]} &= \{0,0,\text{RSPID}\}=\{0,0, \\ &((\text{KDFV[0:23] >>FMKG_SE_VSP[KSPS[0:4]]}) \& \text{FMKG_SE_VSP[KSPMASK [0:5]]}) \mid \\ &\text{FMKG_SE_VSP[KSPID[0:5]]}\} \end{aligned}$$

For the final absolute storage profile ID, the SPID result from this equation is subject to the manipulation with FMBM\_SPICID values. [5.5.4.1.1, “Storage Profile Virtualization per port \(for Rx and Offline ports\).”](#)

Storage profile generation is enabled if FMKG\_SE\_VSP[KSPEN]= 0, otherwise it is disabled. See [Section 5.10.3.12.15, “KeyGen Scheme Entry Virtual Storage Profile Register \(AWR8\\_RFMODE\\_FMKG\\_SE\\_VSP\).”](#)

#### 5.10.4.8 Custom Classifier Base (CCBASE) Update

It is possible to execute a different Custom Classifier for each protocol stack. The motivation is to allow the KeyGen to generate the pointer to the first AD to be executed by the FMan controller Custom Classifier on a per-frame basis.

The KeyGen Custom Classifier bit-select entry modifies CCBASE[0:31], which are programmed in the BMI.

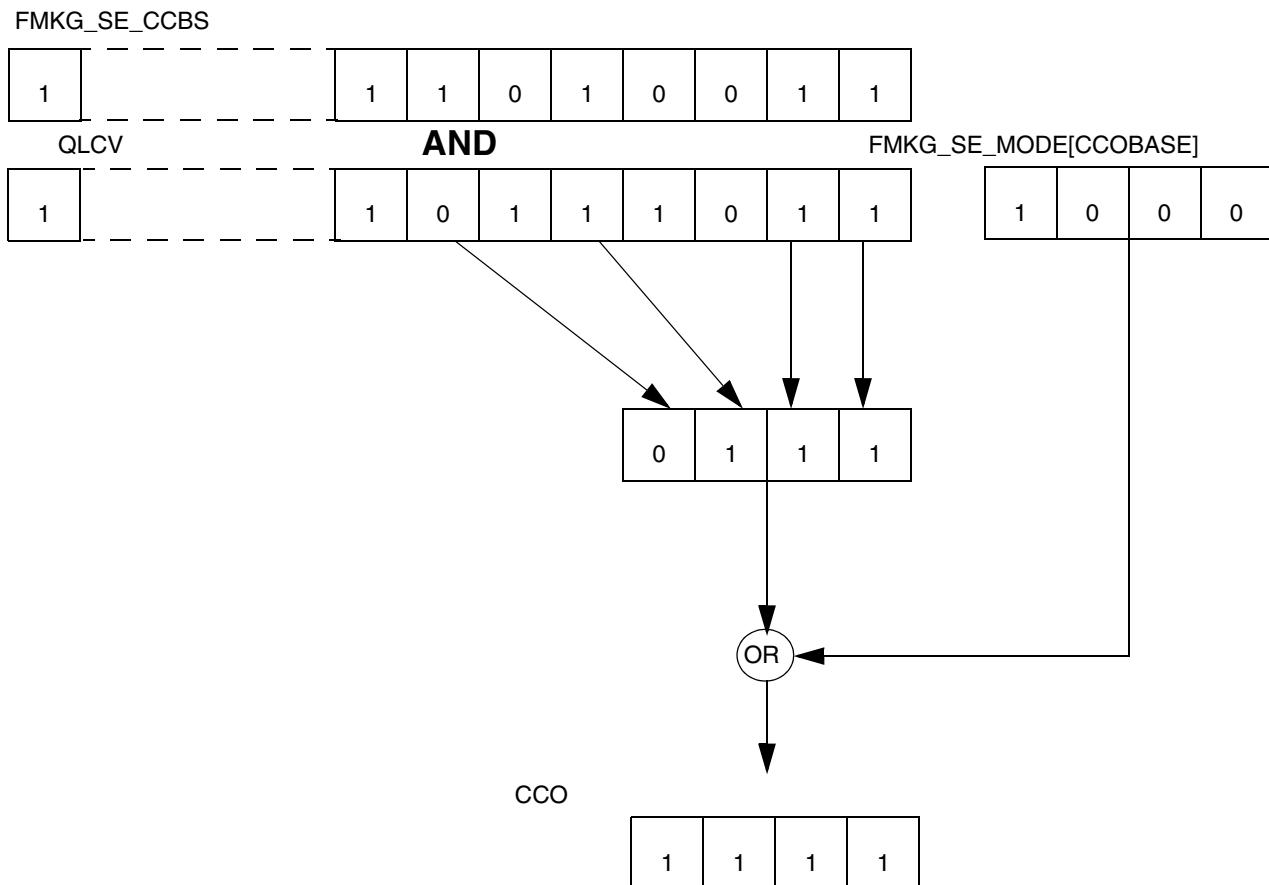
The user can configure which of the qualified LCV bits (QLCV[0:31]; see [Figure 5-349](#)) are used to define the Custom Classifier offset (CCO). The Custom Classifier base is configured by the user through the BMI and it is placed in the internal context (IC) as CCBASE[0:31].

The KeyGen can change the 8 LSBs (CCBASE[24:31]) in the following ways:

- The 4 LSBs (CCBASE[28:31]) are always zero.
- The rest (CCBASE[24:27]) are calculated using FMKG\_SE\_CCBS ([Section 5.10.3.12.12, “KeyGen Scheme Entry Custom Classifier Bit Select Register \(AWR20\\_RFMODE\\_FMKG\\_SE\\_CCBS\)”](#)) and FMKG\_SE\_MODE[CCOBASE] ([Section 5.10.3.12.1, “KeyGen Scheme Entry MODE Register \(AWR1\\_RFMODE\\_FMKG\\_SE\\_MODE\)”](#)).

The KeyGen writes the calculated Custom Classifier offset CCO[0:3] concatenated with 4 zero bits to the IC CCBASE last byte (LSB) only if it is requested by the given NIA. (NIA[CCEN]=1; see [Section 5.10.4.10.1, “KeyGen NIA Action Codes”](#)). The remainder bytes of the CCBASE are programmed in the FMBM\_RCCB, FMBM\_OCCB or (in FMan\_v3) FMBM\_TCCB).

The calculation of the CCO[0:3] is described in [Figure 5-351](#).



**Figure 5-351. Custom Classifier Offset Calculation**

### 5.10.4.9 Next Invoke Action (NIA) Generation

After frame processing, the KeyGen pushes the frame to the next invoked action (NIA). The NIA is determined in the following ways:

- When there is no scheme selection, the NIA is taken from FMKG\_GCR[DEFNIA] ([Section 5.10.3.1, “KeyGen General Configuration Register \(FMKG\\_GCR\)”](#)).
- When a scheme is selected, the NIA is taken from AWR1\_RFMODE\_FMKG\_SE\_MODE[NIA] ([Section 5.10.3.12.1, “KeyGen Scheme Entry MODE Register \(AWR1\\_RFMODE\\_FMKG\\_SE\\_MODE\)”](#)).

The NIA is updated with the calculated PNUM if AWR1\_RFMODE\_FMKG\_SE\_MODE[PL] is set. This means that it is likely that the NIA is the Policer and its PNUM was calculated by the KeyGen ([Section 5.10.4.6, “Policer Profile Number \(PNUM\) Generation”](#)).

### 5.10.4.10 KeyGen Inputs and Outputs

#### 5.10.4.10.1 KeyGen NIA Action Codes

The KeyGen starts to process the frame by receiving the next invoked action (NIA) from the FMan frame processing manager (FPM). See [Section 5.3.5, “FMan User-Configurable Pipeline Architecture—Introducing the NIA.”](#)

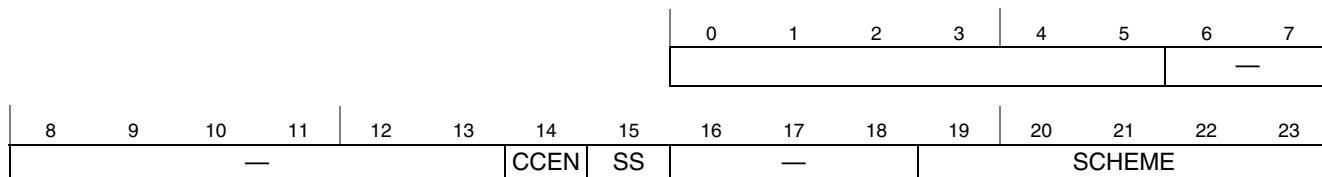


Figure 5-352. KeyGen NIA Bit Mapping

Table 5-397. KeyGen NIA Action Code (AC) Fields Description

Bits	Name	Description
6-13	—	Reserved
14	CCEN	Custom classifier enable 0 Does not write the calculated CC base offset to the internal context (IC) 1 Overwrites the internal context (IC) CCBASE last byte by the calculated CC base offset
15	SS	Scheme selector 0 Scheme is selected according to parse result; SCHEME field is ignored 1 Take the scheme number from SCHEME field
16-18	—	Reserved
19-23	SCHEME	Scheme ID Represents the selected scheme in case the SS field is set

### 5.10.4.10.2 KeyGen Inputs from FMan Memory

The KeyGen reads the Parser result (PR), the frame descriptor (FD) and the frame data.

**Table 5-398. KeyGen Read Data**

Field	Size	Description
PR	32 bytes	Parse result Holds the layers results and the headers offsets
FD	16 bytes	Frame descriptor (FD) Holds the frame length and the frame status
Frame	Up to 256 bytes	Frame data The KeyGen reads the parsed frame data + configurable offset.

### 5.10.4.10.3 KeyGen Outputs to FMan Memory

The KeyGen calculates and writes to the (IC) of the FQID, PNUM, and the last byte of the Custom Classifier base (CCBASE) if NIA[CCEN] is asserted. [Table 5-399](#) describes the information that the KeyGen writes.

**Table 5-399. KeyGen Write Data**

Field	Size	Description
FD[Status]	1 byte	Frame Descriptor (FD) Status bits
AD	8 bytes	Action Descriptor (AD) Writes the action descriptor related to the scheme type
CCBASE	1 byte	Custom classifier base Writes to the last byte of the CCBASE only if NIA[CCEN] is asserted
KS	1 byte	Key size The amount of extracted bytes
HASH	8 bytes	Hashed key Writes the result of the hash 64 bits
KEY	Up to 56 bytes	Key Writes the generated key

### Frame Descriptor (FD) Status

The KeyGen updates the FD status bits if it encounters an error while processing the frame.

**Table 5-400. Frame Descriptor (FD) Status Entry**

Field	Size	Description
EOF	1 bit	Extract out of the frame If the extract offset is bigger than the frame-parsed headers (plus the wanted payload extract—see <a href="#">Section 5.10.3.8, “KeyGen Frame Data Offset Register (FMKG_FDOR)”</a> ) or buffer length (256 bytes), an error occurred. The extract data is taken from the configured default value. 0 No out-of-frame extract occurred. 1 Out-of-frame extract occurred.
NSS	1 bit	No scheme selection If a scheme is not found, classification does not occur. The frame moves to the default NIA without generating a key and without changing the FQID, SPID, operational mode bits and PNUM. 0 Scheme selection occurred. 1 No scheme selection occurred.
KSO	1 bit	Key size overflow If the selected scheme has a scheme error configuration that configured a key size more than 56 bytes. 0 No key size overflow occurred. 1 Key size overflow occurred on the selected scheme.

### Action Descriptor (AD)

The KeyGen writes an action descriptor (AD) to the IC. The AD is composed of an FQID and a Policier profile number (PNUM).

**Table 5-401. Action Descriptor (AD) Entry**

Field	Size	Description
FQID	24 bits	Frame queue ID (FQID) FQID = HK Shift(Hash Result) and HK Mask or Or result or FQ Base
PNUM	8 bits	Policer profile (PNUM) PNUM = PNUM Shift[HK Rot(Hash Result) and HK Mask or Or result] and PNUM Mask or PNUM Base

### 5.10.4.11 KeyGen Debug Functionality

During packet processing, the FMan can trace packet processing flow and trap a packet. Each FMan execution module that took part in packet processing can write debug information to the debug area of the internal context (IC).

All FMan flows are initialized at the FMan Buffer Manager interface (BMI), thus any debug flow should also be initiated at the BMI. Each port can be associated with any of the three debug flows (A, B, and C), one or more regardless of the other ports. The following description pertains to one flow but applies to all.

Upon task dispatch, the KeyGen checks its debug trap conditions for each flow. The KeyGen does one of the following according to its internal debug configuration:

- Attempts to trap the frame by evaluating the configured criteria. A successful trap causes it to write its debug context and propagate the debug signal (continue debug for this flow) to the next processing module during task dispatch. If a frame is detected by traps related to different debug flows (A, B, or C) within the same module, the most verbose trace is applied. For example, if debug

flow A is configured to dump debug data in terse mode and debug flow B is configured for verbose mode, and the criteria for both flows are met, then the verbose trace format is chosen. Failing to trap causes the tracing of that packet within all subsequent modules to stop.

- Traps in bypass state, which acts as a successful trap. This allows the packet to be ignored by the current module, but remain a packet of interest for all downstream FMan modules.

#### 5.10.4.11.1 Debug Trace

KeyGen trace takes place only if debug was requested during task dispatch and if the frame has been trapped (for this action trap, bypass is regarded as a successful trap). The KeyGen writes trace information into the debug section of the internal context (IC). The trace is controlled by TL\_X (trace level for flow X) dedicated for each of the three independent debug flows. This selects the amount of trace information that is stored when the trace is performed. If the frame is identified by more than one debug flow, then the most verbose trace is applied. During frame dispatch, the current debug offset is given. This value is updated according to the amount of trace that was written by the KeyGen so that the updated offset can be delivered to the next module in the frame flow.

The KeyGen writes the contents of [Table 5-402](#) into the debug trace.

**Table 5-402. FMan KeyGen Trace Debug Format**

Verbosity Level		Size	Field Descriptions	
Very verbose trace	Verbose trace	Minimum trace	1 byte	Trace config <ul style="list-style-type: none"> <li>• The selected verbosity level (2 bits)</li> <li>• Trace size in 4-byte granularity (6 bits)</li> </ul>
			3 bytes	Received NIA
	—	4 bytes	FPM timestamp	
			8 bits	Selected scheme
		8 bits	Internal frame pointer (IFP[16:23])	
			16 bits	Zero
		4 bytes	FD command field	

After the KeyGen has written its trace data, the final debug offset should point to the next available byte in the internal context for the next module. If the debug offset extends beyond the 256 bytes of the internal context, then the trace is not written. When the debug offset is at the end of the buffer and not all of the 10 bytes can be written, the KeyGen writes only the first bytes of the trace subject to available space. The trace size remains the same according to the trace level. The debug offset is in 4-byte granularity. The maximum trace size is 12 bytes with 2 bytes reserved.

#### 5.10.4.11.2 Debug Traps

The KeyGen has five programmable sets of debug traps for each debug flow. The debug traps for a specific flow are related to each other through boolean operations, forming a combined debug event. If a debug event occurs, the next module is signaled during task dispatch that the frame is in an active debug state. The user may choose to bypass a trap, meaning that the particular trap operation is considered successful for any frame.

## 5.10.5 KeyGen Initialization

The KeyGen registers and RAM are initialized to their reset values. Out of reset, all scheme entries and classification plans are zeros. The KeyGen schemes and classification plans can be programmed only in indirect mode.

To initialize the KeyGen, complete the following sequence of initialization steps:

1. Initialize the global register (without enabling the KeyGen) by configuring FMKG\_GCR[EN] = 0.
2. Initialize the KeyGen Port Partition.
3. Initialize the KeyGen Scheme Entry.
4. Initialize the KeyGen Classification Plan Group.
5. Enable the KeyGen by configuring FMKG\_GCR[EN] = 1.

### 5.10.5.1 Initialize the KeyGen Port Partition

At any time, software can use the atomic access registers to update a full port-partition entry. However, only one software entity can use the atomic access registers at any one time. In addition, if Host commands are used, software must not use the atomic access registers, because the FMan Controller also uses them to execute the Host commands.

To initialize the port partition, complete the following sequence of steps:

1. Write to FMKG\_PE\_SP and FMKG\_PE\_CPP the atomic access words in registers.
2. Activate the atomic write action by writing FMKG\_AR with the following:
  - GO = 1
  - RW = 0
  - NUM = don't care
  - HPORTID = the wanted hardware portID
  - SEL = 2
  - WSEL = 0xc000

### 5.10.5.2 Initialize the KeyGen Scheme Entry

At any time, software can use the atomic access registers to update full or partial scheme entry.

#### 5.10.5.2.1 Initialize a Full Scheme Entry

To initialize a full scheme entry, complete the following sequence of steps:

1. Write all the scheme atomic access words in registers FMKG\_SE\_MODE through FMKG\_SE\_MV (for a total of 20 or 21 registers, depending on the statistic counter update).
2. Activate the atomic write action by writing FMKG\_AR with the following:
  - GO = 1
  - RW = 0
  - NUM = scheme number

- SEL = 0
- WSEL = 0x8000 for also changing the scheme statistic packet counter (FMKG\_SE\_SPC)
- OR
- WSEL = 0x0000 for not changing the scheme statistic packet counter

### **5.10.5.2.2 Initialize a Partial Scheme Entry**

To initialize partial scheme parameters by read-modify-write, complete the following sequence of steps:

1. Activate the atomic read action by writing FMKG\_AR with the following:
  - GO = 1
  - RW = 1
  - NUM = scheme number
  - WSEL = don't care (all words are read regardless of WSEL.)
2. Write the selected FMKG\_SE\_\* atomic access registers.
3. Activate the atomic action by writing FMKG\_AR with the following:
  - GO = 1
  - RW = 0
  - NUM = scheme number
4. Set only the selected bits in WSEL (associated with words that need update).

### **5.10.5.3 Initialize the KeyGen Classification Plan Group**

At any time, software can use the atomic access registers to update a full or partial classification plan group entry.

#### **5.10.5.3.1 Initialize a Full Classification Plan Group**

To initialize a full classification plan group, complete the following sequence of steps:

1. Write all the group atomic access words in registers FMKG\_CPE0 through FMKG\_CPE7 (total of eight registers).
2. Activate the atomic write action by writing FMKG\_AR with the following:
  - GO = 1
  - RW = 0
  - NUM = classification plan group number
  - SEL = 1
  - WSEL = 0xFF00

#### **5.10.5.3.2 Initialize a Partial Classification Plan Group**

To initialize partial group parameters by read-modify-write, complete the following sequence of steps:

1. Activate the atomic read action by writing FMKG\_AR with the following:
  - GO = 1

- RW = 1
  - NUM = classification plan group number
  - WSEL = don't care (all words are read regardless of WSEL.)
2. Write the selected FMKG\_CPE\* atomic access registers.
  3. Activate the atomic action by writing FMKG\_AR with the following:
    - GO = 1
    - RW = 0
    - NUM = classification plan group number
  4. Set only the selected bits in WSEL (associated with words that need update).

#### **NOTE**

For FMKG\_CPG0–7, use registers FMKG\_SE\_MODE through FMKG\_SE\_PPC.

### **5.10.6 KeyGen Events**

The types of KeyGen events are described in [Table 5-403](#).

**Table 5-403. Types of KeyGen Events**

	<b>Event Type</b>	<b>Definition</b>
Per frame	Out of frame extract error event	Extracts data from the selected default value and indicates this event in the FD status bit
	No scheme selection	<ul style="list-style-type: none"> <li>• Gets the default NIA FMKG_GCR[DEFNIA]</li> <li>• Asserts a flag in the frame status bits.</li> </ul>
Per scheme	Key size overflow	Error event per scheme. Builds more than a 56-byte key

## 5.11 Frame Manager—Policer

### 5.11.1 Policer Overview

The Policer supports implementation of differentiated services at line speed on the Frame Manager (FMan) receive or offline parsing paths. It holds 256 traffic profiles in internal memory, each profile implementing RFC-2698, RFC-4115, or pass-through mode. Each mode can work in either color-blind or color-aware mode and pass or drop packets according to their resulting color.

The FMan Policer is typically part of the FMan receive path and it enables high-performance implementation of differentiated services on receive traffic running at line speed. It can also be used to protect on-chip cores from excessive traffic or packet rates.

The Policer holds 256 independent traffic profiles; each profile can work in one of the following modes:

- Pass-through—This mode does not implement differentiated services. It can be used for some applications to block/pass selected flows and set a constant color for these flows. The default (disabled) mode is color-aware pass-through with no dropping of packets.
- RFC-2698—A two-rate three-color marker
- RFC-4115—A differentiated service two-rate three-color marker with efficient handling of in-profile traffic

All modes can run in color-aware or color-blind mode. In color-aware mode, packet pre-color information is taken into account by the algorithm, or propagated in pass-through mode. In color-blind mode, the Policer sets the frame color, disregarding the pre-color information. Each Policer profile can program the next module action according to each resulting color. Typically, that action would be used to select a drop or pass action for the packets according to the Policer coloring action.

Any single Policer traffic profile can be used to control either single receive flow traffic or accumulated traffic of multiple receive flows. In addition, Policer profiles can be concatenated to each other by selecting the Policer as the next FMan processing module. Each Policer profile holds its own configuration and state variables, which is referred to as *profile context data*. The full profile context data is stored in a 64-byte entry in the Policer profile RAM (PRAM). A single profile entry is used to shape the traffic accumulated by all flows that are assigned to it and can be configured independently from other profile entries.

On a typical application, Policer profile selection is achieved as follows:

- Selected fields of the Parser results directly map an entry to an internal lookup table residing in FMan memory. The entry from this table selects the Policer profile.
- Selected fields of the Parser results go to the KeyGen function, which then maps the entry to an FMan internal lookup table residing in the FMan memory. The table entry selects the Policer profile.
- Selected fields of the Parser results go to the KeyGen function, which directly maps them to a Policer profile with no lookup table search.
- The Policer has the capability to map the selected profile per port-ID regions to allow virtual MAC separation for the host application.

The packet pre-color is used by profiles configured to work in color-aware mode. The pre-coloring of the packet is achieved as follows:

- When the BMI initializes IC for a new Rx frame, it writes a programmable default color to the color field (FCL) in the FD status.
- The Parser may overwrite the default color based on its parsing decisions. For example, it could consider VLAN a priority and/or IP TOS, physical port-ID and IP addresses for the color selection.
- A Policier profile that works in color-aware mode reads the packet color field and uses it in its defined algorithm. It may then overwrite the color field with a new color based on its traffic policing results. Pre-color value of  $2'b11$  can be treated as either GREEN, YELLOW, RED or override (pass-through) based on the profile configuration. A Policier profile that works in color-blind mode ignores the pre-color information and works according to its color-blind algorithm definition.

Each profile entry includes configuration and run-time variables. The configuration selects the algorithm (pass-through, RFC-2698, or RFC-4115), how the results are treated, calculation units, accuracy control, and other algorithm-specific parameters. The run-time variables include status of token buckets, last time-stamp value, and profile's statistic counters.

When the Policier completes the algorithm on a single packet and re-colors it, it returns the control back to the FPM with an indication of the next processing module. The next processing module is programmable per final frame color. The following are examples of the next processing modules:

- BMI—In the case of a packet colored GREEN or YELLOW, it would normally not be dropped. If a packet is colored RED, a quick drop of the RED packets is possible, indicated by the NIA coding. Based on NIA, the BMI would either initiate DMA access to copy the most updated frame IC from FMan memory to external memory and pass the control to the QMI, or close the task in case of packet drop.
- Policier—In the case of profile concatenation, one or more profiles can be reiterated through the Policier and select a new profile number. This configuration allows multiple policing schemes to run on the same flows, or aggregate multiple profiles into one profile.

## 5.11.2 Policier Features

- Holds 256 independent profiles. Each profile has the following features:
  - Supports pass-through mode
    - In color-aware mode, packet pre-color is copied to the packet color
    - In color-blind mode, the default color is programmable per profile
    - Configurable next module and command per packet color. Each color action can be programmed to pass or drop the packet.
    - Programmable treatment of packets with “override” pre-color
  - Supports RFC-2698—A two-rate three-color marker
    - Works in either color-blind or color-aware mode
    - Two configurable traffic rates: peak information rate (PIR) and committed information rate (CIR)

- Holds two token buckets P and C. Each token bucket has associated burst (bucket) size: peak burst size (PBS) and committed burst size (CBS)
  - Support for PACKET or BYTE rate modes. For PACKET mode PIR, CIR, PBS and CBS measure full packets. For BYTE mode they measure byte count.
  - Support different packet length interpretations in BYTE mode
  - Fixed point calculations with programmable fixed point location on timestamp representation to prevent accumulated errors
  - Marks packets as either GREEN, YELLOW or RED
  - Configurable next module and command per packet color. Therefore each color action can be programmed to pass or drop the packet.
  - Programmable treatment of packets with “override” pre-color
  - Automatic bucket status fixed point position calibration to maximize accuracy for the selected bucket size
- Supports RFC-4115—A differentiated service two-rate three-color marker with efficient handling of in-profile traffic
  - Work in either color-blind or color-aware mode
  - Configure two traffic rates: excess information rate (EIR) and committed information rate (CIR)
  - Holds two token buckets C and E. Each token bucket has associated burst (bucket) size: excess burst size (EBS) and committed burst size (CBS).
  - Support for PACKET or BYTE rate modes. For PACKET mode EIR, CIR, EBS, and CBS measure full packets. For BYTE mode, they measure byte count.
  - Support different packet length interpretations in BYTE mode
  - Fixed point calculations with programmable fixed point location on timestamp representation to prevent accumulated errors
  - Marks packets as either GREEN, YELLOW or RED
  - Configurable next module and command per packet color. Each color action can be programmed to pass or drop the packet.
  - Programmable treatment of packets with “override” pre-color
  - Automatic bucket status fixed point position calibration to maximize accuracy for the selected bucket size
- Holds global and per profile statistics counters
  - Global total packets counter
  - Global packet coloring statistics counters
  - Per profile GREEN, YELLOW and RED packet counters
  - Per profile active coloring counters (GREEN to YELLOW and GREEN/YELLOW to RED)
- Supports atomic profile updates at run time through configuration registers
- Automatic self “refresh” by dummy packets on all profiles to eliminate timestamp counter wraps
- Profile RAM (PRAM)

- A 16-Kbyte memory holding 256 profiles entries of 64 bytes each
- ECC protection—single bit error correct, double bit error detect, with 32-bit granularity
- Automated self-initialization support
- Virtual MAC separation for all existing Rx and Offline-parsing/Host Command port-ID numbers
  - The Port-ID can modify the profile selection to map a separate profile group,
  - Each profile group has continuous power of two number of profiles,
  - Programmed default profile mapping for other or non-enabled port-ID numbers

### 5.11.3 Policer Modes of Operation

This section discusses the various modes of operation.

#### 5.11.3.1 Disabled Mode

Out of reset, the Policer wakes in disabled mode, indicated by FMPL\_GCR[EN] = 0. The PRAM is self-initialized to all zeros. The Policer can be enabled by setting FMPL\_GCR[EN] = 1.

When FMPL\_GCR[EN] is cleared at run time, the Policer gracefully enters disabled mode. In this sequence, it stops accepting new packets while packets in the middle of processing run to completion. The busy bit in FMPL\_GSR continuously indicates the Policer operational status. Polling FMPL\_GSR[BSY] for a zero value after FMPL\_GCR[EN] is cleared indicates that all Policer tasks have finished.

#### 5.11.3.2 Active Mode

When the user sets FMPL\_GCR[EN], the Policer enters active mode. It is assumed that all the PRAM entries are already initialized at that time. Unused and unmodified entries retain the default configuration (all zeros), which selects color-aware pass-through mode with next invoked action (NIA) taken from FMPL\_GCR[DEFNIA]. See [Section 5.11.5.2, “Policer Profile Operation Modes,”](#) for a detailed description of the Policer active modes.

In this mode, the Policer provides runtime debug features of up to three debug flows and can dump trace data to the debug area in the frame’s internal context.

### 5.11.4 Policer Memory Map and Register Definitions

This section provides a summary of the Policer memory map. [Table 5-404](#) specifies the groups of registers used to configure the Policer. See [Section 5.4.2, “FMan Detailed Memory Map,”](#) for details of Policer memory space within FMan memory space.

**Table 5-404. Policer Memory Map**

Register Offset	Register	Access	Reset Value	Section/Page
<b>General Configuration and Status Registers</b>				
0x000	FMPL_GCR—FMan Policer General Configuration	R/W	0x0050_0002	<a href="#">5.11.4.1/5-479</a>

**Table 5-404. Policer Memory Map (continued)**

Register Offset	Register	Access	Reset Value	Section/Page
0x004	FMPL_GSR—FMan Policer Global Status Register	R	All zeros	<a href="#">5.11.4.2/5-480</a>
0x008	FMPL_EVR—FMan Policer Event Register	w1c	All zeros	<a href="#">5.11.4.3/5-482</a>
0x00C	FMPL_IER—FMan Policer Interrupt Enable Register	R/W	All zeros	<a href="#">5.11.4.4/5-482</a>
0x010	FMPL_IFR—FMan Policer Interrupt Force Register	W	All zeros	<a href="#">5.11.4.5/5-483</a>
0x014	FMPL_EEVR—FMan Policer Error Event Register	w1c	All zeros	<a href="#">5.11.4.6/5-483</a>
0x018	FMPL_EIER—FMan Policer Error Interrupt Enable Register	R/W	All zeros	<a href="#">5.11.4.7/5-484</a>
<b>Global Statistic Counters</b>				
0x020	FMPL_RPC—FMan Policer RED Packets Counter	R/W	All zeros	<a href="#">5.11.4.8/5-485</a>
0x024	FMPL_YPD—FMan Policer YELLOW Packets Counter	R/W	All zeros	<a href="#">5.11.4.9/5-485</a>
0x028	FMPL_RRPC—FMan Policer Recolored RED Packet Counter	R/W	All zeros	<a href="#">5.11.4.10/5-486</a>
0x02C	FMPL_RYPC—FMan Policer Recolored YELLOW Packet Counter	R/W	All zeros	<a href="#">5.11.4.11/5-486</a>
0x030	FMPL_TPC—FMan Policer Total Packet Counter	R/W	All zeros	<a href="#">5.11.4.12/5-487</a>
0x034	FMPL_FLMC—FMan Policer Frame Length Mismatch Counter	R/W	All zeros	<a href="#">5.11.4.13/5-487</a>
0x038–0x08B	Reserved	—	—	—
<b>Profile RAM Access Registers</b>				
0x08C	FMPL_PAR—FMan Policer Profile Action Register	R/W	0xA000_FFFF	<a href="#">5.11.4.14/5-488</a>
0x090	FMPL_PEMODE—FMan Policer Profile Entry Mode	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x094	FMPL_PEGNIA—FMan Policer Profile Entry GREEN Next Invoked Action	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x098	FMPL_PEYNIA—FMan Policer Profile Entry YELLOW Next Invoked Action	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x09C	FMPL_PERNIA—FMan Policer Profile Entry RED Next Invoked Action	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0A0	FMPL_PECIR—FMan Policer Profile Entry Committed Information Rate	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0A4	FMPL_PECBS—FMan Policer Profile Entry Committed Burst Size	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0A8	FMPL_PEPIR_EIR—FMan Policer Profile Entry Peak/Excess Information Rate	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0AC	FMPL_PEPBS_EBS—FMan Policer Profile Entry Peak/Excess Information Rate	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0B0	FMPL_PELTS—FMan Policer Profile Entry Last Timestamp	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0B4	FMPL_PECTS—FMan Policer Profile Entry Committed Token Status	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0B8	FMPL_PEPTS_ETS—FMan Policer Profile Entry Peak/Excess Token Status	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0BC	FMPL_PEGPC—FMan Policer Profile Entry GREEN Packet Counter	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>

**Table 5-404. Policer Memory Map (continued)**

Register Offset	Register	Access	Reset Value	Section/Page
0x0C0	FMPL_PEYPC—FMan Policer Profile Entry YELLOW Packet Counter	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0C4	FMPL_PERPC—FMan Policer Profile Entry RED Packet Counter	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0C8	FMPL_PERYPC—FMan Policer Profile Entry Recolored YELLOW Packet Counter	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0CC	FMPL_PERRPC—FMan Policer Profile Entry Recolored RED Packet Counter	R/W	All zeros	<a href="#">5.11.4.15/5-489</a>
0x0D0–0x0FF	Reserved	—	—	—
<b>Error Capture Registers</b>				
0x100	FMPL_SERC—FMan Policer Soft Error Capture	Mixed	All zeros	<a href="#">5.11.4.16/5-490</a>
0x104	FMPL_UPCR—FMan Policer Uninitialized Profile Capture Register	Mixed	All zeros	<a href="#">5.11.4.17/5-491</a>
0x108	Reserved	—	—	—
<b>Debug Trap and Trace Registers</b>				
0x10c	FMPL_DTRCR—FMan Policer Debug Trace Configuration Register	R/W	All zeros	<a href="#">5.11.4.18/5-492</a>
0x110	FMPL_FADBTCR0—FMan Policer Flow A Debug Trap Configuration Register 0	R/W	All zeros	<a href="#">5.11.4.19/5-494</a>
0x114	FMPL_FADBVALR0—FMan Policer Flow A Debug Value Register 0	R/W	All zeros	<a href="#">5.11.4.20/5-495</a>
0x118	FMPL_FADBTMR0—FMan Policer Flow A Debug Trap Mask Register 0	R/W	All zeros	<a href="#">5.11.4.21/5-496</a>
0x11c	FMPL_FADBTMC—FMan Policer Flow A Debug Trap Match Counter	R/W	All zeros	<a href="#">5.11.4.22/5-496</a>
0x120	FMPL_FADBTCR1—FMan Policer Flow A Debug Trap Configuration Register 1	R/W	All zeros	<a href="#">5.11.4.19/5-494</a>
0x124	FMPL_FADBVALR1—FMan Policer Flow A Debug Value Register 1	R/W	All zeros	<a href="#">5.11.4.20/5-495</a>
0x128	FMPL_FADBTMR1—FMan Policer Flow A Debug Trap Mask Register 1	R/W	All zeros	<a href="#">5.11.4.21/5-496</a>
0x12c	Reserved	—	—	—
0x130	FMPL_FBDBTCR0—FMan Policer Flow B Debug Trap Configuration Register 0	R/W	All zeros	<a href="#">5.11.4.23/5-497</a>
0x134	FMPL_FBDBVALR0—FMan Policer Flow B Debug Value Register 0	R/W	All zeros	<a href="#">5.11.4.24/5-498</a>
0x138	FMPL_FBDBTMR0—FMan Policer Flow B Debug Trap Mask Register 0	R/W	All zeros	<a href="#">5.11.4.25/5-499</a>
0x13c	FMPL_FBDBTMC—FMan Policer Flow B Debug Trap Match Counter	R/W	All zeros	<a href="#">5.11.4.26/5-499</a>
0x140	FMPL_FBDBTCR1—FMan Policer Flow B Debug Trap Configuration Register 1	R/W	All zeros	<a href="#">5.11.4.23/5-497</a>
0x144	FMPL_FBDBVALR1—FMan Policer Flow B Debug Value Register 1	R/W	All zeros	<a href="#">5.11.4.24/5-498</a>
0x148	FMPL_FBDBTMR1—FMan Policer Flow B Debug Trap Mask Register 1	R/W	All zeros	<a href="#">5.11.4.25/5-499</a>
0x14c	Reserved	R	All zeros	—

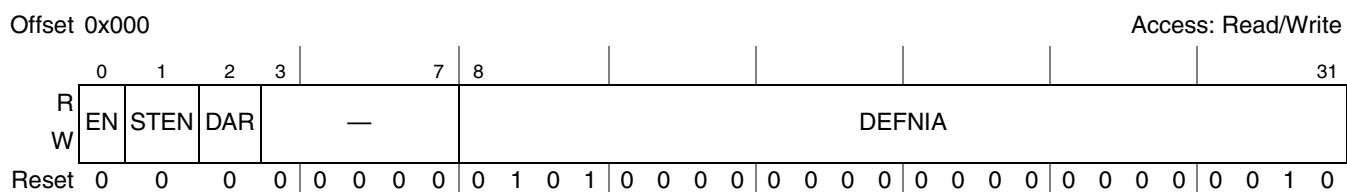
**Table 5-404. Policer Memory Map (continued)**

Register Offset	Register	Access	Reset Value	Section/Page
0x150	FMPL_FCDBTCR0—FMan Policer Flow C Debug Trap Configuration Register 0	R/W	All zeros	<a href="#">5.11.4.27/5-500</a>
0x154	FMPL_FCDBVALR0—FMan Policer Flow C Debug Value Register 0	R/W	All zeros	<a href="#">5.11.4.28/5-501</a>
0x158	FMPL_FCDBTMR0—FMan Policer Flow C Debug Trap Mask Register 0	R/W	All zeros	<a href="#">5.11.4.29/5-502</a>
0x15c	FMPL_FCDBTMC—FMan Policer Flow C Debug Trap Match Counter	R/W	All zeros	<a href="#">5.11.4.30/5-502</a>
0x160	FMPL_FCDBTCR1—FMan Policer Flow C Debug Trap Configuration Register 1	R/W	All zeros	<a href="#">5.11.4.27/5-500</a>
0x164	FMPL_FCDBVALR1—FMan Policer Flow C Debug Value Register 1	R/W	All zeros	<a href="#">5.11.4.28/5-501</a>
0x168	FMPL_FCDBTMR1—FMan Policer Flow C Debug Trap Mask Register 1	R/W	All zeros	<a href="#">5.11.4.29/5-502</a>
0x16c–0x1FF	Reserved	—	—	—
<b>Profile Selection Mapping Registers Per Port-ID</b>				
0x200	FMPL_DPMR—FMan Policer Default Mapping Register	R/W	All zeros	<a href="#">5.11.4.31/5-502</a>
0x204–0x2FF (0x200 + 4xn)	FMPL_PMR1—FMPL_PMR63, - FMan Policer Profile Mapping Registers.	R/W <sup>1</sup>	All zeros	<a href="#">5.11.4.32/5-504</a>
0x300–0xFFFF	Reserved	—	—	—

<sup>1</sup> Only implemented Rx and O/H Port-ID registers are R/W. For all other Port-ID numbers are read-only zeros.

### 5.11.4.1 FMan Policer General Configuration Register (FMPL\_GCR)

The FMPL\_GCR holds global Policer configuration and activation bits.



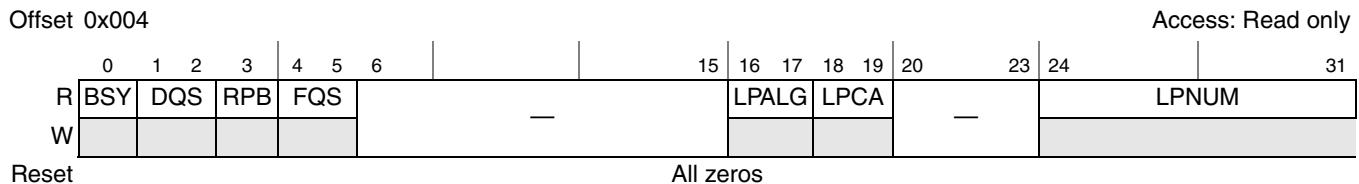
**Figure 5-353. FMan Policer General Configuration Register (FMPL\_GCR)**

**Table 5-405. FMPL\_GCR Field Descriptions**

Bits	Name	Description
0	EN	Enable Policer 0 Policer not enabled. The Policer does not accept new jobs. The status of the packets that already in progress is indicated by FMPL_GSR[BSY] status bit. 1 Policer enabled. When EN = 1, the Policer is active. It gets jobs, and processes packets. When EN is cleared, the Policer performs a graceful disable sequence. It stops accepting new packets and runs existing packets to completion. FMPL_GSR[BSY] is set, indicating that there are still some packets that are being processed. When there are no more packets in progress the FMPL_GSR[BSY] bit is cleared.
1	STEN	Policer global statistic counters enable (not for profile statistic counters) 0 Global statistic counters are not enabled. 1 Enables global statistic counters to be updated.
2	DAR	Disable Auto Refresh This bit disables the profile auto-refresh mechanism. It can be used in combination with the forced refresh bit to explicitly refresh profile timestamp instead of the built-in refresh mechanism. 0 Auto Refresh is enabled 1 Auto Refresh is disabled
3–7	—	Reserved
8–31	DEFNIA	Default Next Invoked Action This field selects the default profile entry NIA when a non-initialized profile is accessed. Out of reset the DEFNIA selects the BMI “Enqueue Frame” command (0x500002). For a detailed NIA description, see <a href="#">Section 5.4.4, “Next Invoked Action (NIA).”</a>

### 5.11.4.2 FMan Policer Global Status Register (FMPL\_GSR)

The FMPL\_GSR provides continuous status bits that reflect the Policer internal state. It can be used by software to detect if the block is busy or idle.



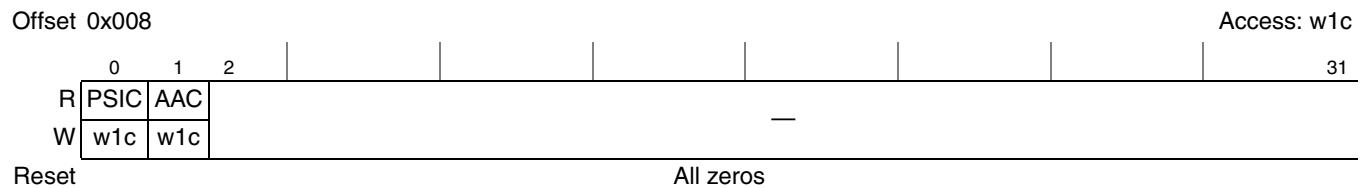
**Figure 5-354. FMan Policer Global Status Register (FMPL\_GSR)**

**Table 5-406. FMPL\_GSR Field Descriptions**

Bits	Name	Description
0	BSY	Policer Busy Indication This bit is continuously reflecting the Policer module packet processing state. When disabling the Policer (by clearing FMPL_GCR[EN]) use this bit to qualify the completion of Policer graceful stop. 0 Policer is not busy. No packets are pending in process. If Policer has been disabled by writing FMPL_GCR[EN]=0, this bit indicates that Policer graceful stop is completed. 1 Policer is busy. There are pending packets in process. If Policer has been disabled by writing FMPL_GCR[EN]=0, this bit indicates that Policer graceful stop is in progress.
1-2	DQS	Policer input Queue Status 00 Policer input queue is empty 01 Policer input queue has at least one pending packet. 10 Policer input queue is full. 11 Reserved
3	RPB	Policer Rate Processor Busy 0 Policer rate processor is idle. No packets are in RP processing stage. 1 Policer rate processor is busy. At least one packet is in RP processing stage.
4-5	FQS	Policer output Queue Status 00 Policer output queue is empty 01 Policer output queue has at least one pending packet. 10 Policer output queue is full. No more packets can be accepted. 11 Reserved
6-15	—	Reserved
16-17	LPALG	Last Profile Algorithm This field holds the last processed profile algorithm in the rate processor. It is updated at the end of each traced profile processing. 00 Pass-through 01 RFC-2698 10 RFC-4115 11 Reserved
18-19	LPCA	Last Profile Coloring Action This field holds the last processed profile coloring action of the rate processor. It is updated at the end of each traced profile processing. For color-blind modes the pre-color is considered GREEN. 00 No Change 01 GREEN to YELLOW 10 GREEN to RED 11 YELLOW to RED
20-23	—	Reserved
24-31	LPNUM	Last Profile Number This field holds the last processed profile number in the rate processor. It is updated at the end of each traced profile processing.

### 5.11.4.3 FMan Policer Event Register (FMPL\_EVR)

The FMPL\_EVR holds run time events of the Policer that are associated with the Policer interrupt. Each event bit has a corresponding enable bit in FMPL\_IER.



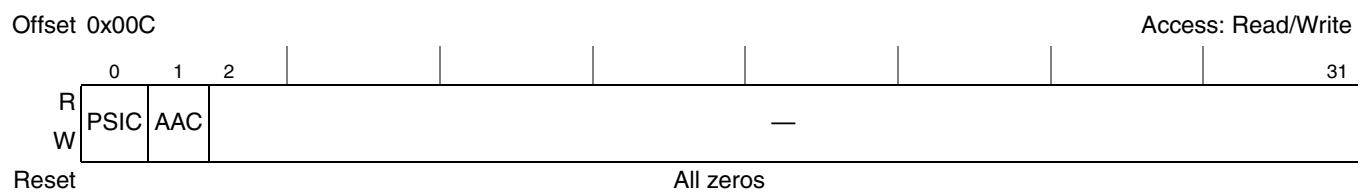
**Figure 5-355. FMan Policer Event Register (FMPL\_EVR)**

**Table 5-407. FMPL\_EVR Field Descriptions**

Bits	Name	Description
0	PSIC	PRAM Self-Initialization Complete status. Write ‘1’ to clear. Writing ‘0’ has no effect. 0 No PRAM self-initialization complete indication since last clearing of this bit. 1 PRAM self-initialization sequence is completed. All PRAM addresses at the end of these process are updated according to FMPL_PExxx registers. If FMPL_IER[PSIC] is set the Policer interrupt is asserted. Out of reset PRAM initialization is done once and PSIC is set when the initialization completes. For details, see <a href="#">Section 5.11.4.14, “FMan Policer Profile Action Register (FMPL_PAR).”</a>
1	AAC	Atomic Action Complete. Write ‘1’ to clear. Writing ‘0’ has no effect. 0 No Atomic Action is completed since last clearing of this bit. 1 Atomic Action initiated by writing ‘1’ to FMPL_PAR[GO] is completed. Another action can be started. For details, see <a href="#">Section 5.11.4.14, “FMan Policer Profile Action Register (FMPL_PAR).”</a>
2–31	—	Reserved

### 5.11.4.4 FMan Policer Interrupt Enable Register (FMPL\_IER)

The FMPL\_IER enables run time events from FMPL\_EVR towards the Policer interrupt line.



**Figure 5-356. FMan Policer Interrupt Enable Register (FMPL\_IER)**

#### **Table 5-408. FMPL\_IER Field Descriptions**

Bits	Name	Description
0	PSIC	<p>PRAM Self-Initialization Complete interrupt enable.</p> <p>0 Policer “PRAM Self-Initialization Complete” interrupt is masked. 1 Policer “PRAM Self-Initialization Complete” interrupt is enabled.</p> <p>The Policer interrupt line is asserted when FMPL_EVR[PSIC] is ‘1’.</p> <p>For details, see <a href="#">Section 5.11.4.14, “FMan Policer Profile Action Register (FMPL_PAR).”</a></p>
1	AAC	<p>Atomic Action Complete interrupt enable.</p> <p>0 Policer “Atomic Update Complete” interrupt is masked. 1 Policer “Atomic Update Complete” interrupt is enabled.</p> <p>The Policer interrupt line is asserted when FMPL_EVR[AAC] is ‘1’.</p> <p>For details, see <a href="#">Section 5.11.4.14, “FMan Policer Profile Action Register (FMPL_PAR).”</a></p>
2–31	—	Reserved

#### 5.11.4.5 FMan Policer Interrupt Force Register (FMPL\_IFR)

The FMPL\_IFR is a debug verification assistance register. Writing 1 to any valid bit in this register sets the corresponding bit in FMPL\_EVR.



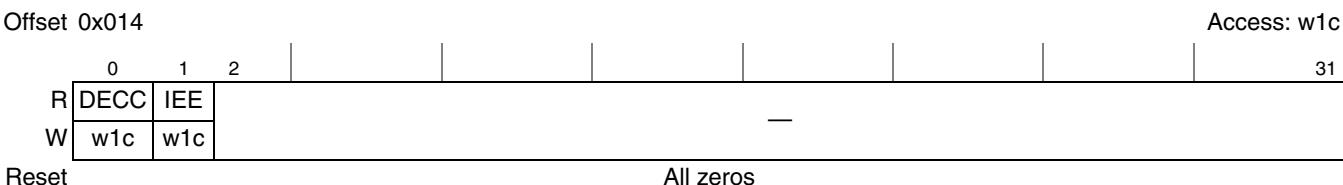
**Figure 5-357. FMan Policer Interrupt Force Register (FMPL\_IFR)**

**Table 5-409. FMPL\_IFR Field Descriptions**

Bits	Name	Description
0	PSIC	Writing 1 to sets the FMPL_EVR[PSIC] status bit. Writing 0 has no effect.
1	AAC	Writing 1 sets the FMPL_EVR[AAC] status bit. Writing 0 has no effect.
2–31	—	Reserved

#### 5.11.4.6 FMan Policer Error Event Register (FMPL\_EEVR)

The FMPL\_EEVR holds run time error events of the Policer that are associated with the Policer error interrupt. Each event bit has a corresponding mask bit in FMPL\_EIER.



**Figure 5-358. FMan Policier Error Event Register (FMPL\_EEVR)**

**Table 5-410. MPL\_EEVR Field Descriptions**

Bits	Name	Description
0	DECC	<p>Double-bit ECC error has been detected on PRAM read access. Write ‘1’ to clear. Writing ‘0’ has no effect. For details on ECC capture, see <a href="#">Section 5.11.4.16, “FMan Policer Soft Error Capture Register (FMPL_SERC).”</a></p> <p>0 No double-bit ECC error has been detected on any PRAM access since last clearing of this bit. 1 Double-bit ECC error has been detected on at least one PRAM access. If FMPL_EIER[DECC] is set the Policer error interrupt is asserted.</p>
1	IEE	<p>Initialization Entry Error. A profile entry with MODE[PI] bit cleared has been selected for packet processing. Write ‘1’ to clear. Writing ‘0’ has no effect. For details on the MODE profile configuration, see <a href="#">Section 5.11.4.33.1, “FMan Policer Profile Entry MODE Configuration Word Register (MODE).”</a> In addition to setting this bit the Policer asserts the IPP bit in the FD status field written back to the frame IC in FMan memory.</p> <p>0 No access to non-initialized profile has been detected since last clearing of this bit. 1 At least one access to non-initialized profile has been detected. This happens when the selected profile MODE[PI]=0. The self-initialization sequence writes zeros to all PRAM locations, therefore self-initialized entries trigger this event. The NIA and Port-ID of the first frame that used uninitialized profile is captured at the FMPL_UPCR register. If FMPL_EIER[IEE] is set the Policer error interrupt is asserted.</p>
2–31	—	Reserved

#### 5.11.4.7 FMan Policer Error Interrupt Enable Register (FMPL\_EIER)

The FMPL\_EIER masks run time error events from FMPL\_EEVR, towards the Policer error interrupt line.

Offset 0x018

## Access: Read/Write



**Figure 5-359. FMan Policier Error Interrupt Enable Register (FMPL\_EIER)**

**Table 5-411. FMPL\_EIER Field Descriptions**

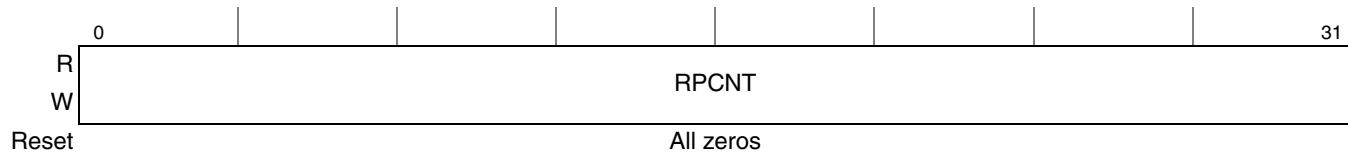
Bits	Name	Description
0	DECC	<p>Double-bit ECC Error interrupt mask.</p> <p>0 Policer “Double-bit ECC Error” interrupt is masked. In addition PRAM double ECC error detection does not trigger FM halt event.</p> <p>1 Policer “Double-bit ECC Error” interrupt is enabled. The Policer error interrupt line is asserted when FMPL_EEVR[DECC] is ‘1’. The first double-bit ECC error PRAM location is captured in the FMPL_SERC register, indicated by FMPL_SERC[CET]=1. Detection of PRAM double ECC error may trigger FM halt event.</p> <p>For details, see <a href="#">Section 5.11.4.16, “FMan Policer Soft Error Capture Register (FMPL_SERC).”</a></p>
1	IEE	<p>Initialization Entry Error interrupt mask.</p> <p>0 Policer “Initialization Entry Error” interrupt is masked.</p> <p>1 Policer “Initialization Entry Error” interrupt is enabled. The Policer error interrupt line is asserted when FMPL_EEVR[IEE] is ‘1’.</p>
2–31	—	Reserved

### 5.11.4.8 FMan Policer RED Packet Counter Register (FMPL\_RPC)

The FMPL\_RPC counts the total number of RED packets that exit the Policer.

Offset 0x020

Access: Read/Write



**Figure 5-360. FMan Policer RED Packet Counter Register (FMPL\_RPC)**

**Table 5-412. FMPL\_RPC Field Descriptions**

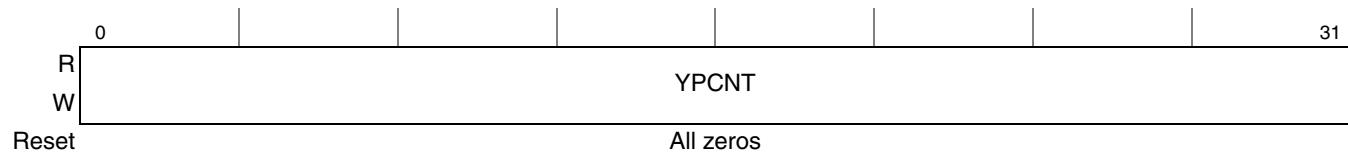
Bits	Name	Description
0–31	RPCNT	RED Packet Counter - free running counter. This is the total count of packets RED packets passed by the Policer. It includes both packets that are pre-colored RED (in color-aware mode) and packets recolored RED by the Policer. Writes to the counter set its value and next increment starts from the written value.

### 5.11.4.9 FMan Policer YELLOW Packet Counter Register (FMPL\_YPC)

The FMPL\_YPC counts the total number of YELLOW packets that exit the Policer.

Offset 0x024

Access: Read/Write



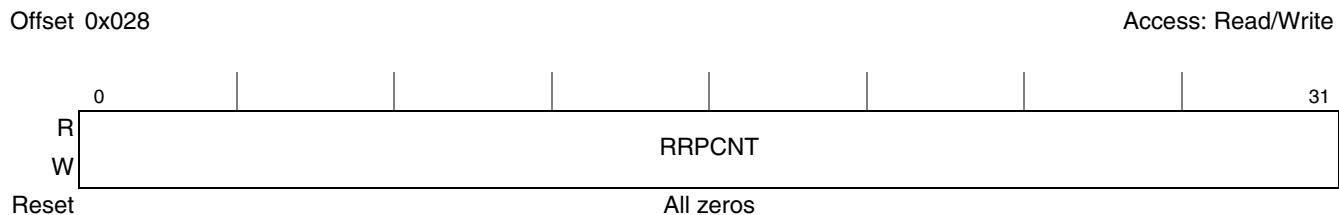
**Figure 5-361. FMan Policer YELLOW Packet Counter Register (FMPL\_YPC)**

**Table 5-413. FMPL\_YPC Field Descriptions**

Bits	Name	Description
0–31	YPCNT	YELLOW Packet Counter. This is the total count of YELLOW packets passed by the Policer. It includes both packets that are pre-colored YELLOW (in color-aware mode) and packets recolored YELLOW by the Policer. Writes to the counter set its value and next increment starts from the written value.

### 5.11.4.10 FMan Policer Recolored RED Packet Counter Register (FMPL\_RRPC)

The FMPL\_RRPC counts the number of packets that changed color to RED by the Policer. This is a subset of FMPL\_RPC packet count, indicating active color changes.



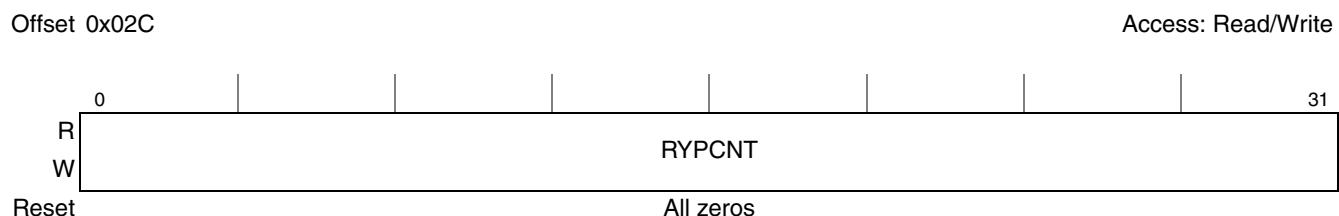
**Figure 5-362. FMan Policer Recolored RED Packet Counter Register (FMPL\_RRPC)**

**Table 5-414. FMPL\_RRPC Field Descriptions**

Bits	Name	Description
0–31	RRPCNT	Recolored RED Packet Counter. This is the total count of packets that changed color to RED by the Policer. It does not include packets that are pre-colored RED (in color-aware mode). Writes to the counter set its value and next increment starts from the written value.

### 5.11.4.11 FMan Policer Recolored YELLOW Packet Counter Register (FMPL\_RYPC)

The FMPL\_RYPC counts the number of packets that changed color to YELLOW by the Policer. This is a subset of FMPL\_YPC packet count, indicating active color changes.



**Figure 5-363. FMan Policer Recolored YELLOW Packet Counter Register (FMPL\_RYPC)**

**Table 5-415. FMPL\_RYPC Field Descriptions**

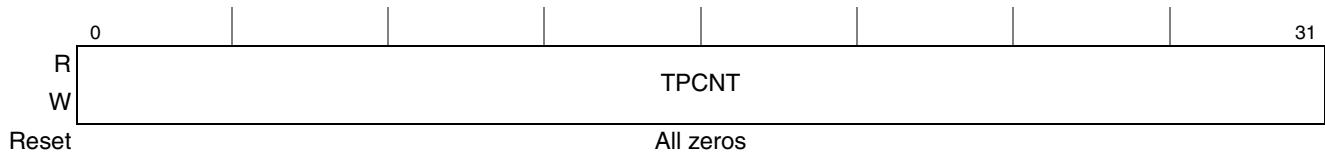
Bits	Name	Description
0–31	RYPCNT	Recolored YELLOW Packet Counter. This is the total count of packets that changed color to YELLOW by the Policer. It does not include packets that are pre-colored YELLOW (in color-aware mode). Writes to the counter set its value and next increment starts from the written value.

### 5.11.4.12 FMan Policer Total Packet Counter Register (FMPL\_TPC)

The FMPL\_TPC counts the total number of packets passed in the Policer.

Offset 0x030

Access: Read/Write



**Figure 5-364. FMan Policer Total Packet Counter Register (FMPL\_TPC)**

**Table 5-416. FMPL\_TPC Field Descriptions**

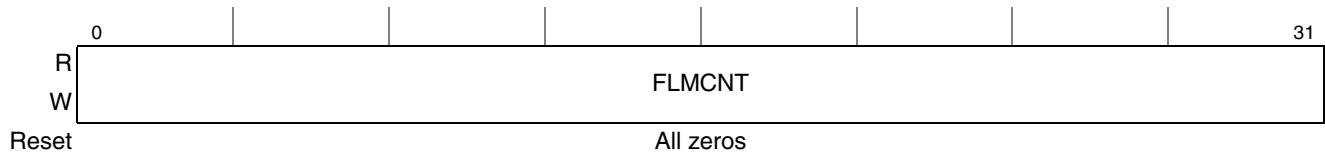
Bits	Name	Description
0–31	TPCNT	Total Packet Counter. This is the total count of packets passed in the Policer on all profiles. Writes to the counter set its value and next increment starts from the written value.

### 5.11.4.13 FMan Policer Frame Length Mismatch Counter Register (FMPL\_FLMC)

The FMPL\_FLMC counts the number of packets with length mismatch indicated by an offset value of 0xFF in the selected Parser result entry or when the calculated frame offset result is greater than the packet full length provided by FD length.

Offset 0x034

Access: Read/Write



**Figure 5-365. FMan Policer Frame Length Mismatch Counter Register (FMPL\_FLMC)**

**Table 5-417. MPL\_FLMC Field Descriptions**

Bits	Name	Description
0–31	FLMCNT	Frame Length Mismatch Count. This is the total count of mismatches in the frame length offset selection for rate calculations. Mismatch occurs when the Parser result selected by FMPL_PEMODE[FLS] was not parsed and therefore contains an offset value of 0xFF (BMI initialization default). Another error condition that gets counted is when the calculated frame length is not a positive number due to selected offset greater than the frame length. Writes to the counter set its value and next increment starts from the written value. For details on frame length selection and length mismatch detection, see <a href="#">Section 5.11.4.33.1, “FMan Policer Profile Entry MODE Configuration Word Register (MODE).”</a>

### 5.11.4.14 FMan Policer Profile Action Register (FMPL\_PAR)

The FMPL\_PAR is used for accessing a selected profile entry in an atomic operation or for activating the PRAM self initialization sequence. On specific profile access, it either initiates atomic read of the full profile entry to registers, or updates partial or full profile entry by an atomic write access. The FMPL\_PAR enables back-to-back profile accesses without software check of status bits. It also provides completion status bits of profile entry access or PRAM self initialization that can create an interrupt event for an event driven activation scheme.

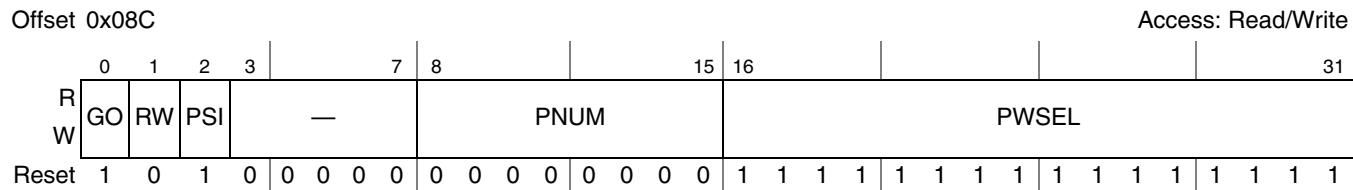


Figure 5-366. FMan Policer Profile Action Register (FMPL\_PAR)

Table 5-418. FMPL\_PAR Field Descriptions

Bits	Name	Description
0	GO	<p>Activate the atomic profile entry access or PRAM self initialization. Writing '1' when GO=0 and at the same time writing '0' to PSI starts an atomic read or atomic write access to the selected profile entry, according to the setting of RW, PNUM and PWSEL. Writing '1' when GO=0 and at the same time writing '0' to RW and '1' to PSI starts PRAM self-initialization sequence. The GO bit is set after hard reset negation to initialize the PRAM once with all zeros. It is self-clearing at the end of the atomic profile access or at the end of PRAM self-initialization. When the atomic access is complete the FMPL_EVR[AAC] status bit is set. If FMPL_IER[AAC] is set as well the Policer interrupt line is asserted.</p> <p>0 Policer profile atomic access is idle. 1 Activate Policer atomic access. The access types, profile selection and field select bits are configured RW, by PNUM and PWSEL.</p>
1	RW	<p>Read/Write access type.</p> <p>0 Profile entry atomic access is a write. At the end of the atomic access the contents of all FMPL_PE* configuration registers associated with set PWSEL mask bits are copied into the PRAM entry selected by PNUM. The write access is atomic and does not interfere with the selected profile updates due to packet processing.</p> <p>1 Profile entry atomic access is a read. At the end of the atomic access all contents of the PRAM entry selected by PNUM are copied to their associated FMPL_PE* configuration registers. The read access is atomic and does not get interference by the selected profile updates due to packet processing.</p>
2	PSI	<p>PRAM (Profile RAM) Self Initialization. This bit activates self-initialization sequence of the Policer PRAM. It can be set only if at the same write to FMPL_PAR GO is written with '1', and RW is written with '0'. During self-initialization the Policer updates all PRAM entries with the values stored in FMPL_PE* registers. For details, see <a href="#">Section 5.11.4.15, “FMan Policer Profile Entry Access Word Registers (FMPL_PE*)”</a>.</p> <p>At the end of PRAM self initialization the GO bit is self-cleared while PSI stays asserted. In addition the FMPL_EVR[PSIC] status bit is set. If FMPL_IER[PSIC] mask bit is set the Policer interrupt is asserted as well. Out of hard reset PSI and GO are both set and PRAM runs self-initialization to all zeros. At the end of the initialization GO is self cleared. If the Policer is enabled it starts processing incoming packets after PRAM initialization is complete.</p>

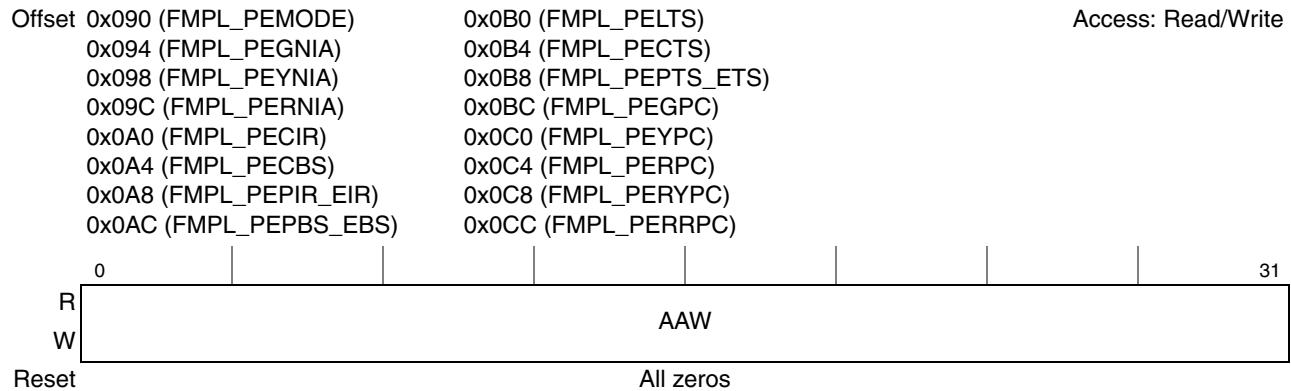
**Table 5-418. FMPL\_PAR Field Descriptions (continued)**

Bits	Name	Description																																																																				
3–7	—	Reserved																																																																				
8–15	PNUM	Profile Number. This field selects the profile number to be accessed by the atomic read/write access.																																																																				
16–31	PWSEL	<p>PWSEL[0:15] - Profile Write Select. This field provides bit-wise selection mask of the profile 32-bit entries for the atomic write access. A field associated with set PWSEL bit is written by an atomic write from its associated FMPL_PE* register. A field associated with cleared PWSEL bit is not written. Atomic read always copy all profile words to their associated FMPL_PE* registers. Following are the PWSEL bits associations to configuration registers and profile entry fields:</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>PWSEL BIT</th> <th>Associated Reg.</th> <th>Associated PRAM Entry Field</th> <th>Page</th> </tr> </thead> <tbody> <tr><td>0</td><td>FMPL_PEMODE</td><td>MODE</td><td><a href="#">5.11.4.33.1/5-506</a></td></tr> <tr><td>1</td><td>FMPL_PEGNIA</td><td>GREEN Next Invoked Action</td><td><a href="#">5.11.4.33.2/5-509</a></td></tr> <tr><td>2</td><td>FMPL_PENYNA</td><td>YELLOW Next Invoked Action</td><td><a href="#">5.11.4.33.3/5-509</a></td></tr> <tr><td>3</td><td>FMPL_PERNIA</td><td>RED Next Invoked Action</td><td><a href="#">5.11.4.33.3/5-509</a></td></tr> <tr><td>4</td><td>FMPL_PECIR</td><td>CIR - Committed Burst Rate</td><td><a href="#">5.11.4.33.4/5-510</a></td></tr> <tr><td>5</td><td>FMPL_PECBS</td><td>CBS - Committed Burst Size</td><td><a href="#">5.11.4.33.6/5-511</a></td></tr> <tr><td>6</td><td>FMPL_PEPIR_EIR</td><td>PIR_EIR - Peak/Excess Information Rate</td><td><a href="#">5.11.4.33.6/5-511</a></td></tr> <tr><td>7</td><td>FMPL_PEPBS_EBS</td><td>PBS_EBS - Peak/Excess Burst Size</td><td><a href="#">5.11.4.33.8/5-512</a></td></tr> <tr><td>8</td><td>FMPL_PELTS</td><td>LTS - Last Timestamp</td><td><a href="#">5.11.4.33.8/5-512</a></td></tr> <tr><td>9</td><td>FMPL_PECTS</td><td>CTS - Committed Token Status</td><td><a href="#">5.11.4.33.10/5-513</a></td></tr> <tr><td>10</td><td>FMPL_PEPS_ETS</td><td>PTS_ETS - Peak/Excess Token Status</td><td><a href="#">5.11.4.33.11/5-514</a></td></tr> <tr><td>11</td><td>FMPL_PEGPC</td><td>GPC - GREEN Packet Counter</td><td><a href="#">5.11.4.33.11/5-514</a></td></tr> <tr><td>12</td><td>FMPL_PEPYPC</td><td>YPC - YELLOW Packet Counter</td><td><a href="#">5.11.4.33.13/5-515</a></td></tr> <tr><td>13</td><td>FMPL_PERRPC</td><td>RPC - RED Packet Counter</td><td><a href="#">5.11.4.33.13/5-515</a></td></tr> <tr><td>14</td><td>FMPL_PERYPC</td><td>RYPC - Recolored YELLOW Packet Counter</td><td><a href="#">5.11.4.33.15/5-515</a></td></tr> <tr><td>15</td><td>FMPL_PERRPC</td><td>RRPC - Recolored RED Packet Counter</td><td><a href="#">5.11.4.33.15/5-515</a></td></tr> </tbody> </table>	PWSEL BIT	Associated Reg.	Associated PRAM Entry Field	Page	0	FMPL_PEMODE	MODE	<a href="#">5.11.4.33.1/5-506</a>	1	FMPL_PEGNIA	GREEN Next Invoked Action	<a href="#">5.11.4.33.2/5-509</a>	2	FMPL_PENYNA	YELLOW Next Invoked Action	<a href="#">5.11.4.33.3/5-509</a>	3	FMPL_PERNIA	RED Next Invoked Action	<a href="#">5.11.4.33.3/5-509</a>	4	FMPL_PECIR	CIR - Committed Burst Rate	<a href="#">5.11.4.33.4/5-510</a>	5	FMPL_PECBS	CBS - Committed Burst Size	<a href="#">5.11.4.33.6/5-511</a>	6	FMPL_PEPIR_EIR	PIR_EIR - Peak/Excess Information Rate	<a href="#">5.11.4.33.6/5-511</a>	7	FMPL_PEPBS_EBS	PBS_EBS - Peak/Excess Burst Size	<a href="#">5.11.4.33.8/5-512</a>	8	FMPL_PELTS	LTS - Last Timestamp	<a href="#">5.11.4.33.8/5-512</a>	9	FMPL_PECTS	CTS - Committed Token Status	<a href="#">5.11.4.33.10/5-513</a>	10	FMPL_PEPS_ETS	PTS_ETS - Peak/Excess Token Status	<a href="#">5.11.4.33.11/5-514</a>	11	FMPL_PEGPC	GPC - GREEN Packet Counter	<a href="#">5.11.4.33.11/5-514</a>	12	FMPL_PEPYPC	YPC - YELLOW Packet Counter	<a href="#">5.11.4.33.13/5-515</a>	13	FMPL_PERRPC	RPC - RED Packet Counter	<a href="#">5.11.4.33.13/5-515</a>	14	FMPL_PERYPC	RYPC - Recolored YELLOW Packet Counter	<a href="#">5.11.4.33.15/5-515</a>	15	FMPL_PERRPC	RRPC - Recolored RED Packet Counter	<a href="#">5.11.4.33.15/5-515</a>
PWSEL BIT	Associated Reg.	Associated PRAM Entry Field	Page																																																																			
0	FMPL_PEMODE	MODE	<a href="#">5.11.4.33.1/5-506</a>																																																																			
1	FMPL_PEGNIA	GREEN Next Invoked Action	<a href="#">5.11.4.33.2/5-509</a>																																																																			
2	FMPL_PENYNA	YELLOW Next Invoked Action	<a href="#">5.11.4.33.3/5-509</a>																																																																			
3	FMPL_PERNIA	RED Next Invoked Action	<a href="#">5.11.4.33.3/5-509</a>																																																																			
4	FMPL_PECIR	CIR - Committed Burst Rate	<a href="#">5.11.4.33.4/5-510</a>																																																																			
5	FMPL_PECBS	CBS - Committed Burst Size	<a href="#">5.11.4.33.6/5-511</a>																																																																			
6	FMPL_PEPIR_EIR	PIR_EIR - Peak/Excess Information Rate	<a href="#">5.11.4.33.6/5-511</a>																																																																			
7	FMPL_PEPBS_EBS	PBS_EBS - Peak/Excess Burst Size	<a href="#">5.11.4.33.8/5-512</a>																																																																			
8	FMPL_PELTS	LTS - Last Timestamp	<a href="#">5.11.4.33.8/5-512</a>																																																																			
9	FMPL_PECTS	CTS - Committed Token Status	<a href="#">5.11.4.33.10/5-513</a>																																																																			
10	FMPL_PEPS_ETS	PTS_ETS - Peak/Excess Token Status	<a href="#">5.11.4.33.11/5-514</a>																																																																			
11	FMPL_PEGPC	GPC - GREEN Packet Counter	<a href="#">5.11.4.33.11/5-514</a>																																																																			
12	FMPL_PEPYPC	YPC - YELLOW Packet Counter	<a href="#">5.11.4.33.13/5-515</a>																																																																			
13	FMPL_PERRPC	RPC - RED Packet Counter	<a href="#">5.11.4.33.13/5-515</a>																																																																			
14	FMPL_PERYPC	RYPC - Recolored YELLOW Packet Counter	<a href="#">5.11.4.33.15/5-515</a>																																																																			
15	FMPL_PERRPC	RRPC - Recolored RED Packet Counter	<a href="#">5.11.4.33.15/5-515</a>																																																																			

### 5.11.4.15 FMan Policer Profile Entry Access Word Registers (FMPL\_PE\*)

The FMPL\_PE\* are used by an atomic access sequence to a selected profile entry. An atomic write access does not interfere with process packet profile updates. An atomic read access provides coherent data without interference from process packet profile updates.

For details on atomic access sequence, see [Section 5.11.4.14, “FMan Policer Profile Action Register \(FMPL\\_PAR\).”](#) For details on profile entry memory map, see [Section 5.11.4.33, “Policer Profile Entry Memory Map.”](#)



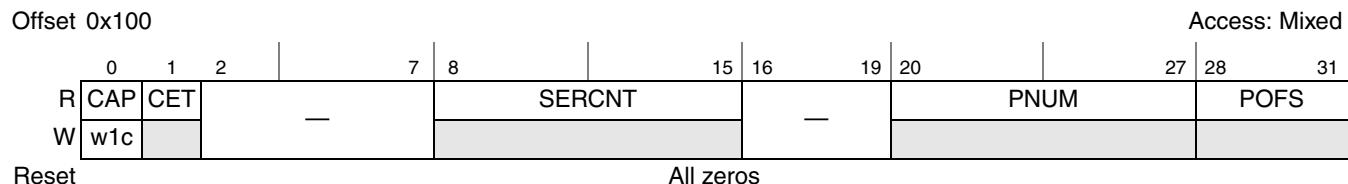
**Figure 5-367. FMan Policer Profile Entry Access Word Registers (FMPL\_PE\*)**

**Table 5-419. FMPL\_PE\* Field Descriptions**

Bits	Name	Description
0–31	AAW	<p>Atomic Access Word.</p> <p>Prior to atomic write action or PRAM self initialization sequence (by asserting FMPL_PAR[PSI]) all the used access words should be initialized with the values to be copied to their associated PRAM entries. Only words with asserted FMPL_PAR[PWSEL] bits associated with them are written to the PRAM entries. For proper profile configuration that uses rate metering it is recommended select full token buckets by setting FMPL_PECTS and FMPL_PEPTS_ETS to 0xFFFFFFFF. It is also recommended to set FMPL_PELTS to 0x00000000. This configuration ensures clean start of the profile activity.</p> <p>On atomic read action all words from the profile entry are updated from the PRAM entry regardless of PLPAR[PWSEL] value. For details on word selection, see <a href="#">Section 5.11.4.14, “FMan Policer Profile Action Register (FMPL_PAR).”</a></p> <p>For information on access word meaning at the profile, see <a href="#">Section 5.11.4.33, “Policer Profile Entry Memory Map.”</a></p>

#### 5.11.4.16 FMan Policer Soft Error Capture Register (FMPL\_SERC)

The FMPL\_SERC counts single-bit ECC errors, and captures PRAM profile and 32-bit word offset of either single-bit or double-bit ECC error. The first single-bit error detection locks the captured fields from additional single-bit error detections. The first double-bit error detection locks the captured fields from any additional error detection. Software can clear the capture lock to enable additional soft error detections.



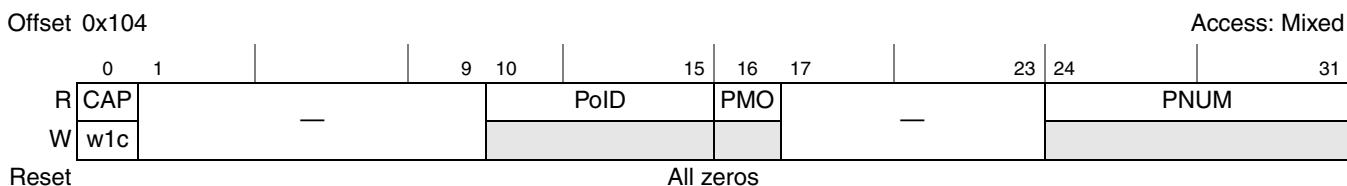
**Figure 5-368. FMan Policer Soft Error Capture Register (FMPL\_SERC)**

**Table 5-420. FMPL\_SERC Field Descriptions**

Bits	Name	Description
0	CAP	Captured Error Indication. 0 No Error Captured in PNUM and POFS fields. Any single ECC error would set the CAP bit and its profile number and internal offset would be captured and locked in the PNUM and POFS fields. 1 Single or double ECC error is captured. The PNUM and POFS fields hold the profile number and internal offset and are locked from re-capturing on another ECC error.
1	CET	Captured Error Type. 0 Single-Bit ECC Error is captured if CAP=1. No ECC error captured if CAP=0. 1 Double ECC error is captured. CET is never asserted if CAP=0. If CAP=1 and CET=1, then the FMPL_EEVR[DECC] error status bit is set. If FMPL_EIER[DECC]=1, then a double-bit ECC error also asserts the Policer error interrupt. In addition the double ECC error indication may trigger FM halt event.
2-7	—	Reserved
8-15	SERCNT	Soft Error Counter. This counter accumulates single-bit ECC errors that have been detected and corrected by the PRAM ECC logic. Double-bit ECC errors are not counted.
16-19	—	Reserved
20-27	PNUM	Profile Number Capture. This field captures the profile number on which a ECC error has been detected and corrected. When CAP=1 and CET=0 this field is locked from additional single ECC errors but could capture double-bit ECC errors. When CAP=1 and CET=1 the field is locked from capturing either single-bit or double-bit ECC errors. When CAP is cleared PNUM is cleared and ready for additional soft error captures.
28-31	POFS	Profile Offset Capture. This field captures the profile 32-bit word on which a ECC error has been detected and corrected. When CAP=1 and CET=0 this field is locked from additional single ECC errors but could capture double-bit ECC errors. When CAP=1 and CET=1 the field is locked from capturing either single-bit or double-bit ECC errors. When CAP is cleared POFS is cleared and ready for additional soft error captures.

#### 5.11.4.17 FMan Policer Uninitialized Profile Capture Register (FMPL\_UPCR)

The FMPL\_UPCR captures the Port-ID, profile number and the profile mapping option for the frame which was mapped to the uninitialized profile (PI bit of the profile MODE entry is cleared). When an invalid access occurs it sets the FMPL\_EEVR[IEE] status bit. The FMPL\_UPCR locks the captured fields of the first frame which caused this event until software clears the CAP bit. In addition the Policer sets the IPP bit in the FD status word that is updated in the frame IC.



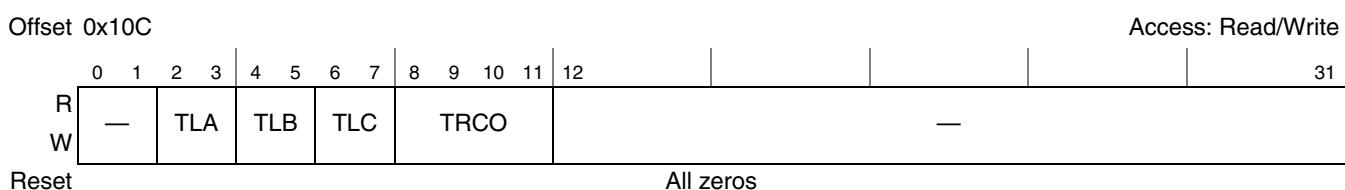
**Figure 5-369. FMan Policer Uninitialized Profile Capture Register (FMPL\_UPCR)**

**Table 5-421. FMPL\_UPCR Field Descriptions**

Bits	Name	Description
0	CAP	Captured Error Indication. This bit is set by access to uninitialized profile (profile entry MODE[PI]=0). Software clears it by writing ‘1’. While set, this bit locks FMPL_UPCR from capturing another access to uninitialized profile. 0 No Errors captured and values of PoID, PMO and PNUM fields are not valid. If any uninitialized profile is accessed the bit would be set, PoID, PMO and PNUM would capture the access attributes, and FMPL_EEVR[IEE] error status bit would be set to indicate an error event. For details, see <a href="#">Section 5.11.4.6, “FMan Policer Error Event Register (FMPL_EEVR).”</a> 1 An uninitialized profile entry access has been detected and captured. The PoID, PMO and PNUM fields hold the captured profile details.
1–9	—	Reserved
10–15	PoID	Port-ID. Reflects Port-ID of the frame which used uninitialized profile for rate calculation. If this port ID does not match an enabled FMPL_PMRn, the profile has been selected through FMPL_DPMR.
16	PMO	Profile Mapping Option. Valid only when CAP=1. This bit captures NIA[8] which indicates if the non initialized profile number (PNUM) is taken from absolute profile number (NIA[16:23]) or manipulated by the selected port-ID by the matching mapping register (FMPL_PMRn) configuration or by the default mapping register (FMPL_DPMR). For details, see <a href="#">Section 5.11.4.32, “FMan Policer Profile Mapping Registers (FMPL_PMRn),”</a> and <a href="#">Section 5.11.4.31, “FMan Policer Default Profile Mapping Register (FMPL_DPMR).”</a> 0 PNUM represents an absolute profile number, which was calculated according to PNUM and Port-ID configurations defined at the FMPL_PMRn or FMPL_DPMR. 1 PNUM represents an absolute profile number from.
17–23	—	Reserved.
24–31	PNUM	Profile Number. Valid only when CAP = 1. This field captures the non initialized profile number. If PMO=0 the appropriate FMPL_PMRn or FMPL_DPMR setting can be used to interpret which bits are taken from PoID and which are taken from PNUM. Otherwise PNUM bits capture NIA[16:23]. For details, see <a href="#">Section 5.11.4.32, “FMan Policer Profile Mapping Registers (FMPL_PMRn),”</a> and <a href="#">Section 5.11.4.31, “FMan Policer Default Profile Mapping Register (FMPL_DPMR).”</a>

### 5.11.4.18 FMan Policer Debug Trace Configuration Register (FMPL\_DTRCR)

The FMPL\_DTRCR controls the Policer debug trace level for flows A, B and C and collaboration of debug traps when not all debug flows are in use. Dumping of trace data is conditioned by a match in its associated flow, and by a non-zero trace level defined for that flow. When more than one flow has a match, the highest trace level of the matching flows is selected. For information on debug trace data per trace level, see [Section 5.11.5.3, “Debug Trace.”](#)



**Figure 5-370. FMan Policer Debug Trace Configuration Register (FMPL\_DTRCR)**

**Table 5-422. FMPL\_DTRCR Field Descriptions**

Bits	Name	Description
0–1	—	Reserved
2–3	TLA	<p>Trace Level of Flow A</p> <p>This field selects the verbosity level of trace information dumped by the Policer into the debug area of a frame's IC when flow A is enabled and flow A traps result in a match.</p> <p>00 Trace Disabled 01 Minimum trace data dumped to FMan memory. 10 More trace data dumped to FMan memory. 11 Verbose trace data dumped to FMan memory.</p>
4–5	TLB	<p>Trace Level of Flow B</p> <p>This field selects the verbosity level of trace information dumped by the Policer into the debug area of a frame's IC when flow B is enabled and flow B traps result in a match.</p> <p>00 Trace Disabled 01 Minimum trace data dumped to FMan memory. 10 More trace data dumped to FMan memory. 11 Verbose trace data dumped to FMan memory.</p>
6–7	TLC	<p>Trace Level of Flow C</p> <p>This field selects the verbosity level of trace information dumped by the Policer into the debug area of a frame's IC when flow C is enabled and flow C traps result in a match.</p> <p>00 Trace Disabled 01 Minimum trace data dumped to FMan memory. 10 More trace data dumped to FMan memory. 11 Verbose trace data dumped to FMan memory.</p>
8–11	TRCO	<p>Trap collaboration</p> <p>To provide a larger number of trap criteria for debug flow A, the traps that belong to debug flows B and/or C can be chained as a continuation of the traps for debug flow A. This lowers the number of active debug flows, but allows the criteria for debug flow A to be more extensive. The chained trap set then acts as one trap entity with a single boolean result. The debug flows (B and/or C) that donate their traps to debug flow A are put into bypass, since they would no longer have the trap resources needed to define match criteria. If debug flows B or C are not participants in the collaboration chain, then their trapping behavior will remain active and operate independently as configured. Since Policer supports two trap conditions per debug flow, a maximum of 6 trap conditions may be chained if debug flows B and C participate in a collaboration set with debug flow A.</p> <p>0000 No trap collaboration. Each flow uses its own trap registers. 0001 Flow A trap registers are chained with flow B trap registers. Debug flow B is in bypass mode and can dump debug trace data according to TRB setting. 0010 Flow A trap registers are chained with flow C trap registers. Debug flow C is in bypass mode and can dump debug trace data according to TRC setting. 0011 Flow A trap registers are chained with flow B trap registers and then with flow C trap registers. Flows B and C are in bypass mode and can dump debug trace data according to TRB and TRC setting respectively. Any other value is reserved.</p>
12–31	—	Reserved

### 5.11.4.19 FMan Policer Flow A Debug Trap Configuration Registers (FMPL\_FADBTn)

The FMPL\_FADBTn registers define the fields used for comparison, the comparison operators, and the logic operation that combines this trap result with the next one if it exists for debug flow A.

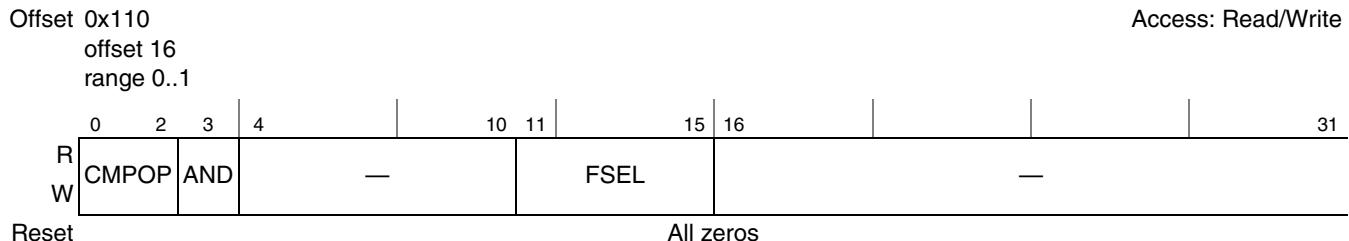


Figure 5-371. FMan Policer Flow A Debug Trap Configuration Registers (FMPL\_FADBTn)

Table 5-423. FMPL\_FADBTn Field Descriptions

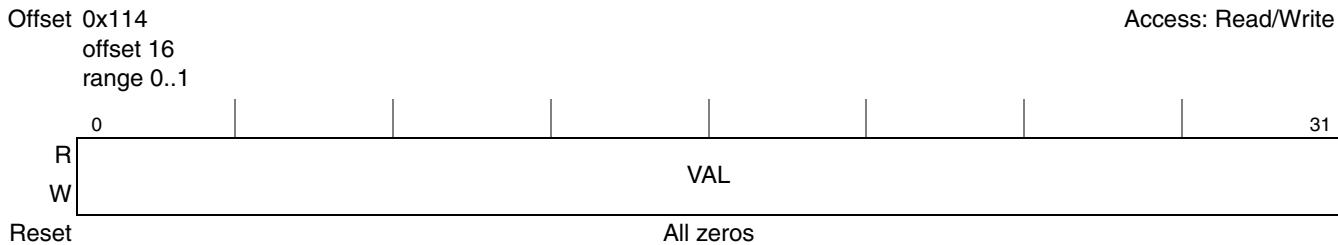
Bits	Name	Description
0-2	CMPOP	Compare operator. Mask is defined in FMPL_FADBTMRn. Value is defined in FMPL_FADBTVALn. 000 Trap disabled (never match) 001 Always match 010 Match if (selected field & mask) equals value 011 Match if (selected field & mask) does not equal value 100 Match if (selected field & mask) is greater than value 101 Match if (selected field & mask) is less or equal to value 110 Match if (selected field & mask) is less than value] 111 Match if (selected field & mask) is greater or equal to value
3	3 AND	AND This field combines individual trap results into a trap equation. By default, it is an OR operation. If all next traps are disabled (or this is the last trap), they have no effect on the combined match result. The combined match evaluation is done in ascending order from trap number n to trap number n + 1, and without precedence of AND over OR. For example: TRAP1    TRAP2 && TRAP3 results in equation equivalent to (TRAP1    TRAP2) && TRAP3. 0 OR between this trap result and the next trap result 1 AND between this trap result and the next trap result
4-10	—	Reserved

**Table 5-423. FMPL\_FADBTMR $n$  Field Descriptions (continued)**

Bits	Name	Description
11–15	FSEL	<p>Field selection</p> <p>Each encoding of FSEL identifies the data sources for the values that will be used in this trap's comparison operation, and each such field option may consist of a single value or a concatenation of multiple whole or partial values. The selected field data from its original sources is then compared to the value in FMPL_FADBTVALx combined with the mask defined in the FMPL_FADBTMRx register. Fields with fewer than 32 bits are left padded with zeros to form a 32-bit value.</p> <ul style="list-style-type: none"> <li>00000 Received NIA (24-bit value, matched against FMPL_FADBTVALx[8:31])</li> <li>00001 Received Port-ID (6-bit value, matched against FMPL_FADBTVALx[26:31])</li> <li>00010 Absolute Profile Number selected (8-bit value, matched against FMPL_FADBTVALx[24:31])</li> <li>00011 Concatenation of {Input Color (2 bits), Coloring Action Taken (2 bits), IEE (1 bit), Frame length mismatch (1 bit)} matched against FMPL_FADBTVALx[26:31])</li> <li>00100 Timestamp value at time of packet arrival to the Policer (32 bits, matched against FMPL_FADBTVALx[0:31])</li> <li>00101 CTS updated bucket status (32 bits, matched against FMPL_FADBTVALx[0:31])</li> <li>00110 PTS_ETS updated bucket status (32 bits, matched against FMPL_FADBTVALx[0:31])</li> <li>00111—11111 Reserved</li> </ul>
16–31	—	Reserved

#### 5.11.4.20 FMan Policer Flow A Debug Value Registers (FMPL\_FADBVALR $n$ )

The FMPL\_FADBVALR $n$  contain the value against which the selected field identified by FMPL\_FADBTMR $n$ [FSEL] is compared.



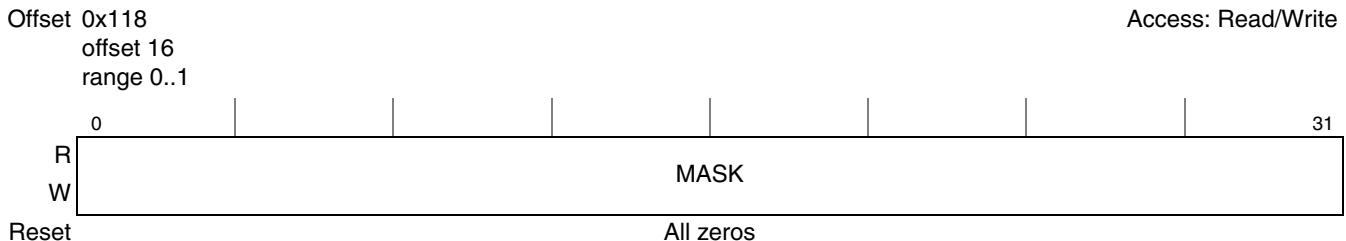
**Figure 5-372. FMan Policer Flow A Debug Value Registers (FMPL\_FADBVALR $n$ )**

**Table 5-424. FMPL\_FADBVALR $n$  Field Descriptions**

Bits	Name	Description
0–31	VAL	<p>Value</p> <p>This value is used for comparison against the selected field, indicated by FMPL_FADBTMR<math>n</math>[FSEL], using the mask specified in FMPL_FADBTMR<math>n</math>.</p>

### 5.11.4.21 FMan Policer Flow A Debug Trap Mask Registers (FMPL\_FADBTMR $n$ )

The mask value specified in the FMPL\_FADBTMR $n$  is ANDed bit-wise with the selected value for comparison, identified by FMPL\_FADBTCR $n$ [FSEL], and the result is compared to FMPL\_FADBVALR $n$ [VAL].



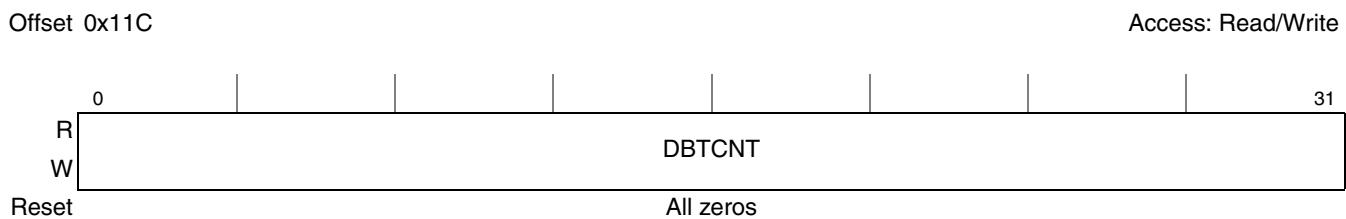
**Figure 5-373. FMan Policer Flow A Debug Trap Mask Registers (FMPL\_FADBTMR $n$ )**

**Table 5-425. FMPL\_FADBTMR $n$  Field Descriptions**

Bits	Name	Description
0–31	MASK	MASK field This field gets bit-wise ANDed with the selected field, identified by FMPL_FADBTCR $n$ [FSEL], and the result is compared to FMPL_FADBVALR $n$ [VAL]. Only the designated bits from the selected field pass the AND function.

### 5.11.4.22 FMan Policer Flow A Debug Trap Match Counter Register (FMPL\_FADBTMC)

The FMPL\_FADBTMC counts the number of match events on debug flow A traps.



**Figure 5-374. FMan Policer Flow A Debug Trap Match Counter Register (FMPL\_FADBTMC)**

**Table 5-426. FMPL\_FADBTMC Field Descriptions**

Bits	Name	Description
0–31	DBTCNT	Debug trap event match counter This is the total count of packets that matched the debug trap criteria defined by FMPL_FADBTCR $n$ , FMPL_FADBVALR $n$ , and FMPL_FADBTMR $n$ for flow A. Writes to the counter set its value, and the next increment starts from the written value. For details of debug functionality, see <a href="#">Section 5.11.5.5.1, “Packet-Based Debug Trace.”</a>

### 5.11.4.23 FMan Policer Flow B Debug Trap Configuration Registers (FMPL\_FBDBTCR $n$ )

The FMPL\_FBDBTCR $n$  define the fields used for comparison, the comparison operators, and the logic operation that combines this trap result with the next one if it exists for debug flow B.

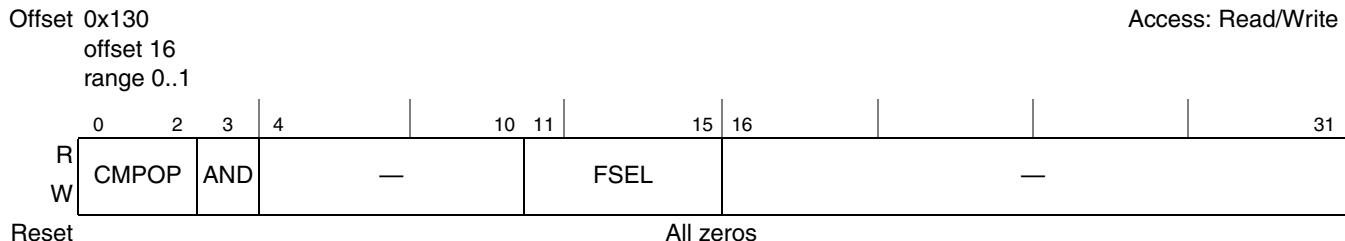


Figure 5-375. FMan Policer Flow B Debug Trap Configuration Registers (FMPL\_FBDBTCR $n$ )

Table 5-427. FMPL\_FBDBTCR $n$  Field Descriptions

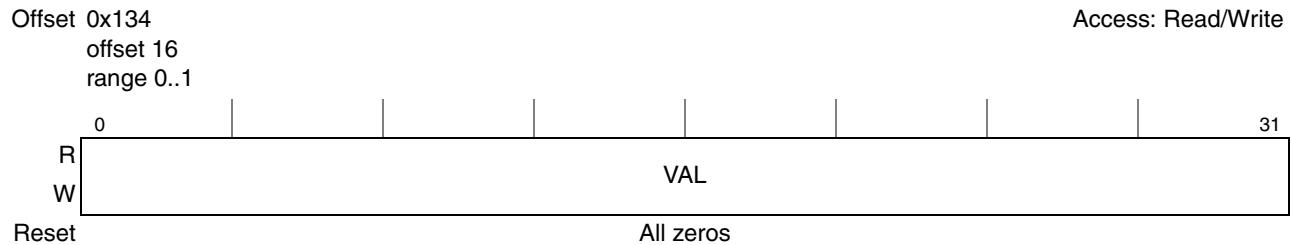
Bits	Name	Description
0–2	CMPOP	Compare operator. Mask is defined in FMPL_FBDBTMR $n$ . Value is defined in FMPL_FBDBTVAL $n$ . <ul style="list-style-type: none"> <li>000 Trap disabled (never match)</li> <li>001 Always match</li> <li>010 Match if (selected field &amp; mask) equals value</li> <li>011 Match if (selected field &amp; mask) does not equal value</li> <li>100 Match if (selected field &amp; mask) is greater than value</li> <li>101 Match if (selected field &amp; mask) is less or equal to value</li> <li>110 Match if (selected field &amp; mask) is less than value]</li> <li>111 Match if (selected field &amp; mask) is greater or equal to value</li> </ul>
3	AND	AND <p>This field combines individual trap results into a trap equation. By default, it is an OR operation. If all next traps are disabled (or this is the last trap), they have no effect on the combined match result. The combined match evaluation is done in ascending order from trap number <math>n</math> to trap number <math>n + 1</math>, and without precedence of AND over OR. For example: TRAP1    TRAP2 &amp;&amp; TRAP3 results in equation equivalent to (TRAP1    TRAP2) &amp;&amp; TRAP3.</p> <ul style="list-style-type: none"> <li>0 OR between this trap result and the next trap result</li> <li>1 AND between this trap result and the next trap result</li> </ul>
4–10	—	Reserved

**Table 5-427. FMPL\_FBDTCR*n* Field Descriptions (continued)**

Bits	Name	Description
11–15	FSEL	<p>Field selection</p> <p>Each encoding of FSEL identifies the data sources for the values that will be used in this trap's comparison operation, and each such field option may consist of a single value or a concatenation of multiple whole or partial values. The selected field data from its original sources is then compared to the value in FMPL_FBDTVALn combined with the mask defined in the FMPL_FBDTMRn register. Fields with fewer than 32 bits are left padded with zeros to form a 32-bit value.</p> <ul style="list-style-type: none"> <li>00000 Received NIA (24-bit value, matched against FMPL_FBDTVALn[8:31])</li> <li>00001 Received Port-ID (6-bit value, matched against FMPL_FBDTVALn[26:31])</li> <li>00010 Absolute Profile Number selected (8-bit value, matched against FMPL_FBDTVALn[24:31])</li> <li>00011 Concatenation of {Input Color (2 bits), Coloring Action Taken (2 bits), IEE (1 bit), Frame length mismatch (1 bit)} matched against FMPL_FBDTVALn[26:31])</li> <li>00100 Timestamp value at time of packet arrival to the Policer (32 bits, matched against FMPL_FBDTVALn[0:31])</li> <li>00101 CTS updated bucket status (32 bits, matched against FMPL_FBDTVALn[0:31])</li> <li>00110 PTS_ETS updated bucket status (32 bits, matched against FMPL_FBDTVALn[0:31])</li> <li>00111—11111 Reserved</li> </ul>
16–31	—	Reserved

#### 5.11.4.24 FMan Policier Flow B Debug Value Registers (FMPL\_FBDVALR $n$ )

The FMPL\_FBDVALR $n$  contain the value against which the selected field identified by FMPL\_FBDTCR $n$ [FSEL] is compared.



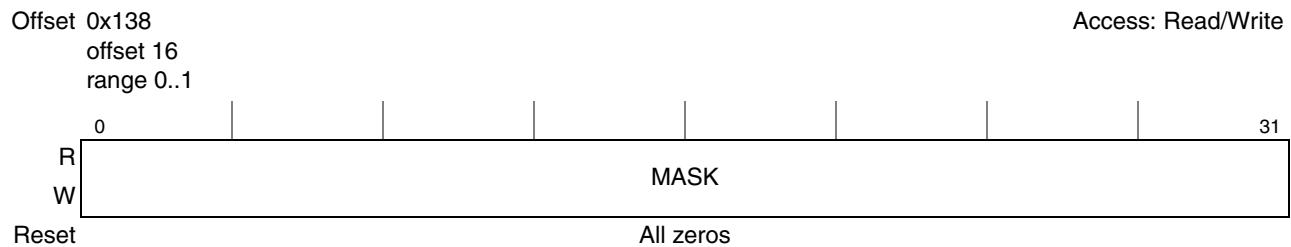
**Figure 5-376. FMan Policier Flow B Debug Value Registers (FMPL\_FBDBVALRn)**

**Table 5-428. FMPL\_FBDVALRn Field Descriptions**

Bits	Name	Description
0–31	VAL	Value. This value is used for comparison against the selected field, indicated by FMPL_FBDTCR $n$ [FSEL], using the mask specified in FMPL_FBDTMR $n$ .

### 5.11.4.25 FMan Policier Flow B Debug Trap Mask Registers (FMPL\_FBDDBTMR $n$ )

The mask value specified at the FMPL\_FBDDBTMR $n$  is ANDed bit-wise with the selected value for comparison and the result is compared to the FMPL\_FBDVALR $n$ [VAL].



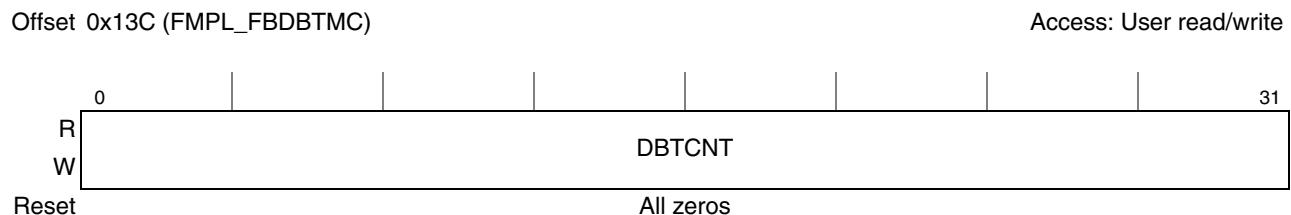
**Figure 5-377. FMan Policier Flow B Debug Trap Mask Registers (FMPL\_FBDDBTMR $n$ )**

**Table 5-429. FMPL\_FBDDBTMR $n$  Field Descriptions**

Bits	Name	Description
0–31	MASK	MASK field. This field gets bit-wise ANDed with the selected field, identified by FMPL_FBDTCR $n$ [FSEL], and the result is compared to FMPL_FBDVALR $n$ [VAL]. Only the designated bits from the selected field pass the AND function.

### 5.11.4.26 FMan Policier Flow B Debug Trap Match Counter Register (FMPL\_FBDDBTMC)

The FMPL\_FBDDBTMC counts the number of match events on debug flow B traps.



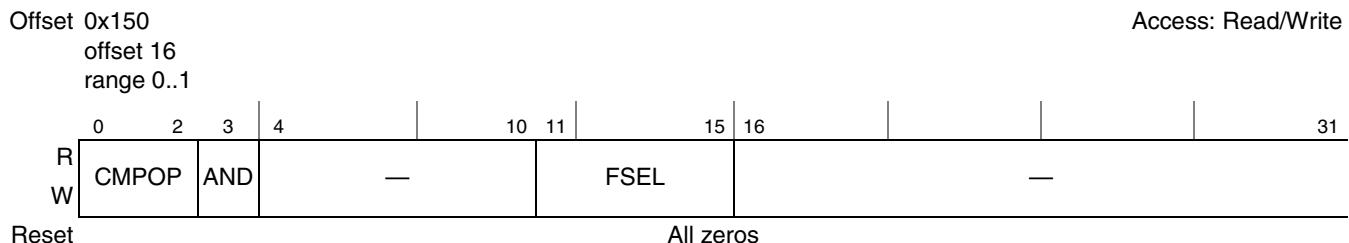
**Figure 5-378. FMan Policier Flow B Debug Trap Match Counter Register (FMPL\_FBDDBTMC)**

**Table 5-430. FMPL\_FBDDBTMC Field Descriptions**

Bits	Name	Description
0–31	DBTCNT	Debug Trap event match Counter. This is the total count of packets that matched the debug trap criteria defined by FMPL_FBDTCR $n$ , FMPL_FBDVALR $n$ , and FMPL_FBDDBTMR $n$ registers for flow B. Writes to the counter set its value and next increment starts from the written value. For details of debug functionality, see <a href="#">Section 5.11.5.5.1, “Packet-Based Debug Trace.”</a>

### 5.11.4.27 FMan Policer Flow C Debug Trap Configuration Registers (FMPL\_FCDBTCR $n$ )

The FMPL\_FCDBTCR $n$  define the fields used for comparison, the comparison operators, and the logic operation that combines this trap result with the next one if it exists for debug flow C.



**Figure 5-379. FMan Policer Flow C Debug Trap Configuration Registers (FMPL\_FCDBTCR $n$ )**

**Table 5-431. FMPL\_FCDBTCR $n$  Field Descriptions**

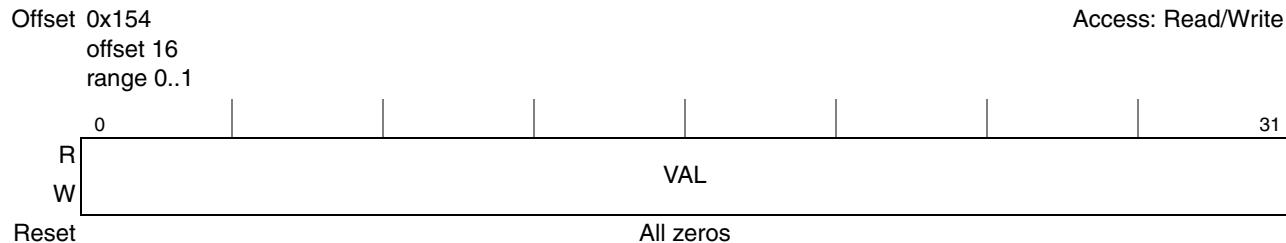
Bits	Name	Description
0-2	CMPOP	Compare operator. Mask is defined in FMPL_FCDBTMR $n$ . Value is defined in FMPL_FCDBTVAL $n$ . 000 Trap disabled (never match) 001 Always match 010 Match if (selected field & mask) equals value 011 Match if (selected field & mask) does not equal value 100 Match if (selected field & mask) is greater than value 101 Match if (selected field & mask) is less or equal to value 110 Match if (selected field & mask) is less than value 111 Match if (selected field & mask) is greater or equal to value
3	AND	AND This field combines individual trap results into a trap equation. By default, it is an OR operation. If all next traps are disabled (or this is the last trap), they have no effect on the combined match result. The combined match evaluation is done in ascending order from trap number $n$ to trap number $n + 1$ , and without precedence of AND over OR. For example: TRAP1    TRAP2 && TRAP3 results in equation equivalent to (TRAP1    TRAP2) && TRAP3. 0 OR between this trap result and the next trap result 1 AND between this trap result and the next trap result
4-10	—	Reserved

**Table 5-431. FMPL\_FCDBTCR $n$  Field Descriptions (continued)**

Bits	Name	Description
11–15	FSEL	<p>Field selection</p> <p>Each encoding of FSEL identifies the data sources for the values that will be used in this trap's comparison operation, and each such field option may consist of a single value or a concatenation of multiple whole or partial values. The selected field data from its original sources is then compared to the value in FMPL_FCDBTVAL<math>n</math> combined with the mask defined in the FMPL_FCDBTMR<math>n</math> register. Fields with fewer than 32 bits are left padded with zeros to form a 32-bit value.</p> <ul style="list-style-type: none"> <li>00000 Received NIA (24-bit value, matched against FMPL_FCDBTVAL<math>n</math>[8:31])</li> <li>00001 Received Port-ID (6-bit value, matched against FMPL_FCDBTVAL<math>n</math>[26:31])</li> <li>00010 Absolute Profile Number selected (8-bit value, matched against FMPL_FCDBTVAL<math>n</math>[24:31])</li> <li>00011 Concatenation of {Input Color (2 bits), Coloring Action Taken (2 bits), IEE (1 bit), Frame length mismatch (1 bit)} matched against FMPL_FCDBTVAL<math>n</math>[26:31])</li> <li>00100 Timestamp value at time of packet arrival to the Policer (32 bits, matched against FMPL_FCDBTVAL<math>n</math>[0:31])</li> <li>00101 CTS updated bucket status (32 bits, matched against FMPL_FCDBTVAL<math>n</math>[0:31])</li> <li>00110 PTS_ETS updated bucket status (32 bits, matched against FMPL_FCDBTVAL<math>n</math>[0:31])</li> <li>00111—11111 Reserved</li> </ul>
16–31	—	Reserved

#### 5.11.4.28 FMan Policer Flow C Debug Value Registers (FMPL\_FCDBVALR $n$ )

The FMPL\_FCDBVALR $n$  contain the value against which the selected field identified by FMPL\_FCDBTCR $n$ [FSEL] is compared.



**Figure 5-380. FMan Policer Flow C Debug Value Registers (FMPL\_FCDBVALR $n$ )**

**Table 5-432. FMPL\_FCDBVALR $n$  Field Descriptions**

Bits	Name	Description
0–31	VAL	<p>Value</p> <p>This value is used for comparison against the selected field at the FMPL_FCDBTCR<math>n</math> with mask specified at FMPL_FCDBTMR<math>n</math> applied to it.</p>

### 5.11.4.29 FMan Policier Flow C Debug Trap Mask Registers (FMPL\_FCDBTMR $n$ )

The mask value specified at the FMPL\_FCDBTMR $n$  is ANDed bit-wise with the selected value for comparison and the result is compared to the FMPL\_FCDBVALR $n$ [VAL].



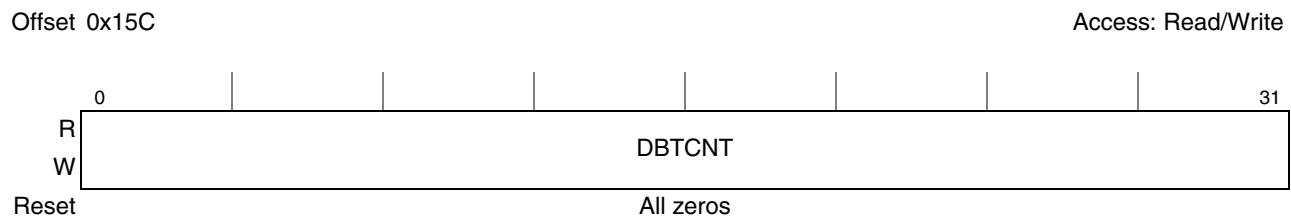
**Figure 5-381. FMan Policier Flow C Debug Trap Mask Registers (FMPL\_FCDBTMR $n$ )**

**Table 5-433. FMPL\_FCDBTMR $n$  Field Descriptions**

Bits	Name	Description
0–31	MASK	MASK field. This field gets bit-wise ANDed with the selected field, identified by FMPL_FCDBTCR $n$ [FSEL], and the result is compared to FMPL_FCDBVALR $n$ [VAL]. Only the designated bits from the selected field pass the AND function.

### 5.11.4.30 FMan Policier Flow C Debug Trap Match Counter Register (FMPL\_FCDBTMC)

The FMPL\_FCDBTMC counts the number of match events on debug flow C traps.



**Figure 5-382. FMan Policier Flow C Debug Trap Match Counter Register (FMPL\_FCDBTMC)**

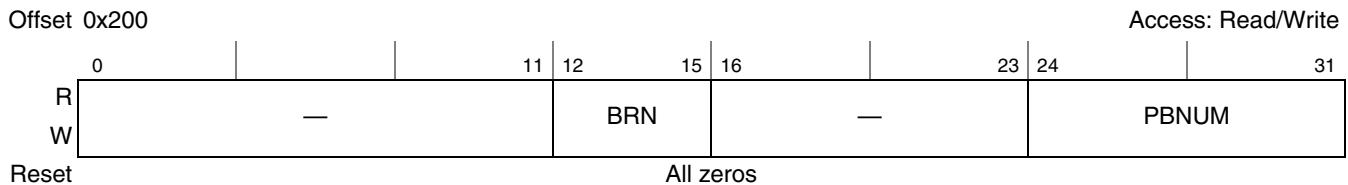
**Table 5-434. FMPL\_FCDBTMC Field Descriptions**

Bits	Name	Description
0–31	DBTCNT	Debug Trap event match Counter. This is the total count of packets that matched the debug trap criteria defined by FMPL_FCDBTCR $n$ , FMPL_FCDBVALR $n$ , and FMPL_FCDBTMR $n$ registers for flow C. Writes to the counter set its value and next increment starts from the written value. For details of debug functionality, see <a href="#">Section 5.11.5.5.1, “Packet-Based Debug Trace.”</a>

### 5.11.4.31 FMan Policier Default Profile Mapping Register (FMPL\_DPMR)

The default profile mapping register is used to manipulate profile selection passed to the Policier when the port ID is not supported or when the V bit is cleared in its associated FMPL\_PMR $n$  register. The address manipulation is activated when NIA[8] is zero, otherwise the absolute profile number is taken directly

from NIA[16:23]. The mapped profile number is calculated as a combination of NIA[16:23] and base address configuration. For NIA details, see [Section 5.4.4, “Next Invoked Action \(NIA\).”](#) For details on port mapping registers, see [Section 5.11.4.32, “FMan Policer Profile Mapping Registers \(FMPL\\_PMRn\).”](#) The relationship between the port ID and the PLPMRx register which represents it is summarized in [Table 5-436.](#)



**Figure 5-383. FMan Policer Default Profile Mapping Register (FMPL\_DPMR)**

**Table 5-435. FMPL\_DPMR Field Descriptions**

Bits	Name	Description
0–11	—	Reserved
12–15	BRN	<p>Bit Replacement Number.</p> <p>This field selects the number of most significant bits from profile selection (PNUM) that are replaced by their corresponding PNUM bits in case of no Port-ID match on any of FMPL_PMRn registers. Assuming the profile number is represented by N bits (in the Policer N=8), the following bit replacements are done.</p> <ul style="list-style-type: none"> <li>0000 Full bit replacement. In case of a match select profile number {PNUM[0:N-1]}.</li> <li>This option maps a single profile to one of <math>2^N</math> specific locations.</li> <li>0001 In case of a match select profile number {PNUM[0:N-2],PNUM[N-1]}.</li> <li>This option maps 2 profiles from into one of <math>2^{(N-1)}</math> base locations.</li> <li>0010 In case of a match select profile number {PNUM[0:N-3],PNUM[N-2:N-1]}.</li> <li>This option maps 4 profiles into one of <math>2^{(N-2)}</math> base locations.</li> <li>0011 In case of a match select profile number {PNUM[0:N-4],PNUM[N-3:N-1]}.</li> <li>This option maps 8 profiles into one of <math>2^{(N-3)}</math> base locations.</li> <li>0100 In case of a match select profile number {PNUM[0:N-5],PNUM[N-4:N-1]}.</li> <li>This option maps 16 profiles into one of <math>2^{(N-4)}</math> base locations.</li> <li>0101 In case of a match select profile number {PNUM[0:N-6],PNUM[N-5:N-1]}.</li> <li>This option maps 32 profiles into one of <math>2^{(N-5)}</math> base locations.</li> <li>0110 In case of a match select profile number {PNUM[0:N-7],PNUM[N-6:N-1]}.</li> <li>This option maps 64 profiles into one of <math>2^{(N-6)}</math> base locations.</li> <li>0111 In case of a match select profile number {PNUM[0:N-8],PNUM[N-7:N-1]}.</li> <li>This option maps 128 profiles into one of <math>2^{(N-7)}</math> base locations.</li> <li>1000 No bit replacement for N=8. In case of a match select profile number {PNUM[N-8:N-1]}.</li> <li>When N=8 this option maps all 256 profiles into one group.</li> <li>For N&gt;8 continue bit replacement in the same manner (256 profiles mapped to multiple groups)</li> <li>1001:1100 Reserved for N=9..12</li> <li>1101:1111 Reserved.</li> </ul>
16–23	—	Reserved
24–31	PNUM	<p>Profile Base Number.</p> <p>This field maps the base profile number where the group of mapped profiles is located. In case of a match the number of PNUM most significant bits selected by BRN is concatenated with the remaining least significant bits from the profile selection (PNUM). The number of profile entries in the group is always a power of two.</p>

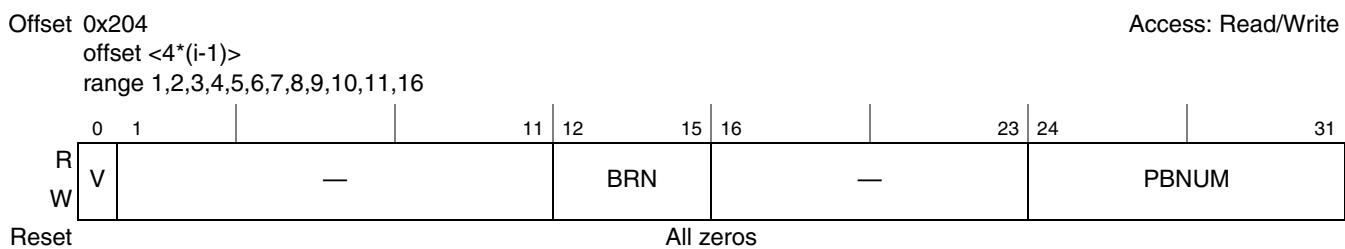
#### 5.11.4.32 FMan Policer Profile Mapping Registers (FMPL\_PMR $n$ )

The profile mapping registers can be used to manipulate profile selection passed to the Policer. When NIA[8] is zero the actual profile number is calculated as a combination of NIA[16:23] and base address configuration in the selected FMPL\_PMR $n$  register. Otherwise the absolute profile number is taken directly from NIA[16:23]. For NIA details, see [Section 5.4.4, “Next Invoked Action \(NIA\).”](#) The port ID selects one matching FMPL\_PMR $n$  to be used for profile mapping. The relationship between the port ID and the PLPMRx register that represents it is summarized in [Table 5-436](#).

**Table 5-436. Port ID to FMPL\_PMRn Mapping**

Port ID	Port Source	Mapping Register	Address Offset
0x01	Off-line Parsing/Host Command 1	FMPL_PMR1	0x204
0x02	Off-line Parsing/Host Command 2	FMPL_PMR2	0x208
0x03	Off-line Parsing/Host Command 3	FMPL_PMR3	0x20C
0x04	Off-line Parsing/Host Command 4	FMPL_PMR4	0x210
0x05	Off-line Parsing/Host Command 5	FMPL_PMR5	0x214
0x06	Off-line Parsing/Host Command 6	FMPL_PMR6	0x218
0x07	Off-line Parsing/Host Command 7	FMPL_PMR7	0x21C
0x08	1Gbit 0 Rx	FMPL_PMR8	0x220
0x09	1Gbit 1 Rx	FMPL_PMR9	0x224
0x0a	1Gbit 2 Rx	FMPL_PMR10	0x228
0x0b	1Gbit 3 Rx	FMPL_PMR11	0x22C
0x10	10Gbit 0 Rx	FMPL_PMR16	0x240
Other Values	Default port mapping	FMPL_DPMR	0x200

When NIA[8] = 0 the profile mapping is done as follows: selected number of PNUM most significant bits is replaced by a base profile number value bits from FMPL\_PMR $n$ . The remaining least significant bits are taken from the NIA least significant PNUM bits. The number of replaced bits is selected by FMPL\_PMR $n$ [BRN] and their values are taken from FMPL\_PMR $n$ [PBNUM]. This forms a continuous group of profile entries associated with the specified Port-ID: The group size is a power of two, its first profile selected by FMPL\_PMR $n$ [PBNUM] msbs, and the profile entry offset within that group is taken from PNUM lsbs. It enables virtual core separation by dividing the Policer profile entry address space between all existing port IDs plus one default port ID.



**Figure 5-384. FMan Policer Profile Mapping Registers (FMPL\_PMRn)**

**Table 5-437. FMPL\_PMRn Field Descriptions**

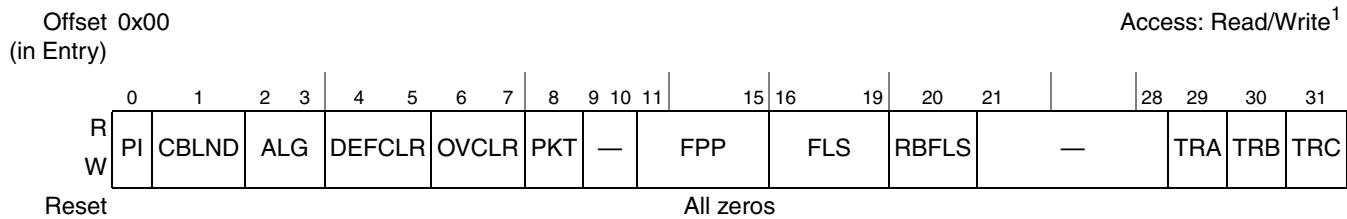
Bits	Name	Description
0	V	Port-ID Valid Bit. 0 Profile mapping register is not valid. If this register port-ID is accepted it would not match and go to the default mapping register (FMPL_DPMR). 1 Profile mapping register is valid. If the NIA Port-ID field selects this register and NIA[8]=0, NIA[16:23] will be manipulated as using the BRN and PNUM fields.
1–11	—	Reserved
12–15	BRN	Bit Replacement Number. This field selects the number of most significant bits from the profile selection (PNUM) that are replaced by their corresponding PNUM bits in case of a Port-ID match. Assuming the profile number is represented by N bits (in the Policier N=8), the following bit replacements are done. 0000 Full bit replacement. In case of a match select profile number {PNUM[0:N-1]}. This option maps a single profile to one of $2^N$ specific locations. 0001 In case of a match select profile number {PNUM[0:N-2],PNUM[N-1]}. This option maps 2 profiles into one of $2^{(N-1)}$ base locations. 0010 In case of a match select profile number {PNUM[0:N-3],PNUM[N-2:N-1]}. This option maps 4 profiles into one of $2^{(N-2)}$ base locations. 0011 In case of a match select profile number {PNUM[0:N-4],PNUM[N-3:N-1]}. This option maps 8 profiles into one of $2^{(N-3)}$ base locations. 0100 In case of a match select profile number {PNUM[0:N-5],PNUM[N-4:N-1]}. This option maps 16 profiles into one of $2^{(N-4)}$ base locations. 0101 In case of a match select profile number {PNUM[0:N-6],PNUM[N-5:N-1]}. This option maps 32 profiles into one of $2^{(N-5)}$ base locations. 0110 In case of a match select profile number {PNUM[0:N-7],PNUM[N-6:N-1]}. This option maps 64 profiles into one of $2^{(N-6)}$ base locations. 0111 In case of a match select profile number {PNUM[0:N-8],PNUM[N-7:N-1]}. This option maps 128 profiles into one of $2^{(N-7)}$ base locations. 1000 No bit replacement for N=8. In case of a match select profile number {PNUM[N-8:N-1]}. When N=8 this option maps all 256 profiles into one group. For N>8 continue bit replacement in the same manner (256 profiles mapped to multiple groups) 1001:1100 Reserved for N=9.12 1101:1111 Reserved.
16–19	—	Reserved
24–31	PNUM	Profile Base Number. This field maps the base profile number where the group of mapped profiles is located. In case of a match the number of PNUM most significant bits selected by BRN is concatenated with the remaining least significant bits from the profile selection (PNUM). The number of profile entries in the group is always a power of two.

### 5.11.4.33 Policer Profile Entry Memory Map

This section describes the internal memory organization of a single profile entry residing in the internal PRAM. Each profile entry consumes 64 bytes, and the PRAM holds 256 entries. Application access to the memory is performed by indirect mapping, using the atomic access logic. For details on atomic PRAM entry access, see [Section 5.11.4.14, “FMan Policier Profile Action Register \(FMPL\\_PAR\),”](#) and [Section 5.11.4.15, “FMan Policier Profile Entry Access Word Registers \(FMPL\\_PE\\*\).”](#)

The offsets of the profile entry fields below are given relative to the entry base location.

### 5.11.4.33.1 FMan Policer Profile Entry MODE Configuration Word Register (MODE)



**Figure 5-385. FMan Policer Profile Entry MODE Configuration Word Register (MODE)**

<sup>1</sup> Indirect by access logic

**Table 5-438. MODE Field Descriptions**

Bits	Name	Description
0	PI	<p>Profile initialization</p> <p>0 Profile is not initialized (not enabled). Any packet selecting this profile would set the FMPL_EEVR[IEE] error status bit. The ALG, CBLND fields are ignored (implying color-aware pass-through mode). The GNIA, YNIA and RNIA configuration words are ignored as well. The next module and its invoked command are selected by default NIA programmed in FMPL_GCR[DEFNIA]. When this profile is accessed, the Policer also sets the IPP bit in the FD status word that is updated to the frame IC. If FMPL_UPCR[CAP] = 0, it captures and holds the profile details and sets FMPL_UPCR[CAP]. Statistic counters are updated as in color-aware pass-through mode. For clarification, when the packet has OVERRIDE pre-color (0b11) it does not change color and counted by the GPC statistic counter similar to OVCLR configuration of 0b11.</p> <p>1 Profile is initialized. Access to this profile would not set the FMPL_EEVR[IEE] error status bit. The PKT, ALG, CBLND fields select the profile mode of operations. The GNIA, YNIA, and RNIA registers select the action of the profile based on the color result. The IPP bit in the FD status word is not changed (it may have been set by a previous non-initialized profile selection, therefore it is not cleared).</p>
1	CBLND	<p>Color blind mode</p> <p>See <a href="#">Section 5.11.5.2, “Policer Profile Operation Modes,”</a> for details.</p> <p>0 Color-aware operation of the selected algorithm</p> <p>1 Color-blind operation of the selected algorithm</p>
2-3	ALG	<p>Algorithm selection</p> <p>See <a href="#">Section 5.11.5.2, “Policer Profile Operation Modes,”</a> for details.</p> <p>00 Pass-through</p> <p>01 RFC-2698</p> <p>10 RFC-4115</p> <p>11 Reserved</p>
4-5	DEFCLR	<p>Default color</p> <p>This field only affects color blind pass-through mode. In this mode, the Policer re-colors any incoming packet with the DEFCLR value. See <a href="#">Section 5.11.5.2, “Policer Profile Operation Modes,”</a> for details.</p> <p>00 GREEN</p> <p>01 YELLOW</p> <p>10 RED</p> <p>11 Override; next invoked action is selected by GNIA and the packet is counted by the GPC statistics counter as GREEN.</p>

**Table 5-438. MODE Field Descriptions (continued)**

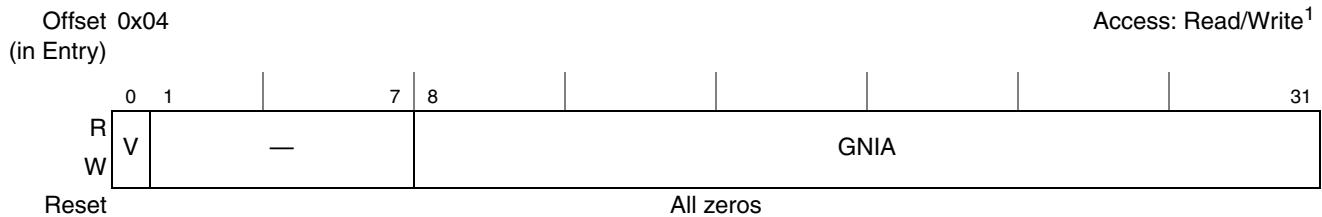
Bits	Name	Description
6-7	OVCLR	<p>Override color</p> <p>This field only affects color-aware modes. It determines the profile response to a pre-color value of 2'b11. See <a href="#">Section 5.11.5.2, “Policer Profile Operation Modes,”</a> for details.</p> <ul style="list-style-type: none"> <li>00 Pre-color value of 11 is recognized as GREEN (00). Final color written back to FMan memory.</li> <li>01 Pre-color value of 11 is recognized as YELLOW (01). Final color written back to FMan memory.</li> <li>10 Pre-color value of 11 is recognized as RED (10). Final color written back to FMan memory.</li> <li>11 Pre-color value of 11 is recognized as GREEN: Packet color is passed with no change (11), next invoked action is selected by GNIA and the packet is counted by the GPC statistics counter as GREEN. In RFC algorithms the buckets are updated as if a GREEN packet is accepted.</li> </ul>
8	PKT	<p>PACKET mode</p> <p>This field is affecting only RFC-2698 and RFC-4115 modes. See <a href="#">Section 5.11.5.2, “Policer Profile Operation Modes,”</a> for details.</p> <ul style="list-style-type: none"> <li>0 BYTE rate mode</li> <li>1 PACKET rate mode</li> </ul>
9-10	—	Reserved
11-15	FPP	<p>Fixed point position</p> <p>This field moves the fixed point position of the timestamp calculations (new timestamp, old timestamp and TDIFF). Each entry can select different view for improved accuracy at its selected rates. See <a href="#">Section 5.11.5.3.2, “Rate Measurement Implementation by Time-Stamp,”</a> for details.</p> <p>FPP is used as signed integer for moving the fixed point position relative to center position (right to bit 15). If FPP is zero (self-initialization state) the timestamp has 16 integer bits and 16 fraction bits. Positive FPP values move the fixed point to the left up to 15 positions. Negative FPP values move it to the right up to 16 positions. Following is a list of FPP accuracy selections:</p> <ul style="list-style-type: none"> <li>10000 Fixed point located right to bit 31. Timestamp is represented by 32 integer bits.</li> <li>10001 Fixed point located right to bit 30. Timestamp is represented by 31 integer bits and one fraction bit.</li> <li>...</li> <li>...</li> <li>11111 Fixed point located right to bit 16. Timestamp is represented by 17 integer bits and 15 fraction bits.</li> <li>00000 Fixed point located right to bit 15. Timestamp is represented by 16 integer bits and 16 fraction bits.</li> <li>00001 Fixed point located right to bit 14. Timestamp is represented by 15 integer bits and 17 fraction bits.</li> <li>...</li> <li>...</li> <li>01110 Fixed point located right to bit 1. Timestamp is represented by two integer bits and 30 fraction bits.</li> <li>01111 Fixed point located right to bit 0. Timestamp is represented by one integer bit and 31 fraction bits.</li> </ul>
16-19	FLS	<p>Frame length selection</p> <p>This field defines what type of frame length will be selected when the profile set to byte rate mode of operation ([PKT]=0). For packet rate mode of operation this field is ignored and length is considered to be equal to 1. If the selected packet length (L2, L3 or L4 offset) was not parsed (offset value is 0xFF) or has greater value than the FD length, the selected length will be ignored and the packet length (according to RBFL bit) is used for rate calculation. In addition the FMPL_FLMC counts this event and the FLM bit is set by the Policer in the FD status field.</p> <ul style="list-style-type: none"> <li>0011 L2 frame length is selected. Length to be used for calculation is: FD length - L2 start offset. If FCS bit is set and FCS bytes have been stripped off the frame, four FCS bytes are added to the calculated length.</li> <li>1011 L3 frame length is selected. Length to be used for calculation is: FD length - L3 start offset. If FCS bit is set and FCS bytes have not been stripped off the frame, four FCS bytes are reduced from the calculated length.</li> <li>1110 L4 frame length is selected. Length to be used for calculation is: FD length - L4 start offset. If FCS bit is set and FCS bytes have not been stripped off the frame, four FCS bytes are reduced from the calculated length.</li> <li>1111 Full frame length is selected. Length to be used for calculation is: FD length. If FCS bit is set and FCS bytes have been stripped off the frame, four FCS bytes are added to the calculated length.</li> </ul>

**Table 5-438. MODE Field Descriptions (continued)**

Bits	Name	Description
20	RBFLS	<p>Roll back frame length selection Defines what frame length is used for rate calculation when the length selected by FLS field is not valid (value of 0xFF is read as selected by FLS field or calculated frame length using the offset selected by FLS field is not a positive number).</p> <p>0 L2 frame length is selected. Length to be used for calculation is: FD length - L2 start offset. If FCS bit is set and FCS bytes have been stripped off the frame, four FCS bytes are added to the calculated length. In case L2 offset is invalid (0xFF) or greater than FD length the full frame length (FD length) is selected.</p> <p>1 Full frame length (FD length) is selected. If FCS bit is set and FCS bytes have been stripped off the frame, four FCS bytes are added to the calculated length.</p>
21	FCS	<p>Consider the packet last 4 FCS bytes in the L2, L3 and L4 packet length calculations. This bit has effect only when ALG selects RFC mode and PKT = 0.</p> <p>0 The Policer ignores FD[STATUS] bit 4, which indicates the packet includes four FCS bytes. It does not consider existence or non-existence of FCS in the packet length calculations. This is a back-compatibility mode.</p> <p>1 The Policer uses FD STATUS bit 4 to detect whether 4 FCS bytes are included in the packet. When FD status bit 4 is zero, it indicates that FCS bytes are stripped from the packet. In the case where the L2 of full packet length is selected by FLS (or by RBFLS), the 4 FCS bytes are added to the calculated length. They are not added for L3 or L4 packet length selections because FCS bytes are considered to be part of the L2 payload. When FD status bit 4 is 1, it indicates that FCS bytes are included in the packet. If this case, L2 or full packet length that already include the four FCS bytes is not changed. When L3 or L4 packet length is selected, the 4 FCS bytes are subtracted from the calculated packet length.</p>
22-28	—	Reserved
29	TRA	<p>Trap profile on flow A This bit supports user debug on this profile for flow A. When the bit is set and the profile is accessed, this condition has a match. It can be used as the third base condition following the conditions defined by the flow A trap registers: FMPL_FADBTCR<math>n</math>, FMPL_FADBVALR<math>n</math>, and FMPL_FADBTRM<math>n</math>.</p> <p>0 Profile access trap is not set. When the profile is accessed, this condition appears as false. To ignore the profile TRA bit, set AND = 0 in flow A trap configuration register FMPL_FADBTR1.</p> <p>1 Profile access trap is set. When the profile is accessed, this condition appears as true.</p>
30	TRB	<p>Trap profile on flow B This bit supports user debug on this profile for flow B. When the bit is set and the profile is accessed, this condition has a match. It can be used as the third base condition following the conditions defined by flow B trap registers: FMPL_DBTCR<math>n</math>, FMPL_FBDBVALR<math>n</math>, and FMPL_FBDBTMR<math>n</math>.</p> <p>0 Profile access trap is not set. When the profile is accessed, this condition appears as false. To ignore the profile TRA bit, set AND = 0 in flow A trap configuration register FMPL_DBTCR1.</p> <p>1 Profile access trap is set. When the profile is accessed, this condition appears as true.</p>
31	TRC	<p>Trap profile on flow C This bit supports user debug on this profile for flow C. When the bit is set and the profile is accessed, this condition has a match. It can be used as the third base condition following the conditions defined by flow B trap registers: FMPL_DBTCR<math>n</math>, FMPL_FCDBVALR<math>n</math>, and FMPL_FCDBTMR<math>n</math>.</p> <p>0 Profile access trap is not set. When the profile is accessed, this condition appears as false. To ignore the profile TRA bit, set AND = 0 in flow B trap configuration register FMPL_DBTCR1.</p> <p>1 Profile access trap is set. When the profile is accessed, this condition appears as true.</p>

### 5.11.4.33.2 GREEN Next Invoked Action Configuration Word Register (GNIA)

The GNIA selects the next module action when a GREEN result packet passes through the Policer.



**Figure 5-386. GREEN Next Invoked Action Configuration Word Register (GNIA)**

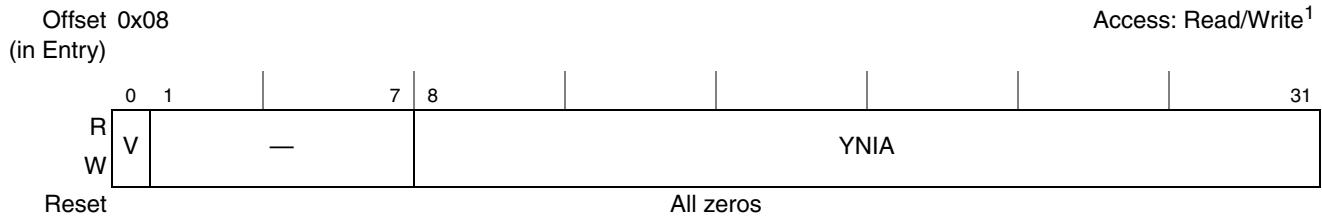
<sup>1</sup> Indirect by access logic

**Table 5-439. GNIA Field Descriptions**

Bits	Name	Description
0	V	Valid Bit. 0 On GREEN coloring result take the Default Next Invoked Action from FMPL_GCR[DEFNIA]. 1 On GREEN coloring result take the Next Invoked Action from the GNIA field.
1–7	—	Reserved
8–31	GNIA	GREEN Next Invoked Action. This field selects the Next Invoked Action if the final packet color is GREEN and V=1. If V=0 this field has no effect. The Next Invoked Action can select the Policer itself as the next module, and by that concatenate profiles entries to each other. In case of virtual port implementation by FMPL_PMRn, the next Port-ID for this purpose is the same Port-ID as the current packet. For detailed NIA description, see <a href="#">Section 5.4.4, “Next Invoked Action (NIA)”</a> .

### 5.11.4.33.3 YELLOW Next Invoked Action Address Configuration Word Register (YNIA)

The YNIA selects the next module action when a YELLOW result packet passes through the Policer. It has no effect on GREEN and RED packets.



**Figure 5-387. YELLOW Next Invoked Action Address Configuration Word Register (YNIA)**

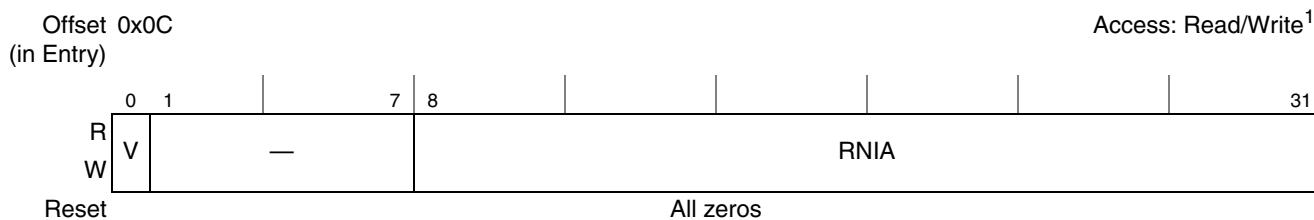
<sup>1</sup> Indirect by access logic

**Table 5-440. YNIA Field Descriptions**

Bits	Name	Description
0	V	Valid Bit. 0 On YELLOW coloring result take the Default Next Invoked Action from FMPL_GCR[DEFNIA]. 1 On YELLOW coloring result take the Default Next Invoked Action from the YNIA field.
1–7	—	Reserved
8–31	YNIA	YELLOW Next Invoked Action. This field selects the Next Invoked Action if the final packet color is YELLOW and V=1. If V=0 this field has no effect. The Next Invoked Action can select the Policer itself as the next module, and by that concatenate profiles entries to each other. In case of virtual port implementation by FMPL_PMRn, the next Port-ID for this purpose is the same Port-ID as the current packet. For detailed NIA description, see <a href="#">Section 5.4.4, “Next Invoked Action (NIA).”</a>

#### 5.11.4.33.4 RED Next Invoked Action Configuration Word Register (RNIA)

The RNIA selects the next module action when a RED result packet passes through the Policer.



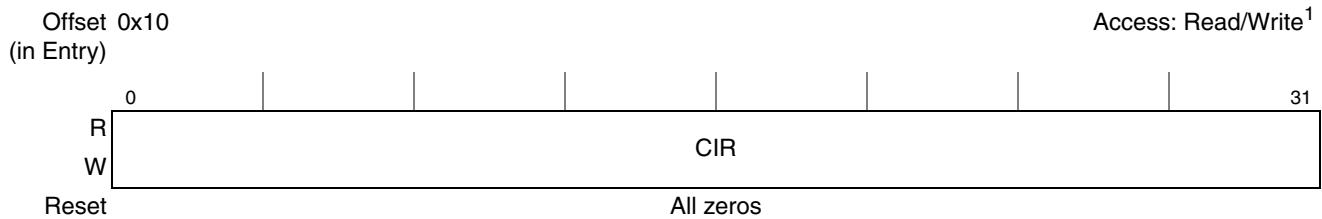
**Figure 5-388. RED Next Invoked Action Configuration Word (RNIA)**

<sup>1</sup> Indirect by access logic

**Table 5-441. RNIA Field Descriptions**

Bits	Field	Description
0	V	Valid Bit. 0 On RED coloring result take the Default Next Invoked Action from FMPL_GCR[DEFNIA]. 1 On RED coloring result take the Default Next Invoked Action from the RNIA field.
1–7	—	Reserved
8–31	RNIA	RED Next Invoked Action. This field selects the Next Invoked Action if the final packet color is RED and V=1. If V=0 this field has no effect. The Next Invoked Action can select the Policer itself as the next module, and by that concatenate profiles entries to each other. In case of virtual port implementation by FMPL_PMRn, the next Port-ID for this purpose is the same Port-ID as the current packet. For detailed NIA description, see <a href="#">Section 5.4.4, “Next Invoked Action (NIA).”</a>

#### **5.11.4.33.5 Committed Information Rate Configuration Word Register (CIR)**



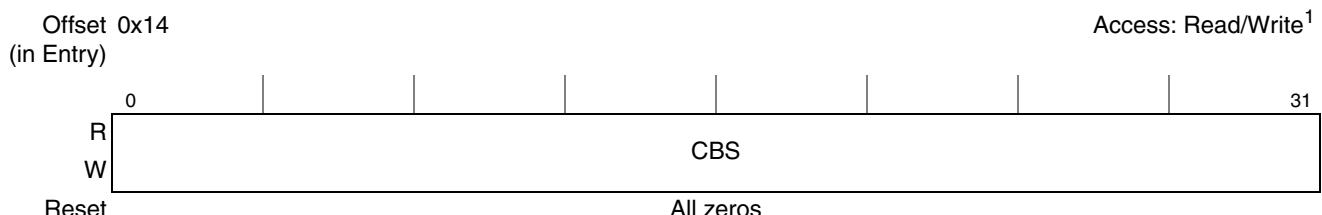
**Figure 5-389. Committed Information Rate Configuration Word (CIR)**

<sup>1</sup> Indirect by access logic

**Table 5-442. CIR Field Descriptions**

Bits	Name	Description
0–31	CIR	<p>Committed Information Rate</p> <p>This field is used only in RFC-2698 and RFC-4115 modes. It provides the committed information rate in bytes per timestamp unit or packets per timestamp unit depending on MODE[PKT] selection. The number is represented as a fixed point value, with the fixed point located right to bit 15 (bits 0-15 for integer and bits 16-31 for fraction). For details, see <a href="#">Section 5.11.5.3.3, “RFC-2698 Profile Mode Description,”</a> and <a href="#">Section 5.11.5.3.6, “RFC-4115 Profile Mode Description.”</a></p>

#### **5.11.4.33.6 Committed Burst Size Configuration Word Register (CBS)**



**Figure 5-390. Committed Burst Size Configuration Word (CBS)**

<sup>1</sup> Indirect by access logic

**Table 5-443. CBS Field Descriptions**

Bits	Name	Description
0–31	CBS	<p>Committed Burst Size</p> <p>This field is used only in RFC-2698 and RFC-4115 modes. It provides the committed burst size in bytes or packets depending on MODE[PKT] selection. CBS is a positive integer. In BYTE mode it should be set to at least the max allowed packet size. It defines the number of bytes on a full committed token bucket. The max packet length interpretation depends on MODE[FLS] programming (L2, L3, L4 or full frame), Parser offset result for the selected packet and MODE[RBFLS] programming.</p> <p>In PACKET mode it defines the number of packets in the committed token bucket. The timestamp unit meaning depends on timestamp count rate and the setting of MODE[FPP].</p> <p>For details on profile mode, see <a href="#">Section 5.11.4.33.1, “FMan Policer Profile Entry MODE Configuration Word Register (MODE)”</a>. For details on CBS, see <a href="#">Section 5.11.5.3.3, “RFC-2698 Profile Mode Description,”</a> and <a href="#">Section 5.11.5.3.6, “RFC-4115 Profile Mode Description.”</a></p>

### 5.11.4.33.7 Peak/Excess Information Rate Configuration Word Register (PIR\_EIR)



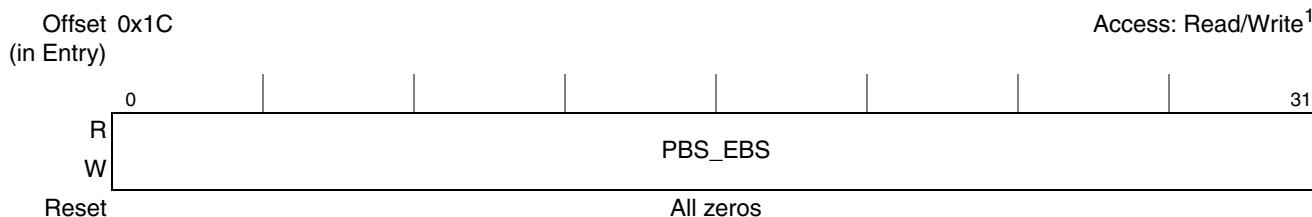
**Figure 5-391. Peak/Excess Information Rate Configuration Word (PIR\_EIR)**

<sup>1</sup> Indirect by access logic

**Table 5-444. PIR\_EIR Field Descriptions**

Bits	Name	Description
0–31	PIR_EIR	<p>Peak/Excess Information Rate</p> <p>This field is used as PIR in RFC-2698 mode or as EIR in RFC-4115 mode, and not used in pass-through mode. It provides the Peak or Excess information rate in bytes per timestamp unit or packets per timestamp unit depending on MODE[PKT] selection. The number is represented as fixed point value, with the fixed point located right to bit 15 (bits 0-15 for integer and bits 16-31 for fraction). The timestamp unit meaning depends on timestamp count rate and the setting of MODE[FPP]. For details, see <a href="#">Section 5.11.5.3.3, “RFC-2698 Profile Mode Description,”</a> and <a href="#">Section 5.11.5.3.6, “RFC-4115 Profile Mode Description.”</a></p>

#### 5.11.4.33.8 Peak/Excess Burst Size Configuration Word Register (PBS\_EBS)



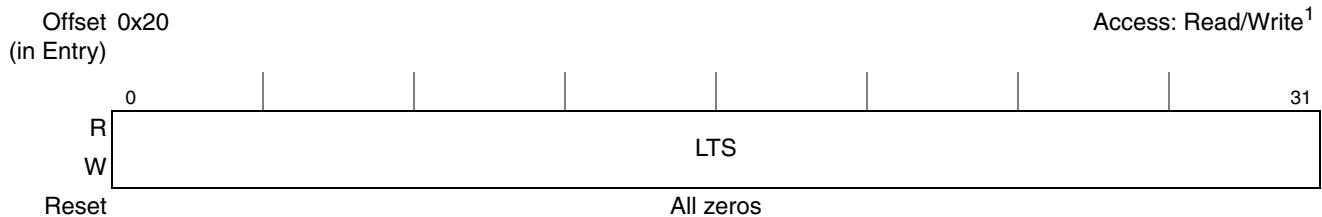
**Figure 5-392. Peak/Excess Burst Size Configuration Word Register (PBS\_EBS)**

## <sup>1</sup> Indirect by access logic

**Table 5-445. PBS\_EBS Field Descriptions**

Bits	Name	Description
0–31	PBS_EBS	<p>Peak/Excess Burst Size</p> <p>This field is used as PBS in RFC-2698 or as EBS in RFC-4115 modes, and not used in pass-through mode. It provides the peak or excess burst size in bytes or packets depending on MODE[PKT] selection. PBS_EBS is a positive integer. In BYTE mode it should be set to at least the max allowed packet size. It defines the number of bytes on a full peak/excess token bucket. The max packet length interpretation depends on MODE[FLS] programming (L2, L3, L4 or full frame), Parser offset result for the selected packet and MODE[RBFLS] programming.</p> <p>In PACKET mode it defines the number of packets in the peak/excess token bucket.</p> <p>For details on profile mode, see <a href="#">Section 5.11.4.33.1, “FMan Policer Profile Entry MODE Configuration Word Register (MODE)</a>. For details on CBS, see <a href="#">Section 5.11.5.3.3, “RFC-2698 Profile Mode Description,”</a> and <a href="#">Section 5.11.5.3.6, “RFC-4115 Profile Mode Description.”</a></p>

#### **5.11.4.33.9 Last Timestamp Variable Register (LTS)**



**Figure 5-393. Last Timestamp Variable Register (LTS)**

<sup>1</sup> Indirect by access logic

**Table 5-446. LTS Field Descriptions**

Bits	Name	Description
0–31	LTS	<p>Last Timestamp Variable</p> <p>The last timestamp variable is storing the arrival time of the previous packet or internal refresh packet. Together with the Timestamp of the current packet it is used for getting the time difference between the two packets (TDIFF) and for update of the CTS and PTS_ETS token bucket status. The LTS field is used only in RFC-2698 or RFC-4115 modes. It is updated on each incoming packet processing as well as on refresh packets. At the end of each processing the current TIMESTAMP is stored in LTS for the next packet event. LTS and TIMESTAMP values are fixed point numbers with configured fixed point location selected by MODE[FPP]. For details on MODE configuration word programming, see <a href="#">Section 5.11.4.33.1, “FMan Policier Profile Entry MODE Configuration Word Register (MODE).”</a> For details on rate measurement and timestamp usage, see <a href="#">Section 5.11.5.3.2, “Rate Measurement Implementation by Time-Stamp.”</a></p>

#### **5.11.4.33.10 Committed Token-Bucket Status Variable Register (CTS)**



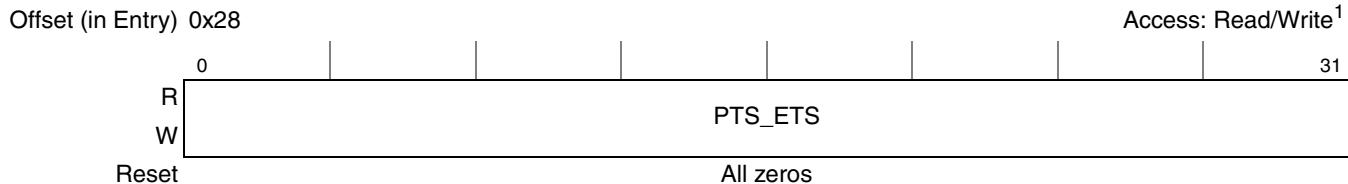
**Figure 5-394. Committed Token-Bucket Status Variable Register (CTS)**

<sup>1</sup> Indirect by access logic

**Table 5-447. CTS Field Descriptions**

Bits	Name	Description
0–31	CTS	<p>Committed Token-Bucket Status Variable.</p> <p>This variable represents the updated number of tokens in the committed token bucket. It is used only in RFC-2698 or RFC-4115 modes. CTS is updated on each incoming packet processing as well as on refresh packets. The CTS value is represented as fixed point number with automatic location of the fixed point to achieve max accuracy. The representation will always have integer bits that exactly cover the integer value of CBS, and the rest of the bits represent the fraction. For example, if CBS=1024 then 10 CTS bits represent the integer and 22 CTS bits represent the fraction.</p> <p>At first initialization and on any profile configuration change it is required to initialize CTS to the same value or greater of CBS (that is, most significant integer bits of CTS are the same as CBS and the rest of the fraction bits are all zeros). For convenience the user can initiate it to 0xFFFFFFFF and the first profile update will automatically update bucket to its full value.</p>

### 5.11.4.33.11 Peak/Excess Token-Bucket Status Variable Register (PTS\_ETS)



**Figure 5-395. Peak/Excess Token-Bucket Status Variable Register (PTS\_ETS)**

<sup>1</sup> Indirect by access logic

**Table 5-448. PTS\_ETS Field Descriptions**

Bits	Name	Description
0–31	PTS_ETS	<p>Peak/Excess Token-Bucket Status Variable.</p> <p>This variable represents the updated number of token in the peak or excess token bucket. It is used only in RFC-2698 as peak token bucket or in RFC-4115 as excess token bucket. PTS_ETS is updated on each incoming packet processing as well as on refresh packets. The PTS_ETS value is represented as fixed point number with automatic location of the fixed point to achieve max accuracy. The representation will always have integer bits that exactly cover the integer value of PBS_EBS, and the rest of the bits represent the fraction. For example, if PBS_EBS=1024 then 10 PTS_ETS bits represent the integer and 22 PTS_ETS bits represent the fraction.</p> <p>At first initialization and on any profile configuration change, it is required that PTS_ETS be initialized to be greater than or equal to PBS_EBS (that is, the most significant integer bits of PTS_ETS are greater than or equal to CBS and the rest of the fraction bits are all zeros). For convenience, the user can initiate it to 0xFFFFFFFF and the first profile update will automatically update the bucket to its full value.</p>

### 5.11.4.33.12 GREEN Packet Counter Variable Register (GPC)



**Figure 5-396. GREEN Packet Counter Variable Register (GPC)**

<sup>1</sup> Indirect by access logic

**Table 5-449. GPC Field Descriptions**

Bits	Name	Description
0–31	GPC	<p>GREEN Packet Counter Variable</p> <p>This is an integer counter that accumulates the total number of GREEN result packets sent by the current Policer profile. The counter is updated on every incoming packet. It is not affected by internal refresh packets.</p>

### 5.11.4.33.13 YELLOW Packet Counter Variable Register (YPC)



**Figure 5-397. YELLOW Packet Counter Variable Register (YPC)**

<sup>1</sup> Indirect by access logic

**Table 5-450. YPC Field Descriptions**

Bits	Name	Description
0–31	YPC	YELLOW Packet Counter Variable. The counter is updated on every incoming packet. It is not affected by internal refresh packets.

### 5.11.4.33.14 RED Packet Counter Variable Register (RPC)



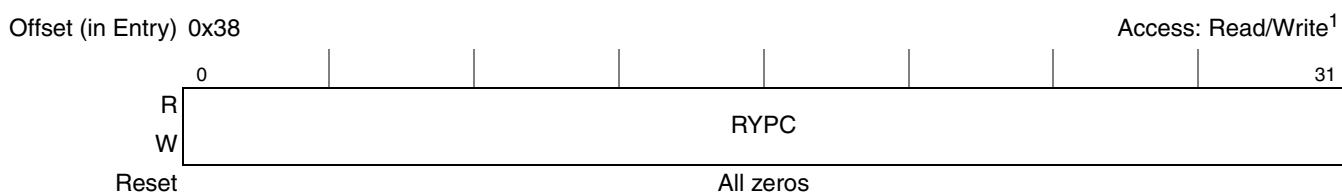
**Figure 5-398. RED Packet Counter Variable Register (RPC)**

<sup>1</sup> Indirect by access logic

**Table 5-451. RPC Field Descriptions**

Bits	Name	Description
0–31	RPC	RED Packet Counter Variable This is an integer counter that accumulates the total number of RED result packets sent by the current Policer profile. This number includes both packets that changed color from GREEN/YELLOW and packets that kept their original RED color. The counter is updated on every incoming packet. It is not affected by internal refresh packets.

### 5.11.4.33.15 Recolored YELLOW Packet Counter Variable Register (RYPC)



**Figure 5-399. Recolored YELLOW Packet Counter Variable Register (RYPC)**

<sup>1</sup> Indirect by access logic

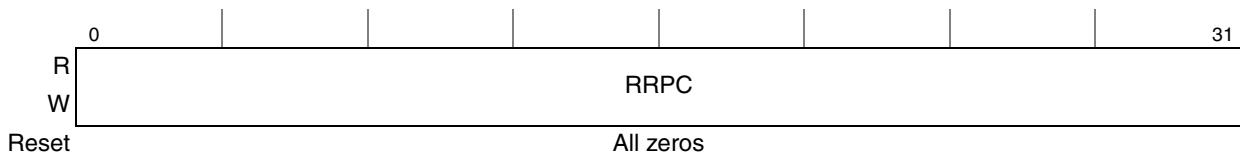
**Table 5-452. RYPC Field Descriptions**

Bits	Name	Description
0–31	RYPC	Recolored YELLOW Packet Counter Variable This is an integer counter that accumulates the total number of GREEN packets that are recolored to YELLOW by the current Policer profile. In color-blind mode all YELLOW packets are considered to be recolored. The counter is updated on every incoming packet. It is not affected by internal refresh packets.

#### 5.11.4.33.16 Recolored RED Packet Counter Variable Register (RRPC)

Offset (in Entry) 0x3C

Access: Read/Write<sup>1</sup>



**Figure 5-400. Recolored RED Packet Counter Variable Register (RRPC)**

<sup>1</sup> Indirect by access logic

**Table 5-453. RRPC Field Descriptions**

Bits	Name	Description
0–31	RRPC	Recolored RED Packet Counter Variable This is an integer counter that accumulates the total number of GREEN or YELLOW packets that are recolored to RED by the current Policer profile. In color-blind mode all RED packets are considered to be recolored. The counter is updated on every incoming packet. It is not affected by internal refresh packets.

### 5.11.5 Policer Functional Description

#### 5.11.5.1 Policer Flow Description

This section provides a description of the Policer flows from incoming packet to end results.

##### 5.11.5.1.1 Receiving an Incoming Packet

Incoming packets are assigned to the Policer though the NIA of the previous processing module.

The following incoming packet data is used by the Policer:

- TNUM—Task number unique to each incoming packet
- Debug level—Enables different debug levels per frame. For a definition of Policer operations in each debug level, see [Section 5.11.5.5.1, “Packet-Based Debug Trace.”](#)
- Port-ID—Distinguishes the flow per MAC or other FMan virtual channel. Can be used as part of the profile selection mapping
- NIA—the 16 least significant bits (lsbs) select the Policer profile (either directly or together with the Port-ID).

- ICP—Internal context pointer, which is a 24-bit FMan memory address that points to the frame IC. It is used as a base to read required fields from the frame IC.
- Debug offset—Points to the next 4-byte aligned offset where debug data can be appended. When the Policer works in one of its debug modes, it appends its own debug data and updates the debug offset depending on the debug level. The debug offset increment granularity is 4 bytes.
- Frame internal offset—Used by the Policer to compensate packet length calculations when frame data starts on a non-zero offset. The Parser header offset results are always referring to the beginning of the data buffer while FD frame length is not affected by the internal offset. Therefore, packet length calculation is increased by this value. If the internal offset value is greater than the Parser offset, the Policer treats it as frame length mismatch and triggers frame length roll-back to L2 offset, or, in case it is also greater than L2 offset, the full length is selected.

The 8 lsbs of the NIA of the incoming packet select the Policer profile. This selection can be modified according to the Port-ID such that different ports are mapped to separate groups of profile numbers while the lower NIA bits select internal offset within that group.

The Policer uses the FD length and the Parser offset results to calculate the effective frame length in bytes for RFC-2698 or RFC-4115 operation (packet length is ignored in PACKET mode). According to the selected profile layer, the Policer may take FD length or subtract from it the Parser L2 offset, L3 offset or L4 offset result to extract the metered data unit.

If the configured offset is not parsed (the selected layer offset value is 0xFF initiated by the BMI and not overridden by the Parser) or not valid (selected offset is greater than the FD length value, resulting in negative length calculation result), the full frame length or the L2 offset is used instead of the selected layer offset. The selection of which length to roll-back is done per profile by MODE[RBFLS] configuration.

When it is set to roll back to the L2 offset, the Policer validates that L2 offset is parsed and valid. In case the L2 offset is either not parsed (value of 0xFF) or not valid (greater than the FD length) the FD LENGTH field is used for the rate calculation. Any length mismatch indicated above that results in different packet length calculation, the configured selection is counted by the FMPL\_FLMC. In addition, the Policer sets the FLM bit in the FD status. When the MODE[FCS] bit is set, the Policer uses FD STATUS bit 4 to detect the existence or non-existence of 4 FCS bytes at the end of the packet and fixes the length calculation accordingly. In this mode, L2 or full packet length includes the FCS bytes (add four bytes if STATUS bit 4 is cleared), and L3 or L4 packet length does not include the FCS bytes (subtract four bytes STATUS bit 4 is set).

For details on the configuration options, see [Section 5.11.4.33.1, “FMan Policier Profile Entry MODE Configuration Word Register \(MODE\).”](#) For details on frame length error event counting, see [Section 5.11.4.13, “FMan Policier Frame Length Mismatch Counter Register \(FMPL\\_FLMC\).”](#)

The incoming packet holds two-bit pre-color information that is used in color-aware operation and ignored in color-blind modes. It is also paired with the current TIMESTAMP value when working in RFC-2698 or RFC-4115 mode.

In addition to the incoming packet attributes, the Policier fetches its configuration and run-time variables from the selected profile PRAM entry.

### **5.11.5.1.2 Avoiding Calculation Errors in Policer Profile Using Refresh Packets**

The refresh packet mechanism eliminates wraps on the Policer profile's last time stamp variable (LTS). The TIMESTAMP value from the FMan FPM holds the 32-bit integer part of the central time base used by all the Policer profiles. If a certain profile is not accessed by regular packet mapping for a long time, there could be a full wrap in the difference between the current TIMESTAMP value and the Policer profile entry last time stamp (LTS) variable, resulting in major calculation errors.

The refresh packet updates the profile token bucket status (CTS and PTS\_ETS) and saves an updated version of the TIMESTAMP value in LTS. All profiles are scanned at low rate and refreshed multiple times over the full TIMESTAMP wrap period.

Refresh packets are internal to the Policer and do not involve interaction with other FMan modules. They only update the profile buckets status and the LTS value, but do get traffic metering and do not update statistic counters.

### **5.11.5.1.3 Atomic Profile Update**

The Policer provides the user an atomic profile access logic that can read or write the profile context in PRAM under live traffic conditions. Note that the atomic profile access logic can only be accessed by one software entity at a time. If Host commands are used to access the profile, then software should not use the atomic profile access logic as the FMan controller uses this mechanism to execute the Host Commands.

For coherent profile write, the full profile entry or selected words of the profile into the Policer profile entry access word registers (FMPL\_PE\*). Then, the Policer profile action register (FMPL\_PAR) should be written to initiate atomic write access to the selected profile entry words.

#### **NOTE**

It is the user's responsibility to initialize full token buckets in FMPL\_PECTS and FMPL\_PEPTS\_ETS (that is, their integer portion should be equal to CBS and PBS\_EBS respectively).

Note that FMPL\_PECTS and FMPL\_PEPTS\_ETS have automatic fixed point locations such that the number of integer bits of FMPL\_PECTS exactly covers the CBS value, and the number of integer bits of FMPL\_PEPTS\_ETS exactly covers the PBS\_EBS value. For convenience, FMPL\_PECTS and FMPL\_PEPTS\_ETS can be initialized to 0xFFFF\_FFFF to indicate full status. They automatically get updated to adjusted full bucket value when the profile is first selected by an incoming packet or by a refresh packet.

### **5.11.5.2 Policer Profile Operation Modes**

Each of the Policer profiles is fully configured in the internal PRAM and can work independently of the others. The profile can work in one of the following modes:

- Pass-through
- RFC-2698—A two-rate three-color marker
- RFC-4115—A differentiated service two-rate three-color marker with efficient handling of in-profile traffic

Each mode can be internally configured to operate as color-aware or color-blind, and can separately configure the next invoked action (NIA) for any of the color results (such as BMI commands to drop or keep the packet with specified color using the GNIA, YNIA, or RNIA profile entry configuration words).

The most common usage is selection between dropping or keeping RED colored packets using the RNIA configuration. Each profile entry can configure measurement units to be BYTE or PACKET. A non-initialized profile is filled with all zeros after hard reset. This set-up implicitly selects pass-through, color-aware mode, and the NIA is selected by FMPL\_MCR[DEFNIA]. Packets with OVERRIDE pre-color (0b11) have per-profile programmable treatment. They may be considered as per-color GREEN, YELLOW, or RED, or alternatively, they may keep their override color though the Policer profile. In the latter case, these packets get counted as GREEN packets by the profile MODE[GPC] statistics counter.

On standard applications, the used PRAM entries are initialized by software before the Policer is enabled. This can be indicated by setting MODE[PI] to 1 in the initialization value. When a profile is accessed with MODE[PI] = 0, it is treated as an invalid access to a non-initialized profile. This event sets FMPL\_EEVR[IEE] status bit and the Policer sets the IPP bit in the FD status word in the updated frame IC. The FMPL\_UPCR captures and locks the details of the first access. The lock is released after software clears FMPL\_UPCR[CAP].

### 5.11.5.2.1 Pass-Through Modes

When a Policer profile is configured to work in pass-through mode, no traffic measurements are performed. Depending on the setting of configuration fields, the next packet color could be either propagated from the incoming packet or overwritten by the profile default color. The GNIA, YNIA, and RNIA settings select the invoked action for any of the packet color results.

In pass-through mode, there is no usage of the traffic parameters configured in the CIR, CBS, PIR\_EIR, PBS\_EBS, CTS, PTS\_ETS, and LTS fields. Statistic counters are updated according to the specific setting.

### 5.11.5.2.2 Color-Aware Pass-Through Mode

The profile is transparent to all packet colors. It is the recommended initialization of unused profile entries and can be used as a place-holder for later assignment of the profile entry to a different algorithm.

In color-aware pass-through mode, all incoming packets are passed with no color change to the next processing module. According to the packet pre-color, GNIA, YNIA, or RNIA is used to select the next action. The GPC, YPC, and RPC statistics counters count packets with GREEN, YELLOW, or RED output colors, respectively. The RYPC and RRPC counters do not count because no color change is done by the Policer.

If packet pre-color is OVERRIDE (11), then it is interpreted according to MODE[OVCLR] field. It can be treated as GREEN, YELLOW, or RED pre-color and passed with that color result, or propagated as with no change as OVERRIDE color. For details, see [Section 5.11.4.33.1, “FMan Policer Profile Entry MODE Configuration Word Register \(MODE\).”](#)

### **5.11.5.2.3 Color-Blind Pass-Through Mode**

All incoming packets are passed to the next processing module. The next color is forced to be the value programmed in the profile default color field MODE[DEFCLR]. The next processing module would be as specified in the GNIA, YNIA, and RNIA according to the MODE[DEFCLR] value.

Only one of the GPC, YPC and RPC statistic counters is advancing according to MODE[DEFCLR] setting. The RYPC and RRPC statistics counters do not advance since pre-coloring is ignored. If MODE[DEFCLR] = 11 packets are counted by GPC.

This mode can be used to override color for a full stream of packets, or temporarily drop a full stream of packets.

### **5.11.5.3 Traffic Metering and Marking Modes**

The following sections describe the principles of metering incoming traffic rates and how they are used in the Policer implementation of RFC-2698 and RFC-4115 algorithms.

#### **5.11.5.3.1 Token Bucket Metering Principles**

The implementation of differentiated service algorithms uses token buckets for traffic metering. This section describes the general concept of token bucket metering. The metering methods defined in RFC-2698 and RFC-4115 use similar data structures with some differences on the metering decisions. The next sections describe the Policer implementation of the two algorithms and their specific configuration setup.

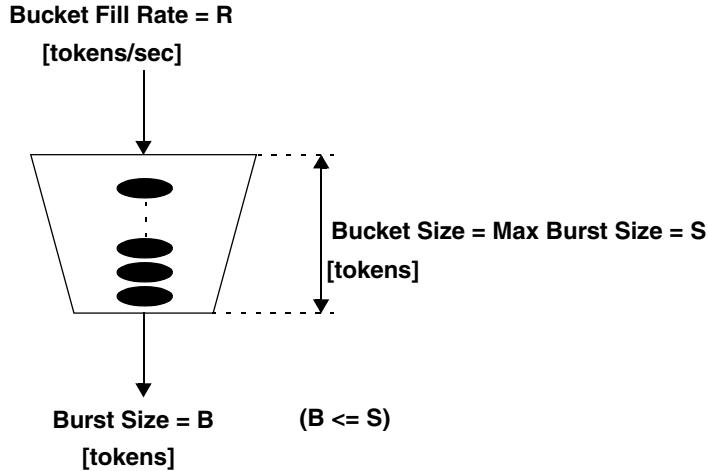
A token bucket is a mechanism to measure and enforce traffic rates. It uses tokens as the basic measurement unit. Bucket size and incoming packet burst size are defined in units of tokens, while traffic rate is defined in units of tokens per second.

A bucket with size S can hold up to S tokens and is referred to as max burst size. The max burst size is the maximum size of the packet network data in units of tokens, and excluding link layer headers and overhead (the Policer offers different options of packet network data size calculation, configured per profile). An incoming packet with size B can hold up to S tokens of network data. The bucket size S should be configured to be greater or equal to the longest packet network data, such that a full token bucket can always hold at least one full burst.

Tokens are added to the bucket at rate of R tokens per second, up to the bucket size S. Therefore one token is added to the bucket every  $1/R$  seconds as long as the bucket is not full.

When a packet with size B ( $B \leq S$ ) arrives in the bucket it passes unchanged (if the bucket holds at least B tokens) and B tokens are removed from the bucket. If the bucket holds fewer than B tokens the packet is identified as exceeding its defined stream rate and no tokens are removed from the bucket. In such a case

the packet could be either dropped or marked (colored), depending on the required response to this event. Figure 5-401 illustrates a token bucket traffic measurement system.



**Figure 5-401. Token Bucket Concept**

A two-rate metering algorithm implements two token buckets working in parallel; each bucket is configured to meter traffic with different characteristics. Each measurement system is characterized by its traffic parameters: Traffic Rate  $R$  [tokens/sec], and Max Burst Size  $S$  [tokens]. Parameter  $S$  is also defining the token bucket size such that any burst exceeding  $S$  tokens is identified as non-conforming. The unit of the token could vary on different algorithms (it could be bits, bytes, or full packets, depending on the measurement goals). The Policer implementation supports BYTE and PACKET unit modes.

### 5.11.5.3.2 Rate Measurement Implementation by Time-Stamp

The Policer implementation does not increase all token buckets' size every  $1/R$  seconds. Instead it attaches a timestamp value for each arriving packet and checks the time gap between current arrival and the previous packet arrival that is stored in the profile context at run-time (LTS—last timestamp). Assuming the delta time is TDIFF, then the number of tokens to be added to the bucket is  $\text{TDIFF} \cdot R$ , limited by the max burst size (which also defines the bucket size).

To prevent accumulated errors, the rate parameters and calculations use a fixed point representation such that time representation is divided to integer and fraction portions.

TDIFF accuracy depends on the timestamp count rate. All Policer actions are timed by timestamp units which are defined by timestamp count rate and the interpretation of it by the profile fixed point position. For example, the timestamp could be used for measuring microseconds with integer and fraction bits, such that the programmed numbers left to the fixed point position represent Mbyte/sec (in BYTE mode) or Mpacket/sec (in PACKET mode). The fixed point position that divides the timestamp counter to integer and fraction (per profile) can be moved to account for very high traffic rates (in ranges of GByte/sec) or very low traffic rates (in ranges of few packets per second). Moving the fixed point to the right increases the timestamp unit resolution of the integer bits and improves metering accuracy at high traffic rates. Moving the fixed point to the left increases the timestamp unit resolution of the fraction bits and improves metering accuracy at low traffic rates. For example, the timestamp value could be set to 16-bit integer and

16-bit fraction and its count pre-scaling configured (in the FPM) to count integer LSB (bit 15) at rate of 1 MHz, providing rate measurements of Mbyte/sec or Mpacket/sec.

The FPM timestamp prescale logic allows precision count to convert different clock frequencies to absolute time measurements as required by the application.

#### 5.11.5.3.3 RFC-2698 Profile Mode Description

The RFC-2698 two-rate three-color marker algorithm is used for implementing of differentiated services by mapping different traffic types to different profiles. The algorithm meters an IP packet stream and marks its packets as either GREEN, YELLOW or RED.

A packet is marked RED if it exceeds the peak information rate (PIR); otherwise it is marked YELLOW if it exceeds its committed information rate (CIR); otherwise it is marked GREEN. The algorithm is used to enforce peak rate needs on ingress traffic. The Policer can drop RED packets by setting the profile RNIA configuration word to BMI “Discard Frame” command, or pass the RED packets like other colored packets by setting the profile RNIA configuration to BMI “Enqueue Frame” command. In practice there is no limitation from configuring any command in GNIA, YNIA and RNIA (for example, drop YELLOW packets or forward YELLOW packets to another Policer profile where they could be aggregated and recolored RED).

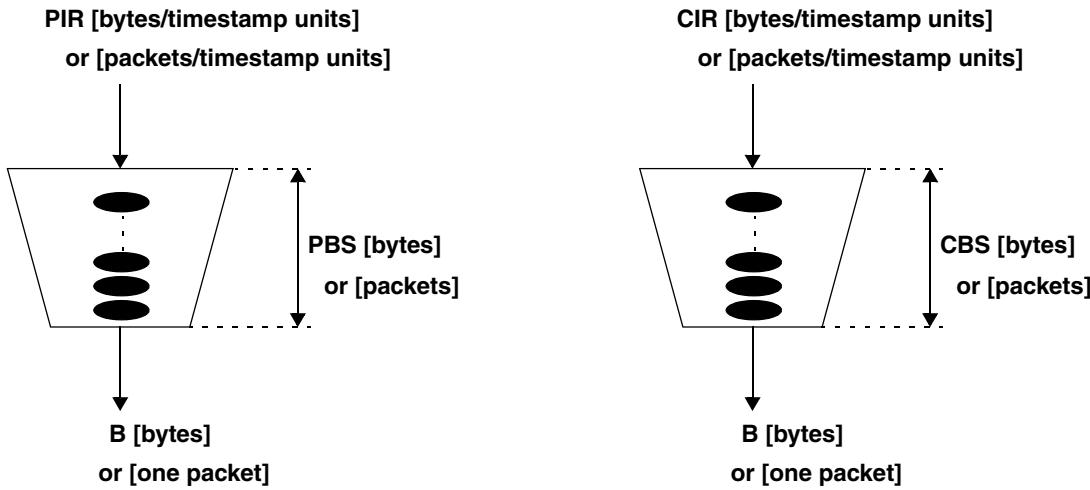
Metering can operate in one of two modes: color-blind and color-aware. In color-blind mode the metering algorithm assumes that the packet stream is uncolored. In color-aware mode it assumes that a preceding entity has pre-colored the incoming packet stream so that the packet is either GREEN, YELLOW or RED. Based on Meter result, a following Marker may recolor the IP packet.

The RFC-2698 algorithm is an extension of RFC-2697 that meters only one rate. Its definition duplicates the token buckets to measure two rates in parallel: CIR and PIR.

#### 5.11.5.3.4 RFC-2698 Profile Mode Configuration

- Select algorithm MODE[ALG] = ‘01’.
- Select color-blind/color-aware operation (MODE[CBLND]).
  - In color-aware mode, select override color interpretation (MODE[OVCLR]).
- Select Measurement Unit Mode to BYTE or PACKET (MODE[PKT]).
  - In BYTE mode, select Parser offset to be used for packet length calculation (MODE[FLS] and MODE[RBFLS]). Select how the Policer considers FCS bytes by MODE[FCS].
- Select timestamp floating point position (MODE[FPP]) to fit the actual traffic rates. For PACKET mode with low traffic rates, move the fixed point to the left to increase fraction accuracy. For BYTE mode with high traffic rates, move the fixed point to the right to increase integer accuracy. Best accuracy is achieved when the PIR configuration representation occupies the most significant bit.
- Configure traffic parameters.
  - Set peak information rate (PIR\_EIR used as PIR).
  - Set peak burst size (PBS\_EBS used as PBS).
  - Set committed information rate (CIR).
  - Set committed burst size (CBS).

- Initialize the Metering Buckets to be full (write them with 0xFFFF\_FFFF). The Policer truncates PTS and CTS automatically to fit PBS and CBS respectively. It also automates the fixed point position of each token bucket status to maximize calculation accuracy by assigning integer bits that cover PBS or CBS programming and assigns the remaining LSBs to be in the fraction part.
  - Peak rate token bucket size (PTS\_ETS used as PTS).
  - Committed rate token bucket size (CTS).



**Figure 5-402. RFC-2968 Token Buckets Implementation**

PIR and CIR are measured in bytes per timestamp units or packets per timestamp units depending on the measurement unit mode. The timestamp is distributed by the FPM and treated by the Policer as a 32-bit fixed point number. Timestamp unit is defined as an increment rate of its integer part. By default (MODE[FPP] = 0) the timestamp value representation is a 16-bit integer and a 16-bit fraction. However, the integer part resolution can be increased to support high traffic rates by moving the fixed point position to the right. Alternatively the fraction bits resolution can be increased to support low traffic rates by moving the fixed point position to the left. If, for example, a 16-bit integer and 16-bit fraction partition are used and the timestamp integer part is configured by the FPM prescale logic to count at 1-MHz rate, then the PIR and CIR are measured in Mbytes/sec or Mpacket/sec, subject to the unit configuration. For a given rate its configuration representation is doubled for each right movement of the fixed point position. The best measurement accuracy is achieved when the max rate representation (PIR) occupies the most significant configuration bit, or the fixed point position moved all the way to the right, whichever comes first.

In BYTE unit mode (the native RFC-2698 definition) PIR and CIR account for configured packet size selection using the MODE[FLS] programming. Originally RFC-2698 defines packet size to be IP header and payload, excluding link layer (L2) headers. Using MODE[FLS], the Policer can extract from the frame IC Parser results with offset of L2, L3 or L4 headers. It then selects either full frame or subtracts from the frame length the selected header offset to get the desired packet size. When MODE[FCS] is set the Policer uses FD STATUS bit 4 to detect existence of FCS bytes and add them to L2 or full frame length (in case they were missing) or subtract them from L3 or L4 length (in case they exist).

In PACKET mode each incoming packet is counted as one token. PIR and CIR are the number of packets per timestamp units. PBS and CBS are the bucket sizes and can be set to one or more packets.

The Policer configuration represents PIR and CIR as 32-bit fixed point numbers, divided to 16-bit integer and 16-bit fraction. PIR must be equal or greater than CIR.

The PBS and CBS are positive integers measured in bytes (BYTE mode) or packets (PACKET mode). In BYTE mode they should be set to be equal or greater than the size of the largest possible IP packet in the stream. In PACKET mode they should be set to integer number of one or more packets. The P and C full bucket sizes are equal to PBS and CBS respectively.

#### 5.11.5.3.5 RFC-2698 Profile Mode Metering and Marking

Number of token on buckets P and C (PTS and CTS respectively) are run-time variables updated on each arrival of packet belonging to the profile and measured in bytes (BYTE mode) or packets (PACKET mode). The buckets are initialized to be full, that is, PTS(0) = PBS and CTS(0) = CBS.

The timestamp value is sampled whenever a new packet mapped to the profile arrives, and saved in profile entry TS field after completion of the packet processing. Therefore when the new packet arrives the difference between the sampled timestamp value and the previously saved LTS value (TDIFF) is the number of timestamp units between the two packet arrivals.

According to the algorithm PTS and CTS bucket sizes increment by one at rate of PIR and CIR, up to the size of PBS and CBS respectively. The same result is achieved by calculation at the packet arrival time: adding to the TDIFF\*PIR to PTS and TDIFF\*CIR to CTS and truncating the results to PBS and CBS respectively if the results exceed the bucket sizes. PTS and CTS are represented by fixed-point numbers to prevent accumulated computation errors (number of tokens is updated upon packet arrival and must accumulate a non-integer timestamp value multiplied by non-integer rate value). The integer bits of PTS are automatically set to cover PBS value and the rest of the bits are used as fraction. Similarly the integer bits of CTS are automatically set to cover CBS value and the rest of the bits are used as fraction. This setting maximizes the accuracy of accumulated tokens as long as the bucket fills up.

When a new packet with size B arrives to the system, PTS and CTS are first updated as described above to reflect the most updated bucket status, and then the profile metering is applied on the packet.

In color-blind mode, the following metering is applied:

- If  $B > PTS$  the packet is marked RED
- Else if  $B > CTS$  the packet is marked YELLOW and decrement PTS by B.
- Else the packet is marked GREEN and decrement both PTS and CTS by B.

In color-aware mode, the following metering is applied:

- If B has been pre-colored RED or if  $B > PTS$  the packet is marked RED
- Else if B has been pre-colored YELLOW or if  $B > CTS$  the packet is marked YELLOW and decrement PTS by B.
- Else the packet is marked GREEN and decrement both PTS and CTS by B.

The difference between color-blind and color-aware modes is the honoring of pre-color information. In the color-aware mode, a packet would never be degraded from RED to YELLOW/GREEN or from YELLOW to GREEN.

If the packet has been colored RED and the profile RNIA is configured to BMI discard packet command, the BMI would drop the packet. Otherwise, if RNIA is configured to BMI enqueue frame command, the BMI would copy the frame IC from FMan memory to external memory and forward enqueue command to the QMI. The profile programming allows other usage scenarios by providing GNIA, YNIA and RNIA for GREEN, YELLOW or RED packets respectively. Each NIA may also select the Policer as next module and concatenate Policer profiles per color to achieve complex policing schemes.

#### 5.11.5.3.6 RFC-4115 Profile Mode Description

RFC-4115 is a modified version of RFC-2698 with a slightly different definition of traffic parameters, metering and marking methods. It has been defined to improve in-profile service and therefore called, “A Differentiated Service Two-Rate Three-Color Marker with Efficient Handling of in-Profile Traffic”.

The main difference between RFC-4115 and RFC-2698 is the number of conformance tests that a packet has to pass before it is colored GREEN. In RFC-2698 it has to pass two conformance tests while in RFC-4115 it only need one conformance test. Therefore traffic that is inside the profile limits (CIR) could potentially have a performance hit for a CPU-based implementation. However, the Policer implementation provides the same processing time for both algorithms by parallel execution of the two conformance tests. Therefore, the differences between the two algorithms are minor and selection depends on the application requirements.

The RFC-4115 implementation uses the same data structures as RFC-2698, with the exception that Peak Information Rate (PIR) and Peak Burst Size (PBS) are replaced by Excess Information Rate (EIR) and Excess Burst Size (EBS) respectively.

#### 5.11.5.3.7 RFC-4115 Profile Mode Configuration

- Select algorithm MODE[ALG] = ‘10’.
- Select color-blind/color-aware operation (MODE[CBLND]).
  - In color-aware mode select override color interpretation (MODE[OVCLR]).
- Select Measurement Unit Mode to BYTE or PACKET (MODE[PKT]).
  - In BYTE mode select Parser offset to be used for packet length calculation (MODE[FLS] and MODE[RBFLS]). Select how the Policer considers FCS bytes by MODE[FCS].
- Select timestamp floating point position (MODE[FPP]) to fit the actual traffic rates. For PACKET mode with low traffic rates move the fixed point to the left to increase fraction accuracy. For BYTE mode with high traffic rates move the fixed point to the right to increase integer accuracy.
- Configure traffic parameters.
  - Set committed information rate (CIR).
  - Set committed burst size (CBS),
  - Set excess information rate (EIR).
  - Set excess burst size (EBS).

- Initialize the Metering Buckets to be full (write them with 0xFFFFFFFF. The Policer truncates CTS and ETS automatically to fit CBS and EBS respectively. It also automates the fixed point position of each token bucket status to maximize calculation accuracy by assigning integer bits that covers the CBS or EBS programming and assigning the remaining LS bits to be in the fraction part).
  - Committed Rate Token Bucket Size (CTS)
  - Excess Rate Token Bucket Size (ETS)

EIR and CIR are measured in bits-per-second in RFC-4115, but measured as bytes-per-timestamp units or packets-per-timestamp units in the Policer implementation (depending on the measurement unit mode). To get convenient conversion from bytes per timestamp units to bytes/second, the timestamp integer part can be set to count at a rate of 1 MHz. This would achieve measurement at units of 1 Mbytes/sec, which is equivalent to 8 Mbits/sec.

The EIR and CIR are represented by 32-bit fixed point numbers with a 16-bit integer and a 16-bit fractional portion. The timestamp value is also represented as 32-bit fixed point number where the fixed point position can be configured to adapt the measurement accuracy for very high or very low traffic rates. For a given rate, its configuration representation is doubled for each right movement of the timestamp fixed point position. The best measurement accuracy is achieved when the max rate representation (either EIR or CIR) occupies the most significant configuration bit, or the fixed point position moved all the way to the right, whichever comes first.

The CIR and EIR can be set independently of each other. Alternatively, the CIR and EIR can be linked together by defining a burst duration parameter, T, where  $T = CBS/CIR = EBS/EIR$ .

The CBS and EBS are positive numbers measured in bytes (BYTE mode) or packets (PACKET mode). In BYTE mode, they should be set to a value greater than the expected maximum length of an incoming PDU (Protocol Data Unit). In PACKET mode, they should be set to the integer value of one or more packets. The C and E full bucket sizes are equal to CBS and EBS, respectively.

The PDU interpretation is configured using the MODE[FLS] programming. In most cases, it is similar to RFC-2698 definition (IP header and payload, excluding link layer headers and overhead). Using MODE[FLS], the Policer can extract from the frame IC parse results the offsets of L2, L3 or L4 headers. It then selects either full frame length or subtract from the frame length the selected header offset to get the desired packet size. When MODE[FCS] is set, the Policer uses FD STATUS bit 4 to detect the existence of FCS bytes and add them to L2 or full frame length (in case they were missing) or subtract them from L3 or L4 length (in case they exist).

### **5.11.5.3.8 RFC-4115 Profile Mode Metering and Marking**

The number of tokens on buckets C and E (CTS and ETS, respectively) are run-time variables updated on each arrival of packets belonging to the profile and measured in bytes (BYTE mode) or packets (PACKET mode). The buckets are initialized to be full, that is,  $CTS(0) = CBS$  and  $ETS(0) = EBS$ .

The timestamp value is sampled whenever a new packet mapped to the profile arrives, and saved in the profile entry TS field after completion of the packet processing. Therefore, when the new packet arrives, the difference between the sampled timestamp value and the previously saved LTS value (TDIFF) is the number of timestamp units between the two packet arrivals.

According to the algorithm, CTS and ETS bucket sizes increment by one at the rate of CIR and EIR, up to the size of CBS and EBS respectively. The same result is achieved by calculation at the packet arrival time: adding TDIFF\*CIR to CTS and TDIFF\*EIR to ETS and truncating the results to CBS and EBS respectively if the results exceed the bucket sizes. CTS and ETS are represented by fixed-point numbers to prevent accumulated computation error (number of tokens is updated on a packet arrival and has to accumulate a non-integer TDIFF value multiplied by a non-integer rate value). The integer bits in CTS are automatically set to cover the CBS value and the rest of the bits are used as the fractional portion. Similarly, the integer bits ETS are automatically set to cover EBS value and the rest of the bits are used as the fractional portion. This setting maximizes the accuracy of accumulated tokens as long as the bucket fills up.

When a new packet with size B arrives to the system, CTS and ETS are first updated as described above to reflect the most updated bucket status, and then the profile metering is applied on the packet.

In color-aware mode: the following metering is applied:

- If B has been pre-colored GREEN
  - If  $B < CTS$ , the packet is GREEN and CTS decrements by B.
  - Else if  $B < ETS$ , the packet is YELLOW and ETS decrements by B.
  - Else the packet is RED
- If B has been pre-colored YELLOW
  - If  $B < ETS$ , the packet is YELLOW and ETS decrements by B.
  - Else the packet is RED
- If B has been pre-colored RED it is not tested against the two buckets and the color remains RED.

In color-blind mode, all packets are assumed to be pre-colored GREEN and the operation is the same as defined in the color-aware for this case.

If the packet has been colored RED and the profile RNIA is configured to BMI discard packet command, the BMI drops the packet. Otherwise if RNIA is configured to the BMI enqueue frame command, the BMI copies the frame IC from FMan memory to external memory and forwards the enqueue command to the QMI. The profile programming allows other usage scenarios by providing GNIA, YNIA and RNIA for GREEN, YELLOW or RED packets respectively. Each NIA may also select the Policer as next module and concatenate Policer profiles per color to achieve complex policing schemes.

#### **5.11.5.4 Functional Description—Next Invoked Action (NIA)**

The Policer uses the NIA to calculate the Policer profile of the incoming packet.

#### **5.11.5.5 Debug Support**

The Policer has support for the following different levels of debug:

- Packet-based debug trace
- Flow-based debug trace

### 5.11.5.5.1 Packet-Based Debug Trace

A packet-based debug trace can be enabled by any of three independent debug flows, initialized in the BMI. Whenever a debug flow is active, the Policer debug logic associated with the selected flow is activated.

Each debug flow's trap logic can analyze frame attributes and packet metering results to produce a match. A debug flow can also be bypassed by setting an always-match condition.

When the trap logic associated with a debug flow is activated and no match is found, the flow's debug condition is cleared. When the trap equation gets a match, the flow's debug condition remains in effect and continues to the next module. Trace data may be dumped to the debug area of the frame's internal context according to the trace level defined for the corresponding debug flow in FMPL\_DTRCR. For details, see [Section 5.11.4.18, “FMan Policier Debug Trace Configuration Register \(FMPL\\_DTRCR\).”](#) In addition, a trap match statistics counter is incremented for each debug flow in the Policier.

### 5.11.5.5.2 Debug Flow Traps

Each debug flow can contain one or more traps, which consist of a set of registers that can combine multiple base criteria into an equation with AND or OR relationships between them. The result of such an equation is a combined, single debug event with a boolean true/false value that corresponds to a match or no-match condition. Each trap condition is defined by a set of the following registers:

- Flow A/B/C debug trap configuration register (FMPL\_FxDBTCR $n$ ). See ([Section 5.11.4.19, “FMan Policier Flow A Debug Trap Configuration Registers \(FMPL\\_FADBTCR \$n\$ \).”](#))
- Flow A/B/C debug value register (FMPL\_FxDBVALR $n$ ). See [Section 5.11.4.20, “FMan Policier Flow A Debug Value Registers \(FMPL\\_FADBVALR \$n\$ \).”](#)
- Flow A/B/C debug trap mask register (FMPL\_FxDBTMR $n$ ). See [Section 5.11.4.21, “FMan Policier Flow A Debug Trap Mask Registers \(FMPL\\_FADBTMR \$n\$ \).”](#)

The debug trap configuration register, FMPL\_FxDBTCR $n$ , defines the compare operation (CMPOP), AND/OR logic function with the next trap condition (AND), and the selected field for this condition (FSEL). Each encoding of FSEL identifies the data sources for the values that will be used in the trap's comparison operation, and each such field option may consist of a single value or a concatenation of multiple whole or partial values. Fields with fewer than 32 bits are left padded with zeros to form a 32-bit value. The selected field and the mask value from FMPL\_FxDBTMR $n$  pass through a bit-wise AND operation, and the result is compared to the value from FMPL\_FxDBVALR $n$  according to the compare operation defined by CMPOP. The true/false trap match result is either logically ANDed or logically ORed with the next trap result, according to the AND bit selection.

Each of the Policier profiles holds three trap bits in its MODE configuration entry: TRA, TRB and TRC for flows A, B, and C, respectively. The profile trap bit is always considered to be the last trap in the chain, with value of 1 representing a match. If, for example, flow A has set two trap conditions in FMPL\_FADBTCR0, and FMPL\_FADBTCR1, its chain of matches is FMPL\_FADBTCR0, FMPL\_FADBTCR1, and then the selected profile TRA bit value. The value of the AND bit of the first two traps form a boolean equation. Only when the whole equation result is true, is a match found on flow A. This logic applies to all debug flows in Policier. See [Section 5.11.4.33.1, “FMan Policier Profile Entry](#)

“**MODE Configuration Word Register (MODE)**,” for the configuration of profile debug flow settings for TRA, TRB, and TRC.

#### **5.11.5.5.3 Debug Trace**

When a debug flow is activated and its trap set produces a match, the Policer can dump trace data to the debug area of the frame’s IC.

Each of the Policer debug flows defines its trace verbosity level through 2-bit TLA, TLB, and TLC fields in the debug trace configuration registers (FMPL\_FxDTRCR). A value of 2b’00 disables trace data of its associated flow. Values 2b’01 to 2b’11 define increasing levels of minimum, verbose, and highly verbose trace options. [Table 5-454](#) describes the trace data dumped in each trace verbosity level.

**Table 5-454. FMan Policer Debug Trace Format**

Verbosity Level		Size	Field Descriptions
Very verbose trace (20 bytes)	Minimum trace (4 bytes)	1 byte	Trace level and size The two msbs reflect the trace level. The 6 lsbs hold the size of the trace data dumped by the Policer in bytes.
		3 bytes	Policer input NIA This is the NIA received by the Policer.
		4 bytes	Timestamp value This is the TSB value snapshot at the time of the trace data dump to FMan memory.
	Verbose trace (8 bytes)	1 byte	FCS action (bits 0–1), Port-ID (bits 2–7) The 2-bit FCS action field shows MODE[FCS] configuration effect on the packet length calculation: 00 FCS bit cleared, or length adjustment not enabled because profile is not metering in BYTE mode 01 FCS set, added four bytes to packet length (got packet without FCS, L2 or full packet length) 10 FCS set, subtracted four bytes from packet length (got packet with FCS, L3 or L4 packet length) 11 FCS set and profile works in BYTE mode, but packet length did not need adjustment The 6-bit port ID associated with the processed frame
		2 bytes	Profile MODE configuration These bytes provide in condensed format important configuration fields from the profile entry MODE register. For details, see <a href="#">Table 5-456</a> .
		2 bytes	Packet length This field provides 16 lsbs of the effective packet length calculated on the accessed profile (in a normal case, it is a reduction of the selected Parser header offset from the full frame length). The 20-bit packet length is a concatenation between this field and the 4 MSBs from PLMSB field in the debug flags entry. For detailed description of debug flags and PNUM fields, see <a href="#">Table 5-455</a> .
		3 bytes	Debug flags and PNUM These bytes provide the debug match trigger, flags, and absolute profile number mapped for this frame. They also hold the 4 msbs of a 20-bit packet length. For a detailed description of debug flags and PNUM fields, see <a href="#">Table 5-455</a> .
		4 bytes	Profile configured CBS For details, see <a href="#">Section 5.11.4.33.6, “Committed Burst Size Configuration Word Register (CBS).”</a>
		4 bytes	Profile CTS The fixed point representation of CTS is known according to the CBS value. For details, see <a href="#">Section 5.11.4.33.10, “Committed Token-Bucket Status Variable Register (CTS).”</a>
		4 bytes	Profile configured PBS_EBS For details, see <a href="#">Section 5.11.4.33.8, “Peak/Excess Burst Size Configuration Word Register (PBS_EBS).”</a>
		4 bytes	Profile PTS_ETS The fixed point representation of PTS_ETS is known according to the PBS_EBS value. For details, see <a href="#">Section 5.11.4.33.11, “Peak/Excess Token-Bucket Status Variable Register (PTS_ETS).”</a>

**Table 5-455** describes the debug flags and PNUM dump fields.

**Table 5-455. Debug Flags and PNUM Dump Field Descriptions**

Bits	Name	Description
0	FLM	Frame length mismatch 0 Frame Length calculation has been selected by MODE[FLS]. 1 Frame Length calculation has been rolled back by MODE[RBFLS].
1	FATM	Flow A trap match This bit indicates that flow A traps and MODE[TRA] match, and therefore, flow A is one of the triggers for dumping debug data. In collaboration mode, this bit indicates the last concatenated trap match result.
2	FBTM	Flow B trap match This bit indicates that flow B traps and MODE[TRB] match, and therefore, flow B is one of the triggers for dumping debug data. If flow B is bypassed due to trap collaboration mode with flow A, this bit is set and flow B traps are associated with flow A match conditions.
3	FCTM	Flow C trap match This bit indicates that flow C traps and MODE[TRC] match, and therefore, flow C is one of the triggers for dumping debug data. If flow C is disabled due to trap collaboration mode with flow A, this bit is set and flow C traps are associated with flow A match conditions.
4–7	—	Reserved
8–15	PNUM	Profile number This is the absolute selected profile number.
16–17	PPC	Packet pre-color This field shows the packet pre-color. In color-blind mode, the pre-color is implicitly GREEN. 00 Packet pre-color is GREEN 01 Packet pre-color is YELLOW 10 Packet pre-color is RED 11 Packet pre-color is OVERRIDE
18–19	PC	Packet color This field shows the final packet color. 00 Packet Color is GREEN 01 Packet Color is YELLOW 10 Packet Color is RED 11 Packet Color is OVERRIDE
20–23	PLMSB	Packet length MSB This field holds 4 msbs of packet length and should be concatenated with the 16-bit packet length dump data. This provides support for 20-bit packet lengths that could be represented in the frame descriptor (FD).

**Table 5-456** describes the debug mode dump fields.

**Table 5-456. Debug Mode Dump Field Descriptions**

Bits	Name	Description
0	PI	Profile initialization bit This bit indicates the selected profile MODE[PI] configuration bit. 0 Profile has not been explicitly initialized and used default pass-through mode. The packet color is not changed, and THE NIA was taken from FMPL_GCR[DEFNIA]. 1 Profile has been explicitly initialized and its configuration entries used for rate processing.
1	CBLND	MODE register color blind (CBLND) bit

**Table 5-456. Debug Mode Dump Field Descriptions (continued)**

Bits	Name	Description
2–3	ALG	MODE register algorithm selection (ALG) field
4–7	FLS	MODE register frame length selection (FLS ) field
8	PKT	MODE register packet mode (PKT) bit
9	RBFLS	MODE register roll back frame length selection (RBFLS ) bit
10	FLSRES	Frame length selection (FLS) result This field is valid only if the profile FLS field selected a non-L2 header offset, and a frame length mismatch has been detected. It is therefore qualified by the asserted FLM bit in the debug flag's trace dump. 0 If FLM is asserted in the trace dump debug flags, the L2 header has been selected by RBFLS roll back action. Otherwise, the header selected by FLS has been used for frame length calculation. 1 Full frame length has been selected by RBFLS roll back action.
11–15	FPP	MODE register fixed point position (FPP) field

When multiple debug flows are enabled and get a match, the highest verbosity level of the matching flows is selected as the associated frame's trace level.

To provide a larger number of trap criteria for debug flow A, the traps that belong to debug flows B and/or C can be chained as a continuation of the traps for debug flow A. This lowers the number of active debug flows, but allows the criteria for debug flow A to be more extensive. The chained trap set then acts as one trap entity with a single boolean result. The debug flows (B and/or C) that donate their traps to debug flow A are put into bypass, since they would no longer have the trap resources needed to define match criteria. If debug flows B or C are not participants in a collaboration chain, then their trapping behavior will remain active and operate independently as configured. Since Policer supports two trap conditions per debug flow, a maximum of 6 trap conditions may be chained if debug flows B and C both participate in a collaboration set with debug flow A.

When debug flow A traps are concatenated through the FMPL\_DTRCR[TRCO] field with another flow's trap registers, the other flow is disabled and put into bypass. This setting associates the entire chain of trap registers to flow A and the trace verbosity level, in case of a match, is selected by FMPL\_DTRCR[TLA]. In addition, the profile entry TRA is chained in this case to the last trap that has been associated to flow A.

#### 5.11.5.5.4 Flow-Based Debug Trace

The flow-based debug trace is a subset of the packet-based debug trace. Trace data is dumped for an enabled flow whenever its debug traps are configured to the always-match condition. The only condition in such a case is that a frame had been previously tagged as belonging to a given debug flow (A, B, or C) at BMI, and not further qualification is necessary.

To configure a flow so that it always dumps data, configure the base trap condition of the selected flow as follows:

- Configure the selected flow FMPL\_FnDBTCR0 (flow A/B/C debug trap configuration register 0) with CMPOP = 0b01 (always match) and AND = 0b0 (OR with next trap). This ensures a match of the selected flow regardless of the evaluation of any remaining trap conditions for the debug flow. The Policer dumps trace data according to the configured trace level for the associated debug

flow. For details, see [Section 5.11.4.19, “FMan Policer Flow A Debug Trap Configuration Registers \(FMPL\\_FADBTCRn\).”](#)

## 5.11.6 Initialization Information

### 5.11.6.1 Reset Initialization

#### 5.11.6.1.1 Hard Reset Sequence

Out of hard reset, all the configuration registers are reset to their defined reset value. The Policer performs automated PRAM initialization by writing zeros to all profile entries. While the self-initialization is in progress, the FMPL\_PAR[PSI] and FMPL\_PAR[GO] bits are set. The self-initialization sequence runs for 512 Policer clocks and at the end of the process the FMPL\_PAR[GO] is self-cleared. In addition, the FMPL\_EVR[PSIC] status bit is set.

Software can optionally poll FMPL\_EVR[PSIC] or enable its interrupt by setting FMPL\_IER[PSIC] to get an indication when the PRAM initialization is completed. After PRAM is initialized, software can write 1 to clear FMPL\_EVR[PSIC].

#### 5.11.6.1.2 Activating the PRAM Self Initialization

Software can initiate PRAM self-initialization sequence at any time. See [Section 5.11.6.3.2, “Full PRAM Re-Initialization,”](#) for details.

#### 5.11.6.1.3 FMan Soft Reset Initialization

The Policer gets a soft reset input from the FMan. The following actions are triggered by a soft reset:

- The Policer stops packet processing immediately. Any packet in progress may not complete its processing and not reflected in the statistic counters.
- All internal state registers and FIFOs are reset.
- All configuration registers are reset to their defined reset values, except for global statistic counters that keep their pre-reset state. User may initialize the statistic counters later by register write accesses. Note that the statistic counter values may miss packets that got to the Policer but did not complete their processing.
- The PRAM is not re-initialized and keeps all profile configurations as well as accumulated statistics before the soft reset. User may clear it later by activating the self initialization sequence.

### 5.11.6.2 Profile Entry Initialization

Out of hard reset, all profile entries are zeroed by the PRAM self-initialization sequence. At any time, software can use the atomic access registers to update full or partial profile entry. The following sequence describes an update of a full entry:

- Optional: ensure FMPL\_EVR[AAC] is cleared (only required if polling used to qualify end of access, or if an interrupt event is enabled for event-driven operation).

- Write all the profile atomic access words in registers FMPL\_PEMODE through FMPL\_PERRPC (total of 16 32-bit registers). To initialize FMPL\_PECTS and FMPL\_PEPTS\_ETS to “full bucket” condition, write them with 0xFFFF\_FFFF and initialize FMPL\_PELTS to all zeros. The Policer updates bucket status in CTS and PTS\_ETS based on CBS and PBS\_EBS on the first packet event. It also updates LTS to its correct value.
- Activate the atomic write action by writing FMPL\_PAR with: GO = 1, RW = 0, PSI = 0, PNUM = Profile Number, PWSEL = 0xFFFF (select all words).
- Optional: to ensure the last write action is completed, either poll when FMPL\_PAR[GO] is cleared, or when FMPL\_EVR[AAC] is set. If FMPL\_IER[AAC] is set, the Policer interrupt is asserted when the update is complete. The access completion is fast and in most cases on the first poll the action is already complete. At the end of the update, clear FMPL\_IER[AAC] for the next event.

A group of profiles with the same parameters can be initialized by repeating the write to FMPL\_PAR while changing only the PNUM field value on each write. The Policer ensures completion of an atomic operation in back-to-back writes without requiring software to poll FMPL\_EVR[AAC].

Multiple profiles can be partially updated by setting the appropriate bits in FMPL\_PAR[PWSEL]. The following sequence should be followed to initialize partial profile parameters by read-modify-write:

- Optional: ensure FMPL\_EVR[AAC] is cleared (only required if polling is used to qualify the end of access, or if the interrupt event is enabled for event-driven operation).
- Activate the atomic read action by writing FMPL\_PAR with: GO = 1, RW = 1, PSI = 0, PNUM = Profile Number, PWSEL = don’t care (all words are read regardless of PWSEL).
- Optional: Wait for setting of FMPL\_EVR[AAC] and then clear FMPL\_EVR[AAC] by writing 1 (in case of working in event driven mode).
- Write selected FMPL\_PE\* profile atomic access registers. If the profile configuration is being modified, also initialize full buckets by writing FMPL\_PECTS and FMPL\_PEPTS\_ETS to 0xFFFF\_FFFF. This setting ensures the token buckets are full, and the Policer updates these field values based on CBS and PBS\_EBS values on the first packet event.
- Activate the atomic action by writing FMPL\_PAR with: GO = 1, RW = 0, PSI = 0, PNUM = Profile Number, and set only the selected bits in PWSEL (associated with words that need to be updated).
- Optional: to ensure the action is completed, either poll when FMPL\_PAR[GO] is cleared, or when FMPL\_EVR[AAC] is set. If FMPL\_IER[AAC] is set, the Policer interrupt is asserted when the update is complete. In most cases, the completion is fast and on the first poll the action would already be completed.

### 5.11.6.3 PRAM Re-Initialization after Software Reset

After software reset (see [Section 5.11.6.1.3, “FMan Soft Reset Initialization”](#)), the PRAM is still loaded with the previous profile configurations, dynamic token buckets, last time stamp (LTS), and statistic counters values. It is possible to re-initialize profiles separately, one by one, or all profiles at once.

The full PRAM re-initialization is a preferable option if all profiles remain unchanged. If profile parameter changes are required for a number of profiles, the combination of specific profile re-initialization and full PRAM re-initialization is advised.

### 5.11.6.3.1 Specific Profile Re-initialization

The following sequence should be followed to re-initialize profile entry:

- Optional: ensure FMPL\_EVR[AAC] is cleared.
- Write selected FMPL\_PE\* profile atomic access registers. Initialize full buckets by writing FMPL\_PECTS and FMPL\_PEPTS\_ETS to 0xFFFF\_FFFF. This setting ensures the token buckets are full, and the Policer updates the fields values based on CBS and PBS\_EBS values on the first packet event.
- Activate the atomic action by writing FMPL\_PAR with: GO = 1, RW = 0, PSI = 0, PNUM = Profile Number, and set only selected bits in PWSEL (associated with words that need to be updated). If profile parameters did not change, the PWSEL should select the FMPL\_PECTS and FMPL\_PEPTS\_ETS words and optionally the statistic counters (if previously gathered statistic is not required).
- Optional: to ensure the action is completed, either poll when FMPL\_PAR[GO] is cleared, or when FMPL\_EVR[AAC] is set. If FMPL\_IER[AAC] is set, the Policer interrupt is asserted when the update is complete. In most cases, the completion is fast and on the first poll the action would already be completed.

Note that polling FMPL\_EVR[AAC] or FMPL\_PAR[GO] is not required when updating multiple profiles. Therefore, software can update the buckets and statistic counters of multiple profiles by a simple sequence of write accesses to FMPL\_PAR and only changing the PNUM field.

### 5.11.6.3.2 Full PRAM Re-Initialization

The following sequence should be followed to re-initialize all profiles:

- Ensure FMPL\_EVR[AAC] is cleared.
- Initialize full buckets by writing FMPL\_PECTS and FMPL\_PEPTS\_ETS to 0xFFFF\_FFFF. This setting ensures the token buckets are full, and the RP updates the fields values based on CBS and PBS\_EBS values on the first packet event.
- Optional: ensure FMPL\_EVR[PSIC] is cleared (if working in event-driven mode).
- Initiate PRAM self-initialization by writing FMPL\_PAR with: GO = 1, RW = 0, PSI = 1, PNUM = don't care, and set the selected (or all) bits in PWSEL according the required update. The PRAM self initialization sequence starts when there are no more packets in processing.
- Optional: software can poll FMPL\_EVR[PSIC] or enable its interrupt by setting FMPL\_IER[PSIC] to get an indication when the PRAM initialization is completed. If it is an event-driven operation, after PRAM has been initialized, software should write 1 to clear FMPL\_EVR[PSIC].

Note that polling FMPL\_EVR[PSIC] or FMPL\_PAR[GO] is not required to start updating specific profiles. Any write access to FMPL\_PAR or FMPL\_PE\* while the initialization sequence is not complete (FMPL\_PAR[GO] = 1) gets stalled. However, caution should be taken because it may cause long wait states (up to 512 clocks).

#### 5.11.6.4 Profile Mapping Initialization

The Policer profile number and the Port-ID are manipulated together to get the absolute profile number in the PRAM. This feature is used for virtual port separation.

Each Port-ID is associated with one of the FMPL\_PMR $n$  registers. For details on the Port-ID mapping, see [Section 5.11.4.32, “FMan Policer Profile Mapping Registers \(FMPL\\_PMR \$n\$ \)](#). If a non-supported Port-ID arrives or if FMPL\_PMR $n$ [V] = 0 for the register associated with the Port-ID, the default mapping is selected FMPL\_DPMR. This register is implicitly valid and has similar configuration fields as FMPL\_PMR $n$ . For details on the default mapping register, see [Section 5.11.4.31, “FMan Policer Default Profile Mapping Register \(FMPL\\_DPMR\)](#).

Out of reset, all the profile mapping registers are disabled (FMPL\_PMR $n$ [V] = 0) and the FMPL\_DPMR reset value maps all incoming packets to profile number zero. Software can then enable and initialize the FMPL\_PMR $n$  registers and customize each port-ID mapping.

The following example shows how to map 32 profile selections from the 10-Gbit MAC Rx (Port-ID 0x10) to a continuous group of profiles residing in the Policer PRAM, starting at offset 0xE0 (profiles 0xE0–0xFF):

- The 10-Gbit RX uses Port-ID 0x10 and managed by FMPL\_PMR11. Write this register with the following field values:
  - V = 1: Port is valid
  - PBNUM = 11100000 (0xE0):. This is the base address pointing to the first profile number.
  - BRN = 0101: Selecting the actual profile number to be {PBNUM[0:2], PNUM[3:7]}). This maps a continuous group of 32 profiles to base address 0xE0 (pointed by PBNUM[0:2]).

In addition to the above configuration, it is required that the Policer NIA from the 10-Gbit Rx MAC Port-ID clears NIA[8]. For orthogonal operation of different Port-ID numbers, it is required that software configures the FMPL\_PMR $n$  registers without overlaps.

#### 5.11.6.4.1 Next Invoked Action (NIA)

The Policer uses the NIA to calculate the Policer profile for the current packet. If NIA[8] = 1, the PNUM field selects an absolute profile number (overriding the profile mapping registers). If NIA[8] = 0, the actual PNUM is calculated according to the configurations of the FMan Policer mapping registers (see [Section 5.11.4.32, “FMan Policer Profile Mapping Registers \(FMPL\\_PMR \$n\$ \)](#)). [Table 5-457](#) shows a NIA example.

**Table 5-457. NIA Example**

NIA[8:23]	Selected PNUM
0x8005	NIA[8] = 1, therefore PNUM = 5
0x0005	NIA[8] = 0, therefore PNUM is calculated according to the configurations of the FMan Policer mapping registers.

### 5.11.6.5 Profile Concatenation Initialization

Policer profiles can be concatenated to each other to perform complex functions, such as profile aggregation or checking the same flow with multiple criteria (such as limiting both data rate and packet rate). Concatenation is achieved by configuring the GNIA, YNIA, and RNIA fields of the profile entry to select the Policer as the next module, and specifying the next profile number. It is recommended to use profile concatenation with absolute profile numbering (NIA[8] = 1) to eliminate looping in the same profile group. It is the software's responsibility to prevent looping conditions such as iterating on the same profile number.

The following example shows configuration of aggregation function: In addition to each port-ID policing algorithm, it limits the overall packet rate of GREEN, YELLOW, and RED packets. It assumes all the Port-ID mapping registers (FMPL\_PMR $n$  and FMPL\_DPMR) implement virtual Port-ID separation and do not map directly to profile numbers 253, 254, and 255; therefore, software can assign these profile numbers for packet rate limiting of aggregated GREEN, YELLOW and RED packets.

#### Example 5-4. Input Policier NIA with NIA[8] Cleared

---

Configure all profiles (except for 253–255) as follows:

- Select the RFC type, color-aware/blind mode, rate/burst parameters, BYTE/PACKET mode as defined for each Port-ID specific policing algorithm.
- Configure GNIA = 0x4C80FD (ORR = 0, module = Policier, Absolute Profile Number = 253)
- Configure YNIA = 0x4C80FE (ORR = 0, module = Policier, Absolute Profile Number = 254)
- Configure RNIA = 0x4C80FF (ORR = 0, module = Policier, Absolute Profile Number = 255)

Configure profile entry 253 as follows:

- Select PACKET mode, color-aware, and RFC rate/burst parameters that define the overall GREEN packet rate.
- Configure GNIA = 0x500002 (ORR = 0, module=BMI, Enqueue Frame command)
- Configure YNIA = 0x4C80FE (ORR = 0, module = Policier, Absolute Profile Number = 254)
- Configure RNIA = 0x4C80FF (OR = 0, module = Policier, Absolute Profile Number = 255)

Configure profile entry 254 as follows:

- Select PACKET mode, color-aware, and RFC rate/burst parameters that define the overall YELLOW packet rate.
- Configure GNIA = 0x500002 (OR = 0, module = BMI, “Enqueue Frame” command). This field should never be reached since color-aware mode honors YELLOW pre-color.
- Configure YNIA = 0x500002 (OR = 0, module = BMI, “Enqueue Frame” command)
- Configure RNIA = 0x4C80FF (OR = 0, module = Policier, Absolute Profile Number = 255)

Configure profile entry 255 as follows:

- Select PACKET mode, color-aware, and pass-through mode.
- Configure GNIA = 0x500002 (OR = 0, module = BMI, “Enqueue Frame” command). This field should never be reached since color-aware mode honors RED pre-color.

- Configure YNIA = 0x500002 (OR = 0, module = BMI, “Enqueue Frame” command). This field should never be reached since color-aware mode honors RED pre-color.
- Configure RNIA = 0x500002 or 0x5000C1 (OR = 0, module = BMI, “Enqueue Frame” or “Discard Frame” command, depending on application requirement).

In the configuration shown in [Example 5-4](#), note the following:

- All the GREEN packets are aggregated by concatenation to profile number 253. This profile limits the overall GREEN packet rate. If the result is still GREEN, the packet is forwarded to the BMI with Enqueue command. If the result is YELLOW, the packet is concatenated to profile number 254 to be counted as part of the overall YELLOW packets. If the result is RED, the packet is concatenated to profile number 255 to be counted as part of the overall RED packets.
- All YELLOW packets are measured by profile 254 (in addition to packets colored YELLOW by profile 253). This profile works in color-aware mode and limits the overall YELLOW packet rate; therefore its color result would never be GREEN. If the result of profile 254 stays YELLOW, the packet is forwarded to the BMI with Enqueue command. If the result is RED, the packet is concatenated to profile number 255 to be counted as part of the overall RED packets.
- Profile 255 gets all the RED packets, including packets colored RED by profiles 253 and 254. Because it works in color-aware mode, its output color is always RED. Therefore, it can work in pass-through mode and direct all the packets to the BMI with either “Enqueue Frame” or “Discard Frame” commands, depending on the application requirement. Effectively, this profile only provides statistics of packet with final RED result, after all aggregation is done.

### 5.11.6.6 Writing to the Statistic Counters

The global statistic counters can be written at any time and continue counting from the written value. The per profile statistic counters, which reside at PRAM, can be accessed through the profile action registers (FMPL\_PAR) by setting the required counter values in FMPL\_PE\* and asserting their associated word select on FMPL\_PAR[PWSEL].

After soft reset, all statistic counters as well as the PRAM keep their pre-reset values for analysis purposes. To start from clean conditions, software must re-initialize them by clearing the statistic counters and programming the profile entries in PRAM.

### 5.11.7 Application Information

This section includes some application examples using the Policer.

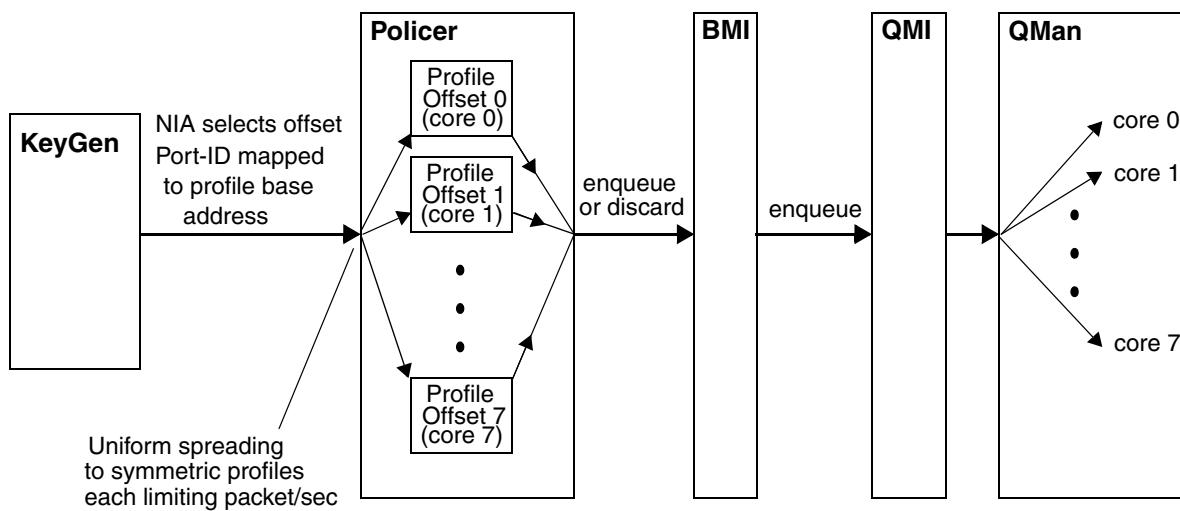
#### 5.11.7.1 Load Spreading and Policing Traffic per Core

[Figure 5-403](#) shows an application example that uses the Policer in combination with the KeyGen for uniform load spreading of selected Port-ID incoming packets between on-chip cores. The Policer limits the packet rate of each core.

- The KeyGen hash function selects FQID. Selected FQID bits used for profile offset and the port-ID number is mapped to the base address of the Policer profile group. Each entry in the profile group

is programmed with the same configuration and limits the packet rate towards one of eight on-chip cores.

- When a new FQ is added to the system, the QMan configuration that routes the frame to specific core should match the KeyGen bits that selects the Policer profile offset. This configuration achieves a one-to-one relation between the selected Policer profile and the target core that services the same packet. For example, if the Policer offset is selected by bits from the FQID, the same routing selection should be made at the QMan.
- Each Policer profile can generate either enqueue or discard commands to the BMI. Only packet rate within the core limits continues from the BMI, though the QMI to the QMan.
- The QMan mapping of the frame FQID routes the frame to the queue serviced by the selected core and matches its associated Policer profile.



**Figure 5-403. Load Spreading Application Example**

### 5.11.7.2 Profile Concatenation for Combined PACKET/BYTE Based Policing

Figure 5-404 shows an application example that runs two Policer iterations on the same flow. The first iteration limits the flow data rate, while the second iteration limits the flow packet rate. This configuration achieves a double policing scheme that could drop frames based on either of the criteria.

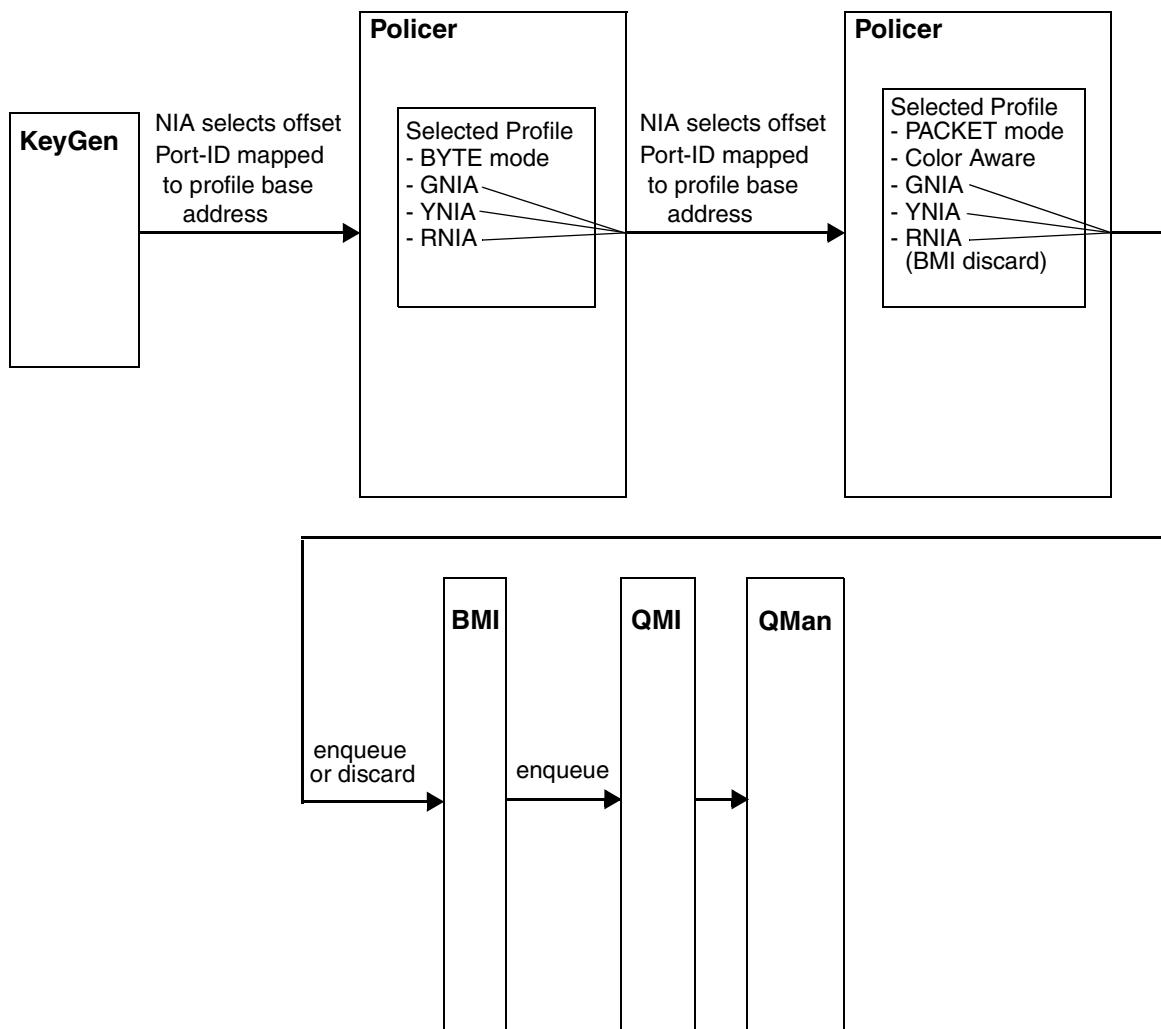


Figure 5-404. Double Policing Scheme by Profile Concatenation

### 5.11.7.3 Profile Concatenation for Aggregating Multiple Streams

Figure 5-405 shows an application example that runs two Policer iterations to aggregate multiple streams into one shared profile. The double policing scheme can color frames on each port based on one of its dedicated configurations, and then color and optionally drop frames based on the aggregated byte or packet rate. This enables resource sharing where the sum of each port-allowed traffic rate is higher than the allowed aggregated traffic rate. When only part of the ports are loaded, all traffic is passing. However, when all ports are loaded, the aggregated traffic rate exceeds the system capabilities and therefore the

second Policer iteration can be used to limit the rate. The pre-coloring achieved per each port is used in the shared profile considerations, which works in color-aware mode.

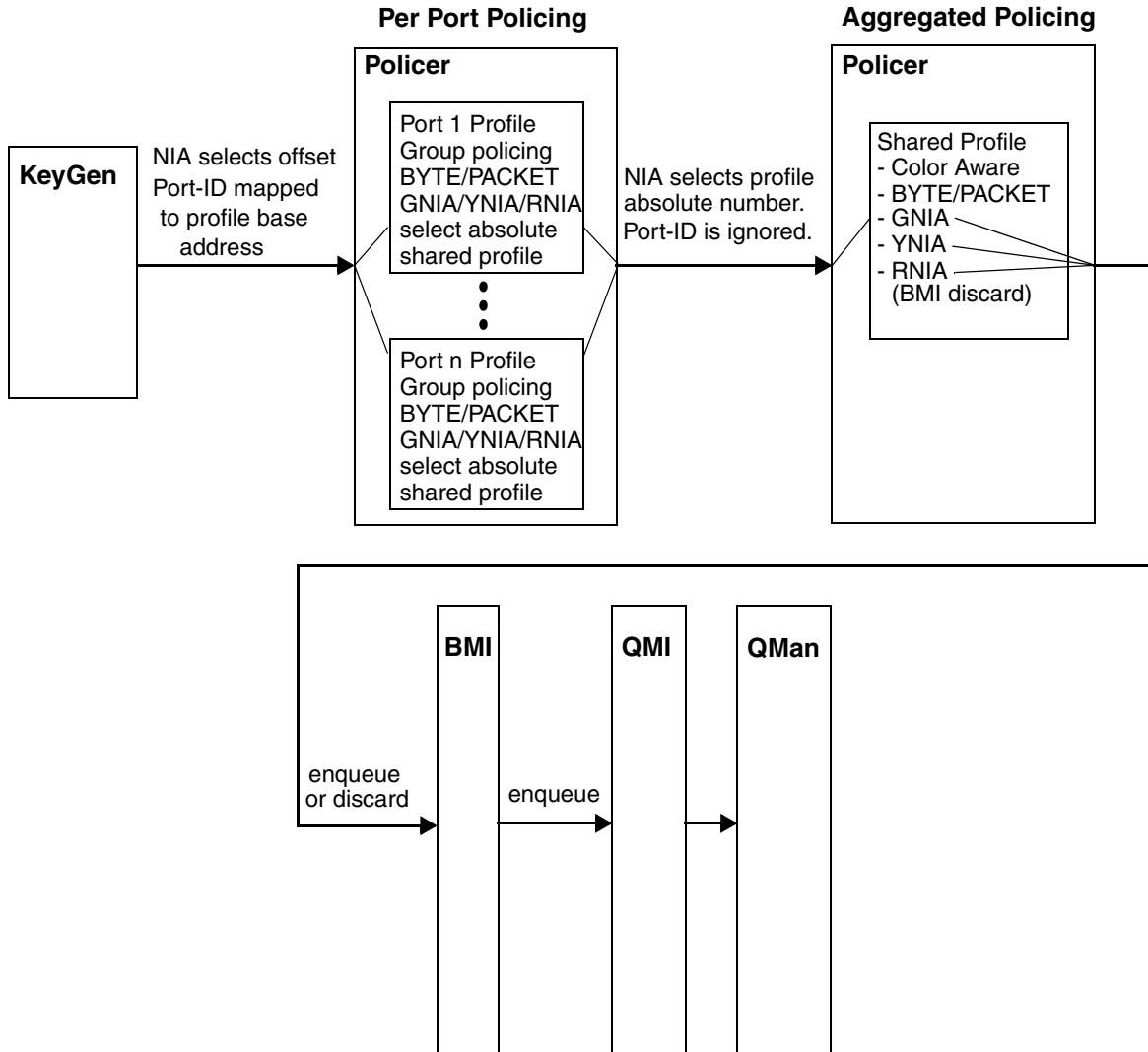


Figure 5-405. Aggregating Multiple Streams by Profile Concatenation

#### 5.11.7.4 Profile Concatenation with Per Color Aggregation

Similar to the usage example described in [Section 5.11.7.3, “Profile Concatenation for Aggregating Multiple Streams,”](#) the final bucket can be split to three buckets for GREEN, YELLOW, and RED traffic aggregation. Then the GREEN bucket aggregation result packets may be further mapped to YELLOW or colored RED, and YELLOW bucket aggregation result packets may be recolored RED. This usage example can provide separately aggregated rates for GREEN, YELLOW, and RED at the system level.



## 5.12 Frame Manager—FMan Controller

### 5.12.1 FMan Controller Overview

The FMan Controller is a configurable engine used for a number of purposes:

- Custom classifier (this is the primary function of the FMan controller):  
The custom classifier may be invoked after KeyGen processing has completed and can be operational in both the Ethernet receive flow and the offline flow. While the KeyGen module is able to distribute frames into queues according to a hash result, the custom classifier may be configured to perform exact matches on a combination of frame headers. The custom classifier uses trees of action descriptors (AD) to perform a variety of actions (such as table lookups) and produces a result descriptor (RD) that contains the necessary information for the continuation of the frame processing in the next module or the next look-up table. The custom classifier supports up to 16 trees of descriptors.
- Ethernet independent mode:  
The Ethernet independent mode is an Ethernet controller. It uses Buffer Descriptor ring structures as the interface to the hosts for transmission or reception of frames. The QMan, BMan, Parser and KeyGen are not used in this mode.
- FMan controller host commands:  
The FMan controller host commands are used to initialize or update or read some data structures in the FMan. This mechanism allows the virtualization of these services in a multicore environment. In addition, the host commands allow for atomic updates to complex table entries, allowing for the FMan to use the tables dynamically, during the update process. The data structures which are accessed by the host commands are: Policer Profiles, KeyGen Schemes, and assistance for dynamic update of custom classifier tables. Additional structures may be supported as defined in subsequent chapters.
- Custom classifier functional extensions  
The custom classifier functional extensions, use the Action Descriptor (AD) programming model of the custom classifier to provide extensions as described below.
  - Header manipulation:  
Header manipulation command descriptors are used to perform operations on the frame header: Remove, Insert, Replace, Protocol Specific Commands. A user programmable number of HMCDs may be invoked for multiple header manipulation operations. See [Section 5.12.10, "Header Manipulation Command Descriptors \(HMCDs\)."](#)
  - IP Reassembly / Fragmentation
  - Statistics
  - Deep Sleep Auto response
  - IPsec acceleration
  - Frame replication
- Extensions to FMan modules for parsing, error checking, flow direction etc.

Three action descriptor (AD) types are used during the custom classifier process:

- Table descriptor—Specifies the matching table parameters on which the look-up is performed.
- New classification result—Specifies which module (for example, the Policer) to initiate next and contains the FQID, virtual storage profile and the optional Policer profile.
- Keep classification result—Specifies which module to initiate next. The FQID and optional Policer profile (PP) are not overwritten and remain the same as before the custom classifier stage.

The first AD is accessed based on CCBASE composed of the BMI custom classifier base address and the KeyGen offset. See [Section 5.10, "Frame Manager—Key Generator,"](#) for more information. The BMI custom classifier base address may differ per port due to the host memory partitioning. When a frame is received and a specific key is extracted from the frame, a look-up operation is executed on the matching table. The look-up's result is either a match or a mismatch. When a match occurs, the action descriptor (AD) is taken from the corresponding offset of the AD table. If a mismatch occurs the default AD is taken from the last entry of the action descriptor table, which is specifically allocated for this purpose.

The look-up is divided into several types according to the expected fields that are extracted. The types split between dedicated routines exist for specific protocol fields in addition to generic routines. The generic routines enabling look-up with respect to unspecified protocol fields.

The lookup key is formed from fields which are extracted either from the frame headers or from the internal context (IC). As an example, the user may need the logical port ID or one of the code results from the IC, as part of the lookup-key.

## 5.12.2 FMan Controller Features Summary

Key features of the FMan Controller include the following:

- Support for nested look-ups
- Comparison tables (including key and potential local mask), supporting up to 255 entries
- Performance considerations
  - Up to three look-ups per packet
  - Comparison table size is up to 128 bytes
- Key size up to 56 bytes (valid table entry sizes: 1, 2, 4, 8, 16, 24, 32, 40, 48 or 56 bytes)
- Global mask up to 4 bytes
- Local mask to support range/longest prefix match for any key size
- Look-up actions using a key extracted from the received frame are as follows:
  - Common extracted protocols fields are as follows:
    - Ethernet MAC destination or source addresses
    - Ethernet type
    - VLAN1 or Last VLAN TCI (use global MASK)
    - PPPoE PPP PID
    - MPLS or Last MPLS (use global MASK)
    - IPv4 TOS (use global MASK)
    - IPv4PTYPE

- IPv4 src. or/and IPv4 dest. addresses
- IPv4 dest. address with local mask
- IPv6 class/flow (use global MASK)
- IPv6 src. or IPv6 dest. addresses
- IPv6 dest. address with local mask
- GRE PType
- MinEnc Protocol
- MinEnc src. or/and dest. addresses
- UDP/TCP (L4) port src. or/and port dest.
- Generic protocol fields are as follows:
  - Programmable key offset (key offset up to 256)
  - Programmable key size (key size up to 56)
  - Option for offset from a known header in the Parser Result
- Look-up actions using a key extracted from the received internal frame context are as follows:
  - Generic protocol fields are as follows:
    - Programmable key offset (key offset up to 256)
    - Programmable key size (key size up to 56)
  - Use indexed look-up
- Up to 16 first ADs (roots) per port
- Supports TTL/Hop limit equal to one identification
- Support for dynamic updating of custom classifier tables
- Header manipulation command descriptors
  - Header removal/insert/replace of programmable number of bytes
  - L2 header
    - Remove of L2 header
    - DSCP to VLAN priority bits translation
  - Removal/insert of MPLS header.
  - Update specific fields in the frame header
    - IPv4: TOS, TTL, ID, Src, Dst
    - IPv6: Traffic Class, Hop Limit, TTL, Src, Dst
  - Header replace of protocol specific header.
    - IPv4
    - IPv6
    - UDP
    - TCP
  - IPv4 with IPv6 replacement.
  - IPv6 with IPv4 replacement.

- Update IP length and IP/UDP/TCP Checksum if needed
- User defined order of the header manipulation command descriptors
- Parse after Header manipulation.
- UDP/TCP Checksum calculation (FMan\_v3 only).
- Support for Deep Sleep Auto Response for the following protocols:
  - Address Resolution Protocol for Ethernet/ IPv4.
  - ICMPv4
  - ICMPv6
  - Neighbor Discovery
  - SNMP

### 5.12.3 FMan Controller NIA—Action Codes

NIA stands for next invoked action. The processing flow in FMan requires different processing modules to perform different tasks. When a module completes a task on a certain flow it must decide which module should continue with the task, and the next required step.

The FMan controller receives commands from other modules within the FMan through the FPM dispatch bus. This table gives a detailed description of the commands the FMan controller can receive through the dispatch bus.

**Table 5-458. FMan controller Dispatch Commands**

Source Modules	Action Code	Task Name	Affected Flows	Description
KeyGen, BMI	0x06	Custom Classifier	RX, O/H	FMan controller performs custom classifier procedure.
BMI	0x08	Independent Mode transmit	TX	FMan controller performs independent mode transmitter procedure.e
BMI	0x0a	Independent Mode receive	RX	FMan controller performs independent mode receiver procedure.
BMI	0x0c	Host Commands	O/H	Host Commands: FMan controller update/read Policier Profile/KeyGen SCHEME and assistance for dynamic update of custom classifier tables.
BMI	0x0e	Pop to Next Step	RX,O/H	Check next engine to execute.
QMI	0x10	Pre-BMI Fetch Frame Header	O/H	See <a href="#">Section 5.12.21, "Pre/Post BMI Fetch NIAs."</a>
BMI	0x12	Post-BMI Fetch	O/H	See <a href="#">Section 5.12.21, "Pre/Post BMI Fetch NIAs."</a>
QMI	0x18	Pre-BMI Fetch Full Frame	O/H	This NIA is relevant only for FMan_v3 or higher. See <a href="#">Section 5.12.21, "Pre/Post BMI Fetch NIAs."</a>
CC/Policer/KG	0x1A	Pre-BMIPreToEn queue	RX,O/H	<p><u>FMan_v3:</u></p> <ol style="list-style-type: none"> <li>1. IP reassembly functionality.</li> <li>2. IPsec ECN/DSCP propagation for Pre-SEC traffic.</li> <li>3. Responsible for DMA write of the frame header/SG list first entry (if FWD=1) in case one of the following entities has changed it: Pre/Post BMI Fetch (on OP after SEC where UDP Length/EtherType/ECN/DSCP fields may be updated), Header Manipulation (should be placed on each port that HM is working either OP/RX).</li> <li>4. This is a work around for the multiple VSP per error queue - firmware will overwrite the VSP with “ErrorRSPID” as programmed in <a href="#">Table 5-540</a>.</li> <li>5. Provide another work around for HW bug - if “VSPDIsOPFix” bit is set in <a href="#">Table 5-540</a>.</li> </ol> <p>This NIA continues to BMIPreToEnq NIA.</p> <p>If this NIA is instantiated, then the Post-BMI Prepare To Enqueue NIA (0x14 or 0x22) must also be instantiated.</p>

**Table 5-458. FMan controller Dispatch Commands**

Source Modules	Action Code	Task Name	Affected Flows	Description
CC/Police/KG	0x1E	Pre-BMIDiscard Frame	RX,O/H	<p><u>FMan_v3:</u></p> <ol style="list-style-type: none"> <li>IP reassembly functionality.</li> <li>IPsec ECN/DSCP propagation for Pre-SEC traffic.</li> <li>Responsible for DMA write of the frame header/SG list first entry (if FWD=1) in case one of the following entities has changed it: Pre/Post BMI Fetch (on OP after SEC where UDP Length/EtherType/ECN/DSCP fields may be updated), Header Manipulation (should be placed on each port that HM is working either OP/RX)</li> <li>This is a work around for the multiple VSP per error queue - firmware will overwrite the VSP with "ErrorRSPID" as programmed in <a href="#">Table 5-540</a>.</li> <li>Provide another work around for HW bug - if "VSPDIsOPFix" bit is set in <a href="#">Table 5-540</a>.</li> </ol> <p>This NIA continues to BMI Discard Frame NIA.</p>
Keygen	0x20	Pre-Custom Classifier	O/H	This NIA restores the Flow ID field by copying the last 2 bytes of the IC into IC[0x16].
BMI, FMan Controller	0x22	Post-BMI Prepare To Enqueue	RX, O/H	<p>This NIA completes the work performed by the Variable IP Version/Length features (VIPV_EN,VIPL_EN) of the preceding IPsec Manipulation TD. This NIA must be called with ORR cleared.</p> <p>If this NIA is instantiated, then the Pre-BMI Prepare To Enqueue NIA (0x1A) must also be instantiated.</p>
BMI	0x24	Post Transmission	Tx	<p>This NIA features a set of actions to be done after frame transmission, depending on Context A [A1] field.</p> <p>To support this NIA user should:</p> <ul style="list-style-type: none"> <li>Set A2[NL] bit to allow this feature.</li> <li>Set FMBM_TCMNE to 0x24.</li> </ul>
CC/Police/KG	0x28	Pre-BMIPreToEn queueNonIPACC Offload	RX,O/H	<p>FMan_v3 only if IP acceleration offloading is not enabled. This NIA continues to BMI Prepare To Enq NIA.</p> <p>This is a work around for the multiple VSP per error queue - firmware overwrites the VSP with "ErrorRSPID" as programmed in <a href="#">Table 5-540</a>.</p> <p>Provide another work around for HW bug - if "VSPDIsOPFix" bit is set in <a href="#">Table 5-540</a>.</p>
CC/Police/KG	0x2A	Pre-BMIDiscard FrameNonIPACC Offload	RX,O/H	<p>FMan_v3 only if IP acceleration offloading is not enabled.</p> <p>This NIA continues to BMI Discard Frame NIA.</p> <p>This is a work around for the multiple VSP per error queue - firmware overwrites the VSP with "ErrorRSPID" as programmed in <a href="#">Table 5-540</a>.</p> <p>Provide another work around for HW bug - if "VSPDIsOPFix" bit is set in <a href="#">Table 5-540</a>.</p>

**Table 5-458. FMan controller Dispatch Commands**

Source Modules	Action Code	Task Name	Affected Flows	Description
BMI	0x2C	Pop to Next Step NonIPACCOffload	RX,O/H	FMan_v3 only if IP acceleration offloading is not enabled. This NIA should be programmed in the FMBM_xCMNE register. This is a work around for the multiple VSP per error queue - firmware overwrites the VSP with “ErrorRSPID” as programmed in <a href="#">Table 5-540</a> . Provide another work around for HW bug - if “VSPDisOPFix” bit is set in <a href="#">Table 5-540</a> .
Parser	0x2E	Deep Sleep Auto Response	Rx	FMan controller performs the deep sleep auto response functionality. See <a href="#">Section 5.12.13,“Deep Sleep Auto Response.”</a>

## 5.12.4 FMan Controller Memory Map/Register Definition

The following registers are used to download configuration data to the FMan Controller.

The size of the FMan Controller configuration data space varies in various versions of the FMan. See SoC DPAA instruction chapter for the size in a specific device. The FMan Controller configuration data download is accomplished using the registers listed in [Table 5-459](#).

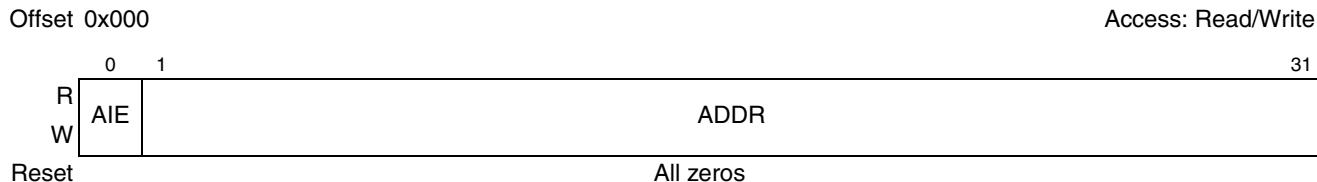
**Table 5-459. FMan controller Configuration Data Download Registers**

FMan Controller Offset <sup>1</sup>	Name	Access	Reset	Section/Page
0x000	FMCDADDR—FMan Controller configuration data, address register	R/W	All zeros	<a href="#">5.12.4.1/5-550</a>
0x004	FMCDDATA—FMan Controller configuration data, data register	R/W	All zeros	<a href="#">5.12.4.2/5-551</a>
0x00C	FMCDREADY—FMan Controller configuration data ready register	R/W	All zeros	<a href="#">5.12.4.3/5-551</a>
0x800–0xFFFF	FMCDDATA—Multiple addresses for FMCDDATA register (bursts are allowed for faster download.)	R/W	All zeros	<a href="#">5.12.4.2/5-551</a>

<sup>1</sup> See [Table 5-17, “FMan Memory Map Regions,”](#) for details of FMan controller memory space within FMan memory space.

### 5.12.4.1 FMan Controller Configuration Data Address Register (FMCDADDR)

The FMan Controller configuration data address register is used to program the address of the FMan Controller configuration data memory. The data associated with this address is then written to the FMCDDATA register. Address auto increment can be configured to allow repeated accesses to the FMCDDATA register without manually updating the FMCDADDR register. The initial value for the auto increment value is the value written to the ADDR field of FMCDADDR.



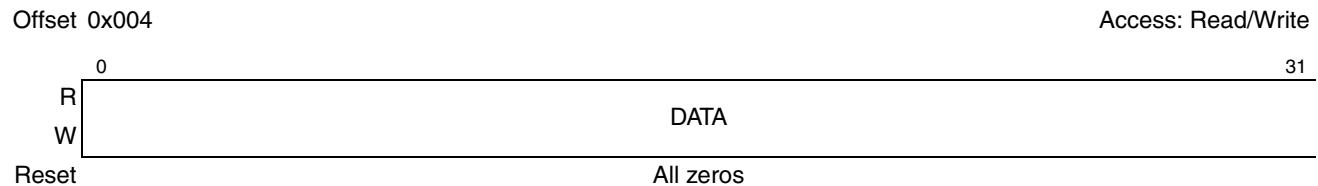
**Figure 5-406. FMan Controller Configuration Data, Address Register**

**Table 5-460. FMCDADDR Field Descriptions**

Bits	Name	Description
0	AIE	Auto Increment Enable 0 Auto increment is disabled. 1 Auto increment is enabled. ADDR is automatically incremented for the next FMan Controller configuration data word (4 bytes).
1–31	ADDR	Address. Bits 30–31 should be cleared (address should be word aligned). This address is updated every read or write during auto increment (a value of 4 is added with each operation).

### 5.12.4.2 FMan Controller Configuration Data, Data Register (FMCDDATA)

The FMan Controller configuration data, data register is used to access the FMan Controller configuration data memory. An access to the FMCDDATA register results in an access to the corresponding FMan Controller configuration data memory location. The address of the access is determined by the FMCDADD register.



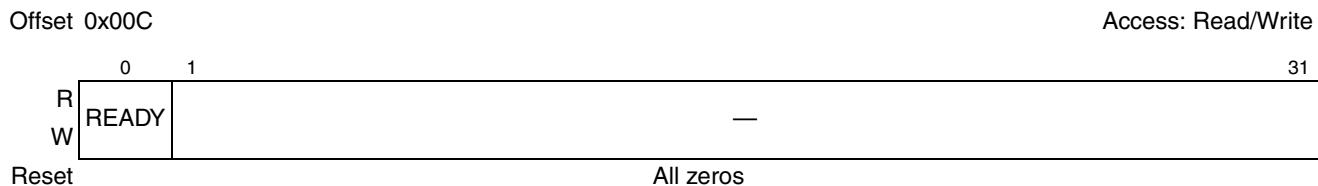
**Figure 5-407. FMan Controller Configuration Data, Data Register**

**Table 5-461. FMCDDATA Field Descriptions**

Bits	Name	Description
0–31	DATA	Data written to or read from the FMan configuration data memory at the specified FMCDADD[ADDR] address.

### 5.12.4.3 FMan Controller Configuration Data Ready Register (FMCDREADY)

FMan Controller configuration data ready register is used to allow the FMan controller to access the FMan Controller configuration data after it has been downloaded. The reset value of FMCDREADY blocks the FMan controller from accessing the FMan Controller configuration data until it is first downloaded.



**Figure 5-408. FMan Controller Configuration Data Ready Register**

**Table 5-462. FMCDREADY Field Descriptions**

Bits	Name	Description
0	READY	FMan Controller configuration data ready bit 0 FMan controller access to FMan Controller configuration data memory is blocked. 1 FMan controller access to FMan Controller configuration data memory is permitted.
1–31	—	Reserved

### 5.12.4.4 FMan Controller Configuration Data Download Flow

The FMan controller configuration data is loaded to the data memory by using FMCDADDR and FMCDDATA registers. The configuration data is loaded in 4-byte granularity by writing the address to FMCDADDR and writing the data to FMCDDATA. To reduce the complexity of loading the configuration

data, the auto increment feature may be used. This table presents the configuration data download flow with auto increment.

**Table 5-463. FMan Controller Configuration Data Download Flow**

Operation		Comment
Write 0x80000000 to FMCDADDR.		Initialize FMCDADDR ADDR field to zero and enable auto increment.
Memory barrier		This ensures that the ADDR is updated before proceeding.
For n=0 to configuration_data_size/4		Loop over all of the configuration data.
Loop block	Write configuration data word n to FMCDDATA.	Write configuration data to memory.
While( (n%4)!=0 )		Continue filling the memory until we have a multiple of 16 bytes (this is done to avoid ECC errors).
Loop block	Write 0xFFFF_FFFF to FMCDDATA.	Write to FMCDDATA in order to pad configuration data.
	n++	Increment loop counter.
Memory barrier		This ensure that the data is updated before proceeding.
Write 0x80000000 to FMCDREADY.		This allows access by the FMan controller to the configuration data.

## 5.12.5 FMan Controller Functional Description

### 5.12.5.1 Matching Table Structure

Matching tables are used to perform comparisons between the input key and the entries in the table. When a match is found, the corresponding custom classifier action descriptor (AD) entry is chosen. Upon a mismatch, the last custom classifier AD entry is selected. This implies that the last entry of the matching table is always ignored and not valid. As shown in [Figure 5-409](#), the matching table consists of the key entries and the optional key local masks. Up to 255 entries can reside in the matching table, where an entry may be only a key or a key and local mask pair. The size of the table must not exceed 128 bytes in order to accommodate 18Mpps performance.

The algorithm for locating a key in the table can be performed in one of the following ways:

- No mask—Each entry is regarded as if it is masked by the smallest valid size without truncating any portion of the key. Valid entry sizes are 1, 2, 4, 8, 16, 24, 32, 40, 48, or 56 bytes. The key is always aligned to the most significant bytes of the match table entry. For example, if searching for an Ethernet MAC address, which is 6 bytes in length, each entry in the table must be 8 bytes long, where the 6 most significant bytes are used for the entry and the 2 least significant bytes are ignored in the comparison process.
- Global mask—The same as no mask, but using a programmable global mask of 4 bytes for masking the relevant bits of the input key and the table entry values. Global mask is valid only for keys up to 4 bytes. This method can be useful for MPLS or VLAN, for example, where only specific bits out of the entry are relevant for the search. The global mask must be aligned to the most significant bytes in the custom classifier table descriptor. For example, if looking for VLAN TCI, which is

only 2 bytes long, the user must program the global mask aligned to the left as 0xFFFF\_0000 (or any other value instead of ‘0xFF’), rather than being aligned to the right.

- Local mask—The same as no mask, but for each entry in the table there is a specific mask of the same length that determines which bits are relevant for the search. This mask is bit-wise and applies to both the input key and the applicable table entry. This method can be useful for an LPM algorithm or for a range search. The mask is always aligned to the most significant bytes of the match table entry.

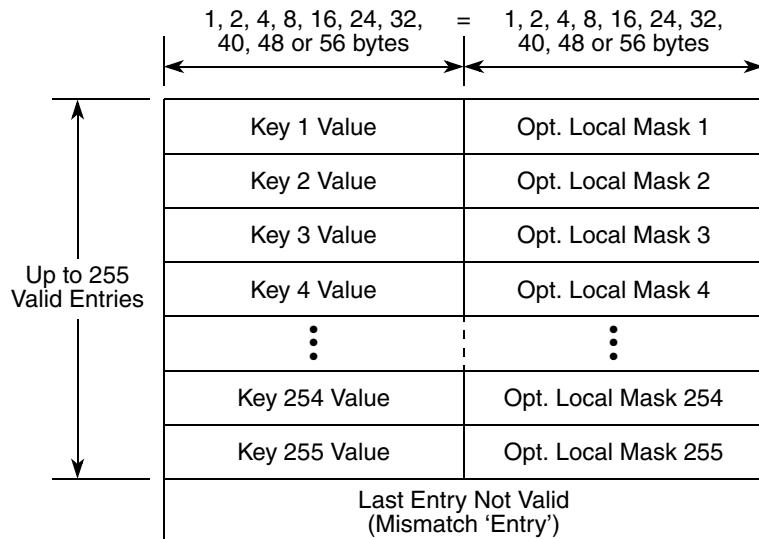


Figure 5-409. Matching Table Structure

### 5.12.5.2 TTL/Hop Limit Equals One Identification

The FMan controller supports identification of IPv4 TTL or IPv6 hop limit equals one. The user programs this by setting the custom classifier action descriptor (AD) of table type with the parse-code field set to “TTL/Hop Limit Identification”. In this case the custom classifier AD is not used as a typical table type because there is no matching table algorithm. If the TTL/Hop limit field value equals one, then the selected custom classifier AD is taken from the custom classifier AD base address. Otherwise (if TTL/Hop Limit is greater than 1) the selected custom classifier AD is taken from the custom classifier AD base address plus 16 bytes.

### 5.12.5.3 Look-Up on Internal Context Fields

The FMan controller supports look-up not only on fields (keys) resides in the incoming frame but also on fields which resides in the Internal Context. This can be very useful for look-up on the Key that the KeyGen generated or for look-up on the parser layer-x code results. For more information about this Custom Classifier Parse Code see [Table 5-467](#) for Generic\_5\_Off\_IC\_GMASK Parse-Code.

### 5.12.5.4 Using KeyGen HASH Result for Indexed Look-Up

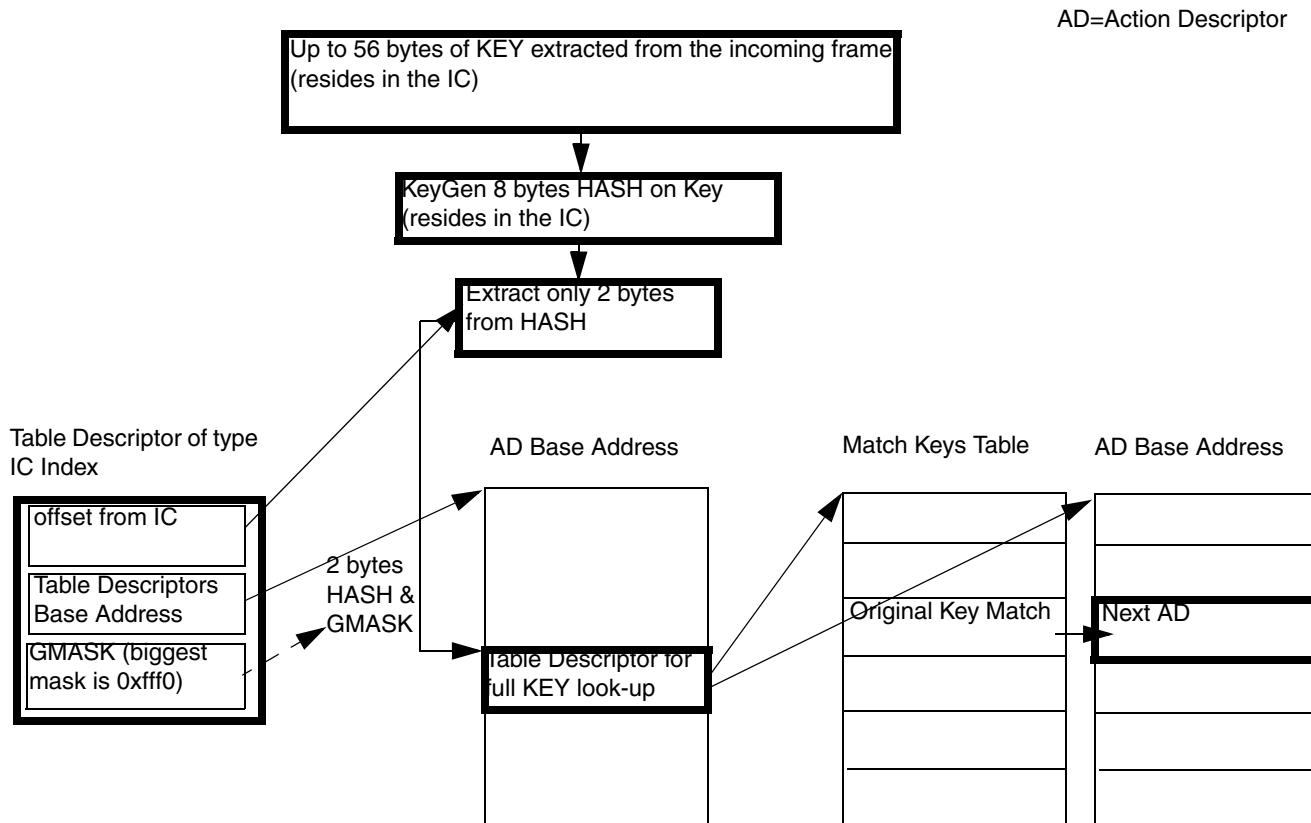
When dealing with big number of possible KEY results, for example, 2K possible MAC source address, the look-up tables are too big to handle. In-order to be able to handle such big look-up searches, the FMan

controller uses the KeyGen HASH result to be able to limit the search on much smaller look-up tables. These tables contains the Keys that have contention from the KeyGen HASH result.

[Figure 5-410](#) describes the flow for using KeyGen HASH result for indexed look-up. The flow starts with the KeyGen that perform hash function on the KEY extracted from the incoming frame, and generate an HASH result. Only 2 bytes are taken from this HASH result and is used as an index to select the hash bucket. This bucket should contain all possible Keys which may have contention on the same HASH result.

A Table Descriptor of type IC Index look-up is defined. This Table Descriptor is used to point to the hash buckets, according to 2 bytes taken from the HASH result, which are Table Descriptors that perform look-up on the Key which is the input to the KeyGen HASH function. This Table Descriptor provides a way to distinguish between all possible KeyGen Keys that may fall into the same bucket. The Table Descriptor can be of type IC Look-up (See [Section 5.12.5.3,"Look-Up on Internal Context Fields"](#)), or from any other type that enable to look-up on the exact Key that the KeyGen extracted from the incoming frame and based on it the HASH is calculated.

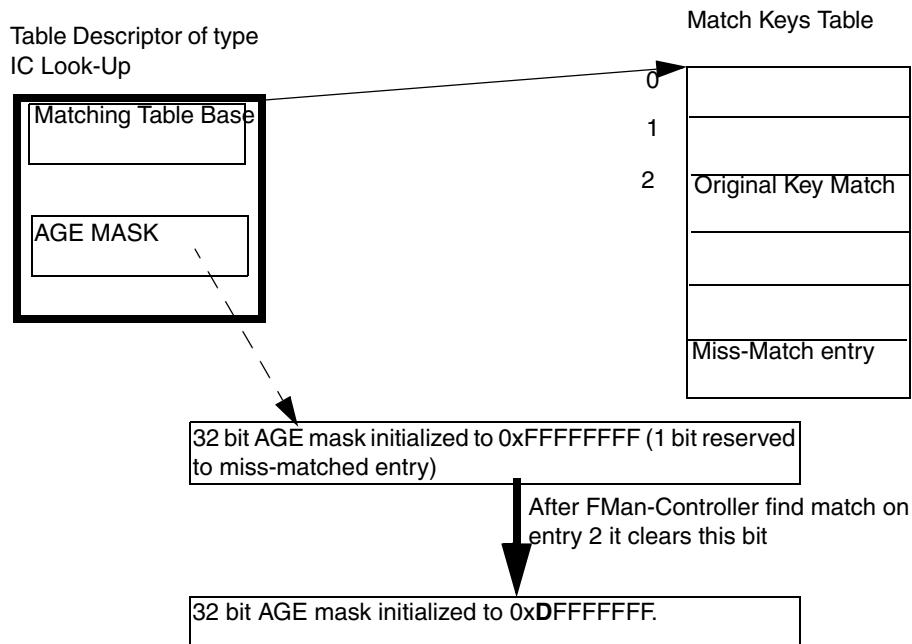
For example if the Look-up is performed on the MAC source field (key), and the KeyGen performs HASH on this KEY. Only 2 bytes are taken from this HASH result is used as an index (with a programmable MASK of 2 bytes with 4 lsb always cleared) from Base of Table Descriptors of Type MAC source address or alternatively IC look-up on the Key field itself.



**Figure 5-410. KeyGen/Custom Classifier HASH Look-Up Flow**

Aging mechanism is supported to enable application to monitor if a certain Key is accessed for certain period of time. This mechanism is implemented only on Lookup on internal context type (in order to

perform Key look-up). For this purpose a Operation Code of type “Lookup on internal context with Aging” is defined. Up to 31 keys that belongs to the same HASH bucket can have aging support (the table descriptor GMASK field is used for this purpose). [Figure 5-411](#) shows how this aging mechanism is working. For each Key including the last miss-match entry there is a corresponding bit in the AGE mask according to the location of the Key in the matching table. The application should set periodically the relevant age bit, using a special host command, and the FMan controller is responsible to clear this bit when a match is found on the specific key (or a miss-match which means clearing the miss-match bit according to its location). In this way the application can have monitor if a specific key has been accessed and if not it can be removed from the table.



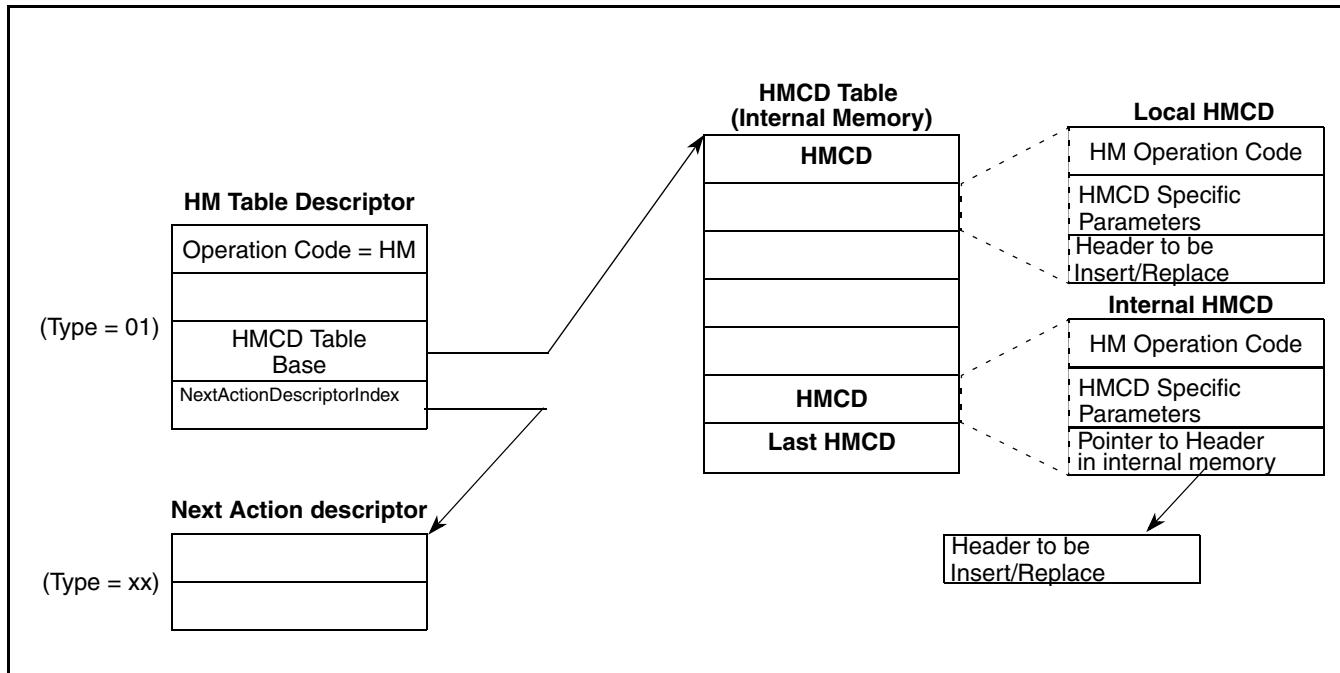
**Figure 5-411. IC Look-Up with Aging mechanism**

### 5.12.5.5 Header Manipulation (HM)

Header manipulation can be performed to change the incoming frame header for termination or interworking flow requirements. Header modification can be configured on a per-flow basis or for a user-determined group of flows.

An header manipulation table descriptor (HMTD) is used to perform header manipulation command descriptors (HMCDs). An HMTD can be located at first AD address or can be chosen as a result of custom classifier action descriptor (match or mismatch). The HMTD can be the last AD in the Custom Classifier (CC) flow. For more information see [Section 5.12.9, "Header Manipulation Table Descriptor \(HMTD\)"](#).

Every HMTD points to an HMCD table, which is placed in the internal memory (MURAM). The size of each HMCD table must not exceed 256 bytes. [Figure 5-412](#) shows a basic header manipulation flow.



**Figure 5-412. Basic Header Manipulation Flow**

The HMTD includes one or more commands (HMCDs). The commands are executed one by one from the first to the last one.

The HMCDs can be either internal or local. Internal HMCDs include a pointer to a memory location (internal memory) where the header to be inserted or replaced with is found. With local HMCDs, the HMCD itself contains the header to be inserted or replaced with. For more information see [Section 5.12.10, "Header Manipulation Command Descriptors \(HMCDs\)"](#).

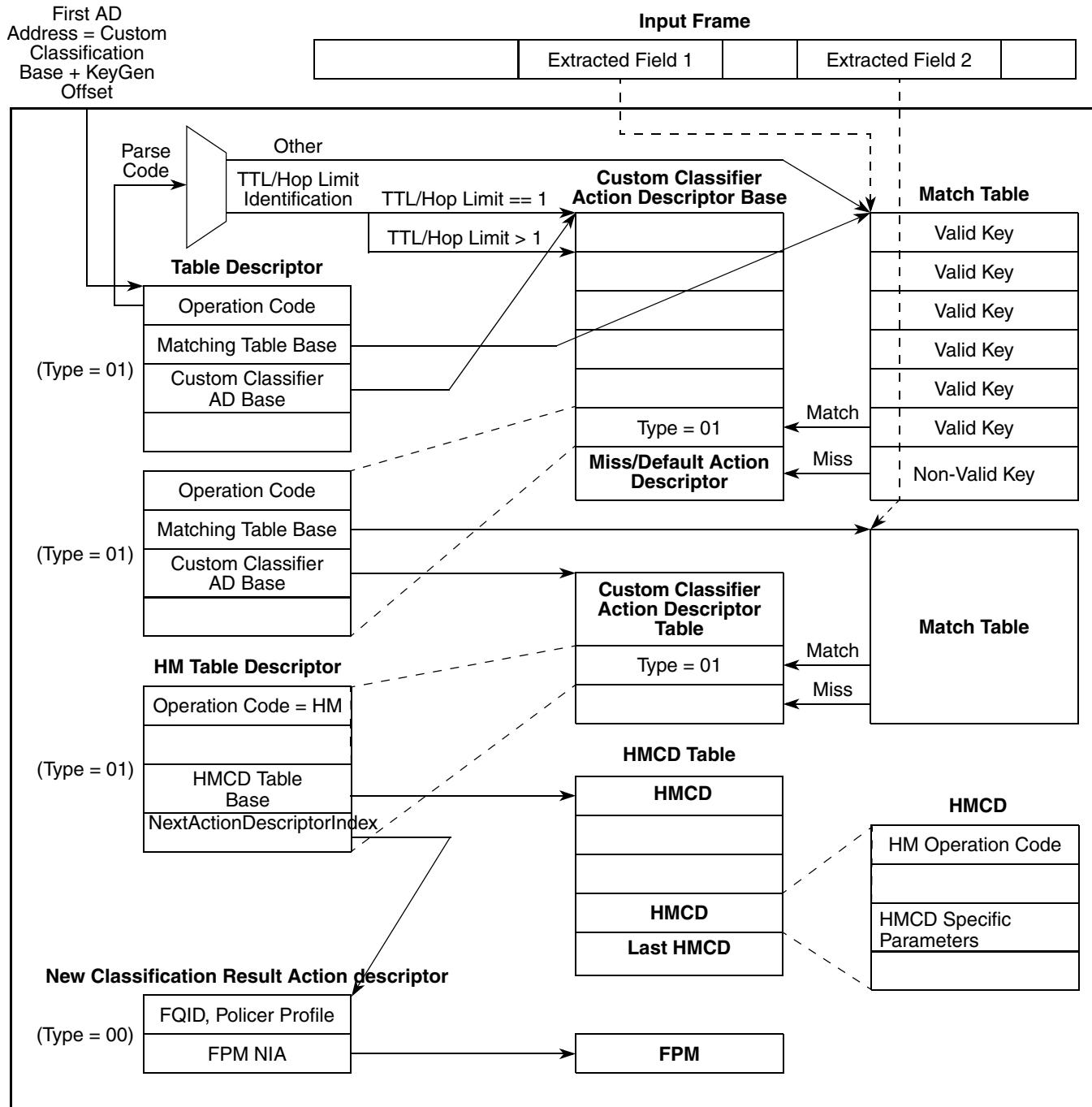
### 5.12.5.6 Statistics Gathering in Custom Classifier

The following Statistics Gathering features are available:

- Per-flow frame count (See StatisticCounter field in [Section 5.12.6.2, "Keep Classification Result"](#) and [Section 5.12.6.1, "New Classification Result"](#)).
- Frame and Byte count statistics (See [Section 5.12.18, "Statistics Gathering"](#))
- Statistics per Frame Length Range (See [Section 5.12.18, "Statistics Gathering"](#))
- Conditional Statistics that count the actual transmitted frames/bytes. (See [Section 5.12.18, "Statistics Gathering"](#))

### 5.12.5.7 Custom Classifier and HM Generic Flow

This figure shows an example of a custom classifier and HM generic flow.



**Figure 5-413. Custom Classifier and HM Generic Flow Example**

- Figure 5-413 describes an example flow of the custom classifier. Execution starts based on custom Classifier base+KeyGen offset which points to first action descriptor (AD). This AD is of type table AD (Type 01). The AD describes the first table. Extract field 1 is extracted out of the frame and

match against the list of keys. On the last key of the table a match is found. The corresponding AD is also a table descriptor which points to the next table. Extract field 2 is extracted and matched against the keys of the second table. There we have a match too however this time the next AD is header manipulation table descriptor (HMTD - Type01). The HMTD uses to perform header manipulation points to an HMCD table, which are placed in the internal or external memory. The HMCD points to the last AD, new classification result type (Type 0). Thus execution continues through the FPM.

## 5.12.6 Custom Classifier Action Descriptors (ADs)

There are three types of custom classifier action descriptors:

- New classification result
- Keep Classification result
- Table Descriptor—Need to initiate the next internal FMan controller action (according to the NIA).

### 5.12.6.1 New Classification Result

Need to initiate the next module (for example, the Policer), and the FQID (and optional Policier Profile) are taken from this AD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type=00															FQID[0-7]
2																FQID [8-23]
4	EBD	EBAD	FWD	NL	CWD	NENQ	—	VSPE								Policer Profile
6																NextActionDescriptorIndex
8	Extended Mode	Statistic Enable	NAD EN	—	FR	—	NO_C SPEN	OVOM								FPM NIA[0-7]
A																FPM NIA[8-23]
C																StatisticCounter / Next Frame Replicator Member Index
E																StatisticCounter

Figure 5-414. New Classification Result Action Descriptor (AD) (Type = 00)

**Table 5-464. New Classification Action Descriptor (Type = 00)**

Offset	Bits	Name	Description
0	0–1	Type	Action Descriptor Type. Set to 00. 00 New Classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are taken from this Action Descriptor. 01 Table Descriptor. Describes the matching table arguments. 10 Keep classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are not over written. 11 Reserved for dynamic update of custom classifier tables.
	2–7	RSPID	FMan_v3 devices: Relative Storage Profile ID
	8–31	FQID	Frame Queue ID.
4	0	EBD	FMan_v3 devices: Operational Mode bits. Bit description are the same as <a href="#">Section 5.4.3.1.1, "Frame Queue Descriptor (FQD) Context A."</a>
	1	EBAD	
	2	FWD	
	3	NL	
	4	CWD	
	5	NENQ	
	6	—	
	7	VSPE	
	8–15	Policer Profile	Policer Profile.
	16–31	NextActionDescriptorIndex	Index to the next Action Descriptor. The pointer to the next Action Descriptor is equal to this value multiplied by 16 (no base is added to this result). <b>Note:</b> Valid only when NADEN nextActionDescriptorEnable bit is set. Otherwise must be cleared. This field is supported in FMan_v3 or if IP acceleration is enabled. Otherwise must be cleared.

**Table 5-464. New Classification Action Descriptor (Type = 00)**

Offset	Bits	Name	Description
8	0	ExtendedModeEn	Extended Mode Enabled bit must be set if one of the following bits is set: ‘StatisticEn’, ‘NextActionDescriptorEnable’, ‘Frame Replicator’ . This field is supported in FMan_v3 or if IP acceleration is enabled. Otherwise must be cleared.
	1	StatisticEn	Valid only if ‘ExtendedModeEn’ bit is set and FR bit is clear. If set the Statistic counter is incremented for each access to this Custom Classifier Action Descriptor on event of received frame.This field is supported in FMan_v3 or if IP acceleration is enabled. Otherwise must be cleared.
	2	NextActionDescriptorEnable (NADEN)	Valid only if ‘ExtendedModeEn’ bit is set. If set the FMan-controller proceeds to the next Action Descriptor according to nextActionDescriptorIndex field.This field is supported in FMan_v3 or if IP acceleration is enabled. Otherwise must be cleared.
	3	Reserved	Must be cleared.
	4	FR	FMan_v3 devices: Frame Replicator. Valid only if ‘ExtendedModeEn’ bit is set. When set, indicates that this Action Descriptor is a part of Action Descriptors list for Frame Replication. In this case “Next Frame Replicator Member Index” in offset 0xC is used as an index of the next member Action Descriptor in the Frame Replicator list. When this bit is set, StatisticCounter is not supported.
	5	Reserved	Must be cleared.
	6	NO_CSPEN	FMan_v3 devices: No CC Virtual Storage Profile Enable Update. 0 - VSPE bit programmed in this CC AD overwrites the ICAD[VSPE] bit. 1 - The ICAD[VSPE] is preserved. In any case the RSPID is updated according to CC AD.
	7	OVOM	FMan_v3 devices: Override Operational Mode Bits. Controls how Operational Mode Bits in this Action Descriptor are reflected on the FPM bus. 0 - Operational Mode Bits are OR'd with the FPM bus. 1 - Operational Mode Bits are COPY'd into the FPM bus.
	8-31	FPM NIA[0-23]	FPM Next Invoked Action.

**Table 5-464. New Classification Action Descriptor (Type = 00)**

Offset	Bits	Name	Description
C	0-15	StatisticCounter / Next Frame Replicator Member Index	If FR bit is clear, this field is used as Statistic counter High bytes. Statistic counter is enabled only if 'statistic En' bit is set and FR bit is clear. This counter is incremented on any access to this Classification Action Descriptor. Must be cleared by CPU at initialization. FMan_v3 devices: If FR bit is set, this field is used as an index to the next Frame Replicator member Action Descriptor. The pointer to the next Action Descriptor in the Frame Replicator list is equal to this value multiplied by 16 (no base is added to this result). This field is supported in FMan_v3 or if IP acceleration is enabled. Otherwise must be cleared.
	16-31	StatisticCounter	If FR bit is clear, this field is used as Statistic counter Low bytes. Statistic counter is enabled only if 'statistic En' bit is set and FR bit is clear. This counter is incremented on any access to this Classification Action Descriptor. Must be cleared by CPU at initialization. If FR bit is set, this field must be clear. This field is supported in FMan_v3 or if IP acceleration is enabled. Otherwise must be cleared.

### 5.12.6.2 Keep Classification Result

Need to initiate the next module (for example, the Policer), and the FQID (and optional PP) are not overwritten.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type=10	PD								—						
2								—								
4	EBD	EBAD	FWD	NL	CWD	NENQ	—						—			
6															NextActionDescriptorIndex	
8	Extended Mode	Statistic Enable	NAD EN	—	FR	—	OVOM								FPM NIA[0-7]	
A															FPM NIA[8-23]	
C															StatisticCounter / Next Frame Replicator Member Index	
E															StatisticCounter / —	

**Figure 5-415. Keep Classification Action Descriptor (Type = 10)**

**Table 5-465. Keep Classification Action Descriptor (Type=10)**

Offset	Bits	Name	Description
0	0-1	Type	Action Descriptor Type. Set to 10. Same as description in <a href="#">Table 5-464</a> .
	2	PD	Policer Disabled. 0 - Policer Enable for this Action Descriptor. This means that the next module is the Policer and the Policer profile is taken from the KeyGen output. 1 - Policer Disable for this Action Descriptor. This means that the next module is NOT the Policer.
	3-31	Reserved	Must be cleared.
4	0	EBD	FMan_v3 devices: Operational Mode bits. Bit description are the same as <a href="#">Section 5.4.3.1.1,"Frame Queue Descriptor (FQD) Context A."</a>
	1	EBAD	
	2	FWD	
	3	NL	
	4	CWD	
	5	NENQ	
	6-7	—	
4	8-15	Reserved	Must be cleared.
	16-31	NextActionDescriptorIndex	See description in <a href="#">Table 5-464</a> .
8	0	ExtendedModeEn	See description in <a href="#">Table 5-464</a> .
	1	StatisticEn	See description in <a href="#">Table 5-464</a> .
	2	NextActionDescriptorEnable (NADEN)	See description in <a href="#">Table 5-464</a> .
	3	Reserved	Must be cleared.
	4	FR	See description in <a href="#">Table 5-464</a> .
	5-6	Reserved	Must be cleared.
	7	OVOM	FMan_v3 devices: Override Operational Mode Bits. Controls how Operational Mode Bits in this Action Descriptor are reflected on the FPM bus. 0 - Operational Mode Bits are OR'd with the FPM bus. 1 - Operational Mode Bits are COPY'd into the FPM bus.

**Table 5-465. Keep Classification Action Descriptor (Type=10)**

Offset	Bits	Name	Description
	8-31	FPM NIA[0-23]	FPM Next Invoked Action (see <a href="#">Section 5.4.4, "Next Invoked Action (NIA)"</a> ). If PD is zero, this NIA must be the Policer and since the Policer profile is taken from the previous internal context action descriptor the last 8 bits are don't care and should be cleared. If PD is zero, the Policer NIA should be with NIA[8] bit cleared, meaning that Policer profile is relative to the configuration registers of the port ID.
C	0-31	StatisticCounter/Next Frame Replicator Member Index	Same as description in Table <a href="#">Table 5-464</a> .

### 5.12.6.3 Table Descriptor

Need to initiate the next internal FMan controller action (according to the NIA).

This figure shows the table descriptor, type 01.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15					
0	Type=01	Key Length										Custom Classifier Action Descriptor Base [0-7]									
2	Custom Classifier Action Descriptor Base [8-23]																				
4	Match Table Entries No										LM	Match Table Pointer [0-6]									
6	Match Table Pointer [7-22]																				
8	—	OffsetFromParserResult			Offset																
A	—															OperationCode					
C	GMASK/AGE MASK																				
E																					

**Figure 5-416. Table Descriptor (Type = 01)**

This table describes the fields of the table descriptor, type 01.

**Table 5-466. Table Descriptor (Type = 01)**

Offset	Bits	Name	Description
0	0–1	Type	<p>Action Descriptor Type. Set to 01.</p> <p>00 New classification result. Need to initiate the next module (for example, Policier) and the FQID (and optional Policier Profile) are taken from this Action Descriptor.</p> <p>01 Table Descriptor. Describes the matching table arguments.</p> <p>10 Keep Classification result. Need to initiate the next module (for example, Policier) and the FQID (and optional Policier Profile) are not overwritten.</p> <p>11 Reserved for dynamic update of custom classifier tables.</p>
	2–7	Key Length	Key length. Indicates the size of the key minus 1. Possible values are 0 (1 byte) to 55 (56 bytes). Valid only for generic routines and shim headers. Set to 0 if not valid.
	8–31	Custom Classifier Action Descriptor Base	<p>Custom Classifier Action Descriptor FMan memory Base address. This table must be 16 byte aligned and its size is 16 times the number one greater than the number of entries in the matching table.</p> <p><b>Important:</b> User must define the default (mismatch) Action Descriptor entry and locate it at the end of the table. This entry is one entry in addition to the total number of entries in the matching table.</p> <p><b>Note:</b> If the Operation Code is IPv4 TTL or IPv6 Hop Limit there are only 2 entries for this table. See <a href="#">Section 5.12.5.2, "TTL/Hop Limit Equals One Identification,"</a> for description of the table.</p> <p><b>Note:</b> If the Operation Code is IC Index then the number of entries is a function of the GMASK. For example if GMASK=0x00000ff0 then number of Action Descriptor is up to 256.</p>
4	0–7	Match Table Entries No.	<p>Match Table Entries Number. One plus this number corresponds to the number of entries in the matching table. All the entries in this table are valid except the last entry, which is ignored. In case of a mismatch on the valid entries the result is the Action Descriptor that corresponds to the last entry.</p> <p>For example, setting the Match Table Entries No. to 7, means that user must allocate 8 entries for the matching table with last entry which is not valid (only 7 entries are valid) and need to allocate 8 entries for the custom classifier Action Descriptor. In case that there is no match on any of the first 7 entries in the matching table the selected custom classifier Action Descriptor is entry number 8.</p> <p>The maximum entries supported is 256. As such, the maximum value for this field is 255. Not valid if the Operation Code is IPv4 TTL or IPv6 Hop Limit or IC Index.</p> <p><b>Note:</b> A key and local mask pair is considered only one entry.</p>
	8	LM	<p>Local Mask. Not valid if the Operation Code is IPv4 TTL or IPv6 Hop Limit or IC Index.</p> <p>0 No local mask for this table. The table contains only key values.</p> <p>1 Local mask is in use for this table. Each key value in the table has a corresponding mask. The key and mask size for each entry are identical.</p> <p><b>Note:</b> Operation Codes that require initialization of GMASK may ONLY use LM enabled if the GMASK value is all 0xffffffff (meaning that GMASK is not functional). If GMASK is functional (some KEY bits are masked, others are not) the user must not use LM mode.</p>
	9–31	Match Table Pointer	Matching table internal pointer. Pointer to the internal FMan memory matching table base. Alignment of the matching table must be set to 16 bytes. Maximum size is (255*entry size) bytes. Refer to <a href="#">Section 5.12.5, "FMan Controller Functional Description,"</a> for description of the matching table. Not valid if the Operation Code is IPv4 TTL or IPv6 Hop Limit or IC Index.

**Table 5-466. Table Descriptor (Type = 01) (continued)**

Offset	Bits	Name	Description
8	0–2	Reserved	Must be set to 0.
	3–7	OffsetFromParserResult	Offset from ‘Parser Result Array’ in which the specific key offset is located. Must be cleared if not used. Valid values are 16 to 31. Valid only for generic routines when user specifies an offset of a specific entry from the Parser Result Array. See, <a href="#">Section 5.9, “Frame Manager—Parser,”</a> for more information.
	8–15	Offset	<p>Offset of key from start of frame or from internal frame context. Valid only in case of generic routines or in case of IC Index. Valid values are 0 to 255. Specify the offset of extracted key from the beginning of frame (if the offset from the Parser Result is 0) or from a specific Parser Result Array key offset field (if the offset from the Parser Result is greater than 0).</p> <p><b>Important note</b> - The offset is from the beginning of the internal buffer (which is always aligned to 256), in case that offset from the Parser Result is 0. This means that user must take into consideration the internal frame offset since the minimum value for the offset can be this value. If offset from the Parser Result is not 0, the internal frame offset is already reflected in the parser offset.</p> <p>In case of Generic 5 and Generic 6 Operation Code - specify the offset of extracted key from the beginning of internal frame context.</p> <p>For IC Index look-up specify the offset from the IC base from where the 2 bytes of the Index should be extracted. For example in case of HASH result field, the Offset must be programmed between 0x48-0x4E (HASH field in IC).</p>
	16–23	—	Reserved. Must be cleared.
	24–31	OperationCode	Operation Code. Specifies the specific/generic action type. See <a href="#">Table 5-467</a> for more details.
	C	GMASK/AGE MASK	<p>Global Mask. User specifies the Global Mask for Key size which are less than or equal to 4. For more detailed usage of this field refer to <a href="#">Table 5-416</a> and to the LM field description. The global mask must be aligned to the most significant bytes. For example, if looking for VLAN TCI, which is only 2 bytes long, the user must program the global mask aligned to the left (e.g. 0xffff0000).</p> <p>For Operation Code of type IC Index, the GMASK is used to filter the 2 bytes from IC, that is used as an Index to the Action Descriptor array. Only 2 least significant bytes of GMASK are valid with the 4 l.s.b always cleared (since each Action Descriptor size is 16 bytes). The maximum bits that may be set are 12 (0x0000FFF0), which enable up to 4K entries in the array.</p> <p>For Operation Code “Generic_6_Off_IC_AGE_MASK”this field is used as AGE Mask. User should initialize this field to 0xFFFFFFFF. Each bit corresponds to the KEY location in the matching table. On each match of KEY the FMan-Controller clear the corresponding bit in the AGE Mask. For more information see <a href="#">Section 5.12.5.4, “Using KeyGen HASH Result for Indexed Look-Up.”</a></p>

This table describes the operation codes.

**Table 5-467. \Operation Code Description**

Name	Operatio n Code <sup>1</sup>	Description
Eth MAC des.	0x00	Ethernet destination MAC 6 bytes. In this case each Match entry is of 8 bytes with 2 least significant bytes cleared.
Eth MAC src.	0x01	Ethernet source MAC 6 bytes. In this case each Match entry is of 8 bytes with 2 least significant bytes cleared.
Eth Type	0x02	Ethernet Type 2 bytes.
VLAN1 TCI	0x03	VLAN1 TCI (Tag Control Information) 2 bytes. User must also set GMASK to valid value, in order to distinguish between PRI/CFI/VLAN ID.
VLAN Last TCI	0x04	VLAN Last TCI (Tag Control Information) 2 bytes. User must also set GMASK to valid value, in order to distinguish between PRI/CFI/VLAN ID.
PPPoE PID	0x05	PPPoE PID 2 bytes.
MPLS	0x06	MPLS 4 bytes. User must also set GMASK to valid value.
MPLS Last	0x07	MPLS Last 4 bytes. User must also set GMASK to valid value.
IPv4 des.	0x08	IPv4 destination address (4 bytes).
IPv4 TOS	0x09	IPv4 TOS (1 byte). User must also set GMASK to valid value.
IPv4 Protocol	0x0A	IPv4 Protocol (1 byte).
IPv4 src.	0x0B	IPv4 source address (4 bytes).
IPv4 src. and des.	0x0C	IPv4 source and destination addresses (8 bytes)
IPv6 Class/Flow	0x0D	IPv6 Class/Flow (4 bytes). User must also set GMASK to valid value, in order to distinguish between class and flow
IPv6 NextHdr	0x0E	IPv6 NextHdr (1 byte).
IPv6 des.	0x0F	IPv6 destination address (16 bytes).
IPv6 src.	0x10	IPv6 source address (16 bytes).
GRE Protocol	0x11	GRE protocol Type (2 bytes).
MinEnc Protocol	0x12	MinEnc protocol (1 byte).
MinEnc IP des.	0x13	MinEnc IP destination address (4 bytes).
MinEnc IP src.	0x14	MinEnc IP source address (4 bytes).
MinEnc IP des. and src.	0x15	MinEnc IP destination and source addresses (8 bytes)
TunIPv4 des.	0x16	TunIPv4 destination address (4 bytes).
TunIPv4 TOS	0x17	TunIPv4 TOS (1 byte). User must also set GMASK to valid value
TunIPv4 Protocol	0x18	TunIPv4 Protocol (1 byte).
TunIPv4 src.	0x19	TunIPv4 source address (4 bytes).
TunIPv4 src. and des.	0x1A	TunIPv4 source and destination addresses (8 bytes).

**Table 5-467. \Operation Code Description (continued)**

Name	Operation Code <sup>1</sup>	Description
TunIPv6 Class/Flow	0x1B	TunIPv6 Class/Flow (4 bytes). User must also set GMASK to valid value, in order to distinguish between class and flow.
TunIPv6 NextHdr	0x1C	TunIPv6 NextHdr (1 byte).
TunIPv6 des.	0x1D	TunIPv6 destination address (16 bytes).
TunIPv6 src.	0x1E	TunIPv6 source address (16 bytes).
Layer 4 src. port	0x1F	Layer 4 (TCP or UDP) source port (2 bytes).
Layer 4 des. port	0x20	Layer 4 (TCP or UDP) destination port (2 bytes).
Layer 4 src. and des. ports	0x21	Layer 4 (TCP or UDP) source and destination ports (4 bytes).
Shim1	0x22	Shim 1. Key length should be programmed in the Action Descriptor.
Shim2	0x23	Shim 2. Key length should be programmed in the Action Descriptor.
IPPidOffset	0x24	Last IP Protocol ID Offset. Key length should be programmed in the Action Descriptor. In case last IP Protocol ID Offset Key length must be set to 0.
Generic_1_Off_OffsetParserResult_GMASK	0x25	Generic action 1. Key length should be programmed in the Action Descriptor. Offset and offsetFromParserResult must be valid. GMASK should be valid in case that key is less than or equal to 4 bytes, or set to 0xffffffff in case that key length is bigger than 4 bytes.
Generic_2_OffsetParserResult_GMASK	0x26	Generic action 2. Key length should be programmed in the Action Descriptor. offsetFromParserResult must be valid. GMASK should be valid in case that key is less than or equal to 4 bytes, or set to 0xffffffff in case that key length is bigger than 4 bytes.
Generic_3_Off	0x27	Generic action 3. Key length should be programmed in the Action Descriptor. Offset must be valid.
Generic_4_Off_GMASK	0x28	Generic action 4. Key length should be programmed in the Action Descriptor. Key size must be less than or equal to 4 bytes. Offset must be valid. GMASK should be valid.
IPv4 TTL	0x29	IPv4 TTL identification. If this field value equals to one, then the selected custom classifier Action Descriptor is taken from the custom classifier Action Descriptor Base address. Otherwise (TTL>1) the selected custom classifier Action Descriptor is taken from custom classifier Action Descriptor Base address plus 16 bytes.
IPv6 Hop Limit	0x2A	IPv6 Hop Limit identification. If this field value equals to one, then the selected custom classifier Action Descriptor is taken from the custom classifier Action Descriptor Base address. Otherwise (Hop Limit>1) the selected custom classifier Action Descriptor is taken from custom classifier Action Descriptor Base address plus 16 bytes.
Generic_5_Off_IC_GMASK	0x2B	This feature is not available on P4080 rev1. Generic action 5. The key is extracted from the internal frame context and not from the frame content. Key length should be programmed in the Action Descriptor. Offset must be valid. GMASK should be valid in case that key is less than or equal to 4 bytes, or set to 0xffffffff in case that key length is bigger than 4 bytes. Example for possible values for the offset could be 0x50 for the beginning of the KeyGen extracted KEY or 0x20 for the beginning of the parser results in-order to identify the layer x code result.

**Table 5-467. \Operation Code Description (continued)**

Name	Operation Code <sup>1</sup>	Description
IC Index	0x2C	This feature is not available on P4080 rev1. IC Index Look-Up. For more information see <a href="#">Section 5.12.5.4, "Using KeyGen HASH Result for Indexed Look-Up."</a> User must program the Action Descriptor Base Address, the GMASK, the offset and the Parse-Code itself. All other fields are irrelevant. The GMASK is used to filter the 2 bytes from IC that is used as an Index to the Action Descriptor array. Only 2 least significant bytes of GMASK are valid with the 4 l.s.b always cleared. The maximum bits that may be set are 12 (0x0000FFF0), which enable up to 4K entries of Action Descriptor array.
Generic_6_Off_IC_AGE_MASK	0x2D	This feature is not available on P4080 rev1. Generic action 6. The key is extracted from the internal frame context and not from the frame content. Key length should be programmed in the Action Descriptor. Offset must be valid. AGE MASK (GMASK) is used for supporting Aging mechanism. See <a href="#">Section 5.12.5.4, "Using KeyGen HASH Result for Indexed Look-Up."</a> Example for possible values for the offset could be 0x50 for the beginning of the KeyGen extracted KEY.
CAPWAP Fragmentation Condition Check	0x2E	Next Generation CAPWAP Fragmentation Condition Check. See <a href="#">Table 5-534</a> .
CAPWAP Manipulation	0x2F	Next Generation CAPWAP Manipulation Table Descriptor. See <a href="#">Table 5-537</a> .
CAPWAP Reassembly	0x30	Next Generation CAPWAP Reassembly Table Descriptor. See <a href="#">Table 5-535</a> .
CAPWAP Fragmentation	0x33	Next Generation CAPWAP fragmentation Table Descriptor. See <a href="#">Table 5-536</a> .
IP Manipulation	0x34	IP Manipulation. For more information see <a href="#">Table 5-529</a> and <a href="#">Figure 5-490</a> .
HMTD	0x35	Header Manipulation Table Descriptor (HMTD). For more information see <a href="#">Section 5.12.9, "Header Manipulation Table Descriptor (HMTD)"</a> .
STD	0x36	Statistics Table Descriptor (STD). For more information See <a href="#">Section 5.12.18, "Statistics Gathering"</a> .
IP Fragmentation	0x74	IP Fragmentation. For more information, see <a href="#">Figure 5-491</a> and <a href="#">Table 5-530</a> .
Frame Replicator	0x75	Frame Replicator. For more information, see <a href="#">Section 5.12.20, "Frame Replicator Support"</a> .
IP DSCP	0x76	IP DSCP. User should locate the DSCP value in the 6 lsb of each lookup entry. 2 msb of the lookup entry should be cleared. User should set GMASK to a valid value.
IP Reassembly	0xB4	IP Reassembly. For more information, see <a href="#">Figure 5-492</a> and <a href="#">Table 5-531</a> . "IP Reassembly Table Descriptor."
IPsec Manipulation	0xF4	IPsec Manipulation table descriptor. See <a href="#">Figure 5-493</a> .

**Note:**

<sup>1</sup> Local Mask is enabled by setting LM bit in the Custom Classifier Table Descriptor. For more details, see [Table 5-466](#).

## 5.12.7 First Custom Classifier Action Descriptor Location

The first custom classifier Action Descriptor is used as a root for a search tree. The search tree is built according to the protocol headers that the user must identify. The first Action Descriptor is accessed by the offset generated by the KeyGen from a base address which is programmed in the BMI programming

model. Refer to [Section 5.10.3.12.12, "KeyGen Scheme Entry Custom Classifier Bit Select Register \(AWR20\\_RFMODE\\_FMKG\\_SE\\_CCBS\)"](#), and [Section 5.10.3.12.1, "KeyGen Scheme Entry MODE Register \(AWR1\\_RFMODE\\_FMKG\\_SE\\_MODE\)"](#), for more information about the calculation of the Custom Classifier offset. The base address can be different per port due to the host memory partitioning but must be 256 byte aligned. The custom classifier base address is set in the BMI FMBM\_RCCB registers per MAC receive port or in the FMBM\_OCCB registers for offline parsing.

### 5.12.8 Custom Classifier Search Tree Example

This is an example of a custom classifier search tree for the following look up scenario: MAC destination address, VLAN1 TCI field and IPv4 destination address.

The first custom classifier Action Descriptor, which is from table descriptor type with Ethernet MAC destination Operation Code, is accessed using the custom classifier base and KeyGen offset. After the first match is found the corresponding custom classifier Action Descriptor entry of table descriptor type (the entry with the same offset as the matching entry) is taken from the custom classifier Action Descriptor base with VLAN1 TCI Operation Code and GMASK entry valid. After the second match is found the corresponding custom classifier Action Descriptor entry of table descriptor type (the entry with the same offset as the matching entry) is taken from the custom classifier Action Descriptor base with IP destination address Operation Code. This third look up results with custom classifier Action Descriptor entry with new classification result, where the FMan controller updates the internal context with this result (in the Action Descriptor field) and initiates the next FPM module according to the NIA field.

### 5.12.9 Header Manipulation Table Descriptor (HMTD)

Header Manipulation Table Descriptor (HMTD) is valid if:

- AD[Type] = 01
- AD[OperationCode] = 0x35 (HM)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type=01			—			—	PAHM	NADE N	—	—	—	—	—	—	
2	—			—							—					
4																Internal HMCD Table Base Pointer
6																Internal HMCD Table Base Pointer (cont)
8																NextActionDescriptorIndex
A			—													OperationCode = 0x35 (HM)
C									—							
E									—							

Figure 5-417. HM Table Descriptor (Type = 01)

This table describes the fields of the HM table descriptor.

**Table 5-468. HM Table Descriptor (Type = 01)**

Offset	Bits	Name	Description
0	0–1	Type	Action Descriptor Type. Set to 01.
	2–8	—	Reserved. Must be cleared.
9	PAHM	Parse after HM. If this bit is set, the parser is triggered after all HMCDs have been performed. When the parser stops, <ul style="list-style-type: none"><li>• If the NADEN==1, the next AD (or TD) in the CC flow will be as it is determined in the NextActionDescriptorIndex field.</li><li>• If the NADEN==0, the execution continues through the FPM according to the NIA.</li></ul> Notes <ul style="list-style-type: none"><li>• This bit can be set by the user only if the parse tree starting point (the lower header layer of the frame) is not changed by the HMCDs. It is the user's responsibility to ensure this restriction. For more information refer to <a href="#">Section 5.12.12, "Parse After Header Manipulation."</a></li></ul>	
	NADEN	NextActionDescriptorEnable (NADEN) 0 This is the last Action Descriptor. The next internal FMan controller action will be initiated according to the current NIA. (The FQID is not overwritten). 1 The FMan-controller proceeds to the next Action Descriptor (or Table Descriptor) according to NextActionDescriptorIndex field.	
	11–15	—	Reserved. Must be cleared.
	2	0–7	Reserved. Must be cleared.
	0x3	2 bytes	Reserved. Must be cleared.
		3 bytes	Internal HMCD Base Pointer Pointer to the HMCD table which is placed in the internal memory (MURAM). The size HMCD table must not exceed 256 bytes. This base must be eight byte aligned.
0x8	0–15	NextActionDescriptorIndex	Index to the next Action Descriptor. The pointer to the next Action Descriptor is equal to this value multiplied by 16 (no base is added to this result).
	16–23	—	Reserved. Must be cleared.
	24–31	OperationCode	Operation Code. Must be set to 0x35. See <a href="#">Table 5-467</a>
0xC	0–31	—	Reserved. Must be cleared.

## 5.12.10 Header Manipulation Command Descriptors (HMCDs)

Header manipulation descriptors are used to perform the following operations on the frame header.

- Remove
- Insert
- Replace
- Protocol Specific Commands

The user invokes a user programmable number of HMCDs, subject to some restrictions which are described in [Section 5.12.11, "Restrictions on the Usage of HMCDs."](#)

The length of the Header Manipulation Command Descriptor is variable and is determined by the header Manipulation Opcode. The first byte of the HMCD contains an opcode that distinguishes the operation being performed on the frame.

HMCDs can be either internal or local. Internal HMCDs include a pointer to a memory location (internal memory) where the header to be inserted or replaced with is found. With local HMCDs, the HMCD itself contains the header to be inserted or replaced with.

[Figure 5-418](#) shows a general header manipulation command descriptor.



**Figure 5-418. General Header Manipulation Command Descriptor (HMCD)**

[Table 5-469](#) gives the HMCD descriptions.

**Table 5-469. Header Manipulation Command Descriptor OPCODES**

HdrMan OPCODE	Description	Size (Bytes)
0x00	Reserved	
0x01	Header Removal Command Descriptor	4
0x02	Local Header Insert Command Descriptor	user programmable <sup>1</sup>
0x03	Internal Header Insert Command Descriptor	8
0x04	Reserved	
0x05	Local Header Replace Command Descriptor	user programmable <sup>2</sup>
0x06	Internal Header Replace Command Descriptor	8
0x07	Reserved	
0x08	Protocol Specific Header Removal Command Descriptor	4
0x09	Internal Protocol Specific Header Insert Command Descriptor	8
0x0A	Reserved	
0x0B	VLAN Priority Update Command Descriptor	4/8
0x0C	Local IPv4 Update Command Descriptor	4/8/12/16
0x0D	Reserved	
0x0E	Local TCP/UDP Update Command Descriptor	8
0x0F	Reserved	
0x10	Local IPv6 Update Command Descriptor	4/20/36
0x11	Reserved	
0x12	Internal IP Header Replace Command Descriptor	12

**Table 5-469. Header Manipulation Command Descriptor OPCODES (continued)**

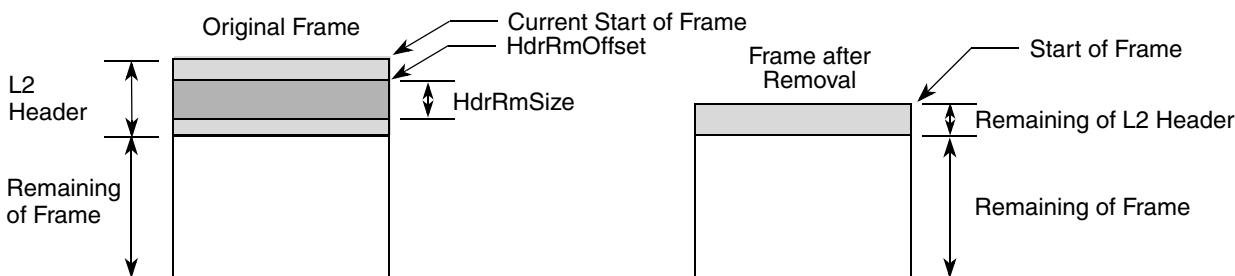
HdrMan OPCODE	Description	Size (Bytes)
0x14	UDP/TCP Checksum Calculation Command Descriptor (FMan_v3 only)	4
0x15	Remove header till known parser result. (FMan_v3 only) See <a href="#">Table 5-484</a> .	4
0x16	Local Insert UDP or UDP-Lite header. (FMan_v3 only) See <a href="#">Table 5-485</a> .	12
0x17	Local Insert L3 (IPv4 or IPv6) header. (FMan_v3 only) See <a href="#">Table 5-486</a> .	user programmable
0x18	Remove/Local Insert CAPWAP header. (FMan_v3 only) See <a href="#">Table 5-487</a> .	user programmable
0x19	Replace Field in Header. See <a href="#">Section 5.12.10.18, "Replace Field in Header Command Descriptor</a>	8
Rest	—	

<sup>1</sup> This HMCD must be padded to a size which is multiple of four bytes.

<sup>2</sup> This HMCD must be padded to a size which is multiple of four bytes.

### 5.12.10.1 Header Removal Command Descriptor

This descriptor is used to remove header fields (up to L3 header) from the frame. The size of this HMCD is 4 bytes.



**Figure 5-419. Header Removal Command**

[Figure 5-420](#) shows the header removal command descriptor.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0															
Offset + 2															

**Figure 5-420. Header Removal Command Descriptor**

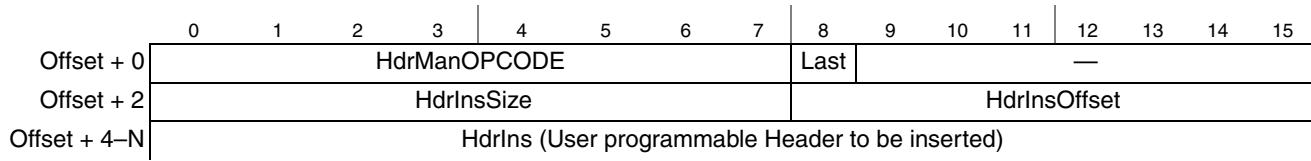
[Table 5-470](#) describes the header removal command descriptor fields.

**Table 5-470. Header Removal Command Descriptor Field Descriptions**

Offset	Bit	Name	Description
Base+ 0x0	0–7	HdrMan OPCODE	Opcode = 0x01
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9–15	Reserved	Clear to zero.
Base _0x2	0–7	HdrRmSize	Size of header, in bytes, to be removed from the frame.
	8–15	HdrRmOff	Offset from the beginning of the frame in bytes, where the header is being removed. This offset is relative to the beginning of the frame after all header manipulations that took place before the invocation of this HMCD.

### 5.12.10.2 Local Header Insert Command Descriptor

This descriptor is used to insert header fields (up to L3 header) into the frame. The size of this HMCD is user programmable. [Figure 5-421](#) shows the local header insert command descriptor.



**Figure 5-421. Local Header Insert Command Descriptor**

[Table 5-471](#) describes the local header insert command descriptor. fields.

**Table 5-471. Local Header Insert Command Descriptor Field Descriptions**

Offset	Bit	Name	Description
Base+ 0x0	0–7	HdrMan OPCODE	Opcode = 0x02
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9–15	—	Reserved. Bits must be cleared.
Base _0x2	0–7	HdrInsSize	Size of header, in bytes, to be inserted in frame. In the process of header manipulation, the frame must not increase in size more than FOF bytes beyond the size of the original incoming frame.
	8–15	HdrInsOff	Offset from the beginning of the frame in bytes, where the header is being inserted. This offset is relative to the beginning of the frame after all header manipulations that took place before the invocation of this HMCD.
Base+0x4–Base+0xN	8–15	HdrIns	User programmable Header to be inserted.

### 5.12.10.3 Internal Header Insert Command Descriptor

This descriptor is used to insert header fields (up to L3 header) into the frame. The size of this HMCD is 8 bytes. With this HMCD the header is fetched from internal multiuser RAM. [Figure 5-422](#) shows the internal header insert command descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	HdrManOPCODE	Last	—													
Offset + 2	HdrInsSize		HdrInsOffset													
Offset + 4	—		IntHdrInsPtr													
Offset + 6	IntHdrInsPtr															

**Figure 5-422. Internal Header Insert Command Descriptor**

[Table 5-472](#) describes the internal header insert command descriptor. fields.

**Table 5-472. Internal Header Insert Command Descriptor Field Descriptions**

Offset	Bit	Name	Description
Base+ 0x0	0–7	HdrMan OPCODE	Opcode = 0x03
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9–15	—	Reserved. Bits must be cleared.
Base _0x2	0–7	HdrInsSize	Size of header, in bytes, to be inserted in frame. In the process of header manipulation, the frame must not increase in size more than FOF bytes beyond the size of the original incoming frame.
	8–15	HdrInsOff	Offset from the beginning of the frame in bytes, where the header is being inserted. This offset is relative to the beginning of the frame after all header manipulations that took place before the invocation of this HMCD.
Base + 0x4	0–7	—	Reserved. Bits must be cleared.
	8–15	IntHsrlnsPtr	Pointer to internal multiuser RAM memory location for header to be inserted.
Base + 0x6	0–15		

### 5.12.10.4 Local Header Replace Command Descriptor

This descriptor is used to replace header fields (up to L3 header) in the frame. The size of this HMCD is user programmable. [Figure 5-423](#) shows the local header replace command descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	HdrManOPCODE	Last	—													
Offset + 2	HdrRepSize		HdrRepOffset													
Offset + 4–N	HdrRep (User programmable Header to be inserted)															

**Figure 5-423. Local Header Replace Command Descriptor**

[Table 5-473](#) describes the local header replace command descriptor fields.

**Table 5-473. Local Header Replace Command Descriptor Field Descriptions**

Offset	Bit	Name	Description
Base+ 0x0	0–7	HdrMan OPCODE	Opcode = 0x05
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9–15	—	Reserved. Clear to zero.
Base _0x2	0–7	HdrRepSize	Size of header, in bytes, to be replaced in frame.
	8–15	HdrRepOffset	Offset from the beginning of the frame in bytes, where the header is being replaced. This offset is relative to the beginning of the frame after all header manipulations that took place before the invocation of this HMCD.
Base+0x4– Base+0xN	8–15	HdrRep	User programmable Header to be inserted.

### 5.12.10.5 Internal Header Replace Command Descriptor

This descriptor is used to replace header fields (up to L3 header) in the frame. The size of this HMCD is 8 bytes. With this HMCD the header is fetched from internal multiuser RAM. [Figure 5-424](#) shows the internal header replace command descriptor.



**Figure 5-424. Internal Header Replace Command Descriptor**

[Table 5-474](#) describes the internal header replace command descriptor fields.

**Table 5-474. Internal Header Replace Command Descriptor Description**

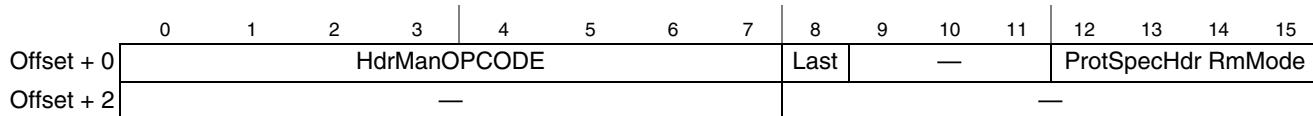
Offset	Bit	Name	Description
Base+ 0x0	0–7	HdrMan OPCODE	Opcode = 0x06
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9–15	Reserved	Reserved. Clear to zero.
Base _0x2	0–7	HdrRepSize	Size of header, in bytes, to be replaced in frame.
	8–15	HdrRepOff	Offset from the beginning of the frame in bytes, where the header is being replaced. This offset is relative to the beginning of the frame after all header manipulations that took place before the invocation of this HMCD.

**Table 5-474. Internal Header Replace Command Descriptor Description**

Offset	Bit	Name	Description
Base + 0x4	0–7	—	Reserved. Clear to zero.
	8–15	IntHdrRepPtr	Pointer to internal multiuser RAM memory location for the replacing header.
Base + 0x6	0–15		

### 5.12.10.6 Protocol Specific Header Removal Command Descriptor

This descriptor is used to remove a protocol specific header from the frame. This HMCD uses the parser results offsets so it can be invoked only after the invocation of the parser. The size of this HMCD is 4 bytes. This HMCD is used when the user cannot foresee exactly the size of the protocol specific header as in Ethernet/802.3 with or without QTags. [Figure 5-425](#) shows the protocol specific header removal command descriptor.



**Figure 5-425. Protocol Specific Header Removal Command Descriptor**

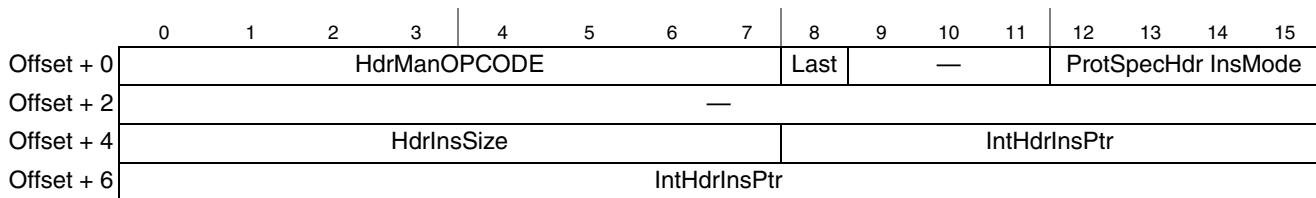
[Table 5-475](#) describes the protocol specific header removal command descriptor fields.

**Table 5-475. Protocol Specific Header Removal Command Descriptor Descriptions**

Offset	Bit	Name	Description																																																									
Base + 0	0–7	HdrManOPCODE	Opcode = 0x08																																																									
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor																																																									
	9–11		Reserved. Clear to zero.																																																									
	12–15	ProtSpecHdr RmMode	<p>Protocol specific header removal Mode</p> <p>0000 Remove Ethernet/802.3 MAC header.</p> <ul style="list-style-type: none"> <li>This HMCD is able to recognize Ethernet/802.3 MAC headers of the following type: Ethernet (EType) and 802.3 SNAP header with unlimited stacked QTags. This HMCD removes the Ethernet/802.3 MAC header until the header which follows the Ethernet/802.3 MAC header.</li> </ul> <table border="1"> <tr> <td colspan="2">QTag Format</td> <td colspan="2">TCI or VLAN Tag</td> <td>QTag</td> </tr> <tr> <td>Type=UserProg</td> <td>VPri</td> <td>CFI</td> <td>VID</td> <td></td> </tr> <tr> <td>2 Bytes</td> <td>3 Bits</td> <td>1 Bit</td> <td>12 Bits</td> <td></td> </tr> </table> <p>Ethernet/802.3 Frame Format with Two Stacked QTags</p> <table border="1"> <tr> <td>MACdst</td> <td>MACsrc</td> <td>QTag1</td> <td>QTag2</td> <td>Etype/Size</td> </tr> <tr> <td>6 Bytes</td> <td>6 Bytes</td> <td>4 Bytes</td> <td>4 Bytes</td> <td>2 Bytes</td> </tr> </table> <p>0001 Remove the stacked QTags. (Unlimited stacked QTags).</p> <p>0010 Remove Ethernet/802.3 MAC header + MPLS header.</p> <ul style="list-style-type: none"> <li>This HMCD removes the MPLS and Ethernet/802.3 MAC header until the L3 header (IPv4 or IPv6) which follows the MPLS header.</li> </ul> <table border="1"> <tr> <td>MPLS Label Format</td> <td>Label Value</td> <td>Exp</td> <td>S</td> <td>TTL</td> </tr> <tr> <td></td> <td>20 Bits</td> <td>3 Bits</td> <td>1 Bit</td> <td>8 Bits</td> </tr> </table> <p>IP over Ethernet Frame Format with Stacked QTags and multi MPLS labels.</p> <table border="1"> <tr> <td>MACdst</td> <td>MACsrc</td> <td>N*QTags</td> <td>MPLS Etype</td> <td>N*MPLS Labels</td> <td>IPv4/IPv6</td> </tr> <tr> <td>6 Bytes</td> <td>6 Bytes</td> <td>4*N Bytes</td> <td>2 Bytes</td> <td>4*N Bytes</td> <td></td> </tr> </table> <p>IP over Ethernet Frame Format with Stacked QTags.</p> <table border="1"> <tr> <td>MACdst</td> <td>MACsrc</td> <td>N*QTags</td> <td>IP Etype</td> <td>IPv4/IPv6</td> </tr> <tr> <td>6 Bytes</td> <td>6 Bytes</td> <td>4*N Bytes</td> <td>2 Bytes</td> <td></td> </tr> </table> <p>0011 Remove the MPLS header. (Unlimited MPLS labels).</p> <ul style="list-style-type: none"> <li>The MPLS Etype will be replaced by the following IP header Etype. (IPv4 or IPv6).</li> </ul> <p>0100–1111 Reserved</p>	QTag Format		TCI or VLAN Tag		QTag	Type=UserProg	VPri	CFI	VID		2 Bytes	3 Bits	1 Bit	12 Bits		MACdst	MACsrc	QTag1	QTag2	Etype/Size	6 Bytes	6 Bytes	4 Bytes	4 Bytes	2 Bytes	MPLS Label Format	Label Value	Exp	S	TTL		20 Bits	3 Bits	1 Bit	8 Bits	MACdst	MACsrc	N*QTags	MPLS Etype	N*MPLS Labels	IPv4/IPv6	6 Bytes	6 Bytes	4*N Bytes	2 Bytes	4*N Bytes		MACdst	MACsrc	N*QTags	IP Etype	IPv4/IPv6	6 Bytes	6 Bytes	4*N Bytes	2 Bytes	
QTag Format		TCI or VLAN Tag		QTag																																																								
Type=UserProg	VPri	CFI	VID																																																									
2 Bytes	3 Bits	1 Bit	12 Bits																																																									
MACdst	MACsrc	QTag1	QTag2	Etype/Size																																																								
6 Bytes	6 Bytes	4 Bytes	4 Bytes	2 Bytes																																																								
MPLS Label Format	Label Value	Exp	S	TTL																																																								
	20 Bits	3 Bits	1 Bit	8 Bits																																																								
MACdst	MACsrc	N*QTags	MPLS Etype	N*MPLS Labels	IPv4/IPv6																																																							
6 Bytes	6 Bytes	4*N Bytes	2 Bytes	4*N Bytes																																																								
MACdst	MACsrc	N*QTags	IP Etype	IPv4/IPv6																																																								
6 Bytes	6 Bytes	4*N Bytes	2 Bytes																																																									
Base + 2	—		Reserved. Clear to zero.																																																									

### 5.12.10.7 Internal Protocol Specific Header Insert Command Descriptor

This descriptor is used to insert a protocol specific header from the frame. This HMCD uses the parser results offsets so it can be invoked only after the invocation of the parser. The size of this HMCD is 8 bytes. This HMCD is used when the user cannot foresee exactly the size of the protocol specific header as in Ethernet/802.3 with or without QTags. [Figure 5-426](#) shows the internal protocol specific header insert command descriptor.



**Figure 5-426. Internal Protocol Specific Header Insert Command Descriptor**

**Table 5-476** describes the internal protocol specific header insert command descriptor fields.

**Table 5-476. Internal Protocol Specific Header Insert Command Descriptor Descriptions**

Offset	Bit	Name	Description																																
Base + 0	0–7	HdrManOPCODE	Opcode = 0x09																																
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor																																
	9–11		Reserved. Clear to zero.																																
	12–15	ProtSpecHdr InsMode	<p>Protocol specific header insert Mode 0000 Insert MPLS header. Single MPLS label or more.</p> <ul style="list-style-type: none"> <li>This HMCD is able to recognize Ethernet/802.3 MAC headers of the following type: Ethernet (EType) and 802.3 SNAP header with unlimited stacked QTags. This HMCD insert MPLS label(s) at the end of the L2 header. The user has to prepare the required header to be inserted in frame. i.e. MPLS Etype + N*MPLS Labels. The original IP</li> </ul> <table border="1"> <tr> <td>MPLS Label Format</td> <td>Label Value</td> <td>Exp</td> <td>S</td> <td>TTL</td> </tr> <tr> <td></td> <td>20 Bits</td> <td>3 Bits</td> <td>1 Bit</td> <td>8 Bits</td> </tr> </table> <p>IP over Ethernet Frame Format with Stacked QTags and multi MPLS labels.</p> <table border="1"> <tr> <td>MACdst</td> <td>MACsrc</td> <td>N*QTags</td> <td>MPLS Etype</td> <td>N*MPLS Labels</td> <td>IPv4/IPv6</td> </tr> <tr> <td>6 Bytes</td> <td>6 Bytes</td> <td>4*N Bytes</td> <td>2 Bytes</td> <td>4*N Bytes</td> <td></td> </tr> </table> <p>IP over Ethernet Frame Format with Stacked QTags.</p> <table border="1"> <tr> <td>MACdst</td> <td>MACsrc</td> <td>N*QTags</td> <td>IP Etype</td> <td>IPv4/IPv6</td> </tr> <tr> <td>6 Bytes</td> <td>6 Bytes</td> <td>4*N Bytes</td> <td>2 Bytes</td> <td></td> </tr> </table> <p>Etype (or MPLS Etype, if the frame already includes an MPLS header) is replaced by the new MPLS Etype and the N*MPLS labels are inserted right after it.</p> <p>0001 Insert and update of MPLS header. Single MPLS label or more.</p> <ul style="list-style-type: none"> <li>The user has to prepare the required header to be inserted in frame. i.e. MPLS Etype + N*MPLS Labels. The original IP Etype (or MPLS Etype, if the frame already includes an MPLS header) is replaced by the new MPLS Etype and the N*MPLS labels are inserted right after it. If the new MPLS header is the last one, the last MPLS label is updated according to the following L3 IP header. The last MPLS label = “2” for IPv6 and “0” for IPv4. The MPLS S bit is also automatically updated. (set to “1” for last MPLS header and set to “0” for non last MPLS header).</li> </ul> <p>0002–1111 Reserved</p>	MPLS Label Format	Label Value	Exp	S	TTL		20 Bits	3 Bits	1 Bit	8 Bits	MACdst	MACsrc	N*QTags	MPLS Etype	N*MPLS Labels	IPv4/IPv6	6 Bytes	6 Bytes	4*N Bytes	2 Bytes	4*N Bytes		MACdst	MACsrc	N*QTags	IP Etype	IPv4/IPv6	6 Bytes	6 Bytes	4*N Bytes	2 Bytes	
MPLS Label Format	Label Value	Exp	S	TTL																															
	20 Bits	3 Bits	1 Bit	8 Bits																															
MACdst	MACsrc	N*QTags	MPLS Etype	N*MPLS Labels	IPv4/IPv6																														
6 Bytes	6 Bytes	4*N Bytes	2 Bytes	4*N Bytes																															
MACdst	MACsrc	N*QTags	IP Etype	IPv4/IPv6																															
6 Bytes	6 Bytes	4*N Bytes	2 Bytes																																
Base + 0x2	0–15	—	Reserved. Clear to zero.																																
Base + 0x4	0–7	HdrlnsSize	Size of header, in bytes, to be inserted in frame. In the process of header manipulation, the frame must not increase in size more than FOF bytes beyond the size of the original incoming frame.																																
	8–15	IntHsrlnsPtr	Pointer to internal multiuser RAM memory location for header to be inserted.																																
Base + 0x6	0–15																																		

### 5.12.10.8 VLAN Priority Update Command Descriptor

This descriptor is used to update the VLAN priority field. This HMCD uses the parser results offsets so it can be invoked only after the invocation of the parser. The size of this HMCD is 4 or 8 bytes. See [Figure 5-427](#).

[Figure 5-427](#) shows the VLAN priority update command descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	HdrManOPCODE							Last	—	—	—	—	VPriHdr RepMode			
Offset + 2	—							—	—	—	—	—	Vpri Default Value			
Offset + 4 (Optional)	— (Optional)							IntHdrRepPtr (Optional)								
Offset + 6 (Optional)	IntHdrRepPtr (Optional)															

**Figure 5-427. VLAN Priority Update Command Descriptor**

[Table 5-477](#) describes the VLAN priority update command descriptor fields.

**Table 5-477. VLAN Priority Update Command Descriptor Descriptions**

Offset	Bit	Name	Description
Base + 0	0–7	HdrManOPCODE	Opcode = 0x0B
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9	—	Reserved. Clear to zero.
	10–12	—	Reserved. Clear to zero.
	13–15	VPriHdr RepMode	Protocol specific header replace Mode 000 Replace the Vpri field in outer most VLAN Tag with a default value (Vpri Default Value). Do not change the value of the VID portion of Q Tag. 001 DSCP to VLAN priority bits translation. Replace Vpri field in outer most VLAN Tag with value taken from IPv4 TOS or IPv6 TC field Mapping Table (6 bits Diffserv DSCP field). Do not change the value of the VID portion of Q Tag. See <a href="#">Section 5.12.10.8.1, "IP_DSCP_to_VPri_Table."</a> If IP header is not found in the frame a default value (Vpri Default Value) is used. 001–111 Reserved
Base + 2	0–12	—	Reserved. Clear to zero.
	13–15	Vpri Default Value	The Q Tag default value if the IP header is not found.
Base + 0x4 (Optional)	0–7	—	This field is included in this HMCD only if IntHdrRepPtr field is used. Reserved. Clear to zero.
	8–15	IntHdrRepPtr	This field is included in this HMCD only if it is used. If VPriHdrRepMode = 000, this field is not used If VPriHdrRepMode = 001, this field is pointer to the a 32 byte table IP_DSCP_to_VPri_Table in the internal multiuser RAM memory. See <a href="#">Section 5.12.10.8.1, "IP_DSCP_to_VPri_Table."</a> If VPriHdrRepMode = 001–111, this field is not used
Base + 0x6 (Optional)	0–15		

### 5.12.10.8.1 IP\_DSCP\_to\_VPri\_Table

This table is pointed by IntHdrRepPtr field as programmed in the VLAN Priority Update Command Descriptor. See [Section 5.12.10.8,"VLAN Priority Update Command Descriptor."](#)

The 6 DSCP bits (The six most significant bits of IPv4 TOS or IPv6 TC) are marked in the VPri field of the Q-Tag according to this translation table.

Offset <sup>1</sup>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	IP_DSCP_0x0				IP_DSCP_0x1				IP_DSCP_0x2				IP_DSCP_0x3			
Offset + 2		IP_DSCP_0x4				IP_DSCP_0x5				IP_DSCP_0x6			IP_DSCP_0x7			
....	...				...				...				...			
Offset + 1E		IP_DSCP_0x3C				IP_DSCP_0x3D			IP_DSCP_0x3E				IP_DSCP_0x3F			

**Figure 5-428. IP\_DSCP\_to\_VPri\_Table**

<sup>1</sup> Offset from IntHdrRepPtr

**Table 5-478. IP\_DSCP\_to\_VPri\_Table**

Offset	Bits	Name	Description	Initialized by
Offset + 0	0–3	IP_DSCP_0x0	If IP_DSCP field in IP header is equal to 0x0, VPri in Q-Tag is marked with this value. The MSBit is not used.	CPU
	4–7	IP_DSCP_0x1	If IP_DSCP field in IP header is equal to 0x1, VPri in Q-Tag is marked with this value. The MSBit is not used.	CPU
	8–11	IP_DSCP_0x2	If IP_DSCP field in IP header is equal to 0x2, VPri in Q-Tag is marked with this value. The MSBit is not used.	CPU
	12–15	IP_DSCP_0x3	If IP_DSCP field in IP header is equal to 0x3, VPri in Q-Tag is marked with this value. The MSBit is not used.	CPU
Offset + 2	0–3	IP_DSCP_0x4	If IP_DSCP field in IP header is equal to 0x4, VPri in Q-Tag is marked with this value. The MSBit is not used.	CPU
	4–7	IP_DSCP_0x5	If IP_DSCP field in IP header is equal to 0x5, VPri in Q-Tag is marked with this value. The MSBit is not used.	CPU
	8–11	IP_DSCP_0x6	If IP_DSCP field in IP header is equal to 0x6, VPri in Q-Tag is marked with this value. The MSBit is not used.	CPU
	12–15	IP_DSCP_0x7	If IP_DSCP field in IP header is equal to 0x7, VPri in Q-Tag is marked with this value. The MSBit is not used.	CPU
...			....	
Offset + 1F	12–15	IP_DSCP_0x3F	If IP_DSCP field in IP header is equal to 0x3F, VPri in Q-Tag is marked with this value. The MSBit is not used.	CPU

### 5.12.10.9 Local IPv4 Update Command Descriptor

This command descriptor is used to replace IPv4 fields in the IP header of a frame. It automatically generates the IP checksum and optionally can update the UDP/TCP checksum. If the incoming frame contains TCP or UDP (the original UDP checksum!= 0), the UDP/TCP checksum is recalculated based

on original checksum and the change in relevant header fields. This HMCD uses the parser results offsets so it can be invoked only after the invocation of the parser.

**Figure 5-429** shows the local IPv4 update command descriptor. If IPv4 header is not found, the frame is not affected by this HMCD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Offset + 0	HdrManOPCODE			Last	Reserved			—									
Offset + 2	IP TOS			IP ID mode	IPSrc	IPDSt	—	IP TOS mode	IP TTL								
Offset + 4 (Optional)	Parameter1 (Optional)																
Offset + 6 (Optional)	Parameter2 (Optional)																
Offset + 8 (Optional)	Parameter3 (Optional)																
Offset + a (Optional)																	
Offset + c (Optional)																	
Offset + e (Optional)																	

**Figure 5-429. Local IPv4 Update Command Descriptor**

**Table 5-479** describes the local IPv4 update command descriptor fields.

**Table 5-479. Local IPv4 Update Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0–7	HdrManOPCODE	Opcode = 0x0C
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9–14	—	Reserved. Bits must be cleared.
	15	—	Reserved. Bits must be cleared.
Base+ 0x2	0–7	IP TOS	If IP TOS mode=1 then the original IP TOS (8 bits of TOS field in the frame) will be replaced by this field. Otherwise must be cleared.
	8	IP ID mode	0 No Change in IP ID. 1 The IP ID in the incoming frame, is replaced by the IP ID field (See Parameter1). The copied IP ID field (in the internal memory) will be incremented by 1.
	9	IP Src	0 IP Src address remain unchanged. 1 IP Src address should be replaced by Parameter1 or Parameter2.

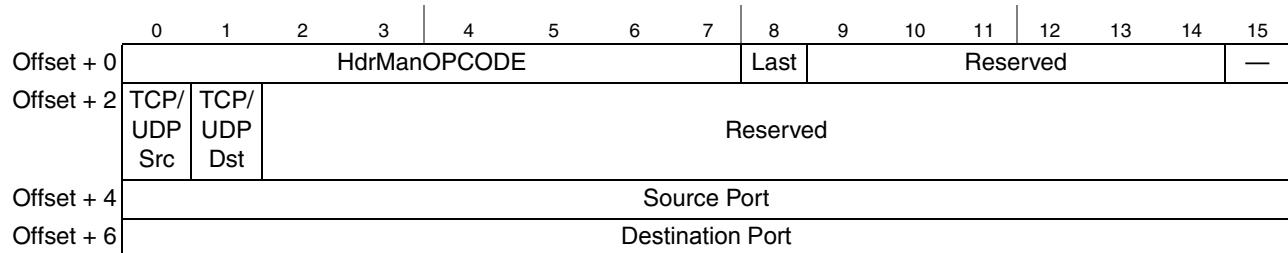
**Table 5-479. Local IPv4 Update Command Descriptor (continued)**

Offset	Bit	Name	Description
	10	IP Dst	0 IP Dst address remain unchanged. 1 IP Dst address should be replaced by Parameter1, 2 or 3.
	11–12	—	Reserved. Bits must be cleared.
	13–14	IP TOS mode	0 No Change in IP TOS. 1 The 8 bits of the IP TOS in the incoming frame, is replaced by the 8 bits of IP TOS field of this HMCD. 2 Reserved 3 Reserved.
	15	IP TTL	0 No change in The Time to Live field. 1 Time to Live field will be decremented by 1. Note: If the TTL==0, it is not decremented by 1.
Base + 0x4		Parameter1 (optional)	This field is included in this HMCD only if it is used. If IP ID = 1, this field (bits 8–31) is used as a pointer to internal multiuser RAM memory for IP ID fields to be inserted. (bits 0–7 are reserved bits in this case) If IP ID = 0 <ul style="list-style-type: none"> <li>If IP Src = 0 and IP Dst = 0, this field is not used</li> <li>If IP Src = 0 and IP Dst = 1, this field is used for IP Dst value</li> <li>If IP Src = 1 and IP Dst = 0, this field is used for IP Src value</li> <li>If IP Src = 1 and IP Dst = 1, this field is used for IP Src value</li> </ul>
Base + 0x8		Parameter2 (optional)	This field is included in this HMCD only if it is used. If IP Src = 0 and IP Dst = 0, this field is not used If IP ID = 0 <ul style="list-style-type: none"> <li>If IP Src = 0 and IP Dst = 0, this field is not used</li> <li>If IP Src = 0 and IP Dst = 1, this field is not used</li> <li>If IP Src = 1 and IP Dst = 0, this field is not used</li> <li>If IP Src = 1 and IP Dst = 1, this field is used for IP Dst value</li> </ul> If IP ID = 1 <ul style="list-style-type: none"> <li>If IP Src = 0 or IP Dst = 0, this field is not used</li> <li>If IP Src = 0 and IP Dst = 1, this field is used for IP Dst value</li> <li>If IP Src = 1 and IP Dst = 0, this field is used for IP Src value</li> <li>If IP Src = 1 and IP Dst = 1, this field is used for IP Src value</li> </ul>
Base + 0xc		Parameter3 (optional)	This field is included in this HMCD only if it is used. If IP ID = 0, this field is not used If IP ID = 1 <ul style="list-style-type: none"> <li>If IP Src = 0 and IP Dst = 0, this field is not used</li> <li>If IP Src = 0 and IP Dst = 1, this field is not used</li> <li>If IP Src = 1 and IP Dst = 0, this field is not used</li> <li>If IP Src = 1 and IP Dst = 1, this field is used for IP Dst value</li> </ul>

### 5.12.10.10 Local TCP/UDP Update Command Descriptor

This command descriptor is used to replace ports in the TCP/UDP header of a frame. It optionally can update the TCP/UDP checksum. For TCP or UDP with original UDP checksum!= 0, the UDP/TCP checksum is recalculated based on original checksum and the change in relevant header fields. This HMCD uses the parser results offsets so it can be invoked only after the invocation of the parser.

[Figure 5-430](#) shows the internal TCP/UDP update command descriptor. If TCP/UDP header is not found, the frame is not affected by this HMCD. The size of this HMCD is 8 bytes.



**Figure 5-430. Local TCP/UDP Update Command Descriptor**

[Table 5-480](#) describes the local TCP/UDP update command descriptor fields.

**Table 5-480. Local TCP/UDP Update Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0-7	HdrManOPCODE	Opcode = 0x0E
	8	Last	0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9-14	—	Reserved. Bits must be cleared.
	15	—	Reserved. Bits must be cleared.
Base+ 0x2	0	TCP/UDP Src	0 TCP/UDP Src port remain unchanged. 1 TCP/UDP Src port should be replaced.
	1	TCP/UDP Dst	0 TCP/UDP Dst port remain unchanged. 1 TCP/UDP Dst port should be replaced.
	2-15	—	Reserved. Bits must be cleared.
Base + 0x4	0-15	Source Port	TCP/UDP Source port.
Base + 0x6	0-15	Destination Port	TCP/UDP Destination port.

### 5.12.10.11 Local IPv6 Update Command Descriptor

This command descriptor is used to replace IPv6 fields/addresses in the IP header of a frame. It optionally can update the UDP/TCP checksum. If the incoming frame contains TCP or UDP (the original UDP checksum!= 0), the UDP/TCP checksum is recalculated based on original checksum and the change in relevant header fields. This HMCD uses the parser results offsets so it can be invoked only after the invocation of the parser.

[Figure 5-431](#) shows the local IPv6 update command descriptor. If IPv6 header is not found, the frame is not affected by this HMCD.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15						
Offset + 0	HdrManOPCODE								Last	Reserved				—								
Offset + 2	IP Traffic Class								IPSrc	IPDst	Reserved		IP Traffic Class mode		IP Hop Limit							
Offset + 4 (Optional)	Parameter1 (Optional)																					
Offset + 13 (Optional)	Parameter2 (Optional)																					
Offset + 14 (Optional)																						
Offset + 23																						

**Figure 5-431. Local IPv6 Update Command Descriptor**

Table 5-481 describes the local IPv6 update command descriptor fields.

**Table 5-481. Local IPv6 Update Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0–7	HdrManOPCODE	Opcode = 0x10
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9–14	—	Reserved. Bits must be cleared.
	15	—	Reserved. Bits must be cleared.
Base+ 0x2	0–7	IP Traffic Class	If Traffic Class mode=1 then the original Traffic Class (8 bits of Traffic Class field in the frame) will be replace by this field. Otherwise must be cleared.
	8	IP Src	0 IP Src address remain unchanged. 1 IP Src address should be replaced by Parameter1.
	9	IP Dst	0 IP Dst address remain unchanged. 1 IP Dst address should be replaced by Parameter1 or Parameter2.
	10–12	—	Reserved. Bits must be cleared.
	13–14	IP Traffic Class mode	0 No Change in IP Traffic Class mode. 1 The 8 bits of the Traffic Class in the incoming frame, is replaced by the 8 bits of IPv6 Traffic Class field of this HMCD 2 Reserved. 3 Reserved.
	15	IPv6 HopLimit	0 No change in the Time to Hop Limit field. 1 Time to Hop Limit field will be decremented by 1. Note: If the HOP Limit ==0, it is not decremented by 1.
Base + 0x4 (optional)		Parameter1 (optional)	This field is included in this HMCD only if it is used. If IP Src = 0 and IP Dst = 0, this field is not used If IP Src = 0 and IP Dst = 1, this field is used for IP Dst value If IP Src = 1 and IP Dst = 0, this field is used for IP Src value If IP Src = 1 and IP Dst = 1, this field is used for IP Src value

**Table 5-481. Local IPv6 Update Command Descriptor (continued)**

Offset	Bit	Name	Description
Base + 0x14 (optional)		Parameter2 (optional)	This field is included in this HMCD only if it is used. If IP Src = 0 and IP Dst = 0, this field is not used If IP Src = 0 and IP Dst = 1, this field is not used If IP Src = 1 and IP Dst = 0, this field is not used If IP Src = 1 and IP Dst = 1, this field is used for IP Dst value

### 5.12.10.12 Internal IP Header Replace Command Descriptor

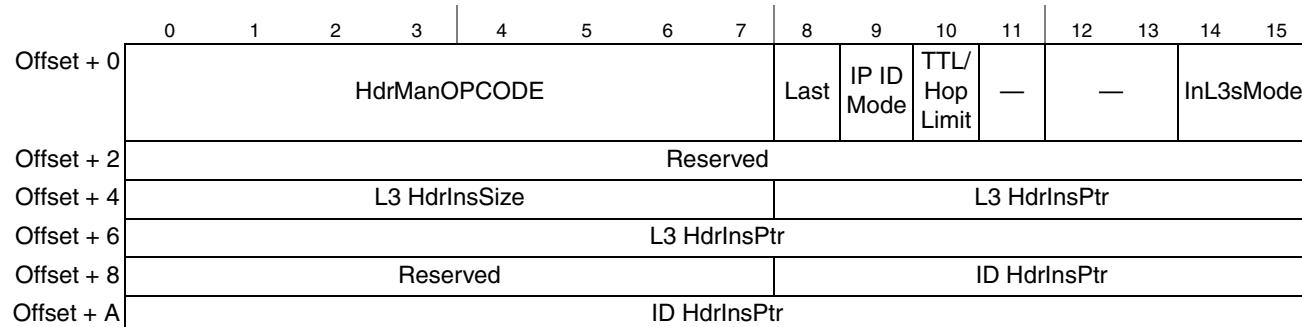
This command descriptor replace IPv4 or IPv6 header in the frame. The Ethernet type and length field of the IP header are automatically updated. The checksum field in the IPv4 header which is inserted, should be clear by the user. The IPv4 checksum is calculated and inserted by this HMCD (this operation is skipped for IPv6 due to the lack of checksum field in its header). The UDP/TCP checksum can be updated.

The protocol/next header and total length/payload length fields in the IPv4/IPv6 header which is inserted, should be clear by the user. The new protocol/next header field will be duplicated from the original IPv4/IPv6 header.

Note that the incoming IPv4 or IPv6 frame must include L4 header (UDP/TCP/SCTP/DCCP) that is supported by the parser. IP encapsulation and routing header are not supported by this command.

The size of this HMCD is 12 bytes. This HMCD uses the parser results offsets so it can be invoked only after the invocation of the parser.

Figure 5-432 shows the internal IP header replace command descriptor.



**Figure 5-432. Internal IP Header replace Command Descriptor**

Table 5-482 describes the internal IP header replace command descriptor fields.

**Table 5-482. Internal IP Header replace Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0–7	HdrMan OPCODE	Opcode = 0x12
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9	IP ID mode	0 No Change in IP ID. 1 The IP ID in the incoming frame, is replaced by the IP ID field (See ID HdrInsPtr). The copied IP ID field (in the internal memory) will be incremented by 1. Note: The ID field in the IPv4 header which is inserted, should be clear by the user.
	10	TTL/HopLimit	0 No change in the Time to Live or Time to Hop Limit field. If InsL3Mode = 00 1 Time to Live field is duplicated from the Hop Limit field decremented by 1. If InsL3Mode = 01 1 Hop Limit field is duplicated from the Time to Live field decremented by 1. Note: If the TTL/HOP Limit ==0, it is not decremented by 1. The Time to Live or Time to Hop Limit field in the IP header which is inserted, should be clear by the user.
	11	—	Reserved. Bits must be cleared.
	12–13	—	Reserved. Bits must be cleared.
	14–15	InsL3Mode	00 Replace IPv4 with IPv6 header. 01 Replace IPv6 with IPv4 header. 10 Reserved 11 Reserved
Base+ 0x2	0–15	—	Reserved. Bits must be cleared.
Base + 0x4	0–7	L3 HdrInsSize	Size of L3 (IP) header, in bytes, to be replaced in frame. In the process of header manipulation, the frame must not increase in size more than (256-FOF) bytes beyond the size of the original incoming frame.
	8–15	L3 HdrInsPtr	Pointer to internal multiuser RAM memory location for L3 header to be inserted.
Base + 0x6	0–15	—	
Base + 0x8	0–7	—	Reserved. Bits must be cleared.
	8–15	ID HdrInsPtr	Pointer to internal multiuser RAM memory location for IP ID field to be inserted.
Base + 0xA	0–15	—	

### 5.12.10.13 UDP/TCP Checksum Calculation Command Descriptor

Available only in FMan\_v3 and only on Offline Port.

This feature supports frames up to 9216B (9KB) only. UDP/TCP checksum calculation is not performed on IP fragment.

This command descriptor calculates the checksum of UDP and TCP and updates the checksum field in the header. No changes are made to non UDP\TCP frames.

In order to enable UDP/TCP Checksum Calculation:

- OP BMI must bring in the full frame in order to update the full running sum in the Parser Result Array. This can be done in one of two ways:
  - NIA 0x50020c (BMI Action Code =0x20c, Fetch Frame and Execute).
  - NIA 0x18 (FMan Controller Action Code = 0x18, Pre-BMI Fetch Full Frame) for more details see [Section 5.12.21,"Pre/Post BMI Fetch NIAs."](#)
- User must set Routing Header Enable bit in Parser IPv6 HXS Configuration.

This command is used as follows:

1. Parser must be invoked before using this command with FD[CMD][DCL4C]=1 (to avoid parsing error in L4 HXS)
2. Since this command uses the Parse Result, after the parser no frame/header manipulation is allowed till invocation of this command.

This command clears the FD[CMD][DTC] bit (which shares the same location with FD[CMD][DCL4C] bit) in order to avoid re-calculation of the checksum by the TX port.

The size of this HMCD is 4 bytes.

[Figure 5-433](#) shows the UDP/TCP Checksum Calculation command descriptor.



**Figure 5-433. UDP/TCP Checksum Calculation Command Descriptor**

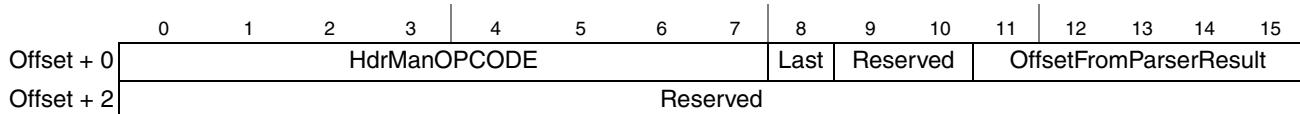
[Table 5-483](#) describes the UDP/TCP Checksum Calculation command descriptor fields.

**Table 5-483. UDP/TCP Checksum Calculation Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0-7	HdrMan OPCODE	Opcode = 0x14
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9-15	Reserved.	Reserved. Bits must be cleared.
Base+ 0x2	0-15	Reserved	Reserved. Bits must be cleared.

### 5.12.10.14 Remove Header till known parser result Command Descriptor

The following HM command removes header till a known offset indicated by parser results offset fields. This command should be used before any other HM commands. Valid only for FMan\_v3. [Figure 5-434](#) shows remove header till known parser result Command Descriptor.



**Figure 5-434. Remove header till known parser result Command Descriptor**

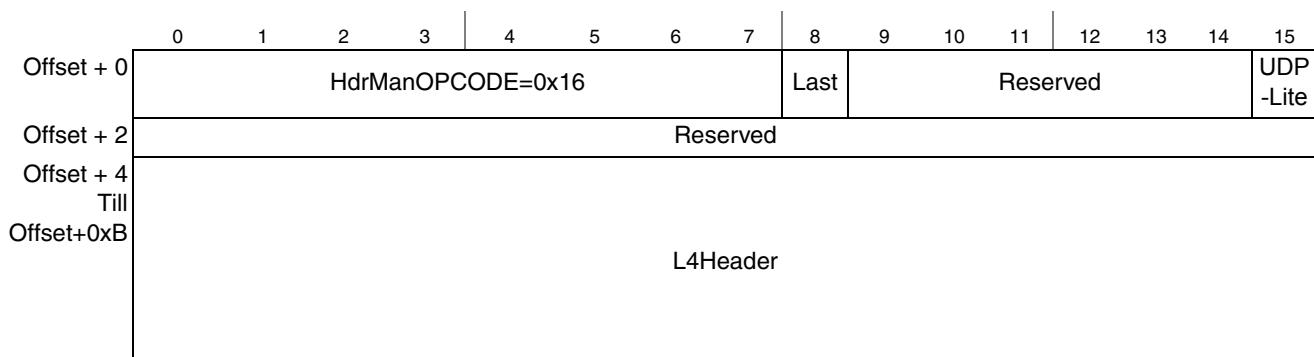
Table 5-484 describes the remove header till known parser result command descriptor fields.

**Table 5-484. Remove header till known parser result Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0–7	HdrMan OPCODE	Opcode = 0x15
	8	Last	0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9–10	Reserved.	Reserved. Bits must be cleared.
	11–15	OffsetFromParser Result	Offset from ‘Parser Result Array’ in which the specific field offset is located. Valid values are 16 to 31. The value specified in this offset in the Parser Result Array, is the bytes number removed from the start of the frame. See FMan Parser chapter for more information.
Base+ 0x2	0–15	Reserved	Reserved. Bits must be cleared.

### 5.12.10.15 Local Insert UDP or UDP-Lite Header Command Descriptor

The following HM command enables inserting UDP or UDP-Lite header. UDP-Lite may be used only if the preceding header is IPv6. It should precede the Local Insert L3 (IPv4 or IPv6). It must be applied only on the beginning of the frame, only remove/Local Insert CAPWAP header HMCD may precede this HMCD. Figure 5-435 shows Local Insert UDP or UDP-Lite Header Command Descriptor.



**Figure 5-435. Local Insert UDP or UDP-Lite Header Command Descriptor**

Table 5-485 describes the Local Insert UDP or UDP-Lite Header command descriptor fields.

**Table 5-485. Local Insert UDP or UDP-Lite Header Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0-7	HdrMan OPCODE	Opcode = 0x16
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9-14	Reserved	Reserved. Bits must be cleared.
	15	UDP-Lite	UDP-Lite. 0 - UDP. 1 - UDP-Lite.
Base+ 0x2	0-15	Reserved	Reserved. Bits must be cleared.
Base+ 0x4 Till Base + 0xB	0-15	L4Header	UDP or UDP Lite header. Must set the checksum field to 0. Must set the checksum coverage field in case of UDP-Lite to 8. Should set length field for UDP header to 0.

### 5.12.10.16 Local Insert L3 (IPv4 or IPv6) Header Command Descriptor

The following HM command enables inserting L3 header; either IPv4 (with options if required) or IPv6 (with extensions if required). If UDP header insertion is also required then first user needs to insert the UDP header and then insert the L3 header. It must be applied only on the beginning of the frame.

Figure 5-436 shows Local Insert L3 Header Command Descriptor.

L4 checksum is calculated if the next header is UDP or UDP-Lite and user set the ‘CalcCS’ bit in the following table. In this case Local Insert UDP or UDP-Lite HMCD should precede this command. Insert HMCDIPv4 header checksum is always calculated.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	HdrManOPCODE=0x17							Last	DF_MODE	—	—	—	Calc CS	PQoS MODE	—	—
Offset + 2	L3HdrInsSize							LastPIDOffset								
Offset + 4	Reserved							IDHdrInsPtr								
Offset + 6	IDHdrInsPtr							LastIPv6DestinationOffset								
Offset + 8	LastIPv6DestinationOffset							Reserved								
Offset + A	Reserved							L3Header								
Offset + C	L3Header							—								

**Figure 5-436. Local Insert L3 Header Command Descriptor**

Table 5-486 describes the Local Insert L3 Header Command Descriptor fields.

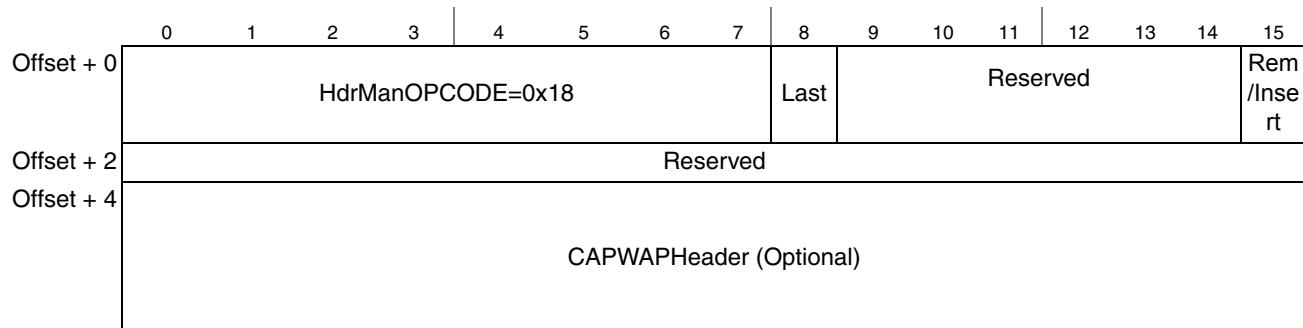
**Table 5-486. Local Insert L3 Header Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0-7	HdrMan OPCODE	Opcode = 0x17
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9	DF_MODE	DF (Don't Fragment) mode. Valid only for IPv4 header. 0 IPv4 DF bit in the user template (L3Header field) is not changed. 1 IPv4 DF bit in the user template is overwritten with the LSB (bit) of the IC HASH field next-to-last byte. The next-to-last HASH byte is configured to be overwritten by the IC located in external memory (meta data) when RPD is set.
	10-12	Reserved	Reserved. Bits must be cleared.
	13	CalcCS	Calculate checksum. User may set this bit only if the preceding HMCD is Local Insert UDP/UDPLite HMCD.
	14-15	PQoS_MODE	Propagate QoS mode. 0 IPv4 TOS or IPv6 traffic class field in the user template (L3Header field) is not changed. 1 IPv4 TOS or IPv6 traffic class field is overwritten with the IC parser results last HASH byte. This byte is configured to be overwritten by the IC located in external memory (meta data) when RPD is set. This mode may be preceded with CAPWAP Manipulation Table descriptor using the same PQoS_MODE. 2-3 Reserved.
Base+ 0x2	0-7	L3HdrInsSize	L3 header size in byte units. L3 header may include options in case of IPv4 or extensions in case of IPv6.
	8-15	LastPIDOffset	Last Protocol ID offset. User program the offset within the L3 inserted header where the last PID is located.
Base+ 0x4	0-7	Reserved	Reserved. Bits must be cleared.
	8-31	IDHdrInsPtr	Pointer to internal multiuser RAM memory location (allocated 2 bytes) or IPv4 ID field to be inserted.
Base+ 0x8	0-31	LastIPv6DestinationOffset	This field must be set to 0 if not routing header extension exists in the IPv6 inserted template. This field indicates the offset of the last IPv6 destination address in the IPv6 inserted template. Needed for UDP checksum calculation.
	8-31	Reserved	Reserved. Bits must be cleared.
Base+ 0xC	0-15	L3Header	L3 header template (either IPv4 or IPv6). IP version field must be set correctly. IP Length field should be cleared. If IPv4 header need to clear ID and checksum fields. The size of the header is indicated in L3HdrInsSize field. User must set the Last PID field to indicate what is the next L4 protocol. User should set the HLEN field for IPv4 header to the correct value.

### 5.12.10.17 Remove/Local Insert CAPWAP Header Command Descriptor

The following CAPWAP Remove/Local Insert CAPWAP header manipulation perform CAPWAP header insertion or CAPWAP header removal. The command considers the options field in the CAPWAP header. It must be applied only on the beginning of the frame. [Figure 5-437](#) shows Removal/Local Insert of

CAPWAP Header Command Descriptor. This command may precede the Insert Local UDP or UDP-Lite HMCD. User should avoid using the Local Insert HMCD when CAPWAP header insertion is required.



**Figure 5-437. Remove/Local Insert CAPWAP Header Command Descriptor**

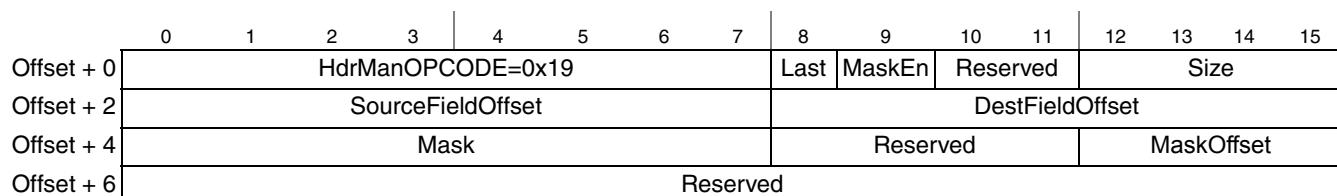
Table 5-487 describes the Remove/Local Insert CAPWAP Header Command Descriptor fields.

**Table 5-487. Remove/Local Insert CAPWAP Header Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0-7	HdrMan OPCODE	Opcode = 0x18
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9-14	Reserved	Reserved. Bits must be cleared.
	15	Rem/Insert	CAPWAP Header Remove or CAPWAP Header Local Insert. 0 - Remove CAPWAP Header. 1 - Insert CAPWAP Header.
Base+ 0x4	0-15	CAPWAPHeader	Valid only if Inserting a CAPWAP Header. User prepare the CAPWAP header template. The HDR field must be set correctly according to CAPWAP header length. The CAPWAP Fragment offset and Fragment ID should be programmed to zero.

### 5.12.10.18 Replace Field in Header Command Descriptor

The following Replace Field in Header manipulation replaces a field (up to 8 bytes) in the header of the frame by a value placed in the IC Hash field. In one of the bytes, a mask can be enabled in order to change only few bits within the byte. Figure 5-438 shows Replace Field in Header Command Descriptor. This command may precede any HM command.



**Figure 5-438. Replace Field in Header Command Descriptor**

Table 5-488 describes the Replace Field in Header Command Descriptor fields.

**Table 5-488. Replace Field in Header Command Descriptor**

Offset	Bit	Name	Description
Base+ 0x0	0-7	HdrMan OPCODE	Opcode = 0x19
	8	Last	Last HMCD 0 Not last Header Manipulation Command Descriptor 1 Last Header Manipulation Command Descriptor
	9	MaskEn	Mask Enable. 0 - No mask is required. 1 - Mask is required on one of the byte to be replaced. The Mask and OffsetForMask fields should be updated accordingly.
	10-11	Reserved	Reserved. Should be cleared.
	12-15	Size	The number of bytes to be replaced. The bytes to be replaced should be consecutive. This number includes the byte affected by the Mask if enabled. The size is limited to 8 bytes.
Base+ 0x2	0-7	SourceFieldOffset	Source Field Offset. The field that will be inserted (source field) should be placed in the Hash field of the IC. The SourceFieldOffset represents the offset from Hash field where the field is located. The SourceFieldOffset must be up to 7. The field located at SourceFieldOffset is configured to be overwritten by the IC located in external memory (meta data) when RPD is set.
	8-15	DestFieldOffset	Destination Field Offset. The DestFieldOffset represents the offset from the start of frame, where the field should be replaced.
Base+ 0x4	0-7	Mask	Valid only if MaskEn is set. This byte represents the mask done on the byte specified by the MaskOffset. In the byte Mask, bit set to "1" represents bit that will be replaced and bit set to "0" represents bit that will stay intact after this HM. For example, if the user wants to change only the 2 lsb bits, Mask should be equal to: 0x03.
	8-11	Reserved	Reserved. Should be cleared.
	12-15	MaskOffset	Valid only if MaskEn is set. This field represents the offset to the byte affected by the Mask Offset within the replaced field. For example, if the user wants to replace 3 bytes but in the second byte the user wants to replace only a few bits, set as follows: - Size = 3. - MaskEn = 1. - Mask = 0xA2 (only 3 bits will be replaced) - MaskOffset to 1.
Base+ 0x6	0-15	Reserved	Reserved. Should be cleared.

### 5.12.11 Restrictions on the Usage of HMCDs

The programming of the HMCD must comply to the following restrictions:

- The Header manipulation can be done only on the first (256-FOF) bytes of the incoming frame (i.e. original frame before any manipulation). FOF is a user programmable parameter. The maximum value of this parameter is 256 bytes.
- The maximum number of bytes that may be inserted in a frame is FOF. This limitation is relevant for any single HMCD or any sequence of HMCDs.
  - The user must set the FD[offset] (or first SG entry offset) to be big enough to include also the inserted header size.
- If a UDP/TCP Checksum Calculation HMCD exists it must be the first manipulation command.
- Other Header manipulation commands have to be preformed from the lowest frame layer and up.
  - If the user removes/inserts fields from a layer, the offset for this layer and all the lower layers will be changed and will not match the parser results offsets. In this case if the user needs to change something in a lower layer based on the parser results, the parser has to be rerun beforehand (See [Section 5.12.12, "Parse After Header Manipulation"](#)). This is the reason why it is recommend to perform the HMCDs from the lowest frame layer and up. Note that if only update commands are in use, there isn't any restriction. However, it is the user's responsibility to adhere to this restriction. In case of a wrong HMCD sequence, a disrupted header manipulation might occur.
    - Examples:
      - If the MPLS header is removed, the offset of the VLAN will be changed.
      - If the IPv4 header is replaced with IPv6, the offset of the L2 header will be changed.
      - If the L4 header is only updated, no offset will be changed. So in this case there isn't any restriction regarding the HMCDs that follow.

## 5.12.12 Parse After Header Manipulation

Due to the Header Manipulation commands, the frame header can be updated. Sometimes these changes can lead to a substantial change in the frame header.

For example,

1. The user can replace IPv4 header with IPv6 header. In this case, it might be required to rerun the parser after the Header Manipulation and by that to update the parser results.
2. If the MPLS header is removed, the offset of the VLAN header in the frame will be changed. In this case, if the VLAN has to be replaced\removed by using the parser results offsets, it is required to rerun the parser after the Header Manipulation and by that to update the parser results.

In order to enable rerunning the parser after the HM was performed, the user has to set the PAHM bit in the HMTD. In this case, after all HMCDs have been performed (the commands are executed one by one from the first one to the last one), the HMTD triggers the parser.

When the parser stops,

- If HMTD[NADEN]==1, the next AD in the CC flow will be as it is determined in the NextActionDescriptorIndex field.
- If HMTD[NADEN]==0, the execution continues through the FPM according to the NIA.

## **NOTE**

The HMTD can trigger the parser only if the starting point of the parse tree (the lower layer header of the frame) is not changed by the HMCDs. For example, if the first header layer is a L2 header, the user can update the L2 header or add/remove upper layers but the L2 header must stay the first layer of the frame.

Figure 5-439 shows an example of Parse after HM generic flow.

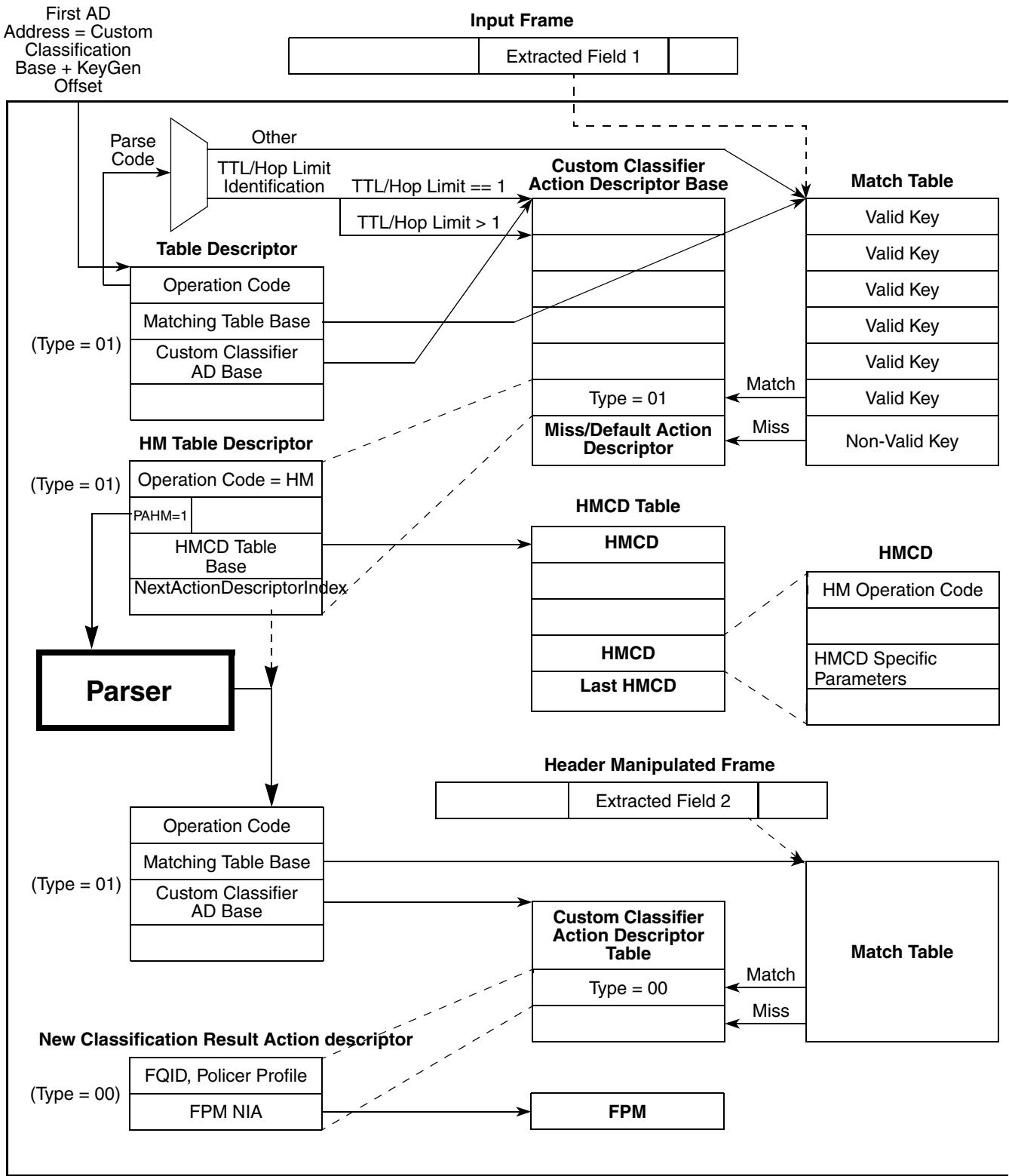


Figure 5-439. Parse after HM Generic Flow Example

Figure 5-439 describes an example flow of parse after header manipulation:

- The HMTD is chosen as a result of the first custom classifier AD.
- After all HMCDs have been performed, the parser is triggered.
- The parser results are updated.
- When the parser stops, the HMTD leads (NextActionDescriptorIndex) to the next AD, the second custom classifier AD.
- The second custom classifier AD uses the updated parser results to extract field 2 and matches it against the keys of the second table.
- A lookup match has been found and in response a new classification result AD (Type 0) is chosen.
- The execution continues through the FPM.

## 5.12.13 Deep Sleep Auto Response

### 5.12.13.1 Introduction to Deep Sleep Auto Response

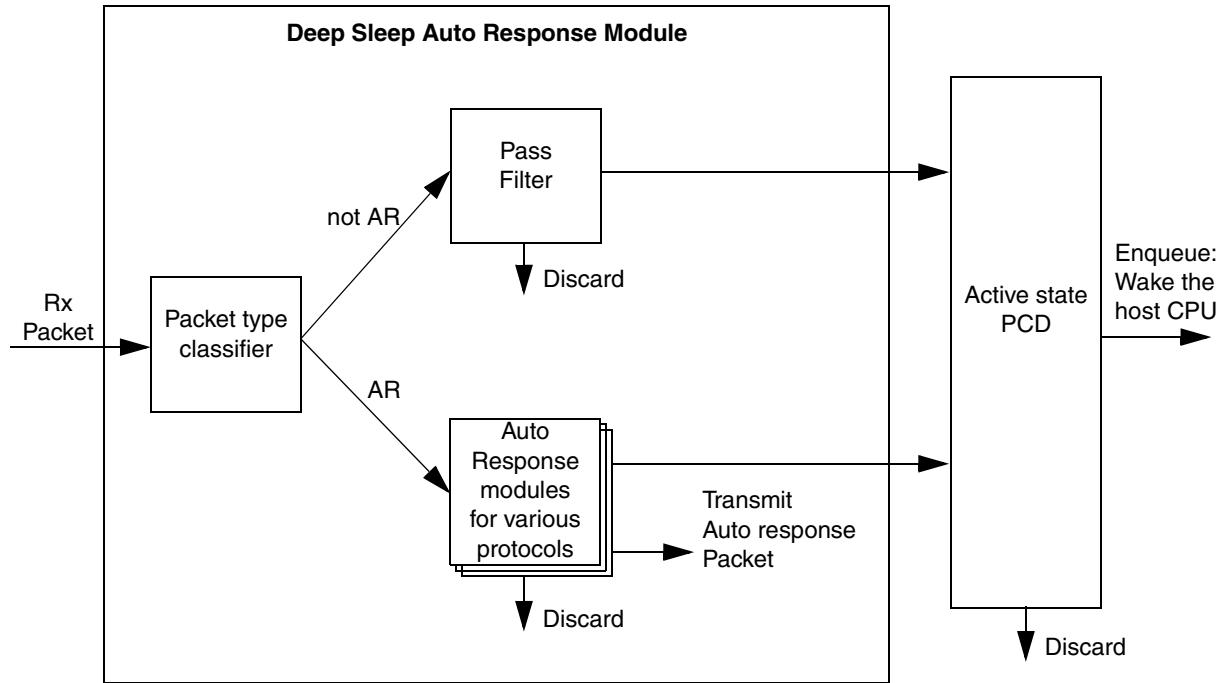
Many electronic systems are required to provide their full functionality only during relatively short periods while during the rest of the time these system are required to maintain only network presence. This fact along with growing concern for energy conservation, were the motivation for developing the ECMA-393 “ProxZzzy for sleeping hosts” standard. The FM and within it the FMan controller Deep Sleep Auto Response module provide functionality for implementing this standard. The main idea is the usage of two distinct operation modes: the ‘Active’ mode in which the whole device including the host CPU is active and full functionality is available, and the ‘Low Power’ (or ‘Deep Sleep’) mode in which most of the device is powered down but some parts remain active to provide the network presence. The FMan controller Deep Sleep Auto Response module is able to respond automatically to many packet types without the need to wake the host CPU and thus it allows the extension of periods in which the device is in low power mode.

The Deep Sleep Auto Response module supports the following protocols:

- Address Resolution Protocol (ARP) for IPv4/Ethernet.
- Neighbor Discovery (ND), for IPv6
- ICMP Echo Request, for both IPv4 and IPv6
- SNMP

Figure 5-440 , ‘Deep Sleep Auto Response main flow’ shows the flow of packets through the main parts of the Deep Sleep Auto Response module. The received packet enters the packet type classifier. In case of error frame as indicated by the hardware Parser (e.g. IP header checksum error), in case of IP fragment and in case of IP tunneling, the packet is discarded and statistics counter is incremented. In case that the incoming packet contains one of the protocols that may be handled automatically, it is sent to appropriate handler (e.g. handler for ARP auto response), otherwise the packet is sent to the pass filter. The auto response handler (ARP handler, SNMP handler etc.) may decide to respond to the packet automatically (transmit a response packet), discard the packet or send the packet to the active state PCD. The decision depends on specific packet content and it may depend on configuration of the auto response module. Packets that cannot be handled by the auto response module are sent to the pass filter. The purpose of this module is to avoid wake up of the host CPU as much as possible. The user configures the pass filter such

that only a subset of all the incoming packets will pass toward the active state PCD. The active state PCD is the same PCD tree which is used when the device is in active mode. In case that the PCD result is enqueue, a wake of the whole device is initiated.



**Figure 5-440. Deep Sleep Auto Response main flow**

### 5.12.13.2 Auto Response Pass Filter

As explained above, the function of the auto response pass filter is to decide which packets should be discarded and which packets should pass to the active state PCD. This filtering is applied to packets that are not handled by any of the auto response modules.

- Non IP filtering: Non IP packets are discarded.
- IP filtering: IPv4 packets are filtered according to the value in their Protocol field (for IPv6 ‘Next Header’ field is used in this stage). The value in the Protocol field is looked up in a table. The action (discard of pass to next stage) in case of hit/miss is configurable per the whole table.
- UDP filtering: UDP packets that pass the IP filtering stage are filtered according to the value in their Source Port and Destination Port fields. The value in the Source Port and Destination Port fields is looked up in a table. This table contains values as well as masks that can be used to enable/disable the comparison per each bit. The action (discard of pass to next stage) in case of hit/miss is configurable per the whole table.
- TCP filtering: TCP packets that pass the IP filtering stage are filtered according to the value in their TCP control flags field (the byte at offset 13 of the TCP header) and then according to their Source Port and Destination Port fields. The TCP control flags filtering allow TCP packets with certain

TCP control flags set to pass. Packets that pass this stage continue to ports based filtering. The value in the Source Port and Destination Port fields is looked up in a table. This table contains values as well as masks that can be used to enable/disable the comparison per bit. The action (discard or pass to next stage) in case of hit/miss is configurable per the whole table.

### 5.12.13.3 Soft Parser for Deep Sleep Auto Response

Soft Parser can be used to expand the functionality of the hardware parser. Generally Soft Parser may change any parse result field but since Deep Sleep Auto Response relies on some fields it is required to preserve their value. Following is a list of parse result fields that should be preserved for proper operations:

- L2R
- L3R
- NxtHdr
- Running Sum
- IP PID Offset
- LastETypeOffset
- IPOffset\_1
- IPOffset\_n
- NxtHdrOffset

### 5.12.13.4 Deep Sleep Auto Response Programming Model

Deep Sleep Auto Response has common parameters and specific per protocol parameters. The common parameters are described in [Table 5-489](#).

The common parameters include general configuration, pointers to specific per protocol parameters that reside in separate descriptors and pointers to the Pass Filter tables: IP protocol table, TCP port table and UDP port table. These parameters should be initialized by software.

The root pointer to the common parameters descriptor, ArCommonDescPtr, should be initiated by software at offset 0x00 in the FMan Controller parameters page as described in [Section 5.12.23,"FMan-Controller Parameters Page per Port."](#)

**Table 5-489. Auto Response Common Parameters Descriptor**

Offset	Bits	Name	Description
General Parameters			
0x00	0-7	ARTxPort	Auto Response Transmit Port number. Frames transmitted by the auto response module will be transmitted to this port. Normally this should be equivalent to the auto response receive port number.
	8-31	Reserved	Reserved. Should be cleared.
0x04	0-31	ActiveHPNIA	Active mode Hardware Parser NIA. This field should be programmed to the content of the FMBM_RFPNE register that existed before entering sleep mode. The FMan Controller will use this value upon directing a frame processing from the auto response module to the active mode classification flow.

<b>Offset</b>	<b>Bits</b>	<b>Name</b>	<b>Description</b>
0x08	0-15	SNMPPort	SNMP Port. Defines the UDP port number that is used for SNMP. Typically this field should be programmed to the value 161.
	16-31	MACStationAddr	MAC Station Address (upper 2 bytes). This field should be initiated by software to indicate the link-layer (MAC) address of the auto response port. It is used by all relevant protocols.
0x0C	0-31	MACStationAddr	MAC Station Address (lower 4 bytes). This field should be initiated by software to indicate the link-layer (MAC) address of the auto response port. It is used by all relevant protocols.

Offset	Bits	Name	Description
Filtering Parameters			
0x10	0-10	Reserved	Reserved. Should be cleared.
	12	IPProtocolTblPass	IP Protocol Table acts as a pass table (i.e. when this bit is set a hit results in pass). 0 - Discard the packet in case of hit in the table. Pass the packet in case of miss. 1 - Pass the packet in case of hit in the table. Discard the packet in case of miss. When IP Protocol filtering is disabled (IPProtocolFiltTblPtr == 0x0000_0000) this bit is not valid and any incoming IP packet will be sent to active state PCD.
	13	UDPPortTblPass	UDP Protocol Table acts as a pass table (i.e. when this bit is set a hit results in pass). 0 - Discard the packet in case of hit in the table. Pass the packet in case of miss. 1 - Pass the packet in case of hit in the table. Discard the packet in case of miss. When UDP Port filtering is disabled (UDPPortFiltTblPtr == 0x0000_0000) this bit is not valid and any incoming UDP packet will be sent to active state PCD.
	14	TCPPortTblPass	TCP Protocol Table acts as a pass table (i.e. when this bit is set a hit results in pass). 0 - Discard the packet in case of hit in the table. Pass the packet in case of miss. 1 - Pass the packet in case of hit in the table. Discard the packet in case of miss. When TCP Port filtering is disabled (TCPPortFiltTblPtr == 0x0000_0000) this bit is not valid and any incoming TCP packet that passed the TCP control filtering will be sent to active state PCD.
	15	Reserved	Reserved. Should be cleared.
	16-31	TCPControlPass	These bits define rules for passing TCP packets through the auto response pass filter based on the value in their TCP control field (the two bytes at offset 12 of the TCP header). The pass filter performs a bitwise ‘and’ operation between this field and the bytes at offset 12 of the TCP header. In case that any of the bits in the result byte is set, the packet is allowed to pass through. Note that all packets after the initial SYN packet have the ACK flag set.
0x14	0-7	IPProtocolTblSize	Number of IP Protocol filtering table entries. In case no IP Protocol filtering is required the value of this field should be set to 0x00.
	8-15	UDPPortTblSize	Number of UDP Port filtering table entries. In case no UDP Port filtering is required the value of this field should be set to 0x00.
	16-23	TCPPortTblSize	Number of TCP Port filtering table entries. In case no TCP Port filtering is required the value of this field should be set to 0x00.
	24-31	Reserved	Reserved. Should be cleared.
0x18	0-31	IPProtocolFiltTblPtr	IP Protocol Filter Table Pointer. A pointer to the IP Protocol table that is used for filtering of non auto response IP packets. Each entry in this table is a single byte. The auto response pass filter compares the values in this table to the value in the IP Protocol field of the incoming packet. The action in case of hit or miss is defined by the IPProtocolTblPass bit (see above). In case no IP Protocol filtering is required the value of this field should be set to 0x0000_0000.
0x1C	0-31	UDPPortFiltTblPtr	UDP Protocol Filter Table Pointer. A pointer to the UDP Protocol table that is used for filtering of non auto response UDP packets. Each entry has a format as defined in <a href="#">Figure 5-442 ,‘TCP/UDP port table entry’</a> . The auto response pass filter compares the values in this table to the values in the Source Port and Destination Port fields of the incoming UDP packet. The action in case of hit or miss is defined by the UDPPortTblPass bit (see above). In case no UDP Port filtering is required the value of this field should be set to 0x0000_0000.

<b>Offset</b>	<b>Bits</b>	<b>Name</b>	<b>Description</b>
0x20	0-31	TCPPortFiltTblPtr	<p>TCP Protocol Filter Table Pointer. A pointer to the TCP Protocol table that is used for filtering of non auto response TCP packets. Each entry has a format as defined in <a href="#">Figure 5-442 ,‘TCP/UDP port table entry’</a>. The auto response pass filter compares the values in this table to the values in the Source Port and Destination Port fields of the incoming TCP packet. The action in case of hit or miss is defined by the UDPPortTblPass bit (see above).</p> <p>In case no TCP Port filtering is required the value of this field should be set to 0x0000_0000.</p>
0x24	0-31	Reserved	Reserved. Should be cleared.

Offset	Bits	Name	Description
Protocols Descriptors Pointers			
0x28	0-31	ARPDescriptorPtr	ARP Descriptor Pointer. Pointer to the ARP descriptor. To disable the ARP auto response functionality, set the value of this pointer to NULL (0x00000000).
0x2C	0-31	NDDescriptorPtr	Neighbor Discovery Descriptor. A pointer to the ND Descriptor. To disable the ND auto response functionality, set the value of this pointer to NULL (0x00000000).
0x30	0-31	ICMPv4Ptr	ICMPv4 Descriptor pointer. To disable the ICMPv4 Echo auto response functionality, set the value of this pointer to NULL (0x00000000).
0x34	0-31	ICMPv6Ptr	ICMPv6 Descriptor pointer. To disable the ICMPv6 Echo auto response functionality, set the value of this pointer to NULL (0x00000000).
0x38	0-31	SNMPDescriptorPtr	SNMP Descriptor pointer. A pointer to the SNMP descriptor. See <a href="#">Figure 5-454 ,‘SNMP Descriptor’</a> for details. To disable the SNMP auto response functionality, set the value of this pointer to NULL (0x00000000).
0x3C	0-31	StatisticsPtr	Pointer to Deep Sleep Auto Response Statistics table.

Figure 5-441 illustrates the auto response programming model.

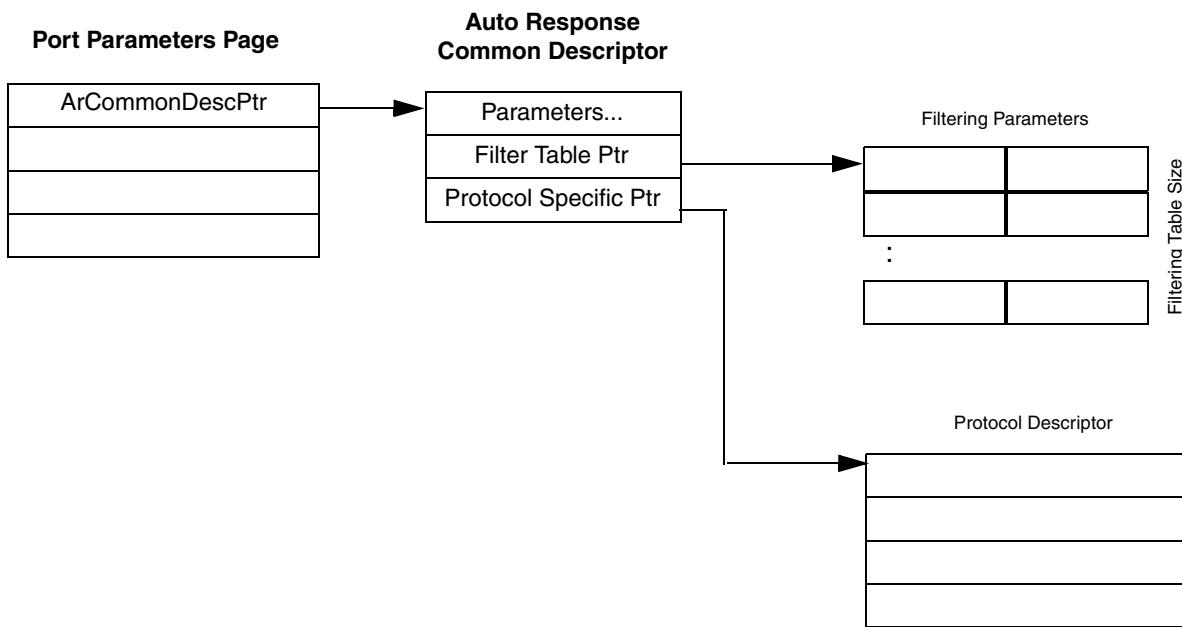


Figure 5-441. Auto Response Programming Model

#### 5.12.13.4.1 Filtering Tables Formats

The IP protocol filtering table is a simple list of bytes. The TCP/UDP port tables contain also a mask as shown in [Figure 5-442](#) below.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																SrcPort
2																DstPort
4																SrcPortMask
6																DstPortMask

**Figure 5-442. TCP/UDP port table entry**

[Table 5-490](#) describes the fields of TCP/UDP port table entry:

**Table 5-490. TCP/UDP port table entry fields**

Offset	Bits	Name	Description
0x0	0–15	SrcPort	Source port. UDP or TCP source port value.
0x2	0–15	DstPort	Destination port. UDP or TCP destination port value.
0x4	0–15	SrcPortMask	Source port mask. A mask for the SrcPort value (TCP or UDP). A ‘0’ in a bit of the mask indicates that the corresponding bit in SrcPort should be ignored. A ‘1’ indicates that the corresponding bit should be compared.
0x6	0–15	DstPortMask	Destination port mask. A mask for the DstPort value (TCP or UDP). A ‘0’ in a bit of the mask indicates that the corresponding bit in DstPort should be ignored. A ‘1’ indicates that the corresponding bit should be compared.

#### 5.12.13.4.2 Deep Sleep Auto Response Statistics table

The Deep Sleep Auto Response Statistics table contains several 32 bits counters. They should be initialized to the value 0 and they are incremented by the Deep Sleep Auto Response functionality as described below.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x00																DSAR_Discarded
0x02																
0x04																DSAR_Err_Discarded
0x06																
0x08																DSAR_Frag_Discarded
0x0A																
0x0C																DSAR_Tunnel_Discarded
0x0E																
0x10																DSAR_ARP_Discarded
0x12																
0x14																DSAR_IP_Discarded
0x16																
0x18																DSAR_TCP_Discarded
0x1A																
0x1C																DSAR_UDP_Discarded
0x1E																
0x20																DSAR_ICMPv6_Checksum_Err
0x22																
0x24																DSAR_ICMPv6_Other_Type
0x26																
0x28																DSAR_ICMPv4_Other_Type
0x2A																

**Figure 5-443. Deep Sleep Auto Response Statistics table**

Table 5-491 describes the statistics table fields:

**Table 5-491. Deep Sleep Auto Response Statistics table fields**

Offset	Bits	Name	Description
0x00	0–15	DSAR_Discarded	This counter is incremented whenever the Auto Response pass filter discards an incoming packet.
0x02	0–15		
0x04	0–15	DSAR_Err_Discarded	This counter is incremented whenever the Auto Response pass filter discards an incoming packet with parsing error (as indicated by the PHE bit in FD[STATUS]).
0x06	0–15		
0x08	0–15	DSAR_Frag_Discarded	This counter is incremented whenever the Auto Response pass filter discards an incoming IP fragment.
0x0A	0–15		
0x0C	0–15	DSAR_Tunnel_Discarded	This counter is incremented whenever the Auto Response pass filter discards an incoming IP tunneled packet.
0x0E	0–15		
0x10	0–15	DSAR_ARP_Discarded	This counter is incremented whenever the Auto Response pass filter discards an incoming ARP Packet.
0x12	0–15		

**Table 5-491. Deep Sleep Auto Response Statistics table fields**

Offset	Bits	Name	Description
0x14	0–15	DSAR_IP_Discarded	This counter is incremented whenever the Auto Response pass filter discards an incoming IP packet.
0x16	0–15		
0x18	0–15	DSAR_TCP_Discarded	This counter is incremented whenever the Auto Response pass filter discards an incoming TCP packet.
0x1A	0–15		
0x1C	0–15	DSAR_UDP_Discarded	This counter is incremented whenever the Auto Response pass filter discards an incoming UDP packet.
0x1E	0–15		
0x20	0–15	DSAR_ICMPv6_Checksum_Err	This counter is incremented whenever Auto Response module receives an ICMP v6 packet with checksum error. The packet is discarded.
0x22	0–15		
0x24	0–15	DSAR_ICMPv6_Other_Type	This counter is incremented whenever the Auto Response module receives an ICMP v6 packet and the type is not echo and not Neighbor Solicitation/Advertisement.
0x26	0–15		
0x28	0–15	DSAR_ICMPv4_Other_Type	This counter is incremented whenever the Auto Response module receives an ICMP v4 packet and the type is not echo.
0x2A	0–15		

## 5.12.13.5 Address Resolution Protocol (ARP) Deep Sleep Auto Response

### 5.12.13.5.1 Introduction to ARP Deep Sleep Auto Response

The FMan Controller supports auto response for Ethernet-IPv4 Address Resolution Protocol (ARP), which comply with the ECMA-393 standard.

This solution main features allow:

- ARP echo detection - The FMan Controller inspects the Sender Protocol Address (SPA) and the Sender Hardware Address (SHA) ARP header fields for a case that these fields include one of the device's own IP-MAC bindings.
- ARP conflict detection - The FMan Controller inspects SPA and SHA ARP header fields for an IP-MAC binding in which the IP address is one of the device's own addresses, but the binding itself is not one of the device's own bindings. The host is responsible to manage an ongoing conflict.
- ARP Ethernet-IPv4 validation - The FMan Controller inspects the six first bytes of the ARP header to assure the network layer is IPv4 and the link layer is Ethernet.
- VLAN, IP address and MAC address binding - The FMan Controller looks for a match between the VLAN ID and IP address of the incoming frame against its VLAN - IP - MAC list. Only the first (outer) VLAN tag is supported in case of multiple tags.
- ARP Auto-Response - The FMan Controller replies to packets which require a reply according to the RFC 826 standard.

The following features are not supported by the ARP auto response function:

- The ARP auto-response module does not manage entries of IP-MAC addresses of other devices nor it makes any updates to MAC-IP bindings given by the user as parameters.

- The ARP auto-response module does not initiate ARP probing or ARP announcing (see RFC 5227). The host CPU shall manage these features if required during active mode.
- The ARP auto-response module does not support ARP announcements, also called ‘gratuitous ARP’ messages. These are usually broadcast as an ARP request containing the sender's protocol address (SPA) in the target field (TPA=SPA) with the target hardware address (THA) set to zero, or ARP reply with the sender's hardware and protocol addresses (SHA and SPA) duplicated in the target fields (TPA=SPA, THA=SHA). Such messages will be discarded.

### 5.12.13.5.2 Functional Description of ARP Deep Sleep Auto Response

During Deep Sleep mode, the following actions are taken on an incoming ARP frame:

- The FMan Controller checks the validity of the ARP header. In a case in which the header is not a valid Ethernet/IPv4 ARP the packet is dropped, and the FMan Controller increments a counter (INVAL\_CNT) and finishes processing the frame.
- The FMan Controller compares ARP Header SPA field (see RFC 826) to the list of IPv4 addresses:
  - If an echo case is detected the frame is dropped, the FMan Controller increments the echo counter (ECHO\_CNT) and finishes processing the frame.
  - If an IPv4 address conflict detected (RFC 5227) the FMan Controller increments the CD\_CNT counter (regardless of CDEN value). If conflict detection enable bit is set in the ARP Descriptor the FMan Controller proceeds to the ‘Active’ mode processing flow. According to the ‘Active mode’ configuration, this can cause a system wake-up or frame discard. If the conflict detection enable bit is cleared, the FMan Controller continues as follows.
- The FMan Controller compares ARP header Target Protocol Address (TPA) field (see RFC 826) and if a VLAN tag exists, the VLAN ID of the incoming frame to the list of the device’s VLAN IDs and IPv4 addresses.

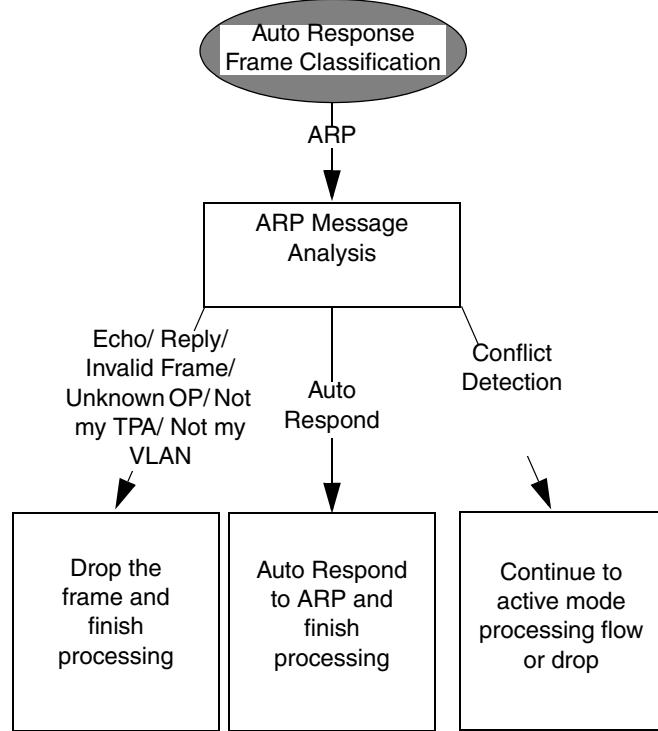
#### NOTE

A value of ‘0’ in the VLAN ID field of the list indicates entry with no VLAN tag or a null VLAN ID. These entries can only match incoming frames with no VLAN tag or a null VLAN ID. There is no distinguishing between the two cases.

- If the ARP header TPA field contains an IPv4 address which exists in the list, and the VLAN ID of the incoming frame (if exists) matches the VLAN ID in that entry of the list, the FMan Controller examines the ‘OP’ (operation) field in the ARP header:
  - If ARP[OP] == 1 (ARP Request) the FMan Controller replies to this packet as stated in RFC 826, increments a counter (AR\_CNT) and finishes processing the frame.
  - If ARP[OP] == 2 (ARP Reply) the FMan Controller increments a counter (RATM\_CNT) and drops the packet.
  - Else, the packet is dropped and the FMan Controller increments a counter (UKOP\_CNT) and finishes processing the frame.
- If the ARP header TPA field contains an IPv4 address which exists in the list, VLAN tag exists, but the VLAN ID of the incoming frame is different than the VLAN ID in the list of that entry, the FMan Controller increments the NMVLAN\_CNT counter and drops the packet.

- If the ARP header TPA field contains an IPv4 address which does not exist in the IPv4 list, the packet is dropped and the FMan Controller increments a counter (NMTP\_CNT) and finishes processing the frame.

The following illustration demonstrates the above functionality:



**Figure 5-444. Deep Sleep ARP Flow Illustration**

#### 5.12.13.5.3 Deep Sleep Auto Response ARP Descriptor

The operation of ARP Deep Sleep Auto Response is configured in the ARP descriptor data structure and in additional related data structures that are pointed to by fields in the ARP descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0										Reserved						CDEN	
2										NumOfBindings							
4										BindingsPointer							
6										StatisticsPointer							
8																	
A																	
C										Reserved							
E																	

**Figure 5-445. ARP Descriptor**

Table 5-492 describes the fields of the ARP Descriptor:

**Table 5-492. ARP Descriptor**

Offset	Bits	Name	Description
0x0	0–14	Type	Reserved
	15	CDEN	Conflict Detection Enable 0 - Conflict detection is disabled. The host CPU will not be woken upon conflict detection and the packet will be treated normally. However a counter (CD_CNT) will be incremented. 1 - Conflict detection is enabled. Ethernet-IPv4 ARP frames which contains address conflict will cause the FMan Controller to proceed to the ‘Active’ mode processing flow. According to the ‘Active mode’ configuration, this can cause a system wake-up or frame discard.
0x2	0–15	NumOfBindings	Number of entries in the binding list.
0x4	0–15	BindingsPointer	Bindings Pointer. This points to an VLAN-IPv4 Addresses Bindings list.
0x6	0–15		
0x8	0–15	StatisticsPointer	Statistics Pointer. This field points to the ARP Descriptor’s statistics data structure.
0xA	0–15		
0xC	0–15	Reserved	Reserved. Must be cleared.
0xE	0–15		

#### 5.12.13.5.4 Deep Sleep Auto Response ARP Data Structures

##### VLAN-IPv4 Addresses Bindings

The VLAN-IPv4 addresses bindings is a list with variable number of entries, set by the software driver. The number of entries is defined by the NumOfBindings parameter in the ARP descriptor.

Every valid IP address of the port should have an entry in the list.

Each entry in the list contains the parameters described in Table 5-493. The list is searched from the first entry until a match or until the end of the list according to the NumOfBindings value.

**Table 5-493. VLAN-IPv4 Binding List Structure**

Name	Size	Description
Ipv4Addr	32 bit	IPv4 Address.
VlanId	16 bit	12 bits VLAN ID. The 4 left-most bits should be clear. This field should be 0x0000 for an entry with no VLAN tag or a null VLAN ID.
Reserved	16 bit	Reserved

##### ARP Statistics Counters

Table 5-494 describes the statistics counters for the ARP function. The statistics counters data structure is pointed by the ‘StatisticsPointer’ field in the ARP descriptor. All counters are 32 bits long. The counters should be initialized to all zeros by software.

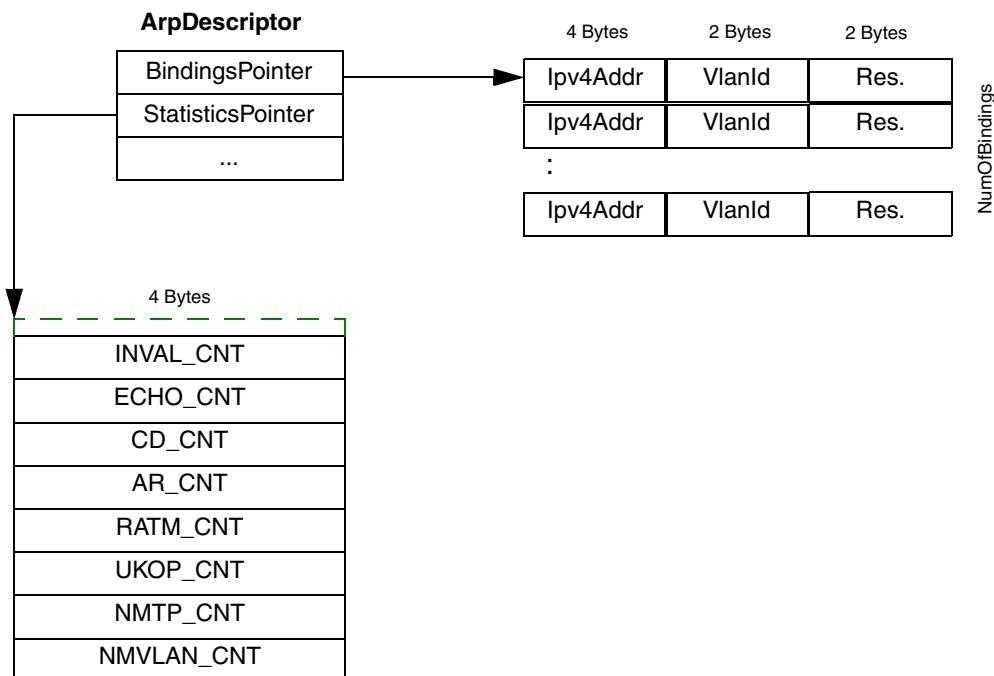
**Table 5-494. ARP Statistics Counters**

Offset	Name	Description
0x00	INVAL_CNT	Invalid ARP IPv4-Ethernet counter
0x04	ECHO_CNT	Echo counter
0x08	CD_CNT	Conflict Detection counter
0x0C	AR_CNT	Auto-Response counter
0x10	RATM_CNT	Replies Addressed To Me counter
0x14	UKOP_CNT	Unknown Operation counter
0x18	NMTP_CNT	Not my TPA counter
0x1C	NMVLAN_CNT	Not My VLAN counter

For details of when the counters are incremented see [Section 5.12.13.5.2,"Functional Description of ARP Deep Sleep Auto Response."](#)

## Data Structure Summary Illustration

The ARP data structure is illustrated in [Figure 5-446](#)



**Figure 5-446. Deep Sleep Auto Response ARP Data Structures illustration**

## 5.12.13.6 Internet Control Message Protocol (ICMP) Deep Sleep Auto Response

### 5.12.13.6.1 Introduction to ICMP Deep Sleep Auto Response

The FMan Controller supports auto response to Ethernet-IPv4 ICMP Echo messages, according to the RFC792 standard. ICMP Echo messages are commonly used with the ‘ping’ application.

This solution main features allow:

- ICMP message validation - The FMan Controller inspects the ICMP header and validates that it has legal values and correct checksum.
- VLAN and IP address lookup - The FMan Controller looks for a match between the VLAN ID and IP address of the incoming frame against its VLAN - IP. Only the first (outer) VLAN tag is supported in case of multiple tags.
- ICMP Echo Auto-Response - The FMan Controller replies with an ICMP Echo Reply message to valid incoming ICMP Echo messages.

### 5.12.13.6.2 IPv4 ICMP Deep Sleep Auto Response Functional Description

During Deep Sleep mode, the following actions are taken on an incoming IPv4 ICMP (ICMPv4) frame:

- The FMan Controller checks the validity of the ICMP header. This includes the header fields and checksum calculation. In a case in which the header is not a valid IPv4 ICMP Echo message the packet is dropped, the FMan Controller increments the INVAL\_CNT counter and finishes processing the frame.
  - A valid ICMP message must have a value of 8 in the Type field, 0 in the code field and a correct checksum.
  - Upon a checksum error, the checksum error counter (CSERR\_CNT) is incremented.
- The FMan Controller compares IPv4 header Destination Address (DA), and if a VLAN tag exists, the VLAN ID of the incoming frame to the list of the device’s VLAN IDs and IPv4 addresses.

#### NOTE

A value of ‘0’ in the VLAN ID field of the list indicates entry with no VLAN tag or a null VLAN ID. These entries can only match incoming frames with no VLAN tag or a null VLAN ID. There is no distinguishing between the two cases.

- If the IPv4 DA field contains an IPv4 address which exists in the list, and the VLAN ID of the incoming frame (if exists) matches the VLAN ID in that entry of the list, the ICMPv4 auto response counter (AR\_CNT) is incremented. The FMan Controller will respond with an ICMP Echo Reply message.
- If the IPv4 DA field contains an IPv4 address which exists in the list, VLAN tag exists, but the VLAN ID of the incoming frame is different than the VLAN ID in the list of that entry, the FMan Controller increments the NMVLAN\_CNT counter and drops the packet.
- If the IPv4 DA field contains an IPv4 address which does not exist in the IPv4 list, the packet is dropped and the FMan Controller increments a counter (NMIP\_CNT) and finishes processing the frame.

- Upon a valid ICMP Echo message detection and a VLAN ID and IP address match, the FMan Controller prepares the ICMP Echo Reply message and transmits it according to the RFC 792 standard.
  - The ICMP Type field is set to 0.
  - The ICMP header checksum is calculated.
  - The Source and Destination IP addresses are switched.
  - The TTL field is set to 255.
  - The IP header checksum is recalculated.
  - Any other header or payload fields are retained.
  - The Ethernet (MAC) Source and Destination addresses are switched.

### 5.12.13.6.3 Deep Sleep Auto Response ICMPv4 Descriptor

The operation of IPv4 ICMP Deep Sleep Auto Response is configured in the ICMPv4 descriptor data structure and in additional related data structures that are pointed to by fields in the ICMPv4 descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																Reserved
2																NumOfBindings
4																BindingsPointer
6																
8																StatisticsPointer
A																
C																Reserved
E																

Figure 5-447. ICMPv4 Descriptor

Table 5-495 describes the fields of the ICMPv4 Descriptor:

Table 5-495. ICMPv4 Descriptor

Offset	Bits	Name	Description
0x0	0–15	Type	Reserved
0x2	0–15	NumOfBindings	Number of entries in the binding list.
0x4	0–15	BindingsPointer	Bindings Pointer. This points to an VLAN-IPv4 Addresses Bindings list.
0x6	0–15		
0x8	0–15	StatisticsPointer	Statistics Pointer. This field points to the ICMPv4 statistics data structure.
0xA	0–15		
0xC	0–15	Reserved	Reserved. Must be cleared.
0xE	0–15		

#### 5.12.13.6.4 Deep Sleep Auto Response ICMPv4 Data Structures

##### VLAN-IPv4 Addresses Bindings

The VLAN-IPv4 addresses bindings is a list with variable number of entries, set by the software driver. The number of entries is defined by the NumOfBindings parameter in the ICMPv4 descriptor.

Every valid IP address of the port should have an entry in the list.

Each entry in the list contains the parameters described in [Table 5-496](#). The list is searched from the first entry until a match or until the end of the list according to the NumOfBindings value.

**Table 5-496. VLAN-IPv4 Binding List Structure**

Name	Size	Description
Ipv4Addr	32 bit	IPv4 Address.
VlanId	16 bit	12 bits VLAN ID. The 4 left-most bits should be clear. This field should be 0x0000 for an entry with no VLAN tag or a null VLAN ID.
Reserved	16 bit	Reserved

##### ICMPv4 Statistics Counters

[Table 5-497](#) describes the statistics counters for the ICMPv4 function. The statistics counters data structure is pointed by the ‘StatisticsPointer’ field in the ICMPv4 descriptor. All counters are 32 bits long. The counters should be initialized to all zeros by software.

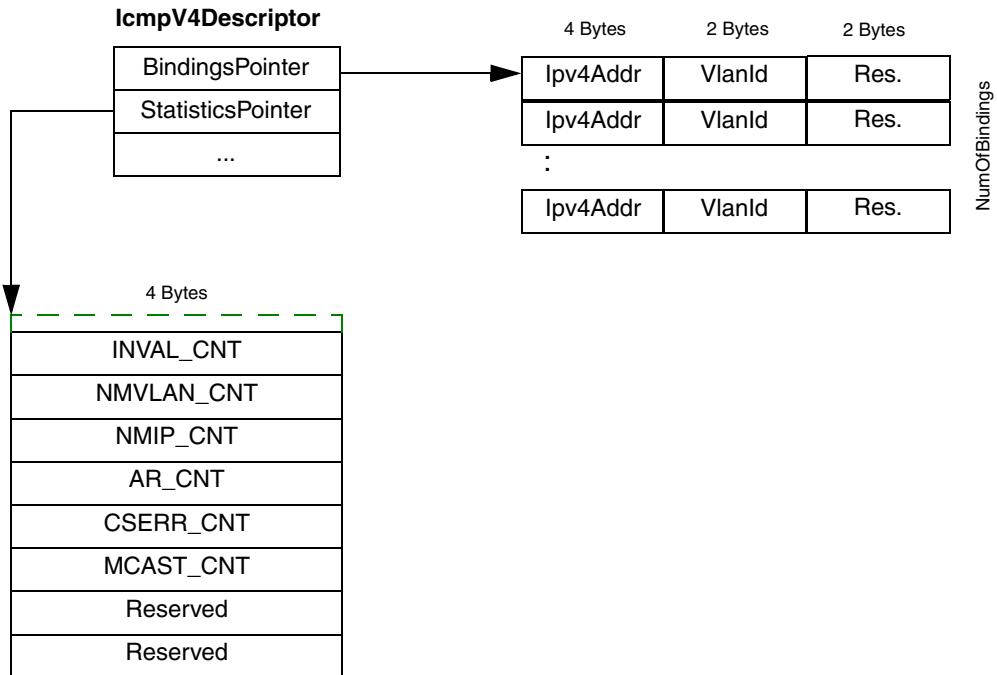
**Table 5-497. ICMPv4 Statistics Counters**

Offset	Name	Description
0x00	INVAL_CNT	Invalid ICMPv4 header counter
0x04	NMVLAN_CNT	Not My VLAN counter
0x08	NMIP_CNT	Not My IP counter
0x0C	AR_CNT	Auto-Response counter
0x10	CSERR_CNT	Checksum Error counter
0x14	Reserved	Reserved
0x18	Reserved	Reserved
0x1C	Reserved	Reserved

For details of when the counters are incremented see [Section 5.12.13.6.2, "IPv4 ICMP Deep Sleep Auto Response Functional Description."](#)

##### Data Structure Summary Illustration

The ICMPv4 data structure is illustrated in [Figure 5-448](#).



**Figure 5-448. Deep Sleep Auto Response ICMPv4 Data Structures illustration**

### 5.12.13.7 Internet Control Message Protocol (ICMPv6) Echo Auto Response

#### 5.12.13.7.1 Introduction to ICMPv6 Echo Deep Sleep Auto Response

The FMan Controller supports auto response for Ethernet-IPv6 ICMP Echo messages, according to the RFC 4443 standard. ICMPv6 Echo messages are commonly used with the ‘ping’ application.

Every ICMPv6 message is preceded by an IPv6 header and zero or more IPv6 extension headers. The ICMPv6 header is identified by a Next Header value of 58 in the immediately preceding header.

This solution main features allow:

- ICMPv6 message validation - The FMan Controller inspects the ICMPv6 header and validates that it has legal values and correct checksum.
  - Note that unlike ICMPv4, the ICMPv6 checksum is calculated on a pseudo-header that includes IPv6 header fields, as described in RFC 2460, section 8.1 “Upper-Layer Checksums”
- VLAN and IP address lookup - The FMan Controller looks for a match between the VLAN ID and IP address of the incoming frame against its VLAN - IP. Only the first (outer) VLAN tag is supported in case of multiple tags.
- ICMP Echo Auto-Response - The FMan Controller replies with an ICMP Echo Reply message to valid incoming ICMP Echo messages.

## **NOTE**

The ECMA-393 standard does not specifically defines ICMPv6 Echo support.

### **5.12.13.7.2 ICMPv6 Echo Auto Response Functional Description**

During Deep Sleep mode, the following actions are taken on an incoming IPv6 ICMP (ICMPv6) frame:

- The FMan Controller checks the validity of the ICMPv6 header. This includes the IP pseudo-header fields, the ICMPv6 header fields and a checksum calculation.
  - In case the ICMPv6 Type field is ICMPv6 Echo (128), but the ICMPv6 Echo descriptor pointer in the Common descriptor (ICMPv6Ptr) is null, the processing flow is directed to the Pass Filter.
  - In case the ICMPv6 Type field is ICMPv6 Echo (128), but the Code field is not 0, the FMan Controller increments the INVAL\_CNT counter and finishes processing the frame.
  - In case the ICMPv6 Type value is not supported by the auto-response module, the Common statistics table ICMPv6 Other Type counter (DSAR\_ICMPv6\_Other\_Type) is incremented.
  - Upon a checksum error, the checksum error of the Common statistics table Checksum Error counter (DSAR\_ICMPv6\_Checksum\_Err) is incremented.
- The FMan Controller compares IPv6 header Destination Address (DA), and if a VLAN tag exists, the VLAN ID of the incoming frame to the list of the device's VLAN IDs and IPv6 addresses.

## **NOTE**

A value of ‘0’ in the VLAN ID field of the list indicates entry with no VLAN tag or a null VLAN ID. These entries can only match incoming frames with no VLAN tag or a null VLAN ID. There is no distinguishing between the two cases.

- If the IPv6 DA field contains an IPv6 address which exists in the list, and the VLAN ID of the incoming frame (if exists) matches the VLAN ID in that entry of the list, the ICMPv6 auto response counter (AR\_CNT) is incremented. The FMan Controller will respond with an ICMPv6 Echo Reply message.
- If the ICMPv6 pseudo-header contains an IPv6 address which exists in the list, VLAN tag exists, but the VLAN ID of the incoming frame is different than the VLAN ID in the list of that entry, the FMan Controller increments the NMVLAN\_CNT counter and drops the packet.
- If the ICMPv6 pseudo-header contains an IPv6 address which does not exist in the IPv6 list, the packet is dropped and the FMan Controller increments a counter (NMIP\_CNT) and finishes processing the frame.
- Upon a valid ICMP Echo message detection and a VLAN ID and IP address match, the FMan Controller prepares the ICMP Echo Reply message and transmits it according to the RFC 4443 standard.
  - The ICMPv6 Type field is set to Echo Reply
  - The ICMPv6 pseudo-header checksum is calculated.
  - The Source and Destination IP addresses are switched.

- The IP Hop Limit field is set to 255.
- Any other header or payload fields are retained.
- The Ethernet (MAC) Source and Destination addresses are switched.

### 5.12.13.7.3 Deep Sleep Auto Response ICMPv6 Descriptor

The operation of IPv6 ICMP Deep Sleep Auto Response is configured in the ICMPv6 descriptor data structure and in additional related data structures that are pointed to by fields in the ICMPv6 descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																Reserved
2																NumOfBindings
4																BindingsPointer
6																
8																StatisticsPointer
A																
C																Reserved
E																

**Figure 5-449. ICMPv6 Descriptor**

[Table 5-498](#) describes the fields of the ICMPv6 Descriptor:

**Table 5-498. ICMPv6 Descriptor**

Offset	Bits	Name	Description
0x0	0–15	Reserved	Reserved
0x2	0–15	NumOfBindings	Number of entries in the binding list.
0x4	0–15	BindingsPointer	Bindings Pointer. This points to an VLAN-IPv6 Addresses Bindings list.
0x6	0–15		
0x8	0–15	StatisticsPointer	Statistics Pointer. This field points to the ICMPv6 statistics data structure.
0xA	0–15		
0xC	0–15	Reserved	Reserved. Must be cleared.
0xE	0–15		

### 5.12.13.7.4 Deep Sleep Auto Response ICMPv6 Data Structures

#### VLAN-IPv6 Addresses Bindings

The VLAN-IPv6 addresses bindings is a list with variable number of entries, set by the software driver. The number of entries is defined by the NumOfBindings parameter in the ICMPv6 descriptor.

Every valid IP address of the port should have an entry in the list.

Each entry in the list contains the parameters described in [Table 5-499](#). The list is searched from the first entry until a match or until the end of the list according to the NumOfBindings value.

**Table 5-499. VLAN and IPv6 Addresses List Structure**

Parameter Name	Size	Field	Bits	Description
Ipv6Addr	4x32 bit			IPv6 Address. Array of 4 x 32 bits (total 128 bits)
VlanId	16 bit	VID	15:0	The 12 right-most bits (11:0) are VLAN ID (VID). VID should be 0x000 for an entry with no VLAN tag or a null VLAN ID. The 4 left-most bits (15:12) are reserved and should be cleared.
Reserved	16 bit			Reserved.

**NOTE**

The entry format is kept identical to the Neighbor Discovery (ND) list, so the tables can be shared.

**ICMPv6 Statistics Counters**

[Table 5-500](#) describes the statistics counters for the ICMPv6 function. The statistics counters data structure is pointed by the ‘StatisticsPointer’ field in the ICMPv6 descriptor. All counters are 32 bits long. The counters should be initialized to all zeros by software.

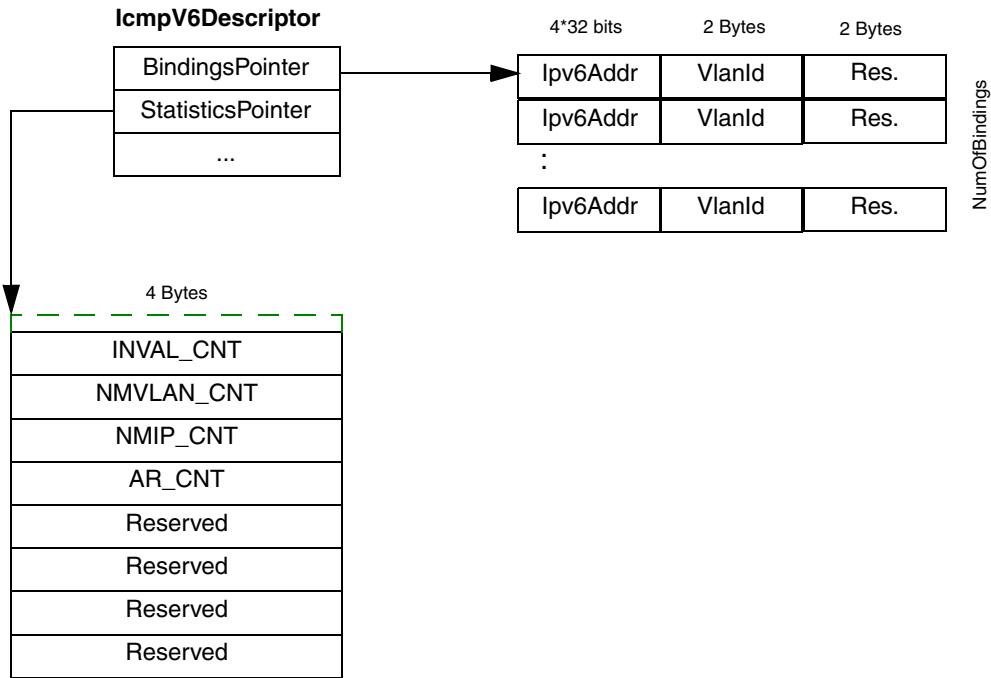
**Table 5-500. ICMPv6 Statistics Counters**

Offset	Name	Description
0x00	INVAL_CNT	Invalid ICMPv6 header counter
0x04	NMVLAN_CNT	Not My VLAN counter
0x08	NMIP_CNT	Not My IP counter
0x0C	AR_CNT	Auto-Response counter
0x10	Reserved	Reserved
0x14	Reserved	Reserved
0x18	Reserved	Reserved
0x1C	Reserved	Reserved

For details of when the counters are incremented see [Section 5.12.13.7.2, "ICMPv6 Echo Auto Response Functional Description."](#)

**Data Structure Summary Illustration**

The ICMPv6 data structure is illustrated in [Figure 5-450](#).



**Figure 5-450. Deep Sleep Auto Response ICMPv6 Data Structures illustration**

### 5.12.13.8 Neighbor Discovery Protocol Auto Response

#### 5.12.13.8.1 Introduction to IPv6 Neighbor Discovery Deep Sleep Auto Response

The FMan Controller supports auto response for Ethernet-IPv6 Neighbor Discovery messages, according to the RFC 4861 standard. The Neighbor Discovery Protocol is used for address autoconfiguration of nodes, discovery of other nodes on the link, determining the Link Layer addresses of other nodes, duplicate address detection. Neighbor Discovery messages are sent as part of an ICMPv6 packet.

The Neighbor Discovery Protocol defines five ICMPv6 packet types for the purpose of Router Solicitation, Router Advertisement, Neighbor Solicitation, Neighbor Advertisement, and Redirect. Neighbor Solicitations messages are used by nodes to determine the Link Layer address of a neighbor, and Neighbor Advertisements are used by nodes to respond to a Neighbor Solicitation message.

Neighbor Solicitation messages can also be used to determine if more than one node has been assigned the same unicast address.

This solution main features allow:

- ICMPv6 message validation - The FMan Controller inspects the ICMPv6 header and validates that it has legal values and correct checksum.
  - Note that unlike ICMPv4, the ICMPv6 checksum is calculated on a pseudo-header that includes IPv6 header fields, as described in RFC 2460, section 8.1 “Upper-Layer Checksums”

- VLAN and IP address lookup - The FMan Controller looks for a match between the VLAN ID and Target Address of the incoming message, and its VLAN - IP. Only the first (outer) VLAN tag is supported in case of multiple tags.
- Neighbor Discovery Auto-Response - The FMan Controller replies with a Neighbor Advertisement message to valid incoming Neighbor Solicitation messages.
- Support for global and link-local IPv6 addresses.
- Support Multicast-node Solicited address.

This following features are not supported:

- Router Solicitation, Router Advertisement and Redirect messages. The node is functioning as a Host, not as a Router.
- Receiving unsolicited Neighbor Advertisement messages. The FMan Controller does not update IPv6-MAC address bindings (of other hosts) during sleep mode.
- Initiation of unsolicited Neighbor Advertisement messages.

#### **5.12.13.8.2 Neighbor Discovery Auto Response Functional Description**

During Deep Sleep mode, the following actions are taken on an incoming IPv6 Neighbor Discovery frame:

- The FMan Controller checks the validity of the ICMPv6 header. This includes the IP pseudo-header fields, the ICMPv6 header fields and a checksum calculation.
  - A valid ND message must have a value of 135 in the ICMPv6 Type field, 0 in the Code field and a correct checksum value.
  - The IP Hop Limit field should have a value of 255.
  - The ICMP length (derived from the IP length) should be 24 or more octets.
  - The Target Address must not be a multicast address.
  - All included ICMP options must have a length that is greater than zero.
  - If the IP source address is the unspecified address, the IP destination address should be a solicited-node multicast address.
  - If the IP source address is the unspecified address, must not include a source link-layer address option in the message.
- In case the ICMPv6 Type field is Neighbor Solicitation (135) or Neighbor Advertisement (136), but the Neighbor Discovery pointer in the Common descriptor (NDDescriptorPtr) is null, the processing flow is directed to the Pass Filter.
- In case the ICMPv6 Type value is not supported by the auto-response module, the Common statistics table ICMPv6 Other Type counter (DSAR\_ICMPv6\_Other\_Type) is incremented.
- In case of receiving an unsolicited Neighbor Advertisement messages, the FMan Controller increments the USADVERT\_CNT and drops the frame.
- In case the header is not a valid Neighbor Solicitation message the packet is dropped, the FMan Controller increments the INVAL\_CNT counter and finishes processing the frame.

- In case of receiving an unsolicited Neighbor Advertisement messages, the FMan Controller increments the USADVERT\_CNT.
- Upon a checksum error, the checksum error of the Common statistics table Checksum Error counter (DSAR\_ICMPv6\_Checksum\_Err) is incremented.
- The FMan Controller analyzes the Neighbor Solicitation message and acts according to the rules defined in RFC 4861 (Neighbor Discovery in IPv6).
  - The Source IP address, Destination IP address, Target address and other message fields are checked.
  - If this is a multicast packet, and the SolicitedAddr field in the ND descriptor is not 0xFFFFFFFF, the solicited node multicast address is compared to the multicast groups existing in the ND descriptor. In case the group address does not match, the packet is silently discarded, and the NMMCAST\_CNT counter is incremented. If SolicitedAddr is 0xFFFFFFFF, the solicited node multicast address is ignored, and only the Target Address is compared.
- The FMan Controller compares the Target Address (DA), and if a VLAN tag exists, the VLAN ID of the incoming frame to the list of the device's VLAN IDs and IPv6 addresses.

#### **NOTE**

A value of '0' in the VLAN ID field of the list indicates entry with no VLAN tag or a null VLAN ID. These entries can only match incoming frames with no VLAN tag or a null VLAN ID. There is no distinguishing between the two cases.

- If the Target Address contains an IPv6 address which exists in the list, and the VLAN ID of the incoming frame (if exists) matches the VLAN ID in that entry of the list, the following actions are taken:
  - If the Target Address matches a table address, the FMan Controller will respond with a Neighbor Advertisement message. In this case the Neighbor Discovery auto response counter (AR\_CNT) is incremented.

#### **NOTE**

In a special case that the Target Address matches a table address, the source IP address is a unicast address, but the ICMPv6 Source Link-layer Address option is omitted, the NSLLA\_CNT counter will be incremented and the processing will move to the active state.

In such situations, the device will first have to use Neighbor Discovery to determine the link-layer address of its neighbor (i.e., send out a multicast Neighbor Solicitation) before it is possible to respond with a Neighbor Advertisement message (see RFC 4861). This must be done by the host CPU in wake mode.

- If the VLAN ID of the incoming frame does not match any VLAN ID entry of the list, the packet is silently discarded, and the NMVLAN\_CNT counter is incremented.
- If the Target Address of the incoming frame does not match any IP address entry of the list, the packet is silently discarded, and the NMIP\_CNT counter is incremented.

- The Neighbor Advertisement message is composed according to the rules defined by RFC 4861.

When replying with a Neighbor Advertisement message, the following actions are taken:

- The ICMPv6 Type field is set to Neighbor Advertisement, 136. The ICMPv6 Option Type field is set to Target link-layer address.
- The ICMP Solicited ('S') field is set according to the description [Table 5-501](#).
- The IP Hop Limit field is set to 255 (left unchanged from the neighbor solicitation message).
- The ICMPv6 'Target Address' (IPv6 address) is set to the Target Address field in the Neighbor Solicitation message that prompted this advertisement.
- The ICMPv6 'Target link-layer address' (MAC address) option is set to the device own MAC address (as set in the Common descriptor, MACStationAddr).
- The IPv6 Destination Address is set according to RFC 4861:
  - If the source of the solicitation is the unspecified address, the all-nodes multicast address of the link-local scope (FF02::1).
  - Otherwise, the unicast Source Address of the solicitation.
- The IPv6 Source Address is set to the Target Address field in the Neighbor Solicitation message that prompted this advertisement.
- The Ethernet header MAC Destination Address is set as follows:
  - If the source of the solicitation is the unspecified address, the MAC address is set to an IPv6 multicast MAC address according to RFC 2464: the first two octets are 0x3333 and the last four octets are the last four octets of IP DST address (all-nodes address in this case, FF02::1, so the MAC address is set to 0x33\_33\_00\_00\_00\_01).
  - If the source of the solicitation is NOT the unspecified address, the 'Source Link-Layer Address' from the received Neighbor Solicitation message.
- Any other header or payload fields are retained.
- The ICMPv6 pseudo-header checksum is calculated.
- The Ethernet header MAC Source Address is set to the MACStationAddr.

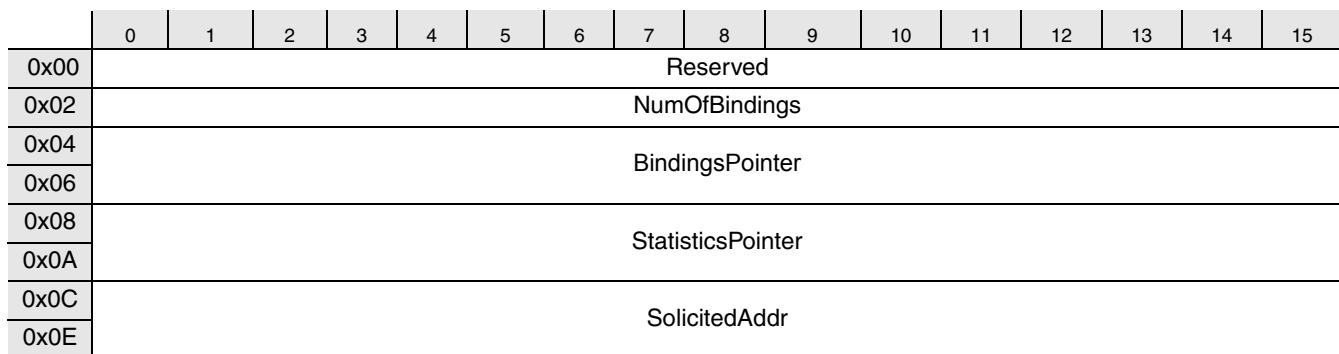
[Table 5-501](#) summarizes the response to various incoming Neighbor Solicitation packets upon a hit in the lookup table.

**Table 5-501. Neighbor Solicitation Response Cases**

Source IP	Destination IP	Source Link-layer Address	Action	Neighbor Advertisement Message Parameters
Unspecified address	Solicited-node multicast address	Must not exist	Response with a Neighbor Advertisement message.	<ul style="list-style-type: none"> <li>Solicited flag: 0</li> <li>Target Address: as in the Neighbor Solicitation message.</li> <li>Target Link-layer Address: MACStationAddr</li> <li>IP Destination: All-nodes multicast address.</li> <li>MAC Destination: 0x333300000001</li> <li>MAC Source: MACStationAddr</li> </ul>
NOT the Unspecified address (the source unicast address)	Solicited-node multicast address or the Target Address (unicast)	Exists	Response with a Neighbor Advertisement message.	<ul style="list-style-type: none"> <li>Solicited flag: 1</li> <li>Target Address: as in the Neighbor Solicitation message.</li> <li>Target Link-layer Address: MACStationAddr</li> <li>IP Destination: the IP Source from the Neighbor Solicitation message.</li> <li>MAC Destination: the Source link-layer address from the Neighbor Solicitation message.</li> <li>MAC Source: MACStationAddr</li> </ul>
NOT the Unspecified address (the source unicast address)	The Target Address. (unicast)	Does not exist	Move to active state.	

#### 5.12.13.8.3 Deep Sleep Auto Response Neighbor Discovery Descriptor

The operation of IPv6 Neighbor Discovery Deep Sleep Auto Response is configured in the ND descriptor data structure and in additional related data structures that are pointed to by fields in the ND descriptor.



**Figure 5-451. ND Descriptor**

[Table 5-502](#) describes the fields of the Neighbor Discovery Descriptor:

**Table 5-502. Neighbor Discovery Descriptor**

Offset	Bits	Name	Description
0x00	0–15	Reserved	Reserved
0x02	0–15	NumOfBindings	Number of entries in the binding list. Note that the list is common for both the temporary and assigned addresses.
0x04	0–15	BindingsPointer	Bindings Pointer. This points to an VLAN-IPv6 Addresses Bindings list.
0x06	0–15		
0x08	0–15	StatisticsPointer	Statistics Pointer. This field points to the Neighbor Discovery statistics data structure.
0x0A	0–15		
0x0C	0–15	SolicitedAddr	Solicited-node Multicast Address. All addresses must be within the same multicast group. The upper 8 left-most bits should normally be cleared, while the lower 24 bits represent the solicited node multicast address group. A special value of 0xFFFFFFFF in this field indicates the FMan Controller to skip the multicast group comparison.
0x0E	0–15		

#### 5.12.13.8.4 Deep Sleep Auto Response Neighbor Discovery Data Structures

##### VLAN-IPv6 Addresses Bindings

The VLAN-IPv6 addresses bindings is a list with variable number of entries, set by the software driver. The number of entries is defined by the NumOfBindings parameter in the ND descriptor.

Every valid assigned or temporary IP address of the port should have an entry in the list.

The MAC address is not part of this list, but exists in the global Auto Response descriptor. See [Section 5.12.13.4, "Deep Sleep Auto Response Programming Model."](#)

Each entry in the list contains the parameters described in [Table 5-503](#). The list is searched from the first entry until a match or until the end of the list according to the NumOfBindings value.

**Table 5-503. VLAN and IPv6 Addresses List Structure**

Parameter Name	size	Field	Bits	Description
Ipv6Addr	4x32 bit			IPv6 Address. Array of 4 x 32bits (total 128 bits)
VlanId	16 bit	VID	15:0	The 12 right-most bits (11:0) are VLAN ID (VID). VID should be 0x000 for an entry with no VLAN tag or a null VLAN ID. The 4 left-most bits (15:12) are reserved and should be cleared.
Reserved	16 bit			Reserved.

##### NOTE

The entry format is identical to the ICMPv6 Echo list, so the tables can be shared.

## Neighbor Discovery Statistics Counters

[Table 5-504](#) describes the statistics counters for the Neighbor Discovery function. The statistics counters data structure is pointed by the ‘StatisticsPointer’ field in the Neighbor Discovery descriptor. All counters are 32 bits long. The counters should be initialized to all zeros by software.

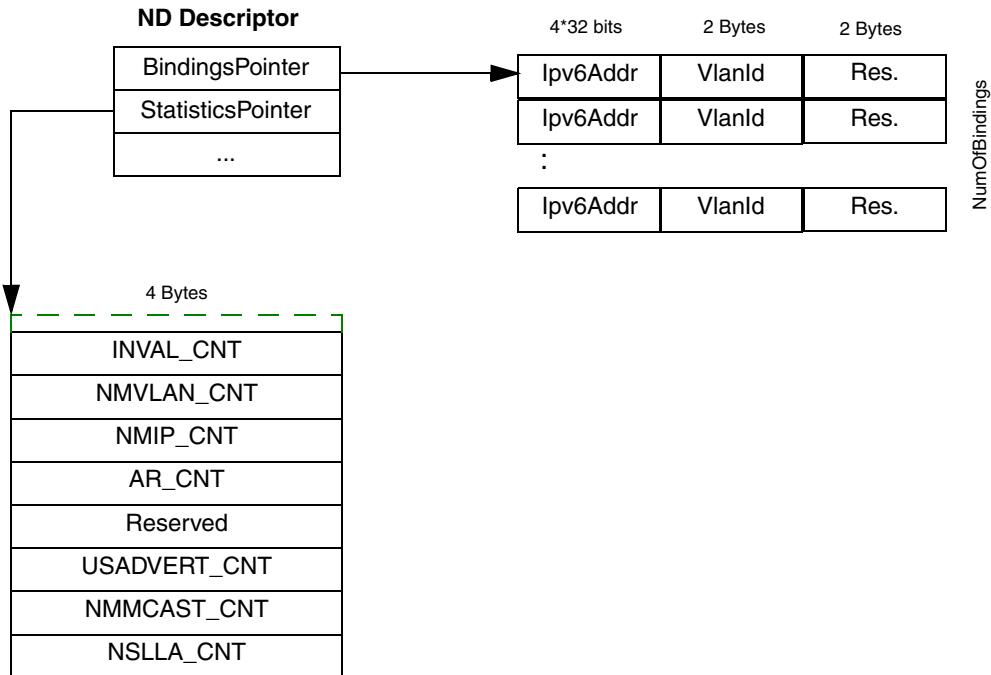
**Table 5-504. Neighbor Discovery Statistics Counters**

Offset	Name	Description
0x00	INVAL_CNT	Invalid Neighbor Discovery message counter
0x04	NMVLAN_CNT	Not My VLAN counter
0x08	NMIP_CNT	Not My IP counter
0x0C	AR_CNT	Auto-Response counter
0x10	Reserved	Reserved.
0x14	USADVERT_CNT	Unsolicited Neighbor Advertisements counter
0x18	NMMCAST_CNT	Not My Multicast group counter
0x1C	NSLLA_CNT	No Source Link-Layer Address counter. Indicates that there was a match on a Target Address of a packet that its source IP address is a unicast address, but the ICMPv6 Source Link-layer Address option is omitted

For details of when the counters are incremented see [Section 5.12.13.8.2, "Neighbor Discovery Auto Response Functional Description."](#)

## Data Structure Summary Illustration

The Neighbor Discovery data structure is illustrated in [Figure 5-452](#).



**Figure 5-452. Deep Sleep Auto Response Neighbor Discovery Data Structures illustration**

### 5.12.13.9 SNMP Deep Sleep Auto Response

#### 5.12.13.9.1 Introduction to SNMP Deep Sleep Auto Response

Simple Network Management Protocol (SNMP) is an internet protocol for managing devices in the network. The SNMP deep sleep auto response functionality parses incoming SNMP packets (requests) and responds to a subset of these requests automatically without the need to wake up the host CPU. Thus this functionality extends the periods in which the host CPU remains in deep sleep low power mode. The SNMP protocol is defined in a set of RFC documents. The main RFC documents relevant for this implementation are RFC 1156, RFC 1157 and RFC 1441. ECMA 393 standard defines the operations of a proxy for sleeping hosts for various protocols including SNMP. There are 3 versions of the SNMP: SNMPv1, SNMPv2 and SNMPv3 but ECMA 393 requires that the proxy shall support only SNMPv1 and SNMPv2.

#### 5.12.13.9.2 Key features of SNMP Deep Sleep Auto Response

- Configurable UDP port used for identifying SNMP messages.
- Configurable list of OID-value pairs (cached OIDs).
- Automatic response to Get-request for cached OIDs.

- Support for getall\_flag that defines the behavior in case of Get-request for non cached OID (as defined in ECMA 393): When getall\_flag is not enabled a “noSuchValue” is returned in the response message. When getall\_flag is enabled a wake of host CPU is initiated.
- Automatic response for Get-next-request. The OID-VALUE of lexicographic next (cached) OID are returned.
- Supports wake of host CPU for Set-request (it is also possible to drop these packets based on the value in the community string in the incoming message and the SNMP auto response configuration).

### 5.12.13.9.3 Functional description of SNMP Deep Sleep Auto Response

The SNMP protocol allows an ‘SNMP manager’ to manage an ‘SNMP agent’ by means of SNMP messages. Each SNMP agent contains a set of objects that may be managed. This set of objects is called the Management Information Base (MIB). SNMP defines various operations that may be performed on the managed objects. SNMP messages are used for implementing these operations. Each object in the MIB has a unique identifier called Object ID (OID). The OID is used within the SNMP messages to identify the objects that are being accessed. The deep sleep auto response functionality supports a subset of the SNMP operations/messages as follows.

- Get-request: a request to read the value of one or multiple objects. The objects to read are identified by their OID values.
- Get-next-request: a request to read the value of one or multiple objects. For each OID in the Get-next-request message the agent should return the value associated with the lexicographically next OID. This message may be used by the SNMP manager to implement a table walk on MIB objects.
- Set-request: a request to write value(s) to one or multiple objects.

Typically Get-request and Get-next-request are handled by auto response and Set-request results in wake of the host CPU. However, the exact operation (auto response, drop or wake of the host) depends on specific user configuration and on the exact content of the SNMP message.

This implementation supports SNMP messages that are sent over IP/UDP protocol stack. A configurable UDP port number is used for identifying the SNMP messages. Typically UDP port 161 is used for this purpose. All SNMP messages share the same general message format as shown in [Figure 5-453](#) below.

IP	UDP	version	community	PDU-type	request-id	error-status	error-index	variable bindings
----	-----	---------	-----------	----------	------------	--------------	-------------	-------------------

**Figure 5-453. SNMP message format**

All SNMP message fields (from version field to variable bindings field) are encoded according to Basic Encoding Rules (BER). This encoding is defined in ITU standard X.690. Generally each entity is encoded as a Type, Length, Value. Following is a description of the functionality per SNMP message field.

#### version

This field contains an integer value. Messages containing the values 1 or 2 are processed. Messages with other values are discarded and the SNMP error counter is incremented.

## community

This field contains an octet string that is used as sort of password for allowing access to the MIB objects. The SNMP descriptor (see [Figure 5-454](#)) contains pointers to ‘read only community string’ and ‘read write community string’ that are stored in FMan controller memory. For Get-request and Get-next-request the community string in the incoming message is compared to the ‘read only community string’ and ‘read write community string’ in FMan controller memory. In case of a match to any of the configured community strings the message is processed. In case of a mismatch to both strings, the message is discarded. For Set-request when the pointer to ‘read write community string’ contains a valid value (i.e. it is not a null pointer) the community string is compared to the ‘read write community string’. In case of a match the message is sent to active state PCD. In case of mismatch or when the string pointer is NULL the packet is discarded.

## PDU-type

This field identifies the type of the SNMP message. Get-request, Get-next-request and Set-request messages are processed. Other messages are discarded.

## request-id

This field contains an integer. The value in this field is copied to the Response message.

## error-status

In case that the auto response function encounters an error while processing the variable bindings, the auto response function fills this field of the outgoing message with a code that indicates the type of error as defined in the SNMP RFCs. Note that error response messages will contain only the error status and index and the variable bindings part of the message will be copied from the incoming Get-request or Get-next-request message.

## error-index

In case that the auto response function encounters an error while processing the variable bindings, the auto response function fills this field of the outgoing message with the index of the variable that had an error.

## variable bindings

This is a list of OID-VALUE pairs. In case of Get-request and Get-next-request the auto response functionality parses the variable bindings in the incoming SNMP message and builds the variable bindings of the outgoing Response message. The SNMP deep sleep auto response programming model contains an OIDs table. The OIDs table represents an ordered list of OID-VALUE pairs, such that an entry with lexicographically larger OID will follow an entry with lexicographically smaller OID.

In case of Get-request, the auto response functionality searches the OIDs table looking for exact match on the OID. In case of a hit the auto response functionality will use the ‘result’ information from the OIDs table to fill the VALUE in the response message. In case of a miss the functionality depends on the setting of the getall\_flag (see [Table 5-505](#) for details). This operation is repeated for each OID-Value pair in the incoming message. Note that error response messages will contain only the error status and index and the variable bindings part of the message will be copied from the incoming Get-request or Get-next-request message.

In case of Get-next-request, the auto response functionality searches the OIDs table looking for an entry with lexicographically larger OID than the OID in the incoming message. The OID and VALUE in this entry are used for filling the OID and VALUE fields in the response message. This operation is repeated for each OID-value pair in the incoming message. In case that there is no entry with lexicographically larger OID, the functionality depends on the setting of the getall\_flag (see [Table 5-505](#) for details). Note that error response messages will contain only the error status and index and the variable bindings part of the message will be copied from the incoming Get-request or Get-next-request message.

Incoming messages with up to 32 variable bindings will be responded. Incoming message with more than 32 variable binding will wake the host.

#### 5.12.13.9.4 SNMP Deep Sleep Auto Response Programming model

##### SNMP Descriptor

The operation of SNMP Deep Sleep Auto Response is configured in the SNMP descriptor data structure and in additional related data structures that are pointed to by fields in the SNMP descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	getall_flag															
															reserved	
2															MaxSnmpMsgLength	
4															NumOfIPv4Addresses	
6															NumOfIPv6Addresses	
8															IPv4AddrTblPtr	
A																
C															IPv6AddrTblPtr	
E																
10															RdOnlyCommunityStrPtr	
12																
14															RdWrCommunityStrPtr	
16																
18															OidsTablePtr	
1A																
1C															OidsTableSize	
1E																
20															StatisticsPtr	
22																

Figure 5-454. SNMP Descriptor

Table 5-505 describes the fields of SNMP Descriptor:

**Table 5-505. SNMP Descriptor fields**

Offset	Bits	Name	Description
0x0	0	getall_flag	For Get-request getall_flag functions as defined in ECMA-393 0: For uncached OID respond with the value noSuchName. 1: For uncached OID wake the host. For Get-next-request (not defined in ECMA-393) getall_flag functions as follows: 0: For OID that has no lexicographic next OID respond with the value noSuchName. 1: For OID that has no lexicographic next OID wake the host.
	1–15	reserved	Reserved. Must be cleared.
0x2	0–15	MaxSnmpMsgLength	Maximum SNMP message length. The SNMP auto response module monitors the length of the SNMP message that it builds and verifies that it will not exceed the value in this parameter. If the size of the generated response message exceeds MaxSnmpMsgLength, then the auto response module sends the “tooBig” status message as defined in RFC1157. SNMP autoreponse module uses buffers from the Tx port (as defined by the ARTxPort in the auto response common parameters descriptor) for building the SNMP response message. It is the responsibility of higher layer software modules to set MaxSnmpMsgLength in accordance to the number and size of buffers allocated to the Tx port. There should be enough buffer memory for sending one largest SNMP message (including the Ethernet/IP/UDP layers). In case that there are not enough buffers for sending a message (due to wrong setting of MaxSnmpMsgLength), the system will hang.
0x4	0–15	NumOfIPv4Addresses	Number of entries in the IPv4 Address table. When IPv4AddrTblPtr is NULL (0x00000000) this parameter is ignored.
0x6	0–15	NumOfIPv6Addresses	Number of entries in the IPv6 Address table. When IPv6AddrTblPtr is NULL (0x00000000) this parameter is ignored.
0x8	0–15	IPv4AddrTblPtr	Pointer to IPv4 addresses table. See the table entry format below in <a href="#">IPv4 Addresses Table</a> . For SNMP packets that are carried on IPv4 the SNMP auto response module verifies that the destination address of the IPv4 and the VLAN ID match one of the entries in the IPv4 addresses table. In case that the VLAN/IPv4 address does not match any entry in the table, the packet is sent to the auto response pass filter. If the IPv4AddrTblPtr contains NULL (0x00000000), then all SNMP over IPv4 packets are processed by the SNMP auto response module regardless of their VLAN/IPv4 address.
0xA	0–15		
0xC	0–15	IPv6AddrTblPtr	Pointer to IPv6 addresses table. See the table entry format below in <a href="#">IPv6 Addresses Table</a> . For SNMP packets that are carried on IPv6 the SNMP auto response module verifies that the destination address of the IPv6 and the VLAN ID match one of the entries in the IPv6 addresses table. In case that the VLAN/IPv6 address does not match any entry in the table, the packet is sent to the auto response pass filter. If the IPv6AddrTblPtr contains NULL (0x00000000), then all SNMP over IPv6 packets are processed by the SNMP auto response module regardless of their VLAN/IPv6 address.
0xE	0–15		
0x10	0–15	RdOnlyCommunityStrPtr	Pointer to the read only community string. The read only community string should reside in FMAN controller memory and it should be encoded in BER format.
0x12	0–15		
0x14	0–15	RdWrCommunityStrPtr	Pointer to the read write community string. The read write community string should reside in FMAN controller memory and it should be encoded in BER format.
0x16	0–15		

**Table 5-505. SNMP Descriptor fields**

<b>Offset</b>	<b>Bits</b>	<b>Name</b>	<b>Description</b>
0x18	0–15	OidsTablePtr	Pointer to the OIDs table. Each entry in the table has a format as defined in <a href="#">Figure 5-455</a> and <a href="#">Table 5-508</a> below. The table should represent a lexicographically ordered list. Thus the first entry in the table should point to the lexicographically first OID and each successive entry should point to lexicographic successor.
0x1A	0–15		
0x1C	0–15	OidsTableSize	OIDs table size. The number of entries (i.e. the number of OID-VALUE pairs) in the OIDs table. Oids table should contain up to 256 entries.
0x1E	0–15		
0x20	0–15	StatisticsPtr	Pointer to the SNMP statistics table. Each entry in the table is a 32 bits statistics counter.
0x22	0–15		

## IPv4 Addresses Table

The IPv4 addresses table contains a list of VLAN/IPv4 addresses. Each entry in the table contains a pair of VLAN and IPv4 address. The number of entries in the table is defined by NumOfIPv4Addresses in the SNMP descriptor. See the description of IPv4AddrTblPtr in SNMP descriptor. Note that the format of this table is identical to the format of the VLAN-IPv4 Addresses Bindings table in ARP auto response module. Thus it is possible to share a common table.

**Table 5-506. IPv4 Addresses Table entry**

<b>Name</b>	<b>Size</b>	<b>Description</b>
Ipv4Addr	32 bit	IPv4 Address.
VlanId	16 bit	12 bits VLAN ID. The 4 left-most bits should be clear. This field should be 0x0000 for an entry with no VLAN tag or a null VLAN ID.
Reserved	16 bit	Reserved

## IPv6 Addresses Table

The IPv6 addresses table contains a list of VLAN/IPv6 addresses. Each entry in the table contains a pair of VLAN and IPv6 address. The number of entries in the table is defined by NumOfIPv6Addresses in the SNMP descriptor. See the description of IPv6AddrTblPtr in SNMP descriptor. Note that the format of this table is identical to the format of the VLAN-IPv6 Addresses Bindings table in ICMPv6 auto response module. Thus it is possible to share a common table.

**Table 5-507. IPv6 Addresses Table entry**

<b>Name</b>	<b>Size</b>	<b>Description</b>
Ipv6Addr	4x32 bit	IPv6 Address. Array of 4 x 32 bits (total 128 bits)
VlanId	16 bit	The 12 right-most bits (11:0) are VLAN ID (VID). VID should be 0x000 for an entry with no VLAN tag or a null VLAN ID. The 4 left-most bits (15:12) are reserved and should be cleared.
Reserved	16 bit	Reserved.

## OIDs Table

The OIDs table points to the OID-value pairs that are handled by the SNMP proxy. Each entry in the table points to a single OID-value pair. The format of a table entry is defined below.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0									OidSize							
2										ResSize						
4											OidPtr					
6																
8											ResValOrPtr					
A																
C												Reserved				
E																

**Figure 5-455. OIDs table entry**

[Table 5-508](#) describes the fields of OIDs table entry:

**Table 5-508. OIDs table entry fields**

Offset	Bits	Name	Description
0x0	0–15	OidSize	OID size. Number of octets in the OID that is pointed by OidPtr.
0x2	0–15	ResSize	Result size. Number of octets in the result that is associated to the OID.
0x4	0–15	OidPtr	Pointer to the OID. The OID should reside in FMAN controller memory and it should be encoded in BER format excluding the type and length fields of BER i.e. only the value of the OID should reside in memory.
0x6	0–15		
0x8	0–15	ResValOrPtr	The functionality of this field depends on the number of octets in the result (indicated by ResSize). In case of 4 or less octets, the result resides directly in this field. In case of more octets, this field is a pointer to the result which resides in FMAN controller memory. In any case the result should be encoded in BER format (including type, length and value).
0xA	0–15		
0xC	0–15	Reserved	Reserved. Must be cleared.
0xE	0–15		

## SNMP Statistics Table

The SNMP statistics table contains several 32 bits SNMP statistics counters. The counters should be initialized to the value 0 and are incremented by the SNMP auto response functionality as described below.

**Figure 5-456. SNMP statistics table**

Table 5-509 describes the fields of SNMP statistics table:

**Table 5-509. SNMP statistics table fields**

Offset	Bits	Name	Description
0x0	0–15	SnmpErrCnt	SNMP Error Counter. This counter is incremented for each SNMP packet that contains an error. The following conditions are considered errors: <ul style="list-style-type: none"> <li>• Wrong BER type octet</li> <li>• Bad length fields (more than 3 octets or first octet equals 0x80).</li> <li>• Inconsistent values in length fields (e.g. length in UDP length is not consistent to length in SNMP).</li> <li>• Wrong SNMP version (version field does not contain 1 or 2).</li> <li>• Non zero error index or error status</li> </ul>
0x2	0–15		
0x4	0–15	SnmpCommunityErrCnt	SNMP Community Error Counter. This counter is incremented for each SNMP packet that contains a wrong community value.
0x6	0–15		
0x8	0–15	SnmpTotalDiscardCnt	SNMP Total Discard Counter. This counter is incremented for each SNMP packet that is discarded by the SNMP autoresponse functionality.
0xA	0–15		
0xC	0–15	SnmpGetReqCnt	SNMP Get Request Counter. This counter is incremented for each SNMP Get request.
0xE	0–15		
0x10	0–15	SnmpGetNextReqCnt	SNMP Get Next Request Counter. This counter is incremented for each SNMP Get Next request.
0x12	0–15		

### 5.12.14 FMan Controller Important Often-Overlooked Details

- Each table descriptor has its own unique Action Descriptor table and Matching table.
  - Initialize the FMBM\_RCCB or FMBM\_OCCB per port to determine the custom classifier base. This base address must be 256 byte aligned. Refer to [Section 5.5.4.6.14, "Tx Custom Classifier Base Register \(FMBM\\_TCCB\)"](#), and [Section 5.5.4.7.11, "O/H Custom Classifier Base Register \(FMBM\\_OCCB\)"](#). Different ports may have the same custom classifier base.
  - Initialize KeyGen scheme CCBS and CCOBASE so that the correct first Action Descriptor is accessed.

- Initialize the FMBM\_RFCA or FMBM\_OFCA register per port. This register programs the FMan controller MR[0:5] (which determines the mode attributes) and programs the OR and SYNC fields. User must initialize MR[2] to 1 (this assures proper data handling in FMan memory) and SYNC ‘10’, which requests synchronization (needed for SYNC command). To assure ordering of frames user must set the OR bit in FMBM\_RFCA or FMBM\_OFCA registers (needed for order definition) and also set the ORR bit in the NIA which resides in the FMBM\_RFENE or FMBM\_OFENE registers (needed for order restoration). Refer to [Section 5.5.4.5, "Rx Port Register Descriptions,"](#) and [Section 5.5.4.7, "Offline Port/Host Command Port Registers Description."](#)
- The first Action Descriptor pointer is used as a root to a search tree and can not be used as a middle node in a pre-defined search tree. A search tree defines all the possible flows for incoming frame look-ups.

### 5.12.14.1 Dynamic Update of Custom Classifier and HM Tables

Dynamic update of custom classifier and header manipulation tables is supported. This means that the user prepares the new Action Descriptor and all its related data structures (it may be a new look up tree), ultimately replacing the old Action Descriptor in a synchronized fashion.

The user may choose one of three flows to dynamically update custom classifier tables.

#### 5.12.14.1.1 Direct Table Access Direct Access Sync Flow

The following is the first flow, which uses an FPM external request mechanism, and may only be used by hypervisor.

1. Prepare the new custom classifier Action Descriptor and all of its corresponding structures.
2. Substitute the first four bytes of the old Action Descriptor with the information shown in [Figure 5-457](#).
3. Perform SYNC via an external request 0 mechanism, which may be done only by hypervisor. User must assert a request by setting FMFP\_EXTC[INV0], invoking external request 0 TNUM. User must wait until this bit is cleared by the FMan.
4. Substitute the old Action Descriptor with the new Action Descriptor, updating the first 4 bytes last. That is, the user must copy the new Action Descriptor into the old Action Descriptor in reverse order.
5. Perform SYNC via external request 0 mechanism, which may be done only by hypervisor. User must assert a request by setting FMFP\_EXTC[INV0], invoking external request 0 TNUM. User must wait until this bit is cleared by the FMan.
6. All old Action Descriptor corresponding data structures are free and ready for deallocation. Note that since the 16 bytes of the new Action Descriptor are actually copied to the 16 bytes of the old Action Descriptor the user may now deallocate the new Action Descriptor memory.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type=11				—											New AD pointer [0-7]
2																New AD pointer [8-23]

**Figure 5-457. Old AD First 4 Bytes Temporary Value (Type = 11)**

**Table 5-510. Old AD First 4 Bytes Temporary Value (Type = 11)**

Offset	Bits	Name	Description
0	0–1	Type	AD Type. Set to 11.
	2–7	—	Reserved
	8–31	New AD pointer	Pointer to the new AD

#### 5.12.14.1.2 Direct Table access Host Command Sync Flow

The following is the second flow, which uses host commands for SYNC, and may be used by any host:

1. Prepare the new custom classifier Action Descriptor and all of its corresponding structures.
2. Substitute the first four bytes of the old Action Descriptor with the information shown in [Figure 5-457](#).
3. Perform SYNC via provided SYNC host command. Wait for host command completion.
4. Substitute the old Action Descriptor with the new Action Descriptor, updating the first 4 bytes last. That is, the user must copy the new Action Descriptor into the old Action Descriptor in reverse order.
5. Perform SYNC via provided SYNC host command. Wait for host command completion.
6. All old Action Descriptor corresponding data structures are free and ready for deallocation. Note that because the 16 bytes of the new Action Descriptor are actually copied to the 16 bytes of the old Action Descriptor the user may now deallocate the new Action Descriptor memory.

#### 5.12.14.1.3 Full Host Command Flow

The following is the third flow, which uses host commands for dynamic update of custom classifier tables, and may be used by any host.

1. Prepare the new custom classifier Action Descriptor and all of its corresponding structures.
2. Perform dynamic update of custom classifier tables host command. Wait for host command completion.
3. All old Action Descriptor corresponding data structures are free and ready for deallocation. Note that since the 16 bytes of the new Action Descriptor are actually copied to the 16 bytes of the old Action Descriptor the user may now deallocate the new Action Descriptor memory.

## 5.12.15 Ethernet Independent Mode (IM)

### 5.12.15.1 IM Introduction

The Ethernet independent mode (IM) uses legacy BD ring structures as the interface to the hosts for transmission or reception of frames rather than using the hardware blocks such as QMan, BMan, hardware Parser and KeyGen. User must still initialize the FMan DMA block, FMan memory, the BMI block and the FPM. This mode is useful for the boot process before all hardware is initialized. It can also be useful if an application does not wish to initialize all hardware blocks for simplicity reasons. This mode may also benefit development and debug efforts.

### 5.12.15.2 IM Features

- Enables each MAC to work in Ethernet independent mode
- Promiscuous mode - all frames are stored in Rx Queue
- Multi buffer data structure for frame storage for both Rx and Tx
  - User programmable buffer size in Rx (MRBLR)
- One queue descriptor on Rx
  - Rx queue descriptor manages a ring of Buffer Descriptors (BDs) pointing to data buffers
- One queue descriptor on Tx
  - Tx queue descriptor manages a ring of Buffer Descriptors (BDs) pointing to data buffers
- Statistics
  - Rx Queue Busy counter
- Receiver network management and diagnostics
  - Receive frame interrupt
  - Busy (out of buffers) interrupt
  - Rx Error indication on the RxBD (Frame too long, too short, OV, CRC error)
- See [Section 5.5, "Frame Manager—Buffer Manager Interface \(BMI\),"](#) for more available MAC features and statistics
- Ethernet Media Access Controller for more available MAC features and statistics
- Graceful Stop
  - Stop transmission between adjacent complete frames.

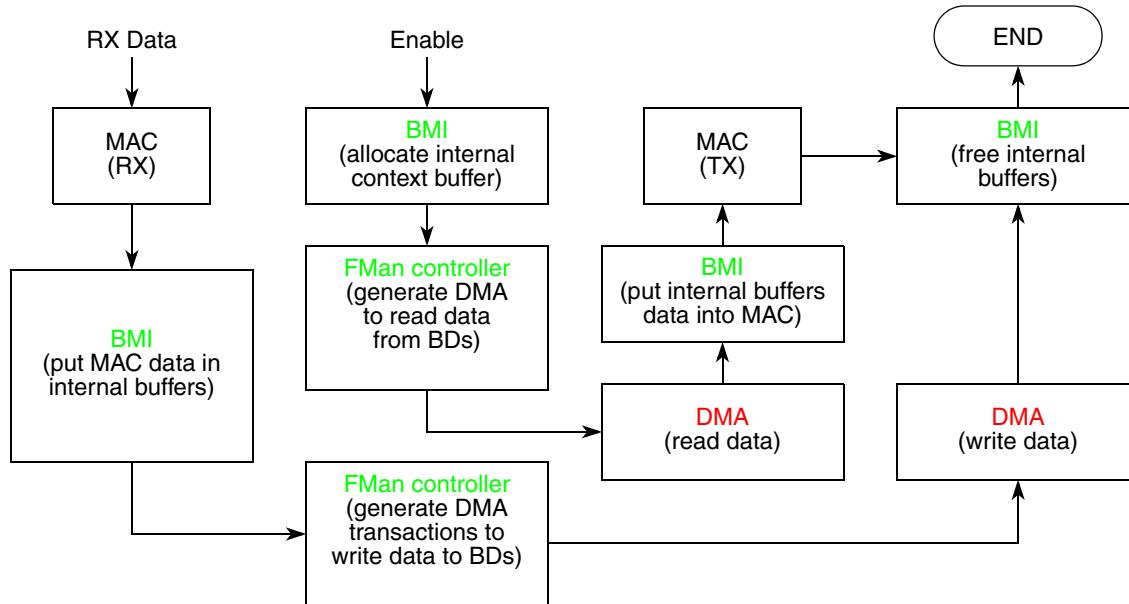
### 5.12.15.3 Frame Transmitter Overview

The transmission flow starts with the MAC enabling the BMI which asserts a request to the FMan controller, conveying the location of the internal context buffer. At this point the FMan controller allocate internal FMan memory buffers. The FMan controller fills the allocated internal buffers with valid data read from the external buffer once the BDs are ready (TxBD[R] is set). When the entire frame is available in the internal buffers the FMan controller requests BMI to transmit the frame using the appropriate NIA. The BMI then transfers the data from the internal buffers to the MAC, waits until the MAC completes

transmission of each internal buffer, and then frees the internal buffer. For more information about the transmitter flow refer to [Figure 5-458](#).

For stopping transmission user should assert Graceful-Stop. This operation assures nice stop between adjacent complete frames. When switching a transmit port from IM to Normal mode user must assert Graceful-Stop to return all open TNUMs and resources back to the FPM. For more information about this process refer to [Section 5.12.15.6, "IM Application Notes."](#)

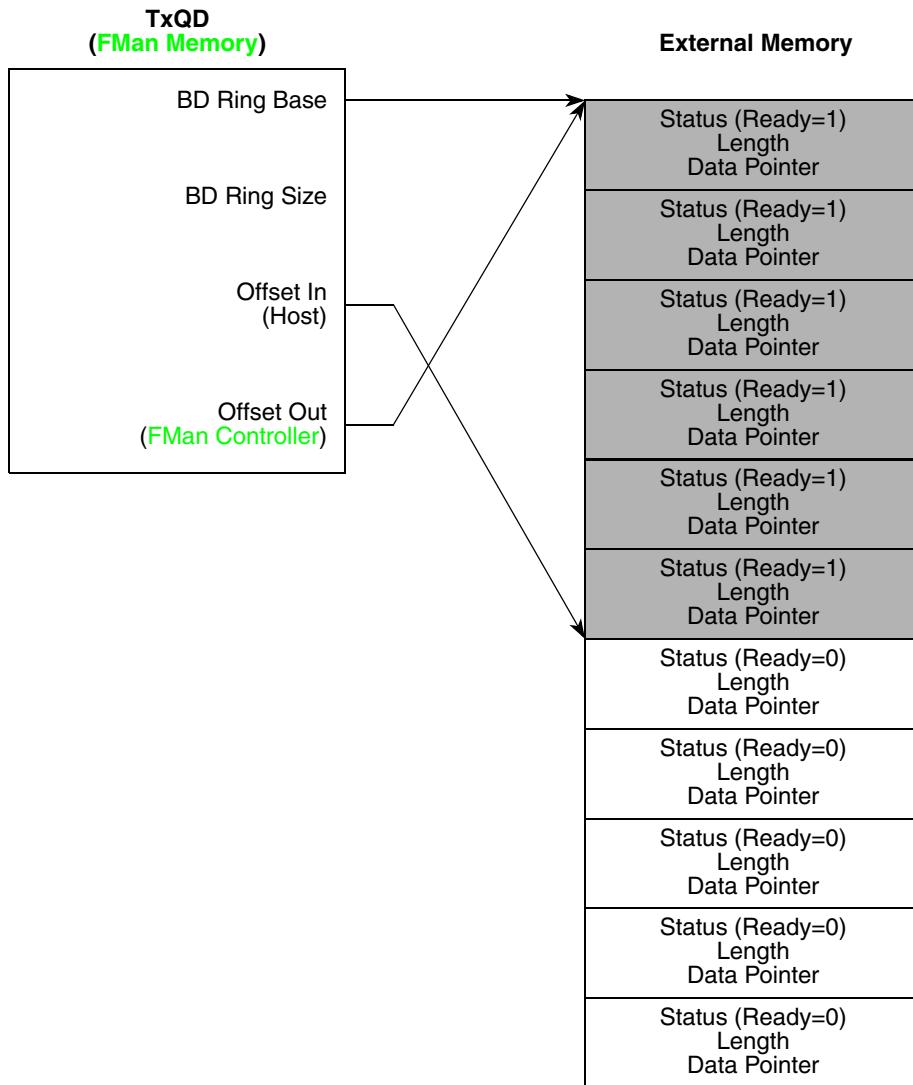
There is one transmit queue descriptor (TxQD) which is located in FMan memory. This TxQD is used to point to a BD ring located in external memory.



**Figure 5-458. Independent Mode Flow**

#### NOTE

User must associate only one FMan controller to the Tx port while in independent mode.



**Figure 5-459. Transmit Structures**

#### 5.12.15.4 Frame Receiver Overview

The reception process begins when the MAC interface receives the data and passes it to the BMI. The BMI transfers the data coming from the MAC into internal FMan memory buffers. At this point the FMan controller reads the BDs from external memory and writes the data from the internal buffers to the external buffers. It uses the MRBLR value to determine how much data to write to each BD buffer pointer. When the entire frame is completely received, the FMan controller requests BMI initiation via the FPM. Next, the BMI releases the internal buffers, internal context, and frees the TNUM.

There is only one receive queue descriptor (RxQD) which is located in FMan memory. The receive queue descriptor points to a BD ring located in external memory. Only promiscuous mode is supported, which means that all frames are transferred to the RxQD.

## NOTE

User must associate only one FMan controller to the Rx port while in independent mode.

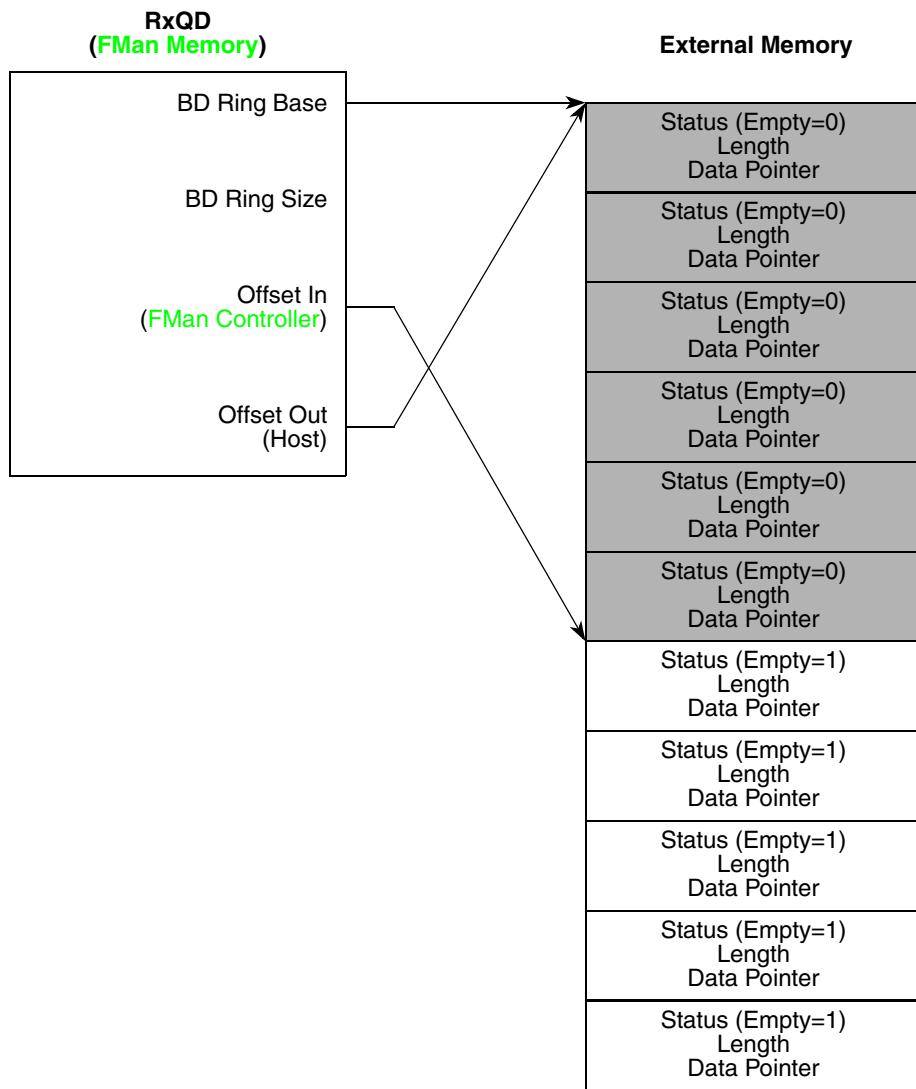


Figure 5-460. Receive Structures

### 5.12.15.5 IM Programming Model

#### 5.12.15.5.1 Parameter RAM

Each MAC that uses independent mode has its own global parameter RAM. This global parameter RAM can be common for both Rx and Tx or also can be managed separately (in this case the Rx/Tx relevant parameters are noted in a special column in [Table 5-511](#)). The Independent mode global parameter RAM pointer must be initialized using the BMI Rx Frame Queue ID register (See [Section 5.5.4.5.15, "Rx Frame Queue ID Register \(FMBM\\_RFQID\)."](#) ) and the BMI Tx Confirmation Frame Queue ID register (See [Section 5.5.4.6.9, "Tx Confirmation Frame Queue ID Register \(FMBM\\_TCFQID\)"](#))

**Table 5-511. Independent Mode Global Parameter RAM**

Offset	Bits	Rx/Tx	Name	Description
0x0	0–31	Rx/Tx	Independent Mode Register	Independent Mode Register. See <a href="#">Table 5-512</a> .
0x4	0–31	Rx	RxDQ ptr	Receive Queue Descriptor pointer. RxDQ size is 32 bytes. Should be 8 byte aligned. User must set this pointer to Independent mode page address + 0x20.
0x8	0–31	Tx	TxDQ ptr	Transmit Queue Descriptor pointer. TxDQ size is 32 bytes. Should be 8 byte aligned. User must set this pointer to Independent mode page address + 0x40.
0xC	0–15	Rx	MRBLR	Maximum Receive Buffer Length. When frame receive size is bigger than this value, the frame is spread over more than one buffer according to MRBLR partitions. User should set this value to the power of 2 argument and the result must be bigger or equal to 256. For example to get buffer size up to 256 user should set 8 in the MRBLR value, and for getting buffer size up to 1024 user should set 10 in the MRBLR value.
	15–31	Rx	RxDQ BSY CNT	RxDQ0 busy counter. Should be cleared.
0x10–0x1F	—	Rx/Tx	—	Should be cleared
0x20–0x3F	—	Rx	RxDQ	RxDQ queue descriptor. See description in <a href="#">Table 5-513</a> .
0x40–0x5F	—	Tx	TxDQ	TxDQ queue descriptor. See description in <a href="#">Table 5-515</a> .
0x60–0xFF	—	Rx/Tx	—	Should be cleared.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	—	GBL	—	—	—	Graceful-Stop	—	—	—	—	—	—	—	—	—	—
2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

**Figure 5-461. Independent Mode Register**

**Table 5-512. Independent Mode Register Field Descriptions**

Bits	Name	Description
0–1	—	Reserved
2	GBL	Global. Setting GBL enables snooping of data buffers and BDs.

### 5.12.15.5.2 Rx Queue Descriptor

The Rx queue descriptor is a table that describes the associated BD ring.

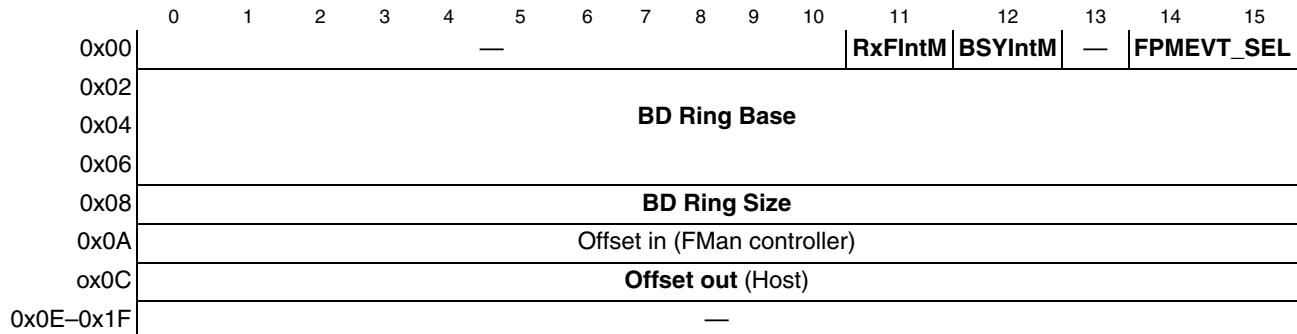


Figure 5-462. Rx Queue Descriptor

Table 5-513. Rx Queue Descriptor Field Descriptions

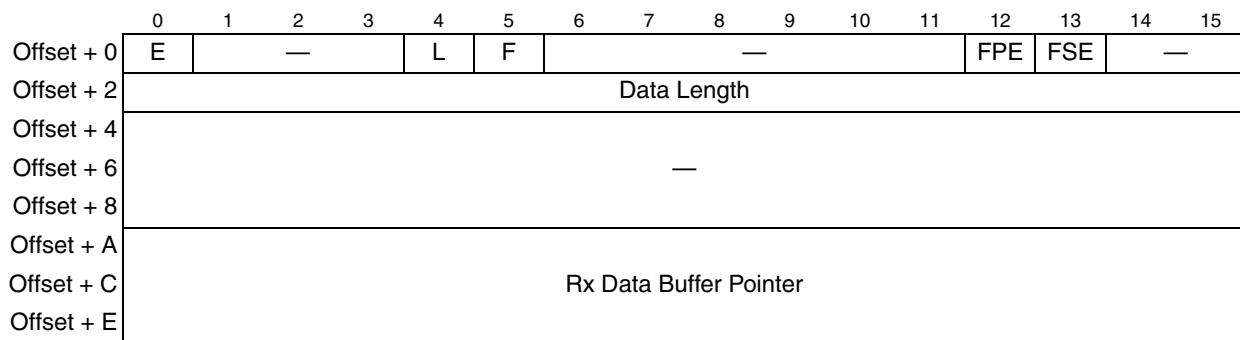
Offset	Bit	Name	Description
0x00	0–10	—	Reserved
	11	<b>RxFIntM</b>	RxF Interrupt mask. 0 - no interrupt on receive frame. 1 - set interrupt for receive frame. For more information see <a href="#">Section 5.12.22, "FMan Controller Event Register."</a>
	12	<b>BSYIntM</b>	RxBD busy Interrupt mask. 0 - no interrupt on receive busy event. 1 - set interrupt for receive busy event. For more information see <a href="#">Section 5.12.22, "FMan Controller Event Register."</a>
	13	—	Reserved
	14–15	<b>FPMEVT_SEL</b>	FPM Event 0-3 select. Indicated which of the FPM FMan controller event registers (FMFP_CEVO-3) is selected for this RxQD. For more information see the FPM chapter. <b>Note:</b> User can enable interrupt/clear the events by setting the corresponding bits in the FMFP_CEE0-3/FMFP_CEVO-3.
0x02	0–15	<b>BD Ring Base (0–15)</b>	BD Base Address. Address of first BD in the Rx BD ring (bits 0–15). Bits 0–11 are reserved and should be set to 0; bits 12–15 are used to accommodate 36-bit addressing.
0x04	0–31	<b>BD Ring Base (16–47)</b>	BD Base Address. Address of first BD in the Rx BD ring (bits 16–47).
0x08	0–15	<b>BD Ring Size</b>	Size in byte units of all the BD ring. Equal to number of BDs multiplied by 16.
0x0A	0–15	Offset in	Offset from the BD base to the next BD in the BD ring. Initialize to 0. Managed by the FMan controller. <b>Note:</b> When all the buffers are full then Offset in precede Offset Out by 1 entry only. This means that the maximum number of full BDs is BD ring size minus one.

**Table 5-513. Rx Queue Descriptor Field Descriptions (continued)**

Offset	Bit	Name	Description
0x0C	0–15	<b>Offset Out</b>	Offset to the next BD to be fetched by the Host. Updated by the host. Initialized to zero. Then host still need to check that the BD[E] bit is cleared before updating the BD and advancing Offset Out.
0x0E–0x1F	—	—	Reserved

### 5.12.15.5.3 RxBD

The RxBD is a 16-byte structure that contains the information about the buffer pointer and frame size.



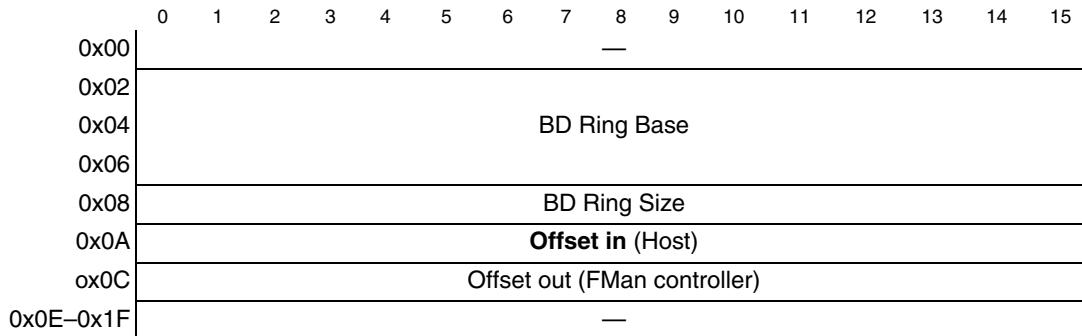
**Figure 5-463. RxBD data structure**

**Table 5-514. RxBD Data Structure Field Descriptions**

Offset	Bits	Name	Description
0x00	0	<b>E</b>	<p>Empty</p> <p>0 The buffer associated with this RxBD is full or reception terminated due to an error. The core can examine or read to any fields of this RxBD. The FMan controller does not use this BD as long as E = 0.</p> <p>1 The associated buffer is empty. The RxBD and buffer are owned by the FMan controller. Once E = 1, the core should not write any fields of this RxBD.</p>
	1–3	—	Reserved, should be cleared.
	4	<b>L</b>	<p>Last in frame. Set by the FMan controller when this buffer is the last in a frame. This implies the end of the frame or a reception error, in which case one or more of the CRE, FTL, FTS and OV bits are set. The FMan controller writes the number of frame octets to the data length field.</p> <p>0 Not the last buffer in a frame.</p> <p>1 Last buffer in a frame.</p>
	5	<b>F</b>	<p>First in frame. Set by the Ethernet controller when this buffer is the first in a frame.</p> <p>0 Not the first buffer in a frame.</p> <p>1 First buffer in a frame.</p>
	6–11	—	Reserved, should be cleared.
	12	<b>FPE</b>	Frame Physical Error: One of the following event were detected: Rx FIFO overflow, FCS error, code error, running disparity error (SGMII and TBI modes), FIFO parity error. PHY Sequence error, PHY error control character detected. Asserted by the BMI.
	13	<b>FSE</b>	Frame Size Error: According to the frame size in the FD the user determines one of the following error options Frame too long - Frame size exceeds max_length_frame (programmable). Asserted by the BMI.
	14–15	—	Reserved, should be cleared.
	0–15	Data Length	Data Length of this buffer. If L bit is set, this field contains the total frame length. If the received frame contains CRC, then this length also includes the CRC additional bytes (Subject to Ethernet controller programming)
0x04	0–31	—	Reserved, should be cleared.
0x08	0–15	—	Reserved, should be cleared.
0x0A	0–15	<b>Rx data buffer pointer (0–15)</b>	Rx data buffer pointer (bits 0–15). Initialized by the Host. Bits 0–11 are reserved and should be set to 0; bits 12–15 are used to accommodate 36-bit addressing.
0x0C	0–31	<b>Rx data buffer pointer (16–47)</b>	Rx data buffer pointer (bits 16–47). Initialized by the Host.

### 5.12.15.5.4 Tx Queue Descriptor

The Tx queue descriptor is a table that describes the associated BD ring.



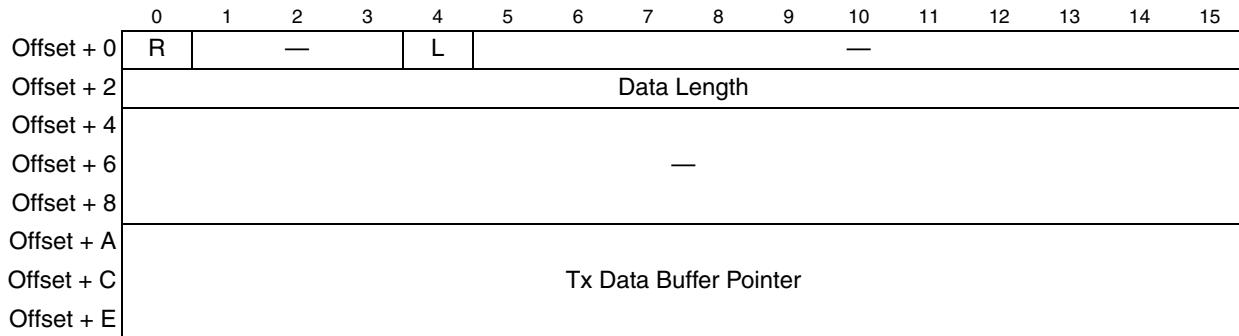
**Figure 5-464. Tx Queue Descriptor**

**Table 5-515. Tx Queue Descriptor Field Descriptions**

Offset	Bit	Name	Description
0x00	0–15	—	Reserved
0x02	0–15	<b>BD Ring Base (0-15)</b>	BD Base Address. Address of first BD in the Tx BD ring (bits 0-15). Bits 0-11 are reserved and should be set to 0; bits 12-15 are used to accommodate 36-bit addressing.
0x04	0–31	<b>BD Ring Base (16-47)</b>	BD Base Address. Address of first BD in the Tx BD ring (bits 16-47).
0x08	0–15	<b>BD Ring Size</b>	Size in byte units of all the BD ring. Equal to number of BDs multiplied by 16.
0x0A	0–15	<b>Offset in</b>	Offset to the first BD in the frame to be written by the Host. Updated by the host. Initialized to zero. The host must advance Offset in <b>only</b> after all the frame is located in the BDs with Ready bit set. <b>Note:</b> When all the buffers are full then Offset in precede offset out by 1 entry only. This means that the maximum number of full BDs is BD ring size minus one. Before using a BD for transmission the host must check first that the BD[R] bit is cleared.
0x0C	0–15	Offset out	Offset from the BD base to the next BD in the BD ring. Initialize to 0. Managed by the FMan controller
0x0E-0x1F	—	—	Reserved

### 5.12.15.5.5 TxBD

The TxBD is a 16 byte structure containing information about the buffer pointer and frame size.



**Figure 5-465. TxBD Data Structure**

**Table 5-516. TxBD Data Structure Field Descriptions**

Offset	Bits	Name	Description
0x00	0	<b>R</b>	Ready 0 The buffer associated with this BD is not ready for transmission; the user can manipulate this BD or its associated buffer. The FMan Controller clears the R bit to indicate that the buffer has been sent. 1 The buffer is ready to be sent. The buffer is either waiting or in the process of being sent. The user cannot change fields in this BD or its associated buffer once R = 1.
	1–3	—	Reserved, should be cleared.
	4	<b>L</b>	Last in frame. Set by the Host when this buffer is the last in a frame. 0 Not the last buffer in a frame. 1 Last buffer in a frame. Data length reflect data size in this buffer only.
	5–15	—	Reserved, should be cleared.
	0–15	<b>Data Length</b>	Data Length of this buffer. Set by the Host.
0x04	0–31	—	Reserved, should be cleared.
0x08	0–15	—	Reserved, should be cleared.
0x0A	0–15	<b>Tx data buffer pointer (0–15)</b>	Tx data buffer pointer (bits 0–15). Set by the Host. Bits 0–11 are reserved and should be set to 0; bits 12–15 are used to accommodate 36-bit addressing.
0x0C	0–31	<b>Tx data buffer pointer (16–47)</b>	Tx data buffer pointer (bits 16–47). Set by the Host.

### 5.12.15.6 IM Application Notes

- The Independent mode global parameter RAM pointer must be initialized using the BMI Rx Frame Queue ID register (See [Section 5.5.4.5.15, "Rx Frame Queue ID Register \(FMBM\\_RFQID\)."](#) ) and

the BMI Tx Confirmation Frame Queue ID register (See [Section 5.5.4.6.9, "Tx Confirmation Frame Queue ID Register \(FMBM\\_TCFQID\)"](#)). See [Section 5.12.15.5.1, "Parameter RAM."](#)

- Initialize the FMBM\_RFCA or FMBM\_TFCA register per port. This register programs the FMan controller MR[0:5], which determines the mode attributes. The user must initialize MR[3] to 1, assuring extended address mode for external memory accesses. The user should clear both FMBM\_TFCA[OR] and FMBM\_RFCA[OR], since the ordering process if any is controlled only internally by the FMan controller. See [Section 5.5.4.6.8, "Tx Frame Attributes Register \(FMBM\\_TFCA\)"](#), and [Section 5.5.4.5.10, "Rx Frame Attributes Register \(FMBM\\_RFCA\)"](#).
- User must associate only one FMan controller to the Tx/Rx ports while in independent mode. This is done by setting FMFP\_PRC[CPA] (FPM Port ID Control) so that each even port ID corresponds to FMan controller #1 and each odd port ID corresponds to FMan controller #2.
- The BMI internal buffer allocation must not begin from offset 0 in FMan memory. FMBM\_CFG1[FBPO] must be greater than 0.
- Clear FMFP\_MXD[DISP\_LIM] to assure that the FPM dispatch limit mechanism is disabled when IM is enabled.
- Clear FMBM\_RIM[FOF]. Rx internal frame margin is not supported for the IM Rx port.
- FMBM\_RCFG[FDOVR] dictates BMI behavior upon the reception of an error frame. If FDOVR is cleared the received errored frame is discarded. If FDOVR is set the received errored frame continues to the IM Rx BD with an error indication. The value written in FMBM\_REFQID has no meaning for IM.
- Set FMBM\_SPICID to specify ICID associated to the IM Tx or Rx port. This allow DMA transactions to be authenticated and translated as needed before entering the memory interconnect.
- Set FMBM\_RCFG[IM] to enable IM on the Rx port.
- Set FMBM\_TCFG[IM] to enable IM on the Tx port.
- When switching transmit port from working in IM to Normal mode application must assert Graceful-Stop using the following process-
  - Clear FMBM\_TCFG[EN] bit in order to disable transmit port and leave FMBM\_TCFG[IM] bit still set.
  - Set Graceful-Stop bit in the Independent Mode register. See [Table 5-512](#).
  - Wait till FMBM\_TST[BSY] bit is cleared to assure that port is not busy.
  - Clear FMBM\_TCFG[IM] bit and set FMBM\_TCFG[EN] bit to re-enable transmit in normal mode.

## 5.12.16 Host Commands

### 5.12.16.1 Host Command Introduction

Host commands enable port virtualization and free load from the cores. The available host commands are:

- Init and Update
  - Policer Profile
  - KeyGen Scheme

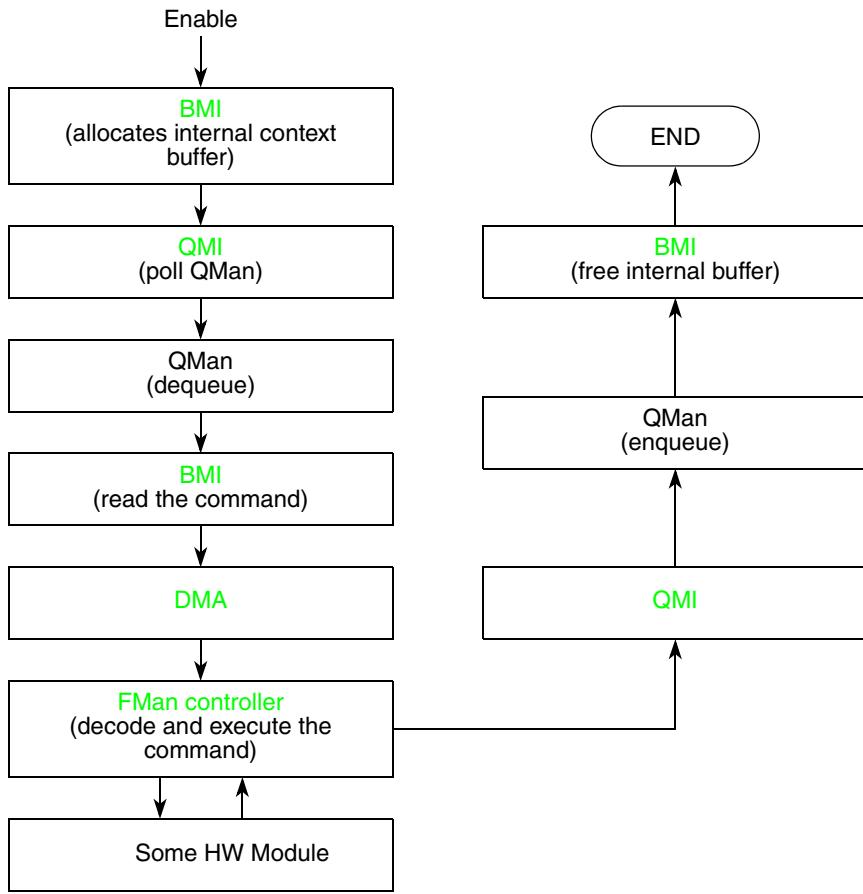
- Assistance for dynamic update of custom classifier tables
  - SYNC
  - Dynamic update of custom classifier tables
- Aging support

### **5.12.16.2 Host Command Features**

- Support for host commands through QMan
- Statistics read/write commands
- Policer Profile Host command
  - Support for Read/Write access for configuration registers and profile entry fields
  - Read full profile entry (64 bytes)
  - Update partial or full profile entry
- KeyGen Host Command
  - KeyGen scheme update host command
    - Support error write indication
  - KeyGen classification plan update host command
  - KeyGen port partition configuration update host command
  - Support Read/Write access KeyGen scheme (up to 92 bytes each entry)
  - Support Read/Write access classification plan partition (32 bytes each partition, 4 bytes each CP entry)
  - Support Read/Write access KeyGen port partition (8 bytes)
- Sequence number support for frame confirmation
- Assistance for dynamic update of custom classifier tables
- Aging support

### **5.12.16.3 Host Command Flow**

Host commands are dequeued from the QMan. After the completion of the host command the result and confirmation are transferred to the software through the QMan. For this purpose the user must initialize the Tx Confirmation Frame QID in the FMBM\_TCFQID. The FMan controller decodes the host command from a frame that the host prepares for transmission according to a specific host command template described below. [Figure 5-466](#) shows a general flow for host commands.

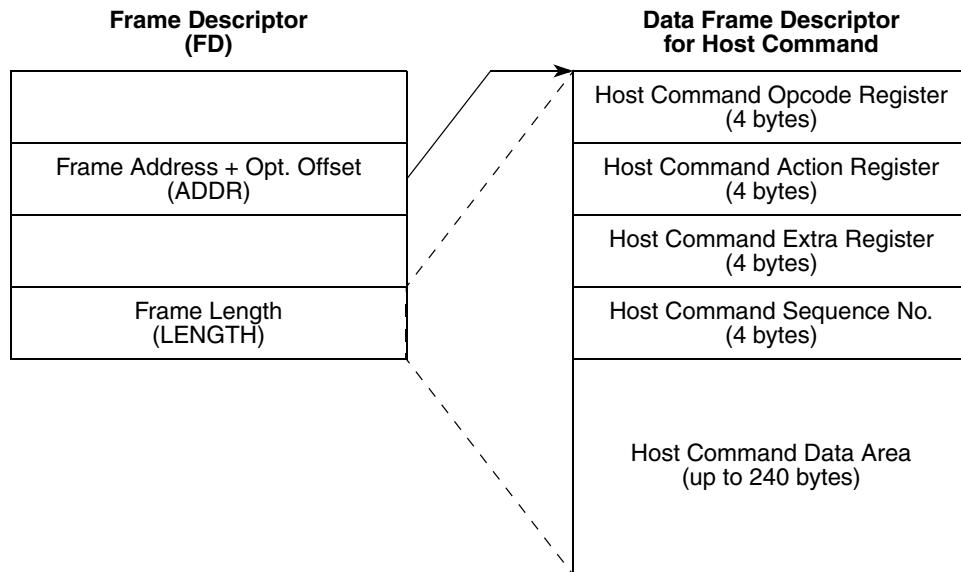


**Figure 5-466. Host Command Flow**

The BMI allocates an internal frame context buffer and the QMI polls for FD from the QMan. After the QMan dequeues FD from a FQ the BMI generates a DMA transaction to read the command. The DMA reads the command data and, after verifying that the DMA read has finished, the FMan controller decodes the host command from the contents of the buffer and processes it. The QMI enqueues the host command result/confirmation on a queue through QMan. Finally, the BMI frees internal buffers and the internal context of the command.

#### 5.12.16.4 Host Command Programming Model

The FMan controller decodes the host command from the first word of the buffer and determines the host command type. [Figure 5-467](#) shows the descriptor of the frame for a host command. The user must prepare the host command FD according to the following data structures.



**Figure 5-467. Data Frame Description for Host Command (External Memory)**

In order to support atomic operations for host commands, the user must configure a single port designated only for host commands and set only one TNUM available for this port.

#### 5.12.16.4.1 Host Command Opcode Register (HCOR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	—	GBL	—								—					
2			—													Opcde

**Figure 5-468. Host Command Opcode Register**

**Table 5-517. Host Command Opcode Register Description**

Offset	Bits	Name	Description
0	0–1	Reserved	Must be cleared
	2	GBL	Global. Setting GBL enables snooping of BMan data buffers.
	3–15	Reserved	Must be cleared

**Table 5-517. Host Command Opcode Register Description (continued)**

Offset	Bits	Name	Description
2	0–7	Reserved	Must be cleared
	8–15	Opcode	Host Command Opcode. 0x00 - Policier Profile Host Command. 0x01 - KeyGen Host Commands. 0x02 - SYNC. For more information see <a href="#">Section 5.12.14.1, "Dynamic Update of Custom Classifier and HM Tables."</a> 0x03 - Dynamic update of custom classifier tables. For more information see <a href="#">Section 5.12.14.1, "Dynamic Update of Custom Classifier and HM Tables."</a> 0x04 - Aging support - On P4080 rev1 this feature is not supported. Enable the host to set a specific bit or bits (according to the corresponding KEY or KEYS) in the Generic_6_Off_IC_AGE_MASK 'AGE MASK' field in an atomic transaction. 0x10 - IP Reassembly Timeout Configuration Command. 0x11 - IP Fragmentation Host Command.  0x13 - Dynamic update of custom classifier tables when the old table to be replaced is with operation code Generic_6_Off_IC_AGE_MASK (0x2D) and user plan to update, add or remove entries from this table. Other Opcodes - Reserved.

#### 5.12.16.4.2 Host Command Action Register (HCAR)

##### Policer Profile and KeyGen

The host command action register uses the platform of the FMan Policier Profile Action Register (FPLPAR) and the KeyGen Action Register (KGAR).

The read/write operation uses the atomic access sequence to select profile entry, scheme number, and classification plan group. An atomic write access does not interfere with process packet profile or scheme updates. An atomic read access provides coherent data without interference from process packet profile/scheme updates.

During an atomic write action these registers are write-protected until the operation is completed and a write access gets wait states. During an atomic read action these registers are pending for updated data and any read from them gets wait states until the action is complete to reflect the latest data.

Figure 5-469 describes the Policier Profile Host Command Action Register. For fields description, see [Section 5.11.4.14, "FMan Policier Profile Action Register \(FMPL\\_PAR\)."](#)  If the profile mapping option, HCER[PMO], is zero the host command calculates the PNUM subject to the original PNUM supplied by the HCAR[PNUM] and HCER[PortID] configurations which are defined by FMPL\_PMRx or FMPL\_DPMR. Upon host command completion the HCAR[PNUM] is updated with the calculated PNUM (absolute PNUM) for both write/read commands in order to be visible to the host. If the profile mapping option, HCER[PMO], is one the PNUM is supplied by HCAR[PNUM].

This figure describes the KeyGen Scheme/Classification Plan/Port Partition Host Command Action register. For field descriptions, see [Section 5.10.3.11, "KeyGen Action Register \(FMKG\\_AR\)"](#).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	RW	PSI	0												PNUM
2																PWSEL

**Figure 5-469. Policer Profile Host Command Action Register**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	RW	ER					SEL								NUM
2																PORTID

**Figure 5-470. KeyGen Scheme/Classification Plan/Port Partition Host Command Action Register**

Writing to the KeyGen scheme can create an error if the Port ID does not match the scheme, causing the Action Register [ER] bit to be set. The user must verify it in the host command Action Register in the return buffer area.

## SYNC and Dynamic Update of Custom Classifier Tables

A SYNC command does not require HCAR initialization. Dynamic update of custom classifier tables, however, does require initialization of the HCAR according to the template in [Figure 5-457](#).

## Aging Support

Enable the host to set a specific bit (according to the corresponding KEY) in the Generic\_6\_Off\_IC\_AGE\_MASK ‘AGE MASK’ field in an atomic transaction. This is useful for HASH function, as described in [Section 5.12.5.4, "Using KeyGen HASH Result for Indexed Look-Up."](#) HCAR consists of the internal memory pointer of the Generic\_6\_Off\_IC\_AGE\_MASK Action Descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				—												Generic_6_Off_IC_AGE_MASK AD ptr[0-7]
2																Generic_6_Off_IC_AGE_MASK AD ptr[8-23]

**Figure 5-471. HCAR for Aging Support**

## 5.12.16.4.3 Host Command Extra Register (HCER)

### Policer Profile and KeyGen

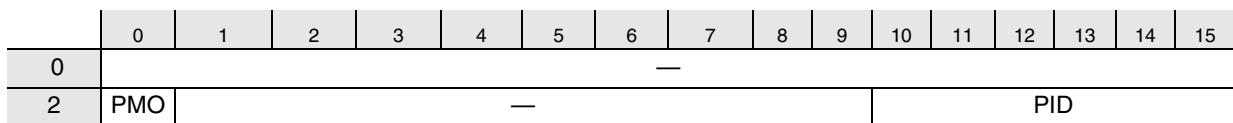
This register holds the KeyGen scheme write select bits which are needed only for a scheme write host command or for a Read/Write Policer profile host command.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																KEYGSSEL
2																—

**Figure 5-472. KeyGen Scheme Host Command Extra Register**

**Table 5-518. KeyGen Scheme Host Command Extra Register Field Descriptions**

Offset	Bits	Name	Description
0	0–15	KEYGSSEL [0-15]	KeyGen Scheme Write Select - Valid only for the KeyGen Scheme write command.
2	0–4	KEYGSSEL [16-20]	This is 23 bit selection mask for KeyGen Scheme write command. Each bit is set for writing one of the KGSE_* registers. A field associated with cleared KEYGSSEL bit is not written by the host command. In case that KGSE_SPC bit is set, meaning that the user want to update the Statistic Packet Counter, the user should also set the HCAR[WSEL(0)] bit. Description of the KeyGen Entry (KGE) Access Register can be found <a href="#">Section 5.10.3.11,"KeyGen Action Register (FMKG_AR)." </a>
5–15	Reserved		Must be cleared



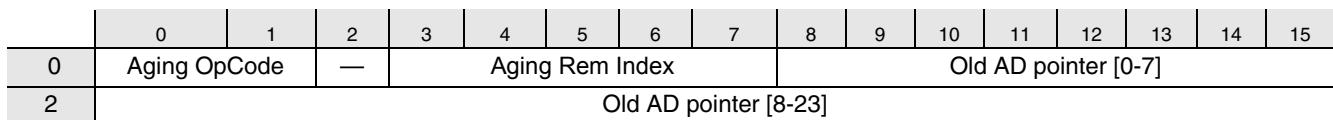
**Figure 5-473. Policer Profile Host Command Extra Register**

**Table 5-519. Policer Profile Host Command Extra Register Field Descriptions**

Offset	Bits	Name	Description
0	0–15	Reserved	Must be cleared
2	0	PMO	Profile Mapping Option. This bit indicates if the PNUM provided in HCAR is taken as absolute profile number or is an input to the PNUM calculation - according to the port-ID and its configuration registers FMPL_PMRx or FMPL_DPMR (default). 0 - Relative PNUM profile calculation - according to the port-ID and its configuration registers FMPL_PMRx or FMPL_DPMR (default). 1 - PNUM represents an absolute profile number.
1–9	Reserved		Must be cleared
10–15	PID		Port-ID. Valid only if PMO is cleared and reflects the PORT ID from which the configuration registers are taken. If this port ID does not match an enabled FMPL_PMRx register, the profile is selected through FMPL_DPMR.

### SYNC and Dynamic Update of Custom Classifier Tables

A SYNC command does not require HCER initialization. Dynamic update of custom classifier tables, requires initialization of the HCER as described in [Figure 5-474](#) and in [Table 5-520. “HCER for Dynamic Update of Custom Classifier Tables Command description.”](#) For more information refer to [Section 5.12.14.1,"Dynamic Update of Custom Classifier and HM Tables."](#)



**Figure 5-474. HCER for Dynamic Update of Custom Classifier Tables Command**

**Table 5-520. HCER for Dynamic Update of Custom Classifier Tables Command description**

Offset	Bits	Name	Description
0	0-1	Aging OpCode	Valid only if the old table descriptor is with aging support (Generic_6_Off_IC_AGE_MASK operation code is 0x2D) and user plan to update, add or remove entries from this table, otherwise must be cleared. 0 - Update entries in the look-up table that supports aging. Keys are not updated in this case. 1 - User removes entries from the look-up table that supports aging. In this case the field "Aging Rem Index" field indicates which key position is removed. 2 - User adds entries to the look-up table that supports aging.
	2	Reserved	Must be cleared
	3-7	Aging Rem Index	Valid only if the old table descriptor is with aging support (Generic_6_Off_IC_AGE_MASK operation code is 0x2D) and user plan to add or remove entries from this table, otherwise must be cleared. Valid only if "Aging OpCode" field is set to 1 (remove entries) and optional values are 0-30. Indicates the key index to be removed.
	8-15	Old AD pointer [0-7]	This field contains the Old Action Descriptor pointer [0-7].
2	0-15	Old AD pointer [8-23]	This field contains the Old Action Descriptor pointer [8-23].

## Aging Support

HCER is used for indicating which Age bit (or bits) to set in the AGE\_MASK field which resides in the Generic\_6\_Off\_IC\_AGE\_MASK. It is used as a mask that is or'ed with the existing AGE\_MASK field.

### NOTE

On command completion the user get into this field the AGE\_MASK field before setting the new bit (or bits). In this way the application identify which bits were cleared by the FMan-Controller before setting them.

### NOTE

User may set more than 1 bit at a time - for example user can set all of the AGE\_MASK bits. Notice that the last bit in the MASK always corresponds to the miss-matched KEY.



**Figure 5-475. HCER for Aging Support**

### 5.12.16.4.4 Host Command Sequence Number (HCSN)

The user may maintain a sequence number per host command in order to keep track of the completion order of host commands. This programming is optional and is performed by initializing the host command

sequence number field in the host command data buffer area. After the completion of the host commands the result of the host command and its confirmation are transferred to the software through the QMan so the user may verify its sequence number.

#### 5.12.16.4.5 Host Command Data Area

The host command data area is up to 240 bytes located at the FD buffer address (plus optional FD offset) at offset 16. This area is used for the interaction between host and FMan controller.

#### Policer Profile and KeyGen

The host command area description for the Policer Profile and KeyGen is:

- Write—the host offers the register values to be updated and the FMan controller writes them to the hardware registers.
- Read—the FMan controller reads all the hardware registers and updates their values in this area.

Separate data areas exist for Policer Profile, KeyGen Scheme, Classification plan, Port Partition, as noted below.

**Table 5-521. Host Command Data Area Description**

Offset	State	Name	Description
0x0-0x3C	HCOR[Opcode] = 0x00	PLPE registers	Read/Write Policer Profile Registers host command field. Read command - All 16 PLPE_* registers is written by the FMan controller to the data area, for host handling. Write command - The host need to initialize only the relevant PLPE_* registers, according to HCAR[PWSEL] bits, in a continuous manner. For more information see <a href="#">Section 5.11.4.15, "FMan Policer Profile Entry Access Word Registers (FMPL_PE*)"</a> .
0x0-0x50	HCOR[Opcode] = 0x01 HCAR[SEL] = 0	KGSE registers	Read/Write KeyGen Scheme Registers host command field. Read command - All 23 KGSE_* registers is written by the FMan controller to the data area, for host handling. Write command - The host need to initialize only the relevant KGSE_* registers, according to HCER[KEYGSSEL] bits, in a continuous manner. Description of the KeyGen Entry (KGE) Access Register can be found <a href="#">Section 5.10.3.11, "KeyGen Action Register (FMKG_AR)"</a> .
0x0-0x1C	HCOR[Opcode] = 0x01 HCAR[SEL] = 1	KeyGen Classification Plan	Read/Write KeyGen Classification Plan Registers host command field. Read command - All 8 Classification Plan (group) is written by the FMan controller to the data area (first 8 KGSE_* registers), for host handling. Description of the KeyGen Entry (KGE) Access Register can be found <a href="#">Section 5.10.3.11, "KeyGen Action Register (FMKG_AR)"</a> . Write command - The host need to initialize only the relevant KGSE_* registers, according to HCAR[WSEL] bits, in a continuous manner. For more information see <a href="#">Section 5.10.3.13.1, "KeyGen Classification Plan Entry Register (AWR&lt;1+i&gt;_RFMODE_FMKG_CPE&lt;i&gt;)"</a> .

**Table 5-521. Host Command Data Area Description (continued)**

0x0-0x08	HCOR[Opcode] = 0x01 HCAR[SEL] = 2	Port Partition Configuration	Read/Write KeyGen Port Partition Configuration Registers host command field. Read command - KGPE_SP & KGPE_CPP is written by the FMan controller to the data area (first 2 KGSE_* registers), for host handling. Write command - The host need to initialize only the relevant KGSE_* registers, according to HCAR[WSEL] bits, in a continuous manner. For more information see <a href="#">Section 5.10.3.14.1, "KeyGen Port Entry Scheme Partition Register (AWR1_RFMODE_FMKG_PE_SP),"</a> and <a href="#">Section 5.10.3.14.2, "KeyGen Port Entry Classification Plan Partition Register (AWR2_RFMODE_FMKG_PE_CPP)." </a>
----------	--------------------------------------	------------------------------	--

### **SYNC, Dynamic update of Custom Classifier Tables and Aging Support**

SYNC, dynamic update of custom classifier tables commands and Aging support do not require the data area initialization therefore Host command data area should not be allocated.

#### **5.12.16.5 Host Command Application Notes**

In order to support atomic operation for host commands, user should set single port per FMan which is destined only for host commands and set only one TNUM available for this port. This is performed by setting the BMI host command port parameters register as follows: FMBM\_PP\_x[MXT] = 0 and FMBM\_PP\_x[EXT] = 0.

After the completion of the host command the result of the host command and its confirmation is transferred to the software through the QMan. This confirmation FQID is determined by either FMBM\_OFQID (which is the default Tx Confirmation Frame QID), the QMan FQ context or the FD[CFQ] value in case that FD[FCO] bit is set.

#### **NOTE**

If FD[FCO] bit is set then FD[CFQ] value must not be equal to zero. This is because this FQID is used for host command confirmation.

Since internal context is not updated into external memory by the host command, the user should not define internal context in the Tx external buffer.

The user must initialize the FMBM\_OFCA for a host command port. This register programs the FMan controller MR[0:5], which determines the mode attributes. User must initialize MR[3] to 1. This assures extended address mode for external memory accesses.

Clear FMBM\_OIM[FOF]. O/H internal frame margin is not supported for the Host Command port.

#### **5.12.17 IP Acceleration Programming Model Additions**

##### **5.12.17.1 IP Fragmentation**

The IP Fragmentation function includes the following high-level features:

- Fragmentation of IP packets into IP fragments according to RFC 791 and RFC 2460.

- Support the following interface:
  - Offline Port
- Protocol Stack supported:
  - Ethernet
  - IP
- Configurable MTU per flow.
- Ordering is preserved between fragmented frames and regular frames.
- Ordering is preserved between fragmented frames and next fragmented frames.
- The input frame's Internal Context is copied to each fragment according to user configuration
- Statistics counters per flow.
- Input packet can be fragmented up to 16 fragments.
- On FMan\_v3 Virtual Storage Profile must be disabled on the port if IP fragmentation is enabled.

For IPv4, the ID field of the original frame is copied to each of its fragments. For IPv6, a unique ID is generated for each frame, as configured by the user, and this ID is copied to each of the frame's fragments.

Fragmentation of fragments is not supported.

### **5.12.17.1.1 IP Fragmentation Functional Description**

#### **IP Fragmentation Ordering**

The IP Fragmentation function preserves the order between fragments and regular frames with respect to the last fragment of each frame. That is, fragments of different frames may interleave and regular frames may appear between fragments, but the last fragment of each frame is guaranteed to preserve the original ordering.

For example, given frames R1,R2,R3,R4, where R2 is fragmented to F2,L2, and R3 is fragmented to F3,L3, the following sequences are considered ordered:

- **R1,F2,L2,F3,L3,R4**
- **F2,R1,L2,F3,L3,R4**
- **F2,R1,F3,L2,L3,R3**

#### **IP Fragmentation Process**

A frame is fragmented into fragments, each represented by a Scatter/Gather (S/G) list. The S/G list contains at least two entries:

- Entry that points to the header (Eth + IP header)
- Entry that points to the payload belonging to this fragment.

If the input frame is a S/G frame, then the fragment's payload may span over multiple S/G buffers and therefore the fragment will have more than 2 S/G entries.

In order to build a fragment, a new external buffer is allocated. This buffer is either allocated from the original frame's BPID or from the user-specified BPID in the IP Fragmentation Table Descriptor. This

buffer contains the S/G entries and the fragment's L2 header, IP header and unfragmentable part for IPv6 fragments. This buffer optionally contains the Internal Context portion that was configured (via FMBM\_OICP) to be written to external memory.

## IP Fragments Structure

IP Fragments are constructed by pointing to the original frame buffer(s) in order to avoid buffer copies. This results in multiple pointers that point to the same buffer. Only one of these multiple pointers is marked as belonging to a “Real BPID”. All other pointers are marked as belonging to “BPID NULL”. This enables correct de-allocation of these pointers, as described in [Section , "De-Allocation of IP Fragments."](#) The following diagrams illustrate the IP Fragments' structures that result from a single buffer frame and from a Scatter Gather frame.

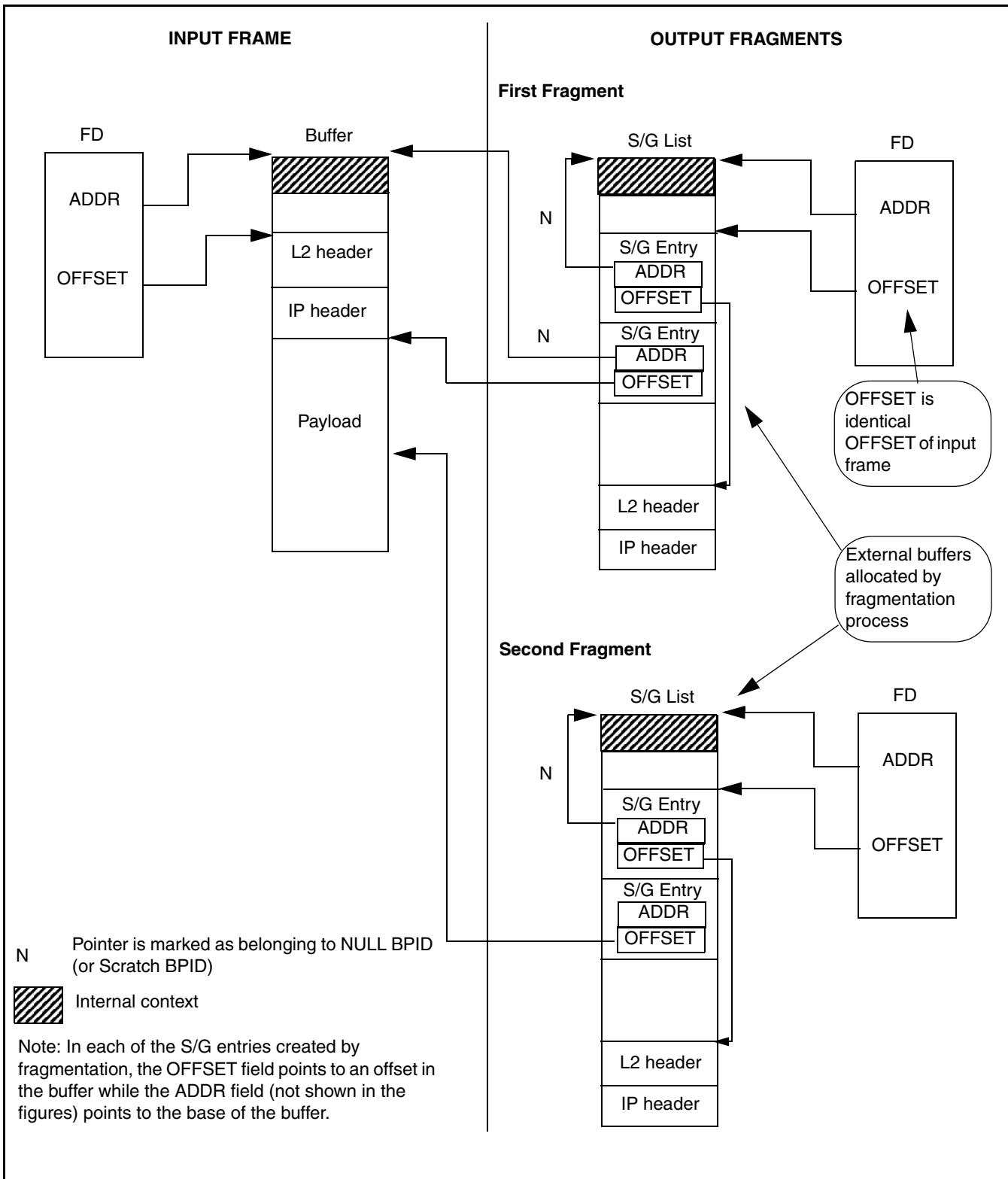


Figure 5-476. Single Buffer Frame Fragmentation

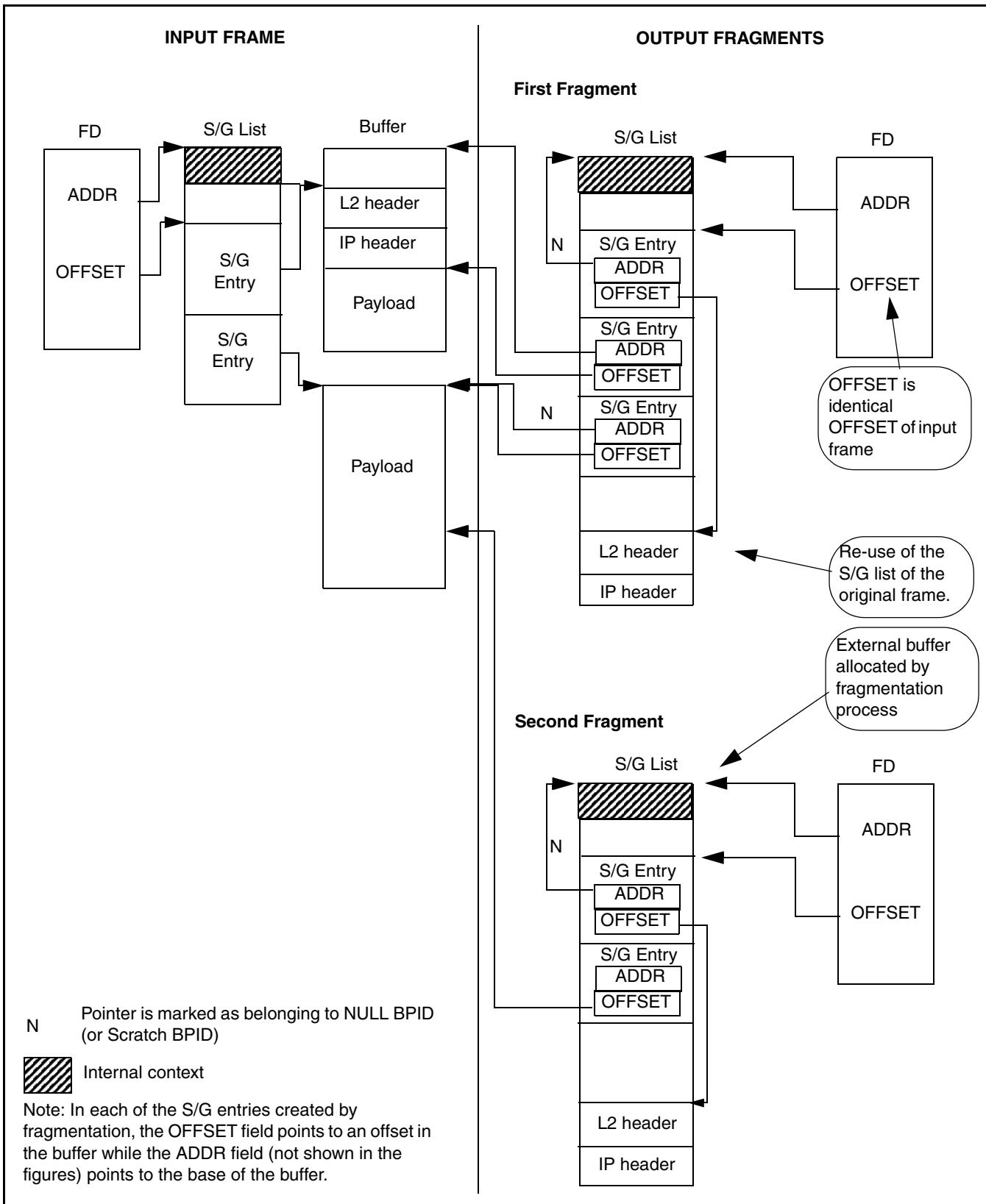


Figure 5-477. S/G Frame Fragmentation

During the fragmentation process, if the external buffer allocation fails, then the AllocFailureCounter is incremented and the FMan-Controller re-tries the allocation until it succeeds.

## Initialization of IP Fragmentation

1. On FMan\_v3 Virtual Storage Profile must be disabled on the port if IP fragmentation is enabled.
2. In order to enable fragmentation, two Table Descriptors must be configured: The IP Manipulation Table Descriptor and the IP Fragmentation Table Descriptor. The IP Manipulation Table Descriptor must define the MTU value and must forward processing to the IP Fragmentation Table descriptor (see [Section 5.12.17.4.1, "IP Manipulation Custom-Classification Table Descriptor"](#) and [Section 5.12.17.4.2, "IP Fragmentation Table Descriptor"](#) for additional initialization information of these table descriptors).
3. The Frame Offset (FOF) programmed for the fragmentation port must be big enough to store  $n * 16$  bytes where n represents the maximum number of Scatter/Gather entries needed for the fragment construction.
  - For a single buffer input frame:  $n = 2$  (FOF=32 bytes).
  - For a Scatter/Gather input frame:  
 $n = \text{round-up}(\text{MTU} / \text{min input buffer size in the Scatter/Gather entries}) + 1$ . For typical operation setting  $n=4$  (FOF=64 bytes) is sufficient.
4. The FOF value must be set such that:
  - For IPv4:  $(256 - \text{FOF}) \geq \text{L2 header size} + \text{IP header size} + \text{size of all IP options that are to be copied to each fragment}$ .
  - For IPv6:  $(256 - \text{FOF}) \geq \text{L2 header size} + \text{Unfragmentable part} + \text{Fragment header size}$ .
5. The frame buffer sizes must be:
  - For a single-buffer input frame, the buffer size must not exceed  $2^{13}$ .
  - For a Scatter/Gather input frame, the buffer size of each Scatter/Gather buffer must not exceed  $2^{13}$ .
6. For an input Scatter/Gather frame:
  - For IPv4, the first Scatter/Gather entry's buffer size must accommodate the L2 header size + IP header size + size of all IP options that are to be copied to each fragment.  
In addition, the input frame's S/G buffer size must accommodate the FD[offset] + 256 bytes.
  - For IPv6, the first Scatter/Gather entry's buffer size must accommodate the L2 header size + Unfragmentable part + Fragment header size.  
In addition, the input frame's S/G buffer size must accommodate the FD[offset] + 256 bytes.
7. During fragmentation, a new buffer is allocated in external memory for each fragment. This buffer is allocated either from the BPID of the frame being fragmented or from the user-configured BufferPoolID in the IP Fragmentation Table Descriptor. The buffer size must accommodate FD[offset] + 256 bytes.
8. For Frame Manger versions that are earlier than v3, the IP Fragmentation Host Command must be issued as described in [Section , "IP Fragmentation Host Command Data Structure."](#)

- Frames sent to the IP Fragmentation function must not contain an Ethernet FCS.

## De-Allocation of IP Fragments

Since the structure of IP Fragments includes multiple pointers to the same buffer, their de-allocation must be done according to the following rules:

- The pointers marked as belonging to “BPID NULL” must be de-allocated first.
- The pointer marked as belonging to a “Real BPID” must be de-allocated last.
- The de-allocation process results in multiple de-allocations of the same address; one of which is the de-allocation to the “real BPID”.

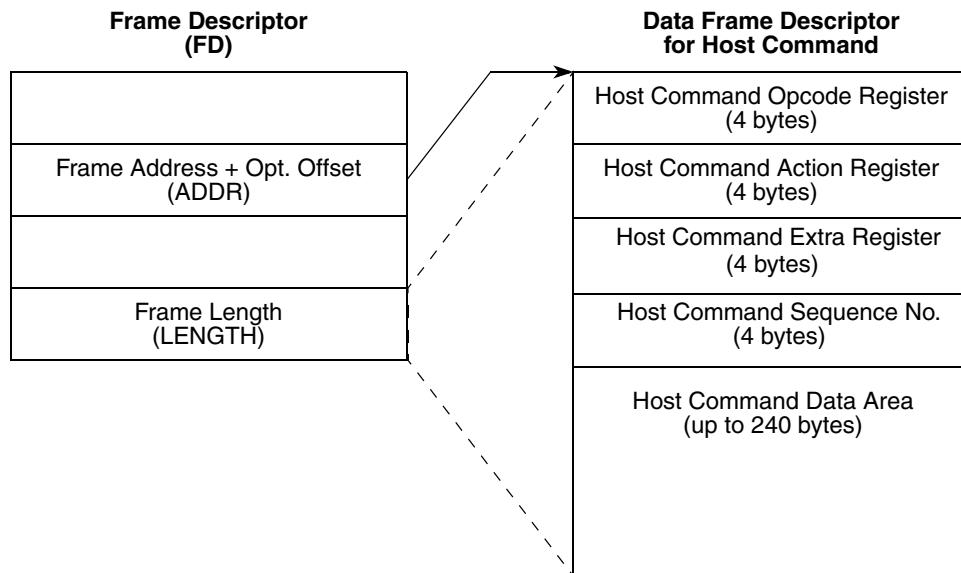
### 5.12.17.1.2 IP Fragmentation Data Structures

The IP Fragmentation Table Descriptor is described in [Section 5.12.17.4.2, "IP Fragmentation Table Descriptor."](#)

#### IP Fragmentation Host Command Data Structure

The IP Fragmentation Host Command is used to either fill up a Scratch Buffer Pool or to empty out a Scratch Buffer Pool. During initialization, the Scratch Buffer Pool is filled up with as many buffers as requested by the caller. During cleanup, the Scratch Buffer Pool is emptied of all its buffers. Cleanup should be performed only after all fragments created by the IP Fragmentation process have released all their buffers. The number of buffers that were cleaned up is returned to the caller.

The following figure describes the parameters that should be programmed in the Data Frame Descriptor for the IP Fragmentation host command.



**Figure 5-478. Data Frame Descriptor For IP Fragmentation Host Command**

- The Host Command Opcode Register (HCOR), Opcode Field should be set to 0x11. For detailed information about the Host Command Opcode Register (HCOR) See [Section 5.12.16.4.1, "Host Command Opcode Register \(HCOR\)."](#)
- The Host Command Action Register (HCAR) contains the sub-command (Fill or Empty) and the BPID.
- The Host Command Extra Register (HCER) contains the number of buffers as described in the table below.
- The Host Command Sequence No. Register is used as described in details in [Section 5.12.16.4.4, "Host Command Sequence Number \(HCSN\)."](#)
- The Host Command Data Area is not used.

The IP Fragmentation Host Command Action register is described below.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	SUBCMD										BPID					
2	—															

**Figure 5-479. IP Fragmentation Host Command Action Register**

**Table 5-522. IP Fragmentation Host Command Action Register**

Offset	Bits	Name	Description
0	0-7	SUBCMD	Sub Command 0 - Fill. Pool is filled with buffers. 1 - Empty. Pool is emptied of all its buffers.
	8-15	BPID	Buffer Pool ID. This is the ID of the Scratch Pool. This pool must be allocated prior to calling the host command.
2	0-15	Reserved.	Reserved. Must be cleared.

The IP Fragmentation Host Command Extra register is described below

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NumBuffers															
2	—															

**Figure 5-480. IP Fragmentation Host Command Extra Register**

**Table 5-523. IP Fragmentation Host Command Extra Register**

Offset	Bits	Name	Description
0	0-31	NumBuffers	Number of Buffers. For SUBCMD=0, this is the number of buffers to fill the scratch pool with. The host command returns the number of buffers that could not be filled. A zero value indicates success while a non-zero value indicates failure. The failure value is an approximation of the number of buffers that could not be filled. For SUBCMD=1, this is the number of buffers that were emptied from the scratch pool.

## 5.12.17.2 IP Reassembly

### 5.12.17.2.1 Overview

The FMan Controller performs reassembly on IP protocol fragments. The IP Reassembly function provides the following high-level features:

- Reassembly of IP fragments into IP packets according to RFC 791 and RFC 2460.
- Support the following interfaces:
  - 10 Gigabit Ethernet
  - Gigabit Ethernet
  - Offline Port
- Protocol Stack supported:
  - Ethernet
  - IP
- IP fragments may be received in order or out-of-order.
- Maintains order between reassembled frames and regular frames received on the same flow.
- User configurable reassembly timeout.
- Minimum size of First/Middle fragment is limited according to user configuration. This is not valid for CAPWAP reassembly.
- Statistics counters.
- Error handling for malformed fragments.
- Support for up to 2K simultaneous frames from the same flow or from different flows to be reassembled concurrently.
- Reassembly of a fragments does not lose indications of congestion.

IP Reassembly supports a maximum of 16 fragments per frame. Each fragment must reside in a single buffer (not in a Scatter/Gather frame).

### 5.12.17.2.2 Terminology

- **Open reassembly frame:** A frame that started the reassembly process; i.e. at least one fragment of this frame was handled but not all the fragments.

- **Opening fragment:** The first handled fragment of a frame (not necessary the fragment with frag offset=0)
- **Closing fragment:** The last handled fragment of a frame. This fragment completes the frame. No more fragments are expected for the reassembly of the frame.
- **RFD:** Reassembled Frame Descriptor.
- **IP Packet Length:** for IPv4, this is the IP Total Length. For IPv6, this is the IP Payload Length + 40.

### 5.12.17.2.3 IP Reassembly Functional Description

#### IP Reassembly Process

The IP Reassembly function can be applied to IP fragments residing on an RX port or on an Offline port. When an IP frame is recognized as an IP fragment (by the Parser), it is later on in the flow steered to a Keygen scheme that is programmed to generate an IP Reassembly key. IPv4 fragments are steered to a different Keygen scheme from IPv6 fragments. An example IPv4 IP Reassembly key may contain the SourceAddress, DestinationAddress, Protocol and ID. An example IPv6 Reassembly key may contain the SourceAddress, DestinationAddress and ID. All IP fragments with an identical IP Reassembly key value are considered to be part of the same IP frame.

After an IP Reassembly key is generated, processing is directed to the IP Reassembly Table Descriptor which triggers the IP Reassembly function. The IP Reassembly function searches for the key in its IP Reassembly Automatic Learning Hash table. The search consists of a hash lookup followed by an exact match. The hash lookup uses a programmable number of LSBs from the key's hash value as an index into the hash table to retrieve one set. For FMan\_v3 devices some bits are also used for SPID selection (see later in this section). The set is then searched for an exact match with the full key value. If the exact match fails, then this is the opening fragment that arrived for this frame. Therefore, a new entry is created in the set, a new Reassembly Frame Descriptor (RFD) is associated with this entry and a new Scatter/Gather list is built. If the exact match succeeds, then this fragment is added to the already existing Scatter/Gather list that contains all the fragments of this frame that have been accumulated so far. Once all fragments of a given frame have arrived, the reassembly is completed and the reassembled Scatter/Gather frame is enqueued to the user-configured queue. The L2 header and IP header of the reassembled frame are those belonging to the first fragment with all the necessary IP header updates that must be performed as total length, IPv4 checksum, IPv6 fragment extension removal etc. See [Figure 5-483](#).

External buffers are allocated during the reassembly process. Each reassembled frame requires one additional external buffer allocation, the Fman Controller writes to this buffer the scatter gather list for the reassembled frame. The Buffer Pool ID field which is used should be at least (256+input fragment FD[offset] ignoring the 4 lsb of this offset) bytes. The FD[offset] (ignoring the 4 lsb of this offset) of the S/G list and ICID fields are taken from the input fragment. User should use the same fragment offset for all fragments that belongs to the same reassembled frame on the OP. User should take care that the Scatter/Gather list will not overwrite the internal context by configuring the FMBM\_xICP parameters accordingly.

For FMan\_v3 devices the storage profile selection is according to the storage profile of the opening fragment. In case that the first fragment (fragment with offset zero) of the frame corresponds to different storage profile than the opening fragment then one of two possible scenarios occurs:

1. Mark “Non Consistent SP” bit in the FD[Status].
2. Reassembled frame is enqueued into a special “Non Consistent Storage Profile FQID” without further PCD flow.

Reassembled frames are marked by setting “IP Reassembled” bit in the FD[Status].

## IP Reassembly Frame Ordering

IP Reassembly maintains the order between reassembled frames and regular frames residing in the same queue. The ordering is determined by the closing fragment of the reassembled frame. Regular frames residing in the same queue before a closing fragment are forwarded before the corresponding reassembled frame. For example, if the case is F,M1,M2,L,R,M3 (where M3 is the closing fragment), then the regular frame is enqueued before the reassembled frame (R, F,M1,M2,M3,L). Frames residing in the same queue after the closing fragment are forwarded after the corresponding reassembled frame. For example, if the case is F,M1,M2,L,M3,R (where M3 is the fragment that completes the frame), then the reassembled frame is enqueued before the regular frame (F,M1,M2,M3,L,R).

IP Reassembly maintains order between received reassembled frames residing in the same queue.

When a fragment that completes frame1 is located before a fragment that completes frame2 then frame1 is enqueued before frame2. For example, if the case is L1,F2,F1,L2 (where F1 is the fragment that completes frame1 and L2 is the fragment that completes frame2), then frame1 is enqueued before frame2.

## IP Reassembly Congestion Indication

Reassembly of a fragments does not lose indications of congestion. This means that if any fragment of an IP packet to be reassembled has the CE codepoint set, then the FMan Controller sets the CE codepoint on the reassembled packet. However, this will not occur if any of the other fragments contributing to this reassembly carries the Not-ECT codepoint. In this case the reassembled frame will be discarded.

## IP Reassembly Error Handling

The IP Reassembly function checks for error fragments. The following types of fragments are considered to be error fragments:

- Duplicate fragment.
- Overlap fragment.
- Short fragment (First or Middle Fragment with size < configured minimum fragment size). This check is not valid for CAPWAP reassembly.
- Check if successfully CAPWAP reassembled frame exceeds maximum frame size.
- Out of range fragment (Fragment offset is beyond the end of the Last fragment).
- First or Middle fragment whose IP payload size is not a multiple of 8.

- Middle or Last fragment in which the fragment offset \* 8 + fragment IP payload length + IP header length exceeds 64Kbytes. This check is not valid for CAPWAP.
- Fragment that exceeds 16 fragments for the frame.
- Fragment which resides in a Scatter/Gather format.
- Fragment which carries Not-ECT codepoint and any other fragment of this IP packet to be reassembled has the CE codepoint set.
- Fragment which carries CE codepoint set and any other fragment of this IP packet to be reassembled has the Not-ECT codepoint.

If an error fragment is encountered then it is either discarded or enqueued to an error queue as per user configuration, the related counters are updated and the reassembly function continues. If a “good” fragment arrives at a later time, then reassembly may succeed. If not, then the frame will eventually expire as described in [Section ,“IP Reassembly TimeOut.”](#)

The IP Reassembly function may fail due to lack of resources:

- No available entries in the IP Reassembly Automatic Learning Hash table
- No available RFDs
- No available internal/external buffers for building the Scatter/Gather list

In such cases, the fragment being handled is either discarded or enqueued to an error queue as per user configuration and the corresponding busy counters are updated.

All error fragments and TimeOut frames are marked as errors according to the FD[status][IPRE] error code field - see the FD Status/Command description for the bits locations.

## **IP Reassembly TimeOut**

The TimeOut function limits the time of reassembly. An IP reassembly TimeOut table is used for recording the timestamp of each fragment. The update of this table depends on the configuration of the TimeOutMode bit initialized in the IP Reassembly Common Parameters Table.

If the TimeOutMode bit is zero, the update is as follows:

Upon processing of any fragment of a frame, the current time is stored as a timestamp in the IP Reassembly TimeOut table. This table lists the timestamp of the latest processed fragment for each frame. In this case, the TimeOut limits the time between consecutive fragments processing.

If the timeOutMode bit is set, the update is as follows:

Upon processing of the opening fragment of a frame, the current time is stored as a timestamp in the IP Reassembly TimeOut table. This table lists the timestamp of the first processed fragment for each frame. In this case, the TimeOut limits the time for the all reassembled frame.

The TimeOut process runs in the background and checks for expired frames/fragments. If an open reassembly frame (at least one fragment was processed but not all the fragments) is found to be too old compared to a configured expiration delay, the frame is enqueued to a TimeOut Queue configured in the

IP Reassembly Common Parameters Table. If the user would like to discard timed-out frames, as recommended, then the TimeOutQueue should be configured to 0.

The time period at which the timeout check is triggered is configurable. For more information see parameter TSBS in [Table 5-527. “IP Reassembly Timeout Configuration Host Command HCAR and HCER.”](#)

If a frame is expired, its reassembly process is halted and the ‘total TimeOut counter’ is incremented.

If the reassembly is halted, the partially reassembled frame can be either discarded (recommended mode) or can be enqueued to the user-configured TimeOut Queue. This queue is configured in the IP Reassembly Common Parameters table. Note that the partially reassembled frame may not yet be reassembled in-order, i.e the fragments may appear out-of-order in the scatter/gather list. They appear in the order they were processed. A description of a partially reassembled frames can be found at [Partially Reassembled Frame Descriptor](#).

The IP Reassembly TimeOut Configuration Host Command is used to set up a TimeOut background routine.

For detailed information about the Host Command procedure and the Host Command flow, see [Section 5.12.16, "Host Commands."](#)

For a description of the IP Reassembly TimeOut Configuration Host Command data structures, See [Section , "IP Reassembly Internal Buffer Pool Management Structure."](#)

## IP Reassembly Statistics

The counters maintained by the IP Reassembly function are described in [Table 5-524. “IP Reassembly Parameters Table”](#) and in [Table 5-525. “IP Reassembly Common Parameters Table.”](#) Counters maintained in the IP Reassembly Parameters Table are maintained per IP Version. i.e. IPv4 counters are maintained separately from IPv6 counters. Counters maintained in the IP Reassembly Common Parameters Table are an accumulation of both IPv4 and IPv6 statistics.

## Initialization of IP Reassembly

IP Reassembly is initialized and configured by the Host CPU. The required initialization steps are as follows:

- The Parser must be configured to identify IPv4 fragments and IPv6 fragments and to steer each one to its corresponding Keygen scheme using the Soft Parser.
- Keygen must have a scheme for IPv4 fragments and a scheme for IPv6 fragments. Each scheme must generate an IP Reassembly key and a hash value and trigger the appropriate IP Reassembly Table Descriptor (either IPv4 or I Pv6 IP Reassembly Table Descriptor). See key size restrictions related to the SetSize parameter in [Table 5-524. “IP Reassembly Parameters Table.”](#)
- All IP Reassembly data structures must be allocated and initialized as described in [Section 5.12.17.2.4, "IP Reassembly Data Structures."](#)
- The IP Reassembly TimeOut Configuration Host Command with sub command ‘Activate Timeout Task’ must be issued as described in [Section , "IP Reassembly Internal Buffer Pool Management Structure."](#)

- The BMI internal buffer allocation must not begin from offset 0 in the FMAN internal memory. i.e. FMBM\_CFG1[FBPO] must be greater than 0.
- In order to assure as much as possible that each fragment resides in a single buffer (not in a Scatter/Gather format) user need to do the following:
  - Configure an internal BMI memory size that is large enough to hold the largest expected IP Fragment multiplied by number of TNUMs allocated for this port, in addition to one extra internal buffer per TNUM.
  - Configure an external buffer pool with buffer sizes that are large enough to hold the largest expected IP Fragment.
- For FMan\_v3 devices user need to allocate 256 bytes “page” for the IP reassembly port by configure its pointer in FMBM\_xGPR. For more information refer to [Section 5.12.23,"FMan-Controller Parameters Page per Port."](#)

#### 5.12.17.2.4 IP Reassembly Data Structures

The following figures provides an overview of the IP Reassembly data structures and an example of the data structure of a reassembled frame. Subsequent sub-sections provide details on each data structure.

The figure below illustrates the relationship between the different IP Reassembly data structures. The figure is an example of a system that performs IP Reassembly on both IPv4 and IPv6 fragments. If IP Reassembly is to be performed on more than one port, then all the data structures in this figure, aside for the TimeOut hardware Trigger Mechanism, are duplicated for each port. The TimeOut hardware Trigger Mechanism is always shared by all IP Reassembly functions.

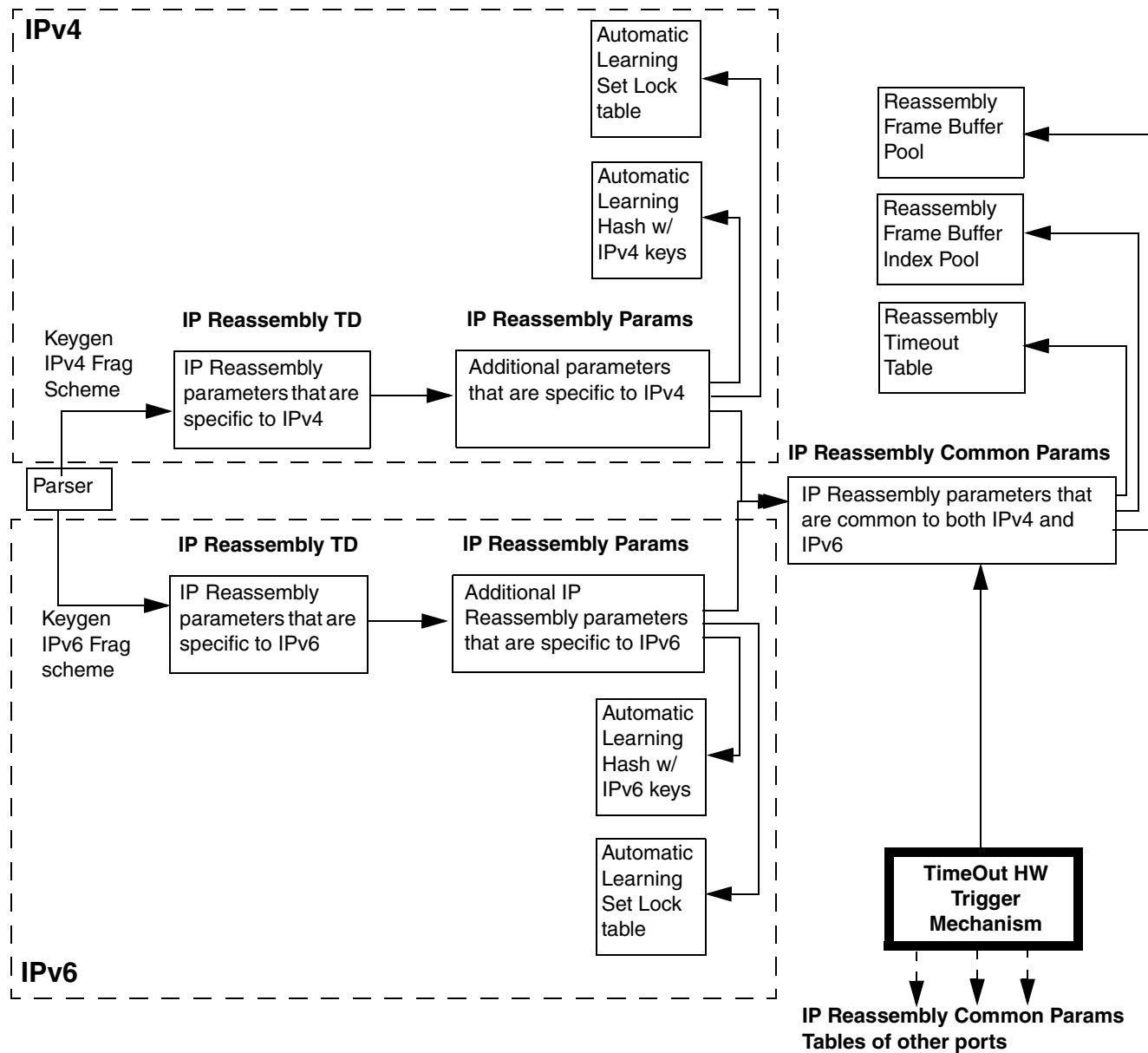


Figure 5-481. Example Per-Port IP Reassembly Data Structures

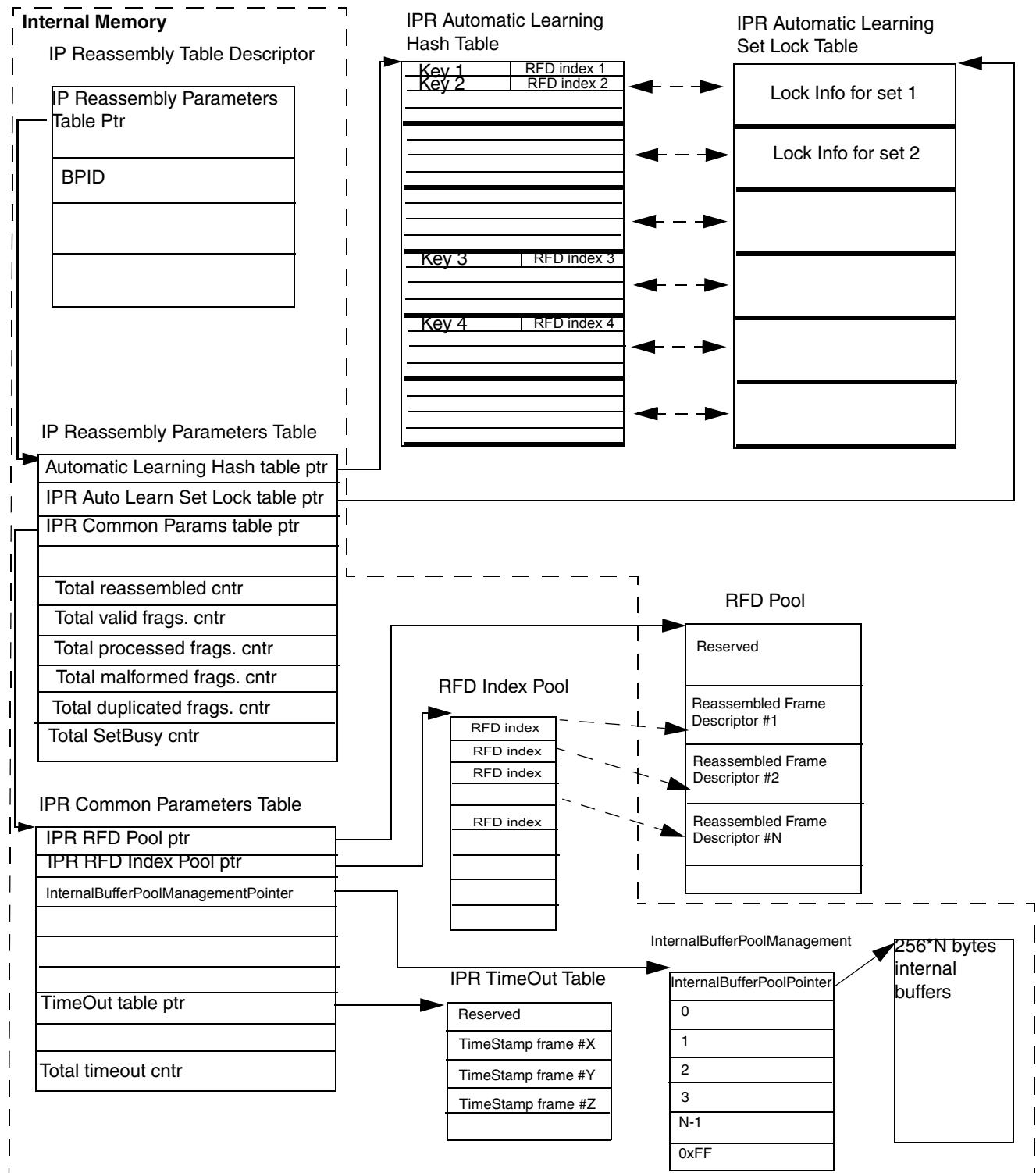
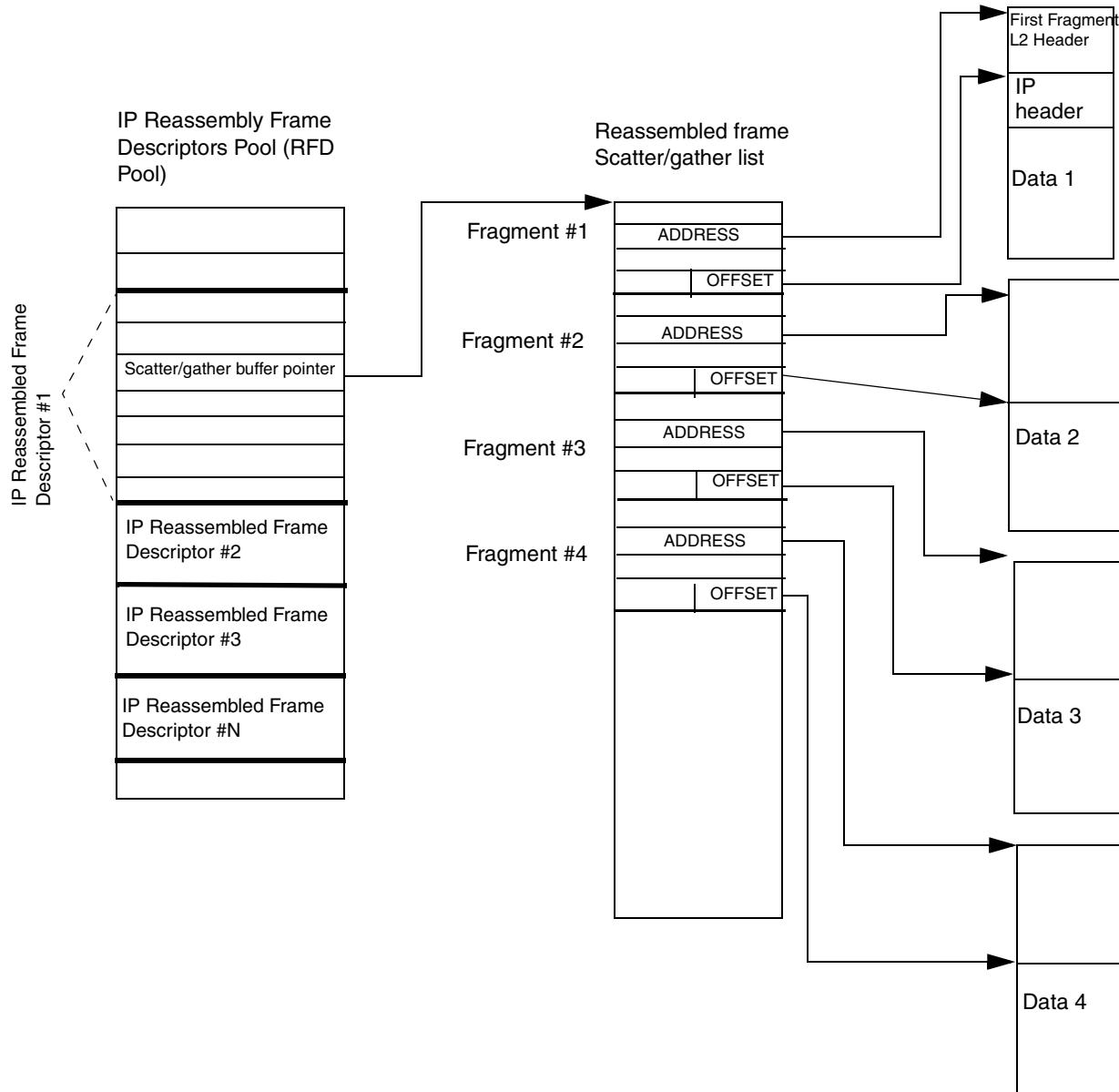


Figure 5-482. IP Reassembly Data Structures Overview



**Figure 5-483. Example of a Reassembled IP Frame**

## IP Reassembly Parameters Table

The IP Reassembly Parameters table contains parameters that are specific to either the IPv4 reassembly function or to the IPv6 reassembly function. If both IPv4 reassembly and IPv6 reassembly is required, then two separate IP Reassembly Parameter tables are required. This table is located in the internal memory. The pointer to this table is configured in the IP Reassembly Table descriptor, see [Section 5.12.17.4.3,"IP Reassembly Table Descriptor."](#)

The IP Reassembly Parameters Table base address should be 8-byte aligned.

[Figure 5-484 ,‘IP Reassembly Parameters Table’](#) describes the IP Reassembly Parameters Table:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	—	—	—	—	WaysNumber	—	—	—	—	—	—	—	SetSize	—	—	—
0x2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x4	—	—	—	—	—	—	—	—	—	—	—	—	IPReassCommonParamsTblPointer[0-7]	—	—	—
0x6	—	—	—	—	—	—	—	—	—	—	—	—	IPReassCommonParamsTblPointer[8-23]	—	—	—
0x8	—	—	—	ICID_AL	—	—	—	—	—	—	—	—	—	—	—	—
0xA	—	—	EICID_AL	—	—	—	—	—	—	—	—	—	AutoLearnHashTblPointer	—	—	—
0xC	—	—	—	—	—	—	—	—	—	—	—	—	AutoLearnHashTblPointer(cont)	—	—	—
0xE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x10	—	—	—	iICID_SL	—	—	—	—	—	—	—	—	—	—	—	—
0x12	—	—	EICID_SL	—	—	—	—	—	—	—	—	—	AutoLearnHashTblPointer	—	—	—
0x14	—	—	—	—	—	—	—	—	—	—	—	—	AutoLearnSetLockTblPointer(cont)	—	—	—
0x16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x18	—	—	—	—	MinFragSize	—	—	—	—	—	—	—	—	—	—	—
0x1A	—	—	—	—	MaxReassembledFrameLength	—	—	—	—	—	—	—	—	—	—	—
0x1C	—	—	—	—	—	—	—	—	—	—	—	—	TotalSuccessfullyReassembledFramesCounter	—	—	—
0x1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x20	—	—	—	—	—	—	—	—	—	—	—	—	TotalValidFragmentsCounter	—	—	—
0x22	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x24	—	—	—	—	—	—	—	—	—	—	—	—	TotalProcessedFragmentsCounter	—	—	—
0x26	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x28	—	—	—	—	—	—	—	—	—	—	—	—	TotalMalformedFragmentsCounter	—	—	—
0x2A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x2C	—	—	—	—	—	—	—	—	—	—	—	—	TotalSetBusyCounter	—	—	—
0x2E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x30	—	—	—	—	—	—	—	—	—	—	—	—	TotalDiscardedFragsCounter	—	—	—
0x32	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x34	—	—	—	—	—	—	—	—	—	—	—	—	TotalMoreThan16FragsCounter	—	—	—
0x36	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x38	—	—	—	—	—	—	—	—	—	—	—	—	ExceedMaxReassembledFrameLength	—	—	—
0x3A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x3C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x3E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

**Figure 5-484. IP Reassembly Parameters Table**

The following table describes the fields of the IP Reassembly Parameters Table:

**Table 5-524. IP Reassembly Parameters Table**

Offset	Bits	Name	Description
0x0	0-3	Reserved	Reserved. Must be set to zero
	4-7	WaysNumber	Number of Ways-per-set in the IP Automatic learning Hash table. Any number between 1 and 8 is valid.
	8-11	Reserved	Reserved. Must be set to zero
	12-15	SetSize	<p>Size of a set in the Automatic Learning Hash table.</p> <p>4 - size of set is 16 bytes      5 - size of set is 32 bytes      6 - size of set is 64 bytes      7 - size of set is 128 bytes      8 - size of set is 256 bytes      All other values are reserved.</p> <p>The size of a set is determined by the extracted key size and the number of ways-per-set as follows:  <math>\text{WaySize} = (\text{extracted key size} + 4)</math> for FMan_v3 devices rounded-up to the next multiple of 8.  <math>\text{Size of set} = (\text{WaySize} * \text{WaysNumber})</math> rounded-up to the next power of 2.</p> <p>For example, if the extracted key is an IPv6 IP Reassembly key (source address, destination address, ID) and the WaysNumber=4, then      For devices later than and including FMan_v3:  <ul style="list-style-type: none"> <li>• WaySize = <math>(16+16+4 + 4)</math> rounded-up to next multiple of 8 = 40</li> <li>• Size of set = <math>(40 * 4)</math> rounded-up to next power of 2 = 256</li> </ul>     For a 256 byte set, the SetSize field should be set to 8.</p> <p>For devices later than and including FMan_v3:      If the extracted key size is a multiple of 8 or 1 to 3 byte less than the next multiple of 8, then keygen should append 4 bytes of zero padding to the extracted key. Otherwise, Keygen should not add any padding. For example:       <ul style="list-style-type: none"> <li>• If the extracted key size is 16 bytes, then 4 bytes of zeros should be appended to the key, resulting in a key size of 20 bytes and a way size of 24.</li> <li>• If the extracted key size is 12 bytes, then no padding should be added and the way size is 16 bytes.</li> <li>• If the extracted key size is 13 bytes, then 4 bytes of zeros should be appended to the key, resulting in a key size of 17 bytes and a way size of 24.</li> </ul>     Note that the maximum size of the extracted key is 52 bytes.</p>
0x2	0-15	AutoLearnHashKeyMask	<p>Automatic Learning Hash Key Mask.</p> <p>This mask is applied to the 16LSB of the IC[hash value]. The result is used as an index into the Automatic Learning Hash table. Thus this field determines how many sets are in the Automatic Learning Hash table.</p>
0x4	0-7	Reserved	Reserved. Must be set to zero
	8-23	IPReassCommonParamsTbl Pointer	Pointer to the IP Reassembly Common Parameters table. This table is located in the internal memory. Its size is 64 bytes and its base address should be 8-byte aligned. See <a href="#">Table 5-525. "IP Reassembly Common Parameters Table"</a>

**Table 5-524. IP Reassembly Parameters Table (continued)**

Offset	Bits	Name	Description
0x8	0-1	Reserved	Reserved. Must be set to zero.
	2-7	ICID_AL	Isolation context identifier (ICID). This ICID is used by the IP Reassembly process to access the table pointed at by the AutoLearnHashTblPointer.
	8-15	Reserved	Reserved. Must be set to zero.
0xA	0-1	Reserved	Reserved. Must be set to zero.
	2-3	EICID_AL	Extended isolation context identifier (ICID). This field contains the 2 LSBs of the ICID_AL field.
	4-7	Reserved	Reserved. Must be set to zero.
0xB	5 bytes	AutoLearnHashTblPointer	<p>Pointer to the IP Reassembly Automatic Learning Hash Table. The size of this table is determined by the number of sets and the set size.</p> <p>For example, if AutoLearnHashKeyMask=0xF and SetSize=8, then: Total table size = 16 * 256 bytes.</p> <p>It is recommended that the total number of entries in this table (number of sets * number of ways) will be twice the number of frames that are expected to be reassembled simultaneously.</p> <p>This table resides in external memory. The table entries must be initialized to zero. This table's base address should be aligned to the size of a set, as programmed in SetSize (For example, if SetSize = 4, then it should be aligned to 16). See <a href="#">Section , "IP Reassembly Frame Descriptor"</a> for details.</p>
0x10	0-1	Reserved	Reserved. Must be set to zero.
	2-7	ICID_SL	Isolation context identifier (ICID). This ICID is used by the IP Reassembly process to access the table pointed at by the AutoLearnSetLockTblPointer.
	8-15	Reserved	Reserved. Must be set to zero.
0x12	0-1	Reserved	Reserved. Must be set to zero.
	2-3	EICID_SL	Extended isolation context identifier This field contains the 2 LSBs of the ICID_SL field.
	4-7	Reserved	Reserved. Must be set to zero.
0x13	5 bytes	AutoLearnSetLockTblPointer	<p>Pointer to the Set Lock table.</p> <p>The size of this table is (number of sets in the IP Reassembly Automatic Learning Hash table)*4 bytes. This table resides in external memory and its base address should be 4-byte aligned. The table entries must be set to zero.</p>

**Table 5-524. IP Reassembly Parameters Table (continued)**

Offset	Bits	Name	Description
0x18	0-15	MinFragSize	<p>Not Valid for Next Generation CAPWAP reassembly. First/Middle fragment minimum size in Bytes.</p> <p>This is the minimum size allowed for a first/middle fragment. First/Middle fragments whose IP packet length is shorter than MinFragSize are considered to be a malformed fragments by the IP Reassembly function.</p> <p>For IPv6 MinFragSize must be &gt;= 256.</p>
	16-31	MaxReassembledFrameLength	<p>Valid only for Next Generation CAPWAP reassembly. For IP Reassembly must be set to 0. The maximum CAPWAP reassembled frame length in bytes.</p> <p>if MaxReassembledFrameLength == 0, any successful reassembled frame length is considered as a valid length.</p> <p>if MaxReassembledFrameLength &gt; 0, a successful reassembled frame which its length exceeds this value is considered as an error frame (FD status[CRE] bit is set).</p>
0x1C	0-31	TotalSuccessfullyReassembledFramesCounter	<p>Counts the number of successfully reassembled frames.</p> <p>Must be initialized to zero.</p>
0x20	0-31	TotalValidFragmentsCounter	<p>Counts the total number of valid fragments that have been processed for all frames.</p> <p>Must be initialized to zero.</p>
0x24	0-31	TotalProcessedFragmentsCounter	<p>Counts the number of processed fragments (valid and error fragments) for all frames.</p> <p>Must be initialized to zero.</p>
0x28	0-31	TotalMalformedFragmentsCounter	<p>Counts the number of malformed fragments processed for all frames.</p> <p>Malformed fragments include:</p> <ul style="list-style-type: none"> <li>• Duplicate fragments</li> <li>• Overlap fragments</li> <li>• Short fragments (First or Middle fragment with size &lt; configured minimum fragment size)</li> <li>• Out of range fragments (Fragment offset is beyond the end of the Last fragment)</li> <li>• First or Middle fragments whose IP payload size is not a multiple of 8</li> <li>• Middle or Last fragments in which the fragment offset * 8 + fragment IP payload length + IP header length exceeds 64Kbytes</li> <li>• Fragment which carries Not-ECT codepoint and any other fragment of this IP packet to be reassembled has the CE codepoint set.</li> <li>• Fragment which carries CE codepoint set and any other fragment of this IP packet to be reassembled has the Not-ECT codepoint.</li> </ul> <p>Must be initialized to zero.</p>
0x2C	0-31	TotalSetBusyCounter	<p>Counts the number of times a busy condition occurs when attempting to access an IP Reassembly Automatic Learning Hash set.</p> <p>Must be initialized to zero.</p>
0x30	0-31	TotalDiscardedFragsCounter	<p>Counts the number of fragments discarded by the reassembly process. Once discarded by the reassembly process, these fragments may either be enqueued to an error queue or discarded by the BMI, as per user configuration of the BMI. This counter does not include fragments that were discarded due to TimeOut. Must be initialized to zero.</p>
0x34	0-31	TotalMoreThan16FragsCounter	<p>Counts the fragment occurrences in which the number of fragments-per-frame exceeds 16.</p> <p>Must be initialized to zero.</p>

**Table 5-524. IP Reassembly Parameters Table (continued)**

Offset	Bits	Name	Description
0x38		ExceedMaxReassembledFrameLength	Valid only for Next Generation CAPWAP Reassembly. Counts the number of times that a successful reassembled frame length exceeds MaxReassembledFrameLength value. In case MaxReassembledFrameLength == 0, this counter never incremented. Must be initialized to zero.
0x3C		Reserved	Reserved. Must be set to zero.

### IP Reassembly Common Parameters Table

The IP Reassembly Common Parameters table contains parameters that are common to both the IPv4 reassembly function and IPv6 reassembly function. The pointer to the IP Reassembly Common Parameters table is configured in the IP Reassembly Parameters table. See the IPReassCommonParamsTblPointer field in [Table 5-524](#) for size and alignment requirements.

The figure below describes the IP Reassembly Common Parameters Table:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	TimeOutMode	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TimeOutFQID[0-7]
0x2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TimeOutFQID[8-23]
0x4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	ReassFrmDescIndexPoolPointer[0-7]
0x6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	ReassFrmDescIndexPoolPointer[8-23]
0x8	—	—	ICID_RFD	—	—	—	—	—	—	—	—	—	—	—	—	—
0xA	—	EICID_RFD	—	—	—	—	—	—	—	—	—	—	—	—	—	ReassFrmDescPoolPointer
0xC	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	ReassFrmDescPoolPointer(cont)
0xE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	ExpirationDelay
0x10	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TimeOutTblPointer[0-7]
0x12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TimeOutTblPointer[8-23]
0x14	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	ExpirationDelay
0x16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	ExpirationDelay
0x18	InternalBufferPoolManagementIndex	—	—	—	—	—	—	—	—	—	—	—	—	—	—	InternalBufferPoolManagementPointer(msb)
0x1A	—	InternalBufferPoolManagementPointer (lsb)	—	—	—	—	—	—	—	—	—	—	—	—	—	InternalBufferPoolManagementPointer (lsb)
0x1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x20	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TotalTimeOutCounter
0x22	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x24	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TotalRFDPoolBusyCounter
0x26	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x28	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TotalInternalBufferBusy
0x2A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x2C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TotalExternalBufferBusy
0x2E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x30	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TotalSgFragmentCounter
0x32	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x34	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TotalDmaSemaphoreDepletionCounter
0x36	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
0x38	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	TotalNCSPCounter
0x3C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	DiscardMask

**Figure 5-485. IP Reassembly Common Parameters Table**

The table below describes the fields of the IP Reassembly Common Parameters Table:

**Table 5-525. IP Reassembly Common Parameters Table**

Offset	Bits	Name	Description
0x0	0	TimeOutMode	TimeOut Mode. 0-The Expiration Delay defines the maximum delay between the current time and the last handled fragment. 1-The Expiration Delay defines the maximum delay between the current time and the first handled fragment of the frame.
	1-7	Reserved	Reserved. Must be set to zero.
	8-31	TimeOutFQID	TimeOut FQID. Recommended value for this field is 0; in this way timed-out frames are discarded on FMan level. Time-out frames are enqueued to this FQID if this field is not 0.
0x4	0-7	Reserved	Reserved. Must be set to zero.
	8-31	ReassFrmDescIndexPoolPointer	Pointer to the IP Reassembly Frame Descriptor Indexes Pool. This pool resides in the internal memory. The number of entries in this pool is identical to the number of entries in the ReassFrmDescPool. The size of each entry is 2 bytes. This table's base address should be 256-byte aligned. The entries in this table should contain indexes starting with 1 up to the maximum number of frames that are allowed to be reassembled simultaneously + 128. The last entry in this pool must contain the index zero. See <a href="#">IP Reassembly Frame Descriptor</a> for details.
0x8	0-1	Reserved	Reserved. Must be set to zero.
	2-7	ICID_RFD	Isolation context identifier (ICID) This ICID is used by the IP Reassembly process to access the table pointed at by the ReassFrmDescPoolPointer.
	8-15	Reserved	Reserved. Must be set to zero.
0xA	0-1	Reserved	Reserved. Must be set to zero.
	2-3	EICID_RFD	Extended isolation context identifier (EICID) This field contains the 2 LSBs of the ICID_RFD field.
	4-7	Reserved	Reserved. Must be set to zero.
0xB	5 bytes	ReassFrmDescPoolPointer	Pointer to the IP Reassembly Frame Descriptors Pool. This pool resides in external memory. The number of entries in this pool should be equal to the maximum number of frames that are allowed to be reassembled simultaneously +129. The size of each entry is 64 bytes. This table's base address should be 64-byte aligned. See <a href="#">IP Reassembly Frame Descriptor</a> for details.
0x10	0-7	Reserved	Reserved. Must be set to zero.
	8-31	TimeOutTblPointer	Pointer to the TimeOut table. This table resides in the internal memory. The number of entries in this table is identical to the number of entries in the ReassFrmDescPool. Each entry is 8 bytes and should be set to zero. The table's base address should be 8-byte aligned. See <a href="#">IP Reassembly Frame Descriptor</a> .

**Table 5-525. IP Reassembly Common Parameters Table (continued)**

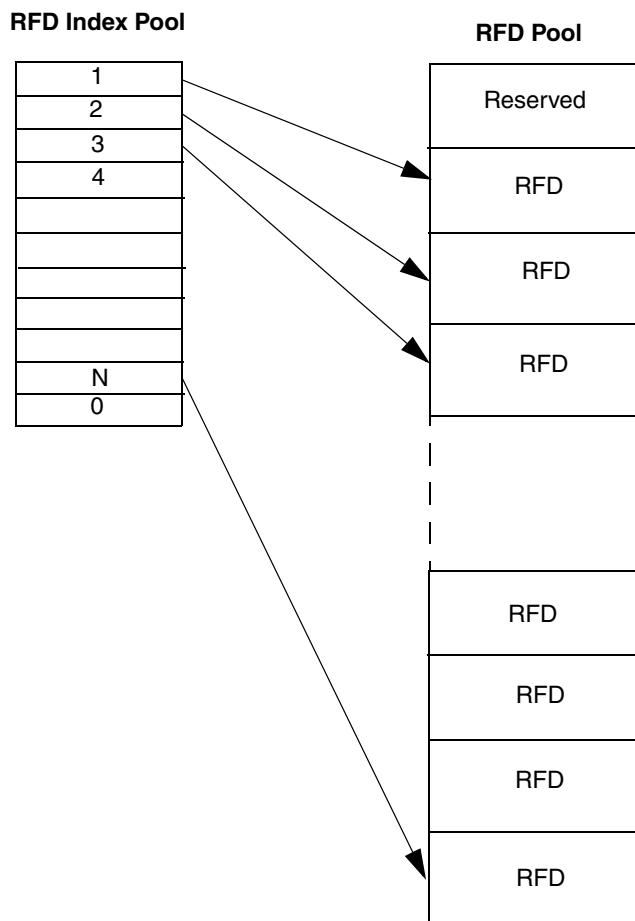
Offset	Bits	Name	Description
0x14	0-31	ExpirationDelay	Expiration Delay for IP Reassembly TimeOut. ExpirationDelay is expressed in the time units configured in the FMFP_TSC1 register. The timeout function invokes periodically a timeout task to check if timeout expired for any of the frames that are reassembled currently. Thus the accuracy of the expiration delay depends on how often this timeout task is invoked. The parameter TSBS of the IP reassembly configuration host command controls the frequency of timeout task invocation and thus controls the accuracy. See <a href="#">Table 5-527. “IP Reassembly Timeout Configuration Host Command HCAR and HCER.”</a>
0x18	0-7	InternalBufferPoolManagementIndex	Internal Buffer Pool Index. Should be initialized to 4. FMan Controller uses this field as an index to the next valid entry in the Internal Buffer Pool Management.
	8-31	InternalBufferPoolManagementPointer	Internal Buffer Pool Management Pointer. Should be 4 bytes aligned. User should allocate 5+N bytes for this structure. N indicates number of TNUMs belonging to the port (or ports) that uses this structure. For more information see <a href="#">Section ,“IP Reassembly Internal Buffer Pool Management Structure.”</a>
0x1C	0-15	Reserved	Reserved. Must be set to zero.
0x1E	16-31	Reserved	Reserved. Must be set to zero.
0x20	0-31	TotalTimeOutCounter	Counts the number of TimeOut occurrences. Must be initialized to zero.
0x24	0-31	TotalRFDPoolBusyCounter	Counts the number of failed attempts to allocate a Reassembly Frame Descriptor. Must be initialized to zero.
0x28	0-31	TotalInternalBufferBusy	Counts the number of times an internal buffer busy occurred. During the reassembly process, the allocation of an internal buffer from the Internal Buffer Pointers Pool is required. If there is no free buffer, this counter is incremented.
0x2C	0-31	TotalExternalBufferBusy	Counts the number of times external buffer busy occurred. For each reassembled frame, the allocation of an external buffer is required. If there is no free buffer, this counter is incremented.
0x30	0-31	TotalSgFragmentCounter	Counts the number of Scatter/Gather fragments. Any fragment that uses Scatter/Gather format is an error fragment.
0x34	0-31	TotalDmaSemaphoreDepletionCounter	Counts the number of failed attempts to allocate a DMA semaphore. When a DMA semaphore can not be allocated, the current fragment is discarded. If this counter is incremented the DMA Semaphore CAM number of entries should be incremented (FMDMMR[CEN] register) and/or the number of TNUMs allocated for this port should be decremented (FMBM_PP_X[MXT/EXT] register). Must be initialized to zero.
0x38	0-31	TotalNCSPCounter	Valid only for FMan_v3. Counts the number of Non Consistent Storage Profile events for successfully reassembled frames. Must be initialized to zero.

## IP Reassembly Frame Descriptor

The IP Reassembly Frame Descriptors (RFD) Pool contains RFDs that are used during reassembly to accumulate information about each reassembled frame. The IP Reassembly Frame Descriptors Index Pool

contains indexes into the RFD pool and manages allocation/de-allocation of these RFDs. Pointers to these two pools reside in the IP Reassembly Common Parameters Table. See the ReassFrmDescPoolPointer and ReassFrmDescIndexPoolPointer fields in the IP Reassembly Common Parameters Table described in [Table 5-525](#). Size and alignment requirements for these pools are also described in the IP Reassembly Common Parameters Table.

Each entry in the RFD Index pool contains an index to the IP Reassembly Frame Descriptors Pool starting with index 1. Index 0 represents an empty entry and should be placed in the last entry of the RFD Index Pool. The first entry of the RFD pool table is reserved. The RFD Pool and RFD Index Pool are illustrated in the figure below.



**Figure 5-486. RFD Index Pool and RFD Pool**

### Partially Reassembled Frame Descriptor

A frame that fails reassembly due to timeout may be forwarded to the CPU as a partially reassembled frame which is represented as a Scatter/Gather frame. The frame is marked as an error frame according to FD[status][IPRE] error code field. (See the FD Status/Command description for the bits locations). For a detailed description of a standard DPAA Scatter/Gather frame format, see

## IP Reassembly Internal Buffer Pool Management Structure

IP reassembly process requires internal buffer pool for its normal processing. User need to allocate these internal buffers by defining the Internal buffer Pool Management structure and allocate the internal buffers pool. Number of internal buffers should be equal to the number of TNUMs belonging to the port (or ports) that uses this structure.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
2																
4					0											1
6						2										3
8						4										5
3+N							N-1									0xFF

**Figure 5-487. Internal Buffer Pool Management Structure**

**Table 5-526. Internal Buffer Pool Management Structure**

Offset	Bits	Name	Description	
0x0	0-31	InternalBufferPool Pointer	Internal Buffer Pool Pointer. Should be aligned to 256 bytes. User allocates 256*N bytes for this Internal buffer Pool, N indicates number of TNUMs belonging to the port (or ports) that uses this structure.	
0x4	0-7	—	0.	
	8-15	—	1.	
	16-23	—	2.	
	24-31	—	3.	
3+N	0-7	—	N-1.	
4+N	0-7	—	0xFF. This value indicates the end of the internal buffer pool index.	

## IP Reassembly TimeOut Data Structure

Each entry of the IP Reassembly TimeOut table is 8 bytes long. The number of entries in the IP Reassembly TimeOut table should be equal to the number of entries in the IP Reassembly Frame Descriptors Pool. The IP reassembly TimeOut table is located in the internal memory. Since RFD index 0 is not allowed, the first 8 bytes of the IP Reassembly TimeOut table are reserved.

The figure below describes an entry in the IP Reassembly TimeOut table.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0								—								
2								—								
0																TimeStamp
2																

Figure 5-488. IP reassembly TimeOut Entry

### IP Reassembly TimeOut Configuration Host Command Data Structure

The following figure describes the parameters that should be programmed in the Data Frame Descriptor for the IP Reassembly TimeOut Configuration host command.

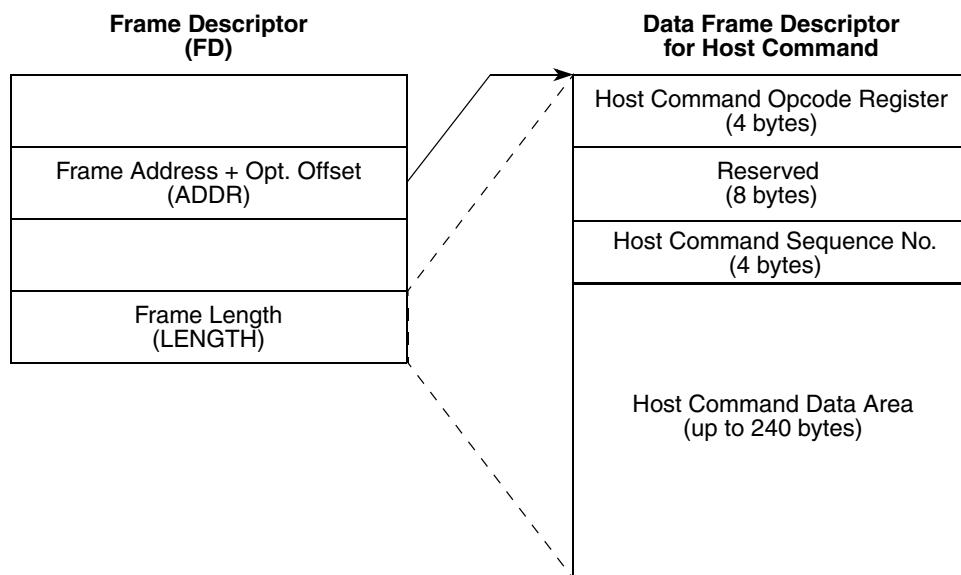


Figure 5-489. Data Frame Descriptor For IP Reassembly TimeOut Configuration Host Command

- Host Command Opcode Register (HCOR), Opcode Field should be set to 0x10 to apply IP Reassembly TimeOut Configuration Host Command. For detailed information about the Host Command Opcode Register (HCOR) See [Section 5.12.16.4.1, "Host Command Opcode Register \(HCOR\)."](#)
- TimeOut Configuration Information is placed in Host Command Action Register (HCAR) and Host Command Extra Register (HCER) as described in [Table 5-527. “IP Reassembly Timeout Configuration Host Command HCAR and HCER.”](#)
- Host Command Sequence No. Register is used as described in details in [Section 5.12.16.4.4, "Host Command Sequence Number \(HCSN\)."](#)
- The IP Reassembly Timeout functionality consumes internal buffers out of the total budget allocated to the Host Command port. The number of consumed buffers depends on the number of active timeout functions (i.e. the number of times that the host command was issued with ‘activate’

subcommand less the number of times that the host command was issued with ‘disable’ subcommand). The number of consumed buffers equals to the number of active timeout functions plus 1. In case that the default number of buffers is insufficient, it is possible to change it by writing to FMBM\_PFS[IFSZ].

**Table 5-527. IP Reassembly Timeout Configuration Host Command HCAR and HCER**

Register	Offset	Name	Description
HCAR	0x0	SUBCMD	<p>Sub Command. The timeout configuration host command has several sub commands. This field indicates the sub command when the host issues the command.</p> <p><b>0 - ActivateTimeout Task.</b> This sub command activates a timeout task (to scan an associated timeout table). The IPRCPT parameter identifies the IP reassembly function and TSBS defines the period (and therefore the accuracy) for the timeout task.</p> <p><b>1 - Disable Timeout Task.</b> This sub command disables a timeout task. IPRCPT must be specified.</p> <p>Other values in this field are reserved.</p>
	0x1	RESULT	<p>This is the result of the host command. Host commands are returned to confirmation queue and this field indicates if and what error conditions were encountered.</p> <p>0x0 - Command completed with no error 0x1 - Error: failed to allocate TNUM 0x2 - Error: failed to allocate internal buffer (increase the number of buffers allocated to the host command port). 0x3 - Error: 'Disable Timeout Task' with invalid IPRCPT 0x4 - Error: too many timeout tasks 0x5 - Error: invalid sub command</p> <p>Other values in this field are reserved.</p>
	0x2-0x3	-	Reserved
HCER	0x0	TSBS	<p>Time Stamp Bus Scale. Valid values are 0-31 (0- corresponds to FMFP_TSP[0],...,31- corresponds to FMFP_TSP[31]). Controls which bit of the TimeStamp register (FMFP_TSP) is used to determine the (TimeOut Request Time)/2. Affect the half time interval between consecutive IP Reassembly TimeOut routine requests.</p> <p>Assuming that bit FMFP_TSP[23] is toggled each 1us (user configures this using FMFP_TSC1): If TSBS=24, the TimeOut check will be triggered every 1us. If TSBS=14, the TimeOut check will be triggered every 1ms. If TSBS=4, the TimeOut check will be triggered every 1sec.</p> <p>The time period between TimeOut checks is - <math>2^{(24- TSBS)} \times 1\text{us}</math>.</p>
	0x1-0x3	IPRCPT	IP Reassembly Common Parameter Table Pointer.

## 5.12.17.3 IPsec

### 5.12.17.3.1 IPsec Features

IPsec features that FMan-Controller supports are:

- Support IPsec ESP tunnel mode.
- Support IPv4 and/or IPv6 packets may be tunneled within IPv4.
- Identify IPsec error frames.

- Automatically update UDP total length on outer header after encryption.
- For Ingress packets, the size of the outer IPv4 header or IPv6 extensions may vary for different frames belonging to a given SA.
- For Egress packets, the IP Version of the inner packet may vary for different frames belonging to a given SA.
- ECN/DSCP fields propagation.
- For Egress packets, FMan controller removes for short frames any optional L2 padding or other bytes that are not related to the IP packet before performing the encryption.
- Support extended anti-replay window size.

### 5.12.17.3.2 IPsec Introduction

IPsec errors are still directed to the SEC output queue. The FMan Controller checks for these kind of errors, and in case it identifies an error it sets a special “Non-FMAN” error bit in the FD status (bit number 9 (NFE-Non FMan error) as indicated in the [Offline Port Frame Descriptor Command/Status](#) section of the FMan chapter).

For IPsec tunnel frames with inner IP version not the same as the outer IP version, the FMan controller automatically updates the EtherType field in the Ethernet header.

The FMan controller updates the UDP header Length field if it exists on the outer header after encryption.

For Ingress packets, the size of the outer IPv4 header or IPv6 extensions may vary for different frames belonging to a given SA. FMan Controller sets the FD[status] bits as indicated in the DECO Protocol Override Register (DPOVRD) in the Security (SEC) Reference Manual. Updating the IP Header Length field enables support for this feature.

For Egress packets, the IP Version of the inner packet may vary for different frames belonging to a given SA. IPsec encapsulation sets the Next Header field of the ESP trailer according to the IP Version of the frame being encapsulated. The Next Header field is set to 4 for IPv4 and 41 for IPv6. This feature is enabled via the VIPV\_EN option of the IPsec Manipulation Table Descriptor.

For Egress packets, FMan controller removes for short frames any optional L2 padding or other bytes that are not related to the IP packet before performing the encryption.

IPv4 TOS[ECN] field and IPv6 Traffic Class[ECN] are propagated according to RFC 6040. On encryption user can work with compatibility mode which means that the ECN is set to NOT-ECT value, or in normal mode which means that inner ECN bits are copied into outer ECN bits. On decryption the behavior of ECN propagation is according to [Table 5-528](#). Un-used combinations are considered as error frames, and FMan Controller set a special “Non-FMAN” error bit in the FD status (bit number 9 as indicated in the FMAN chapter) and clears the high nibble (bits 0-3) of the FD status so user can distinguish this error from other SEC errors.

IPv4 TOS[DSCP] field and IPv6 Traffic Class[DSCP] can be configured to be copied from inner to outer or vice versa. Another option can be on encryption not change the value of this field that was set by the SEC, and on decryption not change the inner value of this field. On Encapsulation the recommendation is to copy DSCP from inner header into outer header. On Decryption the recommendation is to NOT change the inner DSCP.

**Table 5-528. RFC 6040 IP in IP Decapsulation behaviors**

Arriving Inner Header	Arriving Outer Header			
	Not-ECT	ECT (0)	ECT (1)	CE
Not-ECT	Not-ECT	<drop>	<drop>	<drop>
ECT (0)	ECT (0)	ECT (0)	ECT (1)	CE
ECT (1)	ECT (1)	<drop>	ECT (1)	CE
CE	CE	CE	<drop>	CE

Support extended anti-replay window size than the SEC hardware supports. When this mode is enabled the SEC anti-replay window mechanism should be disabled. After SEC decryption, this mechanism checks if the authenticated (validated) SEC packet sequence number is:

1. lower than the anti-replay lower window limit (LATE) or
2. same sequence number already received in the anti-replay window (REPLAY).

In these cases of LATE or REPLAY the FMan Controller sets a special “Non-FMAN” error bit in the FD status (bit number 9 as indicated in the FMAN chapter) and sets the other bits to indicate anti-replay error in the same manner as the SEC does. This means that for LATE FD[status]=0x40400083 and for REPLAY FD[status]=0x40400084. If the packet sequence number is bigger than the anti-replay upper window limit and it was authenticated, then the window is shifted to this new value.

## 5.12.17.4 Table Descriptors Additions

### 5.12.17.4.1 IP Manipulation Custom-Classification Table Descriptor

IP Manipulation Table Descriptor functionality:

- Determines whether to perform IP Fragmentation.
- Gathers statistics.
- Supports scatter/gather Frame Descriptor format.

#### NOTE

- IP Manipulation Table Descriptor may be preceded by Keep/New Result Action Descriptor - in this case it may adopt the NIA from this AD.

Figure 5-490 and Table 5-529 describes IP Manipulation Table Descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type	CNIA		—	—	—	—	—	—	—	—	—	—	—	—	NextActionDescriptorPointer[0-7]
2																NextActionDescriptorPointer[8-23]
4					—								—			—
6													—			
8									MTU							
A					—								OperationCode=0x34			
C									TotalFrameCount[0-15]							
E									TotalFrameCount[16-31]							

Figure 5-490. IP Manipulation Table Descriptor

Table 5-529. IP Manipulation Table Descriptor

Offset	Bits	Name	Description
0	0-1	Type	Action Descriptor Type. MUST Set to 01. 00 - New classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are taken from this Action Descriptor. 01 - Table Descriptor. 10 - Keep Classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are not overwritten. 11 - Reserved for dynamic update of custom classifier tables.
	2	ChangedNextInvokedAction (CNIA)	Changed Next Invoked Action. This bit controls where processing will be forwarded to. This bit may be set only if MTU = 0xFFFF. 0 - If IP Fragmentation is required, processing is directed to the IP Fragmentation process. If no IP Fragmentation is required, then IP Manip directs processing to the PreBMIPrepareToEnq FMan Controller NIA. 1 - IP Manip directs processing to the NIA that is set by the preceding Keep or New Action Descriptor. <b>Note:</b> If CNIA is set, a PreBMIPrepareToEnq NIA followed by a BMIPrepareToEnq NIA must reside in the chain following this Table Descriptor before QMIEnq.
	3-4	Reserved	Reserved. Must be cleared
	5	Reserved	Reserved. Must be cleared.
	6	Reserved	Reserved. Must be cleared.
	7	Reserved	Reserved. Must be cleared.
	8-31	Next ActionDescriptor Pointer	Next Action Descriptor FMan memory Pointer. Points to the next action descriptor in Action Descriptor chain. Memory allocation requirementsFMan memory: 16 bytes.
4	0-31	Reserved	Reserved. Must be cleared

Offset	Bits	Name	Description
8	0-15	MTU	Maximum Transmit Unit. Used to determine to terminate action chain or to continue to the next Action Descriptor pointed by Next Action Descriptor Pointer. MTU >= IP packet Length or MTU = 0xFFFF. This Table Descriptor is the Last Action Descriptor in chain. MTU < IP packet Length. Next Action Descriptor to be processed is pointed by Next Action Descriptor pointer.
	16-23	Reserved	Reserved. Must be cleared
	24-31	OperationCode	Operation Code. Specifies the specific/generic action type. See <a href="#">Table 5-467</a> , IP Manipulation. MUST be set to 0x34.
C	0-31	TotalFrameCount	Total frame counter. Counts each frame processed by this Table Descriptor. MUST be initialized to zero.

#### 5.12.17.4.2 IP Fragmentation Table Descriptor

The IP Fragmentation table descriptor triggers the IP Fragmentation function. Please refer to Section 5.12.17.1, "IP Fragmentation" for a full description of the IP Fragmentation function.

The figure below describes the IP Fragmentation Table Descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type		DFAction	BPID En	OptCo unter En	—										FragmentedFramesCounter
2																FragmentedFramesCounter (cont)
4																GeneratedFragmentsCounter
6																
8					BufferPoolID											ScratchAllocFailureCounter
A					AllocFailureCounter											OperationCode=0x74
C					ScratchBufferPoolID											FragIDPtr
E																FragIDPtr (cont)

Figure 5-491. IP Fragmentation Table Descriptor

**Table 5-530. IP Fragmentation Table Descriptor**

Offset	Bits	Name	Description
0	0-1	Type	Action Descriptor Type. Set to 01. 00 - New Classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are taken from this Action Descriptor. 01 - Table Descriptor. 10 - Keep classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are not over written. 11 - Reserved for dynamic update of custom classifier tables.
	2-3	DFAction	Don't Fragment Action. If an IP packet is larger than MTU and its DF bit is set, then this field will determine the action to be taken. 00 - Discard packet. 01 - Fragment packet and continue normal processing. The DF field of the resulting fragments is set to zero. 10 - Continue normal processing without fragmenting the packet. 11 - Reserved.
	4	BPIDEn	0 - The original frame's BPID is used by the IP Fragmentation function for buffer allocation. 1 - The BufferPoolID in this Table Descriptor is used by the IPFragmentation function for buffer allocation.
	5	OptCounterEn	Options Counter Enable. 0 - Options counter is disabled. 1 - A counter is incremented each time an IPv4 frame with IPv4 Options is encountered and the COPIED flag on one of the options is cleared. The counter is incremented at most once per frame. The counter is located on the Port Page. See OptCounter in <a href="#">Section 5.12.23, "FMan-Controller Parameters Page per Port."</a>
	6-7	Reserved	Reserved. Must be cleared.
	8-15	FragmentedFramesCounter	Fragmented Frames Counter. This counter counts the number of fragmented frames.
2	0-15		
4	0-31	GeneratedFragmentsCounter	Generated Fragments Counter. This counter counts the number of fragments generated.
8	0-7	BufferPoolID	If BPIDEn is set, then this Buffer Pool ID is used to allocate buffers during IP fragmentation. For the required buffer length of this pool, see <a href="#">Section , "Initialization of IP Fragmentation."</a>
	8-15	ScratchAllocFailureCounter	Valid only for non FMan_v3 devices. Should be initialized to 0. Scratch Buffer Pool Allocation failure counter. This counter is incremented each time allocation from the Scratch Buffer Pool fails. In order to avoid allocation failures, the Scratch Buffer Pool should be enlarged. See Scratch Buffer PoolID for additional information.

**Table 5-530. IP Fragmentation Table Descriptor (continued)**

Offset	Bits	Name	Description
A	0-7	AllocFailureCounter	Allocation failure counter. External buffers are allocated during the fragmentation process in order to build the Scatter/Gather output fragments. Buffers are allocated from the Buffer Pool (BPID) of the original frame being fragmented or from the BufferPoolID programmed in this table. For the required buffer length of this pool, see <a href="#">Section , "Initialization of IP Fragmentation."</a> This counter is incremented each time the allocation fails. Aside for incrementing the counter, the fragmentation process continues to re-try to allocate an external buffer until the allocation succeeds. Each subsequent re-try does not increment the counter. This field must be initialized to zero by the CPU.
	8-15	OperationCode	Operation Code. Specifies the specific/generic action type. This code should be set to 0x74.
C	0-7	ScratchBufferPoolID	Scratch Buffer Pool ID. This Buffer Pool is required by the fragmentation process in order to ensure correct release operation of the frames and fragments. For Frame Manager version v3 and later, this must be set to 0xFF.
C	8-31	FragIDP[0-23]	Pointer to fragment ID. The fragmentation process will use this ID for the next IPv6 frame to be fragmented and will then increment this value. This pointer points to a 4-byte area in the internal memory. The user must allocate these 4 bytes and initialize them to 0. Multiple IP Fragmentation Table Descriptors may or may not point to the same location in memory. However, all IP Fragmentation Table Descriptors belonging to the same IP-source/IP-destination flow MUST point to the same location in memory in order to ensure a unique ID generation for each frame in the flow.

### 5.12.17.4.3 IP Reassembly Table Descriptor

The IP Reassembly table descriptor triggers the IP Reassembly function. The IP Reassembly Table Descriptor contains parameters that are specific to either the IPv4 reassembly function or to the IPv6 reassembly function. If both IPv4 reassembly and IPv6 reassembly is required, then two separate IP Reassembly Table Descriptors are required. Please refer to Section 5.12.17.2, "IP Reassembly" for a full description of the IP Reassembly function.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type=01				—											IPReassParamsTblPointer[0-7]
2																IPReassParamsTblPointer[8-23]
4			—									—				
6				BufferPoolID								—				
8							—									
A				—												OperationCode=0xB4
C	NCSPF QIDM	—		—												Non Consistent SP FQID[0-7]
E																Non Consistent SP FQID[8-23]

**Figure 5-492. IP Reassembly Table Descriptor**

**Table 5-531. IP Reassembly Table Descriptor**

Offset	Bits	Name	Description
0	0-1	Type	Action Descriptor Type. Set to 01. 00 - New classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are taken from this Action Descriptor. 01 - Table Descriptor. 10 - Keep Classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are not overwritten. 11 - Reserved for dynamic update of custom classifier tables.
	2-7	Reserved	Reserved. Must be cleared.
	8-31	IPReassParamsTblPointer	Pointer to the IP Reassembly Parameters Table. see <a href="#">Section , "IP Reassembly Parameters Table."</a>
4	0-15	Reserved	Reserved. Must be cleared.

Offset	Bits	Name	Description
6	0-7	BufferPoolID	<p>Buffer pool ID (BPID). Valid only for Non FMan_v3 devices. For FMan_v3 devices should be cleared.</p> <p>External buffers are allocated during the reassembly process. Each reassembled frame requires one additional external buffer allocation, the Fman Controller writes to this buffer the scatter gather list for the reassembled frame. The Buffer Pool ID field should be configured to the pool from which external buffers will be fetched. These buffers should be at least (256+input fragment FD[offset] ignoring the 4 lsb of this offset) bytes.</p> <p><b>Note:</b> For FMan_v3 devices the Buffer Pool ID used is the same buffer Pool ID as for the input fragment. These buffers should be at least (256+input fragment FD[offset] ignoring the 4 lsb of this offset) bytes.</p>
	8-15	Reserved	Reserved. Must be cleared.
8	0-23	Reserved	Reserved. Must be cleared.
	24-31	OperationCode	Operation Code = 0xB4. IP Reassembly.
C	0	NCSPFQIDM	<p>Valid only for FMan_v3 devices. Non Consistent Storage Profile FQID Mode.</p> <p>0 - Disabled.</p> <p>1 - Enabled. On event of Non Consistent Storage Profile the reassembled frame is enqueued into special queue as indicated in field “Non Consistent SP FQID”. For more information see <a href="#">Section ,“IP Reassembly Process.”</a></p>
	1	Reserved	Reserved. Must be cleared.
	2-7	Reserved	Reserved. Must be cleared.
	8-31	Non Consistent SP FQID	Valid only for FMan_v3 devices. Non Consistent Storage Profile FQID. Valid only if NCSPFQIDM is set. For more information see <a href="#">Section ,“IP Reassembly Process.”</a>

#### 5.12.17.4.4 IPsec Manipulation Table Descriptor

IPsec Manipulation table descriptor is responsible to support the following IPsec features:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type=01	—	DEC	VIPV_EN	ECN_EN	CDSCP_EN	VIPL_E_N									AntiReplayWindowPointer[0-7]
2																AntiReplayWindowPointer[8-23]
4																ARWSIZEFactor
6																NextActionDescriptorIndex
8	—	NADEN		—												OuterIPHdrLen
A	—			—												OperationCode = 0xF4
C																—
E																

Figure 5-493. IPsec Manipulation Table Descriptor

Table 5-532. IPsec Manipulation Table Descriptor (Type = 01)

Offset	Bits	Name	Description
0x0	0–1	Type	Action Descriptor Type. Set to 01.
	2	—	Reserved. Must be cleared.
	3	DEC	ECN/DSCP Decryption flow. 0 - Encryption flow. Removing optional short frames L2 padding or other bytes that are not related to the IP packet before performing the encryption. 1 - Decryption flow.
	4	VIPV_EN	Variable IP Version Enable. When enabled, the OuterIPHdrLen field must be valid. 0 - Variable IP Version is disabled. 1 - Variable IP Version is enabled.
	5	ECN_EN	Valid only for Ethernet frames. ECN propagation Enabled. 0 - ECN propagation is disabled. 1 - ECN propagation is enabled.
	6	CDSCP_EN	Valid only for Ethernet frames. Copy DSCP Enable. 0 - DSCP field is not propagated. 1 - DSCP field is propagated.

**Table 5-532. IPsec Manipulation Table Descriptor (Type = 01) (continued)**

Offset	Bits	Name	Description
	7	VIPL_EN	Variable IP header length (or IPv6 extensions length) enabled. May be set only on Pre-SEC port for decryption flow. 0 - All frames belonging to one IPsec SA have the same IP header length (or IPv6 extensions length). 1 - Frames belonging to one IPsec SA may have different IP header length (or IPv6 extensions length).
	8-31	AntiReplayWindow Pointer[0-23]	Anti-replay window pointer. Valid only if AntiReplayWindowSize>0. Must be 4 byte aligned. User should allocate in internal memory (MURAM) $2^x$ bits plus 4 bytes, where $2^x$ is at least window size + 32. User must clear this area. For example: 96<window size <= 224 need to allocate 32+4=36 bytes. 224<window size<=480 need to allocate 64+4=68bytes. 480<window size<= 992 need to allocate 128+4=132 bytes. 992<window size<= 2016 need to allocate 256+4=260 bytes.
0x4	0-11	AntiReplayWindowSize[0-11]	Anti-replay window size. Valid only for decryption mode. Only the 12 most significant bit of the window size are written and the 4 lsb of the window size are always assumed to be 0. This means actual window size is always multiplication of 16. 0 - Anti-replay mode is disabled after decryption. Using SEC anti-replay mode is optional. 1 - 4094- Anti-replay mode is enabled and window size is set to this value multiplied by 16. SEC anti-replay window mechanism must be disabled.
	12-15	ARWSizeFactor	Anti-replay window size plus 32 in power of 2 units minus 5. Valid only if AntiReplayWindowSize>0. For example: 96<window size <= 224 need to set 8-5=3. 224<window size<=480 need to set 9-5=4. 480<window size<= 992 need to set 10-5=5. 992<window size<= 2016 need to set 11-5=6.
	16-31	NextActionDescriptorIndex	Next Action Descriptor Index. Valid only if NADEN=1. Index to the next Action Descriptor. The pointer to the next Action Descriptor is equal to this value multiplied by 16 (no base is added to this result).
0x8	0-1	—	Reserved. Must be cleared.
	2	NADEN	Next Action Descriptor Index Enable. If set the FMan-controller proceeds to the next Action Descriptor according to nextActionDescriptorIndex field, otherwise it is the last table descriptor for this classification process.
	3-7	—	Reserved. Must be cleared.
	8-15	OuterIPHdrLen	Outer IP Header Length. Valid if VIPV_EN=1. Specifies the length of the outer IP header to be used for IPsec ESP Tunnel mode encapsulation.
	16-19	—	Reserved. Must be cleared.
	20-23	—	Reserved. Must be cleared.
	24-31	OperationCode	Operation Code. Must be set to 0xF4. See <a href="#">Table 5-467</a>
0xC	0-31	—	Reserved. Must be cleared.

## 5.12.18 Statistics Gathering

The following statistics gathering features are available through the Statistics Table Descriptor:

- Frame and byte count statistics
- Statistics per Frame Length Range (FLR)
- Conditional Statistics that count the actual transmitted frames/bytes.

Only one of the above 3 types of statistics can be enabled by a single Statistics Table Descriptor.

The byte-count statistics reflect the length indicated by the Frame Descriptor Length field.

### 5.12.18.1 Frame and Byte Count Statistics

The Frame and Byte Count Statistics are gathered per flow. Each counter is 32-bit wide. The figure below shows the memory structure of the frame and byte count statistics. This structure resides in internal memory.

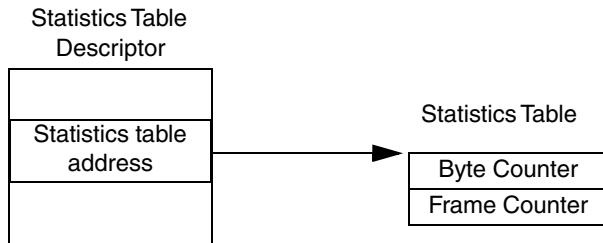
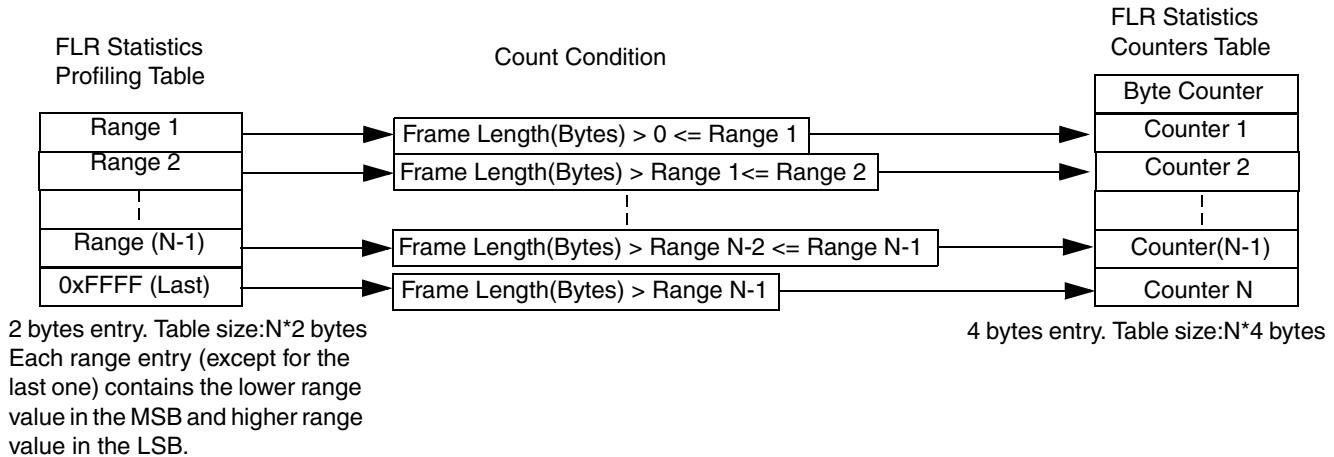


Figure 5-494. Frame and Byte Count Statistics

### 5.12.18.2 Frame Length Range Statistics (FLR)

The Frame Length Range Statistics categorizes frames according to their length. The number of frame length ranges and the range values are configurable. The number of frames is counted per range. The number of bytes is counted globally for all the ranges (See FLR Statistics Profiling Table Address in [Table 5-533](#)). Each counter is 32-bit wide. The figure below shows the memory structure of the FLR statistics counters. This structure resides in internal memory.



**Figure 5-495. Frame Length Range Statistics**

**NOTE**

The FLR Statistics Profiling Table can be shared by 2 or more Statistics Table Descriptors (i.e. Same FLR Statistics Profiling Table Address).

### 5.12.18.3 Conditional Statistics (Fman\_v3 only)

The Conditional Statistics count only frames that are actually transmitted. The byte count reflects the actual size of the transmitted frame. Each counter is 32-bit wide. The figure below shows the memory structure of the Conditional Statistics counters. This structure resides in internal memory.

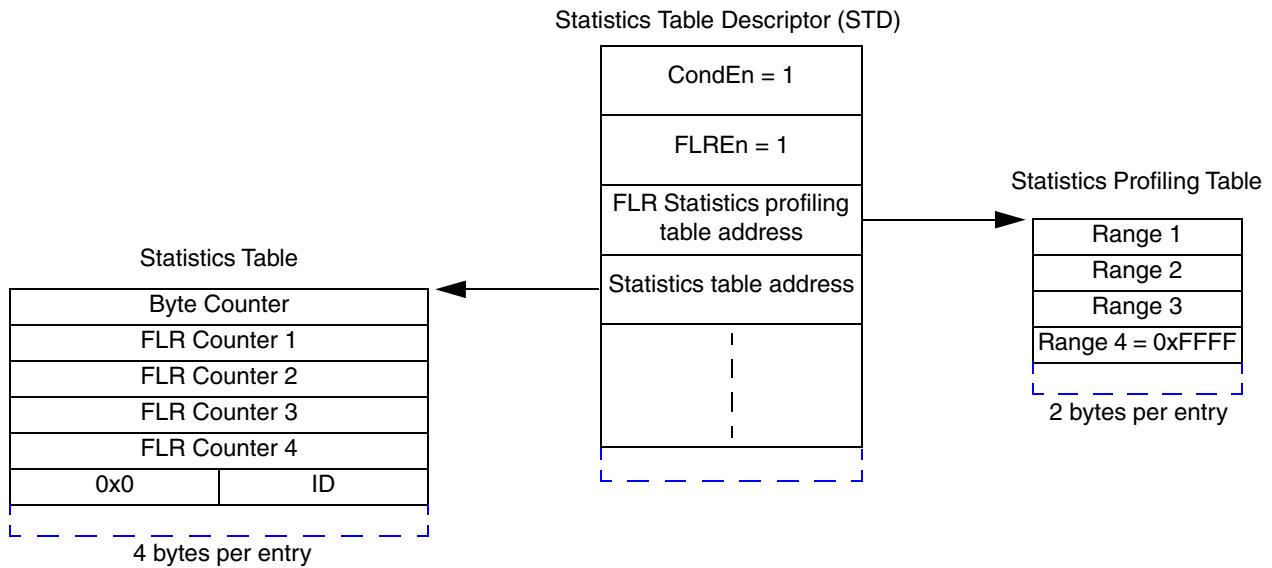


Figure 5-496. Conditional Statistics

#### 5.12.18.3.1 Configuring the Conditional Statistics

To configure the Conditional Statistics, the user should:

- Configure the Statistics Table Descriptor (STD) as follows:
  - Set STD[CondEn] = 1
  - Set STD[FLREn] = 1
  - STD[FLR Statistics Profiling Table Address] should contain the address of a 4-entry FLR Statistics Profiling Table as described in [Section 5.12.18.2, "Frame Length Range Statistics \(FLR\)."](#)
  - STD[Statistics Table Address] should contain the address of the following structure:
    - Byte counter (4 bytes). Should be initialized to zero.
    - 4 FLR counters (4 bytes each). Should be initialized to zero.
    - Zero padding (2 bytes). Must be initialized to zero.
    - ID field (2 bytes). The user should manage this field as described in "[Set FQD\[ContextA\]\[A2\]\[NL\] = 1](#)".
- Configure the BMI register FMBM\_TCMNE to contain the FMan Controller Post Transmission NIA value (0x24).
- Configure BMI to transfer the IC[Hash] field from the O/H port to the Tx port (FMBM\_xICP registers).
- For each of the Tx dequeue queues in which statistics are gathered:
  - Set FQD[ContextA][A1V] = 1
  - Set FQD[ContextA][A1] = 1

- Set FQD[ContextA][A2V] = 1
- Set FQD[ContextA][A2][NL] = 1

### 5.12.18.3.2 Conditional Statistics Dynamic Changes

To make dynamic changes to the Conditional Statistics, the StatisticsTable[ID] field, located at offset 0x16 from the base of the Statistics Table, should be managed by the user in the following way:

- The user should maintain a 2 byte unsigned global integer which is incremented by 1, for each conditional STD creation/modification. The counter may wrap, but a value of zero should be skipped.
- Each time a conditional STD is created/modified, the StatisticsTable[ID] field should be set to the current value of the global variable above.
- Each time a conditional STD is deleted, the StatisticsTable[ID] field should be set to zero.
- Before issuing a Host Command for a dynamic change, the user must first change the StatisticsTable[ID] as described above and verify that change was applied and only then issue the Host Command.

### 5.12.18.3.3 Conditional Statistics Restrictions

- Conditional Statistics are supported starting with FMan\_v3.
- Conditional Statistics support only Frame Length Range statistics.
- IP Fragmentation must not be applied after Conditional Statistics.
- Keygen must not be applied after Conditional Statistics.
- After Conditional Statistics, another Conditional Statistics must not be invoked.
- The FLR Statistics Profiling Table and Statistics Table must not be de-allocated at any time.

### 5.12.18.4 Statistics Table Descriptor (STD)

The following figure shows the structure of the Statistics Table Descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type=01				—											FLR Statistics profiling table address [0-7]
2																FLR Statistics profiling table address [8-23]
4																—
6																—
8																NextActionDescriptorIndex
A	NAD En	FLR En	Cond En	—	—	—	—	—	—	—	—	—	—	—	—	OperationCode = 0x36 (STD)
C				—	—	—	—	—	—	—	—	—	—	—	—	Statistics table address [0-7]
E																Statistics table address [8-23]

Figure 5-497. Statistics Table Descriptor (Type = 01)

This table describes Statistics table descriptor fields:

**Table 5-533. Statistics Table Descriptor fields description (Type = 01)**

Offset	Bits	Name	Description
0	0–1	Type	Type - Table Descriptor, Set to 0b01.
	2–7	—	Reserved. Must be cleared.
	8–31	FLR Statistics profiling table address	Valid only if FLREn bit == ‘1’, else this field is reserved and should be cleared. Points to a FMan Memory data structure that contains N Frame length range Thresholds (Last entry is a constant last flag 0xFFFF). The thresholds are 16 bits size (See <a href="#">Section 5.12.18.1, “Frame and Byte Count Statistics”</a> for more information). In FMan memory, Allocate ( $N^2$ ) bytes aligned to 8.
4	0–31	—	Reserved. Must be cleared.
0x8	0–15	NextActionDescriptorIndex	Index to the next Custom Classifier structure. The pointer to the next Custom Classifier structure is equal to this value multiplied by 16 (no base is added to this result).
	16	NADEn	NextActionDescriptorEnable (NADEn) 0 - This is the last Custom Classifier structure. The Statistics Table Descriptor (STD) does not change any of the pre-determined Custom Classifier result (i.e. NIA/FQID) 1 - The FMan-controller proceeds to the next Custom Classifier structure (Action/Table Descriptor) according to NextActionDescriptorIndex field.
	17	FLREn	Frame Length Range Enable. 1 - Enables Frame Length Range statistics (FLR). ‘FLR Statistics profiling table address’ and ‘Statistics table address’ fields must be initialized. 0 - Frame Length Range statistics is disabled. Only Frame/Byte Statistics are gathered. ‘Statistics table address’ field must be initialized.
	18	CondEn	Conditional Statistics Enable. Valid only for FMan_v3. 1 - Conditional Statistics are enabled. FLREn must also be set. See <a href="#">Section 5.12.18.3, “Conditional Statistics (Fman_v3 only)”</a> . 0 - Conditional Statistics are disabled. Either Frame/Byte Count statistics or FLR statistics are enabled.
	19–23	—	Reserved. Must be cleared.
	24–31	OperationCode	Operation Code. Must be set to 0x36. See <a href="#">Table 5-467</a>
0xC	0–7	—	Reserved. Must be cleared.
0xC	8–31	Statistics table address	Statistics Table Address If FLREn=0 (Non-FLR mode), points to the Frame/Byte Statistics Table. (See <a href="#">Section 5.12.18.1, “Frame and Byte Count Statistics”</a> ). Table must be 8-byte aligned. If FLREn=1 (FLR mode), points to the FLR Statistics Counters Table. (See <a href="#">Section 5.12.18.2, “Frame Length Range Statistics (FLR)”</a> ). In FMan memory, Allocate $((N + 1) * 4)$ bytes aligned to 8.

## 5.12.19 Next Generation CAPWAP Programming Model Additions

Next Generation CAPWAP is targeted for FMan\_v3 devices. This package offers the following features:

- CAPWAP encapsulation and decapsulation
- CAPWAP header insertion/removal

- CAPWAP DTLS header insertion/removal
- L2/L3 (IPv4 and IPv6)/ L4 (both UDP and UDP-Lite) header insertion capabilities
- Support DTLS encryption/description flows with SEC error checks
- Support propagation of TOS field per frame on egress flow
- Support CAPWAP fragmentation and CAPWAP reassembly
  - All features are identical to IP fragmentation and IP reassembly with the following additions
    - Support single buffer CAPWAP reassembly
- Support classification on CAPWAP/802.11 data frames

### 5.12.19.1 CAPWAP Fragmentation and CAPWAP Reassembly

The CAPWAP fragmentation and CAPWAP reassembly processes are identical to the IP Reassembly and IP Fragmentation. Please refer to [Section 5.12.17.1, "IP Fragmentation"](#) and [Section 5.12.17.2, "IP Reassembly."](#) Therefore in this sections any reference to IP is also relevant to CAPWAP. Any reference to IPRE (IP reassembly error) should be considered as CRE (CAPWAP Reassembly Error).

Unlike IP Reassembly, CAPWAP Reassembly can only run on OP, and supports single buffer CAPWAP reassembly. Non-Consistent Storage Profile event or mode is not relevant for CAPWAP reassembly. In addition there is no minimum fragment size check for the non last fragments, instead there is check for exceeding maximum size reassembled frame which is not valid for IP reassembly. See more details [Table 5-524](#) for parameter called ‘ExceedMaxReassembledFrameLength’.

#### NOTE

CAPWAP fragmentation process requires that input frame (frame header just before entering CAPWAP fragmentation TD) is starting with CAPWAP header or CAPWAP DTLS header.

CAPWAP reassembly process requires that input fragment (fragment header just before entering CAPWAP reassembly TD) is starting with CAPWAP header

### 5.12.19.2 Next Generation CAPWAP Table Descriptor Additions

#### 5.12.19.2.1 CAPWAP Fragmentation Condition Check Custom-Classification Table Descriptor

IP Fragmentation Condition Check Table Descriptor functionality:

- Determines whether to perform CAPWAP Fragmentation.
- Gathers statistics.

#### NOTE

- CAPWAP Fragmentation Condition Check Table Descriptor may be preceded by Keep/New Result Action Descriptor - in this case it may adopt the NIA from this AD.

[Figure 5-498](#) and [Table 5-534](#) describes CAPWAP Fragmentation Condition Check Table Descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15									
0	Type	CNIA		3	4	—	5	6	7	8	9	10	11	12	13	14	15								
2	NextActionDescriptorPointer[0-7]																								
4	NextActionDescriptorPointer[8-23]																								
6	—																								
8	MTU																								
A	—							OperationCode=0x2E																	
C	TotalFrameCount[0-15]																								
E	TotalFrameCount[16-31]																								

Figure 5-498. CAPWAP Fragmentation Condition Check Table Descriptor

Table 5-534. CAPWAP Fragmentation Condition Check Table Descriptor

Offset	Bits	Name	Description
0	0-1	Type	Action Descriptor Type. MUST Set to 01. 00 - New classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are taken from this Action Descriptor. 01 - Table Descriptor. 10 - Keep Classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are not overwritten. 11 - Reserved for dynamic update of custom classifier tables.
	2	ChangedNextInvokedAction (CNIA)	Changed Next Invoked Action. This bit controls where processing will be forwarded to. This bit may be set only if MTU = 0xFFFF. 0 - If CAPWAP Fragmentation is required, processing is directed to the CAPWAP Fragmentation process. If no CAPWAP Fragmentation is required, then frame is directed to the PreBMIPrepareToEnq FMan Controller NIA. 1 - Frame is directed to the NIA that is set by the preceding Keep or New Action Descriptor. <b>Note:</b> If CNIA is set, a PreBMIPrepareToEnq NIA followed by a BMIPrepareToEnq NIA must reside in the chain following this Table Descriptor before QMIEnq.
	3-7	Reserved	Reserved. Must be cleared.
	8-31	Next ActionDescriptor Pointer	Next Action Descriptor FMan memory Pointer. Points to the next action descriptor in Action Descriptor chain. Memory allocation requirements FMan memory: 16 bytes.
4	0-31	Reserved	Reserved. Must be cleared

Offset	Bits	Name	Description
8	0-15	MTU	Maximum Transmit Unit considering CAPWAP packet length including CAPWAP header. Used to determine to terminate action chain or to continue to the next Action Descriptor pointed by Next Action Descriptor Pointer. MTU >= CAPWAP packet Length or MTU = 0xFFFF. This Table Descriptor is the Last Action Descriptor in chain. MTU < IP CAPWAP packet Length. Next Action Descriptor to be processed is pointed by Next Action Descriptor pointer.
	16-23	Reserved	Reserved. Must be cleared
	24-31	OperationCode	Operation Code. Specifies the specific/generic action type. See <a href="#">Table 5-467</a> , CAPWAP Fragmentation Condition Check. MUST be set to 0x2E.
C	0-31	TotalFrameCount	Total frame counter. Counts each frame processed by this Table Descriptor. MUST be initialized to zero.

### 5.12.19.2.2 CAPWAP Reassembly Table Descriptor

The CAPWAP Reassembly table descriptor triggers the CAPWAP Reassembly function.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type=01				—											CAPWAPReassParamsTblPointer[0-7]
2																CAPWAPReassParamsTblPointer[8-23]
4			—									—				
6			—									—				
8					—											
A			—													OperationCode=0x30
C							—									
E								—								

**Figure 5-499. CAPWAP Reassembly Table Descriptor**

**Table 5-535. CAPWAP Reassembly Table Descriptor**

Offset	Bits	Name	Description
0	0-1	Type	Action Descriptor Type. Set to 01. 00 - New classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are taken from this Action Descriptor. 01 - Table Descriptor. 10 - Keep Classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are not overwritten. 11 - Reserved for dynamic update of custom classifier tables.
	2-7	Reserved	Reserved. Must be cleared.
	8-31	CAPWAPReassParamsTblPointer	Pointer to the CAPWAP Reassembly Parameters Table. see <a href="#">Section , "IP Reassembly Parameters Table."</a>
4	0-15	Reserved	Reserved. Must be cleared.

Offset	Bits	Name	Description
6	0-15	Reserved	Reserved. Must be cleared.
8	0-23	Reserved	Reserved. Must be cleared.
	24-31	OperationCode	Operation Code = 0x30. CAPWAP Reassembly.
C	0-31	Reserved	Reserved. Must be cleared.

### 5.12.19.2.3 CAPWAP Fragmentation Table Descriptor

The CAPWAP Fragmentation table descriptor triggers the CAPWAP Fragmentation function. This table descriptor should be preceded by the CAPWAP Fragmentation Condition Check Table Descriptor which is responsible for checking the MTU against the CAPWAP packet length.

The figure below describes the CAPWAP Fragmentation Table Descriptor.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type	—	BPID En	Capwa pHead erOptio nsCom pressE n	—	—	—	—	—	—	—	—	—	—	—	FragmentedFramesCounter
2	FragmentedFramesCounter (cont)															
4																GeneratedFragmentsCounter
6																
8	BufferPoolID															—
A	AllocFailureCounter															OperationCode=0x33
C	ScratchBufferPoolID															FragIDPtr
E	FragIDPtr (cont)															

Figure 5-500. CAPWAP Fragmentation Table Descriptor

**Table 5-536. CAPWAP Fragmentation Table Descriptor**

Offset	Bits	Name	Description
0	0-1	Type	Action Descriptor Type. Set to 01. 00 - New Classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are taken from this Action Descriptor. 01 - Table Descriptor. 10 - Keep classification result. Need to initiate the next module (for example, Policer) and the FQID (and optional Policer Profile) are not over written. 11 - Reserved for dynamic update of custom classifier tables.
	2-3	Reserved	Reserved. Must be cleared.
	4	BPIDEn	0 - The original frame's BPID is used by the CAPWAP Fragmentation function for buffer allocation. 1 - The BufferPoolID in this Table Descriptor is used by the CAPWAP Fragmentation function for buffer allocation.
	5	CapwapHeaderOptionsCompressEn	CAPWAP Header Options Compress Enable mode. When this mode is enabled then only the first fragment include the CAPWAP header options field (if user provides it in the input frame) and all other fragments exclude the CAPWAP options field (CAPWAP header is updated accordingly). 0 - CAPWAP Header Options Compress Enable mode is disabled. 1 - CAPWAP Header Options Compress Enable mode is enabled.
	6-7	Reserved	Reserved. Must be cleared.
	8-15	FragmentedFramesCounter	Fragmented Frames Counter. This counter counts the number of fragmented frames.
2	0-15		
4	0-31	GeneratedFragmentsCounter	Generated Fragments Counter. This counter counts the number of fragments generated.
8	0-7	BufferPoolID	If BPIDEn is set, then this Buffer Pool ID is used to allocate buffers during CAPWAP fragmentation. For the required buffer length of this pool, see <a href="#">Section , "Initialization of IP Fragmentation."</a>
	8-15	Reserved	Reserved. Must be cleared.
A	0-7	AllocFailureCounter	Allocation failure counter. External buffers are allocated during the fragmentation process in order to build the Scatter/Gather output fragments. Buffers are allocated from the Buffer Pool (BPID) of the original frame being fragmented or from the BufferPoolID programmed in this table. For the required buffer length of this pool, see <a href="#">Section , "Initialization of IP Fragmentation."</a> This counter is incremented each time the allocation fails. Aside for incrementing the counter, the fragmentation process continues to re-try to allocate an external buffer until the allocation succeeds. Each subsequent re-try does not increment the counter. This field must be initialized to zero by the CPU.
	8-15	OperationCode	Operation Code. Specifies the specific/generic action type. This code should be set to 0x33.

**Table 5-536. CAPWAP Fragmentation Table Descriptor (continued)**

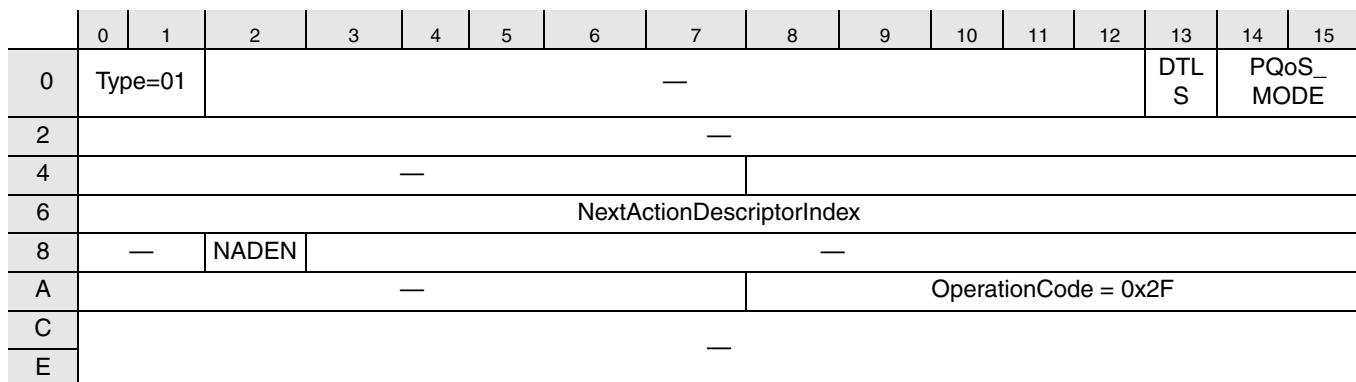
Offset	Bits	Name	Description
C	0-7	ScratchBufferPoolID	Scratch Buffer Pool ID. must be set to 0xFF.
C	8-31	FragIDP[0-23]	Pointer to CAPWAP fragment ID. The fragmentation process will use this ID for the next CAPWAP frame to be fragmented and will then increment this value. This pointer points to a 2-byte area in the internal memory. The user must allocate these 2 bytes and initialize them to 0. Multiple CAPWAP Fragmentation Table Descriptors may or may not point to the same location in memory. However, all CAPWAP Fragmentation Table Descriptors belonging to the same CAPWAP tunnel MUST point to the same location in memory in order to ensure a unique ID generation for each frame in the tunnel.

#### 5.12.19.2.4 CAPWAP Manipulation Table Descriptor

CAPWAP Manipulation Table descriptor is used for the following functionality:

- If the flow continues to SEC DTLS encryption:
  - Clear the 3 msb in the FD status which are used as SEC command bits.
  - Addition of CAPWAP DTLS header.
  - Optionally propagate QoS per frame: user should locate the QoS (IPv4 TOS or IPv6 traffic Class) in the Internal Context area of the external buffer meta-data and it should be programmed to overwrite the last byte in the HASH field in the IC and set the RPD bit for this frame.

CAPWAP Manipulation table descriptor is described in [Figure 5-501](#).



**Figure 5-501. CAPWAP Manipulation Table Descriptor**

**Table 5-537. CAPWAP Manipulation Table Descriptor (Type = 01)**

Offset	Bits	Name	Description
0x0	0-1	Type	Action Descriptor Type. Set to 01.
	2-12	—	Reserved. Must be cleared.

**Table 5-537. CAPWAP Manipulation Table Descriptor (Type = 01) (continued)**

Offset	Bits	Name	Description
	13	DTLS	DTLS. 0 - This flow does not continue to DTLS encryption. 1 - This flow continues to SEC DTLS encryption. Insert CAPWAP DTLS header.
	14-15	PQoS_MODE	Propagate QoS mode. 0 - IPv4 TOS or IPv6 traffic class field in the user template (L3Header field in the following Insert Local L3 HM command) is not changed. 1 - IPv4 TOS or IPv6 traffic class field is overwritten with the IC parser results last HASH byte. This byte is configured to be overwritten by the IC located in external memory (meta data) when RPD is set. This mode should be followed with “Local Insert L3” HMCD using the same PQoS_MODE. 2 - 3 - Reserved.
0x2	0-15	—	Reserved. Must be cleared.
0x4	0-15	—	Reserved. Must be cleared.
	16-31	NextActionDescriptorIndex	Next Action Descriptor Index. Valid only if NADEN=1. Index to the next Action Descriptor. The pointer to the next Action Descriptor is equal to this value multiplied by 16 (no base is added to this result).
0x8	0-1	—	Reserved. Must be cleared.
	2	NADEN	Next Action Descriptor Index Enable. If set the FMan-controller proceeds to the next Action Descriptor according to nextActionDescriptorIndex field, otherwise it is the last table descriptor for this classification process.
	3-23	—	Reserved. Must be cleared.
	24-31	OperationCode	Operation Code. Must be set to 0x2F. See <a href="#">Table 5-467</a> .
0xC	0-31	—	Reserved. Must be cleared.

## 5.12.20 Frame Replicator Support

Fman controller supports Frame Replicator feature, which enables the same incoming frame to be used by several consumers. Each replication has its own Frame Descriptor (FD), each one can point to a different copy of the frame and can be enqueued to a different frame queue. This mode of operation is supported with the appropriate configuration of operational mode bits (See [Section 5.12.6.1,"New Classification Result"](#) and [Section 5.12.6.2,"Keep Classification Result"](#)).

The user should program the Frame Replicator Source Table Descriptor ([Section 5.12.20.1,"Frame Replicator Source Table Descriptor \(FRSTD\)"](#)) and a chain of action descriptors which are the Frame Replicator Group members ([Section 5.12.6.1,"New Classification Result"](#) and [“Section 5.12.6.2, Keep Classification Result”](#)). In each action descriptor member ‘ExtendedModeEn’ bit and ‘FR’ bit should be set. In each action descriptor member, except for the last one, NL bit should be set. “Next Frame Replicator Member Index” field, multiplied by 16, points to the next Frame Replicator member action descriptor. If bit NADEN is set, classification of the current member can continue to next custom classifier table descriptor (for example, Header Manipulation).

After each frame’s replication is enqueued to QMI, action code 0xE is used by the BMI. See below are some examples that show chain of action descriptors for Frame Replicator.

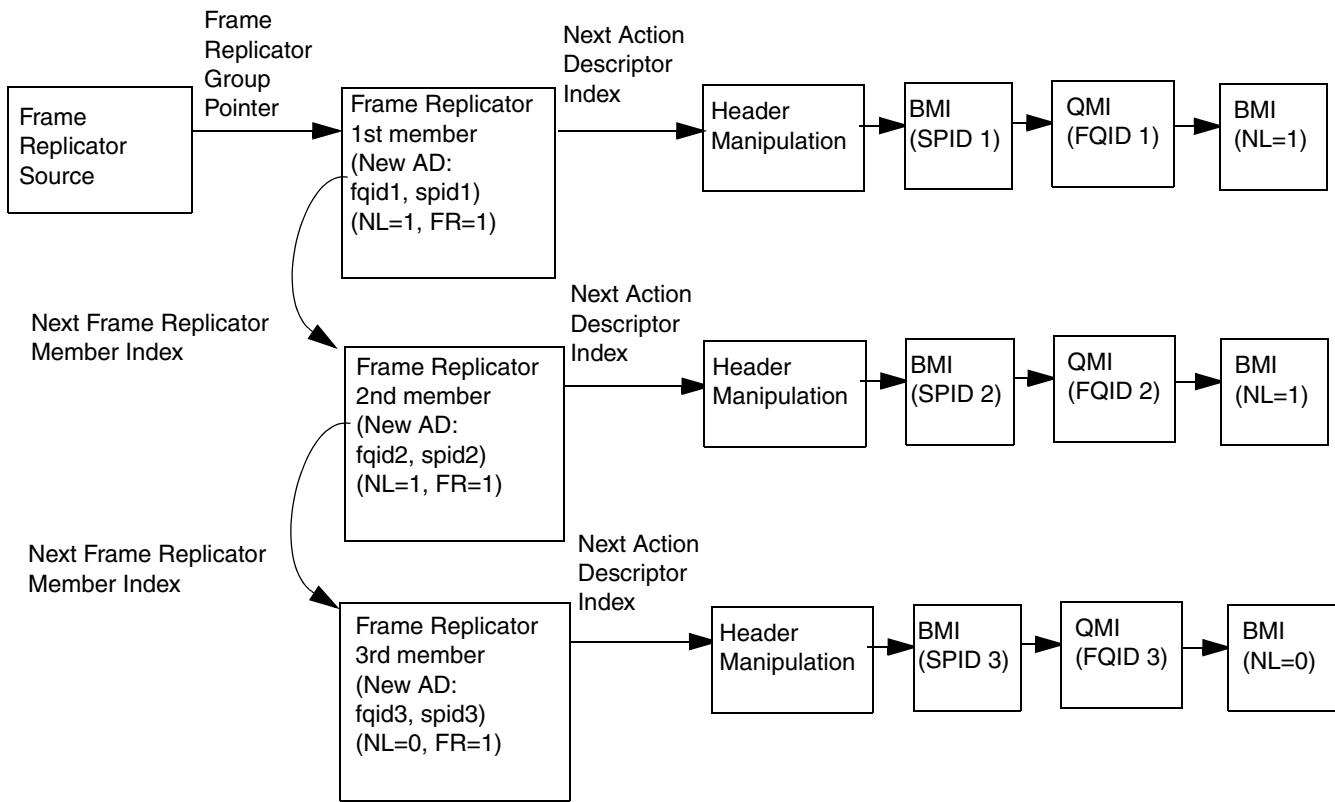
[Figure 5-502](#) shows a Frame Replicator group. The Frame Replicator Source Table Descriptor points to the first Frame Replicator member. Each member’s “Next Action Descriptor Index” points to a Header Manipulation Table Descriptor. Note that Header Manipulation following each member (except for the last member) is allowed only if it does not change the frame’s header protocol stack and size. No such limitation exists for the last member. Each member’s “Next Frame Replicator Member Index” points to the next member in the list. All members are “New AD” which have their own SPID (Storage Profile ID) and FQID values. Note that Header Manipulation following each member requires allocation of an external buffer for every member. The NL bit specifies which members are not the last in the list. Using the list the frame is replicated 3 times. Three FDs, each points to a different copy, are enqueued to different FQIDs.

Each member of the frame replicator group must have its own copy of the replicated buffer. This means that when working on offline port VSP must be enabled on this port. For Rx port frame replication may work even with VSP disabled as long as each member still gets its own replicated buffer.

### NOTE

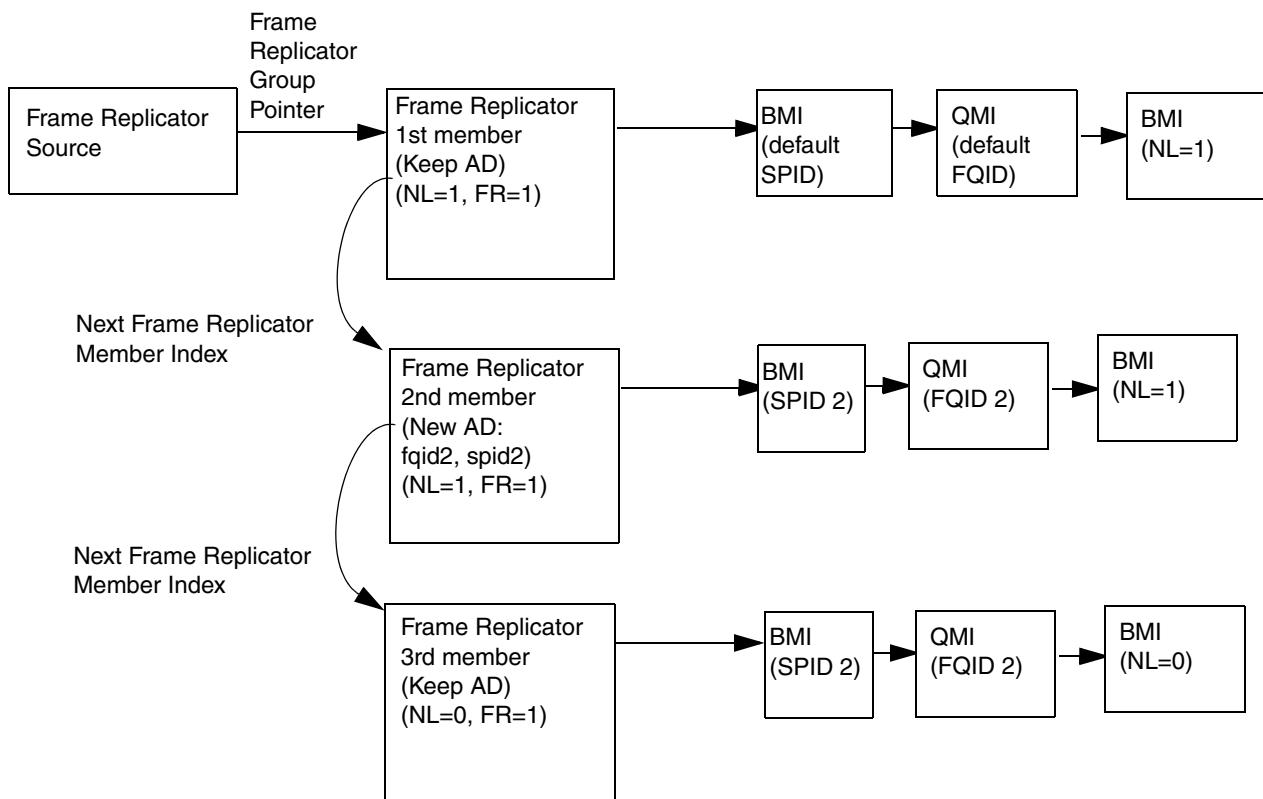
Frame replication cannot work on the same port as IP fragmentation.

IP reassembled frames cannot be subject for frame replication on the same port. If this feature is needed then all traffic which is subject to frame replication needs to be recycled on another offline port before being replicated.



**Figure 5-502. CC list of AD for Frame Replication**

Figure 5-503 shows a Frame Replicator group. The first AD assumes the frame was classified prior to the FMan controller, therefore it is a “keep AD”. The second AD causes the frame to be replicated with a new FQID and SPID. The third AD is again a “keep AD”, therefore it uses the previous FQID and SPID. As a result, first frame is enqueued to a frame queue according to its classification results prior to the FMAN controller, second frame is enqueued to another frame queue, and third frame is enqueued to the same frame queue as the second.



**Figure 5-503. CC list of AD for Frame Replicator (with “Keep” ADs)**

### 5.12.20.1 Frame Replicator Source Table Descriptor (FRSTD)

Frame Replicator Source Table Descriptor (FRSTD) is valid if:

- AD[Type] = 01
- AD[OperationCode] = 0x75

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Type=01				—		—		—			—				
2									—							
4				—												Frame Replicator Group Pointer
6																Frame Replicator Group Pointer (cont)
8									—							
A				—												OperationCode = 0x75
C									—							
E									—							

**Figure 5-504. Frame Replicator Source Table Descriptor (Type = 01)**

This table describes the fields of the Frame Replicator Source table descriptor.

**Table 5-538. Frame Replicator Source Table Descriptor (Type = 01)**

Offset	Bits	Name	Description
0x0	0–1	Type	Action Descriptor Type. Set to 01.
	2–7	—	Reserved. Must be cleared.
	8	—	Reserved. Must be cleared.
	9–31	—	Reserved. Must be cleared.
0x4	0–7	—	Reserved. Must be cleared.
	8–31	Frame Replicator Group Pointer	Pointer to the Frame Replicator Group which is placed in the internal memory (MURAM). This base must be 16 byte aligned.
0x8	0–23	—	Reserved. Must be cleared.
	24–31	OperationCode	Operation Code. Must be set to 0x75. See <a href="#">Table 5-467</a>
0xC	0–31	—	Reserved. Must be cleared.

### 5.12.21 Pre/Post BMI Fetch NIAs

See [Section 5.12.3, "FMan Controller NIA—Action Codes"](#) for a complete list of FMan Controller NIAs.

The PreBMIFetchFrameHdr, PreBMIFetchFullFrame and PostBMIFetch NIAs perform a set of actions as configured by the user. When used, the PreBMIFetchFrameHdr NIA should precede the BMIFetchFrameHdr, the PreBMIFetchFullFrame NIA should precede the BMIFetchFullFrame and the PostBMIFetch NIA should follow BMIFetch.

The PostBMIFetch NIA continues execution to the NIA as configured in the Parameter page per port[PostBMIFetchContNIA]. See [Table 5-540](#).

The PreBMIFetchFrameHdr and PostBMIFetch NIAs perform a set of actions according to the OPCODE that is passed to it per queue. The OPCODE is passed in the ContextA/B fields as follows:

- For FMan\_v3, these are the 4lsb of the frame's ContextA.

After PreBMIFetchFrameHdr, these bits are cleared.

The NIA parameters are represented by the 4-bit OPCODE. Each OPCODE determines which actions to perform as follows:

- 0: frame did not come from SEC. Configure DCL4C (see below).
- 1: Ethernet frame came from SEC-IPsec: configure DCL4C, check SEC errors, fix EtherType field, Update UDP length
- 2: Ethernet frame came from SEC-IPsec: configure DCL4C, check SEC errors, fix EtherType field
- 3: Ethernet frame came from SEC-IPsec: configure DCL4C, check SEC errors, fix EtherType field, Update UDP length, set RPD
- 4: Ethernet frame came from SEC-IPsec: configure DCL4C, check SEC errors, fix EtherType field, set RPD

- 5: Ethernet frame came from SEC-IPsec: configure DCL4C, check SEC errors, fix EtherType field, Update UDP length, set RPD, perform IPsec Manipulation (e.g. Propagate TOS/DSCP/ECN field.)
- 6: Ethernet frame came from SEC-IPsec: configure DCL4C, check SEC errors, fix EtherType field, set RPD, perform IPsec Manipulation (e.g. Propagate TOS/DSCP/ECN field/support anti-replay window.)
- 7: frame did not come from SEC. Set DCL4C.
- 8: frame did not come from SEC. Clear RPD.
- 9: CAPWAP DTLS frame after DTLS decryption: SEC error check, CAPWAP version check, remove CAPWAP DTLS header. If there is CAPWAP version error set the CHE bit in the FD status.
- 10: CAPWAP DTLS frame after DTLS encryption: SEC error check, copy QoS field from CAPWAP DTLS header into last byte of HASH and clear this byte in the CAPWAP DTLS header.
- 12: IP frame after SEC encryption: SEC error check, Update UDP length.
- all the rest: reserved for future use - behavior is unpredictable

#### **NOTE**

For FMan\_v3 devices user must not configure to overwrite the parser results in the IC when using OPCODES 3-6.

Configuring DCL4C means that DCL4C is set under the following conditions:

- Fetching only frame header
- Frame is too large to bring into OP (Frames larger than jumbo; i.e. 9216Bytes are considered too large). In this case, full frame fetch is cancelled as well.

In case frame came from SEC and it is of Scatter/Gather format firmware adjust the FD[offset] field to be the same as the offset from the first SG entry so it will support BMI internal context write.

#### **NOTE**

The buffer size where the SG list resides in case the frame that came from SEC is of SG format - must be at least 256 bytes plus the adjusted offset.

SEC error check:

- User shall NOT set the FQD[Context A][ICMD] bit (Ignore Command).
- If one of the 4 msb in the FD[status] is set in case frame came from SEC, then the Fman Controller sets FD[status][9] bit (clear all other bits) and the original FD[status] is saved in the IC.
- Bit FD[status][9] is set as an indication to the application that a SEC error occurred.
- On FMan\_v3, if FD[Length] = 0 the frame is directed to BMI Prepare to Enqueue, otherwise the flow continues to BMI Fetch. Handling of FD[status][9] by the BMI should be configured by the user. (e.g. by configuring the FMBM\_OFSDM or FMBM\_OFSEM registers).

For FMan\_v3 devices SEC HW supports ESP trailer removal and enabling storage profile also solves the FMan S/G compatible issues.

The Pre-BMI Fetch (FrameHdr or FullFrame) NIA must always be used together with the Post-BMI Fetch NIA and can only be used on Offline Ports. The actual execution of the chosen actions are performed either by PreBMIFetch or PostBMIFetch or both.

## 5.12.22 FMan Controller Event Register

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	RxF	BSY								—						
2								—								

Figure 5-505. FPM FMan Controller Event Register

Table 5-539. FPM FMan Controller Event Register Field Descriptions

Bits	Name	Description
0	RxF	Receive frame interrupt event
1	BSY	BD ring busy interrupt event
3-31	—	Reserved. Should be cleared

## 5.12.23 FMan-Controller Parameters Page per Port

Fman Controller parameters page per offline/RX port is valid only for FMan\_v3 devices. User need to allocate 256 bytes for this page and set it in the FMBM\_xGPR register. It should be 8 bytes aligned. This page must be always allocated per offline/RX port for Fman\_v3.

Table 5-540. Parameter Page per Port

Offset	Bits	Name	Description
0x00	0-31	ArCommonDescPtr	Auto Response Common Descriptor Pointer. See <a href="#">Section 5.12.13.4, "Deep Sleep Auto Response Programming Model."</a> This field should be initiated by software when the Auto Response function exists.
0x04-0x0F	—	Reserved	Reserved. Should be cleared.
0x10	0-31	IPv4 IPR Direct Scheme NIA	IPv4 direct scheme NIA is needed for IP reassembly automatic learning process - this scheme builds the Key needed for identifying the reassembled frame context (usually it is 3-tuple + ID for IPv4).
0x14	0-31	IPv6 IPR Direct Scheme NIA	IPv6 direct scheme NIA is needed for IP reassembly automatic learning process - this scheme builds the Key needed for identifying the reassembled frame context (usually it is 2-tuple + ID for IPv6).
0x18-0x2F	—	Reserved	Reserved. Should be cleared.
0x30	0-31	OptCounter	Options Counter. This counter is incremented each time an IPv4 frame with IPv4 Options is encountered by the IP Fragmentation function and the COPIED flag on one of the options is cleared. The counter is incremented at most once per frame.
0x34-0x3F	—	Reserved	Reserved. Should be cleared.

Offset	Bits	Name	Description
0x40	0	VSPDisableOPFixEn	0 - No operation. 1 - This is an Offline Port - Apply fix for the HW bug of not releasing external buffers in case discard decision and VSPE=0. Requires initialization of "DiscardMask" field.
	1	IPaccOffloadingEnable	Firmware need to have indication to go to the correct NIA in case of PHE error check. 0 - IPACC offloading features are disabled. 1 - IPACC offloading features are enabled (IPR/IPF/IPsec/HM/Frame replicator).
	2-22	Reserved	Reserved. Should be cleared.
	23	1	must be set to 1.
	24-25	Reserved	Reserved. Should be cleared.
	26-31	ErrorRSPID	Relative Storage Profile ID for error cases. Requires initialization of 'ErrorPlusDiscardMask'.
0x44	0-31	ErrorPlusDiscardMask	OR'ed mask for both error mask and discard mask (see FMBM_xFSEM and FMBM_xFSMD).
0x48	0-31	DiscardMask	Discard mask (see FMBM_xFSMD). If FDOV is set for this port this mask should be cleared. Must be initialized if IPR is enabled or if 'VSPDIsOPFix' is set.
0x4C	—	Reserved	Reserved. Should be cleared.
0x50	0-31	PostBMIFetchContNIA	Indicates the NIA PostBMIFetch need to use after it completes processing.

### 5.12.24 General FMan-Controller Application Notes

- Clear FMFP\_MXD[DISP\_LIM] to assure that the FPM dispatch limit mechanism is disabled when Any FMan Controller package is enabled.
- The FMan Controller performs parking of certain number of TNUMs according to [Table 5-541](#). This means that the user should consider that the total available TNUMs in the system is the total TNUM number (128) minus the TNUMs that are parked by the FMan Controller.

**Table 5-541. TNUMs Parked by the FMan Controller**

Feature	No. of parked TNUMs	Description
Dynamic update of CC tables	1	Perform SYNC via an external request 0 mechanism. Valid for all FMan devices.
IPR TO	1 + No. of activated TO tasks	IP reassembly TO. Valid for all FMan devices.

- IP Acceleration offloading requires from performance considerations at least 16 TNUMs per Rx/Offline port which may be different than the default recommended values. This requirement may affect the FIFO size allocated for each of these ports since 16 TNUMs means at least  $(2*16+5)*256 = 9.25$  Kbytes. Note that if there are not enough available TNUMs in the system user may consider to allocate less TNUMs per Rx/Offline port according to the nature of the offloading required on this specific port.

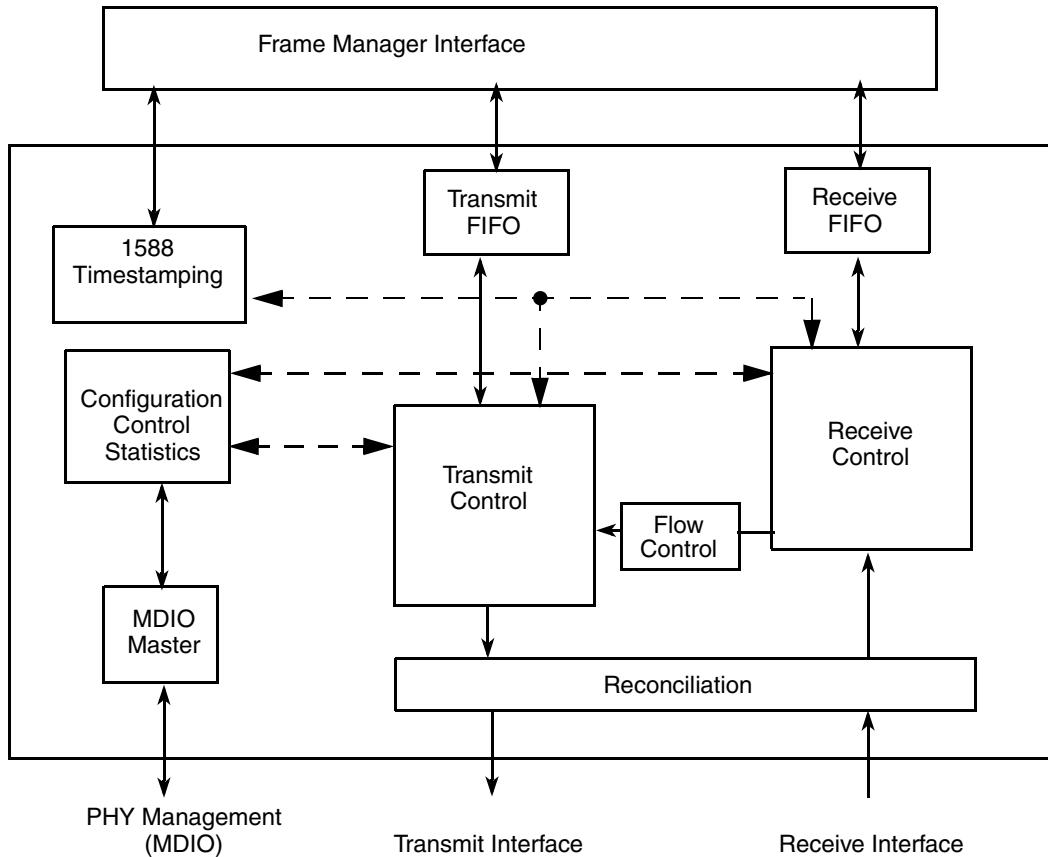
- IP Acceleration offloading requires DMA transaction ordering per port which means setting FMDM\_MR[27]=0 and FMDM\_MR[AID\_ovrd]=0.

# Chapter 6

## Multirate Ethernet Media Access Controller (mEMAC)

### 6.1 mEMAC Overview

The multirate Ethernet media access controller (mEMAC) interfaces to 10Gbps and below Ethernet/IEEE 802.3 networks via either external RGMII interfaces or XFI/SGMII using the high-speed SerDes interfaces. [Figure 6-1](#) shows the mEMAC block diagram.



**Figure 6-1. Ethernet MAC Block Diagram**

### 6.2 mEMAC Features Summary

The Ethernet MAC controller includes the following features:

- Full MAC layer and reconciliation sub-layer implementation compliant with IEEE 802.3ae specification

- EEE (Energy Efficient Ethernet) 10G interface and MII/GMII signaling according to the IEEE802.3az specification
- Lane, data alignment, PHY error and local/remote fault signaling handled by the reconciliation sub-layer
- CRC-32 checking with optional forwarding of the FCS field to the user application
- CRC-32 generation and append on transmit or forwarding of user application provided FCS selectable on a per-frame basis
- 8 MAC address comparison on receive and one MAC address overwrite on transmit for NIC applications.
- Selectable promiscuous frame receive mode and transparent MAC address forwarding on transmit
- Multicast address filtering with 64-bin hash code lookup table on receive reducing processing load on higher layers
- Ethernet pause frame (802.3 Annex 31A) termination providing fully automated flow control without any user application overhead
- Priority Flow Control (PFC) frame support allowing 8 classes for higher layer congestion management
- Magic packet detection
- Programmable frame maximum length providing support for any frame up to 32K (for example, Jumbo frame or any tagged frame)
- Receive detection of VLAN tagged frames according to IEEE 802.1Q and double VLAN Tags (Stacked VLANs)
- Dynamic inter packet gap (IPG) calculation for WAN applications
- Deficit Idle Counter (DIC) for optimized performance with minimum IPG for LAN applications
- Clock and data rate decoupling with programmable asynchronous FIFOs
- 802.3 basic and mandatory managed Objects statistic counters and IETF Management Information Database (MIB) package (RFC2665) and Remote Network Monitoring (RMON) counters
- Programmable Clause 22 and Clause 45 MDIO Master interface for PHY device configuration and management
- Interrupt generation for error and Normal events

The Ethernet MAC controller IEEE 1588 timestamping support includes these features:

- Precise time-stamping of ingress frames and egress frames

## 6.3 mEMAC External Signals Descriptions

This section defines Ethernet MAC external interfaces signals.

**Table 6-1** lists the network interface signals.

**Table 6-1. Ethernet MAC Network Interface Signals**

Signal Name	Function	Reset State
MII_RXDV	MII receive data valid, input	—
CRS	carrier sense, input	—
COL	collision, input	—
MII_RXD[3:0]	MII receive data bits, input	—
MII_RXER	MII receive error, input	—
MII_RXCLK	MII receive clock, input	—
MII_TXD[3:0]	MII transmit data bits, output	0000
MII_TXEN	MII transmit data enabled, output	0
MII_TXER	MII transmit error, output	0
MII_TXCLK	MII transmit clock, input	—
RGMII_CLKREF125	RGMII oscillator source for transmit clock	—
RGMII_TXC	RGMII transmit clock, output	0
RGMII_TXD[3:0]	RGMII (TXC rising) - Transmit data bits 3:0, output RGMII (TXC falling) - Transmit data bits 7:4, output	0000
RGMII_TXCTL	RGMII (TXC rising) - Transmit data enable, output RGMII (TXC falling) - Transmit data error, output	0
RGMII_RXC	RGMII receive clock, input	—
RGMII_RXCTL	RGMII (RXC rising) - Receive data valid, input RGMII (RXC falling) - Receive data error, input	—
RGMII_RXD[3:0]	RGMII (RXC rising) - Receive data bits 3:0, input RGMII (RXC falling) - Receive data bits 7:4, input	—
EMIn_MDC	PHY management clock, output	0
EMIn_MDIO	PHY management data, bidirectional	Hi-Z (input)

### 6.3.1 mEMAC Detailed Signal Descriptions

**Table 6-2** offers detailed signal descriptions of the Ethernet MAC controller interface signals.

**Table 6-2. Ethernet MAC Signals—Detailed Signal Descriptions**

Signal	I/O	Description
MII_RXDV	I	MII receive data valid
CRS	I	PHY asserts CRS when the receive or transmit medium is non-idle. This signal asserts asynchronously with respect to MII clocks.
COL	I	PHY asserts COL upon detection of a collision on the medium. This signal asserts asynchronously with respect to MII clocks.
MII_RXD[3:0]	I	MII receive data. Valid when MII_RXDV is asserted.
MII_RXER	I	MII receive error.

**Table 6-2. Ethernet MAC Signals—Detailed Signal Descriptions (continued)**

<b>Signal</b>	<b>I/O</b>	<b>Description</b>
MII_RXCLK	I	MII receive clock supplied by the PHY.
MII_TXD[3:0]	O	MII transmit data. Valid data is sent to the PHY when MII_TXEN is asserted.
MII_TXEN	O	MII transmit data valid. When asserted, the MAC is indicating that valid data is present on MII_TXD[3:0].
MII_TXER	O	MII transmit coding error.
MII_TXCLK	I	MII transmit clock supplied by the PHY.
RGMII_CLKREF125	I	RGMII transmit 125 MHz source. This signal must be generated externally with a crystal or oscillator, or is sometimes provided by the PHY.
RGMII_TXC	O	RGMII transmit clock. Provides timing reference during 125 MHz, 25 MHz, 2.5 MHz transmissions
RGMII_TXD[3:0]	O	RGMII transmit data. Data bits 3-0 are transmitted on the rising edge of RGMII_TXC, and data bits 7-4 are transmitted on the falling edge of RGMII_TXC
RGMII_TXCTL	O	TX_EN and TX_ERR are transmitted on this signal on rising and falling edges of RGMII_TXC.
RGMII_RXC	I	RGMII receive clock. It is a continuous clock (2.5,25 or 125 MHz) that provides a timing reference for RGMII_RXCTL and RGMII_RXD.
RGMII_RXCTL	I	RX_DV an RX_ERR are received on this signal on the rising and falling edges of RGMII_RXC.
RGMII_RXD[3:0]	I	RGMII receive data. Data bits 3-0 are received on the rising edge of RGMII_RXC, and data bits 7-4 are received on the falling edge of RGMII_RXC
EMIn_MDC	O	Management data clock. This signal is a clock (typically 2.5 MHz) supplied by the MAC (IEEE set minimum period of 400 ns or a frequency of 2.5 MHz, but the device may be configured to higher speed if supported by the PHY.) The frequency can be modified.
EMIn_MDIO	I/O	Management data input/output.
		<b>State Meaning</b> Asserted/Negated—EMIn_MDIO is a bidirectional signal to input PHY-supplied status during management read cycles and output control during MII management write cycles. Addressed using memory-mapped registers.
		<b>Timing</b> Asserted/Negated—This signal is required to be synchronous with the EMIn_MDC signal.

## 6.4 mEMAC Memory Map/Register Definition

The Ethernet MAC device is programmed by control/status registers (CSRs). The CSRs are used for mode control, interrupts, and to extract status information.

Important things to consider when accessing the mEMAC registers are as follows:

- All accesses to and from the registers must be made as 32-bit accesses.
- There is no support for accesses of sizes other than 32 bits.

- Writes to reserved register bits must always preserve the previous value, as changing the value of reserved bits may have unintended side-effects.
- Unless otherwise specified, the read value of unmapped registers or of reserved bits in mapped registers is not defined, and must not be assumed to be 0.

### 6.4.1 mEMAC Top-Level Memory Map

The Ethernet MAC controller and the Ethernet management interface is allocated 2 Kbytes of memory-mapped space, as indicated in [Table 6-3](#). (See [Section 5.4.2, “FMan Detailed Memory Map,”](#) for details of Ethernet MAC controller memory space within FMan memory space.)

**Table 6-3. Ethernet MAC Memory Map Summary**

Address Offset	Function
0x000–0x02C	General control and status registers
0x030–0x03C	MDIO Ethernet management interface registers
0x040–0x0FC	General control and status registers
0x100–0x1FC	Rx Statistics counters registers
0x200–0x2FC	Tx Statistics counters registers
0x300–0x304	Line interface control registers

### 6.4.2 mEMAC Detailed Memory Map

The Ethernet MAC controller memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in CCSRBAR as defined in [Chapter 2, “Memory Map.”](#)) plus the block base address (specified in [Section 5.4.2, “FMan Detailed Memory Map”](#)), plus the offset of the specific register to be accessed. Note that all memory-mapped registers must only be accessed as 32-bit quantities.

[Table 6-4](#) lists the offset, name, and a cross-reference to the complete description of each register. The offsets to the memory map table are defined for the Ethernet MAC controller starting at 0x000 address offset and continues to 0x7FF (2 KB region represented by 512 registers of 4 bytes each).

**Table 6-4. Module Memory Map**

mEMAC Offset	Name	Access	Reset	Section/Page
<b>General Control and Status Registers</b>				
0x008	COMMAND_CONFIG—Control and configuration register	R/W	0x0000_0840	<a href="#">6.4.3.1.1/6-10</a>
0x00C	MAC_ADDR_0—Lower 32 bits of the first 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.2/6-12</a>
0x010	MAC_ADDR_1—Upper 16 bits of the first 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.3/6-13</a>
0x014	MAXFRM—Maximum frame length register	R/W	0x0000_0600	<a href="#">6.4.3.1.4/6-13</a>
0x01C	RX_FIFO_SECTIONS—Receive FIFO configuration register	R/W	Device-specific	<a href="#">6.4.3.1.5/6-14</a>
0x020	TX_FIFO_SECTIONS—Transmit FIFO configuration register	R/W	Device-specific	<a href="#">6.4.3.1.6/6-14</a>
0x02C	HASHTABLE_CTRL—Hash table control register	W	0x0000_0000	<a href="#">6.4.3.1.7/6-15</a>

**Table 6-4. Module Memory Map (continued)**

mEMAC Offset	Name	Access	Reset	Section/Page
0x040	IEVENT—Interrupt event register	R/W	0x0000_0060	<a href="#">6.4.3.1.8/6-16</a>
0x044	TX_LENGTH—Transmitter inter-packet-gap register	R/W	0x0000_000C	<a href="#">6.4.3.1.9/6-17</a>
0x04C	IMASK—Interrupt mask register	R/W	0x0000_0000	<a href="#">6.4.3.1.10/6-18</a>
0x054	CL01_PAUSE_QUANTA - CL0,1 Pause quanta register	R/W	0x0000_0000	<a href="#">6.4.3.1.11/6-19</a>
0x058	CL23_PAUSE_QUANTA - CL2,3 Pause quanta register	R/W	0x0000_0000	<a href="#">6.4.3.1.12/6-20</a>
0x05C	CL45_PAUSE_QUANTA - CL4,5 Pause quanta register	R/W	0x0000_0000	<a href="#">6.4.3.1.13/6-20</a>
0x060	CL67_PAUSE_QUANTA - CL6,7 Pause quanta register	R/W	0x0000_0000	<a href="#">6.4.3.1.14/6-20</a>
0x064	CL01_PAUSE_THRESH - CL0,1 Pause quanta threshold register	R/W	0x0000_0000	<a href="#">6.4.3.1.15/6-21</a>
0x068	CL23_PAUSE_THRESH - CL2,3 Pause quanta threshold register	R/W	0x0000_0000	<a href="#">6.4.3.1.16/6-21</a>
0x06C	CL45_PAUSE_THRESH - CL4,5 Pause quanta threshold register	R/W	0x0000_0000	<a href="#">6.4.3.1.17/6-21</a>
0x070	CL67_PAUSE_THRESH - CL6,7 Pause quanta threshold register	R/W	0x0000_0000	<a href="#">6.4.3.1.18/6-22</a>
0x074	RX_PAUSE_STATUS - Receive pause status register	R	0x0000_0000	<a href="#">6.4.3.1.19/6-22</a>
0x80	MAC_ADDR_2—Lower 32 bits of the 2nd 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.20/6-23</a>
0x84	MAC_ADDR_3—Upper 16 bits of the 2nd 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.21/6-23</a>
0x88	MAC_ADDR_4—Lower 32 bits of the 3rd 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.22/6-23</a>
0x8C	MAC_ADDR_5—Upper 16 bits of the 3rd 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.23/6-24</a>
0x90	MAC_ADDR_6—Lower 32 bits of the 4th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.24/6-24</a>
0x94	MAC_ADDR_7—Upper 16 bits of the 4th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.25/6-24</a>
0x98	MAC_ADDR_8—Lower 32 bits of the 5th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.26/6-25</a>
0x9C	MAC_ADDR_9—Upper 16 bits of the 5th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.27/6-25</a>
0xA0	MAC_ADDR_10—Lower 32 bits of the 6th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.28/6-25</a>
0xA4	MAC_ADDR_11—Upper 16 bits of the 6th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.29/6-26</a>
0xA8	MAC_ADDR_12—Lower 32 bits of the 7th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.30/6-26</a>
0xAC	MAC_ADDR_13—Upper 16 bits of the 7th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.31/6-26</a>
0xB0	MAC_ADDR_14—Lower 32 bits of the 8th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.32/6-27</a>
0xB4	MAC_ADDR_15—Upper 16 bits of the 8th 48-bit MAC address	R/W	0x0000_0000	<a href="#">6.4.3.1.33/6-27</a>
0xB8	LPWAKE_TIMER - EEE Low Power Wakeup Timer register	R/W	0x0000_2000	<a href="#">6.4.3.1.34/6-27</a>
0xBC	SLEEP_TIMER - Transmit EEE Low Power Timer register	R/W	0x0000_0000	<a href="#">6.4.3.1.35/6-28</a>
0x0E0	STATN_CONFIG - Statistics configuration register	R/W	0x0000_0000	<a href="#">6.4.3.1.36/6-28</a>
<b>Rx Statistics Counter Registers</b>				
<b>10G MAC- 64-bit registers: To read, read given address first (LSBs), followed by read to address+4 (MSBs) otherwise - 32-bit registers</b>				
0x100	REOCT_L	R	0x0000_0000	<a href="#">6.4.3.2.1/6-29</a>
0x104	REOCT_U	R	0x0000_0000	<a href="#">6.4.3.2.1/6-29</a>

**Table 6-4. Module Memory Map (continued)**

mEMAC Offset	Name	Access	Reset	Section/Page
0x108	ROCT_L	R	0x0000_0000	<a href="#">6.4.3.2.2/6-29</a>
0x10C	ROCT_U	R	0x0000_0000	<a href="#">6.4.3.2.2/6-29</a>
0x110	RALN_L	R	0x0000_0000	<a href="#">6.4.3.2.3/6-29</a>
0x114	RALN_U	R	0x0000_0000	<a href="#">6.4.3.2.3/6-29</a>
0x118	RXPF_L	R	0x0000_0000	<a href="#">6.4.3.2.4/6-30</a>
0x11C	RXPF_U	R	0x0000_0000	<a href="#">6.4.3.2.4/6-30</a>
0x120	RFRM_L	R	0x0000_0000	<a href="#">6.4.3.2.5/6-30</a>
0x124	RFRM_U	R	0x0000_0000	<a href="#">6.4.3.2.5/6-30</a>
0x128	RFCS_L	R	0x0000_0000	<a href="#">6.4.3.2.6/6-30</a>
0x12C	RFCS_U	R	0x0000_0000	<a href="#">6.4.3.2.6/6-30</a>
0x130	RVLAN_L	R	0x0000_0000	<a href="#">6.4.3.2.7/6-31</a>
0x134	RVLAN_U	R	0x0000_0000	<a href="#">6.4.3.2.7/6-31</a>
0x138	RERR_L	R	0x0000_0000	<a href="#">6.4.3.2.8/6-31</a>
0x13C	RERR_U	R	0x0000_0000	<a href="#">6.4.3.2.8/6-31</a>
0x140	RUCA_L	R	0x0000_0000	<a href="#">6.4.3.2.9/6-31</a>
0x144	RUCA_U	R	0x0000_0000	<a href="#">6.4.3.2.9/6-31</a>
0x148	RMCA_L	R	0x0000_0000	<a href="#">6.4.3.2.10/6-32</a>
0x14C	RMCA_U	R	0x0000_0000	<a href="#">6.4.3.2.10/6-32</a>
0x150	RBCA_L	R	0x0000_0000	<a href="#">6.4.3.2.11/6-32</a>
0x154	RBCA_U	R	0x0000_0000	<a href="#">6.4.3.2.11/6-32</a>
0x158	RDRP_L	R	0x0000_0000	<a href="#">6.4.3.2.12/6-32</a>
0x15C	RDRP_U	R	0x0000_0000	<a href="#">6.4.3.2.12/6-32</a>
0x160	RPKT_L	R	0x0000_0000	<a href="#">6.4.3.2.13/6-33</a>
0x164	RPKT_U	R	0x0000_0000	<a href="#">6.4.3.2.13/6-33</a>
0x168	RUND_L	R	0x0000_0000	<a href="#">6.4.3.2.14/6-33</a>
0x16C	RUND_U	R	0x0000_0000	<a href="#">6.4.3.2.14/6-33</a>
0x170	R64_L	R	0x0000_0000	<a href="#">6.4.3.2.15/6-33</a>
0x174	R64_U	R	0x0000_0000	<a href="#">6.4.3.2.15/6-33</a>
0x178	R127_L	R	0x0000_0000	<a href="#">6.4.3.2.16/6-34</a>
0x17C	R127_U	R	0x0000_0000	<a href="#">6.4.3.2.16/6-34</a>
0x180	R255_L	R	0x0000_0000	<a href="#">6.4.3.2.17/6-34</a>

**Table 6-4. Module Memory Map (continued)**

mEMAC Offset	Name	Access	Reset	Section/Page
0x184	R255_U	R	0x0000_0000	<a href="#">6.4.3.2.17/6-34</a>
0x188	R511_L	R	0x0000_0000	<a href="#">6.4.3.2.18/6-34</a>
0x18C	R511_U	R	0x0000_0000	<a href="#">6.4.3.2.18/6-34</a>
0x190	R1023_L	R	0x0000_0000	<a href="#">6.4.3.2.19/6-35</a>
0x194	R1023_U	R	0x0000_0000	<a href="#">6.4.3.2.19/6-35</a>
0x198	R1518_L	R	0x0000_0000	<a href="#">6.4.3.2.20/6-35</a>
0x19C	R1518_U	R	0x0000_0000	<a href="#">6.4.3.2.20/6-35</a>
0x1A0	R1519X_L	R	0x0000_0000	<a href="#">6.4.3.2.21/6-35</a>
0x1A4	R1519X_U	R	0x0000_0000	<a href="#">6.4.3.2.21/6-35</a>
0x1A8	ROVR_L	R	0x0000_0000	<a href="#">6.4.3.2.22/6-36</a>
0x1AC	ROVR_U	R	0x0000_0000	<a href="#">6.4.3.2.22/6-36</a>
0x1B0	RJBR_L	R	0x0000_0000	<a href="#">6.4.3.2.23/6-36</a>
0x1B4	RJBR_U	R	0x0000_0000	<a href="#">6.4.3.2.23/6-36</a>
0x1B8	RFRG_L	R	0x0000_0000	<a href="#">6.4.3.2.24/6-36</a>
0x1BC	RFRG_U	R	0x0000_0000	<a href="#">6.4.3.2.24/6-36</a>
0x1C0	RCNP_L	R	0x0000_0000	<a href="#">6.4.3.2.25/6-37</a>
0x1C4	RCNP_U	R	0x0000_0000	<a href="#">6.4.3.2.25/6-37</a>
0x1C8	RDRNTP_L	R	0x0000_0000	<a href="#">6.4.3.2.26/6-37</a>
0x1CC	RDRNTP_U	R	0x0000_0000	<a href="#">6.4.3.2.26/6-37</a>

**Tx Statistics Counter Registers**

**10G MAC- 64-bit registers: To read, read given address first (LSBs), followed by read to address+4 (MSBs)  
otherwise - 32-bit registers**

0x200	TEOCT_L	R	0x0000_0000	<a href="#">6.4.3.2.27/6-37</a>
0x204	TEOCT_U	R	0x0000_0000	<a href="#">6.4.3.2.27/6-37</a>
0x208	TOCT_L	R	0x0000_0000	<a href="#">6.4.3.2.28/6-38</a>
0x20C	TOCT_U	R	0x0000_0000	<a href="#">6.4.3.2.28/6-38</a>
0x218	TXPF_L	R	0x0000_0000	<a href="#">6.4.3.2.29/6-38</a>
0x21C	TXPF_U	R	0x0000_0000	<a href="#">6.4.3.2.29/6-38</a>
0x220	TFRM_L	R	0x0000_0000	<a href="#">6.4.3.2.30/6-38</a>
0x224	TFRM_U	R	0x0000_0000	<a href="#">6.4.3.2.30/6-38</a>
0x228	TFCS_L	R	0x0000_0000	<a href="#">6.4.3.2.31/6-39</a>
0x22C	TFCS_U	R	0x0000_0000	<a href="#">6.4.3.2.31/6-39</a>

**Table 6-4. Module Memory Map (continued)**

mEMAC Offset	Name	Access	Reset	Section/Page
0x230	TVLAN_L	R	0x0000_0000	<a href="#">6.4.3.2.32/6-39</a>
0x234	TVLAN_U	R	0x0000_0000	<a href="#">6.4.3.2.32/6-39</a>
0x238	TERR_L	R	0x0000_0000	<a href="#">6.4.3.2.33/6-39</a>
0x23C	TERR_U	R	0x0000_0000	<a href="#">6.4.3.2.33/6-39</a>
0x240	TUCA_L	R	0x0000_0000	<a href="#">6.4.3.2.34/6-40</a>
0x244	TUCA_U	R	0x0000_0000	<a href="#">6.4.3.2.34/6-40</a>
0x248	TMCA_L	R	0x0000_0000	<a href="#">6.4.3.2.35/6-40</a>
0x24C	TMCA_U	R	0x0000_0000	<a href="#">6.4.3.2.35/6-40</a>
0x250	TBCA_L	R	0x0000_0000	<a href="#">6.4.3.2.36/6-41</a>
0x254	TBCA_U	R	0x0000_0000	<a href="#">6.4.3.2.36/6-41</a>
0x260	TPKT_L	R	0x0000_0000	<a href="#">6.4.3.2.37/6-41</a>
0x264	TPKT_U	R	0x0000_0000	<a href="#">6.4.3.2.37/6-41</a>
0x268	TUND_L	R	0x0000_0000	<a href="#">6.4.3.2.38/6-41</a>
0x26C	TUND_U	R	0x0000_0000	<a href="#">6.4.3.2.38/6-41</a>
0x270	T64_L	R	0x0000_0000	<a href="#">6.4.3.2.39/6-42</a>
0x274	T64_U	R	0x0000_0000	<a href="#">6.4.3.2.39/6-42</a>
0x278	T127_L	R	0x0000_0000	<a href="#">6.4.3.2.40/6-42</a>
0x27C	T127_U	R	0x0000_0000	<a href="#">6.4.3.2.40/6-42</a>
0x280	T255_L	R	0x0000_0000	<a href="#">6.4.3.2.41/6-42</a>
0x284	T255_U	R	0x0000_0000	<a href="#">6.4.3.2.41/6-42</a>
0x288	T511_L	R	0x0000_0000	<a href="#">6.4.3.2.42/6-43</a>
0x28C	T511_U	R	0x0000_0000	<a href="#">6.4.3.2.42/6-43</a>
0x290	T1023_L	R	0x0000_0000	<a href="#">6.4.3.2.43/6-43</a>
0x294	T1023_U	R	0x0000_0000	<a href="#">6.4.3.2.43/6-43</a>
0x298	T1518_L	R	0x0000_0000	<a href="#">6.4.3.2.44/6-43</a>
0x29C	T1518_U	R	0x0000_0000	<a href="#">6.4.3.2.44/6-43</a>
0x2A0	T1519X_L	R	0x0000_0000	<a href="#">6.4.3.2.45/6-44</a>
0x2A4	T1519X_U	R	0x0000_0000	<a href="#">6.4.3.2.45/6-44</a>
0x2C0	TCNP_L	R	0x0000_0000	<a href="#">6.4.3.2.46/6-44</a>
0x2C4	TCNP_U	R	0x0000_0000	<a href="#">6.4.3.2.46/6-44</a>
<b>Line Interface Control Registers</b>				
0x300	IF_MODE—Interface Mode Control Register	R/W	0x0000_000x	<a href="#">6.4.3.3.1/6-44</a>

**Table 6-4. Module Memory Map (continued)**

mEMAC Offset	Name	Access	Reset	Section/Page
0x304	IF_STATUS—Interface Status Register	R	0x0000_0000	<a href="#">6.4.3.3.2/6-45</a>
<b>Ethernet Management Interface Registers</b> (see <a href="#">Table 5-17</a> for base addresses for the MDIO registers)				
0x030	MDIO_CFG—Management interface configuration register	R/W	0x0000_1448	<a href="#">6.4.3.4.1/6-46</a>
0x034	MDIO_CTRL—Management interface control register	R/W	0x0000_0000	<a href="#">6.4.3.4.2/6-48</a>
0x038	MDIO_DATA—Management interface data register	R/W	0x0000_0000	<a href="#">6.4.3.4.3/6-49</a>
0x03C	MDIO_ADDR—Management interface register address register	W	0x0000_0000	<a href="#">6.4.3.4.4/6-49</a>

## 6.4.3 mEMAC Memory-Mapped Register Descriptions

This section provides a detailed description of all the Ethernet MAC controller registers. Because all of the Ethernet MAC controller registers are 32 bits wide, only 32-bit register accesses are supported.

### 6.4.3.1 mEMAC General Control and Status Registers

This section describes general control and status registers used for both transmitting and receiving Ethernet frames. All of the registers are 32 bits wide.

#### 6.4.3.1.1 Command and Configuration Register (COMMAND\_CONFIG)

Offset 0x008																Access: Read/Write				
R W	0 M G	1	2	3	4	5	6	7	REG_LOWP _RXETY	TX_LO WP_E NA	8	9	10	11	12	13	14	15	SEND _IDLE	
Reset																All zeros				
R W	—	CNT_FRM _EN	SWR	TXP	XGLP	TX_ADDR _INS	PAUSE_IGN	PAUSE _FWD	CRC	PAD	PROMIS	WAN	—	RX_EN	TX_EN	28	29	30	31	
Reset																0	0	0	0	0

**Figure 6-2. Command and Configuration Register (COMMAND\_CONFIG)****Table 6-5. COMMAND\_CONFIG Field Descriptions**

Bits	Name	Description
0	MG	Magic Packet detection enable. 0 - normal operation or Magic packet mode has exited with reception of a valid Magic Packet 1 - all received frames are searched for the Magic Packet pattern. The MAC is placed in discard mode, hence no frames will be received by the application any more. However, statistics will continue normally.
1	—	Reserved

**Table 6-5. COMMAND\_CONFIG Field Descriptions (continued)**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
2	RXSTP	Rx stop 0 - normal operation 1 - The MAC is placed in discard mode, hence no new frames will be received by the application any more. However, statistics will continue normally
3–6	—	Reserved
7	REG_LOWP_RXETY	0 - (default) Rx low power indication is asserted to PCS as soon as the RS layer detects the Low Power Idle sequence from the line. 1- Rx low power indication will assert to PCS only when the RX FIFO is empty. That is, the assertion of the low power indication is delayed until all data has been retrieved by the application.
8	TX_LOWP_ENA	Transmit Low Power Idle Enable. 0 - (default), the MAC operates in normal mode. 1 - the MAC completes the transmission of the current Frame and generates Low Power Idle Sequences to the line. It is advised to inspect IEVENT[TX_EMPTY] is set before enabling the LPI.
9	—	Reserved
10	SFD	Disable check of SFD (0xd5) character at frame start. 1 - the frame is accepted even if the SFD byte following the preamble is not 0xd5. 0 - (default) a frame is accepted only if the SFD byte is found with value 0xd5. Note: Relevant only in 10G mode. In all other modes the SFD character must be 0xd5 and this setting has no effect.
11	—	Reserved
12	PFC_MODE	Enable Priority Flow Control (PFC) mode of operation. 0 - (default) the MAC uses standard Link Pause frames 1 - the MAC will transmit and accept PFC frames.
13	—	Reserved
14	NO_LEN_CHK	Payload length check disable 0 MAC compares the frame payload length with the frame Length/Type field. 1 Payload length check is disabled.
15	SEND_IDLE	Suppresses any frame transmissions and forces IDLE on the transmit interface instead of frames. Note: does not have an effect on fault handling (i.e. reception of local fault will still cause transmit of remote fault). Must be 0 for normal operation
16–17	—	Reserved
18	CNT_FRM_EN	Control frame reception enable 0 Only Pause frames are accepted (all other control frames are rejected). 1 All control frames are accepted.
19	SWR	Software Reset. Self clearing bit. 1 - resets all statistic counters as well as the transmit and receive FIFOs. It should be issued after all traffic has been stopped as a result of clearing the rx/tx enable bits.
20	TXP	Enable padding of frames in transmit direction (1, default). 0 - the MAC will not extend frames from the application to a minimum of 64 bytes, allowing to transmit too short frames (violating the Ethernet minimum size requirement). Must be 1 for normal operation
21	XGLP	10G interface /GMII loopback enable 0 Configured for normal (non-loopback) operation. 1 Configured for 10G interface /GMII internal loopback mode - also MAC is isolated from PCS

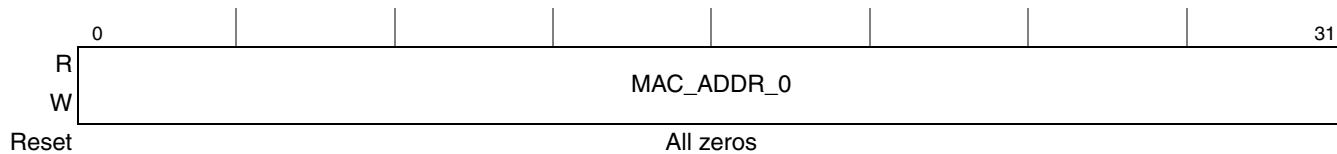
**Table 6-5. COMMAND\_CONFIG Field Descriptions (continued)**

Bits	Name	Description
22	TX_ADDR_INS	Transmit source MAC address insertion 0 MAC transmits the source MAC address unmodified from user application. 1 MAC overwrites the source MAC address with the MAC programmed address.
23	PAUSE_IGN	Ignore Pause frame quanta 0 MAC stops transmit process for the duration specified in the Pause frame quanta of a received Pause frame. 1 MAC ignores received Pause frames.
24	PAUSE_FWD	Terminate/forward received Pause frames 0 MAC terminates and discards received Pause frames. 1 MAC forwards Pause frames to the user application.
25	CRC	Terminate/forward CRC of received frames (This bit is only applicable if COMMAND_CONFIG[PAD_EN] is clear.) 0 MAC strips CRC from received frames. 1 MAC forwards CRC of received frames to the user application.
26	PAD	Frame padding removal in receive path enable 0 MAC does not remove padding prior to forwarding frames to the user application. 1 MAC removes padding prior to forwarding frames to the user application.
27	PROMIS	Promiscuous operation enable 0 Unicast frames with a destination address not matching the MAC addresses (defined by registers, MAC_ADDR_0/1 till MAC_ADDR_14/15) are rejected. 1 All frames are received without any MAC address filtering.
28	WAN	WAN mode enable (When changing the MAC mode, verify the correct setting of the transmit inter-packet-gap in TX_LENGTH[LEN].) 0 Configure MAC for LAN mode. 1 Configure MAC for WAN mode.
29	—	Reserved
30	RX_EN	MAC receive path enable 0 MAC receive path is disabled 1 MAC receive path is enabled.
31	TX_EN	MAC transmit path enable 0 MAC transmit path is disabled 1 MAC transmit path is enabled.

**6.4.3.1.2 First MAC Lower Address Register (MAC\_ADDR\_0)**

Offset 0x00C

Access: Read/Write

**Figure 6-3. First MAC Lower Address Register (MAC\_ADDR\_0)**

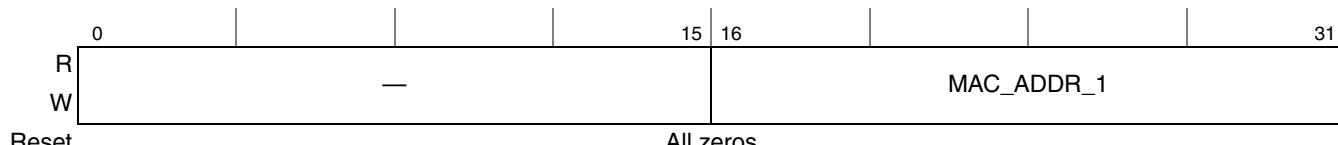
**Table 6-6. MAC\_ADDR\_0 Field Descriptions**

Bits	Name	Description
0–31	MAC_ADDR_0	The lower 32 bits of the first 48-bit MAC address. (MAC_ADDR_0[31] = LSB of the MAC address.)

**6.4.3.1.3 First MAC Upper Address Register (MAC\_ADDR\_1)**

Offset 0x010

Access: Read/Write



Reset

All zeros

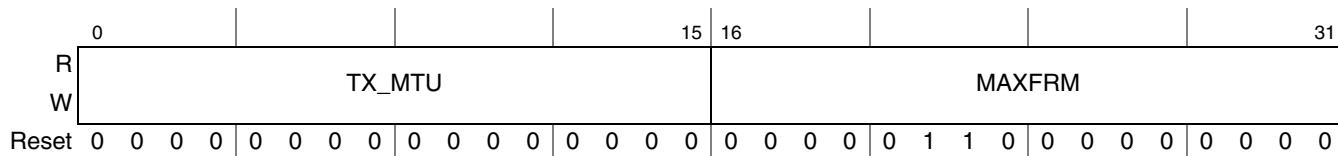
**Figure 6-4. First MAC Upper Address Register (MAC\_ADDR\_1)****Table 6-7. MAC\_ADDR\_1 Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	MAC_ADDR_1	The upper 16 bits of the first 48-bit MAC address. (MAC_ADDR_1[31] = MAC address bit 32, and MAC_ADDR_1[16] = MAC address bit 47.)

**6.4.3.1.4 Maximum Frame Length Register (MAXFRM)**

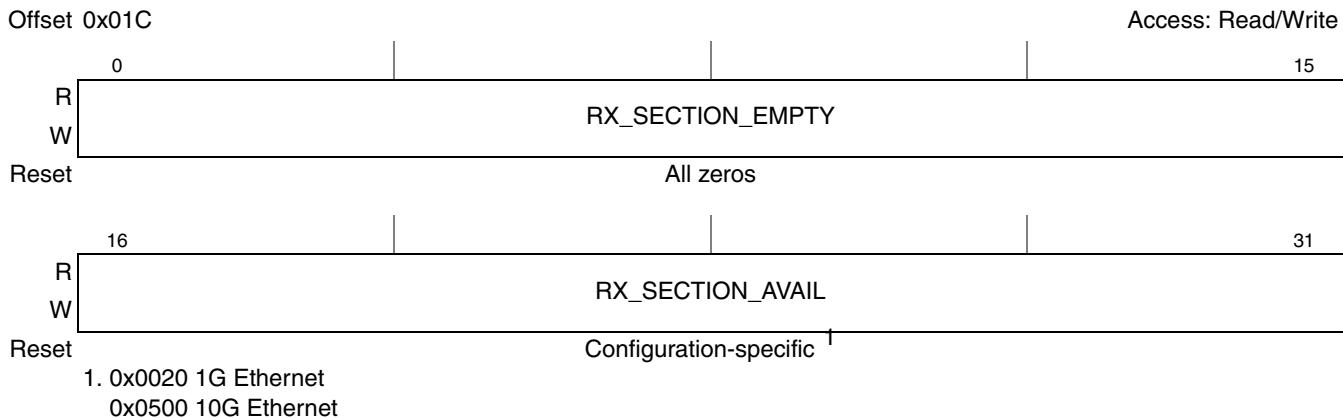
Offset 0x014

Access: Read/Write

**Figure 6-5. Maximum Frame Length Register (MAXFRM)****Table 6-8. MAXFRM Field Descriptions**

Bits	Name	Description
0–15	TX_MTU	Allows to set a different maximum length on transmit. It is used for statistics purposes only and has no effect on the MAC transmit operation. When set to 0 MAXFRM is used instead.
16–31	MAXFRM	Maximum supported received frame length. The MAC supports reception of any frame size up to 32,736 bytes (0x7FE0). Typical settings are 0x05EE (1,518 bytes) for standard frames. Default setting is 0x0600 (1,536 bytes). Received frames that exceed this stated maximum are truncated.(A truncated frame which is anyway corrupted can be in the range of (MAXFRM+4) to (MAXFRM-7). When not using crc forwarding a truncated frame can be in the range of MAXFRM to (MAXFRM-7)).

#### 6.4.3.1.5 Receive FIFO Sections Register (RX\_FIFO\_SECTIONS)

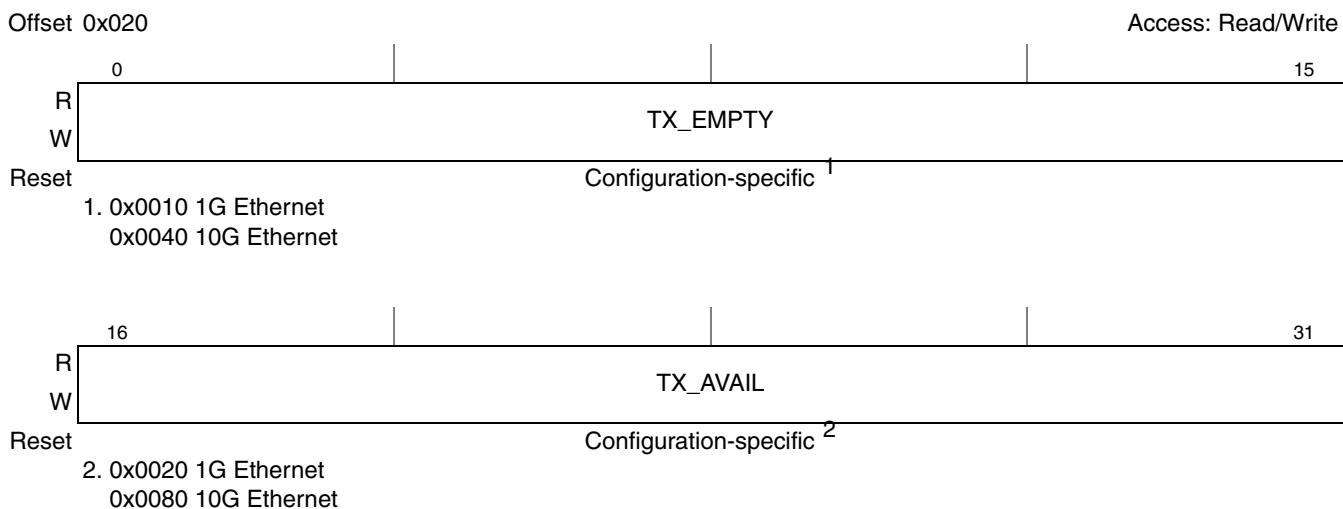


**Figure 6-6. RX\_FIFO\_SECTIONS Register Definition**

**Table 6-9. RX\_FIFO\_SECTIONS Field Descriptions**

Bits	Name	Description
0–15	RX_SECTION_EMPTY	RX section empty threshold. Value is in steps of 8-byte FIFO words. When FIFO level rises above this mark and PFC_MODE=0, pause XOFF packet is transmitted. When FIFO level goes below this mark and PFC_MODE=0, pause XON packet is transmitted. A value of 0s disables this function.
16–31	RX_SECTION_AVAIL	RX section available threshold. Value is in steps of 8-byte FIFO words. Note - this value should not be modified.

#### 6.4.3.1.6 Transmit FIFO Sections Register (TX\_FIFO\_SECTIONS)



**Figure 6-7. TX\_FIFO\_SECTIONS Register Definition**

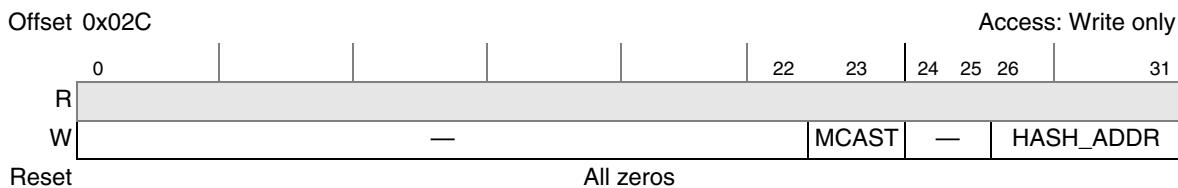
**Table 6-10. TX\_FIFO\_SECTIONS Field Descriptions**

Bits	Name	Description
0–15	TX_EMPTY	<p>TX section empty threshold. Value is in steps of 8-byte FIFO words.</p> <p>Note - When PFC_MODE=0 this value depends on MAC TX FIFO size: 0x0010 or 0x0040.</p> <p>When PFC_MODE=1 program: for 10G mode - 0x0036, for 2.5G mode - 0x0010, for 1G mode - 0x0004.</p>
16–31	TX_AVAIL	<p>TX section available threshold. Value is in steps of 8-byte FIFO words.</p> <p>When FIFO level reaches this mark, MAC starts frame transmission (unless complete frame is in FIFO).</p> <p>For 1G rate TX_AVAIL should be programmed to 0x0020.</p> <p>For 2.5G rate TX_AVAIL should be programmed to 0x0020.</p> <p>For 10G rate MAC TX_AVAIL should be programmed to 0x0019 (25d).</p>

#### 6.4.3.1.7 Hash Table Control Register (HASHTABLE\_CTRL)

The HASHTABLE\_CTRL is used to create the hash table which controls frame acceptance.

Hash table entries are initialized to 0 after reset, so hash table should be programmed.



**Figure 6-8. HASHTABLE\_CTRL Register Definition**

**Table 6-11. HASHTABLE\_CTRL Field Descriptions**

Bits	Name	Description
0–22	—	Reserved
23	MCAST_EN	Multicast frame acceptance for the specified hash entry. 0 Reject - Received frames with a DA that produces the corresponding hash code are not accepted. 1 Accept - Received frames with a DA that produces the corresponding hash code are accepted.
24–25	—	Reserved
26–31	HASH_ADDR	Hash table address code. Hash code is defined as $\{\text{^DA[47:40]}, \text{^DA[39:32]}, \text{^DA[31:24]}, \text{^DA[23:16]}, \text{^DA[15:8]}, \text{^DA[7:0]}\}$ Any frame whose DA produces any of the accepted 6-bit hash codes specified in the hash table will be accepted. ( $\wedge$ denotes XOR).

### 6.4.3.1.8 Interrupt Event Register (IEVENT)

The IEVENT reflects the occurrence of various errors or functional events. All bits are cleared by writing '1' except for bits 25-27 which are status bits.

												Access: Mixed					
Offset 0x040															15		
R	0	1	2													—	
	PCS	AN	LT													—	
W	w1c	w1c	w1c													—	
Reset	All zeros																
R	—	MGI	TS_ECC_ER	RX_OVFL	TX_UNFL	TX_OVFL	TX_ECC_ER	RX_ECC_ER	LI_FAULT	RX_EMPTY	TX_EMPTY	RX_LOWP	—	REM_FAULT	LOC_FAULT		
W	—	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	—	w1c	w1c		
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0		

Figure 6-9. Interrupt Event Register (IEVENT)

Table 6-12. IEVENT Field Descriptions

Bits	Name	Description
0	PCS	10G interface - PCS event (see MDIO section in the SerDes chapter of the SoC Reference Manual) GMII - link synchronization event (see MDIO section in the SerDes chapter of the SoC Reference Manual)
1	AN	10G interface - Auto-negotiation event (see MDIO section in the SerDes chapter of the SoC Reference Manual) GMII - Auto-negotiation status (see MDIO section in the SerDes chapter of the SoC Reference Manual)
2	LT	10G interface - Link Training event (see MDIO section in the SerDes chapter of the SoC Reference Manual) GMII - new page received by auto-negotiation function (see MDIO section in the SerDes chapter of the SoC Reference Manual)
3–16	—	Reserved
17	MGI	Magic packet detection indication event 0 Magic packet was not detected. 1 Magic packet was detected when COMMAND_CONFIG[MG] was set. COMMAND_CONFIG[MG] is cleared upon magic packet detection.
18	TS_ECC_ER	Timestamp FIFO ECC error event 0 A frame was not received with timestamp FIFO ECC error. 1 A frame was received with timestamp FIFO ECC error.
19	RX_FIFO_OVFL	Receive FIFO overflow event. 0 The receive FIFO has not overflowed. 1 The receive FIFO overflowed.
20	TX_FIFO_UNFL	Transmit FIFO underflow event. 0 The transmit FIFO has not underflowed. 1 The transmit FIFO underflowed.

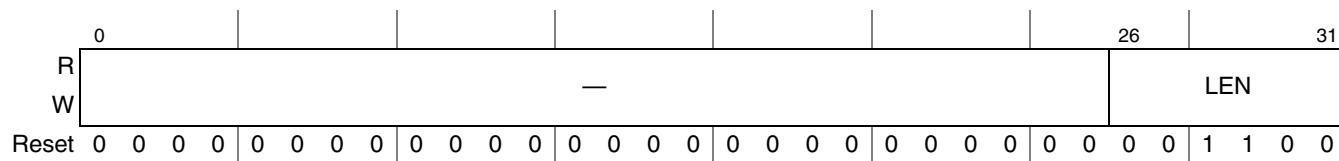
**Table 6-12. IEVENT Field Descriptions (continued)**

Bits	Name	Description
21	TX_FIFO_OVFL	Transmit FIFO overflow event. 0 The transmit FIFO has not overflowed. 1 The transmit FIFO overflowed.
22	TX_ECC_ER	Transmit frame ECC error event. 0 A frame was not transmitted with an ECC memory error. 1 A frame was transmitted with an ECC memory error.
23	RX_ECC_ER	Receive frame ECC error event. 0 A frame was not received with an ECC memory error. 1 A frame was received with an ECC memory error.
24	LIFAULT	Link Interruption fault event (10G interface) 0 A Link Interruption fault condition did not occur. 1 A Link Interruption fault condition occurred
25	RX_EMPTY	Receive fifo empty event 0 An empty receive fifo condition did not occur. 1 An empty receive fifo condition occurred
26	TX_EMPTY	Transmit fifo empty event 0 An empty transmit fifo condition did not occur. 1 An empty transmit fifo condition occurred
27	RXLWP	Low Power Idle event 0 A Low Power Idle condition did not occur. 1 A Low Power Idle condition occurred.
28–29	—	Reserved
30	REM_FAULT	Remote fault event (10G interface) 0 A remote fault condition did not occur. 1 A remote fault condition occurred.
31	LOC_FAULT	Local fault event (10G interface) 0 A local fault condition did not occur. 1 A local fault condition occurred.

**6.4.3.1.9 Transmit Inter-Packet Gap Length Register (TX\_LENGTH)**

Offset 0x044

Access: Read/Write

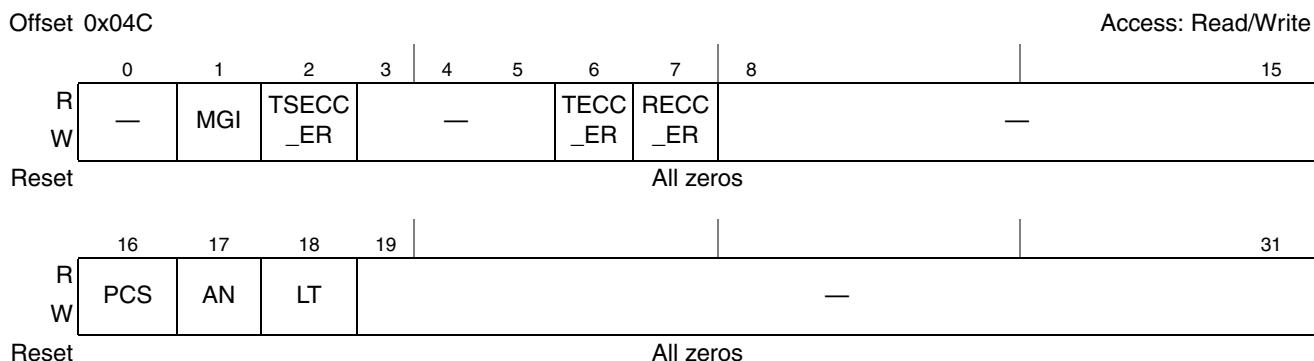
**Figure 6-10. Transmit Inter-Packet Gap Length Register (TX\_LENGTH)**

**Table 6-13. TX\_LENGTH Field Descriptions**

Bits	Name	Description
0–25	—	Reserved
26–31	LEN	<p>Transmit inter-packet gap value.</p> <p>If configured for LAN mode (per COMMAND_CONFIG[WAN_MODE]):</p> <p>LEN represents the number of octets in steps of 4. Valid values for 10G interface are 8, 12, 16, ... 100. DIC is fully supported for any setting &gt; 8. The default of 0xC (12 octets) is required to conform to IEEE 802.3ae. Valid values for GMII are 12 ...30.</p> <p>If configured for WAN mode (per COMMAND_CONFIG[WAN_MODE]):</p> <p>LEN represents the stretch factor. Valid values are 4–15. The stretch factor is calculated as (value+1)*8. A default of 0xC (12 stretch factor) is required to conform to IEEE802.3ae (that is, 13*8=104). A larger value shrinks the IPG (increasing bandwidth).</p>

#### **6.4.3.1.10 Interrupt Mask Register (IMASK)**

The IMASK is used to mask the generation of an interrupt for several errors or functional events reflected on IEVENT register. Error interrupt is generated by bits 2,6-7. Normal interrupt is generated by bits 1,16-18.



**Figure 6-11. Interrupt Mask Register (IMASK)**

**Table 6-14. IMASK Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0	—	Reserved
1	MGI	Magic packet detection indication normal interrupt mask 0 masked 1 enabled
2	TSECC_ER	Timestamp FIFO ECC error interrupt mask 0 masked 1 enabled
3–5	—	Reserved
6	TECC_ER	Transmit frame ECC error interrupt mask. 0 masked 1 enabled

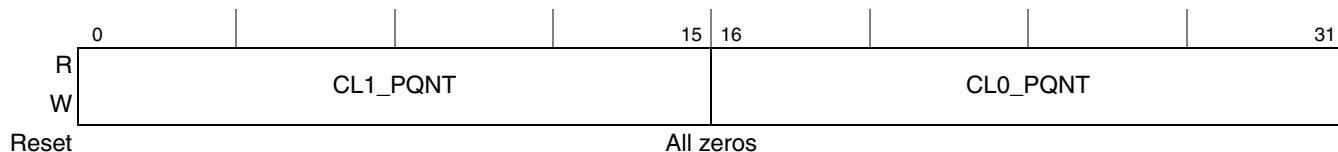
**Table 6-14. IMASK Field Descriptions (continued)**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
7	RECC_ER	Receive frame ECC error interrupt mask. 0 masked 1 enabled
8–15	—	Reserved
16	PCS	10G Interface - PCS event interrupt mask (see MDIO section in the SerDes chapter of the SoC Reference Manual) GMII - link synchronization event interrupt mask (see MDIO section in the SerDes chapter of the SoC Reference Manual) 0 masked 1 enabled
17	AN	10G Interface - Auto-negotiation event interrupt mask (see MDIO section in the SerDes chapter of the SoC Reference Manual) GMII - Auto-negotiation status interrupt mask (see MDIO section in the SerDes chapter of the SoC Reference Manual) 0 masked 1 enabled
18	LT	10G Interface - Link Training event interrupt mask (see MDIO section in the SerDes chapter of the SoC Reference Manual) GMII - new page received by auto-negotiation function interrupt mask (see MDIO section in the SerDes chapter of the SoC Reference Manual) 0 masked 1 enabled
19–31	—	Reserved

#### 6.4.3.1.11 CL01 Pause Quanta Register (CL01\_PAUSE\_QUANTA)

Offset 0x054

Access: Read/Write



Reset

**Figure 6-12. CL01\_PAUSE\_QUANTA Register****Table 6-15. CL01\_PAUSE\_QUANTA Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–15	CL1_PQNT	Value to be sent for the PFC quanta value for that class when a class XOFF is triggered.
16–31	CL0_PQNT	Value to be sent for the PFC quanta value for that class when a class XOFF is triggered. When normal pause mode is enabled, only CL0_PAUSE_QUANTA is used.

#### 6.4.3.1.12 CL23 Pause Quanta Register (CL23\_PAUSE\_QUANTA)

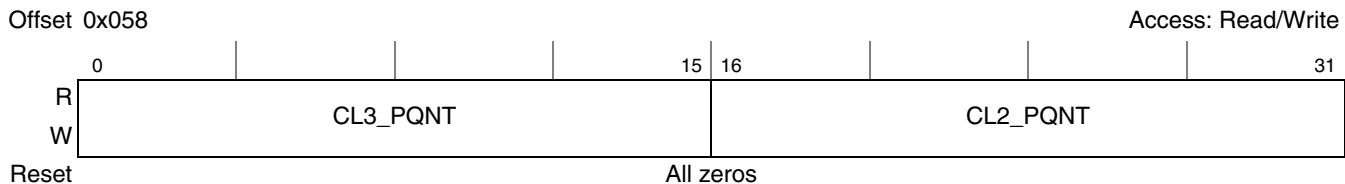


Figure 6-13. CL23\_PAUSE\_QUANTA Register

Table 6-16. CL23\_PAUSE\_QUANTA Field Descriptions

Bits	Name	Description
0–15	CL3_PQNT	Value to be sent for the PFC quanta value for that class when a class XOFF is triggered.
16–31	CL2_PQNT	Value to be sent for the PFC quanta value for that class when a class XOFF is triggered.

#### 6.4.3.1.13 CL45 Pause Quanta Register (CL45\_PAUSE\_QUANTA)

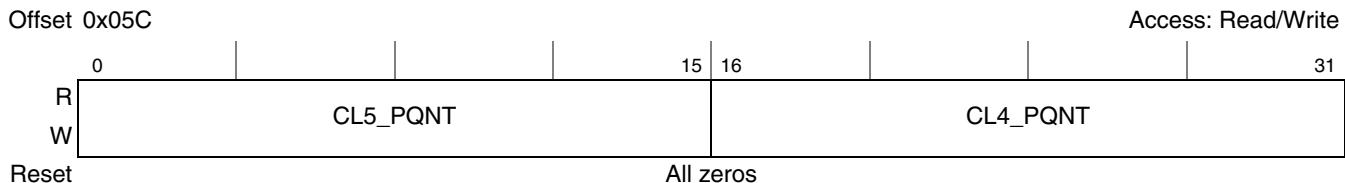


Figure 6-14. CL45\_PAUSE\_QUANTA Register

Table 6-17. CL45\_PAUSE\_QUANTA Field Descriptions

Bits	Name	Description
0–15	CL5_PQNT	Value to be sent for the PFC quanta value for that class when a class XOFF is triggered.
16–31	CL4_PQNT	Value to be sent for the PFC quanta value for that class when a class XOFF is triggered.

#### 6.4.3.1.14 CL67 Pause Quanta Register (CL67\_PAUSE\_QUANTA)

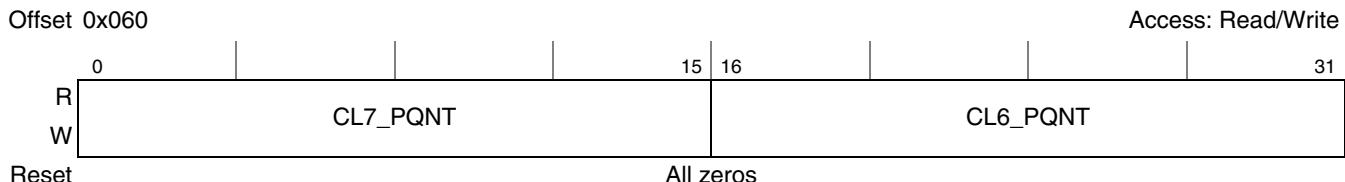


Figure 6-15. CL67\_PAUSE\_QUANTA Register

Table 6-18. CL67\_PAUSE\_QUANTA Field Descriptions

Bits	Name	Description
0–15	CL7_PQNT	Value to be sent for the PFC quanta value for that class when a class XOFF is triggered.
16–31	CL6_PQNT	Value to be sent for the PFC quanta value for that class when a class XOFF is triggered.

#### 6.4.3.1.15 CL01 Pause Quanta Threshold Register (CL01\_PAUSE\_THRESH)

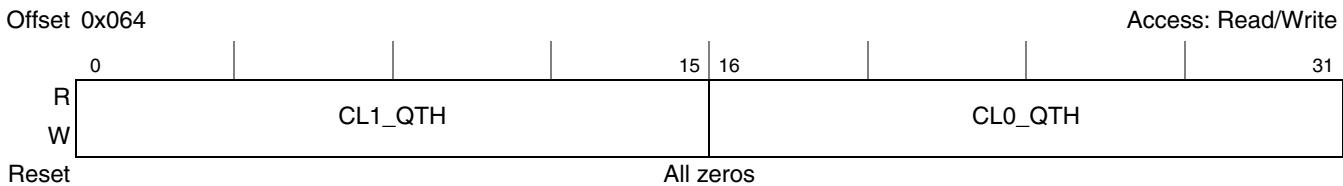


Figure 6-16. CL01\_PAUSE\_THRESH Register

Table 6-19. CL01\_PAUSE\_THRESH Field Descriptions

Bits	Name	Description
0–15	CL1_QTH	When a PFC quanta timer counts down and reaches this value, a refresh pause frame should be sent with the programmed full quanta value if a pause condition still exists. Value should be greater or equal to 2 and less than CL1_PQNT.
16–31	CL0_QTH	When a PFC quanta timer counts down and reaches this value, a refresh pause frame should be sent with the programmed full quanta value if a pause condition still exists. When normal pause mode is enabled, CL0_QTH is used for refreshing pause frames. Value should be greater or equal to 2 and less than CL0_PQNT.

#### 6.4.3.1.16 CL23 Pause Quanta Threshold Register (CL23\_PAUSE\_THRESH)

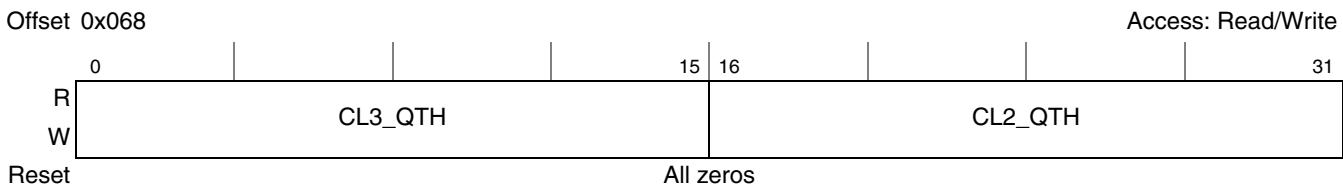


Figure 6-17. CL23\_PAUSE\_THRESH Register

Table 6-20. CL23\_PAUSE\_THRESH Field Descriptions

Bits	Name	Description
0–15	CL3_QTH	When a PFC quanta timer counts down and reaches this value, a refresh pause frame should be sent with the programmed full quanta value if a pause condition still exists. Value should be greater or equal to 2 and less than CL3_PQNT.
16–31	CL2_QTH	When a PFC quanta timer counts down and reaches this value, a refresh pause frame should be sent with the programmed full quanta value if a pause condition still exists. Value should be greater or equal to 2 and less than CL2_PQNT.

#### 6.4.3.1.17 CL45 Pause Quanta Threshold Register (CL45\_PAUSE\_THRESH)

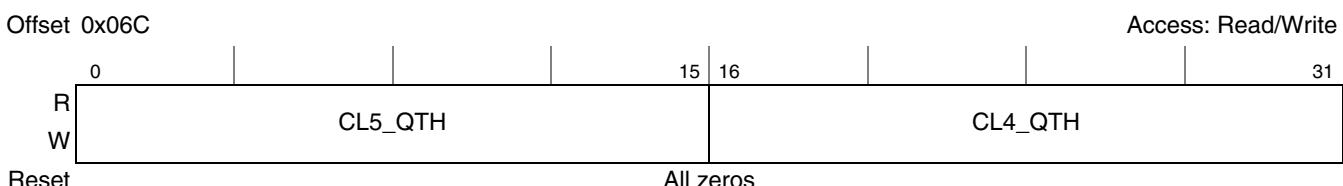


Figure 6-18. CL45\_PAUSE\_THRESH Register

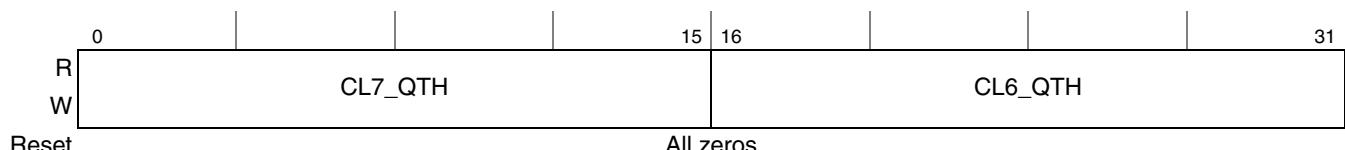
**Table 6-21. CL45\_PAUSE\_THRESH Field Descriptions**

Bits	Name	Description
0–15	CL5_QTH	When a PFC quanta timer counts down and reaches this value, a refresh pause frame should be sent with the programmed full quanta value if a pause condition still exists. Value should be greater or equal to 2 and less than CL5_PQNT.
16–31	CL4_QTH	When a PFC quanta timer counts down and reaches this value, a refresh pause frame should be sent with the programmed full quanta value if a pause condition still exists. Value should be greater or equal to 2 and less than CL4_PQNT.

**6.4.3.1.18 CL67 Pause Quanta Threshold Register (CL67\_PAUSE\_THRESH)**

Offset 0x070

Access: Read/Write

**Figure 6-19. CL67\_PAUSE\_THRESH Register****Table 6-22. CL67\_PAUSE\_THRESH Field Descriptions**

Bits	Name	Description
0–15	C7_QTH	When a PFC quanta timer counts down and reaches this value, a refresh pause frame should be sent with the programmed full quanta value if a pause condition still exists. Value should be greater or equal to 2 and less than CL7_PQNT.
16–31	CL6_QTH	When a PFC quanta timer counts down and reaches this value, a refresh pause frame should be sent with the programmed full quanta value if a pause condition still exists. Value should be greater or equal to 2 and less than CL6_PQNT.

**6.4.3.1.19 Receive Pause Status Register (RX\_PAUSE\_STATUS)**

Offset 0x074

Access: Read Only

**Figure 6-20. RX\_PAUSE\_STATUS Register****Table 6-23. RX\_PAUSE\_STATUS Field Descriptions**

Bits	Name	Description
0–23	—	Reserved
24–31	PSTAT	Pause statuses. One bit for each of the 8 classes : {7,6,5,4,3,2,1,0}.

#### 6.4.3.1.20 2nd MAC Lower Address Register (MAC\_ADDR\_2)

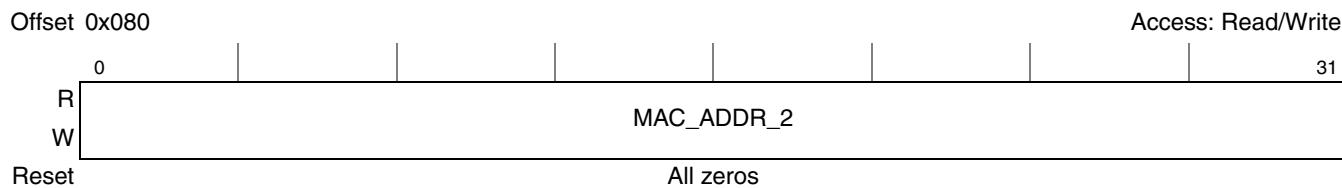


Figure 6-21. 2nd MAC Lower Address Register (MAC\_ADDR\_2)

Table 6-24. MAC\_ADDR\_2 Field Descriptions

Bits	Name	Description
0–31	MAC_ADDR_2	The lower 32 bits of the 2nd 48-bit MAC address. (MAC_ADDR_0[31] = LSB of the MAC address.)

#### 6.4.3.1.21 2nd MAC Upper Address Register (MAC\_ADDR\_3)

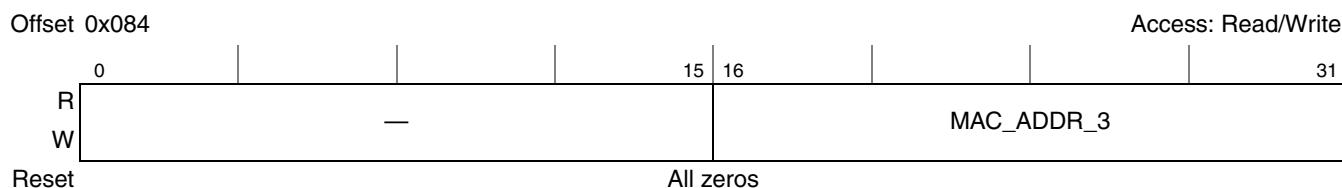


Figure 6-22. 2nd MAC Upper Address Register (MAC\_ADDR\_3)

Table 6-25. MAC\_ADDR\_3 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	MAC_ADDR_3	The upper 16 bits of the 2nd 48-bit MAC address. (MAC_ADDR_1[31] = MAC address bit 32, and MAC_ADDR_1[16] = MAC address bit 47.)

#### 6.4.3.1.22 3rd MAC Lower Address Register (MAC\_ADDR\_4)

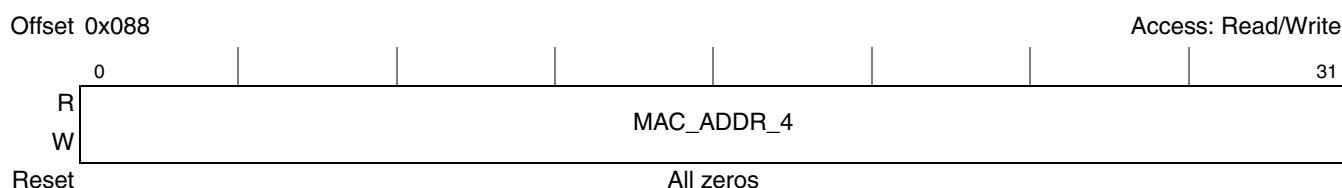


Figure 6-23. 3rd MAC Lower Address Register (MAC\_ADDR\_4)

Table 6-26. MAC\_ADDR\_4 Field Descriptions

Bits	Name	Description
0–31	MAC_ADDR_4	The lower 32 bits of the 3rd 48-bit MAC address. (MAC_ADDR_0[31] = LSB of the MAC address.)

#### 6.4.3.1.23 3rd MAC Upper Address Register (MAC\_ADDR\_5)

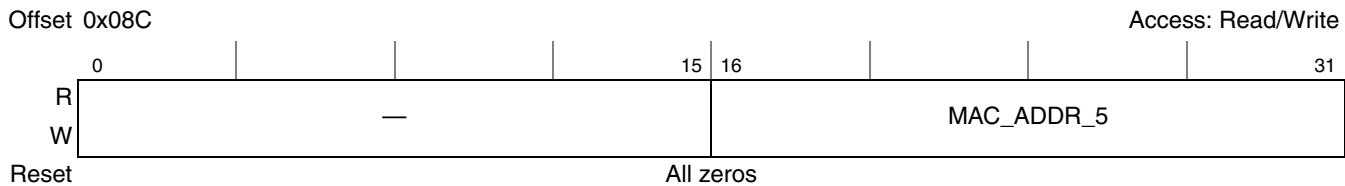


Figure 6-24. 3rd MAC Upper Address Register (MAC\_ADDR\_5)

Table 6-27. MAC\_ADDR\_5 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	MAC_ADDR_5	The upper 16 bits of the 3rd 48-bit MAC address. (MAC_ADDR_1[31] = MAC address bit 32, and MAC_ADDR_1[16] = MAC address bit 47.)

#### 6.4.3.1.24 4th MAC Lower Address Register (MAC\_ADDR\_6)

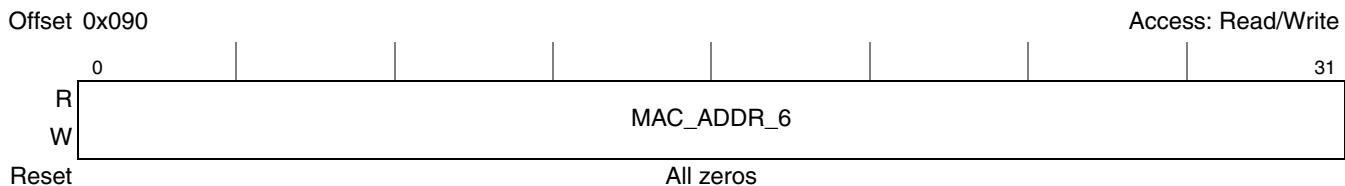


Figure 6-25. 4th MAC Lower Address Register (MAC\_ADDR\_6)

Table 6-28. MAC\_ADDR\_6 Field Descriptions

Bits	Name	Description
0–31	MAC_ADDR_6	The lower 32 bits of the 4th 48-bit MAC address. (MAC_ADDR_0[31] = LSB of the MAC address.)

#### 6.4.3.1.25 4th MAC Upper Address Register (MAC\_ADDR\_7)

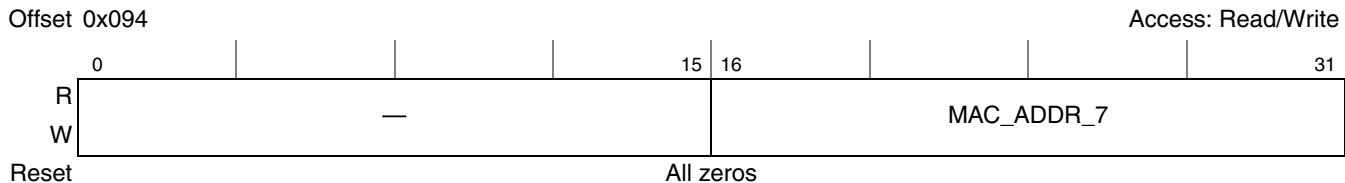


Figure 6-26. 4th MAC Upper Address Register (MAC\_ADDR\_7)

Table 6-29. MAC\_ADDR\_7 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	MAC_ADDR_7	The upper 16 bits of the 4th 48-bit MAC address. (MAC_ADDR_1[31] = MAC address bit 32, and MAC_ADDR_1[16] = MAC address bit 47.)

#### 6.4.3.1.26 5th MAC Lower Address Register (MAC\_ADDR\_8)

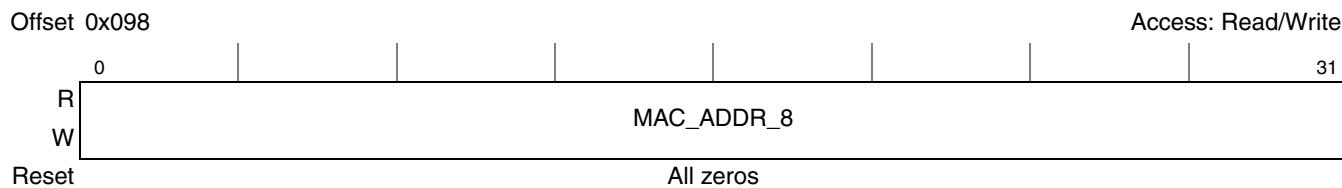


Figure 6-27. 5th MAC Lower Address Register (MAC\_ADDR\_8)

Table 6-30. MAC\_ADDR\_8 Field Descriptions

Bits	Name	Description
0-31	MAC_ADDR_8	The lower 32 bits of the 5th 48-bit MAC address. (MAC_ADDR_0[31] = LSB of the MAC address.)

#### 6.4.3.1.27 5th MAC Upper Address Register (MAC\_ADDR\_9)

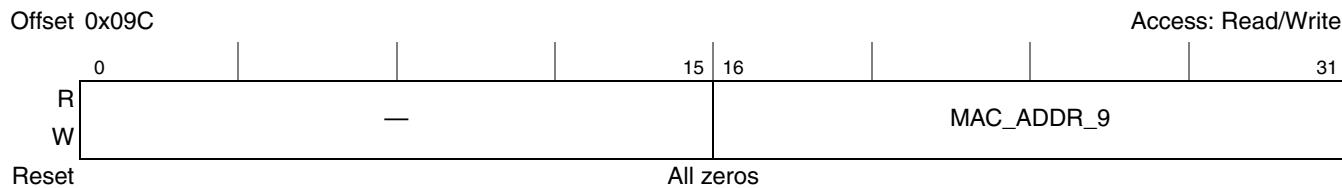


Figure 6-28. 5th MAC Upper Address Register (MAC\_ADDR\_9)

Table 6-31. MAC\_ADDR\_9 Field Descriptions

Bits	Name	Description
0-15	—	Reserved
16-31	MAC_ADDR_9	The upper 16 bits of the 5th 48-bit MAC address. (MAC_ADDR_1[31] = MAC address bit 32, and MAC_ADDR_1[16] = MAC address bit 47.)

#### 6.4.3.1.28 6th MAC Lower Address Register (MAC\_ADDR\_10)

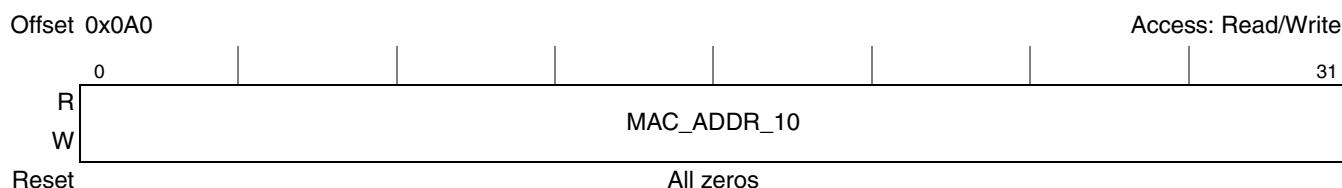


Figure 6-29. 6th MAC Lower Address Register (MAC\_ADDR\_10)

Table 6-32. MAC\_ADDR\_10 Field Descriptions

Bits	Name	Description
0-31	MAC_ADDR_10	The lower 32 bits of the 6th 48-bit MAC address. (MAC_ADDR_0[31] = LSB of the MAC address.)

#### 6.4.3.1.29 6th MAC Upper Address Register (MAC\_ADDR\_11)

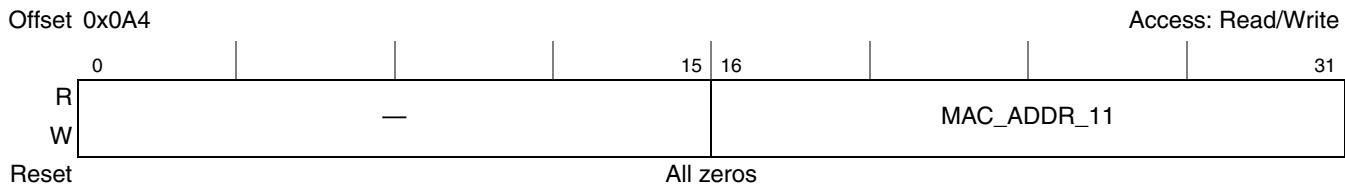


Figure 6-30. 6th MAC Upper Address Register (MAC\_ADDR\_11)

Table 6-33. MAC\_ADDR\_11 Field Descriptions

Bits	Name	Description
0-15	—	Reserved
16-31	MAC_ADDR_11	The upper 16 bits of the 6th 48-bit MAC address. (MAC_ADDR_1[31] = MAC address bit 32, and MAC_ADDR_1[16] = MAC address bit 47.)

#### 6.4.3.1.30 7th MAC Lower Address Register (MAC\_ADDR\_12)

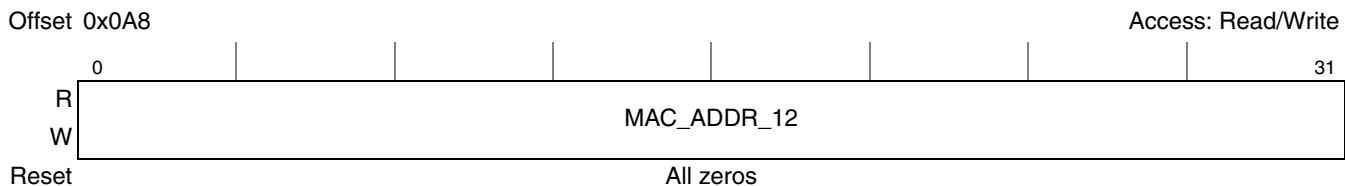


Figure 6-31. 7th MAC Lower Address Register (MAC\_ADDR\_12)

Table 6-34. MAC\_ADDR\_12 Field Descriptions

Bits	Name	Description
0-31	MAC_ADDR_12	The lower 32 bits of the 7th 48-bit MAC address. (MAC_ADDR_0[31] = LSB of the MAC address.)

#### 6.4.3.1.31 7th MAC Upper Address Register (MAC\_ADDR\_13)

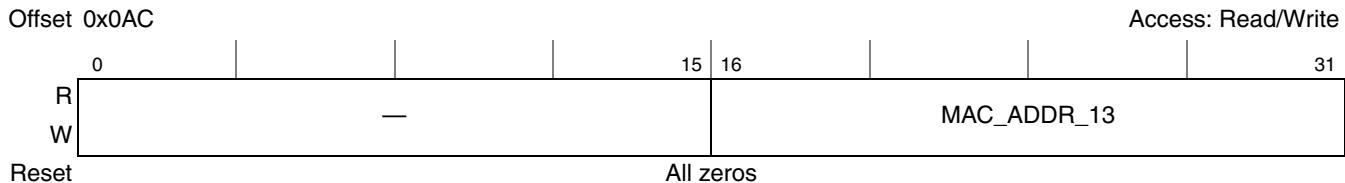


Figure 6-32. 7th MAC Upper Address Register (MAC\_ADDR\_13)

Table 6-35. MAC\_ADDR\_13 Field Descriptions

Bits	Name	Description
0-15	—	Reserved
16-31	MAC_ADDR_13	The upper 16 bits of the 7th 48-bit MAC address. (MAC_ADDR_1[31] = MAC address bit 32, and MAC_ADDR_1[16] = MAC address bit 47.)

#### 6.4.3.1.32 8th MAC Lower Address Register (MAC\_ADDR\_14)



Figure 6-33. 8th MAC Lower Address Register (MAC\_ADDR\_14)

Table 6-36. MAC\_ADDR\_14 Field Descriptions

Bits	Name	Description
0-31	MAC_ADDR_14	The lower 32 bits of the 8th 48-bit MAC address. (MAC_ADDR_0[31] = LSB of the MAC address.)

#### 6.4.3.1.33 8th MAC Upper Address Register (MAC\_ADDR\_15)

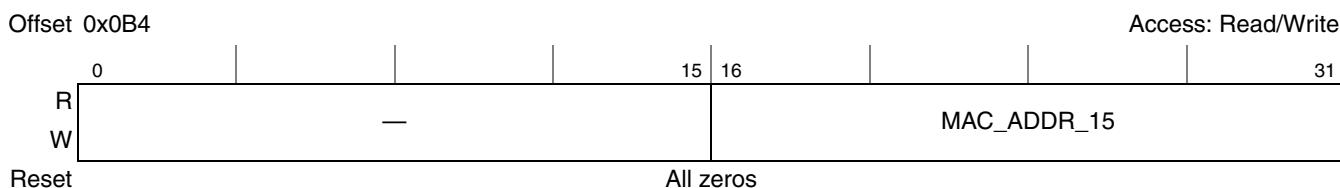


Figure 6-34. 8th MAC Upper Address Register (MAC\_ADDR\_15)

Table 6-37. MAC\_ADDR\_15 Field Descriptions

Bits	Name	Description
0-15	—	Reserved
16-31	MAC_ADDR_15	The upper 16 bits of the 8th 48-bit MAC address. (MAC_ADDR_1[31] = MAC address bit 32, and MAC_ADDR_1[16] = MAC address bit 47.)

#### 6.4.3.1.34 EEE Low Power Wakeup Timer Register (LPWAKE\_TIMER)

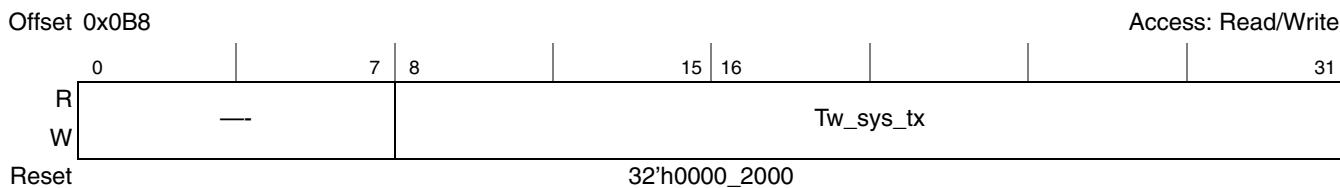


Figure 6-35. EEE Low Power Wakeup Timer Register (LPWAKE\_TIMER)

Table 6-38. LPWAKE\_TIMER Field Descriptions

Bits	Name	Description
0-7	—	Reserved
8-31	Tw_sys_tx	Defines the number of FMan clock cycles (which represents time) required by the PHY to wait before transmitting a new frame after the application has indicated it wants to end the low power state.

#### 6.4.3.1.35 Transmit EEE Low Power Timer Register (SLEEP\_TIMER)



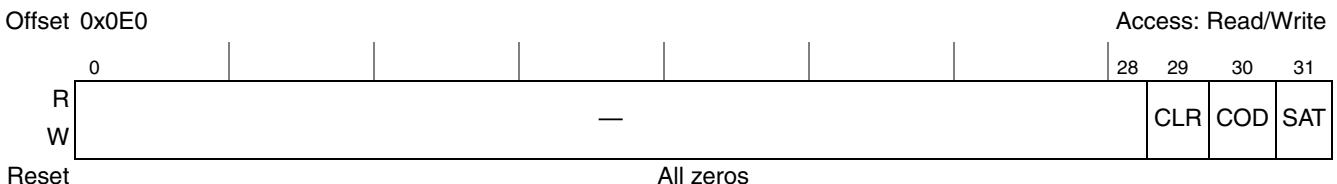
**Figure 6-36. Transmit EEE Low Power Timer Register (SLEEP\_TIMER)**

**Table 6-39. SLEEP\_TIMER Field Descriptions**

Bits	Name	Description
0-7	—	Reserved
8-31	SLEEP_T	Defines the number of FMan clock cycles (which represents time) where Tx is idle before MAC transmits low power EEE. A value of 0 does not activate low power EEE transmission.

#### 6.4.3.1.36 Statistics Configuration Register (STATN\_CONFIG)

Used to configure features of the statistics module.



**Figure 6-37. STATN\_CONFIG Register**

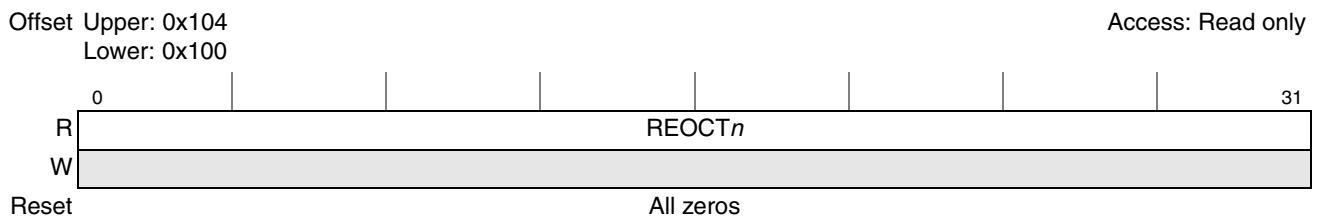
**Table 6-40. STATN\_CONFIG Field Descriptions**

Bits	Name	Description
0-28	—	Reserved
29	CLR	1 - all counters will be reset to 0. Note that this takes at least 32 FMan clock cycles. Self resetting bit, reads 0 always. The soft-reset from command config will also trigger this counter clear function.
30	CLR_ON_RD	0 - (default) counters are not affected by read. 1 - a read to a counter resets it to 0.
31	SATURATE	0 - (default) counters are wrapping around 1- the counters saturate at the maximum value. Typically used in combination with CLR_ON_RD.

#### 6.4.3.2 mEMAC Statistics Counter Registers

The Ethernet MAC controller offers statistics counters, which can be used to implement the statistics required in IEEE802.3 basic, mandatory and recommended Management information packages [2, clause 30]. In addition it can be used to implement the applicable objects of the Management Information Base (MIB,MIB-II) according to IETF RFC2665 (including update to 10Gb/s). For monitoring applications the RMON counters are available according to IETF RFC2819.

#### **6.4.3.2.1 Receive Ethernet Octets Counter (REOCT $n$ )**

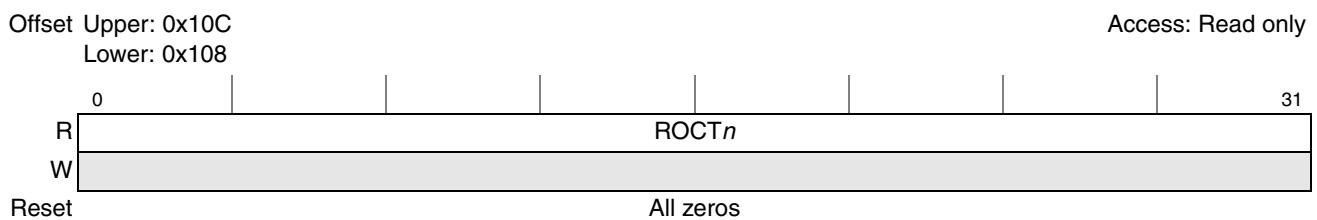


**Figure 6-38. Receive Ethernet Octets Counter (REOCT $n$ )**

**Table 6-41. REOCT*n* Field Description**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	REOCT <i>n</i>	Incremented for each octet received in both good and bad packets.

#### **6.4.3.2.2 Receive Octets Counter Register (ROCT $n$ )**

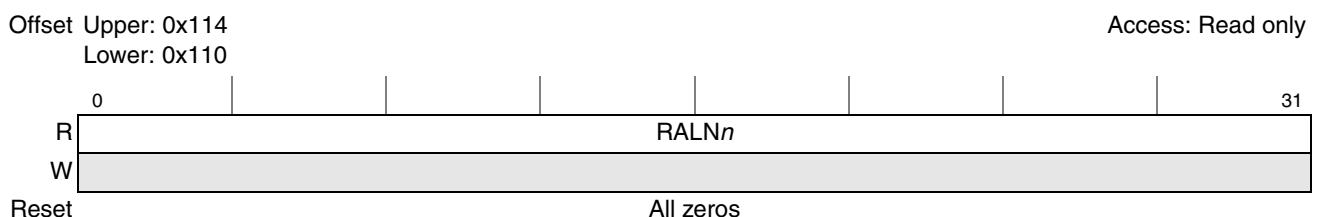


**Figure 6-39. Receive Octets Counter Register (ROCTn)**

**Table 6-42. ROCTn Field Description**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0-31	ROCT $n$	Incremented for each octet received except preamble (that is, Header, Payload, Pad and FCS) for all valid frames and valid Pause frames received.

#### 6.4.3.2.3 Receive Alignment Error Counter Register (RALN $n$ )

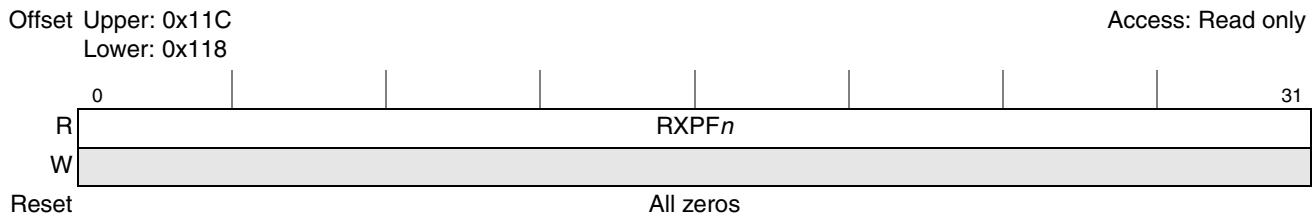


**Figure 6-40. Receive Alignment Error Counter Register (RALN $n$ )**

**Table 6-43. RALN<sub>n</sub> Field Description**

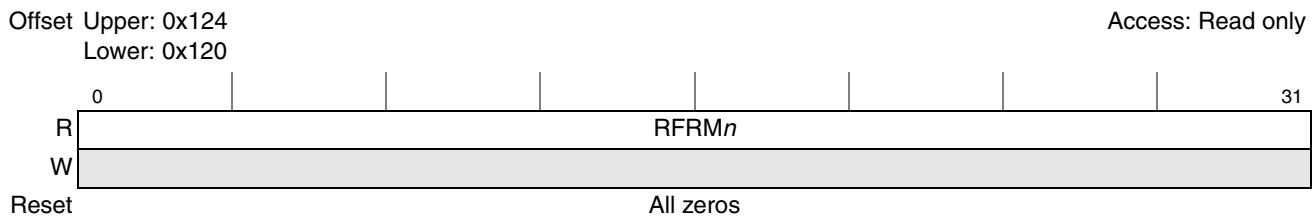
<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	RALN $n$	Incremented for each frame received with an alignment error (optional used for wrong SFD)

#### 6.4.3.2.4 Receive Valid Pause Frame Counter Register (RXPF $n$ )

Figure 6-41. Receive Valid Pause Frame Counter Register (RXPF $n$ )Table 6-44. RXPF $n$  Field Description

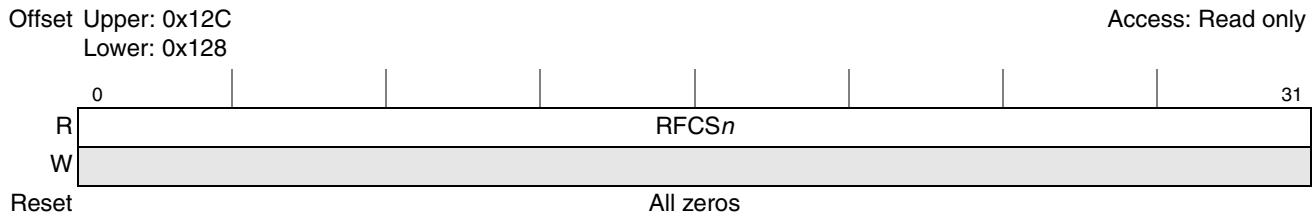
Bits	Name	Description
0–31	RXPF $n$	Incremented for each valid Pause frame received (regular and PFC).

#### 6.4.3.2.5 Receive Frame Counter Register (RFRM $n$ )

Figure 6-42. Receive Frame Counter Register (RFRM $n$ )Table 6-45. RFRM $n$  Field Description

Bits	Name	Description
0–31	RFRM $n$	Incremented for each frame received without error, including Pause frames.

#### 6.4.3.2.6 Receive Frame Check Sequence Error Counter Register (RFCSe $n$ )

Figure 6-43. Receive Frame Check Sequence Error Counter Register (RFCSe $n$ )Table 6-46. RFCSe $n$  Field Description

Bits	Name	Description
0–31	RFCSe $n$	Incremented for each frame received with a CRC-32 error but the frame is otherwise of correct length.

#### 6.4.3.2.7 Receive VLAN Frame Counter Register (RVLAn)

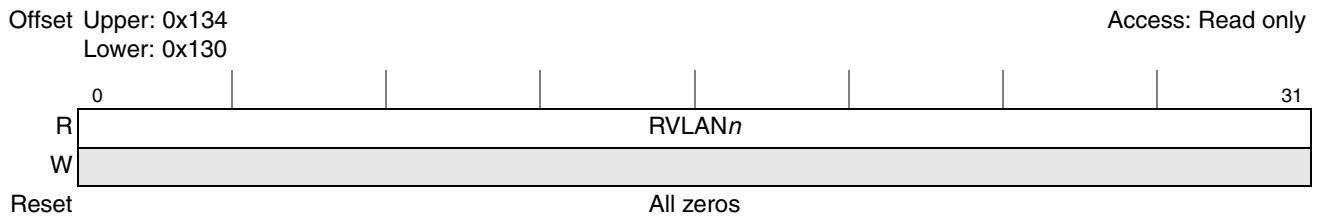


Figure 6-44. Receive VLAN Frame Counter Register (RVLAn)

Table 6-47. RVLAn Field Description

Bits	Name	Description
0–31	RVLAn	Incremented for each valid VLAN tagged frame received.

#### 6.4.3.2.8 Receive Frame Error Counter Register (RERRn)

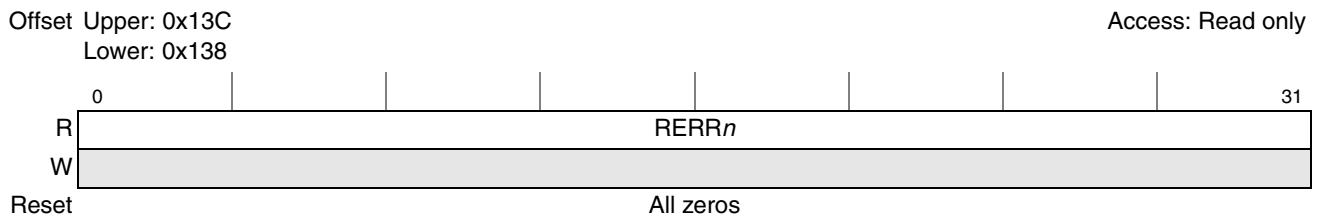


Figure 6-45. Receive Frame Error Counter Register (RERRn)

Table 6-48. RERRn Field Description

Bits	Name	Description
0–31	RERRn	Incremented for each frame received with an error (except for undersized/fragment frame) : <ul style="list-style-type: none"> <li>• FIFO overflow error</li> <li>• CRC error</li> <li>• Payload length error</li> <li>• Jabber and oversized error</li> <li>• Alignment error (if supported)</li> <li>• Reception of PHY/PCS error indication (0xFE, not a code error)</li> </ul>

#### 6.4.3.2.9 Receive Unicast Frame Counter Register (RUCAn)

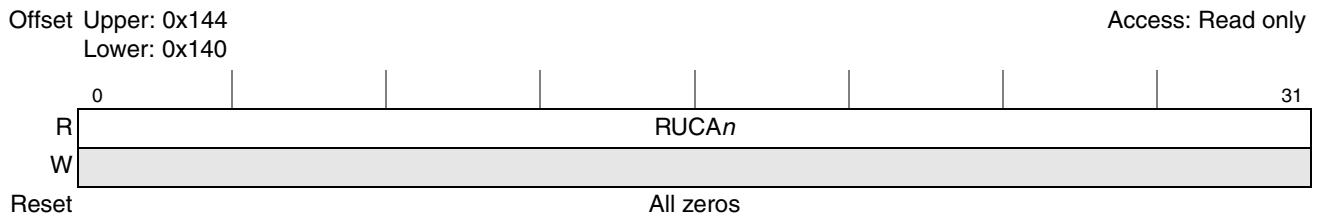


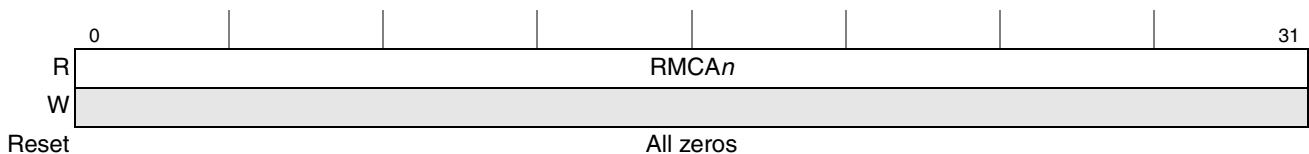
Figure 6-46. Receive Unicast Frame Counter Register (RUCAn)

**Table 6-49. RUCAn Field Description**

Bits	Name	Description
0–31	RUCA $n$	Incremented for each valid frame received in which bit 0 of the destination address was “0”.

#### **6.4.3.2.10 Receive Multicast Frame Counter Register (RMCA $n$ )**

Offset Upper: 0x14C Access: Read only  
Lower: 0x148



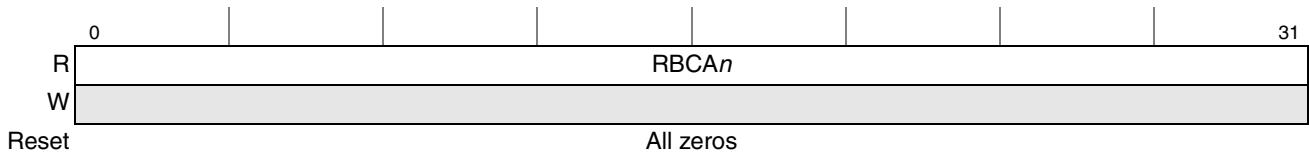
**Figure 6-47. Receive Multicast Frame Counter Register (RMCA $n$ )**

**Table 6-50. RMCA*n* Field Description**

Bits	Name	Description
0-31	RMCA $n$	Incremented for each valid frame received in which bit 0 of the destination address was “1” but not the broadcast address (all bits set to “1”). This count excludes Pause frames.

#### **6.4.3.2.11 Receive Broadcast Frame Counter Register (RBCAn)**

Offset Upper: 0x154 Access: Read only  
Lower: 0x150



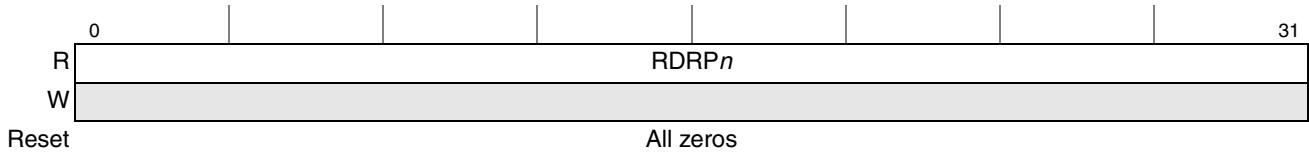
**Figure 6-48. Receive Broadcast Frame Counter Register (RBCAn)**

**Table 6-51. RBCAn Field Description**

Bits	Name	Description
0–31	RBCAn	Incremented for each valid frame received in which all bits of the destination address were “1”.

#### **6.4.3.2.12 Receive Dropped Packets Counter Register (RDRPn)**

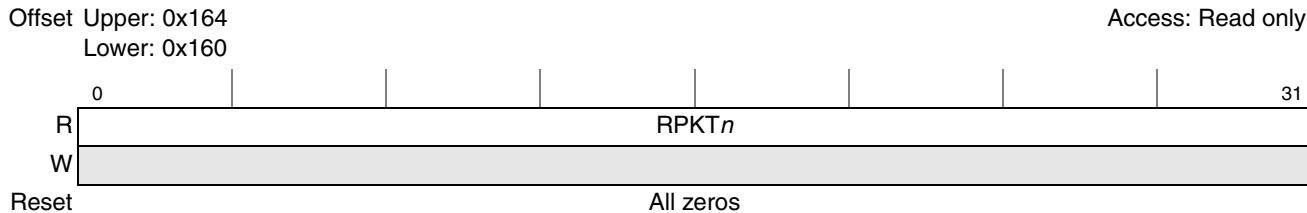
Offset Upper: 0x15C Access: Read only  
Lower: 0x158



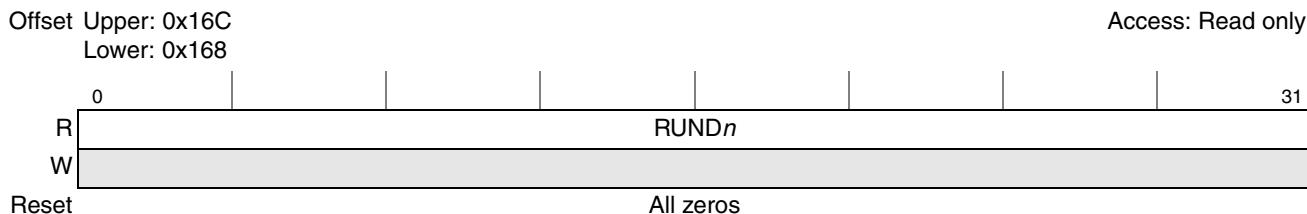
**Figure 6-49. Receive Dropped Packets Counter Register (RDRPn)**

**Table 6-52. RDRP $n$  Field Description**

Bits	Name	Description
0–31	RDRP $n$	Incremented for each dropped packet due to internal errors. Occurs when a receive FIFO overflows. Includes also packets truncated as a result of the receive FIFO overflow.

**6.4.3.2.13 Receive Packets Counter Register (RPKT $n$ )****Figure 6-50. Receive Packets Counter Register (RPKT $n$ )****Table 6-53. RPKT $n$  Field Description**

Bits	Name	Description
0–31	RPKT $n$	Incremented for each good or bad packet received.

**6.4.3.2.14 Receive Undersized Packet Counter Register (RUND $n$ )****Figure 6-51. Receive Undersized Packet Counter Register (RUND $n$ )****Table 6-54. RUND $n$  Field Description**

Bits	Name	Description
0–31	RUND $n$	Incremented for each packet received that was less than 64 octets long with a good CRC.

**6.4.3.2.15 Receive 64-Octet Packet Counter Register (R64 $n$ )****Figure 6-52. Receive 64-Octet Packet Counter Register (R64 $n$ )**

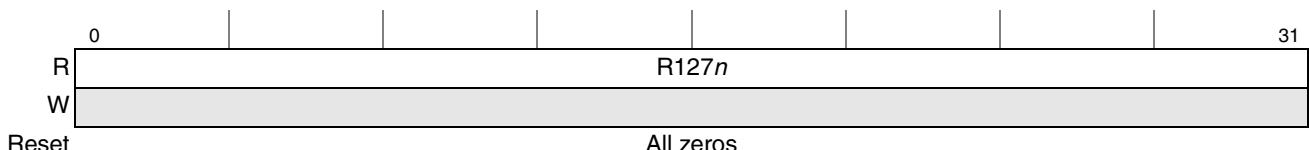
**Table 6-55. R64n Field Description**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	R64n	Incremented for each 64-octet frame received, good or bad.

#### **6.4.3.2.16 Receive 65- to 127-Octet Packet Counter Register (R127n)**

Offset Upper: 0x17C  
Lower: 0x178

Access: Read only



**Figure 6-53. Receive 65- to 127-Octet Packet Counter Register (R127n)**

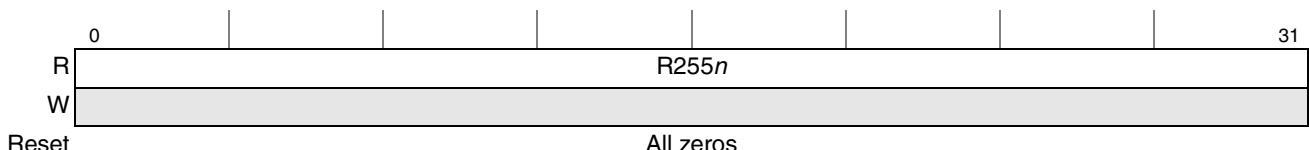
**Table 6-56. R127n Field Description**

Bits	Name	Description
0–31	R127n	Incremented for each 65- to 127-octet frame received, good or bad.

#### **6.4.3.2.17 Receive 128- to 255-Octet Packet Counter Register (R255n)**

Offset Upper: 0x184  
Lower: 0x180

Access: Read only



**Figure 6-54. Receive 128- to 255-Octet Packet Counter Register (R255n)**

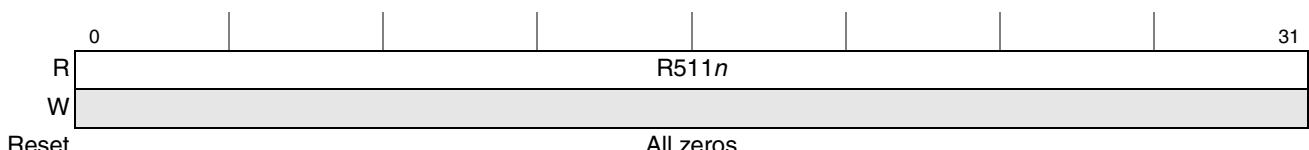
**Table 6-57. R255*n* Field Description**

Bits	Name	Description
0–31	R255 <i>n</i>	Incremented for each 128- to 255-octet frame received, good or bad.

#### **6.4.3.2.18 Receive 256- to 511-Octet Packet Counter Register (R511n)**

Offset Upper: 0x18C  
Lower: 0x188

Access: Read only



**Figure 6-55. Receive 256- to 511-Octet Packet Counter Register (R511n)**

**Table 6-58. R511n Field Description**

Bits	Name	Description
0–31	R511 <i>n</i>	Incremented for each 256- to 511-octet frame received, good or bad.

#### **6.4.3.2.19 Receive 512- to 1023-Octet Packet Counter Register (R1023n)**

Offset Upper: 0x194

Access: Read only

The diagram shows a memory location  $R1023n$ . A vertical line labeled "0" at its bottom end points to a horizontal bar representing memory. The memory bar is divided into two sections: a white section labeled "R1023n" and a gray section labeled "All zeros". The total length of the memory bar is labeled "31" at its right end.

**Figure 6-56. Receive 512- to 1023-Octet Packet Counter Register (R1023n)**

**Table 6-59. R1023*n* Field Description**

Bits	Name	Description
0–31	R1023 <i>n</i>	Incremented for each 512- to 1023-octet frame received, good or bad.

#### **6.4.3.2.20 Receive 1024- to 1518-Octet Packet Counter Register (R1518n)**

Offset Upper: 0x19C  
Lower: 0x198

Access: Read only

Diagram illustrating the state of the register R1518n:

- The register is 32 bits wide, indexed from 0 to 31.
- Bit 0 is labeled "R".
- Bit 31 is labeled "31".
- The current state of the register is shown below:
  - A bar labeled "W" contains the value "Reset".
  - A bar labeled "All zeros" shows all bits set to zero.

**Figure 6-57. Receive 1024- to 1518-Octet Packet Counter Register (R1518n)**

**Table 6-60. R1518*n* Field Description**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	R1518n	Incremented for each 1024- to 1518-octet frame received, good or bad.

#### **6.4.3.2.21 Receive 1519- to Max-Octet Packet Counter Register (R1519Xn)**

Offset Upper: 0x1A4  
Lower: 0x1A0

Access: Read only

Lower: 0xTA0

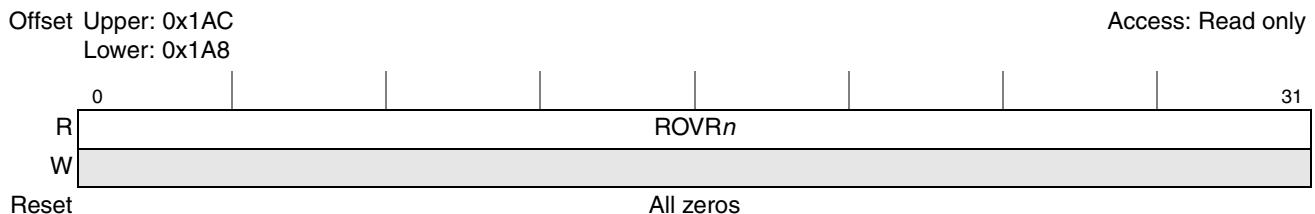
0								31
R	R1519Xn							
W								

Reset All zeros

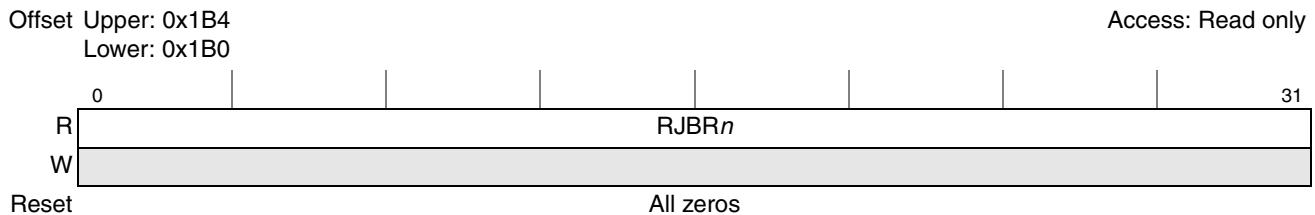
**Figure 6-58 Receive 1519- to Max-Octet Packet Counter Register (R1519X<sub>n</sub>)**

**Table 6-61. R1519Xn Field Description**

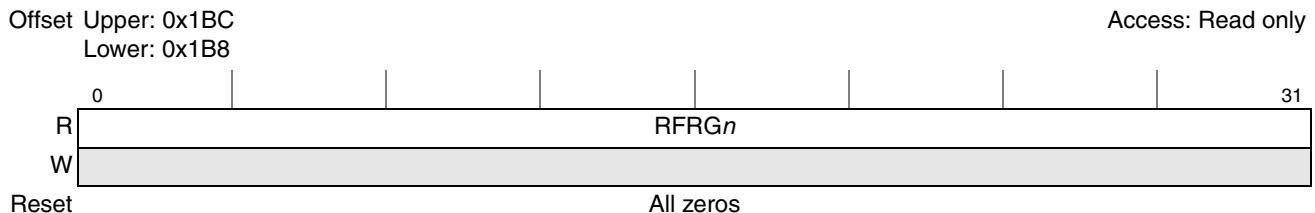
Bits	Name	Description
0–31	R1519Xn	Incremented for each 1519-octet frame and larger (up to the maximum frame length specified in the MAX_FRM register) received, good or bad.

**6.4.3.2.22 Receive Oversized Packet Counter Register (ROVRn)****Figure 6-59. Receive Oversized Packet Counter Register (ROVRn)****Table 6-62. ROVRn Field Description**

Bits	Name	Description
0–31	ROVRn	Incremented for each packet which is longer than the maximum frame length specified in the MAXFRM register received with a good frame check sequence.

**6.4.3.2.23 Receive Jabber Packet Counter Register (RJBRn)****Figure 6-60. Receive Jabber Packet Counter Register (RJBRn)****Table 6-63. RJBRn Field Description**

Bits	Name	Description
0–31	RJBRn	Incremented for each packet which is longer than the maximum frame length specified in the MAX_FRM register received with a bad frame check sequence.

**6.4.3.2.24 Receive Fragment Packet Counter Register (RFRGn)****Figure 6-61. Receive Fragment Packet Counter Register (RFRGn)**

**Table 6-64. RFRGn Field Description**

Bits	Name	Description
0–31	RFRG $n$	Incremented for each packet which is shorter than 64 octets received with a wrong CRC.

#### **6.4.3.2.25 Receive Control Packet Counter Register (RCNP $n$ )**

Offset Upper: 0x1C4

Access: Read only

RCNP $n$

**Reset** All zeros

All zeros

**Figure 6-62. Receive Control Packet Counter Register (RCNPn)**

**Table 6-65. RCNP*n* Field Description**

Bits	Name	Description
0–31	RCNP $n$	Incremented for each valid control packet (type 0x8808) but not for Pause packets

#### **6.4.3.2.26 Receive Dropped Not Truncated Packets Counter Register (RDRNTPn)**

Offset Upper: 0x1CC

Access: Read only

The diagram shows the RDRNTPn register structure. It consists of a horizontal line with seven vertical tick marks. The first tick mark is labeled '0' below it. The last tick mark is labeled '31' above it. The label 'RDRNTPn' is centered below the line.

W

ANSWER

**Figure 6-68. Receive-Filtered-Net-Truncated-Packets-Counter Register (RPNTRc)**

T-11-2-00-BBBNTB-F1-HD - 11

Bits	Name	Description
0–31	RDRNTP $n$	Incremented for each fully dropped packet (not truncated) due to internal errors. Occurs when a receive FIFO overflows.

#### 6.4.3.2.27 Transmit Ethernet Octets Counter (TEOCT $n$ )

Offset | Upper: 0x204

Access: Read only

Upper: 0x204  
Lower: 0x200

Diagram illustrating the structure of the TEOCT $n$  register:

- The register is 32 bits wide, indexed from 0 to 31.
- The bit at index 0 is labeled "R".
- The bits from index 1 to index 30 are labeled "TEOCT $n$ ".
- The bit at index 31 is labeled "Reset".
- The line "W" is shown below the bit 0 position.

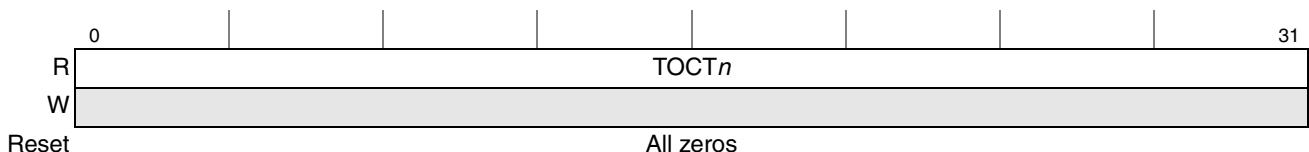
**Figure 6-64. Transmit Ethernet Octets Counter (TEOCT $n$ )**

**Table 6-67. TEOCT<sub>n</sub> Field Description**

Bits	Name	Description
0–31	TEOCT $n$	Incremented for each octet transmitted in both good and bad packets.

#### **6.4.3.2.28 Transmit Octets Counter Register (TOCT $n$ )**

Offset Upper: 0x20C Access: Read only  
Lower: 0x208



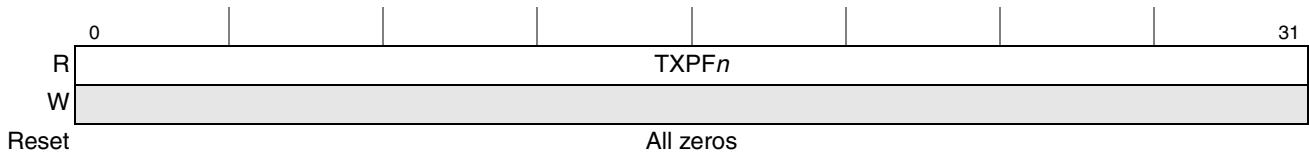
**Figure 6-65. Transmit Octets Counter Register (TOCTn)**

**Table 6-68. TOCTn Field Description**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	TOCT <i>n</i>	Incremented for each octet transmitted except preamble (that is, Header, Payload, Pad and FCS) for all valid frames and valid Pause frames transmitted.

#### **6.4.3.2.29 Transmit Valid Pause Frame Counter Register (TXPFn)**

Offset Upper: 0x21C Access: Read only  
Lower: 0x218



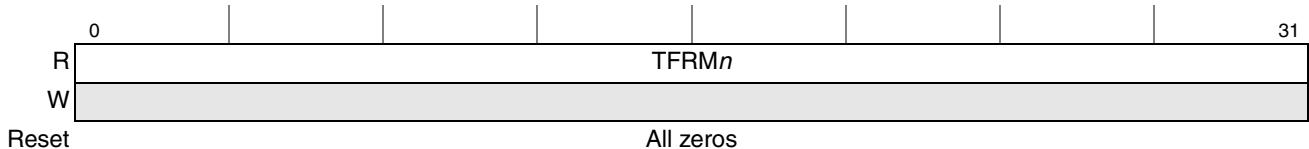
**Figure 6-66. Transmit Valid Pause Frame Counter Register (TXPFn)**

**Table 6-69. TXPF*n* Field Description**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	TXPF <i>n</i>	Incremented for each valid Pause frame transmitted (regular and PFC).

#### **6.4.3.2.30 Transmit Frame Counter Register (TFRM $n$ )**

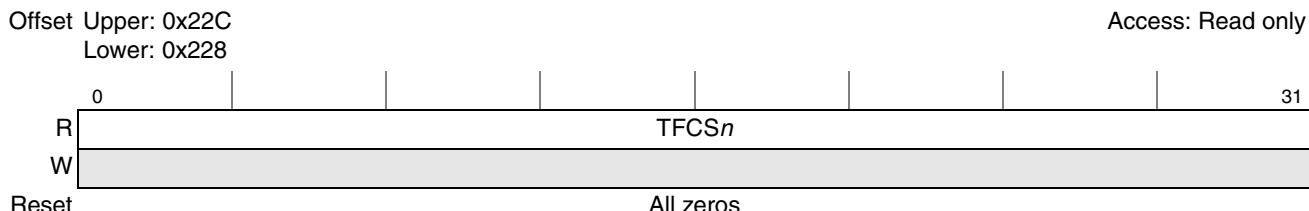
Offset Upper: 0x224 Access: Read only  
Lower: 0x220



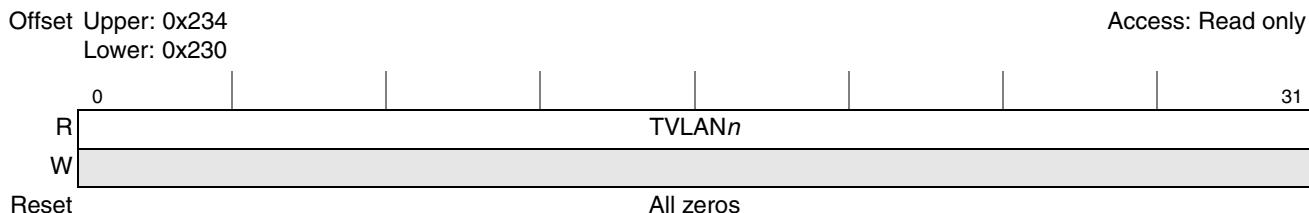
**Figure 6-67. Transmit Frame Counter Register (TFRM $n$ )**

**Table 6-70. TFRM $n$  Field Description**

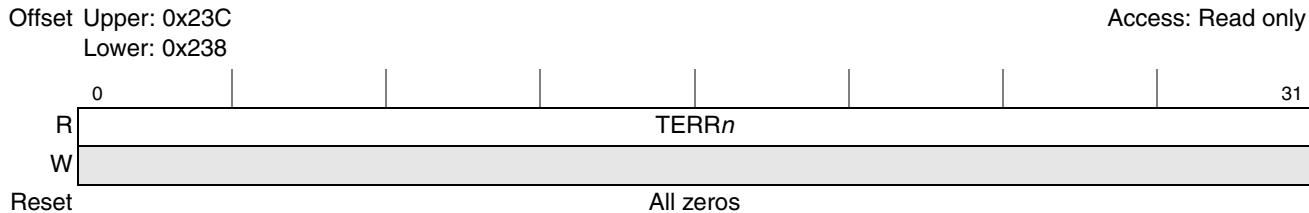
Bits	Name	Description
0–31	TFRM $n$	Incremented for each frame transmitted without error, including Pause frames.

**6.4.3.2.31 Transmit Frame Check Sequence Error Counter Register (TFCS $n$ )****Figure 6-68. Transmit Frame Check Sequence Error Counter Register (TFCS $n$ )****Table 6-71. TFCS $n$  Field Description**

Bits	Name	Description
0–31	TFCS $n$	Incremented for each frame transmitted with a CRC-32 error except for underflows.

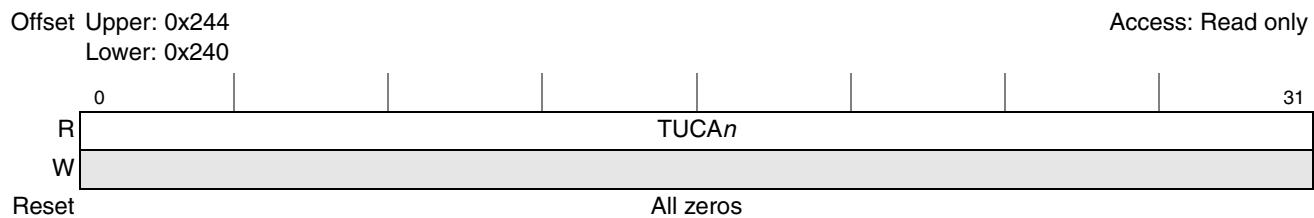
**6.4.3.2.32 Transmit VLAN Frame Counter Register (TVLAN $n$ )****Figure 6-69. Transmit VLAN Frame Counter Register (TVLAN $n$ )****Table 6-72. TVLAN $n$  Field Description**

Bits	Name	Description
0–31	TVLAN $n$	Incremented for each valid VLAN tagged frame transmitted.

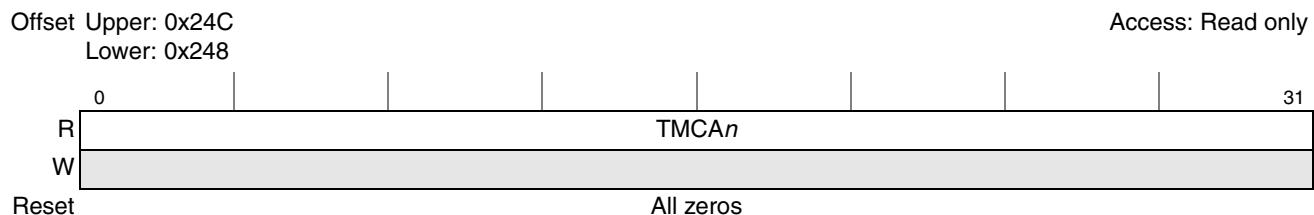
**6.4.3.2.33 Transmit Frame Error Counter Register (TERR $n$ )****Figure 6-70. Transmit Frame Error Counter Register (TERR $n$ )**

**Table 6-73. TERR $n$  Field Description**

Bits	Name	Description
0–31	TERR $n$	Incremented for each frame transmitted with an error: <ul style="list-style-type: none"> <li>• FIFO overflow error</li> <li>• FIFO underflow error</li> <li>• memory double ECC errors</li> </ul>

**6.4.3.2.34 Transmit Unicast Frame Counter Register (TUCAn)****Figure 6-71. Transmit Unicast Frame Counter Register (TUCAn)****Table 6-74. TUCAn Field Description**

Bits	Name	Description
0–31	TUCAn	Incremented for each valid frame transmitted in which bit 0 of the destination address was “0”.

**6.4.3.2.35 Transmit Multicast Frame Counter Register (TMCAAn)****Figure 6-72. Transmit Multicast Frame Counter Register (TMCAAn)****Table 6-75. TMCAAn Field Description**

Bits	Name	Description
0–31	TMCAAn	Incremented for each valid frame transmitted in which bit 0 of the destination address was “1” but not the broadcast address (all bits set to “1”).

#### 6.4.3.2.36 Transmit Broadcast Frame Counter Register (TBCAn)

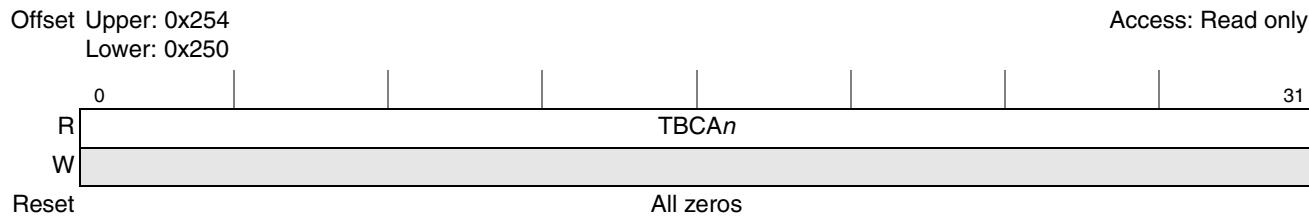


Figure 6-73. Transmit Broadcast Frame Counter Register (TBCAn)

Table 6-76. TBCAn Field Description

Bits	Name	Description
0–31	TBCAn	Incremented for each valid frame transmitted in which all bits of the destination address were “1”.

#### 6.4.3.2.37 Transmit Packets Counter Register (TPKTN)

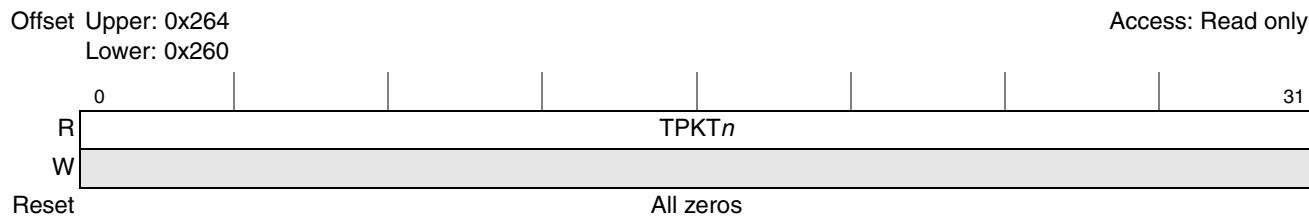


Figure 6-74. Transmit Packets Counter Register (TPKTN)

Table 6-77. TPKTN Field Description

Bits	Name	Description
0–31	TPKTN	Incremented for each good or bad packet transmitted.

#### 6.4.3.2.38 Transmit Undersized Packet Counter Register (TUNDn)

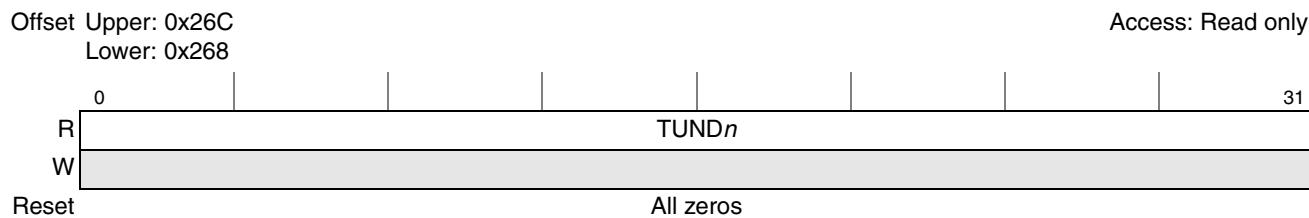
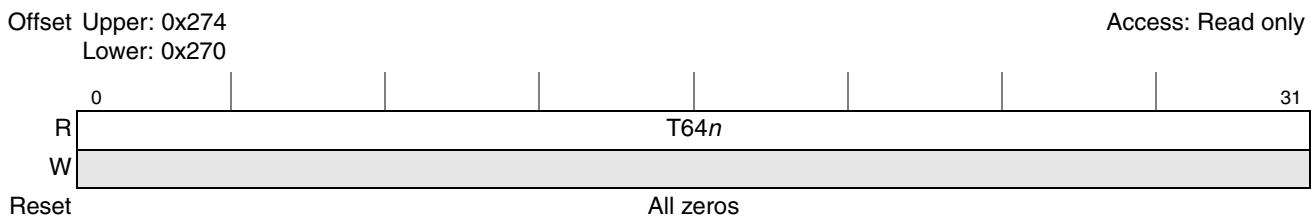


Figure 6-75. Transmit Undersized Packet Counter Register (TUNDn)

Table 6-78. TUNDn Field Description

Bits	Name	Description
0–31	TUNDn	Incremented for each packet transmitted that was less than 64 octets long with a good CRC.

#### **6.4.3.2.39 Transmit 64-Octet Packet Counter Register (T64n)**

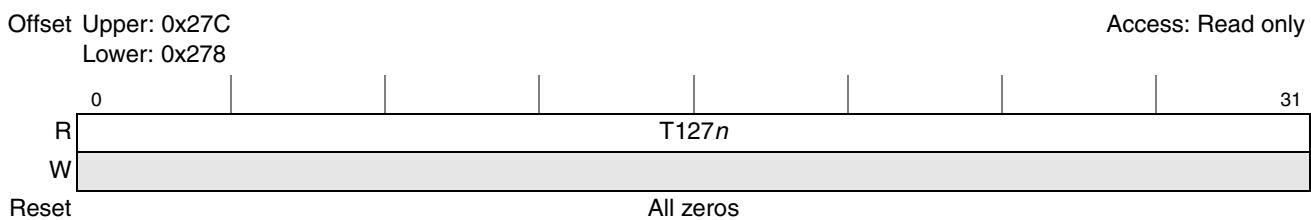


**Figure 6-76. Transmit 64-Octet Packet Counter Register (T64n)**

**Table 6-79. T64n Field Description**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	T64n	Incremented for each 64-octet frame transmitted, good or bad.

#### **6.4.3.2.40 Transmit 65- to 127-Octet Packet Counter Register (T127n)**

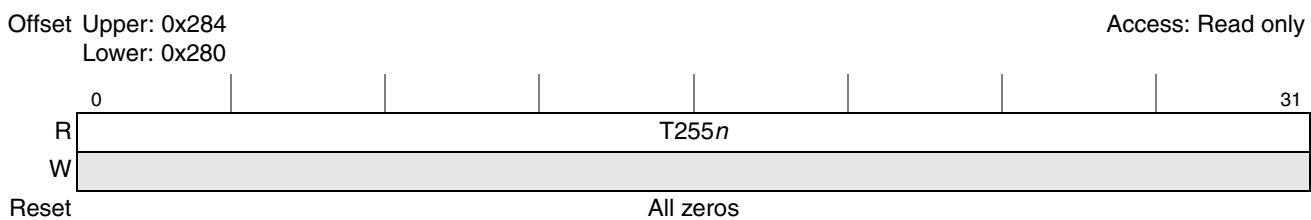


**Figure 6-77. Transmit 65- to 127-Octet Packet Counter Register (T127n)**

**Table 6-80. T127*n* Field Description**

Bits	Name	Description
0–31	T127 <i>n</i>	Incremented for each 65- to 127-octet frame transmitted, good or bad.

#### **6.4.3.2.41 Transmit 128- to 255-Octet Packet Counter Register (T255n)**

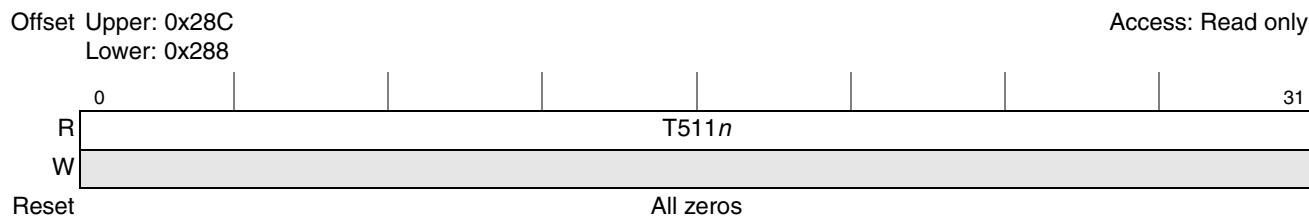


**Figure 6-78. Transmit 128- to 255-Octet Packet Counter Register (T255n)**

**Table 6-81. T255*n* Field Description**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	T255 <i>n</i>	Incremented for each 128- to 255-octet frame transmitted, good or bad.

#### 6.4.3.2.42 Transmit 256- to 511-Octet Packet Counter Register (T511*n*)

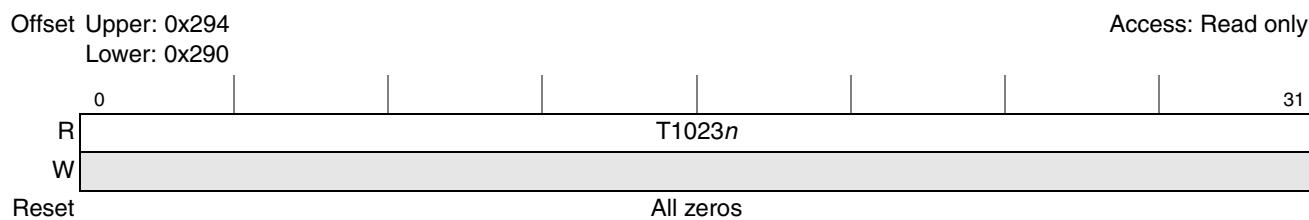


**Figure 6-79. Transmit 256- to 511-Octet Packet Counter Register (T511*n*)**

**Table 6-82. T511*n* Field Description**

Bits	Name	Description
0–31	T511 <i>n</i>	Incremented for each 256- to 511-octet frame transmitted, good or bad.

#### 6.4.3.2.43 Transmit 512- to 1023-Octet Packet Counter Register (T1023*n*)

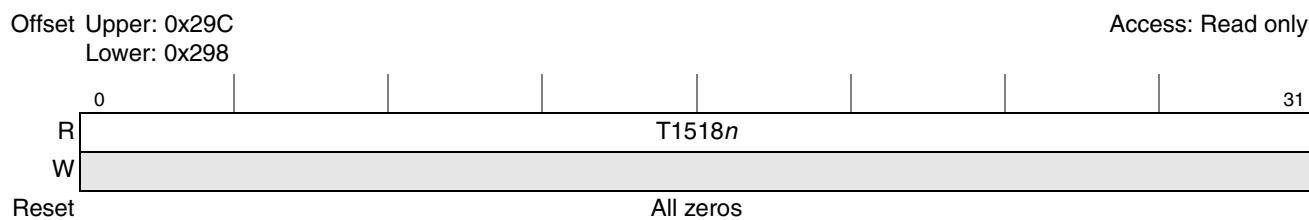


**Figure 6-80. Transmit 512- to 1023-Octet Packet Counter Register (T1023*n*)**

**Table 6-83. T1023*n* Field Description**

Bits	Name	Description
0–31	T1023 <i>n</i>	Incremented for each 512- to 1023-octet frame transmitted, good or bad.

#### 6.4.3.2.44 Transmit 1024- to 1518-Octet Packet Counter Register (T1518*n*)

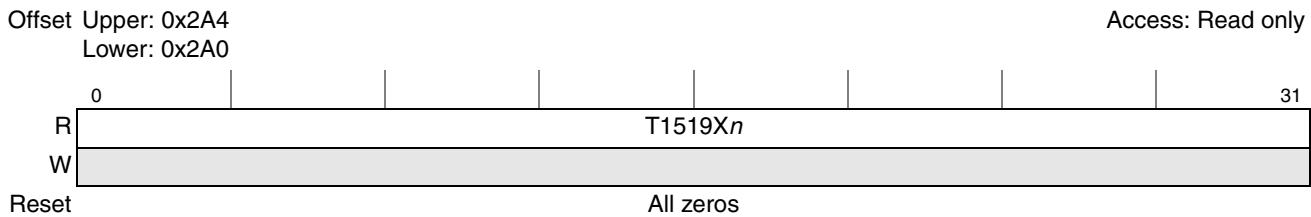


**Figure 6-81. Transmit 1024- to 1518-Octet Packet Counter Register (T1518*n*)**

**Table 6-84. T1518*n* Field Description**

Bits	Name	Description
0–31	T1518 <i>n</i>	Incremented for each 1024- to 1518-octet frame transmitted, good or bad.

#### **6.4.3.2.45 Transmit 1519- to TX\_MTU-Octet Packet Counter Register (T1519Xn)**

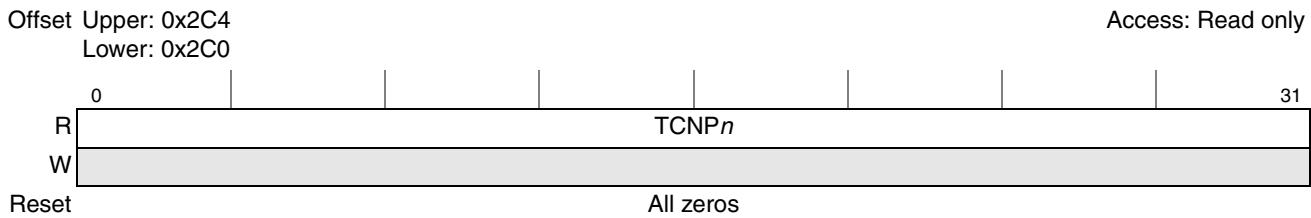


**Figure 6-82. Transmit 1519- to TX\_MTU-Octet Packet Counter Register (T1519Xn)**

**Table 6-85. T1519Xn Field Description**

Bits	Name	Description
0–31	T1519X $n$	Incremented for each 1519-octet frame and larger (up to the maximum frame length specified in the MAX_FRM register) transmitted, good or bad.

#### **6.4.3.2.46 Transmit Control Packet Counter Register (TCNP<sub>n</sub>)**



**Figure 6-83. Transmit Control Packet Counter Register (TCNP*n*)**

**Table 6-86. TCNP*n* Field Description**

Bits	Name	Description
0–31	TCNP <i>n</i>	Incremented for each valid control packet transmitted (type 0x8808) but not for Pause packets

### **6.4.3.3 Line Interface Control Registers**

#### **6.4.3.3.1 Interface Mode Register (IF\_MODE)**

This register defines the type of interface used and RGMII configurations. Software must initialize this register. It is not automatically populated by RCW selections

Offset 0x300

Access: Read/Write

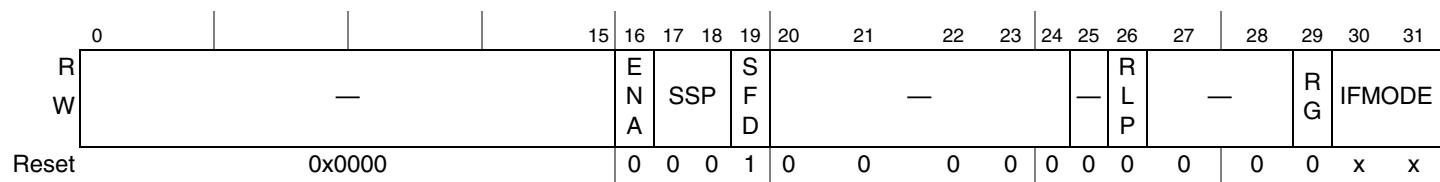


Figure 6-84. IF\_MODE Register

Table 6-87. IF\_MODE Field Description

Bits	Name	Description
0–15	—	Reserved
16	ENA	<ul style="list-style-type: none"> <li>0 - SSP bits determine the RGMII link speed.</li> <li>1 - Enable automatic speed selection - RGMII PHY in-band status information is used to select the speed of operation.</li> </ul> <p><b>Note:</b> This field is applicable when RG=1 and ignored when RG=0.</p>
17–18	SSP	<ul style="list-style-type: none"> <li>00 - 100 Mbps RGMII (valid only if ENA=0)</li> <li>01 - 10 Mbps RGMII (valid only if ENA=0)</li> <li>10 - 1 Gbps RGMII (valid only if ENA=0)</li> <li>11 - Reserved</li> </ul> <p><b>Note:</b> This field is applicable when RG=1 and ignored when RG=0.</p>
19	SFD	<ul style="list-style-type: none"> <li>1 - forces full duplex RGMII mode (bit is valid only if ENA=0)</li> </ul> <p><b>Note:</b> This field is applicable when RG=1 and ignored when RG=0.</p>
20–23	—	Reserved
24–25	—	Reserved
26	RLP	<ul style="list-style-type: none"> <li>0 - normal operation</li> <li>1 - RGMII internal loopback (XGLP should be 0 and INV should be 1)</li> </ul> <p><b>Note:</b> This field is applicable when RG=1 and ignored when RG=0.</p>
27–28	—	Reserved
29	RG	<ul style="list-style-type: none"> <li>0 - non RGMII mode</li> <li>1 - RGMII mode (when GMII mode is selected)</li> </ul>
30–31	IFMODE	<ul style="list-style-type: none"> <li>00 - 10G interface mode</li> <li>01 - MII mode</li> <li>10 - GMII mode (also for RGMII)</li> <li>11 - Reserved</li> </ul>

#### 6.4.3.3.2 Interface Status Register (IF\_STATUS)

Offset 0x304

Access: Read only

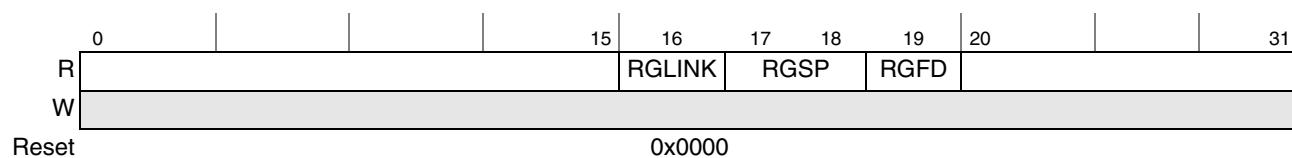


Figure 6-85. IF\_STATUS Register

**Table 6-88. IF\_STATUS Field Description**

Bits	Name	Description
0–15	—	Reserved
16	RGLINK	<ul style="list-style-type: none"> <li>0 - no link is established or RGMII PHY does not support the optional in-band signaling</li> <li>1 - a valid link is established by the RGMII PHY (if it supports the optional in-band signaling)</li> </ul>
17–18	RGSP	<ul style="list-style-type: none"> <li>00 - 10 Mbps (either controlled by in-band RGMII PHY status or by forced speed settings)</li> <li>01 - 100 Mbps (either controlled by in-band RGMII PHY status or by forced speed settings)</li> <li>10 - 1 Gbps (either controlled by in-band RGMII PHY status or by forced speed settings)</li> <li>11 - Reserved</li> </ul>
19	RGFD	<ul style="list-style-type: none"> <li>0 - no link is established or RGMII PHY does not support the optional in-band signaling</li> <li>1 - RGMII full duplex link is established (bit is valid if PHY supports the optional in-band signaling)</li> </ul>
20–31	—	Reserved

#### 6.4.3.4 MDIO Ethernet Management Interface Registers

There are two types of MDIO Ethernet Management Interfaces (EMIs), internal and external. The internal MDIOs support the MDIO-*n* for EMAC*n* port configuration and management. The external MDIOs support PHY device configuration and management through dedicated MDIOs (MDIO1 and MDIO2). Both internal and external MDIOs use the same register formats with some differences noted in this section. See [Table 5-17](#) for the base addresses for each internal and external MDIO.

##### 6.4.3.4.1 MDIO Configuration Register (MDIO\_CFG)

MDIO interrupt is enabled by bit 2.

Offset 0x030												Access: Mixed			
R	0	1	2	3	—	7	8	9	10	—	15				
W	BSY	CMP	CIM	—	—	NEG <sup>1</sup>	EHOLD	—	—	—					
	w1c														
Resets:	0	0	0	0	0	0	0	0	0	0	0				
Internal	0	0	0	0	0	0	0	0	0	0	0				
External	0	0	0	0	0	0	0	0	0	0	0				
	16	—	24	25	26	27	29	30	31	—					
R	MDIO_CLK_DIV <sup>2</sup>								ENC45	PRE	MDIO_HOLD	MDIO_RD_ER	—		
W															
Resets:	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0
Internal	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0
External	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

**Figure 6-86. MDIO\_CFG Register Definition**

<sup>1</sup> Reset value of NEG for external MDIOs is 1

<sup>2</sup> Reset value of MDIO\_CLK\_DIV for external MDIOs is 9'b0

**Table 6-89. MDIO\_CFG Field Descriptions**

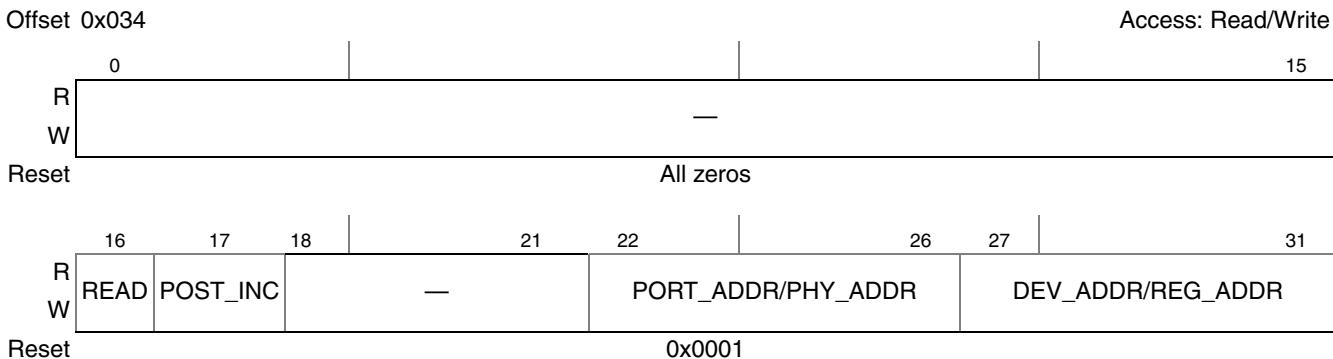
Bits	Name	Description
0	BSY	MDIO busy 0 An MDIO transaction is not occurring; software may access other MDIO registers. 1 An MDIO transaction is occurring.
1	CMP	MDIO command completion event. Bit is cleared by writing '1'. 0 An MDIO command completion did not occur. 1 An MDIO command completion occurred.
2	CIM	MDIO command completion interrupt mask. 0 masked 1 enabled
3–7	—	Reserved
8	NEG	(applicable only for external MDIOs; bit is always 0 for internal MDIOs) 0 normal operation 1 MDIO is driven by master on MDC negative edge (default for external MDIOs)
9	EHOLD	Extended MDIO hold time. Applicable only for external MDIOs, should be set to 0 for internal MDIOs. 0 Do not extend hold time 1 Extend hold time See table in MDIO_HOLD bit field 27–29 description.
10–15	—	Reserved
16–24	MDIO_CLK_DIV	MDIO clock divisor. A value of 5–511 can be set to configure the MDIO clock frequency relative to FMan frequency. Ratio is $(2 \times \text{MDIO\_CLK\_DIV}) + 1$ . Setting the divisor to 0 disables MDC. <b>For internal PCS MDIO</b> Ethernet management interface the default is 0b0_0010_1000 <b>For external MDIO</b> Ethernet management interface the default is 9'b0, which disables MDC. <b>Note:</b> For internal MDIO/MDC accesses, only default MDIO_CLK_DIV value should be used regardless of FMan frequency value. If MDIO_CLK_DIV is programmed with a different value than default setting, then for internal MDC accesses the upper resulting MDC frequency should not exceed 10 MHz.
25	ENC45	Enable Clause 45 support. Select based on PCS. For example, if using XFI or 10GBase-KR, set ENC45 = 1 or if using SGMII or QSGMII, choose ENC45 = 0. 0 Clause 22 transactions are used. 1 Clause 45 transactions are used (default). <b>Note:</b> To switch between ENC45 modes, from a Clause 45 to Clause 22 setting or the reverse, software must add an up to 50 MDC cycle delay.
26	PRE	MDIO preamble disable. 0 Generation of MDIO preamble is enabled (Default operation. Do not change for internal MDIO accesses.) 1 Generation of MDIO preamble is disabled.

**Table 6-89. MDIO\_CFG Field Descriptions (continued)**

Bits	Name	Description																		
27–29	MDIO_HOLD	<p>MDIO hold time.</p> <p><b>For internal MDIO mode</b>, do not change the default value of ‘010’ and EHOLD (bit 9) must be 0.</p> <p><b>For external MDIO mode</b>, use this equation to determine the FMan clock cycles listed in table.</p> $\text{MDIO hold time} = 1 + (2 + 6 \times \text{EHOLD}) \times \text{MDIO\_HOLD}$ <table border="1" data-bbox="722 481 1248 751"> <thead> <tr> <th data-bbox="722 481 866 511">Bit setting</th><th data-bbox="866 481 1248 511">EHOLD = 1</th></tr> </thead> <tbody> <tr> <td data-bbox="722 511 866 540">000</td><td data-bbox="866 511 1248 540">1 FMan clock cycle</td></tr> <tr> <td data-bbox="722 540 866 570">001</td><td data-bbox="866 540 1248 570">9 FMan clock cycles</td></tr> <tr> <td data-bbox="722 570 866 599">010</td><td data-bbox="866 570 1248 599">17 FMan clock cycles</td></tr> <tr> <td data-bbox="722 599 866 629">011</td><td data-bbox="866 599 1248 629">25 FMan clock cycles</td></tr> <tr> <td data-bbox="722 629 866 658">100</td><td data-bbox="866 629 1248 658">33 FMan clock cycles</td></tr> <tr> <td data-bbox="722 658 866 688">101</td><td data-bbox="866 658 1248 688">41 FMan clock cycles</td></tr> <tr> <td data-bbox="722 688 866 718">110</td><td data-bbox="866 688 1248 718">49 FMan clock cycles</td></tr> <tr> <td data-bbox="722 718 866 751">111</td><td data-bbox="866 718 1248 751">57 FMan clock cycles</td></tr> </tbody> </table>	Bit setting	EHOLD = 1	000	1 FMan clock cycle	001	9 FMan clock cycles	010	17 FMan clock cycles	011	25 FMan clock cycles	100	33 FMan clock cycles	101	41 FMan clock cycles	110	49 FMan clock cycles	111	57 FMan clock cycles
Bit setting	EHOLD = 1																			
000	1 FMan clock cycle																			
001	9 FMan clock cycles																			
010	17 FMan clock cycles																			
011	25 FMan clock cycles																			
100	33 FMan clock cycles																			
101	41 FMan clock cycles																			
110	49 FMan clock cycles																			
111	57 FMan clock cycles																			
30	MDIO_RD_ER	<p>MDIO read error</p> <p>1 The last read transaction received no response from a PHY; any data read should be considered invalid. (for example, The PHY address does not match any PHY available on the MDIO bus.)</p>																		
31	—	Reserved																		

#### 6.4.3.4.2 MDIO Control Register (MDIO\_CTL)

The MDIO\_CTL is used to control the MDIO interface.



**Figure 6-87. MDIO\_CTL Register Definition**

**Table 6-90. MDIO\_CTL Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–15	—	Reserved
16	READ	MDIO read initiation. 1 A normal MDIO read transaction is initiated.

**Table 6-90. MDIO\_CTL Field Descriptions (continued)**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
17	POST_INC	MDIO read with address post-increment initiation. 0 (default operation) 1 An MDIO read with Clause 45 address post-increment transaction is initiated
18–21	—	Reserved
22–26	PORT_ADDR/PHY_A_DDR	Five-bit MDIO port address (Clause 45) / PHY address (Clause 22).
27–31	DEV_ADDR/REG_ADDR	Five-bit MDIO device address (Clause 45) / register address (Clause 22).

#### **6.4.3.4.3 MDIO Data Register (MDIO\_DATA)**

The MDIO DATA is used to communicate with an attached PHY.



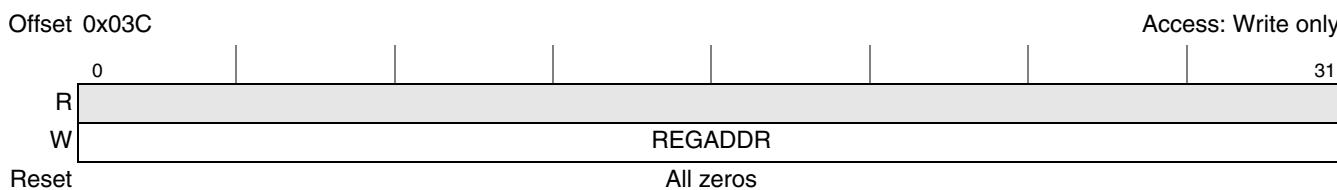
**Figure 6-88. MDIO\_DATA Register Definition**

**Table 6-91. MDIO\_DATA Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	MDIO_DATA	<p>16-bit MDIO data.</p> <ul style="list-style-type: none"> <li>Writing to this register initiates a write transaction to the PHY. For proper operation, the MDIO_CTL register must have been initialized first. The MDIO_CFG[BSY] status bit is set immediately and cleared when the write transaction has completed.</li> <li>Reading this register returns data read from the PHY register specified in MDIO_CTL after a read transaction has completed. Read transactions are initiated by writing a 1 to MDIO_CTL[READ] or MDIO_CTL[POST_INC]. Read data is invalid if MDIO_CFG[BSY] is set.</li> </ul>

#### 6.4.3.4.4 MDIO Register Address Register (MDIO\_ADDR)

The MDIO ADDR is used to communicate with an attached Clause 45 PHY.



**Figure 6-89. MDIO\_ADDR Register Definition**

**Table 6-92. MDIO\_ADDR Field Descriptions**

<b>Bits</b>	<b>Name</b>	<b>Description</b>
0–31	REGADDR	MDIO PHY register address. Address of the register within the Clause 45 PHY device from which data is to be read or to which data is to be written. Writing to this register initiates an address-write transaction to set the PHY's internal address register to the value specified in MDIO_ADDR[REGADDR]. For proper operation, the MDIO_CTL register must have been initialized prior to writing to this register.

## 6.5 mEMAC Functional Description

### 6.5.1 MAC Address Insertion

On each frame received from the transmit FIFO interface, the source MAC address is optionally replaced by the address programmed in the configuration registers:

- MAC\_ADDR\_0 and MAC\_ADDR\_1 (COMMAND\_CONFIG[TX\_ADDR\_INS] set)

or is transparently forwarded to the Ethernet line (COMMAND\_CONFIG[TX\_ADDR\_INS] is cleared).

### 6.5.2 CRC-32 Calculation

The CRC polynomial, as specified in the 802.3 standard, is:

$$FCS(X) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The 32 bits of the CRC value are placed in the FCS field so that the  $x^{31}$  term is the right-most bit of the first octet. The CRC bits are thus transmitted in the following order:  $x^{31}, x^{30} \dots x^1, x^0$ .

### 6.5.3 Unicast destination address recognition

Unless COMMAND\_CONFIG[PROMIS] is set, unicast frames with a destination address not matching the MAC addresses (defined by registers MAC\_ADDR\_0/1 till MAC\_ADDR\_14/15) are rejected

### 6.5.4 MDIO Ethernet Management Interface usage

MDIO\_CFG should be initialized (once) before the flows described below.

#### 6.5.4.1 Clause 45 Read Flow

- 1) Wait for MDIO\_CFG[BSY] = 0.
- 2) MDIO\_CTL must be initialized with proper PORT\_ADDR and DEV\_ADDR and with bits 16-17 cleared.
- 3) Initialize MDIO\_ADDR with proper REGADDR.
- 4) Wait for MDIO\_CFG[BSY] = 0.

- 5) Write MDIO\_CTL with proper PORT\_ADDR and DEV\_ADDR and with bit 16 or bit 17 set.  
 Bit 16 - READ - normal read. Bit 17 - a read and afterwards PHY internal address is incremented by one.
- 6) Wait for MDIO\_CFG[BSY] = 0.
- 7) If the addressed PHY did not respond then MDIO\_CFG[MDIO\_RD\_ER] is set.  
 Otherwise read MDIO\_DATA value.
- Steps 3 & 4 can be skipped if subsequent access to the same PHY register or a read with post-increment was previously performed.

#### **6.5.4.2 Clause 45 Write Flow**

- 1) Wait for MDIO\_CFG[BSY] = 0.
  - 2) MDIO\_CTL must be initialized with proper PORT\_ADDR and DEV\_ADDR and with bits 16-17 cleared.
  - 3) Initialize MDIO\_ADDR with proper REGADDR.
  - 4) Wait for MDIO\_CFG[BSY] = 0.
  - 5) Write MDIO\_DATA with proper value.
- Steps 3 & 4 can be skipped if subsequent access to the same PHY register.

#### **6.5.4.3 Clause 22 Read Flow**

- 1) Wait for MDIO\_CFG[BSY] = 0.
  - 2) Write MDIO\_CTL with proper PHY\_ADDR and REGISTER\_ADDR and with bit 16 set.
  - 3) Wait for MDIO\_CFG[BSY] = 0.
  - 4) If the addressed PHY did not respond then MDIO\_CFG[MDIO\_RD\_ER] is set.
- Otherwise read MDIO\_DATA value.

#### **6.5.4.4 Clause 22 Write Flow**

- 1) Wait for MDIO\_CFG[BSY] = 0.
- 2) MDIO\_CTL must be initialized with proper PHY\_ADDR and REGISTER\_ADDR and with bits 16-17 cleared.
- 3) Write MDIO\_DATA with proper value.

### **6.5.5 Graceful stop**

In order to stop receive:

- 1) Write COMMAND\_CONFIG[RXSTP]=1
- 2) Wait for IEVENT[RX\_EMPTY] to be set
- 3) Write COMMAND\_CONFIG[RX\_EN]=0

In order to stop transmit:

- 1) Disable FM\_BMI proper port etc
- 2) Wait for IEVENT[TX\_EMPTY] to be set
- 3) Write COMMAND\_CONFIG[TX\_EN]=0

## 6.6 Initialization Information

- Set COMMAND\_CONFIG[SWR]
- Initialize IF\_MODE register
- Set STATN\_CONFIG[CLR]
- Initialize MAC\_ADDR\_0, ..., MAC\_ADDR\_15 registers
- Clear IEVENT register
- Initialize IMASK register
- Set COMMAND\_CONFIG[RX\_EN], set COMMAND\_CONFIG[TX\_EN].

# Chapter 7

## IEEE 1588 Timer Module

### 7.1 Overview

The 1588 timer module interfaces to up to an unlimited number of 10/100/1000 MB or 10G Ethernet MACs, providing current time, alarm, and FIFER support.

#### 7.1.1 Features

The 1588 control block includes these distinctive features:

- 1588 Timer
  - External GPIO trigger for time-stamping
  - 2 Time-stamp alarms
  - 3 FIPER pulse generators
  - Phase adjusted output timer clock

### 7.2 Modes of Operation

The 1588 timer module supports all full-duplex modes of the associated Ethernet MACs.

### 7.3 External Signals Description

This section defines the Timer-to-chip signals. [Table 7-1](#) lists the external interface signals.

**Table 7-1. 1588 Timer External Interface Signal Summary**

Signal Name	Function/Description	Reset	I/O
TMR_1588_CLK	Timer 1588 Reference clock input	0	I
TMR_TRIG1	Timer External Trigger1 input	0	I
TMR_TRIG2	Timer External Trigger2 input	0	I
TMR_GCLK	Phase Adjusted Timer Generated Clock output	0	O
TMR_PP1	Timer FIPER Periodic Pulse1 output	0	O
TMR_PP2	Timer FIPER Periodic Pulse2 output	0	O
TMR_PP3	Timer FIPER Periodic Pulse3 output	0	O
TMR_ALARM1	Timer Time-Stamp Alarm1 Trigger output	0	O
TMR_ALARM2	Timer Time-Stamp Alarm2 Trigger output	0	O

### 7.3.1 Detailed Signal Descriptions

Below is a description of the 1588 timer module interface signals. Input signals not used are internally disabled. Output signals not used are driven low.

**Table 7-2. 1588 Timer—Detailed Signal Descriptions**

Signal	I/O	Description
TMR_1588_CLK	I	1588 clock in. External high precision timer reference clock input (chip external input pin).
TMR_TRIG1	I	1588 trigger in 1. External timer trigger input 1. This is an asynchronous general purpose input (chip external input pin).
TMR_TRIG2	I	1588 trigger in 2. External timer trigger input 2. This is an asynchronous general purpose input (chip external input pin).
TMR_GCLK	O	1588 clock out. Phase aligned timer clock divider output (chip external output pin).
TMR_PP1	O	1588 pulse out 1. Timer pulse per period 1. It is phase aligned with 1588 timer clock (chip external output pin)
TMR_PP2	O	1588 pulse out 2. Timer pulse per period 2. It is phase aligned with 1588 timer clock (chip external output pin)
TMR_PP3	O	1588 pulse out 3. Timer pulse per period 3. It is phase aligned with 1588 timer clock (chip external output pin)
TMR_ALARM1	O	Timer current time is equal to or greater than alarm time comparator register. User deactivate this output by writing to the TMR_ALARMn_L register. After deactivating it, the user can rearm the alarm by writing to the TMR_ALARMn_H, enabling the triggered alarm to assert the output (chip external output pin).
TMR_ALARM2	O	Timer current time is equal to or greater than alarm time comparator register. User deactivate this output by writing to the TMR_ALARMn_L register. After deactivating it, the user can rearm the alarm by writing to the TMR_ALARMn_H, enabling the triggered alarm to assert the output (chip external output pin).

### 7.4 Memory Map/Register Definition

The 1588 timer module is programmed by control/status registers (CSRs). The CSRs are used for mode control, interrupts, and to extract status information.

All accesses to and from the registers must be made as 32-bit accesses. There is no support for accesses of sizes other than 32 bits. Writes to reserved register bits must always store the previous value, as changing the value of reserved bits may have unintended side-effects. Unless otherwise specified, the read value of unmapped registers or of reserved bits in mapped registers is not defined, and must not be assumed to be 0.

#### 7.4.1 Top-Level Module Memory Map

The 1588 timer module is allocated 256 bytes of memory-mapped space. The space for the 1588 timer module is divided as indicated in [Table 7-3](#).

**Table 7-3. Module Memory Map Summary**

Register Address	Function
000–07C	General control/status registers
080–0FC	1588 Timer registers

## 7.4.2 Detailed Memory Map

**Table 7-4** lists the address, name, and a cross-reference to the complete description of each register. The offsets to the memory map table are defined for timer starting at 0x000 address offset and goes to 0x0FF (256 byte region represented by 64 registers of 4 bytes each).

In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- Reserved registers and fields must be preserved on writes.
- R/W, R, and W (read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type.
- w1c indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

**Table 7-4. Module Memory Map**

Timer Offset	Name	Access <sup>1</sup>	Reset	Section/page
<b>Timer General Control and Status Registers</b>				
0x000	TMR_ID—Module ID and version register	R	0x0300_0100	<a href="#">7.4.3.1.1/7-4</a>
0x004	TMR_ID2—Module ID and configuration register	R	0x0000_0000	<a href="#">7.4.3.1.2/7-5</a>
0x008–0x07C	Reserved	—	—	—
<b>1588 Timer Registers</b>				
0x080	TMR_CTRL - Timer control register	R/W	0x0001_0001	<a href="#">7.4.3.2.1/7-5</a>
0x084	TMR_TEVENT - Timer stamp event register	R/W	0x0000_0000	<a href="#">7.4.3.2.2/7-7</a>
0x088	TMR_TEMASK - Timer event mask register	R/W	0x0000_0000	<a href="#">7.4.3.2.3/7-8</a>
0x08C–0x090	Reserved	—	—	—
0x094	TMR_STAT - Timer status register	R/W	0x0000_0000	<a href="#">7.4.3.2.5/7-10</a>
0x098	TMR_CNT_H - Timer counter high register	R/W	0x0000_0000	<a href="#">7.4.3.2.5/7-10</a>
0x09C	TMR_CNT_L - Timer counter low register	R/W	0x0000_0000	<a href="#">7.4.3.2.5/7-10</a>
0x0A0	TMR_ADD - Timer drift compensation addend register	R/W	0x0000_0000	<a href="#">7.4.3.2.6/7-11</a>
0x0A4	TMR_ACC - Timer accumulator register	R	0x0000_0000	<a href="#">7.4.3.2.7/7-12</a>
0x0A8	TMR_PRSC - Timer prescale	R/W	0x0000_0002	<a href="#">7.4.3.2.8/7-12</a>

**Table 7-4. Module Memory Map (continued)**

Timer Offset	Name	Access <sup>1</sup>	Reset	Section/page
0x0B0	TMROFF_H - Timer offset high	R/W	0x0000_0000	<a href="#">7.4.3.2.9/7-13</a>
0x0B4	TMROFF_L - Timer offset low	R/W	0x0000_0000	<a href="#">7.4.3.2.9/7-13</a>
0x0B8	TMR_ALARM1_H - Timer alarm 1 high register	R/W	0xFFFF_FFFF	<a href="#">7.4.3.2.10/7-14</a>
0x0BC	TMR_ALARM1_L - Timer alarm 1 high register	R/W	0xFFFF_FFFF	
0x0C0	TMR_ALARM2_H - Timer alarm 2 high register	R/W	0xFFFF_FFFF	
0x0C4	TMR_ALARM2_L - Timer alarm 2 high register	R/W	0xFFFF_FFFF	
0x0C8–0x0CC	Reserved	—	—	—
0x0D0	TMR_FIPER1 - Timer fixed period interval	R/W	0xFFFF_FFFF	<a href="#">7.4.3.2.11/7-14</a>
0x0D4	TMR_FIPER2 - Timer fixed period interval	R/W	0xFFFF_FFFF	
0x0D8	TMR_FIPER3 - Timer fixed period interval	R/W	0xFFFF_FFFF	
0x0DC	Reserved	—	—	—
0x0E0	TMR_ETTS1_H - Time stamp of general purpose external trigger	R	0x0000_0000	<a href="#">7.4.3.2.12/7-16</a>
0x0E4	TMR_ETTS1_L - Time stamp of general purpose external trigger	R	0x0000_0000	
0x0E8	TMR_ETTS2_H - Time stamp of general purpose external trigger	R	0x0000_0000	
0x0EC	TMR_ETTS2_L - Time stamp of general purpose external trigger	R	0x0000_0000	
0x0F0–0x0FC	Reserved	—	—	—

<sup>1</sup> Key: R = read only, WO = write only, R/W = read and write, LH = latches high, SC = self-clearing

## 7.4.3 Memory-Mapped Register Descriptions

This section provides a detailed description of all the 1588 timer module registers. Because all of the registers are 32 bits wide, only 32-bit register accesses are supported.

### 7.4.3.1 General Control and Status Registers

This section describes general control and status registers.

#### 7.4.3.1.1 Module ID Register (TMR\_ID)

The module ID register (TMR\_ID) is a read-only register. The TMR\_ID register is used to identify the 1588 timer block and revision. Currently the 1588 timer ID is 0x0300 and the major revision is 0x01, while the minor revision is 0x00.

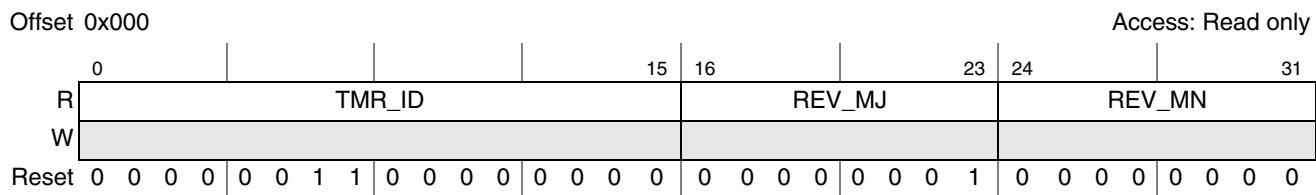
**Figure 7-1. TMR\_ID Register Definition**

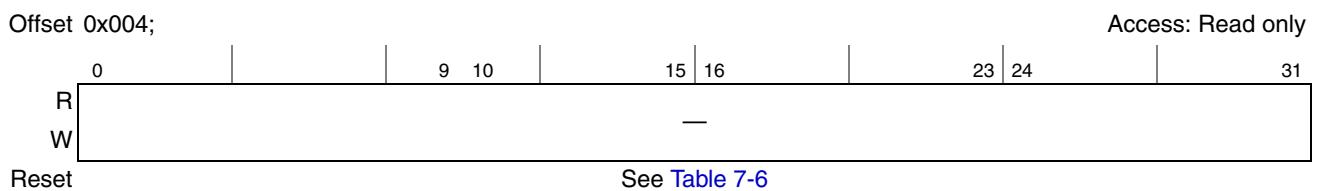
Table 7-5 describes the fields of the TMR\_ID register.

**Table 7-5. TMR\_ID Field Descriptions**

Bit	Name	Description
0–15	TMR_ID	Value identifying the 1588 timer module 0300 Unique identifier for the 1588 timer module.
16–23	REV_MJ	Value identifies the major revision of the 1588 timer module. 00 is the initial version
24–31	REV_MN	Value identifies the minor revision of the 1588 timer module. 00 is the initial revision

#### **7.4.3.1.2 Controller ID Register (TMR\_ID2)**

The controller ID register (TMR\_ID2), shown in Figure 7-2, is a read-only register. The TMR\_ID2 register is used to identify available module configurations.



## Figure 7-2. TMR\_ID2 Register Definition

Table 7-6 describes the fields of the TMR\_ID2 register.

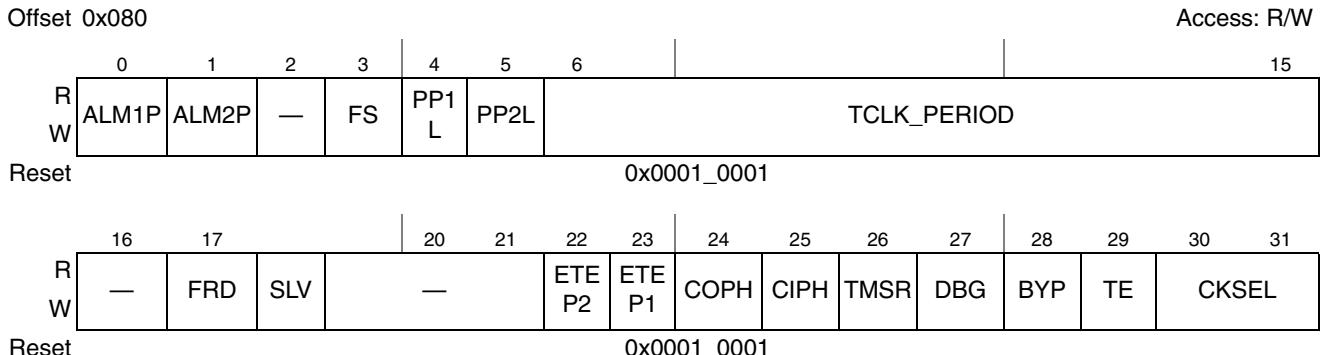
**Table 7-6. TMR\_ID2 Field Descriptions**

Bit	Name	Description
0–31	—	Reserved

### **7.4.3.2 Hardware Assist for IEEE1588 Compliant Timestamping**

#### 7.4.3.2.1 Timer Control Register (TMR\_CTRL)

This register is used to reset, configure, and initialize the 1588 module precision timer clock. [Figure 7-3](#) describes the definition for the TMR\_CTRL register.



**Figure 7-3. TMR\_CTRL Register Definition**

Table 7-7 describes the fields of the TMR\_CTRL register.

**Table 7-7. TMR\_CTRL Register Field Descriptions**

Bits	Name	Description
0	ALM1P	Alarm1 output polarity 0 active high output 1 active low output
1	ALM2P	Alarm2 output polarity 0 active high output 1 active low output
2	—	Reserved
3	FS	FIPER start indication 0 Fiper is enabled through timer enable 1 Fiper is enabled through timer enable and alarm indication.
4	PP1L	Fiper1 pulse loopback mode enabled. 0 Trigger1 input is based upon normal external trigger input. 1 Fiper1 pulse is looped back into Trigger1 input.
5	PP2L	Fiper2 pulse loopback mode enabled. 0 Trigger2 input is based upon normal external trigger input. 1 Fiper2 pulse is looped back into Trigger2 input.
6–15	TCLK_PERIOD	1588 timer reference clock period. The value programmed by user in this field ideally represents the frequency of the phase adjusted clock period. The timer clock counter will increment by TCLK_PERIOD every time the accumulator register overflows. When this field is being used to track time, this clock period must be equal to or larger than the clock period of the timer reference clock. For applications where user does not want the clock period to be added, they can program this field to 1 to count the clock ticks. This field defaulted to 1 to count overflow ticks. Note: For customers sensitive to consistency in time-stamp values, it is recommended the selected frequency is fixed at a period that is an integer multiple value.
16	—	Reserved
17	FRD	FIPER Realignment Disable 0 FIPER Realignment is enabled. 1 FIPER Realignment is disabled.
18	SLV	Timer Master/Slave mode. Determines Master timer counter to be used in a system of multiple Timer blocks effectively synchronizing their time. A standalone Timer block should be set as a Master. When a block is set as a Slave, there can only be one Master timer counter within the system.  When configured as a Slave, both Master and Slave can only operate in same timer clock domain. Therefore one can not select one source, TMR_CTRL[CKSEL], of the timer clock for the Master and a different clock source for the Slave. Each separate Master can select different clock sources. 0 The timer module is in master mode. Current time is calculated by adding TMR_CNT_H/L and TMR_OFFSET_H/L 1 The timer module is in slave mode. Current time is received from a separate master timer module and the values of TMR_CNT_H/L and TMR_OFFSET_H/L are ignored.
19-21	—	Reserved
22	ETEP2	External trigger 1 edge polarity 0 Time stamp on the rising edge of the external trigger 1 Time stamp on the falling edge of the external trigger

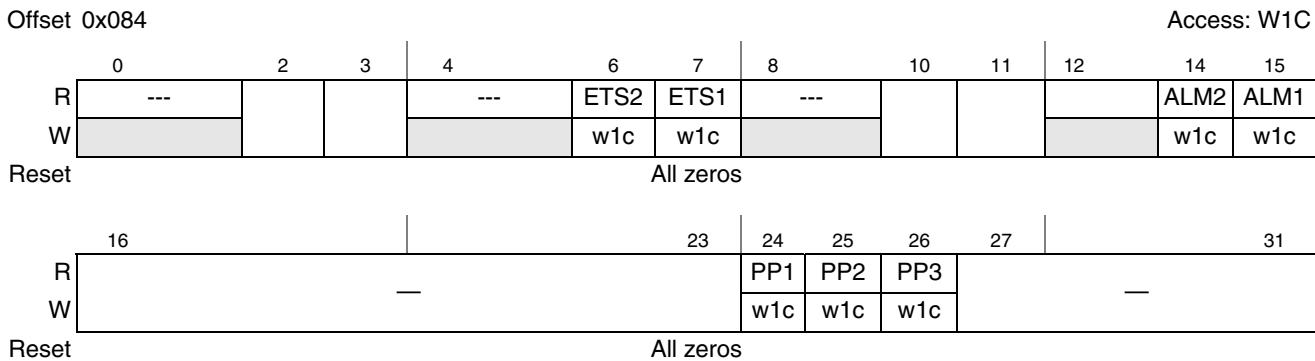
**Table 7-7. TMR\_CTRL Register Field Descriptions (continued)**

Bits	Name	Description
23	ETEP1	External trigger 2 edge polarity 0 Time stamp on the rising edge of the external trigger 1 Time stamp on the falling edge of the external trigger
24	COPH	Generated clock output phase. 0 non-inverted divided clock is output 1 inverted divided clock is output
25	CIPH	External oscillator input clock phase. 0 non-inverted frequency tuned timer input clock 1 inverted frequency tuned timer input clock
26	TMSR	Timer soft reset. When enabled, it resets all the timer registers and state machines for the 1588 module. 0 Normal operation 1 Place entire timer in reset except control and configuration registers. This bit is self-clearing. After TMSR is cleared, before attempting timer clock domain register accesses, the user will need to poll TMR_STAT[Timer Ref Clk active]. Refer to <a href="#">Table 7-20</a> and <a href="#">Table 7-21</a> .  User programmable registers are not reset by the soft reset e.g. TMR_CTRL, TMR_TEMASK, TMR_ADD, TMR_PRSC, TMROFF_H/L, TMR_ALARMn, and TMR_FIPERn. Note: Software must ensure all associated MACs with timestamping enabled are quiesced (not time-stamping packets) before asserting TMR_CTRL[TMSR]
27	—	Reserved
28	BYP	Bypass drift compensated clock 0 64-bit clock counter is incremented on the accumulator overflow 1 64-bit clock counter is directly driven from the external oscillator ignoring accumulator overflow
29	TE	1588 timer enable. If not enabled, all the timer registers and state machines are disabled. 0 timer not enabled 1 timer enabled and resume normal operation
30–31	CKSEL	1588 Timer reference clock source select. Default is 01. The selected clock is limited by the fastest associated MAC receive clock. The selected timer clock can not be slower than one seventh (1:7) of the MAC receive clock.  The results of timer clock domain register read/write are boundedly undefined if an inactive 1588 reference clock is selected. Refer to <a href="#">Table 7-20</a> for a list of timer clock domain registers.  After updating TMR_CTRL[CKSEL], user should poll TMR_STAT[RCD] or wait at least 3 timer clocks before reading timer related registers.  00 External high precision timer reference clock (TMR_1588_CLK) - maximum speed is limited to (ipg_clk*90%) 01 MAC system clock (ipg_clk) 10 reserved 11 External Input pin Real-Time Clock (RTC) oscillator - maximum speed is limited to (ipg_clk*90%)

#### 7.4.3.2.2 Timer Event Register (TMR\_TEVENT)

The IEEE 1588 timer implementation requires several interrupt event signals. The 1588 timer interrupt does not require any interrupt coalescing. Software may poll this register at any time to check for pending interrupts. If an event occurs and its corresponding enable bit is set in the event mask register (TEMASK),

the event also causes a hardware interrupt at the PIC. A bit in the timer event register is cleared by writing a 1 to that bit position. [Figure 7-4](#) describes the definition for the TMR\_TEVENT register.



**Figure 7-4. TMR\_TEVENT Register Definition**

[Table 7-8](#) describes the fields of the TMR\_TEVENT register fields for the timer.

**Table 7-8. TMR\_TEVENT Register Field Descriptions**

Bits	Name	Description
6	ETS2	External trigger 2 new timestamp sampled in TMR_ETTS2_H/L 0 external trigger new timestamp not sampled in TMR_ETTS2_H/L 1 external trigger new timestamp sampled in TMR_ETTS2_H/L
7	ETS1	External trigger 1 new timestamp sampled in TMR_ETTS1_H/L 0 external trigger new timestamp not sampled in TMR_ETTS1_H/L 1 external trigger new timestamp sampled in TMR_ETTS1_H/L
14	ALM2	Current time equaled alarm time register 2 0 alarm time has not been reached yet 1 alarm time has been reached
15	ALM1	Current time equaled alarm time register 1 0 alarm time has not been reached yet 1 alarm time has been reached
16–23	—	Reserved
24	PP1	Indicates that a periodic pulse has been generated based on FIPER1 register. 0 periodic pulse not generated 1 periodic pulse generated
25	PP2	Indicates that a periodic pulse has been generated based on FIPER2 register. 0 periodic pulse not generated 1 periodic pulse generated
26	PP3	Indicates that a periodic pulse has been generated based on FIPER3 register. 0 periodic pulse not generated 1 periodic pulse generated
27–31	—	Reserved

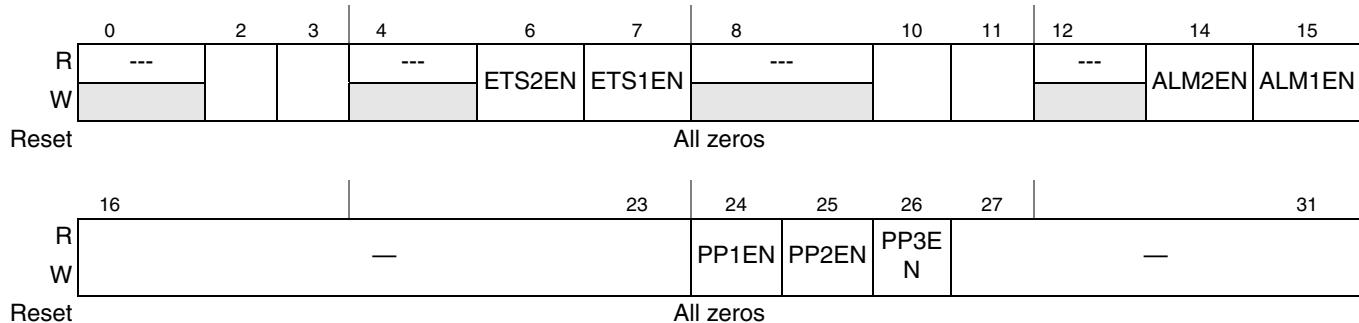
#### 7.4.3.2.3 Timer Event Mask Register (TMR\_TEMASK)

Timer event mask register. The event mask register provides control over which possible interrupt events in the TMR\_TEVENT register are permitted to participate in generating hardware interrupts to the PIC.

All implemented bits in this register are R/W and cleared upon a hardware reset. [Figure 7-5](#) describes the definition for the TMR\_TEMASK register.

Offset 0x088

Access: Read/Write

**Figure 7-5. TMR\_TEMASK Register Definition**

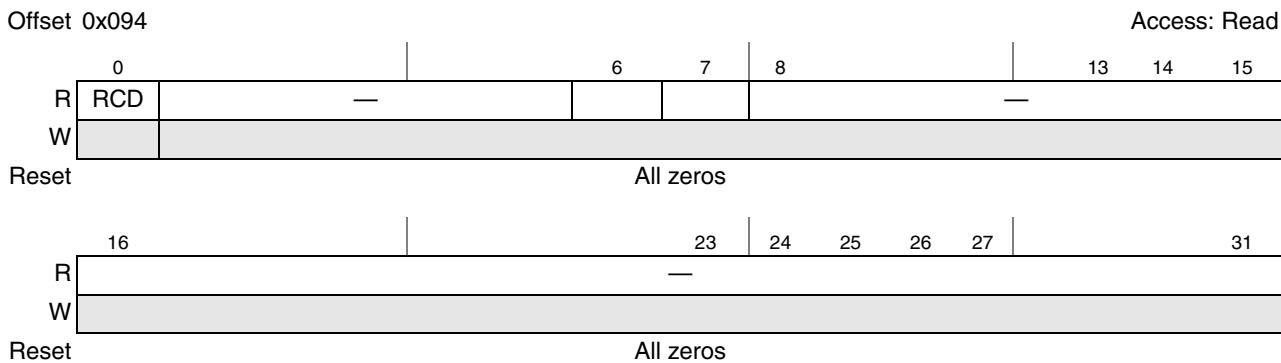
[Table 7-9](#) describes the fields of the TMR\_TEMASK register fields for the timer.

**Table 7-9. TMR\_TEMASK Register Field Descriptions**

Bits	Name	Description
6	ETS2EN	External trigger 2 new timestamp sample event enable
7	ETS1EN	External trigger 1 new timestamp sample event enable
14	ALM2EN	Timer ALM1 event enable
15	ALM1EN	Timer ALM2 event enable
16–23	—	Reserved
24	PP1EN	Periodic pulse event 1 enable
25	PP2EN	Periodic pulse event 2 enable
26	PP3EN	Periodic pulse event 3 enable
27–31	—	Reserved

#### 7.4.3.2.4 Timer Status Register (TMR\_STAT)

The timer TMR\_STAT register returns relevant dynamic timer status that does not generate an interrupt. This register is read only. [Figure 7-7](#) describes the definition for the TMR\_STAT register.

**Figure 7-6. TMR\_STAT Register Definition**

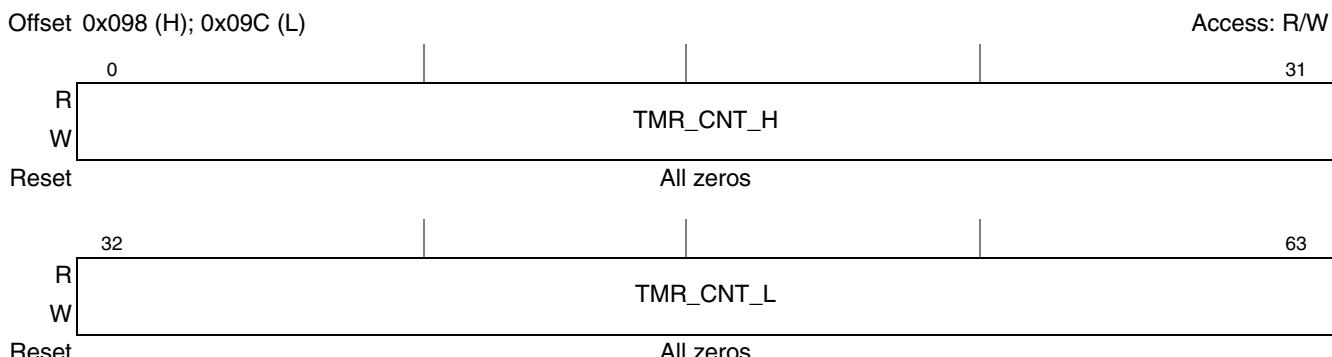
[Table 7-11](#) describes the fields of the TMR\_STAT register.

**Table 7-10. TMR\_STAT Register Field Descriptions**

Bits	Name	Description
0	RCD	Timer Reference Clock Detected. The selected timer clock, TMR_CTRL[CKSEL], is sensed as being active. When it is not active, register read/write accesses to timer clock domain registers will not be allowed. Refer to <a href="#">Table 7-20</a> and <a href="#">Table 7-21</a> . 0 Reference Clock has not been detected as active. Registers in timer clock domain are not allowed to be accessed; reads return 0, writes are ignored. 1 Reference Clock has been detected as active. Registers in timer clock domain are allowed to be accessed.

#### 7.4.3.2.5 Timer Counter Register (TMR\_CNT\_H/L)

If the timer module is in master mode, the timer register (TMR\_CNT\_H/L) represents accurate time in terms of clock ticks or in nano-seconds. Writes to these registers will override the previous time. This is a read/write register. [Figure 7-7](#) describes the definition for the TMR\_CNT\_H/L register. These registers are unused in slave mode.

**Figure 7-7. TMR\_CNT\_H Register Definition**

[Table 7-11](#) describes the fields of the TMR\_CNT\_H/L register.

**Table 7-11. TMR\_CNT\_H/L Register Field Descriptions**

Bits	Name	Description
0-63	TMR_CNT_H/L	<p>Timer counter register. In master mode (TMR_CTRL[TMR_SLV]=0), current time is calculated by adding TMROFF_H/L with the TMR_CNT_H/L counter. This register can be written through the register writes. Writes to the TMR_CNT_L register copies the written value into the shadow TMR_CNT_L register. Writes to the TMR_CNT_H register copies the values written into the shadow TMR_CNT_H register. Contents of the shadow registers are copied into the TMR_CNT_L and TMR_CNT_H registers following a write into the TMR_CNT_H register. Writes to these registers have precedence over the timer increment. The user must write to TMR_CNT_L register first.</p> <p>Reads from the TMR_CNT_L register copies the entire 64-bit clock time of the read enable into the TMR_CNT_H/L shadow registers. Read instruction from the TMR_CNT_H register reads the value stored in the TMR_CNT_H shadow register. The user must read the TMR_CNT_L register first to get correct 64-bit TMR_CNT_H/L counter values.</p> <p>This register value is not used by the timer module in slave mode (TMR_CTRL[TMR_SLV]=1).</p>

#### 7.4.3.2.6 Timer Drift Compensation Addend Register (TMR\_ADD)

Timer drift compensation addend register (TMR\_ADD) is used to hold timer frequency compensation value (FreqCompensationValue). The nominal frequency of the clock counter is determined by the FreqDivRatio and the clock frequency (FreqClock). This register is programmed with  $2^{32}/\text{FreqDivRatio}$ . Frequency division ratio (FreqDivRatio) is the ratio between the frequency of the oscillator (TimerOsc) and the desired clock frequency (NominalFreq). FreqDivRatio is a design constant chosen to be greater than 1.0001. The ADDEND value is added to the 32-bit accumulator register at every rising edge of the oscillator clock (TimerOsc). The clock counter is incremented at every carry pulse of the accumulator. Figure 7-8 describes the definition of the TMR\_ADD register.

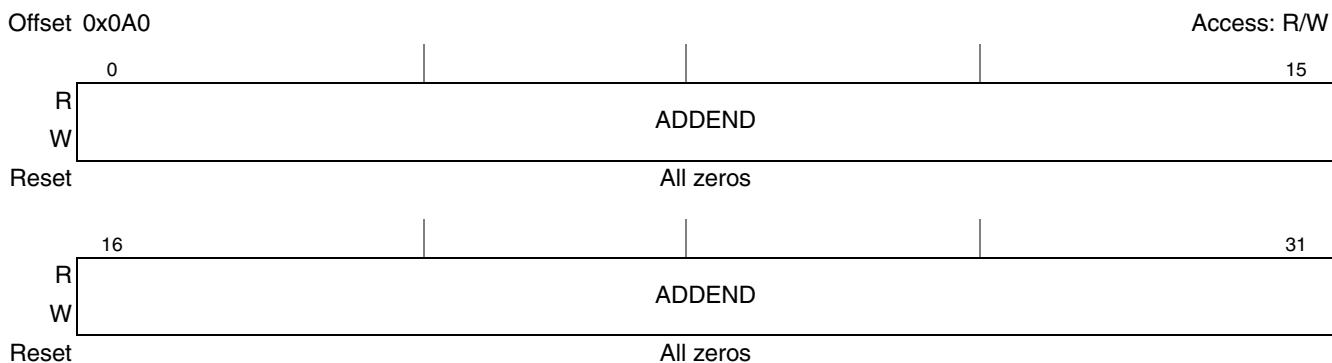
**Figure 7-8. TMR\_ADD Register Definition**

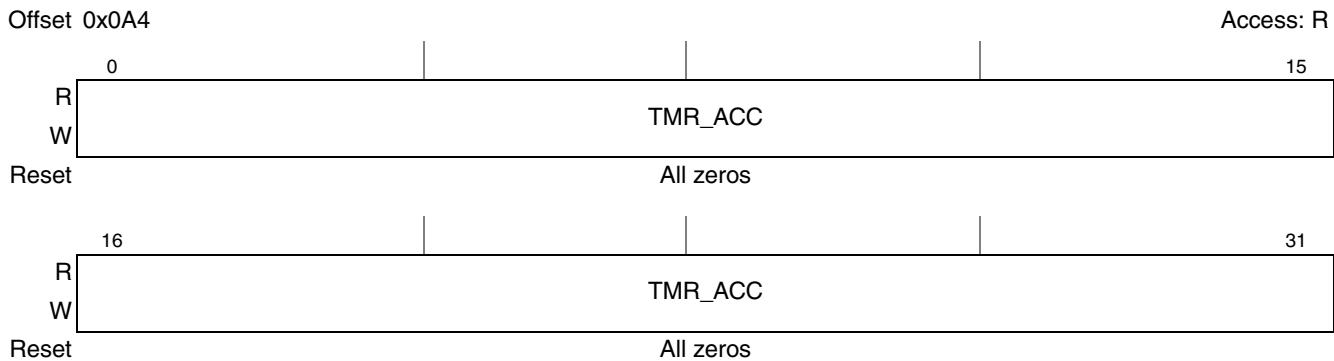
Table 7-12 describes the fields of the TMR\_ADD register fields for the timer.

**Table 7-12. TMR\_ADD Register Field Descriptions**

Bits	Name	Description
0–31	ADDEND	Timer drift compensation addend register value. It is programmed with a value of $2^{32}/\text{FreqDivRatio}$ . For example, TimerOsc = 50 MHz NominalFreq = 40 MHz FreqDivRatio = 1.25 ADDEND = $\text{ceil}(2^{32}/1.25) = 0xCCCC\_CCCD$

#### 7.4.3.2.7 Timer Accumulator Register (TMR\_ACC)

Timer accumulator register accumulates the value of the addend register into it. An overflow pulse of the accumulator is used to increment the timer clock by TMR\_CTRL[TCLK\_PERIOD]. This register is read only in normal operation. Figure 7-9 describes the definition of the TMR\_ACC register.



**Figure 7-9. TMR\_ACC Register Definition**

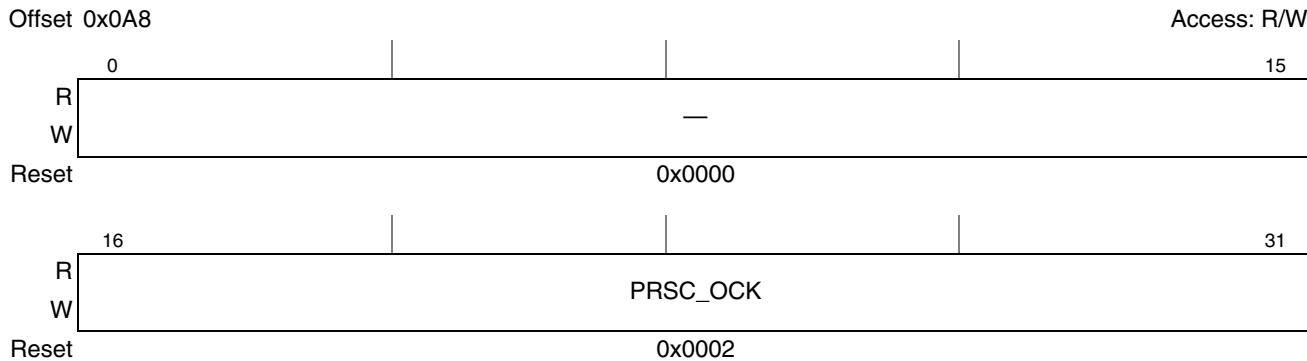
Table 7-13 describes the fields of the TMR\_ACC register.

**Table 7-13. TMR\_ACC Register Field Descriptions**

Bits	Name	Description
0–31	TMR_ACC	32-bit timer accumulator register

#### 7.4.3.2.8 Timer Prescale Register (TMR\_PRSC)

Timer generated output clock prescale register. It is used to adjust output clock frequency that is placed onto the TMR\_GCLK output pin. Figure 7-10 describes the definition for the TMR\_PRSC register.

**Figure 7-10. TMR\_PRSC Register Definition**

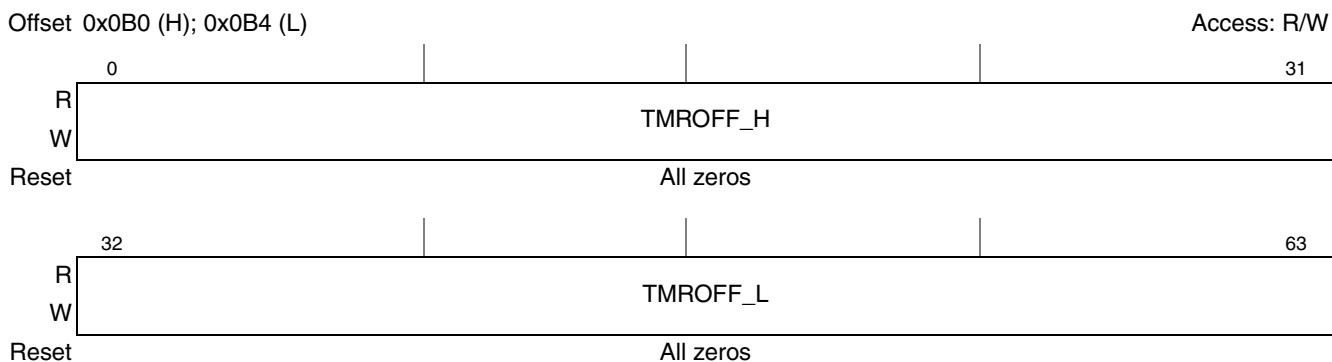
[Table 7-14](#) describes the fields of the TMR\_PRSC register.

**Table 7-14. TMR\_PRSC Register Field Descriptions**

Bits	Name	Description
0–15	—	Reserved
16–31	PRSC_OCK	Output clock division/prescale factor. Output clock is generated by dividing the timer input clock (which is the timer clock resulting from BYP configuration) by this number. Programmed value in this field must be greater than 1. Any value less than 2 is treated as 2.  Note: If TMR_PRSC is an odd value the FIPERn pulse width will vary.

#### 7.4.3.2.9 Timer Offset Register (TMROFF\_H/L)

In master mode, the timer offset register is used to adjust current time by adding its value to the clock counter. [Figure 7-11](#) describes the definition of the TMROFF\_H/L register.

**Figure 7-11. TMROFF\_H/L Register Definition**

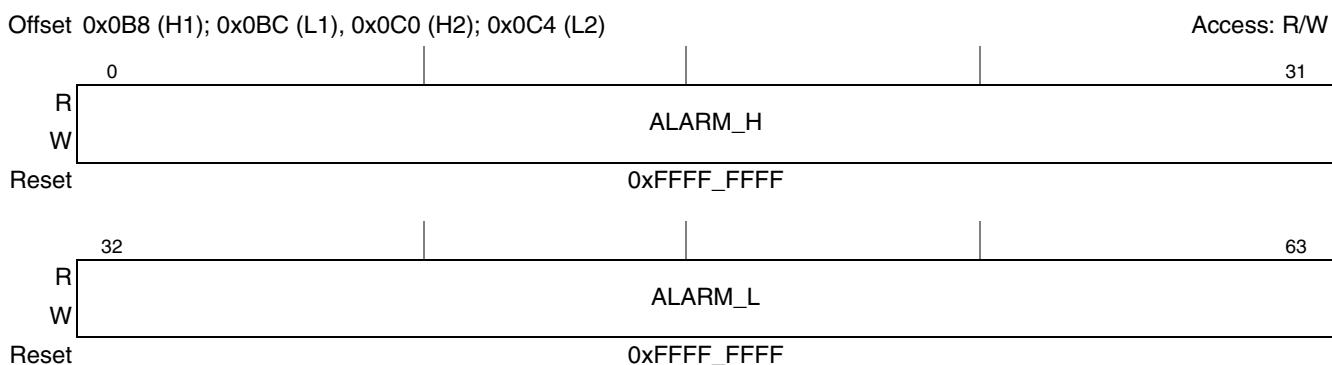
[Table 7-15](#) describes the fields of the TMROFF\_H/L register.

**Table 7-15. TMROFF\_H/L Register Field Descriptions**

Bits	Name	Description
0–63	TMROFF_H/L	Offset value of the clock counter. In master mode (TMR_CTRL[TMR_SLV]=0), current time in is calculated by adding TMROFF_H/L with the timer's counter TMR_CNT_H/L register. This register value is unused by the timer module in slave mode (TMR_CTRL[TMR_SLV]=1)

#### 7.4.3.2.10 Alarm Time Comparator Register (TMR\_ALARM1–2\_H/L)

Alarm time comparator register (TMR\_ALARMn\_H/L). This register holds alarm time for comparison with the current time counter. There are two of these registers for the 1588 timer module. [Figure 7-12](#) describes the definition for the TMR\_ALARMn\_H/L register.

**Figure 7-12. TMR\_ALARM1-2\_H/L Register Definition**

[Table 7-16](#) describes the fields of the TMR\_ALARMn\_H/L register.

**Table 7-16. TMR\_ALARMn\_H/L Register Field Descriptions**

Bits	Name	Description
0–63	ALARM_H/L	Alarm time comparator register. Alarm time comparator register. The corresponding alarm event in TMR_TEVENT is set when the current time counter becomes equal to or greater than the alarm time compare value in TMR_ALARMn_L/H. By default, once the timer is enabled, the Alarm is armed and will fire if it is not deactivated. Writing the TMR_ALARMn_L register deactivates the alarm event after it has fired. Writing the TMR_ALARMn_L followed by the TMR_ALARMn_H register rearms the alarm function with the new compare value. The value programmed in this register must be an integer multiple of TMR_CTRL[TCLK_PERIOD] in order to get correct result. This register is reset to all ones to avoid false alarm after reset. In FS mode the alarm trigger is used as an indication to the FIPER to start down counting. Only alarm 1 supports this mode. In FS mode, alarm polarity bit should be configured to 0 (rising edge).

#### 7.4.3.2.11 Timer Fixed Interval Period Register (TMR\_FIPER1–3)

Timer fixed interval period pulse generator register. It is used to generate periodic pulses. This register is reset with 0xFFFF\_FFFF. The down count register loads the value programmed in the fixed period interval (FIPER). FIPER register must be programmed before the timer is enabled. At every tick of the timer accumulator overflow, the counter decrements by the value of TMR\_CTRL[TCLK\_PERIOD]. It generates a pulse when the down counter value reaches zero. It reloads the down counter in the cycle following a pulse.

Should a user wish to use the TMR\_FIPER1 register to generate a 1 PPS event, the following setup should be used:

- With timer disabled, set TMR\_CTRL[FS]=1,
- Program TMR\_FIPER1 to a value that will generate a pulse every second,
- Program TMR\_ALARM1 to the correct time for the first PPS event, (with TMR\_ARLAM1\_H being written last to arm the ALARM)
- Enable the timer

The logic will then wait for TMR\_ALARM1 to expire before enabling the count down of TMR\_FIPER1. The end result will be that TMR\_FIPER1 will pulse every second after the original timer ALARM1 expired.

Note:

In the case where the PPS signals are required to be phased aligned to the prescale output clock, the alarm value should be configured to **1 clock period less** than the wanted value.

In order to keep tracking the prescale output clock, each time before enabling the FIPER, the user must reset the FIPER by writing a new value to the register. The FIPER\_value must be a multiple of the prescale register value and the tclk\_period value minus tclk\_period setting.

FIPER\_VALUE = (prescale\_value × tclk\_per × N) – tclk\_period, or

TMR\_FIPERN = (TMR\_PRSC × TMR\_CTRL[TCLK\_PERIOD] × N) - tclk\_period

where N is an integer number greater than or equal to 2

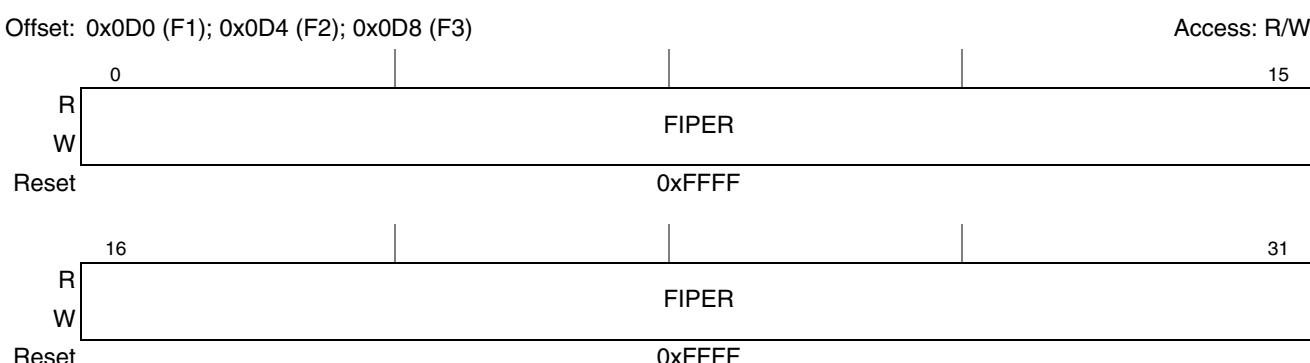
For example:

prescale = 10

tclk\_per = 9

The FIPER can get the following values: 171, 261, 351.....

[Figure 7-13](#) describes the definition for the TMR\_FIPER register.



[Figure 7-13. TMR\\_FIPERn Register Definition](#)

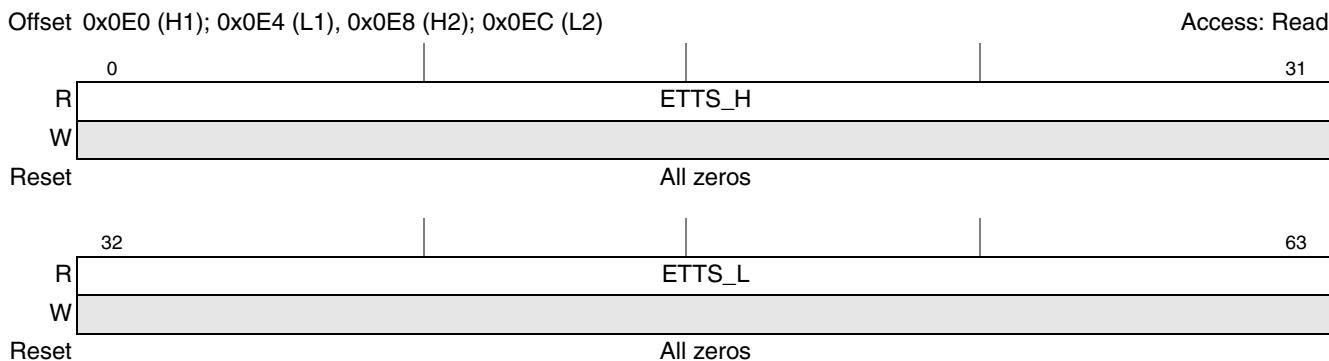
Table 7-17 describes the fields of the TMR\_FIPER register.

**Table 7-17. TMR\_FIPER Register Field Descriptions**

Bits	Name	Description
0–31	FIPER	<p>Fixed interval pulse period register. This field must be programmed to an integer multiple of TMR_CTRL[TCLK_PERIOD] value to ensure a period pulse being generated correctly.</p> <p>Output FIPER pulses are generated 4-5 timer reference clock after TMR_CTRL[TE]=1 and TMR_CTRL[FS]=0 or 1 timer reference clock after ALARM1 triggers.</p> <p>To disable, set TMR_CTRL[FS]=1 and disable ALARM1 (write to TMR_ALARM1_L).</p> <p>To realign the FIPER to a new phase, after FIPER pulsing is active:</p> <ul style="list-style-type: none"> <li>- Set TMR_CTRL[FS]=0, and TMR_CTRL[FRD]=0</li> <li>- Disarm TMR_ALARM1_H/L</li> <li>- Write to TMR_CNT_H/L and/or TMROFF_H/L</li> <li>- Then update TMR_ALARMn_H/L to a future value. The new TMR_ALARMn_H/L value should be calculated to realign the fixed interval pulse to the desired phase.</li> </ul> <p>In order for the FIPER pulse to rise at rising edge of the Prescaler output clock and to have a width a FIPER pulse width = 1*Prescaler output clock (ipp_tmr_gclk) period, one of the following equations must be followed:</p> <p>-- <math>TMR\_FIPERn = (TMR\_PRSC * TMR\_CTRL[TCLK\_PERIOD] * M) - TMR\_CTRL[TCLK\_PERIOD]</math>, where TMR_PRSC is an even number, where M is an integer number greater than or equal to 2.</p> <p>If TMR_PRSC is an odd number, the above equation will produce a FIPERn pulse of width 1.5*Prescaler output clock (ipp_tmr_gclk) period.</p>

#### 7.4.3.2.12 External Trigger Stamp Register (TMR\_ETTS1–2\_H/L)

General purpose external trigger -stamp register (TMR\_ETTSn\_H/L). This register holds time at the programmable edge of the external trigger. This register is read only. Figure 7-14 describes the definition for the TMR\_ETTSn\_H/L register.



**Figure 7-14. TMR\_ETTS1-2\_H/L Register Definition**

Table 7-18 describes the fields of the TMR\_ETTSn\_H/L register.

**Table 7-18. TMR\_ETTS1-2\_H Register Field Descriptions**

Bits	Name	Description
0–63	ETTS_H/L	Time stamp field at the programmable edge of the external trigger. For time-stamping of back-2-back trigger events, the trigger falling edge can be no closer than 5 timer clocks to next trigger rising edge.

## 7.5 Functional Description

### 7.5.1 1588 Timer Module Operation

This section describes the operation of the 1588 timer module.

#### 7.5.1.1 Initialization Sequence

This sections describes which registers are reset due to a hard or software reset and what registers the user must initialize prior to enabling the 1588 timer module.

##### 7.5.1.1.1 Hardware Controlled Initialization

A hard reset occurs when the system powers up. All 1588 timer module registers and control logic are reset to their default states after a hard reset has occurred.

##### 7.5.1.1.2 User Initialization

After the system has undergone a hard reset, software must initialize certain basic 1588 Timer module registers. The results of timer clock domain register read/write are boundedly undefined if an inactive 1588 reference clock is selected. Refer to [Table 7-20](#) for a list of timer clock domain registers. Other registers can also be initialized during this time, but they are optional and must be determined based on the requirements of the system. See [Table 7-3](#) for the register list. [Table 7-19](#) describes the minimum steps for register initialization.

**Table 7-19. Steps for Minimum Register Initialization**

Description
1. Set and clear TMR_CTRL[Soft_Reset]
2. Initialize TMR_CTRL including CKSEL and FS
3. Clear TMR_TEVENT
4. Initialize TMR_TMASK
5. Initialize TMR_ADD
6. Disarm TMR_ALARMn_H/L
7. Initialize TMR_CTRL including Timer Enable

### 7.5.1.2 Soft Reset and Reconfiguring Procedure

Following is a procedure to gracefully reset and reconfigure the timer module:

1. Set TMR\_CTRL[TMSR] bit and disable Timer, TMR\_CTRL[TE]=0
2. Wait for hardware to clear TMR\_CTRL[TMSR] bit
3. Wait minimum of 3 timer clocks + 3 platform clocks (ipg\_clk) before accessing any timer domain registers. Refer to [Table 7-20](#) and [Table 7-21](#).
4. Setup TMR\_CTRL register, including TCLK\_PERIOD and CKSEL (if update is desired)
5. Disarm the alarms with a write to TMR\_ALARM1\_L and TMR\_ALARM2\_L
6. Clear alarm, periodic pulse and external trigger event bits in TMR\_TEVENT
7. Set up TMR\_TEMASK for desired interrupts
8. Load TMR\_ADD with new value when in master mode and not in Bypass mode (if update is desired)
9. Optionally update TMR\_FIPERn, and TMR\_PRSC
10. Update TMR\_CNT\_H/L, and TMR\_OFF\_H/L to set the estimated PTP time (when in master mode)
11. Enable TMR\_CTRL[TE]

**Table 7-20. Timer Clock Domain Register List**

Register Name
TMR_CNT_H - timer counter high register
TMR_CNT_L - timer counter low register
TMR_ADD - Timer drift compensation addend register
TMR_ACC - Timer accumulator register
TMR_PRSC - timer prescale
TMROFF_H - Timer offset high
TMROFF_L - Timer offset low
TMR_ALARM1_H - Timer alarm 1 high register
TMR_ALARM1_L - Timer alarm 1 low register
TMR_ALARM2_H - Timer alarm 2 high register
TMR_ALARM2_L - Timer alarm 2 low register
TMR_FIPER1 - Timer fixed period interval
TMR_FIPER2 - Timer fixed period interval
TMR_FIPER3 - Timer fixed period interval
TMR_ETTS1_H - Time-stamp of general purpose external trigger
TMR_ETTS1_L - Time-stamp of general purpose external trigger
TMR_ETTS2_H - Time-stamp of general purpose external trigger
TMR_ETTS2_L - Time-stamp of general purpose external trigger

**Table 7-21. Platform Clock (ipg\_clk) Domain Register List**

Register Name
TMR_ID—Module ID and version register
TMR_ID2—Module ID and configuration register

**Table 7-21. Platform Clock (ipg\_clk) Domain Register List**

Register Name
TMR_CTRL - Timer control register
TMR_TEVENT - time-stamp event register
TMR_TEMASK - Timer event mask register
TMR_STAT - Timer Status

### 7.5.1.3 Interrupt Handling

The following describes what usually occurs within a timer module interrupt handler:

- If an interrupt occurs, read TMR\_TEVENT to determine interrupt sources. TMR\_TEVENT bits to be handled in this interrupt handler are normally cleared at this time.
- Continue normal execution.

Table 7-22 lists all the interrupts related to the 1588 timer module.

**Table 7-22. 1588 Timer Module Interrupts**

Interrupt	Description	Action taken by the 1588 Timer module
ETSn	External trigger new time-stamp sampled in TMR_ETTSn_H/L.	None.
ALMn	Current time equaled the programmed alarm time.	The alarm output pin will be continuously driven asserted until TMR_ALARMn is disarmed.
PPn	Periodic pulse has been generated.	The FIPERn will continue pulsing until disabled.

## 7.6 Initialization/Application Information

### 7.6.1 Hardware Assist for IEEE 1588 Compliant Timestamping

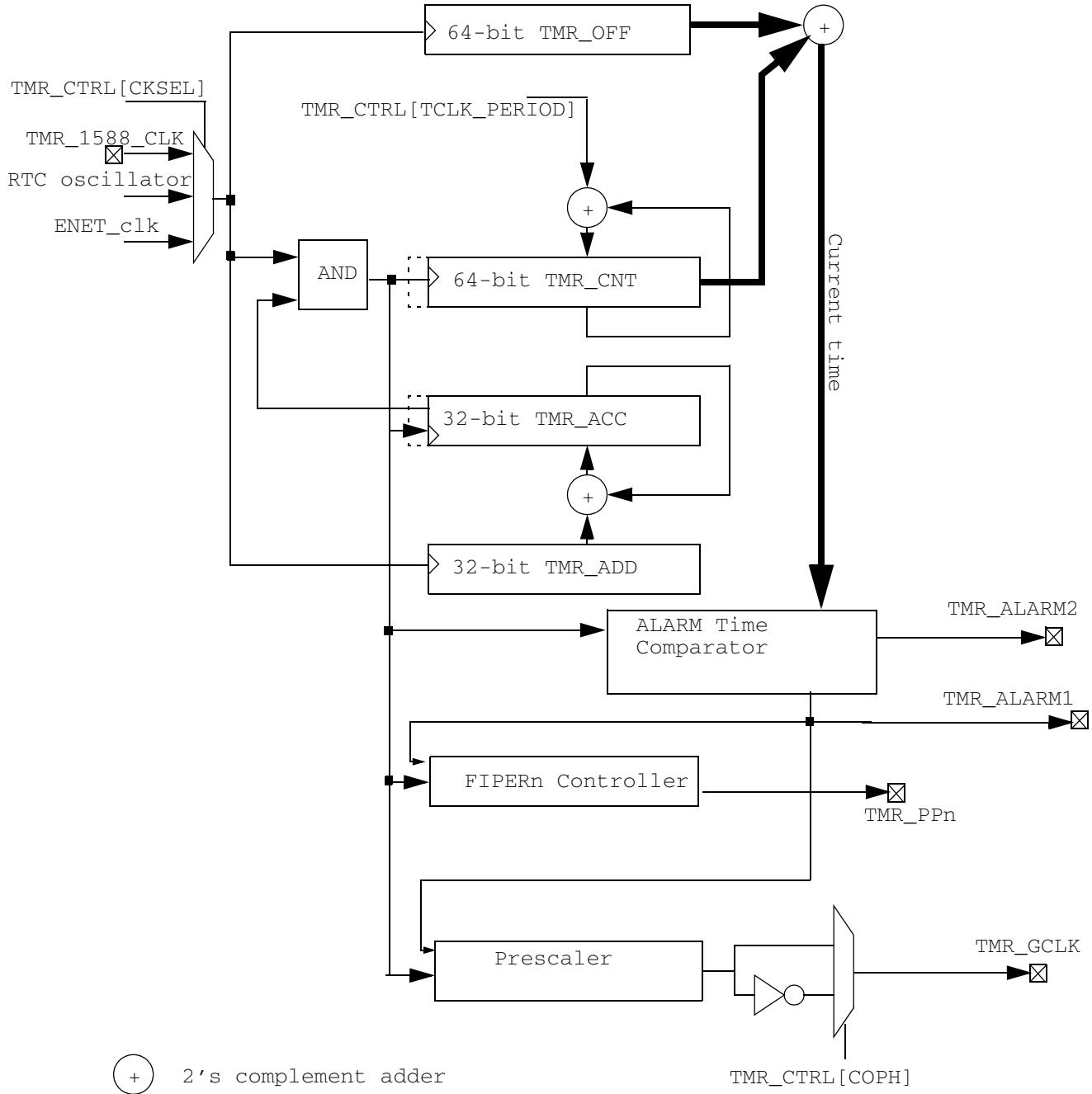
IEEE 1588 specifies a mechanism for synchronizing multiple nodes to the same master clock. The 1588 timer module utilizes a timer clock module to support the IEEE 1588 standard. The following sections describes the features, programming model, and application information.

#### 7.6.1.1 Features

- 64-bit free running timer running from the external oscillator or internal clock
- Programmable timer oscillator clock selection
- Self-correcting precision timer with nano-second resolution
- 64-bit Timer offset register for current time adjustment
- Time stamp capture on two general purpose external trigger through GPIO connection
  - Maskable interrupts on GPIO timestamp trigger
  - Programmable polarity of external trigger (GPIO) edge

- Two 64-bit alarm registers for future time comparison
  - Maskable interrupts on alarm
- Three programmable timer output pulse period phase aligned with 1588 timer clock
  - Maskable interrupts associated with each pulse
- Automatic FIPER pulse realignment in response to timer adjustments
- Maskable timer interrupt event register
- Phase aligned adjustable (divide by N) clock output
- Supports all Ethernet modes supported by the attached Ethernet Controller
- Independent of master and slave mode. Master and slave modes are user programmable.
- Supports timestamp of nano-second resolution

Figure 7-15 shows the general logic used to establish and maintain current time for use in 1588 timestamping.



**Figure 7-15. 1588 Timer Block Diagram**

### 7.6.1.2 Time-Stamping of Packets

The 1588 timer module provides a 64-bit current time value in the selected timer clock domain to client 1588 modules internal to Ethernet controllers. Each client 1588 will take in the current time input and use it's value as the time-stamp for transmitted packets. The 64-bit current time is controlled by timer enable, TMR\_CTRL[TE], selected clock, TMR\_CTRL[CKSEL], bypass mode, TMR\_CTRL[BYP], the timer clock period, TMR\_CTRL[TCLK\_PERIOD], and the ADDEND value, TMR\_ADD registers.

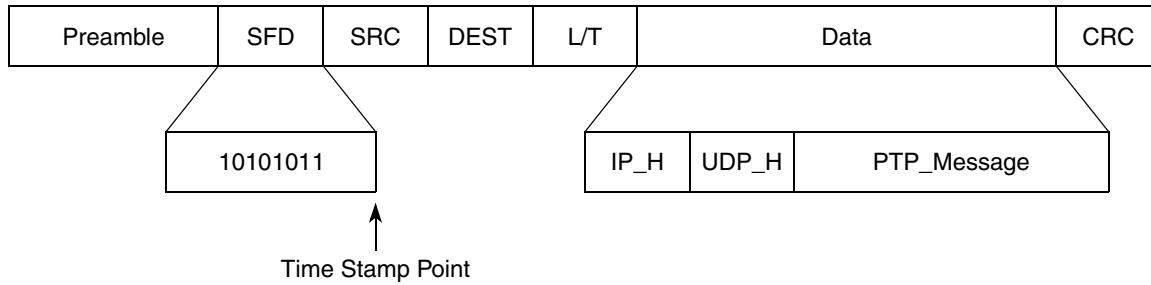
### 7.6.1.3 PTP Packets

PTP packets are embedded in UDP payload with special IP source and destination address and special source and destination ports numbers. Special fields of interest of a PTP packet are listed in [Table 7-23](#).

**Table 7-23. PTP Payload Special Fields**

Layer	Octet (Offset from the SFD)	Field	Value	Comments
Ethernet	12-13	Length/Packet	0x0800	IPv4
IP header	22	Time to live	0x00	Must be 0
IP header	23	IP Protocol	0x11	UDP
IP header	26-29	Source IP Address IANA defines 4 multicast address for the PTP packet		
IP header	30-33	Destination IP Address IANA defines 4 multicast address for the PTP packet	224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132	DefaultPTPdomain AlternatePTPdomain1 AlternatePTPdomain2 AlternatePTPdomain3
UDP header	34-35	Source port number		
UDP header	36-37	Destination port number	319 320	EventPort GeneralPort
UDP data	74	Control	0x0 0x1 0x2 0x3 0x4	Sync Delay_req Follow_up Delay_resp Management

A representation of the PTP packet is shown in [Figure 7-16](#).

**Figure 7-16. PTP Packet Format**

#### 7.6.1.4 Time-Stamp For External Trigger

When 1588 timer module is enabled, TMR\_CTRL[TE]=1, the programmed trigger edge, TMR\_CTRL[ETEPn] will cause the appropriate TMR\_ETSn\_H/L register to be updated with a new time-stamp value. The external trigger is synchronized to the system clock domain,. The time-stamp values is therefore inaccurate by 2 to 3 timer clocks. The TMR\_TEVENT[ETSn] will be set 2-3 system clocks later.

#### 7.6.1.5 User Time Adjustments

The 1588 timer counter once started will increment tracking time per each adjusted clock edge. The user should be aware of several time delays in setting up the 1588, and delays in the time-stamp value to the actual time a frame is transmitted or received so it can make the appropriate adjustments.

##### 7.6.1.5.1 1588 Timer Counter Setup Delays

The TMR\_CNT\_L and TMR\_CNT\_H registers are normally written to set an initial point in time. When the registers are written, once the timer is enabled, it takes 6 timer reference clock cycles before time moves forward, incrementing by TMR\_CTRL[TCLK\_PERIOD].

The counter, TMR\_CNT\_L/H, will not increment until the timer counter is enabled which can be set two ways, TMR\_CTRL[TE]=1 & TMR\_CTRL[Bypass]=1, or TMR\_CTRL[TE]=1 & TMR\_CTRL[Bypass]=0 and Accumulator overflow.

Once timer counter is enabled, it takes 6 clocks before the tclk\_period is first added. After the initial 6 cycle setup delay, time will increment on each subsequent timer phase adjusted clock, or in bypass mode, each timer reference clock positive edge.

##### 7.6.1.5.2 1588 Timer Alarm Delays

When the TMR\_ALARMn\_H/L is written to arm the alarm, it will take 5-6 timer reference clock cycles before it takes affect. Thus, the user must make sure the alarm is set to a future time that is not within 6 clock cycles of the current time.

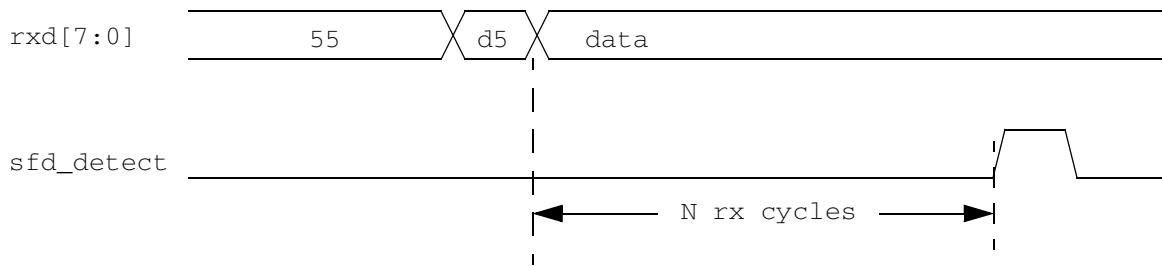
### 7.6.1.5.3 Time-Stamp Delays for Supported Modes

1588 Ethernet modes supported along with the pin to time-stamp delay logic path details are shown in [Table 7-24](#).

**Table 7-24. Ethernet Mode and 1588 Support**

Mode	Delay between Pin and Time-stamp Detection w.r.t. MII
<b>RMII</b>	18 Rx clock cycles + 2-3 timer clock cycles for synchronization
<b>GMII</b>	16 Rx clock cycles + 2-3 timer clock cycles for synchronization
<b>RGMII</b>	18 Rx clock cycles + 2-3 timer clock cycles for synchronization
<b>SGMII</b>	18 Rx clock cycles + 2-3 timer clock cycles for synchronization

Illustration of delays is shown in [Figure 7-17](#).



**Figure 7-17. SFD Delay From Pin to Detection**

# **Appendix A**

## **Revision History**

Because this is the initial publication of the document, there are no changes.

