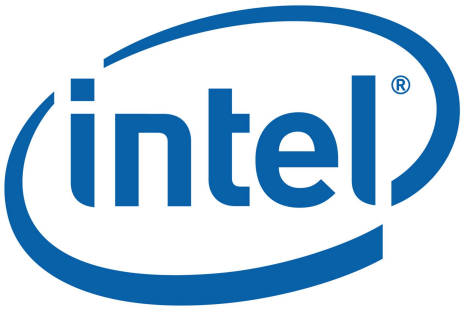


Intel® Ethernet Switch FM4000

24-Port 10G Ethernet L2/L3/L4 Switch/Router

Datasheet

Networking Division (ND)

Revision 3.2
March 2014

LEGAL

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A “Mission Critical Application” is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked “reserved” or “undefined”. Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm.

Intel and Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* Other names and brands may be claimed as the property of others.

Revision History

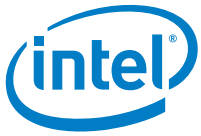
Revision	Date	Comments
3.2	March 26, 2014	General update.
3.1	January 23, 2014	General Update.
3.0	November 25, 2013	General update.
2.6	April 22, 2013	Added note preceding table in Section 16.5.
2.5	April 1, 2013	Changed part number FM4104 to FM4105.
2.4	April 7, 2011	General update (see revision document).
2.3	December 6, 2010	General update (see revision document).
2.2	January 5, 2010	General update (see revision document).
2.1	May 14, 2009	General update (see revision document).
2.0	February 10, 2009	General update (see revision document).
1.4	December 4, 2008	General update (see revision document).
1.31	October 7, 2008	General update (see revision document).
1.2	July 17, 2008	General update.
1.15	July 11, 2008	Formatting updates only.
1.1	July 1, 2008	General update.
1.0	October 30, 2007	Update to Preliminary Datasheet.
0.9	October 3, 2007	General update.
0.55	September 30, 2007	General update (intermediate release).
0.50	April 23, 2007	General update.
0.40	February 19, 2007	First release.

NOTE: ***This page intentionally left blank.***

Contents

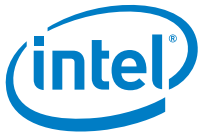
1.0 Introduction 11
1.1 Scope 11
1.2 Product Applicability 11
1.3 Document Organization 12
1.4 Definitions 13
2.0 Architecture Overview 15
2.1 Introduction to Architecture 15
2.2 Frame Parsing 18
2.2.1 Layer 2 19
2.2.2 Layer 3 20
2.2.3 Deep Packet Inspection for IP frames 22
2.3 Frame Processing Pipeline Overview 22
2.3.1 Frame Header 24
2.3.2 Frame Trailer 25
2.3.3 GloRT 26
2.3.4 Ethernet Port Logic 26
2.3.5 Parsing 26
2.3.6 FFU (Filtering and Forwarding Unit) 27
2.3.7 ARP Unit 27
2.3.8 Layer 2 Lookup Unit 28
2.3.9 GloRT Lookup 28
2.3.10 Triggers and Link Aggregation Units 28
2.3.11 Congestion Management 29
2.3.12 Packet Queues 29
2.3.13 Packet Replicator 29
2.3.14 Egress Scheduler and Egress EPL 30
2.4 Intel ISL Tag 30
2.4.1 Frame Type (FTYPE) 31
2.4.2 VLAN Type (VTYPE) and Management Type (MTYPE) 33
2.4.3 Switch Priority (SWPRI) 33
2.4.4 User Bits (USER) 34
2.4.5 VLAN priority (VPRI) 34
2.4.6 VLAN CFI/DEI bit (VCFI) 34
2.4.7 VLAN ID (VID) 34
2.4.8 Source GloRT (SGLORT) 34
2.4.9 Destination GloRT (DGLORT) 35
2.5 F96 tag 35
2.6 Action Codes 35
3.0 Pin Descriptions 39
3.1 Pin Overview 39
3.2 Signal Name Convention 40
3.3 Detailed Pin Descriptions 41
3.3.1 Ethernet Port Pins 41
3.3.2 Power Pins 41
3.3.3 Bus Interface Pins 42
3.3.4 DMA Interface Pins 43
3.3.5 GPIOs and Strapping Pins 43

3.3.6 I2C Pins 45
3.3.7 MDIO Pins 45
3.3.8 LED Pins 45
3.3.9 JTAG Pins 46
3.3.10 Miscellaneous Pins 46
4.0 Ethernet Port Logic (EPL) 47
4.1 Overview 47
4.2 SERDES 48
4.2.1 PLL and Reference Frequency 48
4.2.2 Adjusting Drive Strength and Pre-emphasis 49
4.2.3 Configuration of Polarity and Lane Ordering 49
4.2.4 Testing Facilities 50
4.2.5 Status and Interrupts 52
4.2.6 Auto-negotiation 52
4.2.7 Clause 73 53
4.2.8 Clause 37 56
4.3 Physical Coding Sub-layer (PCS) 58
4.3.1 Frame Format 58
4.3.2 Local and Remote Faults 59
4.3.3 Messaging 59
4.3.4 Balancing IFGs 60
4.4 MAC 60
4.4.1 Frame Parsing 61
4.4.2 Parser Errors 61
4.4.3 Deep Packet Inspection for IP Frames 62
4.4.4 Deep Packet Inspection for Non-IP Frames. 64
4.4.5 EPL Reset 64
5.0 Chip Management 67
5.1 Overview 67
5.2 Boot Controller and Chip Reset 70
5.3 EEPROM Boot Format 73
5.4 Interrupt Controller 74
5.5 I2C Controller 77
5.6 MDIO Controller 82
5.7 GPIO Controller 84
5.8 Frame Handler PLL 85
5.9 Frame Timeout 86
5.10 LED 87
5.11 CPU Interface Controller 88
5.12 CPU Bus Interface 89
5.12.1 Using DATA_HOLD 90
5.12.2 Atomic Accesses 91
5.12.3 Little and Big Endian Support 92
5.13 CPU Frame Transfer 94
5.13.1 Packet Transmission 95
5.13.2 Packet Reception 96
5.13.2.1 Little Endian Packet Transfer 97
5.13.2.2 Big Endian Packet Transfer 97
5.13.3 Packet Transfer DMA Timing 98
5.14 Packet Trapping, Logging and Mirroring 99



5.15 SPI Interface 101
5.15.1 SPI (Serial Peripheral Interface) Controller 101
5.16 In-Band Management 103
5.16.1 Overview 103
5.16.1.1 Management By an Attached CPU 104
5.16.1.2 Management By a Remote CPU 104
5.16.1.3 Basic FIBM Topology 104
5.16.1.4 Other FIBM Topologies 105
5.16.2 Management Switch Bridge 106
5.16.2.1 Valid Frames 106
5.16.3 FIBM Frames 107
5.16.3.1 FIBM Frame Format 107
5.16.3.2 ISL Tag 108
5.16.3.3 CRC Errors 108
5.16.4 Reliable FIBM Communication 109
5.16.4.1 Sequence Numbers 109
5.16.4.2 Scratch Registers 109
5.16.4.3 Reading and Writing Idempotent Registers 109
5.16.4.4 Writing Non-Idempotent Registers 109
5.16.4.5 Reading Non-Idempotent Registers 110
5.16.4.6 Interrupts 110
5.16.4.7 Timing 110
5.16.5 The FIBM Payload Format 110
5.16.5.1 FIBM Request Frame Payload Format 110
5.16.5.2 FIBM Response Frame Payload Format 111
5.16.5.3 Minimum FIBM Frame Payload Length 112
5.16.5.4 Maximum FIBM Frame Payload Length 112
5.16.5.5 FIBM Header Words 112
5.16.6 Interrupts and RESET 113
5.16.6.1 General Interrupt Handling 113
5.16.6.2 Disabling Interrupts 114
5.16.6.3 RESET Initiated by a Remote CPU 114
5.16.6.4 Fatal Interrupts and Self Initiated RESET 114
5.16.6.5 Boot After RESET 115
5.17 Counter Rate Monitoring 115
5.17.1 Per-Monitor Configuration 116
5.17.2 Per-Monitor State 116
5.17.3 General Configuration 117
5.18 JTAG Interface 117
5.18.1 Tap Controller 118
5.18.2 Instruction Register 118
5.18.3 Bypass Register 118
5.18.4 JTAG Scan Chain 118
6.0 Filtering and Forwarding Unit (FFU) 119
6.1 Overview 119
6.2 Frame Header Mapper 121
6.3 Slice Activation and Overloading 122
6.4 Search Key Selection 124
6.5 Slice Cascading 126
6.6 ACL Mappings Examples 127

6.7 Ingress Action 128
6.8 Egress ACLs 131
7.0 Routing 133
7.1 Overview 133
7.1.1 Multi-Chip Routing 134
7.2 ARP Unit 135
7.3 ARP_USED Table 137
7.4 IP Traps & Logs 137
7.4.1 Trapping of IGMP Frames 137
7.4.2 Trapping of IP Frames with Options 137
7.4.3 Trapping and Logging of Frames with TTL=1 or TTL=0 138
7.4.4 Trapping of Frames That Exceed MTU Size 138
7.4.5 Logging Redirectable IP Unicast Frames 139
7.4.6 Logging Routable IP Unicast Frames Received as Layer 2 Multicast 139
8.0 Layer 2 Lookup 141
8.1 Overview 141
8.2 VLANs 143
8.3 MAC Address Table 145
8.4 Layer 2 Lookup Flow 146
8.5 MA Table Lookup, Learning and Aging 147
8.6 MA Table Management 148
8.6.1 Direct Table Access 148
8.6.2 MAC Table Hardware-Accelerated Purging 149
8.6.3 MAC Table Change Notification FIFO 149
8.6.4 MA_TCN_FIFO Overflow Protection 152
8.6.5 MA_TCN_FIFO Interrupts 152
8.7 MAC Address Security 153
8.8 Layer 2 Protocol Traps 154
8.9 IEEE 802.1x – Port Access Control 154
9.0 Port Mapping and Packet Replication 157
9.1 Overview of Port Mapping Unit 157
9.2 Loopback Suppression 159
9.3 Link Aggregation 160
9.3.1 Link Aggregation Glorts 160
9.3.2 Filtering and Pruning 161
9.3.2.1 Example A – LAG within One Switch 162
9.3.2.2 Example B – LAG within a Two-level Fat Tree 163
9.4 IP Multicasting 167
9.4.1 Getting a GloRT 167
9.4.2 Usage of Destination Mask and IP Multicast Table 168
9.4.3 Configuring MTable 169
10.0 Frame Hashing 171
10.1 Overview 171
10.2 Layer 3/4 Hash 172
10.3 Layer 2/3/4 Hash 174
10.4 FM2000 Compatibility 176
11.0 Triggers 177
11.1 Overview 177
11.2 Trigger Match Conditions 177



11.2.1 Configurable Trigger Precedence 180
11.2.2 Random Matching 180
11.3 Trigger Actions 181
11.3.1 Using Triggers to Undrop Frames 183
11.3.2 Rate Limiting 184
11.3.3 Trigger Action Resolution 184
11.4 Trigger Counters and Interrupts 185
12.0 QoS and Congestion Management 187
12.1 Overview 187
12.2 Processing of Frames with Trailing Errors 188
12.3 Differentiated Services (DiffServ) 189
12.4 Processing Priority 189
12.5 Changing Priority 191
12.6 Transmitting Priority 191
12.7 Policing 192
12.8 Monitoring Memory 195
12.9 Flow Control and Rate Limiters 200
12.10 Congestion Notification 204
12.10.1 Virtual Output Queues Congestion Notifications 205
12.10.1.1 VCN Frame Reaction 207
12.10.2 Class-Based Pause Frames 207
12.10.3 Fractional Congestion Notifications 208
12.10.3.1 FCN Rate Reaction 210
12.11 Backward Congestion Notification 211
12.11.1 BCN Congestion Point 215
12.11.1.1 Sampling 215
12.11.1.2 BCN Frame Generation 216
12.11.2 BCN Reaction Point 216
12.11.2.1 BCN Frame Reception 216
12.11.2.2 Rate Limiter Tagging 217
12.12 Queue Delay Measurement 217
13.0 Egress Scheduling and Shaping 219
13.1 Introduction 219
13.2 Definition of terms 220
13.3 Scheduling Algorithm 221
13.3.1 Groups and Sets of Traffic Classes 221
13.3.2 Group Eligibility 222
13.3.3 Class Selection 223
13.3.4 Algorithm Notes 223
13.4 Scheduler Implementation 224
13.4.1 Deficit Round-Robin 224
13.4.2 Bandwidth Shaping 226
13.5 Configuration 227
13.6 Egress Scheduling Examples 227
13.6.1 Strict Priority 227
13.6.2 Weight Controlled Priority Queuing 228
13.6.3 Mixed Strict and Round-Robin Queues 228
13.6.4 Nested Strict Prioritization 229
13.6.5 Deficit Round-Robin with Maximum Bandwidth Limits 229
14.0 Statistics and Monitoring 231

14.1 Frame Counters 231
14.2 Frame Monitoring 232
15.0 Electrical Specification 233
15.1 Absolute Maximum Ratings 233
15.2 Recommended Operating Conditions 233
15.3 Power Supply Sequencing 234
15.4 DC Characteristics of LVTTL PADs 235
15.5 AC Timing Specifications 238
15.5.1 CPU Interface, General Timing Requirements 241
15.5.2 MDIO Interface, General Timing Requirements 242
15.5.3 JTAG Interface 243
16.0 Mechanical Specification 245
16.1 1433-Ball 40mm Package Dimensions 246
16.2 897-Ball 32mm Package Dimensions 248
16.3 529-Ball 25mm Package Dimensions 250
16.4 Heat Sinking 251
16.4.1 Heat Sink Mounting Pressure 252
16.5 Pin Locations 253

1.0 Introduction

1.1 Scope

This functional specification is the basis for the data sheet for the Intel® Ethernet Switch FM4000 series devices, and provides information on the significantly-enhanced feature set over the FM2000 series in the areas of routing, access control lists, congestion management, network scaling, and management.

1.2 Product Applicability

The FM4000 represents a family of products with various port configurations and package sizes, which are listed in Table 1-1:

Table 1-1 FM4000 Products

Part Number	10 GbE Ports	2.5 GbE/1 GbE Ports	Package Type
FM4410	8	10	529-ball
FM4105	2	16	897-ball
FM4112	8	16	897-ball
FM4212	12	0	1433-ball
FM4224	24	0	1433-ball

This document pertains to all variants of the FM4000 platform, although most references are specific to the 24-port 10 GbE version of the device. The part marking and number conventions for Intel Ethernet Switch Family devices are defined as follows:

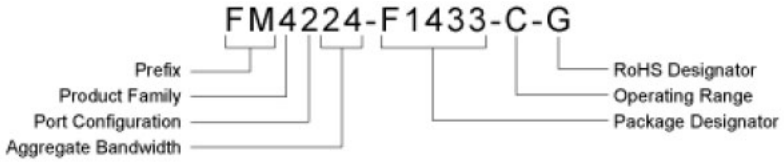


Figure 1-1 Intel Ethernet Switch Family Part Number Convention

Key:

- **Prefix** — **"**FM**"** identifies the device as an Intel Ethernet Switch Family product.
- **Product Family** — Conveys information about the general capabilities of the device, as follows:

  2 = FM2000 (L2)

  3 = FM4000 with L2+ features

  4 = FM4000 with full multi-layer feature set.
- **Port Configuration** — Conveys information about the configuration of the ports on the device, as follows:

  1 = Mostly single-SerDes interfaces (1 GbE, 2.5 GbE operation)

  2 = Mostly quad-SerDes interfaces (10 GbE operation)
- **Aggregate Bandwidth** — Identifies the approximate maximum bandwidth of the device configuration, calculated as:

  ```
  ((n x 10 GbE interfaces) + (m x 2.5 GbE interfaces))/10.
  ```

  Example:

  ```
  ((4 x 10 GbE) + (12 x 2.5 GbE))/10 = 07.
  ```
- **Package Designator** — Optional field. Identifies the package type and ball count of the device, as follows:

  B = Wire-bond BGA

  F = Flip-chip BGA
- **Operating Range** — Identifies the operating conditions for which the device is certified to operate in the following case temperature ranges:

  C = 0 to +85

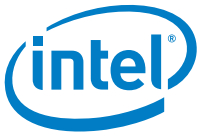
  E = 0 to +105

  I = -40 to +115
- **RoHS Designator** — The presence of a "-G" means that the device is complaint with the RoHS requirements for restrictions on the use of hazardous substances. Compliance is via exemption #15 in the RoHS Directive Annex, which allows for the use of Pb (lead) in the solder bumps used for die attach in flip-chip packages. –G parts have lead-free solder balls on the exterior of the package for PC board die attach. Note that the non-RoHS compliant package meets the RoHS limits for the other five substances, but contains Pb in the external solder balls, which is not allowed by the RoHS directive, and in the solder bumps for die attach. This is often referred to as RoHS 5-of-6 compliant.

1.3 Document Organization

This document is organized as follows:

- Section 1.0, "Introduction"
- Section 2.0, "Architecture Overview"



- Section 3.0, “Pin Descriptions”
- Section 4.0, “Ethernet Port Logic (EPL)”
- Section 5.0, “Chip Management”
- Section 6.0, “Filtering and Forwarding Unit (FFU)”
- Section 7.0, “Routing”
- Section 8.0, “Layer 2 Lookup”
- Section 9.0, “Port Mapping and Packet Replication”
- Section 10.0, “Frame Hashing”
- Section 11.0, “Triggers”
- Section 12.0, “QoS and Congestion Management”
- Section 13.0, “Egress Scheduling and Shaping”
- Section 14.0, “Statistics and Monitoring”
- Section 15.0, “Electrical Specification”
- Section 16.0, “Mechanical Specification”

1.4 Definitions

Table 1-2 Terminology Definitions

Term	Definition
{A,B}	Denotes a bit concatenation of variable A or B where A is most significant bit field and B is least significant bit field. Note that {0,A} means that 0s are added to the left (most significant bits) to pad to the desired size while {A,0} means padded to the right.
Bit Numbering	Bit 0 is the least-significant bit throughout the FM4000 architecture (even if Ethernet standards and specifications suggest otherwise).
Byte	8 bits.
Double Word	64 bits.
EBI	External Bus Interface. Intel's term for the external CPU interface. EBI and CPU Interface are used interchangeably.
FM2000	The original member of the Intel Ethernet Switch Family, the FM2000 is a layer-2 10 GbE switch chip platform which forms the basis for many L2 switch product variants (including the FM2224, FM2212, FM2208, FM2112, FM2104, and FM2103).
FM4000	The FM4000 series is an enhanced multi-layer 10 GbE switch chip platform, which are pin-compatible with their original FM2000 counterparts. The FM4xxx devices are full featured L3 routing devices plus other enhancements such as ACL's, congestion management, increased frame memory and more.
GloRT	Intel-proprietary Global Resource Tag, which is used to pass global identification information from one FM4000 device to another in a network. GloRT is the proper “pronunciation”, although other uses may appear in the document—and have the same meaning (e.g., glort, Glort, GLORT, etc.)
Half-word	16 bits.

Table 1-2 Terminology Definitions (Continued)

Term	Definition
ISL	Intel-proprietary Inter- Switch Link tag, which is used to pass relevant management and control information from one FM4000 device to another in a network.
Jumbo Frame	The maximum jumbo frame size for FM4xxx devices is 16,376 bytes.
Logging	Logging refers to a copy of the frame sent to a local CPU for monitoring purpose.
Mirroring	Mirroring refers to a copy of the frame sent to another port for monitoring purpose.
Packet or Frame	**Packet** — On a typical computer network, data is transmitted in the form of structured and modest-sized packets. Instead of transmitting arbitrary-length strings of data, structured packets Packet or Frame allow error checking and other relevant processing to occur on smaller easier-to-retransmit data. Packetized data also helps to alleviate traffic jams on the network when multiple nodes are contending for a shared network resource. Because packets are typically smaller than the complete data stream, techniques such as time-division multiplexing (TDM) can be used to share and interleave traffic, making it appear that multiple nodes are using the network resource at the same time. **Frame** — While a packet is a small block of data, a Frame is the definition of how packets of data are defined and transported on a specific network. When sending data over a network, both sides of the connection must agree on a common frame format (e.g., when a frame starts, when a frame ends, padding, etc.) Combining terms, an Ethernet packet is sent onto an Ethernet interface using an Ethernet frame format. This document uses both terms interchangeably.
Register Type	Registers are split into fields of the following types: RO: Read-Only (This register cannot be programmed by software. Typically reports a status.) RW: Read-Write (Reading returns the value written.) CW1: Clear-on-Write-1 (Writing 1b to any bit clears that bit.) CW0: Clear-on-Write-0 (Writing 0b to any bit clears that bit.) CR: Clear-on-Read (Reading the register clears the register.) RV: Reserved (For upward compatibility. Always write as zero, ignore on read.)
Segment	A portion of a packet corresponding to one of the common architecturally-significant storage and processing “chunks” that has been defined in the architecture. In FM4xxx devices, a segment is 512 bytes.
Sub-segment	A fragment of a segment that corresponds to the smallest architecturally-significant “atomic unit”. In FM4xxx devices, a sub-segment is 64 bytes.
Trapping	Trapping refers to special frames that are captured by the switch and redirected to a local CPU for processing.
Word	32 bits.
X[0..N	Denotes an array which indexes go from 0 through N inclusively.
X[N:0] or X[0:N]	Denotes a bit range within a variable. The bit range [0:N] indicates that the bit 0 is the most significant while bit range [N:0] indicates that the bit 0 is the least significant.

§ §

2.0 Architecture Overview

2.1 Introduction to Architecture

The main components in the Intel Ethernet Switch FM4000 architecture are shown in Figure 2-1.

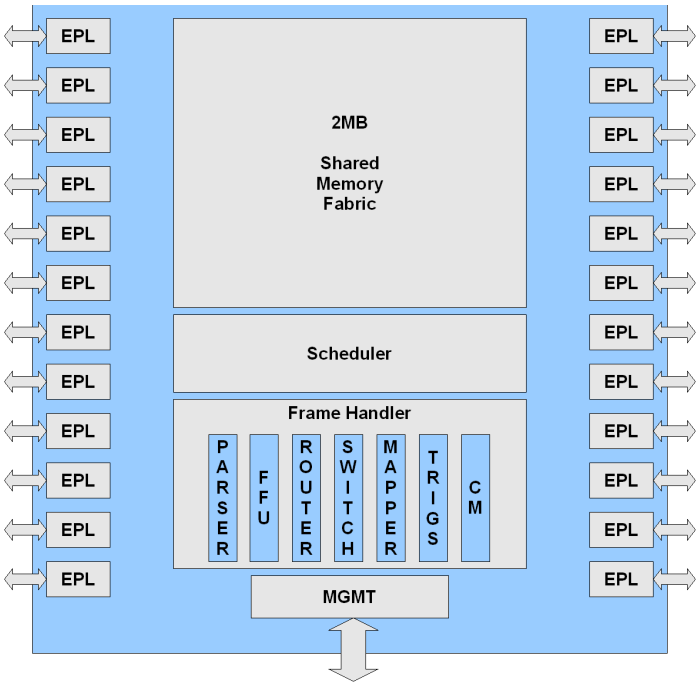


Figure 2-1 FM4000 Block Diagram

Definitions:

- **EPL** — The Ethernet Port Logic interfaces to the XAUI physical interface block, and implements the PCS and MAC layers for transmission and reception. The EPL parses incoming packets to extract the packet headers which are sent to the frame handler for packet processing while saving the entire packet in the shared memory. In the egress direction, the EPL receives segment pointers from the scheduler as well as extra data on the packet allowing the EPL to modify the packet on the way out if needed.
- **Shared Memory Fabric** — The shared memory switch fabric stores incoming packets from ingress EPLs and forward them to egress EPLs upon request from the scheduler.

- **Frame Handler** — The frame handler makes forwarding decision on the packet based on the frame header received from the EPL. The forwarding information is sent to the scheduler.
- **Scheduler** — The scheduler manages free data segments, maintains receive and transmit queues and schedules packets for transmission. The free segments are forwarded as needed to the EPLs which use them to store incoming packets to the right location in the shared memory fabric. The scheduler keeps the list of the segments sent to each EPL and waits for the frame handler forwarding decision before placing the packet at the tail of the proper transmission queue. The scheduler then applies advanced scheduling algorithms to decide which packet to forward to the EPL, and then sends a segment list to the EPL, which uses those segments to retrieve the packet from memory
- **Management** — The management block interfaces to the different components in the device to provide a coherent mechanism to manage the switch as an integrated system.

FM4000 operates as a one-arm IP router combined with an Ethernet layer 2 switch. This is shown in Figure 2-2. In this architecture, incoming packets are first associated with a VLAN (using the VTAG tag if present or by associating a default VLAN) and are then either switched within their respective VLANs or routed across VLANs or both. The decision to switch or route or drop depends on tables stored in the FFU (which includes a large ternary CAM to store IP route entries and access control lists), other tables located in the switch (such as MAC Address table) or in the router (such as ARP table) to complete the operation selected.

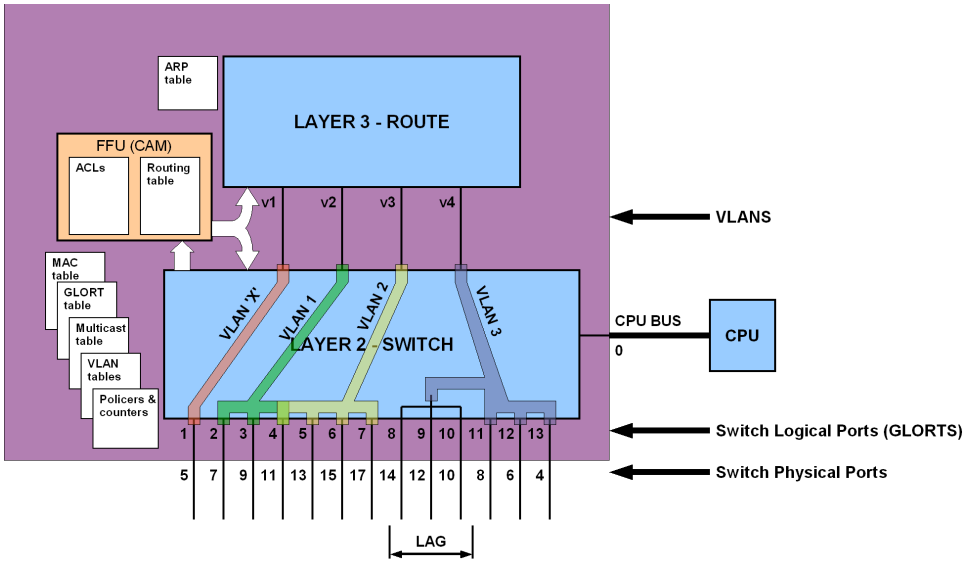


Figure 2-2 FM4000 Switch/Router Concept

The switch includes a set of features such as global resource tags (“GloRTs”), distributed link aggregation, inter-switch tags and advance multicast distribution to allow a set of switches (cluster) to operate as a single switch as shown in the following two illustrations. Figure 2-3 shows a stack arrangement, while Figure 2-4 shows a fat-tree (Clos) architecture. Both examples allow exploitation of the full feature set through a F64 tag use on internal links.

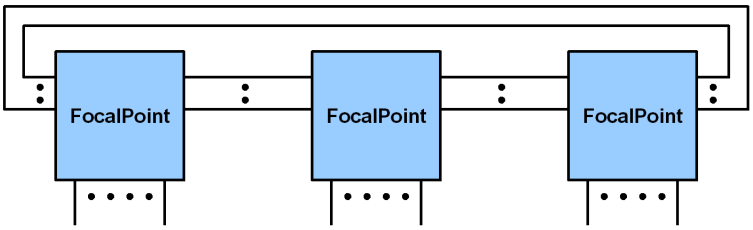


Figure 2-3 FM4000 in a Stack Topology

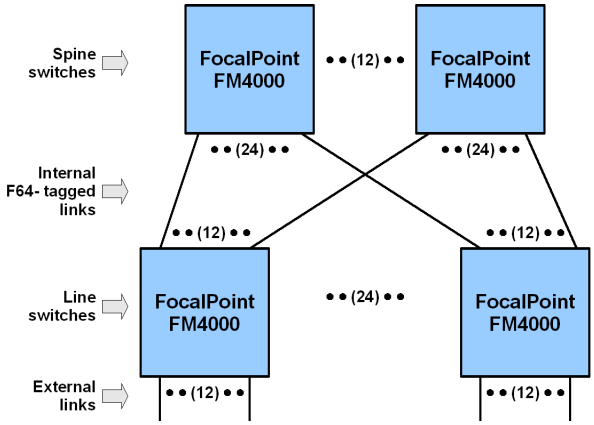


Figure 2-4 FM4000 in a Tightly-Coupled Clos Topology

Figure 2-5 shows a fat-tree using non-Intel-tag-aware switches as spine switches. The loosely coupled architecture uses an F32 tag on internal links which is designed to be compatible with existing switches but only allows a subset of features to operate in this topology. The services not available in this architecture are distributed link aggregation and centralized management such as trapping and logging.

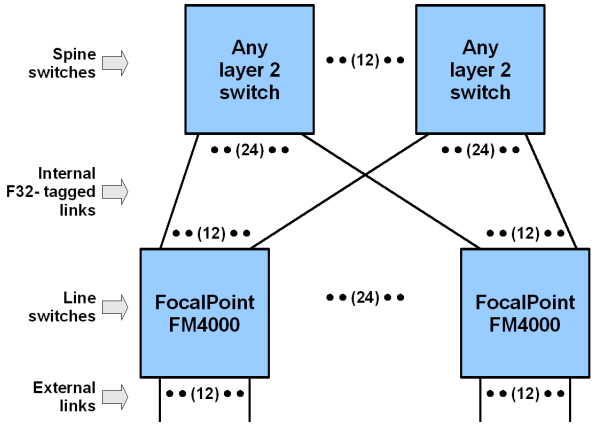


Figure 2-5 FM4000 in a Loosely-Coupled Clos Architecture

2.2 Frame Parsing

The frame parsing is shown in Figure 2-6. The packet received is always stored in the switch fabric. The job of the parser is to extract the useful fields from the header of the packet and present them to the frame processor for processing. The maximum number of bytes passed to the frame processor is 78 bytes.

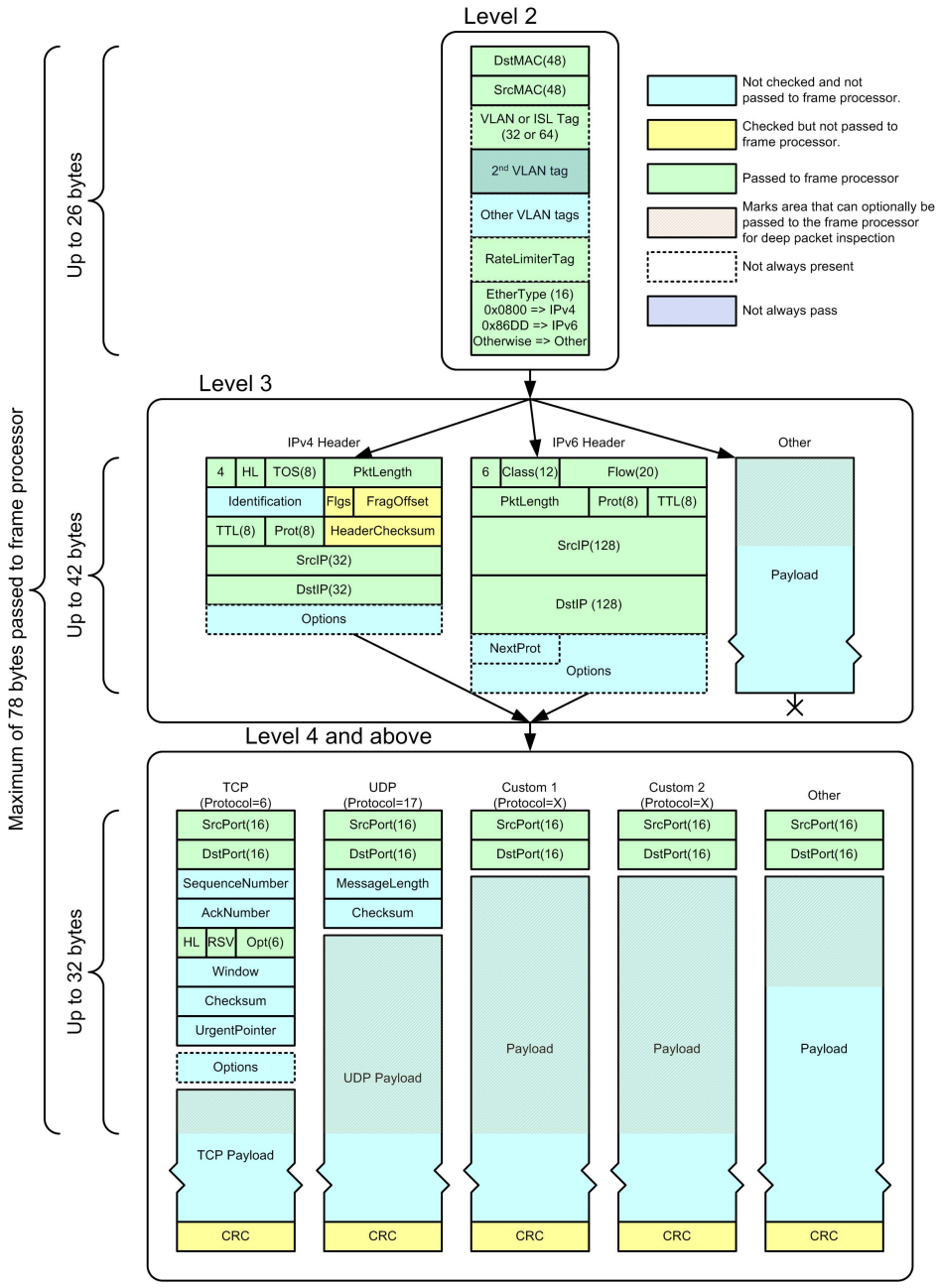
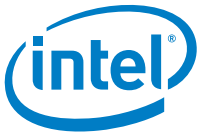


Figure 2-6 Frame Parsing



2.2.1 Layer 2

The layer 2 processing is as follows:

- Destination MAC address
  - — Always forwarded to the frame handler.
- Source MAC address
  - — Always forwarded to the frame handler.
- VLAN tags and ISL processing.
  - — The port logic supports different VLAN tag encapsulation.
  - — First, each port is statically configured to define if an ISL tag is expected. The following options are possible:
    - **No ISL** — If the port is not configured to expect an ISL tag, the parser checks if one of 3 VLAN types is recognized (0x8100 or one of the two user defined VLAN Ethernet type A or B) and, if there is a match, forwards the next 16 bits to the frame handler as VLAN PRI and VLAN ID. The parser then proceeds analyzing the next tags to detect VLAN stacking.
    - **F32** — The parser proceeds as for "No ISL" tag case, but also checks if an F32 tag is present after the VLAN and RLT tags. The F32 tag has a special configurable 16-bit Ethernet type which is followed by the source GloRT. If the F32 tag is found present, the source GloRT is captured and sent to the frame handler. If the F32 is not present, the frame handler associates the default source GloRT to this packet.
    - **F64** — The parser assumes the presence of a 64-bit tag immediately after the DMAC/SMAC addresses and forwards the content to the frame handler and marks the packet as F64 tagged. The parser then proceeds analyzing the next tags to detect VLAN stacking.
    - **F96** — The parser assumes presence of a 96-bit tag immediately after the DMAC/SMAC addresses and forwards the first 64 bits to the frame handler and skips over the next 32 bits and marks the packet as F64 tagged. The parser then proceeds analyzing the next tags to detect VLAN stacking.
    - **X32** — The parser assumes presence of a 32-bit tag immediately after the DMAC/SMAC address. It skips over the first 16 bits, assumes that the next 16 bits contain VLAN PRI and VLAN ID (as it would if the type were 0x8100) and forwards those to the frame handler and marks the packet as X32 tagged. Further stacking is not supported in this mode. The parser assumes that the Ethernet type follows immediately after this header.
    - **X64** — The parser assumes presence of a 64-bit tag immediately after the DMAC/SMAC address. It skips over the first 16 bits, assumes that the next 16 bits contain VLAN PRI and VLAN ID (as it would if the type were 0x8100), forwards those to the frame handler, marks the packet as X64 tagged and skips over the next 32 bits. Further stacking not supported in this mode. The parser assumes that the Ethernet type follows immediately after this header.
    - **X96** — The parser assumes presence of a 64-bit tag immediately after the DMAC/SMAC address. It skips over the first 16 bits, assumes that the next 16 bits contains VLAN PRI and VLAN ID (as it would if the type were 0x8100), forwards those to the frame handler, skips over the next 64 bits and marks the packet as X96 tagged. Further stacking not supported in this mode. The parser assumes that the Ethernet type follows immediately after this header.

— VLAN stacking is supported for non-ISL tagged packets and for F32, F64 and F96 tagged packets. It is not supported for X32, X64 and X96 packets. The different frame formats are shown below. Note that the RLT and F32 tags are optional and that the EPL is configurable for the detection of those fields and if the Ethernet type that is reserved for those.

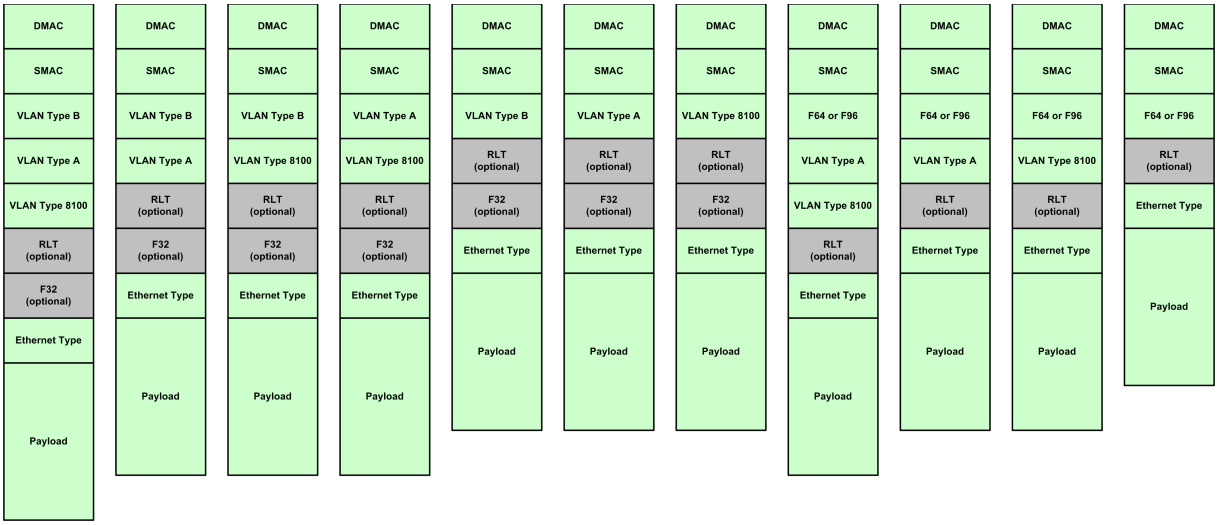


Figure 2-7 VLAN Stacking Options

— Any other VLAN stacking configuration is considered invalid. As soon as the parser determines that an invalid combination is present, parsing stops and the Ethernet type of the packet is captured where parsing stopped. Frames that come in with invalid VLAN tag combinations are switched but cannot be routed.

- Rate Limiter Tag
  - — Optionally forwarded to the frame processor if present. The rate limiter tag is used during congestion management in a tightly coupled system. The port can be configured to skip over this field even if it is present. The RLT tag can be detected only if the VLAN stacking is one of the options recognized.
  - — The rate limiter tag is not detected if the VLAN stack was considered invalid.
- Ethernet Type
  - — Always forwarded to the frame processor.

2.2.2 Layer 3

FM4000 detects the presence of an IPv4 or IPv6 header by comparing the Ethernet types to the known values for IPv4 and IPv6 Ethernet types and by checking the version field (first 4 bits of the layer 3 header). If the Ethertype is 0x0800 (and the IP version is 4), the frame is parsed as IPv4 frame. If the Ethertype is 0x86dd (and the IP version is 6), the frame is parsed as IPv6. If the Ethertype is 0x8808, it is parsed as a MAC Control frame (most likely a PAUSE frame). In any other case, such as a length value (0x0000-0x0600) in this field, the parsing terminates. FM4000 does not attempt to parse LLC or SNAP encoded packets. The port may be configured to always stop parsing after layer 2, but when layer 3 parsing is required, it proceeds as follows:

IPv4:

- The parser always forwards the version, header length, TOS/DS field, packet length, TTL, layer 4 protocol, destination IP and source IP addresses to the frame processor.
- The parser also checks the presence of header options by checking the Internet Header Length (HL) field, stored in bits 4-7 of the IPv4 header. HL is the length of the IPv4 header in 32-bit words, including options. The minimum value of HL is 5, indicating no options; if HL > 5, then options are present. It is not necessary to parse the contents of the IPv4 options. The parser skips over all options.
- The parser also checks if the packet is a fragment, and if it is the first fragment. If it is the first fragment, the layer 4 is processed, otherwise, the parsing stops. The parser detects if it is the first fragment by checking that IPv4 fragment offset in bits 51-63 of the IPv4 header is 0.
- If there is an error during decoding (IP header checksum invalid, invalid HL, etc...), the packet is flagged as having a parse error and the frame is discarded and counted. Note that a trigger action could override the default disposition and possibly trap the packet to the local processor for further processing. Packets that have a parse error are never routed.
- The parser declares the frame as an IP multicast frame if the destination IP address is greater or equal to 224.0.0.0 and declares the frame as an IP unicast otherwise.

IPv6:

- The parser always forwards the class, flow, packet length, protocol, TTL, destination and source IP addresses to the frame processor.
- The parser is also capable of detecting the presence of options in the packet by comparing the protocol field to the known options and skipping over the options if they are present. The parser stops after the current header if the protocol in the next header does not match one of the following:

  0x00 = IPv6 Hop-by-Hop Option (length 8N+8 bytes)

  0x2b = Routing Header for IPv6 (length 8N+8 bytes)

  0x2c = Fragment Header for IPv6 (length 8 bytes; N is always 0)

  0x3c = Destination Options for IPv6 (length 8N+8 bytes)

  0x33 = Authentication Header (length **4N+8** bytes -- not the same as the other options)
- The parser also checks if the packet is a fragment, and if it is the first fragment. If it is the first fragment, the layer 4 is processed. Otherwise, the parsing stops. If an IPv6 packet is fragmented, it carries a Fragment Header option. If it has a Fragment Header, and the fragment offset (bits 16-28 of the Fragment Header option) is nonzero, this is not the first fragment.
- If there is an error during decoding (IP header checksum invalid, invalid HL, etc...), the packet is flagged as having a parse error and the frame is discarded and counted. Note that a trigger action could override the default disposition and possibly trap the packet to the local processor for further processing. Packets that have a parse error are never routed.
- The parser declares the frame as an IP multicast frame if the highest byte of the destination IP address is equal to 255 and declares the frame as an IP unicast otherwise.

Non IPv4 and IPv6 Packets:

- Each port can be configured to forward the first N bytes to the frame processor for processing. The bytes forwarded are mapped to the DIP, SIP, KEYA, KEYB fields of the Frame Filtering and Forwarding unit.

If the IP packet is fragmented, and this is not the first fragment, parsing terminates at this point because the layer 4 header is not available. The parser verifies the IPv4 fragment offset is in bits 51-63 of the IPv4 header; if the fragment offset is nonzero then this is not the first fragment.

If an IPv6 packet is fragmented, it carries a Fragment Header option. If it has a Fragment Header, and the fragment offset (bits 16-28 of the Fragment Header option) is nonzero, this is not the first fragment.

The port may also be configured to always stop parsing after layer 3, regardless of fragmentation. Any layer 4 fields that are either not parsed or not present due to fragmentation are treated as '0' in the frame processing pipeline.

2.2.3 Deep Packet Inspection for IP frames

FM4000 switches have the ability to forward additional frame header bytes to the Frame Handler. This capability is referred to as deep packet inspection. Up to a total of 78 bytes of frame header can be forwarded to the Frame Handler, consisting of normal header bytes and deep packet inspection bytes. This capability is detailed more in the Section 4.0, “Ethernet Port Logic (EPL)”.

2.3 Frame Processing Pipeline Overview

FM4000 Frame Forwarding is designed to handle wire-speed layer 2/3/4 switching in the context of a single-chip or a multi-chip solution in a variety of topologies. The features offered are:

- Global routing across multi-chip in fat tree, ring or meshed topologies.
- Layer 2 switching with optional automatic address learning and security.
- Layer 3 routing for IPv4 and IPv6 including routing across multiple paths (ECMP).
- Basic and extended Access Control Lists (ACLs) for layer 2 / layer 3 / layer 4 and deep packet inspection.
- Snooping of IGMP v1, v2, and v3.
- Link aggregation across multiple links using various information from frame header to derive hashing function.
- Trapping special frames.
- Triggers.
- Congestion Management.
- Logging and/or mirroring.

The frame processor pipeline is shown in Figure 2-8. The different elements of this pipeline are briefly described in Section 2.3.1 through Section 2.3.14.

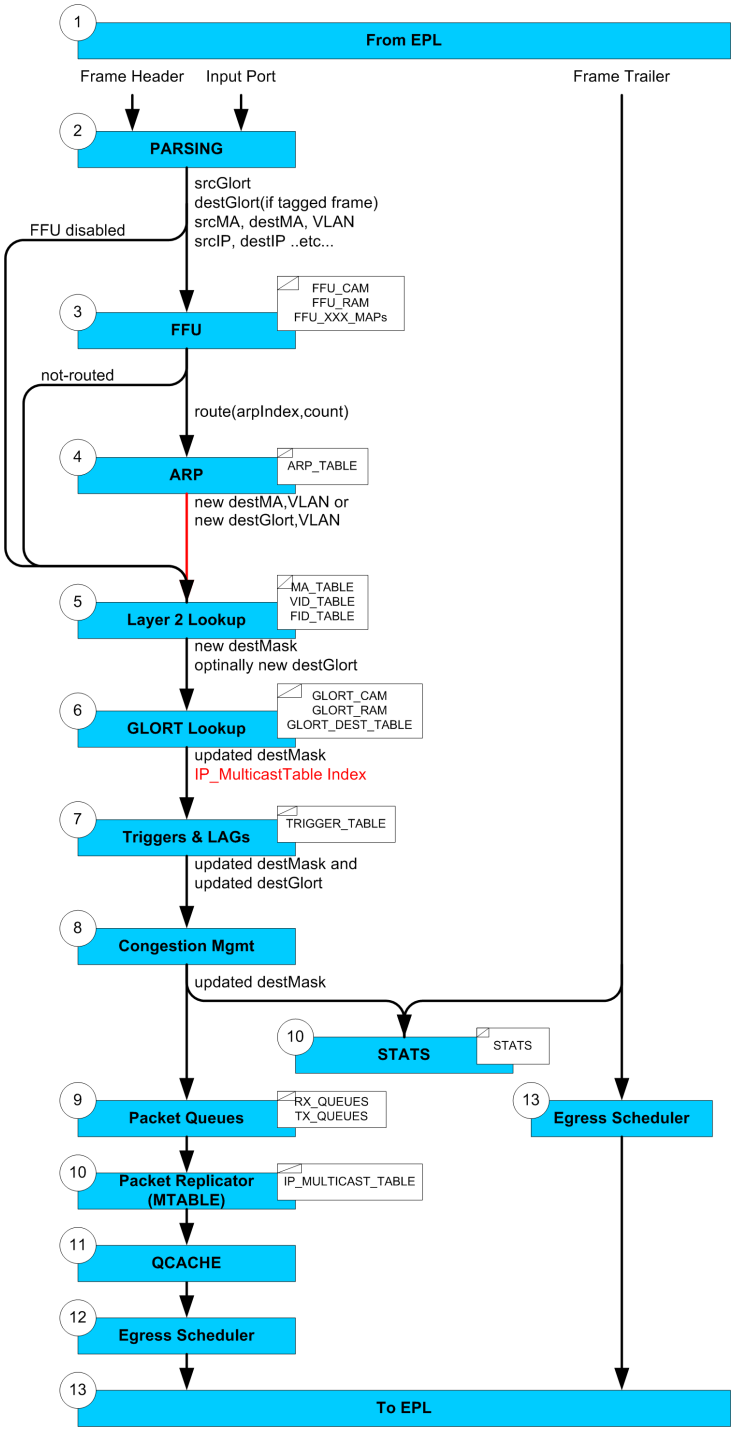


Figure 2-8 Frame Processor Pipeline

2.3.1 Frame Header

The frame header contains the following information:

- Source Port (5 bits)
- Source MAC Address (48 bits)
- Destination MAC Address (48 bits)
- VLAN (12 bits)
- 802.1p priority and CFI/DEI bit (4 bits)
  - The 802.1p and CFI/DEI bit are processed as one 4-bit element inside the switch.
- Ethernet Type (16 bits)
  - The frame Ethernet type is the first type field after VLAN and RLT tags.
- Type of IP packet (IPv4 or IPv6)
- Source IP (32 or 128 bits)
- Destination IP (32 or 128 bits)
- Layer 4 Source Port (16 bits)
- Layer 4 Destination Port (16 bits)
- Layer 4 Options (6 bits)
- Layer 4 Protocols (8 bits)
- IP TOS (8 bits)
- IP TTL (8 bits)
- IP Flow ID (20 bits)
- Deep packet inspection:
  - Extra bytes extracted after the layer 4 decoding for IP packets and after layer 2 Ethernet type for non IP packets. These extra bytes are passed to the FFU for further processing.

• ISL tag:
  - Switch priority (4 bits)
  - Source GLORT (16 bits)
  - Destination GLORT (16 bits)
  - User info (8 bits)
  - Frame Type

Note: All non-applicable fields are automatically set to 0b.

The following provides information on how the frame processor forwards a frame.

- The chip always stores and forwards the first 64 bytes as a minimum.
- The chip must parse the entire header before determining an output port. So if more than 64 bytes of header are being parsed, the forwarding decision is not made until this is complete.
- The forwarding decision is made in parallel with receiving the frame. This requires 50-100 ns of processing time per frame.

- Once the frame destination is known, the frame pointer is put into the scheduler, which determines the next frame to transmit. If the egress port is empty, this is a very small amount of time. If the frame is corrupt (bad CRC), and this corruption can be detected before the frame is selected for dequeue in the scheduler, the frame is dropped in the scheduler instead of being transmitted with bad CRC.
- Once the frame is sent to the egress port, the egress port holds the frame for a some number of EPL cycles (between 6 and 32 cycles) before transmitting. If it gets an indication from the switch element that the frame has been queued enough to guarantee transmission, the hold is released and the frame starts to transmit immediately.

2.3.2 Frame Trailer

The frame trailer contains the following information:

- Packet Length
- End Of Frame Status (good CRC, bad CRC, symbol error, disparity error, oversize, undersize)
  - For statistics
- Packet disposition (discard eligible or forward)
  - Frame disposition, i.e. forward normally or discard if possible. Frames marked for discard eligibility:
    - Do not cause MAC address learning or security violation events.
    - Do not cause trigger interrupts.
    - Cancel PAUSE actions.
    - Are not sampled for congestion notification purpose.
    - Cancel procession of congestion notification frames.
    - Are not forwarded if frame transmission has not yet started (in cut-through mode, the frame transmission may have already been started before the end of frame is received, and the discard flag may come too late to actually delete the frame).
  - The following actions are not changed regardless of whether the frame is marked discard eligible or not:
    - Policers are still applied.
    - FFU counters are still updated.
    - Ingress rate-limiters are still active.
    - SFLOW is still performed.
    - Memory management is still performed.

The EPL contains configuration options to select how to mark a frame (discard or forward) depending on the type of errors encountered. The default is to set the discard flag whenever the frame is erroneous for any reason.

Note: End of frame status and packet disposition are used for different purposes.The end of frame status is used for statistics while the packet disposition is used to decide how to dispose of the packet. As an example, a frame with a bad CRC may optionally be marked as forward normally, but is still be counted as a frame with a bad CRC.

2.3.3 GloRT

A GloRT (short for global resource tag) is a 16-bit number that can be used to identify a specific port, link aggregation group, multicast group, management frames or any other packet destination. The term GloRT is sometimes used interchangeably between GloRT values, GloRT actions and GloRT functions. Each FM4000 device must be configured with the following set of GloRTs:

- **Per-port Source GloRT** — Whenever a frame arrives without an ISL tag, this source GloRT is associated with the frame. If the frame's source MAC address is learned, this GloRT value is stored in the MAC address table.
- **CPU GloRT** — Whenever a frame is trapped or logged or mirrored to the CPU, the top 8 bits of the destination GloRT are set to this value (set in Frame Handler Register TRAP_GLORT). The bottom eight bits are set to a Trap Code value, chosen by hardware to indicate the reason for sending the frame to the CPU. The set of defined trap codes is listed in Section 5.14.

In addition, other feature-specific GloRTs may also be configured:

- **RX Mirror GloRT** — Using the Triggers, a copy of any ingress frame can be copied to one additional RX Mirror destination.
- **TX Mirror GloRT** — FM4000 supports a single TX port mirror. All frames sent to a TX port can be copied to a configurable TX Mirror destination GloRT.

The mapping from GloRT to physical port involves a ternary CAM lookup. This allows the GloRT space to be allocated in a fairly arbitrary manner. The only structure required by the mapping function consists of: (1) a configurable bit range within link aggregation group GloRTs, used to identify individual physical port members; and (2) the fixed eight-bit CPU trap code field within the CPU GloRT.

2.3.4 Ethernet Port Logic

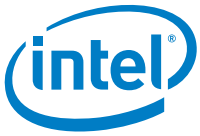
The Ethernet Port Logic includes the PCS and MAC layers and is responsible for extracting the Frame Header fields from the packets received while skipping over superfluous fields. The EPL is detailed in Section 4.0, “Ethernet Port Logic (EPL)”.

2.3.5 Parsing

The frame header received is parsed and a switch priority is associated with the frame using one of the following fields:

- Switch priority (ISL tag)
- 802.1p priority
- TOS
- DiffServ

The priority association is detailed in Section 12.0, “QoS and Congestion Management”.



2.3.6 FFU (Filtering and Forwarding Unit)

The parsed frame header is then presented to a frame Filtering and Forwarding unit which produces a set of orthogonal actions on the frame using different matching rules. The most common actions are: route, permit, deny, count, mirror, trap, and log. This is not an exhaustive list as the FFU also contains less common actions.The FFU contains a 32 x 512 x 36-bit flexible ternary CAM structure along with a set of control registers. The FFU also contains a mapper which maps known fields into smaller entities to improve CAM resource utilization. This mapper includes for example, recognition of specific MAC addresses, compression of IPv6 header and combinations of TCP/UDP port ranges. The FFU organization is detailed in Section 6.0, “Filtering and Forwarding Unit (FFU)”.

The default action is “switch” and “permit” in case no matches are found in the FFU. The entire FFU can be disabled.

IP routing is normally done by loading a routing table into the FFU_CAM (ordered by longest prefix match). Each entry in the table is loaded with a mask, a route entry and a 4-bit field that represents the router's MAC address along with a route action with information about the next hop.The route action includes information on how to index the ARP table.

An IP unicast frame is thus routed if it is addressed to the router's MAC address and is IPv4 (Ethertype is 0800 and header version is 4) or IPv6 (86DD and header version is 6) and the port and the VLAN associated with the frame are marked as routable. If the frame is not addressed to the router's MAC address or comes from a port or VLAN that is not routable, it is switched. If the frame is addressed to the router but is not IPv4 or IPv6, it is trapped to the CPU.

For IP multicast frames, if the frame is not IPv4 or IPv6, or its IP address does not start with the prefix used for IP multicast frames (bits 1110 for IPv4 or bits 1111 1111 for IPv6) and the port or VLAN is not routable, then the multicast frame is L2 switched only. If the frame is an IP frame with a proper IP multicast MAC address, the frame is both switched and routed at the same time. It needs to get switched to multicast destinations on the same VLAN, and also routed to multicast destinations on other VLANs. Although the frame is technically both switched and routed, FM4000 treats it more as being routed, because it looks up the frame by IP address in the FFU, rather than by MAC address in the MAC table. Other entries can coexist in the CAM to produce ACLs allowing specific frames to be denied or permitted and then optionally counted, logged, trapped or mirrored.

2.3.7 ARP Unit

The route action includes an ARP Table index and the number of possible paths. If the number of paths is k, the FM4000 picks a number between 0 and k-1 to add to the ARP Table index, to get the actual index into the ARP table which is used to read the ARP table. This number is picked using a hash of the frame, using the hard threshold algorithm, which means using division (not modulo) to divide the hash range into equal-sized buckets.

The 16 K ARP table contains either a destination GloRT/VLAN or a MAC/VLAN address pair or a VLAN/flag. The GloRT is typically used for IP multicasting (see Section 2.3.3 and Section 9.4) or for specialized multi-chip systems. The VLAN/MAC address is typically used for IP unicast traffic where the VLAN/MAC address specifies the next hop. The VLAN/flag is used for IPv6 unicast when the destination is retrieved from the lower 48 bits of the address. Multicast MAC addresses are classified after the ARP table.

Note: Disabling the FFU causes the routing step to be bypassed as well.

2.3.8 Layer 2 Lookup Unit

The layer 2 Lookup Unit includes a VLAN table (VID_TABLE), a spanning-tree state table (FID_TABLE) and a MAC address table (MA_TABLE). The layer 2 lookup first uses the destination MAC address and the VLAN port to retrieve the logical destination port (GloRT) from the MA_TABLE, and then uses the VLAN to retrieve a spanning-tree number and VLAN disposition, and uses the FID_TABLE to retrieve the spanning tree state. If the destination address is unknown, the packet is flooded using a flood GloRT. The address can also be automatically learned or raise security violations depending on configuration of the switch.

2.3.9 GloRT Lookup

This unit is responsible to map a destination GloRT into a destination mask, IP multicast replicator index and to implement link aggregation.

There are multiple possible sources for the destination GloRTs:

- The frame may contain an ISL tag which contains the destination GloRT
- The FFU may trigger a route command which may include a new destination GloRT
- The ARP table may include a new destination GloRT
- The MAC table may include a new destination GloRT
- A default GloRT is assigned (GloRT-flood, GloRT-broadcast) otherwise

The GloRT Lookup Unit contains three tables: the GloRT CAM, the GloRT RAM and the Destination Table.

The GloRT CAM is a 256-entry ternary CAM used to search the destGloRT. The GloRT RAM is simply a RAM with one-to-one correspondence with the GloRT CAM, and which contains the data associated with each entry of the GloRT CAM. This data includes a base pointer to the destination table, the number of links in the group and one or two sub-indexes retrieved from the destination GloRT itself.

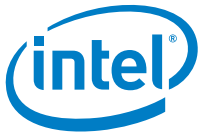
The unit uses this information to compute an index and retrieve a final destination mask which contains one bit per output port where the frame is delivered, along with an IP multicast index, which is only required if multicasting IP frames across multiple VLANs.

The unit also implements link aggregation pruning to load balance traffic across multiple links.

2.3.10 Triggers and Link Aggregation Units

The triggers are low-level elements that can be used to modify traffic flows. There are up to 64 triggers. The details of the triggers can be found in Section 11.0, “Triggers”. The output of a trigger is a destination mask and an updated destination GloRT. The destination mask is ANDed with the link aggregation destination mask.

The link aggregation mask uses the result of hashing over the frame to determine which port of a link aggregation group is used to transmit the frame.



2.3.11 Congestion Management

The congestion management maintains information about the size of the different queues and can mark, discard and/or send congestion management frames back to the originator when the size of queues exceeds certain limits.

2.3.12 Packet Queues

If operating in cut-through mode, the frame handler places the pointer to the packet processed into a temporary receive queue (8 receive queues, one per priority) along with a destination mask and other information about the packet. If operating in store-and-forward mode, the frame handler holds onto the pointer until the end of the packet is received and then places the pointer into the temporary receive queue if and only if the packet is valid. Otherwise it is dropped. Then, a sub-unit of the scheduler copies the packet pointers from the receive queues into the transmit queues (200 transmit queues, one per priority), replicating the pointer if the packet is multicasted across multiple ports.

2.3.13 Packet Replicator

The packet replicator is called each time a packet is de-queued from a transmit queue for transmission. Each packet has multiple attributes including flags to indicate if this packet is to be logged, mirrored and/or multicasted (all could be true at the same time). If the multicast index is used (it is used whenever a multicast packet needs to be replicated across multiple VLANs), the replicator uses this index to read a table which gives a list of VLANs on this port that should receive a copy of this packet.

The packet replicator also checks if a copy must be replicated for mirroring or logging purposes. If there is no logging, mirroring or multicasting across VLANs, then MTABLE is transparent and performs no action other than forwarding the frame to the next stage of the pipeline.

Packets replicated for logging or RX mirroring are exact copies of the packets received except for the Intel ISL tag which may actually be added, updated or removed depending on how the switch is configured. The CRC is also updated if the Intel tag was added, modified or removed. Packets replicated for TX mirroring matches the transmitted packet if and only if the destination mirror port is configured the same way as the normal destination port (same VLAN tagging options, same router address, default VLAN type, etc...), except for the Intel ISL tag which is updated with the destination GloRT for mirror.

2.3.14 Egress Scheduler and Egress EPL

The egress scheduler uses various information about the ports and the queues to determine when to schedule a packet for transmission. The egress scheduler includes traffic shaping and pacing algorithms. Once a frame is scheduled for transmission, the pointer is forwarded to the EPL which retrieves the packet payload from the main memory and alters the packet on its way out if required.

- If a packet is routed, the actions taken are:
  - Change source and destination address and VLAN.
  - Decrement TTL.
  - Compute new CRC.
- If packet is switched, the actions taken are:
  - Change VLAN if needed (add, delete, replace).
  - Compute new CRC if packet is changed.

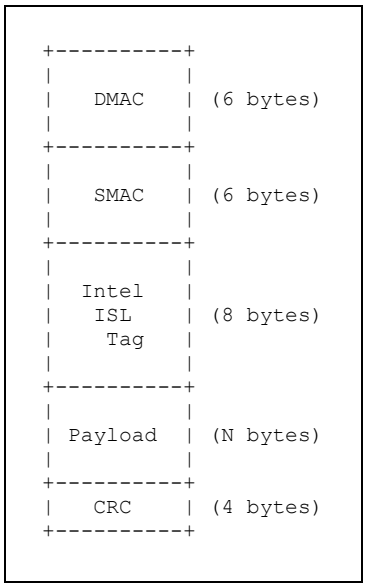
The egress scheduler is detailed in Section 13.0, "Egress Scheduling and Shaping".

The egress EPL is detailed in Section 4.0, "Ethernet Port Logic (EPL)".

2.4 Intel ISL Tag

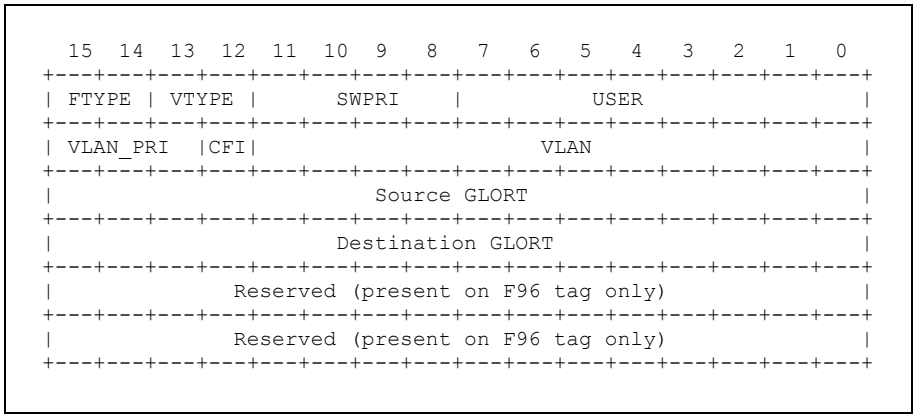
The Intel ISL tag can be attached to frames on links between Intel chips, and between a Intel chip and a CPU. It carries extra frame-specific data that allows multiple chips to work together as a single system. Each port of the switch is configurable to expect the Intel ISL tag on all frames or not. All data fields in the Intel ISL tags are in big-endian format.

When present, the tag is located immediately after the source address as shown below.



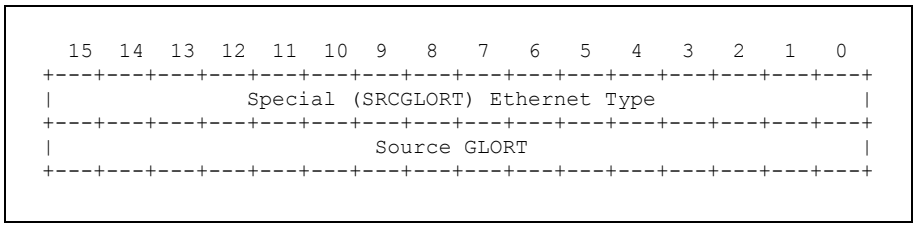
The tag exists in 3 different formats: F32, F64 and F96.

The format of the F64/F96 tag is shown here (most significant byte transmitted first).



The only difference between the F96 and the F64 is the presence of an extra 32 bits reserved for usage by future products. The FM4000 family always transmits 0's in the reserved field and ignores those fields in reception.

The format of the F32 tag is shown here (most significant byte transmitted first).



The F32 tag is designed to allow the source GloRT to be passed through switches that do not support the F64 tag. The source GloRT is required for global address learning and sharing in a tightly coupled cluster implementing Clos architectures. Note that the F32 is optionally present and if present, contains the source GloRT that is associated with this packet. If the source GloRT is found to be 0b, the frame handler associates a default source GloRT.

2.4.1 Frame Type (FTYPE)

The 2-bit frame type code indicates whether a frame has been *routed* since ingress, whether it requires *special delivery*, or whether it is a *management* frame.

00b = Normal

01b = Routed

10b = Special Delivery

11b = Management

Definitions

Normal: The default tag given to all incoming traffic. Learning and loopback suppression are applied to normal traffic in the normal way. By default, strict destination GloRT routing is not used with normal traffic.

Routed: If this is set, the frame still needs to be IP routed (its src MAC changed, its dst MAC and VLAN possibly changed, and its TTL decremented) since ingress. When a frame has been modified, source-GloRT-based loopback suppression no longer applies. Also, since the new src MAC is no longer associated with the source GloRT, there should be no learning of routed frames. (The MAC of the router should already be locked in the MAC table.) By default, strict destination GloRT routing is not used with routed traffic.

Special Delivery: Used for frames which are not part of the standard switched/routed traffic. It includes mirror copies and frames trapped to the CPU, that should be sent unmodified to special-purpose destinations. It also includes low-level protocols that require strict GloRT routing, bypassing link aggregation (such as LACP) or spanning tree (such as STP).

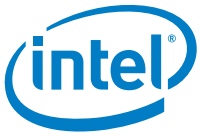
For special delivery frames, the FFU gets a "do not modify" input bit to its scenario selection (though the FFU can be configured to modify them anyway). Learning and loopback suppression do not apply to special delivery frames. By default, special delivery traffic uses strict destination GloRT routing; for trapping and mirroring, this maintains frame ordering (per traffic class), and for LACP, this means frames are directed to a specific physical port in a LAG.

Special delivery frames should not be trapped or logged, because they either originate from the CPU, are already headed to the CPU, or are simply copies of other frames.

Management: If this is set, this frame is for in-band management. Management frames are in general not modified (as with special delivery frames, the FFU is given a "do not modify" flag). By default, management frames use strict destination GloRT routing; i.e., they follow known routes through the system, and they are not affected by LAG hashing or by spanning tree state. Learning and loopback suppression do not apply to management frames. Management frames are normally not trapped or logged because they either originate from the CPU or are already headed to the CPU.

Only frames tagged as management are accepted by the MSB as in-band management. A port may be configured as untrusted, meaning that management frames coming from that port are dropped.

FTYPE	Learning	Loopback Suppression	Trap/Log	Modification Permitted	Strict Dest (by default)
Normal	•	•	•	•	
Routed	•		•	•	
Special Delivery					•
Management					•



Other implications of strict GloRT routing and special delivery frames are listed here:

Mask or Action	Strict GloRT Routing	Special Delivery Frames
PORT_CFG_2 Destination mask	Applied	Not applied
Ingress VLAN mask	Applied	Not applied
Port reflection prevention	Applied	Not applied
Egress VLAN mask	Applied	Not applied
Loopback suppression	Applied	Not applied
Learning	Enabled	Not applied
Lag filter mask	Trigger dependent	Trigger dependent
GloRT calculation	No hashing	No hashing

2.4.2 VLAN Type (VTYPE) and Management Type (MTYPE)

The 2-bit VTYPE field is only used for switched or routed frames and is used to indicate which type of VLAN it is and is encoded as follows:

00b = The original frame did not have any VLAN tag or the VLAN tag was ignored.

01b = The original frame had a VLAN tag of type 0x8100.

10b = The original frame has a VLAN tag of type USER DEFINED “A”.

11b = The original frame has a VLAN tag of type USER DEFINED “B”.

The same 2-bit field is used to carry the type of management frame and is labeled MTYPE in this case. The field is encoded as follows:

00b = Request

01b = Response or interrupt

10b = Reserved

11b = Reserved

2.4.3 Switch Priority (SWPRI)

This field carries a 4-bit value indicating the priority level of the frame. This priority level is expected to be consistent across a cluster of tightly coupled switches, unlike the VLAN priority which may be remapped differently on different ingress ports. The switch priority of a frame determines traffic class and possibly the egress VLAN priority. See Section 12.0 and Section 13.0 for details.

2.4.4 User Bits (USER)

The 8-bit USER field can be used by end-points to carry special information about the frame through a set of switches that are linked through F64 or F96 tags. They can be read and modified by the ACLs or left unchanged and are thus under total user control.

The F32 tag does not transport the USER field and has a narrower interpretation of this field. On egress ports configured for F32, the most significant bit of the USER field controls whether the source GloRT is sent or not. If this bit is set to 1b, this port transmits the source GloRT Ethernet tag. If this bit is not set, this special Ethernet tag is not sent. Similarly, on ingress of those ports, USER is set to 0x80 if the source GloRT Ethernet tag is present and to 0x00 otherwise.

2.4.5 VLAN priority (VPRI)

The priority field of the encapsulated VLAN tag, and occupies the same bit positions that the VLAN priority would occupy in an actual VLAN tag. If the frame type is 0x3 (management frame), or the VLAN Ethertype is 0x0 (no VLAN tag), this field is reserved.

2.4.6 VLAN CFI/DEI bit (VCFI)

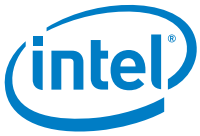
The CFI bit (or DEI bit defined in Q-in-Q) of the encapsulated VLAN tag, and occupies the same bit position that the CFI would occupy in an actual VLAN tag. If the frame type is 0x3 (management frame), or the VLAN Ethertype is 0x0 (no VLAN tag), this field is reserved.

2.4.7 VLAN ID (VID)

The VID field of the encapsulated VLAN tag, and occupies the same bit positions that the VID would occupy in an actual VLAN tag. If the frame type is 0x3 (management frame), or the VLAN Ethertype is 0x0 (no VLAN tag), this field is reserved.

2.4.8 Source GloRT (SGLORT)

The source GloRT indicates the point of origin of the frame in the switch system. If it refers to an external port, it should always be in LAG/port identifier form, to allow for learning and loopback suppression. Note that there is a default source GloRT (PORT_CFG_ISL::SrcFGlort) that gets associated with the frame if the frame comes from a port that is not ISL tagged or if the frame comes from a port that is ISL tagged but the source GloRT specified is 0b.



2.4.9 Destination GloRT (DGLORT)

The destination GloRT indicates the current destination of the frame within the system, whether it be egress port, multicast group, CPU, next routing hop, or something else.

On frames directed to the CPU, trap information is represented by different CPU destination GloRTs. The CPU GloRTs share a common 8-bit prefix, with 8 bits of CPU trap code as the suffix.The trap code indicates, for instance, trap due to triggers (with trigger number included), trap due to a particular SYS_CFG trap setting, etc.

Note: It is not possible to mark a single copy of a frame as both trap to CPU and route to a specific non-CPU destination (unless a multicast dst GloRT is used). If all FM4000 chips have a local CPU, this is not a performance issue. In any case, MTable handles the corner case of wanting to do both on a single physical port, and duplicates the frame.

Since all GloRT comparisons are maskable, it is possible to configure the system to use GloRTs of fewer than 12 bits, and use the remaining bits of the destination GloRT field as user bits. FM4000 may not preserve these bits on a mirror or CPU copy.

2.5 F96 tag

FM4000 can be configured, per port, to use a 3 word (96-bit) ISL tag. In this case, it is referred to as an F96 tag; the first two words of the F96 tag are as above, and the format of the third word is undefined and transported as is. The extra 32 bits are filled with zeros.

2.6 Action Codes

The Frame Processing Pipeline maintains a list of action codes as it processes each frame. At the end of the pipeline (prior to trigger, LAG filtering rules and congestion management), the switch applies the highest precedence action as defined in Table 2-1. This table lists all defined action codes along with an indication if the layer 2 source address is learned or not by default (the triggers have the capability to override the default learning behavior). Lower values have higher precedence than higher values.

After this step, the frame goes through the following final 3 steps (in order):

1. Triggers which may override any prior decision.
2. LAG pruning and filtering.
3. Congestion management.

Table 2-1 Action Codes

Bit Number and Relative Priority	Description	Learning	Category
0 (Highest)	Specially handled frame.	No	Privileged inter-switch frame type
1	Drop due to header parse error or discard eligible.	No	Unable to properly process frame
2	Parity error detected while doing lookup.	No	L2 Privileged Frame Types
3	Trap BPDU.	Yes	
4	Trap 802.1X frames.	Yes	
5	Trap 802.1X frames that were remapped to priority 15.	Yes	
6	Trap GARP frames.	Yes	
7	Trap GARP frames that were remapped to priority 15.	Yes	
8	Trap LACP frames.	Yes	
9	Trap LACP frames that were remapped to priority 15.	Yes	
10	Trap other IEEE reserved address.	Yes	
11	Trap other IEEE reserved address that were remapped to priority 15.	Yes	
12	Drop due to tagging rules violation.	No	L2 Ingress Policies
13	Drop PAUSE frame.	No	
14	Drop due to MAC security violation (new address).	No	
15	Drop due to MAC security violation (moved address).	No	
16	Trap CPU MAC address.	Yes	
17	Trap CPU MAC address and remapped to priority 15.	Yes	
18	Drop due to ingress STP check (non-learning state).	No	
19	Drop due to VLAN ingress membership violation.	No	
20	Drop due to ingress STP check (learning state).	Yes	

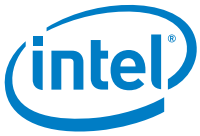


Table 2-1 Action Codes (Continued)

Bit Number and Relative Priority	Description	Learning	Category
21	Drop due to FFU action.	No	L3 Forwarding Policies
22	Trap due to FFU action.	Yes	
23	Trap due to TTL <= 1 for ICMP frames.	Yes	
24	Trap IP frames with options.	Yes	
25	Trap due to MTU violation.	Yes	
26	Trap due to IGMP.	Yes	
27	Trap due to TTL <= 1 for non-ICMP frames.	Yes	
28	Log IP multicast frames to CPU because TTL <= 1.		
29	Drop due to TTL <= 1.	Yes	
30	Drop due to VLAN egress violation.	No	L2 Egress Policies
31	Drop due to flood control of DLF frames (SYS_CFG_1).	No	
32	Drop due to GLORT_CAM miss.	No	
33	Unused.		
34	Drop Policer.	Yes	
35	Drop due to egress STP check.	Yes	
36	Flood due to destination MAC miss in MA_TABLE.	Yes	
37 (Lowest)	Forward normally.	Yes	Successful Forwarding

§ §

NOTE: ***This page intentionally left blank.***

3.0 Pin Descriptions

3.1 Pin Overview

Figure 3-1 provides an overview of the pins used by both FM4000 series switches.

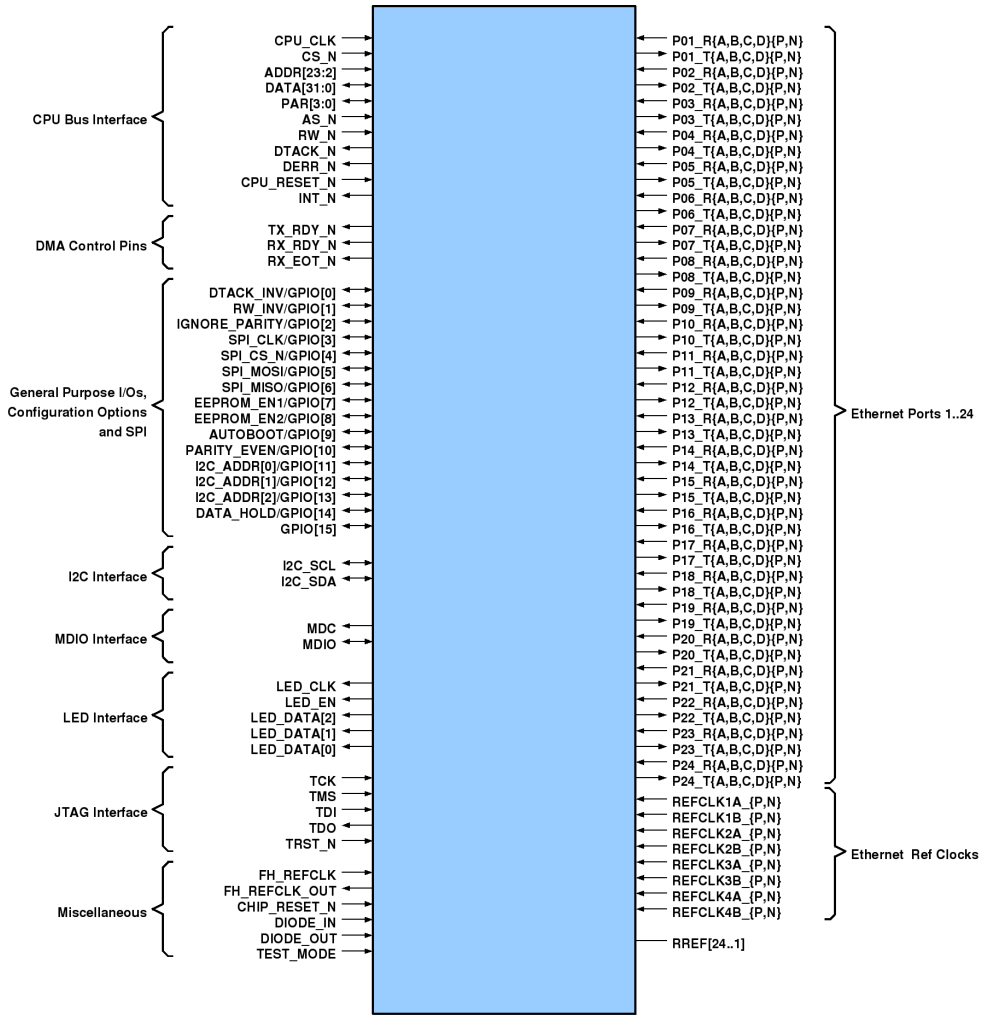


Figure 3-1 Pin Overview

3.2 Signal Name Convention

The signal name convention used is the following:

- **Signal Mnemonic** — The signal mnemonic is a generic name used as a prefix to identify the function of this pin.
- **Negative Logic** — Signal with inverted logic (0b=asserted, 1b=de-asserted) are designated by using the signal mnemonic following by "_N" suffix.
- **Differential Pairs** — Differential signals are designated by using the signal mnemonic followed by "_P" for the positive pin and "_N" for the negative pair.
- **Bus Designation** — A signal that is part of a bus is designated by using the signal mnemonic followed by [n] to designate the pin number in that bus. The entire bus or part of a bus is designated using the [M...N] designation. As an example, DATA[31..0] bus represents 32 pins designated DATA[0], DATA[1], etc...
- **Set** — A set of signals is referenced globally using name{M...N} or name{A,B,C,D}. The actual pin designation is formed by using the signal mnemonic and concatenating one of the values of the set. As an example, the pin set REFCLK{1..4}{A,B}{P,N} represents 16 pins designated REFCLK1AP, REFCLK1AN, REFCLK1BP, etc...
- **Types** — The following pin types are used:
  - IN: Input to the chip
  - OUT: Output from the chip
  - IN/OUT: May be operation as input or output
  - OD: Open drain output
  - OC: Open collector output
  - Power: A pin used for power
  - Ground: A pin used for ground
  - Sense: A pin used for sensing (no input/output concept)
- **Standard** — The following pin standard are used:
  - CML
  - TTL

3.3 Detailed Pin Descriptions

FM4000 pins are described in the following tables. Note that GPIOs pins are latched when CHIP_RESET_N is de-asserted to provide default configurations.

3.3.1 Ethernet Port Pins

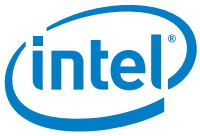
Pin Name	Type	Standard	Usage
P{1..24}_R{A,B,C,D}{P,N}	IN	CML (XAUI, SGMII)	Serdes receive ports. There are four pairs per port where each set {A,B,C,D} of four pairs can be mapped to LANE0..LANE3 or LANE3...LANE0 by software. Unused pins may be left floating; they are internally terminated.
P{1..24}_T{A,B,C,D}{P,N}	OUT	CML (XAUI, SGMII)	SerDes transmit ports (i=[1..24]). There are four pairs per port where each set {A,B,C,D} of four pairs can be mapped to LANE0..LANE3 or LANE3...LANE0 by software. Unused pins may be left floating; they are internally terminated.
REFCLK{1..4}{A,B}_{P,N)	IN	CML	Reference clocks for serdes. There are two possible reference clocks per block of 6 ports.
RREF[24..1]	Sense	N/A	Reference resistor to VDDX.

3.3.2 Power Pins

Pin Name	Type	Standard	Usage
VDD	Power	N/A	Core power (1.2V-1.3V).
VDDX	Power	N/A	Serdes core power (1.0 V-1.2 V).
VDD33	Power	N/A	TTL I/O power.
VTT{1..24}	Power	N/A	Serdes driver power (i=[1..24])
VDD33A	Power	N/A	Analog power for Frame Handler PLL.
VDDA	Power	N/A	Analog power for Serdes PLL. This is a filtered version of the VDDX.

3.3.3 Bus Interface Pins

Pin Name	Type	Standard	Usage
CPU_CLK	IN	LVTTL	Bus interface clock.
ADDR[23:2]	IN	LVTTL	Address inputs.
DATA[31:0]	IN/OUT	LVTTL (6 mA)	Data bus.
PAR[3:0]	IN/OUT	LVTTL (6 mA)	Data parity.
CS_N	IN	LVTTL	Chip select, active low..
AS_N	IN	LVTTL	Address strobe, input, active low..
RW_N	IN	LVTTL	Read/Write. Defines type of transaction (read, write) being requested. Polarity of this signal depends on the RW_INV strapping pin. When RW_INV is pull-down to ground, then read is active high while write is active low. Conversely, when RW_INV is pulled up to VDD33, then read is active low while write is active high.
DTACK_N	OUT	TTL (6 mA)	Data transfer acknowledge. Indicates the completion of a data transfer. This signal is actively asserted for one cycle when data transfer is ready during read or latched during write and then actively de-asserted for 1 cycle, the cycle after, and finally tri-stated for all other cycles. Polarity of this signal is determined by DTACK_INV strapping pin. If DTACK_INV is pulled down to ground at reset, DTACK_N is active low. If DTACK_INV is pulled up to VDD33 at reset, DTACK_N is active high.
DERR_N	OUT	LVTTL (6 mA)	Indicates write data parity errors. Only asserted (and valid) when DTACK_N asserted. Tri-stated otherwise.
TEST_RESET_N	IN	LVTTL	Intel internal use only. Users must tie this pin high.
INTR_N	OD	LVTTL (6 mA)	Interrupt. This pin is active low and asserted whenever an interrupt condition exist in the chip. The pin gets deasserted once all interrupt sources have been cleared. Pull-down current = 30 mA.



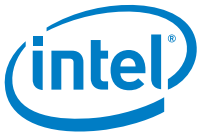
3.3.4 DMA Interface Pins

Pin Name	Type	Standard	Usage
TXRDY_N	OUT	LVTTL (4 mA)	Transmit FIFO ready. Asserted whenever the switch can accept a new word of data for packet transmission from the CPU to the network.
RXRDY_N	OUT	LVTTL (4 mA)	Receive data ready. Asserted whenever the switch has data in its receive FIFO for the CPU.
RXEOT_N	OUT	LVTTL (4 mA)	End of frame indication. Asserted while reading the last data word of a packet to indicate this is the last work of the packet.

3.3.5 GPIOs and Strapping Pins

Pin Name	Type	Standard	Usage
DTACK_INV/GPIO[0]	IN/OUT	LVTTL (6 mA)	Defines polarity of DTACK_N. If connected to ground, DTACK_N is active low. If connected to VDD33, DTACK_N is active high. On FM2000, this pin is permanently used for this function. On FM4000, this pin is sampled at reset to determine DTACK polarity and can be used as a general purpose I/O after reset (input by default).
RW_INV/GPIO[1]	IN/OUT	LVTTL (6 mA)	Defines polarity of RW_N pin when in EBI mode. When connected to ground, then read is active high while write is active low. Conversely, when connected to VDD33, read is active low while write is active high. On FM2000, this pin is permanently used for this function. On FM4000, this pin is sampled at reset and can be used as a general purpose I/O after reset (input by default).
IGNORE_PARITY/GPIO[2]	IN/OUT	LVTTL (6 mA)	Setting this pin to VDD33 disables parity checking on incoming write data. On FM2000, this pin is permanently used for this function. On FM4000, this pin is sampled at reset and can be used as a general purpose I/O after reset (input by default).
SPI_CLK/GPIO[3] SPI_CS_N/GPIO[4] SPI_MOSI/GPIO[5] SPI_MISO/GPIO[6]	IN/OUT	LVTTL (2 mA)	Used for either SPI or general purpose I/O pins. These pins are used as SPI when the switch retrieves its configuration from an external SPI EEPROM (if this enabled) and as GPIO after. The pin directions when the interface is used as SPI are: SPI_CLK: Out SPI_CS_N: Out SPI_MOSI: Out SPI_MISO: In ***Note:*** The pin SPI_MOSI was previously called SPI_SO in the FM2000 and the pin SPI_MISO was called SPI_SI.

Pin Name	Type	Standard	Usage
EEPROM_EN1/GPIO[7] EEPROM_EN2/GPIO[8]	IN/OUT	LVTTL (6 mA)	General purpose I/O pin. These pins are also latched at reset to define the location of the serial EEPROM that holds the configuration of the switch. **EN1 / EN2 / Usage** 0 0 Disable. A local CPU is responsible for configuring the switch. 1 0 SPI. 1 1 I^2C at address 0x50. 0 1 I^2C at address 0x51. The EEPROM is always assumed to require 3 bytes of addressing.
AUTOBOOT/GPIO[9]	IN/OUT	LVTTL (6 mA)	General purpose I/O pin. This pin is also latched at reset to define if the chip automatically starts booting or waits for CPU intervention.
PARITY_EVEN/GPIO[10]	IN/OUT	LVTTL (6 mA)	General purpose I/O pin. This pin is also latched at reset to determine the parity mode on the data bus after reset (high = even, low = odd). This pin must be pulled up or pulled down and cannot be left unconnected. Pull up for compatibility with FM2000 series devices.
I2C_ADDR[0]/GPIO[11]	IN/OUT	LVTTL (6 mA)	General purpose I/O pin. This pin is also latched at reset to set the I^2C slave address of the switch.
I2C_ADDR[1]/GPIO[12]	IN/OUT	LVTTL (6 mA)	General purpose I/O pin. This pin is also latched at reset to set the I^2C slave address of the switch.
I2C_ADDR[2]/GPIO[13]	IN/OUT	LVTTL (6 mA)	General purpose I/O pin. This pin is also latched at reset to set the I^2C slave address of the switch.
DATA_HOLD/GPIO[14]	IN/OUT	LVTTL (6 mA)	General purpose I/O pin. This pin is latched at reset to define DTACK and DATA behavior. If this pin is pulled up, DTACK and DATA (if read) are asserted when needed and remain asserted until CS is de-asserted. If this pin is pulled down, DTACK and DATA are asserted (if read) for one cycle, then DTACK gets actively de-asserted for one cycle and tri-stated after that. For compatibility with FM2000 devices, this pin should be pulled down. It should not be left unconnected.
GPIO[15]	IN/OUT	LVTTL (6 mA)	General purpose I/O pin.



3.3.6 I²C Pins

Pin Name	Type	Standard	Usage
I2C_SCL	OD	LVTTL (6 mA)	I²C Clock. Pull-down current = 30 mA.
I2C_SDA	OD	LVTTL (6 mA)	I²C data. Pull-down current = 30 mA

3.3.7 MDIO Pins

Pin Name	Type	Standard	Usage
MDC	OUT	LVTTL (6 mA)	MDIO clock.
MDIO	OD	LVTTL (6 mA)	MDIO data. Pull-down current = 30 mA.

3.3.8 LED Pins

Pin Name	Type	Standard	Usage
LED_CLK		LVTTL (2 mA)	Serial LED clock.
LED_EN		LVTTL (2 mA)	Serial LED enable. Asserted only on the first bit of the 36-bit frame.
LED_DATA0		LVTTL (2 mA)	Serial LED data. There are 3 bits sent per port. LED_DATA0 is used for ports 1..8, LED_DATA1 is used for ports 9..16 and LED_DATA2 is used for ports 17..24.

3.3.9 JTAG Pins

Pin Name	Type	Standard	Usage
TCK	IN	LVTTL	JTAG clock.
TMS	IN	LVTTL	JTAG control. Internal pull-down.
TDI	IN	LVTTL	JTAG data input. Internal pull down.
TDO	OUT	LVTTL (4 mA)	JTAG data output.
TRST_N	IN	LVTTL	JTAG reset. Internal pull-down. Tie low if not using JTAG

3.3.10 Miscellaneous Pins

Pin Name	Type	Standard	Usage
CHIP_RESET_N	IN	LVTTL	On FM4000, all strapping options are latched while the chip is in reset. Internal pull-down.
DIODE_IN	Sense	Analog	Temperature sensing diode.
DIODE_OUT			
FH_PLL_REFCLK	IN	LVTTL	Reference clock to PLL. Must be between 10 MHz and 70 MHz.
FH_CLKOUT	OUT	LVTTL (4 mA)	Copy of PLL output to FH. Presence of this signal is controlled by PLL configuration register.
TEST_MODE	IN	LVTTL	Defines the mode of operation of the chip. This pin is reserved by Intel for scan testing and must be connected to ground for normal operation.
CONT_EN	IN	LVTTL	Reserved for Intel use. Connect to ground for normal operation.

§ §

4.0 Ethernet Port Logic (EPL)

4.1 Overview

The Ethernet Port Logic includes the following 3 elements:

PMD/PMA:

Physical interface including the following functions:

- Serial transmission and reception
- Symbol locking
- BIST
- 8b/10b encoding and decoding
- Clause 37 and 73 auto-negotiation

PCS:

Reconciliation layer which includes the following functions:

- Lane alignment
- |R| suppressions for clock compensation
- Frame boundary decoding

MAC:

Packet reception and transmission including the following functions:

- Parse packet in reception. Forward pertinent fields to frame processor and forward entire payload to message array. Validate CRC and packet length.
- Retrieve packet from memory and modify packet according to information retrieved from frame processor.

4.2 SERDES

In many Intel Ethernet Switch Family device variants, some number of ports operate in 10M/100M/1G/2.5 Gb/s SGMII, single-SerDes modes. Other ports designed for 10 Gb/s operation use the quad-SerDes mode, each providing 2.5 Gb/s data throughput with a line rate of 3.125 Gb/s after line encoding. Any 10 Gb/s port can also be operated in single-SerDes mode at one of the lower data rates.

Note: For rates lower than 1 Gb/s, the SerDes still operates at 1 Gb/s. This is because, in accordance with the SGMII specification, the frame is elongated by byte replication such that the "byte rate" is always the same a 1 Gb/s operation.

The SerDes feature on-chip termination, providing 100-ohm differential input (Rx) and output (Tx) impedances. The REFCLK inputs also have on-chip, 100-ohm differential terminations.

The SerDes has electrical characteristics compatible with four separate standards:

- IEEE 802.3ae (XAUI)
- IEEE 802.3ak (CX4)
- IEEE 802.3ad (1000Base CX)
- IEEE 802.3ap (1000Base KX, 10GBase KX4, clause 37)

4.2.1 PLL and Reference Frequency

There are 8 CML low-jitter reference clocks for the SERDES, two per group of 6 ports. The mapping of the reference clocks to the physical ports is shown in the next table:

Clock Name	Ports
REFCLK1A and REFCLK1B	1, 3, 5, 7, 9, 11
REFCLK2A and REFCLK2B	2, 4, 6, 8, 10, 12
REFCLK3A and REFCLK3B	13, 15, 17, 19, 21, 23
REFCLK4A and REFCLK4B	14, 16, 18, 20, 22, 24

The EPL_PORT_CTRL defines the clock selected for each port (A or B). The actual SERDES speed is 10x the reference clock selected. The reference clock must be between 100 MHz and 350 MHz. A reference of 125 MHz is suitable for 1 Gb/s data rates (SerDes speed = 1.25 Gb/s) while a reference of 312.5 MHz is suitable for 10 Gb/s data rates (SerDes speed = 3.125 Gb/s). The REFCLK jitter must be kept below 0.1 UI (peak to peak).

4.2.2 Adjusting Drive Strength and Pre-emphasis

The nominal drive current is set to 20 mA through a 1 KΩ ±1% resistor. This value can be modified per port using the SERDES_CTRL_2 register to conserve power, or to increase signal strength. The drive current settings available are:

- 28 mA
- 20 mA
- 10 mA

These values are then digitally adjusted over a multiplier range of 0.6 to 1.3 per lane. The register SERDES_CTRL_2::DEQ allows control of pre-emphasis to adjust slew rate.

4.2.3 Configuration of Polarity and Lane Ordering

The SERDES can be configured for polarity and lane ordering per port per direction. The polarity, normal or reverse, is adjustable per port, per lane and per direction, and is configured in SERDES_CRL_3. The lane ordering, normal or inverted, is also adjustable per port and per direction, and is configured in PCS_CFG_1 (bits RI and TI). Two lane ordering variants are offered as shown in the following table:

	External Lane	
Internal Lane	**Normal**	**Inverted**
0	A	D
1	B	C
2	C	B
3	D	A

Note: In single serdes mode only lane A may be used.

4.2.4 Testing Facilities

Each SERDES lane has one BIST transmitter, one BIST checker and one internal loopback. The bits SERDES_TEST_MODE[BM] define the BIST mode and the loopback. The different modes are listed here:

000b = Disable, normal operation

001b= PRBS (x9+x5+x1), repeat every 511 cycles

010b = High frequency test data = 1010101010

011b = Test data = K28.5 (IDLE)

100b = Low frequency test data = 0001111100

101b = PRBS (x10+x3+x1), repeat every 1023 cycles

110b = PRBS (x9+x4+x1), repeat every 511 cycles

111b = PRBS (x7+x1), repeat every 127 cycles

The BIST transmitters on all 4 lanes are automatically enabled when the BIST mode is set to a value different than 0b. The BIST checkers are activated by writing a 0b into SERDES_TEST_MODE[x]::BS. The values in BIST_ERR_CNT count the number of errors received per lane.

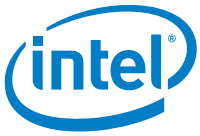
The BIST checker works properly only if symbols are aligned prior to start the checker. The symbol alignment is done by the PCS framer using the comma character as a reference which is the only character to use a series of five 1s or 0s in the normal flow of data. However, as the BIST transmitter may generate this test pattern, it is important to follow the following procedure:

1. Obtain symbol lock prior to enabling BIST transmitter (bits 3-0 of SERDES_STATUS).
2. Disable PCS framer (bit 6 of SERDES_TEST_MODE).
3. Set BIST mode (which automatically enabled the transmitter as well).
4. Enable BIST checker (bit 5 of SERDES_TEST_MODE).
5. Verify BIST_ERR_CNT to detect any error.

The loopback function loops TX pairs back into RX pairs and is controlled by SERDES_TEST_MODE::TestMode.

The EPL block is also capable to transmit a CJPAT pattern on the line. The pattern is defined in IEEE 802.3ae 48A.5 and reproduced here.

- START/PREAMBLE/SFD
  - <FB> 55 55 55 55 55 55 D5
- MODIFIED JTPAT SEQUENCE
  - 0B for 1 Octet (lane 0)
  - 7E for 3 Octets (lanes 1, 2, 3)
  - 7E for 524 Octets—Low Density Transition Pattern
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump



  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - AB for 4 Octets—Phase Jump
  - B5 for 160 Octets—High Density Transition Pattern
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - 7E for 528 Octets—Low-Density Transition Pattern
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - AB for 4 Octets—Phase Jump
  - B5 for 160 Octets—High-Density Transition Pattern
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
  - EB for 4 Octets—Phase Jump
  - F4 for 4 Octets—Phase Jump
- CRC
  - BD 9F 1E AB
- IPG
  - <FD>

4.2.5 Status and Interrupts

The SERDES locks on the |K| characters, this is represented in the SERDES_STATUS::SymbolLock. Also, the SERDES checks for signal detection (at least 75 mV) which is reported per lane in SERDES_IP::LOSn and globally (all lanes) in SERDES_STATUS::SignalDetect (note that in single lane mode, the signal detect of unused lanes is ignored and thus SERDES_STATUS::SignalDetect reflect only the status of the lane currently used).

4.2.6 Auto-negotiation

The FM4000 supports auto-negotiation as defined in clause 37 and clause 73 (introduced in 802.3ap). The EPLs may be placed in this mode after a link comes up to negotiate operation parameters with its link partners. The following registers are used to control this feature. Their usage is detailed in the clause 37 and clause 73 section.

- **AN_CTL**
  - Force AN to take over SERDES from normal traffic and BIST
  - Restart AN (self clear bit)
  - Define the mode of operation (clause 73, clause 37)
- **AN_STATUS**
  - AN status (lock, LCW count, state)
  - AN interrupt pending (timeout of negotiation completed)
- **AN_TIMEOUT** (16 bits)
  - Timeout on exchange before declaring AN failure
- **AN_TX_MSG0** and **AN_TX_MSG1** (48 bits each)
  - Messages to send to remote.
- **AN_RX_MSG0** and **AN_RX_MSG1** (48 bits each)
  - Messages received
- **AN_TX_TIMER**
  - Minimum time to transmit identical page before changing content.
- **Interrupts**
  - Interrupt pins for AN are included in PCS_IP interrupts

4.2.7 Clause 73

Clause 73 uses a relatively low-frequency to encode the auto-negotiation messages. Each message is 48 bits long and is encoded in a 106-bit frames using a Manchester encoding scheme. The frame is shown in Figure 4-1. These frames are exchanged between link partners before the link comes up as the speed and mode of operation is not determined yet. The clause 73 requires a 125 MHz reference clock.

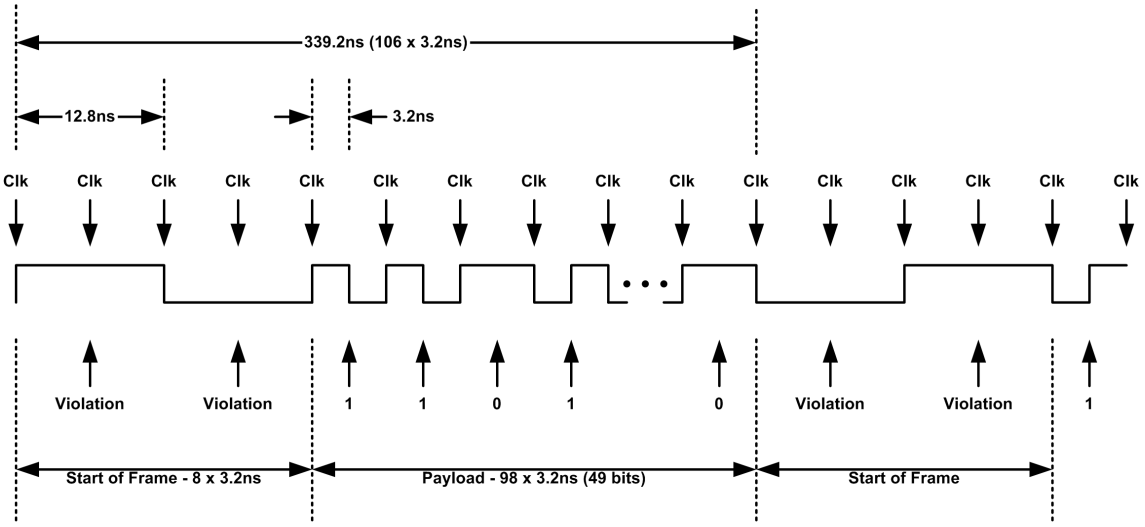


Figure 4-1 EPL Clause 73

The auto-negotiation process is detailed in clause 73 and summarized here:

1. The sender sends the mandatory control word frame repetitively setting the ACK bit to 0b. The control word includes an ID, which is a pseudo random number set by software and is static for the duration of this negotiation. The control word includes an NP bit to indicate if the sender has more pages to send.
2. The receiver expects to receive the mandatory control word and wait for 3 consecutive identical copies of the control word to be received.
3. The sender sends an acknowledge (ACK set to 1b) and echoed the ID from the link partner once 3 identical copies have been received.
4. The receiver waits for an ACK with an echoed ID equal to the one sent. The receiver checks for NP and if set, enters the next page negotiation.
5. If neither the sender nor the receiver has more page to send, the AN is completed.
6. If one or both has next pages to send, they do so at that point starting first by sending the ACK bit to 0b and then waiting for a valid next page before sending the next page. If one side as no next page to send, it sends the NULL page.

The control word format is listed in Table 4-1.

Table 4-1 EPL Clause 73 Control Word Format

Bits	Name	Transmitter	Receiver
4:0	Selector	Fixed (00001). Cannot be set by software.	Checked by hardware.
9:5	Echoed Nonce	Transmitter echoes received "transmitted Nonce" in this field once three valid identical CW have been received.	Checked by hardware to be equal to the transmitted "transmitted Nonce" (if ACK is 1b).
12:10	Pause Abilities	Set by software.	Pause abilities. Read and interpreted by software once negotiation exchange is completed.
13	Fault	Set by software.	Fault at link partner. Read and interpreted by software once negotiation exchange is completed.
14	ACK	Software presets if needed and hardware sets at appropriate time. Set by hardware once three valid identical CW have been received.	Check by hardware.
15	NP	Software presets if needed and hardware sets at appropriate time. Set by hardware once three valid identical CW have been received.	Check by hardware.
20:16	Tx Nonce	ID for this negotiation session. Set by hardware after each restart.	Link partner ID for this negotiation session.
45:21	Mode Abilities	Abilities of this transmitter. Set by software.	Abilities of link partner. Read and interpreted by software.
47:46	FEC Abilities	FEC abilities of this transmitter. Set by software.	FEC abilities of link partner. Read and interpreted by software.

The next page format is listed in Table 4-2.

Table 4-2 EPL Clause 73 Next Page Format

Bits	Name	Transmitter	Receiver
10:0	Message	Set by software.	Read and interpreted by software once negotiation exchange is completed.
11	Toggle	Set by software.	Checked by hardware to be toggling.
12	ACK2	Set by software.	Read and interpreted by software once negotiation exchange is completed.
13	MP	Set by software.	Read and interpreted by software once negotiation exchange is completed.
14	ACK	Software presets if needed and hardware follows once three valid identical CW have been received.	Check by hardware.

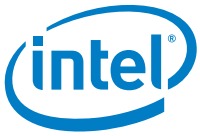


Table 4-2 EPL Clause 73 Next Page Format (Continued)

Bits	Name	Transmitter	Receiver
15	NP	Software presets if needed and hardware follows once three valid identical CW have been received.	Check by hardware.
47:16	Message	Set by software.	Read and interpreted by software.

The exchange with the link partner must include the control word and may include zero, one or more next pages. The basic methodology is to load the first control word into TX_MSG0 and allow an auto acknowledge by presetting the ACK bit into this message and then pre-load TX_MSG1 with the next message if there are any but have this message with the acknowledge bit preset to 0b so that a software intervention is required to complete the exchange. If there are no next pages sent or received, the software does not have to intervene, and an AN_DONE interrupt is received upon completion of the exchange.

The details of the algorithm are as follows:

1. Software initializes the MAC.
    a. The software initializes the serdes and sets up the BIST mode to force a low-frequency pattern on the line.
    b. The software initializes the hardware abilities by loading them into the AN_TX_MSG0 register (fields 'selector' must be set to 1b and 'echoed nonce' must be preset to 0b, hardware updates the echoed nonce at the appropriate time) and setting the ACK bit to 1b. The NP bit should be set either to 1b if there is one extra next page to send and 0b if there are no next page to send.
    c. The software loads a next page into AN_TX_MSG1 if there are any and sets the ACK bit to 0b and the NP bit to 1b or 0b depending of there are further pages to send after that one. The software initializes the 'select' bit in AN_CTL to 0b.
    d. The software then enables the AN_RX_MSG, AN_TIMEOUT, AN_FAIL and AN_DONE interrupt bit in PCS_IM and waits.
    e. The software then enables the auto-negotiation by setting the enable in the AN_CTL register.
    f. The software disables the BIST mode.
2. FM4000 performs control word exchange.
    a. The MAC starts by transmitting the AN_TX_MSG0 as the control word repetitively with ACK bit to 0b.
    b. The MAC waits for three identical valid control words and stores them into AN_RX_MSG0.
    c. The MAC echoes the NONCE received into its message, sets the ACK bit to 1b. If the MAC detects the NONCE received is equal to the NONCE transmitted, the MAC posts an AN_FAIL interrupt.
    d. The MAC waits for the ACK bit to be received to 1b; the updated word is stored in AN_RX_MSG0 (second interrupt posted). If NP is received as 0b and transmitted as 0b, the hardware posts an AN_DONE interrupt. If the NP is received as 1b or transmitted as 1b, an AN_RX_MSG interrupt is posted.
    e. The MAC then starts to transmit the next page stored in AN_TX_MSG1 (if NP is 1b in AN_TX_MSG0) or transmits null pages (if NP is 0b in AN_TX_MSG0). It transmits with the ACK bit to 0b and holds at this stage until software intervention because the AN_TX_MSG1:ACK bit is 0b. In the mean time, the hardware checks for received messages and, if there are at least three identical copies, stores them into AN_RX_MSG1 and posts an AN_RX_MSG1 interrupt.

3. Software waits for RX_MSG interrupts.
   a. Depending of the state of the select bit, the software expects the current message in one of the AN_RX_MSG register (new value stored), prepares the next message in the alternative AN_RX_MSG register (where ACK is set to 0b and NP set depending if there are further pages after the next one or not) and, when ready, sets ACK and ACK2 in the current message and then toggles select in the AN_CTL register to indicate to the hardware to do one next page.
4. Hardware sends next page.
   a. The MAC detects the toggle change in select. It then first completes the current exchange by sending ACK to 1b and a new ACK2 bit.
   b. The MAC waits for ACK 1b from remote, posts a new value in AN_RX_MSG1 and posts an AN_RX_MSG1 interrupt.
   c. The MAC starts to transmit next page sending an ACK bit.
   d. The MAC waits for the ACK bit to be received to 1b, the updated word is stored in AN_RX_MSG0. If NP is received as 0b and transmitted as 0b, the hardware posts an AN_DONE interrupt.
   e. The MAC then starts to transmit the next page stored in AN_TX_MSG1 (if NP is 1b in AN_TX_MSG0) or transmits null pages (if NP is 0b in AN_TX_MSG0). It transmits with the ACK bit set to 0b and hold. The control is returned to step 3.
5. Analyze messages received and apply negotiated values.
   a. The software detects the interrupt and reacts accordingly. If the interrupt is AN_DONE, then the software now has all pages in memory. If the interrupt AN_TIMEOUT or AN_FAIL is detected, software must wait for a while and go back to step 1.
   b. The software then applies the negotiated values and sets the serdes accordingly. If the link does not come up within a certain time, software must go back to step 1.

Note: The 802.3ap doesn't define the 2.5 Gb/s mode. However, the software has complete control over the announced capabilities (C, A, F bits) and the capabilities to exchange new pages with remote.

4.2.8 Clause 37

The process for clause 37 is very similar to the process for clause 73 except that a “page” is 16-bits long and is encoded as normal code words preceded by /C1/ or /C2/ code words. These pages are exchanged between link partners after the link comes up as the pages are transmitted or received using normal 8b/10b encoding/decoding and must be sent after the link is synchronized.

The control word format for clause 37 is listed in Table 4-3:

Table 4-3 EPL Clause 37 Control Word Format

Bits	Name	Transmitter	Receiver
4:0	Selector	Set by software.	Checked by software.
6:5	Duplex	Set by software.	Duplex abilities. Read and interpreted by software once negotiation exchange is completed.

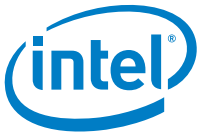


Table 4-3 EPL Clause 37 Control Word Format (Continued)

Bits	Name	Transmitter	Receiver
7:8	Pause	Set by software.	Pause abilities. Read and interpreted by software once negotiation exchange is completed.
9:11	Reserved	Set by software.	Checked by software.
12:13	Fault	Set by software.	Fault at link partner. Read and interpreted by software once negotiation exchange is completed.
14	ACK	Software presets if needed and hardware sets at appropriate time. Set by hardware once three valid identical CW have been received.	Check by hardware.
15	NP	Software presets if needed and hardware sets at appropriate time. Set by hardware once three valid identical CW have been received.	Check by hardware.

The next page format is listed in Table 4-4:

Table 4-4 EPL Clause 37 Next Page Format

Bits	Name	Transmitter	Receiver
10:0	Message	Set by software.	Read and interpreted by software once negotiation exchange is completed.
11	Toggle	Set by software.	Checked by hardware to be toggling.
12	ACK2	Set by software.	Read and interpreted by software once negotiation exchange is completed.
13	MP	Set by software.	Read and interpreted by software once negotiation exchange is completed.
14	ACK	Software presets if needed and hardware follows once three valid identical CW have been received.	Check by hardware.
15	NP	Software presets if needed and hardware follows once three valid identical CW have been received.	Check by hardware.

The process of exchanging messages is exactly the same as for clause 73 except for the following details:

- In clause 73, any message must be transmitted at least three times before changing any content of the message. In clause 37, the specification defines an interval of time rather than a specific number of repeats. This time is configured through the AN_TX_TIMER register.
- There is no NONCE concept in clause 37 and this clause 73 feature is thus disabled.
- There is no fixed header in clause 37 so the hardware still reports any change in the concept, but does not attempt to validate the content.

- The register AN_CTL::AN_Mode offers two clause 37 modes, with transmitter enable and with transmitter disable. The first mode is suitable when FM4000 initiates the negotiation while the second mode is suitable when FM4000 is silently listening to a negotiation request but is not initiating one. In that second mode, the software waits for an interrupt announcing reception of configuration word, and then changes the mode to enabled the transmitter to actually perform the negotiation.

4.3 Physical Coding Sub-layer (PCS)

4.3.1 Frame Format

The frame format in 10 GbE mode is shown in the next table. The value of Dp (Data preamble) is 55h (symbol D21.2), the value of Ds (Data start) is D5h (symbol D21.6). The PCS layer always expects a strict 8-symbol preamble (includes 1x|S|, 6x|Dp| and 1x|Ds|).

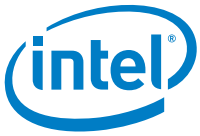
LANE 0	S	Dp	D	D	...	D	A	R	K	R
LANE 1	Dp	Dp	D	D	...	T	A	R	K	R
LANE 2	Dp	Dp	D	D	...	K	A	R	K	R
LANE 3	Dp	Dp	D	D	...	K	A	R	K	R

The frame format in 1 GbE mode is shown in the next table.The PCS layer is programmable (bit SP of PCS_CFG) to either expect a strict 8-byte preamble (bit SP is set to 0b) or a variable size preamble (bit SP is set to 1b). When configured for supporting a variable size preamble, the PCS accepts as a valid preamble any starting sequence of 1x|S|, [0..6]x|Dp|,1x|Ds|).

LANE 0	S	Dp	...	Dp	D	D	...	D	T	R	I

In the 100M mode, the PCS searches for |S| and then samples incoming data every 10 cycles. In the 10M, the PCS samples incoming data every 100 cycles.

Finally, the PCS supports 4-bit miss-alignment in the data part of the frame (|DP| to last |D|). This is enabled using the ND option of PCS_CFG. When this option is enabled, the PCS accepts 0xD5 or 0x5?-0x?D as a valid start of frame and automatically realigns the frame before sending it to the MAC layer. This is particularly useful when the SGMII interface comes from a device that did an MII-SGMII conversion and the size of the pre-amble on the MII was not a multiple of 8 bits. This should only be useful in 10M/100M.



4.3.2 Local and Remote Faults

The PCS level is responsible to handle local faults and remote faults. Local faults are typically sent by the PHY to the PCS to indicate that it detected a fault in the receiver. remote faults are either received or sent.

The PCS does the following:

- Upon reception of at least 4 local fault symbols within a 128 cycles period, the PCS enters into a local fault detect state. The local fault state clears itself when 128 cycles occur without receiving any local fault symbol.
- While in local fault, if the PCS_CONTROL(EF) is set, then the transmitter transmits remote fault symbols to the link partner and the data coming from the MAC is discarded.
- While in remote fault, the transmitter transmits idle symbols to the link partner, the data coming from the MAC is discarded.
- The PCS layer could also be configured by PCS_CONTROL(EL) to transmit remote fault when the link goes down regardless if the local faults are received or not.

Normally, only one type of fault is received at any time. In the unlikely situation where two faults are received, the local faults take precedence.

A cycle is 4 bytes in both 1 GbE and 10 GbE mode.

Note: The PCS implementation offers configuration of the local fault and remote fault symbols (register PCS_CFG_2 and PCS_CFG_3), however compliance to the spec 802.3ae requires that these registers are kept to their default value.

4.3.3 Messaging

The PCS supports simple in-band messaging. Up to 24 bits may be transmitted or received.

Upon receiving an FSIG symbol (IEEE 802.3 Signal Ordered Set), the PCS stores the lower 24 bits into the PCS_CFG_5 and sets the bit PCS_IP[FD] to indicate that an FSIG symbol has been detected. The corresponding interrupt bit mask may be programmed to cause an interrupt when the FSIG is received.

Also, the PCS could be used to transmit an FSIG message. The lower 24 bits are placed in the PCS_CFG_4 register and the user sets the PCS_CONTROL[FS] to force the transmission of the FSIG symbol. This bit self-clears once the symbol is sent. The PCS_IP[FS] is set to indicate that the FSIG has been sent and a corresponding interrupt bit mask may be programmed to cause an interrupt when the FSIG has been sent.

4.3.4 Balancing IFGs

On transmit, it may be necessary for the transmitter to modify the length of the inter-frame gap (IFG) to align the Start control character (first octet of preamble) on lane 0. This is accomplished in one of the following two ways:

- Guarantee minimum IFG. A MAC implementation may incorporate this RS function into its design and always insert additional idle characters to align the start of preamble on a four byte boundary. Note that this reduces the effective data rate for certain packet sizes separated with minimum inter-frame spacing.
- Meet average minimum IFG. Alternatively, the transmitter may maintain the effective data rate by sometimes inserting and sometimes deleting idle characters to align the Start control character. When using this method the RS must maintain a Deficit Idle Count (DIC) that represents the cumulative count of idle characters deleted or inserted. The DIC is incremented for each idle character deleted, decremented for each idle character inserted, and the decision of whether to insert or delete idle characters is constrained by bounding the DIC to a minimum value of zero and maximum value of three. Note that this may result in inter-frame spacing observed on the transmit XGMII that is up to three octets shorter than the minimum transmitted inter-frame spacing specified in Clause 4; however, the frequency of shortened inter-frame spacing is constrained by the DIC rules. The DIC is only reset at initialization and is applied regardless of the size of the IPG transmitted by the MAC sublayer. An equivalent technique may be employed to control RS alignment of the Start control character provided that the result is the same as if the RS implemented DIC as described.

The FM2000 series devices support those two methods. The first method is supported by setting the PCS_CFG_1[DE] to 0b. The second method is supported by setting the PCS_CFG_1[DE] to 1b.

4.4 MAC

The MAC layer is responsible for the following actions:

In the receive direction:

- Writes the entire frame into memory including CRC.
  - The scheduler will have pushed a pointer to the location to write the frame ahead of time.
- Parses the incoming frame and send pertinent information from the header to the frame control.
- Validates the CRC and frame length at the end of frame and indicate to the frame control if:
  - The frame was received with a bad CRC.
  - The frame is too short or too long. Frames that are too long are terminated within a few. bytes after at the maximum length configured has been reached and a bad CRC is appended to the frame when written into memory.
  - A PHY error (disparity error, symbol error) was detected while the frame was received.
  - There was an error detected while writing the frame into memory. Note that the CRC is also marked as incorrect in this case.

In the transmit direction:

- Receives information from the scheduler about the location of the frame to transmit and the changes needed (if any) to be done on the frame.
- Retrieves frame from memory and starts transmission while data is retrieved from the memory. The transmitter changes content of the frame as it being transmitted and compute a new CRC while frame is transmitted.
  - The transmitter also computes the CRC on the frame as retrieved from memory to ensure the data has not been corrupted by alpha particles. This CRC is checked with the final CRC retrieved from memory at the end of frame to indicate if the frame is still valid.
  - The transmitter gives the frame a bad CRC if the receiver has marked it for discard, or if it has a bad CRC when read from the memory, or if the transmitter underflows. Otherwise, it gets a good CRC.
  - The transmitter also pads the frame to MinFrameSize if the PadRuntFrames is set. The padding is done with random bytes, and the CRC is updated accordingly.

4.4.1 Frame Parsing

The job of the frame parser is to extract up to 78 bytes of relevant header fields from the frame and forward it to the Frame Handler for further processing. Section 2.2 gives a detailed view of frame parsing for layer 2 frames, layer 3 frames and for deep packet inspection.

4.4.2 Parser Errors

A header parse error means either that the frame is illegal or malformed, or that the parsing function is unable to report valid results for some other reason.

The following conditions cause header parse error:

- If the frame terminates in the middle of the header. The last 4 bytes of the frame are always treated as the CRC. At 4-bytes before the end of the frame, the frame must not still be in:
  - Any part of the L2 header (Ethertype or before), including VLAN, F64,and other tags.
  - If ParseL3 is enabled:
    - Any part of the IPv4 header, including IPv4 options.
    - Any part of the IPv6 header, including IPv6 options (i.e. IPv6 Hop-by-Hop Option, Routing. Header for IPv6, Fragment Header for IPv6, Destination Options for IPv6, or Authentication Header).
  - If ParseL3 and ParseL4 are enabled:
    - The first 16 bytes of the TCP header (see RFC 3128).
    - The first 8 bytes of the UDP header (see RFC 3128).
- If an IPv4 frame has a bad IP checksum, with ParseL3 enabled.
- If a TCP frame has a fragment offset of 1, with ParseL3 enabled (see RFC 3128).

The following conditions also can cause header parse error, but are not reachable under the default configuration of MAC_CFG_1.MaxFrameSize and MAC_CFG_3.MaxParsingDepth:

- If MaxParsingDepth is reached during any part of the L2 header, or (if ParseL3 is enabled) during the IPv4 or IPv6 header, _not_ including IP options.
- If MaxFrameSize is reached while header parsing is ongoing.

Finally, the following chip malfunction would also cause header parse error:

- If RX overflow (indicated by MAC_IP.RX_S2A_Overflow) occurs during header parsing.

Invalid input symbols or other PCS/PMA-level errors during the frame are treated as causing frame truncation, and can lead to header parse errors in the same way that early frame termination can.

An IP version number other than 4 or 6 does not lead to parse error; it simply means that the parser does not treat the frame as IP. Specifically, the frame is treated as IPv4 only if it has Ethertype 0x0800 _and_ version 4, and as IPv6 only if it has Ethertype 0x86DD_and_ version 6. If it has an IP Ethertype but not the corresponding version number, it is treated as any other unknown L3 protocol.

The parsing does not directly check that the IPv4 header length is greater than or equal to 5; it always assumes a minimum of 5 words in the IPv4 header. However, an IPv4 header length of 0-4 confuses the IP checksum calculation, and very likely leads to parse error for that reason.

Frames with parse errors are dropped by default, with a “ParseErr” bit set in the action mask.

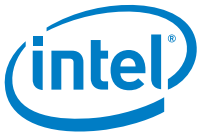
Parse error frames are included in the RX counters as a non-erroneous frame. However, parse error frames are counted individually in the Group 6 counters. Parse error frames also do get counted by the policers, as are all other error frames.

4.4.3 Deep Packet Inspection for IP Frames

The switch assumes the presence of a layer 4+ payload if the packet length is non-zero. The protocols recognized by the switch are TCP (protocol = 0x06), UDP (protocol = 0x11) and two other configurable custom protocols. Note that if the frame is IPv6, the deep inspection does not overwrite the upper bits of the SIP and DIP, the parser just stops after L4D. The processing per protocol is the following:

TCP:

- The parser forwards the source and destination ports to the frame processor.
- The parser can also be configured (per port) to forward the HL/RSV/OPTIONS half-word to the frame processor. If this is done, this half-word is placed in the L4A field and the remaining half-words start at L4B.
- The parser always skips over all TCP options present.
- The parser can also be programmed to forward up to 16 half-words (32 bytes) from the TCP payload for deep packet inspection. The half-words selected to be forwarded are configurable on a per-port basis using a 48-bit vector which indicates which of the next 48 half-words are passed to the frame processor.
  - The parser always stops forwarding after having forwarded either 16 half-words of deep inspection or 78 bytes total.
  - The LSB of the 48-bit vector corresponds to the first half-word after the TCP header.



UDP:

- The parser forwards the source and destination ports to the frame processor.
- The parser can also be programmed to forward up to 16 half-words from the UDP payload for deep packet inspection. The half-words selected to be forwarded are configurable on a port by port basis using a 48-bit vector which indicates which of the next 48 half-words are passed to the frame processor.
  - The parser always stops forwarding after having forwarded either 16 half-words of deep inspection or 78 bytes total.
  - The LSB of the 48-bit vector corresponds to the first half-word after the UDP header.

Custom Protocols:

- The parser also supports two generic custom protocols (protocol ID to be configured by user on each port). If this protocol is recognized, then the first two 16-bit half-words of the packet are forwarded to the frame processor and mapped to source and destination port.
- The parser can then be programmed to forward up to 16 half-words from this protocol for deep packet inspection. The words selected to be forwarded are configurable on a per-port basis using a 48-bit vector which indicates which of the next 48 half-words are passed to the frame processor.
  - The parser always stops forwarding after having forwarded either 16 half-words of deep inspection or 78 bytes total.
  - The LSB of the 48-bit vector corresponds to the third half-word of this protocol (after those mapped to the source and destination port).

Other Protocols:

- The first two 16-bit half words of the packet are forwarded to the frame processor and mapped to source and destination port.
- The parser simply forwards up to 16 more half-words to the frame processor for deep packet inspection. This is configurable per port in the EPL.

The 16 half-words of deep packet inspection are mapped to the following FFU fields:

Half-word	Mapped to FFU	Half-word	Mapped to FFU
0	L4A	8	SIP[63:48]
1	L4B	9	SIP[47:32]
2	L4C	10	DIP[127:112]
3	L4D	11	DIP[111:96]
4	SIP[127:112]	12	DIP[95:80]
5	SIP[111:96]	13	DIP[79:64]
6	SIP[95:80]	14	DIP[63:48]
7	SIP[79:64]	15	DIP[47:32]

4.4.4 Deep Packet Inspection for Non-IP Frames.

If the packet is not an IP packet, then the switch can be programmed to forward the first 16 half-words to the FFU by setting the PARSE_CFG.SendOtherL3 register. The mapping used in this case is the following:

Half-word	Mapped to FFU	Half-word	Mapped to FFU
0	SIP[127:112]	8	DIP[127:112]
1	SIP[111:96]	9	DIP[111:96]
2	SIP[95:80]	10	DIP[95:80]
3	SIP[79:64]	11	DIP[79:64]
4	SIP[63:48]	12	DIP[63:48]
5	SIP[47:32]	13	DIP[47:32]
6	SIP[31:16]	14	DIP[31:16]
7	SIP[15:0]	15	DIP[15:0]

4.4.5 EPL Reset

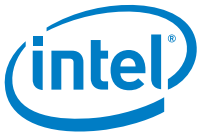
All EPLs are in the reset state during boot. They stay in reset until the API or the application software brings them out of reset during boot. The EPL can be reset as follows:

First time since core reset:

1. Set EPL_PORT_CTRL[i]::IntializeN to 0b.
2. Set EPL_PORT_CTRL[i]::ClockSelect to desired clock (A or B).
3. Set EPL_PORT_CTRL[i]::ResetN to 0b.
4. Start serdes.

Subsequent reset:

1. Shutdown reception by setting EPL_SERDES_CTRL_2::LaneReset, PLLReset and PowerDown to 1s.
2. Ensure EPL is not idling. The transmitter could be placed on hold by setting bandwidth on this port to 0.
3. Wait for at least one frame to drain.
4. Ensure all interrupts are disabled in the EPL (set all EPL IM registers to 1s).
5. Set EPL_PORT_CTRL[i]::ResetN to 1b.
6. Set EPL_PORT_CTRL[i]::IntializeN to 1b.
7. Set EPL_PORT_CTRL[i]::ClockSelect to desired clock (A or B).
8. Set EPL_PORT_CTRL[i]::ResetN to 0b.
9. Start serdes.



The serdes are started using the following procedure:

1. If not already done, shutdown serdes by setting SERDES_CTRL_2::LaneReset, PLLReset and PowerDown to 1s.
2. Program desired equalizer and drive strength in SERDES_CTRL_1.
3. Program desired lane ordering and polarity in PCS_CFG_1.
4. Start Lane A and B (only required if either lane C or D is used).
    a. Power up lane A and B.
    b. Wait 10 µs.
    c. Take PLL_AB out of reset.
    d. Wait 10 µs.
    e. Take lane A and B out of reset.
    f. Wait 10 µs.
5. Start Lane C and D (only required if either lane C or D is used).
    a. Power up lane C and D.
    b. Wait 10 µs.
    c. Take PLL_CD out of reset.
    d. Wait 10 µs.
    e. Take lane C and D out of reset.
    f. Wait 10 µs.

All other parameters may be changed at any time. For auto-negotiation, refer to the auto-negotiation section for changes to this procedure.

§ §

NOTE: ***This page intentionally left blank.***

5.0 Chip Management

5.1 Overview

The FM4000 offers the following functionality:

- External de-multiplexed address/data bus
- Up to 133 MHz
- Even/odd/no parity
- SPI
- I^2C (master and slave)
- MDIO (clause 22 and 45)
- GPIO
- JTAG
- SCAN (not detailed in this section)
- PLL

The chip management block diagram is shown in Figure 5-1.

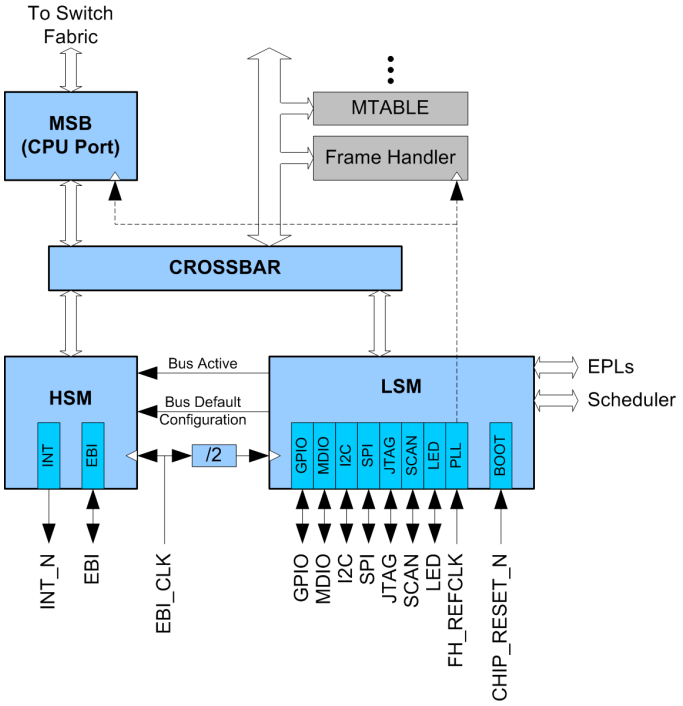


Figure 5-1 Switch Management Block Diagram

It contains the following three independent synchronous units connected through a crossbar:

- **Management-Switch Bridge (MSB)**— This block interfaces to the switch element and is the CPU port to the fabric to transmit or receive frames from the network.
- **High Speed Management (HSM)** — This block contains the CPU interface.
- **Low Speed Management (LSM)** — This contains all low-speed management interfaces and functionality and is responsible for system booting.

The HSM is the first unit active in the system, and includes the following functions:

- Interrupt Controller
- CPU Interface Controller

The LSM is generally the second unit active in the system, and includes the following elements:

- BOOT Controller
- EPL access
- Scheduler access
- Crossbar access
- PLL
- GPIO Controller
- MDIO Controller

- I^2C Controller
- JTAG Controller
- LED Controller
- SPI Controller (only used by Boot controller)

There are five possible requestors coming into the LSM. The priority in descending order is:

1. BOOT
2. XBAR_IF
3. EPL
4. I^2C
5. JTAG

Requestors access the responders according to the priority in the list above. The priority only makes a difference for multiple simultaneous requests. There is no fairness guarantee.

Responders are:

1. XBAR_IF
2. EPL
3. SCHEDULER
4. LED
5. SPI
6. I^2C
7. MDIO
8. GPIO

5.2 Boot Controller and Chip Reset

The FM4000 has several chip reset domains, shown in Figure 5-2.

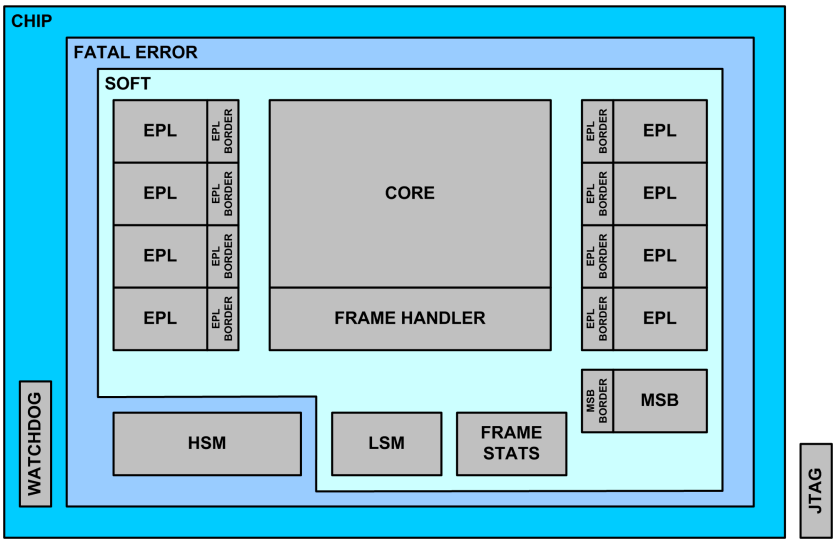
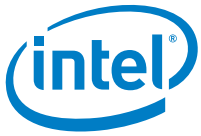


Figure 5-2 Chip Reset Domains

Each of these have differing effects and implementations as described below.

- **Chip Reset** — Controlled via an external pin, CHIP_RESET_N. When this signal is lowered, the reset signals to all domains are asserted except for JTAG. This signal should be set low when power is applied to FM4000 to assure reliable operation. When CHIP_RESET_N is deasserted, all modules are taken out of reset except the Ports (EPL and MSB) and the FH. These modules must be taken out of reset by software. The hold time on this pin after power is applied is 10 ns (min).
- **Fatal Error Reset** — The fatal error reset domain includes the entire chip minus the watchdog.
  - FM4000 includes a watchdog that can be enabled (register WATCHDOG_CFG) to do a self reset when a fatal error is detected (such as a parity error in a Scheduler SRAM). The watchdog circuit has it own reset domain which includes the minimal amount of logic that must remain operational while the chip resets. In particular, this includes a counter which sets the duration of the reset, the fatal error code and a counter of the number of fatal errors. The Watchdog logic resides in them HSM since this module handles all chip interrupts, which is the mechanism by which such fatal errors are indicated. The watchdog asserts the switch reset for 256 cycles.
- **SOFT Reset** — The SOFT_RESET register controls two reset domains; Core and Frame Handler. The Core reset self clears after 256 cycles and includes the LSM module (i.e. the LSM reset domain is tied to the Core domain).The Frame Handler reset (along with the MSB and EPLs reset domains controlled by the LSM) is automatically asserted when the CORE reset is asserted and remains asserted until software clears it. If the switch is reset using the SOFT_RESET register or through the watchdog, the GPIO are not sampled.
- **JTAG Reset** — The JTAG domain is on its own and only applies to the JTAG circuitry.



- **Module Resets** — The EPL's are individually resettable using the EPL_PORT_CTRL register which allows selection of the clock domain as well as control of the EPL core and EPL border resets. The MSB reset is controlled using EPL_PORT_CTRL[0].

The entire reset process is detailed in the following paragraphs.

1. Asserting CHIP_RESET_N or the self-asserting WATCHDOG causes the HSM module to go in reset and, by consequence, causes the Core and LSM to go into reset. The HSM places the RXRDY_N, TXRDY_N and RXEOT_N in tri-state mode.
2. De-asserting CHIP_RESET_N or the self-de-asserting WATCHDOG causes the following to happen.
   — The GPIO pins are sampled, they provide the initial configuration for various elements.
   — The core reset control and LSM reset control are kept asserted for 256 cycles and then self-clear.
3. Deassertion of the core and LSM reset causes the boot controller located in LSM to start to boot the chip. The boot controller does the following:
   — Transfer the content of the fusebox to the VPD registers.
   — Sample AUTOBOOT, EEPROM_ENABLE, EEPROM_ENABLE2 to determine the type of EEPROM used, if booting from EEPROM is enabled and if the booting must start without waiting for the CPU.
   — If booting from EEPROM is enabled and AUTOBOOT is enabled, the boot controller reads and executes the instructions in the EEPROM until the last instruction (END) is retrieved.
   — If booting from EEPROM is disabled or AUTOBOOT is deasserted, the boot controller is stalled until boot instructions are received from the CPU through the BOOT_CTRL register. Progress for each command is reported through the BOOT_STATUS register.
     - AUTOBOOT has no effect if fetching from EEPROM is not enabled.
   — In both cases (CPU assisted boot or EEPROM assisted boot), the boot steps are as follows:
     1. Initialize PLL.
     2. Wait for PLL lock.
        — The maximum lock time is 80ms. The CPU may poll the PLL_STATUS register if desired to reduce waiting time.
     3. Unreset all modules.
        — Write 0b to SOFT_RESET.
     4. Enable MSB and EPL0 by setting ELP_PORT_CTRL[0].ResetN to 1b. Also set ELP_PORT_CTRL.InitializeN to 1b.
     5. Enable FFU by setting SYS_CFG_8, bit 0 to a 1b.
     6. Execute asynchronous RAM repair:
        — For the BOOT ROM:
          - Send command BOOT(“Repair asynchronous RAM”).
        — For the CPU:
          - Write command “Repair asynchronous RAM” into BOOT_CTRL::Command register.
          - Wait for BOOT_STATUS::CommandDone to return to 1b.

7. Execute synchronous RAM repair:
   — For the BOOT ROM:
      • Send command BOOT(“Repair synchronous RAM”).
   — For the CPU:
      • Write command “Repair synchronous RAM” into BOOT_CTRL::Command register.
      • Wait for BOOT_STATUS::CommandDone to return to 1b.
8. Program FFU_INIT_SLICE.
9. Wait for BOOT_STATUS::CommandDone to return to 1b.
10. Wait for BOOT_STATUS::CommandDone to return to 1b.
11. De-assert FH resetExecute scheduler initialization.
   — For the BOOT ROM:
      • Send command BOOT(“Initialize scheduler”).
   — For the CPU:
      • Write command “Initialize scheduler” into BOOT_CTRL::Command register.
      • Wait for BOOT_STATUS::CommandDone to return to 1b.
12. Initialize memory.
   — For the BOOT ROM:
      • Send command BOOT(“Execute memory initialization”).
   — For the CPU:
      • Write command “Execute memory initialization” into BOOT_CTRL::Command register.
      • Wait for BOOT_STATUS::CommandDone to return to 1b.
13. Optionally disable FFU.

5.3 EEPROM Boot Format

The EEPROM boot format consist of a series of instructions encoded as shown in Figure 5-3.

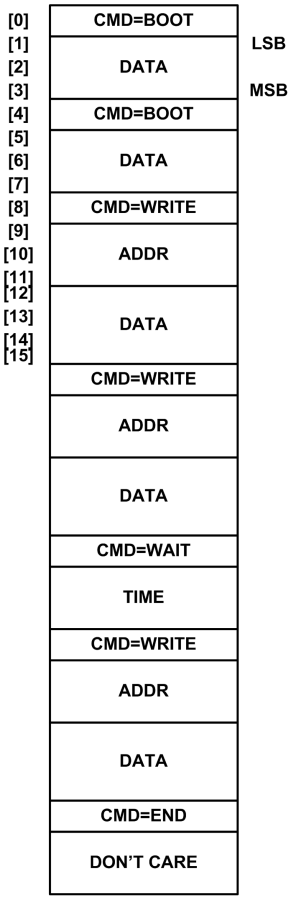


Figure 5-3 Serial EEPROM Format

The SPI controller assumes 3-byte addressing and the I^2C controller assumes 2-byte addressing. It is also possible to use a SPI EEPROM with 2-byte addressing if the data is shifted up by one byte (the lowest byte of the EEPROM is unused). If the switch is reset during boot, the EEPROM should also be reset by cycling the power supply. For more details on how to program the EEPROM, see the FM4000 "EEPROM Programming Application Note".

The command encoding is the following:

Command	Code
WRITE	0x01
WAIT	0x10
BOOT	0x11
END	0xFF

The waiting time is in LSM clock cycles (half the CPU_CLK cycle rate). The data and addresses are always encoded most significant byte (MSB) first. Examples illustrating the byte ordering are:

WRITE 0x40104 0x19 (01 04 01 04 00 00 00 19)

DELAY 10 usec (10 00 00 a6)

BOOTCMD 3 (11 00 00 03)

The boot commands data is as follows:

00b = Initialize scheduler

01b = Repair asynchronous RAM

10b = Initialize memory

11b = Repair synchronous RAM

5.4 Interrupt Controller

Interrupt handling includes the following registers:

- **Interrupt Pending Registers** — The interrupt pending registers contain one bit per interrupt source which gets set when an interrupt condition is detected. The interrupt condition is cleared by writing the corresponding bit to 1b, while writing a 0b has no effect.
- **Interrupt Mask** — The interrupt mask is identical to the interrupt pending and mask out the interrupt source if desired. A masked interrupt (setting bit to 1b) does not post an interrupt.
- **Interrupt Detect Registers** — The interrupt detect registers are a collective representation of the presence of an interrupt source in the system or sub-system. The register INTERRUPT_DETECT is the top register and the bit 31 of that register can be used to mask the global interrupt out.

Each interrupt is either considered a potential fatal error or not. If a potentially fatal interrupt is unmasked, a corresponding fatal code is sent to the interrupt controller and stored into the LAST_FATAL_CODE register, and a watchdog reset is triggered when those interrupts are detected.

Figure 5-4 shows the interrupt hierarchy.

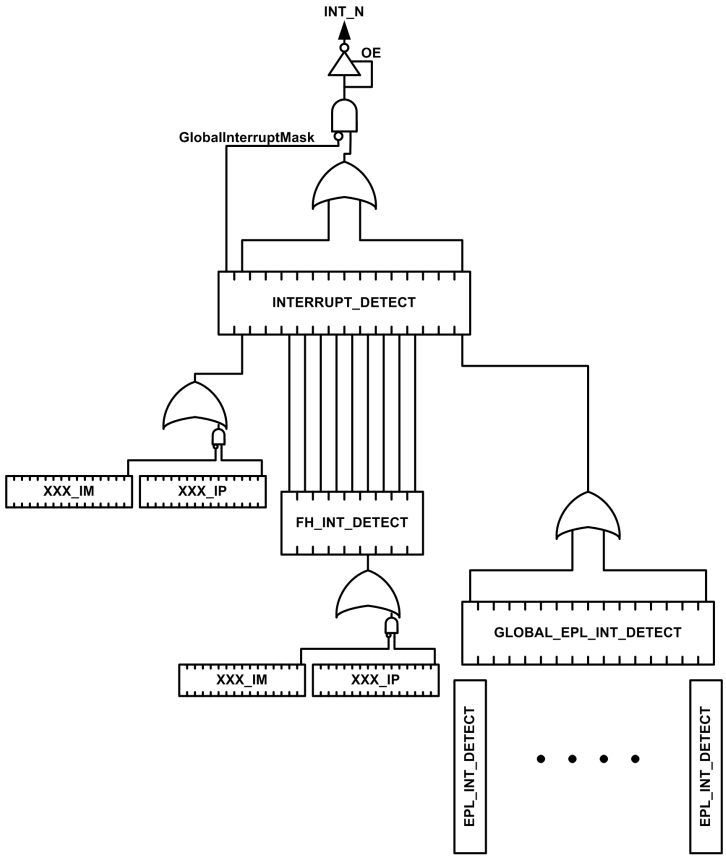


Figure 5-4 Interrupt Hierarchy

The main registers are:

- **INTERRUPT_DETECT** — Located in the HSM. Report the source.
- **LAST_FATAL_CODE** — Located in the HSM. Reports the last fatal code logged. The content of this register is cleared by CHIP_RESET_N or when the CPU is writing a 0b. If the CPU writes anything different, the chip self-resets. The content of this register is not cleared on watchdog reset. The fatal interrupts and associated fatal codes are listed in Table 5-1.
- **FATAL_COUNT** — Located in the HSM, defines the number of times the chip has reset since the last code. The content of this register is incremented for every watchdog reset and is cleared when CHIP_RESET_N is asserted. The CPU can optionally write this register.
- **GLOBAL_EPL_INT_DETECT** — Located in the LSM. Reports which EPL has a pending interrupt.
- **EPL_INT_DETECT** — Located in each EPL, reports the source of interrupt within the EPL (SERDES, PCS, MAC).
- **FH_INT_DETECT** — Located in the FH, reports the source of interrupt within the frame handler. All bits of this register are replicated in the INTERRUPT_DETECT register and the access of this register is normally not required to detect the final interrupt source.

Table 5-1 Fatal Interrupts and Associated Fatal Code

Fatal Interrupt	Fatal Code
Frame Handler Forwarding Table Parity Error	0xC0
Frame Handler Forwarding Table Parity Error	0x22
Scheduler Parity Error (RxQueueLink)	0x20
Scheduler Parity Error (TxQueueLink)	0x21
MSB: IBM CRC Error	0x41
MSB: Invalid IBM Op Error	0x42
MSB: Both of the above errors	0x43

All memory blocks have parity error detections to detect soft errors cause by alpha particles or other phenomena except for the CAMs and the message array.

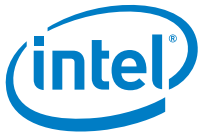
For the message array, the original frame received from the port is entirely saved in the message array along with the CRC. When the frame is transmitted, the EPL retrieves the original packet from memory, and computes the CRC while retrieving and compare to the original CRC at the end of the frame. If the CRC is valid, it is assumed that the packet has not been corrupted. If the CRC is invalid, the packet is transmitted with a bad CRC, and this event is counted in the EPL as a memory corruption event, allowing the software to track those events.

For the CAMs (in FFU and port mapping unit), the lookup circuitry does not allow support for parity as the CAM is a full ternary CAM where only parts of the bits are read. As a consequence, it is recommended for the software to maintain a copy of the content of the CAM and implement a CAM sweeper that periodically compares the content of the CAM in the chip with the one in the memory, and refreshes the CAM entry should it be corrupted. Note that the frequency of errors is very low and a low priority task sweeper should work fine.

For all other RAM blocks, the switch includes an extra parity bit to each entry. The parity bit should be set to 0b by the software, and the switch automatically computes the right value before writing the content into the memory. The switch then validates the parity every time the entry is used and post an interrupt if the parity is found incorrect. Furthermore, any memory read returns if the parity bit is correct or incorrect (reading into RAM block does not cause an interrupt to be posted even if the parity is found incorrect — only traffic processing can cause an interrupt to be posted). Note that writing a parity bit to 1b by the software causes the switch to automatically compute the inverse of the right value, and thus causes an immediate parity error as soon as the entry is used (this is only useful for test purposes). Note that not all tables are software accessible and that parity errors are not always fatal - in fact there are four categories of parity errors:

- **Transient** — Happened on one packet but has no consequence.
- **Cumulative** — Causes some of the memory to be lost but has no effect otherwise. Switch needs if those accumulates.
- **Fatal** — Switch must be reset.
- **Software Repairable** — Software can find which entry is wrong and repair it.

All parity errors clearly describe which category they belong to.



5.5 I²C Controller

The I²C controller supports the following features:

- 100 KHz, 400 KHz operation.
  - The speed is programmable through the I2C_CFG register and the switch supports a larger set.
- Slave mode allowing external masters to read/write registers into the chip.
  - All registers are accessible except for the LCI_TX_FIFO and LCI_RX_FIFO registers. Those have special handling and are not accessible from the I²C, it is thus not possible to send or receive frames via I²C.
- Master mode allowing the switch to access external I²C devices.
- Configurable I²C slave address.
- Boot configuration from I²C serial EEPROM.
- Bus arbitration.
  - The I²C controller attempts to gain the bus only once and return an error if it did not succeed. The software has to retry later on.
- 7-bit addressing mode.

The I2C controller does not support the following features found in some I²C devices:

- General call address.
- 10-bit addressing mode.
- Start byte.
- Mix-speed mode.
- High-speed mode.

Figure 5-5 shows the basic read and write accesses.

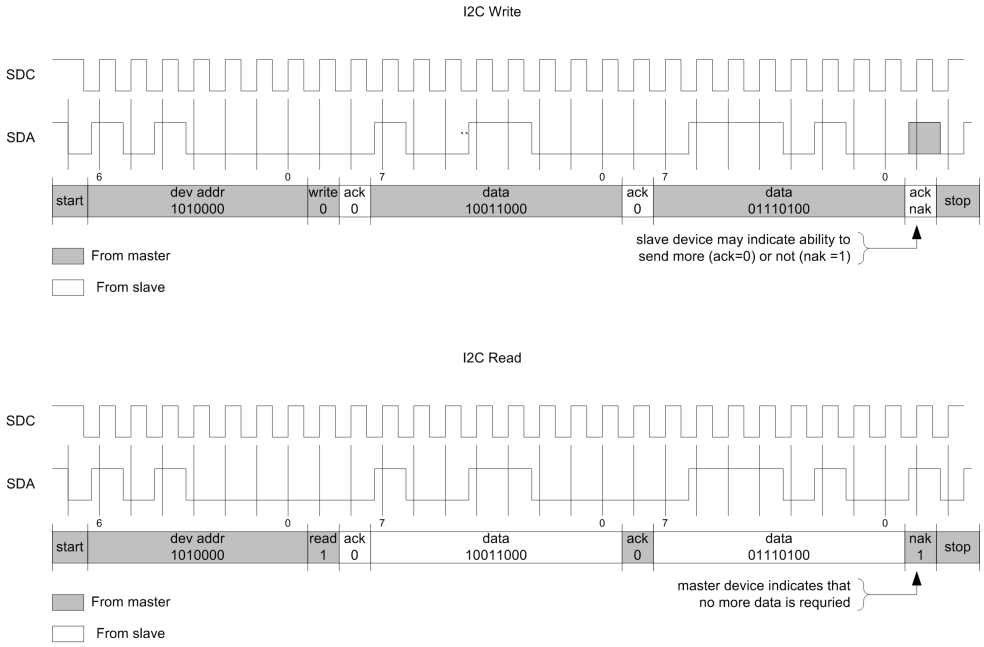


Figure 5-5 I²C Basic Accesses

The pins SDA and SDC are open drain pins with external pull-ups. This structure has the disadvantage of creating slow rising edges which can potentially be incorrectly interpreted as a double transition. To prevent this, the FM4000 device incorporates a digital filtering circuit to ensure that only complete transitions of either SDC or SDA are detected. The filter operates by sampling the SDA and SDC signals with a filter duration based on a counter that uses the EBI reference clock. The filter delay is determined as follows:

```
filter delay = I2C_CFG.DebounceFilterCountLimit * EBI_CLK period / 2
```

For a 50 MHz EBI reference clock, this gives:

```
filter delay = I2C_CFG.DebounceFilterCountLimit * 40 nS
```

Since the EBI reference clock may be different for each design, one value does not work for everyone. However, the default value is on the extreme side for more common reference clock periods. As an example, if the EBI reference clock is 50 MHz and you need to operate the I²C interface at 400 MHz, and your switch is booted from EEPROM, it is recommended that you set the I2C_DebounceFilterCountLimit to 3 in the EEPROM image.

The I²C controller supports a timeout of 100 µs (1/10 of the I²C clock rate) that aborts the current cycle and returns all pins to 1b.

In the slave mode, an external agent on the I²C bus can access any register of the chip as the CPU does.

The slave mode has the following characteristics:

- Slave address is user configurable and defaults to '1000xxx' (0x40-0x47) where ‘xxx’ is defined by the I2C_ADDR configurations strapping pins (derived from pull-up or pull-down on the DMA pins).

- All write accesses start with a 3-byte address field to indicate which address (register or table) to read followed by N 4-byte data words. Address and data must be sent MSB first. Data words are written into consecutive addresses starting with the address given. The address is incremented at the end of the data transfer. The master is at liberty to write any number of words, and it is assumed that the master never attempts to write into an illegal address. It is possible for a write access to only include the 3-0 byte address without any data words. This is used to load an address into the chip for an eventual read (note that the address is saved only if 3 address bytes are sent).
- All read accesses return consecutive 4-byte words starting with the last address used during a write cycle. The actual register is latched just before the first byte of the word is sent. The master is at liberty to read any number of words and terminates with a NAK on the last byte desired, which could any byte.
- The controller supports restart cycles (a stop-start).

This is illustrated in Figure 5-6:

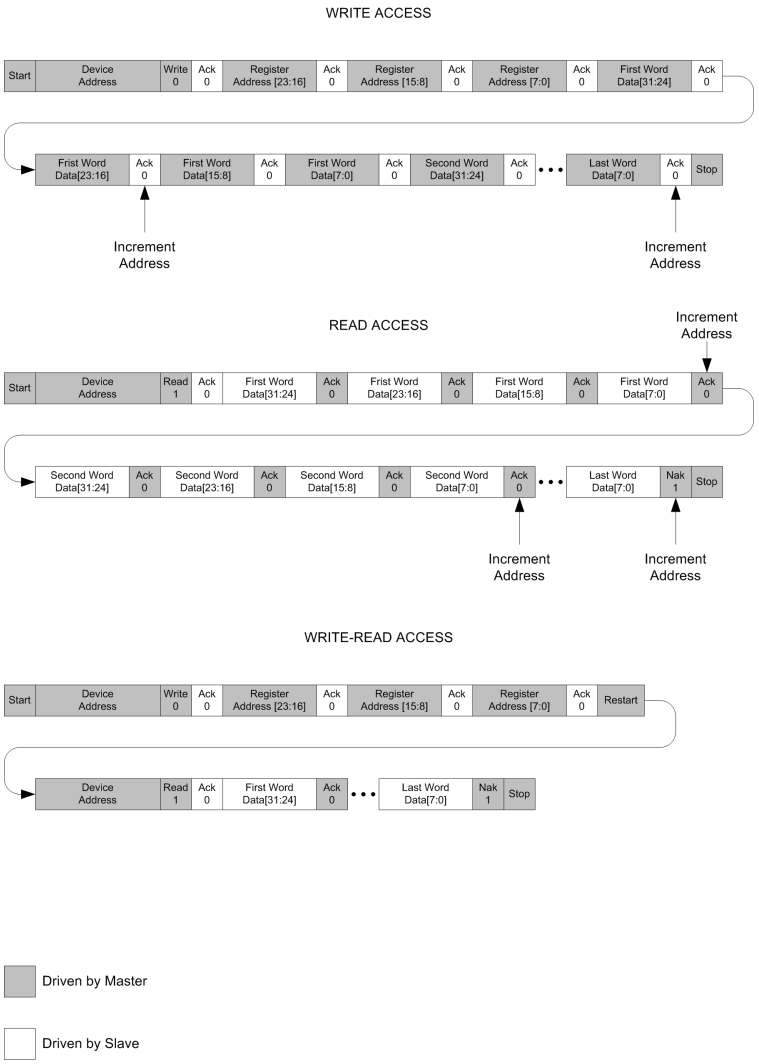


Figure 5-6 Access to FM4000 Internal Registers via I^2C

In the master mode, the I^{2}C controller is capable of issuing automatic I^{2}C accesses:

- **'write'** — Can write up to 8 bytes.
- **'write-read'** — Can write up to 4 bytes and receive up to 4 bytes.
- **'read'** — Can read up to 8 bytes.

The I^{2}C registers are:

- **I2C_CFG**
  - **Enable** (1 bit) — Defines if the switch answers to an I^{2}C address from an external master or not.
  - **Address** (7 bits) — Defines the address to which the switch answers.
  - **Divider** (12 bits) — Defines clock rate as a divider of the LSM base clock (which is CPU_CLK divided by 2).
  - **Filter** (5 bits) — Defines the number of clock for data to be stable before being recognized. This register is used to filter glitches on I^{2}C with bad slew rate.
- **I2C_DATA_W** (32 bits) — Contains the first word for 'write' or 'write-read' commands.
- **I2C_DATA_RW** (32 bits) — Contains the second word for the 'write' or the read for the 'write-read' or 'read' commands.
- **I2C_CTRL** — Used when I^{2}C controller is master.
  - **Address** (8 bits) — Defines the address of the device to access (lower bit ignored)
  - **Command** (4 bits) — Execute command 'write', 'write-read', 'read', 'null'.
  - **LengthW** (3 bits) — Defines the length of the first word. Defines the length in bytes of the first word sent (only 1,2,3,4 are valid values). Only used for 'write' and 'write-read' commands.
  - **LengthRW** (3 bits) — Defines a length in bytes of the second word sent (0,1,2,3,4 where 0 means not sent) for the 'write' command and for the length in bytes of the word to receive (only 1,2,3,4 are valid values) for the 'write-read' and 'read' commands.
  - **LengthSent** (2 bits) — Defines the length sent in case the attached device NAK prematurely. This register is meaningless if the device returned its NAK at the expected location.
  - **Status** (4 bits) — Defines the status of the command (completed, aborted, etc...)

The possible transactions are shown in Figure 5-7:

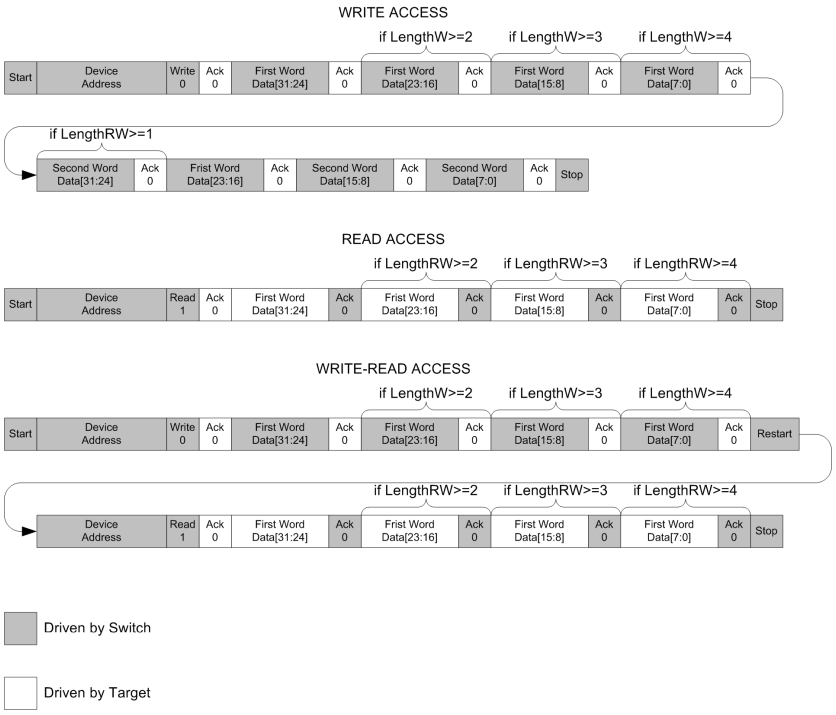


Figure 5-7 Complex I²C Accesses from FM4000

5.6 MDIO Controller

The MDIO controller is designed to allow the CPU to access MDIO devices through the switch. The features supported are:

- Support clauses 22 and 45.
- Master only. The switch cannot be the target for any access.
- 3.3 V only. An external level converter is required for access to 2.5 V or 1.2 V MDIO as defined in IEEE802.3ae specification.

The clause 22 format is shown in Figure 5-8.

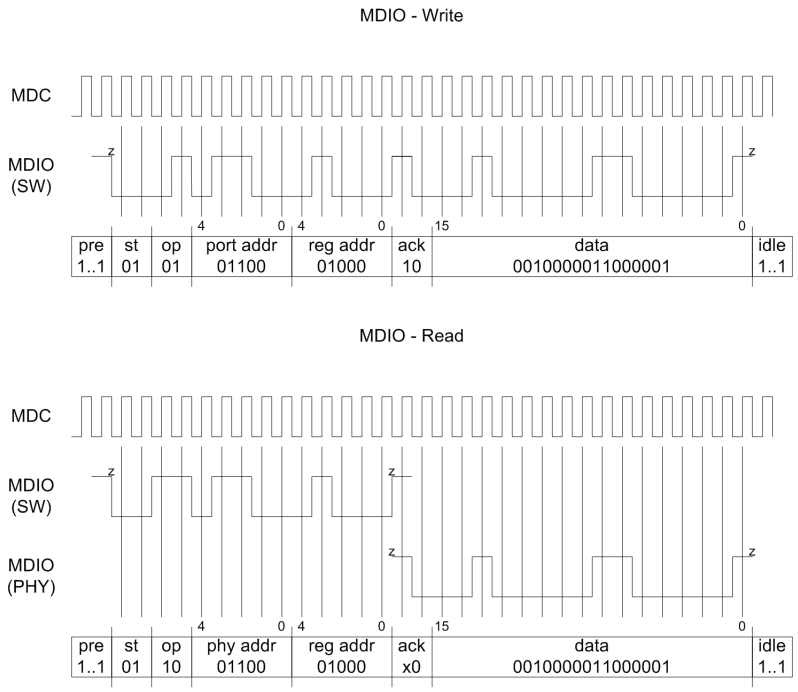


Figure 5-8 Clause 22 MDIO Transaction Format

The clause 45 frame format is shown in Figure 5-9.

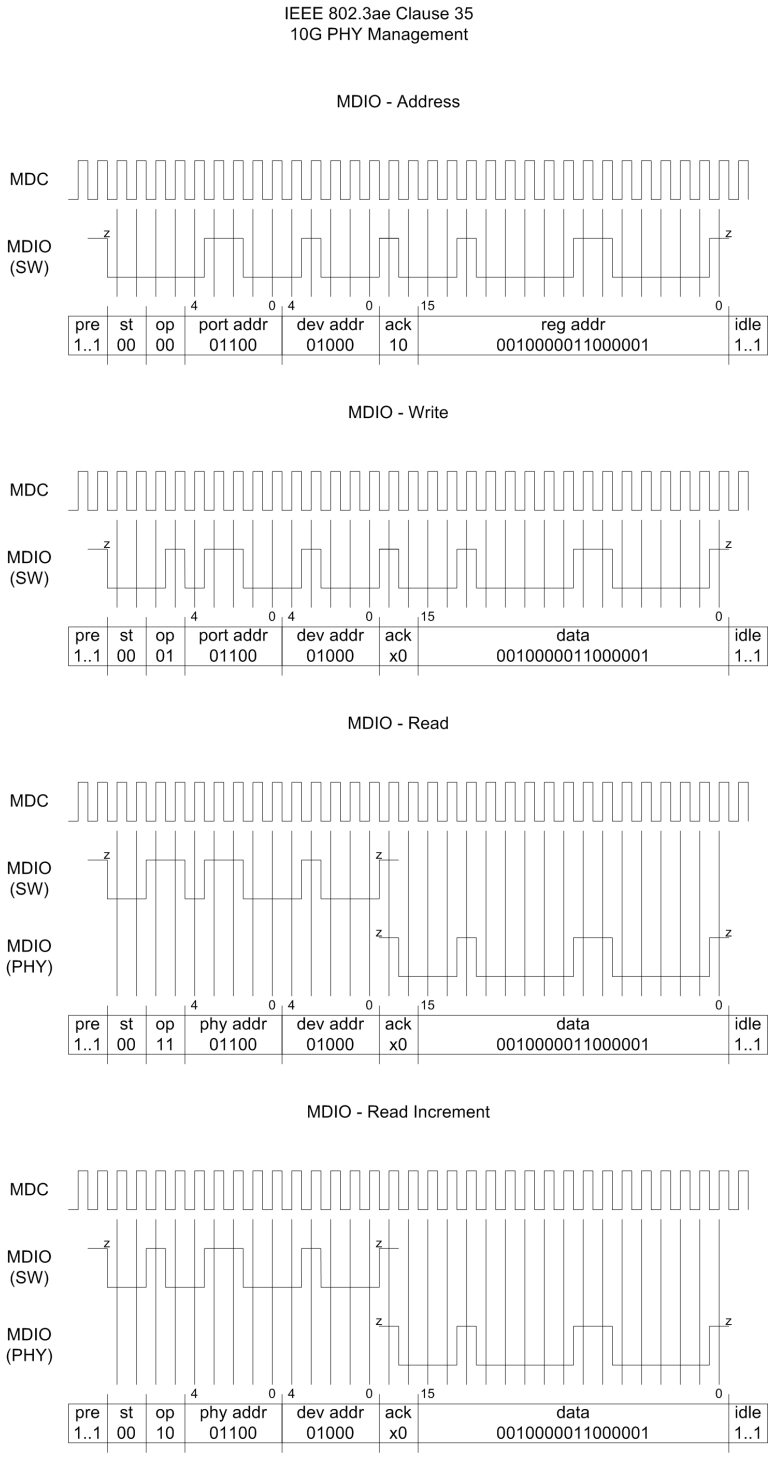


Figure 5-9 Clause 45 MDIO Transaction Format

The register settings that support these transactions are the following:

- **MDIO_CFG**
  - **Divider** (12 bits) — Defines the clock divider (from LSM clock)
  - **Preamble** — Defines if the 32-bit pre-amble is always sent or not.
- **MDIO_DATA** — The data sent or read. Only 16 bits.
- **MDIO_CTRL**
  - **PHY Address** (5 bits)
  - **Device Address** (5 bits) — Note that in 1 GbE mode this field becomes the "register" field.
  - **Register Address** (16 bits) — Note that in 1 GbE mode this field is unused.
  - **Command** (2 bits):
    - **Null**: Do nothing
    - **Write**: Send register address frame and then write frame.
    - **Sequential-read**: Send read command frame only.
    - **Random-read**: Send register address frame followed by a read frame.
  - **Device Type** (1 bit) — Defines if the frame format is compatible with clause 22 (0b) or clause 45 (1b). For the clause 22, the commands "sequential-read" and "random-read" are exactly equivalent.
  - **Status** — Returns the status of the command.

5.7 GPIO Controller

The GPIO controller is 16 bits wide and supports:

- Input, output, open-drain.
- Interrupts per bit.
  - Configurable for low to high or high to low or both.

The following registers are defined for the controller:

- **GPIO_DATA**
- **GPIO_CTRL**
- **GPIO_IP**
- **GPIO_IM**

Note: The GPIO pins are defaulted as inputs (except for SPI_MOSI, SPI_CS_N and SPI_CLK) after reset and that some GPIO pins are used to provide hardware configuration to the chip.The hardware configuration pins (strapping options) are latched when the chip is taken out of reset (CHIP_RESET_N goes from low to high).

Note: SPI_MOSI, SPI_CS_N and SPI_CLK only default to outputs when the latched value of EEPROM_EN is high and the latched value of EEPROM_EN2 is low.

Table 5-2 GPIO and Strapping

GPIO Pin	Hardware Configuration	Default
0	DTACK_INV	Input
1	RW_INV	Input
2	IGNORE_PARITY	Input
3	SPI_CLK	Output
4	SPI_CS_N	Output
5	SPI_MOSI	Output
6	SPI_MISO	Input
7	EEPROM_EN	Input
8	EEPROM_EN2	Input
9	AUTOBOOT	Input
10	PARITY_EVEN	Input
11	I2C_ADDR[0]	Input
12	I2C_ADDR[1]	Input
13	I2C_ADDR[2]	Input
14	DATA_HOLD	Input
15	—	Input

5.8 Frame Handler PLL

The Frame Handler PLL is used to generate a high speed clock for the Frame Handler and the MSB modules. The clock distribution around the PLL is shown in Figure 5-10.

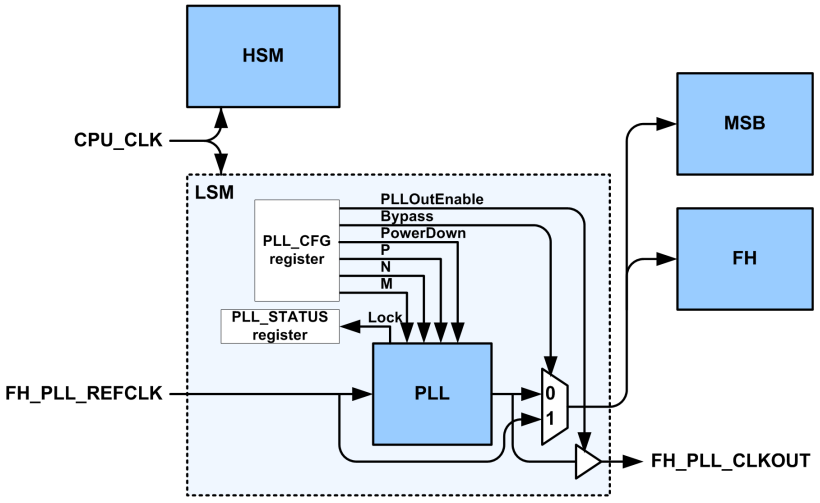


Figure 5-10 Frame Handler PLL and Surrounding Circuitry

The PLL is controlled by the PLL_CFG register located in the LSM module. The PLL reference input clock is a dedicated pin on the package (FH_PLL_REFCLK) and the output of the PLL is normally routed to the FH and the MSB modules.

For debugging purposes, the output of the PLL can optionally be observed on a dedicated pin using the PLL_CFG::PLLOutEnable register field. Also, the MSB and FH can optionally use the FH_PLL_REFCLK rather than the PLL output using the PLL_CFG_::Bypass register field.

The PLL block itself is illustrated in the Figure 5-11, where the roles of the three divisors is evident.

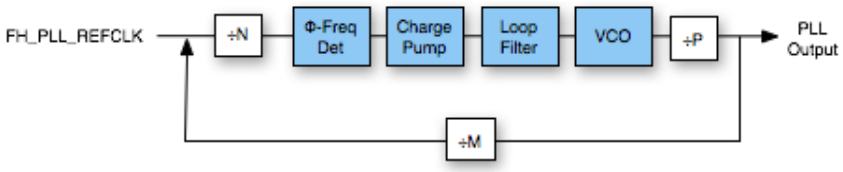


Figure 5-11 Frame Handler PLL Block Diagram

The output of the PLL uses the following formula:

FH_PLL_CLKOUT = FH_PLL_REFCLK * M / N / P

Where:

- N must be between 1 and 16
- M must be between 4 and 128
- FH_PLL_REFCLK must be between 10 MHz and 70 MHz
- FH_PLL_REFCLK has a duty cycle of 40/60 or better
- FH_PLL_REFCLK jitter must be better than 20 ps RMS.
- Vaa and Vdd ripple should be kept between -100 and +100 mV (10 MHz - 2.13 GHz).
- PLL Output, multiplied by P, must be between 70 MHz and 650 MHz

The frequency requirement for the PLL output is 375 MHz to 380 MHz. For example, with a FH_PLL_REFCLK input of 33 MHz, setting M = 23, N = 2 and P = 1 gives a PLL output of 379.5 MHz, which is in the desired range.

The register PLL_STATUS indicates if the PLL locks on the incoming clock or not.

5.9 Frame Timeout

The management block sends a pulse periodically to the scheduler which uses it to flush any old frames from its transmit queue. The scheduler uses the timeout clock to maintain a local clock tick counter, and marks any new frame received with this current clock tick counter. The scheduler the constantly checks the age of each frame as it is de-queued for transmission by computing the difference between the frame tick and the current tick. If the difference is greater than 2, the frame is still removed from the queue but not transmitted and freed (note that multicast frames are freed only once all instances have been de-queued).

The clock period is set via the 28-bit FRAME_TIME_OUT register and is equal to:

```
timeout_period = CFG_TIMEOUT * 2048 / CPU_CLOCK.
```

Frame arrival is not synchronized to the timeout pulse, hence the actual frame timeout extremes could be:

```
timeout_period <= frame_timeout <= timeout_period * 2 + maximum_de-queue_time
```

As an example, if CPU clock is set to 66 MHZ, the timeout period could be:

- If FRAME_TIME_OUT is set to 0x0FFFFFFF, the timeout period is 8310 seconds.
- If FRAME_TIME_OUT is set to 32226, the timeout period is 1 second.
- If FRAME_TIME_OUT is set to 1b, the timeout period is 31 µs.

The scheduler can dequeue at a rate of about 400 MPPS (2.5 ns per frame).

As a special case, writing 1b causes an immediate timeout pulse to be sent to the scheduler, so writing successive '1's' into FRAME_TIME_OUT can be used to artificially accelerate frame time outs.

5.10 LED

The LED interface consists of 4 signals, CLK, DATA0, DATA1, DATA2, and ENABLE, which transmit 3 bits of status data for the LED per port over the time multiplexed data pins. The 3 bits of status for ports 0-8 are placed onto Data0 and the 3 bits of status for ports 9-16 are placed onto Data1 and the 3 bits of status for ports 17-24 are placed onto Data2.

The LED interface is controlled via the LED_CFG register which defines:

- **LED_FREQ** (16 bits) — A divider to derive the LED_CLK from the CPU_CLK. The exact frequency is equal to CPU_CLK/256/LED_FREQ.
- **LED_ENABLE** (1 bit) — Controls if the LED's are enabled (1b) or disabled (0b).
- **LED_INVERT** (1 bit) — Controls the polarity of LED_DATA. The values are (0b) (as shown in the Figure 5-12) or (1b) (invert of what is shown in Figure 5-12).
- **LED_MODE** (1 bit) — Encoding of LED_DATA[2:0].

The overall timing diagram is shown in Figure 5-12.

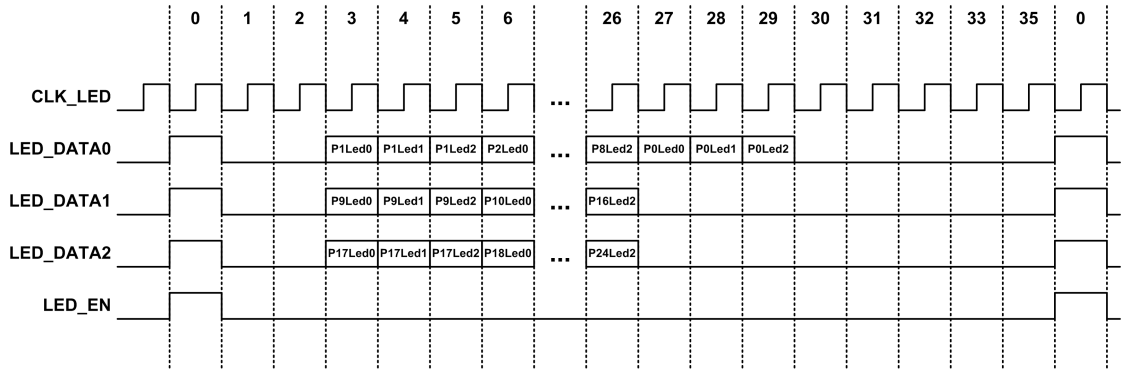


Figure 5-12 LED Timing Diagram

The encoding is as follows:

Table 5-3 LED Definitions

Mode	Usage
0 (Compatible with FM2224	LED0: STATUS Off: Port has no link synch or remote fault error. On: Port has a link synch error or no signal Blinking: Port has a remote fault LED1: RECEIVE Off: Port is not enabled On: Port has link and is enabled Blinking: Port is receiving data LED2: TRANSMIT Off: Port is not transmitting data Blinking: Port is transmitting data
1	LED0: STATUS Off: Port is not operating normally, LED2/LED1 are interpreted as: – LED1 and LED2 are off => port is in reset – LED1 is on and LED2 is off => port is down (no sync) – LED1 is off and LED2 is on => port has detected remote fault This state is momentary, oscillating between this state and any of the link up states. On: Port is operating normally (link up and no fault detected) – LED1: blink when receiving data – LED2: blink when transmitting data

5.11 CPU Interface Controller

The CPU Interface Controller is compatible with the FM2224. It supports the following signals:

- Bus interface:
  - ADDR
  - DATA
  - AS_N
  - CS_N
  - RW_N
  - DTACK_N
  - DERR_N
  - PAR
  - CPU_CLK
- DMA control:
  - RXRDY_N
  - RXEOT_N
  - TXRDY_N

- Other pins:
  - INTR_N

The CPU is configured through configuration pins available in the LSM. The strapping options for CPU are:

- **DTACK_INV** — Defines the polarity of DTACK.
- **RW_INV** (configuration pin for polarity of RW) — Defines the polarity of RW_INV.
- **IGNORE_PARITY** — Defines if parity is used or not.
- **PARITY_EVEN** — Defines the parity type (new for FM4000).

5.12 CPU Bus Interface

The CPU bus interface timing diagram is shown in Figure 5-13 (assumes DTACK_INV and RW_INV are both set to 0b).

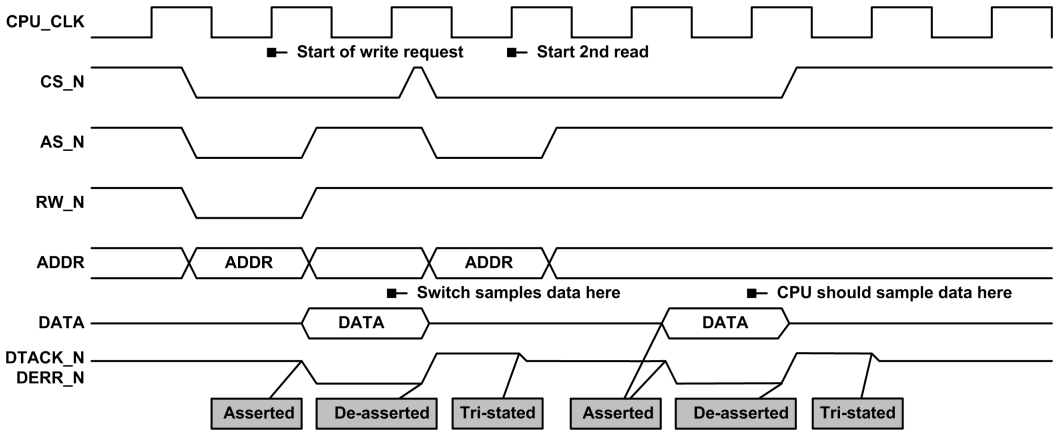


Figure 5-13 CPU Bus Timing Diagram

The CPU bus interface is synchronous and the switch samples/drives at rising edge of the CPU clock and follows a simple protocol:

- A cycle starts when CS_N and AS_N are both asserted (low). The ADDR and RW_N signals are also sampled on the same cycle to determine the register addressed and if the cycle is a read or a write.
  - The interpretation of RW_N depends on the strapping of the pin RW_INV. If the RW_INV is strapped to GND, RW_N=1b is READ and RW_N=0b is WRITE. If the RW_INV is strapped to VDD33, RW_N=1b is WRITE and RW_N=0b is READ.
- For a write cycle, the data is always sampled on the next cycle. Furthermore, the switch has a small write FIFO and asserts a DTACK_N signal on the next cycle as well if this write FIFO is not full. Otherwise, the DTACK_N is delayed until this FIFO has room to store the data. The minimum cycle time is thus 2 clock cycles.

  — The DERR_N is actively driven at the same time as DTACK_N is asserted. It is asserted (logic low) if the data parity is incorrect and de-asserted (logic high) if the data parity is correct. Note that the switch internally cancels the write operation if the parity is incorrect.

- For a read cycle, the switch delays asserting DATA and DTACK_N until data is available, and drives both signals on the same clock cycle. The minimum cycle time is 3 clock cycles.

  — The parity on the data bus is presented at the same time as the data.

- After the read or the write cycle is completed, the switch actively de-asserts DTACK_N for one cycle, and then tri-states the DTACK_N signal. Note that the switch can detect a new read or write cycle on the same cycle on which DTACK_N is actively de-asserted or on any follow-on cycles.
- The bus cycle latency from cycle start to DTACK_N being asserted is variable depending on the register accessed and the clock rate in the chip. The worst case is estimated to be 1us for most typical systems (EPL running at 125 MHz or above, Frame Handler running at 360 MHz or above) and only occurs when accessing Frame Handler registers while the system is fully busy or accessing EPL registers, repetitively.

Differences with FM2000 family:

- The FM2XXX samples write data on the same cycle as CS, AS, ADDR and RW_N. Because of that, CS and AS had to be delayed on multiplex buses by one cycle to allow ADDR to be latched and DATA to be driven on the bus. This is addressed in FM4000 by sampling data on the next clock edge allowing CS and AS to be driven along with the address cycle.
- The FM2XXX waits for CS to fully de-assert for one cycle before starting a new cycle. This restriction has been removed.
- FM4000 is designed to be clocked at up to 133 MHZ versus 66 MHZ for the FM2000 family
- The combination of those 3 modifications could allow the system designer to quadruple the throughput on the CPU bus.

5.12.1 Using DATA_HOLD

The CPU bus interface also support a data holding mode (asserted with DATA_HOLD strapping option) where the DTACK_N (for read and write cycles) and DATA (for read cycles) are maintained asserted until chip select is de-asserted. The DTACK_N and DATA are then immediately tri-stated coincidently with the de-assertion of chip select. This mode is intended for situations where a fixed length cycle is needed, the length of the cycle must be greater than the worst delay expected on DTACK_N (about 16 cycles at 33 MHz).

Figure 5-14 illustrates the effect of the DATA_HOLD option.

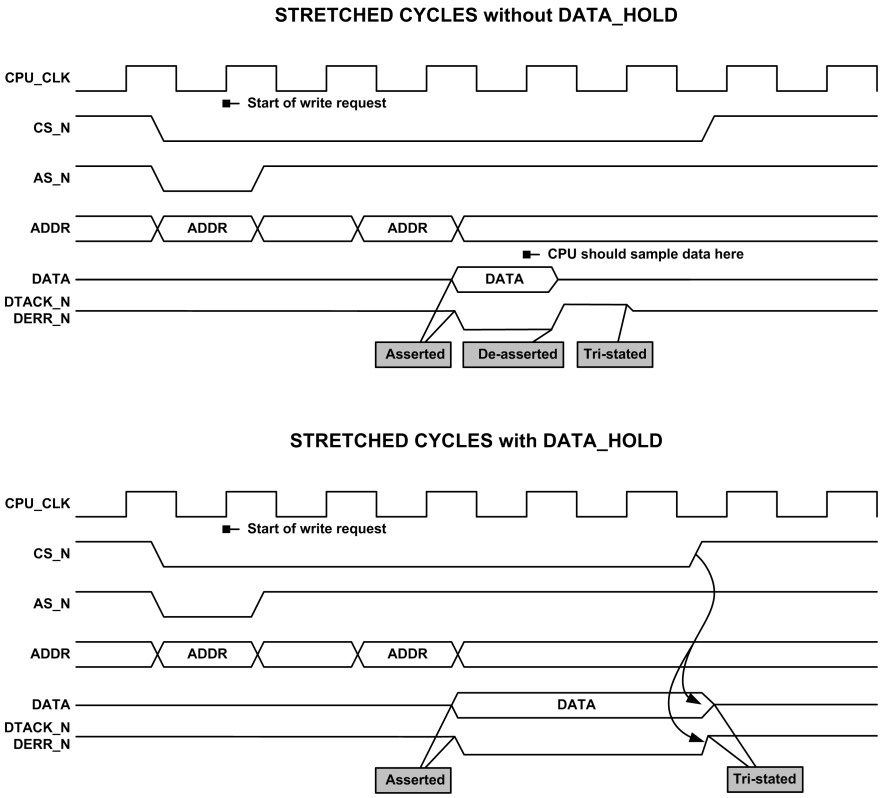


Figure 5-14 Effect on DATA_HOLD on CPU Cycles

5.12.2 Atomic Accesses

The frame handler includes tables that are wider than 32 bits and that need to be accessed atomically. (i.e., the data must be written as one single large word and not as multiple smaller 32-bit words ensuring that the table does not contain an intermediate incorrect value at any point in time or that the software does not read a false value.) The tables that support atomic access are:

- RX_VPRI_MAP
- FFU_SLICE_TCAM
- FFU_SLICE_RAM
- FFU_MAP_MAC
- FFU_MAP_IP_HI
- FFU_MAP_IP_LO
- FFU_MAP_L4_SRC
- FFU_MAP_L4_DST
- FFU_MASTER_VALID

- MA_TABLE
- INGRESS_VID_TABLE
- INGRESS_FID_TABLE
- EGRESS_VID_TABLE
- GLORT_RAM
- GLORT_DEST_TABLE
- POLICER_TABLE
- ARP_TABLE

For those tables, the switch includes extra circuitry to accumulate the data words into temporary registers before performing the actual read or write. The exact process is the following:

For write:

- The software should write the entry starting with the least significant word and terminating with the most significant word.
- The hardware stores the least significant words into temporary cache and then issues a write into the table when the most significant word is written using the content of the temporary registers to complete the entry.
- The software can accelerate loading an entire table with the same value (0 by example) by writing the least significant words only once and load successive entries by writing only the most significant word.

For read:

- The software may read an entry in any order.
- The hardware reads a table entry whenever the index or the table is different from the last index or table read or written, and saves all words read into a temporary cache and then returns the particular word addressed by the software. If the index and table are the same as the last index and table, the content of the cache is used to return the word addressed.
- The software does not have to read all words if they are not needed.

There is only one set of temporary registers for this purpose. Accesses to an atomic table may be interleaved with accesses to non-atomic tables or single registers that are outside of the frame handler without causing problems to either type of access as long as it is understood that the non-atomic accesses may occur out of sequence with the atomic accesses.

5.12.3 Little and Big Endian Support

All registers, regardless of their width, are 32-bit aligned on the memory map. As an example, a 64-bit register is not necessarily aligned on a 64-bit boundary. And all registers greater than 32 bits are accessed least significant word first. As an example, a 128-bit register would be accessed in the following manner:

```
Address X+0: DATA[31:0]   (least significant word)
Address X+1: DATA[63:32]
Address X+2: DATA[95:64]
Address X+3: DATA[127:96] (most significant word)
```

Any large entity in a single large register (such as a 48-bit MAC address) is stored as multiple 32-bits where each 32-bit entity contains up to four bytes and where the least significant byte is assumed to be mapped using the least significant 8 bits of the 32-bit word. For example, the default IEEE assigned LACP frame is stored as follows:

```
MAC Address = 0x0180C2000002 (LACP FRAME)

             31  24 23  16 15   8 7    0
            +------+------+------+------+
Address X+0 | 0xC2 | 0x00 | 0x00 | 0x02 |
            +------+------+------+------+
Address X+1 |      |      | 0x01 | 0x80 |
            +------+------+------+------+
```

The address is transmitted most significant byte first, i.e. starting by 0x01 in this case and terminating with 0x02.

This is the natural encoding for a little-endian processor such as the Intel x86 processor family and any large register may simply be accessed directly. In the case of a big-endian processor such as the PowerPC, the content of any memory must be accessed 32-bits at a time and reassembled into a 64-bit word manually. This is shown in the following example:

Accessing a 64-bit register with a little-endian processor:

```
long long macAddress = 0x0180C2000002LL;
long long *register_ptr;

*register_ptr = macAddress;
```

Accessing a 32-bit register with a bit-endian processor:

```
long long macAddress = 0x0180C2000002LL;;
long long *register_ptr;

*(((unsigned int *) register_ptr)+0) = macAddress;
*(((unsigned int *) register_ptr)+1) = macAddress >> 32;
```

There are only a few registers where this type of access is required and the effect on the CPU is negligible normally (the Intel API takes care of this).

However, the transfer of packets is not negligible and poses a challenge because the byte ordering within memory is be the same between little- and big-endian processors. To avoid the processor having to swap bytes, the switch offers an endianism option for byte ordering in the LCI_CFG register.

This bit only affects the interpretation of byte positions within the packet payload words sent to or received from the CPU. In the big endian configuration, successive bytes of a packet must be stored by placing the first byte in the most significant byte location of memory and moving right. In the little endian configuration, successive bytes of a packet must be stored in the opposite sense, from least significant byte to most. In the case of 32-bit quantities transmitted over a 32-bit bus, the CPU endianness does not matter since all bit fields are defined explicitly. Thus the TX command and RX status words (and in fact all other registers in FM4000) are defined the same for both little- and big-endian CPUs, independently of the LCI_CFGEndianness setting. This is illustrated in the next sections.

5.13 CPU Frame Transfer

FM4000 supports packet transfers between the CPU bus interface and any of the switch's 24 external XAUI ports. Since the FM4000 bus interface only supports slave mode operation, it cannot store or retrieve packet data directly to or from the CPU host memory. Instead, an external bus interface master must individually write or read each word of a packet being sent or received. This external bus master need not be the CPU itself. FM4000 defines three additional bus signals that provide compatibility with certain standalone dual-channel DMA controllers, such as the PLX 9056. Such an arrangement is illustrated in Figure 5-15.

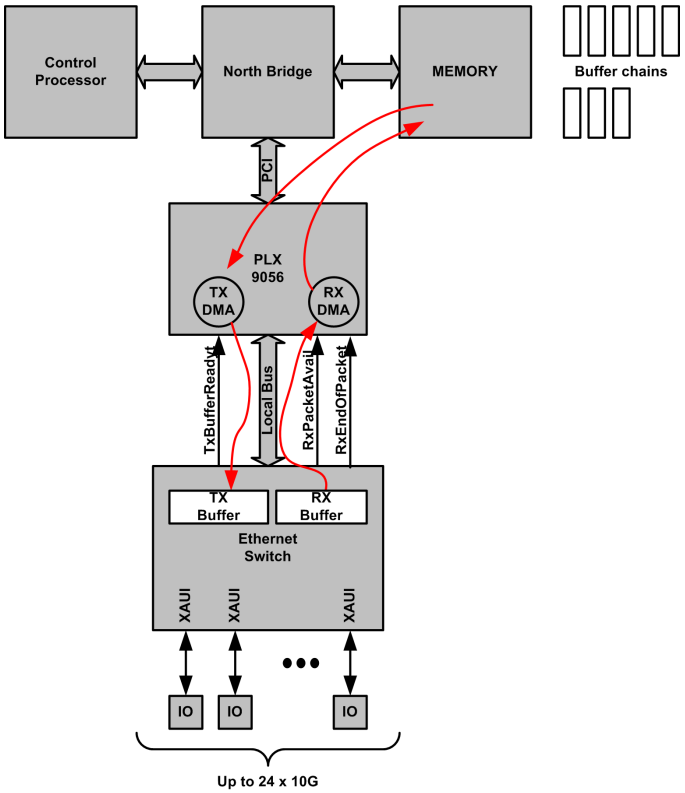
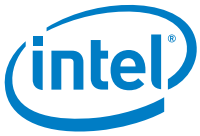


Figure 5-15 DMA Transfer



5.13.1 Packet Transmission

The general protocol for transmitting a packet through the switch is as follows:

1. Determine that the transmit pathway is ready to receive a packet by either reading the TxReady bit of the LCI_STATUS register (non-DMA mode) or by observing the status of the TXRDY_N external pin (DMA mode). If this signal is inactive, the bus master must wait, possibly polling TxReady, until the transfer FIFO is ready. This condition never persists for any prolonged length of time (more than a few bus cycles).
2. Write the packet command word to the LCI_TX_FIFO register. The command word identifies the packet length and whether the FM4000 calculates and attaches the packet's CRC.
3. Write frame payload words to the LCI_TX_FIFO register. A final CRC word should not be specified unless the AttachCRC bit in the command word had been set to zero.

The packet provided to FM4000 must include a properly formatted F64 ISL tag. The tag's contents (Frame Type, Destination GloRT, System Priority, and VLAN) specify the forwarding behavior for the packet. Generally, packets sent by the CPU have their Frame Type field set to either “Special Delivery” or “Management”. For such packets, the standard layer 2 and 3 addressing fields have no effect on the forwarding until the packets leave the F64 ISL tag domain.

More specifically, these packets are handled in the following manner:

- No VLAN, security, or spanning tree checks are performed.
- Standard FFU routing rules do not apply.
- Destination MAC address lookup have no effect.
- Source MAC address is not learned.
- Triggers match only if the “Special Handled Frame” bit is set in their AMASK condition.
- Congestion management policies are applied.

If software wishes the packet to be handled in the standard mode, it should set the ISL tag's Frame Type field to 0x0 (“Normal”).

The format of the TX command word is listed in Table 5-4.

Table 5-4 LCI_TX_CMD Details

Bit(s)	Field Name	Description
0	AttachCRC	If 1b, FM4000 calculates and attach the final CRC word of the packet.
15:1	RSVD	Reserved. Write 0b.
29:16	Length	Length of the packet in bytes (not including the CRC word if AttachCRC==1).
31:30	RSVD	Reserved. Write 0b.

5.13.2 Packet Reception

The general protocol for transmitting a packet through the switch is as follows:

1. Determine that the switch has received a packet for the CPU, either by observing that the RXRDY_N external pin is low or by polling RxReady in the LCI_STATUS register. Note: Polling can be avoided by relying on the NewFrameRecv interrupt in LCI_IP, which is set high whenever a new packet has been received and is ready to be sent on the CPU Interface.
2. Read successive words of the packet from the LCI_RX_FIFO register. The last word received (after the packet's CRC word) is a status word indicating the byte length and error status of the packet. Three conditions indicate the end of the packet transfer (defined when the word at the head of LCI_RX_FIFO is the packet's status word):
   a. The EOT bit in the LCI_STATUS register reads 1b.
   b. The EndOfFrame interrupt in LCI_IP is set.
   c. The RXEOT_N external pin transitions low (timing illustrated below).

The packet received includes an F64 ISL tag. The fields of this tag encode information useful for software:

- Type of frame (normally forwarded vs trapped).
- Source of the packet (in the Source GloRT).
- If the frame was trapped to the CPU, the reason for trapping (in the Destination GloRT).
- Associated system priority and VLAN.

FM4000 supports 2 levels padding.

Level 1 padding is done in MSB to make the packet 32-bits alignment. This is always done by the hardware automatically regardless of HostPadding bit in LCI_CFG register.

Level 2 padding is done in HSM to make the packet 64-bits alignment if HostPadding bit in LCI_CFG is set to 1b.

The format of the RX status word is listed in Table 5-5.

Table 5-5 LCI_RX_STATUS

Bit(s)	Field Name	Description
0	Error	A value of 1b indicates the packet was corrupted in switch memory due to a parity error. The FM4000 discards any packet addressed to the CPU that is received from the network with a bad CRC.
15:1	RSVD	Reserved. Write 0b.
29:16	Length	Length of the packet in bytes.
31:30	RSVD	Reserved. Write 0b.

5.13.2.1 Little Endian Packet Transfer

Interpretation of LCI_TX_FIFO and LCI_RX_FIFO bits when LCI_CFG.Endianness is set to 0b.

Table 5-6 Packet Transmission Format for Little Endian CPU

	31:24 (MSb)	23:26	15:8	7:0 (LSb)
Command Word	Length			AttachCRC
Payload	frame[3]	frame[2]	frame[1]	frame[0]
...				
Payload	X	X	X	frame[Length-1]

Table 5-7 Packet Reception Format for Little Endian CPU

	31:24 (MSb)	23:26	15:8	7:0 (LSb)
Payload	frame[3]	frame[2]	frame[1]	frame[0]
...				
Payload	X	X	X	frame[Length-1]
Status Word	Length			Error

5.13.2.2 Big Endian Packet Transfer

Interpretation of LCI_TX_FIFO and LCI_RX_FIFO bits when LCI_CFG.Endianness is set to 1b.

Table 5-8 Packet Transmission Format for Big Endian CPU

	31:24 (MSb)	23:26	15:8	7:0 (LSb)
Command Word	Length			AttachCRC
Payload	frame[0]	frame[1]	frame[2]	frame[3]
...				
Payload	frame[Length-1]	X	X	X

Table 5-9 Packet Reception Format for Big Endian CPU

	31:24 (MSb)	23:26	15:8	7:0 (LSb)
Payload	frame[0]	frame[1]	frame[2]	frame[3]
...				
Payload	frame[Length-1]	X	X	X
Status Word	Length			Error

5.13.3 Packet Transfer DMA Timing

Packet transmission with an external DMA controller is illustrated in Figure 5-16. The external TXRDY_N signal reflects the state of the TxReady bit the LCI_STATUS register. It is asserted whenever FM4000 is ready to accept a word from the CPU. The DMA controller can write data words to LCI_TX_FIFO as long as this signal is asserted. The RW_N and RW_INV signals must be set to the correct state for proper operation (see Section 5.12).

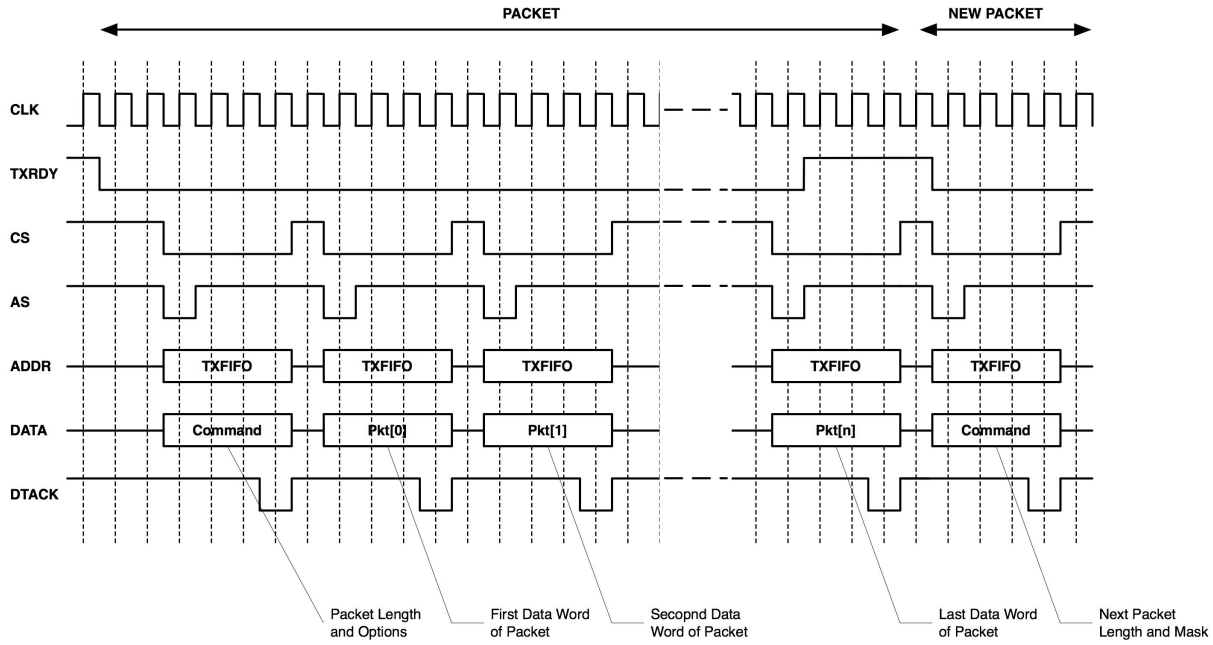


Figure 5-16 DMA Packet Transmission

The timing of packet reception with an external DMA controller is shown in Figure 5-17. The RXREQ_N signal reflects the state of the RxReady bit in the LCI_STATUS register. It is asserted whenever FM4000 has a data word ready to be read from LCI_RX_FIFO. The DMA controller can read LCI_RX_FIFO as long as this signal is asserted. When the last word of the packet transfer is read on the bus (the LCI_RX_STATUS word), the FM4000 asserts the RXEOT_N signal to indicate the end-of-transfer condition. The EOT signal notifies a DMA controller with buffer chaining support to close the current buffer and proceed to the next one in its descriptor list.

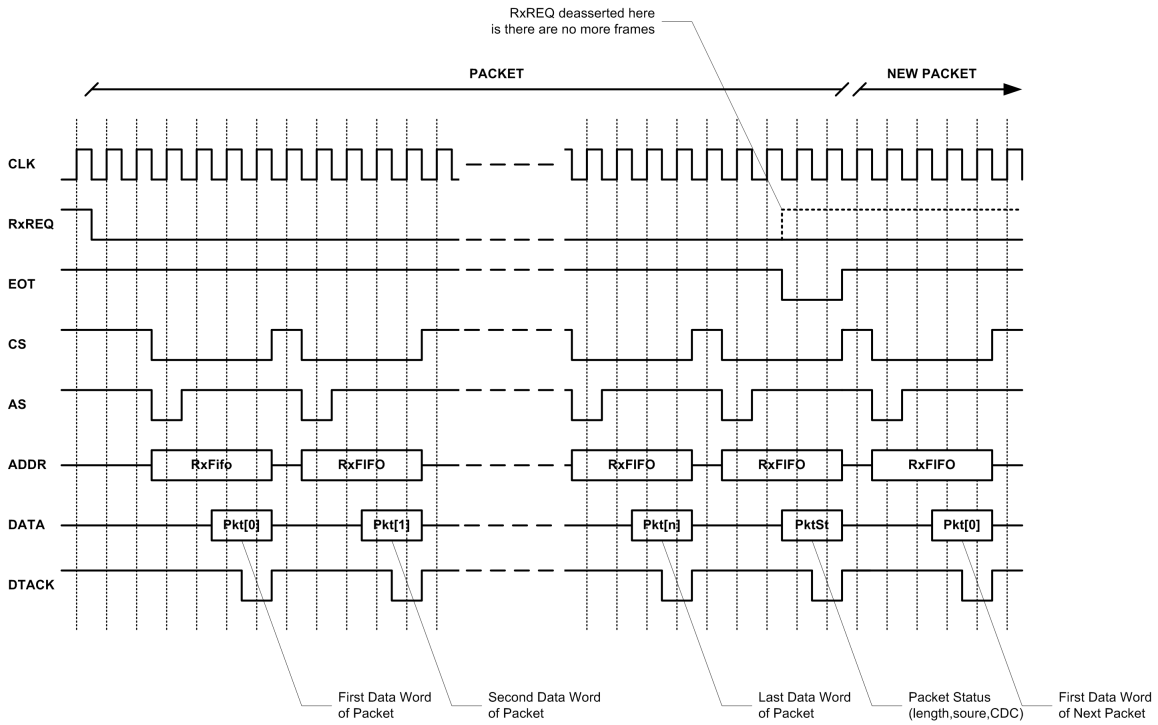


Figure 5-17 DMA Packet Reception

5.14 Packet Trapping, Logging and Mirroring

The switch has the capability to trap, log and mirror packets. The three operations are:

- A trapped packet is a packet that is taken out of the normal forwarding pipeline to be redirected to a CPU for further processing. Most traps are enabled individually in SYS_CFG_XX and PORT_CFG_XX registers but can also be activated by the FFU and the triggers. Examples are: trap IGMP, trap LACP, trap MTU, etc...
  - — Table 5-10 list all possible traps. All traps are sent to the special GloRT 0xFFnn where “nn” is the trap code listed below.
  - — Trap IP traps both unicast and multicast IP frames with options. To differentiate these, a trigger can be used or trapping can be set up using the FFU to have control over unicast vs. multicast. Note that he trap option information is in the least significant bit of the MISC[2:0] TCAM selection field, so a rule can be added to hit on this bit with scenario routable-unicast.
- A logged packet is a packet that is mirrored to the CPU for monitoring purposes.
  - — Logged bit is set in the FFU or the TRIGGERS. Logged packets are sent to the list of ports defined in LOG_MASK using the GloRT defined in the MIRROR_GLORTS::logGlort register field as a destination GloRT.
  - — Logged packets may not be exactly the same as the original frame if the EPL registers controlling VLANs and tagging are not configured the same way.

- A mirrored packet is a packet that is mirrored to another port for monitoring purposes. The switch supports only one TX mirror at any given time but could support a large set of RX mirrors.
  - RX mirrors can be activated via the FFU or the triggers.
  - In both cases, RX and TX mirrors, the mirrored packet may not be exactly the same as the original frame if the EPL registers controlling VLANs and tagging are not configured the same way.
  - For TX mirrors, if the source port and the mirrored port are on the same chip, the packet comes out untagged, however if they are on a different chip, the packet comes out as tagged irrespective of the location of the mirrored port.

A frame can be trapped, logged and mirrored all at the same time. Note that it is possible to cancel logging when a frame is trapped.

Whenever a frame is trapped or logged or to the CPU, the bottom eight bits of the CPU frame's destination GloRT encodes the reason for the trap.

Table 5-10 Frame Trap Codes

Code	Description
0x00..0x3F	Trapped due to a trigger action. The code is set to the trigger number that specified the trap action.
0x40..0x7F	Logged (mirrored to CPU) due to a trigger action. The code is set to the trigger number that specified the log action.
0x80	Trapped due to an FFU TRAP action.
0x81	Logged (mirrored to CPU) due to an FFU LOG action.
0x82	Unrecognized IEEE reserved multicast address trap due to SYS_CFG_1.trapSlow.
0x83	LACP trap due to SYS_CFG_1.trapLACP.
0x84	BPDU trap due to SYS_CFG_1.trapBPDU.
0x85	GARP trap due to SYS_CFG_1.trapGARP.
0x86	IGMP trap due to VLAN_INGRESS_TABLE.TrapIGMP.
0x87	802.1x trap due to SYS_CFG_1.trap8021x.
0x88	Security violation trap due to PORT_CFG_1.SecurityTrap.
0x89..0x8F	Reserved.
0x90	Trap code for unicast ICMP frames with TTL <= 1.
0x91	Trap code for unrecognized IPv4/IPv6 option.
0x92	Trap code for matching of CPU address (SYS_CFG_3/4).
0x93	Trap code for matching EthernetType.
0x94	Trap code for frames that exceeded MTU.
0x95	Trap code for FFU EGRESS log action.

Table 5-10 Frame Trap Codes (Continued)

Code	Description
0x96	Trap code for frames with TTL <= 1.
0x97..0xEF	Reserved.
0xF0..0xFF	Reserved for software use. Possible uses: • CPU GloRT entries in the ARP table, referenced by the FFU's route-glort action • CPU-to-CPU communication.

5.15 SPI Interface

There are three supported instructions which are always aligned to 32 bits:

- **WRITE** (8 bits) — The write command is followed by two arguments: 24 bits (last 2 bits ignored) address and 32 bits data – 64 bits in total.
- **WAIT** (8 bits) — The wait command is followed by 1 argument: 24-bit cycles to wait. Cycles are expressed in terms of the clock used by the CPU Interface – 32 bits total.
- **DONE** (8 bits) — EEPROM sequence is finished. Followed by RSVD (24 bits).

5.15.1 SPI (Serial Peripheral Interface) Controller

A Serial peripheral Interface is needed to access bootstrap code from an off chip ROM. The SPI interface has the following constraints:

- Only supports 3-byte addressing.
- Support of one Chip Select.
  - — The EEPROM size is restricted to 64 Kb-2 Mb. This is sufficient for about 30 K instructions in a 2 Mb part.
- Support of one Mode 0 device (CPOL=0,CPHA=0).
  - — Transmit data on the falling edge of the SPICLK, and receive data on the rising edge of the SPICLK signal — only one CS required)
- Supports frequency of operation up to 40 MHz.

Interoperability note: The SPI works with following parts:

- ST FLASH and EEPROM
- ATMEL FLASH
- Fairchild EEPROM
- AKM EEPROM
- MicroChip EEPROM

Table 5-11 SPI External Pin List

Signal Name	Signal Direction	Description
SPI_MOSI	OUT	Serial Data Output (MOSI, Master-Out- Slave-In, since FM4000 switch is master). Connect to EEPROM serial data input.
SPI_CS_N	OUT	SPI Chip Select (Active Low).
SPI_SCK	OUT	CLOCK for SPI interface.
SPI_MISO	IN	Serial Data Input (MISO, Master-In-Slave-Out, since FM4000 switch is master). Connect to EEPROM serial data output.

A SPI transaction is shown in Figure 5-18 and described as follows:

- Activate SPI_CS_N and assert first data bit.
- On the negative edge the clock, send the following bit stream – MSB first.
  - Send instruction – 8'h3 (I[7:0])
  - Send 3 bytes of address (A[23:0])
- On the positive edge of the clock, receive each bit of data. This continues until BOOT FSM asserts.
- De-activate SPI_CS_N, Tri-state SPI_MOSI.

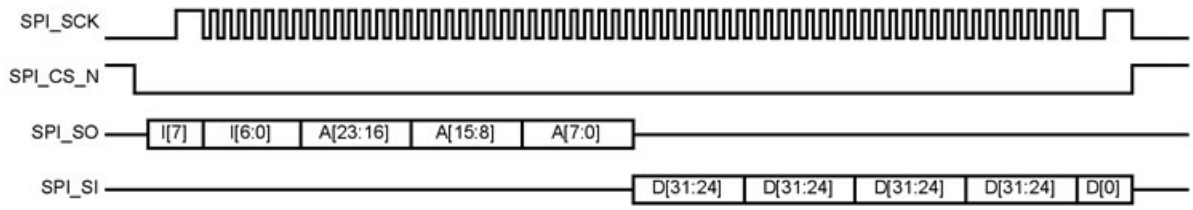


Figure 5-18 SPI Timing Diagram

5.16 In-Band Management

5.16.1 Overview

FTAG In-Band Management (FIBM) is the management of one or more FM4000 chips through management commands encapsulated within Ethernet frames. This means that all FM4000 chips do not require an attached CPU, that they can be managed by a CPU somewhere else in the network, and that one CPU can manage multiple FM4000 chips. This is illustrated in Figure 5-19 (black arrows are EBI buses and red arrows are Ethernet channels).

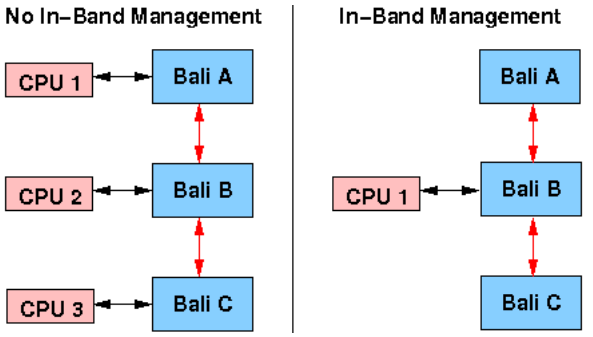


Figure 5-19 In-Band Management

There is no fixed limit to the number of FM4000 chips that one CPU can manage through FIBM.The maximum number is determined by the bandwidth of the EBI bus divided by the bandwidth required for each chip. The bandwidth required for each chip is highly system dependent.

The management operations that can be performed through FIBM are: Register Read, Register Write, Register Read/Write, Delay, and NOP. Data is obviously included with Write requests and Read responses. In a Read/Write operation, a normal Read is performed plus the data is stored in a scratch register. A Delay operation specifies the number of clock cycles before the next FIBM operation is executed. A NOP does not do anything and is only used for padding frames to 64 bytes. These operations are discussed in more detail below.

FIBM must be done in a secure and controlled environment; i.e., within an F64 ISL (Inter-Switch Link) domain. Thus, all FIBM frames must have an F64 ISL tag. Among other things, the F64 ISL tag contains bits that identify the frame as an FIBM frame, the Destination GloRT (Global Resource Tag) for the frame, and the Source GloRT for the frame. The transmission of an FIBM request frame and the receipt of FIBM response frame is shown in Figure 5-20. Management (FIBM or otherwise) is done through Port 0 of the switch.

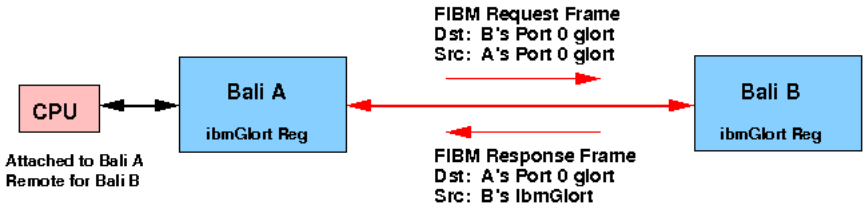


Figure 5-20 FIBM Transmission/Response

5.16.1.1 Management By an Attached CPU

This is the case for FM4000 A in Figure 5-20. While the FM4000 A is not being managed by FIBM, the CPU may be managing other FM4000 chips (like FM4000 B) through FIBM. In this case, FM4000 A receives FIBM request frames from the CPU and forwards them to FM4000 B, and it receives FIBM response frames from FM4000 B and forwards them to the CPU.

5.16.1.2 Management By a Remote CPU

This is the case for FM4000 B in Figure 5-20. The remote CPU sends FIBM request frames to FM4000 B, and FM4000 B does the following:

1. Receives the frame and checks the CRC.
2. Identifies the frame as a valid FIBM request frame.
3. Decodes the FIBM operations in the frame payload.
4. Executes the management operations.
5. Formulates a response frame consisting of data and acknowledgments.
6. Sends an FIBM response frame back to the CPU.

5.16.1.3 Basic FIBM Topology

Figure 5-19 above shows a simple case of using one CPU to manage three FM4000's. FM4000 B is being managed by an attached CPU, and FM4000's A and C are being managed by FIBM. If the CPU wants to issue the same FIBM request to both FM4000 A and C, it can use a multicast GloRT as the destination GloRT for the FIBM request frame. In the FIBM response frames, each chip uses its MSB_IBM_GLORT::ibmGlort as the source GloRT. Thus, the CPU can identify the chip from which each response came. This is the reason that the FIBM response source GloRT is ibmGlort and not the FIBM request destination GloRT.

In the previous paragraph, it was assumed that since the CPU is attached to FM4000 B, it is not managing FM4000 B through FIBM. This is not necessarily the case. The CPU can still manage FM4000 B through FIBM. If the CPU sends a frame to FM4000 B's ibmGlort, the frame follows the path (see Figure 5-23 on page 106):

CPU --> HSM --> Crossring --> MSB --> Port 0 --> Switch --> Port 0 --> MSB

The first time the MSB sees the frame (coming from the HSM/CPU), it unconditionally forwards the frame to Port 0. The switch receives the frame from Port 0, decodes the header, and sends the frame back out on Port 0. The second time that the MSB sees the frame (this time coming from Port 0), it recognizes the frame as a valid FIBM request frame. The MSB processes the frame and sends the FIBM response frame out on Port 0. The FIBM response frame makes it's way back to the CPU via the reverse path:

MSB --> Port 0 --> Switch --> Port 0 --> MSB --> Crossring --> HSM --> CPU

While FM4000 B can be managed by FIBM, it is a less efficient use of FM4000 B's switch resources and of EBI bus bandwidth. However, it does allow the CPU to manage all three chips by FIBM, and it allows the CPU to send FIBM request frames with a multicast destination GloRT to all three chips.

5.16.1.4 Other FIBM Topologies

In Figure 5-21, all three FM4000's are managed by a single CPU and by FIBM. The CPU is attached to an Ethernet port on FM4000 A via a NIC. The potential advantages of this type of configuration are:

- The CPU has a higher bandwidth connection to the system -- 10 Gb/s for Ethernet vs. ~3 Gb/s for EBI.
- The NIC may provide TCP off-loading or other additional features for the CPU.
- The CPU/NIC may be required for some purpose other than FM4000 management.

The disadvantages of this type of configuration are:

- It uses a port on FM4000 A.
- It requires a NIC.
- The NIC may not support the Intel F64 ISL tag. If not, it requires some FFU (Filtering Forwarding Unit) resources on FM4000 A.

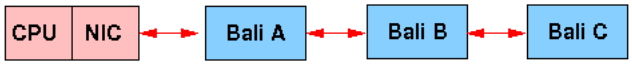


Figure 5-21 Single CPU

Figure 5-22 shows a more realistic system using FIBM. In this example, FM4000's A, B, and C are managed solely by CPU 1, and FM4000's D, E, and F are managed solely by CPU 2, and FM4000's G, H, and I are managed solely by CPU 3. Managing a single FM4000 with multiple CPU's is problematic and is beyond the scope of this document. However, if CPU 2 in Figure 5-22 wanted to do something simple like read the registers in FM4000 A, it could do so.

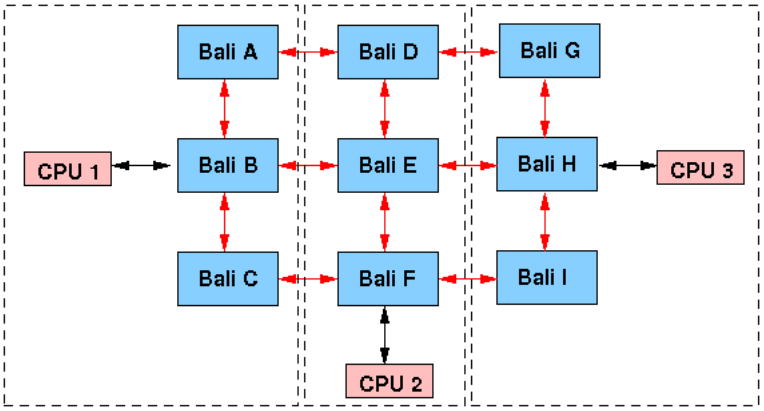


Figure 5-22 Multiple CPUs

5.16.2 Management Switch Bridge

The Management Switch Bridge (MSB) is the unit that actually executes the commands in an FIBM request frame and generates the FIBM response frame. A block diagram of a FM4000 is shown in Figure 5-23. As shown, the MSB connects on one side to Port 0 of the switch and on the other side to the management crossbar. Frames are sent and received via Port 0, and they can be either FIBM or non-FIBM frames. FIBM request frames are received from Port 0 by the MSB, the FIBM operations are executed over the management crossbar, and FIBM response frames are sent out on Port 0. Non-FIBM frames are received by the MSB from Port 0 and transmitted to the HSM (CPU) via the management crossbar. All frames from the HSM (CPU) are sent unconditionally by the MSB to Port 0.

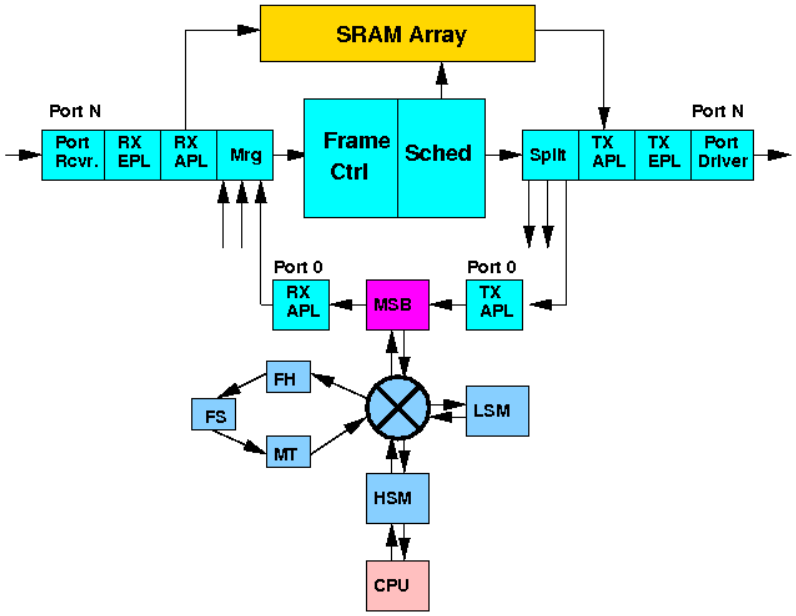
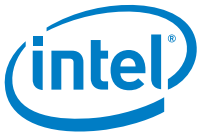


Figure 5-23 Management Switch Bridge

5.16.2.1 Valid Frames

The disposition of frames arriving at the MSB from Port 0 (from a remote CPU) is shown in the following table:

Incoming Frame (from Port 0 to MSB)				
F64 ISL Tag FTYPE	**F64 ISL Tag MTYPE**	**disableIbm Register Bit**	**attachedCpu Register Cit**	**Action**
0b11	0b00	Not set	X	A valid FIBM request frame.
0b11	0b00	Set	X	Only two registers can be accessed (MSB_CFG and INTERRUPT_DETECT).



Incoming Frame (from Port 0 to MSB)				
F64 ISL Tag FTYPE	**F64 ISL Tag MTYPE**	**disableIbm Register Bit**	**attachedCpu Register Cit**	**Action**
Other than the values above.		X	Set	A valid non-FIBM frame. Forwarded to attached CPU.
Other than the values above.		X	Not set	Not valid. Raise an interrupt, count it, discard the frame.

If MSB_CFG::disableIbm is set, FIBM is basically (but not completely) disabled. If it were completely disabled, there would be no way for a remote CPU to enable FIBM and manage the chip. If disableIbm is set:

- NOP, Read, Read/Write and Delay operations work as usual.
- Write operations only work if Address = MSB_CFG_ADDR or Address = INTERRUPT_DETECT_ADDR.
- A Write to any other address is treated as an invalid FIBM op and causes the FIBM response frame to be dropped.

If PORT_CFG_1::dropMgmtISL is set, ISL management frames, including FIBM frames, are dropped.

5.16.3 FIBM Frames

5.16.3.1 FIBM Frame Format

The format of an FIBM frame is shown in the following two tables.

FIBM Frame						
Header				Payload		
Dst MAC	Src MAC	F64 ISL Tag	Ethertype	FIBM Tag	FIBM Op's and Data	CRC
6 bytes	6 bytes	8 bytes	2 bytes	2 bytes	variable	4 bytes

	FIBM Request Frame	**FIBM Response Frame**
Dst MAC		Src MAC of the request frame.
Src MAC		Dst MAC of the request frame.
F64 ISL Tag FTYPE	Must be 0b11.	0b11
F64 ISL Tag MTYPE	Must be 0b00.	0b01
F64 ISL Tag Src GloRT	Destination GloRT for the response frame.	If MSB_CFG::ibmGlortEn is set, MSB_IBM_GLORT::ibmGlort. Otherwise, Dst GloRT of the request frame.
F64 ISL Tag Dst GloRT	Port 0 GloRT of the destination FM4000.	Src GloRT of the request frame.

	FIBM Request Frame	FIBM Response Frame
F64 SWPRI/USER		Same as the request frame.
F64 VPRI/VID	Ignored	Set to 0b.
Ethertype		MSB_INT_FRAME::etherType
FIBM Tag		Same as the request frame.
FIBM Ops & Data	See below.	See below.
CRC	Must be valid.	Normal.

5.16.3.2 ISL Tag

All FIBM frames must have an F64 ISL tag. The ISL tag contains the FTYPE bits, the MTYPE bits, the destination GloRT, the source GloRT, and "other" bits. The definition of the FTYPE and MTYPE bits is shown in the following table. The "other" bits in the ISL tag include SYSPRI, USER, etc. The "other" bits are set by the CPU issuing the FIBM request frame and are duplicated in the ISL tag of the FIBM response frame. Note that there are 4 bits for SYSPRI (System Priority) in the ISL tag, and this determines the priority of the FIBM request and response frames.

F64 ISL Tag FTYPE (Frame Type) bits			F64 ISL Tag MTYPE (Management Type) bits	
0b00	Normal frame		0b00	FIBM request frame
0b01	RX mirror		0b01	FIBM response or interrupt frame
0b10	Modified		0b10	
0b11	Management frame		0b11	

5.16.3.3 CRC Errors

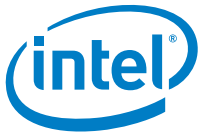
Both the switch and the MSB check the CRC on all frames arriving from a remote CPU, so the only CRC errors that the MSB detects would be the result of internal memory corruption. The MSB processes FIBM request frames header word by header word (in a cut-through style). So, part of the FIBM response frame has already been sent to Port 0 before a bad CDC on the FIBM request frame is detected. Also, any Write requests in the frame have already been executed before a bad CRC is detected; i.e., corrupt data may have been written to registers in the chip. Thus, if the MSB detects a bad CRC on an otherwise valid FIBM frame, it:

1. Raises a fatal interrupt.
2. Sets a bad CRC on the FIBM response frame, and sets a tail error on the frame. This causes the egress EPL to discard the frame.

If the MSB detects a bad CRC on an otherwise valid non-FIBM frame, it:

1. Raises a non-fatal interrupt.
2. Sets the error bit in the tail word and sends the frame on to the attached CPU.

The MSB does not check the CRC on outgoing frames from an attached CPU. However, it appends the CRC if requested.



5.16.4 Reliable FIBM Communication

Ethernet does not provide a reliable communication link; i.e., frames can be dropped during transport trough a network of switches. However, FIBM provides the means for reliable in-band management.

5.16.4.1 Sequence Numbers

Reliable communication over an Ethernet link basically requires two things: 1) acknowledgments, and 2) sequence numbers. An FIBM response frame is returned for every request frame, so the response frame is the acknowledgment for the request frame. The FIBM Tag data in a request frame is echoed in the Tag field of the corresponding response frame. So, the CPU can easily assign each FIBM request frame a 16-bit sequence number in the Tag field.

5.16.4.2 Scratch Registers

There are 16 scratch registers in the MSB (MSB_SCRATCH_[0..15]).They are RW, so they can be accessed with an FIBM Read, Write, or Read/Write operation. With a Read or Write, they are accessed by their address, just like any other register. With a Read/Write operation they are accessed by a 4-bit relative address contained in the Read/Write operation.

5.16.4.3 Reading and Writing Idempotent Registers

An operation is idempotent if it can be repeated and the effect is the same whether it is executed once or twice. An effort has been made to make all of the register operations in FM4000 idempotent. In particular, this means that FM4000 does not have any clear-on-read registers. For idempotent register operations, a remote CPU can:

- Put sequence numbers in the tag field.
- Issue multiple FIBM request frames with unique sequence numbers.
- If an FIBM response frame with a particular sequence number is not received after some period of time, the CPU can just re-send the request frame for that sequence number.

5.16.4.4 Writing Non-Idempotent Registers

If an acknowledgment is not received for a Write request to a non-idempotent register, it is necessary to be able to find out whether the Write operation was executed or not (whether the FIBM request frame got dropped or the FIBM response frame got dropped). A sequence number can be written to a scratch register by the same request frame that writes data to a non-idempotent register. If an FIBM response frame is not received, the CPU can read the scratch register and determine whether the FIBM request frame operations were carried out. The scratch register is idempotent, so the CPU can read it as many times as necessary.

5.16.4.5 Reading Non-Idempotent Registers

If an acknowledgment is not received for a Read request to a non-idempotent register, it is necessary to both 1) be able to find out if the Read operation was executed (whether the FIBM request frame got lost or the FIBM response frame got lost), and 2) be able to find out what the original data was if the Read operation has already been executed.

Sequence numbers can be written to scratch registers as described in Section 5.16.4.2. In addition, FIBM Read/Write operations should be used to read non-idempotent registers. The Read/Write operation has a 22-bit address for the non-idempotent register and a 4-bit address for the scratch register.

The non-idempotent register is read in the same manner as a normal register and, in addition, the register data is written to a scratch register. If no FIBM response frame is received by the CPU, the scratch registers can be read as many times as necessary to determine whether the initial Read/Write operation was carried out (from the sequence number); and, if it was carried out, what the original data was (stored in a scratch register by the Read/Write operation).

5.16.4.6 Interrupts

Since there is no guarantee that a single Interrupt frame is received by the CPU, interrupt frames are sent repeatedly until the interrupt is cleared (see Section 5.16.6).

5.16.4.7 Timing

Since the time for an FIBM request frame to transit from a remote CPU to the chip being managed is indeterminate (especially since there is no guarantee that it arrives at all), FIBM operations cannot be executed at a precise absolute time. If a FM4000 is being managed through FIBM, it must be managed in a way that does not require precise absolute timing. However, FIBM does provide a Delay operation so that FIBM operations can be executed with precise relative timing. A Delay operation specifies the number of MSB clock cycles before the next FIBM operation is executed.

5.16.5 The FIBM Payload Format

All payload data in the following sections is in big-endian format.

5.16.5.1 FIBM Request Frame Payload Format

An example FIBM request frame payload is shown in the following table. Each FIBM operation (Read Request, Write Request, etc.) is specified by an FIBM header word (HW), and if it is a Write Request, it is followed by the appropriate number of data words. Multiple FIBM operations are allowed in a single frame payload. The maximum length of a Read or Write is generally 16 words, but see the note below (Note on Write restrictions:) for certain Write restrictions. Length is the "real length" modulo 16 (Length=0b0000 means 16 words). If it is desired to do a Read or Write Request of more than 16 words, multiple header words must be used.

FIBM Request Frame Payload												
Tag	HW				HW				HW	HW	HW	HW
data	Write	data	data	data	Write	data	data	data	Read	Read	NOP	NOP

5.16.5.2 FIBM Response Frame Payload Format

The following table shows the FIBM response payload for the example FIBM request payload above. Each request header word (except for NOP) is repeated in the response frame. Read request header words are followed by the data. The data for a Write request is not included in the response frame.

FIBM Response Frame Payload										
Tag	HW	HW	HW				HW			
data	Write	Write	Read	data	data	data	Read	data	data	data

Note on Write restrictions:

Software must take care to insure that Writes to certain registers are restricted in burst length as follows:

- Frame Handler registers
  - Less than or equal to 4 words
- Mtable registers
  - Less than or equal to 3 words
- Certain STATS registers less than or equal to 3 words
  - **STATS_RX_TYPE**
  - **STATS_RX_BIN**
  - **STATS_RX_OCTET**
  - **STATS_RX_PKT_PRI**
  - **STATS_RX_ACTION**
  - **STATS_TX_TYPE**
  - **STATS_TX_BIN**
  - **STATS_TX_OCTET**
  - **STATS_TX_INGRESS**

5.16.5.3 Minimum FIBM Frame Payload Length

An FIBM frame header, tag, and tail is 28 bytes, thus a single FIBM Read Request would have an FIBM frame length of 32 bytes. To make the frame meet the minimum Ethernet frame length requirement of 64 bytes, NOP's can be added. A single FIBM Write Request would have a very short FIBM response frame (32 bytes). If MSB_CFG::padIbmResponse is set (the default), FIBM response frames are padded to 64 bytes.

5.16.5.4 Maximum FIBM Frame Payload Length

The maximum number of FIBM operations in a single frame payload is determined by the remote CPU. If the remote CPU submits a large number write requests, the FIBM request frame is very long. If the remote CPU submits a large number of read requests, the FIBM response frame is very long. 21 maximum length (16 word) FIBM operations can be put in a single frame payload without exceeding a 1520-byte Ethernet frame size; i.e., a user can read or write 336 32-bit registers per frame without exceeding a 1520-byte Ethernet frame size. The decision on whether or not to exceed a 1520-byte frame size is left to the user.

5.16.5.5 FIBM Header Words

Each FIBM operation is defined by a 32-bit header word as shown in the following table:

FIBM Header Words

Op	31	30	29	28	27	26	25	24	23	22	21-0
NOP	0	0	0								
Read	0	0	1		Length						Address
Write	0	1	0		Length						Address
Read/Write	0	1	1		ScrAddr						Address
Delay	1	0	0						Data		
Interrupt	1	1	1						Data		

- NOP's are the only operations in an FIBM request frame that are not included in the FIBM response frame. If an FIBM response frame is padded to 64 bytes, it is padded with NOP's.
- The maximum Read and Write Length is for 16 words (Length=0b0000 means 16 words). If a longer Read or Write request is desired, the CPU must break it up into multiple requests.
- A single Read or Write request is not allowed to cross a register address boundary.
- A Read/Write operation reads one register (specified by Address) and returns the data just as a normal Read operation would. In addition, it writes the data to a scratch register (specified by ScrAddr).
- The data field in a Delay operation specifies the number of MSB clock cycles before the next FIBM operation is executed.
- If the MSB receives any FIBM Op other than 0b000, 0b001, 0b010, 0b011 or 0b100, it is considered an Invalid FIBM Op (see below). Also, if there is an early EOF (a Write request at the end of a frame without enough data words), that is treated as an Invalid FIBM Op. However, if there is a Write

request in the middle of a frame without enough data words, there is no way for the MSB to detect this. It simply starts interpreting header words as data and vice versa until some word is found to have an Invalid FIBM Op or the end of the frame is reached.

- For an Invalid FIBM Op:
  - The entire FIBM response frame is dropped.
  - A fatal interrupt is raised (corrupt data may have been written to registers in the chip).
- Interrupt Ops are not sent from a remote CPU to a FM4000 like the other FIBM Ops – they are sent from a FM4000 to a remote CPU. When there is an interrupt, the MSB composes an FIBM interrupt frame with a single Interrupt header word (plus NOP's) and sends it to MSB_IBM_INT::interruptGlort.

5.16.6 Interrupts and RESET

5.16.6.1 General Interrupt Handling

The top level INTERRUPT_DETECT register is in the HSM. With an attached CPU it handles interrupts as normal. However, with in-band management, a remote CPU must be notified of an interrupt. In this case, the HSM asserts an interrupt signal to the MSB, and the signal stays asserted until the interrupt is cleared.

If MSB_CFG::interruptGlortEn is set, the MSB repeatedly sends FIBM Interrupt frames out Port 0 to MSB_IBM_INT::interruptGlort until the interrupt signal from the HSM is not asserted (until the interrupt is cleared). The interval between sending interrupts is determined by MSB_IBM_INT::interruptInterval. This is a 16-bit register and the time unit is 8192 MSB clock cycles (8192/375 = 21.8 usec). Counting starts at 0; i.e., for interruptInterval = N, the actual interval is (N+1)*21.8 μs. This gives a range of ~21.8 μs to ~1.43 s.

The FIBM Interrupt frame sent by the MSB is shown in the following table:

Interrupt Frame		
Dst MAC		0x00000000
Dst MAC		0x00000000
Dst MAC	FTYPE	0b11
	MTYPE	0b01
	SYSPRI	MSB_INT_FRAME::islSysPri (4 bits)
	USER	MSB_INT_FRAME::islUserBits (8 bits)
	VLANPRI	0b000
	VCFI	0b0
	VLAN ID	0x000
	Src GloRT	MSB_IBM_GLORT::ibmGlort
	Dst GloRT	MSB_IBM_INT::interruptGlort
Ethertype		MSB_INT_FRAME::etherType
FIBM Tag		MSB_INTR_CTR_1::interruptSeqCtr
FIBM Interrupt Header Word		0xE0000000

Interrupt Frame	
8 NOP Words	0x00000000
CRC	CRC

MSB_INTR_CTR_1::interruptSeqCtr is the number of FIBM Interrupt frames sent since the current interrupt signal from the HSM was first asserted.

The source GloRT in the ISL tag is MSB_IBM_GLORT::ibmGlort whether MSB_CFG::ibmGlortEn is set or not. If this register is not configured properly, the CPU does not know where the interrupt frames are coming from.

5.16.6.2 Disabling Interrupts

No FIBM interrupt frames are sent if MSB_CFG::interruptGlortEn is not set.

5.16.6.3 RESET Initiated by a Remote CPU

In addition to the normal IP registers that can only be cleared (RW1C) by the CPU, there is an IP (and IM) register in the LSM (the SW_IP register) that can be written (RW) by the CPU. Writing to this register through FIBM allows the CPU to reset the chip.

5.16.6.4 Fatal Interrupts and Self Initiated RESET

With FIBM, there is no guarantee that an interrupt message makes it to the CPU. If this is just because of an isolated instance where the frame got dropped, repeating the interrupt message solves the problem. However, part of the chip may be non-functional (that is why the interrupt got raised in the first place). In that case, even repeated attempts at sending an interrupt message to the CPU will fail.

The same problem exists in the reverse direction — any FIBM frame that the CPU sends to reset the chip might never be able to make it to the appropriate unit within the chip. So, for severe interrupts, a FM4000 that is managed by FIBM must be able to reset itself.

Thus, interrupts have been divided into two types: fatal and non-fatal; and a fatal interrupt causes a FM4000 to reset itself. All of the bits in the IP registers are internally identified as fatal or non-fatal. All of the bits in the IP registers, whether fatal or non-fatal, can be masked by the corresponding IM register.

For the MSB there are two fatal interrupts: MSB_IP::ipIbmCrcError and MSB_IP::ipInvalidIbmOp. Both are considered fatal; because, if they occur, corrupt data may have been written to registers in the chip.

If the INTERRUPT_DETECT register in the HSM has a non-masked, fatal interrupt bit set, the HSM sends a signal to the Watchdog unit in the LSM to reset the chip. It also sends the Watchdog an 8-bit fatal code that indicates the cause of the fatal interrupt. The fatal code is stored in the Watchdog's

LAST_FATAL_CODE register.The Watchdog also has a FATAL_COUNT register which contains the number of times that a fatal reset has occurred. The Watchdog is on a separate reset domain from the rest of the chip, so this data is not lost when the rest of the chip is reset.

5.16.6.5 Boot After RESET

An FM4000 without an attached CPU boots from an EEPROM. The boot sequence does the following:

- Sets MSB_CFG::disableIbm early in the boot process.
- Configures Port 0 appropriately for FIBM (interruptGlort, islSysPri, and islUserBits should be configured, the appropriate ports should be enabled, etc.).
- Writes a non-fatal, non-masked interrupt to the SW_IP register in the LSM.

When the boot sequence is complete:

- The MSB sees that disableIbm is set. This keeps the chip from executing any FIBM Write operations until the CPU has been notified that the chip reset itself (or was externally reset). If disableIbm is set, FIBM Write operations are allowed on only two registers – MSB_CFG and INTERRUPT_DETECT.
- The LSM sees that it has an interrupt, and it sets a bit in the INTERRUPT_DETECT register in the HSM.
- The HSM sees that it has an interrupt, and it asserts the interrupt signal to the MSB.
- The MSB periodically sends out interrupt messages to MSB_IBM_INT::interruptGlort until INTERRUPT_DETECT is cleared.
- The CPU:
  - Receives an interrupt frame.
  - Reads INTERRUPT_DETECT and MSB_CFG::disableIbm. From this, the CPU can determine if a chip reset has occurred.
  - If a chip reset has occurred:
    - Unset MSB_CFG::disableIbm.
- Read the LAST_FATAL_CODE and FATAL_COUNT registers in the Watchdog.
- Clear all interrupts and restore the chip to the proper operational state.
  - If a chip reset has not occurred, clear and process the interrupt as normal.

5.17 Counter Rate Monitoring

The FM4000 provides a general counter rate monitoring service to reduce the need for CPU polling of on-chip counters. Any counters that are expected, under normal circumstances, to increment at some specific bounded rate (possibly 0) are candidates for automatic monitoring. Whenever the actual observed rate exceeds the specified bound, an interrupt is raised, thereby alerting the CPU. A total of 256 counter monitors may be instantiated by software.

Relevant registers are:

- **CRM_CFG_COUNTER[0..255]**
- **CRM_CFG_WINDOW[0..255]**
- **CRM_CFG_LIMIT[0..255]**
- **CRM_LAST_COUNT**
- **CRM_CFG**

- **CRM_IP[0..7]**
- **CRM_IM[0..7]**
- **CRM_INT_DETECT**

5.17.1 Per-Monitor Configuration

Each counter monitor is defined by the following configurable parameters:

Parameter	Bits	Description
CounterAddress	22	Address of the counter to monitor.
CounterSize	2	Software must specify the size of the counter being monitored: 00b = 32 bits [default] 01b = 16 bits 10b = 8 bits 11b = Reserved
RateWindow	24	Window over which RateLimit is defined, in units of the global PollingPeriod parameter. A value of 0b indicates an infinite window (useful in the case of monitoring a counter with an expected rate of 0.)
RateLimit	32	Limit imposed on the specified counter's amount of increase over any given RateWindow period of time. If the actual observed increase is greater than this limit, the bit in COUNTER_MON_IP corresponding to this counter monitor is set.
Enabled	1	Monitor is disabled when set to 0b.

Note: The monitors do not directly support multi-word counter registers. It is expected that the monitors query only the bottom 32 bits of multi-word counters at a sufficiently fast rate (i.e. with a sufficiently small PollingPeriod) to avoid missing 32-bit overflows.

5.17.2 Per-Monitor State

Each monitor maintains the following state which is made visible to software:

Value	Bits	Description
LastCount	32	Last polled counter value. This read-only value is not necessarily expected to be useful to software. It is cleared to zero when the monitor's Enabled bit is set to 0b.
ExceedCount	32	Number of windows over which the RateLimit was exceeded. Cleared to zero on a write of any value. Cleared to zero when the Enabled bit is set to 0b.

Whenever hardware increments a particular monitor i's ExceedCount due to its RateLimit being exceeded, it also sets bit i%32 in CRM_IP[i/32]. If this interrupt bit is not masked in CRM_IM[i/32], an interrupt is reported to the CPU.

5.17.3 General Configuration

In addition to the per-monitor configuration and status registers, there is a single global configuration parameter, applicable to all counter monitors:

Parameter	Bits	Description
PollingPeriod	16	Number of LSM clock cycles between counter samples, in multiples of 1024. The minimum, default, value of 1 gives a polling period of about 20 microseconds, assuming a 50 MHz LSM clock frequency.

A counter's increase from one polling sample to the next is calculated using two's complement arithmetic, so the polling rate constraint is determined by

```
PollingPeriod[seconds] <= Max(2^CounterBits[i] / MaxCounterRate[i])
```

evaluated over all counters (i) being monitored. For example, a single 32-bit, 10 Gb/s port octet counter imposes a maximum PollingPeriod of 2^32 / (10 Gb/s / 8 bits) = 3.4 seconds.

5.18 JTAG Interface

The JTAG controller is compliant to the IEEE 1149.1-2001 specification. The JTAG provides basic external chip debug features:

- Access to an identification register.
- Access to the boundary scan.
- Access to the internal scan chains.
- Ability to Clamp and HighZ all outputs (except SerDes).

The maximum frequency of operation is 40 MHz.

The Supported operations of these registers are:

- **Load IR** (instruction register)
- **Capture** — Initializes/captures/freezes value of register
- **Shift** — Serially shifts in/out value into/out of register.
- **Update** — Validates the contents of the register (i.e., Logic can now use the new value for its internal operation).

The JTAG reset domain is separate and independent from the chip reset domain.

5.18.1 Tap Controller

The tap controller is a finite state machine of 16 states controlled by the 5-pin JTAG interface. It is defined by IEEE 1149.1-2001.

5.18.2 Instruction Register

Supported JTAG instructions are shown in Table 5-12.

Table 5-12 JTAG Instructions

Instruction	Code (6 bits)	Description
IDCODE	x01	Selects the identification register.
SAMPLE/PRELOAD	x02	Select the boundary scan register. Sample input pins to input boundary scan register, preload the output boundary scan register.
EXTEST	x03	Select the boundary scan register. Output boundary scan register cells drive the covered output pins. Input boundary cell registers sample the input pins.
HIGHZ	x06	Selects the bypass register and sets all covered output pins to high impedance.
CLAMP	x07	Forces a known value on the outputs, but uses the bypass register to shorten scan length.
BYPASS	x3F	Selects the bypass register.

5.18.3 Bypass Register

The bypass register is a 1-bit register that connects between TDI and TDO. When the bypass register is selected by the instruction, the data driven on the TDI input pin is shifted out the TDO interface one cycle later.

5.18.4 JTAG Scan Chain

The boundary scan register is a 162-bit deep shift register. Refer to the BSDL description file for pin assignment.

§ §

6.0 Filtering and Forwarding Unit (FFU)

6.1 Overview

The frame header is presented to the Frame Filtering and Forwarding Unit which associates one or more actions with the frame. This unit contains 32 consecutive slices containing both TCAM and SRAM. The last slice(s) can optionally be used for egress ACLs that apply in parallel to multiple egress ports.

Each slice has the following elements:

- Configuration to select which frame header fields are selected for the TCAM comparison
- 512-entry x 36-bit TCAM block used for key comparison
- Hit detection circuitry which can cascade across consecutive slices to create keys larger than 36 bits
- Priority hit detection to determine the hit in each slice with the highest index
- 512-entry x 40-bit SRAM block to store an ingress action(s) associated with the highest priority hit
- The last slice can optionally feed its final 512 hits into the egress ACL unit
- Configuration registers to control the slice behavior for various frame scenarios and key configurations

The overall unit is shown in Figure 6-1.

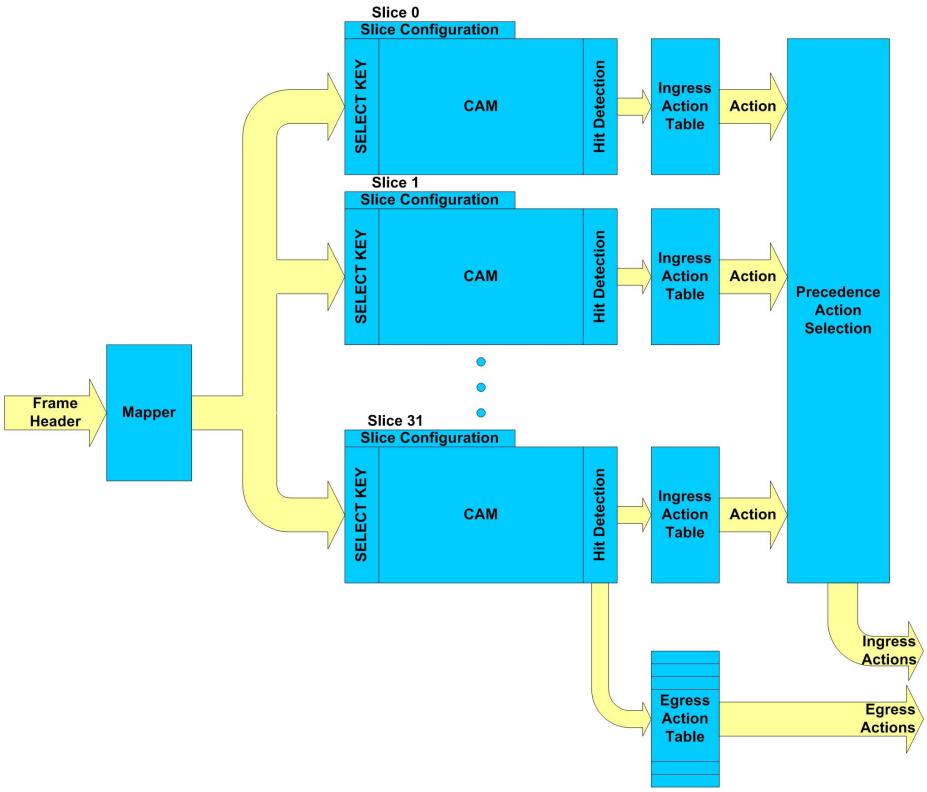


Figure 6-1 FFU Unit

Each TCAM entry has 74 bits of configuration:

- 36-bit mask
- 36-bit key
- 1 valid bit
- 1 case bit (see Section 6.3)

See the FFU_SLICE_TCAM table for exact bit positions.

6.2 Frame Header Mapper

The Frame Header Mapper is used to create smaller mapped keys from some frame header fields to better utilize the TCAM resources. These mapped keys are available in the FFU slices, along with all the original frame fields. Most of these mapped keys are 4 bits and may be used as bits 32..35 of the larger key, leaving 32 bits for another key field such as DIP. The mapper also detects when frames should be routed, based on per-source-port, per-VLAN, and per-DMAC settings. The mapper supports the following functions:

- Maps 8-bit L4 protocol to 4-bit MAP_PROT field.
  - The mapper recognizes up to 8 L4 protocols. If the protocol received matches one of these protocols, the 3-bit MAP_PROT field associated is retrieved, or 0b if the protocol does not match. The 4th bit of MAP_PROT will be 1b if the frame is a non-head IP fragment for IPv4, or if the IPv6 options were too long (which means that the L4 header information is not valid, since a layer 4 header could not be found).
- Maps 16-bit Ethernet type field into 4-bit MAP_TYPE field.
  - The mapper recognizes up to 16 Ethernet types. If the Ethernet type received matches one of the known ones, a 4-bit field associated with the protocol is retrieved. If there is no match, this 4-bit field is set to 0b. Useful for detecting non-IP Ethernet frames, such as ARP. The scenario distinguishes IPv4 and IPv6 from other L3 protocols, so MAP_TYPE is not needed for that purpose.
- Maps 48-bit DMAC and SMAC addresses into a 4-bit MAP_DMAC and MAP_SMAC fields.
  - The mapper recognizes up to 16 Ethernet MAC addresses. If the frame's DMAC address matches one of the known ones (and its validDMAC bit is true), a 4-bit MAP_DMAC field associated with the MAC address is retrieved, a router bit is set, indicating if this DMAC address is an address of one of the virtual routers. If there is no match, this 4-bit field is set to 0b. Likewise, the source MAC address is mapped to MAP_SMAC, but without the router bit. Even if you do not intend to use the MAP_DMAC or MAP_SMAC as TCAM keys, at least one MAC entry must be specified with its “router” bit true to do IP routing. The entry can be configured to ignore a specified number of LSB bits when comparing MAC addresses, which is used for VRRP and can also be used to detect the reserved IEEE multicast ranges.
- Maps physical source port into a 4-bit MAP_SRC field.
  - The mapper retrieves a 4-bit field associated with each source port, along with a routable bit indicating if this port is routable. This mapping is very useful for applying ACL's to a set of equivalent ports without duplicating the TCAM entries.
- Maps 16-bit L4 destination and source ports into a 16-bit MAP_L4_DST and MAP_L4_SRC fields.
  - The mapper distinguishes up to 64 contiguous ranges in the 16-bit port address space. It compares the L4 source port to all port boundaries stored in the FFU_MAP_L4_DST registers and retrieves a new 16-bit MAP_L4_DST from the register if the frame's port is greater or equal to the port stored in the current register and less than the port stored in the next register if and only if the protocol matches the 3-bit protocol retrieved from the FFU_MAP_PROT, the frame is not a non-head-frag, and the entry is marked valid. If there is more than one match, the last match is used. If there is no match, the MAP_L4_DST is set to the original L4_DST (not 0). This usually enables the TCAM to only look at MAP_L4_DST and never need the raw L4_DST, thus making exact-match port comparisons consume no resources. The L4 source port is mapped in a similar way except using independent configuration registers.

- Maps SIP and DIP addresses into a 4-bit MAP_SIP and MAP_DIP fields.
  - The IP address prefix is compared with 16 known IP address prefixes. If the frame's IP address prefix matches one of the known ones (and the appropriate validSIP or validDIP bits is true), a 4-bit field associated with this IP address prefix is retrieved. If there is no match, this 4-bit field is set to 0b. Each register contains a value and the number of least significant bits to ignore. This feature is especially useful when most IPv6 routing or ACL rules use only a few common prefixes. A 32-bit IPv4 address is padded with 0's before being compared to the 128-bit value in the registers.
- Maps VLANs into a 12-bit MAP_VLAN key.
  - Maps all 4096 VLANs into an arbitrary set of VLAN groups and indicates if each VLAN is routable or not. Rules which treat several VLAN's the same can compress those VLAN's into the same VLAN group, thus avoiding TCAM expansion.
- Maps packet length into a 4-bit MAP_LENGTH field.
  - The mapper recognizes up to 16 packet length ranges (from IP packet length field, not the Ethernet frame length). It compares the current IP packet length received to all packet lengths stored in the FFU_MAP_LENGTH registers, and retrieves a 4-bit key stored in each register if the length received is greater or equal to the length stored in the current register and less than the length stored in the next register. Can be used to keep statistics on user-defined length bins, or to drop long packets based on complicated rules, or even to route differently based on length. In all cases, if there is more than one match within one MAP, the highest entry wins.

6.3 Slice Activation and Overloading

A single slice can be used to hold two different type of keys referred to as “cases”. The selection of case “A” or “B” depends on the packet received. The data used to make the decision are:

Table 6-1 Slice Activation and Overloading

Bit(s)	Description
0:1	Frame format: 00b = Not an IP frame 01b = IPv4 10b = IPv6 11b = IPv6 with an IPv4-in-IPv6 DIP. Can optionally use IPv4 routing rules for this frame.

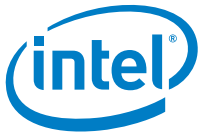


Table 6-1 Slice Activation and Overloading (Continued)

Bit(s)	Description
2:4	Frame handling: 000b = **Switched** — This frame should not be routed, and has a destination GLORT of 0 (i.e. no ISL tag). Ingress ACL's may be applied. 001b = **Switched-GLORT** — This frame should not be routed, and destination GLORT has been set != 0 in the ISL tag by an ingress chip in a multi-chip fabric. Ingress ACL's may be skipped. 010b = **Mgmt** — FTYPE is "mgmt." Generally, the FFU should do nothing to these frames. 011b = **Special** — FTYPE is "special." This usually means mirror traffic or CPU directed control frames, but ACL's may be used. 100b = **Routable** — This frame is a unicast IP frame and the ingress port and ingress VLAN are marked as routable and the DMAC matches a virtual router. This usually means that routing rules should be applied, as well as ingress ACL's. If the frame is actually routed, then it will be marked FTYPE "routed" in the ISL tag for subsequent chips, otherwise it will be FTYPE "normal" and will just be switched. 101b = **Routed-GLORT** — This unicast IP frame was routed by the ingress chip in a multi-chip fabric, and the ISL tag has been marked with FTYPE "routed". Generally ingress ACL's would no longer be applied. However, unicast routing rules must still be applied so that the egress port can make routing modifications to the VLAN/DMAC/TTL of the frame. In this case, the destination GloRT value must be non-zero. 110b = **Routable-multicast** — This frame is IP multicast, the ingress port and ingress VLAN are marked as routable, the destination GloRT is zero. Ingress ACL's and multicast IP routing rules are appropriate. If the frame is in fact routed, then it will be marked FTYPE "routed" in the ISL tag for subsequent chips, otherwise it will be FTYPE "normal" and will just be switched. 111b = **Routed-multicast-GLORT** — This frame is IP multicast and the ingress chip in a multi-chip fabric has already determined that it should be routed. Ingress ACL's are generally not applied. IP multicast routing rules are usually not needed either, since the destination GloRT itself will select the port/vlan pairs to exit on. In this case, the destination GloRT value must be non-zero.

This 5-bit field, referred to as "scenario" is used to index two 32-bit registers (FFU_SLICE_VALID and FFU_SLICE_CASE) which are used to associate a case ("A" or "B") that is used when searching in the TCAM for that slice. The registers are indexed by slice number, and each bit in the register corresponds to one of the 32 scenarios. Each line in the slice also has two bits; valid and case. The FFU will proceed as follows:

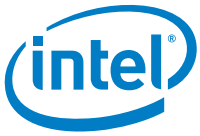
- Retrieve the slice valid bit and slice case bit for each slice from FFU_SLICE_VA,LID and FFU_SLICE_CASE, which depend on frame type and routing conditions
- Skip slices that are not valid for this scenario,
- Use different TCAM keys for case "A" and case "B."

This feature has two purposes; to reduce power and overload slices to hold two orthogonal types of rules, such as IPv4 and IPv6.

6.4 Search Key Selection

The selection of which frame header fields to present to the TCAM depends on the case and is configurable per-slice, per-case number. Each of these registers has 5 x 6-bit fields plus two additional control bits, each field allows setting of up to a 64-way selection. The available fields are:

- Standard keys from Frame Header:
  - DMAC: 48
  - SMAC: 48
  - TYPE:16 (innermost EtherType
  - ISL: 64
    - ISL[63:62] => FTYPE
    - ISL[61:60] => VTYPE or MTYPE
    - ISL[59:56] => Switch Priority
    - ISL[55:48] => Contents found in ISLUSER[7:0]
    - ISL[47:32] => Contents found in VLAN_VPRI[15:0]
    - ISL[31:16] => Source GloRT (SGLORT)
    - ISL[15:0] => Destination GloRT (DGLORT)
  - VLAN_VPRI: 16 (includes outermost VLAN:12, VPRI: 4)
  - VPRI: 4 (part of VLAN_VPRI)
  - PRI: 4 (part of ISL)
  - SIP: 32 or 128
  - DIP: 32 or 128
  - LENGTH:16 (IP header length field)
  - TTL: 8 (IP time-to-live field)
  - PROT: 8 (L4 protocol for IPv4, innermost nextHeader in IPv6)
  - MISC: 8 (5'b0,HeadFrag,DoNotFrag,TrapIPoptions)
  - DS: 8 (DiffServ field of IPv4, or TOS field of IPv6)
  - FLOW: 20 (only for IPv6)
  - L4SRC: 16
  - L4DST: 16
- Keys taken from configurable offsets in L4+ header:
  - L4A: 16
  - L4B: 16
  - L4C: 16
  - L4D:16



- Derived keys from Mapper:
  - MAP_PORT: 4
  - MAP_TYPE: 4
  - MAP_LENGTH: 4
  - MAP_PROT: 4 (includes NHF)
  - MAP_SMAC: 4
  - MAP_DMAC: 4
  - MAC_SIP: 4
  - MAC_DIP: 4
  - MAP_L4SRC: 16
  - MAP_L4DST: 16
  - MAP_VLAN: 16 (includes VPRI as upper 4 bits)
- Other:
  - SRC: 25 (source port bitvector)

Note: The ISL tag received with the frame is originally 64 bits long but is reduced to 48 bits before being presented to the FFU by extracting the VLAN/VPRI/CFI bits from the ISL tag.

6.5 Slice Cascading

Slices can be cascaded together to produce larger search line and/or larger set of actions. Cascading is shown in Figure 6-2.

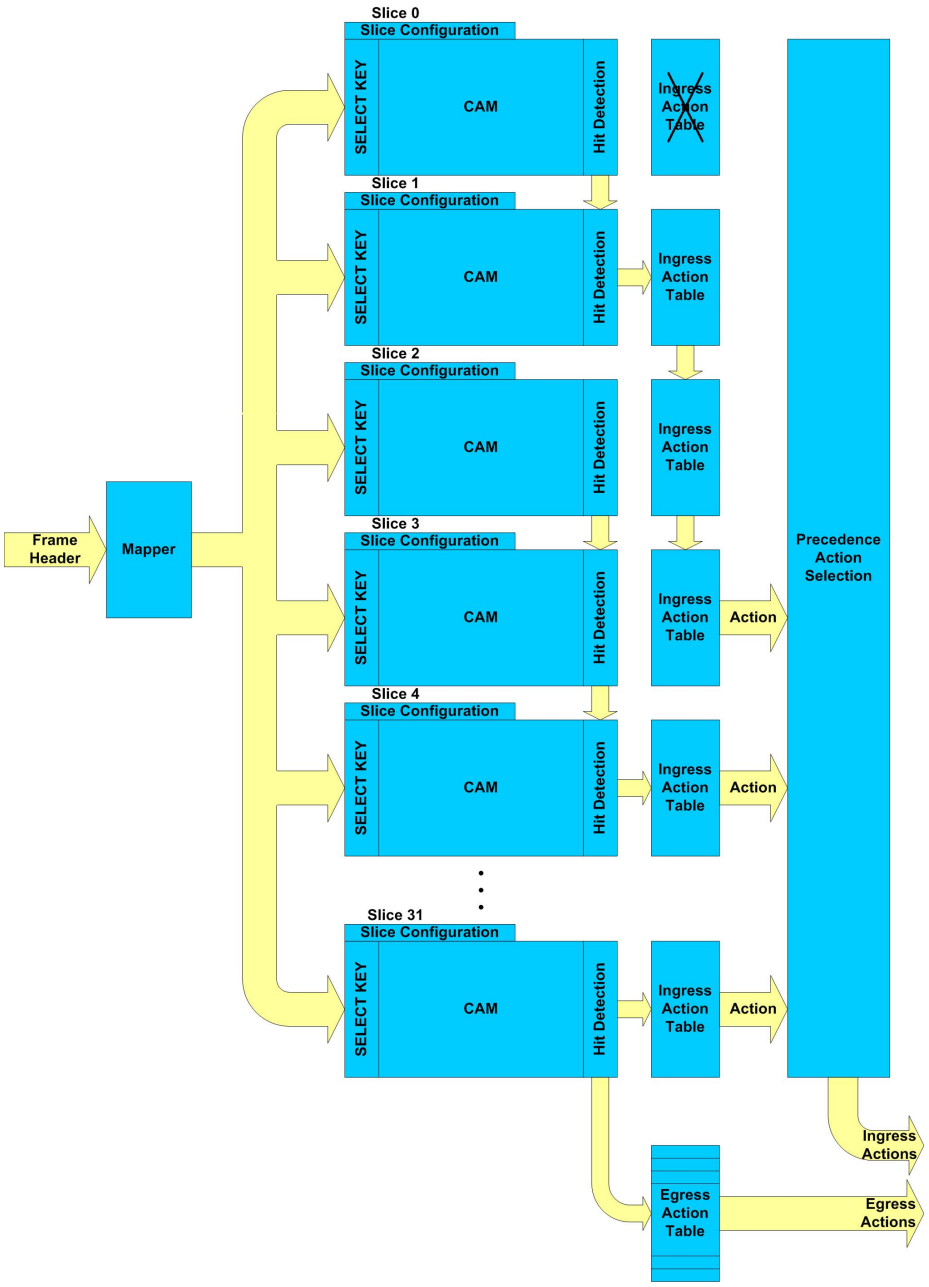
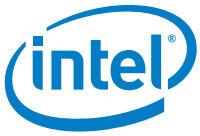


Figure 6-2 FFU Slice Cascading



This example shows two CAM cascades and one ACTION cascade. The two CAM cascades are using block 0-1 and block 2-3-4 respectively. The ACTION cascade is using block 1,2,3.

When two or more CAMs are cascaded together, then the 'hit' line is the highest line that produces a match across all CAM entries together.

The cascading of CAMs and ACTIONS tables are configurable per slice and per case number. The FFU_CASE_CFG and FFU_CASCADE_ACTION registers control the cascading option for each slice and each case, so it is possible to have different cascade arrangements for case “A” and “B”.

6.6 ACL Mappings Examples

The basic ACL's map to this format as shown in following examples, where each line is another slice. Note that these examples are not exhaustive.

```
Layer 2 Simple ACL
==================
 SMAC_LO:32, MAP_SRC:4
 SMAC_HI:16, VLAN_VPRI:16, MAP_TYPE:4

Layer 2 Extended ACL
====================
 SMAC_LO:32, MAP_SRC:4
 DMAC_LO:32, MAP_TYPE:4
 SMAC_HI:16, DMAC_HI:16
 VLAN_VPRI:16, TYPE:16

IPv4 Lookup
====================
 DIP:32, MAP_SRC:4 or MAP_VLAN:4

IPv6 Lookup
====================
 DIP_A:32, MAP_SRC:4
 DIP_B:32, MAP_PROT:4
 DIP_C:32
 DIP_D:32

IPv4 (S,G) for multicast
========================
 DIP:32, MAP_SRC:4
 SIP:32, MAP_PROT:4

IPv4 Layer 3 Simple ACL
=======================
 SIP:32, MAP_SRC:4
 MAPL4DST:16, L4DST:16, MAP_PROT:4

IPv6 Layer 3 Simple ACL
=======================
 SIP_A:32, MAP_SRC:4
 SIP_B:32, MAP_PROT:4
 SIP_C:32
 SIP_D:32
 MAP_L4DST:16, L4DST:16

IPv4 Layer 3 Extended ACL
=========================
 DIP:32, MAP_SRC:4
 SIP:32, MAP_PROT:4
 MAP_L4DST:16, L4DST:16
```

```
 MAP_L4SRC:16, L4SRC:16
 PROT:8, DS:8, TCP_MISC:8

IPv6 Layer 3 Extended ACL
=========================
 SIP_A:32, MAP_SRC:4
 SIP_B:32, MAP_PROT:4
 SIP_C:32
 SIP_D:32
 DIP_A:32
 DIP_B:32
 DIP_C:32
 DIP_D:32
 MAPL4DST:16, L4DST:16
 MAPL4SRC:16, L4SRC:16
 PROT:8, DS:8, TCP_MISC:8
```

6.7 Ingress Action

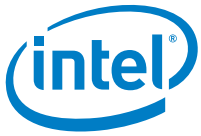
The Filtering and Forwarding Unit produces several different "action fields" which are subsequently used to modify the frame and/or determine its destination. When a TCAM entry "hits" in the TCAM, its associated action entry, which is stored in the FFU_SLICE_SRAM register, is evaluated to determine how the action fields should be modified.

Since multiple TCAM entries may hit (up to one per slice), there may be multiple action entries which attempt to modify the action fields for a given frame. As long as each action entry attempts to modify different action fields, then there is no conflict. However, if one action field is modified by more than one action entry that has hit, the conflict must be resolved.

Each action entry contains a three-bit "precedence" field which is used to resolve such conflicts. If more than one action entry attempts to modify the same action field, the action field is assigned the value from the action entry that has the highest precedence. If more than one action entry for a given action field have equally high precedence, the tie is resolved in favor of the action entry which resides in the higher-numbered slice.

The list of orthogonal action fields are as follows:

- FLAGS (mask: 8, value: 8)
- ROUTE_ARP (arp_index: 14, arp_count: 4) or ROUTE_GLORT (dglort: 16, the frames do not need to be routable for this action to take effect)
- SET_VLAN: 12
- SET_VPRI: 4
- SET_PRI: 4
- SET_DSCP: 6
- SET_USR (mask: 8, value: 8)
- SET_TRIG (mask: 8, value: 8)
- ACTION_A: 16
- ACTION_B: 16
- COUNT_0: 10
- COUNT_1: 10



- COUNT_2: 10
- COUNT_3: 10

The FLAGS/SET_USR/SET_TRIG provide mask/value pairs. Each bit of these actions has its own 3-bit precedence, and each bit makes its own decision about when to replace the bit with a new value. That allows multiple slices to set different bits in the USR/TRIG fields, or to pass some bits through unmodified. Since it is not known what the "user" will do with USR, and masked compares of TRIG in the Trigger unit are possible, this flexibility may be valuable. It has little hardware expense (except added complexity).

FLAGS include several single bit actions which can be set or cleared independently. The defined bits are DROP, NOROUTE, LOG, TRAP, and RX_MIRROR, with 3 bits reserved. If DROP is set to 1b, the frame is dropped. If NOROUTE is set to 1b, the ROUTE action should be ignored, but the frame can still be L2 switched instead of routed. The idea is that by attaching ACL's to physical ports, DROP is done at L2+, whereas attaching an ACL to a VLAN allows the blocking of routing certain types of frames from that VLAN.

The ROUTE is used to define the next hop. There are two type of ROUTE action; ROUTE-GLORT and ROUTE-ARP. The ROUTE-ARP is used for IP routing, while the ROUTE-GLORT is used for special layer 2 switching when the destination cannot be determined using a layer 2 lookup. For ROUTE-ARP, the arp_index picks the base ARP entry, and arp_count specifies the number of ECMP arp entries to hash over, from 1 to 16 (where 16 is encoded as 0).

ACTION_A and ACTION_B are reserved for future use.

The SET_TRIG sets a tag that is passed to the Trigger unit.

The SET_VLAN replaces the ingress VLAN.

The COUNT's represent indices into 4 parallel banks of counters. Each counter bank may be configured to count either frames or bytes. It is also possible to make these counters police or raise interrupts.

The actions are encoded in a 40-bit SRAM as follows:

```
PARITY:1
PRECEDENCE:3
COUNT_BANK:2
COUNT_INDEX:10
ACTION:8
DATA:16 (or 22 overlapped with LSB bits of ACTION)
```

A parity error is OR'ed across all slices and raises an interrupt. This does not actually report the failed index, so the software must search for it.

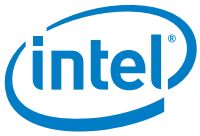
All non-NOP actions are counted somewhere. By convention, the 0 index of the counters is reserved to mean "don't count". See Section 12.7 for more information about the counters/policers.

ACTION chooses among non-counter actions and DATA provides extra arguments. The ACTION cases and relevant data are as follows:

```
ACTION==00XXXXXX -> Reserved
ACTION==01XXXXXX -> Route
  ROUTE-TYPE{21}
    if ARP (route type is 0)
      ARP_COUNT{17:14}
      ARP_INDEX{13:0}
    if GLORT (route type is 1)
      DEST_GLORT{15:0}
ACTION==10XXXCBA -> 8-bit mask/value or 16-bit exclusive actions.
  CBA->FLAGS/SET_USR/SET_TRIG/ACTION_A/ACTION_B
    MASK{7:0}
    VALUE{15:8}
    MASK_DROP{0}
    MASK_TRAP{1}
    MASK_LOG{2}
    MASK_NOROUTE{3}
    MASK_RX_MIRROR{4}
    SET_DROP{8}
    SET_TRAP{9}
    SET_LOG{10}
    SET_NOROUTE{11}
    SET_RX_MIRROR{12}
ACTION==11XXDCBA -> 12-bit, 6-bit, or 4-bit parallel actions.
  A->SET_DSCP
  B->SET_VLAN
  C->SET_VPRI
  D->SET_PRI
  VPRI/PRI is set from DATA{15:12}
  VLAN is set from DATA{11:0}
  DSCP is set from DATA{5:0}
```

In addition to providing a counter combined with any other action, many combination actions are possible, for example:

```
SET_SW_VLAN+SET_SW_PRI
SET_DSCP+SET_NXT_PRI+SET_SW_PRI
DENY+LOG
```



6.8 Egress ACLs

The egress ACL unit follows the last slice. It takes the cascade hit from the last slice, and splits the 512 hits into 32 groups of 16 entries each. Each group can produce a single precedence hit (which is the index with the highest numbered hit within the group). The 'egress' unit can thus produce 32 actions simultaneously (contrary to the slices, which can produce only one ingress action per slice). Each entry of each group has an action associated with it, the action is a 2-bit field:

0b = Drop

1b = Log

The group behavior is controlled by the FFU_EGRESS_CHUNK_CFG registers (32 registers, one per group) which associate an egress port membership mask to the group, and one bit allowing cascading this group to a neighboring group to extend the size of the precedence hit detect. Egress actions only apply to those egress ports in its membership list. Unlike ingress actions, the egress actions are applied in parallel by egress port. On a flood or multicast, each copy will independently check its egress ACL action. The 2 actions can be combined, for example, to drop and log.

If an egress ACL is set with the action = drop for every possible egress port, when it gets to the trigger, the destination mask is all 0's, but the frame Action Code is either Flood (due to destination MAC miss in MA_TABLE) or Forward normally, depending if the DMAC is present in the MAC table or not. The egress ACLs don't affect the action mask.

§ §

NOTE: ***This page intentionally left blank.***

7.0 Routing

7.1 Overview

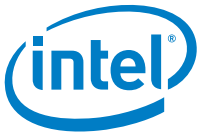
Routing is implemented as a cooperative effort among multiple units. It includes the following steps for a single chip system:

- **RECEIVE & PARSE** — The receive EPL MAC layer parses the packet to detect if the frame is IP and to extract pertinent information for routing purposes. The extracted fields are sent to the frame handler, while the original frame is stored in receive memory.
- **ASSOCIATE** — The frame handler associates a VLAN with the frame (using VLAN received and the PORT_CFG_1 register setting to assign a default or override what was received) and marks the frame as "switched." The egress VLAN is set to be equal to the ingress VLAN.
- **ROUTABLE** — The frame handler then uses FFU_MAP_SRC, FFU_MAP_MAC and FFU_MAP_VLAN registers to determine if the frame is routable. A frame is routable only if the VLAN, the port and the destination MAC address are routable. The frame is then marked as "routable".
- **IP LOOK UP** — The frame handler parses the frame through the FFU TCAM, which can result in having a ROUTE-ARP action associated with the frame. The ROUTE-ARP action includes an ARP index (and an ARP count for ECMP) to find the next hop for this route which is contained in the ARP table.
  - The scenarios selected for the IP route table are "routable" or "routable-multicast".
  - The FFU TCAM can also contain other directives, such as ACL's to change priority. For more details, see Section 6.0, "Filtering and Forwarding Unit (FFU)".
- **ARP LOOKUP** — The ARP table is indexed with the ARP index (if the ARP count is greater than 1, a hash function is used to select one entry) and either returns a VLAN/DMAC pair (arp-mac), a destination GLORT/VLAN pair (arp-glort) or a VLAN for IPv6 addresses with embedded MAC address (arp-embed-mac).
  - The frame is marked "routed" as this point and the frame type in the Intel ISL tag reflects this change of status if the frame exits this chip on a Intel tagged port.
  - Note that arp-mac and arp-embed-mac are typically used for routing IP unicast frames while arp-glort is typically used for IP multicast frames.
    - Routing multicast packets requires replicating the packet within the ingress VLAN to any ports that are listeners for this multicast as well as routing the packet to other VLANs which may require replication to any port as well. This is accomplished by allocating a unique GloRT to each multicast group and by controlling the destination mask directly in the GLORT_DEST_MASK table and also by using a destination VLAN of 0 which is reserved in the layer 2 lookup stage to prevent the packet from being filtered by that unit.
    - The arp-glort returns a 3-bit MTU size index for checking MTU size violations.

- **MA LOOKUP** — The frame handler searches the egress VLAN/destination MAC in the layer 2 lookup table to retrieve a destination GloRT or a destination mask. If the entry is not found, the destination GloRT is set to the flood GloRT if the DMAC is not the broadcast MAC, or to the broadcast GloRT if the DMAC is the broadcast MAC. The frame handler also presents the ingress VLAN/source MAC for address learning. Note that security is also checked at this point if it is enabled.
  - The ingress VLAN spanning tree state is checked after lookup. Learning is canceled if the state is LISTENING or DISABLE. Forwarding is canceled if the state is DISABLE, LISTENING or LEARNING.
  - The egress VLAN membership and forwarding state is also checked if and only if the frame is not a routed multicast frame. The frame is dropped if the forwarding state is DISABLE. The EGRESS VID table also includes a 3-bit MTU index size, which is used only if the arp entry is of type arp-mac or arp-embed-mac.
- **TRAP** — Check traps to determine if the frame must be rerouted to the CPU. Possible IP traps are listed in Section 7.4. Other traps also exist and are listed in Section 8.8.
- **PORT MAP** — The destination GloRT is then used to recover a destination mask from the PORT MAPPING unit. The destination mask indicates which port is going to receive a copy of the frame. There is usually only 1 bit set for a unicast frame unless the frame is flooded, while multicast frames have multiple bits set.
  - The frame also passes through LAG filtering, triggers and congestion management before being forwarded. Those steps refine the destination mask and may change the priority and/or destination GloRT.
- **SCHEDULE** — The frame is then sent to the scheduler for transmission along with new attributes to be added: routed attribute, egress VLAN, egress DMAC, destination mask, destination GloRT, updated DSCP, updated VPRI, updated DSCP, etc.
  - A sub unit in the scheduler (called MTABLE) replicates the frame multiple times on the same port if a multicast frame needs to be replicated across multiple VLANs.
  - An egress VLAN of 0 is not consider valid for the scheduler perspective. So if an egress VLAN of 0 was is for multicast (as recommended), the scheduler assumes that the frame is actually associated with the ingress VLAN instead.
- **TRANSMIT MAC** — The transmit MAC then detect that the frame is marked “routed” and changes the content of the original frame, replacing the fields that need to be replaced; decrementing TTL by 1, replacing DMAC, replacing VLAN, updating DSCP, recomputing IP header, etc.

7.1.1 Multi-Chip Routing

The routing in a multi switch system is slightly more complex. In a multi switch system, each port is either attached to an external device (“external port”) or to an another switch of the multi-chip system (“internal ports”). The transmit MAC of each port is configured accordingly. It is also configured independently to transmit an ISL or not. In a multi-chip system, routing is performed at any step but the actual frame modification (changing VLAN/DMAC, decrementing TTL, etc...) is only done at external ports. This requires that the IP lookup be done on both the ingress switch and the egress switch. The following bullets detail the role of each switch in the system regarding the process detailed above for a single switch system:



Ingress Switch:

The ingress switch performs all the same steps as in the single switch system except that the transmit MAC does not apply the layer 3 modifications on the internal ports. The content of the original frame is, however, modified to include an ISL tag which has the new switch priority and the new destination/source GloRT, but the frame is transmitted with the ingress VLAN and not the egress VLAN, and the DMAC/SMAC remain unchanged. The ISL tag FTYPE also identifies this frame as "routed".

- The IP addresses may be sorted into 2 categories: those that exit only on internal ports and those that may exit on either internal or external ports. The first category does not need to be included in the "routable" scenario, while the second category must be included in both the "routed-glort" and "routable" scenario.

Intermediate Switches:

The intermediate switch does not need an IP route table if all ports are internal ports. The frame is then switched based on the destination GloRT. The step IP LOOKUP is done but should return no route action and the ARP lookup is skipped. The MA LOOKUP is still done to learn the source MAC address using the source GloRT received as part of the ISL tag, but does not do a destination lookup, as the destination GloRT is already known. The PORT MAP determines the next ports for the frame. The TRANSMIT MAC simply forwards the original frame unchanged (unless other directives in the FFU such as ACL changes priority, user field or other fields).

- The FFU may potentially be disabled in the intermediate units to reduce latency and power. The egress VLAN is automatically set to 0 if the packet was a routed packet. This ensures that the layer 2 processing does not filter out the packet in any way.

Egress Switch:

The egress switch performs the same steps as a single switch system.

Arbitrary topologies, i.e. entering and exiting the multi-switch system at any point, are obtained by simply loading the appropriate IP lookup table in the FFU under the right scenarios.

7.2 ARP Unit

The packet is routed if an only if the FFU returns a route-arp action. The inputs to the ARP units are:

- A source MAC and ingress VLAN
  - — These are not changes by the ARP unit.
- The route action from the FFU including the following fields:
  - — ARP index (14 bits)
  - — Path count (4 bits)

The outputs of this unit are:

- An updated egress VLAN and destination MAC.
  - — If the ARP unit is not traversed (meaning the packet is not routed), the destination MAC is not changed, while the egress VLAN is set to 0 for a Intel ISL-tagged packet received with a frame type "routed" with an FFU disabled. The egress VLAN is set equal to the ingress VLAN in all other cases.

  - If the unit is traversed, the destination MAC is only changed for arp-mac and arp-embed-mac ARP entries but the egress VLAN is systematically set to the VLAN defined in the arp entry.
- An updated DGLORT
  - The destination GLORT is updated to the value defined in the arp-glort if the ARP entry is this type. Otherwise, the destination GloRT associated with the packet is not changed.
- A 3-bit MTU index if the entry was a arp-glort entry

The unit computes a hashing function using IP fields before indexing the table. The output of this hash function is used to compute the final index and is typically used for implementation of ECMP. For details on the hashing function, see Section 10.0, "Frame Hashing".

Note: The ARP table entry 0 is reserved and cannot be used for routing.

The 14-bit ARP index received as part of the route-arp command points to the first entry in the ARP table. The 4-bit path count defines the number of entries, including the first one, that could be used as alternative paths (a value of 0 in the path count means 16 entries) and uses the result of the hash function to compute an ARP index:

```
ARP index = (base ARP index) + (hash * (# of paths) / (max hash))
```

The ARP table is 16 K x 64 and contains three fields:

- Type of entry (2 bits)
  - Indicates the type of entry and how the rest of the table is interpreted.
  - If MAC address:
    - Next-hop MAC address (48 bits)
    - Egress VLAN (12 bits)
  - If MAC address extracted from IPv6:
    - Egress VLAN (12 bits)
  - If GLoRT:
    - Destination GloRT (Always replace the received destination GloRT).
    - 3-bit MTU size index.
    - VLAN (Should be set to 0b for routing IP multicast frames).
- Virtual Router ID (8 bits).
- Parity (1 bit)

Note: If a destination GLORT is specified, a Layer2 Lookup is still done but only for a source address lookup to allow learning and security check. If a MAC address is specified, the layer 2 lookup is done normally (both destination and source).

The routed frame VRID is set by this table and pass through the pipeline to the egress port. If the virtual routing id is 0, the source address of the frame is replaced by the one defined in the SRC_MAC_LO/HI registers in the EPL. If the virtual routing id is not 0, the source address is replaced by the one defined in the SRC_MAC_VIRTUAL_LO/HI registers in the EPL, with the VRID substituted for the low byte of the MAC address.

7.3 ARP_USED Table

The ARP_USED table is a 16 K x 1 table (implemented as a 512x32) to indicate if the corresponding entry in the ARP_TABLE has been used recently or not. The CPU may clear the bit by writing a 1b to it, while writing a 0b has no effect. The hardware automatically sets the bit whenever an entry is used. The software may poll this table periodically to detect which entries where used recently and delete unused entries from the table if needed.

7.4 IP Traps & Logs

The switch is capable of capturing the following IP frames:

- Trap routed IP unicast frames with TTL = 1b or TTL = 0b
- Log routed IP multicast frames with TTL =1b or TTL = 0b
- Trap IGMP Frames
- Trap IP frames with certain options
- Trap MTU size violations
- Log routed IP unicast to same LAN for potential redirect
- Log routed IP unicast received as MAC multicast

7.4.1 Trapping of IGMP Frames

The trapping of IGMP frames can be turned on per VLAN using the VLAN_TABLE::TrapIGMP bit. This is a layer 2 function and is detailed in Section 8.8. Note that trapping IGMP frames is only applicable for IPv4 packets. The equivalent concept for IPv6 is MLD and is implemented as a subset of ICMPv6. It always uses a TTL of 1 and the MLD frames are thus trapped using the TTL trap option, which is detailed in Section 7.4.3.

7.4.2 Trapping of IP Frames with Options

The switch has the capability of trapping IP frames with options to the CPU. The trapping is globally enabled

by turning on the SYS_CFG_ROUTER::trapIPOptions and also requires configuring the PARSE_CFG[i]

register in the EPL to enable the options of interest for each port.

7.4.3 Trapping and Logging of Frames with TTL=1 or TTL=0

The switch has the capability of trapping (unicast) or logging (multicast) IP routed frames with a TTL of 1 or 0. This feature is useful for tracing routes in a network by doing a hop-by-hop route discovery. There are 3 options as configured in SYS_CFG_ROUTER::trapTTL:

For routed unicast frames:

1. Discard frames with TTL 1 or TTL 0.
2. Trap ICMP frames with TTL 1 or TTL 0 and discard all others.
3. Trap all frames with TTL 1 or TTL 0.

For routed multicast frames:

1. Do not route frames with TTL 1 or TTL 0.
2. Log ICMP frames with TTL 1 or TTL 0 and do not route all others.
3. Log all frames with TTL 1 or TTL 0.

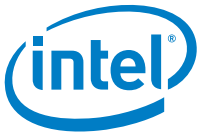
Note: This field only applies to frames that are routed. If a unicast or multicast frame is not routed, then this frame is switched normally regardless of its TTL value. For multicast frames that are both switched and routed, the frame is still switched within the VLAN it was received from regardless of its TTL value. The routing part is conditional on the TTL value received and the configuration of this field.

7.4.4 Trapping of Frames That Exceed MTU Size

The MTU size is checked for any IP frame. The FM4000 supports up to 8 different MTU sizes, which are stored in the MTU_TABLE. A 3-bit index is defined in the ARP_TABLE (only for arp-glort entries) and in the EGRESS_VID_TABLE. The MTU size defined in arp-glort has precedence over the one defined in EGRESS_VID_TABLE and is normally used for multicast, while the EGRESS_VID_TABLE is normally used for unicast. If the actual packet length (as contained in the IP header) is greater than this MTU size, the packet is declared oversized for this VLAN and the parameter SYS_CFG_1:trapMTUViolations defines the disposition of the frame — either trap to CPU or silently discard.

Every egress VLAN has a default MTU size, which can all be set to 16 K if needed. Any routed frame then overwrites the egress VLAN MTU with whatever is in the ARP Table. This means that all IP frames have their MTU enforced. It also requires that one of the MTU sizes needs to be used for the default value.

In the case of multicast, the arp-glort entry is loaded with an MTU index pointing to the smallest MTU possible on any VLAN on the distribution list for this multicast group. There is no per-VLAN control of MTU checking for multicast groups as the MTU is checked before the replication across VLAN is started.



7.4.5 Logging Redirectable IP Unicast Frames

The switch can detect if a unicast routed packet is headed back to the same VLAN it came from. When this occurs, the switch captures the source IP address and the destination IP address in the ARP_REDIRECT_SIP and ARP_REDIRECT_DIP registers, waits for the end of frame to check if the CRC was correct and, if it was, sets the interrupt ARP_IP::Redirect bit. The contents of the ARP_REDIRECT_SIP and ARP_REDIRECT_DIP registers are frozen from the moment they are written until the software clears the interrupt ARP_IP::Redirect bit or the CRC is proved wrong. Only one redirect event can be captured and processed at a time. Any other redirect event occurring in the same interval of time is discarded.

7.4.6 Logging Routable IP Unicast Frames Received as Layer 2 Multicast

This is the case where there is a broadcast MAC address on an IP unicast frame. The switch detects if a unicast IP frame is received on a routable MAC multicast address, and there is route entry for this frame. The switch performs the normal layer 2 multicast in hardware and logs the frame to the CPU so the software stack can route this unicast frame. The frame is not routed by the hardware in this case, unless the broadcast MAC is added to the SRC_MAC table.

§ §

NOTE: ***This page intentionally left blank.***

8.0 Layer 2 Lookup

8.1 Overview

The layer 2 lookup unit uses the following fields from the frame header to alter the course of the packet.

- Ingress VLAN association / Source MAC Address / Source GORT / Source Port
  - — Used for learning, security checking and ingress VLAN/FID (Forwarding Information Database) filtering.
- Egress VLAN association / Destination MAC Address / Destination GLORT
  - — Used for forwarding and egress VLAN/FID filtering.

The outputs of the layer 2 lookup unit are:

- A default destination mask.
- An updated destination GloRT if the destination GloRT was received as 0 to this unit (it is not changed if the destination GloRT was different than 0 at the entry of this unit).

The outputs rules are as follows:

- If the destination MAC address is unknown, the destination GloRT is set to the default Flood GloRT (MA_TABLE_CFG_2::FloodGlort). If the destination MAC was the broadcast MAC address (all 1s), or the MAC address matches in the table and resolves to a destination mask (instead of a GloRT), it is set to the XcastGlort(MA_TABLE_CFG_2::XcastGlort).
- If the destination MAC address is known and the entry is of type MAC-DMASK, the destination mask is first set to this value, and the destination GloRT is updated to the default broadcast GloRT (MAC_CFG_2:XcastGlort) if it was received as 0 (and unchanged otherwise).
- If there was a hit and the entry is of type MAC-DGLORT, the destination mask is first set to all 1s and the destination GloRT is updated to the value of this entry if it was received as 0.
- The destination mask is then updated according to ingress and egress VLAN membership as well as ingress and egress spanning tree states.

The layer 2 lookup handling involves the following table structures:

- **MAC Address Table** (MA_TABLE) — Maintains MAC address port (GloRT) associations, either by VLAN or by classes of VLANs. Table entries are either learned dynamically or entered explicitly and locked into the table.
- **Ingress VLAN Table** (INGRESS_VID_TABLE) — Contains security and configuration information associated with the VLAN on which a frame arrives.

- **Egress VLAN Table** (EGRESS_VID_TABLE) — Contains security and configuration information associated with the VLAN on which the frame will be sent.
- **Ingress Spanning Tree Table (Forwarding Information Database)** (INGRESS_FID_TABLE) — Maintains the ingress-related spanning tree state of the spanning tree (FID) to which the frame belongs on ingress.
- **Egress Spanning Tree Table** (EGRESS_FID_TABLE) — Maintains the egress-related spanning tree state of the spanning tree (FID) to which the frame belongs on egress.
- **Tagging Table** (VLANTAG_TABLE) — Determines whether frames egressing on a particular (Port, VLAN) should be 802.1Q-tagged.
- **MA Table Change Notification FIFO** (MA_TCN_FIFO) — Notifies software of hardware-initiated changes to the MAC Address Table. Allows software to stay up-to-date with the state of the MA Table by periodically issuing efficient block reads from this FIFO.

The following features are supported by the layer 2 unit.

- Flooding packets when address are not known.
- Automatic learning of unknown source addresses (enabled on a per port).
  - The switch does not learn broadcast, multicast or address 0 if they are used as a source address.
- Automatic aging.
  - A background aging process invalidates learned MAC addresses from the MA_TABLE that has not been heard from in some configured aging timeout period.
  - Additionally, an accelerated MAC Table purge feature can be enabled by software to immediately eliminate all learned entries from a specified source GloRT and/or FID.
- Posting of new addresses learned or aged to host processor.
  - Notifications are enqueued in a 512-entry FIFO.
  - Learning and aging can be suspended as this FIFO becomes full.
- Posting of security violations (unknown addresses or known addresses on new port) to host processor.
  - The switch has the ability to post only one violation interrupt per address. This is accomplished by learning if possible and associate a invalid port with the entry, thus preventing any packet to go to this address until the host figures what to do with the address.
- Support of up to 4096 spanning trees with states disabled, listening, learning, forwarding per port.
- Up to 4096 VLANs.
- Arbitrary mapping of set of VLANs into spanning trees.
- Output tagging control (enabled/disabled).
  - This table is actually located in the EPL.
- Optional trapping of known addresses (BPDU, LACP, 802.1X).
- Optional trapping of IGMP frames.
- Configurable per VLAN per port membership.
- Configurable per VLAN per port tagging.

8.2 VLANs

The FM4000 supports 802.1q Virtual LANs (VLAN). A Virtual LAN is used to partition the resources of the switch into multiple smaller switches. This separates the issue of how much switching capacity is needed in a network, from the needs of the individual user groups that make up the network. Computers on different VLANs can only communicate with each other over layer 3 protocols, such as IP.

The FM4000 supports the IEEE 802.1q VLAN specification, which generalizes the notion of VLANs by adding a VLAN tag to the frame header. Once frames are tagged, they can traverse many switches with the same VLAN association.

The FM4000 supports the following VLAN association and security capabilities.

- Each port has a default VLAN ID and default priority.
- Per port VLAN association. Ingress rule is one of the following:
  - Untagged packets received on one port are associated with the default VLAN ID and have the default priority configured for this port.
  - For tagged packets, each port can be configured to either: (1) leave the VLAN ID and VLAN priority as is, or (2) overwrite the VLAN ID and VLAN priority fields with the port's default values.
  - If there is more than one VLAN tag, the contents of the inner tags are ignored, and only the content of the outer tag is taken care of (Q-in-Q). However, the switch has the capability to step over multiple levels of VLAN tags and start packet decoding after VLAN tags.
  - The FFU can be programmed to change the VLAN association for any type of packet.
- Per port VLAN “security settings”, which can be set to any of the following:
  - Discard all untagged packets. Ingress rule.
  - Discard all tagged packets. Ingress rule. This feature is useful but not required in IEEE 802.1q.
  - Discard packets if the Ingress port was not part of the member list of that VLAN ID per the INGRESS_VID_TABLE. (VLAN ingress boundary violation).
  - Discard packets if the Egress port was not part of the member list of that VLAN ID per the EGRESS_VID_TABLE. (VLAN egress boundary violation).
  - Further discard rules can be applied using the FFU.

The switch is capable to support up to 3 different VLAN Ethernet types. One of the them is the IEEE reserved VLAN tag (0x8100) and is not configurable. The other 2 are configurable per port. The outer VLAN tag is used, the inner VLAN tags are stepped over.

The format of the VLAN tag is:

- VLAN Tag Protocol identifier, TPID (2 octets)
  - 0x8100
  - Custom VLAN Ethernet Type “A”
  - Custom VLAN Ethernet Type “B”

- User Priority
  - Octet 1, bits 7:5
  - The priority field represents a number 0-7. See IEEE 802.1D for details
- CFI/DEI
  - Octet 1, bits 4
  - FM4000 extends the priority field from 3-bits to 4-bits to include CFI/DEI.
- VID
  - Octet 1, bits 3:0 (top 4 bits), Octet 2 (bottom 8 bits)
  - Encodes the VLAN ID,

    0x000 = The null VLAN ID, indicates the tag header contains only user priority

    0x001-0xFFE = VLAN associated with the packet

    0xFFF = Is not a valid VLAN.

FM4000 maintains the following configuration information for each VLAN:

- **Membership List** — A bit per port indicating whether each port is a member of the VLAN.
- **Spanning Tree Number** — A 12-bit number (FID) identifying the spanning tree in use for the VLAN. If all VLANs are configured with the same FID, Shared VLAN Learning is in effect. If each VLAN has a unique FID, Independent VLAN Learning is in effect. Hybrid SVL/IVL configurations are also supported.\
- **Spanning Tree State** — One of four states, selected by software: Disabled, Listening, Learning, or Forwarding.
- **VLAN Counter Index** — Identifies the Group 11 and 12 counter index to increment when frames arrive on this VLAN.
- **VLAN Trigger ID** — Associates an ID with frames arriving on this VLAN for use in trigger matching rules.

The VLAN information is divided into different tables, listed here:

INGRESS_VID_TABLE

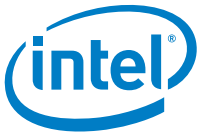
- Port membership (25 x 1 bit)
- Reflect (1 bit) — The reflect usage is described in Section 9.2.
- Counter index (6 bits)
- Spanning tree number (FID) (12 bits)
- Trig id (8 bits)
- Trap IGMP (1 bit)

EGRESS_VID_TABLE

- Port membership (25 x 1 bit)
- Spanning tree number (FID) (12 bits)
- MTU Size Index (3 bits) — For information on MTU size checking, see Section 7.4.4.

INGRESS_FID_TABLE

- Per-port spanning tree state (25 x 2 bits)



EGRESS_FID_TABLE

- Per-Port forwarding state (25 x 1 bit)

VLANTAG_TABLE

- Tag bit (25 x 4096 x 1 bit) — Indicates whether outgoing frames on each (VLAN, Port) pair should be VLAN-tagged.
- This table is implemented in a distributed manner. Each Ethernet Port Logic unit contains a single 4096x1 table.

8.3 MAC Address Table

The MAC Address Table (abbreviated MA Table or MA_TABLE) associates layer 2 MAC addresses with the destination port(s) to which frames addressed to those addresses should be forwarded. The MA Table supports a total of 16,384 entries. It is implemented as a hash table consisting of eight sets of 2048 entries. Entries are classified by FID, allowing for Independent VLAN Learning, Shared VLAN Learning, or any hybrid configuration. Destination ports are specified either by GloRT or by a fixed destination mask. Table entries are either learned dynamically, based on the hardware's observation of source addresses on ingress frames, or entered explicitly and locked into the table.

Each 96-bit MA Table entry includes the following fields:

- **MAC Address** (48 bits)
- **FID** (12 bits)
- **Entry state** (2 bits)

  00b = Not used

  01b = Old (deleted on next aging update)

  10b = New (changed to old on next aging update unless used again)

  11b = Locked
- **Trigger ID** (6 bits)
  - An ID associated with the entry for use in trigger matching rules.
- **Destination type** (1bit)
  - Defines if the destination is a bit mask or a destination GloRT.
- **Destination Mask** (25 bits) **or GloRT** (16 bits)
  - Bit mask for ports associated with this address. One hot encoding for unicast traffic.
  - GloRT (most common)

8.4 Layer 2 Lookup Flow

The overall flow of the layer 2 lookup is shown in Figure 8-1.

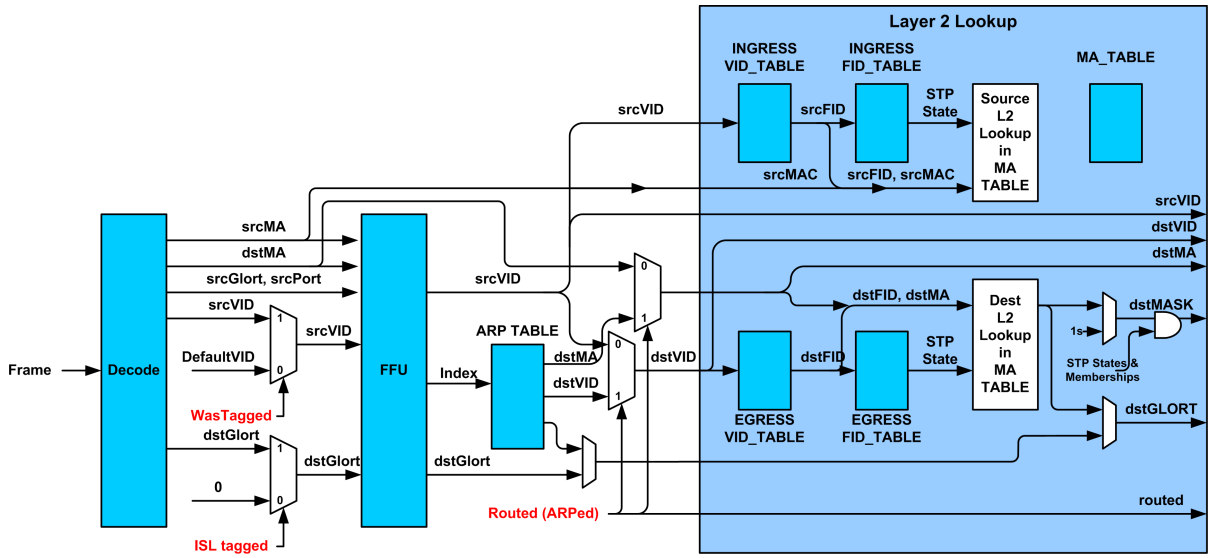
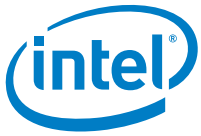


Figure 8-1 Layer 2 Lookup Flow Diagram

The lookup is performed in the following steps:

- The decoder retrieves the VLAN ID (VID) from the packet.
  - The decoder only uses the outer VLAN tag and skips over any inner VLAN tag. The decoder is located in the EPL and can be configured to recognize up to 3 different VLAN tags.
  - A default VID, defined in PORT_CFG_1, is associated with the packet if the frame has no VLAN tag. The PORT_CFG_1 register also includes a flag ("useDefaultVLAN") that, if set, forces the VID to the default VID. The flag "WasTagged" is only set if "useDefaultVLAN" is cleared and the packet is effectively tagged.
- The received VID (srcVID) is then presented to the FFU, which may use it, along with any other information, to associate a new VID. The srcMAC and destMAC is also presented to the FFU but cannot be modified by it.
- If the packet is routed, the ARP table returns a new destination VID and may return a new destination MAC address or a destination GloRT. If the packet is not routed, the destination VID is equal to the srcVID retrieved from the FFU, and the destination MAC address is left unchanged.
  - The ARP table or FFU route-GloRT action may return a destination GloRT, which takes precedence over the destination GloRT received.
- The source VID is used to index the INGRESS_VID_TABLE to retrieve the source FID (spanning tree number) and the port membership. If the receive port is not a member of this VLAN, the packet is dropped. The source FID is then used to index the INGRESS_FID_TABLE to retrieve the spanning tree state of the source port
  - If the spanning tree state is DISABLE, the packet is dropped.



  - If the spanning tree state is LISTENING, the packet is dropped.
    - Some packets (such as BPDU, LAG, 802.1x) can be trapped even when the port is in listening mode. Section 2.6 details the event priority.
  - If the spanning tree state is LEARNING, the packet is dropped but the source address/source FID pair are looked up in the MA table and are learned, if learning is active and the entry is not present.
    - As is the case for LISTENING, some packets are trapped even when the port is in learning mode.
  - If the spanning tree state is FORWARDING, the source MAC address/source FID pair goes through the MA lookup and is learned, if learning is active and the entry is not present.
- The destination VID is used to index the EGRESS_VID_TABLE to retrieve a destination FID and the port membership for this VID. The destination FID is then used to index the EGRESS_FID_TABLE to retrieve the forwarding state of all possible destination ports.
  - The MA is searched for the destination FID and the destination MAC, if there is a hit, then the associated entry is used. If there is a miss, the broadcast GloRT (MA_TABLE_CFG_2::XcastGlort) is used if the packet is a broadcast packet. Otherwise, the flood GloRT is used.
  - A destination mask is derived form the port membership and the forwarding state, where each bit represents one port, and is set only if the port is a member of the destination VLAN and the destination FID is in forwarding mode. This destination mask is then ANDed with the destination mask retrieved from the MA table if there is a hit and the entry contains a mask. Otherwise, this destination mask is fed to the port mapper.

8.5 MA Table Lookup, Learning and Aging

The MA_TABLE is implemented as a 2048 x 8 hash table. The lookup is done by computing an 11-bit index hash from a 60-bit number constructed by concatenating the 48-bit MAC address to the 12-bit FID and reading 8 entries and comparing each one of them with the MAC / FID pair. If any one of the 8 entries match the MAC / FID pair, this is a hit. Otherwise, it is a miss.

Two lookups are done per frame. One for destination MAC address (and associated egress 12-FID derived from VLAN) and one for source MAC address (and associated ingress 12-FID derived from VLAN). The destination lookup is strictly done for all frames, while the source lookup may be relaxed to reduce power consumption. The switch supports 3 modes for source lookup (defined in PORT_CFG_3::SALookupMode):

- **Strict** — Done on all frames, typically required when security is enabled on that port.
- **Relax** — Do lookup only when time permits.
- **Rated** — Do lookup following a certain rate. The rate is configured is MA_TABLE_CFG1 register.

Each source lookup follows the following rules:

- If there is a hit for a source lookup and the address was learned dynamically and the port is the same, the entry timer is refreshed.
- If there is a hit for a source lookup and the address was learned dynamically, and the source GloRT is different, and learning is enabled and the security is not enabled for known address moving to new location, the switch refreshes the entry with the new source GloRT (canonical source GloRT for port members of a LAG). The current trigger id is not changed.

- If there is a miss for a source lookup and learning is enabled, and security is not enabled for new addresses, the switch picks a free entry to store the new entry. The new entry is stored as a dynamic entry, the GloRT is set to the source GloRT (canonical source GloRT for port members of a LAG) and the trigger id is set to the default trigger id configured in MA_TABLE_CFG_1::TrigIdDefault. If there are no free entries, the switch does not learn the new address.

  Note: Hardware first picks a free entry that happens to match the MAC/VID that was previously used (but is now free) and picks the highest index if none match.

- Learning is not done if the source address is a broadcast address, a multicast address or a null address (all 0s), but source lookup is still performed on those addresses.
- Learned entries are always of type GloRT. Entries of type “destinationMask” can only be loaded by the software.

The source GloRT is parsed through the CANONICAL_GLORT_CAM to detect if it is a member of a LAG. If the source GloRT is part of a LAG, the canonical source GloRT for that LAG is used as a source GloRT before storing the address in the MA table. This ensures that future packets coming from different ports in the LAG are all assumed to come from the same 'virtual' port.

Dynamic entries are aged by the switch when aging is enabled. All entries are checked periodically so that 'new' entries are marked 'old' and all 'old' entries are cleared. Static addresses (only the software may load static addresses) are never aged.

Note: The switch supports address learning and aging either automatically or via software intervention. The method is independently set for aging and learning via MA_TABLE_CFG_1::softAging and MA_TABLE_CFG_1::softLearning respectively. If software aging or learning is enabled, the hardware posts the event in the TCN FIFO, but does not perform the actual deletion or addition, or move into the MA table automatically. However, refreshing the age is still done.

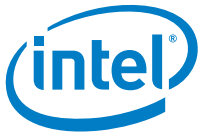
When the switch learns a new MAC address, it selects the highest free set available in the appropriate bin. To minimize the possibility of overwriting entries that are learned after software selects a new locked entry's set number, it is recommended that software populate the bins upward starting from set zero.

8.6 MA Table Management

8.6.1 Direct Table Access

The MAC Table's 16,384 entries are visible to software and can be read and written as atomic 3-word registers. The MAC Table is implemented as 2,048-bin hash table, with each bin consisting of eight sets. Software must directly write locked entries into the table through this interface, taking care to allocate entries to the appropriate hashed index. The MA_TABLE addressing is defined as follows:

Field	Bit(s)	Description
Set	15:13	Identifies one of eight sets in the entry's hash table bin.
Index	12:2	Indexes one of 2048 hash table bins.
Word	1:0	Identifies one of three words in the 96-bit entry. Word number 3 is reserved.



Entries are hashed based on the entry's FID value and MAC address. Given this 60-bit key, a folded Ethernet CRC calculation is performed to provide a 16-bit hash value. One of four bitwise rotations of this value is selected to produce a 12-bit bin index. The rotation used is statically configured for all entries in MA_TABLE_CFG_1. More specifically, the key is interpreted as an eight byte sequence:

bytes[0]	bytes[1]	—	bytes[5]	bytes[6]	bytes[7]
MAC[47:40]	MAC[39:32]		MAC[7:0]	FID[9:8]	FID[7:0]

From this, the sequence's 32-bit CRC value is XOR-folded to give

```
HASH[7:0] = CRC[7:0] ^ CRC[15:8]
HASH[15:8] = CRC[23:16] ^ CRC[31:24]
```

8.6.2 MAC Table Hardware-Accelerated Purging

During port failover events, the direct management interface to the MA Table may be too slow for software to eliminate all stale entries in an acceptable period of time. To accelerate this operation, the switch allows the user to issue a command to automatically purge all entries in the MAC table that match a particular FID and/or GloRT, or unconditionally purge all entries in the table.

The command is issued by writing into the MA_PURGE register. Once complete, a “Purge Complete” notification entry is posted into the TCN FIFO, allowing software to clearly distinguish the entries learned before the purge from those learned after the purge.

To ensure coherency, learning and aging is disabled at the beginning of the command and resumed once the “Purge Complete” notification is posted into the TCN FIFO. The purge command checks one entry of the MA Table every 20 frame handler cycles, allowing frame lookups to still be performed at wire speed while the purge command is in effect. The time to do a complete purge is as follows:

16,384 x 20 / FH_CLOCK (0.87 ms at 375 MHz).

8.6.3 MAC Table Change Notification FIFO

The hardware uses the MAC Table Change Notification FIFO (MA_TCN_FIFO) to communicate MA Table events to software. Each entry in the FIFO contains the reason for the notification and a copy of the changed entry. Some entries also include the actual index and set that was affected in the change.

The MA_TCN_FIFO has a capacity of 512 change notifications.

Relevant registers:

- **MA_TCN_FIFO[0..511,0..3]**
  - **TableEntry** (96 bits) — MAC address lookup entry as encoded in the MA_TABLE. The table entry reported has exactly the same format as the MA_TABLE entries.
  - **Index** (12 bits) — Index of associated MA_TABLE entry.
  - **Set** (3 bits) — Set of associated MA_TABLE entry.
  - **EventType** (3 bits) — Identifies the type of change.

- **MA_TCN_PTR**
  - **Head** (9 bits) — Entry in the MA_TCN_FIFO that software will read next. Updated by software. Read by hardware to determine when the FIFO overflows.
  - **Tail** (9 bits) — Entry in the MA_TCN_FIFO that hardware will write to next. Updated by hardware. RO by software.

The MAC Address Table Change Notification FIFO (MA_TCN_FIFO) is a 512 x 128-bit memory with head and tail pointers exposed to software. When the MAC Table changes in any manner, an event recording that change is written to the tail pointer location in the FIFO, and the tail pointer is incremented. Software is responsible for advancing the head pointer after it has read enqueued events.

When MA_TCN_PTR.Head == MA_TCN_PTR.Tail, the FIFO is empty.

When (MA_TCN_PTR.Tail+1)%512 == MA_TCN_PTR.Head, the FIFO is full.

Note: This simple full condition limits the usable capacity of the MA_TCN_FIFO to 511 entries. In the full condition, if hardware needs to add a 512th event to the FIFO, it instead drops the notification event and raises the Overflow interrupt bit in MA_TCN_IP.

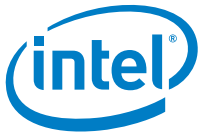
The TCN event types are listed in Table 8-1.

Table 8-1 TCN FIFO Event Type

EventType	Encoding	Description
Learned	0	Entry was learned into the MA_TABLE as a result of a SMAC lookup miss.
Aged	1	The unlocked entry was aged out of the table.
BinFull	2	Entry could not be learned into the MA_TABLE due to an unavailability of free sets in the hashed bin (index).
ParityError	3	A parity error was detected in this entry.
SecurityViolation_New	4	A frame received with this with an unknown SMAC and FID has caused a security violation.
SecurityViolation_Moved	5	A frame received with this with a known SMAC and FID has caused a security violation
PurgeComplete	6	Indicates completion of an MA Table purge operation.

Only one event notification is enqueued in the FIFO per frame. Since two lookups are performed per frame, and not all of the above events are mutually exclusive, the highest severity event is enqueued according to the following precedence ordering:

1. Purge command (Highest)
2. ParityError Destination
3. ParityError Source
4. ParityError Mgmt
5. Learning/Aging.
6. Security Violation
7. Bin Full. (Lowest)



The Learned and Aged event notifications occur exclusively relative to the other event types. The TCN stores the updated entry for learning and aging events so software knows what is currently in the table. Software should already know what was in the table. When aging the macAddress, FID and destMask/destGlort/destType fields all stay the same. When purging, the MA_TABLE entry is written to all 0s.

For SecurityViolation, the TCN FIFO entry contains the following elements:

- TCN FIFO Type: 4 (security violation, new address) or 5 (security violation, moved address)
- Index/Set:
  - Index is valid for both type of events.
  - Set is undefined for a new address while equal to existing entry for an existing address
- ParityError: 0
- Table Entry:
  - MAC Address: Source MAC address that caused the violations
  - FID: The ingress FID
  - State: 0
  - ParityError: 0
  - DestGlort: canonicalized source GloRT
  - DestType: 1 (GloRT)
  - TrigId: 0

For BinFull, the TCN FIFO entry will contain the following elements:

- TCN FIFO Type: 2
- Index/Set: Index is valid, set is undefined.
- ParityError: 0
- Table Entry:
  - MAC Address: Source MAC address that caused the bin full
  - FID: The ingress FID
  - State: 0
  - ParityError: 0
  - DestGlort: canonicalized source GloRT
  - DestType: 1 (GloRT)
  - TrigId: 0

8.6.4 MA_TCN_FIFO Overflow Protection

To protect the CPU from losing table change notifications during periods of excessive learning and/or aging events (perhaps due to malicious denial-of-service flooding), the TCN FIFO supports configurable learning and aging thresholds, which when exceeded temporarily disable learning or aging. The learning and aging mechanisms are re-enabled only after the CPU reads and de-queues enough pending notifications from the FIFO, dropping the number of notification events below the thresholds. The thresholds are set in MA_TCN_CFG_2 as follows:

- **MA_TCN_CFG_2**
  - **LearnedEventsThreshold** — Maximum number events allowed in the FIFO of any type before stopping adding Learning events.
  - **AgedEventsThreshold** — Maximum number of events allowed in the FIFO of any type before stopping adding Aging events.
  - **ErrorEventsLimit** — Maximum number of events allowed in the FIFO of either BinFull or ParityError or Security types before stopping adding those type of events.
  - **FlowControlEnable** — When set to 1, learning and aging will be disabled (flow-controlled) when their thresholds are exceeded.

The Threshold and Limit parameters specify limits on the number of events of these types to allow in the FIFO. In general, when these limits are exceeded, the associated events are not enqueued in the MA_TCN_FIFO, and whenever one is dropped a corresponding overflow interrupt is posted in MA_TCN_IP.

The LearnedEventsThreshold and AgedEventsThreshold limits imply additional behavior unique to these event types. Specifically, when the number of Learned or Aged events in the MA_TCN_FIFO equals their respective limit value, and FlowControlEnable is set to 1, MAC Table learning or aging is temporarily disabled. Once software advances its MA_TCN_PTR.Head pointer by a sufficient amount, reducing the FIFO level below the relevant threshold, the halted learning/aging mechanism resumes.

Note: If FlowControlEnable is 1, setting LearnedEventsThreshold or MaxAgedEventsThreshold to zero has the effect of permanently disabling learning or aging.

The TCN_FIFO entries of type “purge completed” (posted upon completion of the purge command) are not checked against the thresholds. The entry is posted if there is at least one entry free in the TCN_FIFO regardless of the MA_TCN_CFG_2 setting. It is expected that the software issues a purge command and waits to retrieve the “purge completed” from the TCN FIFO before issuing any similar command again, Therefore, the proper approach is to set the thresholds to ensure that at least one entry remains free.

8.6.5 MA_TCN_FIFO Interrupts

Relevant registers are:

- **MA_TCN_CFG_1** — InterruptThreshold, InterruptTimeout — Configuration controlling when the ExceedThreshold interrupt bit becomes active.
- **MA_TCN_IP** — Interrupt bits for each interrupt source.
- **MA_TCN_IM** — Mask bits for each interrupt source.

The following sources of MA_TCN_FIFO interrupts are defined in Table 8-2:

Table 8-2 Sources of MA_TCN_FIFO Interrupts

Interrupt	Description
PendingEvents	Set whenever MA_TCN_PTR.Tail is advanced and (512 + Tail(new) - Head) % 512 > InterruptThreshold, or the time since the last enqueued event exceeds InterruptTimeout.
LearnedOverflow AgedOverflow ParityErrorOverflow BinFullOverflow SecurityViolationOverflow	Indicates that an event of the corresponding type could not be enqueued to the MA_TCN_FIFO for any reason. This will usually be caused by exceeding the LearnedEventsThreshold, AgedEventsThreshold, or ErrorEventsLimit levels, but may also occur due to total FIFO overflow. In the case of Learned and Aged event types, when FlowControlEnable is set, an overflow interrupt also indicates that a MAC address entry was not learned into or aged out of the MA_TABLE, when otherwise it would have been.

The MA_TCN bit in the global FH_INT_DETECT register is set and the interrupt propagates to the CPU whenever a bit is set in MA_TCN_IP and has a zero in the corresponding bit in MA_TCN_IM. Hardware only sets bits from 0b to 1b in MA_TCN_IP. Software must explicitly write to clear the MA_TCN_IP bits, even if the condition that originally caused the interrupt (e.g. Overflow) no longer applies.

8.7 MAC Address Security

MAC address security is a Layer 2 feature in which the switch treats new MAC addresses in a more restricted fashion than the normal policy of learning and flooding. It is not related to VLANs (802.1q) or port access control (802.1x), but it can be used together with those features in numerous system level layer 2 security applications. Security is fundamentally a per-port concept, and violations are checked per port.

There are two meaningful MAC address security checks.

1. Is the Source MAC address in the table?
2. If the Source MAC address is in the table, is the Source MAC address on the correct port?

Unknown MAs or known MAs on the wrong port are considered violations when the security feature is enabled. Two bits are used per port to either enable any of these two conditions to generate a security violation: new unknown mac address, known mac address in new location.

When a frame meets the criteria to be considered a security violation the following actions are performed:

- The frame is flagged as a security violation. There are two possible marking:
  - — New address security violation
  - — Moved address security violation
- The frame can be dropped or trapped using triggers that may trig on either new or moved violations. If not triggers are defined, the frame is dropped by default.
- The frame is counted as a security violation regardless of the disposition of the frame.
- An entry is pushed in the TCN FIFO as a type SECURITY VIOLATION (conditional to FIFO limits). The entry format must be conformed to the entry format in MA table and must be of type GloRT, and must be stored with the canonical source GloRT from which the frame comes from.

- If learning is on, the address that caused the violation is learned anyway, thus preventing the violation to occurs repetitively in the FIFO. If learning is off and the frames causing the security violation keep coming in, each frame causes a FIFO entry to be used until the limit allowed for security violation is reached.

8.8 Layer 2 Protocol Traps

The layer 2 lookup has also the ability to trap frames of some special layer 2 packets or IGMP without using any extra lookup or FFU resources. Each trap is separately enabled (see SYS_CONFIG_1 for layer 2 traps and VID_TABLE for IGMP) and are:

- **BDPU** (Spanning Tree) — Destination address is 0x0180C2000000.
- **LACP** — Link Aggregation Control Protocol: Destination address is 0x0180C2000002.
- **Port Authentication** — Destination address is 0x0180C2000003.
- **GARP** — Both GMRP and GVRP: Destination address is 0x0180C2000020 or 0x0180C2000021.
- **All other IEEE** — Destination address is 0x0180C20000xy. Where.

  ```
  x=0 & y > 3, x=1, or x=2 & y > 1.
  ```

- **IGMP** (per VLAN) — IP frame with protocol equal to 2.
- **CPU port**

When a frame is trapped, it is sent to the CPU GloRT (defined in CPU_GLORT register) instead of being treated as a general multicast address. The switch is capable to re-assign a new switch priority in this case. Any other special protocol or addresses may be trapped using the MA_TABLE triggers or FFU resources.

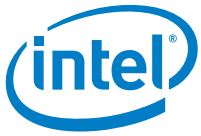
8.9 IEEE 802.1x – Port Access Control

IEEE 802.1x defines “port-based network access control.” The protocol allows port-based network access control that makes use of the physical access characteristics of IEEE 802 LAN infrastructure to provide a means of authenticating and authorizing devices attached to a LAN port that has point-to-point access characteristics, and of preventing access to that port in cases in which the authentication and authorization fail.

There are 3 significant system roles in 802.1x

- **Supplicant** — The side of the Ethernet link which would like access to the services behind the authenticator.
- **Authenticator**— The side the of the Ethernet link that is authenticating and authorizing the supplicant.
- **Authentication Server**— The entity that provides authentication services to the authenticator, and may or may not be co-located with the authenticator.

In standard Ethernet both sides of a link are called peers. In 802.1x they are no longer peers, they are supplicant and authenticator. Either side of a link can take on either role, and in some cases there is mutual authentication.



The supplicant has a PAE (port access entity) and the Authenticator has a PAE. The PAEs control the authentication state of the port. EAPOL (extensible authentication protocol over LAN) messages are sent from the supplicant PAE to the authenticator PAE and from the authenticator PAE to the supplicant PAE. The authenticator PAE may also communicate with an authentication server using EAP (extensible access protocol) messages.

FM4000 can be enabled to trap EAPOL messages automatically and send them to a host processor for further processing. An EAPOL is detected if the destination address is 01-80-C2-00-00-03 (IEEE reserved group address for 802.1x).

FM4000 supports also the following authentication port states:

- **Authorized** — Non-EAPOL frames may be transmitted and received from the port.
- **Non-Authorized-IN** — Frames may be transmitted from the port, however the port is non-authorized for receive, and RX frames are discarded, unless they are EAPOL frames. In this case, the port MAC security is on, learning is off and security trapping is off. No MAC addresses are in the table associated with that port, so all non-trapped packets are discarded. Non dot1x trapped packets are also discarded.

  Note: Currently, other traps are system settings, so it is TBD whether they should be changed to be port settings or they should be discarded in software.

- **Non-Authorized-BOTH** — The port is non-authorized for both transmit and receive, and all frames but EAPOL frames are discarded. The source mask for every port is set so that frames cannot be forwarded to the unauthorized port. The PAE is implemented in-part in the CPU. The CPU can send the PAE's EAPOL packets through this source mask barrier by using the direct mode of LCI transmission, that is in LCI_CFG set Tx Switch Mode to 1.

§ §

NOTE: ***This page intentionally left blank.***

9.0 Port Mapping and Packet Replication

9.1 Overview of Port Mapping Unit

The GloRT lookup is shown in Figure 9-1. The Port Mapping Unit uses the destination GloRT from earlier stages in the pipeline to retrieve the set of destinations for the frame. This set of destination ports is encoded in a destination mask which gets ANDed with a default destination mask generated by the layer 2 lookup module.

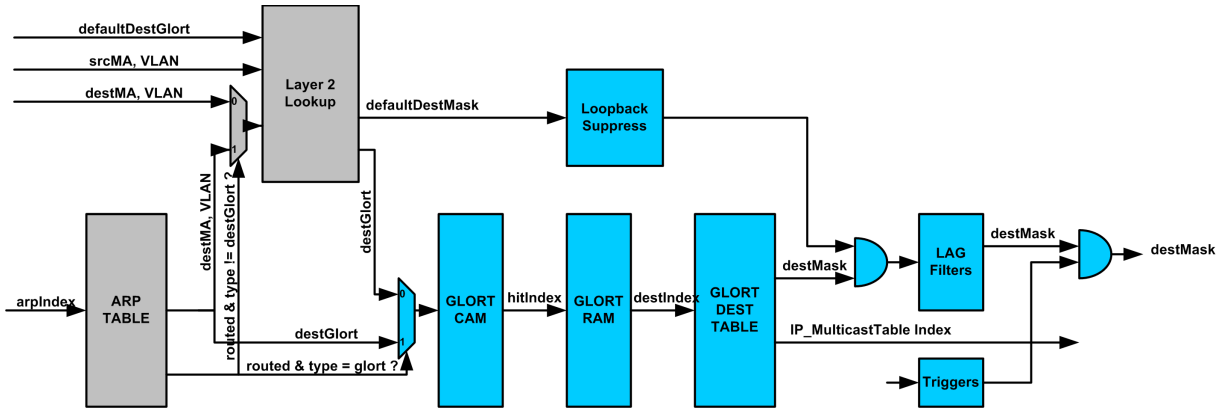


Figure 9-1 GloRT Mapping Unit Block Diagram

The port mapping requires both a destination GloRT and a default destination mask as inputs, and produces an updated destination mask as well as an IP multicast table index for packet replication.

The destination GloRT may come from 3 different sources:

- The Intel ISL tag
  - — A packet that is not tagged will get a default destination GloRT of 0.
- The ARP table
- The MA table

The input destination mask may only come from the layer 2 lookup unit, and is derived from the destination address as well as ingress/egress VLAN membership and spanning tree state.

The destination GloRT is searched into a 256-entry 16-bit full ternary CAM. The GLORT RAM is 256 x 40 bits, and there is a one-to-one correspondence between the GLORT CAM and GLORT RAM. If the search is successful, the highest hit line is selected and the associated GLORT RAM is read. If the destGlort is not found in the CAM, the frame is dropped. A single CAM entry may cover a very large set of contiguous destination GloRTs.

The GLORT RAM contains the following information:

- **ParityError** (1 bit) — Indicates the parity correctness status of each entry.
- **Strict** (2 bits) — Controls the method to compute the index into the Destination Mask Table, and if the destination mask retrieved from this table, is used strictly or ANDed with the destination mask from the layer 2 table or for LAG filtering.
- **DestIndex** (12 bits) — Base index address in the Destination Mask Table.
- **DestCount** (4 bits) — Number of ports in the LAG to which the GloRT entry belongs. A value of 0 represents "16".
- **Range Sub Index A & B** (8 bits each) — Controls the mapping from the GloRT value to the Destination Mask entry (see below).
  - Divided into two 4-bit fields: A/B_Length, A/B_Offset
- **HashRotation** (1 bit) — Selects one of two independent hash values for use in the Destination Mask offset calculation.

The content of the GloRT RAM is then used to compute an index in the Destination Mask Table (GLORT_DEST_TABLE, 4K x 40), which contains the following fields:

- **Destination Mask** (25 bits) — ANDed with the default destination mask produced by the layer 2 lookup processing.
- **IP_Multicast Index** (14 o bits) — Index relevant for IP multicast; detailed in the IP Multicast section.

Data associated with each CAM entry. The entry defines how to compute the index for indexing the GLORT_DEST_TABLE to retrieve the final destination.

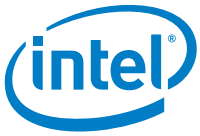
Multiple GloRT values can map to the same GLORT_DEST_TABLE index, so all 64 K possible GloRT values can map into the 4 K table. The Strict field indicates whether the GLORT_DEST_TABLE index is generated deterministically (strict) or by hashing. When strict is used (Strict is 0x3 or it is 0x0 and the ISL tag's FTYPE is either special delivery or management), the index into the GLORT_DEST_TABLE is computed as follows:

```
index = DestIndex + (glort{B}<<width(A)) + glort{A}
```

where:

- **glort{A}** is the value of the bits extracted from the GloRT according to RangeSubIndexA.
- **glort{B}** is the value of the bits extracted from the GloRT according to RangeSubIndexB.
- **width{A}** is the number of bits extracted from the GloRT according to RangeSubIndexA (indicated by bits 22:19 of RangeSubIndexA).

The idea is to provide a single CAM entry to cover multiple LAGs with multiple ports in each LAG where the port numbers are encoded in RangeSubIndexB and the LAG numbers are encoded in RangeSubIndexA. The number of ports in each LAG need not be a power of 2 to make efficient use of the GLORT_DEST_TABLE. If the number of LAGs is not a power of 2, a choice can be made between wasting GLORT_DEST_TABLE entries or consuming additional GLORT_CAM/RAM entries by dividing the LAGs into multiple sets, each set containing a number of LAGs that is a power of 2.



When non-strict is used (Strict is 0x2 or it is 0x0 and the ISL tag's FTYPE is either routed or normal), the index into the GLORT_DEST_TABLE is computed as follows:

```
index = DestIndex + ((hash%DestCount)<<width(A)) + glort{A}
```

where:

- **glort{A}** is the value of the bits extracted from the GloRT according to RangeSubIndexA.
- **width{A}** is the number of bits extracted from the GloRT according to RangeSubIndexA (indicated by bits 22:19 of RangeSubIndexA).
- **hash%DestCount** is a modulo hash over the DestCount number of entries in the GLORT_DEST_TABLE per LAG.

If DestCount is equal to the highest value that would ever be seen encoded by RangeSubIndexB, the difference between strict and non-strict is essentially the difference between hashing over the ports in a LAG and addressing the individual ports specifically (as required by LACP).

The use of sub index A allows for efficient packing of entries in the Destination Mask Table when the entries are shared between different LAGs. The use of the hash function in the index calculation has the effect of balancing traffic across multiple destination masks.

In case of strict routing, the frame hash is not used in the calculation, and the destination GloRT is sufficient to determine where the frame goes. The use of strict routing bypasses any filtering done by layer 2 processing including VLAN ingress and VLAN egress filtering, layer 2 lookup filtering, filtering via global port mask in PORT_CFG_2 and LAG filtering. However, the trigger can still modify the destination mask if active.

9.2 Loopback Suppression

Three tables are used to prevent layer 2 packets from flowing back to the port or the link aggregation they came from. Routed packets are allowed to go back on the port they came from in all circumstances.

The tables are:

- **PORT_CFG_2** (25 x 25 bits)
- **INGRESS_VID_TABLE::Reflect** (4096 x 1 bit)
- **LOOPBACK_SUPPRESS** (25 x 32 bits)
  - The LOOPBACK_SUPPRESS table is instantiated twice in the design as it is used at two different stages. The first set (FH_LOOPBACK_SUPPRESS) is instantiated in the frame handler and is used for unicast packets or multicast packets when they are not replicated across multiple VLANs. The second set (LOOPBACK_SUPPRESS) is used in MTABLE for IP multicast packets that get replicated across multiple VLANs.

The process is the following and only applied for non-routed packets and for non-strict GloRTs:

- The destination mask retrieved from the layer 2 lookup is ANDed with PORT_CFG_2[rxPort].
- The INGRESS_VID_TABLE::Reflect bit is examined. If it is reset, the destination mask bit corresponding to the receive port is cleared. If the reflect bit is set, the destination mask bit corresponding to the receive port is left untouched.

- Then the switch checks every bit of the destination mask. If a bit is found set, the packet is marked to be transmitted to the port indicated by that bit. The source GloRT is then checked to determine if it belongs to the same Link Aggregation Group as the source port. If yes, the bit is cleared.

In the case of IP multicast packets, the loopback suppression is only checked if the ingress VLAN and the egress VLAN are the same.

9.3 Link Aggregation

Link-Aggregation is a means of developing more throughput and redundancy between two switches by aggregating point-to-point links together to form one logical port. This aggregation is transparent to the IEEE MAC. Traffic in the same flow destined for that logical port is sent out one and only one of the physical ports.

In FM4000, all datapath functionality, such as hashing and flood filtering are implemented in hardware. The LACP and Marker frames are trapped and sent to the CPU so that aggregation control process may be implemented in software. The chip's link aggregation operation is fully compatible with IEEE 802.3ad-2000 and IEEE 802.3-2002 clause 43.

Three concepts are used in FM4000 to implement link aggregation:

- Link Aggregation Glorts (Section 9.3.1)
- Filtering (Section 9.3.2)
- Pruning (Section 9.3.2)

These are explained in the next sections.

9.3.1 Link Aggregation Glorts

When ports are aggregated into a LAG, there is a need for two types of global resource addresses:

- **LAG member GloRTs** — These identify the individual physical port members of the LAG. A frame forwarded to such a GloRT egresses from a particular physical port, regardless of the frame's header hashing result.
- **Canonical LAG GloRTs** — These GloRTs identify the trunked port group as a whole, possibly comprising multiple physical ports from different FM4000 devices. A frame identifying this GloRT destination egresses from one of the LAG member ports depending on the frame's header hashing result.

Generally, the CPU sends and receives frames to/from LAG member GloRTs to implement the LACP and Marker protocols, while all other frames are addressed and learned by canonical GloRT. When a frame is sent to the CPU its source GloRT association (configured in PORT_CFG_ISL) must be its LAG member GloRT. This allows the CPU to determine the ingress physical port when such frames are trapped.

However, a frame's source GloRT must be learned into the MAC Address Table by canonical GloRT. Thus a hardware mapping from LAG member GloRT to canonical LAG GloRT must be defined. This mapping function is implemented as a small sixteen-entry CAM, configured in the CANONICAL_GLORT_CAM registers. Figure 9-2 illustrates its operation.

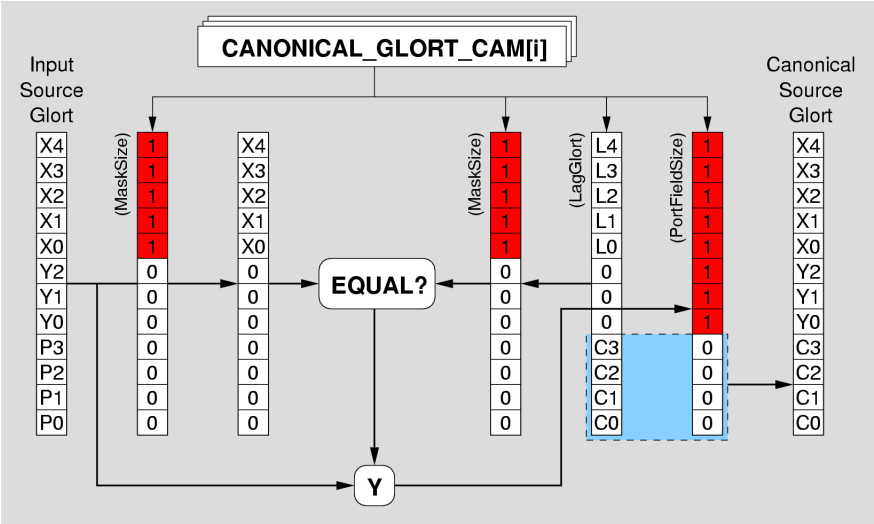


Figure 9-2 Canonical GLORT CAM

In this example, the X0..X4 portion of the GloRT identifies a subset of the overall GloRT space reserved for LAG use. Within that scope, each individual LAG is identified by the Y0..Y2 field. For each LAG, the P0..P3 field identifies both the canonical GloRT (C0..C3) as well as the port member GloRTs (P0..P3 C0..C3).

Note: For a given number of PortFieldSize bits, the maximum number of members per LAG is $2^{PortFieldSize}-1$, since one value must be used for the canonical GloRT value.

If the frame is then forwarded to another switch in the multi-switch system using an ISL tag, the source GloRT must be replaced with the canonical source GloRT so that the next switch does not need to contain the same CANONICAL_GLORT_CAM (the size of this table is tuned for a one chip context and is not a reflection of the entire system capability). If the frame is trapped to the CPU either via special traps or via triggers, the original source GloRT must be left unchanged so that the CPU knowns exactly from which port this frame comes.

9.3.2 Filtering and Pruning

There are two methods to load balance across multiple ports.

- **Filtering** — The destination mask table entry pointed to by the GLORT_RAM includes multiple possible destinations, and a key in LAG_CFG defines which of these ports actually transmits the packet.
- **Pruning** — The GLORT_RAM includes a base pointer, a count and a hash function, which is used to select which destination mask is used.

The filtering method is the recommended method for single switch system while a combination of Pruning and filtering is required for more complex systems.

The exact processing steps performed by the switch are:

1. The hash module computes 2 keys (A and B) from incoming data. One is normally used for balancing traffic across multiple chips, while the other is used to balance traffic across the current chip.

   Note: The chip has different method of computing A and B and thus may load balance traffic differently across multiple stages of switches.

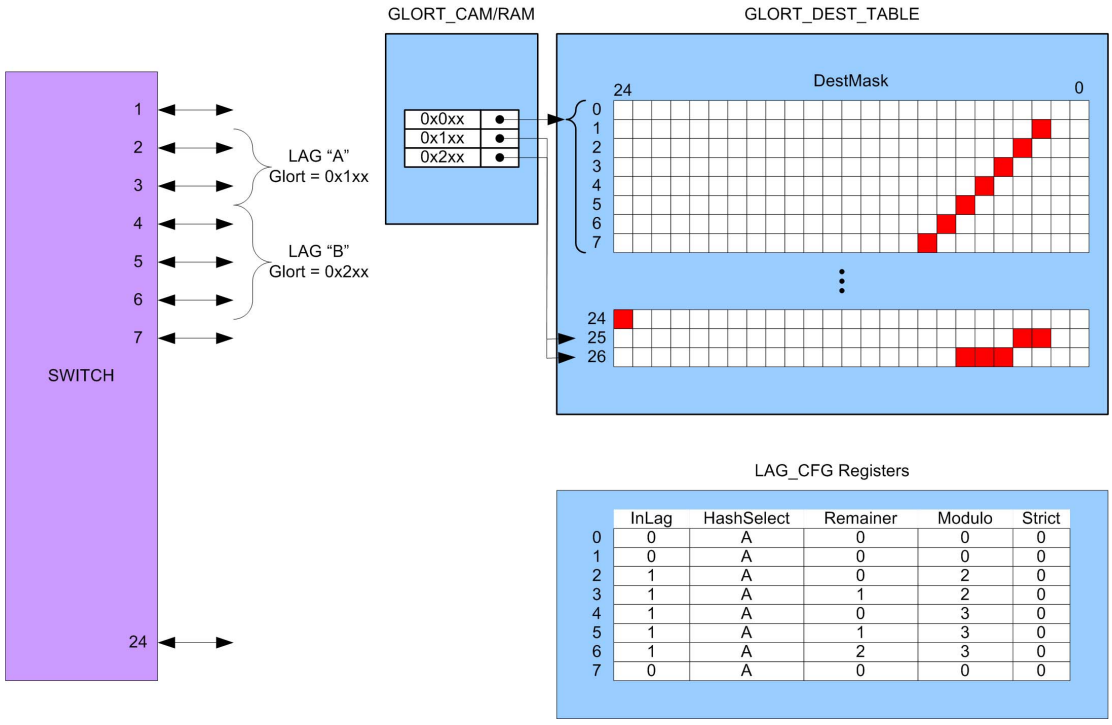
2. The GLORT_RAM defines if a hash function is used, which one (A or B), and which divider is selected (1 to 16). The index is computed from the base index and the remainder of the hash key computed divided by the modulo. This is the pruning step.

3. Then the per-port register set LAG_CFG[0..25] defines the hash function to use (A or B), the divider to use (1 to 16) and the remainder to watch for. The fields for this register are:

   - **HashFunction** (2 bits)
   - **Divider** (4 bits, 0 means 16)
   - **Remainder** (4 bits)
   - **InLag** (1 bit)

9.3.2.1 Example A – LAG within One Switch

In this first example, a single switch supports 2 LAG groups, A and B. The first one has 2 ports (2 and 3), while the second has 3 ports (4, 5 and 6). All other ports are single ports and not members of any LAG.



	InLag	HashSelect	Remainer	Modulo	Strict
0	0	A	0	0	0
1	0	A	0	0	0
2	1	A	0	2	0
3	1	A	1	2	0
4	1	A	0	3	0
5	1	A	1	3	0
6	1	A	2	3	0
7	0	A	0	0	0

Figure 9-3 LAG Filtering in a Single Switch System

The solution is to assign GLORTs 1 through 4 for the individual ports that are not part of a LAG and assign GLORT 0x1XX and 0x2XX for the LAG "A" and the LAG "B" respectively. This is shown in the next table.

Port	Default GLORT
1	1
2	0x101
3	0x102
4	0x201
5	0x202
6	0x203
7	7

For example, when a frame needs to be transmitted to GLORT "0x100", the GLORT_CAM matches 0x1xx and the corresponding GLORT_RAM is read, which points to entry 25 in the GLORT_DEST_TABLE. The destination mask at that location includes port 2 and 3 as possible destinations. The hash key computed from the content of the packet is presented to the lag filtering, which is configured with the same modulo (2) but different remainder (0 or 1), causing half the flows to go to one link while the second half goes to the second link.

9.3.2.2 Example B – LAG within a Two-level Fat Tree

In this second example, five switches are interconnected together to form a multi-switch system. They are all managed from a single point. The system is shown with a single LAG on the external ports that spans two switches. The basic problem to solve is to load balance the traffic from W to both the inner LAG group as well the outer LAG group in such a way that the traffic is well balanced across all links.

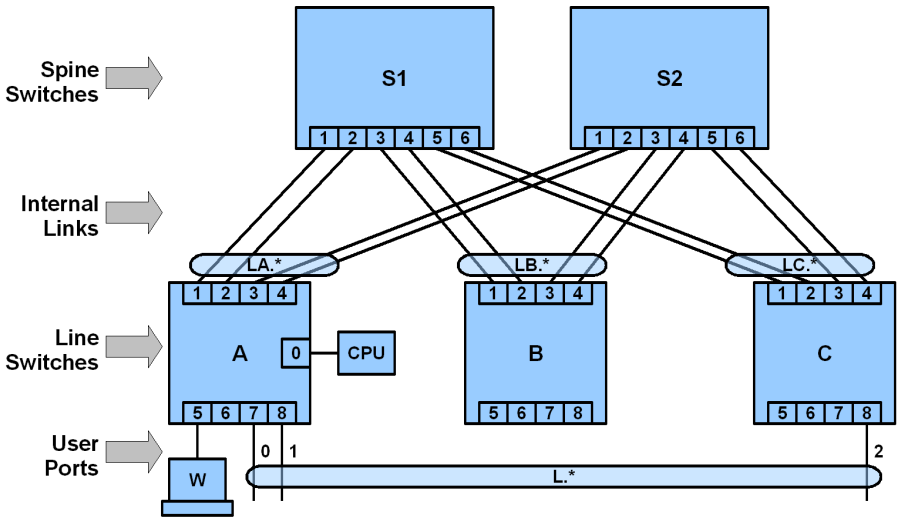


Figure 9-4 LAG Filtering in a Multi Switch System

The following nomenclature is used:

- X.Y identifies a port where "X" is the group/switch it belongs to and "Y" is the port within that group.
- X.* identifies all ports for that switch or group.
- X.0 is the canonical for LAG groups.
- The tuple X.Y can be numerically presented as X * 32 + Y and use directly as a GLORT.

The example above includes the following elements:

- The switch A, B, C are line switches with user ports and internal links.
- The switch S1 and S2 are spine switches and do not include any user ports.
- The LAG L is a link aggregation group from a user group
- The LAGs LA, LB, LC are link aggregation groups used to balance the traffic across the multiple spine chips.
- The workstation W is sending traffic

The requirement is to balance the traffic from workstation W to the different destinations.

The simplest method for implementing this topology would be the following:

- **Switch "A" and "C"** — Two different hash functions are used and configured to generate different keys. The hash function "A" is used to balance toward the spine chip (LAG LA.*) while the hash function "B" is used to balance toward the LAG L.*
- **Switch "A" and "C"** — The GLORT CAM includes the following entries:
  - A.*
  - B.*
  - C.*
  - L.*
  - Broadcast GLORT
- **Switch "A"** — The corresponding GLORT_RAM includes the following entries:
  - A.* points to a block of 25 entries in the GLORT_DEST_MASK using a strict indexing method. Each of the GLORT_DEST_MASK entry points to a specific port.
  - B.* points to a single entry in the GLORT_DEST_MASK which include ports 1,2,3,4 as possible destinations. Any packet going to GLORT B.* are hashed across ports 1,2,3,4 via the LAG filter, which uses a different remainder for each port.
  - C.* points to the same entry as B.*
  - L.* points to a single entry in the GLORT_DEST_MASK which include ports 1,2,3,4,7,8 as possible destinations. Any packet going to GloRT L.* are hashed across ports 1,2,3,4,7,8.
  - Broadcast GLORT points to a single entry in the GLORT_DEST_MASK which includes all ports, including CPU.
- **Switch "C"** — The corresponding GLORT_RAM includes the following entries:
  - A.* points to a single entry in the GLORT_DEST_MASK which includes ports 1,2,3,4 as possible destinations
  - B.* points to the same entry as A.*

— C.* points to a block of 25 entries in the GLORT_DEST_MASK using a strict indexing method. Each entry of the GLORT_DEST_MASK points to a specific port.

— L.* points to a single entry in the GLORT_DEST_MASK which includes ports 1,2,3,4,8 as possible destinations

— Broadcast GLORT points to a single entry in the GLORT_DEST_MASK which includes all ports except CPU.

The configuration is illustrated in Figure 9-5 (only pertinent ports are shown).

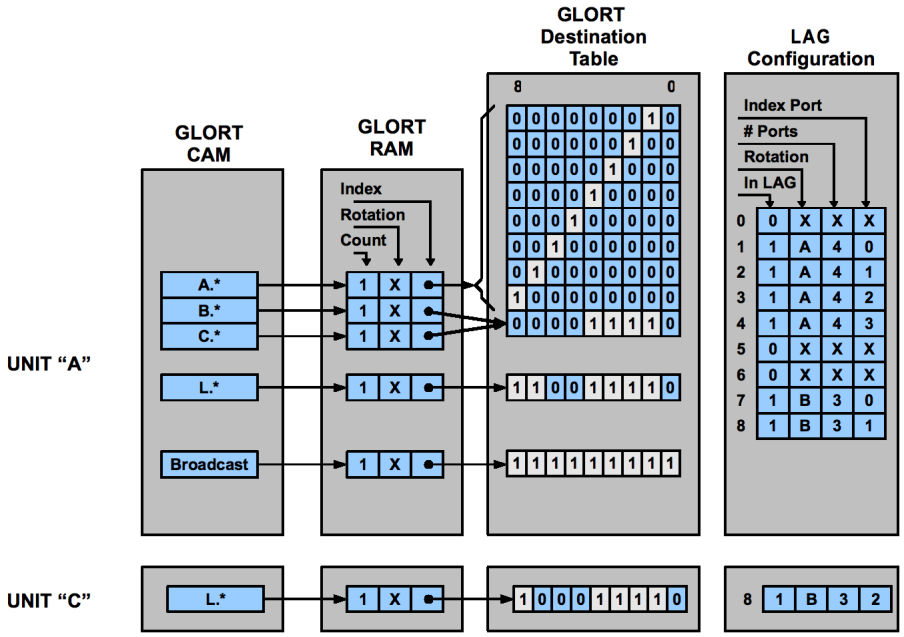


Figure 9-5 Configuration Example for LAG Filtering in a Multi-Switch System

This solution is simple and may be extended easily to support a very large configuration but has the inconvenient characteristic of duplicating some of the packets going from W to L.*. The packets going to L.2 are properly filtered out from L.0 and L.1 and are hashed across LA.* and finally reach switch "C" and then L.2. However, the packets going to L.0 and L.1 are not filtered out from LA.* and thus are sent twice, once on either L.0 or L.1 and once on the LA.* group. The extra packets sent on LA.* are received by the switch C, which filters them out, avoiding duplication on L.*, but extra bandwidth is lost on the way to spine chips transporting packets that are filtered out anyway.

An alternative approach is to use pruning to avoid sending the packets going to L.2 to the inner links (LA.*). The set up becomes the following:

- **Switch "A" and "C"** — Hashing function usage is not changed.
- **Switch "A" and "C"** — The GLORT CAM configuration is not changed.
- **Switch "A"** — The corresponding GLORT_RAM for L.* is changed:
  — L.* points to three entries in the GLORT_DEST_MASK, one for each port in L.*. The ports L.0 and L.2 points to ports 7 and 8 respectively while the third entry for port L.2 includes the LA.* links. A first level of hashing is used to distribute the traffic across the three entries while a second level of hashing (filtering) is used across the LA.* links.

— BROADCAST points to one entry in the GLORT_DEST_MASK. The filtering is used to ensure that only one packet gets eventually transmitted to the L.* group.

- **Switch "C"** — Similarly, the GLORT_RAM for L.* is changed:
  — L.* points to three entries in the GLORT_DEST_MASK. Two are identical and include LA.* only, and one will include L.2. The hash function is used to distribute the traffic equally among the three entries.
  — BROADCAST points to one entry in the GLORT_DEST_MASK. The broadcast is transmitted on the spine switch and all other ports. The LAG receives only one copy due to the lag filtering still applied in the leaf switches (A,B).
  — Note that the LAG filtering for ports A.7, A.8 and C.8 are not superfluous for unitcast (as the L.* is already pruned), but must remain in place for broadcast.

This new configuration is shown in Figure 9-6

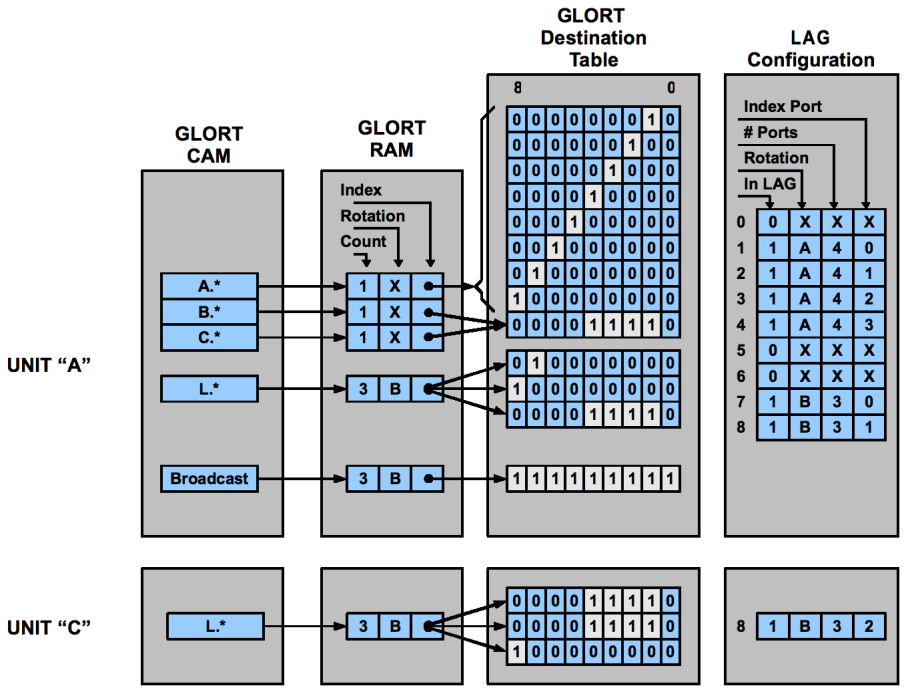


Figure 9-6 Configuration Example for LAG Pruning in a Multi-Switch System

9.4 IP Multicasting

In many basic applications, IP multicast may be treated as a simple layer 2 multicast by loading a specific destination mask in the MA_TABLE or by simply using the GloRT-flood. This however, does not address some common needs of IP multicasting:

- Crossing a LAN boundary (IP multicast routing)
- Having distinct multicasting destinations depending on source addresses

FM4000 is designed to handle complex IP multicasting such as:

- Simultaneous handling of layer 2 switching and layer 3 routing
- Multicasting across different VLANs even on same ports (packet replication)
- Different distribution for different sources
- Wire-speed and low latency in all cases

9.4.1 Getting a GloRT

The first step needed is to assign a GloRT to an IP multicast group. This is done by writing a rule in the FFU which matches the destination IP address. Depending on which multicast routing protocol the high-level software is using, it may be required to match on both source IP address and destination IP address — this is called an (S,G) pair. A route action is attached to the FFU entry to point to the ARP table and a free entry in the ARP table is then used to load the GloRT.

In the case of IP Multicast, the GloRT always comes directly from the ARP Table, while the MAC Table is not involved in determining the destination (however, learning is still enabled). This is because there is a deterministic relationship between IP multicast addresses and Ethernet multicast addresses, so one can proceed from the IP address directly without needing to convert to a MAC address. For non-IP multicast, the MAC table is still used, and may contain either a destination mask or a GloRT, but that is outside the scope of this discussion.

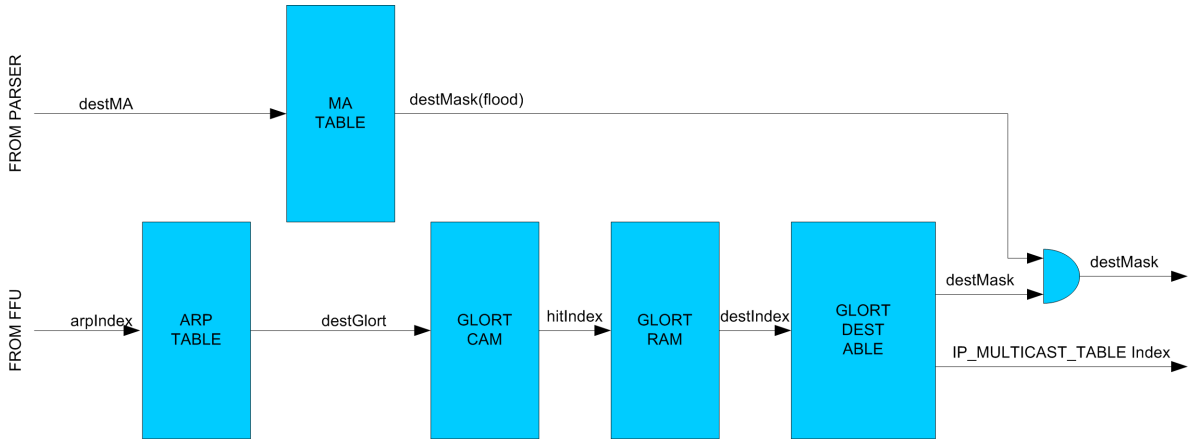


Figure 9-7 Retrieving Destination Mask and an IP_MULTICAST_TABLE Pointer

Once a GloRT is obtained, it goes into the GloRT CAM. For the purposes of this section, we assume the GloRT CAM only matches on the destination GloRT which was obtained from the FFU and/or ARP Table. However, in a multi-chip system, it may be required to match on both source and destination GloRTs. We assume for now a single-chip system where matching is only on the destination GloRT.

Because the GloRT CAM is a CAM, a single entry can match many GloRTs. This gives a great deal of freedom in organizing the GloRT space, but a full discussion of organizing the GloRT space is beyond the scope of this document. However, one simple and obvious strategy is to give all the multicast GloRTs a common prefix. This allows handling all the multicast groups with a single entry in the GloRT CAM.

Upon hitting an entry in the GloRT CAM, the hit line number is used to index the GLORT_RAM and some associated data is exposed. This associated data tells how to compute an index into the Destination Mask Table (GLORT_DEST_TABLE). The process starts with a base pointer into the Destination Mask Table. Two different things can be added to the base pointer. First, a component can be added which depends on the frame hash — this is for link aggregation purposes - and is mostly outside the scope of this section. Secondly, a component can be added which comes from the original GloRT. This is how a single entry in the GloRT CAM for all multicast groups can expand into a separate entry in the Destination Mask Table for each multicast group.

The entry in the Destination Mask Table provides a destination mask and an MTable pointer.

9.4.2 Usage of Destination Mask and IP Multicast Table

The destination mask obtained from the Destination Mask Table is not used directly, but is used to derive two other masks. (Actually, it is used to derive three other masks, but the Forward Normally mask, which is derived by masking out the dropped ports but not adding in the mirror ports, is not important to understanding IP Multicast.)

One of these masks is the destination mask which is given to the scheduler. It indicates which ports has at least one copy of the frame transmitted on them. It is derived from the Destination Mask Table by masking out any ports which have been dropped by Frame Control (e.g. by the FFU, triggers, WRED, etc.), and then adding in any ports to which the frame is being mirrored. (Such as the CPU port, or actual mirror ports.)

The other mask is the “MTable mask”, which is a constant for a given multicast group (i.e. it doesn't depend on FFU actions or anything else transient). The MTable mask is simply the mask from the Destination Mask Table, with internal ports removed. In a single-chip system, all ports are external, so the MTable mask is simply the mask from the Destination Mask Table. This mask indicates which ports have entries in the Multicast Table for this multicast group. Since IP multicast replication only needs to occur on external ports, there is no reason for the internal ports to have entries in the Multicast Table. (There is a register in Frame Control where you can configure which ports are internal and which are external.)

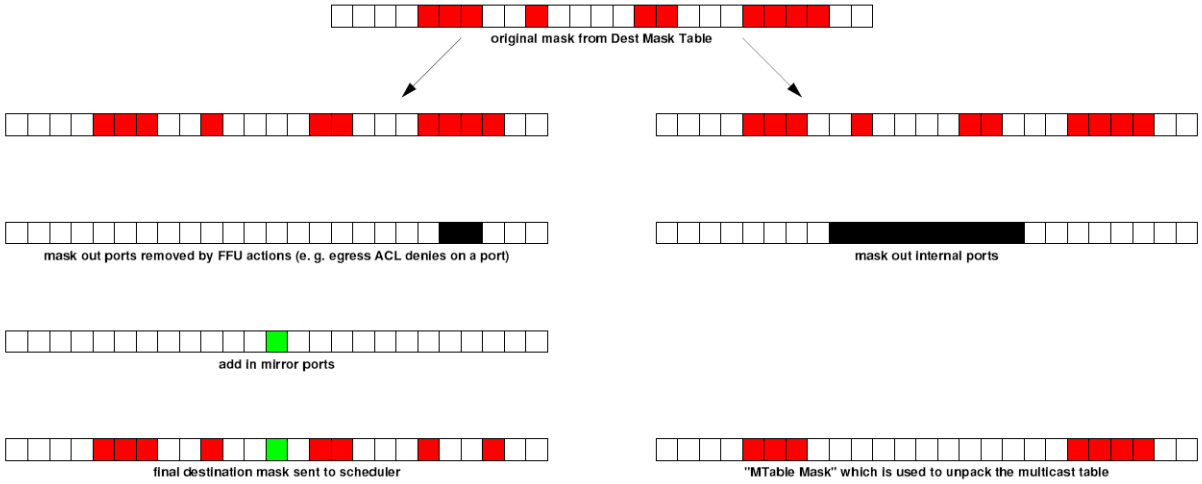


Figure 9-8 Derivation of the Scheduler Destination Mask and the MTable Mask

9.4.3 Configuring MTable

For IP multicast frames, multicast replication occurs in two stages. The first stage is the normal multicast replication, which occurs in RX_QUEUES, where a single frame is turned into one frame on each port in the destination mask. The second stage is only for IP multicast, and is implemented by the Frame Replicator (MTABLE) unit. This second stage creates multiple copies of a frame, each with a different VLAN, on a single port. The number of copies and the VLAN numbers are defined in the Multicast Table (IP_MLTICAST_TABLE). The Multicast Table has 16384 entries, and each entry is 30 bits wide:

- **parityError** (1 bit)
- **VLAN** (12 bits)
- **nextPtr** (14 bits)
- **tail** (1 bit)
- **novlan** (1 bit) — Normally, novlan is false, and the specified VLAN is attached to the frame. Setting the novlan bit to true causes one of two special behaviors to happen. If novlan is 1 and vlan is 0, the frame is sent out with an unmodified VLAN, just as if this port had not been in the MTable mask. If novlan is 1 and vlan is anything other than 0, this entry is skipped and we just move on to the next entry in the linked list.
- **skip** (1 bit) — The entry marked skip is simply skipped while doing replication. This allows the software to eliminate entries in the table without having to recreate the table.

Each entry represents a single VLAN ID which should be sent out on a particular port. The entries are linked together to form a separate linked list of VLANs for each port. The head of each linked list is found by combining the MTable Pointer from the destination mask table, which indicates the beginning of the contiguous region of memory occupied by a particular multicast group, with an offset derived from the MTable Mask. The MTable Mask indicates the ports which are members of this multicast group. There is a linked list for each set bit in the MTable Mask. MTable Pointer + 0 is the first entry of the

linked list for the lowest-numbered bit set in MTable Mask. MTable Pointer + 1 is the first entry of the linked list for the second-lowest numbered bit set in MTable Mask, etc.

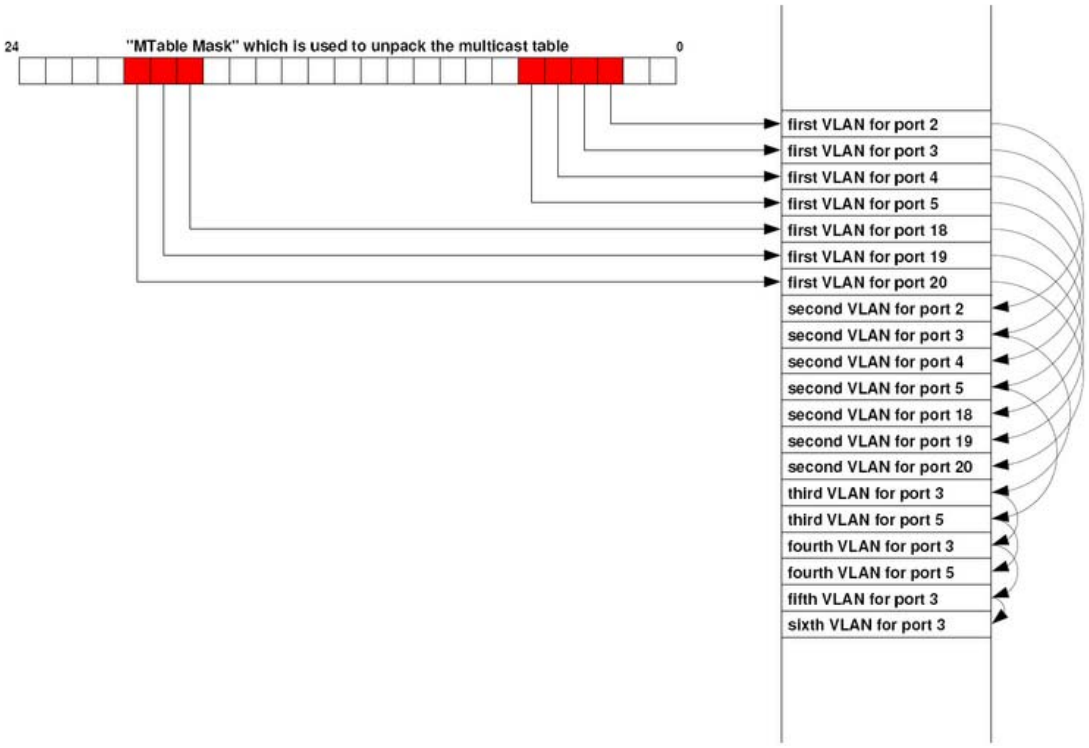


Figure 9-9 Operation of the Scheduler MTable Mask and the IP Multicast Table

For each packet sent, the VLAN retrieved is compared to the current VLAN associated with the frame. If this is the same VLAN, then the frame is deemed switched and the flag indicating that the frame shall be routed is cleared for this frame. If this is not the same VLAN, then the TTL is checked and if it is 1, then the frame is discarded.

§ §

10.0 Frame Hashing

10.1 Overview

For purposes of ECMP and Link Aggregation, two hash values are calculated from the header fields of each frame:

- **Layer 3/4 Hash** (36 bits) — For ECMP.
- **Layer 2/3/4 Hash** (48 bits) — For local and distributed link aggregation (filtering and pruning, respectively).

The keys to these hash functions are constructed in a configurable manner to provide the following features:

- **Symmetry** — Hash value remains the same when source and destination fields are swapped.
- **Static field dependence** — Support for including a specific set of header fields in the hash function.
- **Dynamic field dependence** (based on frame type) — Certain fields can be omitted or included when a frame is IPv4/IPv6.

In a multi-chip switch, it may be desirable to load balance frames over different ECMP sets or LAGs in a statistically independent fashion. For example, the same hash function should not be used to distribute traffic over the second-layer links of a 3-tier fat tree as the first-layer links. Thus, multiple independent frame hashes are calculated for a given frame, with configuration settings determining which of these hashes applies to a given frame and device. Any given device can apply a maximum of three independent hash values to a given frame: one of three independent choices for ECMP, and two of four independent choices for link aggregation.

Binning of the selected hash value is performed using division for ECMP and modulo for link aggregation. Division (also known as Hash Threshold) has the advantage of providing better stability of the bin mappings when the number of bins is changed (cf RFC 2992). This property is only important for ECMP. Therefore, in the interest of maintaining backwards compatibility, link aggregation continues to employ modulo binning. Both functions provide equally balanced hash binning.

Table 10-1 Binning Functions

Type	Equation
Division binning (ECMP)	index = base + (hash * bin_count) / 4096
Modulo binning (Link Aggregation)	index = base + hash % bin_count

Figure 10-1 illustrates the frame hashing data flow in the Frame Processing Pipeline.

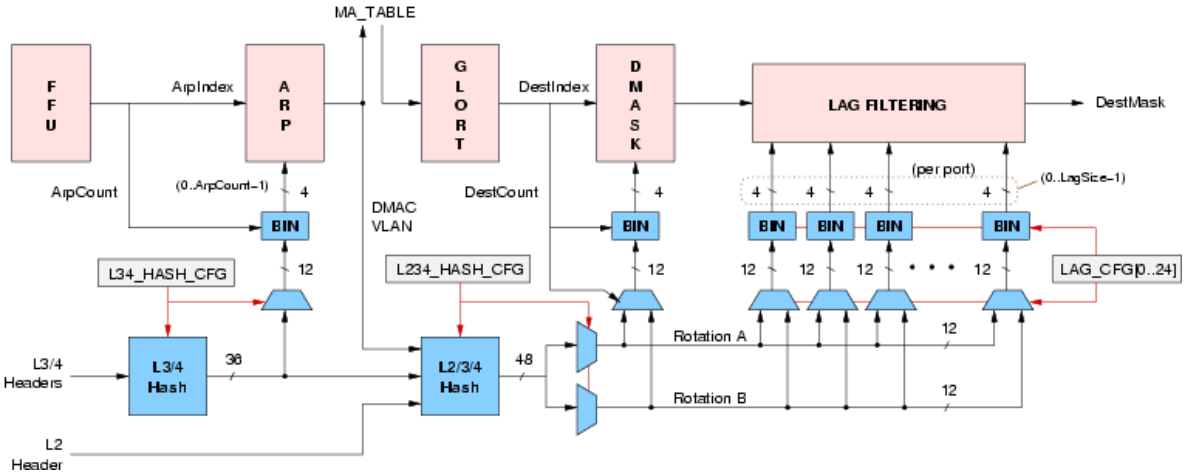


Figure 10-1 Frame Hashing

Note: Although the L2/3/4 hash produces four 12-bit rotations, only two are available for use on a given chip. This restriction limits the amount of parallel binning circuitry in the implementation. In the register definitions, these two globally-selected rotations are referred to as Rotations A and B.

10.2 Layer 3/4 Hash

The 36-bit layer 3/4 hash value is calculated using two 32-bit CRC functions and a 12-bit permutation table:

```
H34[31:0]  = CRC32(0xEDB88320, Bytes[0..41])
H34[35:32] = CRC32(0x82F63B78, Bytes[0..41])[3:0]
```

In this notation, the first parameter of the CRC32 function specifies the CRC polynomial; the second parameter specifies the byte sequence input key to the CRC. The 0xEDB88320 polynomial corresponds to the standard 802.3 Ethernet CRC32.

The input byte sequence is assigned from up to 42 bytes of the frame header:

Bytes 15:0	Bytes 31:16	Bytes 36:32	Byte 37	Bytes 39:38	Bytes 41:40
SIP	DIP	L3FLOW	L4PROT	L4SRC	L4DST

The particular values used in the SIP, DIP, L3FLOW, L4PROT, L4SRC, and L4DST fields are determined based on the L34_HASH_CFG configuration and the contents of the packet header, as illustrated in the following tables.

Table 10-2 Layer 3/4 Hashing SIP Field

	UseSIP	Symmetric	Bytes 11:0	Bytes 15:12
IPv4	1	0	0	SIP[31:0]
	1	1	0	SymA(SIP[31:0], DIP[31:0]*)
IPv6	1	0	SIP[127:32]	SIP[31:0]
	1	1	SymA(SIP[127:32], DIP[127:32]*)	SymA(SIP[31:0], DIP[31:0]*)
IPV4/IPv6	0	x	0	

* DIP field is zero if UseDIP == 0.

Note: The SymA and SymB functions return a convolution of its arguments such that SymA(x,y) == SymA(y,x). (Similar for SymB, referenced below.)

Table 10-3 Layer 3/4 Hashing DIP Field

	UseDIP	Symmetric	Bytes 27:16	Bytes 31:28
IPv4	1	0	0	DIP[31:0]
	1	1	0	SymB(SIP[31:0]*, DIP[31:0])
IPv6	1	0	DIP[127:32]	DIP[31:0]
	1	1	SymB(SIP[127:32]*, DIP[127:32])	SymB(SIP[31:0]*, DIP[31:0])
IPV4/IPv6	0	x	0	

* SIP field is zero if UseSIP == 0.

Table 10-4 Layer 3/4 Hashing Flow Field

	Byte 32	Bytes 35:33[19:0]	Byte 35[7:4]	Byte 36
IPv4	DiffServ[5:0] & DiffServMask	0	0	ISL_USER & UserMask
IPv6	TrafficClass[5:0] & DiffServMask	FlowLabel & FlowLabelMask	0	

The flow identification (Flow Label) fields are each individually bit-masked to provide fine control over which bits are included in the hash value. The masks are configured in L34_FLOW_HASH_CFG. The ISL_USER field comes from the ISL tag, or from the default specified in PORT_CFG_ISL if the frame is not F64-tagged.

Table 10-5 Layer 3/4 Hashing Protocol Field

	UsePROT	Symmetric	Byte 37
IPv4	1	x	Protocol
IPv6	1	x	NextHeader
IPv4/IPv6	0	x	0

Table 10-6 Layer 3/4 Hashing Layer 4 Fields

	UseL4{SRC,DST}	Symmetric	Byes {39:38, 41:40}
UseTCP==1 and Protocol is TCP UseUDP==1 and Protocol is UDP UsePROT1==1 and Protocol is PROT1 UsePROT2==1 and Protocol is PROT2	1	0	{SRC,DST}_PORT
	1	1	{SymA(SRC_PORT, DST_PORT), SymB(SRC_PORT, DST_PORT)}
	0	x	0
Otherwise	x	x	0

10.3 Layer 2/3/4 Hash

The 48-bit L2/3/4 hash value used for link aggregation pruning and filtering is calculated in a similar manner to the L3/4 hash, using an additional 16 layer 2 bytes from the frame header:

```
H234[31:0]  = (~FrameIsIP | UseL2ifIP) * CRC32(0xEDB88320, Bytes[0..15]) ^ UseL34 * H34[31:0]
H234[35:32] = (~FrameIsIP | UseL2ifIP) * CRC32(0x82F63B78, Bytes[0..15])[3:0] ^ UseL34 * H34[35:32]
H234[47:36] = (~FrameIsIP | UseL2ifIP) * CRC32(0x82F63B78, Bytes[0..15])[15:4] ^ UseL34 * H34[11:0]
```

Four different “rotations” of the 48-bit value are used to derive four distinct hash values:

- Rotation 0: H234[11:0]
- Rotation 1: H234[23:12]
- Rotation 2: H234[35:24]
- Rotation 3: H234[47:36]

from which may be chosen Rotation A and Rotation B .

When the FM2000Compatible configuration bit is set to 1b, hash rotation 0 (bits 11:0) is defined to give a FM2000-compatible hash value:

```
H234[11:0] = CRC32(0xEDB88320, Bytes[0..15])[7:0] ^ CRC32(0xEDB88320, Bytes[0..15])[15:8] ^
CRC32(0xEDB88320, Bytes[0..15])[23:16] ^ CRC32(0xEDB88320, Bytes[0..15])[31:24]
```

This definition gives an exclusively layer 2 hash value that is folded down to 8 bits to be backwards compatible with the FM2000 hash function when FM2000Compatible is 1. When FM2000Compatible is 0, the 48-bit H234 hash value provides four independent hash functions that optionally include L2 or L3/4 header fields, depending on configuration settings and the type of frame.

The UseL2ifIP and UseL34 configuration bits in L234_HASH_CFG provide the following options for dynamic header field dependence:

UseL2ifIP	UseL34	Semantics
1	1	H234 always includes the maximum amount of header information.
1	0	H234 only includes layer 2 header fields.
0	1	H234 only includes layer 3/4 header fields when the frame is IP. If the frame is non-IP, the layer 2 header is used for hashing.
0	0	H234 is always zero for IP frames. Otherwise it uses the layer 2 header. Invalid configuration.

The sixteen layer 2 input bytes to the CRC are defined in terms of the following fields:

Bytes 5:0	Bytes 11:6	Bytes 13:12	Bytes 15:14
DMAC	SMAC	TYPE	VLAN

The settings in L234_HASH_CFG control how these fields are constructed from the frame's layer 2 header.

Table 10-7 Layer 2 Hashing DMAC Field

UseDMAC	Symmetric	FM2000Compatible	Bytes 5:0
1	0	x	DMAC
1	1	0	SymA(DMAC, SMAC*)
0	1	1	EvenBits(DMAC ^ SMAC)**
0	0	1	OddBits(DMAC ^ SMAC)**
0	x	x	0

* SMAC field is zero if UseSMAC == 0.
** FM2000-compatible symmetric hash functions. EvenBits() sets even bits to DMAC^SMAC, odd bits to zero; vice versa for OddBits().

Note: The DMAC used in the hash function is always the DMAC as received on ingress.

Table 10-8 Layer 2 Hashing SMAC Field

UseSMAC	Symmetric	FM2000Compatible	Bytes 11:6
1	0	x	SMAC
1	1	0	SymB(DMAC*, SMAC)
0	1	1	EvenBits(DMAC ^ SMAC)**
0	0	1	OddBits(DMAC ^ SMAC)**
0	x	x	0

* DMAC field is zero if UseDMAC == 0.
** FM2000-compatible symmetric hash functions. EvenBits() sets even bits to DMAC^SMAC, odd bits to zero; vice versa for OddBits().

Note: The SMAC used in the hash function is always the SMAC as received on ingress.

Table 10-9 Layer 2 Hashing Type Field

	UseTYPE	Bytes 13:12
EtherType < 0x600	1	0
EtherType ≥ 0x600	1	EtherType
	0	0

Note: The EtherType included in the hash is the layer 2 header's first non-VLAN EtherType.

Note: The EtherType included in the hash is the layer 2 header's first non-VLAN EtherType. In contrast, FM2000 includes the first EtherType in its frame hashing. Thus, if backwards compatibility with FM2000 is required, UseType should be set to zero.

Table 10-10 Layer 2 Hashing VID/VPRI Fields

EtherType	UseVPRI	UseVID	Byte 14[7:4]	Bytes 15:14[11:0]
0x8100 0x9100 0x9200	0	1	0	VLAN ID
	1	0	VPRI	0
	1	1	VPRI	VLAN ID
	0	0	0	0
Non-VLAN Type	x	x	0	0

The VLAN ID used in the hash function is the final associated value, prior to any modifications due to routing. In other words, it is the value at the output of the FFU. Similarly, the VPRI included in the hash is the final associated value at the output of the FFU.

10.4 FM2000 Compatibility

Multi-chip systems using both FM2000 and FM4000 devices may, depending on the hardware learning configuration, require consistent frame hashing across the two generations of switches. For example, in a fat tree with learning enabled in the spine chips, it is important that (a) the hash function be symmetric and (b) all edge switches use the same hash function. Table 10-11 summarizes the configuration requirements that are necessary to achieve consistent hashing between FM4000 and FM2000 devices.

Table 10-11 Summary of Hash Configuration Requirements for FM2000 Compatibility

FM4000 Requirement	FM2000 Requirement	Motivation
FM2000Compatible == 1 UseSMAC == 0 UseDMAC == 0	—	Select FM2000 Compatibility mode.
UseType == 0	IncludeTypeAndSource == 0	See note above about differences between FM4000 & FM2000 in their handling of EtherType.

§ §

11.0 Triggers

11.1 Overview

In addition to the trapping, ACLs, routing, discarding, and forwarding rules described above that implement various protocols, the switch also contains a general set of rules for modifying frames at the last stage of the pipeline. These rules are user programmable and are referred to as "triggers." A total of 64 triggers are supported.

A trigger is defined by two parts: a Match Condition and an Action Specification. The Match Condition is a programmable Boolean expression. If all of the conditions defined in the expression are true, the trigger "fires" and the Action Specification is applied to the packet. For a given packet, any number of triggers may fire. In the case of conflicting Action Specifications, an Action Resolution step determines exactly how the packet is handled.

While triggers are very general, their placement in the overall frame processing pipeline imposes a fundamental limit to their use. Specifically, the triggers are evaluated after the GLORT mapping stage, egress ACLs, and layer 2 trapping, but before link aggregation filtering and congestion management. The implications of this placement in the pipeline are detailed in the Trigger Actions section below.

11.2 Trigger Match Conditions

Each trigger specifies a list of conjunctive match conditions involving properties of the frame and corresponding configured trigger parameters. If all of the match conditions evaluate true, the entire trigger matches and is said to ''fire." The programmable match conditions are listed in Table 11-1.

Table 11-1 Programmable Trigger Match Conditions

Condition	Frame Property	Trigger Parameter(s)
MatchSA	TrigID (6 bits) from the Layer 2 lookup of the frame's source MAC Address. This field is 0b if there is no match in the MA Table.	SA_ID (6 bits)
MatchDA	TrigID (6 bits) from the Layer 2 lookup of the frame's destination MAC Address. This field will be 0b if there was no match in the MA Table.	DA_ID (6 bits)
MatchHitSA	Source MAC Address match in the Layer 2 MA Table. If a source lookup was performed and the address was found in the table, this condition evaluates to true.	—
MatchHitDA	Destination MAC Address match in the Layer 2 MA Table. If the address was found in the table, this condition evaluates to true.	—

Table 11-1 Programmable Trigger Match Conditions (Continued)

Condition	Frame Property	Trigger Parameter(s)
MatchHitSADA	Source or Destination MAC Address match in the Layer 2 MA Table. The match condition is true if either (or both) addresses are found in the Layer 2 MA Table.	—
MatchVlan	TrigID (6 bits) from the EGRESS_VID_TABLE lookup.	VID_ID (6 bits)
MatchFFU	TrigID (8 bits) set from the FFU's SET_TRIG action. If no FFU rule hit specifies a SET_TRIG action, this condition cannot match for any configured value of FFU_ID and FFU_Mask. The default FFU_ID produced by the FFU is 0.	FFU_ID (8 bits) FFU_Mask (8 bits)
MatchSwitchPri	Associated switch priority (4 bits).	SwitchPri (4 bits)
MatchEtherType	First non-VLAN EtherType (16 bits).	EtherType (16 bits) EtherTypeMask (16 bits)
MatchDestGlort	Destination GLORT (16 bits) associated with the frame.	DestGlort (16 bits) DestGlortMask (16 bits)
MatchFrameClass	The frame class as defined by the Layer 2 Destination MAC Address: 0 = Unicast 1 = Broadcast 2 = Multicast If the corresponding bit in the configured FrameClassMask is set to '1', this trigger condition evaluates to true.	FrameClassMask (3 bits)
MatchSrcPort	Ingress port number (5 bits). If the corresponding source port bit in the configured SrcPortMask is set to 1b, this trigger condition evaluates to true.	SrcPortMask (25 bits)
MatchDestPort	Destination port mask (25 bits). If the corresponding destination port bit in the configured DestPortMask is set to 1b, this trigger condition evaluates to true.	DestPortMask (25 bits)
MatchRouted	Value of the mark_routed bit from the FFU ROUTE action (0 if no such action was applied to the frame).	RoutedMask (2 bits)
MatchFTYPE	ISL tag FTYPE field.	FtypeMask (4 bits)

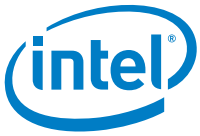


Table 11-1 Programmable Trigger Match Conditions (Continued)

Condition	Frame Property	Trigger Parameter(s)
MatchHandlerAction	The frame processing pipeline maintains a bit vector of all actions applicable to each frame that it handles. For example, if a frame arrives on a port in the Disabled spanning tree state, a corresponding action bit is set, indicating that the frame should be dropped. At the end of the pipeline, a precedence resolution is performed over all applicable actions to determine the final disposition of the frame. By matching against arbitrary bits in this action bit vector, the triggers have the capability to overrule nearly any handling decision made by the processing pipeline. In addition to the action codes listed in Section 2.6, the following action codes also apply: 38: Header was too long to be completely parsed. 39: Frame copied to the CPU as a result of FFU action. 40: Frame copied to the CPU as a result of ACL action. 41 Frame was mirrored as a result of FFU action. 42: IP unicast frame was logged to the CPU because it had an L2 multicast address 43: Multicast ICMP frame was copied to the CPU because it's TTL was <= 1. 44: Multicast frame was copied to the CPU because it's TTL was <= 1.	HandlerActionMask (64 bits)
MatchRandom	One of two 24-bit pseudo-random numbers is associated with the frame. The trigger matches if the number is less than or equal to 2^RandomThreshold.	RandomNumber (1 bit) RandomThreshold (5 bits)
MatchByPrecedence	—	MatchByPrecedence (1 bit)

Most of the match conditions require a configured Match Case value specifying the sense of the match test:

0 = Match if the frame property does not equal the trigger parameter.

1 = Match if the frame property does equal the trigger parameter.

2 = Match unconditionally.

Since the MatchFrameClass, MatchSrcPort, and MatchHandlerAction conditions already provide these matching semantics by taking a mask as their configured parameters, they do not require a Match Case assignment. All triggers have default Match Case values of 2 for all conditions. The SrcPortMask default value of 0x0 disables the triggers from matching all frames. This mask must be set to some nonzero value to activate a trigger.

The default values of MatchHandlerAction and HandlerActionMask are defined such that normally a frame that is dropped earlier in the pipeline does not match against any trigger. Only frames that are flooded or forwarded normally match a trigger with these default settings. This behavior can be overridden by either clearing MatchHandlerAction or by changing HandlerActionMask. The action codes are defined in Section 2.6.

11.2.1 Configurable Trigger Precedence

The MatchByPrecedence condition controls whether otherwise matching triggers are applied to the frame in parallel or in a precedence fashion based on their constant trigger number. If MatchByPrecedence is set to 1b, the trigger can only fire if no other lower-numbered trigger matches since (and including) the last trigger with MatchByPrecedence==0. If MatchByPrecedence is set to 0b, the trigger fires unconditionally, assuming all other match conditions are satisfied. Table 11-2 illustrates this behavior:

Table 11-2 Trigger Precedence

Trigger Number	MatchByPrecedence	Matches?	Fires?	Explanation
0	0	Y	y	Evaluated in parallel.
1	0	N	N	Begins a new precedence block.
2	1	N	N	
3	1	Y	Y	First trigger in precedence block to fire.
4	1	Y	N	Excluded by Trigger 3 firing.
5	0	Y	Y	Begins a new precedence block.
6	1	Y	N	Excluded by Trigger 5.
7	1	N	N	
8	1	Y	N	Excluded by Trigger 5
…				

Note: Hardware always processes trigger #0 as if its MatchByPrecedence is zero, regardless of what is specified in the register configuration.

11.2.2 Random Matching

The triggers support statistical rate-based firing with the MatchRandom condition. This condition matches based on a comparison between a 24-bit pseudo-random number and a RandomThreshold rate parameter configured per trigger. Two random numbers are calculated globally, with each trigger specifying which of the two is to be used in its RandomThreshold comparison. Additionally, the MatchRandomIfLess parameter determines whether the condition matches based on a less-than-or-equal-to comparison or a greater-than-or-equal-to comparison.

The match condition is determined by the following expression:

```
If (MatchRandomIfLess)
      Match = (RandomNumberGenerator[Trig[i].RandomNumber] <= 2^Trig[i].RandomThreshold)
Else
      Match = (RandomNumberGenerator[Trig[i].RandomNumber] >= 2^Trig[i].RandomThreshold
```

On each clock cycle, a new 24-bit RandomNumberGenerator value is calculated with 24 iterations of a 31-bit LFSR. Assuming a 375 MHz Frame Handler clock frequency, the period of the pseudo-random number sequence is 5.7 seconds. The two LFSRs are seeded differently, providing two statistically independent random numbers.

The MatchRandom condition does not require a Match Case (equal/not-equal/unconditional) configuration. To disable rate-constrained matching, RandomThreshold should be set to the maximum value (or any value equal or larger than 24); otherwise, some value between 0 and 23 should be selected.

11.3 Trigger Actions

Each trigger specifies a list of actions to apply to the frame. Nine sets of mutually exclusive actions are defined in Table 11-3.

Table 11-3 Trigger Actions

Trigger Action Set	Trigger Action	Associated Configuration	Description
ForwardingAction	0 - Leave as-is	—	No change in forwarding.
	1 - Forward	NewDestGlort (16 bits) NewDestGlortMask (16 bits)	Forward the frame with a new destination GloRT. Destination GloRT bits are overridden from NewDestGlort whose corresponding NewDestGlortMask bits are 1b. This action can also be used to undo a drop decision from earlier in the frame processing pipeline (see Section 11.3.1).
	2 - Redirect	NewDestMask (25 bits) FilterNewDestMask (1 bit) NewDestGlort (16 bits) NewDestGlortMask (16 bits)	Override destination mask to NewDestMask. If FilterNewDestMask is set to 0b, link aggregation filtering is not applied to this new destination mask. If FilterNewDestMask is set to 1b, LAG filtering is applied, even if the frame's FTYPE equals 2 or 3. NewDestGlort is applied as for the Forward case.
	3 - Drop	DropMask (25 bits)	Do not forward to any ports i with DropMask[i]==1.

Table 11-3 Trigger Actions (Continued)

Trigger Action Set	Trigger Action	Associated Configuration	Description
TrapAction	0 - Leave as-is	—	No change in CPU trapping.
	1 - Trap	CpuTruncate (1 bit)	Trap to CPU (overrides ForwardingAction). Truncation of this frame is enabled if CpuTruncate is set to 1b. In order for the trapped frame to be truncated, the corresponding CpuTruncationEnable in the egress port must also be set.
	2 - Log	CpuTruncate (1 bit)	Log to CPU with optional truncation. When truncation is enabled, only the copy of the frame sent to the CPU is truncated (and only as long as the corresponding CpuTruncationEnable bit is set in the egress port.)
	3 – Do not Trap or Log	—	Overrides a trap or log action specified from an earlier point in the frame processing pipeline.
MirroringAction	0 - Leave Unchanged	—	No change in mirroring disposition.
	1 - Mirror	MirrorPort (5 bits) MirrorGlort (16 bits) MirrorTruncate (1 bit)	Mirror the frame to the specified port with the specified GloRT. This action overrides the FFU's mirror action. Truncation of the mirrored frame is enabled by setting MirrorTruncate to 1b. In order for the mirrored frame to be truncated, the corresponding MirrorTruncationEnable bit in the egress port must also be set.
SwitchPriAction	0 - Leave as-is	—	No change in the frame's associated switch priority.
	1 - Reassign Switch Priority	NewSwitchPri (4 bits)	Associate NewSwitchPri with the frame, for purposes of congestion management and egress queueing. ***Note:*** This has no effect on the FFU's
VlanAction	0 - Leave as-is	—	No change in VLAN.
	1 - Reassign egress VLAN	NewVlan (12 bits)	Override egress VLAN to the specified value. ***Note:*** This only affects L2-switched and L3-unicast frames. IP multicast replication, if applicable, still produces frames with the same egress VLANs as if this action was not specified.
LearningAction	0 - Leave as-is	—	No change in learning.
	1 - Disable Learning	—	Do not learn the source L2 address into the MAC Address Table.
	2 - Force Learning	—	Learn this frame's source L2 address, even if it will be dropped.
RateLimitAction	0 - Leave as-is	—	Do not apply a Trigger Rate Limiter to this frame.
	1 - Rate Limit	RateLimitNum (4 bits)	Assign this frame to one of 16 Trigger Rate Limiters. A drop mask is applied to the frame if the specified rate limiter's bandwidth limit is exceeded.

As illustrated in Figure 11-1, triggers are applied after GloRT mapping, egress ACLs, and action resolution, but before LAG filtering and congestion management.

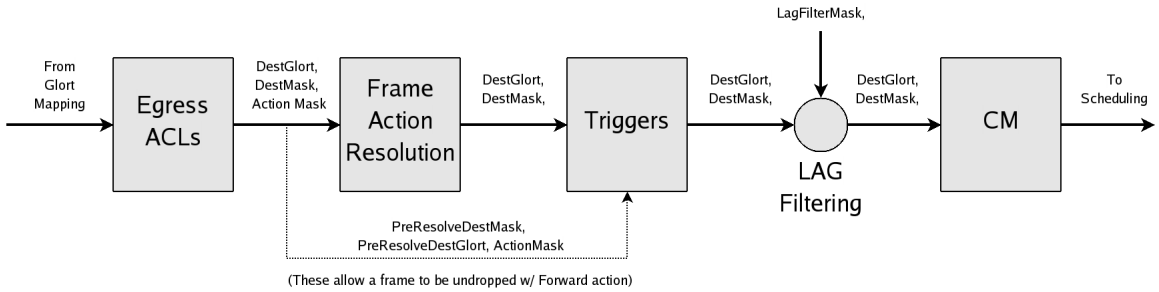


Figure 11-1 Trigger Application

This location in the frame processing pipeline imposes the following constraints on the trigger actions:

- Port numbers and destination mask bits apply to pre-filtered LAG membership ports, so the MatchDestPort condition cannot match on individual physical ports belonging to a LAG. The Redirect ForwardAction offers a FilterNewDestMask configuration, which when set to 0b bypasses LAG filtering and allows frames to be directed to specific physical ports.
- Any assigned destination glorts and port masks are applied to the frame directly and must be mutually consistent. There is no further mapping of the specified GloRT.
- Frames are still subject to dropping due to congestion even if they are trapped, redirected, mirrored or otherwise forwarded normally by the triggers.

11.3.1 Using Triggers to Undrop Frames

The Frame Action Resolution step in the Frame Processing Pipeline applies various hard-coded precedence rules to determine a final handling action for a given frame. For example, a frame might ingress on an invalid (port, VLAN) pair, yet it might also match a Trap FFU rule. The action resolution step determines that the security violation in this case has higher precedence than the Trap rule, causing the frame to be dropped. Since these precedence rules are hard-coded (see the table of Action Codes in Section 2.6), there may be a need to fine-tune these precedence decisions in certain applications. The triggers provide this capability by supporting the MatchHandlerAction match condition in conjunction with the Forwarding action.

For purposes of overriding the action resolution decision, the most important operation the triggers provide is “undropping” a frame. This behavior is specified by selecting ForwardingAction==1. In this case, if the frame was dropped, the destination mask and GloRT prior to the Frame Action Resolution step is restored. The frame's action handling code is also restored to its original value (this matters only for Group 6 packet counter classification.)

11.3.2 Rate Limiting

The triggers implement sixteen drop-based rate limiters. A given frame may be mapped to one or more of these rate limiters, although only one rate limiter is changed and only one drop mask is applied to a given frame. If the rate limiter is out of profile, the frame's destination mask is filtered by a configured drop mask. Example applications of this feature include the following:

- Limit chip-wide proportion of flooding traffic to protect against denial-of-service attacks.
- Limit low-priority traffic sent or trapped to the CPU in a rate-controlled manner rather than in a watermark-controlled manner.
- L2 policing: Limit bandwidth directed to a particular DMAC, VLAN, or egress port.

The trigger rate limiter algorithm is a simplification of the FFU-indexed tri-color marking policers described in Section 12.7. Specifically, each rate limiter consists of a single token bucket with configured Capacity and Rate parameters. Every sixteen Frame Handler clock cycles, the token bucket is refilled by:

```
refillAmount = (RateMantissa * 32) >> (RateExponent + 2)
```

where refillAmount is in units of 1/16 bytes, RateMantissa is a 12 bit quantity and RateExponent is a 2-bit quantity. These parameter definitions give a rate limit range between 1.46 MB/s and 47.8 GB/s.

Note: The maximum chip-wide aggregate bandwidth is 10Gb/s / 8 * 24 = 30GB/s.

If a rate limiter applied to a frame depletes its token bucket credit, a DropMask associated with the rate limiter is aggregated (OR'd) with the DropMask determined from all applicable ForwardingAction==3 (Drop) actions. The frame's final destination mask is then filtered by the DropMask.

11.3.3 Trigger Action Resolution

Note: "Trigger Action Resolution" is not to be confused with the "Frame Action Resolution" step in the Frame Processing Pipeline described in Section 11.3.1.

When multiple matching triggers fire in parallel (due to the use of MatchByPrecedence==0), the set of actions specified by these triggers must be resolved to a single non-conflicting set. There are three general rules governing this resolution process:

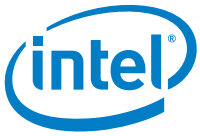
1. Whenever possible, non-conflicting actions are all applied.
2. Otherwise, higher precedence dispositions override less specific ones. The action values are defined such that a higher value indicates higher precedence. For example, the "Override VLAN" case of VlanAction==1 would override the "Leave as-is" case of VlanAction==0.
3. In all other cases, lower-numbered triggers override the actions of higher-numbered triggers.

Rule 1 applies to the following limited set of actions:

- ForwardingAction==3 (Drop). In this case, each trigger's DropMask is applied:

  ```
  NewDestMask = ~OR(DropMask[i]) & DestMask
  ```

  The OR is evaluated over all matching triggers i with ForwardingAction==3 and all out-of-profile rate limiters applied to the frame. The "DestMask" referenced in the above equation is the final post-trigger DestMask, reflecting the result of all redirect, trap, etc. actions applied to the frame.



Note: In the case of mirrored frames, a given frame may be mapped to one or more of these rate limiters, although only one rate limiter is changed and only one drop mask is applied to a given frame. In this case, the trigger with the lowest ID number is applied.

Rule 2 applies in all other cases whenever ActionValue[i] > ActionValue[j] for any two matching triggers i and j. ActionValue[i] is given precedence.

Rule 3 applies in the remaining cases when ActionValue[i]==ActionValue[j] for any two matching triggers i and j. ActionValue[min(i,j)] applies.

In the case that a frame is completely dropped to all destinations due to trigger actions, the frame's action code is set to “TriggerDrop” and is classified accordingly by the Group 6 packet counters.

11.4 Trigger Counters and Interrupts

Whenever a trigger fires, the count associated with that trigger is incremented.

Each trigger is associated with a bit in one of the TRIGGER_IP registers. Whenever a trigger fires, the corresponding IP bit is set.

§ §

NOTE: ***This page intentionally left blank.***

12.0 QoS and Congestion Management

12.1 Overview

FM4000 switches include the following traffic management features:

- Traffic Classification
- Traffic Policing
- Traffic Shaping
- Class Based Flow Control
- Congestion Management
- 802.1q
- DiffServ

FM4000 switches use the following frame attributes to manage the flow:

- VLAN priority
  - The switch processes the ingress/egress VLAN priority as a 4-bit entity combining the actual PRI field in the VLAN packet with the CFI bit. The encoding is as follow:
  - VPRI{3:1} = VLAN priority
  - VPRI{0} = CFI
- DSCP (for DiffServ)
- Switch priority (16-level, derived from VLAN priority or DSCP field or ISL tag)
- Traffic class (derived from switch priority)

The QOS pipeline contains the following elements:

1. Classification
   - Associates a switch priority from VLAN priority and/or DSCP field
2. Re-prioritization
   - Changes priority (VLAN, switch or DSCP) from frame content using FFU
3. Limiting Ingress Rate
   - Limits input rate using PAUSE frames

4. Policing Incoming Traffic
   - Measures input rate per flow and changes switch priority or DSCP if the flow exceeds certain thresholds
5. Monitoring Memory
   - Tracks memory utilization and discards if required
6. Rate Control
   - Activates rate limiters (2 per port) if incoming traffic is above a certain threshold.
7. Congestion Notification
   - Notifies upstream nodes of presence of congestion
8. Scheduling
   - Sending frame according to rate and priority

12.2 Processing of Frames with Trailing Errors

FM4000 switches can operate in cut-through mode where decisions about the frame disposition or accounting may be made before the end of frame is received. The exact disposition of frames with trailing errors (invalid encoding, bad CRC, oversized, etc.) varies across the sub-units of the QoS and Congestion Management unit.

Policer

The policer's role is to check that ingress traffic meets certain criteria and to apply priority changes if the traffic exceeds certain limits. This is applied regardless of whether the frame has a termination error or not, as the frame can possibly be transmitted even if it is an erroneous frame.

Memory Management

The role of memory management is to check that the memory usage is within the boundaries defined by the user. A frame with a CRC error still uses some amount of memory and it must be accounted for, as are non-errored frames.

Note: Note that frames with invalid CRCs that are not yet transmitted are quickly erased from memory and use memory only momentarily, but they are still accounted for.

Scheduler

The scheduler may have received the order to transmit a frame before the end of frame was received. If this is the case, the erroneous frame is still accounted for as a frame being transmitted, and is included in the traffic shaping computations. If the frame was never transmitted (dropped before scheduling occurred), the frame is not included in the traffic shaping functions.

Ingress Rate Limiters

Ingress rate limiters are used for transmitting PAUSE frames back to the source if the incoming traffic exceeds certain bandwidth limitations. To avoid generating PAUSE frames on invalid traffic, the ingress rate limiters do not include erroneous frames in the rate computation.

Triggers Rate Limiters

Triggers in general are used to apply switching actions (logging, mirroring, redirection) if ingress frames meet certain criteria. The triggers are applied before the end of frame is received, and the triggers rate limiters do the same for consistency. Hence, the triggers rate limiters include erroneous frames in the rate computation.

12.3 Differentiated Services (DiffServ)

DiffServ uses an 8-bit field in the IP header to carry priority information about the frame. This field structure is shown here:

0	1	2	3	4	5	6	7
DCSP						ECN	

The Differentiated Services Code Point (DSCP) defines the class of service, while the Explicit Congestion Notification (ECN) reports congestion in the forwarding direction. FM4000 supports traffic conditioning as required by DiffServ, which includes the following: meter, marker, shaper and dropper.

- The "meter" and "marker" are detailed together in Section 12.7.
- The "shaper" is detailed in Section 13.0, "Egress Scheduling and Shaping".
- The "dropper" is detailed in Section 12.7.

12.4 Processing Priority

Figure 12-1 illustrates the priority handling in the switch.

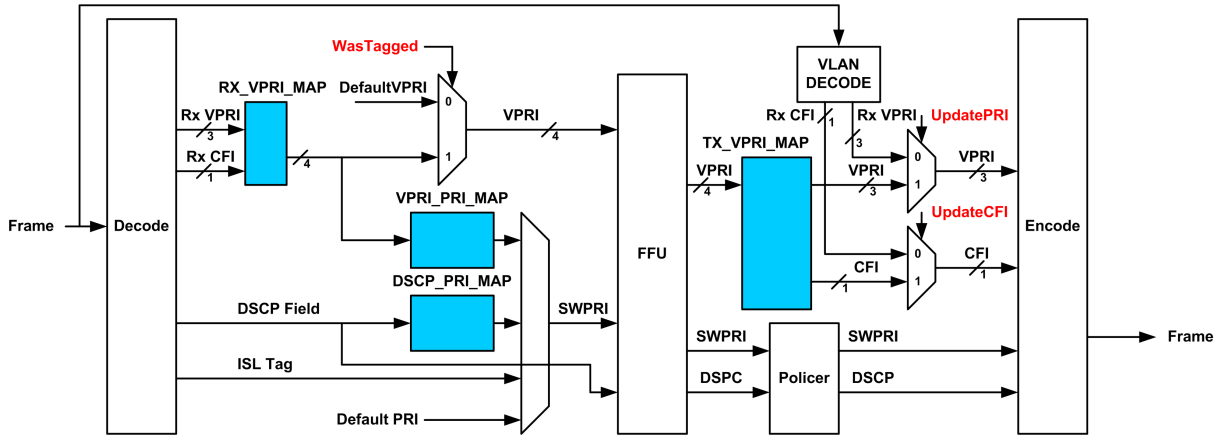


Figure 12-1 Priority Processing Block Diagram

Each frame may include one or more of the following fields:

- VLAN priority
- DSCP/TOS
- Intel TAG switch priority

The purpose of the traffic classification entity is to derive an internal switch priority (16 levels), a DSCP and a VLAN priority for each frame from the information retrieved within the frame. The steps are:

Step 1: Define VLAN Priority

The RX VLAN user priority (including the DEI/CFI bit) is mapped into a 4-bit internal VPRI if present, using the RX_VPRI_MAP register. A per-port configurable VLAN default priority field (PORT_CFG_1::defaultVPRI) is used to assign a VPRI if the frame did not include a VLAN priority tag. This default priority is also used if the PORT_CFG_1::AssumeUntag is set.

The VPRI then gets presented to the FFU, which may optionally be configured to change this value. The final VPRI is then sent to the output port, and is regenerated back into the TX VLAN user priority through a TX_VPRI_MAP register (the MAC_CFG_2::EnableVLANPriUpdate and MAC_CFG_2::EnableVCfiUpdate control if the priority present in the frame is updated or not. This applies only if the VLAN tag is actually updated, and does not apply if the VLAN tag is dropped or added). These mapping registers are different per port, which allows different ports to assign different meanings to their vlan user priorities. However, they must agree on the internal 4-bit VPRI.

Step 2: Define DSCP

The DSCP is normally taken from the IP header, unless the useDefaultDSCP bit of PORT_CFG_1 is 1b, in which case it is overwritten with the defaultDSCP field from the same register. The DSCP is presented to the FFU, which may change it, and then to the policer.

Step 3: Derive Switch Priority

The internal switch priority can be assigned from four sources, controlled by bits of PORT_CFG_1 and using a default value as follows:

- If the frame is ISL tagged and the SwitchPriorityFromISL bit is true, the switch priority is set to the ISL Priority field.
- Otherwise, if the frame is IP and the SwitchPriorityFromDSCP bit is true, the switch priority is retrieved from the global DSCP_PRI_MAP register which maps the 64 possible DSCP codes into a 4-bit switch priority code.
- Otherwise, if the frame has a VLAN tag and the SwitchPriorityFromVLAN bit is true, the switch priority is retrieved from the global VPRI_PRI_MAP table, which maps the 16 possible VPRI code into a 4-bit switch priority level.
- Otherwise, the per-port defaultPriority field in the PORT_CFG_PRI register is used.

If both the DSCP and VLAN priorities are enabled and present, there is a SwitchPriorityPrefersDSCP to break the tie. If 1b, the DSCP priority is preferred over VPRI.

12.5 Changing Priority

After this initial priority association, all three priority fields (VPRI, SWPRI, and DSCP) travel down the pipeline in parallel. Various units can modify one or more of these fields, which are eventually put into the outgoing VLAN tag, ISL tag, and DSCP field, if those tags or fields exist.

Via FFU

The FFU can inspect all three of the priority fields in the matching conditions for rules, and can execute commands to modify these fields independently. The encoding of the action is such that a single command can re-assign VPRI, switch priority, and DSCP together, but the same value is used for VPRI and switch priority. Compound actions or different rules can set VPRI and switch priority independently. See Section 6.0, "Filtering and Forwarding Unit (FFU)" for details.

The FFU can make very complex priority associations using all fields of the header, such as TCP port ranges, source subnets, and so forth. The priority association can be encoded in the VLAN tag, the ISL tag and/or the DSCP field for subsequent hops.

Via Policing

FFU rules can map traffic flows to one of four banks of counters or policers. If a bank is configured to police packets, it produces a tri-color marking (green/yellow/red) based on the state of its token buckets and configuration parameters. Based on the yellow or red markings, the POLICING unit optionally modifies the switch priority or the DSCP field but cannot change the VPRI field.

The DSCP modification is specified by the POLICER_DSCP_DOWN_MAP a 64 x 6-bit mapping table that specifies what the DSCP value is after downgrading. The ingress DSCP value indexes into the POLICER_DSCP_DOWN_MAP to produce a down graded egress DSCP value.

The ingress color is inferred from the POLICER_DSCP_DOWN_MAP table. Ingress DSCP values that downgrade back to the same value are originally red, ingress DSCP values that downgrade to red values are originally yellow, and ingress DSCP values that downgrade to yellow values are originally green.

Via Triggers

Near the end of the pipeline the TRIGGER unit can also modify the switch priority values. There is also special case handling of TRAP and control frames which can modify the switch priority. The final switch priority is used for congestion management and queue selection in the scheduler.

12.6 Transmitting Priority

The VPRI, Switch Priority and DSCP are sent to the MAC for transmission. The VPRI is translated into a new 4-bit VPRI using the TX_PRI_MAP table and is optionally encoded for transmission into the transmitted packet under the following conditions:

- If the packet was received tagged and needs to be transmitted untagged, the VLAN tag is deleted and the updated VPRI is not used.
- If the packet was received tagged and needs to be transmitted tagged, the VLAN tag is updated with the new VID, and the new VPRI is optionally updated.
- If the packet was received untagged and needs to be tagged, a VLAN tag is added using the new VID and new VPRI.

- If the packet was received untagged and needs to remain untagged, the new VID and new VPRI are not used.

The SWPRI is transmitted as part of the ISL tag if present.

The DSCP is used to update the DSCP field of IP frames.

The flag "WasTagged" indicates if the frame was actually received tagged and the PORT_CFG_1::AssumedUntagged is not set.

The flag "UpdatePri" is set to 1b if a VLAN tag is added, or if an existing VLAN is updated and the option MAC_CFG_1::EnableVPriUpdate is set.

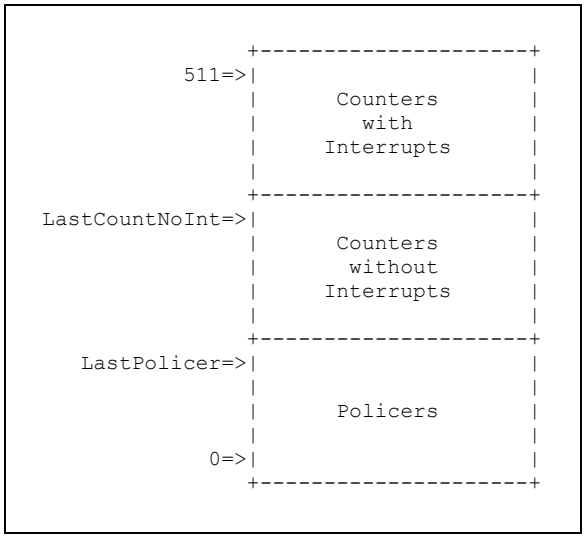
The flag "UpdateCfi" is set to 1b if a VLAN tag is added, or if an existing VLAN is updated and the option MAC_CFG_2::EnableVCfiUpdate is set (Note that CFI is named DEI in Q-in-Q).

The rules to add, update or delete a VLAN tag are detailed in Section 8.2.

12.7 Policing

FM4000 includes a policing unit which monitors the incoming traffic rate and can change the priority if needed. The policing unit includes four banks of 511 x 128-bit memory (note that valid indexes are from 1 to 511 while 0 is reserved as no action). Each bank can be split into three areas; a) policers (bottom of bank), b) counters without interrupts (middle of bank), c) counters with interrupts (top of bank).

The FFU may assign a bank/index pair to a specific action and can generate up to four indexes (one per bank) for a particular frame if there are multiple matches in the FFU and they hit different counter banks. The policing unit has the capability to handle the 4 actions simultaneously, one action per bank. This per-bank action (policing, counting, counting and interrupting) depends on the value of the index relative to the limits sets for this bank.



The counters without interrupts consist of two 64-bit counters; one counts bytes while the second counts frames. The counters with interrupts are identical but can optionally generate interrupts, provided the interrupt mask for this bank is reset.

The policers themselves are implemented using dual token buckets named “committed” and “excess.” Tokens are used from the committed bucket first. When the committed TB is depleted, packets conceptually are marked as yellow and uses the excess TB's tokens.

The policer uses either the switch priority or DSCP to retrieve a frame color (green, yellow, red), and uses the current state of the token bucket to assign a new color if needed. The selection between switch priority or DSCP is controlled per bank by the POLICER_CFG::IngressColorSource register field). The actual color is deduced by analyzing the down map.

- If there are 2 levels down, current color must be green.
- If there is only one level down, current color must be yellow.
- If there is no level down, the color is red.

The new color assigned can only be redder (green to yellow or red, yellow to red) than the incoming color, and can be used to change the switch priority or the DSCP field or both when the frame is transmitted, or to simply drop the frame. This is shown in Figure 12-2.

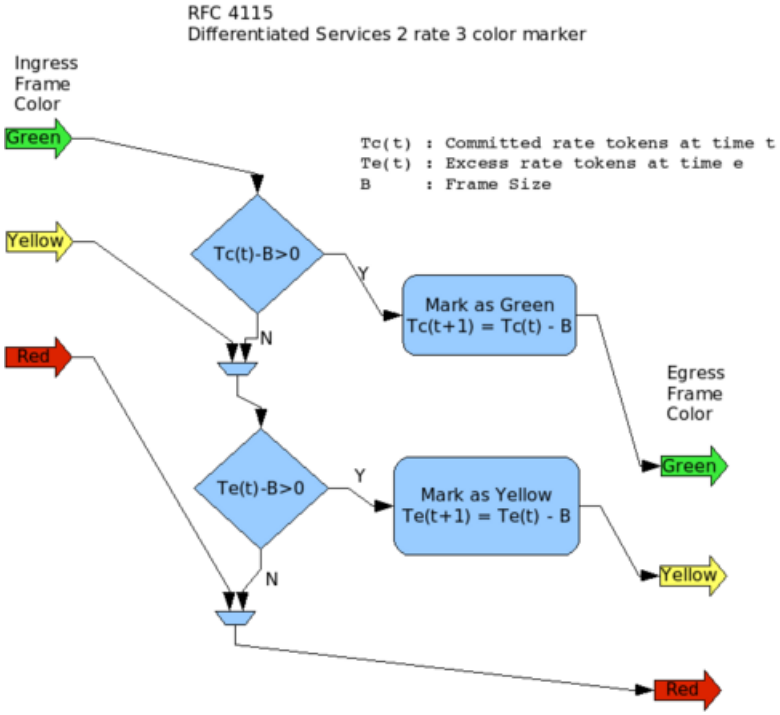


Figure 12-2 Differentiated Services Color Marking

The rules for combining the results from the four banks are detailed in the following list.

Note: Rules (1-4) are applied to obtain an egress DSCP and an egress Switch Priority color, then rules (5-6) are applied independently for DSCP and Switch Priority.

1. Each of the four banks of policers produces a color regardless of whether the policer was hit or not.
2. If a bank was hit and the MarkDSCP bit is set, the policer's output color is determined from the policer's coloring rules. Otherwise the policer's output color is assumed green.
3. The combined bank color is the lowest color of the four banks (most reddish).

4. The frame's egress color is only permitted to be equally or more reddish than the ingress color (for example, a yellow frame may egress yellow or red, but never green). The ingress color is from either DSCP, Switch Priority or assumed green as determined by IngressColorSource.
5. The egress DSCP or Switch Priority value is set to the egress DSCP or Switch Priority color resulting from rules (1-4) by looking up the POLICER_*_DOWN_MAP with the constraint in rule (6).
6. The egress DSCP or Switch Priority value's color can only be made equally or more reddish than the ingress value's color (For example, if the egress color from rules (1-4) is green, but the ingress color was Yellow, the egress value is unchanged and stays Yellow).

As an example, a green frame comes in and hits two banks. The first bank sets the color to red but only MarkDSCP is set for that bank, while the second bank sets the color to yellow and has MarkDSCP and MarkSWPRI both set. In that case the frame exits with an updated DSCP to red (worst case for DSCP) and an updated switch priority to yellow (worst color for switch priority).

Finally the disposition of the frame, forwarded or dropped, is controlled by each policer bank using the POLICER_TABLE[..]::PolicerCommitedAction and POLICER_TABLE[..]::PolicerExcessAction fields, where the drop action from any bank takes precedence over the forward action of any other bank.

The policer includes the following registers:

- **POLICER_TABLE** (4 x 512 x 128-bits)
  - Counter/policers rates.
  - In case of policers, defines the committed or excess rates and disposition of the frame if these rates are exceeded.
- **POLICER_CFG** (4 x 32-bits)
  - Defines partitioning of the bank between policers, counters without interrupts and counters with interrupts.
  - Defines if the switch priority or DSCP is used to retrieve current color.
  - Defines if DSCP downgrading is allowed or not.
  - Defines if switch priority downgrading is allowed or not.
- **POLICER_IP** (1 x 32-bits)
  - Contains interrupts pending.
- **POLICER_IM** (1 x 32 bits)
  - Masks interrupts or not.
- **POLICER_DSCP_DOWN_MAP**
  - Defines the next downgrade DSCP code from current DSCP (used only if DSCP downgrade is turned on).
- **POLICER_SWPRI_DOWN_MAP**
  - Defines the next downgrade switch priority from current switch priority (used only if switch priority downgrade is turned on).

The processing algorithm is the following:

- At the beginning of the frame:
  - Check token bucket against limit and apply color.

- At the end of the frame (or periodically with a frame of 0 length):
  - When tail of frame goes past, read previous tokens and timestamp value from POLICER_TABLE.
  - Decrement tokens value by frame size.
  - Compute tokens to add using the rate mantissa and exponent fields from the POLICER_TABLE.
  - Limit new_tokens to bucket capacity C.

```
new_tokens = (new_tokens > C)? C: new_tokens
```

  - Add new_tokens to tokens value.

12.8 Monitoring Memory

FM4000 is a shared memory fabric where the central packet memory is accessible to all ports on a first come first served basis. However, for congestion management purposes, FM4000 can partition the traffic into different shared memory partitions (SMP) using the traffic class to which the frame belongs, and monitors memory utilization for those SMPs. The switch can then be configured to trigger actions (PAUSE or DROP or DO NOT QUEUE) when certain watermarks are reached.

Note: The traffic class is obtained through the SWITCH_TO_CLASS register set, which associates a traffic class to each switch priority.

The switch contains 2 SMPs. The CM_SMP_MEMBERSHIP register is used to map the eight traffic classes into SMPs using the following assignment:

0 = Not part of any SMP

1 = SMP 0

2 = SMP 1

Any other assignment causes the frame to be dropped.

There are two views of the memory utilization; one “receive” view and one “transmit” view. The are shown in Figure 12-3 and Figure 12-4, respectively.

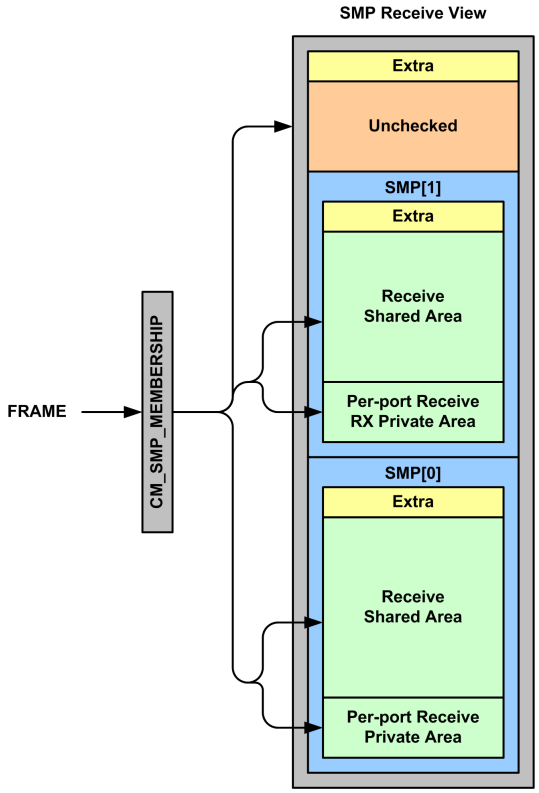


Figure 12-3 Shared Memory Partitioning - Receive

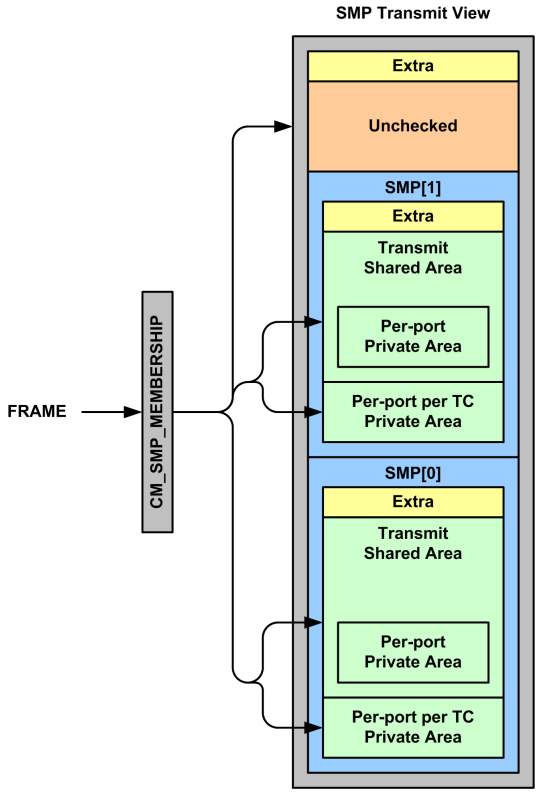


Figure 12-4 Share Memory Partitioning - Transmit

As shown, the incoming frame uses the CM_SMP_MEMBERSHIP register to determine which SMP to use (SMP 0, SMP 1 or unchecked). Once the SMP is known, the frame is checked against the different receive and transmit watermarks. If the frame is accepted, it is queued for transmission. The frame is accounted on the receive side (CM_RX_xxxx_USAGE) and on the transmit side (CM_TX_xxxx_USAGE) at the end of frame once the size of the frame is known. A multicasted frame is checked and accounted once on the receive side, and is checked and accounted for every possible destination on the transmit side.

The accounting is per 512-byte segment. The decision to accept a frame is made at the beginning of the frame, but the accounting is done at the end of frame. Thus, an "extra" area must be reserved for in-flight frames.

The CM_XXX_USAGE registers are used to track memory usage. They are incremented for frames that are scheduled for transmission (forwarded), and are decremented once the frame has been transmitted or flushed by the scheduler. A frame is disposed of when it has been successfully transmitted on all ports or flushed out of all possible transmit queues. Frames that are discarded before being queued are not accounted, as they do not take any memory space.

- **CM_GLOBAL_USAGE** — Tracks global memory usage. Updated for all forwarded frames.
- **CM_RX_USAGE[0..24]** — Tracks per-port memory usage. Updated for all forwarded frames.

- **CM_SHARED_SMP_USAGE[0..1]** — Tracks usage of receive shared area in each SMP. This per-SMP counter is only incremented once the per-port receive private area is exhausted, and is decremented by the portion of segments required to bring the per-port usage down to the private watermark for that port.
- **CM_RX_SMP_USAGE[0..24][0..1]** — Tracks usage of per-port per-SMP memory usage.
- **CM_TX_SMP_USAGE[0..24][0..1]** — Tracks memory usage per port per SMP on the transmit side. Updated for all forwarded frames transmitted to this port and associated with this SMP.
- **CM_TX_TC_USAGE[0..24][0..7]** — Tracks memory usage per-port, per-traffic-class. Updated for all destinations of each forwarded frames.

Watermarks are defined in number of 512-byte segments, and defines the threshold above which the action is taken. The switch offers the following watermarks:

- **CM_GLOBAL_WM** (1 x 16-bits) — Defines an upper-bound limit to memory utilization for new frames. This limit is typically set to the maximum memory minus the maximum frame size times the number of active ports. The CM_GLOBAL_WM register needs to be set sufficiently below the top of memory so that the device never requests more memory than it has. Requesting more memory than what is available can cause frame corruption, or potentially chip deadlock.
- **CM_SHARED_WM** (16 x 16-bits) — Defines the upper-bound limit to usage of shared memory per switch priority for new frames. This limit is typically set to the desired SMP size minus area for in-flight frames, and area for receive private areas.
- **CM_RX_SMP_PRIVATE_WM** (25 x 2 x 16-bits) — Defines the size of the per-port per-SMP receiver private area.
- **CM_RX_SMP_HOG_WM** (25 x 2 x 16-bits) — Defines the upper-bound limit to the receive queue size per-port per SMP.
- **CM_TX_SMP_PRIVATE_WM** (25 x 2 x 16-bits) — Defines the size the per-port per-SMP transmitter private area. Note that this excludes the per-port per-traffic class private area.
- **CM_TX_TC_PRIVATE_WM** (25 x 8 x 16-bits) — Defines the size the per-port per-traffic-class transmitter private area.
- **CM_TX_SMP_HOG_WM** (25 x 4 x 16-bits) — Defines the upper-bound limit to the transmitter queue size per-port per SMP. This excludes the per-port per-traffic class and the per-port per-SMP private area.
- **CM_TX_HOG_MAP** (8 x 2-bits) — Maps the traffic classes associated with the frame to one of the four hog watermarks associated with each port.
- **CM_RX_SMP_PAUSE_WM** (25 x 2 x 16-bits) — Defines a PAUSE ON and PAUSE OFF per-port per-SMP trip points to send PAUSE frames to an ingress port.
- **CM_SHARED_SMP_PAUSE_WM** (2 x 2 x 16-bits) — Defines the PAUSE ON and PAUSE OFF per-SMP trip points to send PAUSE frames to an ingress port.

Figure 12-5 shows the different watermarks and usage counters per port.

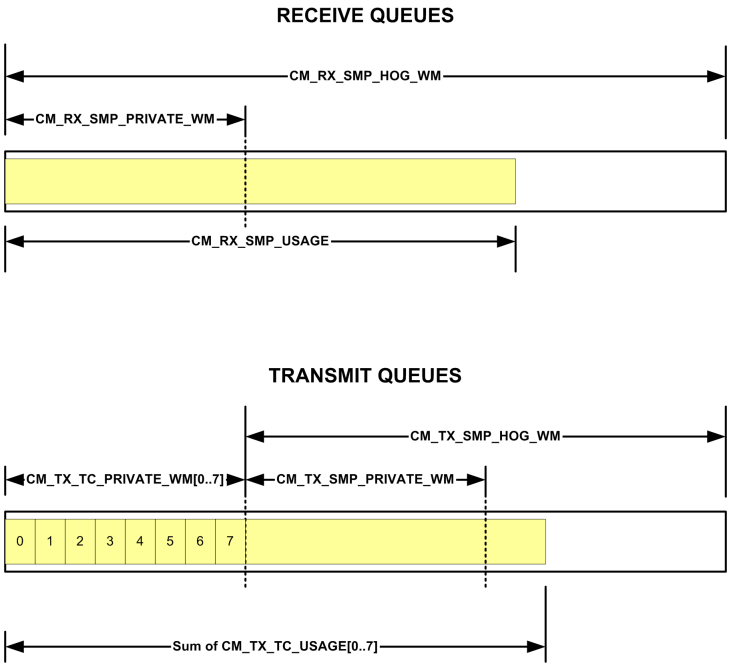


Figure 12-5 Receive and Transmit Queues Watermarks

Generally speaking, a frame is dropped if there is no private space available and the frame exceeds the hog or switch priority or global watermarks. Assumes the following logical expressions:

OverRXP = CM_RX_SMP_USAGE[port][smp] >= CM_RX_PRIVATE_WM[port][smp]

OverRXH = CM_RX_SMP_USAGE[port][smp] >= CM_RX_SMP_HOG_WM[port][smp]

OverRXS = CM_SHARED_SMP_USAGE[smp] >= CM_SHARED_WM[swpri]

OverTXTCP = CM_TX_TC_USAGE[port][tc] >= CM_TX_TC_PRIVATE_WM[port][tc]

OverTXP = CM_TX_SMP_USAGE[port][smp] >= CM_TX_SMP_PRIVATE_WM[port][smp]

OverTXH = CM_TX_SMP_USAGE[port][smp] >= CM_TX_SHARED_HOG_WM[port][TX_HOG_MAP(tc)]

OverG = CM_GLOBAL_USAGE >= CM_GLOBAL_WM

Then a frame is dropped if:

Drop = OverG | (OverRXP & OverTXTCP & OverTXP & (OverTXH | OverRXH | OverRXS))

Note: Note that some memory management registers cannot be changed on-the-fly while there is traffic being switched through the fabric, as the switch has internal states involving usage of private versus shared area.

The registers that need special update procedures are the following:

- **CM_SMP_MEMBERSHIP**
- **CM_RX_SMP_PRIVATE_WM**
- **CM_TX_TC_PRIVATE_WM**
- **CM_TX_SMP_PRIVATE_WM**

For those registers, the only safe time to change them is when the corresponding affected queues are empty. The proper procedure to update them requires the software to stop traffic and wait for the queue to drain using accelerated timeout before changing them. The method suggested is the following:

1. Set PORT_CFG_2 to 0x0 for all ports 0 through 24.
   - Writing PORT_CFG_2 to 0x0 causes all incoming frames to be dropped.
2. Set egress rate limiters to 0.
3. Write 0xFFFFFFFF to FRAME_TIME_OUT.
   - Writing a value to FRAME_TIME_OUT causes immediate marking of all “new” frames to “old,” and causes immediate deletion of any frames that were already marked “old.”
4. Poll CM_GLOBAL_USAGE periodically until seeing no change for a certain period.
   - The amount of time to wait is equal to the time to transmit the largest frame at the lowest supported speed with some margin.
5. Repeat steps 3-4 until the CM_GLOBAL_USAGE is 0.
6. Change watermarks as desired.
7. Restore frame timeout to original value.
8. Restore egress rate limiters.
9. Restore PORT_CFG_2.

12.9 Flow Control and Rate Limiters

FM4000 supports two types of PAUSE frames:

- IEEE 802.3 PAUSE frames — PAUSE frames which contain a single pause interval applicable to all traffic classes.
- CISCO Class Based PAUSE frames — PAUSE frames which contain a bit vector to identify which class to pause, and a 16-bit pause time for each class paused.

Pause frames can be sent for three conditions:

Rate-based Pause:

There are 2 rate limiters, one per SMP. The rate limiter monitors the incoming traffic rate for each SMP. They start transmitting PAUSE frames if a programmable rate threshold is exceeded, and stop transmitting PAUSE frames when the rate is reduced to below a second programmable threshold. Aside from user activation, the rate limiters can become active upon receiving a congestion notification frame from a remote switch.

Memory Usage-based Pause:

There is a per port pause based on CM_RX_SMP_PAUSE_WM. There is also a shared memory pause based on CM_SHARED_SMP_PAUSE_WM and the per port receive private memory usage.

A pause_on is sent if any of the rate based pause, per port pause or shared memory pause conditions evaluate to pause on. A pause_off is sent if all of the three pause conditions evaluate to pause_off.

Also, FM4000 has the capacity to react to reception of PAUSE frames. If it is a Cisco class-based PAUSE frame, the PAUSE frame indicates which class to pause and for how long. If it is an IEEE frame, all

classes are stopped for the prescribed time. There is also an 8-bit mask available, one bit per traffic class, to disable pausing on specific traffic classes.

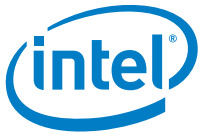
Finally, FM4000 also supports transmit rate limiters, which are detailed in Section 13.0, "Egress Scheduling and Shaping".

The following registers control the pause and rate limiter behavior:

- **CM_RX_SMP_PAUSE_WM** (25 x 2 x 32-bits) — Defines the limit in number of segments that each RX can use per SMP before starting or stopping PAUSE frames. The register contains an ON limit (pause is activated when the number of segments is above this limit) and an OFF limit (pause is deactivated when the number of segments is below or equal to this limit). The hardware assumes that the ON limit is greater than the OFF limit.
- **CM_SHARED_SMP_PAUSE_WM** (2 x 32-bits) — Defines the limit in number of segments of the global pool can be used per SMP before starting or stopping PAUSE frames. The register contains an ON limit (pause is activated when the number of segments is above this limit) and an OFF limit (pause is deactivated when the number of segments is below or equal to this limit). The hardware assumes that the ON limit is greater than the OFF limit. Ports that are using only their RX private memory will not get paused.
- **CM_PORT_CFG** (25 x 32-bits) — Defines configuration for that port:
  - RX PAUSE frame processing (in conjunction with SYS_CFG_1)
    - Incoming PAUSE frames are passed from the EPL to be processed in the Frame Handler, regardless of pause configuration. Pause frame is identified by Ethertype = 8808 and MAC address 01-80-c2-00-00-01.

      Note: The IEEE specifies that PAUSE frames be dropped if the Ethertype is 8808 or if the MAC address is 01-80-c2-00-00-01, but the FM4000 SYS_CFG_1:DropPause bit requires that both conditions be met. By using a combination of ACL's (to drop on PAUSE DMAC) and triggers (to drop on Ethertype 8808), the specified behavior can be implemented.
    - Process incoming RX PAUSE frames as IEEE 802.3 PAUSE frames.
    - Process incoming RX PAUSE frames as Class-based (CISCO-defined) PAUSE frames.
    - PAUSE decimator to use by this port
    - Traffic class that can be paused.
    - Action per rate limiter; drop or pause
    - Rate limiter index
- **CM_PAUSE_RESEND_INTERVAL** (1 x 16-bits) — Defines the PAUSE resend interval in quanta.
- **CM_PAUSE_DECIMATOR** (8 x 32-bits) — Defines up to 8 PAUSE decimators. The decimator registers define the clock ratios between the internal clock and the PAUSE tic. The support of eight decimators allows the switch to support up to eight different data rates.
  - The value stored in the decimator is equal to 512 * FH_CLK / PORT_SPEED /25 and is expressed as (K+1) / (1 + M/(N+1)).
  - Example 1, link at 10 GbE with FH at 375 MHz:
    - Q = 512 * 375 MHz / 10 GbE / 25 = 0.768 = 1/1.302 = ~1/(1+3/10)
    - K = 0
    - M = 3
    - N = 9

  - Example 2, link at 1 GbE with FH at 360 MHz:
    - Q = 512 * 360 MHz / 1 GbE / 25 = 7.37 = ~8 / (1 + 1/12)
    - K = 7
    - M = 1
    - N = 11
- **RX_RATE_LIM_USAGE** (25 x 2 x 32-bits) — Defines the current RX rate limiter counter. This value is stored as a fixed decimal point number (24 bits units and 8 bits fraction). The value is decremented whenever a frame is received by the frame size received if and only if the frame had a good CRC and was not dropped. The value is unconditionally incremented every 25 frame handler clock cycles by the fixed value stored in the RX_RATE_LIM_CFG up to the limit set in that same register.
- **RX_RATE_LIM_CFG** (25 x 2 x 32-bits) — Defines the token bucket parameters for each rate limiter. It contains the following fields:
  - Saturation point (Capacity) — Defined in 64-byte units.
  - Fixed point number (8 bits for header, 7 bits for fraction) representing the number of bytes to add to the token bucket every 25 frame handler clocks.
    - As an example, for FH at 375 MHz a rate of 1 Gb/s is equivalent to:
      - bytes/period = 1 Gb/s * 25 / 375 MHz / 8 = 8.333 = 8 + 85/256
      - units = 8
      - fraction = 85
    - The minimum rate supported for 375 MHz is:
      - units = 0
      - fraction = 1
      - rate = (1/256) * 375 MHz * 8 / 25 = 0.46 Mb/s
- **CN_RATE_LIM_CPID** (2 x 25-bits) — Indicates if a rate limiter is active for each SMP and each port.
- **TX_RATE_LIM_USAGE** (25 x 8 x 32-bits) — Defines the current TX rate limiter counter. Detailed in Section 13.5.
- **TX_RATE_LIM_CFG** (24 x 32-bits) — Defines the token bucket parameters for each rate limiter. It contains the following fields:
  - Saturation point (Capacity) — Defined in 64-byte units.
  - Fixed point number (8 bits for header, 7 bits for fraction) representing the number of bytes to add to the token bucket every 25 frame handler clocks.
    - As an example, for FH at 375 MHz a rate of 1 Gb/s is equivalent to:
      - bytes/period = 1 Gb/s * 25 / 375 MHz / 8 = 8.333 = 8 + 85/256
      - units = 8
      - fraction = 85
    - The minimum rate supported for 375 MHz is:
      - units = 0



— fraction = 1

— rate = (1/256) * 375 MHz * 8 / 25 = 0.46 Mb/s

The receiver flow control algorithm is the following:

1. At beginning of a frame:
   a. Check CM_XXX_USAGE against watermarks (as detailed in previous section) and forward or discard frame accordingly.
   b. Check if the current RX_RATE_LIM_USAGE is smaller than the drop level defined in the RX_RATE_LIM_THRESHOLD::DROP register field and drop the frame if the current usage is lower than this value.
2. At end of frame:
   a. Update the CM_XXX_USAGE registers.
   b. The switch priority associated with that frame is used to index the CM_SMP_MEMBERSHIP and retrieves the SMP associated with that frame.
   c. The RX_RATE_LIM_USAGE is decremented by the size of the frame regardless of whether the frame has a valid CRC or not (the counter can go negative).
   d. The port pause state is set ON if any of the following conditions are met:
      - The new rate limiter usage is smaller than 0 and the rate limiter pause is enabled for that SMP (CM_PORT_CFG::SMP{0,1}_RL_PAUSE).
      - The CM_SHARED_SMP_USAGE is greater than or equal to CM_SHARED_SMP_PAUSE_WM::PauseOn.
      - The CM_RX_SMP_USAGE is greater than or equal to CM_RX_SMP_PAUSE_WM::PauseOn.
   e. The port's pause condition remains on as long as an off condition is not met.
3. Periodically (every 25 frame handler clocks):
   a. The RX_RATE_LIM_USAGE::Capacity is incremented by a fixed-point decimal number defined in RX_RATE_LIM_CFG register and saturates at the limit defined in the same register.
   b. If the rate limiter was active and the capacity goes above the PAUSE_OFF watermark defined in the RX_RATE_LIM_THESHOLD, the rate limiter is de-activated and the pause is canceled if the PAUSE off watermarks have not yet been reached.
   c. Resend PAUSE fames periodically as long as either a rate limiter or congestion is present.
4. When the frame transmission is completed:
   a. Update the CM_XXX_USAGE registers.
   b. The switch priority associated with that frame is used to index the CM_SMP_MEMBERSHIP and retrieve the SMP associated with that frame.
   c. The port pause state is set OFF if the pause state was ON, and all of the following condition are met:
      - The CM_SHARED_SMP_USAGE is greater than or equal to CM_SHARED_SMP_PAUSE_WM::PauseOn.
      - The CM_RX_SMP_USAGE is greater than or equal to CM_RX_SMP_PAUSE_WM::PauseOn.
      - The rate limiter is still active.

12.10 Congestion Notification

FM4000 supports congestion notification (CN), allowing the switch to transmit a CN frame toward the sources during congestion. Two types of frames can be generated:

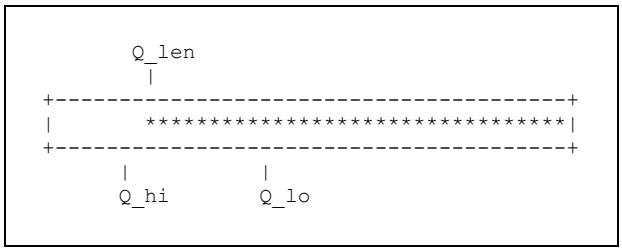
- A multicast frame which contains the state of each queue:
  - This frame is called the VCN frame for Virtual-output-queue Congestion Notification frame. The frame is transmitted each time the congestion status changes. The frame is multicasted back to the end points, which may choose to change their transmission rate and traffic partitioning upon reception of this frame.
  - This method is particularly useful to implement a lossless virtual output queue fabric.
- A unicast frame sent back to the source port:
  - This frame is called the FCN frame for Fractional Congestion Notification frame. The frame is a multi-hop back-pressure notification frame sent from the congestion point back to the source port within the current ISL domain.
  - The frame is trapped at the source port at the boundary of the ISL domain and triggers transmission of a PAUSE frame, where the PAUSE interval is derived from the level of congestion at the congestion point.

The register set defined to support this function are:

- **CN_GLOBAL_CFG_1** and **CN_GLOBAL_CFG_2** (2 x 32-bits) — Defines global congestion notification parameters:
  - Mode of operation (FCN, VCN).
  - Sample period (FCN).
  - Ethernet type for CN frames.
  - Source GloRT for FCN frames. A value of 0 is replaced by the PORT_CFG_ISL[0]::srcGlort.
  - Queue unit size (power of 2).
- **CN_SAMPLE_CFG** (1 x 32-bits) — Defines the sampling parameters globally.
- **CN_SMP_THRESHOLD** (25 x 2 x 32-bits) — Defines the high (Q_hi) and low (Q_low) thresholds per SMP to emit a positive and a negative congestion notification frame.
- **CN_SMP_CFG** (25 x 2 x 32-bits) — Defines the equilibrium threshold (used for FCN).

12.10.1 Virtual Output Queues Congestion Notifications

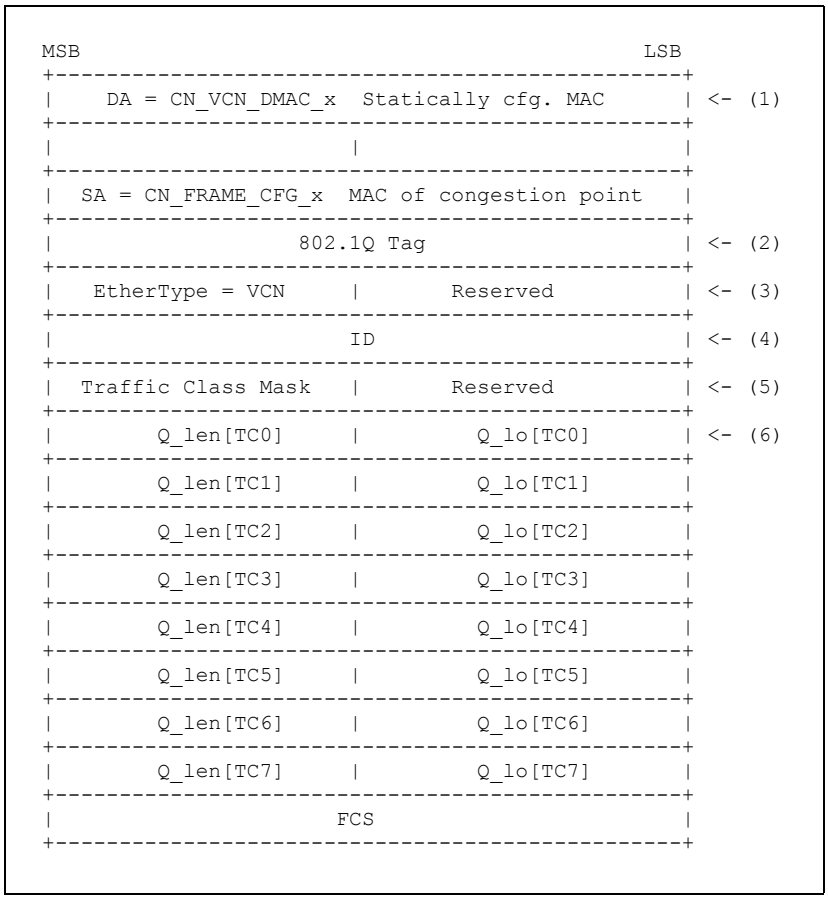
FM4000 implements loss-less virtual output queues flow control by generating VCN flow control frames when a queue reaches a certain depth. The switch tracks utilization per-traffic class and per-port and uses thresholds to trigger transmission of congestion notifications frames to end points. The queue parameters are shown here:



The actual Q_LEN is the CM_TX_TC_USAGE[x][c] register values (per port "x" per traffic class "c"). The Q_HI an Q_LO thresholds are stored in the CN_SMP_THRESHOLD registers (per port per SMP). A queue is declared congested when Q_LEN crosses the Q_HI watermark (going from below Q_HI to above or equal to Q_HI). A queue becomes uncongested if it was congested and the Q_LEN crosses the Q_LO watermark (going from above or equal to Q_LO to below Q_LO).

The switch emits a VCN frame whenever any of the 8 queue status is changed (becomes congested or becomes uncongested), and also emits a VCN frame periodically if any queue remains congested (the period is defined in the CN_GLOBAL_CFG_1 register).

The VCN frame format is shown next.



Notes:

1. The multicast address is globally configured by the registers CN_VCN_DMAC_1 (common 32 bit prefix for all ports) and CN_VCN_DMAC_2 (16-bit suffix individually programmed for each port).
2. The 802.1Q tag is sourced from the register CN_FRAME_CFG_2 (field VLAN_PRI and VLAN_ID).
3. The EtherType is defined in CN_GLOBAL_CFG_1::EtherType and is user configurable.
4. ID[4:0] contains the egress port number for which this VCN frame is sent.
5. Traffic Class Mask:
   - Bit i set => Pause traffic class i
   - Bit i cleared => Unpause traffic class i
6. Q_len[TCi] = CM_TX_TC_USAGE[TCi] * SEGMENT_SIZE / 8. This value can be used or not depending on how smart intelligent subscriber cards are.

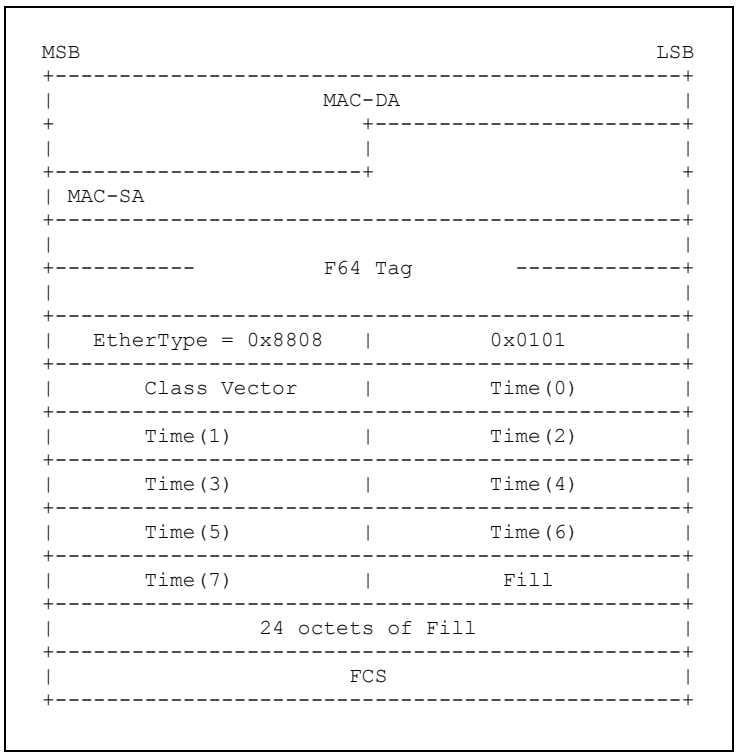
12.10.1.1 VCN Frame Reaction

PAUSE per traffic class is essentially limited to PAUSE per Shared Memory Partition, since traffic classes are associated with one of the two SMP's. Because of this, VCN-based congestion management is best implemented with a FM4000 switch in tandem with a class-based PAUSE-capable NPU. In this mode the FM4000 ingress ports are programmed to pass the VCN frames on to an external device. See the "Congestion Management Application Note" for details on this.

12.10.2 Class-Based Pause Frames

Note: This is a standard implementation of Priority Flow Control (PFC) that has been defined by the IEEE. At the ingress, the switch can generate Class-based Pause frames based on how traffic classes are assigned to the two shared memory partitions. If a memory partition fills past a programmed watermark, pause frames are generated for all traffic classes assigned to that memory partition. At the egress, the switch can respond to Class-based Pause frames with a traffic class assigned to each of eight egress queues.

The Class-based Pause frame format is shown here:



The special fields are:

- **Class Vector** — Map to describe which timers are valid (LSB = e[7..0], e[n] corresponds to time[n])
- **Time(n)** — The pause time measured in units of pause quanta equal to 512 bit times.

12.10.3 Fractional Congestion Notifications

Note: This is a pre-standard and simplified implementation of Congestion Notification that is being defined by the IEEE. There are also limitations on its use that are dependent on usage conditions. Contact Intel for details.

The switch implements intelligent congestion notification using a probability method. In this mode of operation a configurable fraction of frames are sampled when a configurable queue length is exceeded. The switch proceeds as follows:

At the congestion point:

The switch checks the queue length (CM_TX_SMP_USAGE) when a frame is queued. If the queue size exceeds a certain threshold (CN_SMP_CFG::EQ), the switch potentially samples this frame (the fraction of frames that are sampled is defined by a sampling probability defined in the CN_SAMPLE_CFG) and generates an FCN frame back to the source.

At the reaction point:

All switches can be configured (CN_RATE_ACTION_MASK and CN_FORWARD_MASK) to either trap FCN frames, forward them or discard them. If they are trapped, the switch activates one of the rate limiters to slow down the source. The rate limiter selected depends on the content of the CPID of the frame received. If the FCN frames are forwarded, the switch simply forwards the frame to the port without taking any further action.

Note: The rate limiter operating parameters are the ones predefined for the associated SMP and port.

This is illustrated in Figure 12-6. The incoming traffic (yellow frames) get congested at switch "C". This switch generates FCN frames back to the source by sampling incoming traffic. The switches "C" and "B" are configured to forward frames when they detect an FCN frame. The switch "A" is configured to trap the frame and activate a rate limiter, which generates a PAUSE frame back to the source.

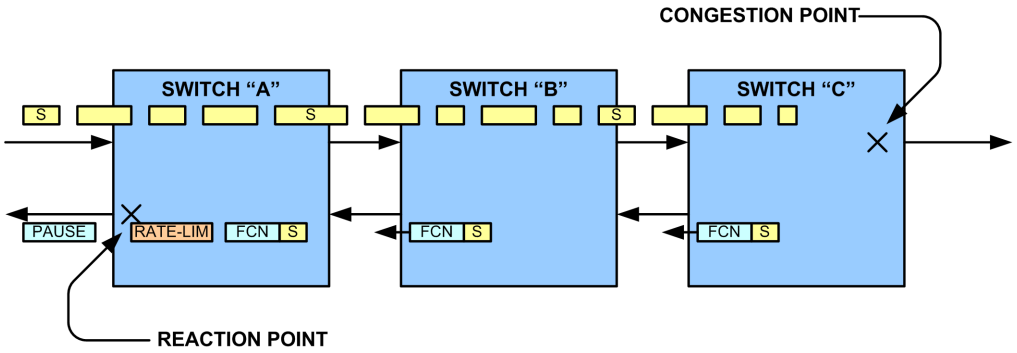
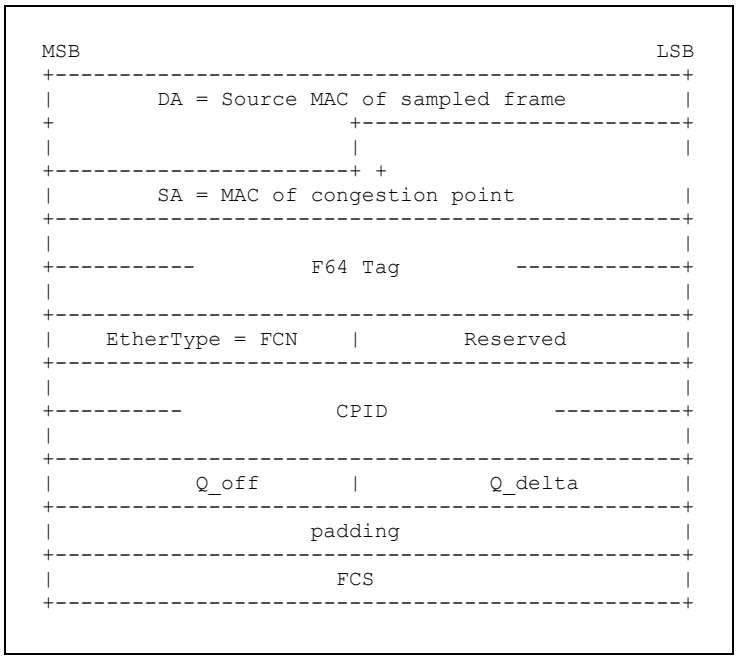


Figure 12-6 Congestion Notification

The FCN frame format is shown here:



The fields are:

- **DA** — Equal to the source address of the sampled frame.
- **SA** — Equal to the source address of the congestion point (defined in CN_FRAME_CFG for this port).
- **F64 Tag** — The destination GloRT is from the sampled frame's source GloRT. The source GloRT is from CN_GLOBAL_CFG_2::CNSrcGlort.

  If sampled frame had VLAN tag then:

  — ISL VLAN_PRI, CFI = CN_FRAME_CFG.Priority

  — ISL VLAN = VID from sampled frame

  Otherwise sampled frame had no VLAN tag, so:

  — ISL VLAN_PRI, CFG = 0

  — ISL VLAN = 0
- **EtherType** — The Ethernet type is defined in CN_GLOBAL_CFG_1::EtherType register.
- **Reserved** — Set to 0.
- **CPID** — Detailed below.
- **Q_OFF** — The difference between the equilibrium point (defined in CN_SMP_CFG) and the current queue length (CM_TX_SMP_USAGE). Saturates at – CN_SMP_CFG::EQ only. Does not saturate in the positive direction.
- **Q_DELTA** — The difference between the current CM_TX_SMP_USAGE and the previously sampled CM_TX_SMP_USAGE. Saturates at 2 * CN_SMP_CFG.EQ and – 2 * CN_SMP_CFG.EQ.

The format of the CPID is shown here:

CPID 64 bits			
Switch ID (48 bits)	Egress Port (6 bits)	Ingress Port (6 bits)	SMP ID (4 bits)

with the following fields:

- The switch ID is equal to the Source Address
- The egress port is the egress port which experiences the congestion (congestion point)
- The ingress port is the port from which the sampled frame comes
- The SMP ID is the SMP domain

12.10.3.1 FCN Rate Reaction

When a FCN frame arrives at a FM4000 device, and the device is configured to process the FCN frame, the ingress rate limiter is activated to pause the source port for a random period of time.

The backoff duration is determined by the following algorithm:

```
FCN(Q_off,Q_delta) frame arrives on smp s and is destined for port p

if (Qoff < 0) {
  q[9:0] = (-Qoff>=1024?) 1023: -Qoff[9:0]
  r_time = (LFSR[9:0] * q) * 2 ^ CN_FB_CFG.FCN_BF[3:0]
  timer = min(CN_BACKOFF_BYTETIME, r_time)

  //store timer value into token bucket usage register
  RX_RATE_LIM_USAGE[p][s] = -1 * timer
}
```

The backoff duration is effectively the length of time needed to refill the token bucket, RX_RATE_LIM_USAGE[p][s] at the configured rate.

12.11 Backward Congestion Notification

Note: This is a pre-standard and simplified implementation of Congestion Notification that is being defined by the IEEE. There are also limitations on its use that are dependent on usage conditions. Contact Intel for details.

The BCN includes the following functions:

- CPID Cache.
- Support reception, deletion and transmission of rate limiter tags.
- Detection of rate limiter tags at congestion point and transmission of BCN to increase bandwidth.
- Computation of the rate limiter parameters (accrual rate and capacity) from level of congestion

The extensions are enabled when the CN_GLOBAL_CFG::Mode is set to 3.

The process used is as follows:

At the congestion point:

The switch checks the queue length (CM_TX_SMP_USAGE) when a frame is queued. If the queue size exceeds a configured threshold (CN_SMP_CFG::EQ), the switch potentially samples this frame (using the sampling probability defined in the CN_PORT_CFG) and generates a BCN frame back to the source.

If the queue size is lower than a configured threshold (CN_SMP_CFG::EQ), the switch checks if this frame carries a valid RLT tag which identified this congestion point. If yes, the switch samples the frame (using the sampling priority defined in the CN_PORT_CFG) and generates a BCN frame back to the source.

The BCN frame generated has a DA equal to the SA of the sampled frame, and contains information about the source point and the status of the queue, as well as selected fields from the sampled frame. The BCN frame is a layer 2 frame which is switched back to the source up to the reaction point where it trips a rate limiter. The BCN frame are not layer 3 and cannot be routed. If the DA is equal to a routing point, the switch can be configured to trap the BCN to the CPU or silently discard it.

At the reaction point:

All switches can be configured (CN_RATE_ACTION_MASK & CN_FORWARD_MASK) to either trap BCN frames, forward them to the egress port or delete them. If they are trapped, the switch activates one of the rate limiters to slow down the source. The rate limiter selected depends on the content of the CPID of the frame received. If the BCN frames are forwarded, the switch simply forwards the frame to the port without taking any further action.

Note: The rate limiter operating parameters are computed from the BCN parameters at the reaction point using queue status information from the frame received.

If a rate limiter is active because of reception of a BCN frame, all future frames coming from this port receive an RLT tag when they exit the switch (provided RLT is enabled on this port). The RLT tag is used at the congestion point.

Figure 12-7 illustrates the concept.

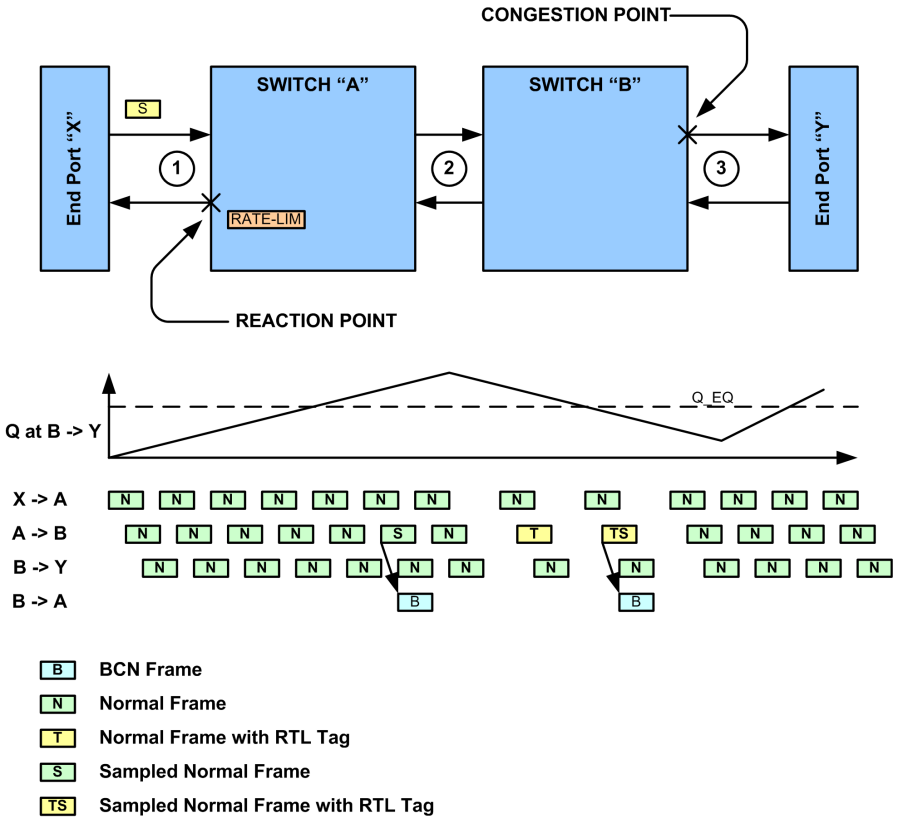
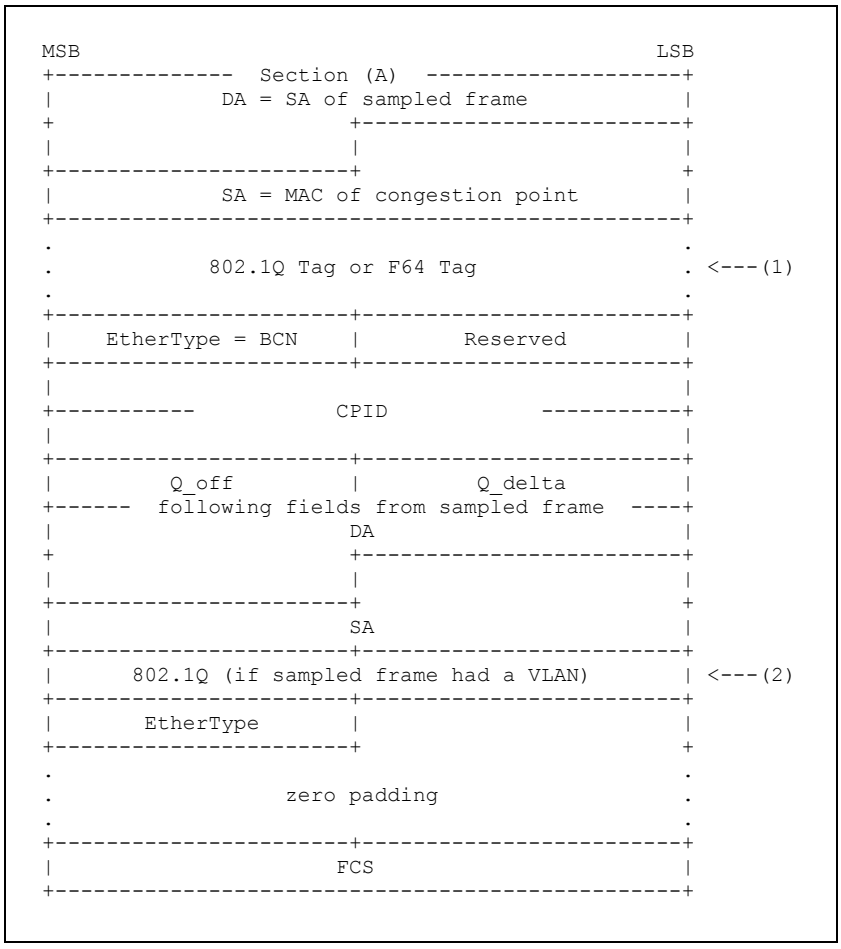


Figure 12-7 Backward Congestion Notification

The BCN frame format is shown here:



1. The BCN frame is always generated with an ISL tag however on egress, the ISL tag may be stripped depending on the EPL configuration. BCN frames can only be generated with Intel ISL tag or with standard VLAN tag without tag. Other tags are not supported.
2. If the sampled frame had VLAN_PRI in the ISL tag or if the sampled frame had 802.1Q tag. Otherwise this field is 0.

The fields are:

- **DA** — Equal to the source address of the sampled frame.
- **SA** — Equal to the source address of the congestion point (defined in CN_FRAME_CFG for this port).
- **F64 Tag** — The destination GloRT is from the sampled frame's source GloRT or zero if the sampled frame was not ISL tagged. The source GloRT is from CN_GLOBAL_CFG_2.CNSrcGlort.

  If sampled frame had VLAN tag then:

  — ISL VLAN_PRI, CFI = CN_FRAME_CFG.Priority

  — ISL VLAN = VID from sampled frame as seen by the EPL (pre-association stage)

  Otherwise sampled frame had no VLAN tag, so:

  — ISL VLAN_PRI, CFG = 0

  - ISL VLAN = 0
- **BCN EtherType** — TBD
- **Reserved** — Set to 0
- **CPID** — Detailed below.
- **Q_OFF** — The difference between the equilibrium point (defined in CN_SMP_CFG) and the current queue length (CM_TX_SMP_USAGE). Saturates at CN_SMP_CFG::EQ and – CN_SMP_CFG::EQ. If CM_TX_SMP_USAGE exceeds CN_SMP_CFG::SC then Q_OFF is zero.
- **Q_DELTA** — The difference between the current CM_TX_SMP_USAGE and the previously sampled CM_TX_SMP_USAGE. Saturates at 2 * CN_SMP_CFG::EQ and – 2 * CN_SMP_CFG::EQ. If CM_TX_SMP_USAGE exceeds CN_SMP_CFG::SC then Q_DELTA is zero.
- **BCN Payload**:
  - **DA** — Destination MAC in sampled frame.
  - **SA** — Source MAC in sampled frame.
  - **802.1Q tag** — If sampled frame had VLAN, the VID is as seen by the EPL (pre-association stage).
  - **EtherType** — EtherType in sampled frame.

The format of the CPID is shown here:

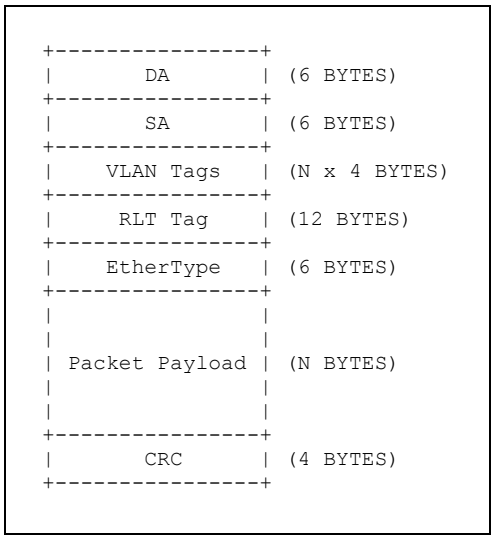
CPID 64 bits			
Switch ID (48 bits)	Egress Port (6 bits)	Ingress Port (6 bits)	SMP ID (4 bits)

with the following fields:

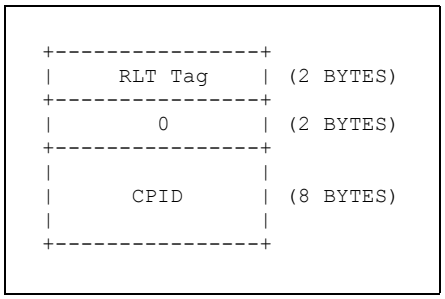
- The switch ID is equal to the Source Address.
- The egress port is the egress port which experience the congestion (congestion point).
- The ingress port is the port from which the sampled frame is coming from.
- The SMP ID is the SMP domain.

Note: The FM4000 has only 2 SMP domains, SMP ID bits 3:1 are thus ignored.

The RTL tag is added by the egress port immediately after the VLAN headers as shown here:



The RLT tag is 8 bytes long and is show here:



12.11.1 BCN Congestion Point

12.11.1.1 Sampling

Each frame passing through a egress port and SMP pair contributes to a byte counter in the Frame Handler. When the counter decrements to zero due to outgoing frames, a flag is set in the Frame Handler to indicate that the next frame is sampled for BCN. The counter is then re-initialized to a random value depending on CN_PORT_CFG.SampleMin and CN_PORT_CFG.SampleJitter.

12.11.1.2 BCN Frame Generation

After a frame has been sampled, a BCN frame may or may not be generated, depending on the contents of the sampled frame.

If the sampled frame contains an RLT:

- a BCN frame is generated if CN_SMP_CFG::EQ or CN_SMP_CFG:SC has been exceeded.
- a BCN frame is generated if CN_SMP_CFG::EQ has not been exceeded and the CPID in the RLT matches the CPID of this egress queue after masking with CN_CPID_MASK.

If the sampled frame had no RLT:

- a BCN frame is generated if CN_SMP_CFG::EQ or CN_SMP_CFG::SC has been exceeded.

12.11.2 BCN Reaction Point

12.11.2.1 BCN Frame Reception

When a BCN frame arrives, the forwarding destination mask is computed through the frame lookup. Depending on the destination mask and the configuration of CN_FORWARD_MASK and CN_RATE_ACTION_MASK, a rate limiter needs to be activated if the CPID in the BCN frame is not in the CPID cache. Otherwise the current state for the rate limiter is updated.

If a rate limiter needs to be activated, the CPID in the BCN frame is written to an entry CPID cache (CN_CPID_TABLE[i], CPID_i), and CN_RATE_LIM_CPID is written with the index (i) into the CPID cache.

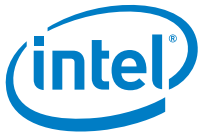
If the CPID is already in the CPID cache, the rate limiter's parameters are updated according to the following algorithm.

```
function compute_Fb(frame f) =
  if (CN_GLOBAL_CFG_2.fbQDelta) {
    Fb = (f.Q_off<<3) - (f.Q_delta<<3)<<CN_FB_CFG.BCN_W
  } else {
    Fb = f.Q_off<<3
  }
  return Fb

Note: Fb is normalized to 512 byte segments by the above function.
Note: The variable R shown below refers to the RX_RATE_LIM_CFG.RATE register.

---Initial state for the rate limiter---
backoff_mode = 0

--- Whenever a new BCN frame arrives---
bcn = receive_bcn() //BCN frame received
if bcn == BCN(0,0) {
  if (backoff_mode = 0) {
    tokens = -random(0, CN_BACKOFF_BYTETIME.Time)
    R = CN_RATE_LIM_CFG.Min
    backoff_mode = 1
  } else {
    tokens = -random(0, CN_BACKOFF_BYTETIME.Time)
    R = R / 2
  }
} else {
  Fb = compute_Fb(recv_frame)
  if (Fb == 0) {
```



```
      //do nothing, no rate change
    } else if (Fb < 0) {
      //rate decrease
      R' = R * ( 1 - (|Fb| * 2 ^ CN_GLOBAL_CFG_1.BCN_GD))
      //decrease limited to minimum rate
      R = max(R', CN_RATE_LIM_CFG.Min)
      //assign new owner CPID for this rate limiter
      currCPID = bcn.CPID
    } else {
      //rate increase, Fb>0
      //increase rate only if owner is asking for increase
      if (bcn.CPID == currCPID) {
        if (backoff_mode == 1) {
          backoff_mode = 0
          //restore rate to min, unless already above min
          R = max(R, CN_RATE_LIM_CFG.Min)
        }
        R' = R + (Fb * Ru * 2 ^ CN_GLOBAL_CFG_1.BCN_GI)
        //rate increase saturates at max rate
        R = min(R', CN_RATE_LIM_CFG.Max)
      }
    }
  }
  ---During token bucker update---
  When updating token bucket state with refill tokens,
  if (backoff_mode = 1 and tokens goes above 0) {
    backoff_mode = 0
  }
```

12.11.2.2 Rate Limiter Tagging

A rate limiter is considered active if the configured rate is less than the CN_RATE_LIM_CFG.Max value.

Whenever a frame goes through an active rate limiter, an RLT is added that corresponds to the CPID for that rate limiter.

The RLT is removed when a RLT frame goes out a port that has MAC_CFG_2.StripRLTOnEgress set.

12.12 Queue Delay Measurement

FM4000 includes a Queue Delay Measurement feature to permit network operators to determine if and by what amount frames are being delayed at each switch within the network. Unlike simple end-to-end measurements, this may permit a problem to be isolated to a handful of choke points in the network topology.

Operators may specify up to two transmit queues to monitor per chip. Each monitor can be configured to either perform exponential-weighted decay averaging or peak tracking. In either case, delay is measured as the elapsed wall-time from when frames enter the monitor transmit queue to when the same frames drain from the queue. This ensures that the estimate is reliable even if low level flow control modulates the rate at which the queue is drained.

The functional behavior is as follows:

- Up to two per-tx, per-tc queues can be monitored simultaneously.
- For each monitored queue, one frame is tracked at a time.

- If configured to track peak values, the newest delay measurement replaces the previous recorded one if the delay is greater in the newer estimate than in the older one.
- If configured for an exponential-weighted average, the newest delay measurement is added to the previous average according to the following equation:

```
Dn + 1 = Dn + (FrameDelay - Dn) > > QDM_CFG::w
```

- If Dn is zero, Dn + 1 = FrameDelay.
- The internal precision of the counter is measured in \frac{1}{2^{12}} tick increments.
- Idle time does not weigh into the average.
- When a sampled frame's latency exceeds the available representation, a bit denoting that this has occurred is set and the latency is assume to have been the maximum representable interval (231 ticks).
- For the purposes of the measurement, latency is taken as the time from when the head of a frame enters the tx queue until the time when the head of a frame leaves the tx queue. Under IP Multicast, the replication of frames delays the departure of later copies. This latency is not tabulated. The latency cost of IPMS traffic appears only to the extent that it creates congestion in the outbound tx queue and induces additional latency on subsequent frames.

The following registers are used to access this feature:

- **QDM_CFG** — Used to activate QDM functionality on a specific TX,TC pair, and set the parameters of the averaging.
  - — A tx field denoting the egress port to which the measurement apparatus should bind
  - — A tc field denoting the queue of the egress port to which the measurement apparatus should bind
  - — A w field denoting the weighting value to use for the exponential-weighted averaging (EWA)
  - — A mode field denoting whether to use EWA (mode 0) or Peak tracking (mode 1)
- **QDM_FRAME_CNT** — Used to specify for how many frames the measurement is to take place. A value of 0x0 disables the queue delay measurement. A value of 0xFFFF enables perpetual sampling. The value in the register is updated in hardware as frames are sampled.
- **QDM_INSTRUMENT** — Contains a 31-bit value denoting the computed queue delay. The time-base is measured in FH clock ticks. The 32nd bit is set if an overflow occurs during the measurement process. The management CPU is expected to clear the register at the beginning of any new sampling.

§ §

13.0 Egress Scheduling and Shaping

13.1 Introduction

FM4000 egress scheduling provides the following features:

- Eight Traffic Classes
  - — Eight independent egress scheduling queues per port.
- Per-port and Per-class Pause Support
  - — Support for standard IEEE 802.3 pause frames, which temporarily halts the scheduling of frames to an entire egress port.
  - — Support for Cisco-defined class-based pause frames, which temporarily halts the scheduling on a per-class basis.
- Strict Priority Scheduling
  - — Strict-high classes are scheduled before all frames of lower priority.
  - — Strict-low classes are scheduled only if all higher classes have no frames to send.
- Deficit Weighted Round-Robin (DWRR) Scheduling
  - — Each traffic class is guaranteed a minimum bandwidth allocation.
  - — Supports Rate Controlled Priority Queuing, where lower-priority classes are scheduled only if all higher-priority classes either have no frames to send or have exceeded their minimum bandwidth guarantees.
- Traffic Shaping
  - — Egress bandwidth of each traffic class can be limited to some maximum limit.
  - — Queuing delay, not discard, is used to enforce the bandwidth limits.
- Group-Based DWRR Scheduling
  - — Multiple traffic classes can be mapped to the same bandwidth shaping group.
  - — DWRR bandwidth guarantees and shaping limits apply to groups, not the individual classes.
  - — Within a group, higher-priority classes are strictly preferred over lower-priority classes.

13.2 Definition of terms

Table 13-1 Definition of Terms

Quantity	Notation	Definition
Traffic Class	c	Identifies one of eight egress queues from which the frame is scheduled. Also known as "fabric priority" and "scheduler priority". Mapped from the frame's switch priority (via the SWITCH_PRI_TO_CLASS table.) The traffic class numeric value is significant when multiple classes are allocated to the same DRR group. In that context, higher numbered classes are strictly preferred over lower numbered classes.
Scheduling Group	i	A set of traffic classes that share a DRR minimum bandwidth allocation. Groups are defined as contiguous sets of traffic classes. That is, for $i = \{c_1, c_2,...c_n\}$, $c_2 = c_1+1$, $c_3 = c_2+1$, etc.) In the absence of strict-high prioritized frames, the total egress bandwidth over these classes is guaranteed to never fall below the minimum limit (Q_i), assuming the group stays collectively backlogged. Like bandwidth shaping groups, DRR groups are defined as a series of contiguous traffic classes.
Bandwidth Shaping Group	g	A set of egress traffic classes for a given port which share a common maximum bandwidth allocation. The total egress bandwidth over these classes is guaranteed to never exceed the group's upper-bound limit (UB_g). Shaping groups are defined by a class-to-group (many-to-one) mapping. Although the hardware supports a fully general mapping, shaping groups should be configured to be supersets of scheduling groups.
Priority Set	s	Contiguous traffic classes of equal priority form a priority set. A priority set is preferred in scheduling if its traffic class members have higher numeric values than those of another priority.
Strict Priority	SP_c	An attribute configured per traffic class. When set, the traffic class effectively has infinite DC_g, preempts lower-priority classes, and is preempted by higher-priority classes. The strict priority attribute must be uniformly set within each priority set. Additionally, it should only be assigned to contiguous sets starting from the outermost ones (numbers 0 and 7). Such high-numbered (high priority) classes are said to have strict-high priority; low-numbered classes have strict-low priority.
DRR Group	i	A scheduling group that is set to be non-strictly scheduled. That is, $SP_c == 0$ for all classes c in the group. DRR groups are scheduled according to a Deficit Weighted Round-Robin algorithm whenever all strict-high classes have no eligible queued frames.
DRR Quantum	Q_i	Also known as "DRR Weight." A measure of the proportion of a port's bandwidth allocated to a given DRR group. The scheduler attempts to provide no less than this proportion of bandwidth to the associated DRR group. Q_i is measured in bytes and must be no smaller than the maximum frame size of the network (MTU).
Deficit Count	DC_i	Dynamic state indicating the portion of DRR quantum remaining in a scheduling round. The deficit count is decremented as frame segments egress the device. The counter is incremented by Q_i at the end of a scheduling round. DC i is signed and never greater than Q_i.
Scheduling Round	—	A scheduling iteration over all traffic classes. The end of a scheduling round occurs when no DRR group can be scheduled without replenishing its deficit count.
Average Bandwidth Utilization	B_g	The average egress bandwidth utilization of a particular shaping group, as measured over some period of time.
Upper-bound Bandwidth Limit	UB_g	For each shaping group, the Egress Scheduler guarantees that $B_g <= UB_g$.

13.3 Scheduling Algorithm

13.3.1 Groups and Sets of Traffic Classes

Ingress frames are associated with a switch priority as they traverse the frame processor pipeline. The switch priority is then mapped into one of the eight traffic classes by the scheduler. Classes are numbered 0 through 7, where 7 is the highest. Frames are then queued in a per-port, per-traffic-class queue. A per-port, class-to-group mapping associates each class into one of the eight bandwidth shaping groups, numbered 0 through 7, where 7 is the highest priority. Consecutive groups can be organized into group sets where the group set priority is determined by the group having the highest number in the set.

Figure 13-1 illustrates the hierarchy of traffic classes, scheduling groups, priority sets, and shaping groups. The diagram also indicates how this example configuration would be specified in the SCHED_GROUP_CFG and SCHED_FH_CLASS_TO_GROUP registers. The scheduling group and priority set memberships are specified with a per-class bit vector. A value of 0b indicates that class *c* belongs to the same group or set as class *c*+1; a value of 1b indicates the beginning of a new group or set. The group or set number is then designated as the highest-numbered class belonging to the group/set. Shaping groups are configured as an arbitrary class-to-group mapping.

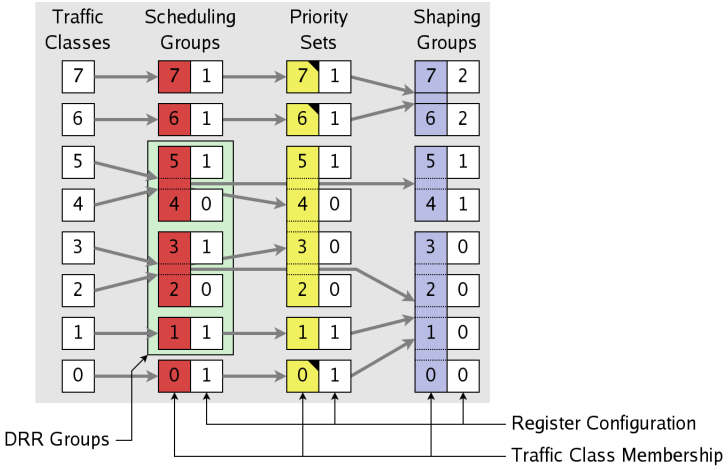


Figure 13-1 Traffic Class and Group Hierarchy

The notched corners of the priority sets indicate classes with strict attributes set to 1. In this example, there are six scheduling groups (numbered 7, 6, 5, 3, 1, and 0), three DRR groups, five priority sets, two strict-high priority sets, one strict-low priority set, and three shaping groups.

More rigorously, the configuration requirements can be expressed as follows:

SchedulingGroupBoundary[7] == 1 PrioritySetBoundary[7] == 1 ShapingGroupBoundary[7] == 1	There is always at least one group/set. (Read Only)
PrioritySetBoundary[c] => SchedulingGroupBoundary[c]	Priority sets are supersets of scheduling groups.
ShapingGroupBoundary[c] => SchedulingGroupBoundary[c]	Shaping groups are supersets of scheduling groups. ***Note:*** This is not an absolute hardware ShapingGroupBoundary[c] => SchedulingGroupBoundary[c] requirement. However, unpredictable scheduling will result if this property is not satisfied.)
StrictPriority[c] => (StrictPriority[c-1] *and* StrictPriority[0]) *or*(StrictPriority[c+1] *and* StrictPriority[7])	Strict priority can only be specified on "outer" traffic classes.
~PrioritySetBoundary[c] => StrictPriority[c] == StrictPriority[c+1]	Strict priority must be assigned consistently throughout a priority set.

It is expected that in the majority of applications, scheduling groups and shaping groups are left at their default configuration, namely identity mappings of the traffic classes.

13.3.2 Group Eligibility

The scheduler determines which scheduling groups are eligible for transmission by considering the pause status of each class and the current bandwidth consumption of each group. More precisely, a group is eligible if both of the following conditions hold:

1. It contains at least one class that is not paused, and that class has at least one frame queued for transmission.
2. The group's bandwidth utilization is under its shaping limit (i.e., The shaping group *g* that contains the scheduling group *i* has $B_g < UB_g$.

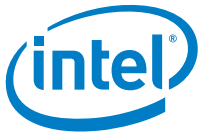
The pause state of a class is controlled by the reception of pause frames from the port's link partner. FM4000 supports two types of pause frames:

- Standard IEEE 802.3 pause.
- Class-based pause as defined by Cisco.

In the IEEE case, receiving a pause frame causes all of the port's traffic classes to become ineligible for a period of time specified by the frame's pause value. If the pause time is zero, all paused traffic classes immediately become un-paused.

Class-based pause frames have similar semantics, except a pause frame can apply to a specific set of traffic classes, and each class is assigned its own pause time.

The second eligibility condition represents an internally generated source of egress queue pausing. Each shaping group's cumulative bandwidth utilization is tracked over a configured time window. If the group consumes too much of the port's bandwidth, it exhausts its bandwidth credit and the constituent scheduling groups become ineligible. The shaping function is implemented using a leaky bucket algorithm detailed below in Section 13.4.



13.3.3 Class Selection

Given a non-empty set of eligible groups as defined in the previous section, the scheduler selects a single group from which the next frame is scheduled.

1. Any DRR groups in the eligibility set with $DC_i \leq 0$ are pruned from the eligibility set as long as there is at least one group in the eligibility set with $DC_i > 0$.
2. If all DRR groups in the eligibility set have $DC_i \leq 0$, then Q_i is added to each DC_i, marking the beginning of a new DRR scheduling round. The LastDRRGroup state is reset to -1.
3. Among all eligible groups, the scheduler selects the group that matches the earliest of the following rules. When multiple groups match, the group that is numerically greatest is chosen.
   a. The group has a strict-high prioritization.
   b. LastDRRGroup, unless the value is -1.
   c. A group belonging to a different and numerically higher priority set than LastDRRGroup, or any DRR group if LastDRRGroup is -1.
   d. A DRR group *i* belonging to the same priority set as LastDRRGroup, with *i* < LastDRRGroup.
   e. Any group in the eligibility set.
4. If the selected group is a DRR group, then LastDRRGroup is set to that group number. If it is not, LastDRRGroup is set to -1.

The next scheduled frame is then chosen from the highest-numbered class that satisfies all of the following conditions:

- Belongs to the selected scheduling group.
- Has at least one frame to be scheduled.
- Is not paused.

13.3.4 Algorithm Notes

FM4000's scheduling algorithm has the following characteristics:

- Once a particular DRR group is chosen for scheduling, the group continues to be serviced until one of the following conditions arises:
  — Its deficit count becomes zero or negative.
  — It becomes ineligible (due to emptying its queue or due to pause).
  — A strict-high priority group becomes eligible.
- A strict-low priority group is scheduled only if none of the other groups, including DRR groups, are eligible. A frame being scheduled from such a group implies that all DRR groups have maximal deficit counts and are waiting for eligible frames to send.
- When the DRR scheduling switches to a different priority set, the round-robin pointer state within the prior priority set is lost. Thus the (non-strict) group prioritization only influences the iteration order through the DRR groups within a given scheduling round; it does not provide independent DRR schedulers. This feature is intended to reduce the scheduling latency of particular DRR groups relative to others. The latency preference only applies when the lower-priority groups are not perennially backlogged.

- The class number represents a nested strict prioritization of classes belonging to the same DRR group. Two classes mapped to the same scheduling group are guaranteed a minimum egress bandwidth, represented by the group's Q_i (in the absence of strict-high priority frames). However, the higher-numbered class starve the lower-numbered class as long as it remains eligible.
  - In most applications, this nested strict priority behavior is be needed, and the scheduling groups can be left at their default one-class-per-group configuration.
- Configuring two classes to have strict priority gives the same scheduling behavior in all of the following scenarios:
  - Both classes belong to the same scheduling group.
  - Each class belongs to its own unique scheduling group and priority set.
  - Each class belongs to its own scheduling group, but share the same priority set.

  It is therefore recommended, in the interest of avoiding confusion, that the scheduler be configured such that:

  StrictPriority[c] => PrioritySetBoundary[c] => SchedulingGroupBoundary[c]

13.4 Scheduler Implementation

13.4.1 Deficit Round-Robin

Note: The terms "Deficit Weighted Round-Robin" and "Deficit Round-Robin" are both widely used, and reference the same scheduling algorithm. In this document we use the latter term to be consistent with the original publication of this algorithm (by M. Shreedhar and G. Varghese in 1995.)

The original specification of the DRR algorithm assumed store-and-forward switching: the length of the current frame determines whether the next frame in the same queue can be transmitted or not. However, in FM4000 as in any other high-performance cut-through switch, the next scheduling decision must be made in a speculative manner, before the length of the previously scheduled frame is known. Therefore, FM4000's DRR implementation must delay its decrement of the scheduled frame's deficit count relative to the standard algorithm. We refer to this variant of the DRR algorithm as "Delayed Deficit Round-Robin" (DDRR). It is comparable to Cisco's MDRR queuing algorithm.

Pseudo-code for the DDRR algorithm is provided here. The pseudo-code assumes a queuing process that assigns ingress frames to their appropriate egress queues, referenced by scheduling group. The Dequeue() and isEmpty() functions are standard queue operators. This pseudo-code only describes the DDRR behavior of FM4000's egress scheduler. It does not cover its interaction with pause-based or shaping-based eligibility, priority sets (strict or otherwise), or nested class-based strict prioritization. These interactions are specified above in Section 13.3.

```
DDRR ==
    // Initialization
    i = 7
    DC[0..7] = 0

    // Round-robin scheduling
    WHILE (true) {
        // DC updated such that each group can always send at least one frame
        // per scheduling round, even when Q is set less than MTU.
        DC[i] = max(DC[i] + Q[i], 1)
        // Scheduling decision made without considering length of Frame
        WHILE (DC[i] > 0 && !Group[i].isEmpty()) {
            Frame = Group[i].Dequeue()
            Transmit(Frame)
            DC[i] -= Frame.LengthPlusIFG()
        }
        IF (Group[i].isEmpty()) DC[i] = 0
        i = i>0 ? i-1 : 7
    }
```

Beyond the delayed evaluation of DC_i when scheduling a frame, this algorithm deviates from the standard DRR algorithm in two respects:

- When adding Q_i to DC_i, DC_i is rounded up to 1 byte. In the standard DRR algorithm, it is asserted that Q_i be set greater than or equal to the network's MTU. By rounding DC_i up to 1, DDRR permits Q_i to be set smaller than the minimum frame size (e.g. zero). DDRR schedules exactly one frame, regardless of its length, per scheduling round from such groups. This provides for per-frame (bandwidth independent) round-robin alternation.
- An additional inter-frame gap plus preamble byte count penalty is added to the length of each frame when decrementing DC_i. This correction properly models the cost of scheduling a frame. It has the effect of normalizing between two classes, one of which continuously sends 64-byte frames, the other MTU frames. Without it, the class sending MTU frames would get less than its promised bandwidth.

Given a set of Q_i (all greater than or equal to MTU), the guaranteed minimum bandwidth of a particular group, as a proportion of the port's total egress bandwidth, can be expressed as:

$$LB_i = \frac{Q_i}{\sum_i Q_i}$$

If any Q_i is less than MTU, then the minimum bandwidth allotted to each group becomes dependent on the actual frames queued. Furthermore, the presence of strict-high classes in the scheduler configuration also disturbs the DRR groups' lower bound bandwidth guarantees.

13.4.2 Bandwidth Shaping

So far in this section, the shaping function has been described as a simple comparison between a measured bandwidth per shaping group (B_g) and some configured upper-bound rate limit (UB_g). Groups remain eligible for scheduling as long as their bandwidth remains lower than the bound: $B_g < UB_g$. This is a simplification of the actual bandwidth shaping algorithm. The actual shaping function is implemented using a token bucket algorithm, which has the following characteristics:

- Every unit of time, $1/UB_g$ bytes of credit (tokens) are added to the bucket.
- Credit is subtracted from the bucket's token count as bytes belonging to the shaping group egress the device.
- When the token count goes less than or equal to zero, all associated scheduling groups become ineligible.
- The capacity of the bucket determines the maximum amount of burstiness that is permitted in the shaping group's egress bandwidth profile. Credit stops accumulating in the token bucket once this capacity is reached.

In FM4000, the token bucket parameters have the following definitions:

Time Unit	T	25 frame handler clock cycles, in nanoseconds (~66.7)
Capacity	C	Number of bytes.
RateUnit	r	Number of bytes credited every time unit.
RateFrac		Fractional number of bytes credited every time unit. Each unit of RateFrac represents 1/128 bytes.

Note that the UB_g parameter used above as a proportion of a port's egress bandwidth, is r / (T* R), where R is the total bandwidth of the egress port (nominally 1.25 bytes/ns for a 10 Gb/s port). With these parameters, the maximum burst (MB) a shaping group can transmit (starting from an idle, fully credited, state) is:

$$MB = C \cdot \left(1 + \frac{r}{R \cdot T}\right) + MTU$$

This burst occurs over MB/R nanoseconds. The additional MTU term is seen in the worst case because scheduling decisions are made without regard for the lengths of frames. A maximum length frame can be scheduled when a group's token count is at its minimum positive value (1), causing the count to go negative by up to $MTU - 1$. This requirement of cut-through switches is the same property that defines the "Delayed" aspect of FM4000's DRR implementation. This non-ideality only occurs on short timescales; the scheduler guarantees that the UB_g bound is satisfied on timescales larger than $2 * MB/R$.

As in the DRR implementation, an inter-frame gap plus preamble byte-count penalty (of 20 bytes) is subtracted from the credit count at the end of each transmitted frame. This correction properly models the higher bandwidth cost of scheduling small frames versus large frames. These penalty bytes should be considered when interpreting the maximum burst byte count calculation given above.

13.5 Configuration

The following table lists the configuration registers relevant for egress scheduling.

Table 13-2 Egress Scheduling Configuration Registers

Register Name	Size	Description
SWITCH_PRI_TO_CLASS_1 SWITCH_PRI_TO_CLASS_2	2 x 32-bits	Maps switch priority to traffic class (an arbitrary 16-to-8 mapping table). ***Note:*** This is global for all ports.
SCHED_GROUP_CFG SCHED_FH_GROUP_CFG	25 x 24-bits	Defines the scheduling group and priority set boundaries as bit vectors over traffic classes. Also specifies the per-class strict priority attribute. These mappings can be configured uniquely per-port. ***Note:*** These two registers are intentionally redundant and must be configured identically in order for the scheduling to operate correctly.
SCHED_SHAPING_GROUP_CFG	25 x 24-bits	Specifies per-port mappings from traffic class to shaping group.
SCHED_DRR_Q	25 x 8 x 24-bits	Defines the DRR quantum per group per port. ***Note:*** Value *MAX* is a special value and is equivalent to an infinite weight.
TX_RATE_LIM_CFG	25 x 8 x 32-bits	Defines the token bucket bandwidth limit parameters per shaping group, per-port.
TX_RATE_LIM_USAGE	25 x 8 x 8-bits	Returns the current bandwidth usage per shaping group.

13.6 Egress Scheduling Examples

13.6.1 Strict Priority

This example is a simple strict priority among eight different classes. Each class has its strict bit set and is assigned its own scheduling group and priority set. On every scheduling cycle the frame with the highest priority group is the one sent.

Class	Class-to-Group Map	Scheduling Group Boundary	DRR Quantum	Priority Set Boundary	Strict Priority	UB Limit
7	7 -> 7	1	X	1	1	100%
6	6 -> 6	1	X	1	1	100%
5	5 -> 5	1	X	1	1	100%
4	4 -> 4	1	X	1	1	100%
3	3 -> 3	1	X	1	1	100%
2	2 -> 2	1	X	1	1	100%

Class	Class-to-Group Map	Scheduling Group Boundary	DRR Quantum	Priority Set Boundary	Strict Priority	UB Limit
1	1 -> 1	1	X	1	1	100%
0	0 -> 0	1	X	1	1	100%

Note: Since all groups have strict priority, their quantum values are ignored by the scheduler. This is indicated with "X" in the table.

13.6.2 Weight Controlled Priority Queuing

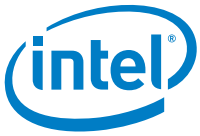
In this example, each class has its own unique priority and DRR quantum. An elected lower priority group continues to be serviced until there are no eligible frames for that group, or its quantum is exhausted. At that point, the highest priority group with eligible frames and positive *DC* is elected and serviced.

Class	Class-to-Group Map	Scheduling Group Boundary	DRR Quantum	Priority Set Boundary	Strict Priority	UB Limit
7	7 -> 7	1	Q_7	1	0	100%
6	6 -> 6	1	Q_6	1	0	100%
5	5 -> 5	1	Q_5	1	0	100%
4	4 -> 4	1	Q_4	1	0	100%
3	3 -> 3	1	Q_3	1	0	100%
2	2 -> 2	1	Q_2	1	0	100%
1	1 -> 1	1	Q_1	1	0	100%
0	0 -> 0	1	Q_0	1	0	100%

13.6.3 Mixed Strict and Round-Robin Queues

In this example, class 7 has strict high priority, class 0 has strict low priority, and classes 1 through 6 are serviced round-robin. Note that configuring a DRR group's quantum to less than the minimum frame size has the effect of restricting it to send no more than one frame per scheduling round.

Class	Class-to-Group Map	Scheduling Group Boundary	DRR Quantum	Priority Set Boundary	Strict Priority	UB Limit
7	7 -> 7	1	X	1	1	100%
6	6 -> 6	1	0	1	0	100%
5	5 -> 5	1	0	0	0	100%
4	4 -> 4	1	0	0	0	100%
3	3 -> 3	1	0	0	0	100%
2	2 -> 2	1	0	0	0	100%



Class	Class-to-Group Map	Scheduling Group Boundary	DRR Quantum	Priority Set Boundary	Strict Priority	UB Limit
1	1 -> 1	1	0	0	0	100%
0	0 -> 0	1	X	1	1	100%

13.6.4 Nested Strict Prioritization

Suppose there are four queues A, B, C, and D. A is to be strictly prioritized over B, but otherwise the groups {A,B}, {C}, and {D} are to share bandwidth according to some specified weights. In this example, classes 7 through 4 are not used.

Class	Class-to-Group Map	Scheduling Group Boundary	DRR Quantum	Priority Set Boundary	Strict Priority	UB Limit
3 (A)	3 -> 3	1	Q_{AB}	1	0	100%
2 (B)	2 -> 2	0	X	0	0	100%
1 (C)	1 -> 1	1	Q_C	0	0	100%
0 (D)	0 -> 0	1	Q_D	0	0	100%

13.6.5 Deficit Round-Robin with Maximum Bandwidth Limits

Suppose there are four queues A, B, C, and D that are to share the egress port's bandwidth equally. However, the sum of C and D's egress bandwidth must never exceed 50% of the port's capacity, even when queues A and B are empty.

Class	Class-to-Group Map	Scheduling Group Boundary	DRR Quantum	Priority Set Boundary	Strict Priority	Shaping Group Map	UB Limit
3 (A)	3 -> 3	1	MTU	1	0	2	100%
2 (B)	2 -> 2	1	MTU	0	0	1	100%
1 (C)	1 -> 1	1	MTU	0	0	0	50%
0 (D)	0 -> 0	1	MTU	0	0	0	

§ §

NOTE: ***This page intentionally left blank.***

14.0 Statistics and Monitoring

14.1 Frame Counters

FM4000 devices maintain numerous per-port and switch-wide frame counters that provide management with statistical information about the state of the switch and of the network in general. The majority of these counters are provided for compatibility with network monitoring-related IETF RFCs: 2819 (RMON), 3273 (High-Capacity RMON), 2613 (SMON), and 4502 (RMON2).

FM4000's counters are organized into a collection of counter groups. Most groups are defined such that only one of its counters is updated per ingress or egress frame. Table 14-1 summarizes FM4000's counter groups.

Table 14-1 List of Counter Banks

Group	Description	Type	Unit	Number of Counters	Standard
1	RX Frame Type	Per-Port, RX	Frames	25 x 6	RMON
2	RX Frame Length Bin	Per-Port, RX	Frames	25 x 16	RMON
3	RX Frame Octets	Per-Port, RX	Octets	25 x 1	RMON
4	RX Frame Counters per Priority	Per-Port, RX	Frames	25 x 8	SMON
5	RX Octets per Priority	Per-Port, RX	Octets	25 x 8	SMON
6	RX Forwarding Action	Per-Port, RX	Frames	25 x 25	—
—	RX Counter Drops	Per-Port, RX	Frames	25 x 1	RMON
7	TX Frame Type	Per-Port, TX	Frames	25 x 7	RMON
8	TX Frame Length Bin	Per-Port, TX	Frames	25 x 12	RMON
9	TX Frame Octets	Per-Port, TX	Octets	25 x 1	RMON
—	TX Counter Drops	Per-Port, TX	Frames	25 x 1	RMON
10	Congestion Notification	Global, RX	Frames	25 x 24	—
11	VLAN Frames	Per-VLAN, RX	Frames	64 x 2	SMON

Table 14-1 List of Counter Banks (Continued)

Group	Description	Type	Unit	Number of Counters	Standard
12	VLAN Octets	Per-VLAN, RX	Octets	64 x 2	SMON
13	Trigger counters	Per-Trigger, RX	Frames	64	—

Note: A variety of layer 3/4 packet header and forwarding properties can be counted in a configurable way by using the FFU counters. Each FFU counter tallies both the packet count and octet count of each frame counted.

Note on counter performance limitations:

The first 9 groups of counters as listed in Table 14-1are implemented in SRAM, and this limitation only applies to those counter groups. The Frame Handler mediates counting activity and causes, at each clock cycle, 13 simultaneous counting events (one for each group), of which 9 target the SRAM stats block. The SRAM stats block does not keep up with the 9 groups of counters in the worst performance case (line rate 64-byte packets and certain specific packet sequences). The Frame Handler implements a short FIFO to compensate for bursts of counting events, but this FIFO may become overloaded, causing the counting event to be dropped and an error to be counted.

Dropping counting events is very rare in actual operation, as it requires sustained bursts of 64-byte packets. One work-around is to disable two groups out of the first 9 groups (the ones that are less useful to you). The fact that a count event is dropped does not cause the frame to be mishandled or dropped. Packets flow normally and only stats are affected.

14.2 Frame Monitoring

All FM4000 switch devices support two traffic monitoring mechanism: RX Mirroring and TX Mirroring.

RX Mirroring is enabled by either FFU or Trigger actions. When enabled, the mechanism causes an unmodified copy of the ingress frame to be sent to a destination mirror port. RX Mirroring is quite flexible in that the selection of source frames for mirroring depends on the matching conditions and precedence resolution of the FFU and triggers. In the case of the FFU, a single destination mirror GloRT and port applies to any frames selected for mirroring. In the case of the triggers, the mirror destination port and GloRT may be specified uniquely for each trigger action. If the ingress frame is dropped for any reason, the mirrored copy is still forwarded (subject to congestion management).

TX Mirroring is enabled by the TX_MIRROR_FH and TX_MIRROR registers. It is less flexible than RX mirroring in that only a single “source” egress port may be mirrored to a single “destination” egress port. When enabled, all frames destined to the source egress port are mirrored. To guarantee that all mirrored copies are identical to their corresponding source frames, the destination TX mirror port's EPL registers must be configured the same as the source port's registers.

For either mirroring mechanism, further modification of the copied frames by downstream FM4000 devices is prevented by setting their ISL tag FTYPE field to 0x2 (Special Delivery) on egress.

§ §

15.0 Electrical Specification

15.1 Absolute Maximum Ratings

Table 15-1 Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Units
Core Voltage	V_{DD}	-0.3	2	V
SerDes Supply Voltage	V_{DDX}	-0.3	2	V
SerDes Bias Voltage	V_{DDA}	-0.3	2	V
Transmitter Termination Voltage	V_{TT}	-0.3	2	V
LVTTL Power Supply	V_{DD33}	-0.3	3.9	V
PLL Analog power supply	V_{DDA33}	-0.3	3.9	V
Case Temp under bias	–	–	130	°C
Storage Temp	–	-65	150	°C
ESD	–	-2000	2000	V

15.2 Recommended Operating Conditions

Table 15-2 Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Units
Core Voltage	V_{DD}	–	1.25	–	V
SerDes Supply Voltage	V_{DDX}	1.14	1.20	1.26	V[1,3]
SerDes Supply Voltage (when using SGMII)	V_{DDX}	0.95	1.00	1.05	V[2,3]
SerDes Bias Voltage	V_{DDA}	1.14	1.20	1.26	V
SerDes Bias Voltage (when using SGMII)	V_{DDA}	0.95	1.00	1.05	V

Table 15-2 Recommended Operating Conditions (Continued)

Parameter		Symbol	Min	Typ	Max	Units
LVTTL Power Supply		V_{DD33}	3.14	3.30	3.47	V
PLL Analog power supply		V_{DDA33}	3.14	3.30	3.47	V
Transmitter Termination Voltage		V_{TT}	V_{DDX}	1.50	1.80	V
Operating Temp (Case)	Commercial	–	0	–	85	°C
	Extended	–	0	–	105	°C
	Industrial	–	-40	–	115	°C

1. Connect a 1.2 K resistor from RREF to VDDX for 1.2 V operation.
2. Connect a 1.0 K resistor from RREF to VDDX for 1.0 V operation.
3. Sharing VDD and VDDX is not recommended.

15.3 Power Supply Sequencing

The only requirement concerning the sequence in which power supplies should be brought up is that VDD should reach at least 90% of its specified value 1 ms before VDD33 is powered on as shown in Figure 15-1. In addition, it is recommended that VDD33 be ramped down prior to removing VDD. Finally, RESET_N should be asserted at power down.

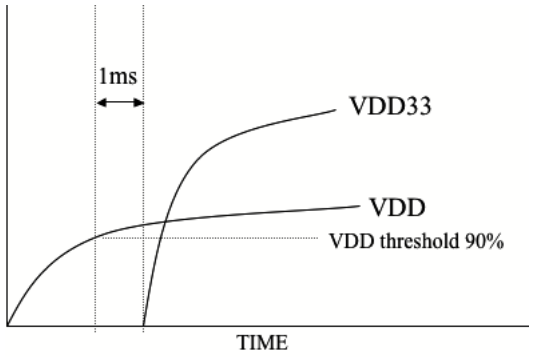


Figure 15-1 Power Supply Sequencing

15.4 DC Characteristics of LVTTL PADs

There are three types of LVCMOS PADs: 2 mA, 4 mA and 6 mA. The characteristics are shown in Table 15-3, Table 15-4 and Table 15-5, respectively.

The 2 mA LVTTL are used for the following signals:

- LED_CLK, LED_EN, LED_DATA[1..3]
- SPI_CLK, SPI_CS_N, SPI_MISO, SPI_MOSI

The 4 mA LVTTL are used for the following signals:

- FH_PLL_CLKOUT
- RXEOT_N, RXRDY_N
- TDO
- TXRDY_N

The 6 mA LVTTL are used for the following signals:

- AUTOBOOT
- DATA[0..31], DATA_HOLD
- DERR_N
- DTACK_INV, DTACK_N
- EEPROM_EN1, EEPROM_EN2,
- I2C_ADDR[0..2]
- GPIO_15
- IGNORE_PARITY
- INTR_N
- MDC, PARITY_EVEN
- PAR[0..3]
- RW_INV

Table 15-3 DC Characteristics of 2 mA LVTTL Outputs

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
HIGH Force Tri-State output leakage	I_{OZH}	V_{DD} = Max V_O = V_{DD}	-1	–	1	µA
LOW Force Tri-State output leakage	I_{OZL}	V_{DD} = Max V_O = GND	-1	–	1	µA
Output HIGH Current	I_{ODH}	V_{DD} = 1.2 V V_{DD33} = 3.3 V V_O = 1.5	–	-9	–	mA
Output LOW Current	I_{ODL}	V_{DD} = 1.2 V V_{DD33} = 3.3 V V_O = 1.5	–	10	–	mA

Table 15-3 DC Characteristics of 2 mA LVTTL Outputs (Continued)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
Output HIGH Voltage	V_{OH}	V_{DD} = Min V_{DD33} = Min I_{OH} = -0.4 mA	V_{DD33} - 0.2	–	–	V
Output HIGH Voltage	V_{OH}	V_{DD} = Min V_{DD33} = Min I_{OH} = -4.0 mA	V_{DD33} - 0.5	–	–	V
Output LOW Voltage	V_{OL}	V_{DD} = Min V_{DD33} = Min I_{OL} = -0.4 mA	–	–	0.2	V
Output LOW Voltage	V_{OL}	V_{DD} = Min V_{DD33} = Min I_{OL} = -4.0 mA	–	0.2	0.4	V
Short Circuit Current	I_{OS}	V_{DD} = Max V_O = GND	–	–	-16	mA
Power Supply Quiescent Current	I_{AA}	V_{DD} = Max V_{DD33} = Max	–	–	74	µA
Power Supply Quiescent Current	I_{AA}	Tri-stated	–	–	-1	µA

Table 15-4 DC Characteristics of 4 mA LVTTL Outputs

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
HIGH Force Tri-State output leakage	I_{OZH}	V_{DD} = Max V_O = V_{DD}	-1	–	1	µA
LOW Force Tri-State output leakage	I_{OZL}	V_{DD} = Max V_O = GND	-1	–	1	µA
Output HIGH Current	I_{ODH}	V_{DD} = 1.2 V V_{DD33} = 3.3 V V_O = 1.5	–	-17	–	mA
Output LOW Current	I_{ODL}	V_{DD} = 1.2 V V_{DD33} = 3.3 V V_O = 1.5	–	20	–	mA
Output HIGH Voltage	V_{OH}	V_{DD} = Min V_{DD33} = Min I_{OH} = -0.4 mA	V_{DD33} - 0.2	–	–	V
Output HIGH Voltage	V_{OH}	V_{DD} = Min V_{DD33} = Min I_{OH} = -4.0 mA	V_{DD33} - 0.5	–	–	V
Output LOW Voltage	V_{OL}	V_{DD} = Min V_{DD33} = Min I_{OL} = -0.4 mA	–	–	0.2	V
Output LOW Voltage	V_{OL}	V_{DD} = Min V_{DD33} = Min I_{OL} = -4.0 mA	–	0.2	0.4	V
Short Circuit Current	I_{OS}	V_{DD} = Max V_O = GND	–	–	-32	mA
Power Supply Quiescent Current	I_{AA}	V_{DD} = Max V_{DD33} = Max	–	–	74	µA
Power Supply Quiescent Current	I_{AA}	Tri-stated	–	–	-1	µA

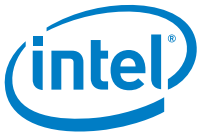


Table 15-5 DC Characteristics of 6 mA LVTTL Outputs

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
HIGH Force Tri-State output leakage	I_{OZH}	V_{DD} = Max V_O = V_{DD}	-1	–	1	µA
LOW Force Tri-State output leakage	I_{OZL}	V_{DD} = Max V_O = GND	-1	–	1	µA
Output HIGH Current	I_{ODH}	V_{DD} = 1.2 V V_{DD33} = 3.3 V V_O = 1.5	–	-26	–	mA
Output LOW Current	I_{ODL}	V_{DD} = 1.2 V V_{DD33} = 3.3 V V_O = 1.5	–	-30	–	mA
Output HIGH Voltage	V_{OH}	V_{DD} = Min V_{DD33} = Min I_{OH} = -0.4 mA	V_{DD33} - 0.2	–	–	V
Output HIGH Voltage	V_{OH}	V_{DD} = Min V_{DD33} = Min I_{OH} = -4.0 mA	V_{DD33} - 0.5	–	–	V
Output LOW Voltage	V_{OL}	V_{DD} = Min V_{DD33} = Min I_{OL} = -0.4 mA	–	–	0.2	V
Output LOW Voltage	V_{OL}	V_{DD} = Min V_{DD33} = Min I_{OL} = -4.0 mA	–	0.2	0.4	V
Short Circuit Current	I_{OS}	V_{DD} = Max V_O = GND	–	–	-48	mA
Power Supply Quiescent Current	I_{AA}	V_{DD} = Max V_{DD33} = Max	–	–	74	µA
Power Supply Quiescent Current	I_{AA}	Tri-stated	–	–	-1	µA

The data listed in Table 15-6 Applies to all LVTTL inputs or input/outputs when in input mode

Table 15-6 DC Characteristics of LVTTL Inputs

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
Input HIGH Level (Input and I/O pins)	V_{IH}	Guaranteed Logic HIGH Level	2	–	V_{DD33} + 0.5	V
Input LOW Level (Input and I/O pins)	V_{IL}	Guaranteed Logic LOW Level	-0.3	–	0.8	V
Input Hysteresis	V_H	it0	–	5	–	mV
Input Hysteresis	V_H	it2	–	200	–	mV
Input HIGH Current (Input pins)	I_{IH}	V_{DD} = Max V_I = V_{IH}(Max)	–	–	±1	µA
Input HIGH Current (I/O pins)	I_{IH}	V_{DD} = Max V_I = V_{CC}	–	–	±1	µA

Table 15-6 DC Characteristics of LVTTL Inputs (Continued)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Units
Input LOW Current (Input pins)	I_{IL}	V_{DD} = Max V_I = GND	–	–	±1	µA
Input LOW Current (I/O pins)	I_{IL}	V_{DD} = Max V_I = GND	–	–	±1	µA
Clamp Diode Voltage	V_{IK}	V_{DD} = Min I_{IN} = -18 mA	–	-0.7	-1.2	V
Quiescent Power Supply Current	I_{DD33L}	V_{DD} = Max V_{CC} = Max V_{IN} =GND	–	0.1	10	µA
Quiescent Power Supply Current	I_{DD33H}	V_{DD} = Max V_{CC} = Max V_{IN}=V_{DD}	–	0.1	10	µA

15.5 AC Timing Specifications

Table 15-7 XAUI Transmitter Characteristics

Symbol	Parameter	Min	Typ	Max	Units
V_{SW}	Output voltage (peak-to-peak, single-ended)	200[1]	500	750[2]	mV
$V_{DIFF-PP}$	Output voltage (peak-to-peak, differential)	400[1]	1000	1500[2]	mV
V_{OL}	Low-level output voltage	–	Vtt - 1.5 * V_{SW}	–	–
V_{OH}	High-level output voltage	–	Vtt - 0.5 * V_{SW}	–	–
V_{TCM}	Transmit common-mode voltage[3]	–	Vtt - V_{SW}	–	–
J_{TT} @ 1.25 Gb/S	Transmitter Total Jitter (Peak-Peak)[4]	–	–	0.24	UI
	Random jitter component (RJ)	–	–	0.12	
	Deterministic jitter component (DJ)	–	–	0.12	
J_{TT} @ 3.125 Gb/s	Transmitter Total Jitter (Peak-Peak)[4]	–	–	0.35	UI
	Random jitter component (RJ)	–	–	0.18	
	Deterministic jitter component (DJ)	–	–	0.17	
Z_{OSE}	Single Ended Output Impedance	40	50	60	Ohms

Table 15-7 XAUI Transmitter Characteristics (Continued)

Symbol	Parameter	Min	Typ	Max	Units
Z_D	Differential Output Impedance	80	100	120	Ohms
T_{TR}, T_{TF}	Rise, fall times of differential outputs[5]	80	–	110	ps

1. HiDrv bit set to 0, LoDrv bit set to 1 in SERDES_CNTL_2 register, and Current Drive bits set to 1100 in SERDES_CNTL_1 register.
2. VTT = 1.8V, HiDrv bit set to 1, LoDrv bit set to 0 in SERDES_CNTL_2 register – see, and Current Drive bits set to 0011 in SERDES_CNTL_1 register.
3. AC coupled operation only.
4. Based on CJPAT.
5. 20% to 80%.

Table 15-8 XAUI Receiver Characteristics

Symbol	Parameter	Min	Typ	Max	Units
V_{LOS}	Low signal differential input threshold voltage	85	–	–	mV
V_{IN}	Differential input voltage, peak to peak	170	–	2000	mV
V_{RCM}	Common mode voltage	–	0.70	–	V
T_{RR}, T_{RF}	Rise, fall times of differential inputs	–	–	160	ps
T_{DSkew}	Differential pair skew	–	–	15	ps
T_{Skew}	Lane-to-lane skew	–	–	12.8	ns
J_{RT} @ 1.25 Gb/S	Total Jitter Tolerance[1]	–	–	0.71	UI
	Random jitter component (RJ)	–	–	0.26	
	Deterministic jitter component (DJ)	–	–	0.45	
J_{TT} @ 3.25 Gb/s	Total Jitter Tolerance[1]	–	–	0.65	UI
	Random jitter component (RJ)	–	–	0.24	
	Deterministic jitter component (DJ)	–	–	0.41	
Z_{IN}	Impedance, single-ended	40	50	60	–
L_{DR}	Differential return loss[2]	10	–	–	dB
V_{RHP}	Hot plug voltage (applied with power on or off)[3]	-0.5	–	1.6	V

1. CJPAT
2. Frequency range of 100 MHz to 1.875 GHz.
3. Without damage to any signal pin.

Table 15-9 REFCLK Input Characteristics

Symbol	Parameter	Min	Typ	Max	Units
V_{il}	Low-level CML input voltage.	0		Vddx - 0.5	V
V_{ih}	High-level CML input voltage		Vddx		V
V_{sw}	Differential input voltage swing	0.4		0.8	V
	Frequency	100	[1]	400	MHz
	Duty Cycle	40	50	60	%
	Stability		50	100	ppm
	Skew between P & N			0.05	RCIU[2]
J_{rc}	Input jitter RMS			1.94	ps[3]
T_{rise}/T_{fall}	Rise/Fall time		0.2	0.25	RCIU[2]

1. Frequency for XAUI or 2.5 GbE must be 312.5 MHz. Frequency for 1 GbE must be 125.0 MHz.
2. RCUI references the REFCLK period.
3. BER 10^{-16}

Table 15-10 Clock Input Requirements

Symbol	Parameter	Min	Typ	Max	Units
T_{FJ}	Frame handler clock jitter	–	–	20	ps RMS
T_{FD}	Frame handler clock duty cycle	40	–	60	%
F_{FR}	Frame handler clock frequency range	5	–	70	MHz
V_{CSW}	CPU clock voltage swing	2.97	–	3.63	V
T_{CS}	CPU clock frequency stability	-100	–	100	ppm
T_{CD}	CPU clock duty cycle	40	–	60	%
T_{CRF}	CPU clock rise/fall times	–	–	8	ns
F_{CR}	CPU clock frequency range	0	–	100	MHz

15.5.1 CPU Interface, General Timing Requirements

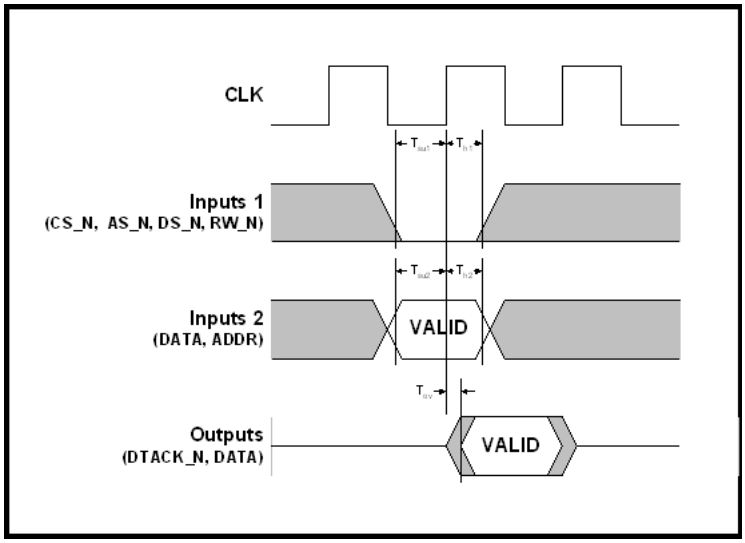


Figure 15-2 CPU Signal Timing

Table 15-11 CPU Interface Timing Constraints

Parameter	Symbol	Min	Typ	Max	Units	Test Conditions
Setup time for CS_N, AS_N, DS_N and RW_N, to rising edge of clock	T_{su1}	20	–	–	ns	–
Hold time for CS_N, AS_N, DS_N and RW_N, to rising edge of clock	T_{h1}	0.5	–	–	ns	–
Setup time for ADDR and DATA(in) to rising edge of clock	T_{su2}	20	–	–	ns	–
Hold time for ADDR and DATA(in) to rising edge of clock	T_{h2}	0.5	–	–	ns	–
Output valid for DTACK_N and DATA(out) to rising edge of clock	T_{ov}	0	–	3.5	ns	–

15.5.2 MDIO Interface, General Timing Requirements

The MDIO interface timing is shown in Figure 15-3. The MDC (output clock) is derived from the CPU clock with the output frequency set in the MDIO_CFG register. The MDIO output signal is also timed to the CPU clock.

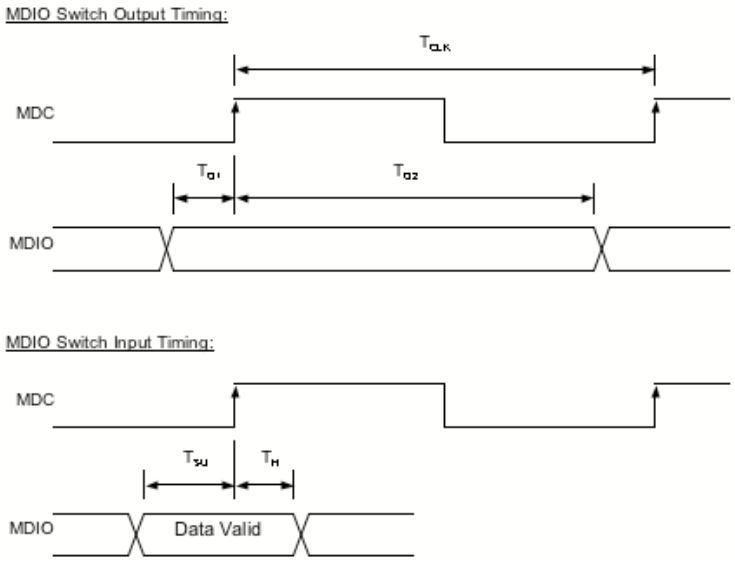
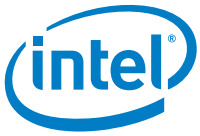


Figure 15-3 MDIO Interface Timing

Table 15-12 MDIO Interface Timing Constraints

Parameter	Symbol	Min	Typ	Max	Units	Comments
MDC output clock period	T_{CLK}	–	–	–	–	Programmable period using the MDIO_CFG::Divider register. Period = (CPU_CLK/2)/Divider.
MDIO switch output timing before MDC clock edge	T_{O1}	–	–	–	–	8 times the CPU clock.
MDIO switch output timing after MDC clock edge	T_{O2}	–	–	–	–	T_{CLK} - T_{O1}
MDIO input set-up time relative to MDC	T_{SU}	10.0	–	–	ns	–
MDIO input hold time relative to MDC	T_H	0.0	–	–	ns	–



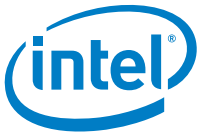
15.5.3 JTAG Interface

The JTAG interface follows standard timing as defined in the *IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture, 2001*.

When not using the JTAG interface, either drive the TCK pin with an external clock, or drive the TRST_N pin low. Conversely, when using the JTAG interface, assert TRST_N along with chip reset to ensure proper reset of the JTAG interface prior to use.

§ §

NOTE: ***This page intentionally left blank.***



16.0 Mechanical Specification

This section provides a set of mechanical drawings for the FM4000 series of products. There are three package sizes used for these products as listed below.

Table 16-1 Products and Package Sizes

Part Number	Package Type	Notes
FM4410	529-Ball, 25 mm	
FM4105	897-Ball, 32 mm	Any 2 of the 8 XAUI ports can be used.
FM4112	897-Ball, 32 mm	
FM4212	1433-Ball, 40 mm	Any 12 of the 24 XAUI ports can be used.
FM4224	1433-Ball, 40 mm	

16.1 1433-Ball 40mm Package Dimensions

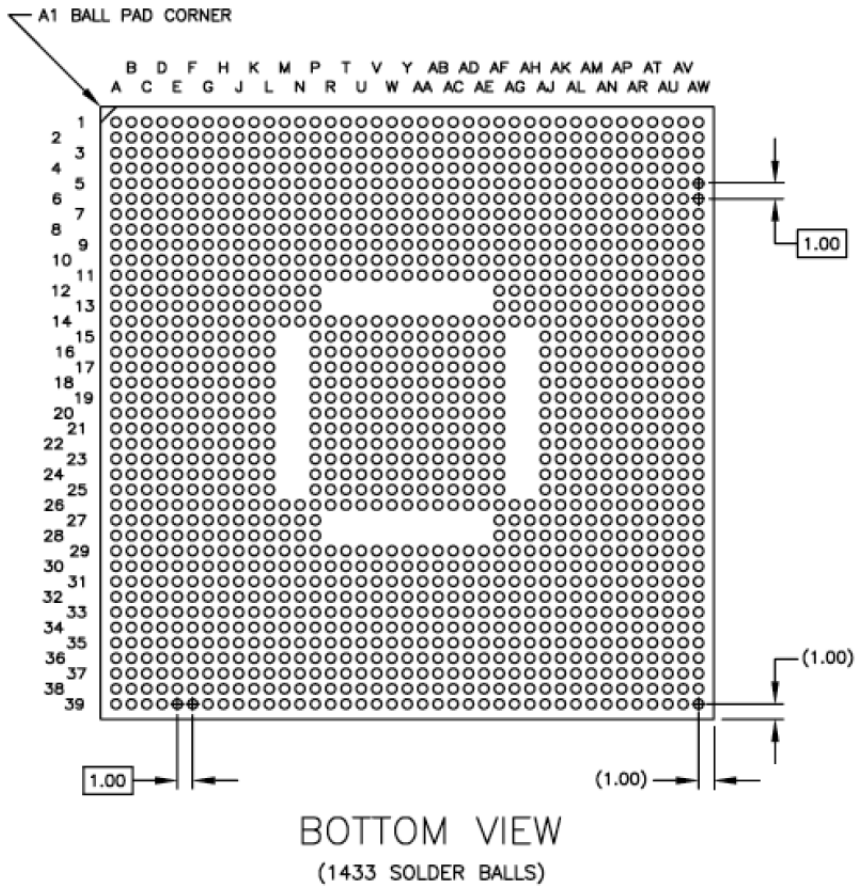
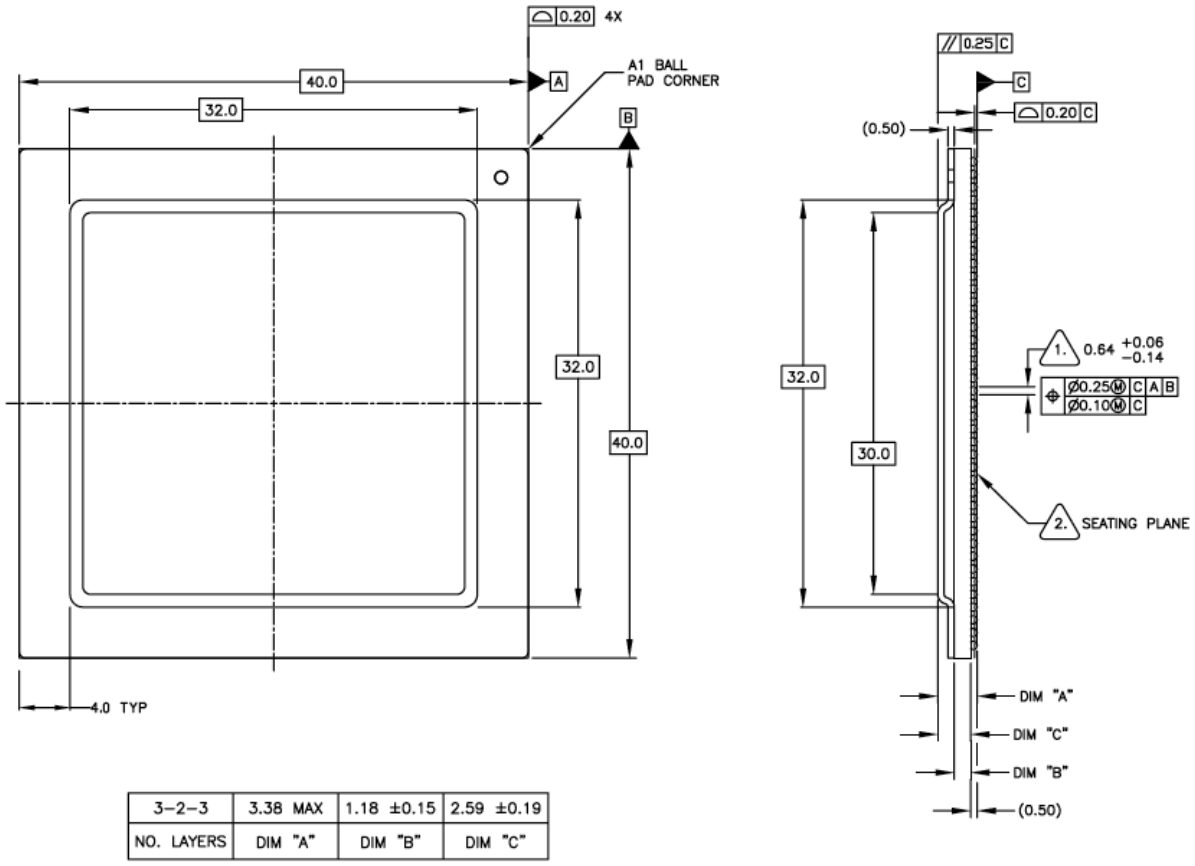


Figure 16-1 1433-Ball Package Bottom View



3–2–3	3.38 MAX	1.18 ±0.15	2.59 ±0.19
NO. LAYERS	DIM "A"	DIM "B"	DIM "C"

1433-Ball Package Top View

Notes:

- Dimension 1 is measured at the maximum solder ball diameter, parallel to primary datum C.
- Primary datum C and seating plane 2 are defined by the spherical crowns of the solder balls.

16.2 897-Ball 32mm Package Dimensions

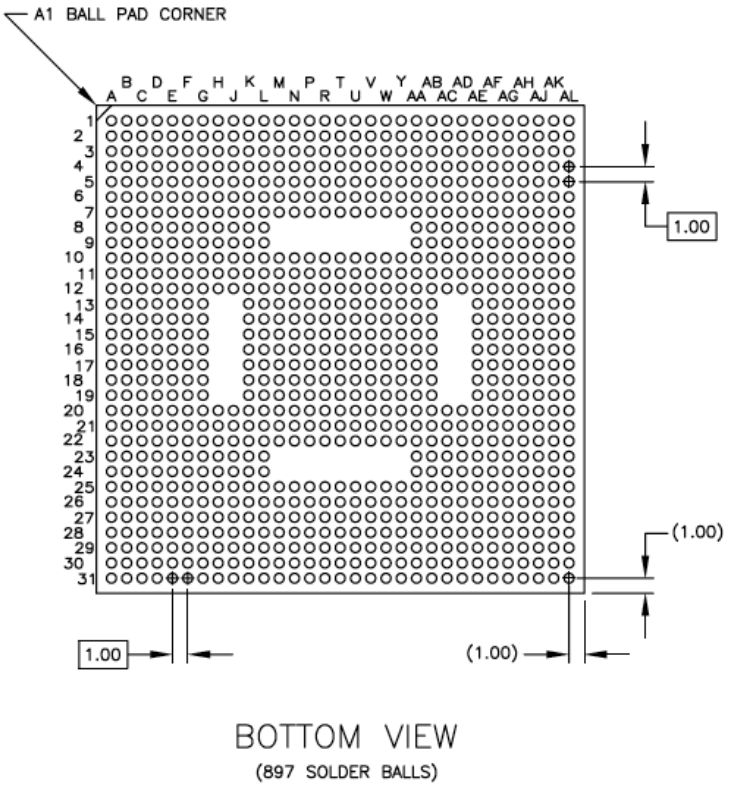
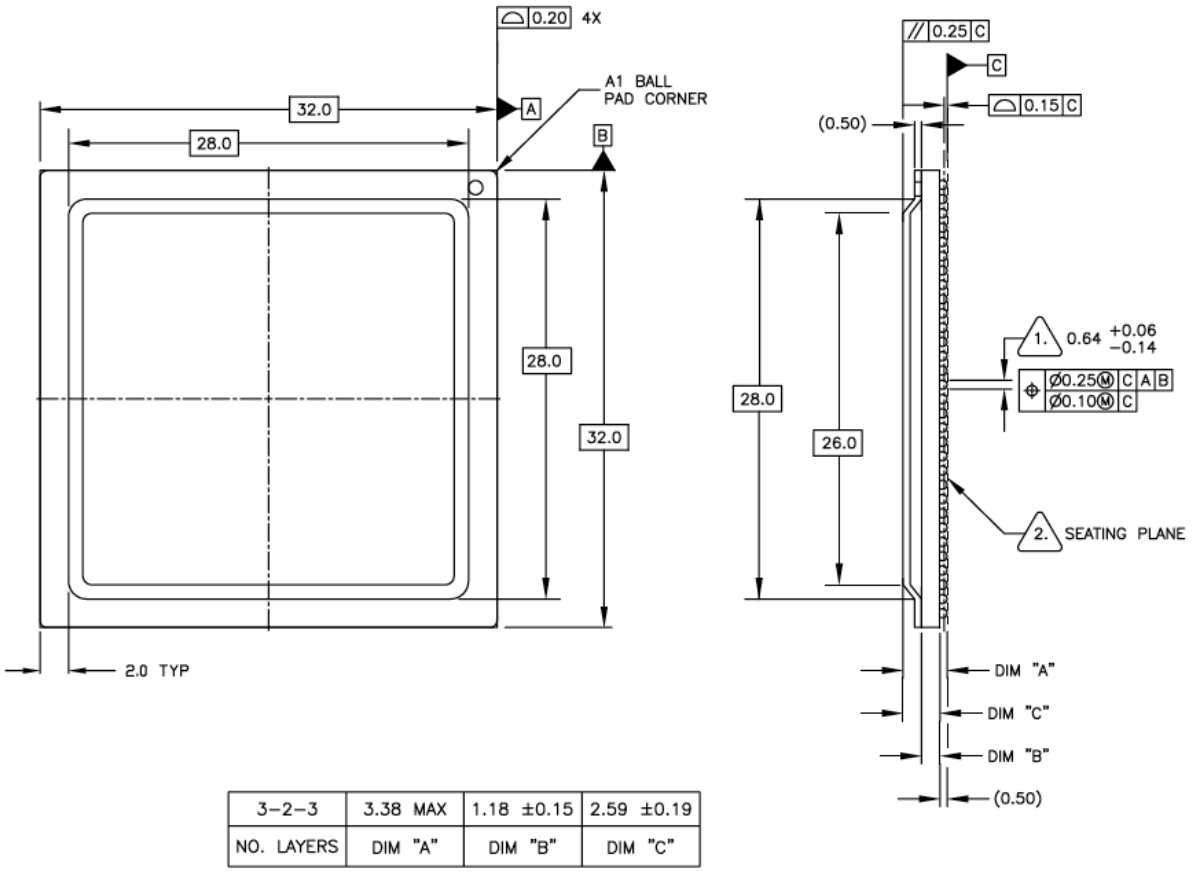


Figure 16-2 897-Ball Package Bottom View



3–2–3	3.38 MAX	1.18 ±0.15	2.59 ±0.19
NO. LAYERS	DIM "A"	DIM "B"	DIM "C"

Figure 16-3 897-Ball Package Top View

Notes:

- Dimension 1 is measured at the maximum solder ball diameter, parallel to primary datum C.
- Primary datum C and seating plane 2 are defined by the spherical crowns of the solder balls.

16.3 529-Ball 25mm Package Dimensions

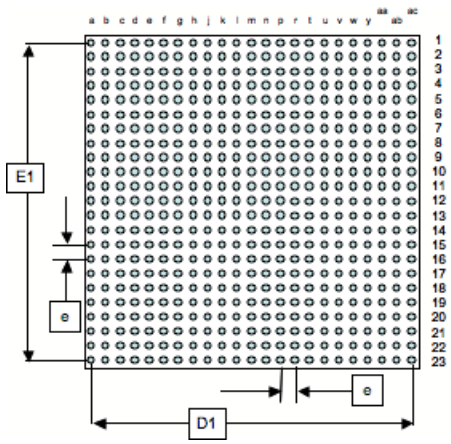


Figure 16-4 529-Ball Package Bottom View

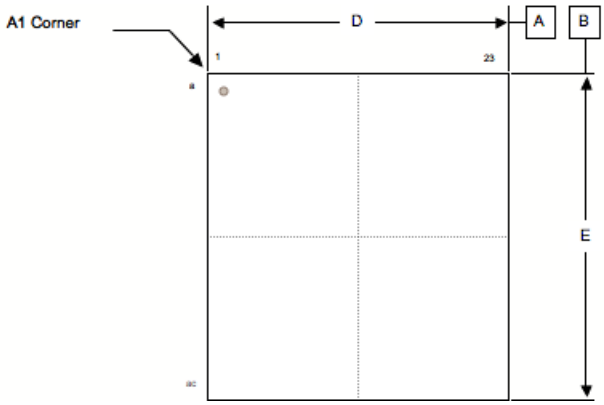


Figure 16-5 529-Ball Package Top View

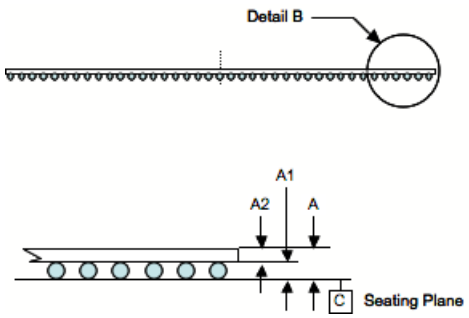


Figure 16-6 529-Ball Package Side View

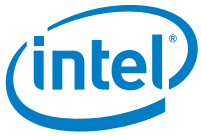


Table 16-2 529-Ball Package Dimensions

Dimensional References			
Reference	**Min**	**Nom**	**Max**
A	2.14	2.42	2.70
A1	0.40	0.50	0.60
A2	1.74	1.92	2.10
D	24.90	25.00	25.10
D1	22.0 BSC		
E	24.90	25.00	25.10
E1	22.0 BSC		
e	1.00 BSC		
M	23		
N	529		

Ref.: JEDEC MS-034 B

Notes:

- All dimensions are in millimeters. "e" represents the basic solder ball grid pitch.
- "M" represents the basic solder ball matrix size.
- Symbol "N" is the maximum allowable number of balls after depopulating.
- Primary datum C and Seating Plane are defined by the spherical crowns of the solder balls.
- Package surface is Ni plated. Black spot (or circular etch) for pin 1 identification.
- Dimensions and tolerance per ASME Y14.5M 1994

16.4 Heat Sinking

It is anticipated that a heat sink will be required for most applications. Intel has successfully used two copper heat sinks designed by Intel and manufactured by *Radian Heatsinks* of Santa Clara, CA (www.radianheatsinks.com), which when reasonable air flow is present, leads to acceptable case temperatures.

The two heat sink designs that have been tested have Radian model numbers FUL003 and FUL004/5. The FUL003 design occupies a smaller area of approximately 64 mm x 64 mm, and is taller at just over 25 mm in height. The FUL004/5 design occupies a larger footprint of 80 mm x 100 mm, but is only 8.5 mm in height. The thermal characteristics (Theta-CA) of the two heat sink designs are approximately equal.

The overarching goal of heat sink design is to keep the operating case temperature of the device below its maximum allowed value. This also ensures that the junction stays below its maximum allowable temperature of 125 °C. With a junction-to-case thermal resistance (Theta-JC) of only 0.1 °C/W, even the highest allowed value of case temperature, 115 °C, keeps the junction temperature below 125 °C. This is true even assuming a worst case power dissipation approaching 48 W, which results in a 4.8 °C rise in junction temperature over case temperature (48 W x 0.1 °C/W = 4.8 °C).

An effective thermal design solution requires knowledge of several parameters of the device package and heat sink, and of the expected ambient temperature and air flow over the device. The relevant package and heat sink parameters are listed in Table 16-3.

Table 16-3 Thermal Parameters

Parameter	Description		Value
Theta-JC	Thermal resistance, junction to case.		0.1 °C/W
Psi-JB[1,2]	Thermal resistance, junction to board.		3.5 °C/W
Theta-CA	Thermal resistance, case to ambient with heat sink in place.	100 LFM	1.2 °C/W (typ)
		150 LFM	0.9 °C/W (typ)
		200 LFM	0.75 °C/W (typ)
TCASE(max)	Maximum case temperature.	Commercial, -C	85 °C
		Extended, -E	105 °C
		Industrial, -I	115 °C

1. This parameter is similar to Theta-JB as defined by JEDEC, except that the presence of an isothermal cold ring is not assumed. The value of Psi-JB is more relevant to real world use scenarios.
2. The PC board is assumed to be at ambient temperature.

With a knowledge of these parameters and the ambient air temperature surrounding the device, the degree of airflow required to keep the case temperature below the specified maximum can be determined. Alternatively, if the airflow is known, the maximum ambient temperature can be calculated.

As an example, if the known parameters are:

- FM4224-F1433-E, 105 °C max case temperature
- 48W maximum power dissipation
- Airflow = 150 LFM

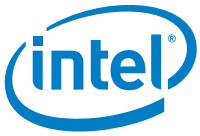
Then the overall temperature rise from case to ambient can be calculated as:

48 x Theta-CA || Psi-JB

With the values of Theta-CA and Psi-JB from Table 16-3, this equates to a temperature rise of approximately 35 °C. With a maximum case temperature of 105 °C, a maximum ambient temperature of 70 °C is allowed. If higher ambient temperatures are desired, an increase in airflow or the use of the Industrial temperature range (-I) device may be required.

16.4.1 Heat Sink Mounting Pressure

The maximum allowable pressure on the package surface when mounting a heat sink is 15 psi.



16.5 Pin Locations

Note: The FM4105 uses the 32 mm 897-pin package, but supports only two XAUI interfaces. These two XAUI interfaces are on ports P01 and P02 in Table 16-4.

Table 16-4 PIN Locations — Alphabetical Order

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
ADDR[23..2]	ADDR[2]: T32 ADDR[3]: T33 ADDR[4]: T34 ADDR[5]: T35 ADDR[6]: T36 ADDR[7]: T37 ADDR[8]: T38 ADDR[9]: T39 ADDR[10]: U31 ADDR[11]: U32 ADDR[12]: U33 ADDR[13]: U34 ADDR[14]: U35 ADDR[15]: U36 ADDR[16]: U37 ADDR[17]: U38 ADDR[18]: U39 ADDR[19]: V33 ADDR[20]: V34 ADDR[21]: V35 ADDR[22]: V36 ADDR[23]: V37	ADDR[2]: K29 ADDR[3]: K25 ADDR[4]: L25 ADDR[5]: L26 ADDR[6]: L27 ADDR[7]: L28 ADDR[8]: L29 ADDR[9]: L30 ADDR[10]: L31 ADDR[11]: M25 ADDR[12]: M26 ADDR[13]: M27 ADDR[14]: M28 ADDR[15]: M29 ADDR[16]: M30 ADDR[17]: M31 ADDR[18]: N26 ADDR[19]: N27 ADDR[20]: N28 ADDR[21]: N29 ADDR[22]: N30 ADDR[23]: N31	
AS_N	W35	P30	
CHIP_RESET_N	N33	J27	M19
CONT_EN	N31	G23	
CPU_CLK	Y37	R29	T23
CS_N	W34	P29	

Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
DATA[31..0]	DATA[0]: AB33 DATA[1]: AB34 DATA[2]: AB35 DATA[3]: AB36 DATA[4]: AB37 DATA[5]: AC31 DATA[6]: AC32 DATA[7]: AC33 DATA[8]: AC34 DATA[9]: AC35 DATA[10]: AC36 DATA[11]: AC37 DATA[12]: AC38 DATA[13]: AC39 DATA[14]: AD31 DATA[15]: AD32 DATA[16]: AD33 DATA[17]: AD34 DATA[18]: AD35 DATA[19]: AD36 DATA[20]: AD37 DATA[21]: AD38 DATA[22]: AD39 DATA[23]: AE31 DATA[24]: AE32 DATA[25]: AE33 DATA[26]: AE34 DATA[27]: AE35 DATA[28]: AE36 DATA[29]: AE37 DATA[30]: AE38 DATA[31]: AE39	DATA[0]: V31 DATA[1]: V25 DATA[2]: V26 DATA[3]: V27 DATA[4]: V28 DATA[5]: V29 DATA[6]: V30 DATA[7]: W25 DATA[8]: W26 DATA[9]: W27 DATA[10]: W28 DATA[11]: W29 DATA[12]: W30 DATA[13]: W31 DATA[14]: Y25 DATA[15]: Y26 DATA[16]: Y27 DATA[17]: Y28 DATA[18]: Y29 DATA[19]: Y30 DATA[20]: Y31 DATA[21]: AA26 DATA[22]: AA27 DATA[23]: AA28 DATA[24]: AA29 DATA[25]: AA30 DATA[26]: AA31 DATA[27]: AB27 DATA[28]: AB28 DATA[29]: AB29 DATA[30]: AB30 DATA[31]: AB31	
DERR_N	V30	L24	
DIODE_IN	AH33	AB26	L18
DIODE_OUT	AJ33	AC26	L19
DTACK_N	U30	J25	
FH_PLL_CLKOUT	AD29	AC24	P17
FH_PLL_REFCLK	AE29	AD24	R20
GPIO[0]/DTACK_INV	AA33	AC27	J18
GPIO[1]/RW_INV	AC29	AB24	T21
GPIO[2]/IGN_PAR	W33	P28	M18

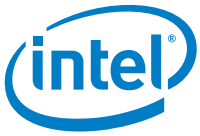


Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
GPIO[3]/SPI_SCK	R36	K31	H18
GPIO[4]/SPI_CS_N	R35	J30	G17
GPIO[5]/SPI_MOSI	R34	J31	G18
GPIO[6]/SPI_MISO	R37	K30	H17
GPIO[7]/EEP_EN1	R31	J23	M20
GPIO[8]/EEP_EN2	AF31	AD23	M21
GPIO[9]/AUTOBOOT	W37	N25	N21
GPIO[10]/PARITY_EVEN	AF29	AE23	T22
GPIO[11]/I2C_ADDR[0]	AF33	AE24	U19
GPIO[12]/I2C_ADDR[1]	AG30	AC28	U20
GPIO[13]/I2C_ADDR[2]	AG31	AC29	U21
GPIO[14]/DATA_HOLD	AG32	AC30	U22
GPIO[15]	AG33	AC31	U23
I2C_SCL	AB32	T27	R18
I2C_SDA	AB31	U27	R19
INTR_N	W36	P31	P19
LED_CLK	P29	G24	K17
LED_DATA0	R29	H24	K18
LED_DATA1	T29	J24	N18
LED_DATA2	U29	K24	P18
LED_EN	V29	K23	J17
MDC	V31	K27	N19
MDIO	V32	K28	N20
NC	N30, P30, P31, P32, P33, R30, R32, R33, R38, R39, T30, W32, Y32, AA32, AF32	G25, H23, H25, J26, J28, J29, K26, R27, AB23, AC23	G19, G20, G21, G22, G23, H19, H20, H21, H22, H23, J19, J20, J21, J22, J23, K19, K20, K21, K22, K23

Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P01_R{A..D}{P,N}	P01_RAN: AN35 P01_RAP: AN36 P01_RBN: AL35 P01_RBP: AL36 P01_RCN: AJ35 P01_RCP: AJ36 P01_RDN: AG35 P01_RDP: AG36	P01_RAN: AL27 P01_RAP: AL28 P01_RBN: AJ27 P01_RBP: AJ28 P01_RCN: AG27 P01_RCP: AG28 P01_RDN: AE27 P01_RDP: AE28	P01_RAN: W19 P01_RAP: Y19 P01_RBN: W2 P01_RBP: Y20 P01_RCN: W21 P01_RCP: Y21 P01_RDN: W22 P01_RDP: Y22
P01_T{A..D}{P,N}	P01_TAN: AN38 P01_TAP: AN39 P01_TBN: AL38 P01_TBP: AL39 P01_TCN: AJ38 P01_TCP: AJ39 P01_TDN: AG38 P01_TDP: AG39	P01_TAN: AL30 P01_TAP: AL31 P01_TBN: AJ30 P01_TBP: AJ31 P01_TCN: AG30 P01_TCP: AG31 P01_TDN: AE30 P01_TDP: AE31	P01_TAN: AB19 P01_TAP: AC19 P01_TBN: AB20 P01_TBP: AC20 P01_TCN: AB21 P01_TCP: AC21 P01_TDN: AB22 P01_TDP: AC22
P02_R{A..D}{P,N}	P02_RAN: N35 P02_RAP: N36 P02_RBN: L35 P02_RBP: L36 P02_RAN: N35 P02_RAP: N36 P02_RBN: L35 P02_RBP: L36	P02_RAN: G27 P02_RAP: G28 P02_RBN: E27 P02_RBP: E28 P02_RAN: C27 P02_RAP: C28 P02_RBN: A27 P02_RBP: A28	P02_RAN: E22 P02_RAP: D22 P02_RBN: E21 P02_RBP: D21 P02_RAN: E20 P02_RAP: D20 P02_RBN: E19 P02_RBP: D19
P02_T{A..D}{P,N}	P02_TAN: N38 P02_TAP: N39 P02_TBN: L38 P02_TBP: L39 P02_TCN: J38 P02_TCP: J39 P02_TDN: G38 P02_TDP: G39	P02_TAN: G30 P02_TAP: G31 P02_TBN: E30 P02_TBP: E31 P02_TCN: C30 P02_TCP: C31 P02_TDN: A30 P02_TDP: A31	P02_TAN: B22 P02_TAP: A22 P02_TBN: B21 P02_TBP: A21 P02_TCN: B20 P02_TCP: A20 P02_TDN: B19 P02_TDP: A19
P03_R{A..D}{P,N}	P03_RAN: AR33 P03_RAP: AT33 P03_RBN: AR35 P03_RBP: AT35 P03_RCN: AR37 P03_RCP: AT37 P03_RDN: AR39 P03_RDP: AT39	P03_RAN: AG19 P03_RAP: AH19 P03_RBN: AG21 P03_RBP: AH21 P03_RCN: AG23 P03_RCP: AH23 P03_RDN: AG25 P03_RDP: AH25	P03_RAN: W17 P03_RAP: Y17

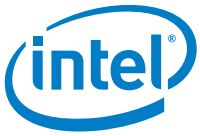


Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P03_T{A..D}{P,N}	P03_TAN: AV33 P03_TAP: AW33 P03_TBN: AV35 P03_TBP: AW35 P03_TCN: AV37 P03_TCP: AW37 P03_TDN: AV39 P03_TDP: AW39	P03_TAN: AK19 P03_TAP: AL19 P03_TBN: AK21 P03_TBP: AL21 P03_TCN: AK23 P03_TCP: AL23 P03_TDN: AK25 P03_TDP: AL25	P03_TAN: AB17 P03_TAP: AC17
P04_R{A..D}{P,N}	P04_RAN: E39 P04_RAP: D39 P04_RBN: E37 P04_RBP: D37 P04_RCN: E35 P04_RCP: D35 P04_RDN: E33 P04_RDP: D33	P04_RAN: E25 P04_RAP: D25 P04_RBN: E23 P04_RBP: D23 P04_RCN: E21 P04_RCP: D21 P04_RDN: E19 P04_RDP: D19	P04_RAN: E17 P04_RAP: D17
P04_T{A..D}{P,N}	P04_TAN: B39 P04_TAP: A39 P04_TBN: B37 P04_TBP: A37 P04_TCN: B35 P04_TCP: A35 P04_TDN: B33 P04_TDP: A33	P04_TAN: B25 P04_TAP: A25 P04_TBN: B23 P04_TBP: A23 P04_TCN: B21 P04_TCP: A21 P04_TDN: B19 P04_TDP: A19	P04_TAN: B17 P04_TAP: A17
P05_R{A..D}{P,N}	P05_RAN: AJ25 P05_RAP: AK25 P05_RBN: AJ27 P05_RBP: AK27 P05_RCN: AJ29 P05_RCP: AK29 P05_RDN: AJ31 P05_RDP: AK31	P05_RAN: AG11 P05_RAP: AH11 P05_RBN: AG13 P05_RBP: AH13 P05_RCN: AG15 P05_RCP: AH15 P05_RDN: AG17 P05_RDP: AH17	P05_RAN: W11 P05_RAP: Y11 P05_RBN: W12 P05_RBP: Y12 P05_RCN: W13 P05_RCP: Y13 P05_RDN: W14 P05_RDP: Y14
P05_T{A..D}{P,N}	P05_TAN: AM25 P05_TAP: AN25 P05_TBN: AM27 P05_TBP: AN27 P05_TCN: AM29 P05_TCP: AN29 P05_TDN: AM31 P05_TDP: AN31	P05_TAN: AK11 P05_TAP: AL11 P05_TBN: AK13 P05_TBP: AL13 P05_TCN: AK15 P05_TCP: AL15 P05_TDN: AK17 P05_TDP: AL17	P05_TAN: AB11 P05_TAP: AC11 P05_TBN: AB12 P05_TBP: AC12 P05_TCN: AB13 P05_TCP: AC13 P05_TDN: AB14 P05_TDP: AC14

Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P06_R{A..D}{P,N}	P06_RAN: L31 P06_RAP: K31 P06_RBN: L29 P06_RBP: K29 P06_RCN: L27 P06_RCP: K27 P06_RDN: L25 P06_RDP: K25	P06_RAN: E17 P06_RAP: D17 P06_RBN: E15 P06_RBP: D15 P06_RCN: E13 P06_RCP: D13 P06_RDN: E11 P06_RDP: D11	P06_RAN: E14 P06_RAP: D14 P06_RBN: E13 P06_RBP: D13 P06_RCN: E12 P06_RCP: D12 P06_RDN: E11 P06_RDP: D11
P06_T{A..D}{P,N}	P06_TAN: H31 P06_TAP: G31 P06_TBN: H29 P06_TBP: G29 P06_TCN: H27 P06_TCP: G27 P06_TDN: H25 P06_TDP: G25	P06_TAN: B17 P06_TAP: A17 P06_TBN: B15 P06_TBP: A15 P06_TCN: B13 P06_TCP: A13 P06_TDN: B11 P06_TDP: A11	P06_TAN: B14 P06_TAP: A14 P06_TBN: B13 P06_TBP: A13 P06_TCN: B12 P06_TCP: A12 P06_TDN: B11 P06_TDP: A11
P07_R{A..D}{P,N}	P07_RAN: AR25 P07_RAP: AT25 P07_RBN: AR27 P07_RBP: AT27 P07_RCN: AR29 P07_RCP: AT29 P07_RDN: AR31 P07_RDP: AT31	P07_RAN: AG3 P07_RAP: AH3 P07_RBN: AG5 P07_RBP: AH5 P07_RCN: AG7 P07_RCP: AH7 P07_RDN: AG9 P07_RDP: AH9	P07_RAN: W6 P07_RAP: Y6 P07_RBN: W7 P07_RBP: Y7 P07_RCN: W8 P07_RCP: Y8 P07_RDN: W9 P07_RDP: Y9
P07_T{A..D}{P,N}	P07_TAN: AV25 P07_TAP: AW25 P07_TBN: AV27 P07_TBP: AW27 P07_TCN: AV29 P07_TCP: AW29 P07_TDN: AV31 P07_TDP: AW31	P07_TAN: AK3 P07_TAP: AL3 P07_TBN: AK5 P07_TBP: AL5 P07_TCN: AK7 P07_TCP: AL7 P07_TDN: AK9 P07_TDP: AL9	P07_TAN: AB6 P07_TAP: AC6 P07_TBN: AB7 P07_TBP: AC7 P07_TCN: AB8 P07_TCP: AC8 P07_TDN: AB9 P07_TDP: AC9
P08_R{A..D}{P,N}	P08_RAN: E31 P08_RAP: D31 P08_RBN: E29 P08_RBP: D29 P08_RCN: E27 P08_RCP: D27 P08_RDN: E25 P08_RDP: D25	P08_RAN: E9 P08_RAP: D9 P08_RBN: E7 P08_RBP: D7 P08_RCN: E5 P08_RCP: D5 P08_RDN: E3 P08_RDP: D3	P08_RAN: E9 P08_RAP: D9 P08_RBN: E8 P08_RBP: D8 P08_RCN: E7 P08_RCP: D7 P08_RDN: E6 P08_RDP: D6

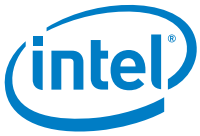


Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P08_T{A..D}{P,N}	P08_TAN: B31 P08_TAP: A31 P08_TBN: B29 P08_TBP: A29 P08_TCN: B27 P08_TCP: A27 P08_TDN: B25 P08_TDP: A25	P08_TAN: B9 P08_TAP: A9 P08_TBN: B7 P08_TBP: A7 P08_TCN: B5 P08_TCP: A5 P08_TDN: B3 P08_TDP: A3	P08_TAN: B9 P08_TAP: A9 P08_TBN: B8 P08_TBP: A8 P08_TCN: B7 P08_TCP: A7 P08_TDN: B6 P08_TDP: A6
P09_R{A..D}{P,N}	P09_RAN: AJ17 P09_RAP: AK17 P09_RBN: AJ19 P09_RBP: AK19 P09_RCN: AJ21 P09_RCP: AK21 P09_RDN: AJ23 P09_RDP: AK23	P09_RAN: AB9 P09_RAP: AA9	P09_RAN: W1 P09_RAP: Y1 P09_RBN: W2 P09_RBP: Y2 P09_RCN: W3 P09_RCP: Y3 P09_RDN: W4 P09_RDP: W4
P09_T{A..D}{P,N}	P09_TAN: AM17 P09_TAP: AN17 P09_TBN: AM19 P09_TBP: AN19 P09_TCN: AM21 P09_TCP: AN21 P09_TDN: AM23 P09_TDP: AN23	P09_TAN: AE9 P09_TAP: AD9	P09_TAN: AB1 P09_TAP: AC1 P09_TBN: AB2 P09_TBP: AC2 P09_TCN: AB3 P09_TCP: AC3 P09_TDN: AB4 P09_TDP: AC4
P10_R{A..D}{P,N}	P10_RAN: L23 P10_RAP: K23 P10_RBN: L21 P10_RBP: K21 P10_RCN: L19 P10_RCP: K19 P10_RDN: L17 P10_RDP: K17	P10_RAN: L9 P10_RAP: K9	P10_RAN: E4 P10_RAP: D4 P10_RBN: E3 P10_RBP: D3 P10_RCN: E2 P10_RCP: D2 P10_RDN: E1 P10_RDP: D1
P10_T{A..D}{P,N}	P10_TAN: H23 P10_TAP: G23 P10_TBN: H21 P10_TBP: G21 P10_TCN: H19 P10_TCP: G19 P10_TDN: H17 P10_TDP: G17	P10_TAN: H9 P10_TAP: G9	P10_TAN: B4 P10_TAP: A4 P10_TBN: B3 P10_TBP: A3 P10_TCN: B2 P10_TCP: A2 P10_TDN: B1 P10_TDP: A1

Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P11_R{A..D}{P,N}	P11_RAN: AR17 P11_RAP: AT17 P11_RBN: AR19 P11_RBP: AT19 P11_RCN: AR21 P11_RCP: AT21 P11_RDN: AR23 P11_RDP: AT23	P11_RAN: AB7 P11_RAP: AA7	
P11_T{A..D}{P,N}	P11_TAN: AV17 P11_TAP: AW17 P11_TBN: AV19 P11_TBP: AW19 P11_TCN: AV21 P11_TCP: AW21 P11_TDN: AV23 P11_TDP: AW23	P11_TAN: AE7 P11_TAP: AD7	
P12_R{A..D}{P,N}	P12_RAN: E23 P12_RAP: D23 P12_RBN: E21 P12_RBP: D21 P12_RCN: E19 P12_RCP: D19 P12_RDN: E17 P12_RDP: D17	P12_RAN: L7 P12_RAP: K7	
P12_T{A..D}{P,N}	P12_TAN: B23 P12_TAP: A23 P12_TBN: B21 P12_TBP: A21 P12_TCN: B19 P12_TCP: A19 P12_TDN: B17 P12_TDP: A17	P12_TAN: H7 P12_TAP: G7	
P13_R{A..D}{P,N}	P13_RAN: AR9 P13_RAP: AT9 P13_RBN: AR11 P13_RBP: AT11 P13_RCN: AR13 P13_RCP: AT13 P13_RDN: AR15 P13_RDP: AT15	P13_RAN: AG1 P13_RAP: AH1	P13_RAN: U5 P13_RAP: U4

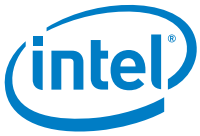


Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P13_T{A..D}{P,N}	P13_TAN: AV9 P13_TAP: AW9 P13_TBN: AV11 P13_TBP: AW11 P13_TCN: AV13 P13_TCP: AW13 P13_TDN: AV15 P13_TDP: AW15	P13_TAN: AK1 P13_TAP: AL1	P13_TAN: U2 P13_TAP: U1
P14_R{A..D}{P,N}	P14_RAN: E15 P14_RAP: D15 P14_RBN: E13 P14_RBP: D13 P14_RCN: E11 P14_RCP: D11 P14_RDN: E9 P14_RDP: D9	P14_RAN: E1 P14_RAP: D1	P14_RAN: G5 P14_RAP: G4
P14_T{A..D}{P,N}	P14_TAN: B15 P14_TAP: A15 P14_TBN: B13 P14_TBP: A13 P14_TCN: B11 P14_TCP: A11 P14_TDN: B9 P14_TDP: A9	P14_TAN: B1 P14_TAP: A1	P14_TAN: G2 P14_TAP: G1
P15_R{A..D}{P,N}	P15_RAN: AJ9 P15_RAP: AK9 P15_RBN: AJ11 P15_RBP: AK11 P15_RCN: AJ13 P15_RCP: AK13 P15_RDN: AJ15 P15_RDP: AK15	P15_RAN: AC5 P15_RAP: AC4	
P15_T{A..D}{P,N}	P15_TAN: AM9 P15_TAP: AN9 P15_TBN: AM11 P15_TBP: AN11 P15_TCN: AM13 P15_TCP: AN13 P15_TDN: AM15 P15_TDP: AN15	P15_TAN: AC2 P15_TAP: AC1	

Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P16_R{A..D}{P,N}	P16_RAN: L15 P16_RAP: K15 P16_RBN: L13 P16_RBP: K13 P16_RCN: L11 P16_RCP: K11 P16_RDN: L9 P16_RDP: K9	P16_RAN: J5 P16_RAP: J4	
P16_T{A..D}{P,N}	P16_TAN: H15 P16_TAP: G15 P16_TBN: H13 P16_TBP: G13 P16_TCN: H11 P16_TCP: G11 P16_TDN: H9 P16_TDP: G9	P16_TAN: J2 P16_TAP: J1	
P17_R{A..D}{P,N}	P17_RAN: AA11 P17_RAP: AA10 P17_RBN: AC11 P17_RBP: AC10 P17_RCN: AE11 P17_RCP: AE10 P17_RDN: AG11 P17_RDP: AG10	P17_RAN: W5 P17_RAP: W4	
P17_T{A..D}{P,N}	P17_TAN: AA8 P17_TAP: AA7 P17_TBN: AC8 P17_TBP: AC7 P17_TCN: AE8 P17_TCP: AE7 P17_TDN: AG8 P17_TDP: AG7	P17_TAN: W2 P17_TAP: W1	
P18_R{A..D}{P,N}	P18_RAN: N11 P18_RAP: N10 P18_RBN: R11 P18_RBP: R10 P18_RCN: U11 P18_RCP: U10 P18_RDN: W11 P18_RDP: W10	P18_RAN: N5 P18_RAP: N4	

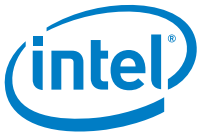


Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P18_T{A..D}{P,N}	P18_TAN: N8 P18_TAP: N7 P18_TBN: R8 P18_TBP: R7 P18_TCN: U8 P18_TCP: U7 P18_TDN: W8 P18_TDP: W7	P18_TAN: N2 P18_TAP: N1	
P19_R{A..D}{P,N}	P19_RAN: AR1 P19_RAP: AT1 P19_RBN: AR3 P19_RBP: AT3 P19_RCN: AR5 P19_RCP: AT5 P19_RDN: AR7 P19_RDP: AT7	P19_RAN: AE5 P19_RAP: AE4	P19_RAN: T5 P19_RAP: T4
P19_T{A..D}{P,N}	P19_TAN: AV1 P19_TAP: AW1 P19_TBN: AV3 P19_TBP: AW3 P19_TCN: AV5 P19_TCP: AW5 P19_TDN: AV7 P19_TDP: AW7	P19_TAN: AE2 P19_TAP: AE1	P19_TAN: T2 P19_TAP: T1
P20_R{A..D}{P,N}	P20_RAN: E7 P20_RAP: D7 P20_RBN: E5 P20_RBP: D5 P20_RCN: D3 P20_RCP: D3 P20_RDN: E1 P20_RDP: D1	P20_RAN: G5 P20_RAP: G4	P20_RAN: H5 P20_RAP: H4
P20_T{A..D}{P,N}	P20_TAN: B7 P20_TAP: A7 P20_TBN: B5 P20_TBP: A5 P20_TCN: B3 P20_TCP: A3 P20_TDN: B1 P20_TDP: A1	P20_TAN: G2 P20_TAP: G1	P20_TAN: H2 P20_TAP: H1

Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P21_R{A..D}{P,N}	P21_RAN: AJ1 P21_RAP: AK1 P21_RBN: AJ3 P21_RBP: AK3 P21_RCN: AJ5 P21_RCP: AK5 P21_RDN: AJ7 P21_RDP: AK7	P21_RAN: AA5 P21_RAP: AA4	P21_RAN: P5 P21_RAP: P4
P21_T{A..D}{P,N}	P21_TAN: AM1 P21_TAP: AN1 P21_TBN: AM3 P21_TBP: AN3 P21_TCN: AM5 P21_TCP: AN5 P21_TDN: AM7 P21_TDP: AN7	P21_TAN: AA2 P21_TAP: AA1	P21_TAN: P2 P21_TAP: P1
P22_R{A..D}{P,N}	P22_RAN: L7 P22_RAP: K7 P22_RBN: L5 P22_RBP: K5 P22_RCN: L3 P22_RCP: K3 P22_RDN: L1 P22_RDP: K1	P22_RAN: L5 P22_RAP: L4	P22_RAN: K5 P22_RAP: K4
P22_T{A..D}{P,N}	P22_TAN: H7 P22_TAP: G7 P22_TBN: H5 P22_TBP: G5 P22_TCN: H33 P22_TCP: G3 P22_TDN: H1 P22_TDP: G1	P22_TAN: L2 P22_TAP: L1	P22_TAN: K2 P22_TAP: K1
P23_R{A..D}{P,N}	P23_RAN: AA5 P23_RAP: AA4 P23_RBN: AC5 P23_RBP: AC4 P23_RCN: AE5 P23_RCP: AE4 P23_RDN: AG5 P23_RDP: AG4	P23_RAN: U5 P23_RAP: U4	P23_RAN: N5 P23_RAP: N4

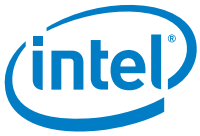


Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
P23_T{A..D}{P,N}	P23_TAN: AA2 P23_TAP: AA1 P23_TBN: AC2 P23_TBP: AC1 P23_TCN: AE2 P23_TCP: AE1 P23_TDN: AG2 P23_TDP: AG1	P23_TAN: U2 P23_TAP: U1	P23_TAN: N2 P23_TAP: N1
P24_R{A..D}{P,N}	P24_RAN: N5 P24_RAP: N4 P24_RBN: R5 P24_RBP: R4 P24_RCN: U5 P24_RCP: U4 P24_RDN: W5 P24_RDP: W4	P24_RAN: R5 P24_RAP: R4	P24_RAN: L5 P24_RAP: L4
P24_T{A..D}{P,N}	P24_TAN: N2 P24_TAP: N1 P24_TBN: R2 P24_TBP: R1 P24_TCN: U2 P24_TCP: U1 P24_TDN: W2 P24_TDP: W1	P24_TAN: R2 P24_TAP: R1	P24_TAN: L2 P24_TAP: L1
PAR[3..0]	PAR[0]: AA34 PAR[1]: AA35 PAR[2]: AA36 PAR[3]: AA37	PAR[0]: T28 PAR[1]: T29 PAR[2]: U28 PAR[3]: U29	
REFCLK[4..1]{A,B}	RCK1AN: AG27 RCK1AP: AF27 RCK1BN: AG26 RCK1BP: AF26 RCK2AN: P27 RCK2AP: N27 RCK2BN: P26 RCK2BP: N2 RCK3AN: AG13 RCK3AP: AF13 RCK3BN: AG14 RCK3BP: AF14 RCK4AN: P13 RCK4AP: N13 RCK4BN: P14 RCK4BP: N14	RCK1AN: AD21 RCK1AP: AC21 RCK1BN: AD20 RCK1BP: AC20 RCK2AN: J21 RCK2AP: H21 RCK2BN: J20 RCK2BP: H20 RCK3AN: AD11 RCK3AP: AC11 RCK3BN: AD12 RCK3BP: AC12 RCK4AN: J11 RCK4AP: H11 RCK4BN: J12 RCK4BP: H12	RCK1AN: U16 RCK1AP: T16 RCK1BN: U15 RCK1BP: T15 RCK2AN: H16 RCK2AP: G16 RCK2BN: H15 RCK2BP: G15 RCK3AN: U7 RCK3AP: T7 RCK3BN: U8 RCK3BP: T8 RCK4AN: H7 RCK4AP: G7 RCK4BN: H8 RCK4BP: G8

Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
RESERVED. Leave open.			P20, P21, P22, R21, R22, R23
RREF[24..1]	RREF01: AK36 RREF02: K36 RREF03: AT36 RREF04: D36 RREF05: AK28 RREF06: K28 RREF07: AT28 RREF08: D28 RREF09: AK20 RREF10: K20 RREF11: AT20 RREF12: D20 RREF13: AT12 RREF14: D12 RREF15: AK12 RREF16: K12 RREF17: AD10 RREF18: T10 RREF19: AT4 RREF20: D4 RREF21: AK4 RREF22: K4 RREF23: AD4 RREF24: T4	RREF01: AH28 RREF02: D28 RREF03: AH22 RREF04: D22 RREF05: AH14 RREF06: D14 RREF07: AH6 RREF08: D6 RREF09: AD8 RREF10: H8 RREF11: AD6 RREF12: H6 RREF13: AH2 RREF14: D2 RREF15: V6 RREF16: P7 RREF17: U6 RREF18: R7 RREF19: V7 RREF20: P6 RREF21: U7 RREF22: R6 RREF23: T7 RREF24: T6	RREF01: Y23 RREF02: D23 RREF03: Y18 RREF04: D18 RREF05: Y15 RREF06: D15 RREF07: Y10 RREF08: D10 RREF09: Y5 RREF10: D5 RREF13: V4 RREF14: F4 RREF19: R5 RREF20: J5 RREF21: R4 RREF22: J4 RREF23: M5 RREF24: M4
RW_N	Y36	R28	
RXEOT_N	Y35	P27	
RXRDY_N	Y33	P25	
TCK	AC30	AB25	T19
TDI	AB30	AA25	T18
TDO	AF30	AE25	U18
TEST_RESET_N	Y30	T26	
TESTMODE	T31	L23	R17
TMS	AE30	AD25	U17
TRST_N	AD30	AC25	T20
TXRDY_N	Y34	P26	

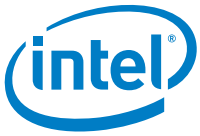


Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
VDD	D8, D16, D24, D32, E8, E16, E24, E32, F4, F5, F6, F7, F8, F9, F15, F16, F17, F23, F24, F25, F31, F32, F33, F34, F35, F36, G34, H34, J34, K8, K16, K24, K32, K34, L8, L16, L24, L32, L34, M4, M5, M6, M7, M8, M9, M10, M11, N6, P6, P17, P18, P22, P23, R17, R18, R22, R23, T14, T15, T16, T17, T18, T19, T20, T21, T22, T23, T24, T25, T26, U14, U15, U16, U17, U18, U19, U20, U21, U22, U23, U24, U25, U26, W6, Y4, Y5, Y6, Y14, Y15, Y16, Y17, Y18, Y22, Y23, Y24, AA6, AC14, AC15, AC16, AC17, AC18, AC19, AC20, AC21, AC22, AC23, AC24, AC25, AC26, AD14, AD15, AD16, AD17, AD18, AD19, AD20, AD21, AD22, AD23, AD24, AD25, AD26, AE17, AE18, AE22, AE23, AF6, AF17, AF18, AF22, AF23, AG6, AH4, AH5, AH6, AH7, AH8, AH9, AH10, AH11, AJ8, AJ16, AJ24, AJ32, AJ34, AK8, AK16, AK24, AK32, AK34, AL34, AM34, AN34, AP4, AP5, AP6, AP7, AP8, AP9, AP15, AP16, AP17, AP23, AP24, AP25, AP31, AP32, AP33, AP34, AP35, AP36, AR8, AR16, AR24, AR32, AT8, AT16, AT24, AT32	K15, K16, K17, L15, L16, L17, M10, M11, M12, M13, M14, M15, M16, M17, M18, M19, M20, M21, M22, N10, N11, N12, N13, N14, N15, N16, N17, N18, N19, N20, N21, N22, R13, R14, R18, R19, T10, T11, T12, T13, T14, T18, T19, T20, U13, U14, U18, U19, W10, W11, W12, W13, W14, W15, W16, W17, W18, W19, W20, W21, W22, Y10, Y11, Y12, Y13, Y14, Y15, Y16, Y17, Y18, Y19, Y20, Y21, Y22, AA15, AA16, AA17, AB15, AB16, AB17	C2, C5, C10, C18, C23, G10, G11, G12, G13, G14, H9, H11, H13, J6, J8, J10, J12, J14, J16, K6, K7, K9, K11, K13, K15, L8, L10, L12, L14, L16, M7, M9, M11, M13, M15, N8, N10, N12, N14, N16, P6, P7, P9, P11, P13, P15, R6, R8, R10, R12, R14, R16, T9, T11, T13, U10, U11, U12, U13, U14, AA2, AA5, AA10, AA18, AA23
VDD33	W25, W26, W38, W39, Y25, Y26, Y38, Y39, AA25, AA26, AA38, AA39	R21, R22, R30, R31, T21, T22, T30, T31, U21, U22, U30, U31	L17, M17, M22, M23, N17, N22, N23
VDDA	C4, C12, C20, C28, C36, J4, J12, J20, J28, K37, T3, T9, AD3, AD9, AK37, AL4, AL12, AL20, AL28, AU4, AU12, AU20, AU28, AU36	C2, C6, C14, C22, D29, F3, J8, K3, P3, V3, AB3, AC8, AF3, AH29, AJ2, AJ6, AJ14, AJ22	E5, E10, E15, E18, E23, F5, H6, M6, T6, V5, W5, W10, W15, W18, W23
VDDA33	AB29	AB29	T17

Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
VDDX	B4, B12, B20, B28, B36, C2, C3, C5, C6, C10, C11, C13, C14, C18, C19, C21, C22, C26, C27, C29, C30, C34, C35, C37, C38, H4, H12, H20, H28, H37, J2, J3, J5, J6, J10, J11, J13, J14, J18, J19, J21, J22, J26, J27, J29, J30, J37, K38, L37, M37, N12, N28, P3, P9, P12, P28, R3, R9, T2, T8, U3, U9, V3, V9, AB3, AB9, AC3, AC9, AD2, AD8, AE3, AE9, AF3, AF9, AF12, AF28, AG12, AG28, AH37, AJ37, AK38, AL2, AL3, AL5, AL6, AL10, AL11, AL13, AL14, AL18, AL19, AL21, AL22, AL26, AL27, AL29, AL30, AL37, AM4, AM12, AM20, AM28, AM37, AU2, AU3, AU5, AU6, AU10, AU11, AU13, AU14, AU18, AU19, AU21, AU22, AU26, AU27, AU29, AU30, AU34, AU35, AU37, AU38, AV4, AV12, AV20, AV28, AV36	B2, B6, B14, B22, B29, C4, C5, C7, C8, C12, C13, C15, C16, C20, C21, C23, C24, C29, D30, E29, F2, F4, F6, F7, F8, F9, F29, G15, G16, G17, H3, J6, K2, K4, K8, M3, M6, M7, N6, N7, P2, P4, T3, V2, V4, W6, W7, Y3, Y6, Y7, AB2, AB4, AB8, AC6, AD3, AE15, AE16, AE17, AF2, AF4, AF6, AF7, AF8, AF9, AF29, AG29, AH30, AJ4, AJ5, AJ7, AJ8, AJ12, AJ13, AJ15, AJ16, AJ20, AJ21, AJ23, AJ24, AJ29, AK2, AK6, AK14, AK22, AK29	C3, C7, C8, C12, C13, C16, C20, C21, D16, G3, H3, K3, L3, N3, P3, T3, U3, Y16, AA3, AA7, AA8, AA12, AA13, AA16, AA20, AA21
VSS	A2, A4, A6, A8, A10, A12, A14, A16, A18, A20, A22, A24, A26, A28, A30, A32, A34, A36, A38, B8, B16, B24, B32, C1, C7, C8, C9, C15, C16, C17, C23, C24, C25, C31, C32, C33, C39, E2, E4, E6, E10, E12, E14, E18, E20, E22, E26, E28, E30, E34, E36, E38, F1, F2, F3, F10, F11, F12, F13, F14, F18, F19, F20, F21, F22, F26, F27, F28, F29, F30, F37, F38, F39, G2, G4, G6, G8, G10, G12, G14, G16, G18, G20, G22, G24, G26, G28, G30, G32, G33, G37, H8, H16, H24, H32, H33, H35, H39, J1, J7, J8, J9, J15, J16, J17, J23, J24, J25, J31, J32, J33, K33, K35, K39, L2, L4, L6, L10, L12, L14, L18, L20, L22, L26, L28, L30, L33, M1, M2, M3, M12, M13, M14, M26, M27, M28, M29, M30, M31, M32, M33, M34, M35, M39, N3, N9, N29, N32, N34, N37, P1, P5, P7, P11, P15, P16, P19, P20, P21, P24, P25, P34, P35, P36, P37, P38, P39, R6, R14, R15, R16, R19, R20, R21, R24, R25, R26, T1, T5, T6, T7, T11, U6, V1, V5, V6, V7, V11, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23,V24, V25, V26, V38, V39, W3, W9, W14, W15, W16, W17, W18, W19, W20, W21, W22, W23, W24, W29, W30, W31, Y1, Y2, Y3, Y7, Y8, Y9, Y10, Y11, Y19, Y20, Y21, Y29, Y31, AA3, AA9, AA14, AA15, AA16, AA17, AA18,	A2, A4, A6, A8, A10, A12, A14, A16, A18, A20, A22, A24, A26, A29, B10, B18, B26, B27, B31, C1, C3, C9, C10, C11, C17, C18, C19, C25, C26, D10, D18, D26, D27, D31, E4, E6, E8, E10, E12, E14, E16, E18, E20, E22, E24, E26, F1, F5, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24, F25, F26, F27, F31, G3, G10, G11, G12, G13, G14, G18, G19, G20, G21, G22, G26, G29, H1, H5, H10, H22, H26, H27, H28, H29, H30, H31, J3, J7, J9, J10, J22, K1, K5, K6, K10, K11, K12, K13, K14, K18, K19, K20, K21, K22, L3, L6, L8, L10, L11, L12, L13, L14, L18, L19, L20, L21, L22, M1, M5, N3, P1, P5, P10, P11, P12, P13, P14, P15, P16, P17, P18, P19, P20, P21, P22, R3, R10, R11, R12, R15, R16, R17, R20, R25, R26, T1, T5, T15, T16, T17, T25, U3, U10, U11, U12, U15, U16, U17, U20, U25, U26, V1, V5, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, W3, Y1, Y5, AA3, AA6, AA8, AA10, AA11, AA12, AA13, AA14, AA18, AA19, AA20, AA21, AA22, AA23, AB1, AB5, AB6,	A5, A10, A15, A16, A18, A23, C1, C4, C6, C9, C11, C14, C15, C17, C19, C22, E16, F1, F3, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, G6, G9, H10, H12, H14, J1, J3, J7, J9, J11, J13, J15, K8, K10, K12, K14, K16, L6, L7, L9, L11, L13, L15, L20, L21, L22, L23, M1, M3, M8, M10, M12, M14, M16, N6, N7, N9, N11, N13, N15, P8, P10, P12, P14, P16, P23, R1, R3, R7, R9, R11, R13, R15, T10, T12, T14, U6, U9, V1, V3, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, W16, AA1, AA4, AA6, AA9, AA11, AA14, AA15, AA17, AA19, AA22, AC5, AC10, AC15, AC16, AC18, AC23

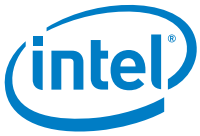


Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
	AA19, AA20, AA21, AA22, AA23, AA24, AA29, AA30, AA31, AB1, AB5, AB6, AB7, AB11, AB14, AB15, AB16, AB17, AB18, AB19, AB20, AB21, AB22, AB23, AB24, AB25, AB26, AB38, AB39, AC6, AD1, AD5, AD6, AD7, AD11, AE6, AE14, AE15, AE16, AE19, AE20, AE21, AE24, AE25, AE26, AF1, AF5, AF7, AF11, AF15, AF16, AF19, AF20, AF21, AF24, AF25, AF34, AF35, AF36, AF37, AF38, AF39, AG3, AG9, AG29, AG34, AG37, AH1, AH2, AH3, AH12, AH13, AH14, AH26, AH27, AH28, AH29, AH30, AH31, AH32, AH34, AH35, AH39, AJ2, AJ4, AJ6, AJ10, AJ12, AJ14, AJ18, AJ20, AJ22, AJ26, AJ28, AJ30, AK33, AK35, AK39, AL1, AL7, AL8, AL9, AL15, AL16, AL17, AL23, AL24, AL25, AL31, AL32, AL33, AM8, AM16, AM24, AM32, AM33, AM35, AM39, AN2, AN4, AN6, AN8, AN10, AN12, AN14, AN16, AN18, AN20, AN22, AN24, AN26, AN28, AN30, AN32, AN33, AN37, AP1, AP2, AP3, AP10, AP11, AP12, AP13, AP14, AP18, AP19, AP20, AP21, AP22, AP26, AP27, AP28, AP29, AP30, AP37, AP38, AP39, AR2, AR4, AR6, AR10, AR12, AR14, AR18, AR20, AR22, AR26, AR28, AR30, AR34, AR36, AR38, AU1, AU7, AU8, AU9, AU15, AU16, AU17, AU23, AU24, AU25, AU31, AU32, AU33, AU39, AV8, AV16, AV24, AV32, AW2, AW4, AW6, AW8, AW10, AW12, AW14, AW16, AW18, AW20, AW22, AW24, AW26, AW28, AW30, AW32, AW34, AW36, AW38	AB10, AB11, AB12, AB13, AB14, AB18, AB19, AB20, AB21, AB22, AC3, AC7, AC9, AC10, AC22, AD1, AD5, AD10, AD22, AD26, AD27, AD28, AD29, AD30, AD31, AE3, AE10, AE11, AE12, AE13, AE14, AE18, AE19, AE20, AE21, AE22, AE26, AE29, AF1, AF5, AF10, AF11, AF12, AF13, AF14, AF15, AF16, AF17, AF18, AF19, AF20, AF21, AF22, AF23, AF24, AF25, AF26, AF27, AF31, AG4, AG6, AG8, AG10, AG12, AG14, AG16, AG18, AG20, AG22, AG24, AG26, AH10, AH18, AH26, AH27, AH31, AJ1, AJ3, AJ9, AJ10, AJ11, AJ17, AJ18, AJ19, AJ25, AJ26, AK10, AK18, AK26, AK27, AK31, AL2, AL4, AL6, AL8, AL10, AL12, AL14, AL16, AL18, AL20, AL22, AL24, AL26, AL29	
VTT[24..1]	VTT01: AH36, AH38, AM36, AM38 VTT02: H36, H38, M36, M38 VTT03: AT34, AT38, AV34, AV38 VTT04: B34, B38, D34, D38 VTT05: AK26, AK30, AM26, AM30 VTT06: H26, H30, K26, K30 VTT07: AT26, AT30, AV26, AV30	VTT01: AF28, AF30, AK28, AK30 VTT02 B28, B30, F28, F30 VTT03: AH20, AH24, AK20, AK24 VTT04: B20, B24, D20, D24 VTT05: AH12, AH16, AK12, AK16 VTT06: B12, B16, D12, D16 VTT07: AH4, AH8, AK4, AK8	VTT01: AB23 VTT02: B23 VTT05: AB15 VTT06: B15 VTT07: AB10

Table 16-4 PIN Locations — Alphabetical Order (Continued)

Signal	Pin Location (40 mm Package)	Pin Location (32 mm Package)	Pin Location (25 mm Package)
	VTT08: B26, B30, D26, D30 VTT09: AK18, AK22, AM18, AM22 VTT10: H18, H22, K18, K22 VTT11: AT18, AT22, AV18, AV22 VTT12: B18, B22, D18, D22 VTT13: AT10, AT14, AV10, AV14 VTT14: B10, B14, D10, D14 VTT15: AK10, AK14, AM10, AM14 VTT16: H10, H14, K10, K14 VTT17: AB8, AB10, AF8, AF10 VTT18: P8, P10, V8, V10 VTT19: AT2, AT6, AV2, AV6 VTT20: B2, B6, D2, D6 VTT21: AK2, AK6, AM2, AM6 VTT22: H2, H6, K2, K6 VTT23: AB2, AB4, AF2, AF4 VTT24: P2, P4, V2, V4	VTT08: B4, B8, D4, D8 VTT09: AE8 VTT10: G8 VTT11: AE6 VTT12: G6 VTT13: AG2 VTT14: E2 VTT15: AD4 VTT16: H2 VTT17: Y4 VTT18: M2 VTT19: AD2 VTT20: H4 VTT21: Y2 VTT22: M4 VTT23: T2 VTT24: T4	VTT08: B10 VTT09: AB5 VTT10: B5
VTT (no port association)			B16, B18, F2, J2, M2, R2, V2, AB16, AB18

§ §

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Intel:

EZFM4224F1433D S LJQL FBFM4410F529E S LJM7 EZFM4112F897E S LJPB EZFM4112F897L S LJUV EZFM4212F1433C S LJMW EZFM4224F1433C S LKAB

OCEAN CHIPS
Океан Электроники
Поставка электронных компонентов

Компания «Океан Электроники» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Поставка оригинальных импортных электронных компонентов напрямую с производств Америки, Европы и Азии, а так же с крупнейших складов мира;
- Широкая линейка поставок активных и пассивных импортных электронных компонентов (более 30 млн. наименований);
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Помощь Конструкторского Отдела и консультации квалифицированных инженеров;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Поставка электронных компонентов под контролем ВП;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- При необходимости вся продукция военного и аэрокосмического назначения проходит испытания и сертификацию в лаборатории (по согласованию с заказчиком);
- Поставка специализированных компонентов военного и аэрокосмического уровня качества (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Actel, Aeroflex, Peregrine, VPT, Syfer, Eurofarad, Texas Instruments, MS Kennedy, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits, General Dynamics и др.);

Компания «Океан Электроники» является официальным дистрибьютором и эксклюзивным представителем в России одного из крупнейших производителей разъемов военного и аэрокосмического назначения «**JONHON**», а так же официальным дистрибьютором и эксклюзивным представителем в России производителя высокотехнологичных и надежных решений для передачи СВЧ сигналов «**FORSTAR**».

AVIC JONHON

«JONHON» (основан в 1970 г.)

Разъемы специального, военного и аэрокосмического назначения:

(Применяются в военной, авиационной, аэрокосмической, морской, железнодорожной, горно- и нефтедобывающей отраслях промышленности)

«FORSTAR» (основан в 1998 г.)

ВЧ соединители, коаксиальные кабели, кабельные сборки и микроволновые компоненты:

(Применяются в телекоммуникациях гражданского и специального назначения, в средствах связи, РЛС, а так же военной, авиационной и аэрокосмической отраслях промышленности).

FORSTAR®

Телефон: 8 (812) 309-75-97 (многоканальный)
Факс: 8 (812) 320-03-32
Электронная почта: ocean@oceanchips.ru
Web: http://oceanchips.ru/
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, д. 2, корп. 4, лит. А