FocalPoint FM4000

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

Datasheet

Dec 6, 2010 (Revision 2.3)

Order Number: FM4000-DS-2.3

Table of Contents

Table of Tables

Table of Figures

Revision History

Revision	Date	Content
0.40	02/19/2007	• First release.
0.50	04/23/2007	• General update
0.55	09/30/2007	• General update (intermediate release)
0.90	10/03/2007	• General update
1.0	09/30/2007	• Update to Preliminary Datasheet
1.1	07/01/2008	• General update
1.15	07/11/2008	• Formatting update only
1.2	07/17/2008	• General update
1.31	10/7/2008	• General update (see revision document)
1.4	12/4/2008	• General update (see revision document)
2.0	2/10/2009	• General update (see revision document)
2.1	5/14/2009	• General update (see revision document)
2.2	1/5/2010	• General update (see revision document)
2.3	12/6/2010	• General update (see revision document)

Chapter 1 - Introduction

1.1 Scope

FocalPoint™ is the family name for Fulcrum's 10G Ethernet switch chip products, which includes the Tahoe L2 switch chip platform (FM2000 series) and the Bali multi-layer switch chip platform (FM3000 and FM4000 series). This functional specification is the basis for the data sheet for the FM4000 Bali devices, and provides information on the significantly-enhanced feature set over Tahoe in the areas of routing, access control lists, congestion management, network scaling, and management. The FM3000 devices are covered in a separate datasheet and for the rest of this datasheet, "Bali" should be understood to signify the FM4000 devices.

1.2 Product Applicability

Bali represents a family of products with various port configurations and package sizes which are listed in the following table:

Part Number		10G Ports	2.5G/1G Ports	Package Type
FM3410	FM4410	8	10	529-ball
FM3103	FM4103	2	4	897-ball
FM3104	FM4104	2	8	897-ball
FM3112	FM4112	8	16	897-ball
FM3208	FM4208	8	0	897-ball
FM3212	FM4212	12	0	1433-ball
FM3220	FM4220	20	0	1433-ball
FM3224	FM4224	24	0	1433-ball

This document pertains to all variants of the Bali platform, although most references are specific to the 24-port 10G version of the device. The part marking and number conventions for FocalPoint devices are defined as follows:

Figure 1: FocalPoint Part Marking

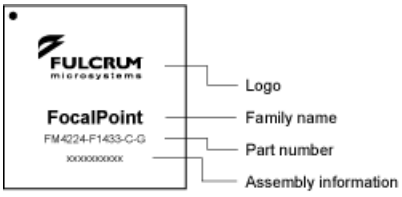
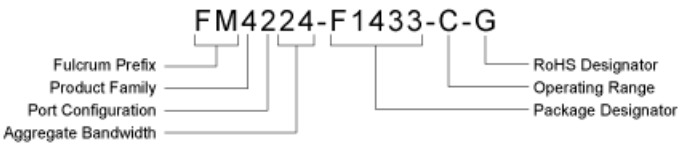


Figure 2: FocalPoint Part Number Convention



Key:

- Fulcrum prefix: "FM" identifies the device as a Fulcrum product.
- Product family: Conveys information about the general capabilities of the device, as follows:
  - 2: Tahoe (L2)
  - 3: Bali with L2+ features
  - 4: Bali 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 (1G, 2.5G operation)
  - 2: Mostly quad-SerDes interfaces (10G operation)
- Aggregate bandwidth: Identifies the approximate maximum bandwidth of the device configuration, calculated as:
  ((n x 10G interfaces) + (m x 2.5G interfaces))/10. Example: ((4 x 10G) + (12 x 2.5G))/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

The document is organized in the following elements:

- Chapter 1 – Introduction
- Chapter 2 – Overview
- Chapter 3 – Pin Description
- Chapter 4 – Ethernet Port Logic (EPL)
- Chapter 5 – Management
- Chapter 6 – Frame Filtering and Forwarding (FFU)
- Chapter 7 – Routing

- Chapter 8 – Layer 2 Lookup Tables
- Chapter 9 – Port Mapping
- Chapter 10 – Frame Hashing
- Chapter 11 – Triggers
- Chapter 12 – Congestion Management
- Chapter 13 – Scheduling
- Chapter 14 – Statistics
- Chapter 15 – All Registers
- Chapter 16 – Electrical Specifications
- Chapter 17 – Mechanical Specifications

1.4 Definitions

Name	Definition
FocalPoint™	The family name for Fulcrum's 10G Ethernet switch chip products, which includes the Tahoe L2 switch chip platform and the Bali multi-layer switch chip platforms.
Tahoe	The original member of the FocalPoint family, Tahoe is a layer-2 10G Ethernet switch chip platform which forms the basis for many FocalPoint L2 switch product variants (including the FM2224, FM2212, FM2208, FM2112, FM2104, and FM2103).
Bali	The newest member of the FocalPoint family, Bali is an enhanced multi-layer 10G Ethernet switch chip platform which forms the basis for new FocalPoint switch product variants, all of which are pin-compatible with their original Tahoe counterparts. The Bali FM4xxx devices are full featured L3 routing devices plus other enhancements such as ACL's, congestion management, increased frame memory and more. The Bali FM3xxx devices are similar to the FM4xxx, except that L3 routing is not included.
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 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 interchangably; for now they shall be interpreted as synonymous. In the future, this document will be edited to standardize on the term packet.
Byte	8 bits
Half-word	16 bits
Word	32 bits
Double Word	64 bits

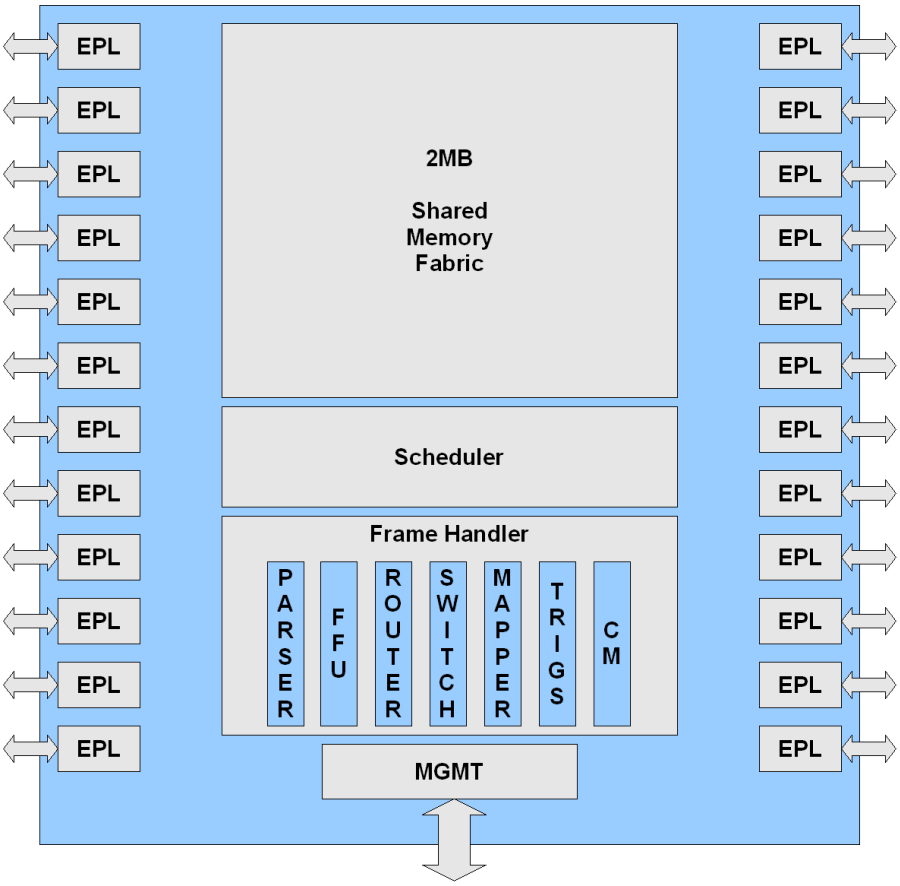
Name	Definition
Segment	A portion of a packet corresponding to one of the common architecturally-significant storage and processing "chunks" that has been defined in the architecture. In FocalPoint FM3xxx and FM4xxx devices, a segment is 512 bytes.
Sub-segment	A fragment of a segment that corresponds to the smallest architecturally-significant "atomic unit". In FM3xxx and FM4xxx devices, a sub-segment is 64 bytes.
GloRT	Fulcrum-proprietary **Glo**bal **R**esource **T**ag, which is used to pass global identification information from one Bali 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.)
Jumbo Frame	The maximum jumbo frame size for FM3xxx and FM4xxx devices is 16,376 bytes.
ISL	Fulcrum-proprietary **I**nter- **S**witch **L**ink tag, which is used to pass relevant management and control information from one Bali device to another in a network.
EBI	External Bus Interface. Fulcrum's term for the external CPU interface. EBI and CPU Interface are used interchangeably.
Bit numbering	Bit 0 is the least-significant bit throughout the Bali architecture (even if Ethernet standards and specifications suggest otherwise).
Register type	Registers are split into fields which can be of the following types: • RO: Read-only • RW: Read-write (reading returns the value written) • CW1: Clear on write 1 • CW0: Clear on write 0 • CR: Self clear have being read. • RV: Reserved for future usage (must be written as 0, will return 0 on read)
Trapping	Trapping refers to special frames that are captured by the switch and redirected to a local CPU for processing.
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.
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.
{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.

Chapter 2 - Architecture Overview

2.1 Introduction to Architecture

The main components in the FocalPoint architecture are shown in Figure 3.

Figure 3: FocalPoint Block Diagram



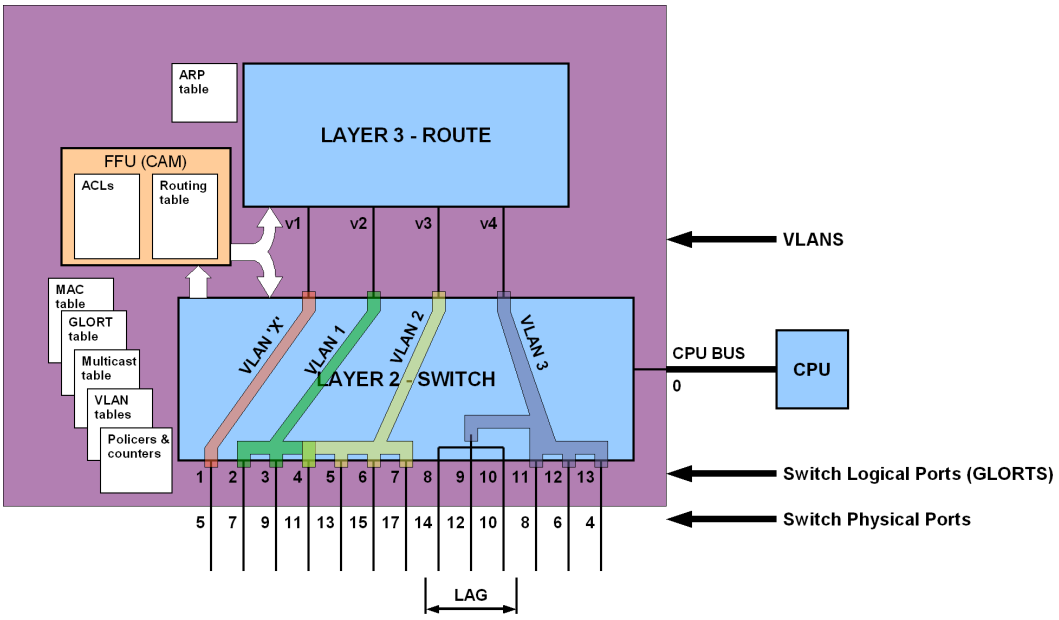
- **EPL**: The Ethernet Port Logic interfaces to the XAUI physical interface block, and implements the PCS and MAC layers for transmission and reception. The EPL parses incoming packets to extract the packet headesr which are sent to the frame handler for packet processing while saving the entire packet 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 will then apply advanced scheduling algorithms to decide which packet to forward to the EPL and will then send a segment list to the EPL which will use 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.

FocalPoint operates as a one-arm IP router combined with an Ethernet layer 2 switch. This is shown in Figure 4. 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 4: FocalPoint 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 2 figures. The Figure 5 shows a stack arrangement, Figure 6 shows a fat-tree (Clos) architecture. Both examples allow exploitation of the full feature set through a F64 tag use on internal links.

Figure 5: FocalPoint in a Stack Topology

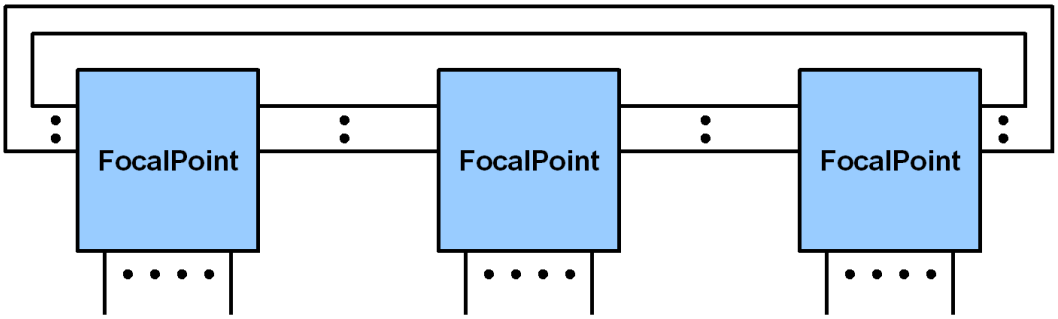
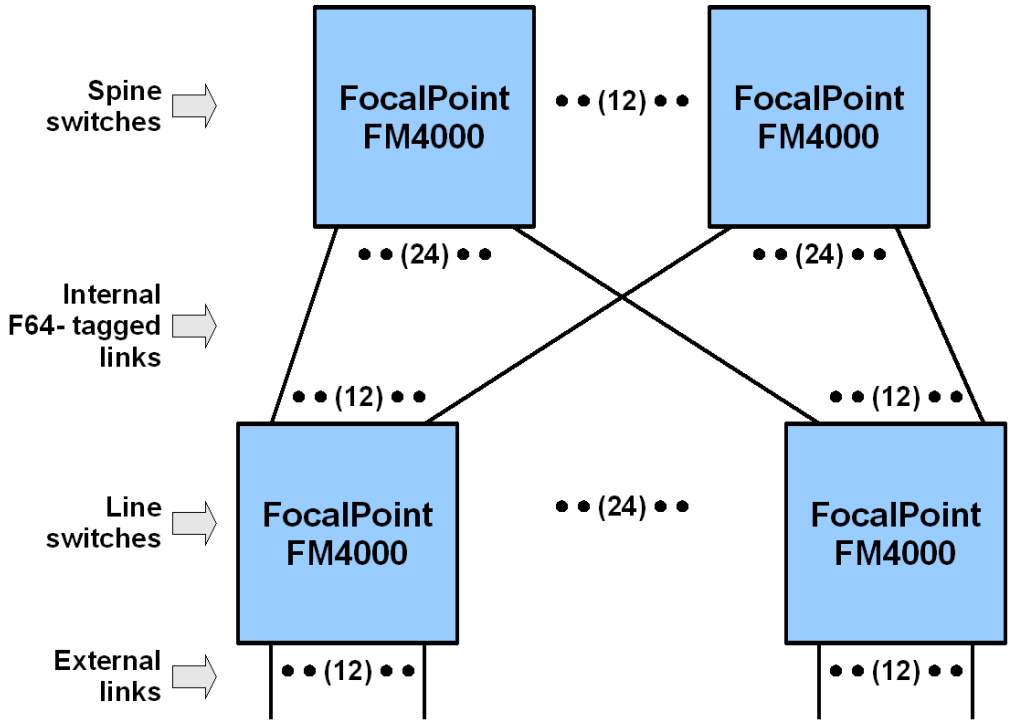
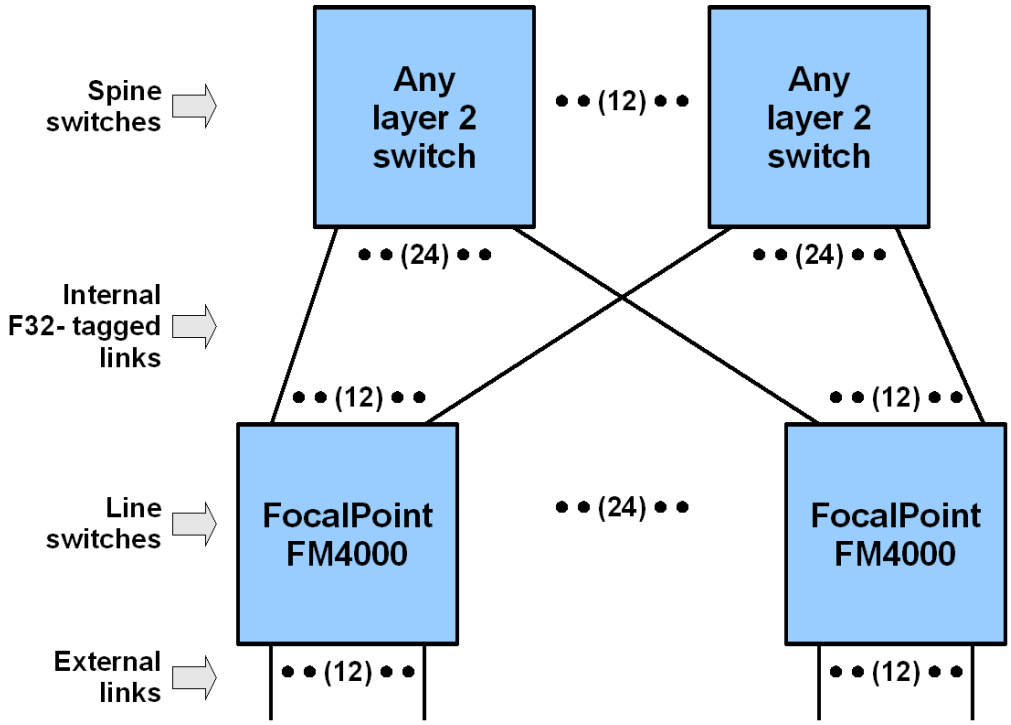


Figure 6: FocalPoint in a Tightly Coupled Clos Topology



The Figure 7 shows a fat-tree using non-Fulcum 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.

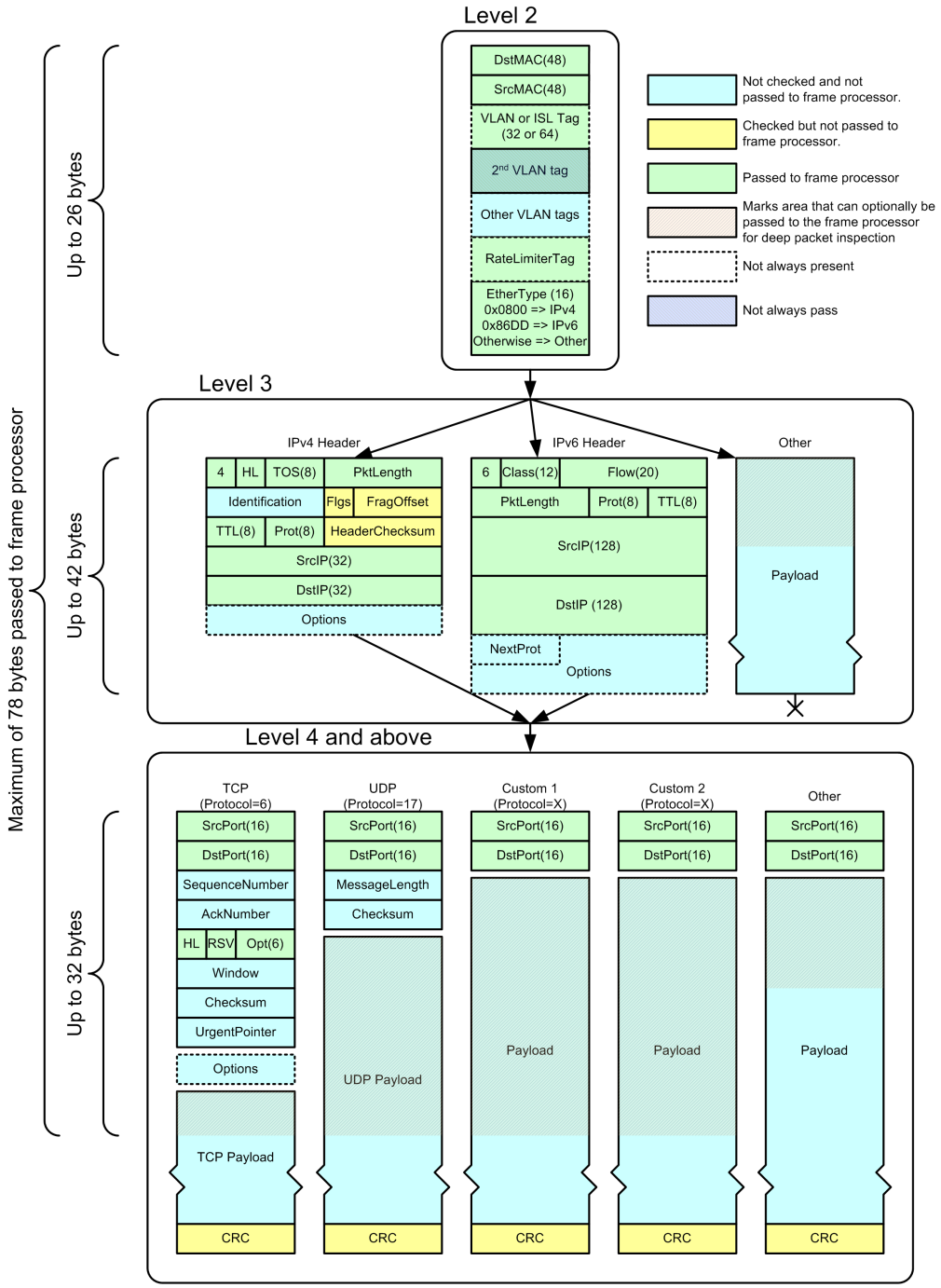
Figure 7: FocalPoint in a Loosely Coupled Clos Archicture



2.2 Frame Parsing

The frame parsing is shown in Figure 8. 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 8: Frame Parsing

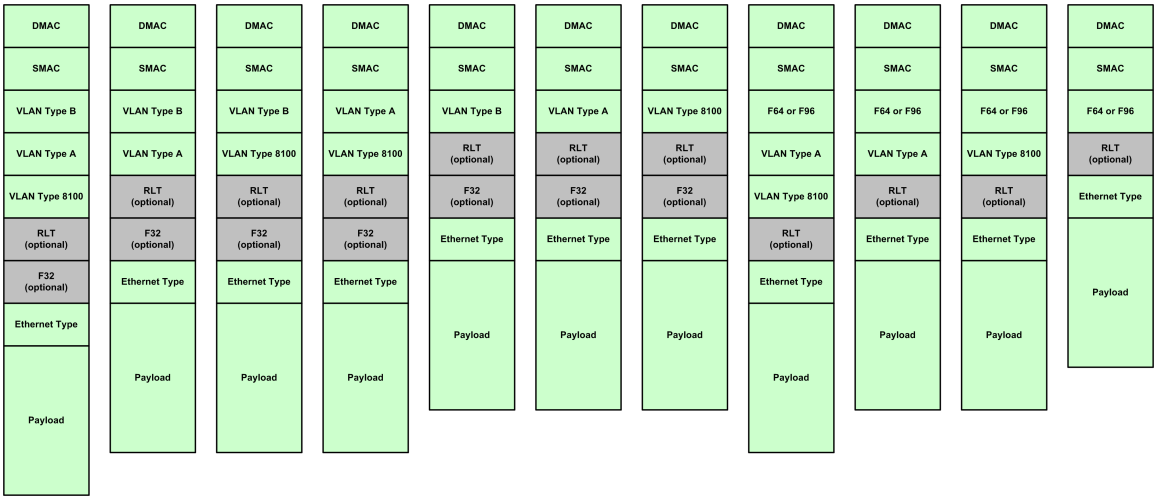


2.2.1 Layer 2

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, then 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 proceed as for "No ISL" tag case but will also check 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, then the source GloRT is captured and sent to the frame handler. If the F32 is not present, then the frame handler will associate 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 9: VLAN Stacking Options



- o Any other VLAN stacking configuration will be 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 will be switched but cannot be routed.
- Rate Limiter Tag
  - o 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.
  - o The rate limiter tag is not detected if the VLAN stack was considered invalid.
- Ethernet Type
  - o Always forwarded to the frame processor.

2.2.2 Layer 3

FocalPoint 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), then the frame is parsed as IPv4 frame. If the ethertype is 0x86dd (and the IP version is 6), then 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. FocalPoint 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
  - o 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.
  - o 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 will skip 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, then the layer 4 will be 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...), then the packet is flagged as having a parse error and the frame will be 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 doesn't 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, then the layer 4 will be processed. Otherwise, the parsing stops. If an IPv6 packet is fragmented, it will carry a Fragment Header option. If it has a Fragment Header, and the fragment offset (bits 16-28 of the Fragment Header option) is nonzero, then this is not the first fragment.
    - If there is an error during decoding (IP header checksum invalid, invalid HL, etc...), then the packet is flagged as having a parse error and the frame will be 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 will be 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, then parsing terminates at this point because the layer 4 header will not be 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 will carry a Fragment Header option. If it has a Fragment Header, and the fragment offset (bits 16-28 of the Fragment Header option) is nonzero, then 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 will be treated as '0' in the frame processing pipeline.

2.2.3 Deep Packet Inspection for IP frames

FocalPoint switches have the ability to forward additional frame header bytes to the Frame Handler. This capability is refered 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 capabitlity is detailed more in the Ethernet Port Logic (EPL) chapter.

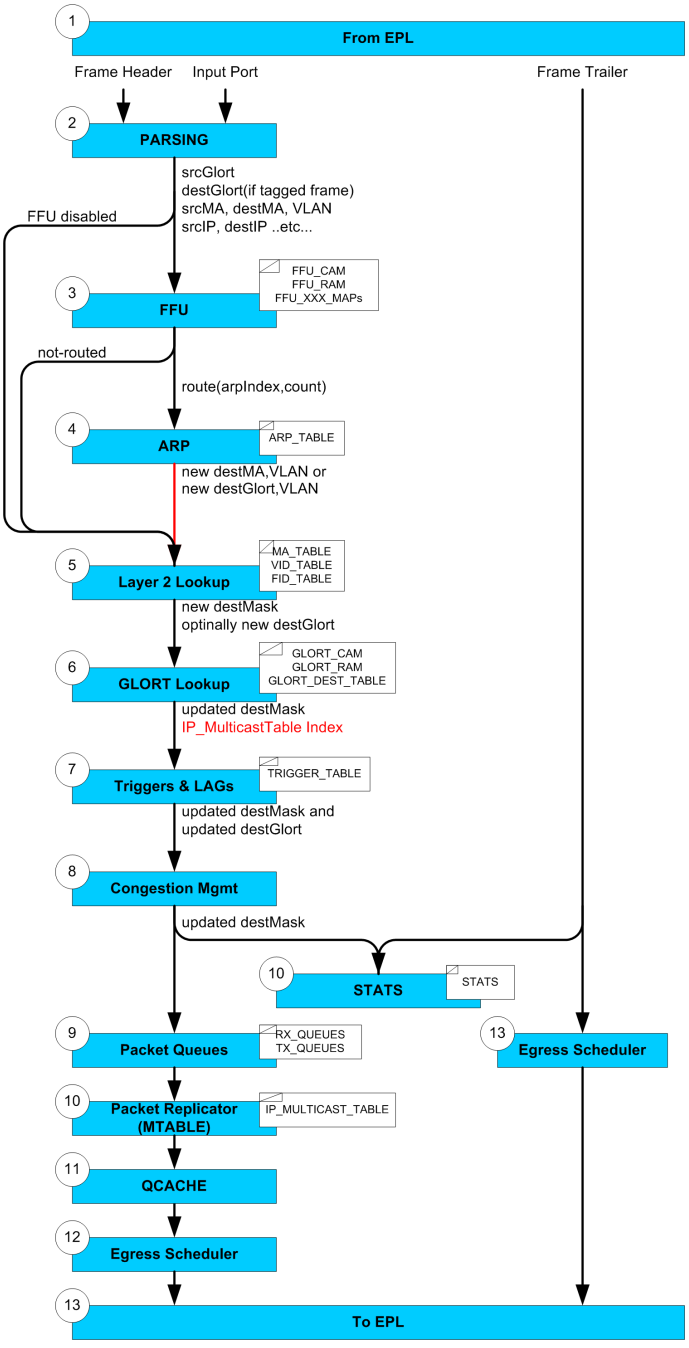
2.3 Frame Processing Pipeline Overview

FocalPoint 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:

- o Global routing across multi-chip in fat tree, ring or meshed topologies
- Layer 2 switching with optional automatic address learning and security
- o 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 10.

Figure 10: Frame Processor Pipeline



The different elements of this pipeline are briefly described below:

2.3.1 Frame Header

The f rame 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 unapplicable fields are automatically set to 0.*

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

- The chip will always store and forward 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 will not be 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, then 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, then 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 will hold 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, then the hold will be released and the frame will start to transmit immediately.

2.3.2 Frame Trailer

The f rame 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:
    - will not cause MAC address learning or security violation events,
    - will not cause trigger interrupts,
    - will cancel PAUSE actions,
    - will not be sampled for congestion notification purpose,
    - will cancel procession of congestion notification frames,
    - will not be forwarded if frame transmission has not started yet (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 that 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 it will 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 FocalPoint 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 will be stored in the MAC address table.
- **CPU GloRT**. Whenever a frame is trapped or logged or mirrored to the CPU, the top eight 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 Chapter 16.

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**. FocalPoint 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 the Ethernet Port Logic chapter.

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, or DiffServ. The priority association is detailed in the congestion management chapter.

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 32x512x36-bit flexible ternary CAM structure along with a set of control registers. The FFU also contains a mapper which will map 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 the FFU section in this chapter.

The default action is “switch” and “permit” in case no matches are found in the FFU. The entire FFU can be disabled.
IP routing will normally be 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, then it is switched. If the frame is addressed to the router but is not IPv4 or IPv6, then 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, then 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 desinations on other VLANs. Although the frame is technically both switched and routed, FocalPoint 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, then FocalPoint 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 will be 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 16K 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 GloRT unit and IP Multicasting) 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 that disabling the FFU will cause 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 will first use the destination MAC address and the VLAN port to retrieve the logical destination port (GloRT) from the MA_TABLE and will then use the VLAN to retrieve a spanning-tree number and VLAN disposition and use the FID_TABLE to retrieve the spanning tree state. If the destination address is unknown, then 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 will be 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 Chapter 11, "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 will use the result of hashing over the frame to determine which port of a link aggregation group will be 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 mutliple 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 Fulcrum ISL tag which may actually be added, updated or removed depending on how the switch is configured. Tthe CRC is also updated if the Fulcrum tag was added, modified or removed. Packets replicated for TX mirroring will match 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 Fulcrum 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 shapping 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 then the actions taken are:
  - Change source and destination address and VLAN
  - Decrement TTL
  - Compute new CRC
- If packet is switched then the actions taken are:
  - Change VLAN if needed (add, delete, replace)
  - Compute new CRC if packet is changed

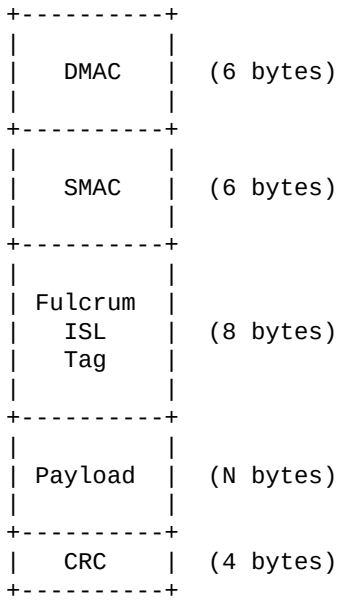
The egress scheduler is detailed in Chapter 13, "Egress Scheduling and Shaping".

The egress EPL is detailed in the EPL chapter.

2.4 Fulcrum ISL Tag

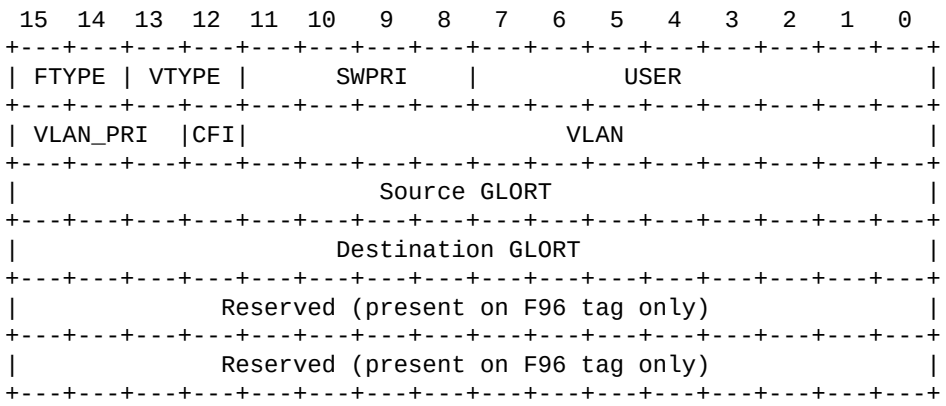
The Fulcrum ISL tag can be attached to frames on links between Fulcrum chips, and between a Fulcrum chip and a CPU. It carries extra frame-specific data that allows multiple chips to work together as a single system. Each port of the switch is configurable to expect the Fulcrum ISL tag on all frames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

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



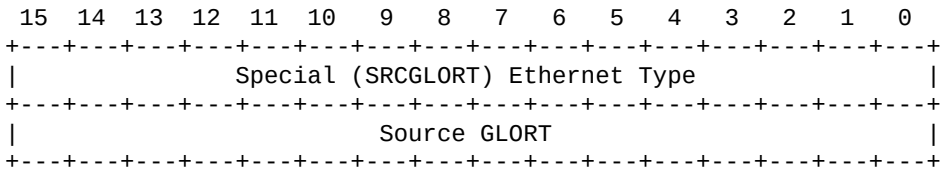
The tag exists in 3 different formats: F32, F64 and F96.

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 will always transmit 0s in the reserved field and ignore 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 0, then the frame handler will associate 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.

```
0x0: normal
0x1: routed
0x2: special delivery
0x3: management
```

Normal: this is 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, then 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: this is 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 will get 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, then 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 will be accepted by the MSB as in-band management. A port may be configured as untrusted, meaning that management frames coming from that port will be dropped.

	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 enabled
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:

- '00': the original frame didn't have any VLAN tag or the VLAN tag was ignored
- '01': the original frame had a VLAN tag of type 0x8100
- '10': the original frame has a VLAN tag of type USER DEFINED “A”
- '11': 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:

- '00': request
- '01': response or interrupt
- '10': reserved
- '11': 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 chapters 12 and 13 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 doesn't 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 1, then this port will trasmit the source GloRT Ethernet tag. If this bit is not set, then 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)

This is 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), then this field is reserved.

2.4.6 VLAN CFI/DEI bit (VCFI)

This is 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), then this field is reserved.

2.4.7 VLAN id (VID)

This is 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), then 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 0.

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's 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 FocalPoint chips have a local CPU, this won't be 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. FocalPoint may not preserve these bits on a mirror or CPU copy.

2.5 F96 tag

FocalPoint can be configured, per port, to use a 3 word (96-bit) ISL tag. In this case, it will be 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 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.

Note that after this step, the frame will then go through the following final 3 steps (in order):

- Triggers which may override any prior decision.
- LAG pruning and filtering.
- Congestion management.

Table 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	
3	Trap BPDU	Yes	L2 Privileged Frame Types
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	
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	

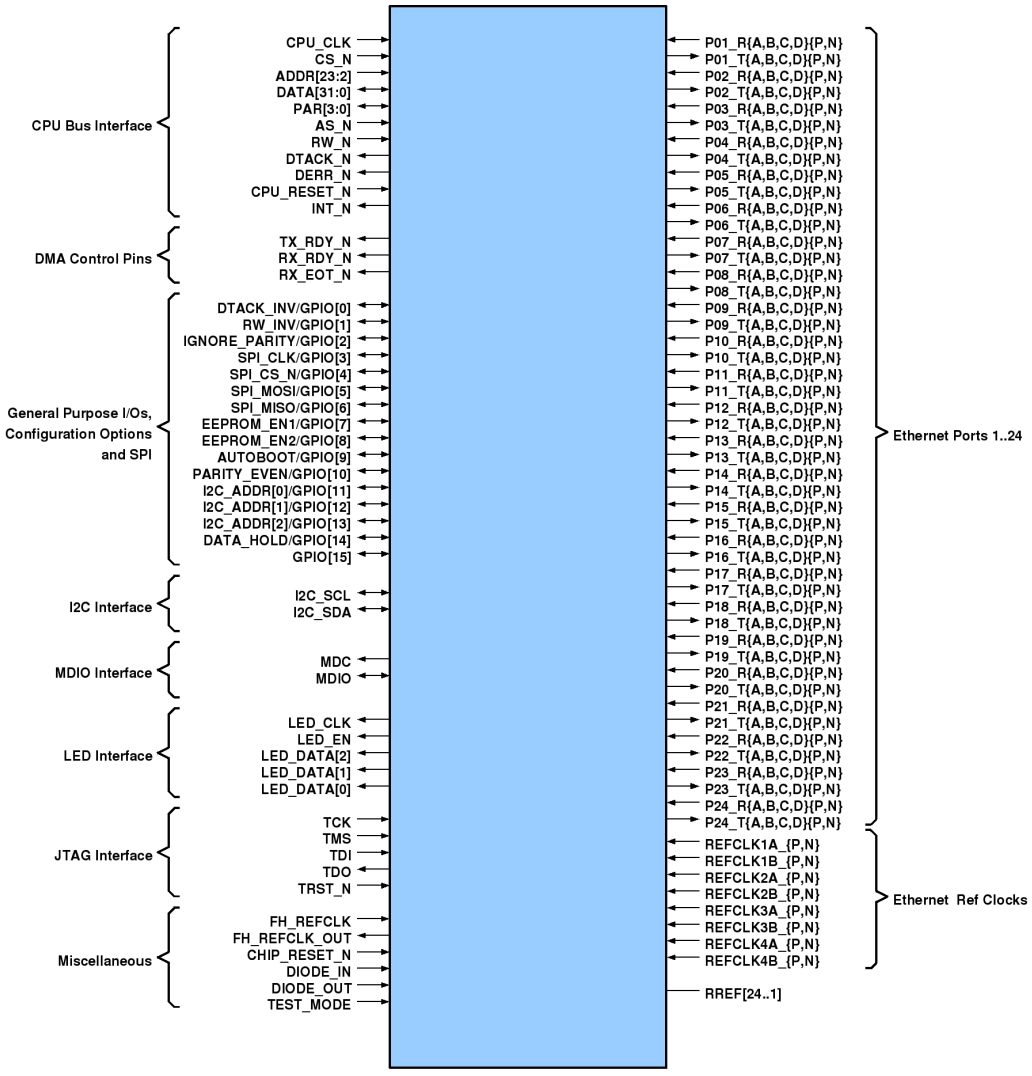
Bit Number and Relative Priority	Description	Learning	Category
26	Trap due to IGMP	Yes	
27	Trap due to TTL <= 1 for non-ICMP frames	Yes	
28	Log IP multicast frames to CPU because TTL <= 1		
29	Drop due to TTL <= 1	Yes	
30	Drop due to VLAN egress violation	No	L2 Egress Policies
31	Drop due to flood control of DLF frames (SYS_CFG_1)	No	
32	Drop due to GLORT_CAM miss.	No	
33	Unused bit		
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

Chapter 3 - Pin Description

3.1 Pin Overview

Figure 11 provides an overview of the pins used by both FM4000 and FM3000 series switches

Figure 11: 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 (0=asserted, 1=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 Description

FocalPoint pins are described in Table 1. Note that GPIOs pins are latched when CHIP_RESET_N is de-asserted to provide default configurations.

Table 2: Pin Description

Pin Name	Type	Type	Usage
Ethernet Port Pins			
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.

Pin Name	Type	Type	Usage
RREF[24..1]	Sense	N/A	Reference resistor to VDDX
Power Pins			
VDD	Power	N/A	Core power (1.2V-1.3V)
VDDX	Power	N/A	Serdes core power (1.0V-1.2V)
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.
Bus Interface Pins			
CPU_CLK	In	LVTTL	Bus interface clock
ADDR[23:2]	In	LVTTL	Address inputs.
DATA[31:0]	In/Out	LVTTL, 6mA	Data bus
PAR[3:0]	In/Out	LVTTL, 6mA	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, 6mA	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, then DTACK_N is active low. If DTACK_INV is pulled up to VDD33 at reset, then DTACK_N is active high.
DERR_N	Out	LVTTL, 6mA	Indicates write data parity errors. Only asserted (and valid) when DTACK_N asserted. Tri-stated otherwise.
TEST_RESET_N	In	LVTTL	Fulcrum internal use only. Users must tie this pin high.
INTR_N	Open-drain	LVTTL, 6mA	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.

Pin Name	Type	Type	Usage
DMA Interface Pins			
TXRDY_N	Out	LVTTL, 4mA	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, 4mA	Receive data ready. Asserted whenever the switch has data in its receive FIFO for the CPU.
RXEOT_N	Out	LVTTL, 4mA	End of frame indication. Asserted while reading the last data word of a packet to indicate this is the last work of the packet.
GPIOs and Strapping Pins			
DTACK_INV/GPIO[0]	In/Out	LVTTL, 6mA	Defines polarity of DTACK_N. If connected to ground, then DTACK_N is active low. If connected to VDD33, then DTACK_N is active high. On FM2000, this pin is permanently used for this function. On FM3000/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, 6mA	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 FM3000/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, 6mA	Setting this pin to VDD33 will disable parity checking on incoming write data. On FM2000, this pin is permanently used for this function. On FFM3000/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 In/Out In/Out In/Out	LVTTL, 2mA	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.

<table>
<tr><th>Pin Name</th><th>Type</th><th>Type</th><th colspan="3">Usage</th></tr>
<tr><td>EEPROM_EN1/GPIO[7]

EEPROM_EN2/GPIO[8]</td><td>In/Out</td><td>LVTTL, 6mA</td><td colspan="3">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.</td></tr>
<tr><td></td><td></td><td></td><td>EN1</td><td>EN2</td><td>Usage</td></tr>
<tr><td></td><td></td><td></td><td>0</td><td>0</td><td>Disable. A local CPU is responsible to configure the switch.</td></tr>
<tr><td></td><td></td><td></td><td>1</td><td>0</td><td>SPI</td></tr>
<tr><td></td><td></td><td></td><td>1</td><td>1</td><td>I2C at address 0x50</td></tr>
<tr><td></td><td></td><td></td><td>0</td><td>1</td><td>I2C at address 0x51</td></tr>
<tr><td></td><td></td><td></td><td colspan="3">The EEPROM is always assumed to require 3 bytes of addressing.</td></tr>
<tr><td>AUTOBOOT/GPIO[9]</td><td>In/Out</td><td>LVTTL, 6mA</td><td colspan="3">General purpose I/O pin.

This pin is also latched at reset to defines if the chip will automatically start booting or wait for the CPU intervention.</td></tr>
<tr><td>PARITY_EVEN/GPIO[10]</td><td>In/Out</td><td>LVTTL, 6mA</td><td colspan="3">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.</td></tr>
<tr><td>I2C_ADDR[0]/GPIO[11]</td><td>In/out</td><td>LVTTL, 6mA</td><td colspan="3">General purpose I/O pin.

This pin is also latched at reset to set the I2C slave address of the switch.</td></tr>
<tr><td>I2C_ADDR[1]/GPIO[12]</td><td>In/out</td><td>LVTTL, 6mA</td><td colspan="3">General purpose I/O pin.

This pin is also latched at reset to set the I2C slave address of the switch.</td></tr>
<tr><td>I2C_ADDR[2]/GPIO[13]</td><td>In/out</td><td>LVTTL, 6mA</td><td colspan="3">General purpose I/O pin.

This pin is also latched at reset to set the I2C slave address of the switch.</td></tr>
<tr><td>DATA_HOLD/GPIO[14]</td><td>In/out</td><td>LVTTL, 6mA</td><td colspan="3">General purpose I/O pin.

This pin is latched at reset to define DTACK and DATA behavior. If this pin is pulled up, then DTACK and DATA (if read) are asserted when needed and remain asserted until CS is de-asserted. If this pin is pulled down, then 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.</td></tr>
<tr><td>GPIO[15]</td><td>In/out</td><td>LVTTL, 6mA</td><td colspan="3">General purpose I/O pin.</td></tr>
</table>

Pin Name	Type	Type	Usage
I2C Pins			
I2C_SCL	Open-drain	LVTTL, 6mA	I2C Clock. Pull-down current = 30mA
I2C_SDA	Open-drain	LVTTL, 6mA	I2C data. Pull-down current = 30mA
MDIO Pins			
MDC	Out	LVTTL, 6mA	MDIO clock
MDIO	Open-drain	LVTTL, 6mA	MDIO data. Pull-down current = 30mA
LED Pins			
LED_CLK	Out	LVTTL, 2mA	Serial LED clock.
LED_EN	Out	LVTTL, 2mA	Serial LED enable. Asserted only on the first bit of the 36-bit frame.
LED_DATA0	Out	LVTTL, 2mA	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.
JTAG Pins			
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, 4mA	JTAG data output
TRST_N	In	LVTTL	JTAG reset. Internal pull-down. Tie low if not using JTAG
Miscellaneous Pins			
CHIP_RESET_N	In	LVTTL	On FocalPoint, all strapping options are latched while the chip is in reset. Internal pull-down.
DIODE_IN DIODE_OUT	Sense	Analog	Temperature sensing diode.
FH_PLL_REFCLK	In	LVTTL	Reference clock to PLL. Must be between 10MHZ and 70MHZ.
FH_CLKOUT	Out	LVTTL, 4mA	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 Fulcrum for scan testing and must be connected to ground for normal operation.
CONT_EN	In	LVTTL	Reserved for Fulcrum use. Connect to ground for normal operation.

Chapter 4 - Ethernet Port Logic

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 FocalPoint device variants, some number of ports operate in 10M/100M/1G/2.5Gbps SGMII, single-SerDes modes. Other ports designed for10Gbps operation, use the quad-SerDes mode, each providing 2.5Gbps data throughput with a line rate of 3.125Gbps after line encoding. Any 10Gbps port can also be operated in single-SerDes mode at one of the lower data rates. Note that for rates lower than 1Gbps, the SerDes still operates at 1G. 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 1Gbps 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 3 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 register EPL_PORT_CTRL defines the clock selected for each port (A or B). The actual SERDES speed will be 10x the reference clock selected. The reference clock must be between 100MHZ and 350MHZ. A reference of 125MHZ is suitable for 1Gbps data rates (SerDes speed = 1.25G) while a reference of 312.5MHZ is suitable for 10Gbps data rates (SerDes speed = 3.125G). 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 1K Ω ±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. See the SERDES_CTRL_1::DTX field in the SERDES registers. Finally, 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, 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 table below:

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

Note that in single serdes mode only lane A may be used.

4.2.4 Testing Facilities

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

- 0 – Disable, normal operation
- 1 – PRBS (x9+x5+x1), repeat every 511 cycles
- 2 – High frequency test data = 1010101010
- 3 – Test data = K28.5 (IDLE)
- 4 - Low frequency test data = 0001111100
- 5 – PRBS (x10+x3+x1), repeat every 1023 cycles
- 6 – PRBS (x9+x4+x1), repeat every 511 cycles
- 7 - 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 0. The BIST checkers are activated by writing a 0 into SERDES_TEST_MODE[x]::BS. The values in BIST_ERR_CNT count the number of errors received per lane.

The BIST checker will work 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:

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

The loopback function will loop 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 will check for signal detection (at least 75mV) 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).

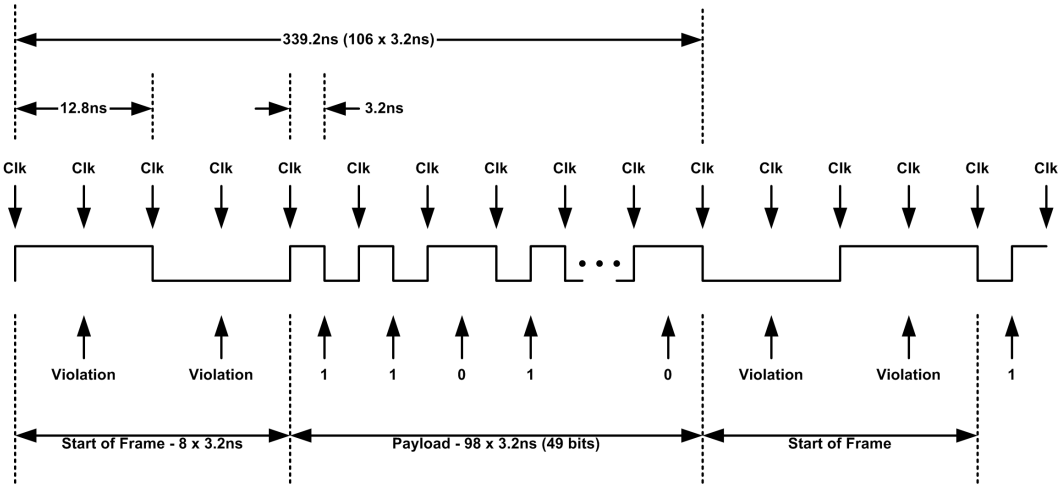
4.2.6 Auto-negotiation

FocalPoint 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 interrups

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 the next figure. 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 125MHZ reference clock.



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 0. 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 1) 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 will enter 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 0 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 the following:

	Control Word		
Bits	**Name**	**Transmitter**	**Receiver**
0-4	Selector	Fixed (00001). Cannot be set by software.	Checked by hardware.
5-9	Echoed Nonce	Transmitter will echoed received “transmitted Nonce” in this field once 3 valid identical CW have been received.	Checked by hardware to be equal to the transmitted “transmitted Nonce” (if ACK is 1).
10-12	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 3 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 3 valid identical CW have been received.	Check by hardware.
16-20	Transmitted Nonce	ID for this negotiation session. Set by software.	Link partner ID for this negotiation session.
21-45	Mode Abilities	Abilities of this transmitter. Set by software.	Abilities of link partner. Read and interpreted by software.
46-47	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 shown here:

	Control Word		
Bits	**Name**	**Transmitter**	**Receiver**
0-10	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 3 valid identical CW have been received.	Check by hardware.
15	NP	Software presets if needed and hardware follows once 3 valid identical CW have been received.	Check by hardware.
16-47	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 0 so that a software intervention is required to complete the exchange. If there are no next pages sent or received, then the software will not have to intervene and an AN_DONE interrupt will be received upon completion of the exchange.

The details of the algorithm are given below.

1. Software initializes the MAC.

    - The software initializes the serdes and sets up the BIST mode to force a low-frequency pattern on the line.
    - The software initializes the hardware abilities by loading them into the AN_TX_MSG0 register (fields 'selector' must be set to 1 and 'echoed nonce' must be preset to 0, hardware will update the echoed nonce at the appropriate time) and sets the ACK bit to 1. The NP bit should be set either to 1 if there is one extra next page to send and 0 if there are no next page to send.

    - The software loads a next page into AN_TX_MSG1 if there are any and sets the ACK bit to 0 and the NP bit to 1 or 0 depending of there are further pages to send after that one. The software initializes the 'select' bit in AN_CTL to 0.

    - The software then enables the AN_RX_MSG, AN_TIMEOUT, AN_FAIL and AN_DONE interrupt bit in PCS_IM and waits.
    - The software then enables the auto-negotiation by setting the enable in the AN_CTL register.

    - The software disables the BIST mode.

2. FocalPoint performs control word exchange.

    - The MAC starts by transmitting the AN_TX_MSG0 as the control word repetitively with ACK bit to 0.
    - The MAC waits for 3 identical valid control words and stores them into AN_RX_MSG0.
    - The MAC echoes the NONCE received into its message, sets the ACK bit to 1. If the MAC detects the NONCE received is equal to the NONCE transmitted, then the MAC posts an AN_FAIL interrupt.
    - The MAC waits for the ACK bit to be received to 1; the updated word is stored in AN_RX_MSG0 (second interrupt posted). If NP is received as 0 and transmitted as 0, then the hardware will post an AN_DONE interrupt. If the NP is received as 1 or transmitted as 1, then an AN_RX_MSG interrupt is posted.
    - The MAC then starts to transmit the next page stored in AN_TX_MSG1 (if NP is 1 in AN_TX_MSG0) or will transmit null pages (if NP is 0 in AN_TX_MSG0). It will transmit with the ACK bit to 0 and will hold at this stage until software intervention because the AN_TX_MSG1:ACK bit is 0. In the mean time, the hardware will check for received messages and, if there are at least 3 identical copies, stored them into AN_RX_MSG1 and post an AN_RX_MSG1 interrupt.

3. Software waits for RX_MSG interrupts.

    - 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 (wher ACK is set to 0 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.

    - The MAC detects the toggle change in select. It then first completes the current exchange by sending ACK to 1 and a new ACK2 bit.
    - The MAC waits for ACK 1 from remote, posts a new value in AN_RX_MSG1 and posts an AN_RX_MSG1 interrupt.
    - The MAC starts to transmit next page sending an ACK bit.
    - The MAC waits for the ACK bit to be received to 1, the updated word is stored in AN_RX_MSG0. If NP is received as 0 and transmitted as 0, then the hardware will post an AN_DONE interrupt.
    - The MAC then starts to transmit the next page stored in AN_TX_MSG1 (if NP is 1 in AN_TX_MSG0) or will transmit null pages (if NP is 0 in AN_TX_MSG0). It will transmit with the ACK bit set to 0 and hold. The control is returned to step 3.

5. Analyze messages received and apply negotiated values.

    - The software will detect the interrupt and react 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, then the software must wait for a while and go back to step 1.
    - The software will then apply the negotiated values and set the serdes accordingly. If the link doesn't come up within a certain time, then the software must go back to step 1.

Note that the 802.3ap doesn't define the 2.5GBPS 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 is shown here:

	Control Word		
Bits	**Name**	**Transmitter**	**Receiver**
0-4	Selector	Set by software.	Checked by software.
5-6	Duplex	Set by software.	Duplex abilities. Read and interpreted by software once negotiation exchange is completed.
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 3 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 3 valid identical CW have been received.	Check by hardware.

The next page format is shown here:

	Control Word		
Bits	**Name**	**Transmitter**	**Receiver**
0-10	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.

	Control Word		
14	ACK	Software presets if needed and hardware follows once 3 valid identical CW have been received.	Check by hardware.
15	NP	Software presets if needed and hardware follows once 3 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 3 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 will still report 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 FocalPoint initiates the negotiation while the second mode is suitable when FocalPoint is silently listening to a negotiation request but is not initiating one. In that second mode, the software will wait for an interrupt announcing reception of configuration word and then will need to change the mode to enabled the transmitter to actually perform the negotiation.

4.3 PCS

4.3.1 PCS – Frame Format

The frame format in 10G 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 1G 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 0) or a variable size preamble (bit SP is set to 1). When configured for supporting a variable size preamble, the PCS will accept 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 will search for |S| and then sample incoming data every 10 cycle. In the 10M, the PCS will sample incoming data every 100 cycle.

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 will accept 0xD5 or 0x5?-0x?D as a valid start of frame and will automatically realign 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.

4.4 PCS – 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 will be received at any time. In the unlikely situation where two faults are received, then the local faults take precedence.

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

Note that 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.

4.5 PCS – Messaging

The PCS will support 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 will set 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 will self clear once the symbol is sent. The PCS_IP[FS] will be 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.

4.6 PCS – Balancing IFGs

On transmit, it may be necessary for the transmitter to modify the length of the interframe gap (IFG) in order 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 will reduce 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 0. The second method is is supported by setting the PCS_CFG_1[DE] to 1.

4.7 MAC

The MAC layer is responsible for the following actions:

- In the receive direction
  - o 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.
  - o Parses the incoming frame and send pertinent information from the header to the frame control.
  - o 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 will also be marked as incorrect in this case.
- In the transmit direction:
  - o 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.
  - o 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 will give the frame a bad CRC if if the receiver has marked it for discard, or if it had a bad CRC when read from the memory, or if the transmitter underflows. Otherwise, it gets a good CRC.

- The transmitter will also pad the frame to MinFrameSize if the PadRuntFrames is set. The padding is done with random bytes and the CRC will be updated accordingly.

4.7.1 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. The Architecture Overview chapter, in the Frame Parsing section, gives a detailed view of frame parsing for layer 2 frames, layer 3 frames and for deep packet inspection.

4.7.2 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 4B 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 will 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 will be 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 will confuse the IP checksum calculation, and will very likely lead 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.

4.7.3 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, then the deep inspection will 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 will always skip 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 will always stop 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 will always stop 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 will always stop 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
  - o The first two 16-bit half words of the packet are forwarded to the frame processor and mapped to source and destination port.
  - o The parser will simply forward 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
0	L4A
1	L4B
2	L4C
3	L4D
4	SIP[127:112]
5	SIP[111:96]
6	SIP[95:80]
7	SIP[79:64]
8	SIP[63:48]
9	SIP[47:32]
10	DIP[127:112]
11	DIP[111:96]
12	DIP[96:80]
13	DIP[79:64]
14	DIP[63:48]
15	DIP[47:32]

4.7.4 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
0	SIP[127:112]
1	SIP[111:96]
2	SIP[95:80]
3	SIP[79:64]
4	SIP[63:48]
5	SIP[47:32]
6	SIP[31:16]
7	SIP[15:0]
8	DIP[127:112]

Half-word	Mapped to FFU
9	DIP[111:96]
10	DIP[95:80]
11	DIP[79:64]
12	DIP[63:48]
13	DIP[47:32]
14	DIP[31:16]
15	DIP[15:0]

4.7.5 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:
  - Set EPL_PORT_CTRL[i]::IntializeN to 0
  - Set EPL_PORT_CTRL[i]::ClockSelect to desired clock (A or B)
  - Set EPL_PORT_CTRL[i]::ResetN to 0
  - Start serdes.
- Subsequent reset:
  - Shutdown reception by setting EPL_SERDES_CTRL_2::LaneReset, PLLReset and PowerDown to 1s.
  - Ensure EPL is not idling. The transmitter could be placed on hold by setting bandwidth on this port to 0.
  - Wait for at least one frame to drain.
  - Ensure all interrupts are disabled in the EPL (set all EPL IM registers to 1s)
  - Set EPL_PORT_CTRL[i]::ResetN to 1
  - Set EPL_PORT_CTRL[i]::IntializeN to 1
  - Set EPL_PORT_CTRL[i]::ClockSelect to desired clock (A or B)
  - Set EPL_PORT_CTRL[i]::ResetN to 0
  - Start serdes.

The serdes are started using the following procedure:

- If not already done, shutdown serdes by setting SERDES_CTRL_2::LaneReset, PLLReset and PowerDown to 1s
- Program desired equalizer and drive strength in SERDES_CTRL_1
- Program desired lane ordering and polarity in PCS_CFG_1
- Start Lane A and B (only required if either lane C or D is used)
  - Power up lane A and B
  - Wait 10us
  - Take PLL_AB out of reset

    - Wait 10us
    - Take lane A and B out of reset
    - Wait 10us
- Start Lane C and D (only required if either lane C or D is used)
    - Power up lane C and D
    - Wait 10us
    - Take PLL_CD out of reset
    - Wait 10us
    - Take lane C and D out of reset
    - Wait 10us

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

Chapter 5 - Chip Management

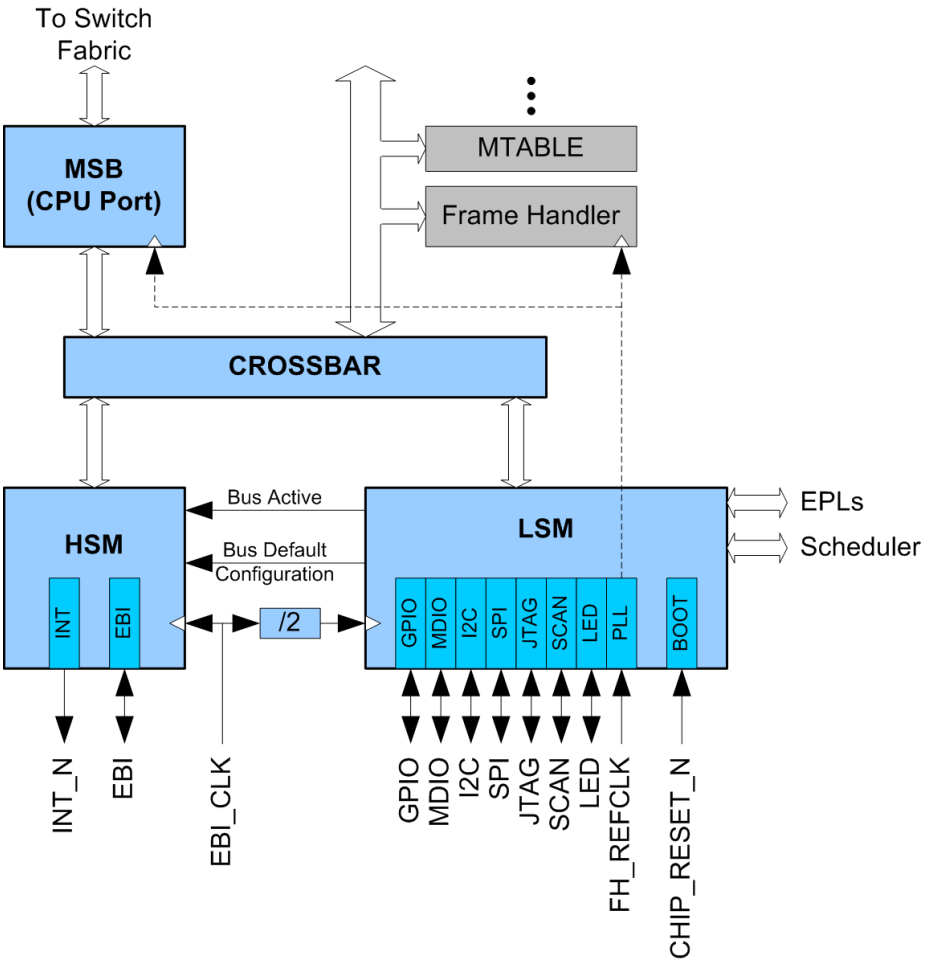
5.1 Overview

FocalPoint offers the following functionality:

- External demultiplexed address/data bus
- Up to 133MHZ
- Even/odd/no parity
- SPI
- I2C (master and slave)
- MDIO (clause 22 and 45)
- GPIO
- JTAG
- SCAN (not detailed in this chapter)
- PLL

The chip management block diagram is shown on Figure 12.

Figure 12: Switch Management Bloc Diagram



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

- MSB, "Management-Switch Bridge".
  - o This block interfaces to the switch element and is the CPU port to the fabric to transmit or receive frames from the network.
- HSM, "High Speed Management".
  - o This block contains the CPU interface.
- LSM, "Low Speed Management".
  - o 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. It includes the following functions:

- Interrupt Controller
- CPU Interface Controller

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

- BOOT Controller
- EPL access
- Scheduler access
- Crossbar access
- PLL
- GPIO Controller
- MDIO Controller
- I2C Controller
- JTAG Controller
- LED Controller
- SPI Controller (only used by Boot controller)

There are 5 possible requestors coming into the LSM, the priority in descending order are:

1. BOOT
2. XBAR_IF
3. EPL
4. I2C
5. JTAG

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

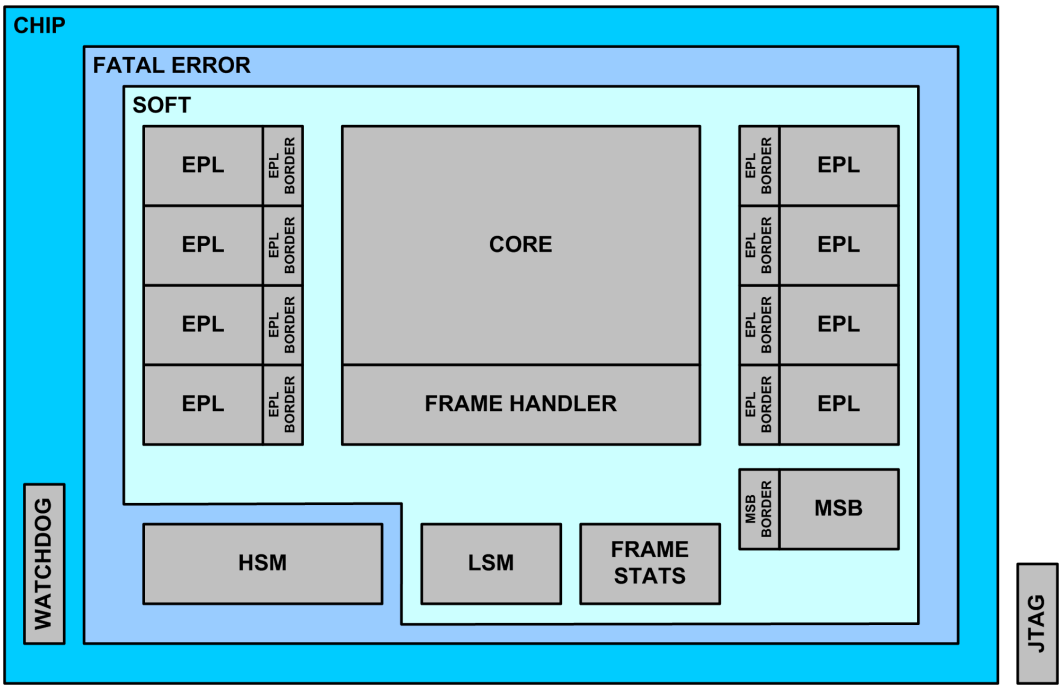
Responders are:

1. XBAR_IF
2. EPL
3. SCHEDULER
4. LED
5. SPI
6. I2C
7. MDIO
8. GPIO

5.2 Boot Controller and Chip Reset

There are several chip reset domains in the chip. They are shown in the next figure and detailed below:

Figure 13: Chip Reset Domains



1. **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 FocalPoint in order 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 10nS (min).
2. **Fatal Error Reset.** The fatal error reset domain includes the entire chip minus the watchdog.

    - FocalPoint 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 the HSM since this module handles all chip interrupts, which is the mechanism by which such fatal errors are indicated. The watchdog will assert the switch reset for 256 cycles.

3. **SOFT Reset**. The SOFT_RESET register controls 2 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.
4. **JTAG Reset**. The JTAG domain is on its own and only applies to the JTAG circuitry.
5. **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 will place the RXRDY_N, TXRDY_N and RXEOT_N in tri-state mode.
2. Deasserting CHIP_RESET_N or the self-deasserting 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 will do 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, then 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, then 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. Intialize 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 0 to SOFT_RESET

        4. Enable MSB and EPL0 by setting ELP_PORT_CTRL[0].ResetN to 1. Also set ELP_PORT_CTRL.InitializeN to 1.
        5. Enable FFU by setting SYS_CFG_8, bit 0 to a 1.
        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 1

        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 1

8. Program FFU_INIT_SLICE
9. Wait for BOOT_STATUS::CommandDone to return to 1Wait for BOOT_STATUS::CommandDone to return to 1Deassert 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 1

10. Initialize memory

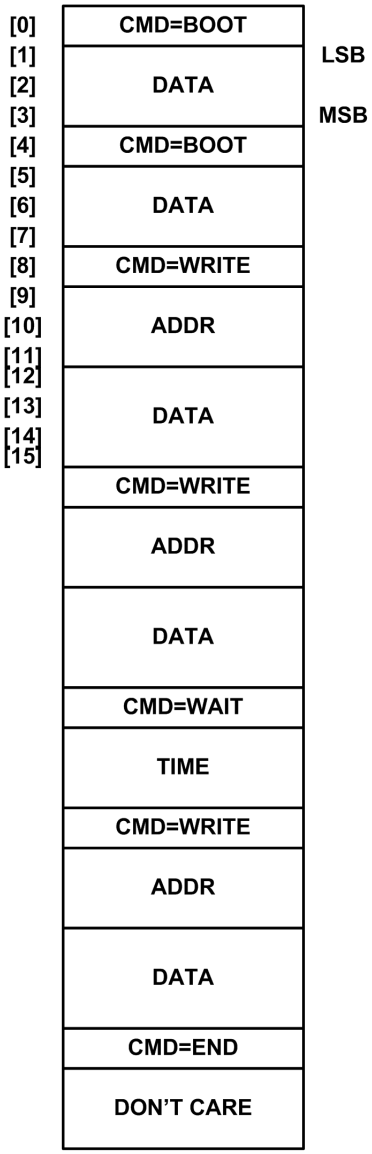
    - For the BOOT rom:
      - Send command BOOT(“Execute memory initialization”)
    - For the CPU:
      - Write command “Execute memory initialization” into BOOT_CTRL::Command register
      - Wait for BOOT_STATUS::CommandDone to return to 1

11. optionally disable FFU

5.3 EEPROM Boot Format

The EEPROM boot format consist of a series of instructions encoded as shown in Figure 14. The SPI controller assumes 3-byte addressing and the I2C 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). For more details on how to program the EEPROM, see the FocalPoint "EEPROM Programming Application Note".

Figure 14: Serial EEPROM Format



The command encoding is the following:

CMD	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:

- 0 => Initialize scheduler
- 1 => Repair asynchronous RAM
- 2 => Initialize memory
- 3 => Repair synchronous RAM

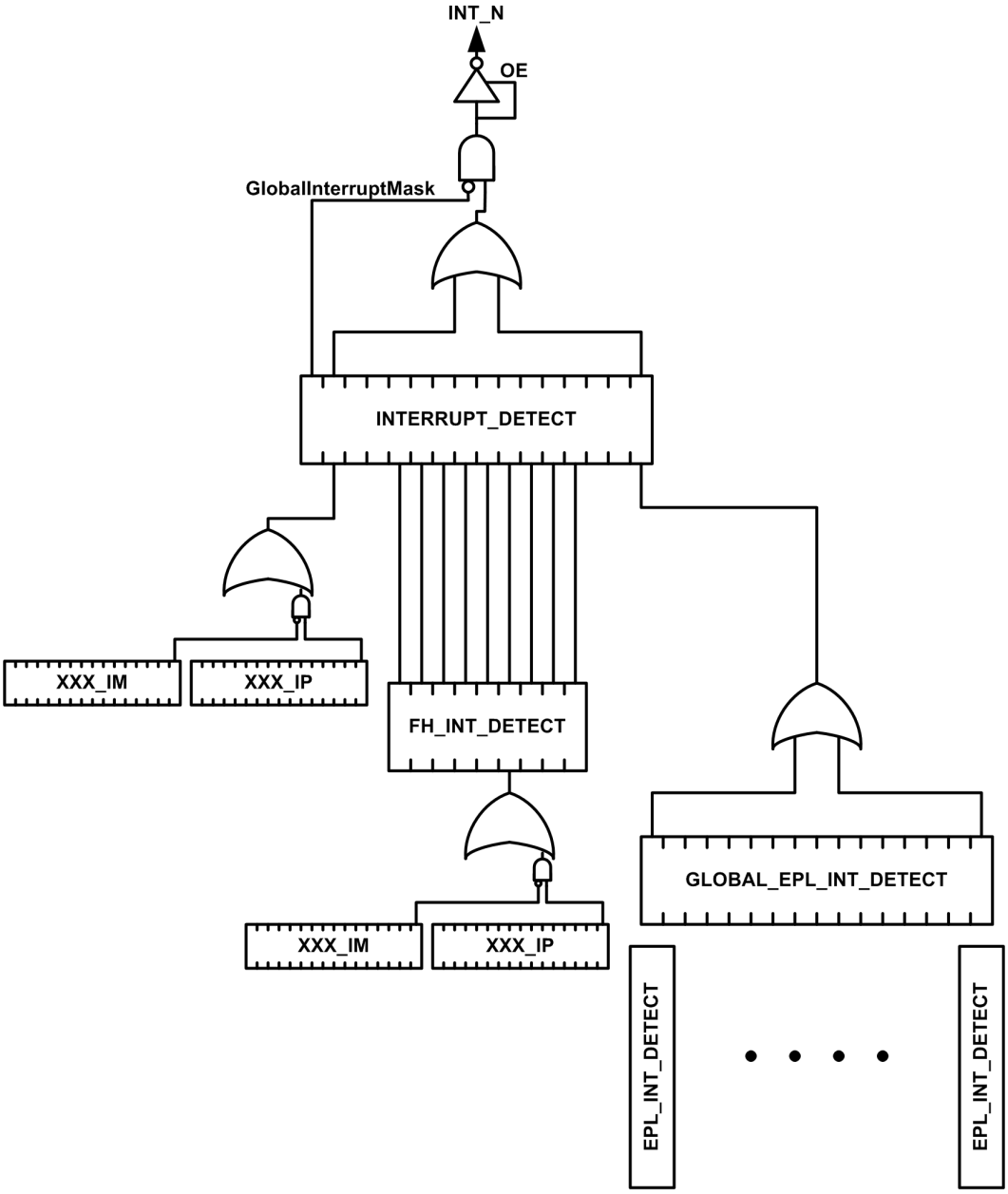
5.4 Interrupt Controller

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 1, while writing a 0 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 1) will 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, then a corresponding fatal code is sent to the interrupt controller and stored into the LAST_FATAL_CODE register and a watchdog reset will be triggered when those interrupts are detected. The Figure 15 shows the interrupt hierarchy.

Figure 15: Interrupt Hierarchy



The main registers are the following:

- INTERRUPT_DETECT
  - o Located in the HSM. Report the source.
- LAST_FATAL_CODE
  - o 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 0. If the CPU writes anything different then the

chip will self reset. The content of this register is not cleared on watchdog reset. The fatal interrupts and associated fatal codes are listed in Table 3.

- 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 3: Fatal Interrupts and Associated Fatal Code

Fatal Interrupt	Fatal Code
Frame Handler Forwarding Table Parity Error	0xC0
EPL Parity Error (VlanTagTable)	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

5.4.1 Memory Errors

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 will retrieve the original packet from memory, compute the CRC while retrieving and compare to the original CRC at the end of the frame. If the CRC is valid, then it is assumed that the packet has not been corrupted. If the CRC is invalid, then the packet will be 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 doesn't 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 will periodically compare the content of the CAM in the chip with the one in the memory and refresh the CAM entry if it gets 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 0 by the software and the switch will automatically compute the right value before writing the content into the memory. Then the switch will validate the parity every time the entry is used and post an interrupt if the parity is found incorrect. Furthermore, any memory read will return if the parity bit is correct or incorrect

(reading into RAM block doesn't 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 1 by the software will cause the switch to automatically compute the inverse of the right value and will thus cause 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 will need 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:

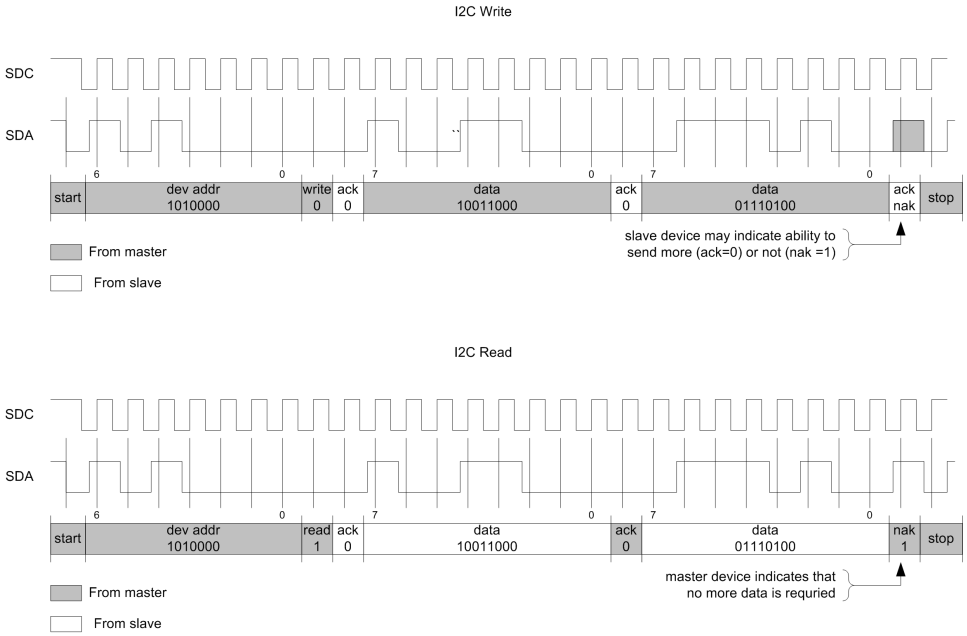
- 100kHz, 400kHz operation.
  - The speed is programmable through the I2C_CFG register and the switch supports a larger set.
- Slave mode allowing external masters to read/write registers into the chip.
  - All registers are accessible except for the LCI_TX_FIFO and LCI_RX_FIFO registers. Those have 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 will attempt to gain the bus only once and return an error if it didn't succeed. The software will have to retry later on.
- 7-bit addressing mode.

The I2C controller doesn't support the following features found in some I2C devices:

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

The following figure shows the basic read and write accesses.

Figure 16: 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 FocalPoint 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 using the EBI clock rate and only reports transitions that are stable for at least 16 cycles.

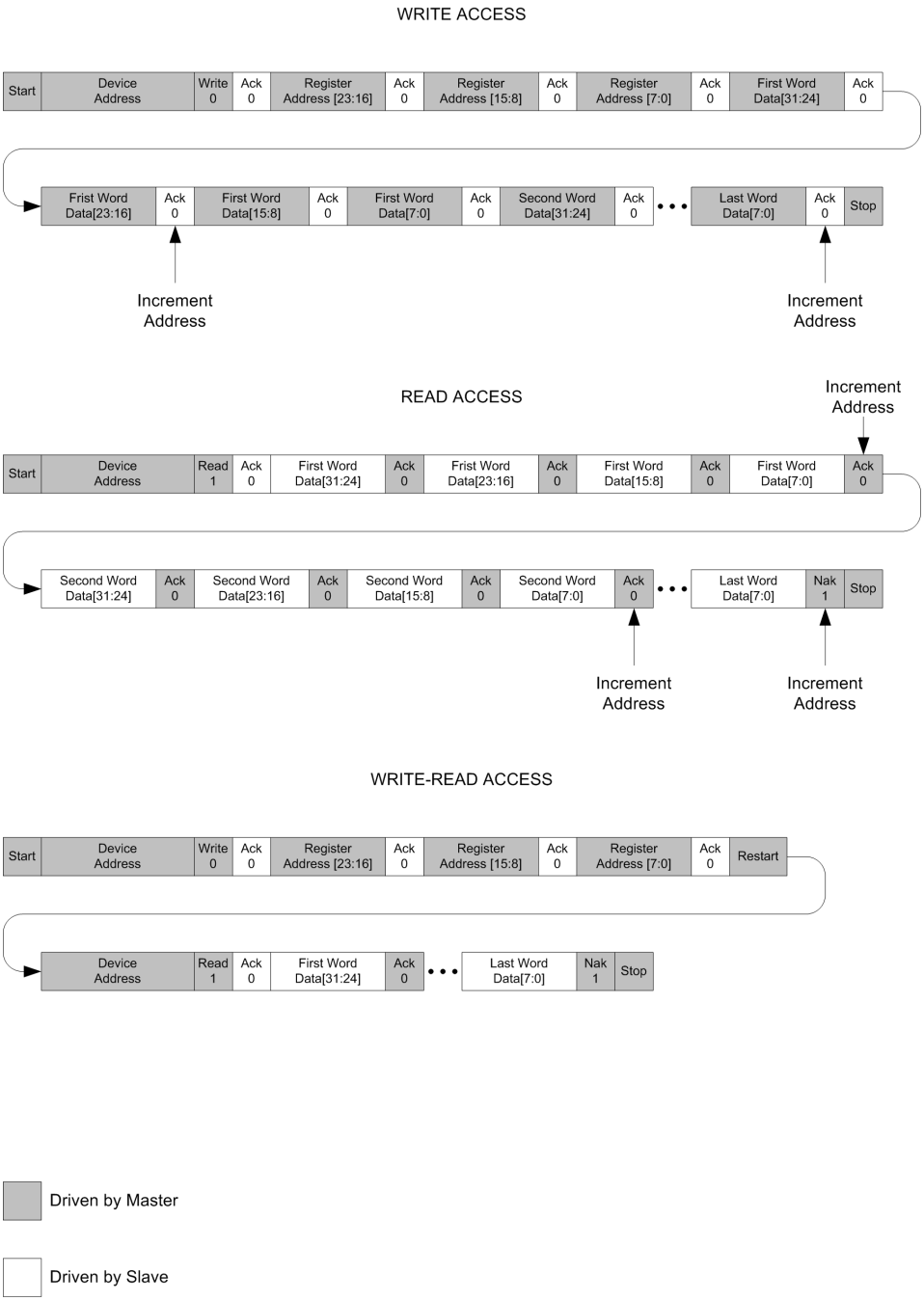
The I2C controller supports a timeout of 100us (1/10 of the I2C clock rate) that aborts the current cycle and returns all pins to 1.

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 will never attempt 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 will terminate with a NAK on the last byte desired, which could any byte.
- The controller will support restart cycles (a stop-start).

This is illustrated in the following example:

Figure 17: Access to FocalPoint Internal Registers via I2C



In the master mode, the I2C controller is capable of issuing automatic I2C accesses:

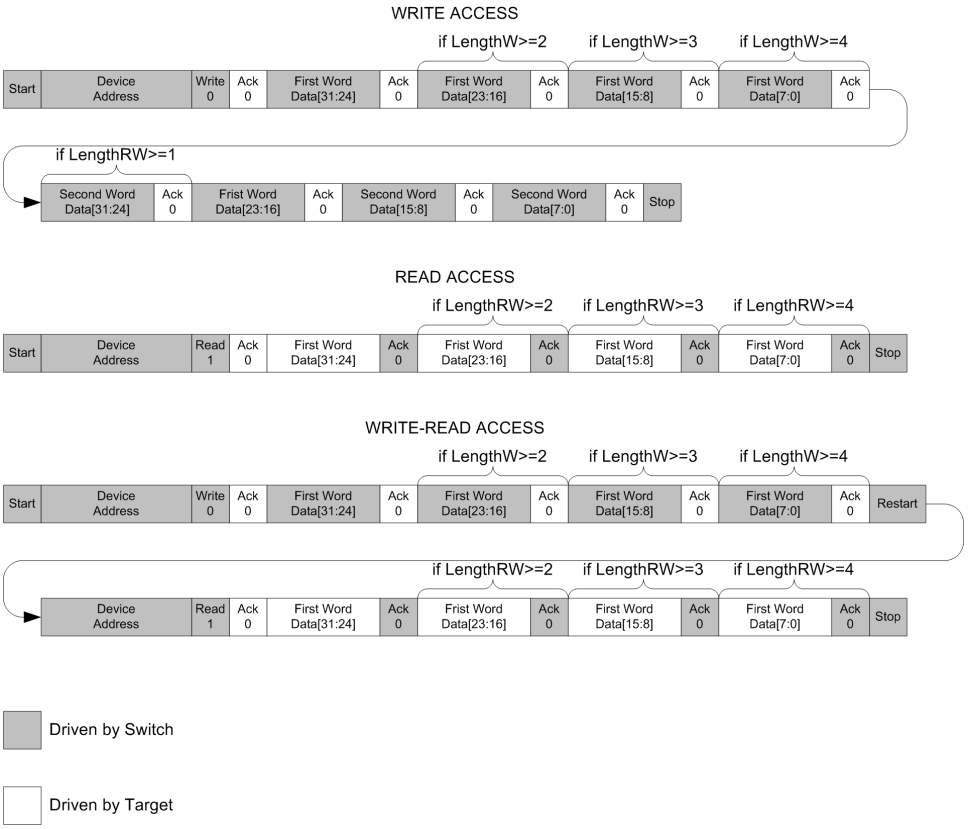
- 'write': can write up to 8 bytes
- 'write-read': can write up to 4 bytes and receive up to 4 bytes
- 'read': can read up to 8 bytes

The I2C registers are:

- I2C_CFG:
  - Enable (1 bit): Defines if the switch will answer to an I2C address from an external master or not
  - Address (7 bits): Defines the address to which the switch will answer
  - 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 I2C 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 I2C 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 the next figure:

Figure 18: Complex I2C Accesses from FocalPoint



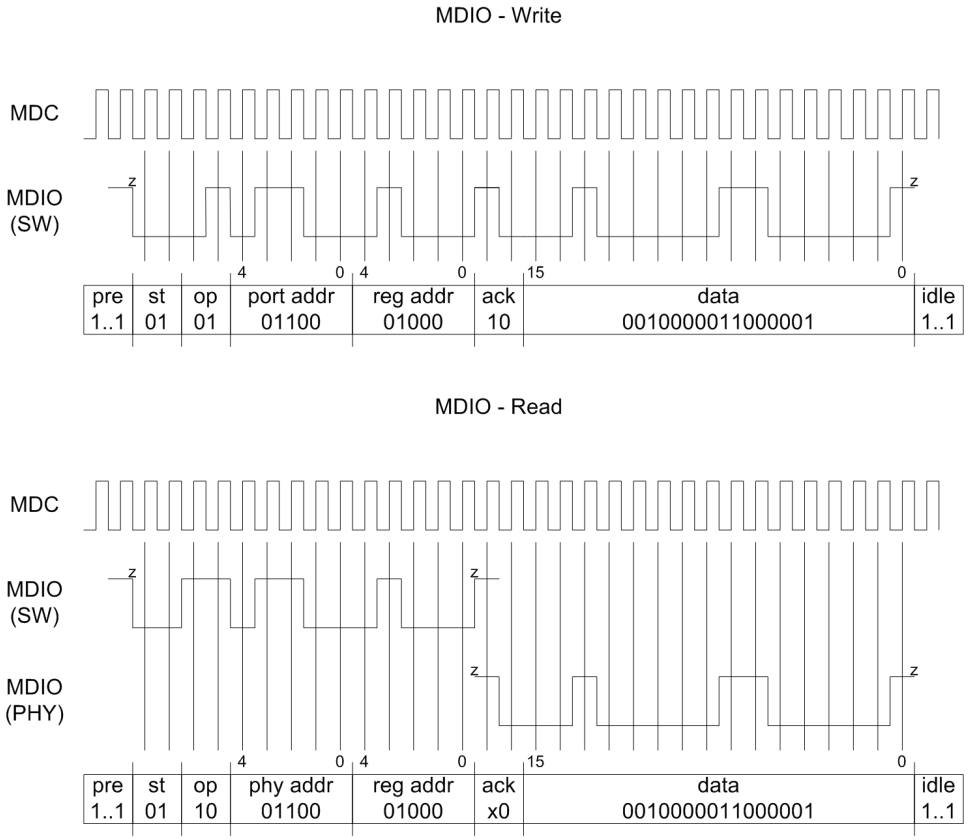
5.6 MDIO Controller

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.
- 3.3V only. An external level converter is required for access to 2.5V or 1.2V MDIO as defined in IEEE802.3ae specification.

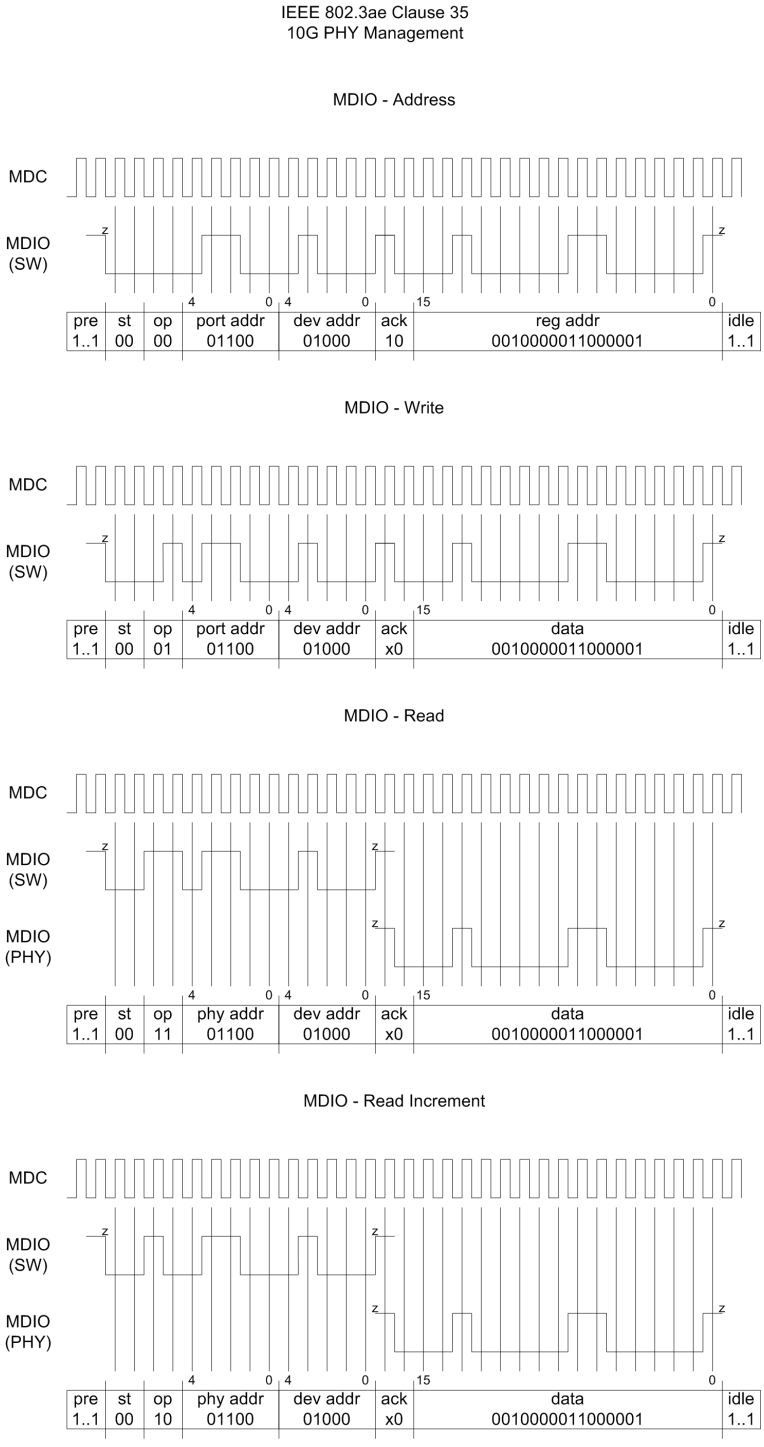
The clause 22 format is shown here:

Figure 19: Clause 22 MDIO Transaction Format



The clause 45 frame format is shown here:

Figure 20: Clause 22 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 1G mode this field becomes the "register" field.)
  - Register address (16 bits) (Note that in 1G 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 frame followed by a read frame.
  - Device type (1 bit)
    - Defines if the frame format is compatible with clause 22 (0) or clause 45 (1). 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 that 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). Also note that 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 4: GPIO and Strapping.

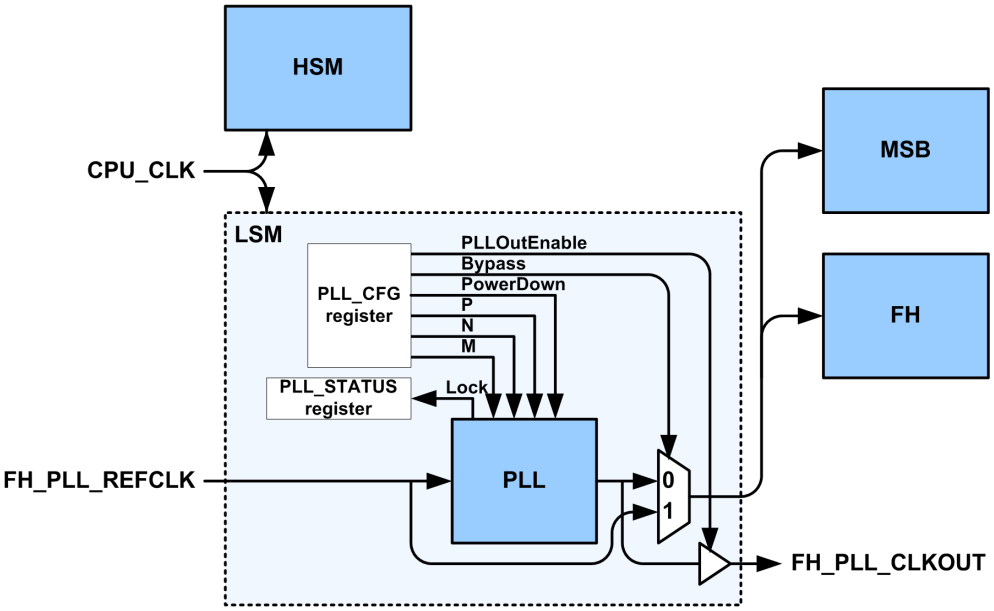
GPIO Pin	Hardware Configuration	Default
0	DTACK_INV	Input
1	RW_INV	Input
2	IGNORE_PARITY	Input

GPIO Pin	Hardware Configuration	Default
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 the next figure.

Figure 21: Frame Handler PLL and surrounding circuitry

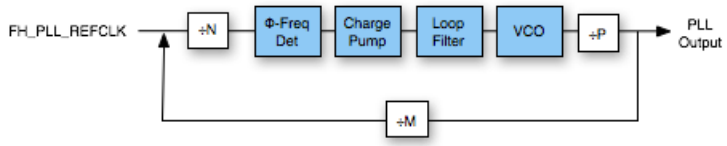


The PLL is controlled by the PLL_CFG register located in the LSM module. The PLL reference input clock is a dedicated pin on the package (FH_PLL_REFCLK) and the output of the PLL is normally routed to the FH and the MSB modules.

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 below, where the roles of the three divisors is evident.

Figure 22: 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 10MHz and 70MHz
- FH_PLL_REFCLK has a duty cycle of 40/60 or better
- FH_PLL_REFCLK jitter must be better than 20ps RMS.
- Vaa and Vdd ripple should be kept between -100 and +100 mV (10MHz - 2.13GHz).
- PLL Output, multiplied by P, must be between 70MHz and 650MHz

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

The register PLL_STATUS will indicate 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 will mark any new frame received with this current clock tick counter. Then, the scheduler 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, then 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 66MHZ, then the timeout period could be:

- If FRAME_TIME_OUT is set to 0x0FFFFFFF, then the timeout period is 8310 seconds.
- If FRAME_TIME_OUT is set to 32226, then the timeout period is 1s.
- If FRAME_TIME_OUT is set to 1, then the timeout period is 31 us.

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

As a special case, writing '1' 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

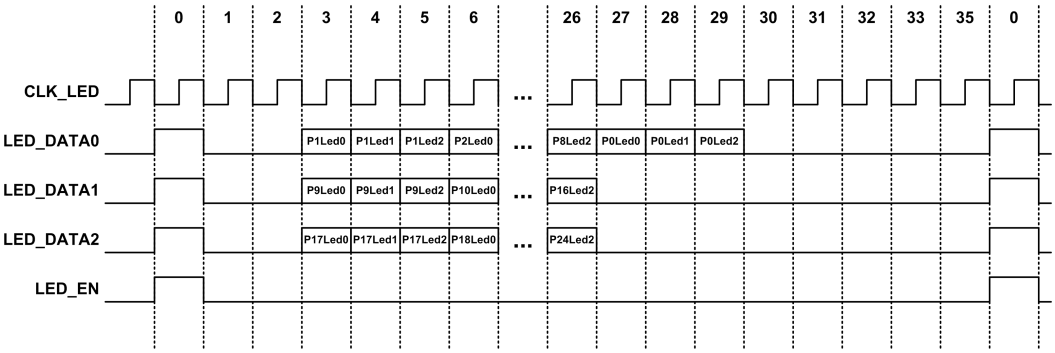
The LED interface consists of 4 signals, CLK, DATA0, DATA1, DATA2, and ENABLE, which transmit 3 bits of status data for the LED per port over the time multiplexed data pins. The 3 bits of status for ports 0-8 are placed onto Data0 and the 3 bits of status for ports 9-16 are placed onto Data1 and the 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 (1) or disabled (0)
- LED_INVERT (1 bit)
  - Controls the polarity of LED_DATA. The values are (0) (as shown in the figure) or (1) (invert of what is shown in the figure)
- LED_MODE (1 bit)
  - Encoding of LED_DATA[2:0]

The overall timing diagram is shown in the next figure.

Figure 23: LED Timing Diagram



The encoding is as follows:

Table 5: LED Definitions

MODE	USAGE
0 (compatible with FM2224)	• LED0: STATUS o Off - Port has no link synch or remote fault error o On - Port has a link synch error or no signal o Blinking - Port has a remote fault • LED1: RECEIVE o Off - Port is not enabled o On - Port has link and is enabled o Blinking - Port is receiving data • LED2: TRANSMIT o Off - Port is not transmitting data o Blinking - Port is transmitting data
1	• LED0 : STATUS o 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 will be momentary, oscillating between this state and any of the link up states. o 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

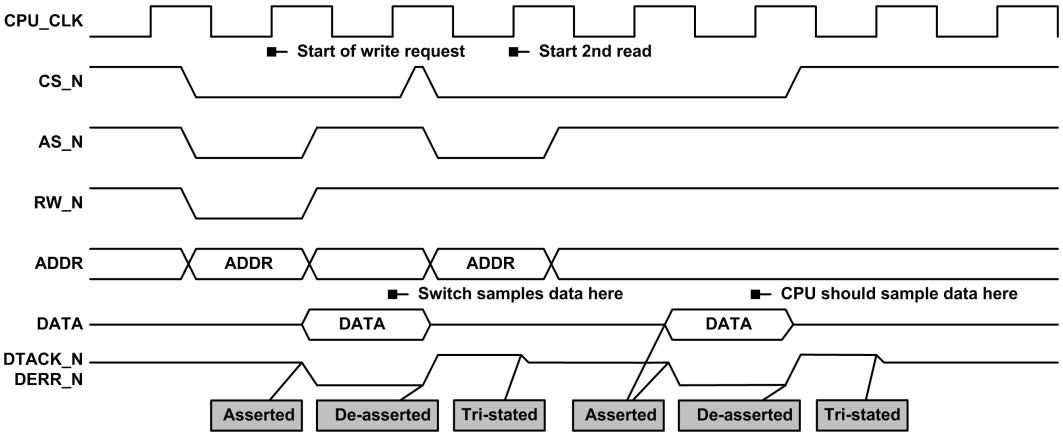
The CPU is configured through configuration pins available in the LSM. The strapping options for CPU are:

- 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 FocalPoint)

5.12 CPU Bus Interface

The CPU bus interface timing diagram is shown in the next figure (assumes DTACK_INV and RW_INV are both set to 0).

Figure 24: 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 then RW_N=1 is READ and RW_N=0 is WRITE. If the RW_INV is strapped to VDD33, then RW_N=1 is WRITE nd RW_N=0 is READ.
- For a write cycle, the data is always sampled on the next cycle. Furthermore, the switch has a small write FIFO and will assert 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 will be 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 will internally cancel the write operation if the parity is incorrect.
- For a read cycle, the switch will delay asserting DATA and DTACK_N until data is available and will drive 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 will actively de-assert DTACK_N for one cycle and then tri-sate 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 125MHZ or above, Frame Handler running at 360MHZ 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:

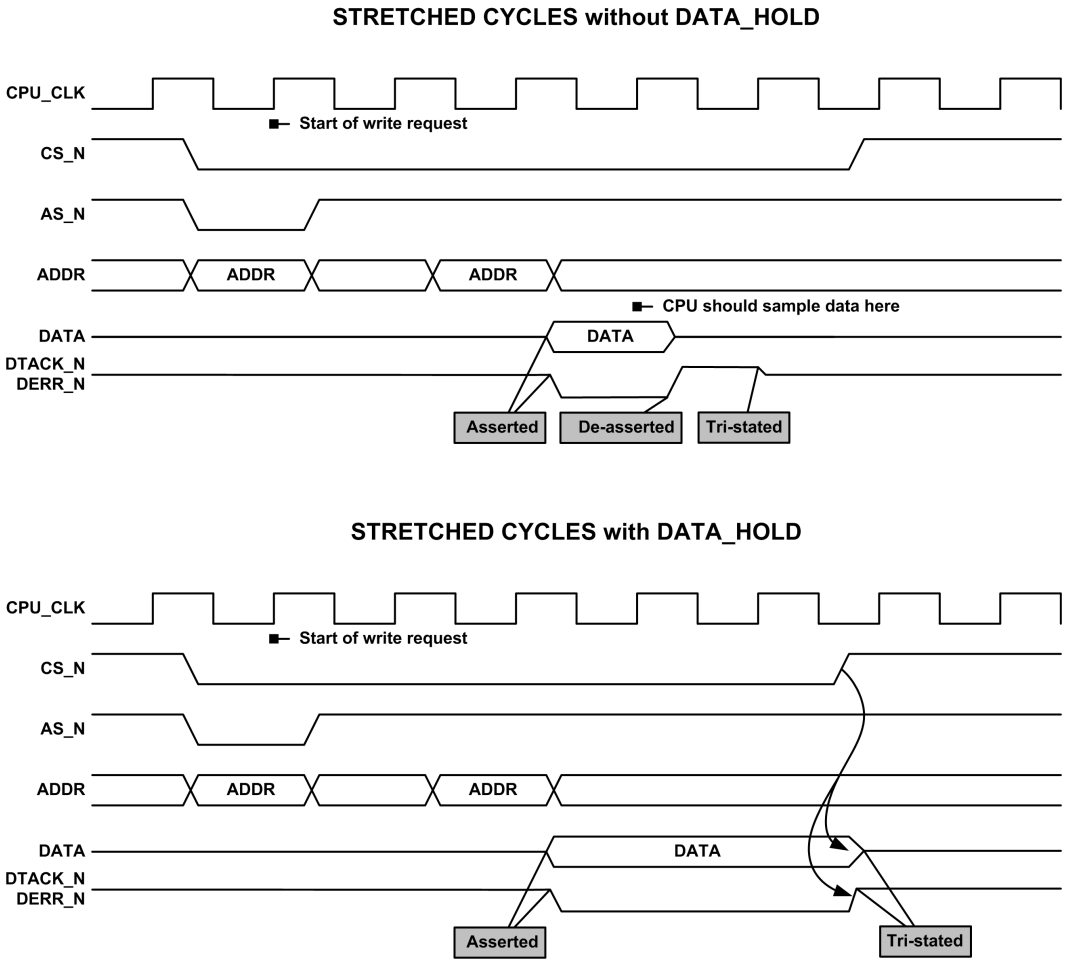
- The FM2XXX samples write data on the same cycle as CS, AS, ADDR and RW_N. Because of that, CS and AS had to be delayed on multiplex buses by one cycle to allow ADDR to be latched and DATA to be driven on the bus. This is addressed in FocalPoint by sampling data on the next clock edge allowing CS and AS to be driven along with the address cycle.

- The FM2XXX waits for CS to fully deassert for one cycle before starting a new cycle. This restriction has been removed.
- FocalPoint is designed to be clocked at up to 133MHZ versus 66MHZ 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 33MHZ). The Figure 25 illustrates the effect of the DATA_HOLD option.

Figure 25: 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 doesn't contain an intermediate incorrect value at any point in time or that the software doesn't read a false value. The tables that supports 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, then the content of the cache is used to return the word addressed.
  - The software doesn't 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) will be 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. By example the default IEEE assigned LACP frame will be 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 will be 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 Fulcrum API takes care of this).

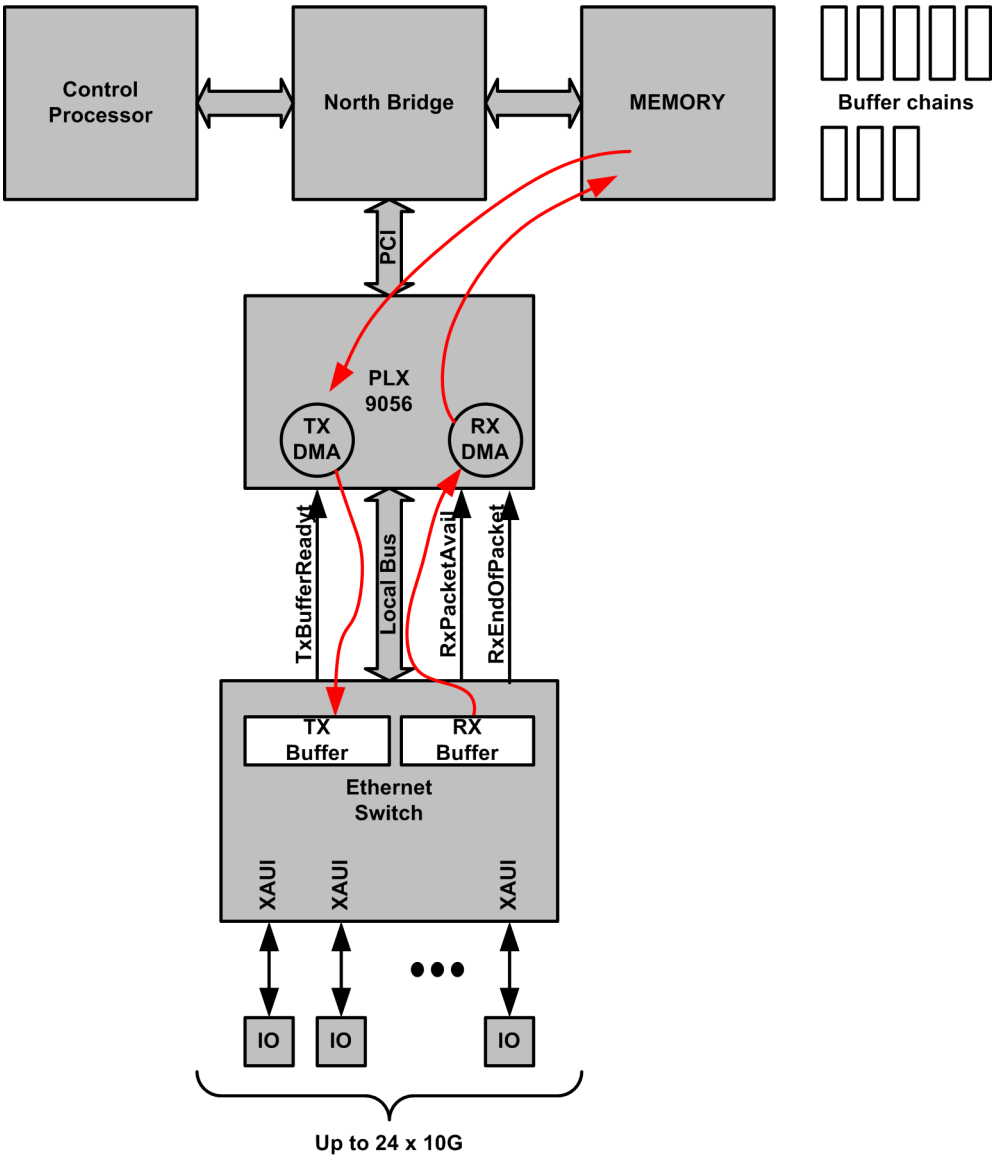
However, the transfer of packets is not negligible and poses a challenge because the byte ordering within memory will not be the same between a little- and big-endian processor. To avoid the processor having to swap bytes, the switch offers an endianism option for byte ordering in the LCI_CFG register.

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 FocalPoint) 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

FocalPoint supports packet transfers between the CPU bus interface and any of the switch's 24 external XAUI ports. Since the FocalPoint 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. FocalPoint 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 the next figure.

Figure 26: DMA Transfer



5.13.1 Packet Transmission

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

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, then the bus master must wait, possibly polling TxReady, until the transfer fifo is ready. This condition will never persist 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 FocalPoint will calculate and attach 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 FocalPoint 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 will 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 will be handled in the following manner:

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

Table 6: LCI_TX_CMD Details

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

5.13.2 Packet Reception

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

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 will be 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 will 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 will read '1'.
    b. The EndOfFrame interrupt in LCI_IP will be set.
    c. The RXEOT_N external pin will transition low (timing illustrated below).

The packet received will include 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.

Focal Point supports 2 levels padding.

Level 1 padding is done in MSB to make the packet 32-bits alignment. This will always be 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 '1'.

The format of the RX status word is provided below.

Table 7: LCI_RX_STATUS

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

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 0.

Table 8: Packet Transmission Format for Little Endian CPU

	31:24 MSb	23:16	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 9: Packet Reception Format for Little Endian CPU

	31:24 MSb	23:16	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 1.

Table 10: Packet Transmission Format for Big Endian CPU

	31:24 MSb	23:16	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

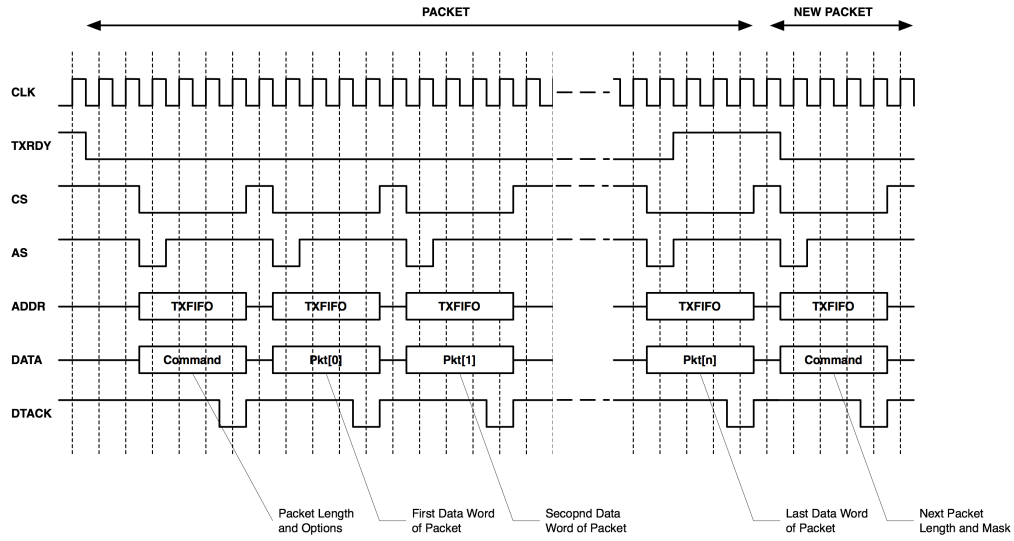
Table 11: Packet Reception Format for Big Endian CPU

	31:24 MSb	23:16	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

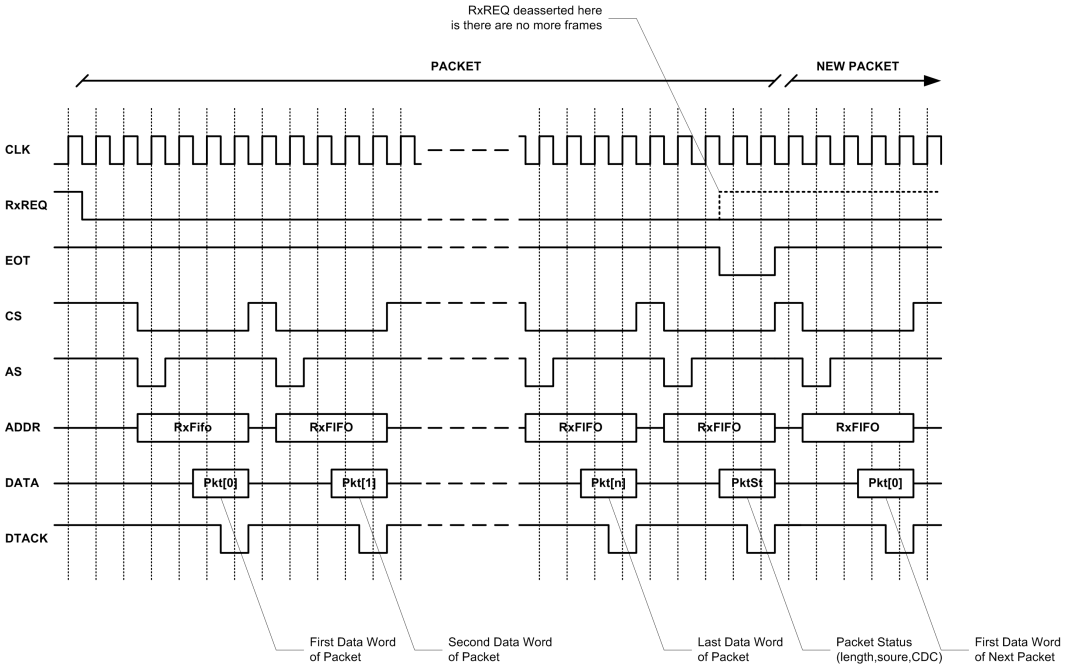
Packet transmission with an external DMA controller is illustrated in the next figure. The external TXRDY_N signal reflects the state of the TxReady bit the LCI_STATUS register. It is asserted whenever FocalPoint is ready to accept a word from the CPU. The DMA controller can write data words to LCI_TX_FIFO 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 CPU Bus Interface section above).

Figure 27: DMA Packet Transmission



The timing of packet reception with an external DMA controller is shown in the next figure. The RXREQ_N signal reflects the state of the RxReady bit in the LCI_STATUS register. It is asserted whenever FocalPoint 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 being read on the bus (the LCI_RX_STATUS word), FocalPoint will assert 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 28: 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 defined below:

- 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...
  - The table below list all possible traps. All traps are sent to the special glort 0xFFnn where 'nn' is the trap code listed below.
  - Trap IP will trap both unicast and multicast IP frames with options, so to differentiate these, a trigger can be used or trapping can be set up using the FFU (the 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) to have control over unicast vs. multicast.
- 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 will encode the reason for the trap.

Table 12: Frame Trap Codes

Code	Description
0x00..0x3F	Trapped due to a trigger action. The code will be set to the trigger number that specified the trap action.
0x40..0x7F	Logged (mirrored to CPU) due to a trigger action. The code will be 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.
0x96	Trap code for frames with TTL <= 1.
0x97..0xEF	Reserved.
0xF0..0xFF	Reserved for software use. Possible uses:

Code	Description
	• 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 32b:

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

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 support 3 byte addressing
- Support of one Chip Select
  - o The EEPROM size is restricted to 64Kb – 2Mb – this is sufficient for about 30k instructions in a 2Mb part.
- Support of one Mode 0 (CPOL=0,CPHA=0 – transmit data on the falling edge of the SPICLK and receive data on the rising edge of the SPICLK signal) device (only one CS required)
- Support 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 13: SPI External Pin List

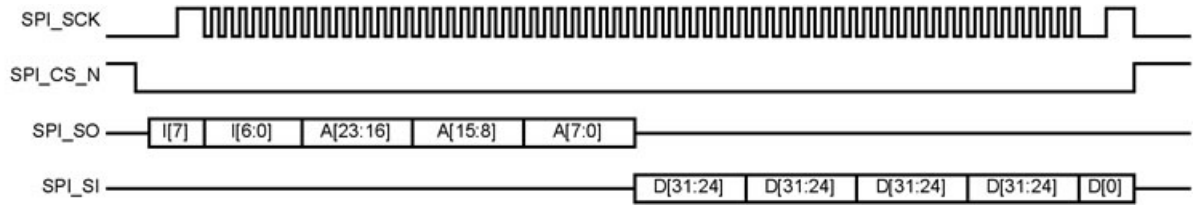
Signal name	Signal direction	Signal description
SPI_MOSI	OUT	Serial Data Output (MOSI, Master-Out- Slave-In, since FocalPoint switch is master). Connect to EEPROM serial data input
SPI_CS_N	OUT	SPI Chip Select (Active Low)
SPI_SCK	OUT	CLOCK for SPI interafce.

SPI_MISO	IN	Serial Data Input (MISO, Master-In-Slave-Out, since FocalPoint switch is master). Connect to EEPROM serial data output.

A SPI transaction is shown in Figure 19 and described below:

- 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 byt es of address (A[23:0])
- On the positive edge of the clock, receive each bit of data. This will continue until BOOT FSM asserts
- De-activate SPI_CS_N, Tri-state SPI_MOSI.

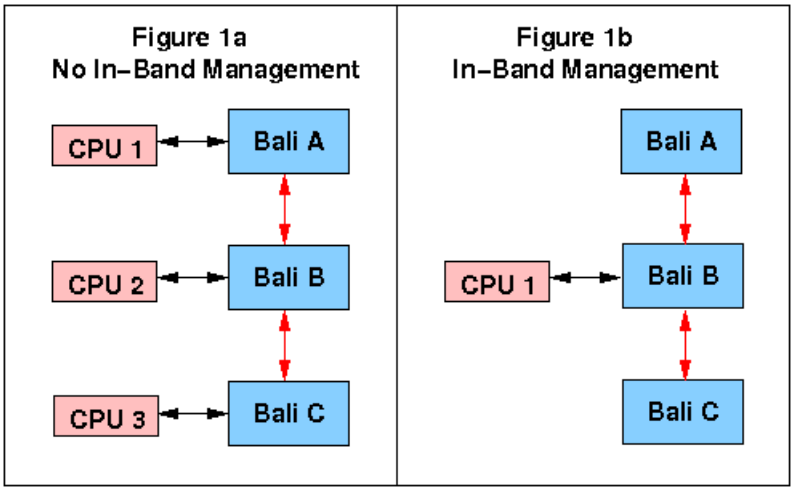
Figure 29: SPI Timing Diagram



5.16 In-Band Management

5.16.1 Overview

Fulcrum In-Band Management (FIBM) is the management of one or more Bali chips through management commands encapsulated within Ethernet frames. This means that all Bali 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 Bali chips. This is illustrated in Figures 1a and 1b below (black arrows are EBI buses and red arrows are Ethernet channels).

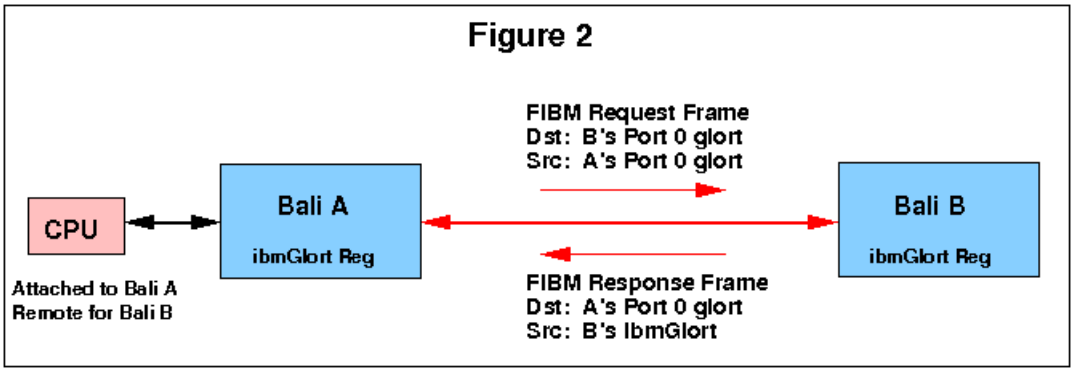


There is no fixed limit to the number of Bali chips that one CPU can manage through FIBM. The maximum number is determined by the bandwidth of the EBI bus divided by the bandwidth required for each chip. The bandwidth required for each chip is highly system dependent.

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 64B. 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 2 below. Management (FIBM or otherwise) is done through Port 0 of the switch.

Figure 2



5.16.1.1 Management By an Attached CPU

This is the case for Bali A in Figure 2 above. While the Bali A is not being managed by FIBM, the CPU may be managing other Bali chips (like Bali B) through FIBM. In this case, Bali A receives FIBM request frames from the CPU and forwards them to Bali B, and it receives FIBM response frames from Bali B and forwards them to the CPU.

5.16.1.2 Management By a Remote CPU

This is the case for Bali B in Figure 2 above. The remote CPU sends FIBM request frames to Bali B, and Bali 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 1b above shows a simple case of using one CPU to manage three Bali's. Bali B is being managed by an attached CPU, and Bali's A and C are being managed by FIBM. If the CPU wants to issue the same FIBM request to both Bali 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 will use it's 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 Bali B, it is not managing Bali B through FIBM. This is not necessarily the case. The CPU can still manage Bali B through FIBM. If the CPU sends a frame to Bali B's ibmGlort, the frame will follow the path: CPU --> HSM --> Crossring --> MSB --> Port 0 --> Switch --> Port 0 --> MSB (see Figure 5). 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 Bali B can be managed by FIBM, it is a less efficient use of Bali 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 3 below, all three Bali's are managed by a single CPU and by FIBM. The CPU is attached to an Ethernet port on Bali A via a NIC. The potential advantages of this type of configuration are:

- The CPU will have a higher bandwidth connection to the system -- 10 Gbps for Ethernet vs. ~3 Gbps 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 Bali management.

The disadvantages of this type of configuration are:

- It uses a port on Bali A.
- It requires a NIC.
- The NIC may not support the Fulcrum F64 ISL tag, and if not it will require some FFU (Filtering Forwarding Unit) resources on Bali A.

Figure 3

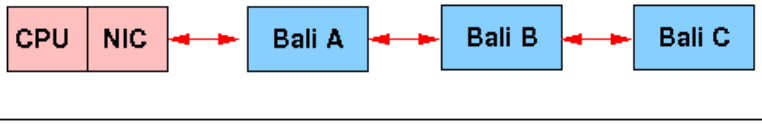
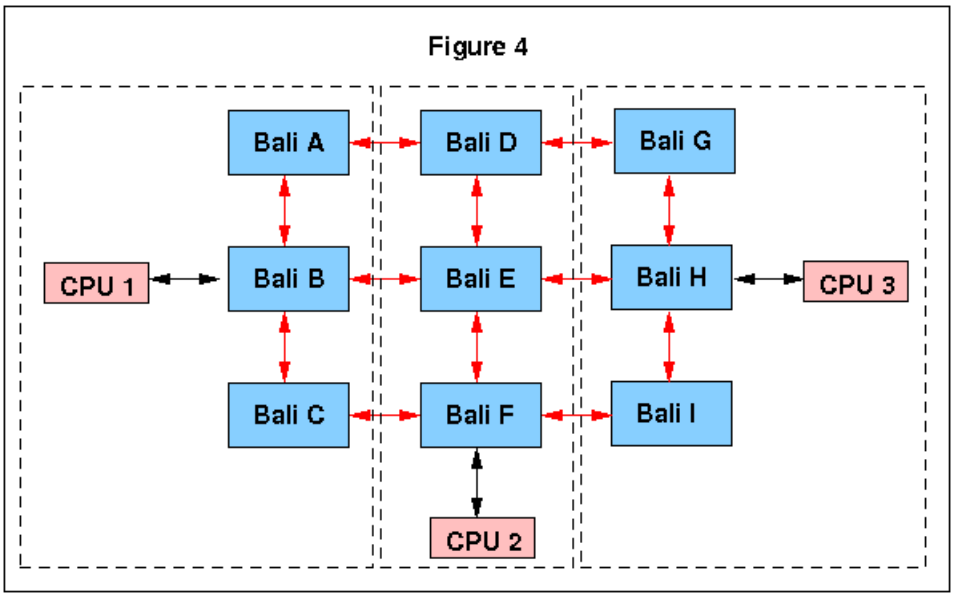
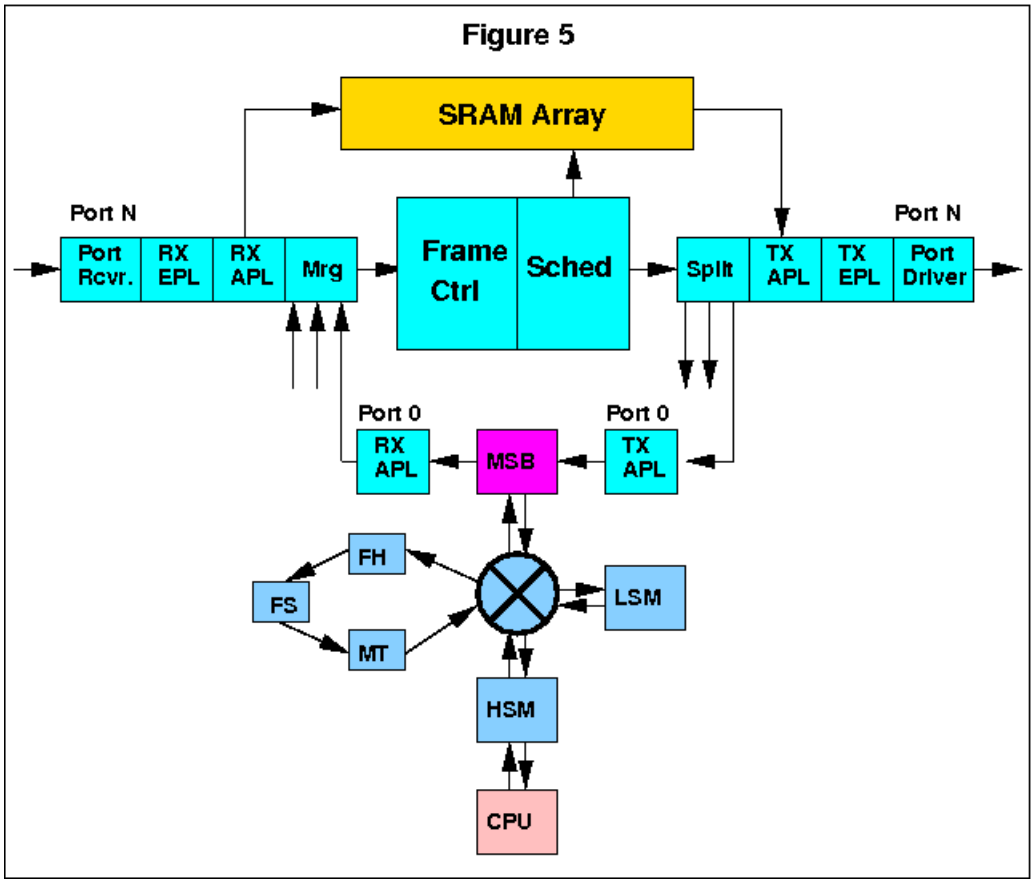


Figure 4 below shows a more realistic system using FIBM. In this example, Bali's A, B, and C are managed solely by CPU 1, and Bali's D, E, and F are managed solely by CPU 2, and Bali's G, H, and I are managed solely by CPU 3. Managing a single Bali with multiple CPU's is problematic and is beyond the scope of this document. However, if CPU 2 in Figure 4 wanted to do something simple like read the registers in Bali A, it could do so.



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 Bali is shown in Figure 5 below. 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.



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 table below:

Incoming Frame (From Port 0 to MSB)				
F64 ISL Tag FTYPE	F64 ISL Tag MTYPE	disableIbm register bit	attachedCpu register bit	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)
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, then:

- 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 will cause the FIBM response frame to be dropped.

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

5.16.3 FIBM Frames

5.16.3.1 FIBM Frame Format

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

FIBM Frame						
Header				Payload		
Dst Mac	Src Mac	F64 ISL Tag	Ethertype	FIBM Tag	FIBM Ops's and Data	CRC
6B	6B	8B	2B	2B	variable	4B

	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 Bali.	Src glort of the request frame.
F64 SWPRI/USER		Same as the request frame.
F64 VPRI/VID	Ignored	Set to 0
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 table below. 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 will already have 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 will:

1. Raise a fatal interrupt.
2. Set a bad CRC on the FIBM response frame, and set a tail error on the frame. This will cause the egress EPL to discard the frame.

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

1. Raise a non-fatal interrupt.
2. Set the error bit in the tail word and send the frame on to the attached CPU.

The MSB does not check the CRC on outgoing frames from an attached CPU. However, it will append 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 4b 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 Bali idempotent. In particular, this means that Bali 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 whether the Read operation was executed or not (whether the FIBM request frame got lost or the FIBM response frame got lost), and 2) if the Read operation has already been executed, be able to find out what the original data was. Sequence numbers can be written to scratch registers as described above. In addition, FIBM Read/Write operations should be used to read non-idempotent registers. The Read/Write operation has a 22b address for the non-idempotent register and a 4b 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 will be received by the CPU, interrupt frames are sent repeatedly until the interrupt is cleared. See the Interrupt section below.

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 will arrive at all), FIBM operations cannot be

executed at a precise absolute time. If a Bali 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 table below. 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 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 table below 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), then FIBM response frames will be padded to 64B.

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

5.16.5.5 FIBM Header Words

Each FIBM operation is defined by a 32b header word as shown in the table below:

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		Scr. Reg. Addr.						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 64B, 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; then 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), then 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 will simply start 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 Bali like the other FIBM Ops – they are sent from a Bali 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 Interupt Handling

The top level INTERRUPT_DETECT register is in the HSM, and, with an attached CPU it will handle interrupts as normal. However, with in-band management a remote CPU must be notified of an interrupt. In this case, the HSM will assert an interrupt signal to the MSB, and the signal will stay asserted until the interrupt is cleared. If MSB_CFG::interruptGlortEn is set, the MSB will repeatedly send 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 16b 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 usec. This gives a range of ~21.8 usec to ~1.43 sec.

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

Interrupt Frame		
Dst MAC		0x00000000
Src MAC		0x00000000
ISL Tag	FTYPE	0b11
	MTYPE	0b01
	SYSPRI	MSB_INT_FRAME::islSysPri (4b)
	USER	MSB_INT_FRAME::islUserBits (8b)
	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
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 will 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 will make it to the CPU. If this is just because of an isolated instance where the frame got dropped, repeating the interrupt message will solve 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 Bali 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 will cause a Bali 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, then the HSM will send a signal to the Watchdog unit in the LSM to reset the chip. It also sends the Watchdog an 8b 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

A Bali without an attached CPU will boot from an EEPROM. The boot sequence will do the following:

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

When the boot sequence is complete:

- The MSB will see 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 will see that it has an interrupt, and it will set a bit in the INTERRUPT_DETECT register in the HSM.
- The HSM will see that it has an interrupt, and it will assert the interrupt signal to the MSB.
- The MSB will periodically send out interrupt messages to MSB_IBM_INT::interruptGlort until INTERRUPT_DETECT is cleared.
- The CPU will:
  - Receive an interrupt frame.
  - Read INTERRUPT_DETECT and MSB_CFG::disableIbm. From this, the CPU can determine whether 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

Bali provides a general counter rate monitoring service in order 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 will be raised thereby alerting the CPU. A total of 256 counter monitors may be instantiated by software.

Relevant registers:

- 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: 0 - 32 bits [default] 1 - 16 bits 2 - 8 bits

Parameter	Bits	Description
RateWindow	24	Window over which RateLimit is defined, in units of the global PollingPeriod parameter. A value of 0 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 will be set.
Enabled	1	Monitor is disabled when set to 0.

Note that the monitors do not directly support multi-word counter registers. It is expected that the monitors will 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 0.
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 0.

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], then an interrupt will be 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, 10Gb/s port octet counter imposes a maximum PollingPeriod of 2^32 / (10Gb / 8b) = 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 40MHZ.

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. Ie. 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 the table below.

Instruction	Code (6b)	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.

Chapter 6 - Frame Filtering and Forwarding Unit

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.

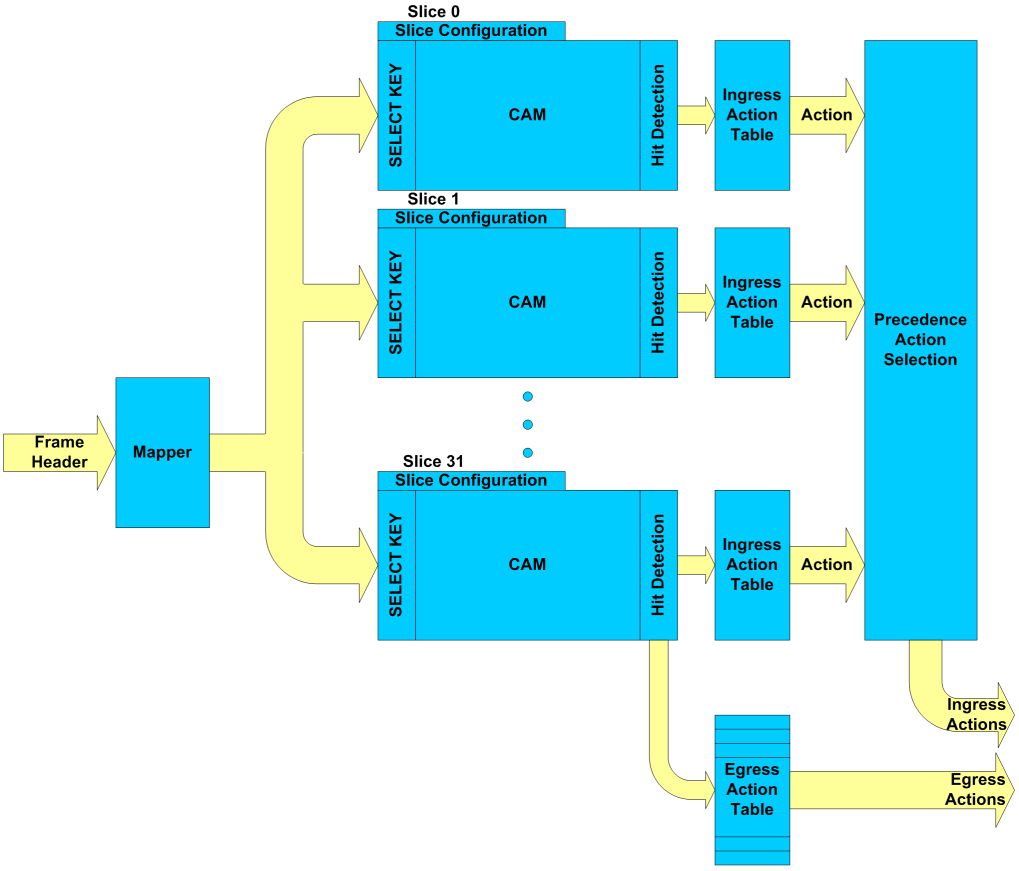
Each slice has the following elements:

- 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 30.

Figure 30: FFU Unit



Each TCAM entry has 74 bits of configuration:

- 36-bit mask
- 36-bit key
- 1 valid bit
- 1 case bit (see section Slice Activation and Overloading section)

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 (see FFU_MAP_PROT registers)
  - o The mapper will recognize up to 8 L4 protocols. If the protocol received matches one of these protocols, then the 3-bit MAP_PROT field associated is retrieved, or 0 if the protocol doesn't

match. The 4th bit of MAP_PROT will be 1 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 (see FFU_MAP_TYPE registers)
  - The mapper will recognize up to 16 Ethernet types. If the Ethernet type received matches one of the known ones, then a 4-bit field associated with the protocol is retrieved. If there is no match, then this 4-bit field is set to 0. Useful for detecting non-IP Ethernet frames, such as ARP. The scenario distinguishes IPv4 and IPv6 from other L3 protocols, so MAP_TYPE isn't needed for that purpose.
- Maps 48-bit DMAC and SMAC addresses into a 4-bit MAP_DMAC and MAP_SMAC fields (see FFU_MAP_MAC registers)
  - The mapper will recognize up to 16 Ethernet MAC addresses. If the frame's DMAC address matches one of the known ones (and its validDMAC bit is true) then a 4-bit MAP_DMAC field associated with the MAC address is retrieved and also sets a router bit indicating if this DMAC address is an address of one of the virtual routers. If there is no match, then this 4-bit field is set to 0. 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 in order to do IP routing. The entry can be configured to ingore 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 (see FFU_MAP_SRC registers)
  - The mapper will retrieve 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 (see FFU_MAP_L4_DST and FFU_MAP_L4_SRC registers)
  - The mapper will distinguish 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 (see FFU_MAP_IP_XXX registers)
  - 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), then a 4-bit field associated with this IP address prefix is retrieved. If there is no match, then this 4-bit field is set to 0. 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 will be padded with 0's before being compared to the 128-bit value in the registers.
  - 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), then a 4-bit field associated with this IP address prefix is retrieved. If there is no match, then this 4-bit field is set to 0. Each register contains a value and the number of least significant bits to ignore. This feature is especially useful when most ACL rules use only a

few common prefixes. A 32-bit IPv4 address will be padded with 0's before being compared to the 128-bit value in the registers.

- Maps VLANs into a 12-bit MAP_VLAN key (see FFU_MAP_VLAN table)
  - 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 (see FFU_MAP_LENGTH registers)
  - The mapper will recognize 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, then 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:

- Bit 0-1: Frame format:
  - 0: Not an IP frame
  - 1: IPv4
  - 2: IPv6
  - 3: IPv6 with an IPv4-in-IPv6 DIP. Can optionally use IPv4 routing rules for this frame.
- Bit 2-4: Frame handling:
  - 0: **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.
  - 1: **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.
  - 2: **mgmt:** FTYPE is "mgmt". Generally, the FFU should do nothing to these frames.
  - 3: **special:** FTYPE is "special". This usually means mirror traffic or CPU directed control frames, but ACL's may be used.
  - 4: **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.
  - 5: **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.
  - 6: **routable-multicast:** This frame is IP multicast, the ingress port and ingress VLAN are marked as routable, the 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.

- 7: **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_VALID 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 using the FFU_SLICE_CASE_CFG[0..31][0..1] registers. Each of these registers has 5 x 6-bit fields plus 2 additional control bits, each field allow setting of up to a 64-way selection. The fields available 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)

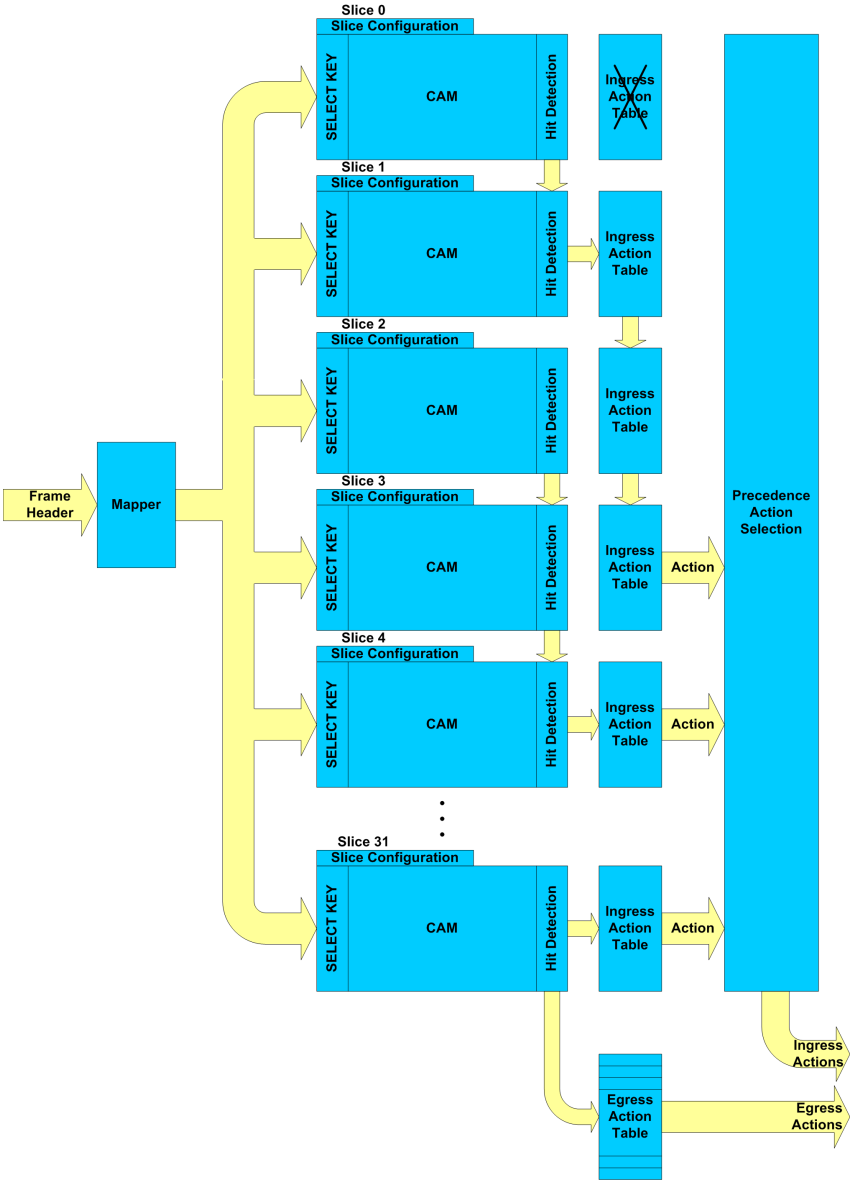
Note that the ISL tag received with the frame is originally 64 bits long but is reduced to 48 bits before being presented to the FFU by extracting the VLAN/VPRI/CFI bits from the ISL tag.

A table in FFU_SLICE_CASE_CFG defines the exact mapping supported by the switch.

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 31.

Figure 31: 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”.

If there is more than one match, then the highest entry wins.

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, then 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, then 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, then 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 1, the frame will be dropped. If NOROUTE is set to 1, 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 will allow 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 the POLICING section 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==00XXXXX -> Reserved
ACTION==01XXXXX -> 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 will be 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 bit field:

- drop
- 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 will be all 0's but the frame Action Code will be 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.

Chapter 7 - 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 will parse 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 will associate 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 mark the frame as “switched”. The egress VLAN is set to be equal to the ingress VLAN.
- ROUTABLE: The frame handler will then use 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 is routable and the port is routable and the destination MAC address is routable. The frame is then marked as “routable”.
- IP LOOK UP: The frame handler will parse 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 example. Consult the FFU chapter for more details.
- ARP LOOKUP: The ARP table is indexed with the ARP index (if the ARP count is greater than 1, then 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 Fulcrum ISL tag will reflect this change of status if the frame exits this chip on a Fulcrum 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 mutlicast 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 will search 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, then the destination glort will be 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 will also present 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 include a 3-bit MTU index size which is used only if the arp entry was 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 at the end of this chapter. Other traps also exist and are listed in the layer 2 chapter.
- 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) will replicate the frame mutliple 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 used for multicast (as recommended), then the scheduler will assume that the frame is actually associated with the ingress VLAN instead.
- TRANSMIT MAC: The transmit MAC will then detect that the frame is marked “routed” and will change 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...

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 will not apply the layer 3 modifications on the internal ports. The content of the original frame will however be modified to include an ISL tag which will have 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 will still be done to learn the source MAC address using the source glort received as part of the ISL tag but will not do a destination lookup as the destination glort is already known. The PORT MAP will determine the

next ports for the frame. The TRANSMIT MAC will simply forward 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 will ensure that the layer 2 processing will not filter out the packet in any way.
- Egress switch:
    - The egress switch will perform 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), then the destination MAC is not changed while the egress VLAN will be set to 0 for a Fulcrum 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, then 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 will compute a hashing function using IP fields before indexing the table. The output of this hash function is is used to compute the final index and is typically used for implementation of ECMP. The details on the hashing function can be found in the hashing chapter. Note that 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 is 16K x 64 and contains three fields:

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

Note that if a destination GLORT is specified, then 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, then 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, then the source address of the frame will be replaced by the one defined in the SRC_MAC_LO/HI registers in the EPL. If the virtual routing id is not 0, then the source address will be 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 16K 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 1 to it while writing a 0 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 = 1 or TTL = 0
- Log routed IP multicast frames with TTL =1 or TTL = 0
- 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

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 the Layer 2 Lookup chapter. 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 below.

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.

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 that 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.

Trapping of frames that exceed MTU size

The MTU size is checked for any IP frame. FocalPoint 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, then the packet is declared oversized for this VLAN and the parameter SYS_CFG_1:trapMTUViolations defines the disposition of the frame - either ttrap to CPU or silently discard.

Every egress VLAN has a default MTU size, which can all be set to 16K if needed. Any routed frame will then overwrite the egress VLAN MTU with whatever is in the ARP Table. This means that all IP frames will 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.

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 frozed from the moment they were 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 will be discarded.

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 that 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..

Chapter 8 - 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 will be 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 will be set to the XcastGlort(MA_TABLE_CFG_2::XcastGlort).
- If the destination MAC address is known and the entry is of type MAC-DMASK, then the destination mask is first set to this value and the destination GloRT will get 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, then the destination mask is first set to all 1s and the destination GloRT will get updated to the value of this entry if it was received as 0.
- The destination mask then gets 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 will not learn broadcast, multicast or address 0 if they are used as a source address.
- Automatic aging
  - A background aging process will invalidate learned MAC addresses from the MA_TABLE that have 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 preventing thus 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 disabed, 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

Bali 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.

Bali 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.

Bali 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 will be associated with the default VLAN ID and will 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, then the content 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 through multiple level 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 (See MAC_VLAN_ETYPE_1 and MAC_VLAN_ETYPE_2). 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
  - Bali 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.

Bali 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, then 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 the port mapping chapter under the loopback suppression section.
  - 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 bits)
  - Spanning tree number (FID) (12 bits)
  - MTU Size Index (3 bits)
    - Consult the routing chapter for MTU size checking.
- 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 else 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)
  - 00: not used

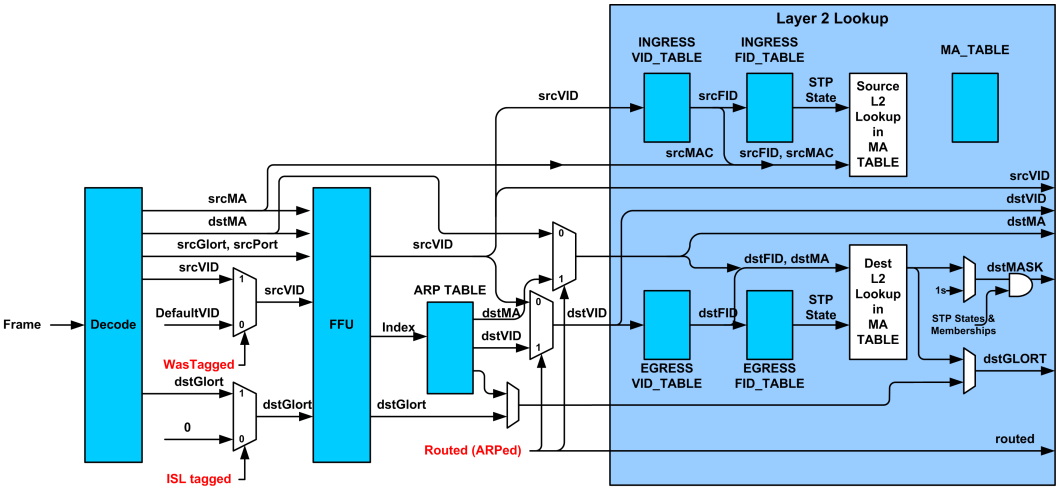
- 01: old (will be deleted on next aging update)
- 10: new (will be changed to old on next aging update unless used again)
- 11: locked

- 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.
  - Or GloRT (most common)

8.4 Layer 2 Lookup Flow

The overall flow of the layer 2 lookup is shown in the next figure.

Figure 32: 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 will only use the outer VLAN tag and will skip 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, will force the VID to the default VID. The flag “WasTagged” is only set if “useDefaultVLAN” is cleared and the packet was 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, then the ARP table will return a new destination VID and may return a new destination MAC address or a destination GloRT. If the packet is not routed, then 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 will take precedence over the destination GloRT received.
- The source VID is used to index the INGRESS_VID_TABLE in order 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, then the packet is dropped.
    - If the spanning tree state is LISTENING, then the packet is dropped.
        - Some packets (such as BPDU, LAG, 802.1x) can be trapped even when the port is in listening mode. The overview chapter details the event priority.
    - If the spanning tree state is LEARNING, then the packet is dropped but the source address / source FID pair will be look up in the MA table and will be 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, then 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 in order 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 will be 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, then the broadcast GloRT (MA_TABLE_CFG_2::XcastGlort) will be used if the packet is a broadcast packet and the flood GloRT will be used otherwise.
    - A destination mask is derived form the port membership and the forwarding state where each bit represent 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 then gets 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, then 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, then 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, then the switch will refresh 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, then the switch will pick a free entry to store the new entry. The new entry will be stored as a dynamic entry, the GloRT will be set to the source GloRT (canonical source GloRT for port members of a LAG) and the trigger id will be set to the default trigger id configured in MA_TABLE_CFG_1::TrigIdDefault. If there are no free entries the switch will not learn the new address. Note that hardware will first pick a free entry that happened to match the MAC/VID that was previously used (but is now free) and will pick 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 will still be 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 the source GloRT is member of a LAG. If the source GloRT is part of a LAG, then 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 such 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 that 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, then the hardware will post the event in the TCN FIFO but will not perform the actual deletion or addition or move into the MA table automatically but refreshing the age will still be 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:

Address Field	Bit Range	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]

The HashRotation definition in MA_TABLE_CFG_1 specifies how this 16-bit hash value is reduced to the 12-bit index.

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 has been posted into the TCN FIFO. The purge command checks one entry of the MA Table every 20 frame handler cycles allowing frame lookups to be still performed at wire speed while the purge command is in effect. The time to do a complete purge is thus 16,384 x 20 / FH_CLOCK (0.87ms at 375MHZ).

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 (96b) - 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 (12b) - Index of associated MA_TABLE entry.
  - Set (3b) - Set of associated MA_TABLE entry.
  - EventType (3b) - Identifies the type of change.
- MA_TCN_PTR:

- Head (9b) - Entry in the MA_TCN_FIFO that software will read next. Updated by software. Read by hardware to determine when the FIFO overflows.
- Tail (9b) - 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 128b 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 that 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 will instead drop the notification event and raise the Overflow interrupt bit in MA_TCN_IP.

The TCN event types are listed in Table 14.

Table 14: 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, then the highest severity event will be 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 will contain 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

In order 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, will 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 will not be 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 will be 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 will resume. Note that if FlowControlEnable is 1, setting LearnedEventsThreshold or MaxAgedEventsThreshold to zero will have the effect of permanently disabling learning or aging.

Note that the TCN_FIFO entries of type “purge completed” (posted upon completion of the purge command) are not checked against the thresholds. So, the entry will be 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 will issue a purge command and wait to retrieve the “purge completed” from the TCN FIFO before issuing any similar command again, so 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:

- 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:

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 will be set and the interrupt will propagate 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

will only set bits from 0 to 1 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:
  - o New address security violation
  - o 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, then 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, then the address that caused the violation will be 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, then each frame will cause 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 listed below:

- 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 in order 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 supplicant is the side of the Ethernet link which would like access to the services behind the authenticator.
- Authenticator – The authenticator is 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.

Bali 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).

Bali supports also the following authentication port states:

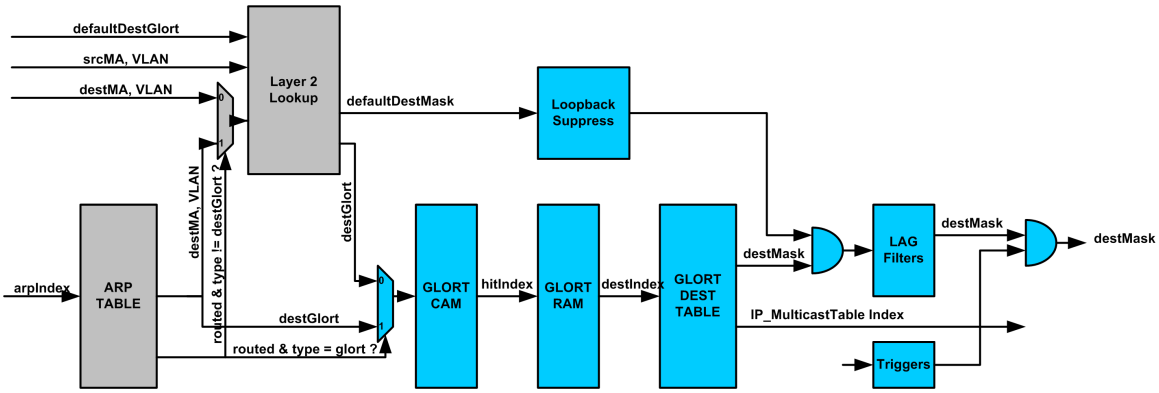
- Authorized – Non-EAPOL frames may be transmitted and received from the port
- Non-Authorized-IN – Frames may be transmitted from the port, however the port is non-authorized for receive, and RX frames are discarded, unless they are EAPOL frames. In this case, the port MAC security is on, learning is 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.

Chapter 9 - Port Mapping and Packet Replication

9.1 Overview of Port Mapping Unit

The GloRT lookup is shown in the Figure 33. The Port Mapping Unit will use 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 33: Glort Mapping Unit Block Diagram (ARP section is not applicable to the FM3000)



The port mapping requires both a destination GloRT and a default destination mask as inputs and will produce 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 Fulcrum 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 256x40 bits and there is a one-to-one correspondence between the GLORT CAM and GLORT RAM. If the search is successful, then 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.
- o IP_Multicast Index (14 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 64K possible GloRT values can map into the 4K table. The Strict field indicates whether the GLORT_DEST_TABLE index will be 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, then 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)
    - o 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, then the destination mask bit corresponding to the receive port is cleared. If the reflect bit is set, then the destination mask bit corresponding to the receive port is left untouched.
- Then the switch will check 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, then the source GloRT is checked to determine if it belongs to the same Link Aggregation Group as the source port. If yes, then 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 FocalPoint, 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 FocalPoint to implement link aggregation:

- Link Aggregation Glorts
- Filtering
- Pruning

Those are explained in the next sections.

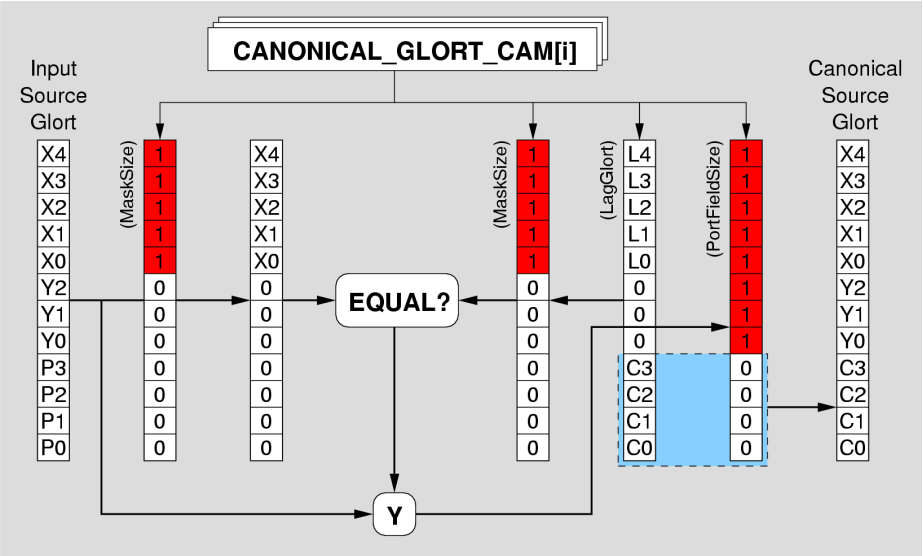
9.3.1 Link Aggregation Glorts

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

- LAG member GloRTs. These identify the individual physical port members of the LAG. A frame forwarded to such a GloRT will egress 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 FocalPoint devices. A frame identifying this GloRT destination will egress 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 in order 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. The diagram below illustrates its operation.

Figure 34: 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 that 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 source GloRT must be replaced with the canonical source GloRT so that the next switch doesn't 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.

The first method is called **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 are going to actually transmit the packet.

The second method is called **Pruning**. The GLORT_RAM includes a base pointer, a count and a hash function which is used to select which destination mask is going to be 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 done 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 one is used to balance traffic across the current chip. Note

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

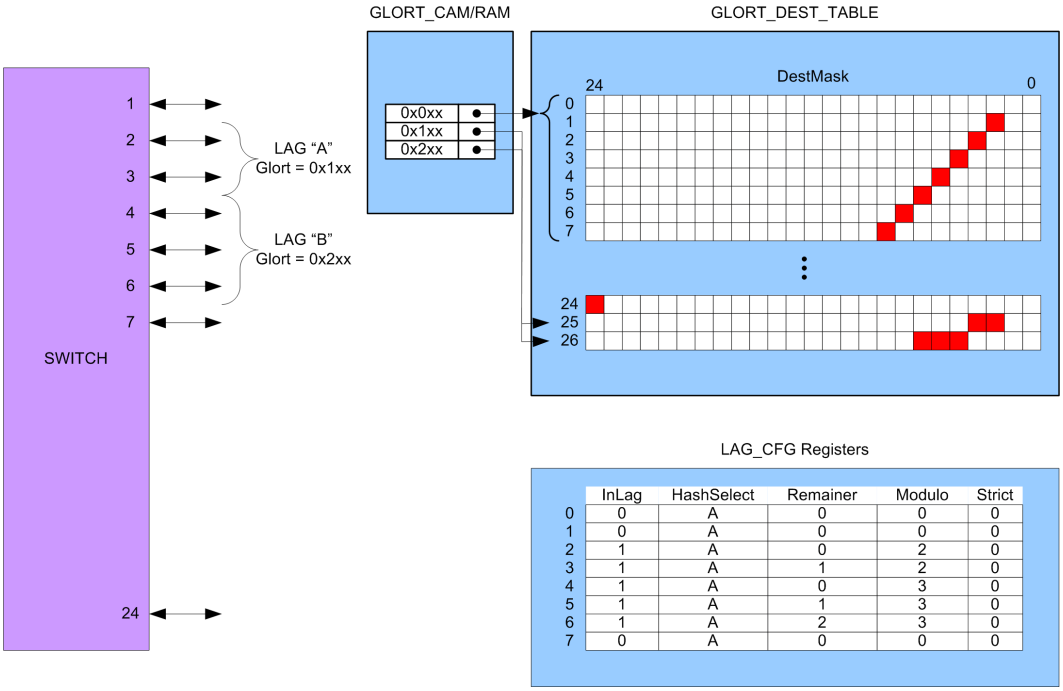
2. The GLORT_RAM defines if a hash function is used, which one (A or B), and which divider is selected (1 to 16). The index is computed from the base index and the remainder of the hash key computed divided by the modulo. This is the pruning step.
3. Then the per-port register set LAG_CFG[0..25] defines 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 thus

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

9.3.3 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.

Figure 35: LAG Filtering in a Single Switch System



The solution is to assign GloRTs 1 through 25 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

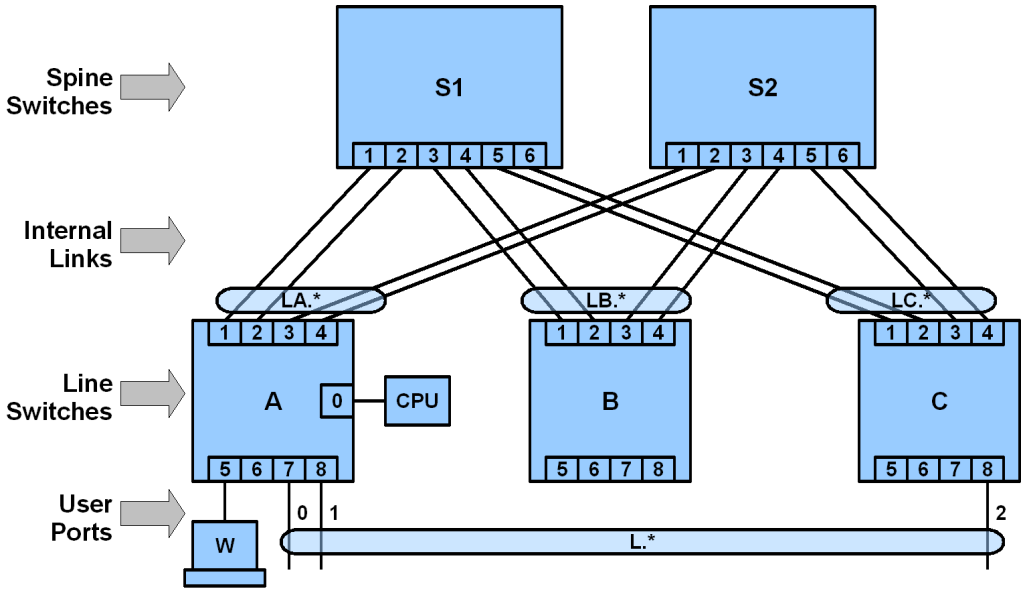
Port	Default Glort
4	0x201
5	0x202
6	0x203
7	7

For example, when a frame needs to be transmitted to GloRT “0x100”, the GLORT_CAM will match 0x1xx and the corresponding GLORT_RAM will be read which will point 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.

9.3.4 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 36: 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 S0 and S1 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

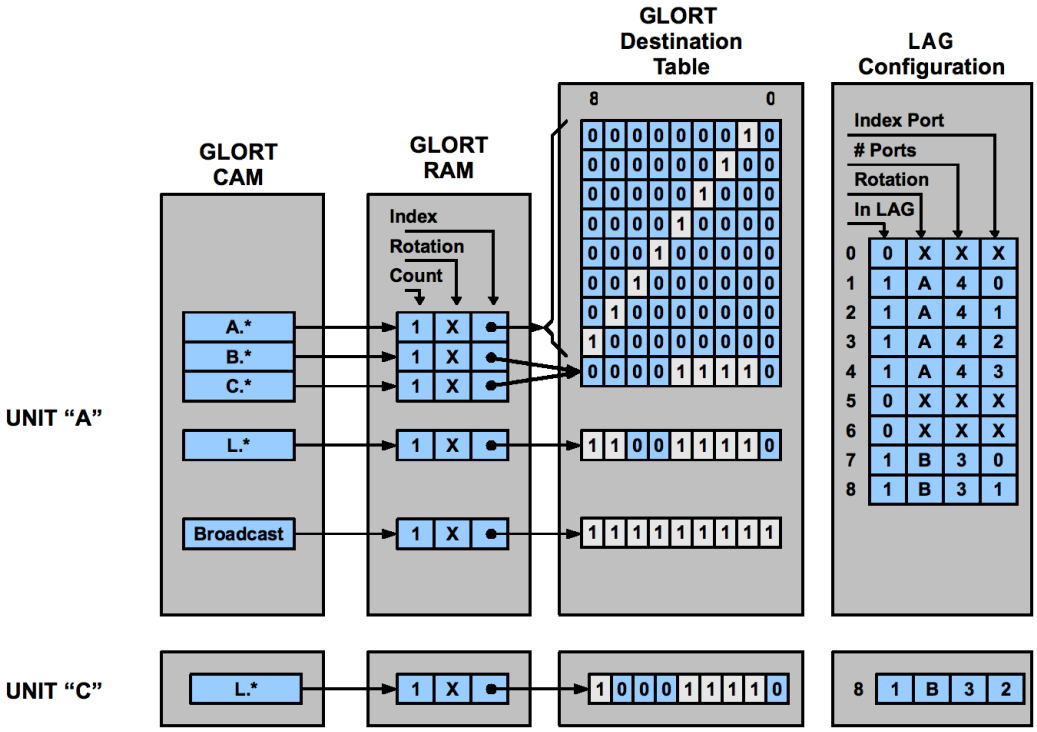
The requirement is to balance the traffic from workstation W to the different destinations.

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 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 will include the following entries:
  - A.*
  - B.*
  - C.*
  - L.*
  - Broadcast Glort
- Switch “A”: The corresponding GLORT_RAM will include 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.* will be hash 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.* will be hashed across ports 1,2,3,4,7,8.
  - Broadcast GloRT points to a single entry in the GLORT_DEST_MASK which will include all ports including CPU.
- Switch “C”: The corresponding GLORT_RAM will include 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 the next figure (only pertinent ports are shown).

Figure 37: 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 will be properly filtered out from L.0 and L.1 and will be hashed across LA.* and finally reach switch "C" and then L.2, but the packets going to L.0 and L.1 will not be filtered out from LA.* and will thus be sent twice, once on either L.0 or L.1 and once on the LA.* group. The extra packets sent on LA.* will be received by the switch C which will filter them out avoiding duplication on L.* but extra bandwidth was lost on the way to spine chips transporting packets that are filtered out anyway.

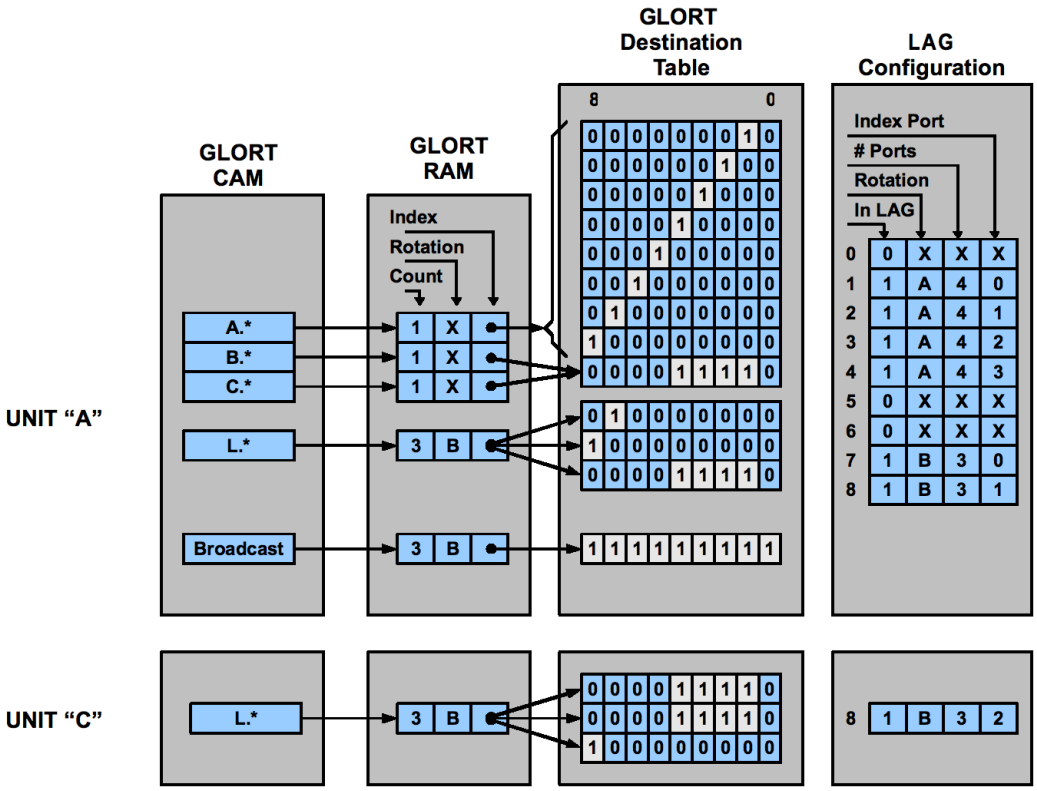
An alternative approach is to use pruning to avoid sending the packets going to L.2 to the inner links (LA.*). The set up becomes the following:

- 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.* will be changed:
  - L.* points to 3 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 3 entries while a second level of hashing (filtering) is used across the LA.* links.
  - BROADCAST points to 1 entry in the GLORT_DEST_MASK. The filterting is used to ensure that only one packet gets eventually transmitted to the L.* group.
- Switch "C": Similarly, the GLORT_RAM for L.* will be changed:
  - L.* points to 3 entries in the GLORT_DEST_MASK. 2 will be identical and will include LA.* only and one will include L.2. The hash function is used to distribute the traffic equally among the 3 entries.

- BROADCAST points to 1 entry in the GLORT_DEST_MASK. The broadcast will be transmitted on the spine switch and all other ports. The LAG will receive 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 wouldn't be superfluous for unitcast (as the L.* is already pruned) but must remain in place for broadcast.

This new configuration is shown in the next figure.

Figure 38: 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

FocalPoint 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

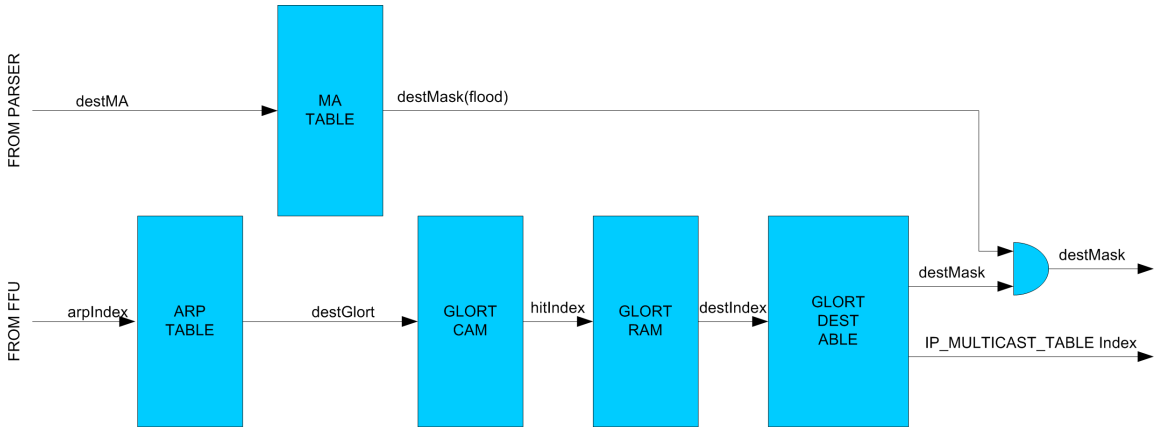
- wire-speed and low latency in all cases

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.

Note that 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 section, which covers only IP multicast.

Figure 39: 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 orgainizing 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.

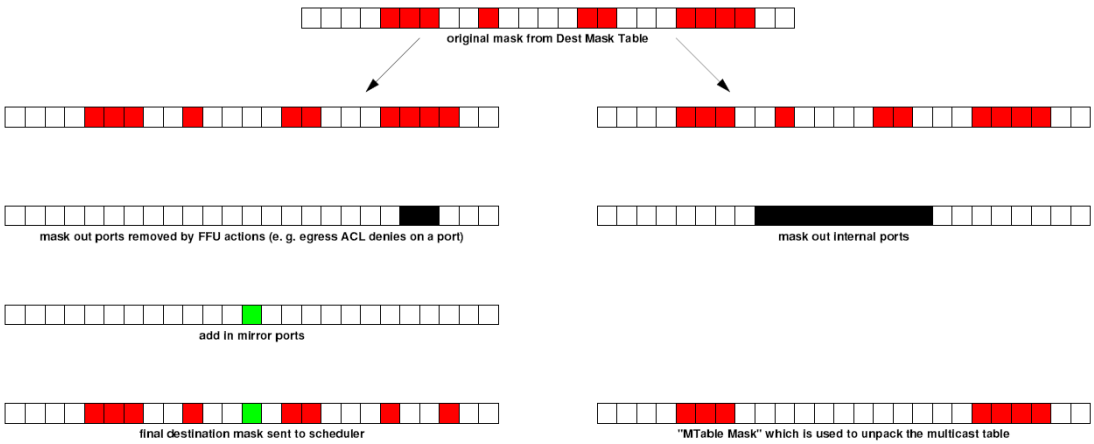
9.4.2 Usage of Destination Mask and IP Multicast Table

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 will have 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 will simply be 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 40: Derivation of the scheduler destination mask and the MTable mask.



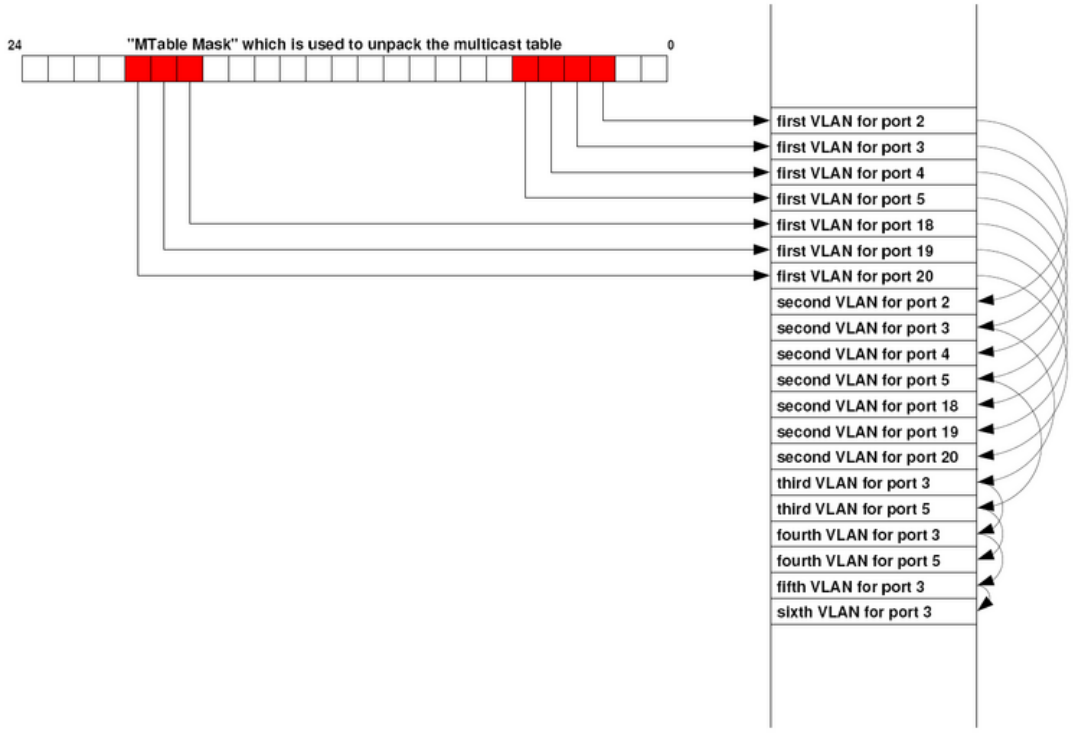
9.4.3 Configuring MTable

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)
    - o 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, then 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, then this entry is skipped and we just move on to the next entry in the linked list.
- skip (1 bit)
    - o 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 41: 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.

Chapter 10 - 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 in order 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.

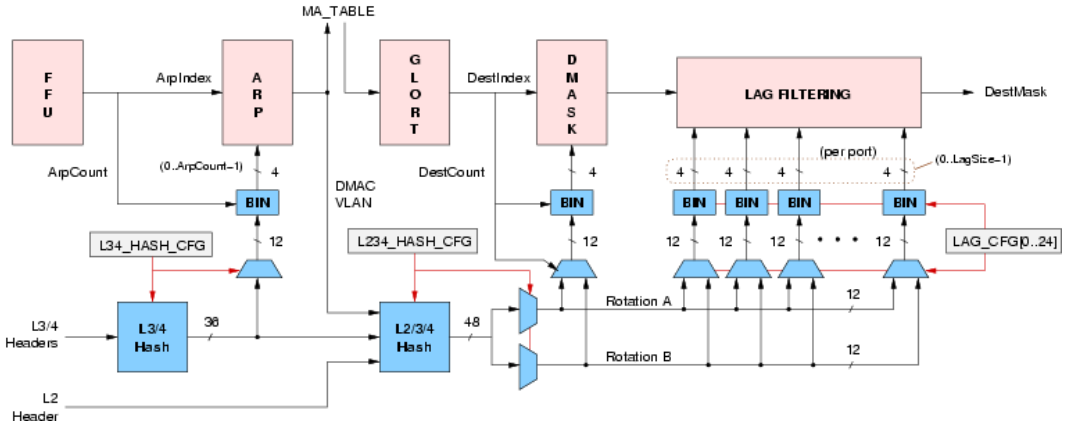
In a multi-chip switch, it may be desirable to load balance frames over different ECMP sets or LAGs in a statistically independent fashion. For example, the same hash function should not be used to distribute traffic over the second-layer links of a 3-tier fat tree as the first-layer links. Thus multiple 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 however, so in the interest of maintaining backwards compatibility with FM2000 devices, link aggregation continues to employ modulo binning. Both functions provide equally balanced hash binning.

Table 15: Binning Functions

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

The following diagram illustrates the frame hashing data flow in the Frame Processing Pipeline.



Note that 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	Byte 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 tables below.

Table 16: 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]*)

	UseSIP	Symmetric	Bytes 11:0	Bytes 15:12
IPv4/IPv6:	0	x	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.)

* - DIP field will be zero if UseDIP == 0.

Table 17: 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 will be zero if UseSIP == 0.

Table 18: 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 in order 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 19: Layer 3/4 Hashing Protocol Field

	UsePROT	Symmetric	Bytes 37
IPv4:	1	x	Protocol
IPv6:	1	x	NextHeader
IPv4/IPv6:	0	x	0

Table 20: Layer 3/4 Hashing Layer 4 Fields

	UseL4{SRC,DST}	Symmetric	Bytes {39:38, 41:40}
UseTCP==1 and Protocol is TCP UseUDP==1 and Protocol is UDP UsePROT1==1 and Protocol is PROT1	1	0	{SRC,DST}_PORT

	UseL4{SRC,DST}	Symmetric	Bytes {39:38, 41:40}
UsePROT2==1 and Protocol is PROT2	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 sixteen 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 (See L234_HASH_CFG in Frame Handler registers).

When the TahoeCompatible configuration bit is set to 1, hash rotation 0 (bits 11:0) is defined to give a Tahoe-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 eight bits to be backwards compatible with the Tahoe hash function when TahoeCompatible is 1. When TahoeCompatible 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, then the layer 2 header will be used for hashing.

UseL2ifIP	UseL34	Semantics
0	0	H234 will always be zero for IP frames; otherwise it will use 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 21: Layer 2 Hashing DMAC Field

UseDMAC	Symmetric	TahoeCompatible	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

Note: The DMAC used in the hash function is always the DMAC as received on ingress.

* - SMAC field will be zero if UseSMAC == 0.

** - Tahoe-compatible symmetric hash functions. EvenBits() sets even bits to DMAC^SMAC, odd bits to zero; vice versa for OddBits().

Table 22: Layer 2 Hashing SMAC Field

UseSMAC	Symmetric	TahoeCompatible	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

Note: The SMAC used in the hash function is always the SMAC as received on ingress.

* - DMAC field will be zero if UseDMAC == 0.

** - Tahoe-compatible symmetric hash functions. EvenBits() sets even bits to DMAC^SMAC, odd bits to zero; vice versa for OddBits().

Table 23: 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. In constrast, Tahoe (FM2000) includes the *first* EtherType in its frame hashing. Thus, if backwards compatibility with Tahoe is required, UseType should be set to zero.

Table 24: Layer 2 Hashing VID/VPRI Fields

EtherType	UseVPRI	UseVID	Bytes 14 [7:4]	Bytes 15:14 [11:0]
Either 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 Tahoe (FM2000) Compatibility

Multi-chip systems using both Tahoe and Bali 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. The following table summarizes the configuration requirements that are necessary in order to achieve consistent hashing between Bali and Tahoe devices.

Table 25: Summary of Hash Configuration Requirements for Tahoe Compatibility

Bali Requirement	Tahoe Requirement	Motivation
TahoeCompatible == 1 UseSMAC == 0, UseDMAC == 0	-	Select Tahoe Compatibility mode.
UseType == 0	IncludeTypeAndSource == 0	See note above about differences between Bali & Tahoe in their handling of EtherType.

Chapter 11 - 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, then 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 will be 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, then the entire trigger matches and is said to "fire." The programmable match conditions are as follows:

Condition	Frame Property	Trigger Parameter(s)
MatchSA	TrigID (6b) from the Layer 2 lookup of the frame's source MAC Address. This field will be 0 if there was no match in the MA Table.	SA_ID (6b)
MatchDA	TrigID (6b) from the Layer 2 lookup of the frame's destination MAC Address. This field will be 0 if there was no match in the MA Table.	DA_ID (6b)
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, then this condition evalutes to true.	-
MatchHitDA	Destination MAC Address match in the Layer 2 MA Table. If the address was found in the table, then this condition evaluates to true.	-
MatchHitSADA	Source *or* Destination MAC Address match in the Layer 2 MA Table. The match condition will be true if either (or both) addresses were found in the Layer 2 MA Table.	-
MatchVlan	TrigID (6b) from the EGRESS_VID_TABLE lookup.	VID_ID (6b)
MatchFFU	TrigID (8b) set from the FFU's SET_TRIG action. If no FFU rule hit specifies a SET_TRIG action, then this condition cannot match for	FFU_ID (8b), FFU_Mask (8b)

Condition	Frame Property	Trigger Parameter(s)
	any configured value of FFU_ID and FFU_Mask. The default FFU_ID produced by the FFU is 0.	
MatchSwitchPri	Associated switch priority (4b).	SwitchPri (4b)
MatchEtherType	First non-VLAN EtherType (16b).	EtherType (16b), EtherTypeMask (16b)
MatchDestGlort	Destination glort (16b) associated with the frame.	DestGlort (16b), DestGlortMask (16b)
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', then this trigger condition evaluates to true.	FrameClassMask (3b)
MatchSrcPort	Ingress port number (5b). If the corresponding source port bit in the configured SrcPortMask is set to '1', then this trigger condition evalutes to true.	SrcPortMask (25b)
MatchDestPort	Destination port mask (25b). If the corresponding destination port bit in the configured DestPortMask is set to '1', then this trigger condition evalutes to true.	DestPortMask (25b)
MatchRouted	Value of the mark_routed bit from the FFU ROUTE action (0 if no such action was applied to the frame).	RoutedMask (2b)
MatchFTYPE	ISL tag FTYPE field.	FtypeMask (4b)
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 will be 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 the Overview chapter's Action code table, 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 mirrorred as a result of FFU action. 42: IP unicast frame was logged to the CPU because it had an L2 multicast address	HandlerActionMask (64b)

Condition	Frame Property	Trigger Parameter(s)
	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.	
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 (1b), RandomThreshold (5b)
MatchByPrecedence	-	MatchByPrecedence (1b)

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 in order 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 will not match against any trigger. Only frames that are flooded or forwarded normally will 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 at the end of the Architecture Overview chapter.

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 1, then 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 0, then the trigger will fire unconditionally, assuming all other match conditions are satisfied. The following table illustrates this behavior:

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	

Trigger Number	MatchByPrecedence	Matches?	Fires?	Explanation
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

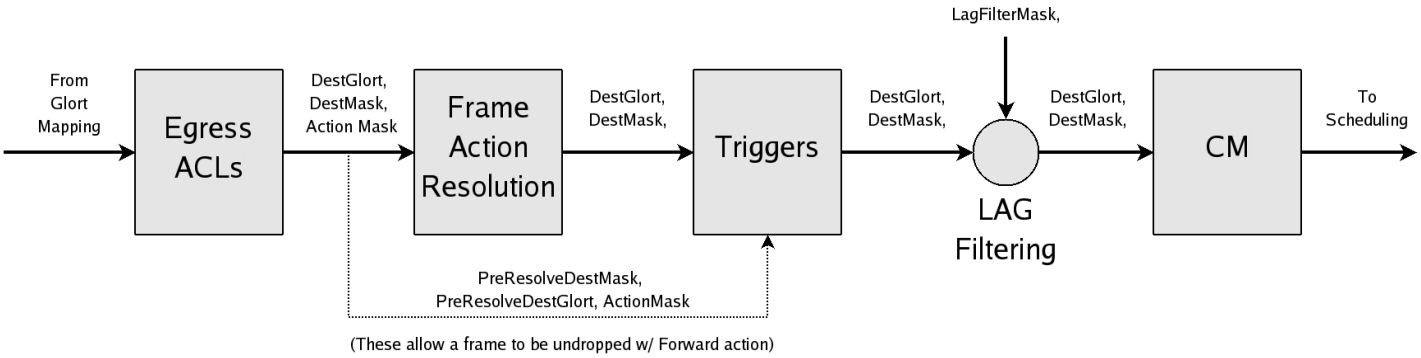
Each trigger specifies a list of actions to apply to the frame. Nine sets of mutually exclusive actions are defined:

Trigger Action Set	Trigger Action	Associated Configuration	Description
ForwardingAction	0 - Leave as-is	-	No change in forwarding.
	1 - Forward	NewDestGlort (16b), NewDestGlortMask (16b)	Forward the frame with a new destination glort. Destination glort bits will be overridden from

Trigger Action Set	Trigger Action	Associated Configuration	Description
			NewDestGlort whose corresponding NewDestGlortMask bits are '1'. This action can also be used to undo a drop decision from earlier in the frame processing pipeline (see “Using Triggers to Undrop Frames” below.)
	2 - Redirect	NewDestMask (25b), FilterNewDestMask (1b), NewDestGlort (16b), NewDestGlortMask (16b)	Override destination mask to NewDestMask. If FilterNewDestMask is set to '0', then link aggregation filtering will not be applied to this new destination mask. If FilterNewDestMask is set to '1', then LAG filtering will be applied, even if the frame's FTYPE equals 2 or 3. NewDestGlort is applied as for the Forward case.
	3 - Drop	DropMask (25b)	Do not forward to any ports i with DropMask[i]==1.
TrapAction	0 - Leave as-is	-	No change in CPU trapping.
	1 - Trap	CpuTruncate (1b)	Trap to CPU (overrides ForwardingAction). Truncation of this frame will be enabled if CpuTruncate is set to 1. In order for the trapped frame to be truncated, the corresponding CpuTruncationEnable in the egress port must also be set.
	2 - Log	CpuTruncate (1b)	Log to CPU with optional truncation. When truncation is enabled, only the copy of the frame sent to the CPU will be 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 (5b), MirrorGlort (16b), MirrorTruncate (1b)	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 1. 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 (4b)	Associate NewSwitchPri with the frame, for purposes of congestion management and egress queueing. Note that this has no effect on the FFU's
VlanAction	0 - Leave as-is	-	No change in VLAN.
	1 - Reassign egress VLAN.	NewVlan (12b)	Override egress VLAN to the specified value. Note: this only affects L2-switched and L3-unicast frames; IP multicast replication, if applicable, will still produce frames with the same egress VLANs as if this action were not specified.

Trigger Action Set	Trigger Action	Associated Configuration	Description
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 (4b)	Assign this frame to one of 16 Trigger Rate Limiters. A drop mask will be applied to the frame if the specified rate limiter's bandwith limit is exceeded.

As illustrated in the diagram below, triggers are applied after Glort mapping, egress ACLs, and action resolution, but before LAG filtering and congestion management.



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 0, bypasses LAG filtering and allows frames to be directed to specific physical ports.
- Any assigned destination glorts and port masks will be 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 the Architectural Overview section), 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 had been dropped the destination mask and glort prior to the Frame Action Resolution step will be restored. The frame's action handling code will also be 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 will be changed and only one drop mask will be applied to a given frame. If the rate limiter is out of profile, then the frame's destination mask will be 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 the Congestion Management chapter. 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 that 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 will be 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 above.

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 will be changed and only one drop mask will be applied to a given frame. In this case, the trigger with the lowest ID number gets 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 will be 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. For more information see Section 3.5.

Each trigger is associated with a bit in one of the TRIGGER_IP registers. Whenever a trigger fires, the corresponding IP bit is set.

Chapter 12 - QoS & Congestion Management

12.1 Overview

FocalPoint switches include the following traffic management features:

- Traffic Classification
- Traffic Policing
- Traffic Shaping
- Class Based Flow Control
- Congestion Management
- 802.1q
- DiffServ

and 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.
6. Congestion Notification
    - Notifies upstream nodes of presence of congestion
7. Scheduling
    - Sending frame according to rate and priority

12.2 Processing of Frames with Trailing Errors

FocalPoint 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 it is an erroneous frame.
- Memory Management
    - The role of memory management is to check that the memory usage is withing the boundaries defined by the user. A frame with a CRC error will still use some amount of memory and it must be accounted for as are non-errored frames. Note that frames with invalid CRCs that are not yet transmitted will quickly be erased from memory and will 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, then the erroneous frame will still be accounted for as a frame being transmitted and will be included in the traffic shaping computations. If the frame was never transmitted (dropped before scheduling occurred), then 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. The ingress rate limiters do not include erroneous frames in the rate computation in order to avoid generating PAUSE frames on invalid traffic.
- 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 the the IP header to carry priority information about the frame. This field structure is shown here:

0	1	2	3	4	5	6	7
DSCP						ECN	

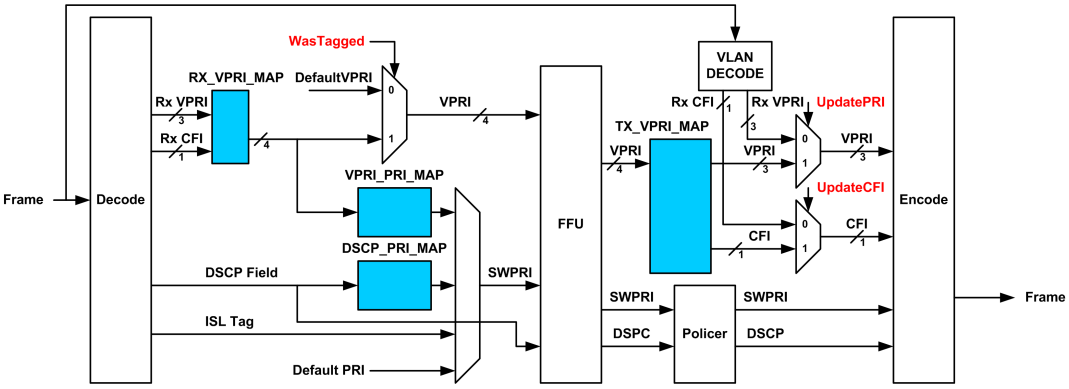
The Differentiated Services Code Point (DSCP) defines the class of service while the Explicit Congestion Notification (ECN) reports congestion in the forwarding direction. FocalPoint supports traffic conditioning as required by DiffServ which includes the following: meter, marker, shaper and dropper.

- The 'meter' and 'marker' are detailed together in the policing section.
- The 'shaper' is detailed in the Scheduler chapter
- The 'dropper' is detailed in the Policing and Congestion Management of this chapter.

12.4 Processing Priority

Figure 42 illustrates the priority handling in the switch.

Figure 42: Priority Processing Block Diagram



Each frame may include one or more of the following fields:

- VLAN priority
- DSCP/TOS
- Fulcrum 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 listed below:

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 didn't 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 then is 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 will 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. They must agree on the internal 4-bit VPRI, however.

Step 2: Define DSCP

The DSCP is normally taken from the IP header unless the useDefaultDSCP bit of PORT_CFG_1 is 1, 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 4 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, then 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 1, the DSCP priority is prefered 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 will eventually be put into the outgoing VLAN tag, ISL tag, and DSCP field, if those tags or fields exist.

Via FFU

The FFU can inspect all 3 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 Chapter 6, Frame Filtering and Forwarding Unit 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 4 banks of counters or policers. If a bank is configured to police packets, it will produce 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 will optionally modify 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 down grading, 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 down grade back to the same value are originally red, ingress DSCP values that down grade to red values are originally yellow and ingress DSCP values that down grade 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 gets translated into a new 4-bit VPRI using the TX_PRI_MAP table and gets optionally encoded for transmission into the transmitted packet under the following conditions:

- If the packet was received tagged and needs to be transmitted untagged, then the VLAN tag is deleted and the update VPRI is not used.
- If the packet was received tagged and needs to be transmitted tagged, then the VLAN tag is update with the new VID and the new VPRI is optionally updated (see MAC_CFG_2 register which controls if the 3-bit priority and/or CFI/DEI gets updated).
- If the packet was received untagged and needs to be tagged, then a VLAN tag is added using the new VID and new VPRI
- If the packet was received untagged and needs to leave untagged, then 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.

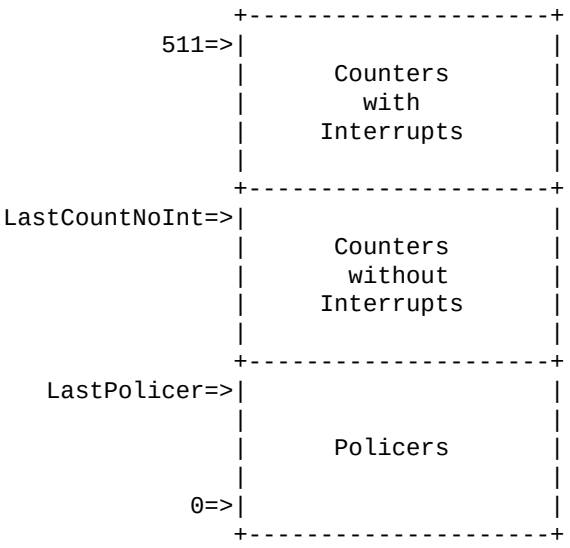
The flag “UpdatePri” is set to 1 if a VLAN tag is added, or if an existing VLAN is updated and the option MAC_CFG_1::EnableVPriUpdate is set.

The flag “UpdateCfi” is set to 1 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 Chapter 8: Layer 2 Lookup.

12.7 Policing

FocalPoint includes a policing unit which monitors the incoming traffic rate and can change the priority if needed. The policing unit includes 4 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 4 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.

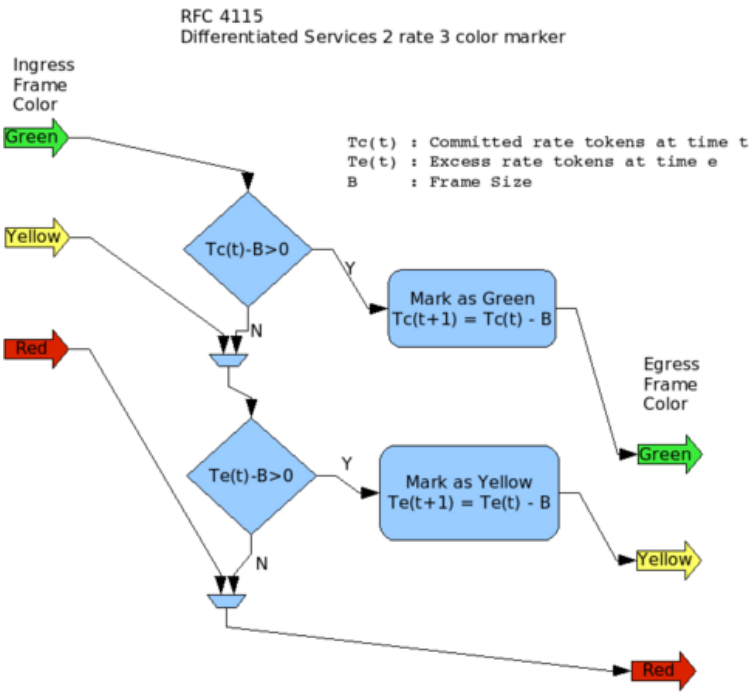


The counters without interrupts consist of two 64-bit counters; one counting bytes while the second counts frames. The counters with interrupts are identical but can optionally generate an interrupt, provided the interrupt mask for this bank is reset.

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 will use the "excess" TB's tokens.

The policer will use either the switch priority or DSCP to retrieve a frame color (green, yellow, red) and will use 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, then current color must be green, if there is only one level down, then current color must be yellow, if there is no level down, then 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 43.

Figure 43: Differentiated Services Color Marking



The rules for combining the results from the four banks are the following:

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 then 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 will be the lowest color of the 4 banks (most reddish).
4. The frame's egress color is only permitted to be equally or more reddish than the ingress color. e.g. 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.

Rules (1-4) are applied to get an egress DSCP and an egress Switch Priority color, then rules (5-6) are applied independently for DSCP and Switch Priority.

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. e.g. if the egress color from rules (1)-(4) is Green but the ingress color was Yellow, the egress value will be unchanged and stay 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 will exit 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

FocalPoint 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, FocalPoint can partition the traffic into different shared memory partitions (SMP) using the traffic class to which the frame belongs (note that the traffic class is obtained through the SWITCH_TO_CLASS register set which associates a traffic class to each switch priority) and will monitor 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.

The switch contains 2 SMPs. The CM_SMP_MEMBERSHIP register is used to map the 8 traffic classes into SMPs using the following assignment:

- 0: Not part of any SMP
- 1: SMP 0
- 2: SMP 1

Any other assignment will cause 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 3 and Figure 4.

Figure 44: Shared Memory Partitioning - Receive

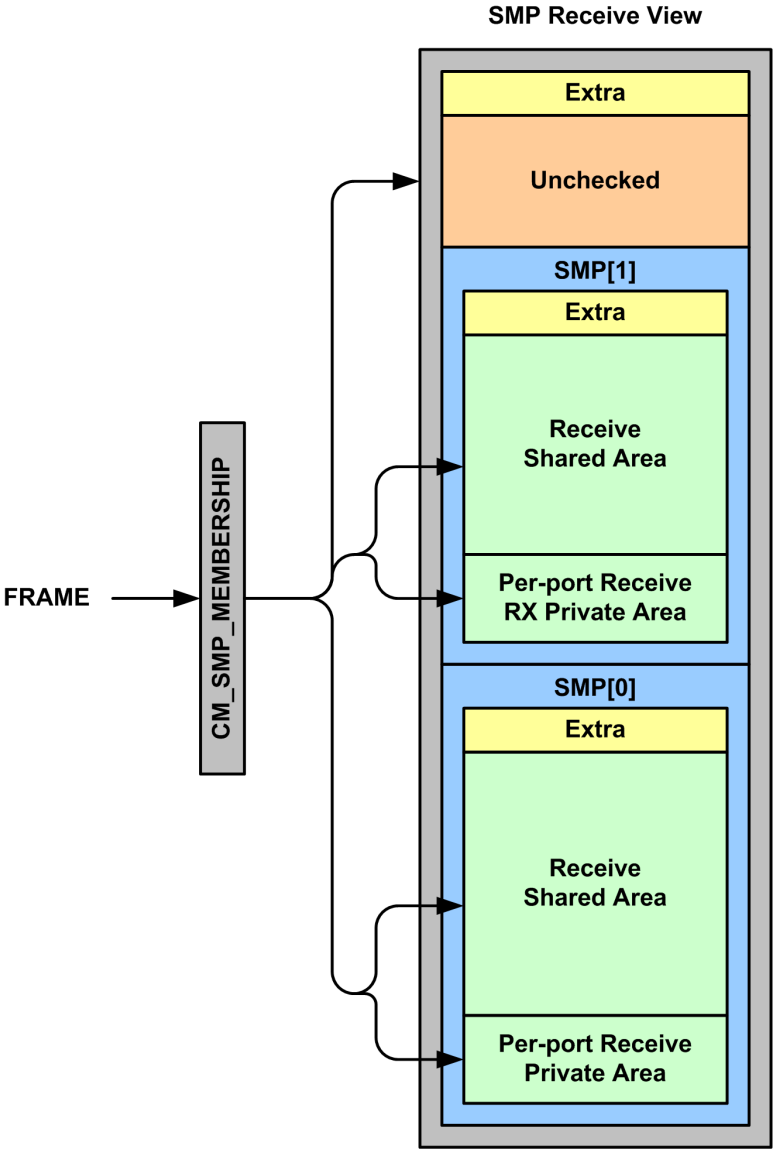
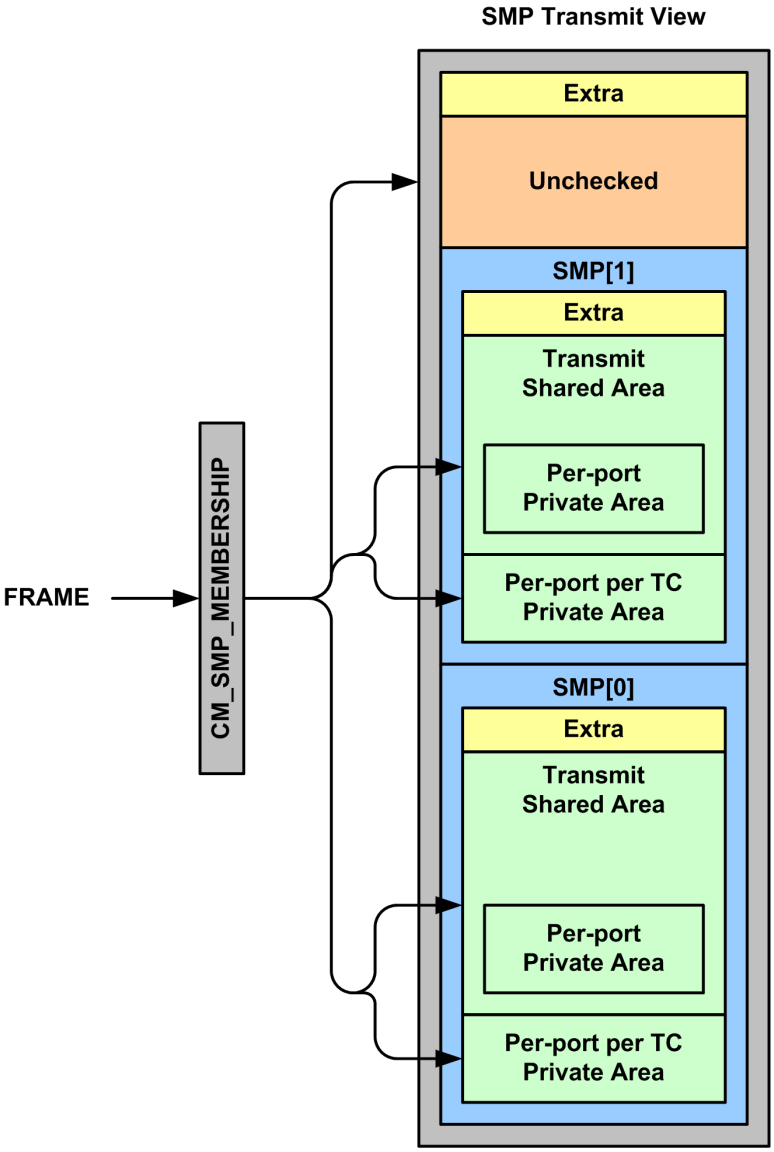


Figure 45: Share Memory Partitioning - Transmit



As shown, the incoming frame will use the CM_SMP_MEMBERSHIP register to determine which SMP to use (SMP 0, SMP 1 or unchecked). Once the SMP is known, the frame will be checked against the different receive and transmit watermarks and then, if accepted, will be queued for transmission. The frame will be 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 will be checked and accounted once on the receive side and will be 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. So, 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 don't 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 will get 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.

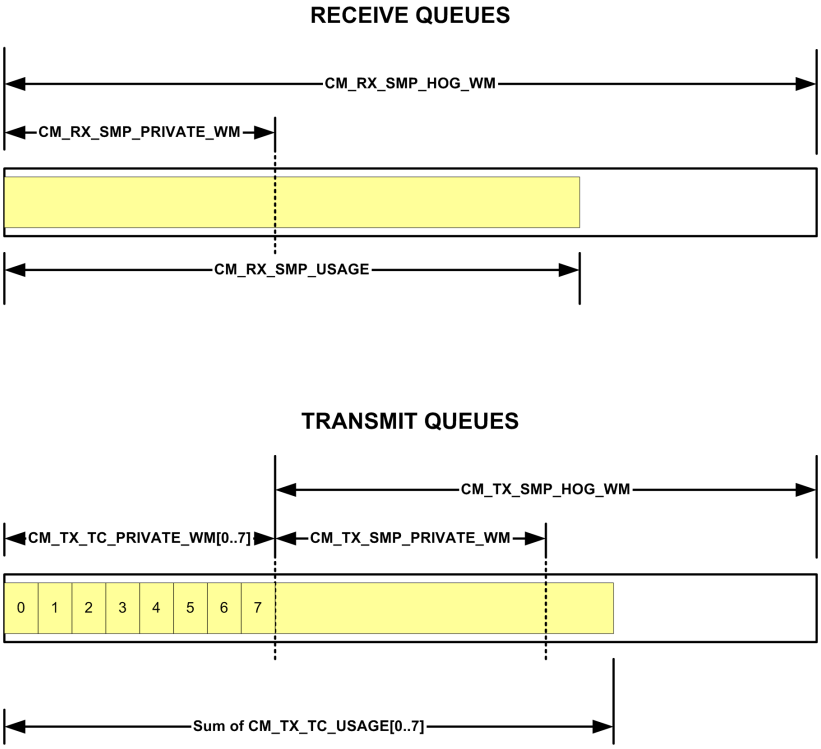
The switch offers the following watermarks (watermarks are defined in number of 512-byte segments and defines the threshold above which the action is taken):

- CM_GLOBAL_WM (1 x 16-bit)
  - 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-bit)
  - 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-bit)
  - Defines the size of the per-port per-SMP receiver private area
- CM_RX_SMP_HOG_WM (25 x 2 x 16-bit)
  - Defines the upper-bound limit to the receive queue size per-port per SMP.
- CM_TX_SMP_PRIVATE_WM (25 x 2 x 16-bit)
  - 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-bit)
  - Defines the size the per-port per-traffic-class transmitter private area.
- CM_TX_SMP_HOG_WM (25 x 4 x 16-bit)
  - 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-bit)
  - o Maps the traffic classes associated with the frame to one of the 4 hog watermarks associated to each port.
- CM_RX_SMP_PAUSE_WM (25 x 2 x 16-bit)
  - o 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-bit)
  - o Defines the PAUSE ON and PAUSE OFF per-smp trip points to send PAUSE frames to an ingress port.

The next figure shows the different watermarks and usage counters per port.

Figure 46: Receive and Transmit Queues Watermarks



Generally speaking a frame gets 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 that some memory management registers can't 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
4. - 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”
5. 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.
6. Repeat steps 3-4 until the CM_GLOBAL_USAGE is 0.
7. Change watermarks as desired.
8. Restore frame timeout to original value.
9. Restore egress rate limiters.
10. Restore PORT_CFG_2.

12.9 Flow Control and Rate Limiters

FocalPoint supports two types of PAUSE frames:

- IEEE 802.3 PAUSE frames
  - o PAUSE frames which contains a single pause interval applicable to all traffic classes
- CISCO Class Based PAUSE frames
  - o 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:
  - o There are 2 rate limiters, one per SMP. The rate limiter monitors the incoming traffic rate for each SMP and will start transmitting PAUSE frames if a programmable rate threshold is exceeded and will 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 conditons evaluate to pause on. A pause_off is sent if all of the three pause conditons evaluate to pause_off.

Also, FocalPoint has the capacity to react to reception of PAUSE frames. If it is a Cisco class based PAUSE frame, then the PAUSE frame will indicate which class to pause and for how long. If it is an IEEE frame, then 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, FocalPoint also supports transmit rate limiters which are detailed in Chapter 13: Egress Scheduling and Shaping..

The following registers control the pause and rate limiter behavior:

- CM_RX_SMP_PAUSE_WM (25 x 2 x 32-bit):
  - Defines the limit in number of segments that each RX can used per SMP before starting or stopping sending PAUSE frames. The register contains an ON limit (pause is activated when the number of segments is above this limit) and an OFF (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-bit)
  - 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 (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 (see Note 1).
      - Process incoming RX PAUSE frames as IEEE 802.3 PAUSE frames.
      - Process incoming RX PAUSE frames as Class-based (CISCO-defined) PAUSE frames.

    Note 1: 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 FocalPoint 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.
    - 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 bit)
  - 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 8 decimators allows the switch to support up to 8 different data rates.
    - The value stored in the decimator is equal to 512 * FH_CLK / PORT_SPEED /25 and shall be expressed as (K+1) / (1 + M/(N+1)).
    - Example 1, link at 10G with FH at 375MHZ:
      - Q = 512 * 375M / 10G / 25 = 0.768 = 1/1.302 = ~1/(1+3/10)
      - K = 0
      - M = 3
      - N = 9
    - Example 2, link at 1G with FH at 360MHZ
      - Q = 512 * 360M / 1G / 25 = 7.37 = ~8 / (1 + 1/12)
      - K = 7
      - M = 1
      - N = 11
- RX_RATE_LIM_USAGE (25 x 2 x 32-bit)
  - 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 cycle 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-bit)
  - 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 375MHZ a rate of 1Gbps is equivalent to:
        - bytes/period = 1G * 25 / 375M / 8 = 8.333 = 8 + 85/256
        - units = 8
        - fraction = 85
      - The minimum rate supported for 375MHZ is:
        - units = 0
        - fraction = 1
        - rate = (1/256) * 375M * 8 / 25 = 0.46Mbps
- 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 Chapter 13: Egress Scheduling and Shaping.
- 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 375MHZ a rate of 1Gbps is equivalent to:
        - bytes/period = 1G * 25 / 375M / 8 = 8.333 = 8 + 85/256
        - units = 8

    - fraction = 85
  - The minimum rate supported for 375MHZ is:
    - units = 0
    - fraction = 1
    - rate = (1/256) * 375M * 8 / 25 = 0.46Mbps

The receiver flow control algorithm is the following:

1. At beginning of a frame:

   - Check CM_XXX_USAGE against watermarks (as detailed in previous section) and forward or discard frame accordingly
   - 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:

   - Update the CM_XXX_USAGE registers.
   - The switch priority associated with that frame is used to index the CM_SMP_MEMBERSHIP and retrieve the SMP associated with that frame.
   - The RX_RATE_LIM_USAGE is decremented by the size of the frame regardless if the frame has a valid CRC or not (the counter can go negative).
   - 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.
   - 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, then the rate limiter is de-activated and the pause is canceled if the PAUSE off watermarks have not been reached yet.
   c. Resend PAUSE fames periodically as long as either a rate limiter or congestion is present.
4. When the frame transmission is completed:

   - Update the CM_XXX_USAGE registers.
   - The switch priority associated with that frame is used to index the CM_SMP_MEMBERSHIP and retrieve the SMP associated with that frame.
   - 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

FocalPoint 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 Vitual-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 gets 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 bit)
  - 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 will get replaced by the PORT_CFG_ISL[0]::srcGlort.
    - Queue unit size (power of 2).
- CN_SAMPLE_CFG (1 x 32 bit)
  - 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 equilibrum threshold (used for FCN).

12.10.1 Virtual Output Queues Congestion Notifications

FocalPoint 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:

```
        |         |
        Q_hi      Q_lo
```

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 below.

```
MSB                                                LSB
+--------------------------------------------------+
|    DA = CN_VCN_DMAC_x  Statically cfg. MAC       | <- (1)
+--------------------------------------------------+
|                        |                         |
+--------------------------------------------------+
|  SA = CN_FRAME_CFG_x  MAC of congestion point    |
+--------------------------------------------------+
|                   802.1Q Tag                     | <- (2)
+--------------------------------------------------+
|   EtherType = VCN     |      Reserved            | <- (3)
+--------------------------------------------------+
|                       ID                         | <- (4)
+--------------------------------------------------+
|  Traffic Class Mask   |      Reserved            | <- (5)
+--------------------------------------------------+
|        Q_len[TC0]     |         Q_lo[TC0]        | <- (6)
+--------------------------------------------------+
|        Q_len[TC1]     |         Q_lo[TC1]        |
+--------------------------------------------------+
|        Q_len[TC2]     |         Q_lo[TC2]        |
+--------------------------------------------------+
|        Q_len[TC3]     |         Q_lo[TC3]        |
+--------------------------------------------------+
|        Q_len[TC4]     |         Q_lo[TC4]        |
+--------------------------------------------------+
|        Q_len[TC5]     |         Q_lo[TC5]        |
+--------------------------------------------------+
|        Q_len[TC6]     |         Q_lo[TC6]        |
+--------------------------------------------------+
|        Q_len[TC7]     |         Q_lo[TC7]        |
+--------------------------------------------------+
|                     FCS                          |
+--------------------------------------------------+
```

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

4. Q_len[TCi] = CM_TX_TC_USAGE[TCi] * SEGMENT_SIZE >> CN_GLOBAL_CFG_2.CNUnitSize. This value can be used or not depending on how smart intelligent subscriber cards are.

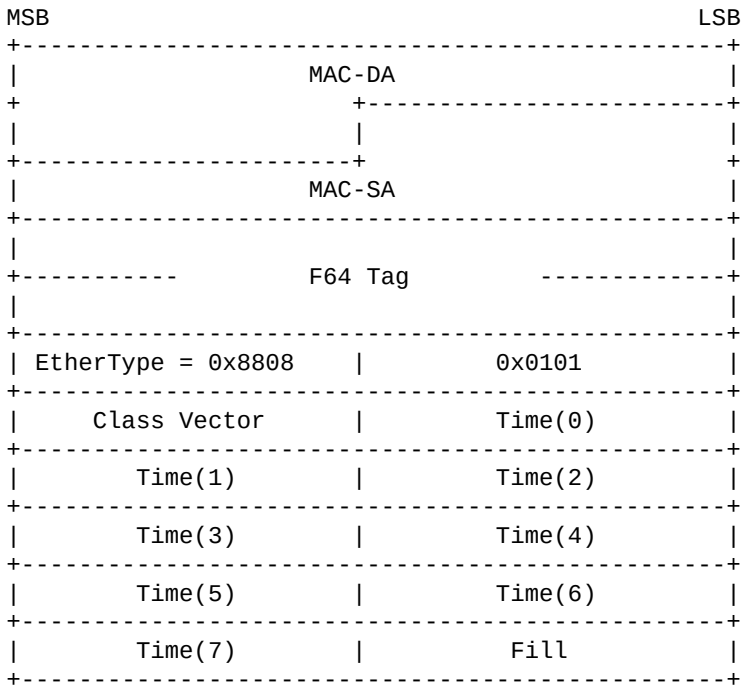
12.10.1.1 VCN Frame Reaction

Note that 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 FocalPoint switch in tandem with a class-based PAUSE capable NPU. In this mode the FocalPoint ingress ports are programmed to pass the VCN frames on to an externa 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 will be 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 8 egress queues.

The Class-based Pause frame format is shown here:



```
|               24 octets of Fill                |
+------------------------------------------------+

|                      FCS                       |
+------------------------------------------------+
```

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.

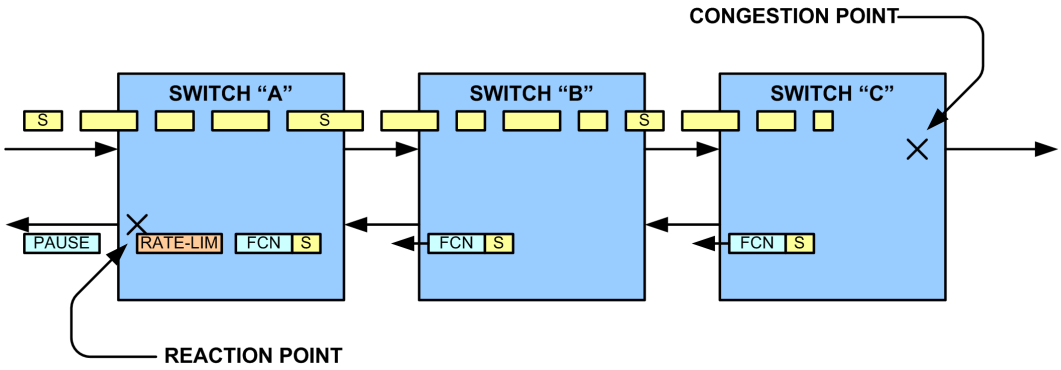
12.10.3 Fractional Congestion Notifications

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 Fulcrum Microsystems 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), then 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, then the switch will activate 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, then the switch will simply forward the frame to the port without taking any further action.
    - The rate limiter operating parameters are the one predefined for the associated SMP and port.

This is illustrated in the next figure. 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 will generate a PAUSE frame back to the source.

Figure 47: Congestion Notification



The FCN frame format is shown here:

```
MSB                                                 LSB
+---------------------------------------------------+
|        DA = Source MAC of sampled frame           |
+                       +---------------------------+
|                       |                           |
+-----------------------+                           +
|        SA = MAC of congestion point               |
+---------------------------------------------------+
|                                                   |
+-----------         F64 Tag         --------------+
|                                                   |
+---------------------------------------------------+
| EtherType = FCN       |        Reserved           |
+---------------------------------------------------+
|                                                   |
+----------           CPID              ----------+
|                                                   |
+---------------------------------------------------+
|        Q_off          |         Q_delta           |
+---------------------------------------------------+
|                     padding                       |
+---------------------------------------------------+
|                       FCS                         |
+---------------------------------------------------+
```

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) in units defined by CN_GLOBAL_CFG_2::CNUnitSize.

    - Staturates 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 in units defined by CN_GLOBAL_CFG_2.CNUnitSize.

    - Staturates at 2 * CN_SMP_CFG.EQ and – 2 * CN_SMP_CFG.EQ.

The format of the CPID is shown below:

```
+-------------------------------------------------+
|    Fulcrum CPID 64 bits                         |
+-------------------------------------------------+
|Switch ID | Egress Port | Ingress Port | SMP ID   |
|  48 bits | 6 bits      | 6 bits       | 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 FocalPoint 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 Fulcrum Microsystems for details.

The BCN includes the following functions:

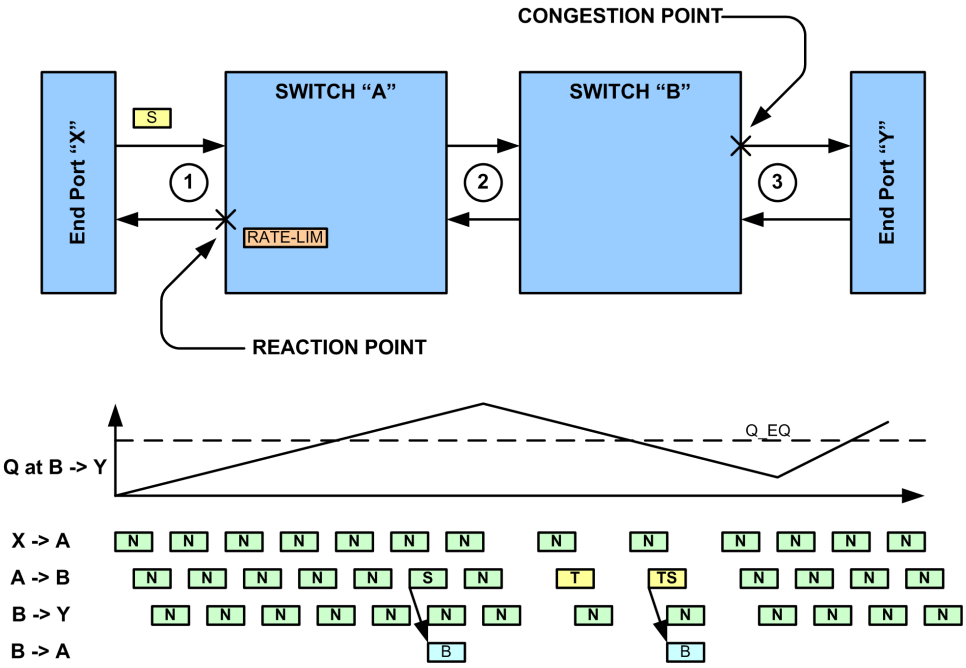
- CPID Cache
- Support reception, deletion and transmission of rate limiter tags.
- Detection of rate limiter tags at congestion point and transmission of BCN to increase bandwidth.
- Computation of the rate limiter parameters (accrual rate and capacity) from level of congestion

The extensions are enabled when the CN_GLOBAL_CFG::Mode is set to 3.

The process used is the following:

- 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), then the switch potentially samples this frame (using the sampling probability defined in the CN_PORT_CFG) and generate a BCN frame back to the source.
  - If the queue size is lower than a configured threshold (CN_SMP_CFG::EQ), then the switch will check if this frame carries a valid RLT tag which identified this congestion point, if yes, then the switch samples the frame (using the sampling priority defined in the CN_PORT_CFG) and generate a BCN frame back to the source.
  - The BCN frame generated will have a DA equal to the SA of the sampled frame and will contain information about the source point, the status of the queue as well as selected fields from the sampled frame. The BCN frame is a layer 2 frame which will be switched back to the source up to the reaction point where it will trig a rate limiter. The BCN frame are not layer 3 and cannot be routed. If the DA is equal to a routing point, then 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 or forward them to the egress port or delete them. If they are trapped, then the switch will activate 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, then the switch will simply forward the frame to the port without taking any further action.
    - 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, then all future frame coming from this port will 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.

The next figure illustrates the concept.



The BCN frame format is shown here:

```
MSB                                                 LSB
+----------------Section (A)-----------------------+
|             DA = SA of sampled frame              |
+                      +--------------------------+
|                      |                          |
+----------------------+                          +
|             SA = MAC of congestion point         |
+--------------------------------------------------+
.                                                  .
.         802.1Q Tag    or    F64 Tag              . <---(1)
.                                                  .
+----------------------+--------------------------+
| EtherType = BCN      |       Reserved           |
+----------------------+--------------------------+
|                                                  |
+-----------          CPID             -----------+
|                                                  |
+----------------------+--------------------------+
|        Q_off         |       Q_delta            |
```

```
 +--------following fields from sampled frame------+
 |                       DA                        |
 +                       +-------------------------+
 |                       |                         |
 +-----------------------+                         +
 |                       SA                        |
 +-----------------------+-------------------------+
 |     802.1Q (if sampled frame had a VLAN)        | <---(2)
 +-----------------------+-------------------------+
 | EtherType             |                         |
 +-----------------------+                         +
 .                                                 .
 .                  zero padding                   .
 .                                                 .
 +-----------------------+-------------------------+
 |                      FCS                        |
 +-------------------------------------------------+
(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 Fulcrum 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) in units defined by CN_GLOBAL_CFG_2.CNUnitSize.
  - Staturates 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 in units defined by CN_GLOBAL_CFG_2.CNUnitSize.
  - Staturates 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 below:

```
+-------------------------------------------------+
|    Fulcrum CPID 64 bits                         |
+-------------------------------------------------+
|Switch ID | Egress Port | Ingress Port | SMP ID   |
|  48 bits | 6 bits      | 6 bits       | 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: Bali 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:

```
+----------------+
|       DA       | (6 BYTES)
+----------------+
|       SA       | (6 BYTES)
+----------------+
|    VLAN Tags   | (N x 4 BYTES)
+----------------+
|     RLT Tag    | (12 BYTES)
+----------------+
|    EtherType   | (6 BYTES)
+----------------+
|                |
|                |
| Packet Payload | (N BYTES)
|                |
```

```
|                |
+----------------+
|       CRC      | (4 BYTES)
+----------------+
```

The RLT tag is 8 bytes long and is show here:

```
+----------------+
|     RLT Tag    | (2 BYTES)
+----------------+
|        0       | (2 BYTES)
+----------------+
|                |
|      CPID      | (8 BYTES)
|                |
+----------------+
```

12.11.1 BCN Congestion Point

12.11.1.1 Sampling

Each frame passing through a egress port and SMP pair will contribute to a byte counter in the Frame Handler. When the counter decrements to zero due to outgoing frames, a flag will be set in the Frame Handler to indicate that the next frame will be sampled for BCN. The counter will then be 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 will be generated if CN_SMP_CFG::EQ or CN_SMP_CFG:SC has been exceeded.
- a BCN frame will be 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 will be 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 fowarding 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 will need to be activated if the CPID in the BCN frame is not in the CPID cache otherwise the current state for the rate limiter will be updated.

If a rate limiter needs to be activated, the CPID in the BCN frame will be written to an entry CPID cache (CN_CPID_TABLE[i], CPID_i) and CN_RATE_LIM_CPID will be written with the index (i) into the CPID cache.

If the CPID is already in the CPID cache, then the rate limiter's parameters will be updated according to the following algorithm.

```
function compute_Fb(frame f) =
  if (CN_GLOBAL_CFG_2.fbQDelta) {
    Fb = f.Q_off   * 2 ^ (CN_GLOBAL_CFG_2.CNUnitSize - 9) -
       ((f.Q_delta * 2 ^ (CN_GLOBAL_CFG_2.CNUnitSize - 9)) << CN_FB_CFG.BCN_W)
  } else {
    Fb = f.Q_off   * 2 ^ (CN_GLOBAL_CFG_2.CNUnitSize - 9)
  }
  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 will be 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

FocalPoint 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 the following:

- 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 will appear 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 is 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 denothing whether to use EWA (mode 0) or Peak tracking (mode 1)

- QDM_FRAME_CNT is 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 will enable 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 will be 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.

Chapter 13 - Egress Scheduling & Shaping

13.1 Introduction

Bali 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 Queueing 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.
  - Queueing 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

Quantity	Notation	Definition
Traffic Class	*c*	Identifies one of eight egress queues from which the frame will be 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 $i = \{ c_1, c_2, \ldots c_n \}$, $c_2 = c_1+1$, $c_3 = c_2+1$, *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.

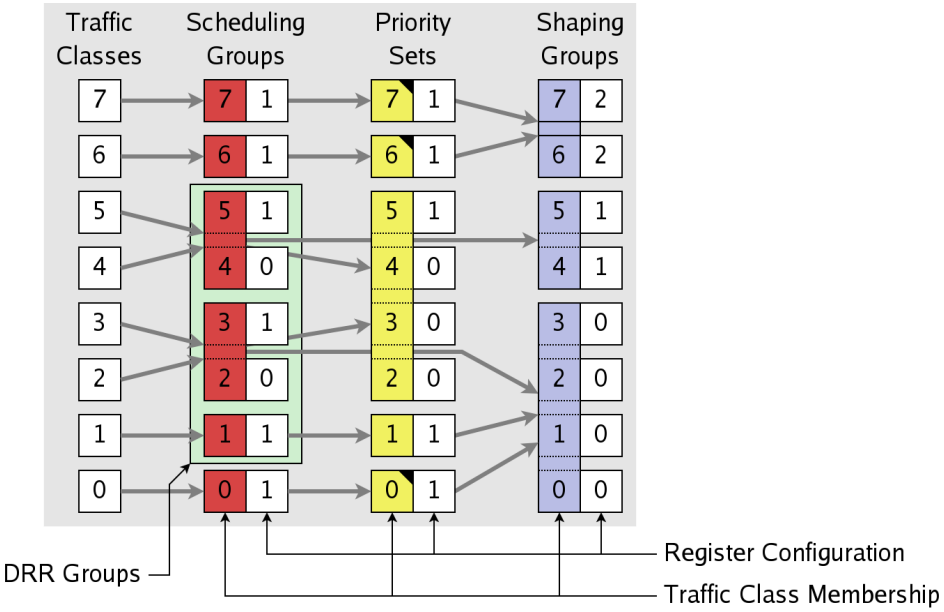
Quantity	Notation	Definition
Bandwidth Shaping Group	*g*	A set of egress traffic classes for a given port which share a common maximum bandwidth allocation. The total egress bandwidth over these classes is guaranteed to never exceed the group's upper-bound limit (UB_g). Shaping groups are defined by 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 another priority set's.
Strict Priority	SP_c	An attribute configured per traffic class. When set, the traffic class will have effectively infinite DC_g , it will preempt lower-priority classes, and it will be 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 will attempt 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 8 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 8 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 determinde by the group having the highest number in the set.

The example below 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 0 indicates that class *c* belongs to the same group or set as class *c+1*; a value of 1 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.



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 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 will be 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. Bali supports two types of pause frames: standard IEEE 802.3 pause and 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 the Implementation section.

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 will be 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. A DRR group *i* belonging to the same priority set as *LastDRRGroup*, with *i* > *LastDRRGroup*.
   f. Any group in the eligibility set.
4. If the selected group is a DRR group, then *LastDRRGroup* is set to that group number. If is not, then *LastDRRGroup* is set to -1.

The next scheduled frame will then be 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

FocalPoint's scheduling algorithm, as described above, has the following characteristics:

- Once a particular DRR group is chosen for scheduling, the group will continue 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 will only be scheduled 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 will be 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 will starve the lower-numbered class as long as it remains eligible.
  - In most applications, this nested strict priority behavior will not 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 data sheet we use the latter term in order 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 Focalpoint 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. Bali's DRR implementation therefore 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 queueing algorithm.

Pseudocode for the DDRR algorithm is provided below. The pseudocode assumes an enqueuing process which assigns ingress frames to their appropriate egress queues, referenced by scheduling group. The *Dequeue()* and *isEmpty()* functions are standard queue operators. This pseudocode only describes the DDRR behavior of FocalPoint'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 the Scheduler Algorithm section.

```
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 will schedule 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 interframe 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

$$LB_i = \frac{Q_i}{\sum_i Q_i}$$

If any Q_i is less than MTU, then the minimum bandwidth allotted to each group becomes dependent on the actual frames queued. Note also that the presence of strict-high classes in the scheduler configuration will also disturb the DRR groups' lower bound bandwidth guarantees.

13.4.2 Bandwidth Shaping

So far in this chapter, 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 FocalPoint, 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 10Gb/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 Bali'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 interframe 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. Refer to the Register Definitions chapter for more details.

Register Name	Size	Description
SWITCH_PRI_TO_CLASS_1 SWITCH_PRI_TO_CLASS_2	2 x 32b	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 24b	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 that 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 24b	Specifies per-port mappings from traffic class to shaping group.
SCHED_DRR_Q	25 x 8 x 24b	Defines the DRR quantum per group per port. Note that value *MAX* is a special value and is equivalent to an infinite weight.
TX_RATE_LIM_CFG	25 x 8 x 32b	Defines the token bucket bandwidth limit parameters per shaping group, per port.
TX_RATE_LIM_USAGE	25 x 8 x 8b	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 8 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%
1	1 -> 1	1	*X*	1	1	100%
0	0 -> 0	1	*X*	1	1	100%

Note that 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 will continue 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%
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 -> 2	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	

Chapter 14 - Statistics and Monitoring

14.1 Frame Counters

FocalPoint 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).

Bali'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. The following table summarizes Bali's counter groups.

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
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 above are implemented in SRAM, and this limitation only applies to those counter groups. The Frame Handler mediates counting activity and will cause, at each clock cycle, 13 simultaneous counting events (one for each group) of which 9 target the SRAM stats block. The SRAM stats block will 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 doesn't cause the frame to be mishandled or dropped. Packets flow normally and only stats are affected.

14.2 Frame Monitoring

All FocalPoint switch devices support two traffic monitoring mechanisms: RX Mirroring and TX Mirroring.

RX Mirroring is enabled by either FFU or Trigger actions and, 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 will still be 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 will be mirrored. In order 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 FocalPoint devices is prevented by setting their ISL tag FTYPE field to 0x2 (Special Delivery) on egress.

Chapter 15 - Register Definitions

15.1 Introduction

This chapter details the entire register set. Registers are grouped together according to their function. The global register map indicates the base address of each group while following register maps list the register in each group along with their offset relative to the base of each group.

All addresses are in 'words' units, hence the "C" expressions:

```
#define FM4000_REG_VITAL_PRODUCT_DATA (0x00000304)
.
.
.
unsigned int *fm4000;
.
.
.
model = fm4000[FM4000_REG_VITAL_PRODUCT_DATA];
.
.
.
```

will produce the right address in a 32 bits CPU.

Many registers are tables which requires one or two indexes. By example the FFU_SLICE_CAM register as two indexes and the actual offset is computed from a simple expression:

```
FFU_SLICE_TCAM[0..31][0..511] => 0x800 + 0x1000*j + 0x2*i
```

In all cases, the index "j" is the left index while the "i" is the right index (FFU_SLICE_TCAM[j][i]).

15.2 Register Map

15.2.1 Register Set Summary

Block	Brief Description	Base Address	Base Size
HSM	HSM Registers	0x00000	2^18=0x40000
LSM	LSM Registers	0x40000	2^18=0x40000
EPL	Ethernet Port Logic Registers (reserved 25 x 1K => 25KW)	0x50000	2^16=0x10000
SCHED2	Scheduler Registers (reserved 4KW)	0x60000	2^16=0x10000
MTABLE	MTABLE Register Set (reserved 64KW)	0xC0000	2^16=0x10000

Block	Brief Description	Base Address	Base Size
STATS_RX_TYPE	Per-Port RX Frame Classification Counters (Group 1)	0x90000	2^14=0x4000
STATS_RX_BIN	Per-Port RX Packet Counters Binned by Length (Group 2)	0x91000	2^14=0x4000
STATS_RX_OCTET	Per-Port RX Octet Counters (Group 3)	0x91380	2^14=0x4000
STATS_RX_PKT_PRI	Per-Port RX Packet Counters per Priority (Group 4)	0x90310	2^14=0x4000
STATS_RX_OCT_PRI	Per-Port RX Octet Counters per Priority (Group 5)	0x92700	2^14=0x4000
STATS_RX_ACTION	Per-Port RX Packet Counters by Forwarding Action (Group 6)	0x92000	2^14=0x4000
STATS_TX_TYPE	Per-Port TX Frame Classification Counters (Group 7)	0x904D0	2^14=0x4000
STATS_TX_BIN	Per-Port TX Packet Counters Binned by Length (Group 8)	0x91460	2^14=0x4000
STATS_TX_OCTET	Tx Octet Counters (Group 9)	0x928C0	2^14=0x4000
MSB	MSB Registers	0x80000	2^16=0x10000
HANDLER	Frame Handler Registers	0x100000	2^20=0x100000
ASSOC	Port Association Registers (reserved 4KW)	0x101000	2^12=0x1000
GLORT	GLORT Control Registers and GLORT Table (reserved 16KW)	0x108000	2^14=0x4000
CM	Congestion Management Registers (reserved 4KW)	0x103000	2^12=0x1000
LAG	Link Aggregation Registers (reserved 4KW)	0x104000	2^12=0x1000
POLICER	Policer Registers (reserved 16KW)	0x10C000	2^14=0x4000
TRIG	Triggers Registers (reserved 4KW)	0x106000	2^12=0x1000
ARP	ARP Control and Table (128KW)	0x120000	2^17=0x20000
FFU	Filtering and Forwarding Unit Registers (256KW address space)	0x140000	2^18=0x40000
STATS_VLAN_PKT	VLAN Packet Counters (Group 11)	0x1A0100	2^8=0x100
STATS_VLAN_OCT	VLAN Octet Counters (Group 12)	0x1A0200	2^8=0x100
STATS_TRIG	Trigger Counters (Group 13)	0x1A0300	2^8=0x100
STATS_EGRESS_ACL	Egress ACL Counters (Group 14)	0x1A0400	2^8=0x100
STATS_CTRL	Statistics Control Registers	0x1A0500	2^8=0x100
L2LOOKUP	Layer 2 Lookup Registers (reserved 128KW)	0x180000	2^17=0x20000

15.2.2 HSM Registers Map

Table 26: HSM Registers Map

Name	Brief Description	Address Offset (Base = 0x00000)
LCI_CFG	Local CPU interface configuration register	0x00001

Name	Brief Description	Address Offset (Base = 0x00000)
LCI_RX_FIFO	Local CPU interface receive FIFO	0x00002
LCI_TX_FIFO	Local CPU interface transmit FIFO	0x00003
LCI_IP	Local CPU interface interrupt status register	0x00004
LCI_IM	Local CPU interface interrupt mask register	0x00005
LCI_STATUS	Local CPU interface status register	0x00006
LAST_FATAL_CODE	Fatal Code received from crossbar	0x00200
INTERRUPT_DETECT	Global interrupt status & control register	0x00201
FATAL_COUNT	Fatal reset count register	0x00202
SOFT_RESET	Soft reset control register	0x00203
WATCHDOG_CFG	Watchdog configuration register	0x00204
PLL_FH_STAT	Frame Handler PPL status register	0x00205
PLL_FH_CTRL	Frame Handler PLL configuration register	0x00206
VITAL_PRODUCT_DATA	Product identification regisgter	0x00304

15.2.3 LSM Registers Map

Table 27: LSM Registers Map

Name	Brief Description	Address Offset (Base = 0x40000)
LSM_INT_DETECT		0x00000
GLOBAL_EPL_INT_DETECT		0x00001
PERR_IP		0x00002
PERR_IM		0x00003
SW_IP	Software Writeable Interrupt	0x00004
SW_IM		0x00005
FRAME_TIME_OUT	Frame timeout duration.	0x00101
BOOT_CTRL *(was CHIP_MODE)*		0x00102
BOOT_STATUS		0x00103
CLK_MULT_1		0x00104
VPD_INFO_1	Chip general information	0x00108
VPD_INFO_2	Chip general information	0x0010A
GPIO_CFG	GPIO Configuration Register	0x00134
GPIO_DATA	GPIO Data Register	0x00135
GPIO_IP	GPIO Interrupt Pending	0x00136
GPIO_IM	GPIO Interrupt Pending	0x00137
I2C_CFG	I2C Configuration	0x00138

Name	Brief Description	Address Offset (Base = 0x40000)
I2C_DATA[0..1]	I2C Data Write Register	0x139 + 0x1*i
I2C_CTRL	I2C Control	0x0013B
MDIO_CFG	MDIO Configuration	0x0013C
MDIO_DATA	MDIO Data Register	0x0013D
MDIO_CTRL	MDIO Control	0x0013E
LED_CFG	LED Configuration Register	0x0013F
EPL_PORT_CTRL[0..24]	EPL Port Control	0x160 + 0x1*i
CRM_CFG_COUNTER[0..255]	Counter rate monitor configuration: Counter and Enable	0x1200 + 0x1*i
CRM_CFG_WINDOW[0..255]	Counter rate monitor configuration: Rate Window	0x1300 + 0x1*i
CRM_CFG_LIMIT[0..255]	Counter rate monitor configuration: Rate Limit	0x1400 + 0x1*i
CRM_LAST_COUNT[0..255]	Last polled counter value	0x1500 + 0x1*i
CRM_EXCEED_COUNT[0..255]	Number of periods over which RateLimit was exceeded.	0x1600 + 0x1*i
CRM_CFG	General Counter Rate Monitor configuration	0x01700
CRM_INT_DETECT	Counter Rate Monitor Interrupt Detect	0x01701
CRM_IP[0..7]	Counter Rate Monitor IP registers	0x1710 + 0x1*i
CRM_IM[0..7]	Counter Rate Monitor IM registers	0x1720 + 0x1*i

15.2.4 Ethernet Port Logic Registers (reserved 25 x 1K => 25KW) Map

Table 28: Ethernet Port Logic Registers (reserved 25 x 1K => 25KW) Map

Name	Brief Description	Address Offset (Base = 0x50000)
SERDES_CTRL_1[0..24]	SERDES equalization and drive	0x0 + 0x400*i
SERDES_CTRL_2[0..24]	SERDES reset, polarity, and comma detection	0x1 + 0x400*i
SERDES_CTRL_3[0..24]	SERDES deassertion count and link count	0x2 + 0x400*i
SERDES_TEST_MODE[0..24]	SERDES Test Mode Control Register	0x3 + 0x400*i
SERDES_STATUS[0..24]	SERDES Status	0x4 + 0x400*i
SERDES_IP[0..24]	SERDES Interrupt Register	0x5 + 0x400*i
SERDES_IM[0..24]	SERDES Interrupt Mask Register	0x6 + 0x400*i
SERDES_BIST_ERR_CNT[0..24]	SERDES BIST Count Register	0x7 + 0x400*i
PCS_CFG_1[0..24]	PCS error handling, lane ordering, and LFSR seed	0x9 + 0x400*i
PCS_CFG_2[0..24]	Local Fault Configuration Register	0xA + 0x400*i
PCS_CFG_3[0..24]	Remote Fault Configuration Register	0xB + 0x400*i
PCS_CFG_4[0..24]	Transmit FSIG Configuration Register	0xC + 0x400*i
PCS_CFG_5[0..24]	Receive FSIG Configuration Register	0xD + 0x400*i

Name	Brief Description	Address Offset (Base = 0x50000)
PCS_IP[0..24]	PCS Interrupt Pending Register	0xE + 0x400*i
PCS_IM[0..24]	PCS Interrupt Mask Register	0xF + 0x400*i
SYNCBUF_CFG[0..24] *(was PACING_RATE)*	PCS Syncbuf Control Register	0x18 + 0x400*i
MAC_CFG_1[0..24]	Frame size and header offset	0x1A + 0x400*i
MAC_CFG_2[0..24]	MAC Configuration 2 Register	0x1B + 0x400*i
MAC_CFG_3[0..24]	MAC Pause Value	0x1C + 0x400*i
MAC_CFG_5[0..24]	MAC PAUSE Address (LOW)	0x1E + 0x400*i
MAC_CFG_6[0..24]	MAC Pause Address (HIGH)	0x1F + 0x400*i
TX_VPRI_MAP_1[0..24] *(was TX_PRI_MAP_1)*	MAC Priority Remapping (priorities 0-7)	0x20 + 0x400*i
TX_VPRI_MAP_2[0..24] *(was TX_PRI_MAP_2)*	MAC Priority Remapping (priorities 8-15)	0x21 + 0x400*i
MAC_STATUS[0..24]	MAC Status Register	0x22 + 0x400*i
MAC_IP[0..24]	MAC Interrupt Pending Register	0x23 + 0x400*i
MAC_IM[0..24]	MAC Interrupt Mask Register	0x24 + 0x400*i
EPL_INT_DETECT[0..24]	EPL Interrupt Source Detect	0x2B + 0x400*i
EPL_LED_STATUS[0..24]	EPL Activity Status Register	0x2A + 0x400*i
STAT_EPL_ERROR1[0..24]	Error Counter	0x25 + 0x400*i
STAT_EPL_ERROR2[0..24]	Error Counter	0x28 + 0x400*i
STAT_TX_BYTECOUNT[0..24]	Transmit Octet Counter	0x2C + 0x400*i
STAT_RX_JABBER[0..24]	RX Jabber Counter	0x29 + 0x400*i
STAT_TX_CRC[0..24]	Transmit CRC Counter	0x27 + 0x400*i
STAT_TX_PAUSE[0..24]	Transmit Pause Counter	0x26 + 0x400*i
SRC_MAC_LO[0..24]	Low 32b of MAC address to be used for routed frames	0x33 + 0x400*i
SRC_MAC_HI[0..24]	High 16b of MAC address to be used for routed frames	0x34 + 0x400*i
SRC_MAC_VIRTUAL_LO[0..24]	Low 32b of MAC address to be used for routed frames	0x35 + 0x400*i
SRC_MAC_VIRTUAL_HI[0..24]	High 16b of MAC address to be used for routed frames	0x36 + 0x400*i
JITTER_TIMER[0..24]	Egress Jitter Timer	0x3D + 0x400*i
PARSE_CFG[0..24]	Frame Parsing Configuration	0x3E + 0x400*i
MAC_VLAN_ETYPE_1[0..24]	C-VLAN Type Configuration	0x66 + 0x400*i
MAC_VLAN_ETYPE_2[0..24] *(was MAC_VLAN_ETYPE)*	S-VLAN or Custom VLAN Type Configuration	0x3F + 0x400*i
PARSE_RLT_1[0..24]	Low 32 bits of CPID	0x40 + 0x400*i

Name	Brief Description	Address Offset (Base = 0x50000)
PARSE_RLT_2[0..24]	Type and upper 16 bits of CPID	0x41 + 0x400*i
TX_TRUNC[0..24]	Truncation Lengths and enables for CPU and Mirrored frames	0x42 + 0x400*i
CPID_0[0..24]	CPID #0 for BCN frames	0x50 + 0x400*i
CPID_1[0..24]	CPID #1 for BCN frames	0x52 + 0x400*i
CPID_2[0..24]	CPID #2 for BCN frames	0x54 + 0x400*i
CPID_3[0..24]	CPID #3 for BCN frames	0x56 + 0x400*i
CPID_4[0..24]	CPID #4 for BCN frames	0x58 + 0x400*i
CPID_5[0..24]	CPID #5 for BCN frames	0x5A + 0x400*i
CPID_6[0..24]	CPID #6 for BCN frames	0x5C + 0x400*i
CPID_7[0..24]	CPID #7 for BCN frames	0x5E + 0x400*i
DI_CFG[0..24] *(was L4_CTL)*	MISC control bits for deep inspection parsing	0x67 + 0x400*i
TCP_WD_MASK_LO[0..24]	TCP half word bit mask for first 32 half-words	0x68 + 0x400*i
TCP_WD_MASK_HI[0..24]	TCP half word bit mask for last 16 half-words	0x69 + 0x400*i
UDP_WD_MASK_LO[0..24]	UDP half word bit mask for first 32 half-words	0x6A + 0x400*i
UDP_WD_MASK_HI[0..24]	UDP half word bit mask for last 16 half-words	0x6B + 0x400*i
L4PROT1_WD_MASK_LO[0..24]	L4 Custom Protocol 1 half word bit mask for first 32 half-words	0x6C + 0x400*i
L4PROT1_WD_MASK_HI[0..24]	L4 Custom Protocol1 half word bit mask for last 16 half-words	0x6D + 0x400*i
L4PROT2_WD_MASK_LO[0..24]	L4 Custom Protocol 2 half word bit mask for first 32 half-words	0x6E + 0x400*i
L4PROT2_WD_MASK_HI[0..24]	L4 Custom Protocol 2 half word bit mask for last 16 half-words	0x6F + 0x400*i
AN_TX_MSG0[0..24] *(was AN_TX_CW_Lo)*	Auto negotiation message 0	0x70 + 0x400*i
AN_TX_MSG1[0..24] *(was AN_TX_CW_Hi)*	Auto negotiation message 1	0x72 + 0x400*i
AN_RX_MSG0[0..24] *(was AN_RX_CW_Lo)*	Auto negotiation message 0	0x74 + 0x400*i
AN_RX_MSG1[0..24] *(was AN_RX_CW_Hi)*	Auto negotiation message 1	0x76 + 0x400*i
AN_CTL[0..24]	Auto-Negotiation Control Bits	0x78 + 0x400*i
AN_STATUS[0..24]	Auto-Negotiation Status	0x79 + 0x400*i
AN_TIMEOUT[0..24] *(was AN_TimeoutValue)*	Auto-Negotiation Timeout Value	0x7A + 0x400*i
AN_TX_TIMER[0..24]	Transmitter timer	0x7B + 0x400*i
VLANTAG_TABLE[0..127] [0..24]	Defines VLAN tagging	0x80 + 0x1*j + 0x400*i

15.2.5 Scheduler Registers (reserved 4KW) Map

Table 29: Scheduler Registers (reserved 4KW) Map

Name	Brief Description	Address Offset (Base = 0x60000)
SCHED_GROUP_CFG[0..24]	Scheduling Group Configuration	0x40 + 0x2*i
FUSE_SEG	*** TO BE SUPPLIED ***	0x000FF
FUSE_PORT	*** TO BE SUPPLIED ***	0x000FE

15.2.6 MTABLE Register Set (reserved 64KW) Map

Table 30: MTABLE Register Set (reserved 64KW) Map

Name	Brief Description	Address Offset (Base = 0xC0000)
TX_MIRROR	Defines the TX mirror source port and destination port.	0x00000
LOG_MASK *(was CPU_MASK)*	Defines the destination mask for logging	0x00001
MIRROR_GLORTS	Defines the destination GloRTs for TX mirror and logged frames	0x04019
LOOPBACK_SUPPRESS[0..24]	GloRTs used by loopback suppression algorithm	0x4000 + 0x1*i
IP_MULTICAST_TABLE[0..16383]	Table of VLANs for IP multicast	0x8000 + 0x1*i

15.2.7 Per-Port RX Frame Classification Counters (Group 1) Map

Exactly one counter in this group is incremented for each frame received.

Table 31: Per-Port RX Frame Classification Counters (Group 1) Map

Name	Brief Description	Address Offset (Base = 0x90000)
STAT_RxUcstPktsNonIP[0..24]	Count of Non-IP Layer 2 unicast frames received.	0x0 + 0x4*i
STAT_RxMcstPktsNonIP[0..24]	Count of Non-IP Layer 2 multicast frames received.	0x70 + 0x4*i
STAT_RxBcstPktsNonIP[0..24]	Count of Non-IP Layer 2 broadcast frames received.	0x2 + 0x4*i
STAT_RxUcstPktsIPv4[0..24]	Count of IPv4 frames received with unicast L2 DMACs.	0x72 + 0x4*i
STAT_RxMcstPktsIPv4[0..24]	Count of IPv4 frames received with multicast L2 DMACs.	0xE2 + 0x4*i
STAT_RxBcstPktsIPv4[0..24]	Count of IPv4 frames received with broadcast L2 DMACs.	0xE0 + 0x4*i
STAT_RxUcstPktsIPv6[0..24]	Count of IPv6 frames received with unicast L2 DMACs.	0x150 + 0x4*i

Name	Brief Description	Address Offset (Base = 0x90000)
STAT_RxMcstPktsIPv6[0..24]	Count of IPv6 frames received with multicast L2 DMACs.	0x1C0 + 0x4*i
STAT_RxBcstPktsIPv6[0..24]	Count of IPv6 frames received with broadcast L2 DMACs.	0x152 + 0x4*i
STAT_RxPausePkts[0..24]	Count of IEEE 802.3 pause packets received.	0x1C2 + 0x4*i
STAT_RxCBPausePkts[0..24]	Count of class-based pause packets received.	0x230 + 0x4*i
STAT_RxSymbolErrors[0..24]	Count of packets received containing symbol errors.	0x232 + 0x4*i
STAT_RxFCSErrors[0..24]	Count of correctly-framed packets received with bad CRC.	0x2A0 + 0x4*i
STAT_RxFrameSizeErrors[0..24]	Count of packets that are undersized or oversized.	0x2A2 + 0x4*i

15.2.8 Per-Port RX Packet Counters Binned by Length (Group 2) Map

Exactly one counter in this group is incremented for each frame received.

Table 32: Per-Port RX Packet Counters Binned by Length (Group 2) Map

Name	Brief Description	Address Offset (Base = 0x91000)
STAT_RxMinto63[0..24]	Number of frames received with MinFrameSize <= length < 64, when MinFrameSize is set less than 64.	0x0 + 0x4*i
STAT_Rx64Pkts[0..24]	Number of frames received with exactly 64 octets.	0x2 + 0x4*i
STAT_Rx65to127[0..24]	Number of frames received with 64 < length < 128.	0x70 + 0x4*i
STAT_Rx128to255[0..24]	Number of frames received with 128 <= length < 256.	0x72 + 0x4*i
STAT_Rx256to511[0..24]	Number of frames received with 256 <= length < 512.	0xE0 + 0x4*i
STAT_Rx512to1023[0..24]	Number of frames received with 512 <= length < 1023.	0xE2 + 0x4*i
STAT_Rx1024to1522[0..24]	Number of frames received with 1024 <= length <= 1522.	0x150 + 0x4*i
STAT_Rx1523to2047[0..24]	Number of frames received with 1523 <= length < 2048.	0x152 + 0x4*i
STAT_Rx2048to4095[0..24]	Number of frames received with 2048 <= length < 4096.	0x1C0 + 0x4*i
STAT_Rx4096to8191[0..24]	Number of frames received with 4096 <= length < 8192.	0x1C2 + 0x4*i
STAT_Rx8192to10239[0..24]	Number of frames received with 8192 <= length < 10240.	0x230 + 0x4*i
STAT_Rx10240toMax[0..24]	Number of frames received with 10240 <= length <= MaxFrameSize, assuming MaxFrameSize (in MAC_CFG_1) is set equal to or larger than 10240.	0x232 + 0x4*i
STAT_RxFragments[0..24]	Counts ingress frames smaller than MinFrameSize (MAC_CFG_1) that are received with incorrect	0x2A0 + 0x4*i

Name	Brief Description	Address Offset (Base = 0x91000)
	alignment or CRC. Received frames smaller than MinFrameSize	
STAT_RxUndersized[0..24]	Counts ingress frames smaller than MinFrameSize (MAC_CFG_1) that are otherwise well formed with a good CRC.	0x2A2 + 0x4*i
STAT_RxOversized[0..24]	Total number of frames received larger than MaxFrameSize (MAC_CFG_1). Includes both frames received with alignment or CRC errors and correctly formatted frames. To obtain the number of oversized frames received without error, subtract STAT_RxJabber from the corresponding EPL register set.	0x310 + 0x4*i

15.2.9 Per-Port RX Octet Counters (Group 3) Map

Exactly one counter in this group is incremented for each frame received.

Table 33: Per-Port RX Octet Counters (Group 3) Map

Name	Brief Description	Address Offset (Base = 0x91380)
STAT_RxOctetsNonIP[0..24]	Total number of octets received from well-formed frames that have non-IP EtherTypes.	0x0 + 0x4*i
STAT_RxOctetsIPv4[0..24]	Total number of octets received from well-formed frames with EtherType 0x0800 (IPv4).	0x2 + 0x4*i
STAT_RxOctetsIPv6[0..24]	Total number of octets received from well-formed frames with EtherType 0x86DD (IPv6).	0x70 + 0x4*i
STAT_RxOctetsError[0..24]	Total number of octets received from frames with incorrect CRC, symbol errors, or bad alignment.	0x72 + 0x4*i

15.2.10 Per-Port RX Packet Counters per Priority (Group 4) Map

Classification of frames is either by ingress user priority or traffic class, depending on the state of PrioritySelect in STATS_CFG. Exactly one counter in this group is incremented for each frame received. Note that frames with incorrect CRC will be included.

Table 34: Per-Port RX Packet Counters per Priority (Group 4) Map

Name	Brief Description	Address Offset (Base = 0x90310)
STAT_RxP0[0..24]	Nubmer of frames received with priority 0.	0x0 + 0x4*i
STAT_RxP1[0..24]	Nubmer of frames received with priority 1.	0x2 + 0x4*i
STAT_RxP2[0..24]	Nubmer of frames received with priority 2.	0x70 + 0x4*i
STAT_RxP3[0..24]	Nubmer of frames received with priority 3.	0x72 + 0x4*i
STAT_RxP4[0..24]	Nubmer of frames received with priority 4.	0xE0 + 0x4*i

Name	Brief Description	Address Offset (Base = 0x90310)
STAT_RxP5[0..24]	Nubmer of frames received with priority 5.	0xE2 + 0x4*i
STAT_RxP6[0..24]	Nubmer of frames received with priority 6.	0x150 + 0x4*i
STAT_RxP7[0..24]	Nubmer of frames received with priority 7.	0x152 + 0x4*i

15.2.11 Per-Port RX Octet Counters per Priority (Group 5) Map

Classification of frames is either by ingress user priority or traffic class, depending on the state of PrioritySelect in STATS_CFG. Exactly one counter in this group is incremented for each frame received. Note that frames with incorrect CRC will be included.

Table 35: Per-Port RX Octet Counters per Priority (Group 5) Map

Name	Brief Description	Address Offset (Base = 0x92700)
STAT_RxOctetsP0[0..24]	Number of octets received from frames of priority 0.	0x0 + 0x4*i
STAT_RxOctetsP1[0..24]	Number of octets received from frames of priority 1.	0x2 + 0x4*i
STAT_RxOctetsP2[0..24]	Number of octets received from frames of priority 2.	0x70 + 0x4*i
STAT_RxOctetsP3[0..24]	Number of octets received from frames of priority 3.	0x72 + 0x4*i
STAT_RxOctetsP4[0..24]	Number of octets received from frames of priority 4.	0xE0 + 0x4*i
STAT_RxOctetsP5[0..24]	Number of octets received from frames of priority 5.	0xE2 + 0x4*i
STAT_RxOctetsP6[0..24]	Number of octets received from frames of priority 6.	0x150 + 0x4*i
STAT_RxOctetsP7[0..24]	Number of octets received from frames of priority 7.	0x152 + 0x4*i

15.2.12 Per-Port RX Packet Counters by Forwarding Action (Group 6) Map

Exactly one counter in this group is incremented for each frame received. Note that frames with incorrect CRC will be included in these counts.

Table 36: Per-Port RX Packet Counters by Forwarding Action (Group 6) Map

Name	Brief Description	Address Offset (Base = 0x92000)
STAT_FIDForwarded[0..24]	Number of frames that were forwarded normally, either unicast or multicast, as a result of a valid MA Table lookup. Also includes layer 2 broadcast frames.	0x0 + 0x4*i
STAT_FloodForwarded[0..24]	Number of frames that were flooded due to a miss in the MA Table.	0x2 + 0x4*i
STAT_SpeciallyHandled[0..24]	Number of frames processed with FTYPE==0x2 (Special Delivery). All standard filtering, forwarding, and lookup rules are bypassed in this case.	0x70 + 0x4*i
STAT_ParseErrDrops[0..24]	Number of frames dropped due to header parse errors.	0x72 + 0x4*i

Name	Brief Description	Address Offset (Base = 0x92000)
STAT_ParityError[0..24]	Number of frames dropped due to memory parity errors encountered in the Frame Processing pipeline.	0xE0 + 0x4*i
STAT_Trapped[0..24]	Number of frames trapped to the CPU for any reason not covered by other counters in this group (e.g. SecurityViolations). These include frames with reserved IEEE multicast addresses (BPDU, 802.1X, LACP, etc.), trapped IP frames (e.g. ICMP), and programmably trapped frames (due to the FFU, triggers, etc.)	0xE2 + 0x4*i
STAT_PauseDrops[0..24]	Number of MAC Control frames dropped (either standard IEEE pause frames or class-based pause frames).	0x150 + 0x4*i
STAT_STPDrops[0..24]	Number of frames dropped due to the ingress port or all egress ports not being in the forwarding state. Note: Multicast frames that are only partially filtered due to a strict subset of egress ports being in the forwarding state will not be included in this count.	0x152 + 0x4*i
STAT_SecurityViolations[0..24]	Number of frames that are dropped or trapped because they are considered a security violation.	0x1C0 + 0x4*i
STAT_VlanTagDrops[0..24]	Number of frames discarded because the frames were untagged, and drop untagged is configured, or the frames were tagged, and dropped tagged is configured.	0x1C2 + 0x4*i
STAT_VlanIngressDrops[0..24]	Number of frames dropped for an ingress VLAN boundary violation.	0x230 + 0x4*i
STAT_VlanEgressDrops[0..24]	Number of unicast frames dropped for an egress VLAN boundary violation. Note: Multicast frames are counted only if they are entirely dropped due to *all* egress ports not belonging to the appropriate VLAN.	0x232 + 0x4*i
STAT_GlortMissDrops[0..24]	Number of frames dropped due to not finding a matching entry in the GLORT_CAM.	0x2A0 + 0x4*i
STAT_FFUDrops[0..24]	Number of frames dropped due to an FFU action.	0x2A2 + 0x4*i
STAT_TriggerDrops[0..24]	Number of frames dropped due to a trigger drop action.	0x310 + 0x4*i
STAT_PolicerDrops[0..24]	Number of frames dropped due to policer rate limitation.	0x312 + 0x4*i
STAT_TTLDrops[0..24]	Number of IP frames dropped due to TTL<=1.	0x380 + 0x4*i
STAT_CMPrivDrops[0..24]	(Congestion Management) Number of frames dropped due to the global privileged watermark.	0x382 + 0x4*i
STAT_SMP0Drops[0..24]	(Congestion Management) Number of frames dropped due to insufficient memory in shared partition 0.	0x3F0 + 0x4*i
STAT_SMP1Drops[0..24]	(Congestion Management) Number of frames dropped due to insufficient memory in shared partition 1.	0x3F2 + 0x4*i

Name	Brief Description	Address Offset (Base = 0x92000)
STAT_RxHog0Drops[0..24]	(Congestion Management) Number of frames dropped due to the SMP 0 RX hog watermark.	0x460 + 0x4*i
STAT_RxHog1Drops[0..24]	(Congestion Management) Number of frames dropped due to the SMP 1 RX hog watermark.	0x462 + 0x4*i
STAT_TxHog0Drops[0..24]	(Congestion Management) Number of frames dropped to all egress ports due to the SMP 0 TX hog watermark.	0x4D0 + 0x4*i
STAT_TxHog1Drops[0..24]	(Congestion Management) Number of frames dropped to all egress ports due to the SMP 1 TX hog watermark.	0x4D2 + 0x4*i
STAT_RateLimit0Drops[0..24]	(Congestion Management) Number of frames dropped due to ingress rate limiting on SMP 0.	0x540 + 0x4*i
STAT_RateLimit1Drops[0..24]	(Congestion Management) Number of frames dropped due to ingress rate limiting on SMP 1.	0x542 + 0x4*i
STAT_BadSMPDrops[0..24]	(Congestion Management) Number of frames dropped due to illegal membership.	0x5B0 + 0x4*i
STAT_TriggerRedirects[0..24]	Number of frames redirected due to trigger action.	0x5B2 + 0x4*i
STAT_Others[0..24]	Number of frames not included in any other group 6 counters.	0x620 + 0x4*i

15.2.13 Per-Port TX Frame Classification Counters (Group 7) Map

Exactly one counter in this group is incremented for each frame that is scheduled for transmission. Each multicast-replicated frame (for both L2 and L3 multicast) is counted individually. This counter group does not include pause frames generated by the switch, since these are not explicitly scheduled; the STAT_TX_PAUSE in the EPL register set counts these frames.
Note that frames with incorrect FCS may be included in the TxUcstPkts, TxMcstPkts, and TxBcstPkts counters due to cut-through. If cut-through is disabled for the egress port, then these counter values will never include frames with incorrect FCS since none will be transmitted.

Table 37: Per-Port TX Frame Classification Counters (Group 7) Map

Name	Brief Description	Address Offset (Base = 0x904D0)
STAT_TxUcstPkts[0..24]	Number of unicast frames transmitted, possibly with incorrect FCS due to cut-through.	0x0 + 0x4*i
STAT_TxBcstPkts[0..24]	Number of broadcast frames transmitted, possibly with incorrect FCS due to cut-through.	0x2 + 0x4*i
STAT_TxMcstPkts[0..24]	Number of multicast frames transmitted, possibly with incorrect FCS due to cut-through.	0x70 + 0x4*i
STAT_TxTimeOutDrop[0..24]	Number of frames purged from memory due to the frame timeout mechanism.	0x72 + 0x4*i
STAT_TxErrorDrop[0..24]	Number of frames that were scheduled for transmission but were dropped due to frame length,	0xE0 + 0x4*i

Name	Brief Description	Address Offset (Base = 0x904D0)
	ingress FCS, or phy-level errors. Frames marked as erroneous on ingress which were transmitted (due to cut-through) will not be included in this counter.	
STAT_TxLoopbackDrop[0..24]	Number of frames that were discarded due to loopback suppression.	0xE2 + 0x4*i

15.2.14 Per-Port TX Packet Counters Binned by Length (Group 8) Map

Exactly one counter in this group is incremented for each non-Pause frame transmitted. Frames transmitted with FCS errors are included in these counts. Note: The length binning is defined in terms of the frame's *ingress* length, which may not equal its egress length due to tagging modifications.

Table 38: Per-Port TX Packet Counters Binned by Length (Group 8) Map

Name	Brief Description	Address Offset (Base = 0x91460)
STAT_TxMinto63[0..24]	Number of frames transmitted with MinFrameSize <= ingress_length < 64, when MinFrameSize is set less than 64.	0x0 + 0x4*i
STAT_Tx64Pkts[0..24]	Number of frames transmitted that were 64 octets long on ingress.	0x2 + 0x4*i
STAT_Tx65to127[0..24]	Number of frames transmitted with 64 < ingress_length < 128.	0x70 + 0x4*i
STAT_Tx128to255[0..24]	Number of frames transmitted with 128 <= ingress_length < 256.	0x72 + 0x4*i
STAT_Tx256to511[0..24]	Number of frames transmitted with 256 <= ingress_length < 512.	0xE0 + 0x4*i
STAT_Tx512to1023[0..24]	Number of frames transmitted with 512 <= ingress_length < 1024.	0xE2 + 0x4*i
STAT_Tx1024to1522[0..24]	Number of frames transmitted with 1024 <= ingress_length <= 1522.	0x150 + 0x4*i
STAT_Tx1523to2047[0..24]	Number of frames transmitted with 1523 <= ingress_length < 2048.	0x152 + 0x4*i
STAT_Tx2048to4095[0..24]	Number of frames transmitted with 2048 <= ingress_length < 4096.	0x1C0 + 0x4*i
STAT_Tx4096to8191[0..24]	Number of frames transmitted with 4096 <= ingress_length < 8192.	0x1C2 + 0x4*i
STAT_Tx8192to10239[0..24]	Number of frames transmitted with 8192 <= ingress_length < 10240.	0x230 + 0x4*i
STAT_Tx10240toMax[0..24]	Number of frames transmitted with 10240 <= ingress_length < MaxFrameSize, assuming MaxFrameSize (in MAC_CFG_1) is set equal to or larger than 10240.	0x232 + 0x4*i

Name	Brief Description	Address Offset (Base = 0x91460)
STAT_TxErrors[0..24]	Number of frames with error, including frames purged from memory due to the frame timeout mechanism, frames that were scheduled for transmission but were dropped due to frame length, ingress FCS, or phy-level errors, and frames that were discarded due to loopback suppression. This should be equal to the sum of STAT_TxTimeOutDrop, STAT_TxErrorDrop, and STAT_TxLoopbackDrop.	0x2A0 + 0x4*i

15.2.15 Tx Octet Counters (Group 9) Map

Table 39: Tx Octet Counters (Group 9) Map

Name	Brief Description	Address Offset (Base = 0x928C0)
STAT_TxOctets[0..24]	Total number of ingress octets transmitted on this port. Note this counter does not accurately represent the total number of octets transmitted on this port, and therefore is not very useful. See also STAT_TX_BYTECOUNT.	0x0 + 0x4*i
STAT_TxErrorOctets[0..24]	Total octet count of frames scheduled for transmission but were dropped due to frame timeout, min/max length bound errors, invalid ingress FCS, or phy-level errors.	0x2 + 0x4*i

15.2.16 MSB Registers Map

Table 40: MSB Registers Map

Name	Brief Description	Address Offset (Base = 0x80000)
MSB_CFG	MSB configuration register.	0x00000
MSB_IBM_GLORT	The glort for the MSB.	0x00001
MSB_IBM_INT	Interrupt glort and interrupt interval.	0x00002
MSB_INT_FRAME	Fields for an FIBM interrupt frame.	0x00003
MSB_STATS_0	FIBM request frame counter.	0x00004
MSB_STATS_1	Non-FIBM frame counter.	0x00005
MSB_STATS_2	Error frame counter.	0x00006
MSB_INTR_CTR_0	Interrupt counter.	0x00007
MSB_INTR_CTR_1	Interrupt seq. counter.	0x00008
MSB_INTR_CTR_2	FIBM CRC error counter.	0x00009
MSB_INTR_CTR_3	Non-FIBM CRC error counter.	0x0000A
MSB_INTR_CTR_4	Frame and no cpu counter.	0x0000B

Name	Brief Description	Address Offset (Base = 0x80000)
MSB_INTR_CTR_5	Invalid FIBM op counter.	0x0000C
MSB_IP	Interrupts pending.	0x0000D
MSB_IM	Interrupt mask.	0x0000E
MSB_RX_EPL_RATE	Things affecting frame transmission rate from MSB to RX EPL.	0x0000F
MSB_SCRATCH_0	Scratch register 0.	0x00010
MSB_SCRATCH_1	Scratch register 1.	0x00011
MSB_SCRATCH_2	Scratch register 2.	0x00012
MSB_SCRATCH_3	Scratch register 3.	0x00013
MSB_SCRATCH_4	Scratch register 4.	0x00014
MSB_SCRATCH_5	Scratch register 5.	0x00015
MSB_SCRATCH_6	Scratch register 6.	0x00016
MSB_SCRATCH_7	Scratch register 7.	0x00017
MSB_SCRATCH_8	Scratch register 8.	0x00018
MSB_SCRATCH_9	Scratch register 9.	0x00019
MSB_SCRATCH_10	Scratch register 10.	0x0001A
MSB_SCRATCH_11	Scratch register 11.	0x0001B
MSB_SCRATCH_12	Scratch register 12.	0x0001C
MSB_SCRATCH_13	Scratch register 13.	0x0001D
MSB_SCRATCH_14	Scratch register 14.	0x0001E
MSB_SCRATCH_15	Scratch register 15.	0x0001F
MSB_TS	Test and set. INTERNAL USE ONLY.	0x00020
MSB_CREDITS	HSM Crossring credits. INTERNAL USE ONLY.	0x00021
MSB_SRAM_REPAIR_0	SRAM b0 repair addr. INTERNAL USE ONLY.	0x00022
MSB_SRAM_REPAIR_1	SRAM b1 repair addr. INTERNAL USE ONLY.	0x00023

15.2.17 Frame Handler Registers Map

Table 41: Frame Handler Registers Map

Name	Brief Description	Address Offset (Base = 0x100000)
SYS_CFG_1	Layer 2 configuration	0x00000
SYS_CFG_3	Top 16 bits of the CPU MAC address	0x00002
SYS_CFG_4	Bottom 32 bits of the CPU MAC address	0x00003
SYS_CFG_7	Aging configuration	0x00006
SYS_CFG_8	Enables various units in the Frame Handler pipeline	0x00007

Name	Brief Description	Address Offset (Base = 0x100000)
PORT_VLAN_IP_1		0x00012
PORT_VLAN_IM_1		0x00013
PORT_VLAN_IP_2		0x00014
PORT_VLAN_IM_2		0x00015
PORT_MAC_SEC_IP		0x00016
PORT_MAC_SEC_IM		0x00017
FH_INT_DETECT	Frame Handler Interrupt Detect	0x00018
SYS_CFG_ROUTER	Global system configuration for ROUTER	0x00019
L34_HASH_CFG	Configures hashing of L3/L4 headers for ECMP and LAG	0x00020
L34_FLOW_HASH_CFG_1	Configures masking of flow fields in L34 hash.	0x00021
L34_FLOW_HASH_CFG_2	Configures masking of flow fields in L34 hash.	0x00022
L234_HASH_CFG	Configures hashing of L2/L3/L4 headers for LAG	0x00023
TX_MIRROR_FH	Defines the TX mirror source port and destination port. Same as the one in MTABLE.	0x0002E
CPU_TRAP_MASK_FH	Defines the CPU mask for trapping	0x0002F
CPU_LOG_MASK_FH	Defines the CPU mask for logging.	0x00030
TRAP_GLORT *(was CPU_GLORT)*	Defines the trap glort when a frame must be trapped.	0x00035
RX_MIRROR_CFG	Configures the rxMirrorGlort and rxMirrorPort	0x00031
PARITY_IP	Parity interrupt register.	0x00032
PARITY_IM	Parity interrupt mask register.	0x00033
SAF_MATRIX[0..24]	Store-and-forward Matrix	0x40 + 0x1*i
FH_LOOPBACK_SUPPRESS[0..24]	GloRTs used by loopback suppression algorithm	0x60 + 0x1*i
INTERNAL_PORT_MASK	Internal port mask is used to remove internal ports form dstMask to generate the MTableMask	0x00080
MGMT_CLK_COUNTER	Determines the interval of mgmt commands issued to the Frame Handler	0x00082
MGMT_FFU_CLK_COUNTER	Determines the interval of ffu mgmt commands issued to the Frame Handler	0x00083

15.2.18 Port Association Registers (reserved 4KW) Map

Table 42: Port Association Registers (reserved 4KW) Map

Name	Brief Description	Address Offset (Base = 0x101000)
PORT_CFG_1[0..24]	VLAN, DSCP, and Priority options by source port.	0x0 + 0x1*i
PORT_CFG_2[0..24]	Destination bitmask per source port.	0x20 + 0x1*i
PORT_CFG_3[0..24]	Security and learning options by source port.	0x40 + 0x1*i

Name	Brief Description	Address Offset (Base = 0x101000)
PORT_CFG_ISL[0..24]	ISL tag defaults per source port.	0x60 + 0x1*i
RX_VPRI_MAP[0..24] *(was RX_PRI_MAP)*	Maps RX VLAN user priority and CFI to internal VPRI per port.	0x80 + 0x2*i
DSCP_PRI_MAP[0..63]	Maps DSCP to SwitchPri.	0xC0 + 0x1*i
VPRI_PRI_MAP[0..15]	Maps VPRI to SwitchPri.	0x100 + 0x1*i

15.2.19 GLORT Control Registers and GLORT Table (reserved 16KW) Map

Table 43: GLORT Control Registers and GLORT Table (reserved 16KW) Map

Name	Brief Description	Address Offset (Base = 0x108000)
GLORT_DEST_TABLE[0..4095]		0x0 + 0x2*i
GLORT_CAM[0..255]		0x2000 + 0x1*i
GLORT_RAM[0..255]		0x2200 + 0x2*i

15.2.20 Congestion Management Registers (reserved 4KW) Map

Table 44: Congestion Management Registers (reserved 4KW) Map

Name	Brief Description	Address Offset (Base = 0x103000)
CM_TX_TC_PRIVATE_WM[0..24] [0..7]	Private watermark per traffic class per port	0x0 + 0x8*j + 0x1*i
CM_TX_TC_USAGE[0..24] [0..7]	Stores the number of segments per traffic class per tx	0x100 + 0x8*j + 0x1*i
CM_TX_SMP_HOG_WM[0..24] [0..3] *(was CM_TX_SHARED_HOG_WM)*	TX hog watermark	0x200 + 0x4*j + 0x1*i
CM_RX_SMP_PAUSE_WM[0..24] [0..1]	RX pause watermark per SMP	0x280 + 0x2*j + 0x1*i
CM_RX_SMP_PRIVATE_WM[0..24] [0..1]	RX private watermark per SMP	0x2C0 + 0x2*j + 0x1*i
CM_RX_SMP_USAGE[0..24] [0..1]	Stores the number of segments per port per SMPs	0x300 + 0x2*j + 0x1*i
CM_TX_SMP_USAGE[0..24] [0..1]	Stores the number of segments per port per SMPs	0x380 + 0x2*j + 0x1*i
CM_TX_SMP_PRIVATE_WM[0..24] [0..1]	TX private watermark per SMP per port	0x3C0 + 0x2*j + 0x1*i
CM_RX_SMP_HOG_WM[0..24] [0..1] *(was CM_RX_SHARED_SMP_HOG_WM)*	Maximum number of segments allowed per receiver per SMP	0x400 + 0x2*j + 0x1*i
CM_PAUSE_RESEND_INTERVAL[0..24]	Periodicity of Pause Message Transmits	0x480 + *i
CM_PORT_CFG[0..24]	Various per-port configuration settings	0x4A0 + *i
CM_RX_USAGE[0..24]	Number of segments used per port	0x4C0 + 0x1*i

Name	Brief Description	Address Offset (Base = 0x103000)
CM_SHARED_WM[0..15] *(was CM_RX_SHARED_WM)*	Shared watermark per switch priority	0x4E0 + 0x1*i
CM_PAUSE_DECIMATION[0..7]	Pause Timer Period Control	0x4F0 + *i
CM_SHARED_SMP_PAUSE_WM[0..1]	RX shared pool pause watermark per SMP	0x4F8 + 0x1*i
CM_SHARED_SMP_USAGE[0..1]	Stores the number of segments in the shared pool for each SMP	0x4FA + 0x1*i
CM_GLOBAL_USAGE	Stores the total number of segments used	0x004FC
CM_GLOBAL_WM	Defines the global watermark	0x004FD
CM_SMP_MEMBERSHIP	Defines the SMP membership per traffic class	0x004FE
CM_TX_HOG_MAP	TX hog map	0x004FF
CN_CPID_MASK	Congestion notification mask	0x00500
CN_VCN_DMAC_2[0..24]	Destination MAC for VCN Frames	0x540 + 0x1*i
CN_RATE_LIM_CPID[0..24] [0..1]	Rate limiter CPID table	0x580 + 0x2*j + 0x1*i
CN_SMP_CFG[0..24] [0..1]	Defines congestion notification parameters per port per SMP	0x5C0 + 0x2*j + 0x1*i
CN_SMP_THRESHOLD[0..24] [0..1]	Defines congestion notification thresholds	0x600 + 0x2*j + 0x1*i
CN_SAMPLE_CFG	Controls CN Byte Interval Sampling	0x00640
CN_RATE_LIM_CFG[0..24]	Congestion Notification Rate Limiter Configuration	0x660 + 0x1*i
CN_CPID_TABLE[0..7]	Congestion notification id table	0x680 + 0x2*i
CN_GLOBAL_CFG_1 *(was CN_GLOBAL_CFG)*	Global configuration for congestion notification (part 1)	0x00692
CN_GLOBAL_CFG_2	Global configuration for congestion notification (part 2)	0x00693
CN_FB_CFG	Congestion notification feedback configuration	0x00694
CN_FORWARD_MASK	Congestion notification forard mask	0x00695
CN_RATE_ACTION_MASK	Congestion notification rate action mask	0x00696
CN_BACKOFF_BYTETIME	Congestion notification backoff byte time	0x00697
CN_FRAME_CFG_1	Congestion notification prefixes	0x00698
CN_FRAME_CFG_2	Congestion notification prefixes	0x00699
CN_VCN_DMAC_1	Destination MAC for VCN Frames	0x0069D
TX_RATE_LIM_CFG[0..24] [0..7]	Token Bucket Capacity for Egress Rate Limiter	0x700 + 0x8*j + 0x1*i
TX_RATE_LIM_USAGE[0..24] [0..7]	Tx rate limiter current usage	0x800 + 0x8*j + 0x1*i
RX_RATE_LIM_CFG[0..24] [0..1]		0x900 + 0x2*j + 0x1*i
RX_RATE_LIM_USAGE[0..24] [0..1]	Rx rate limiter current usage	0x940 + 0x2*j + 0x1*i
RX_RATE_LIM_THRESHOLD[0..24] [0..1]	Rx rate limiter thresholds	0x980 + 0x2*j + 0x1*i
CM_PORT_LIST[0..24]	Congestion Management Port List	0x9C0 + 0x1*i

Name	Brief Description	Address Offset (Base = 0x103000)
SCHED_DRR_Q[0..24] [0..7]	Deficit Round Robin Quantum per port per group	0xA00 + 0x8*j + 0x1*i
SCHED_SHAPING_GROUP_CFG[0..24]	Maps traffic class to bandwidth-sharing traffic-shaping group per port	0xB00 + 0x1*i
SWITCH_PRI_TO_CLASS_1	Maps switch priority to traffic class (pri 0 to 7)	0x00B20
SWITCH_PRI_TO_CLASS_2	Maps switch priority to traffic class (pri 8 to 15)	0x00B21
CM_GLOBAL_CFG		0x00B22
TRAFFIC_CLASS_TO_SCHED_PRI	Traffic class to Scheduler Priority Map	0x00B23
CN_STATS_CFG[0..1]	Congestion notification statistics	0xC00 + 0x1*i
CN_STATS[0..1] [0..7]	Congestion management statistics per port per SMP	0xD00 + 0x8*j + 0x1*i
SCHED_FH_GROUP_CFG[0..24]	Frame Handler Copy of Scheduler Group Configuration	0xF20 + 0x1*i
QDM_CFG[0..1]	Queue Delay Measurement Configuration	0xF40 + 0x1*i
QDM_FRAME_CNT[0..1]	Queue Delay Measurement Configuration	0xF42 + 0x1*i
QDM_INSTRUMENT[0..1]	Queue Delay Measurement Data	0xF44 + 0x1*i

15.2.21 Link Aggregation Registers (reserved 4KW) Map

Table 45: Link Aggregation Registers (reserved 4KW) Map

Name	Brief Description	Address Offset (Base = 0x104000)
LAG_CFG[0..24]	LAG Configuration	0x0 + 0x1*i
CANONICAL_GLORT_CAM[0..15]	Canonical glort lookup CAM	0x20 + 0x1*i

15.2.22 Policer Registers (reserved 16KW) Map

Table 46: Policer Registers (reserved 16KW) Map

Name	Brief Description	Address Offset (Base = 0x10C000)
POLICER_TABLE[0..3] [0..511]		0x0 + 0x800*j + 0x4*i
POLICER_REPAIR[0..3] [0..1]		0x2000 + 0x2*j + 0x1*i
POLICER_CFG[0..3]		0x2008 + 0x1*i
POLICER_IP		0x0200C
POLICER_IM		0x0200D
POLICER_STATUS		0x0200E
POLICER_DSCP_DOWN_MAP[0..63]		0x2100 + 0x1*i
POLICER_SWPRI_DOWN_MAP[0..15]		0x2140 + 0x1*i

15.2.23 Triggers Registers (reserved 4KW) Map

Table 47: Triggers Registers (reserved 4KW) Map

Name	Brief Description	Address Offset (Base = 0x106000)
TRIGGER_CONDITION_CFG[0..63]	General per-trigger match condition configuration.	0x100 + 0x1*i
TRIGGER_CONDITION_PARAM[0..63]	Miscellaneous match condition parameters.	0x140 + 0x1*i
TRIGGER_CONDITION_FFU[0..63]	FFU match parameters.	0x180 + 0x1*i
TRIGGER_CONDITION_TYPE[0..63]	EtherType match parameters.	0x1C0 + 0x1*i
TRIGGER_CONDITION_GLORT[0..63]	Destination glort match parameters.	0x200 + 0x1*i
TRIGGER_CONDITION_RX[0..63]	Source port mask.	0x240 + 0x1*i
TRIGGER_CONDITION_TX[0..63]	Destination port mask.	0x280 + 0x1*i
TRIGGER_CONDITION_AMASK_1[0..63] *(was TRIGGER_CONDITION_AMASK)*	Frame Handler action mask match condition (actions 0 through 31).	0x2C0 + 0x1*i
TRIGGER_CONDITION_AMASK_2[0..63]	Frame Handler action mask match condition (actions 32 through 44).	0x300 + 0x1*i
TRIGGER_ACTION_CFG_1[0..63]	General per-trigger action configuration.	0x340 + 0x1*i
TRIGGER_ACTION_CFG_2[0..63]	General per-trigger action configuration.	0x380 + 0x1*i
TRIGGER_ACTION_GLORT[0..63]	Forwarding action glort assignment.	0x3C0 + 0x1*i
TRIGGER_ACTION_DMASK[0..63]	Forwarding action destination mask assignment.	0x400 + 0x1*i
TRIGGER_ACTION_MIRROR[0..63]	Mirroring action port and glort assignment.	0x440 + 0x1*i
TRIGGER_ACTION_DROP[0..63]	Drop action port mask.	0x480 + 0x1*i
TRIGGER_RATE_LIM_CFG_1[0..15]	Rate limiter configuration (Capacity and Rate).	0x4C0 + 0x1*i
TRIGGER_RATE_LIM_CFG_2[0..15]	Rate limiter configuration (Drop mask).	0x4D0 + 0x1*i
TRIGGER_RATE_LIM_USAGE[0..15]	Rate limiter token-bucket levels (units of 1/16th bytes).	0x580 + 0x1*i
TRIGGER_IP[0..1]	Trigger IP registers	0x4E0 + 0x1*i
TRIGGER_IM[0..1]	Trigger IM registers	0x4E4 + 0x1*i

15.2.24 ARP Control and Table (128KW) Map

Table 48: ARP Control and Table (128KW) Map

Name	Brief Description	Address Offset (Base = 0x120000)
ARP_TABLE[0..16383]	Stores next-hop information for IP routing.	0x0 + 0x4*i
ARP_USED[0..511]	Tracks if hardware has used each ARP entry.	0x10000 + 0x1*i
ARP_IP	ARP Interrupt Pending	0x10201
ARP_IM	ARP Interrupt Mask	0x10202
ARP_REDIRECT_SIP	Redirect source IP	0x10204

Name	Brief Description	Address Offset (Base = 0x120000)
ARP_REDIRECT_DIP	Redirect destination IP	0x10208

15.2.25 Filtering and Forwarding Unit Registers (256KW address space) Map

Table 49: Filtering and Forwarding Unit Registers (256KW address space) Map

Name	Brief Description	Address Offset (Base = 0x140000)
FFU_MAP_SRC[0..24]	Configures source port to 4-bit mapping	0x3B000 + 0x1*i
FFU_MAP_MAC[0..15]	Configures 48-bit to 4-bit MAC address mapping	0x3B100 + 0x2*i
FFU_MAP_VLAN_REPAIR	Repair address for VLAN sram	0x3B180
FFU_MAP_VLAN[0..4095]	Configures 12-bit to 12-bit Vlan mapping	0x3C000 + 0x1*i
FFU_MAP_TYPE[0..15]	Configures 16-bit to 4-bit EtherType mapping	0x3B200 + 0x1*i
FFU_MAP_LENGTH[0..15]	Configures 16-bit to 4-bit IP Length mapping	0x3B300 + 0x1*i
FFU_MAP_IP_LO[0..15]	Configures low 64-bit key of 128-bit to 4-bit IP address mapping	0x3B400 + 0x2*i
FFU_MAP_IP_HI[0..15]	Configures high 64-bit key of 128-bit to 4-bit IP address mapping	0x3B420 + 0x2*i
FFU_MAP_IP_CFG[0..15]	Configures 128-bit to 4-bit IP address mapping	0x3B440 + 0x1*i
FFU_MAP_PROT[0..7]	Configures 8-bit to 3-bit L4 Protocol mapping	0x3B500 + 0x1*i
FFU_MAP_L4_SRC[0..63]	Configures 16-bit to 4-bit L4 source port mapping	0x3B600 + 0x2*i
FFU_MAP_L4_DST[0..63]	Configures 16-bit to 4-bit L4 destination port mapping	0x3B700 + 0x2*i
FFU_INIT_SLICE	Configures FFU slices at power up. The unit contains 33 physical slices and this register is used to select which of those are going to be actually used. The hardware will activate up to ActiveSlices slices and will skip over repair1 and repair2 slices.	0x3B800
FFU_MASTER_VALID	Atomically set validity of all FFU slices and egress ACL chunks	0x3A000
FFU_EGRESS_CHUNK_CFG[0..31]	Configures 16-rule chunks of egress ACLs	0x3D000 + 0x1*i
FFU_EGRESS_CHUNK_VALID[0..31]	Configures valid scenarios for 16-rule chunks of egress ACLs	0x3D100 + 0x1*i
FFU_EGRESS_ACTIONS[0..511]	Configures the egress ACL actions of each rule	0x3D200 + 0x1*i
FFU_SLICE_TCAM[0..31] [0..511]	Configures an entry in a slice's key TCAM	0x0 + 0x1000*j + 0x4*i
FFU_SLICE_SRAM[0..31] [0..511]	Configures an entry in a slice's action SRAM	0x800 + 0x1000*j + 0x2*i
FFU_SLICE_VALID[0..31]	Configures which scenarios are valid in each slice.	0xC00 + 0x1000*i
FFU_SLICE_CASE[0..31]	Configures which cases are used per scenario in each slice	0xC02 + 0x1000*i
FFU_SLICE_CASCADE_ACTION[0..31]	Configures weather an action cascade continues per scenario in each slice	0xC04 + 0x1000*i

Name	Brief Description	Address Offset (Base = 0x140000)
FFU_SLICE_CASE_CFG[0..31] [0..1]	Configures slice behavior in each of two cases	0xC08 + 0x1000*j + 0x1*i

15.2.26 VLAN Packet Counters (Group 11) Map

Note: Pause frames are not included in this counter grouop.

Table 50: VLAN Packet Counters (Group 11) Map

Name	Brief Description	Address Offset (Base = 0x1A0100)
STAT_VlanUcstPkts[0..63]	Count of total unicast frames received with ingress VLANs mapping to this CounterIndex (see INGRESS_VID_TABLE).	0x0 + 0x2*i
STAT_VlanXcstPkts[0..63]	Count of total non-unicast (multicast or broadcast) frames received with ingress VLANs mapping to this CounterIndex (see INGRESS_VID_TABLE).	0x80 + 0x2*i

15.2.27 VLAN Octet Counters (Group 12) Map

Note: Pause frames are not included in this counter grouop.

Table 51: VLAN Octet Counters (Group 12) Map

Name	Brief Description	Address Offset (Base = 0x1A0200)
STAT_VlanUcstOctets[0..63]	Count of total unicast octets received from frames with ingress VLANs mapping to this CounterIndex (see INGRESS_VID_TABLE).	0x0 + 0x2*i
STAT_VlanXcstOctets[0..63]	Count of total non-unicast (multicast or broadcast) octets received from frames with ingress VLANs mapping to this CounterIndex (see INGRESS_VID_TABLE).	0x80 + 0x2*i

15.2.28 Trigger Counters (Group 13) Map

Table 52: Trigger Counters (Group 13) Map

Name	Brief Description	Address Offset (Base = 0x1A0300)
STAT_Trigger[0..63]	Number of times an action of each trigger was applied to a frame. Does not include triggers that fire but are overridden by higher-precedence triggers.	0x0 + 0x2*i

15.2.29 Egress ACL Counters (Group 14) Map

Table 53: Egress ACL Counters (Group 14) Map

Name	Brief Description	Address Offset (Base = 0x1A0400)
STAT_EgressACLPkts[0..24]	Number of egress frames on each port counted by an egress ACL action.	0x0 + 0x2*i
STAT_EgressACLOctets[0..24]	Number of egress bytes on each port counted by an egress ACL action.	0x80 + 0x2*i

15.2.30 Statistics Control Registers Map

Table 54: Statistics Control Registers Map

Name	Brief Description	Address Offset (Base = 0x1A0500)
STATS_CFG[0..24]	Per-port Statistics Counter Configuration	0x0 + 0x2*i
STATS_DROP_COUNT_RX[0..24]	Number of times counter updates to groups 1 through 6 (RX counters) were dropped due to insufficient counter bandwidth.	0x40 + 0x2*i
STATS_DROP_COUNT_TX[0..24]	Number of times counter updates to groups 7 through 9 (TX counters) were dropped due to insufficient counter bandwidth.	0x80 + 0x2*i

15.2.31 Layer 2 Lookup Registers (reserved 128KW) Map

Table 55: Layer 2 Lookup Registers (reserved 128KW) Map

Name	Brief Description	Address Offset (Base = 0x180000)
MA_TABLE[0..16383]	MAC address forwarding table	0x0 + 0x4*i
INGRESS_VID_TABLE[0..4095] *(was VID_TABLE)*	Ingress VLAN table	0x10000 + 0x4*i
EGRESS_VID_TABLE[0..4095]	Egress VLAN table	0x14000 + 0x2*i
INGRESS_FID_TABLE[0..4095] *(was FID_TABLE)*	Ingress spanning-tree forwarding state table	0x16000 + 0x2*i
EGRESS_FID_TABLE[0..4095]	Ingress spanning-tree forwarding state table	0x18000 + 0x1*i
MA_TCN_FIFO[0..511]	MAC Table Change Notification FIFO	0x1E000 + 0x4*i
MA_TCN_PTR	MAC Table Change Pointers	0x1E800
MA_TCN_IP	MAC Table Change Notification Interrupt Pending	0x1E801
MA_TCN_IM	MAC Table Change Notification Interrupt Pending	0x1E802
MA_TABLE_STATUS_3	MAC table status 3	0x1E803
MA_TABLE_CFG_1	Hashing, partitioning, and rates	0x1E804

Name	Brief Description	Address Offset (Base = 0x180000)
MA_TABLE_CFG_2	GloRTs for flooding, broadcast, and multicast	0x1E805
MA_TABLE_CFG_3	Aging and learning	0x1E806
MA_TCN_CFG_1	MAC table change notification configuration	0x1E807
MA_TCN_CFG_2	MAC table change notification configuration	0x1E808
MA_PURGE	MA Table purge command	0x1E809
MTU_TABLE[0..7]	MA Table frame watermark	0x1E810 + 0x1*i

15.3 HSM Registers Details

15.3.1 LCI_CFG

Table 56: LCI_CFG

Name	Bit	Description	Type	Default
endianness	3	Controls the byte ordering for the packet payload when passed to the host. If set to 0, then the byte ordering is little endian. If set to 1, then the byte ordering is big endian.	RW	0x0
hostPadding	4	Controls padding frame size to an even number of words (32bits) or not. If HostPadding is set to '1' and if the number of words in the 32-bit aligned payload is odd, an extra word at the end of the frame will be transfered. The only purpose of this 'hostPadding-controlled' padding is to make the 32-bit aligned payload become a 64-bit aligned payload. Setting HostPadding to '0' will not add an extra word (32bits of '0') to make the payload 64-bit aligned if the payload is not 64-bit aligned already. However, the payload will still always be 32-bit aligned by padding 1~3 bytes if necessary.	RW	0x0

15.3.2 LCI_RX_FIFO

Table 57: LCI_RX_FIFO

Name	Bit	Description	Type	Default
all	31: 0	Receive packet data FIFO. Sucessive reads to this register returns the packet payload until queue is empty. Last word contains length of the packet.	RO	0x0

15.3.3 LCI_TX_FIFO

Table 58: LCI_TX_FIFO

Name	Bit	Description	Type	Default
all	31: 0	Transmit packet data FIFO. The first write is the packet length and the successive writes to this register is the packet payload.	RW	0x0

15.3.4 LCI_IP

Table 59: LCI_IP

Name	Bit	Description	Type	Default
newFrameRecv	1	Indicates if a new frame is received in the LCI_RX_FIFO.	CW1	0x0
endOfFrameSend	2	Indicates the end of a packet transfer to the CPU.	CW1	0x0

Writing '1' into any bit clears the corresponding bit, writing 0 has no effect.

15.3.5 LCI_IM

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 60: LCI_IM

Name	Bit	Description	Type	Default
newFrameRecv	1	New frame receive interrupt mask.	RW	0x1
endOfFrameSend	2	Frame transmitted interrupt mask.	RW	0x1

15.3.6 LCI_STATUS

Table 61: LCI_STATUS

Name	Bit	Description	Type	Default
txReady	0	Indicates the status of the transmit FIFO. If 1, then the transmit FIFO is not full. If 0, then the transmit FIFO is full and the user should not send any more data. This status is reflected and inverted on the pin TXRDY_N.	RO	0x1
rxReady	1	Indicates the status of the receive FIFO. If 1, then the receive FIFO contains data. If 0, then the receive FIFO is empty. This status is reflected and inverted on the pin RXRDY_N.	RO	0x0

Name	Bit	Description	Type	Default
rxEndOfFrame	2	Indicates if the current word is the last word of a receive packet. The last word contains the length of the packet and the CRC status. This status is reflected on RXEOT_N but only while the receive FIFO read is acknowledged.	RO	0x0
testAndSetLock	3	Indicates if the port is currently busy transmitting a packet from congestion notification or in-band management or not. This is for information purpose only.	RO	0x0
pendingTXwords	15: 4	For debugging purposes only. Reports the number of words minus 1, that still need to be written to LCI_TX_FIFO in order to complete a pending frame transmission. Only guaranteed to be correct after a write to LXI_TX_FIFO. This is for information only.	RO	0x0

15.3.7 LAST_FATAL_CODE

Table 62: LAST_FATAL_CODE

Name	Bit	Description	Type	Default
FatalCode	7: 0	Contains the interrupt source for the last fatal interrupt processed.	CW	0x0

The following lists all prefixes:

- 0xC0: FH forwarding table parity error
- 0x22: EPL parity error
- 0x20: Scheduler parity error (RxQueueLink)r
- 0x21: Scheduler parity error (TxQueueLink)r
- 0x41: MSB; IBM CRC error
- 0x42: MSB; Invalid IBM op error
- 0x43: Both of the above errors

15.3.8 INTERRUPT_DETECT

Table 63: INTERRUPT_DETECT

Name	Bit	Description	Type	Default
EPL	0	EPL interrupt pending. Read register GLOBAL_EPL_INT_DETECT to determine which ports have generated interrupts.	RO	0x0
SOFT	1	Software interrupt pending. See SW_IP and SW_IM for the source of the interrupt.	RO	0x0
PERR	2	Scheduler memory parity error pending. See PERR_IP and PERR_IM for the source of the interrupt.	RO	0x0

Name	Bit	Description	Type	Default
GPIO	3	GPIO interrupt pending. See GPIO_IP and GPIO_IM for the source of the interrupt.	RO	0x0
I2C	4	I2C interrupt pending. Write a '1' into I2C_CTRL.InterruptEnable to clear the interrupt.	RO	0x0
MDIO	5	MDIO interrupt pending. Write a '1' into MDIO_CTRL.InterruptEnable to clear the interrupt.	RO	0x0
CRM	6	Counter Rate Monitor interrupt pending. See CRM_INT_DETECT for the CRM that cause the interrupt.	RO	0x0
MSB	9	Port 0 MAC interrupt pending. See MSB_IP and MSB_IM to determine the source of the interrupt.	RO	0x0
VLAN_Egress	10	VLAN egress boundary violation interrupt pending.	RO	0x0
VLAN_Ingress	11	VLAN ingress boundary violation interrupt pending.	RO	0x0
Security	12	Security violation interrupt pending.	RO	0x0
Trigger	13	Trigger interrupt pending.	RO	0x0
PERR_FrameHandler	14	Memory parity error pending in Frame Handler.	RO	0x0
TCN	15	MA Table Change Notification interrupt pending.	RO	0x0
CM	16	Obsolete.	RO	0x0
ARP	17	ARP interrupt pending. Does not apply to FM3000 devices.	RO	0x0
POLICER	18	POLICER interrupt pending.	RO	0x0
RxFIFO	20	RxFIFO (from switch to CPU) has an interrupt pending.	RO	0x0
TxFIFO	21	TxFIFO (from CPU to switch) has an interrupt pending.	RO	0x0
GlobalInterruptMask	31	Global Interrupt Mask. If set to 1, then the interrupt is de-asserted even if there are active interrupts logged in this register.	RW	0x0

15.3.9 FATAL_COUNT

Table 64: FATAL_COUNT

Name	Bit	Description	Type	Default
ResetCount	7: 0	Counts the number of fatal interrupt reset that occured since last time the chip was resetted. This register could be cleared via the SOFT_RESET register.	CW	0x0

15.3.10 SOFT_RESET

Table 65: SOFT_RESET

Name	Bit	Description	Type	Default
SwitchReset	0	Force a reset of the entire chip minus the watchdog. This bit will self clear after 256 cycles.	RW	0x0
CoreReset	1	Force the reset of the asynchronous block. This bit will self clear after 256 cycles.	RW	0x0
MgmtReset	2	Force the reset of the LSM. This bit will self clear after 256 cycles.	RW	0x0
FH_Reset	3	Controls reset on the Frame Handler. This bit is not self clear and must be deasserted by the software.	RW	0x1

15.3.11 WATCHDOG_CFG

Table 66: WATCHDOG_CFG

Name	Bit	Description	Type	Default
Enable	0	Defines if the watchfog is enabled or not.	RW	0x0

15.3.12 PLL_FH_STAT

Table 67: PLL_FH_STAT

Name	Bit	Description	Type	Default
locked	0	Indicates if the PLL has locked or not.	RO	0x0

15.3.13 PLL_FH_CTRL

Table 68: PLL_FH_CTRL

Name	Bit	Description	Type	Default
bypass	0	Enables bypassing the PLL or not. If set to 1, then the FH_REFCLK is directly routed to the frame handler clock bypassing the PLL output. If set to 0, then the PLL oupout is used.	RW	0x0
powerDown	1	Enables PLL or not. If set to 0, then the PLL is active and will try to lock. If set to 1, then the PLL is powered down and will not produce any reference clock to the frame handler.	RW	0x1
P	3: 2	Post divider. The options are: • 0x0: P=1 • 0x1: P=2	RW	0x0

Name	Bit	Description	Type	Default
		• 0x2: P=4 • 0x3: P=8		
M	10: 4	M divider. (Note: setting to 0 causes divider to be 128.)	RW	0x14
N	14: 11	N divider. (Note: setting to 0 causes divider to be 16.)	RW	0x4
PLLoutEnable	15	Enable PLL output or not. If set to 1, then the PLL output is reproduced on the FH_CLKOUT pin. If set to 0, then the PLL output is not reproduced on theFH_CLKOUT and the pin otput is constant (1 or 0). This register doesn't affect the routing of the PLL output to the Frame Handler, only to the external pin.	RW	0x0

15.3.14 VITAL_PRODUCT_DATA

Table 69: VITAL_PRODUCT_DATA

Name	Bit	Description	Type	Default
PartNumber	27: 12	The part number.	RO	0xae19

15.4 LSM Registers Details

15.4.1 LSM_INT_DETECT

Table 70: LSM_INT_DETECT

Name	Bit	Description	Type	Default
SW	0	See SW_IP	RO	0x0
PERR	1	PERR_IP	RO	0x0
GPIO	2	See GPIO_IP	RO	0x0
I2C	3	See I2C registers	RO	0x0
MDIO	4	See MDIO registers	RO	0x0
CRM	5	See CRM_INT_DETECT	RO	0x0

15.4.2 GLOBAL_EPL_INT_DETECT

Table 71: GLOBAL_EPL_INT_DETECT

Name	Bit	Description	Type	Default
port1	1		RO	0x0
port2	2		RO	0x0

Name	Bit	Description	Type	Default
port3	3		RO	0x0
port4	4		RO	0x0
port5	5		RO	0x0
port6	6		RO	0x0
port7	7		RO	0x0
port8	8		RO	0x0
port9	9		RO	0x0
port10	10		RO	0x0
port11	11		RO	0x0
port12	12		RO	0x0
port13	13		RO	0x0
port14	14		RO	0x0
port15	15		RO	0x0
port16	16		RO	0x0
port17	17		RO	0x0
port18	18		RO	0x0
port19	19		RO	0x0
port20	20		RO	0x0
port21	21		RO	0x0
port22	22		RO	0x0
port23	23		RO	0x0

15.4.3 PERR_IP

Parity error detection in the scheduler tables. There are four categories;

- **Transient:** these errors result in data corruption but there is no permanent harm done to the switch and there is no need to reset
- **Cumulative:** more severe than transient, these will result in a bounded (one jumbo frame or less) amount of memory being leaked. It is acceptable to wait for several of these errors to happen before reseting the switch
- **Fatal:**Result in the switch being non-functional in some way and the switch should be reset immediately.
- **User fixable:**Result in the switch being non-functional in some way, but the software is able to detect the error and repair it.

Table 72: PERR_IP

Name	Bit	Description	Type	Default
PacketHeaderParityError	0	Transient error.	CW1	0x0
RxQueuePointerError	1	Transient error.	CW1	0x0
MTableTransientError	2	Transient error.	CW1	0x0

Name	Bit	Description	Type	Default
Reserved	3		CW1	0x0
FreeListError	4	Cumulative Error.	CW1	0x0
ReferenceCountError	5	Cumulative Error.	CW1	0x0
HeadPolarityError	6	Cumulative Error.	CW1	0x0
TxQueueError	7	Cumulative Error.	CW1	0x0
TxQueueFreeList	8	Cumulative Error.	CW1	0x0
SubSegmentError	9	Cumulative Error.	CW1	0x0
MTableCumulativeError	10	Cumulative Error.	CW1	0x0
Reserved	11		CW1	0x0
RxQueueLink	12	Fatal Error.	CW1	0x0
TxQueueLink	13	Fatal Error.	CW1	0x0
Reserved	14		CW1	0x0
MTableTableError	15	User fixable error.	CW1	0x0

Writing '1' into any bit clears the corresponding bit, writing 0 has no effect.

15.4.4 PERR_IM

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 73: PERR_IM

Name	Bit	Description	Type	Default
mask	15: 0		RW	0xffff

15.4.5 SW_IP

Table 74: SW_IP

Name	Bit	Description	Type	Default
SoftIP	31: 0		RW	0x0

15.4.6 SW_IM

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 75: SW_IM

Name	Bit	Description	Type	Default
SoftIM	31: 0		RW	0xffffffff

15.4.7 FRAME_TIME_OUT

Frame timeout duration.

Table 76: FRAME_TIME_OUT

Name	Bit	Description	Type	Default
timeOutMult	27: 0	Defines the frame time out duration. Writing a new value takes effect immediately and writing a 1 causes an immediate timeout pulse to be sent to the scheduler. • timeOutMult > 0: A frame will timeout in 1024 * timeOutMult * LSM clock period. (LSM clock period is 1/2 the CPU clock period) • timeOutMult = 0: Frame timeout disabled.	RW	0x88b8

15.4.8 BOOT_CTRL

****** This register was named CHIP_MODE ******

Table 77: BOOT_CTRL

Name	Bit	Description	Type	Default
Command	3: 0	Available commands: • 0: Initialize scheduler and perform freelist repair • 1: Async memory repair • 2: Memory intialization • 3: Sync memory repair The commands shall normally be executed in the following order: 1, 0, 3 and then 2.	RW	0x0

15.4.9 BOOT_STATUS

Table 78: BOOT_STATUS

Name	Bit	Description	Type	Default
CommandDone	0	Indicates when the command placed in the BOOT_CTRL register has bene executed. This bit is cleared after reset or when the command is started.	RO	0x0

15.4.10 CLK_MULT_1

Table 79: CLK_MULT_1

Name	Bit	Description	Type	Default
SPIMult	7: 0		RW	0x19

15.4.11 VPD_INFO_1

Table 80: VPD_INFO_1

Name	Bit	Description	Type	Default
PortDisableMask	24: 1	Indicates which ports are offered on this chip. If bit # (port#) is set to 1, then the corresponding port is disabled (reset will always be asserted). If set to 0, this port is present.	RW	0x0
Package	31: 25	Indicates package version. • 0: FM4224 • 1: FM4112	RW	0x0
MaskVersion	39: 32	Indicates mask version. Value of 0 means that the mask version has not been programmed yet. If bit[39] is set to '1', the mask version from efuse will be used, otherwise, the mask version from hard macro CHIP_VERSION will be used.	RW	0x0

15.4.12 VPD_INFO_2

Table 81: VPD_INFO_2

Name	Bit	Description	Type	Default
jtagID	10: 0		RO	0x215

15.4.13 GPIO_CFG

Table 82: GPIO_CFG

Name	Bit	Description	Type	Default
Dir	15: 0	Defines the direction of each GPIO pin, 1=output, 0=input.	RW	0x0
OpenDrain	31: 16	Defines if the pin is configured as open drain or normal output pin. This configuratin has no effect if the pin is configured as an input	RW	0x0

15.4.14 GPIO_DATA

Table 83: GPIO_DATA

Name	Bit	Description	Type	Default
data	15: 0	GPIO pin state. Note that writing causes the value to be latched into a 16-bit latch while reading returns the actual pin state. If the pin is configured as open drain, then writing a 1 causes the pin to not be driven while writing a 0 causes the pin to be driven to ground.	RW	0x0
reserved	31: 16		RV	0x0

15.4.15 GPIO_IP

Table 84: GPIO_IP

Name	Bit	Description	Type	Default
detectHigh	15: 0	Indicates if the pin has changed its value from 0 to 1 (1) or not (0). Writing a '1' into this register on the corresponding GPIO pin will clear the interrupt.	CW1	0x0
detectLow	31: 16	Indicates if the pin has changed its value from 1 to 0 (1) or not (0). Writing a '1' into this register on the corresponding GPIO pin will clear the interrupt.	CW1	0x0

Writing '1' into any bit clears the corresponding bit. writing 0 has no effect.

15.4.16 GPIO_IM

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 85: GPIO_IM

Name	Bit	Description	Type	Default
detectHighMask	15: 0	Indicates if the a transition from 0 to 1 should cause an interrupt (0) or should be masked (1).	RW	0xffff
detectLowMask	31: 16	Indicates if the a transition from 1 to 0 should cause an interrupt (0) or should be masked (1).	RW	0xffff

15.4.17 I2C_CFG

Table 86: I2C_CFG

Name	Bit	Description	Type	Default
Enable	0	Enable	RW	0x1

Name	Bit	Description	Type	Default
		• 0: I2C will not answer to I2C commands from other masters • 1: I2C will answer to I2C commands from other masters targetting the address The default value is active. Note that if the chip is configured to retrieve its configuration from serial EEPROM, then the switch will not answer to I2C commands from other masters until the switch has recovered its configuration. In this case the default value could be overwritten by the configuration PROM.		
Addr	7: 1	Defines the I2C address of the switch on a data bus. The default configuration is '1000xxx' where xxx is defined by configuration pins.	RW	0x40
Divider	19: 8	I2C Clock Divider. The I2C clock is equal to CPU_CLK/10/DIVIDER.	RW	0x100
InterruptMask *(was InterruptEnable)*	24	Defines if the I2C interrupt is masked (1) or passed (0) to the interrupt hierarchy.	RW	0x1
DebounceFilterCountLimit	31: 25	This field is to control the debouncing filter circuit.	RW	0x14

15.4.18 I2C_DATA[0..1]

Table 87: I2C_DATA[0..1]

Name	Bit	Description	Type	Default
Data	31: 0	Bytes to be sent out or read from the I2C bus depending on the command type: read, write or write-read. Bytes are sent from, or received to, the most significant byte of the most significant word first. I.e. I2C_DATA[1].Byte3 is used first followed by I2C_DATA[1].Byte2 etc.	RW	0x0

15.4.19 I2C_CTRL

Table 88: I2C_CTRL

Name	Bit	Description	Type	Default
Addr	7: 0	Device to address. Bit 0 is not used and will be completed by the command selected	RW	0x0
Command	9: 8	Command type. The commands are: • 0: Null. • 1: Write. • 2: Write then read. • 3: Read.	RW	0x0

Name	Bit	Description	Type	Default
		Note that a new command starts execution only when this field is changed from 0 to one of the possible commands.		
LengthW	12: 10	Specifies number of bytes to write to the I2C bus for a 'write' or 'write-read', 0 indicates 8 bytes. Sends most significant byte of I2C_DATA[1] first. For a 'write-read', this specifies the number of bytes in the write part of the sequence. Also LengthW and LengthR must not specify more than 8 bytes together for the 'write-read' command otherwise the behavior is undefined.	RW	0x0
LengthR	15: 13	Specifies number of bytes to send for a 'read' or 'write-read', 0 indicates 8 bytes. Sending most significant byte of I2C_DATA[1] first. For a 'write-read', this specifies the number of bytes in the read part of the sequence. Also LengthW and LengthR must not specify more than 8 bytes together for the 'write-read' command otherwise the behavior is undefined.	RW	0x0
LengthSent	18: 16	Indicates the number of bytes sent before a NAK was received. If the targeted device has returned ACK on all bytes sent, then this field will be the number of bytes sent. A value of 0 indicates 8 bytes.	RO	0x0
CommandCompleted	22: 19	Indicates the command status. This field is automatically cleared when the command is started. The codes returned are listed below • 0: running; the command still running or the command is null. • 1: terminated normally; all bytes requested were transfered. • 2: terminated prematurely; the targeted device returned a NAK before all data were sent. • 3: no device; command has terminated prematurely because there was no ACK at the address phase. • 4: timeout: the attached device asserted the clock for longer than 2^30 LSM cycles. • 5: lost arbitration: Arbitration lost to another I2C master. • 6: wait for bus: Switch is waiting for bus. • F: Invalid.	RO	0xF
InterruptPending	23	Defines if a command has completed or not. Writing a '1' clears the interrupt.	CW1	0x0

15.4.20 MDIO_CFG

Table 89: MDIO_CFG

Name	Bit	Description	Type	Default
Divider	11: 0	MDIO Clock Divider. The MDIO clock is equal to CPU_CLK/2/DIVIDER.	RW	0x10
Preamble	12	Defines if the preamble (32 cycles of 1s) is sent (1) or not (0). The default is to send the preamble. Faster access is possible by canceling transmission of preamble.	RW	0x1
InterruptMask *(was InterruptEnable)*	13	Defines if the I2C interrupt is masked (1) or passed (0) to the interrupt hierarchy.	RW	0x1
Reserved	31: 14		RV	0x0

15.4.21 MDIO_DATA

Table 90: MDIO_DATA

Name	Bit	Description	Type	Default
Data	15: 0	Data register. For the command 'read', the content of this register is only updated if the command completed successfully. For the command 'write', the content of the register will be sent to the device (MSB first) and will not be changed once the command completed.	RW	0x0
Reserved	31: 16		RV	0x0

15.4.22 MDIO_CTRL

Table 91: MDIO_CTRL

Name	Bit	Description	Type	Default
Register	15: 0	Register to address.	RW	0x0
Device	20: 16	Sub-device to address.	RW	0x0
PhyAddress	25: 21	Physical device to address.	RW	0x0
Command	27: 26	Command type. The commands are: • 0: Null. • 1: Write. The address frame is always sent. • 2: Sequential-Read. The address frame is not sent and the read frame will cause the next register to be read.	RW	0x0

Name	Bit	Description	Type	Default
		• 3: Random-Read. The address is sent before reading. Note that a new command is starts execution only when this field is changed from 0 to one of the possible commands.		
DeviceType	28	Defines the device type • 0: Clause 22. The MDIO frame structure is for legacy devices and does not contain a 'device' field and the register address is only 5 bits. • 1: Clause 45. The MDIO frame structure is for high-speed devices and contains a 'device' field and the register address is 16 bits.	RW	0x0
Status	30: 29	Indicates the command status. This field is automatically cleared when the command is started. The codes returned are listed below • 0: running; the command still running • 1: terminated normally; all bytes requested were transfered • 2: no device; command has terminated prematurely because there was no ACK at the address phase (only read or write-read may have this status)	RO	0x0
InterruptPending	31	Indicates if a command has been completed. Writing '1' clears the interrupt.	CW1	0x0

15.4.23 LED_CFG

This register defines the behavior of the LED interface.

Table 92: LED_CFG

Name	Bit	Description	Type	Default
LEDFreq	15: 0	Divider of CPU_CLK/256 to produce LED_CLK	RW	0x0
LEDEnable	16	Indicates if LED's are enabled or not.	RW	0x0
LEDInvert	17	Defines polarity of LED_DATA pins (0=normal, 1=invert).	RW	0x0
LEDMode	18	Defines the information encoding method the 3 LED signals produce for each port.	RW	0x0

15.4.24 EPL_PORT_CTRL[0..24]

Controls the port reset and clocking selection for each port. Note that PORT #0 controls the CPU port (MSB).

Table 93: EPL_PORT_CTRL[0..24]

Name	Bit	Description	Type	Default
ResetN	0	Defines if the port is placed in reset mode (0) or active (1).	RW	0x0
SinkMgmtReq *(was BorderResetN)*	1	Defines if the management request to this port should be discarded by the EPL management node. Set this to 1 for all EPLs in reset to enable broadcast writes to the EPLs over the EPL management ring. Otherwise broadcast writes to PORT 31 addresses will be unrolled by the LSM and sent as individual port writes.	RW	0x0
ClockSelect	2	Defines which clock to use for this port; (0) REFCLKA, (1) REFCLKB. Note that this bit is not used for the CPU port.	RW	0x0
InitializeN	3	Indicates whether the EPL should perform datapath initialization. When set to 0, initialization is performed when leaving reset. If the EPL is reset during normal chip operation, setting this bit to 1 before leaving reset will suppress initialization.	RW	0x0

15.4.25 CRM_CFG_COUNTER[0..255]

Table 94: CRM_CFG_COUNTER[0..255]

Name	Bit	Description	Type	Default
Enabled	0	Monitor is disabled when set to 0.	RW	0x0
CounterSize	2: 1	Selects the target counter size: • 0 - 32 bits • 1 - 16 bits • 2 - 8 bits • 3 - Reserved	RW	0x0
CounterAddress	24: 3	Address of the counter to monitor. Can be any address register visible to software. In the case of 64-bit counters, this should be the address of the least significant word.	RW	0x0

15.4.26 CRM_CFG_WINDOW[0..255]

Table 95: CRM_CFG_WINDOW[0..255]

Name	Bit	Description	Type	Default
RateWindow	23: 0	Window over which RateLimit is defined, in units of PollingPeriod. A value of 0 indicates an infinite window.	RW	0x0

15.4.27 CRM_CFG_LIMIT[0..255]

Table 96: CRM_CFG_LIMIT[0..255]

Name	Bit	Description	Type	Default
RateLimit	31: 0	Limit imposed on the specified counter's amount of increase over any given RateWindow period of time.	RW	0x0

15.4.28 CRM_LAST_COUNT[0..255]

Table 97: CRM_LAST_COUNT[0..255]

Name	Bit	Description	Type	Default
LastCount	31: 0	Last polled counter value. Cleared to zero when the monitor's Enable bit is set to 0.	RO	0x0

15.4.29 CRM_EXCEED_COUNT[0..255]

Table 98: CRM_EXCEED_COUNT[0..255]

Name	Bit	Description	Type	Default
ExceedCount	31: 0	Number of windows over which the monitor's RateLimit was exceeded. To clear this bit: • Disable the CRM counter. • Wait for the internal state to be updated. • Read exceedCount. If not zero return to the Wait step.	CW	0x0

15.4.30 CRM_CFG

Table 99: CRM_CFG

Name	Bit	Description	Type	Default
PollingPeriod	15: 0	Number of LSM clock cycles between counter samples, in multiples of 1024.	RW	0x1

15.4.31 CRM_INT_DETECT

Table 100: CRM_INT_DETECT

Name	Bit	Description	Type	Default
IntDetect	7: 0	Indicates that an interrupt in the corresponding CRM_IP register is pending and unmasked.	RO	0x0

15.4.32 CRM_IP[0..7]

Table 101: CRM_IP[0..7]

Name	Bit	Description	Type	Default
InterruptPending	31: 0	Each bit is set to 1 when the corresponding counter monitor's ExceedCount value increments.	CW1	0x0

Writing '1' into any bit clears the corresponding bit, writing 0 has no effect.

15.4.33 CRM_IM[0..7]

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 102: CRM_IM[0..7]

Name	Bit	Description	Type	Default
InterruptMask	31: 0	Interrupt mask bits corresponding to CRM_IP pending bits.	RW	0xffffffff

15.5 Ethernet Port Logic Registers (reserved 25 x 1K => 25KW) Details

15.5.1 SERDES_CTRL_1[0..24]

Table 103: SERDES_CTRL_1[0..24]

Name	Bit	Description	Type	Default
DTX_LaneA	3: 0	Current drive for lane A.	RW	0x0
DTX_LaneB	7: 4	Current drive for lane B.	RW	0x0
DTX_LaneC	11: 8	Current drive for lane C.	RW	0x0
DTX_LaneD	15: 12	Current drive for lane D.	RW	0x0
DEQ_LaneA	19: 16	Emphasis for lane A.	RW	0x0
DEQ_LaneB	23: 20	Emphasis for lane B.	RW	0x0
DEQ_LaneC	27: 24	Emphasis for lane C.	RW	0x0
DEQ_LaneD	31: 28	Emphasis for lane D.	RW	0x0

Current drive is determined by the Dtx setting and the nominal current which is given by hi/lo drive selection in SERDES_CTRL_2
TABLE: values for Dtx and Deq encodings.

value[3:0]	Dtx[3:0] - Actual/Nominal Current	Deq[3:0] - equilization
0x0	1.00	0.00
0x1	1.05	0.04
0x2	1.10	0.08
0x3	1.15	0.12
0x4	1.20	0.16
0x5	1.25	0.20
0x6	1.30	0.24
0x7	1.35	0.28
0x8	0.60	0.32
0x9	0.65	0.36
0xA	0.70	0.40
0xB	0.75	0.44
0xC	0.80	0.48
0xD	0.85	0.52
0xE	0.90	0.60
0xF	0.95	0.65

15.5.2 SERDES_CTRL_2[0..24]

Table 104: SERDES_CTRL_2[0..24]

Name	Bit	Description	Type	Default
LowDrive	3: 0	Used to select nominal current see table below 1 bit per lane	RW	0x0
HighDrive	7: 4	Used to select nominal current See table below 1 bit per lane	RW	0x0
LaneReset	11: 8	Reset of individual Serdes lanes. 1 bit per lane.	RW	0xf
LanePowerDown	15: 12	Power Down of individual Serdes lanes. 1 bit per lane. Note: The FM2224 operates in 4 lane or 1 lane modes only. In the one lane mode, only lane 0 or lane 3 will be enabled.	RW	0xf
PLLResetAB	16	PLL reset of the PLL that covers lanes A and B.	RW	0x1
PLLResetCD	17	PLL reset of the PLL that covers lanes C and D.	RW	0x1
RX_PolarityReversal	21: 18	RX Reverse polarity per Serdes lane 0 - no polarity reversal 1 - reverse polarity	RW	0x0

Name	Bit	Description	Type	Default
TX_PolarityReversal	25: 22	TX Reverse polarity per Serdes lane 0 - no polarity reversal 1 - reverse polarity	RW	0x0
LookForAllCommas	26	Enable Comma Detect on not just K28.5 but also K28.1 and K28.7	RW	0x0

- All 4-bit fields are per lane where the least significant bit is for lane A and most significant bit is for lane D.
- In 1 lane mode only lane A (or lane D if one inverts lanes) needs to be powered-on and setup
- Changing any of these register values will require time for the Serdes to stabalize and for normal operation to start.
- The 2 bit number constructed from 1 bit per lane of the Low Drive field and one bit per lane of the High Drive field is used to encode the nominal drive current, according to the table below

TABLE: Hi/Lo Drive encoding

HiDrv	LoDrv	Nominal Driver Current
0x0	1.00	0.00
0	0	20mA
0	1	10mA
1	0	28mA
1	1	Reserved

15.5.3 SERDES_CTRL_3[0..24]

Table 105: SERDES_CTRL_3[0..24]

Name	Bit	Description	Type	Default
DeassertionCount	19: 0	Once the chip has detected a signal on all active lanes, this field specifies how many clock cycles the signal must remain valid before the chip will assert the SignalDetect bit in the SERDES_STATUS register. When signal is lost on one or more lanes, this is also the number of clock cycles the chip will wait before de-asserting that bit.	RW	0x1312D
LinkCount	31: 20	Once the chip has achieved symbol lock on all active lanes and the lanes have been aligned, this field specifies how many 2^8 clock cycles this state must persist before the chip will assert LinkUp in the PCS_IP register. When symbol lock is lost on one or more lanes or lane alignment is lost, this is also the number of 2^8 clock cycles the chip will wait before de-asserting LinkUp.	RW	0x132

15.5.4 SERDES_TEST_MODE[0..24]

Table 106: SERDES_TEST_MODE[0..24]

Name	Bit	Description	Type	Default
BIST_Mode	2: 0	0x0 - Disabled 0x1 - PRBS, Test Data = x9+x5+1, DE=1 0x2 - Test Data = High-frequency test pattern (0101010101) DE =1 0x3 - Test Data = K28.5(Idle) - Mixed-frequency test pattern Pattern, DE =1 0x4 - Test Data = Low-frequency pattern (0001111100) Pattern, DE =1 0x5 - PRBS, Test Data = x10+x3+1, DE=0 0x6 - PRBS, Test Data = x9+x4+1, DE=1 0x7 - PRBS, Test Data = x7+x1	RW	0x0
TestMode	4: 3	Test Mode 0x0 - normal -default 0x1 - Parallel Loop-back 0x2-0x3 - RSVD Must be stable for minimum 500ns before Serdes Reset for each respective lane is de-asserted	RW	0x0
BIST_Sync	5	Synchronizes the RX BIST checker. When register deasserted allows RX BIST to start checking. Change in state is delayed by 5 cycles to allow for starting of pattern through setting BM and also de-assertion the BS bit.	RW	0x1
FramerEnable	6	Enables PCS framer. The function of the PCS framer is to look for the comma character and instruct the SERDES I/O to shift by a certain number of bits when the comma character is not properly aligned. The PCS framer must be enabled at all time except during SERDES testing using BIST.	RW	0x1
TransmitCJPAT	7	Enables generation and transmission of CJPAT pattern as specified in Annex 48A. Pattern is not checked by RX	RW	0x0

See Annex 48A for more pattern details.

15.5.5 SERDES_STATUS[0..24]

Table 107: SERDES_STATUS[0..24]

Name	Bit	Description	Type	Default
SymbolLock	3: 0	Symbol lock - 1 bit per lane. In 1 lane mode only the 1 active lane should be read for polling the lock status. The other 3 bits are undefined. Bit 0 corresponds to Lane A etc.	RO	0x0
SignalDetect	4	Signal Detect based on all four lanes. There is hysteresis in this status, see SERDES_CTRL_3. In 1 lane mode, the Signal detect is only based on lane 0 or lane 3, depending the lane reversal state.	RO	0x0

In 1 lane mode only the 1 active lane is used to determine the Symbol_Lock and Signal_Detect

15.5.6 SERDES_IP[0..24]

Table 108: SERDES_IP[0..24]

Name	Bit	Description	Type	Default
LOS0	0	Lane A Loss of Signal.	CW1	0x0
OBC0	1	Lane A Out of band Character.	CW1	0x0
DE0	2	Lane A Disparity Error.	CW1	0x0
LOS1	3	Lane B Loss of Signal.	CW1	0x0
OBC1	4	Lane B Out of band Character.	CW1	0x0
DE1	5	Lane B Disparity Error.	CW1	0x0
LOS2	6	Lane C Loss of Signal.	CW1	0x0
OBC2	7	Lane C Out of band Character.	CW1	0x0
DE2	8	Lane C Disparity Error.	CW1	0x0
LOS3	9	Lane D Loss of Signal.	CW1	0x0
OBC3	10	Lane D Out of band Character.	CW1	0x0
DE3	11	Lane D Disparity Error.	CW1	0x0
PhyErrorCount	31: 12	Saturating Error counter - increments once per any kind of error in any lane per cycle. For instance if all 12 errors(3 per lane) were asserted the Error count would increment by 1.	CR	0x0

The interrupt detect field for SERDES_IP only covers bits 11:0, not the counter.
In 1 lane mode only the 1 active lane has any significance in the SERDES_IP - all other lanes should be masked.
Writing '1' into any bit 11:0 clears the corresponding bit; writing 0 has no effect. PhyErrorCount is cleared on read.

15.5.7 SERDES_IM[0..24]

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 109: SERDES_IM[0..24]

Name	Bit	Description	Type	Default
SERDES_IM	11: 0	Bits are associated with SERDES_IP 1 - Mask interrupt. 0 - Do not mask interrupt	RW	0x0fff

15.5.8 SERDES_BIST_ERR_CNT[0..24]

Table 110: SERDES_BIST_ERR_CNT[0..24]

Name	Bit	Description	Type	Default
BEC0	7: 0	Saturating BIST error count for lane A	CW	0x0
BEC1	15: 8	Saturating BIST error count for lane B	CW	0x0
BEC2	23: 16	Saturating BIST error count for lane C	CW	0x0
BEC3	31: 24	Saturating BIST error count for lane D	CW	0x0

This register will overcount errors by approximately 2x (between 1x and 3x), since the error signal is stretched to 2 clock cycles long to ensure synchronization between clock domains.

15.5.9 PCS_CFG_1[0..24]

Table 111: PCS_CFG_1[0..24]

Name	Bit	Description	Type	Default
LFSR_Seed	3: 0	LFSR seed The LFSR seed is 8 bits; the lower nibble and upper nibble are initially loaded with the same value	RW	0xa
ShortPreamble	4	Enable support of shorter preamble in 10M/100M/1G mode only Can detect preambles as short as 2B (|S|,SFD) Do not set this option in 10G mode.	RW	0x0
(was NibbleDetect)	5	OBSOLETE	RW	0x0
MuxSelect	7: 6	Reserved	RW	0x0
IFG	13: 8	Programmable TX inter-frame gap in bytes (0-63B) Spec requires min IFG of 12	RW	0xc
IgnoreAllRX_Errors	14	Ignore All RX PCS errors This includes all the checks that can be turned-off indivudally in the following bits	RW	0x0

Name	Bit	Description	Type	Default
IgnoreDataErrors	15	Ignore Data Errors. These errors are found when non-data characters appear within the frame - bounded by |S| and |T| If this bit is set then control charachters that appear within the body of a frame (Except for |S| which will always truncate an ongoing frame) will not truncate the frame and will isneand instead will be passed forward into the switch as 8b data	RW	0x0
IgnorePreambleErrors	16	Ignore Preamble Errors Can only be used in 4-lane (XAUI) mode only	RW	0x0
IgnoreIFGErrors	17	Ignore inter-frame gap errors If not set then checks that at least 1 idle collumn appears before the start of a frame This bit must be set in 1 lane mode	RW	0x0
EnableDIC	18	Enables the deficit idle count. The DIC counter allows an average of the programmed IFG, usually taken as 12, while forcing alignment of the start of frame to lane zero.	RW	0x0
InvertTX_LaneOrdering	19	Invert TX lane ordering. If set to 0, the lane 0 is mapped to lane A, lane 1 to lane B, lane 2 to lane C and lane 3 to lane D. If set to 1, the lane 0 is mapped to lane D, lane 1 to lane C, lane 2 to lane B and lane 3 to lane A.	RW	0x0
InvertRX_LaneOrdering	20	Invert RX lane ordering. If set to 0, the lane 0 is mapped to lane A, lane 1 to lane B, lane 2 to lane C and lane 3 to lane D. If set to 1, the lane 0 is mapped to lane D, lane 1 to lane C, lane 2 to lane B and lane 3 to lane A.	RW	0x0
EnableLF_Response	21	Enable sending of remote faults on RX and also allow the disabling of TX channel when 4 or more RF seen	RW	0x0
EnableSendingRF	22	Enabling sending remote fault in response to RX link being down	RW	0x0
ForceLF	23	Force continous transmission of local fault symbol (sent after every ||A||) Uses data from PCS_CFG_2	RW	0x0
ForceRF	24	Force continuous transmission of remote fault symbol (sent after every ||A||) Uses data from PCS_CFG_3	RW	0x0

Name	Bit	Description	Type	Default
ForceFSIG	25	Force single transmission of Sequence Ordered Set Bit cleared when FSIG is sent and will also cause FS bit to be asserted in PCS_IP Register Uses data from PCS_CFG_4	RW	0x0
DisableTransmitRS	26	Disable Transmit RS. This will stop accepting new frames from the TX MAC.	RW	0x0
DisableReceiverRS	27	Disable Receive RS No further frames will be allowed through to the RX MAC	RW	0x0
EnableLinkUpNotification	28	Allows the link-up notification to be sent to the switch	RW	0x0
DatapathSelect	30: 29	selects datapath mode 00: 4 lanes (10Gb) 01: 1 lane (1Gb) 10: 1 lane - 1/10 effective data rate (100Mb) 11: 1 lane - 1/100 effective data rate (10Mb) A setting of DatapathSelect greater than 00 is referred to as 1 lane mode	RW	0x0
CheckEndEnable	31	Enables CheckEnd functionality which verifies that the |T| and the following |I| are well formed, and if not will invalidate the preceding frame	RW	0x1

- Bits: 14:17 are used for filtering out garbage. This garbage is not counted, A packet that cannot be initially resolved and is discarded by the PCS as garbage will not be counted in the Ethernet counters as a bad
- Can only alter bits 27:21 during operation all other bits must be stable before Serdes put into operation

15.5.10 PCS_CFG_2[0..24]

Table 112: PCS_CFG_2[0..24]

Name	Bit	Description	Type	Default
LF_TX	23: 0	Local fault value The default value is required for compliance to 802.3ae The LF will be constructed by placing bits [7:0] of this register into the LSB of the LF collumn	RW	0x1

15.5.11 PCS_CFG_3[0..24]

Table 113: PCS_CFG_3[0..24]

Name	Bit	Description	Type	Default
RF_TX	23: 0	Remote fault value The default value is required for compliance to 802.3ae. The RF will be constructed by placing bits [7:0] of this register into the LSB of the RF collumn	RW	0x2

15.5.12 PCS_CFG_4[0..24]

Table 114: PCS_CFG_4[0..24]

Name	Bit	Description	Type	Default
FSIG_TX	23: 0	Transmit FSIG value The FSIG will be constructed by placing bits [7:0] of this register into the LSB of the FSIG collumn FSIG can be used as a proprietary mechanism for sending messages across the Ethernet link	RW	0x0

15.5.13 PCS_CFG_5[0..24]

Table 115: PCS_CFG_5[0..24]

Name	Bit	Description	Type	Default
FSIG_RX	23: 0	Received FSIG value The LSB of the incoming FSIG will be placed in bits [7:0]	RO	0x0

15.5.14 PCS_IP[0..24]

Table 116: PCS_IP[0..24]

Name	Bit	Description	Type	Default
LF_Detected	0	Local Fault Detected This is a status bit and not sticky The switch set this bit when at least 4 LF symbols are received from the line within 128 cycles. The switch reset this bit when no LF symbols are received within 128 cycles.	RO	0x0
RF_Detected	1	Remote Fault Detected This is a status bit and not sticky The switch set this bit when at least 4 RF symbols are received from the line within 128 cycles. The switch reset this bit when no RF symbols are received within 128 cycles.	RO	0x0

Name	Bit	Description	Type	Default
FSIG_Detected	2	FSIG detected FSIG value stored in PCS_CFG_5	CW1	0x0
LF_Sent	3	Local fault sent	CW1	0x0
RF_Sent	4	Remote fault sent	CW1	0x0
FSIG_Sent	5	FSIG Sent	CW1	0x0
LanesMisaligned	6	Lanes Mis-Aligned Should be masked in 1 lane mode	CW1	0x0
PCS_FIFO_Overflow	10: 7	PCS FIFO overflow Lane 1 bit per lane	CW1	0x0
LinkUpDown	11	Link transitioned from being up to being down	CW1	0x0
LinkDownUp	12	Link transitioned from being down to being up	CW1	0x0
LinkUp	13	This bit reflects the current status of the link and is not sticky If this bit is set, then the link is in good working order, i.e. signal is detected (SERDES Status[SD]), symbol locked (SERDES Status[SL]) and lanes are aligned (PCS Status[LA]). Hysteresis on this signal is controlled by register SERDES_CONTROL_3	RO	0x0
FaultChange	14	Indicates that there was a local fault or remote fault status change on the line. Read the LF or RF bit to determine the current status.	CW1	0x0
AN_Done	15	Auto-Negotiation completed normally.	CW1	0x0
AN_Fail	16	Auto-Negotiation did not complete normally.	CW1	0x0
AN_RX_MSG	17	Received a message (AN_RX_MSG0 or AN_RX_MSG1).	CW1	0x0

Writing '1' into any bit clears the corresponding bit, writing 0 has no effect.

15.5.15 PCS_IM[0..24]

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 117: PCS_IM[0..24]

Name	Bit	Description	Type	Default
LF_Detected	0	1 - Mask LF_Detected	RW	0x1
RF_Detected	1	1 - Mask RF_Detected	RW	0x1
FSIG_Detected	2	1 - Mask FSIG_Detected	RW	0x1
LF_Sent	3	1 - Mask LF_Sent	RW	0x1
RF_Sent	4	1 - Mask RF_Sent	RW	0x1
FSIG_Sent	5	1 - Mask FSIG_Sent	RW	0x1

Name	Bit	Description	Type	Default
LanesMisaligned	6	1 - Mask LanesMis-aligned	RW	0x1
PCS_FIFO_Overflow	10: 7	1 - Mask PCS_FIFO_Overflow	RW	0xf
LinkUpDown	11	1 - Mask LinkUpDown	RW	0x1
LinkDownUp	12	1 - Mask LinkDownUp	RW	0x1
LinkUp	13	always set - PCS_IP.LinkUp is excluded from EPL_INT_DETECT.PCS_IP calculation	RO	0x1
FaultChange	14	1 - Mask FaultChange	RW	0x1
AN_Done	15	1 - Mask AN_Done	RW	0x1
AN_Fail	16	1 - Mask AN_Fail	RW	0x1
AN_RX_MSG	17	1 - Mask AN_RX_MSG	RW	0x1

In 1 lane mode should mask all 4 lane specific bits

15.5.16 SYNCBUF_CFG[0..24]

****** This register was named PACING_RATE ******

Table 118: SYNCBUF_CFG[0..24]

Name	Bit	Description	Type	Default
(was PacingRate)	7: 0	OBSOLETE	RW	0x0
SyncBufferWatermark	11: 8	Indicates at what syncbuf level the logic should start deleting ||R|| A value of 0 will force every ||R|| that follows a ||A|| to be deleted.	RW	0x8
DeleteAllR	12	When set will delete all ||R|| before the syncbuf	RW	0x0
UNH_A_Behavior	13	When set will turn-off the ||A|| look-ahead behavior, as required for UNH test compliance.	RW	0x1

15.5.17 MAC_CFG_1[0..24]

Table 119: MAC_CFG_1[0..24]

Name	Bit	Description	Type	Default
HeaderOffset	4: 0	Number of words to skip for proprietary header	RW	0x0
(was CRC_Offset)	11: 6	OBSOLETE	RW	0x0
MaxFrameSize	23: 12	Max Frame Size in words	RW	0x180
MinFrameSize	29: 24	Min Frame Size in words	RW	0x10

If a frame violates the min size frame, the following frame on that port may be corrupted as well. Or it could be discarded if it comes after a runt frame such that the header spacing between the two frames violates MAC_CFG_3.MinEventRate.

15.5.18 MAC_CFG_2[0..24]

Table 120: MAC_CFG_2[0..24]

Name	Bit	Description	Type	Default
DisableRX_MAC	0	When asserted it idles the RX MAC on the next frame boundary. All incoming packets are then discarded and are thus prevented from entering the switch.	RW	0x0
DisableTX_MAC	1	When asserted will stop transmission of frames from this port. Packets still drain from the switch; the link transmits idles and stays in sync.	RW	0x0
CheckParseError *(was DisableRX_Pause)*	2	When set, frame header parse errors will be reported; by default, parse errors cause discard of the frame.	RW	0x1
RuntFrameDiscard	3	Mark the frame as discard eligible if the frame is smaller than the minimum size configured.	RW	0x1
RX_CRC_Discard	4	Mark the frame as discard eligible if the frame received as an RX CRC error.	RW	0x1
RX_OversizeDiscard	5	Mark the frame as discard eligible if the frame is above the maximum size. Once the length of a frame has exceeded Max Frame+8B, its additional data is discarded at the RX MAC regardless of the state of this bit.	RW	0x1
PhyErrorDiscard	6	Mark the frame as discard eligible if bytestream or check end error detected, or if serdes error occurs and is enabled via bit 7 (below)	RW	0x1
SerdesErrorDiscard	7	Allow detection of serdes error (illegal characters or link down) to be factored into Phy error indication in MAC. Not recommended; may cause loss of good frames adjacent to corrupted frames.	RW	0x0
PadRuntFrames	8	Pad frames that violate the Min Size to Min Size. If the frame entered the switch >= Min Size with a good CRC, and it has had a tag removed in the switch, it is padded to Min Size with a good CRC. If the frame entered the switch < Min Size and it cannot be discarded, then it leaves the switch padded to Min Size with a forced bad CRC.	RW	0x1
DrainTX *(was EnableSourcePortPause)*	9	Immediately drains and discards pending TX data from datapath. Will cause underflow if set during frame transmission.	RW	0x0
LengthCheckDiscard	10	When set, will discard frames that have length/type field less than or equal to 1500, and whose payload length is not equal to this value (unless the frame is minimum length, in which case the payload may be larger than the length/type value due to padding).	RW	0x0

Name	Bit	Description	Type	Default
EnableVPriUpdate	11	Defines if the VLAN priority is updated or not. If enabled, then the 4-bit VPRI received from the frame handler will be mapped to a new 3 bit code using TX_PRI_MAP, this new code will be used to update the priority if and only if the VLAN field is actually updated. If disabled, then the VLAN priority is not updated when the VLAN ID is updated. This bit is not used if the existing VLAN tag is removed or if the a new VLAN tag is added or if there is no VLAN to update.	RW	0x0
EnableVCfiUpdate	12	Defines if the VLAN CFI/DEI field is updated or not. If enabled, then the 4-bit VPRI received from the frame handler will be mapped through the 1-bit code using TX_PRI_MAP, and the new 1 bit code will be used to update the CFI/DEI if and only if the VLAN field is actually updated. If disabled, then the VLAN CFI/DEI is not updated when the VLAN ID is updated. This bit is not used if the existing VLAN tag is removed or if the a new VLAN tag is added or if there is no VLAN to update.	RW	0x0
VLAN_EType	14: 13	This register determines the VLAN tag type used when a VLAN tag is added to a frame: • 0x0: reserved • 0x1: 0x8100 • 0x2: MAC_VLAN_ETYPE.STagTypeA • 0x3: MAC_VLAN_ETYPE.STagTypeB	RW	0x1
StripRltOnEgress	15	Defines if RLTs (Rate Limiter Tags) should be allowed on frames egressing on this port. When using RLTs, set this bit to define the boundary of the CN domain.	RW	0x0
EnableRouting	16	When set, enables the replacement of MAC addresses and VLAN ids on outgoing routed frames.	RW	0x1
(was VLAN_EtherType)	31: 16	OBSOLETE	RW	0x0
EnableTTLDecrement	17	When set, enables the decrement of the TTL field on outgoing routed frames.	RW	0x1
EnableDSCPModification	18	When set, enables the modification of the DSCP field on outgoing IP frames.	RW	0x1

- Marking a frame as discard eligible will force the frame to be dropped in store and forward mode and may cause the frame to be dropped in the cut-through mode. If the frame is not dropped and actually forwarded in the cut-through mode, then the frame will be transmitted with a corrupted CRC.
- Overflow always discards. It is not a programmable option.
- A packet is not seen if an overflow occurs on the first word

- If Min frame is set to 64 bytes, and Min Frame Discard is enabled, then garbage inputs will never do more harm than result in a first good frame being discarded on the same port as the last bad frame. If in addition, the data-sheet specs a higher Total Switch Max Frame Rate than (Ports*64 bytes), then Min Frame can be reduced until (1/Min Frame) * Ports = Total Switch Max Frame Rate. If MAC_CFG_2[Min Frame discard] is off, but MAC_CFG_2[Pad to Min Size] is on, then the switch will never discard more than one good frame after the last bad frame per port. However, if Min Frame Discard is off and Pad to Min Size is not enabled, then all guarantees of frame discard are off except that the switch should not get into an illegal state.

15.5.19 MAC_CFG_3[0..24]

Table 121: MAC_CFG_3[0..24]

Name	Bit	Description	Type	Default
TX_PauseValue	15: 0	Pause Value used for TX pause frames in terms of the number of 512 bit times that the link partner needs to Pause. The same value is used for all class in class based flow control.	RW	0xffff
MinEventRate	21: 16	MinEventRate sets the minimum spacing (in words) from the start of one frame to the start of the next. The EPL will discard frames that start with less than this spacing. This limits the max frame rate even when MAC_CFG_1:MinFrameSize is set to less than 64B frames. The main point of this register is to prevent incoming line noise from being interpreted as many short, corrupt frames. Note: there is a cycle added to the configured MinEventRate, for the idle cycle between frames. This is less than the minimum IFG so it makes sense to set MinEventRate==MinFrameSize.	RW	0x10
MaxParsingDepth	31: 24	Maximum depth in words to which a frame may be parsed. If this causes parsing to end during IPv6 options (which have unlimited length), the frame will still be forwarded, but MAP_PROT will indicate that L4 was not parsed. The default value is chosen to protect the datapath from overflow during the header latency.	RW	0x40

15.5.20 MAC_CFG_5[0..24]

Table 122: MAC_CFG_5[0..24]

Name	Bit	Description	Type	Default
PortMACID_Lo	31: 0	Least significant 32 bits of the MAC address. Used as a source address when a PAUSE frame is transmitted.	RW	0x0

15.5.21 MAC_CFG_6[0..24]

Table 123: MAC_CFG_6[0..24]

Name	Bit	Description	Type	Default
PortMACID_Hi	15: 0	Most significant 16 bits of the MAC address. Used as a source address when a PAUSE frame is transmitted.	RW	0x0

15.5.22 TX_VPRI_MAP_1[0..24]

****** This register was named TX_PRI_MAP_1 ******
Remap ingress to egress priority (including CFI bit) This register is indexed by a 4-bit number where bits {3:1} are set to the actual PRI received and bit {0} is set to the actual CFI bit received from the frame processor.

Table 124: TX_VPRI_MAP_1[0..24]

Name	Bit	Description	Type	Default
PRI0	3: 0	Map internal VPRI/CFI 0 to Egress VPRI/CFI	RW	0x0
PRI1	7: 4	Map internal VPRI/CFI 1 to Egress VPRI/CFI	RW	0x2
PRI2	11: 8	Map internal VPRI/CFI 2 to Egress VPRI/CFI	RW	0x4
PRI3	15: 12	Map internal VPRI/CFI 3 to Egress VPRI/CFI	RW	0x6
PRI4	19: 16	Map internal VPRI/CFI 4 to Egress VPRI/CFI	RW	0x8
PRI5	23: 20	Map internal VPRI/CFI 5 to Egress VPRI/CFI	RW	0xA
PRI6	27: 24	Map internal VPRI/CFI 6 to Egress VPRI/CFI	RW	0xC
PRI7	31: 28	Map internal VPRI/CFI 7 to Egress VPRI/CFI	RW	0xE

15.5.23 TX_VPRI_MAP_2[0..24]

****** This register was named TX_PRI_MAP_2 ******
Remap ingress to egress priority (including CFI bit)

Table 125: TX_VPRI_MAP_2[0..24]

Name	Bit	Description	Type	Default
PRI8	3: 0	Map internal VPRI/CFI 8 to Egress VPRI/CFI	RW	0x0
PRI9	7: 4	Map internal VPRI/CFI 9 to Egress VPRI/CFI	RW	0x2
PRI10	11: 8	Map internal VPRI/CFI 10 to Egress VPRI/CFI	RW	0x4
PRI11	15: 12	Map internal VPRI/CFI 11 to Egress VPRI/CFI	RW	0x6
PRI12	19: 16	Map internal VPRI/CFI 12 to Egress VPRI/CFI	RW	0x8
PRI13	23: 20	Map internal VPRI/CFI 13 to Egress VPRI/CFI	RW	0xA
PRI14	27: 24	Map internal VPRI/CFI 14 to Egress VPRI/CFI	RW	0xC
PRI15	31: 28	Map internal VPRI/CFI 15 to Egress VPRI/CFI	RW	0xE

15.5.24 MAC_STATUS[0..24]

Table 126: MAC_STATUS[0..24]

Name	Bit	Description	Type	Default
TX_Status	0	TX is idling: not sending packets	RO	0x0
RX_Status	1	RX is idling: not receiving packets	RO	0x0

15.5.25 MAC_IP[0..24]

Table 127: MAC_IP[0..24]

Name	Bit	Description	Type	Default
RX_RuntError	0	RX Runt error Packet is shorter then the programmed Min frame size in MAC_CFG_1	CW1	0x0
RX_S2A_Overflow	1	Overflow error. This bit is set if a data word has been discarded because either the fabric or the frame control back pressured and data was actually lost. This could only happen once per frame.	CW1	0x0
RX_CRC_Error	2	RX CRC error	CW1	0x0
RX_OversizeError	3	RX Oversized error Packet has exceeded programmed Max frame size in MAC_CFG_1	CW1	0x0
(was Rx_PauseOverflow)	4	RX Pause Overflow Implies that Pause information has been lost Note that this is for debug purpose at the unit level and cannot happen at the system level.	CW1	0x0
RX_PhyError	5	RX PHY error	CW1	0x0
TX_CRC_Error	6	TX CRC error (inclusive of internally created CRC errors which are seperated in TX_CRC_Error2)	CW1	0x0
TX_CRC_Error2	7	TX CRC without RX CRC error - this occurs when bit is corrupted inside the switch Note: If RX is not discarding RX CRC errored packets then these packets will be sent through the swtich and will appear as false internally generated errors. In normal operation all RX CRC errored packets should be discarded	CW1	0x0
TX_Underflow	8	TX underflow	CW1	0x0
VLANTAG_TABLE_ParityErr *(was RX_PauseEnableDeAsserted)*	9	Parity error in VLANTAG_TABLE Egress frame will be discarded; need to rewrite table	CW1	0x0
RXFabricRequestEnableDeAsserted	10	Fabric error. This bit is set whenever the enable signal from the switch array becomes deasserted regardless where we are in the	CW1	0x0

Name	Bit	Description	Type	Default
		frame or if there is any data received at all. This could only happen if the crossbar becomes congested. It is not expected to happen if the chip is operated in normal conditions used for debug puposes only		
CheckEndError	11	CheckEnd error found CheckEnd errors occur when the ||T|| or the next collumn after a ||T|| are incorrectly formatted	CW1	0x0

Writing '1' into any bit clears the corresponding bit, writing 0 has no effect.

15.5.26 MAC_IM[0..24]

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 128: MAC_IM[0..24]

Name	Bit	Description	Type	Default
RX_RuntError	0	1 - Mask RX Runt error	RW	0x1
RX_S2A_Overflow	1	1 - Mask Overflow error.	RW	0x1
RX_CRC_Error	2	1 - Mask RX CRC error	RW	0x1
RX_OversizeError	3	1 - Mask RX Oversized error	RW	0x1
(was Rx_PauseOverflow)	4	1 - Mask RX Pause Overflow	RW	0x1
RX_PhyError	5	1 - Mask RX PHY error	RW	0x1
TX_CRC_Error	6	1 - Mask TX CRC error	RW	0x1
TX_CRC_Error2	7	1 - Mask TX CRC error2	RW	0x1
TX_Underflow	8	1 - Mask TX underflow	RW	0x1
VLANTAG_TABLE_ParityErr *(was RX_PauseEnableDeAsserted)*	9	1 - Mask VLANTAG_TABLE parity error	RW	0x1
RXFabricRequestEnableDeAsserted	10	1 - Mask Fabric error.	RW	0x1
CheckEndError	11	1 - Mask CheckEnd error	RW	0x1

15.5.27 EPL_INT_DETECT[0..24]

Table 129: EPL_INT_DETECT[0..24]

Name	Bit	Description	Type	Default
SERDES_IP	0	There is an interrupt in SERDES_IP	RO	0x0
PCS_IP	1	There is an interrupt in PCS_IP	RO	0x0
MAC_IP	2	There is an interrupt in MAC_IP	RO	0x0

Status bits which are cleared when the underlying cause is cleared

15.5.28 EPL_LED_STATUS[0..24]

Table 130: EPL_LED_STATUS[0..24]

Name	Bit	Description	Type	Default
PortError	0	Port has link sync error or no signal	CR	0x0
RF	1	Port Remote Fault - port has or has sent a remote fault	CR	0x0
PortStatus	2	RX Port Status - RX port has link up	CR	0x0
RX_Receiving *(was RX_Recieving)*	3	RX Port Receiving - RX port receiving data	CR	0x0
TX_Transmitting	4	TX Port Transmitting - TX port transmiting data	CR	0x0

This register clears on a read from the LED state machine. It is possible for the CPU to read this as well, in which case the results are cleared independent of the LED state machine. These bits are not part of the standard interrupt detect mechanism

15.5.29 STAT_EPL_ERROR1[0..24]

Table 131: STAT_EPL_ERROR1[0..24]

Name	Bit	Description	Type	Default
UnderflowCount	7: 0	Number of frame that were terminated early or discarded due to underflow in the TX	CW	0x0
OverflowCount	15: 8	Number of overflowed frames (RX) that were discarded before any information was sent to the FCU	CW	0x0

Writing any value reset this register to 0.

15.5.30 STAT_EPL_ERROR2[0..24]

Table 132: STAT_EPL_ERROR2[0..24]

Name	Bit	Description	Type	Default
InternalCRC_ErrorCount	15: 0	Counts the number of frames that were corrupted within switch This occurs when the RX had no problem but TX detected CRC eror on frame	CW	0x0

Writing any value reset this register to 0.

15.5.31 STAT_TX_BYTECOUNT[0..24]

Table 133: STAT_TX_BYTECOUNT[0..24]

Name	Bit	Description	Type	Default
ByteCount	63: 0	Number of bytes transmitted (see STAT_TxOctets in the statistics section). Writing any value resets the counter to 0.	CW	0x0

15.5.32 STAT_RX_JABBER[0..24]

Table 134: STAT_RX_JABBER[0..24]

Name	Bit	Description	Type	Default
RX_Jabber	15: 0	Number of frames received in which frame size > MaxFrame and the CRC is invalid.	CW	0x0

Writing any value reset this register to 0.

15.5.33 STAT_TX_CRC[0..24]

Table 135: STAT_TX_CRC[0..24]

Name	Bit	Description	Type	Default
TX_CRC_Errors	31: 0	Number of frames transmitted with CRC errors. Part of the RMON counters, even though they are physically located in the MAC.	CW	0x0

Writing any value reset this register to 0.

15.5.34 STAT_TX_PAUSE[0..24]

Table 136: STAT_TX_PAUSE[0..24]

Name	Bit	Description	Type	Default
TX_Pause	31: 0	Number of Pause frames transmitted by the MAC. Part of the RMON counters, even though they are physically located in the MAC.	CW	0x0

Writing any value reset this register to 0.

15.5.35 SRC_MAC_LO[0..24]

Table 137: SRC_MAC_LO[0..24]

Name	Bit	Description	Type	Default
SrcMacLo	31: 0	Low 32b of MAC address to be used for routed frames	RW	0x0

This is the lower 32b of the 1st source MAC address (48b). When a frame is routed, the routing action in the FFU specifies which of the 2 source MAC addresses to use on the outgoing frame.

15.5.36 SRC_MAC_HI[0..24]

Table 138: SRC_MAC_HI[0..24]

Name	Bit	Description	Type	Default
SrcMacHi	15: 0	High 16b of MAC address to be used for routed frames	RW	0x0

This is the higher 16b of the 2nd source MAC address (48b). When a frame is routed, the routing action in the FFU specifies which of the 2 source MAC addresses to use on the outgoing frame.

15.5.37 SRC_MAC_VIRTUAL_LO[0..24]

Table 139: SRC_MAC_VIRTUAL_LO[0..24]

Name	Bit	Description	Type	Default
SrcMacLo	31: 0	Low 32b of MAC address to be used for routed frames	RW	0x0

This is the lower 32b of the 2nd source MAC address (48b). When a frame is routed, the routing action in the FFU specifies which of the 2 source MAC addresses to use on the outgoing frame.

15.5.38 SRC_MAC_VIRTUAL_HI[0..24]

Table 140: SRC_MAC_VIRTUAL_HI[0..24]

Name	Bit	Description	Type	Default
SrcMacHi	15: 0	High 16b of MAC address to be used for routed frames	RW	0x0

This is the higher 16b of the 2nd source MAC address (48b). When a frame is routed, the routing action in the FFU specifies which of the 2 source MAC addresses to use on the outgoing frame.

15.5.39 JITTER_TIMER[0..24]

Table 141: JITTER_TIMER[0..24]

Name	Bit	Description	Type	Default
TXJitterSS	5: 0	Number of EPL clock cycles before starting transmission of a frame that is one sub-segment in length (64 bytes) or less.	RW	0x00
TXJitterSF	13: 8	Number of EPL clock cycles before transmission of a frame that meets the following condition: The writing of the frame is at least one segment (256 bytes) ahead of the reading	RW	0x10

Name	Bit	Description	Type	Default
		of the frame. Note: This applies to store-and-forward traffic, as well as cut-through traffic that has at least a segment in the memory as a result of switch congestion.		
TXJitterCT	21: 16	Number of EPL clock cycles before transmission of a cutthrough frame. This counter applies if the frame is not store-and-forward and the scheduler does not know whether the data path has finished storing the frame when the scheduler schedules the frame.	RW	0x1c

15.5.40 PARSE_CFG[0..24]

Table 142: PARSE_CFG[0..24]

Name	Bit	Description	Type	Default
ISLTag	2: 0	Presence and format of ISL tags in frames sent over this port. • 0x0: none • 0x1: Fulcrum F32 tag • 0x2: Fulcrum F64 tag • 0x3: Fulcrum F96 tag • 0x4: other 32b tag, in place of first VLAN tag • 0x5: other 64b tag, in place of first VLAN tag • 0x6: other 96b tag, in place of first VLAN tag • 0x7: reserved	RW	0x0
SendOtherL3 *(was Send32B)*	4	When set, the first DI_CFG.OtherL3Length bytes after the ethertype will always be forwarded to Frame Handler, even when the ethertype is unrecognized (i.e. not IP, Mac Control, or CN).	RW	0x0
ParseL3	5	Set if L3 (IPv4 and IPv6) frame headers should be parsed.	RW	0x0
ParseL4	6	Set if L4 frame headers should be parsed.	RW	0x0
FlagIPv4Options	7	Set if the presence of IPv4 options should be flagged, for trap or FFU action.	RW	0x0
FlagIPv6HopByHop	8	Set if the presence of the IPv6 hop-by-hop option should be flagged, for trap or FFU action.	RW	0x0
FlagIPv6Routing	9	Set if the presence of the IPv6 routing header should be flagged, for trap or FFU action.	RW	0x0
FlagIPv6Frag	10	Set if the presence of the IPv6 fragment header should be flagged, for trap or FFU action.	RW	0x0

Name	Bit	Description	Type	Default
FlagIPv6Dest	11	Set if the presence of the IPv6 destination options should be flagged, for trap or FFU action.	RW	0x0
FlagIPv6Auth	12	Set if the presence of the IPv6 authentication header should be flagged, for trap or FFU action.	RW	0x0
CNType	31: 16	Ethertype used for CN (Congestion Notification) frames. Frames with this ethertype are always treated as though SendOtherL3 is set.	RW	0x0000

The various IPv4 and IPv6 options that can be flagged are all condensed to a single flag bit; the EPL will not distinguish between different options that have been selected to be flagged.

15.5.41 MAC_VLAN_ETYPE_1[0..24]

Table 143: MAC_VLAN_ETYPE_1[0..24]

Name	Bit	Description	Type	Default
CTagType	15: 0	Type field used in standard VLAN (C-VLAN) tag	RW	0x8100
F32Type	31: 16	Type field used in F32 tag	RW	0xF320

15.5.42 MAC_VLAN_ETYPE_2[0..24]

****** This register was named MAC_VLAN_ETYPE ******
Defines the Ethernet Types that can be recognized as stacked VLAN tag types. Setting either of those fields to 0x8100 (or whatever the value of MAC_VLAN_ETYPE_1.CTagType may be) is equivalent to disabling them. Note that this Ethertype may not be set to the same value as the Ctag type in MAC_VLAN_ETYPE_1.

Table 144: MAC_VLAN_ETYPE_2[0..24]

Name	Bit	Description	Type	Default
STagTypeA *(was VLANEtherTypeA)*	15: 0	Type field used in Service VLAN Tag (S-TAG)	RW	0x88a8
STagTypeB *(was VLANEtherTypeB)*	31: 16	Alternate type field used in Service VLAN Tag (S-TAG)	RW	0x9100

15.5.43 PARSE_RLT_1[0..24]

Table 145: PARSE_RLT_1[0..24]

Name	Bit	Description	Type	Default
CPIDLo	31: 0	Lower 32b to match against CPID; compare to CPID{47:16}.	RW	0x00000000

15.5.44 PARSE_RLT_2[0..24]

Table 146: PARSE_RLT_2[0..24]

Name	Bit	Description	Type	Default
CPIDHi	15: 0	Upper 16b to match against CPID; compare to CPID{63:48}.	RW	0x0000
RLTType	31: 16	Type field used in RLT tag.	RW	0x0000

15.5.45 TX_TRUNC[0..24]

Table 147: TX_TRUNC[0..24]

Name	Bit	Description	Type	Default
CpuTruncationLen	11: 0	Length that CPU frames should be truncated to when CpuTruncate is set in TRIGGER_ACTION_CFG_2 of a firing trigger. Length specified in words (4B)	RW	0x16
RSVD	15: 12	Reserved	RV	0x0
MirrorTruncationLen	27: 16	Length that Mirrored frames should be truncated to when MirrorTruncate in TRIGGER_ACTION_CFG_2 of a firing trigger. Length specified in words (4B)	RW	0x16
RSVD	31: 28	Reserved	RV	0x0

15.5.46 CPID_0[0..24]

Table 148: CPID_0[0..24]

Name	Bit	Description	Type	Default
CPID	63: 0		RW	0x16

15.5.47 CPID_1[0..24]

Table 149: CPID_1[0..24]

Name	Bit	Description	Type	Default
CPID	63: 0		RW	0x16

15.5.48 CPID_2[0..24]

Table 150: CPID_2[0..24]

Name	Bit	Description	Type	Default
CPID	63: 0		RW	0x16

15.5.49 CPID_3[0..24]

Table 151: CPID_3[0..24]

Name	Bit	Description	Type	Default
CPID	63: 0		RW	0x16

15.5.50 CPID_4[0..24]

Table 152: CPID_4[0..24]

Name	Bit	Description	Type	Default
CPID	63: 0		RW	0x16

15.5.51 CPID_5[0..24]

Table 153: CPID_5[0..24]

Name	Bit	Description	Type	Default
CPID	63: 0		RW	0x16

15.5.52 CPID_6[0..24]

Table 154: CPID_6[0..24]

Name	Bit	Description	Type	Default
CPID	63: 0		RW	0x16

15.5.53 CPID_7[0..24]

Table 155: CPID_7[0..24]

Name	Bit	Description	Type	Default
CPID	63: 0		RW	0x16

15.5.54 DI_CFG[0..24]

****** This register was named L4_CTL ******

Table 156: DI_CFG[0..24]

Name	Bit	Description	Type	Default
CustomProtocol1	7: 0	Defines the protocol number for custom protocol 1. To disable, set this field to 0x00.	RW	0x0
CustomProtocol2	15: 8	Defines the protocol number for custom protocol 2. To disable, set this field to 0x00.	RW	0x0

Name	Bit	Description	Type	Default
CaptureTCPFlags *(was CaptureTCP_HL_RSV_OPTIONS_Fields)*	16	When set, on TCP frames, the first deep inspection chunk (L4A) will contain the TCP flags and header length, as copied from the TCP header. Other deep inspection chunks follow in L4B, L4C, etc.	RW	0x0
OtherL3Length	23: 18	Number of bytes following ethertype sent to Frame Handler, for non-IP frames. Note that only an even number of bytes is allowed. The bytes captured in this way will show up in the SIP field followed by the DIP field in the Frame Handler This value must be set high enough to accomodate Mac Control and CN parsing, when needed.	RW	0x20
OtherProtocolLength	29: 24	Defines the number of deep inspection bytes captured in all undefined L4 protocols. Note that only an even number of bytes is allowed.	RW	0x0

15.5.55 TCP_WD_MASK_LO[0..24]

Table 157: TCP_WD_MASK_LO[0..24]

Name	Bit	Description	Type	Default
TCP_WD_Mask_Lo	31: 0	Each Bit determines which half-words from the TCP payload get forwarded to Frame Handler. Bit 0 corresponds to the 1st half-word in the TCP payload.	RW	0x0

15.5.56 TCP_WD_MASK_HI[0..24]

Table 158: TCP_WD_MASK_HI[0..24]

Name	Bit	Description	Type	Default
TCP_WD_Mask_Hi *(was TCP_WD_Mask_Lo)*	15: 0	Bit mask to determine which half-words from the TCP payload get forwarded to Frame Handler. Bit 0 of this register corresponds to the 33rd half-word in the TCP payload.	RW	0x0

15.5.57 UDP_WD_MASK_LO[0..24]

Table 159: UDP_WD_MASK_LO[0..24]

Name	Bit	Description	Type	Default
UDP_WD_Mask_Lo	31: 0	Each Bit determines which half-words from the UDP payload get forwarded to Frame Handler. Bit 0 corresponds to the 1st half-word in the UDP payload.	RW	0x0

15.5.58 UDP_WD_MASK_HI[0..24]

Table 160: UDP_WD_MASK_HI[0..24]

Name	Bit	Description	Type	Default
UDP_WD_Mask_Hi *(was UDP_WD_Mask_Lo)*	15: 0	Bit mask to determine which half-words from the UDP payload get forwarded to Frame Handler. Bit 0 of this register corresponds to the 33rd half-word in the UDP payload.	RW	0x0

15.5.59 L4PROT1_WD_MASK_LO[0..24]

Table 161: L4PROT1_WD_MASK_LO[0..24]

Name	Bit	Description	Type	Default
Prot1_WD_Mask_Lo *(was Prot_WD_Mask_Lo)*	31: 0	Each Bit determines whether a half-word from the Custom Protocol 1 (as defined in DI_CFG.CustomProtocol1) payload gets forwarded to Frame Handler. Bit 0 corresponds to the 1st half-word in the Custom payload.	RW	0x0

15.5.60 L4PROT1_WD_MASK_HI[0..24]

Table 162: L4PROT1_WD_MASK_HI[0..24]

Name	Bit	Description	Type	Default
Prot1_WD_Mask_Hi *(was GRE_WD_Mask_Lo)*	15: 0	Each Bit determines whether a half-word from the Custom Protocol 1 (as defined in DI_CFG.CustomProtocol1) payload gets forwarded to Frame Handler. Bit 0 corresponds to the 33rd half-word in the Custom payload.	RW	0x0

15.5.61 L4PROT2_WD_MASK_LO[0..24]

Table 163: L4PROT2_WD_MASK_LO[0..24]

Name	Bit	Description	Type	Default
Prot2_WD_Mask_Lo *(was Custom_WD_Mask_Lo)*	31: 0	Each Bit determines whether a half-word from the Custom Protocol 2 (as defined in DI_CFG.CustomProtocol2) payload gets forwarded to Frame Handler. Bit 0 corresponds to the 1st half-word in the Custom payload.	RW	0x0

15.5.62 L4PROT2_WD_MASK_HI[0..24]

Table 164: L4PROT2_WD_MASK_HI[0..24]

Name	Bit	Description	Type	Default
Prot2_WD_Mask_Hi *(was Custom_WD_Mask_Lo)*	15: 0	Each Bit determines whether a half-word from the Custom Protocol 2 (as defined in DI_CFG.CustomProtocol2) payload gets forwarded to Frame Handler. Bit 0 corresponds to the 33rd half-word in the Custom payload.	RW	0x0

15.5.63 AN_TX_MSG0[0..24]

****** This register was named AN_TX_CW_Lo ******

Table 165: AN_TX_MSG0[0..24]

Name	Bit	Description	Type	Default
message	47: 0	Message 0 to send.	RW	0x0

15.5.64 AN_TX_MSG1[0..24]

****** This register was named AN_TX_CW_Hi ******

Table 166: AN_TX_MSG1[0..24]

Name	Bit	Description	Type	Default
message	47: 0	Message 1 to send.	RW	0x0

15.5.65 AN_RX_MSG0[0..24]

****** This register was named AN_RX_CW_Lo ******

Table 167: AN_RX_MSG0[0..24]

Name	Bit	Description	Type	Default
message	47: 0	Message 0 received.	RO	0x0

15.5.66 AN_RX_MSG1[0..24]

****** This register was named AN_RX_CW_Hi ******

Table 168: AN_RX_MSG1[0..24]

Name	Bit	Description	Type	Default
message	47: 0	Message 1 received.	RO	0x0000000

15.5.67 AN_CTL[0..24]

Table 169: AN_CTL[0..24]

Name	Bit	Description	Type	Default
AN_Enable	0	Enables and gives Auto-negotiation logic control over Serdes	RW	0x0
AN_RestartRX_Check	1	Resets CW reception logic and must walk through entire Auto-negotiation handshaking steps from scratch	RW	0x0
AN_Select	2	Select next message.	RW	0x0
AN_Mode	4: 3	Defines mode of operation of auto-negotiation. • 0: clause 37, transmit inactive. • 1: clause 37, transmit active. • 2: clause 73.	RW	0x0

15.5.68 AN_STATUS[0..24]

Table 170: AN_STATUS[0..24]

Name	Bit	Description	Type	Default
AN_State	3: 0	Current state of AN FSM: • 0x0: idle • 0x1: start • 0x2: ability match: waiting to receive 3 times • 0x3: waiting for SW to toggle AN_Select • 0x4: waiting to receive 3 times with ack • 0x5: waiting to transmit 6 times with ack • 0x6: ability match for subsequent pages • 0x7: fail • 0x8: pass	RO	0x0
AN_Msg_Select	4	Current MSG number to transmit/receive.	RO	0x0

15.5.69 AN_TIMEOUT[0..24]

****** This register was named AN_TimeoutValue ******

Table 171: AN_TIMEOUT[0..24]

Name	Bit	Description	Type	Default
AN_TimeoutValue	15: 0	Defines the maximum time for AN to complete. The actual time out is equal to the square of this value divided by 125MHZ which is the clock required for AN.	RW	0x0

15.5.70 AN_TX_TIMER[0..24]

Table 172: AN_TX_TIMER[0..24]

Name	Bit	Description	Type	Default
TimerValue	23: 0	Minimum number of clocks that the transmitter will transmit the same configuration words before moving to the next configuration words.	RW	0x0

15.5.71 VLANTAG_TABLE[0..127] [0..24]

Defines for each VLAN if the packet leaves tagged or untagged. This is stored as a 25 x 128 x 32-bit table. The first index is the port and the second index is VLAN / 32. The 32-bit encode the egress tagging option for each group of 32 VLAN entry (one bit per VLAN).

Table 173: VLANTAG_TABLE[0..127] [0..24]

Name	Bit	Description	Type	Default
TagEnable	31: 0	Each bit defines the egress tagging option for each VLAN.	RW	0x0

15.6 Scheduler Registers (reserved 4KW) Details

15.6.1 SCHED_GROUP_CFG[0..24]

This register defines the relative egress scheduling-priority of the traffic-classses.

Table 174: SCHED_GROUP_CFG[0..24]

Name	Bit	Description	Type	Default
SchedulingGroupBoundary	7: 0	Defines groups of classes that use a common a deficit counter, for the purpose of delay-deficit round-robin scheduling classes within a scheduling group have strict priority relative to each other. There is 1 bit per class. If this bit set to 0 then the class belongs to the same group as the class to the left (the class with higher index). If this bit is 1, then the class is at the head of a new group.	RW	0xff
PrioritySetBoundary	15: 8	Defines class clustering. There is one bit per class. If the bit is set to 1, then this is the first class of the set, if the bit is set to 0, then this is not the first class of the set and the class remains part of the same priority set as the class to the left (higher index). All classes in a scheduling group must belong to the same priority set. • 10000000: One large set of 8 classes	RW	0xff

Name	Bit	Description	Type	Default
		• 10001010: 3 sets; classes 7,6,5,4 is one set, classes 3,2 is second set and class 1,0 are third set • 11111111: 8 sets with one class per set		
StrictPriority	23: 16	Defines class priority. There is one bit per class. If this bit set to strict (1), then all packets are drained unless throughput limit is achieved. If set to (0), then packets are drained according to weight. Note that this bit must be set to the same value for all classes within the set (all 0s or all 1s).	RW	0xff

15.6.2 FUSE_SEG

Table 175: FUSE_SEG

Name	Bit	Description	Type	Default
all	31: 0		RV	0x0

15.6.3 FUSE_PORT

Table 176: FUSE_PORT

Name	Bit	Description	Type	Default
all	31: 0		RV	0x0

15.7 MTABLE Register Set (reserved 64KW) Details

15.7.1 TX_MIRROR

TX mirrored frames are send to dstport. srcport tells MTable what the original port was, so that when doing IP multicast replication, MTable can look up the list of VLANs for the original port, rather than the TX mirror port.

Table 177: TX_MIRROR

Name	Bit	Description	Type	Default
dstport	5: 0	Port TX mirror frames are sent to	RW	0x0
srcport	13: 8	Port being TX mirrored	RW	0x0

15.7.2 LOG_MASK

****** This register was named CPU_MASK ******

Defines to which port a copy of the frame is sent if is marked for logging. This is a mask rather than a single port number, so that it is possible to multicast to multiple CPUs if desired.

Note that this register is a copy of the CPU_LOG_MASK_FH register and must be configured the same way.

Table 178: LOG_MASK

Name	Bit	Description	Type	Default
DestMask	24: 0	Destination mask for logged frames.	RW	0x0

15.7.3 MIRROR_GLORTS

When a copy of the frame is sent for logging, it is addressed with a GloRT which is computed by adding logGlort to the trap code for the frame. When a frame is TX mirrored, it is addressed with txMirrorGlort.

Table 179: MIRROR_GLORTS

Name	Bit	Description	Type	Default
logGlort *(was cpuGlort)*	15: 0	Glort base for frames copied for logging purpose. Configure the upper 8 bits. The lower 8 bits are used for the storage of the trap/log code.	RW	0xff00
txMirrorGlort	31: 16	GloRT that TX mirror frames are addressed with	RW	0x0

15.7.4 LOOPBACK_SUPPRESS[0..24]

At the edge of the system, one must avoid retransmitting a frame over the same LAG it arrived on, even if the LAG is distributed across multiple edge chips. This is done by comparing the GloRTs for the source and destination LAG/port— if the GloRTs match, not counting the port-specific bits, then the LAGs are the same. Note that the destination port's GloRT is not necessarily the same as the destination GloRT (which might be multicast), so this comparison needs to be done for all external destination ports in parallel. The comparison is turned off for internal ports.

Table 180: LOOPBACK_SUPPRESS[0..24]

Name	Bit	Description	Type	Default
glort	15: 0	GloRT value	RW	0xffff
mask	31: 16	GloRT mask	RW	0x0

15.7.5 IP_MULTICAST_TABLE[0..16383]

The multicast table is a fungible resource for creating linked lists to specify the VLANs which should be replicated on each port for a particular multicast group.

Table 181: IP_MULTICAST_TABLE[0..16383]

Name	Bit	Description	Type	Default
parity	0	Always write as 0. Reads as 1 if this entry has a parity error.	RW	0x0
tail	1	true means don't follow next pointer	RW	0x0
skip	3	true means don't send a frame for this entry	RW	0x0
vlan	15: 4	vlan	RW	0x0
nxtptr	29: 16	next pointer	RW	0x0

- The skip bit is to help the software keep MTable up-to-date with spanning tree state changes.

15.8 MSB Registers Details

15.8.1 MSB_CFG

Table 182: MSB_CFG

Name	Bit	Description	Type	Default
disableIbm	0	Disable FIBM operation. If this bit is set, FIBM Write operations can only be done on MSB_CFG and INTERRUPT_DETECT.	RW	0x1
attachedCpu	1	A cpu is attached to this chip. If this bit is not set, no frames are forwarded from the MSB to the HSM (cpu). Also, counters and interrupts are set based on this bit.	RW	0x1
padHsmFrame	2	Pad frames from HSM (cpu) to Port 0 (switch) to a length of 64 bytes.	RW	0x1
padIbmResponse	3	Pad FIBM response frames to a length of 64 bytes.	RW	0x1
ibmGlortEn	4	FIBM glort enable. If this bit is set, the source glort for FIBM response frames and FIBM interrupt frames is ibmGlort. If this bit is not set, the source glort for FIBM response frames is the destination glort of the FIBM request frame.	RW	0x0
interruptGlortEn	5	Interrupt glort enable. If this bit is not set, no FIBM interrupt frames are sent.	RW	0x0

15.8.2 MSB_IBM_GLORT

Table 183: MSB_IBM_GLORT

Name	Bit	Description	Type	Default
ibmGlort	15: 0	The source glort for FIBM response frames and FIBM interrupt frames.	RW	0x0

15.8.3 MSB_IBM_INT

Table 184: MSB_IBM_INT

Name	Bit	Description	Type	Default
interruptGlort	15: 0	Destination glort for FIBM interrupt frames.	RW	0x0
interruptInterval	31: 16	Interval for repeating FIBM interrupt frames. Increments are 8192 MSB clock cycles (21.8 usec). Counting starts at 0 -- for interruptInterval=N, the actual interval is (N+1)*21.8 usec. FIBM interrupt frames are sent repeatedly until the interrupt is cleared.	RW	0x00FF

15.8.4 MSB_INT_FRAME

Table 185: MSB_INT_FRAME

Name	Bit	Description	Type	Default
etherType	15: 0	Ethertype for FIBM response frames and FIBM interrupt frames.	RW	0xFFFF
islSysPri	19: 16	F64 ISL tag Sys. Pri. for FIBM interrupt frames.	RW	0xF
islUserBits	27: 20	F64 ISL tag User Bits for FIBM interrupt frames.	RW	0x00

15.8.5 MSB_STATS_0

Table 186: MSB_STATS_0

Name	Bit	Description	Type	Default
ibmFrameCtr	31: 0	Valid FIBM request frames received by MSB from Port 0.	RW	0x0

15.8.6 MSB_STATS_1

Table 187: MSB_STATS_1

Name	Bit	Description	Type	Default
nonIbmFrameCtr	31: 0	Valid non-FIBM frames received by MSB from Port 0.	RW	0x0

15.8.7 MSB_STATS_2

Table 188: MSB_STATS_2

Name	Bit	Description	Type	Default
errorFrameCtr	31: 0	Non valid frames received by MSB from Port 0.	RW	0x0

15.8.8 MSB_INTR_CTR_0

Table 189: MSB_INTR_CTR_0

Name	Bit	Description	Type	Default
interruptCtr	31: 0	Total FIBM interrupt frames sent.	RW	0x0

15.8.9 MSB_INTR_CTR_1

Table 190: MSB_INTR_CTR_1

Name	Bit	Description	Type	Default
interruptSeqCtr	31: 0	FIBM interrupt frames sent since IP bit was set.	RW	0x0

15.8.10 MSB_INTR_CTR_2

Table 191: MSB_INTR_CTR_2

Name	Bit	Description	Type	Default
ibmCrcErrorCtr	31: 0	FIBM request frames received by MSB from Port 0 with a CRC error.	RW	0x0

15.8.11 MSB_INTR_CTR_3

Table 192: MSB_INTR_CTR_3

Name	Bit	Description	Type	Default
nonIbmCrcErrorCtr	31: 0	Non-FIBM frames received by MSB from Port 0 with a CRC error.	RW	0x0

15.8.12 MSB_INTR_CTR_4

Table 193: MSB_INTR_CTR_4

Name	Bit	Description	Type	Default
frameAndNoCpuCtr	31: 0	Non-FIBM frames received by MSB from Port 0 and attachedCpu==0.	RW	0x0

15.8.13 MSB_INTR_CTR_5

Table 194: MSB_INTR_CTR_5

Name	Bit	Description	Type	Default
invalidIbmOpCtr	31: 0	FIBM request frame with an invalid FIBM op code.	RW	0x0

15.8.14 MSB_IP

Table 195: MSB_IP

Name	Bit	Description	Type	Default
ipIbmCrcError	0	IP -- Fatal Error -- FIBM request frame with a CRC error.	CW1	0x0
ipNonIbmCrcError	1	IP -- Non-FIBM frame received from Port 0 with a CRC error.	CW1	0x0
ipFrameAndNoCpu	2	IP -- Non-FIBM frame received from Port 0 and attachedCpu==0.	CW1	0x0
ipInvalidIbmOp	3	IP -- Fatal Error -- FIBM request frame with an invalid FIBM op code.	CW1	0x0

15.8.15 MSB_IM

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 196: MSB_IM

Name	Bit	Description	Type	Default
imIbmCrcError	0	IM -- Fatal Error -- FIBM request frame with a CRC error.	RW	0x1
imNonIbmCrcError	1	IM -- Non-FIBM frame received from Port 0 with a CRC error.	RW	0x1
imFrameAndNoCpu	2	IM -- Non-FIBM frame received from Port 0 and attachedCpu==0.	RW	0x1
imInvalidIbmOp	3	IM -- Fatal Error -- FIBM request frame with an invalid FIBM op code.	RW	0x1

15.8.16 MSB_RX_EPL_RATE

Table 197: MSB_RX_EPL_RATE

Name	Bit	Description	Type	Default
numCycles	7: 0	Number of clock cycles before one is skipped. Default = 5. This means that the MSB will not send data to the RX EPL on every 6th clock cycle. This gives the MSB an effective clock speed of 375 MHz * 5/6 = 312.5 MHz which matches the speed of the RX EPL.	RW	0x5
interFrameGap	11: 8	Inter frame gap. The number of clock cycles after sending a frame before sending the next one.	RW	0x6

15.8.17 MSB_SCRATCH_0

Table 198: MSB_SCRATCH_0

Name	Bit	Description	Type	Default
scratch0	31: 0	Scratch register 0.	RW	0x0

15.8.18 MSB_SCRATCH_1

Table 199: MSB_SCRATCH_1

Name	Bit	Description	Type	Default
scratch1	31: 0	Scratch register 1.	RW	0x0

15.8.19 MSB_SCRATCH_2

Table 200: MSB_SCRATCH_2

Name	Bit	Description	Type	Default
scratch2	31: 0	Scratch register 2.	RW	0x0

15.8.20 MSB_SCRATCH_3

Table 201: MSB_SCRATCH_3

Name	Bit	Description	Type	Default
scratch3	31: 0	Scratch register 3.	RW	0x0

15.8.21 MSB_SCRATCH_4

Table 202: MSB_SCRATCH_4

Name	Bit	Description	Type	Default
scratch4	31: 0	Scratch register 4.	RW	0x0

15.8.22 MSB_SCRATCH_5

Table 203: MSB_SCRATCH_5

Name	Bit	Description	Type	Default
scratch5	31: 0	Scratch register 5.	RW	0x0

15.8.23 MSB_SCRATCH_6

Table 204: MSB_SCRATCH_6

Name	Bit	Description	Type	Default
scratch6	31: 0	Scratch register 6.	RW	0x0

15.8.24 MSB_SCRATCH_7

Table 205: MSB_SCRATCH_7

Name	Bit	Description	Type	Default
scratch7	31: 0	Scratch register 7.	RW	0x0

15.8.25 MSB_SCRATCH_8

Table 206: MSB_SCRATCH_8

Name	Bit	Description	Type	Default
scratch8	31: 0	Scratch register 8.	RW	0x0

15.8.26 MSB_SCRATCH_9

Table 207: MSB_SCRATCH_9

Name	Bit	Description	Type	Default
scratch9	31: 0	Scratch register 9.	RW	0x0

15.8.27 MSB_SCRATCH_10

Table 208: MSB_SCRATCH_10

Name	Bit	Description	Type	Default
scratch10	31: 0	Scratch register 10.	RW	0x0

15.8.28 MSB_SCRATCH_11

Table 209: MSB_SCRATCH_11

Name	Bit	Description	Type	Default
scratch11	31: 0	Scratch register 11.	RW	0x0

15.8.29 MSB_SCRATCH_12

Table 210: MSB_SCRATCH_12

Name	Bit	Description	Type	Default
scratch12	31: 0	Scratch register 12.	RW	0x0

15.8.30 MSB_SCRATCH_13

Table 211: MSB_SCRATCH_13

Name	Bit	Description	Type	Default
scratch13	31: 0	Scratch register 13.	RW	0x0

15.8.31 MSB_SCRATCH_14

Table 212: MSB_SCRATCH_14

Name	Bit	Description	Type	Default
scratch14	31: 0	Scratch register 14.	RW	0x0

15.8.32 MSB_SCRATCH_15

Table 213: MSB_SCRATCH_15

Name	Bit	Description	Type	Default
scratch15	31: 0	Scratch register 15.	RW	0x0

15.8.33 MSB_TS

INTERNAL USE ONLY.

Table 214: MSB_TS

Name	Bit	Description	Type	Default
fbufHasPort0	0	Frame buffer has test and set for Port 0.	RW	0x0
lsmHasPort0	1	LSM has test and set for Port 0.	RW	0x0
fh0HasPort0	2	FH has test and set for Port 0.	RW	0x0
fsHasPort0	3	FS has test and set for Port 0.	RW	0x0
schHasPort0	4	SCH has test and set for Port 0.	RW	0x0
intHasPort0	5	FIBM interrupts have test and set for Port 0.	RW	0x0
hsmHasFbuf	6	HSM has test and set for the frame buffer.	RW	0x0
ibmHasFbuf	7	FIBM has test and set for the frame buffer.	RW	0x0

15.8.34 MSB_CREDITS

INTERNAL USE ONLY.

Table 215: MSB_CREDITS

Name	Bit	Description	Type	Default
hsmCredits	1: 0	Credits for sending frames to the HSM. One credit = 16 words.	RO	0x3

15.8.35 MSB_SRAM_REPAIR_0

INTERNAL USE ONLY.

Table 216: MSB_SRAM_REPAIR_0

Name	Bit	Description	Type	Default
bank0RepairAddr	10: 0	Frame buffer SRAM bank 0 repair address.	RW	0x0
bank0LockBit	11	SRAM bank 0 repair address lock bit. This register can only be written once.	RO	0x0

15.8.36 MSB_SRAM_REPAIR_1

INTERNAL USE ONLY.

Table 217: MSB_SRAM_REPAIR_1

Name	Bit	Description	Type	Default
bank1RepairAddr	10: 0	Frame buffer SRAM bank 1 repair address.	RW	0x0
bank1LockBit	11	SRAM bank 1 repair address lock bit. This register can only be written once.	RO	0x0

15.9 Frame Handler Registers Details

15.9.1 SYS_CFG_1

Table 218: SYS_CFG_1

Name	Bit	Description	Type	Default
trapSlow	0	If 1, then frames of all other IEEE reserved multicast addresses (0x0180C2000004 - 0x0180C200001F and 0x0180C2000022 - 0x0180C200002F) will be trapped to the CPU. If 0, forward normally.	RW	0x1

Name	Bit	Description	Type	Default
trapLACP	1	If 1, then LACP and Marker frames (destination address = 0x0180C2000002) will be trapped to the CPU. If 0, forward normally.	RW	0x1
trapBPDU	2	If 1, then BPDU packets (destination address = 0x0180C2000000) will be trapped to the CPU. If 0, forward BPDU packets normally.	RW	0x1
trapGARP	3	If 1, then GARP packets will be trapped to the CPU. This includes both GMRP and GVRP (destination address = 0x0180c2000020 and destination address = 0x0180C2000021) If 0, forward normally.	RW	0x1
trap802_1x	5	If 1, then 802.1x frames (destination address = 0x0180C2000003) will be trapped to CPU. If 0, then forward normally.	RW	0x1
dropPause	10	This bit is no longer used. PAUSE frames are passed to the Frame Handler regardless of whether PAUSE is enabled or disabled.	RW	0x1
trapMTUViolations	11	Defines disposition for frames that exceed MTU size allowed on the egress VLAN (only applicable for routed frames). If this bit is set, then the frames that exceed the MTU size are trapped to the CPU. If this bit is reset, then the frames are forwarded normally.	RW	0x1
enableTrapPlusLog	12	If set to 1, then trapping and logging can both occur for a given frame. If set to 0, then trapping will take precedence over logging.	RW	0x1
floodControlUnicast	13	If a unicast address is unknown on destination address look-up, it will be flooded unless this bit is set.	RW	0x0
floodControlMulticast	14	If a multicast address is unknown on destination address look-up, it will be flooded unless this bit is set.	RW	0x0

15.9.2 SYS_CFG_3

Table 219: SYS_CFG_3

Name	Bit	Description	Type	Default
CPUMAMSB	15: 0	Top 16 bits of the CPU MAC address	RW	0x0

15.9.3 SYS_CFG_4

Table 220: SYS_CFG_4

Name	Bit	Description	Type	Default
all	31: 0	Bottom 32 bits of the CPU MAC address	RW	0x0

If a frame has a destination address = CPU MAC address, then that packet is sent to the CPU regardless of VLAN association.

15.9.4 SYS_CFG_7

Table 221: SYS_CFG_7

Name	Bit	Description	Type	Default
ageTime	30: 0	MAC table entry age time, t, in terms of Frame handler clock cycles. Table aging proceeds one entry every t+20 FH clock periods. The 16K table requires two cycles through the table to complete the aging process (2*16K*(t+20)*FHClkPer). Example: FH clock 375 MHz (period = 2.67 ns) Timer set to 0x7530 (decimal 30,000) Entries are aged one per 0.08 ms (30,020*2.67ns). Entire table is aging process occurs in 2.62 sec (.08ms*16384*2). 0x0: RSVD	RW	0x7530
disableAging	31	1: Do not age the table 0: Age the table with the age time specified below.	RW	0x1

15.9.5 SYS_CFG_8

Table 222: SYS_CFG_8

Name	Bit	Description	Type	Default
enableFFU	0	If 1, enables FFU, POLICER, and ARP units.	RW	0x0
allowQTagPause	1	If 1, accepts pause frames that have been VLAN tagged (either directly or via isl tagging)	RW	0x0

15.9.6 PORT_VLAN_IP_1

Table 223: PORT_VLAN_IP_1

Name	Bit	Description	Type	Default
all	31: 0	A known unicast address couldn't be forwarded to its destination because the egress port was not in its VLAN membership group, and VLAN unicast tunnel is off, or the destination address is not locked. This does not apply to standard VLAN flooding. The bit number corresponds to the port number of the port of the frame's ingress.	CW1	0x0

15.9.7 PORT_VLAN_IM_1

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 224: PORT_VLAN_IM_1

Name	Bit	Description	Type	Default
Reserved1	0		RV	0x0
mask	24: 1		RW	0xffffff
Reserved2	31: 25		RV	0x0

15.9.8 PORT_VLAN_IP_2

Table 225: PORT_VLAN_IP_2

Name	Bit	Description	Type	Default
all	31: 0	Source port not a member for that VLAN ID. The bit number corresponds to the port number of the port of the frame's ingress.	CW1	0x0

15.9.9 PORT_VLAN_IM_2

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 226: PORT_VLAN_IM_2

Name	Bit	Description	Type	Default
Reserved1	0		RV	0x0
mask	24: 1	For each interrupt: 1 = Mask Interrupt 0 = Do not mask interrupt	RW	0xffffff

Name	Bit	Description	Type	Default
Reserved2	31: 25		RV	0x0

15.9.10 PORT_MAC_SEC_IP

Table 227: PORT_MAC_SEC_IP

Name	Bit	Description	Type	Default
all	31: 0	A security violation occurred on this port. The bit number corresponds to the port number.	CW1	0x0

15.9.11 PORT_MAC_SEC_IM

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 228: PORT_MAC_SEC_IM

Name	Bit	Description	Type	Default
mask	24: 1		RW	0xffffff

15.9.12 FH_INT_DETECT

Frame handler interrupt detect. The bits available in this register are also available in the global INTERRUPT_DETECT register in the HSM.

Table 229: FH_INT_DETECT

Name	Bit	Description	Type	Default
Port_Vlan_IP_1	0	Indicates there is an interrupt on Port_Vlan_1_IP	RO	0x0
Port_Vlan_IP_2	1	Indicates there is an interrupt on Port_Vlan_2_IP	RO	0x0
Port_Mac_Sec_IP	2	Indicates there is an interrupt on Port_Mac_Sec_IP	RO	0x0
Triggers	3	Indicates there is an interrupt on TRIGGERS_IP	RO	0x0
PARITY_IP	4	Indicates there is an interrupt on PARITY_IP	RO	0x0
MA_TCN_IP	5	Indicates there is an interrupt on MA_TCN_IP	RO	0x0
CM_IP	6	Congestion Management interrupt on CM_IP	RO	0x0
ARP_IP	7	Indicates there is an interrupt on ARP_IP	RO	0x0
POLICER_IP	8	Indicates there is an interrupt on POLICER_IP	RO	0x0

15.9.13 SYS_CFG_ROUTER

Table 230: SYS_CFG_ROUTER

Name	Bit	Description	Type	Default
TTLdisposal *(was trapTTL1)*	1: 0	Defines disposition of routed frames with TTL 1 or 0. The options are: • 0 : drop silently • 1 : trap(unicast) or log(multicast) if protocol is ICMP and drop otherwise • 2 : trap(unicast) or log(multicast) all frames Note that 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, then the frame is still switched within the VLAN it was received from regardless of its TTL value while the routing part is conditional to the TTL value received and the configuration of this field.	RW	0x0
trapIPOptions	2	Defines disposition of IP frames with options: • 0 : formware normally • 1 : trap to CPU	RW	0x0

15.9.14 L34_HASH_CFG

Table 231: L34_HASH_CFG

Name	Bit	Description	Type	Default
Symmetric	0	When set, SIP/DIP and L4SRC/L4DST fields are symmetrized prior to hashing.	RW	0x0
UseSIP	1	Use SIP in L3/L4 hash.	RW	0x1
UseDIP	2	Use DIP in L3/L4 hash.	RW	0x1
UsePROT	3	Use PROT in L3/L4 hash.	RW	0x1
UseTCP	4	Use L4SRC/L4DST when PROT is TCP.	RW	0x1
UseUDP	5	Use L4SRC/L4DST when PROT is UDP.	RW	0x1
UsePROT1	6	Use L4SRC/L4DST when PROT is PROT1.	RW	0x0
UsePROT2	7	Use L4SRC/L4DST when PROT is PROT2.	RW	0x0
UseL4SRC	8	Use L4SRC if L4 protocol is enabled.	RW	0x1
UseL4DST	9	Use L4DST if L4 protocol is enabled.	RW	0x1

Name	Bit	Description	Type	Default
ECMP_Rotation	11: 10	Specifies one of three 12-bit hash rotations for use in ECMP binning. The value 0x3 is reserved. Not applicable to FM3000 devices.	RW	0x0
RSVD	15: 12	Reserved.	RV	0x0
PROT1	23: 16	PROT1 programmable L4 protocol.	RW	0x1
PROT2	31: 24	PROT2 programmable L4 protocol.	RW	0x1

15.9.15 L34_FLOW_HASH_CFG_1

Table 232: L34_FLOW_HASH_CFG_1

Name	Bit	Description	Type	Default
DiffServMask	5: 0	Masks the IPv4 DiffServ field in the L34 hash function.	RW	0x0
UserMask	15: 8	Masks the ISL tag's USER field in the L34 hash function.	RW	0x0

15.9.16 L34_FLOW_HASH_CFG_2

Table 233: L34_FLOW_HASH_CFG_2

Name	Bit	Description	Type	Default
FlowLabelMask	19: 0	Masks the IPv6 Flow Label field in the L34 hash function.	RW	0x0

15.9.17 L234_HASH_CFG

Table 234: L234_HASH_CFG

Name	Bit	Description	Type	Default
UseL2ifIP	0	Include the Layer 2 header in the hash function if an IPv4/IPv6 packet.	RW	0x1
UseL34	1	Include the L34 hash value in the L234 hash function.	RW	0x1
Symmetric	2	When set, SMAC/DMAC fields are symmetrized prior to hashing.	RW	0x0
UseDMAC	3	Include DMAC in the L2/L3/L4 hash.	RW	0x1
UseSMAC	4	Include SMAC in the L2/L3/L4 hash.	RW	0x1
UseType	5	Include EtherType field in the L2/L3/L4 hash. If EtherType is less than 0x600, a value of 0 is used for these two bytes.	RW	0x1
UseVPRI	6	Include User (VLAN) Priority field in the L2/L3/L4 hash.	RW	0x1

Name	Bit	Description	Type	Default
UseVID	7	Include VLAN ID field in the L2/L3/L4 hash.	RW	0x1
RotationA	9: 8	Specifies one of four 12-bit H234 rotations for use as Rotation A.	RW	0x1
RotationB	11: 10	Specifies one of four 12-bit H234 rotations for use as Rotation B.	RW	0x1
TahoeCompatible	12	When set, all hash rotations will be backwards compatible with the Tahoe frame hash, and symmetrization will be XOR-based (applicable when Symmetric is 1). Note that setting this bit is a necessary, but not sufficient, condition for achieving Tahoe compatibility. See the Frame Hashing chapter for more details.	RW	0x0

15.9.18 TX_MIRROR_FH

TX mirrored frames are send to dstport. srcport tells MTable what the original port was, so that when doing IP multicast replication, MTable can look up the list of VLANs for the original port, rather than the TX mirror port.

Table 235: TX_MIRROR_FH

Name	Bit	Description	Type	Default
dstport	5: 0	Port TX mirror frames are sent to.	RW	0x0
srcport	13: 8	Port being TX mirrored. A value greater than 24 disables the TX mirroring feature.	RW	0x3f

15.9.19 CPU_TRAP_MASK_FH

When a trapped frame is sent to the CPU, it is sent out on these ports. (This is a mask rather than a single port number, so that it is possible to multicast to multiple CPUs if desired.)

Table 236: CPU_TRAP_MASK_FH

Name	Bit	Description	Type	Default
DestMask	24: 0	Destination mask for CPU.	RW	0x1

15.9.20 CPU_LOG_MASK_FH

When a copy of the frame (logging) is sent to the CPU, it is sent out on these ports. (This is a mask rather than a single port number, so that it is possible to multicast to multiple CPUs if desired.)

Note that this register is a copy of the LOG_MASK register and must be configured the same way.

Table 237: CPU_LOG_MASK_FH

Name	Bit	Description	Type	Default
DestMask	24: 0	Destination mask for CPU.	RW	0x1

15.9.21 TRAP_GLORT

****** This register was named CPU_GLORT ******

Table 238: TRAP_GLORT

Name	Bit	Description	Type	Default
trapGlort	15: 0	Destination glort for CPU (for traps).	RW	0x0

15.9.22 RX_MIRROR_CFG

Table 239: RX_MIRROR_CFG

Name	Bit	Description	Type	Default
rxMirrorGlort	15: 0	glort for rx mirroring	RW	0x0
rxMirrorPort	20: 16	port for rx mirroring	RW	0x0

15.9.23 PARITY_IP

Parity interrupt register. The corresponding bit is set when a parity error is detected in any of the parity checks. When a bit is set in response to a parity error the fh_in_detect register is also set and an interrupt is sent

Table 240: PARITY_IP

Name	Bit	Description	Type	Default
parIpIngressVid *(was parIpVid)*	0	VLAN Table Parity Error.	CW1	0x0
parIpIngressFid *(was parIpFid)*	1	FID Table Parity Error.	CW1	0x0
parIpEgressVid	2	VLAN Table Parity Error.	CW1	0x0
parIpEgressFid	3	FID Table Parity Error.	CW1	0x0
parIpMA	4	MA_TABLE Parity Error.	CW1	0x0
parIpGlrtRam	5	GLORT RAM Parity Error.	CW1	0x0
parGlrtTable	6	GLORT Dest Table Parity Error.	CW1	0x0
parFFUramTable	7	FFU SRAM table parity error.	CW1	0x0
parTCN	8	TCN_FIFO parity error.	CW1	0x0
parARP	9	ARP table parity error.	CW1	0x0

15.9.24 PARITY_IM

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 241: PARITY_IM

Name	Bit	Description	Type	Default
parIpIngressVid *(was parIpVid)*	0	VLAN Table Parity Error.	RW	0x1
parIpIngressFid *(was parIpFid)*	1	FID Table Parity Error.	RW	0x1
parIpEgressVid	2	VLAN Table Parity Error.	RW	0x1
parIpEgressFid	3	FID Table Parity Error.	RW	0x1
parIpMA	4	MA_TABLE Parity Error.	RW	0x1
parIpGlrtRam	5	GLORT RAM Parity Error.	RW	0x1
parGlrtTable	6	GLORT Dest Table Parity Error.	RW	0x1
parFFUramTable	7	FFU SRAM table parity error.	RW	0x1
parTCN	8	TCN_FIFO parity error.	RW	0x1
parARP	9	ARP table parity error.	RW	0x0

15.9.25 SAF_MATRIX[0..24]

Defines cut-through (0) vs store-and-forward (1) from each ingress port to each egress port. The matrix has 25 32-bit entries, one entry per ingress port. The 32-bit entries contains one bit per egress port.

Table 242: SAF_MATRIX[0..24]

Name	Bit	Description	Type	Default
port0	0		RV	(0..24) 0x1
port1	1		RW	(0) 0x1 (1..24) 0x0
port2	2		RW	(0) 0x1 (1..24) 0x0
port3	3		RW	(0) 0x1 (1..24) 0x0
port4	4		RW	(0) 0x1 (1..24) 0x0
port5	5		RW	(0) 0x1 (1..24) 0x0
port6	6		RW	(0) 0x1 (1..24) 0x0
port7	7		RW	(0) 0x1 (1..24) 0x0

Name	Bit	Description	Type	Default
port8	8		RW	(0) 0x1 (1..24) 0x0
port9	9		RW	(0) 0x1 (1..24) 0x0
port10	10		RW	(0) 0x1 (1..24) 0x0
port11	11		RW	(0) 0x1 (1..24) 0x0
port12	12		RW	(0) 0x1 (1..24) 0x0
port13	13		RW	(0) 0x1 (1..24) 0x0
port14	14		RW	(0) 0x1 (1..24) 0x0
port15	15		RW	(0) 0x1 (1..24) 0x0
port16	16		RW	(0) 0x1 (1..24) 0x0
port17	17		RW	(0) 0x1 (1..24) 0x0
port18	18		RW	(0) 0x1 (1..24) 0x0
port19	19		RW	(0) 0x1 (1..24) 0x0
port20	20		RW	(0) 0x1 (1..24) 0x0
port21	21		RW	(0) 0x1 (1..24) 0x0
port22	22		RW	(0) 0x1 (1..24) 0x0
port23	23		RW	(0) 0x1 (1..24) 0x0
port24	24		RW	(0) 0x1 (1..24) 0x0

15.9.26 FH_LOOPBACK_SUPPRESS[0..24]

At the edge of the system, we must avoid retransmitting a frame over the same LAG it arrived on, even if the LAG is distributed across multiple edge chips. This is done by comparing the GloRTs for the source and destination LAG/port— if the GloRTs match, not counting the port-specific bits, then the LAGs are the same. Note that the destination port's GloRT is not necessarily the same as the destination GloRT (which might be multicast), so this comparison needs to be done for all external destination ports in parallel. The comparison is turned off for internal ports.

Table 243: FH_LOOPBACK_SUPPRESS[0..24]

Name	Bit	Description	Type	Default
glort	15: 0	GloRT value	RW	0xffff
mask	31: 16	GloRT mask	RW	0x0

15.9.27 INTERNAL_PORT_MASK

Table 244: INTERNAL_PORT_MASK

Name	Bit	Description	Type	Default
mask	24: 0	Internal ports to remove should be set to 0, all others should be set to 1	RW	0x1ffffff

15.9.28 MGMT_CLK_COUNTER

The MGMT block in Frame Handler can issue a mgmt command every 5 clk cycles, with this register being set to 1. This interval can be slowed down by increasing the register value. For FFU mgmt access, MGMT_FFU_CLK_COUNTER is used to determine the gap. For proper operation this register should be set to no lower than 1

Table 245: MGMT_CLK_COUNTER

Name	Bit	Description	Type	Default
MGMT_CLK_COUNTER	20: 0	Counts the clk ticks before issuing a mgmt access to Frame Handler.	RW	0x4

15.9.29 MGMT_FFU_CLK_COUNTER

For proper FFU mgmt access, this number should be set to 10.

Table 246: MGMT_FFU_CLK_COUNTER

Name	Bit	Description	Type	Default
MGMT_FFU_CLK_COUNTER	20: 0	Counts the clk ticks before issuing a mgmt access to Frame Handler.	RW	0x14

15.10 Port Association Registers (reserved 4KW) Details

15.10.1 PORT_CFG_1[0..24]

Table 247: PORT_CFG_1[0..24]

Name	Bit	Description	Type	Default
defaultVID	11: 0	Default VLAN ID. Applied internally if useDefaultVLAN==1 or the frame is not VLAN	RW	0x1

Name	Bit	Description	Type	Default
		tagged or the VLAN tag has VID==0 (Priority tagged).		
defaultVPRI	15: 12	Default VLAN user priority. This is the internal mapped 4-bit VPRI which will get regenerated to a TX VPRI on each egress port. Applied if useDefaultVLAN==1 or the frame is not VLAN tagged.	RW	0x0
defaultDSCP	21: 16	Default DSCP of this port. Applied internally when useDefaultDSCP==1 or the frame is not IP.	RW	0x0
dropTagged	22	If 1, drop tagged frames. That is, drop if the first EtherType is a VLAN type and the VID is not 0.	RW	0x0
dropUntagged	23	If 1, drop untagged frames. That is, drop if the first EtherType is not a VLAN type or the VID is 0.	RW	0x0
dropMgmtISL	24	If 1, drop ISL tagged management frames.	RW	0x1
useDefaultVLAN	25	If 1, overwrite the VID and VPRI with defaultVID and defaultVPRI regardless of wheather the frame was VLAN tagged or not. This default is applied after SwitchPriority is chosen.	RW	0x0
useDefaultDSCP	26	If 1, overwrite incoming DSCP with defaultDSCP. This default is applied after SwitchPriority is chosen.	RW	0x0
SwitchPriorityFromVLAN	27	If 1 and the frame is VLAN tagged, sets SwitchPriority from the mapped internal VPRI.	RW	0x1
SwitchPriorityFromDSCP	28	If 1 and the frame is IP, map DSCP to SwitchPriority.	RW	0x0
SwitchPriorityFromISL	29	If 1 and the frame is ISL tagged, use ISL priority as SwitchPriority regardless of SwitchPriorityFromVLAN or SwitchPriorityFromDSCP.	RW	0x1
SwitchPriorityPrefersDSCP	30	If 1, SwitchPriority prefers the DSCP mapped priority over VPRI.	RW	0x0
AssumeUntagged *(was EncapsulateVLAN)*	31	If 1, treats the incoming frame as if it were untagged for the purpose of adding/removing VLAN tags on egress. If the frame leaves the switch tagged in 802.1Q mode, it gets an additional stacked VLAN tag. If the frames leaves the switch untagged in 802.1Q mode, then the original VLAN is preserved. This does not change what is presented to the FFU.	RW	0x0

15.10.2 PORT_CFG_2[0..24]

Table 248: PORT_CFG_2[0..24]

Name	Bit	Description	Type	Default
destinationMask	24: 0	A vector for each port i, a bit for each port j, 1 = Port i may send packets to port j. 0 = Port i may not send packets to port j. This feature is used to: • Prevent multicast and broadcast traffic from going out the port it came in on, • Cut loops in statically-configured networks, • Prevent link aggregates from receiving multiple copies of multicast and broadcast traffic. This mask is always "anded" with the destination mask. If the mask is set to all ones (as it is by default), it will have no effect. There is no need to have a default setting of bit i on port i = 0 to prevent loops. The reflect bit in the VLAN table automatically creates this effect. When a frame is routed, the destinationMask has no effect.	RW	0x1ffffff

The Port Based VLAN registers are also viewed as a general port membership list. This is used for other features in the device besides legacy non-802.3q VLANs. The features are:

- Port-based VLAN
- Link Aggregation
- Preventing Loop back

Note: If ingress port frame reflection is enabled, and the per-VLAN frame reflection bit is set for the VLAN associated with a given frame, then a frame may egress its ingress port, if either:

- The frame is flooded for a DLF
- The egress port is the forwarding port as determined by the MAC table
- The frame is a broadcast frame

Note: There is no requirement for a static table entry. This rule supercedes PORT_CFG_2 [1..24].
0 = a frame's egress port may not also be its ingress port.

15.10.3 PORT_CFG_3[0..24]

Table 249: PORT_CFG_3[0..24]

Name	Bit	Description	Type	Default
filterVLANIngress	0	If 1 and the source port does not match the VLAN membership, it is a VLAN boundary violation and the packet is dropped.	RW	0x0

Name	Bit	Description	Type	Default
LearningEnable	1	If 1, source MAC addresses from this port will be learned.	RW	0x1
SecurityType	3: 2	Defines the security type. • 0: No security checks. • 1: Unknown source MAC addresses are a security violation • 2: Known source MAC addresses moving to another port are a security violation. • 3: Both unknown source MAC addresses or known source addresses moving to another port are considered security violation. Note: Port security is not VLAN aware.	RW	0x0
RelaxSourceLookup	6: 5	Defines the MAC address source lookup mode. • 0: Strict source lookup done on all frames. • 1: MAC address source lookups for learning is best effort and may be skipped during periods of high load to help Bali keep up with frame rate. • 2: MAC address source lookup for learning is rated by a token bucket to help reduce power. Note 1: regardless of this register, writing learned entries into the MAC address table is always best-effort.	RW	0x1

15.10.4 PORT_CFG_ISL[0..24]

Table 250: PORT_CFG_ISL[0..24]

Name	Bit	Description	Type	Default
srcGlort	15: 0	Default ISL source Glort to assign to frames arriving on this port if the frames do not have an ISL tag or if the frames do have an ISL tag but the source glort specified is 0.	RW	0x0
USR	23: 16	Default ISL USR bits to assign to frames arriving on this port.	RW	0x0
defaultPriority	27: 24	Default ISL Priority to assign to frames arriving on this port.	RW	0x0

15.10.5 RX_VPRI_MAP[0..24]

****** This register was named RX_PRI_MAP ******
This table defines mapping of received VLAN priority and CFI bit into an internal 4-bit VPRI code per port. The table is first indexed by the port number and then indexed by a 4-bit number where bits {3:1} are set to

the actual PRI received and bit {0} is set to the actual CFI bit received. The table is not used if the packet is not VLAN tagged.

Table 251: RX_VPRI_MAP[0..24]

Name	Bit	Description	Type	Default
pri0	3: 0		RW	0x0
pri1	7: 4		RW	0x0
pri2	11: 8		RW	0x1
pri3	15: 12		RW	0x1
pri4	19: 16		RW	0x2
pri5	23: 20		RW	0x2
pri6	27: 24		RW	0x3
pri7	31: 28		RW	0x3
pri8	35: 32		RW	0x4
pri9	39: 36		RW	0x4
pri10	43: 40		RW	0x5
pri11	47: 44		RW	0x5
pri12	51: 48		RW	0x6
pri13	55: 52		RW	0x6
pri14	59: 56		RW	0x7
pri15	63: 60		RW	0x7

15.10.6 DSCP_PRI_MAP[0..63]

Table 252: DSCP_PRI_MAP[0..63]

Name	Bit	Description	Type	Default
pri	3: 0	SwitchPri to set for this DSCP.	RW	0x0

15.10.7 VPRI_PRI_MAP[0..15]

Table 253: VPRI_PRI_MAP[0..15]

Name	Bit	Description	Type	Default
pri	3: 0	Defines the switch priority associated with each VPRI.	RW	(0) 0x0 (1) 0x1 (2) 0x2 (3) 0x3 (4) 0x4 (5) 0x5 (6) 0x6 (7) 0x7 (8) 0x8 (9) 0x9 (10) 0xa (11) 0xb (12) 0xc (13) 0xd (14) 0xe (15) 0xf

15.11 GLORT Control Registers and GLORT Table (reserved 16KW) Details

15.11.1 GLORT_DEST_TABLE[0..4095]

Defines the destinations for a given glort. (FM4000 only: If an IP_MulticastIndex is specified (value different than 0), then the destination mask is used along the index to retrieve a list of VLANs during IP multicast replication. If the index is set to 0, then the destination mask includes one bit per port to indicate which port will receive a copy of the frame.)

Table 254: GLORT_DEST_TABLE[0..4095]

Name	Bit	Description	Type	Default
ParityError	0	Parity error bit. Setting to 0 on writing force the hardware to set the proper parity, setting this bit to 1 on writing force the hardware to set the parity bit to the wrong value. Reading 0 indicates the parity was good, reading 1 indicates that the parity was incorrect.	RW	0x0
DestMask	25: 1	Contains the destination mask (25 bits).	RW	0x0
IP_MulticastIndex	39: 26	FM4000 only. Defines first entry in the IP multicast table. A value of 0 indicates that the IP_MULTICAST_TABLE is not used when the frame is sent.	RW	0x0

15.11.2 GLORT_CAM[0..255]

Configures an entry in the GLORT CAM. The CAM entries are readable but the value returned will be masked out according to the mask originally written along with the value.

A match occurs when the data searched is equal to the key and the entry is valid. Both the data and the key are ANDed with the mask prior to comparison. Note that a mask of 0 will cause the entry to be automatically invalidated.

The highest index as precedence in case there are more than one match.

Table 255: GLORT_CAM[0..255]

Name	Bit	Description	Type	Default
value	15: 0	Glort value.	RW	0x0
mask	31: 16	Glort Mask. Writing 0 will cause the entry to be invalidated.	RW	0x0

15.11.3 GLORT_RAM[0..255]

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.

The Strict field indicates whether the GLORT_DEST_TABLE index will be 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, then 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).

Table 256: GLORT_RAM[0..255]

Name	Bit	Description	Type	Default
ParityError	0	Parity error bit. Setting to 0 on writing force the hardware to set the proper parity, setting this bit to 1 on writing force the hardware to set the parity bit to the wrong value. Reading 0 indicates the parity was good, reading 1 indicates that the parity was incorrect.	RW	0x0
Strict	2: 1	Determines strict versus hashed destination mask selection. • 0: Strict property is inherited from the ISL tag's FTYPE. • 1: RSVD. • 2: Not strict (hashed). • 3: Strict.	RW	0x0
DestIndex	14: 3	Defines the base index of the first entry into the GLORT_DEST table which contains the destination.	RW	0x0
RangeSubIndexA	22: 15	Defines the position of sub index A (first 4 bits is bit position, second 4 bits is number of bits) to use for indexing into the GLORT_DEST_TABLE.	RW	0x0
RangeSubIndexB	30: 23	Defines the position of sub index B (first 4 bits is bit position, second 4 bits is number of bits) to use for indexing into the GLORT_DEST_TABLE.	RW	0x0
DestCount	34: 31	Defines the number of DestMask entries in the GLORT_DEST_TABLE over which frames will be hashed. A value of 0 is interpreted as 16. For distributed link aggregation pruning, this corresponds to the number of ports in the LAG.	RW	0x0
HashRotation	35	Selects hash Rotation A (0) or Rotation B (1) for calculating the destination mask index.	RW	0x0

15.12 Congestion Management Registers (reserved 4KW) Details

15.12.1 CM_TX_TC_PRIVATE_WM[0..24] [0..7]

Defines the watermark above which a frame can pontentially be not queued.

Table 257: CM_TX_TC_PRIVATE_WM[0..24] [0..7]

Name	Bit	Description	Type	Default
watermark	12: 0	Number of segments.	RW	0x1fff

15.12.2 CM_TX_TC_USAGE[0..24] [0..7]

Stores the number of segments per TX port per traffic class.

Table 258: CM_TX_TC_USAGE[0..24] [0..7]

Name	Bit	Description	Type	Default
count	13: 0	Number of segments.	RO	0x0

15.12.3 CM_TX_SMP_HOG_WM[0..24] [0..3]

****** This register was named CM_TX_SHARED_HOG_WM ******
Defines the watermark, excluding the TX_TC_PRIVATE_WM, above which frames are not queued on this port. There are four hogs watermark per port, the hog watermark selected depends on the CM_TX_HOG_MAP registers.

Table 259: CM_TX_SMP_HOG_WM[0..24] [0..3]

Name	Bit	Description	Type	Default
watermark	12: 0	Number of segments.	RW	0x1fff

15.12.4 CM_RX_SMP_PAUSE_WM[0..24] [0..1]

Defines the watermark per SMP per RX above which pause frames are started or stopped.

Table 260: CM_RX_SMP_PAUSE_WM[0..24] [0..1]

Name	Bit	Description	Type	Default
PauseOn	12: 0	Level at which a pause ON is sent.	RW	0x1fff
PauseOff	28: 16	Level at which a pause OFF is sent.	RW	0x1fff

15.12.5 CM_RX_SMP_PRIVATE_WM[0..24] [0..1]

Defines the private watermark per SMP per RX above which frames can get potentially dropped.

Table 261: CM_RX_SMP_PRIVATE_WM[0..24] [0..1]

Name	Bit	Description	Type	Default
watermark	12: 0	Number of segments.	RW	0x1fff

15.12.6 CM_RX_SMP_USAGE[0..24] [0..1]

Stores the number of segments per port per SMPs

Table 262: CM_RX_SMP_USAGE[0..24] [0..1]

Name	Bit	Description	Type	Default
count	13: 0	Number of segments.	RO	0x0

15.12.7 CM_TX_SMP_USAGE[0..24] [0..1]

Stores the number of segments per port per SMPs

Table 263: CM_TX_SMP_USAGE[0..24] [0..1]

Name	Bit	Description	Type	Default
count	13: 0	Number of segments.	RO	0x0

15.12.8 CM_TX_SMP_PRIVATE_WM[0..24] [0..1]

Defines the private watermark per SMP per TX above which frames can get potentially dropped.

Table 264: CM_TX_SMP_PRIVATE_WM[0..24] [0..1]

Name	Bit	Description	Type	Default
watermark	12: 0	Number of segments.	RW	0x1fff

15.12.9 CM_RX_SMP_HOG_WM[0..24] [0..1]

****** This register was named CM_RX_SHARED_SMP_HOG_WM ******
Defines the maximum number of segments allowed per receiver per SMP before a new frame is discarded.

Table 265: CM_RX_SMP_HOG_WM[0..24] [0..1]

Name	Bit	Description	Type	Default
watermark	12: 0	Number of segments.	RW	0x1fff

15.12.10 CM_PAUSE_RESEND_INTERVAL[0..24]

The periodicity at which a pause ON frame (a frame with pause-timer value non-zero) is generated is given by 25* FH_CLK_PERIOD * INTERVAL (25 is the number of ports including CPU port). This value is configurable per-port. Exact timing of when a pause frame appears depends load conditions, MTU.

Table 266: CM_PAUSE_RESEND_INTERVAL[0..24]

Name	Bit	Description	Type	Default
INTERVAL	15: 0	Interval between pause regenerations	RW	0x100

15.12.11 CM_PORT_CFG[0..24]

Table 267: CM_PORT_CFG[0..24]

Name	Bit	Description	Type	Default
DECIMATOR	2: 0	Controls how Pause Time ticks with respect to the Frame Handler clock ticks. Maps ports to PAUSE_DECIMATION settings.	RW	0x0
PAUSE_TYPE	3	0 - Generate normal pause frames, 1 - Generate class-based pause frames	RW	0x0
DISASSOCIATION_MASK	15: 8	Bit mask enabling pause for each traffic class. An egress queue will not pause if the corresponding bit is set to 0.	RW	0xff
SMP0_RL_PAUSE *(was SMP0_RL_ACTION)*	16	Defines if PAUSE frame is set if the rate limiter usage goes below 0 for SMP0.	RW	0x0
SMP1_RL_PAUSE *(was SMP1_RL_ACTION)*	20	Defines if PAUSE frame is set if the rate limiter usage goes below 0 for SMP1.	RW	0x0

15.12.12 CM_RX_USAGE[0..24]

Stores the number of segments used per port.

Table 268: CM_RX_USAGE[0..24]

Name	Bit	Description	Type	Default
count	13: 0	Number of segments.	RO	0x0

15.12.13 CM_SHARED_WM[0..15]

****** This register was named CM_RX_SHARED_WM ******
Defines the shared watermark per switch priority above which packets are dropped if the shared pool usage, CM_SHARED_SMP_USAGE for the SMP used by this switch priority is above this limit.

Table 269: CM_SHARED_WM[0..15]

Name	Bit	Description	Type	Default
watermark	12: 0	Number of segments.	RW	0x21c

15.12.14 CM_PAUSE_DECIMATION[0..7]

Controls how Pause Time ticks with respect to the Frame Handler clock ticks.

Table 270: CM_PAUSE_DECIMATION[0..7]

Name	Bit	Description	Type	Default
K *(was UP)*	13: 0	Update period, FH clk decimator, every (K+1) * 25 FH clk cycles counters are decremented by 1.	RW	0x0
N *(was CP)*	18: 14	Correction period, every (N+1) * UP * 25 FH Clk cycles the correction value M is subtracted from the pause timer instead of decrementing by 1.	RW	0x9
M *(was CV)*	31: 19	Correction value, unsigned number that is subtracted from the pause time value every (N+1) * (K+1) * 25 FH Clk cycles.	RW	0x3

15.12.15 CM_SHARED_SMP_PAUSE_WM[0..1]

Defines the watermark per SMP compared against CM_SHARED_SMP_USAGE for sending pause frames.

Table 271: CM_SHARED_SMP_PAUSE_WM[0..1]

Name	Bit	Description	Type	Default
PauseOn	12: 0	Level at which a pause ON is sent.	RW	0x1fff
PauseOff	28: 16	Level at which a pause OFF is sent.	RW	0x1fff

15.12.16 CM_SHARED_SMP_USAGE[0..1]

Contains the current number of segments stored in each SMP excluding those used in the RX private queues. A segment is only counted in this shared pool if all private watermarks have been exceeded.

Table 272: CM_SHARED_SMP_USAGE[0..1]

Name	Bit	Description	Type	Default
count	13: 0	Number of segments.	RO	0x0

15.12.17 CM_GLOBAL_USAGE

Stores the total number of segments used currently in the switch.

Table 273: CM_GLOBAL_USAGE

Name	Bit	Description	Type	Default
count	13: 0	Number of segments.	RO	0x0

15.12.18 CM_GLOBAL_WM

Defines the global watermark above which no new frames are accepted.

Table 274: CM_GLOBAL_WM

Name	Bit	Description	Type	Default
watermark	12: 0	Number of segments.	RW	0xfa0

15.12.19 CM_SMP_MEMBERSHIP

Defines the SMP membership per traffic class. The choices are:

- 0: Non SMP controlled
- 1: SMP 0
- 2: SMP 1
- 3-15: Reserved for future use

Frames are dropped for all values other than 0,1,2.

Table 275: CM_SMP_MEMBERSHIP

Name	Bit	Description	Type	Default
tc0	3: 0	Membership for traffic class 0.	RW	0x0
tc1	7: 4	Membership for traffic class 1.	RW	0x0
tc2	11: 8	Membership for traffic class 2.	RW	0x0
tc3	15: 12	Membership for traffic class 3.	RW	0x0
tc4	19: 16	Membership for traffic class 4.	RW	0x0
tc5	23: 20	Membership for traffic class 5.	RW	0x0
tc6	27: 24	Membership for traffic class 6.	RW	0x0
tc7	31: 28	Membership for traffic class 7.	RW	0x0

15.12.20 CM_TX_HOG_MAP

Defines the hog watermark to use for each 8 traffic class.

Table 276: CM_TX_HOG_MAP

Name	Bit	Description	Type	Default
tc0	1: 0	Hog to use for traffic class 0.	RW	0x0
tc1	3: 2	Hog to use for traffic class 1.	RW	0x0
tc2	5: 4	Hog to use for traffic class 2.	RW	0x0
tc3	7: 6	Hog to use for traffic class 3.	RW	0x0
tc4	9: 8	Hog to use for traffic class 4.	RW	0x0
tc5	11: 10	Hog to use for traffic class 5.	RW	0x0
tc6	13: 12	Hog to use for traffic class 6.	RW	0x0
tc7	15: 14	Hog to use for traffic class 7.	RW	0x0

15.12.21 CN_CPID_MASK

Defines the congestion notification mask.

Table 277: CN_CPID_MASK

Name	Bit	Description	Type	Default
Mask	15: 0	Anded with CPID on rate limit tag before matching.	RW	0x0

15.12.22 CN_VCN_DMAC_2[0..24]

This register need not be access atomicly.

Table 278: CN_VCN_DMAC_2[0..24]

Name	Bit	Description	Type	Default
MAC	15: 0		RW	0xeeff

15.12.23 CN_RATE_LIM_CPID[0..24] [0..1]

Table 279: CN_RATE_LIM_CPID[0..24] [0..1]

Name	Bit	Description	Type	Default
index	3: 0	Entry in use CN_CPID_TABLE for this port and this SMP. Manually writing into these registers is unsafe if the CN_GLOBAL_CFG_2.cpidCache bit is set.	RW	0xf

15.12.24 CN_SMP_CFG[0..24] [0..1]

Defines congestion notification parameters per port per SMP. Measured in segments.

Table 280: CN_SMP_CFG[0..24] [0..1]

Name	Bit	Description	Type	Default
EQ	11: 0	Desired equilibrum point.	RW	0x0
SC	27: 16	Severe congestion point.	RW	0x0

15.12.25 CN_SMP_THRESHOLD[0..24] [0..1]

Defines congestion notification thresholds per port per SMPs. Measured in segments.

Table 281: CN_SMP_THRESHOLD[0..24] [0..1]

Name	Bit	Description	Type	Default
Min	11: 0	Used as Q_LOW for VCN.	RW	0x0

Name	Bit	Description	Type	Default
Max	27: 16	Used as Q_HIGH for VCN.	RW	0x0

15.12.26 CN_SAMPLE_CFG

For each port and SMP pair, the frames passing through are sampled randomly for congestion notification depending on the number of bytes seen. After each sample, the port and SMP pair waits to see SampleMin*1024 + LFSR[9:0]<<SampleJitter bytes before sampling again unless SampleJitter = 0xf in which case the port SMP pair waits SampleMin*1024 bytes before sampling again.

Table 282: CN_SAMPLE_CFG

Name	Bit	Description	Type	Default
SampleMin	15: 0	Mininum number of kilobytes passing through this port and belonging to the same SMP before a frame is sampled.	RW	0x064
SampleJitter	19: 16	Determines the random jitter between samples. A random 10 bit number is shifted left by SampleJitter and added to SampleMin to determine the random number of bytes seen between samples. A value of 0xf means random jitter is disabled and the sample period is exactly SampleMin kilobytes.	RW	0xf

15.12.27 CN_RATE_LIM_CFG[0..24]

Defines the rate limiter parameters for reacting to congestion notification frames.

Table 283: CN_RATE_LIM_CFG[0..24]

Name	Bit	Description	Type	Default
Unit	7: 0	Fraction of 1/256. Specifies BCN Rate Unit.	RW	0x01
Min	15: 8	Fraction of 1/256. Specifies BCN R_MIN parameter.	RW	0x15
MaxFrac	23: 16	Fraction of 1/256. Together with Max, specifies the maximum rate for this port.	RW	0x15
Max	30: 24	Units. Together with MaxFrac, specifies the maximum rate for this port.	RW	0x15

15.12.28 CN_CPID_TABLE[0..7]

Table 284: CN_CPID_TABLE[0..7]

Name	Bit	Description	Type	Default
CPID	63: 0		RW	0x0

15.12.29 CN_GLOBAL_CFG_1

****** This register was named CN_GLOBAL_CFG ******
Defines the global configuration registers.

Table 285: CN_GLOBAL_CFG_1

Name	Bit	Description	Type	Default
Mode	1: 0	Defines the operating mode for congestion notification • 0 - congestion notification is disabled • 1 - VCN • 2 - FBCN • Others - Reserved	RW	0x0
SamplePeriod	11: 2	Defines the interval of time before sending a new frame. The unit of time is 256 * Frame Handler clock period.	RW	0x0A
EtherType	31: 16	Defines the ethertype that will be used for all congestion notification frames while operating in one of the CN modes. This value is used during frame generation and parsing of received frames. A frame will be subject to the CN_FORWARD_MASK and CN_RATE_ACTION_MASK settings if and only if its ethertype matches this value and Mode is non-zero.	RW	0x1001

15.12.30 CN_GLOBAL_CFG_2

Defines the global configuration registers.

Table 286: CN_GLOBAL_CFG_2

Name	Bit	Description	Type	Default
CNSrcGlort	15: 0	Defines the source glort for CN frame when they are sent back to the network.	RW	0x0
cpidCache	16	Enables or disables the BCN cpid cache.	RW	0x0
rlIncrease	17	Enables the rate increase reaction • 0 Rate increase disable. Rate limiter will not react to CN rate increase feedback. • 1 Rate increase enable. Rate limiter will react to cN rate increase feedback.	RW	0x0
rlDecrease	18	Enables the rate decrease reaction • 0 Rate decrease disable. Rate limiter will not react to CN rate decrease feedback.	RW	0x0

Name	Bit	Description	Type	Default
		• 1 Rate decrease enable. Rate limiter will react to cN rate decrease feedback.		
fbQDelta	19	Enables use of Qdelta term in the computation of the feedback term (fb) for the rate limiters. • 0 No Qdelta. Fb = f(Qoff) where f is defined by the text in the appropriate congestion management chapter. • 1 Use Qdelta. Fb = f(Qoff, Qdelta)	RW	0x1
backoffMode	21: 20	Defines backoff behavior. • 00 Backoff disable • 01 Proportional backoff (FCN backoff). Backoff time is proportional to Fb see FCN chapter. • 10 Stateful backoff. Backoff when we get Qoff=0, Qdelta=0. Subsequent CN messages cause the token bucket rate to be halved. • 11 stateless backoff. Backoff when we get Qoff=0, Qdelta=0. Subsequent CN messages do not change the token bucket rate.	RW	0x0
localReaction	22	Bypass generation of CN frames if the destination rate limiter is local to this chip. • 0 CN Frames always generated, even for local rate limiters. This means that a CN frame will arrive at the Frame Handler even if it is destined to go to a local rate limiter. • 1 CN Frames bypassed for local rate limiters, when a local rate limiter needs to be updated, the Frame Handler will change the internal registers directly instead of generating a frame that goes through the congestion notification pipeline.	RW	0x0
sendSample	23	Include the sampled frame in the CN frame. • 0 CN frames do not contain sampled frame. • 1 CN frames contain sampled frame.	RW	0x0
qEqSaturate	24	Whether abs(Q_OFFSET) saturates at q_eq • This bit does not have any effect and is set to always saturate	RW	0x0

Name	Bit	Description	Type	Default
ignoreMatch	25	Whether an RL increase may be sent without needing a cpidMatch • 1 a CPID match is not required for a rate-increase to be issued • 0 a CPID match is required for a rate-increase to be issued	RW	0x0
reqFwd	26	Whether to require that the rx port of a sample'd frame have its FORWARD_MASK bit set. This is implicitly enabled when localReaction is set to prevent sampling frames at the Congestion Point wherein the Reaction Point would be on the same chip. Setting this bit and clearing localReaction will disable intra-switch BCN • 1 it is permissible to send a frame • 0 it is not permissible to send a frame	RW	0x1

15.12.31 CN_FB_CFG

Defines congestion notification parameters for feedback at reaction point.

Table 287: CN_FB_CFG

Name	Bit	Description	Type	Default
DriftMultExp	4: 0	Any value over 17 disables Drift drift period is FH_CLK_PERIOD * 25 << DriftMultExp	RW	0xc
BCN_W	9: 8	BCN w shift parameter (unsigned qty)	RW	0x1
BCN_GD *(was BCN_DG)*	21: 16	BCN GD shift parameter (unsigned qty)	RW	0x8
BCN_GI	29: 24	BCN GI shift parameter (signed qty)	RW	0x0
FCN_BF	27: 24	FCN_BACKOFF_FACTOR (signed qty)	RW	0x0

15.12.32 CN_FORWARD_MASK

Defines congestion notification forward mask

Table 288: CN_FORWARD_MASK

Name	Bit	Description	Type	Default
Mask	24: 0	Defines disposition of congestion notification frames to this egress port: • 1: forward the frame on that port • 0: check CN_RATE_ACTION_MASK	RW	0x0

15.12.33 CN_RATE_ACTION_MASK

Defines congestion notification rate action mask

Table 289: CN_RATE_ACTION_MASK

Name	Bit	Description	Type	Default
Mask	24: 0	Defines disposition of congestion notification frames to this egress port: • 0: discard the frame • 1: process the frame The value 1 shall not be selected for ports operating at 10M or 100M.	RW	0x0

15.12.34 CN_BACKOFF_BYTETIME

Defines congestion notification backoff byte time

Table 290: CN_BACKOFF_BYTETIME

Name	Bit	Description	Type	Default
Time	22: 0		RW	0x1388
RandShift	2: 0		RW	0x0

15.12.35 CN_FRAME_CFG_1

Defines the congestion notification prefixes.

Table 291: CN_FRAME_CFG_1

Name	Bit	Description	Type	Default
MA_CPID	31: 0	Defines the upper bits of the MAC address for VCN and FCN congestion notifications frames and CPID prefix for BCN frames.	RW	0x0

15.12.36 CN_FRAME_CFG_2

Defines the congestion notification prefixes.

Table 292: CN_FRAME_CFG_2

Name	Bit	Description	Type	Default
MA_CPID	15: 0	Defines the lower 16 bits of the MAC address for VCN and FCN congestion notifications frames and CPID prefix for BCN frames.	RW	0x0

Name	Bit	Description	Type	Default
VLAN	27: 16	Defines the VLAN for VCN congestion notification frames.	RW	0x0
Priority	31: 28	Defines the priority field for congestion notifications frames.	RW	0x0

15.12.37 CN_VCN_DMAC_1

This register need not be access atomicly.

Table 293: CN_VCN_DMAC_1

Name	Bit	Description	Type	Default
MAC	31: 0		RW	0xeffeff

15.12.38 TX_RATE_LIM_CFG[0..24] [0..7]

Table 294: TX_RATE_LIM_CFG[0..24] [0..7]

Name	Bit	Description	Type	Default
Capacity	16: 0	Defines the saturation value in 64-byte units.	RW	0x100
RateFrac	24: 17	Defines the fraction of a byte to add every 25 FH clock. The fraction is N/256.	RW	0xff
RateUnit	31: 25	Defines the number of bytes to add to the token bucket every 25 FH clock. To set the Frame Handler clock to values less than 280MHz, set this value to 0x7f.	RW	0x6f

15.12.39 TX_RATE_LIM_USAGE[0..24] [0..7]

Table 295: TX_RATE_LIM_USAGE[0..24] [0..7]

Name	Bit	Description	Type	Default
frac	7: 0		RO	0x0
units	31: 8		RO	0x2

15.12.40 RX_RATE_LIM_CFG[0..24] [0..1]

Table 296: RX_RATE_LIM_CFG[0..24] [0..1]

Name	Bit	Description	Type	Default
Capacity	16: 0	Defines the saturation value in 64-byte units.	RW	0x100
RateFrac	24: 17	Defines the fraction of a byte to add every 25 FH clock. The fraction is N/256.	RW	0xff

Name	Bit	Description	Type	Default
RateUnit	31: 25	Defines the number of bytes to add to the token bucket every 25 FH clock. To set the Frame Handler clock to values less than 280MHz, set this value to 0x7f.	RW	0x6f

15.12.41 RX_RATE_LIM_USAGE[0..24] [0..1]

Table 297: RX_RATE_LIM_USAGE[0..24] [0..1]

Name	Bit	Description	Type	Default
frac	7: 0		RO	0x0
units	31: 8		RO	0x2

15.12.42 RX_RATE_LIM_THRESHOLD[0..24] [0..1]

Table 298: RX_RATE_LIM_THRESHOLD[0..24] [0..1]

Name	Bit	Description	Type	Default
off	15: 0	Sets the level of RX_RATE_LIM_USAGE at which to generate a PAUSE_OFF after rate-limiting in 1-byte quantity.	RW	0x0
drop	31: 16	Sets the level of RX_RATE_LIM_USAGE at which to frames are dropped in 1-byte quantity.	RW	0x0

15.12.43 CM_PORT_LIST[0..24]

Table 299: CM_PORT_LIST[0..24]

Name	Bit	Description	Type	Default
port	4: 0	Defines the list of ports that are active for congestion management purpose. By default all ports are included in the list. The benefit of reducing the list is to reduce power in smaller devices.	RW	(0) 0x0 (1) 0x1 (2) 0x2 (3) 0x3 (4) 0x4 (5) 0x5 (6) 0x6 (7) 0x7 (8) 0x8 (9) 0x9 (10) 0xa (11) 0xb (12) 0xc (13) 0xd (14) 0xe (15) 0xf (16) 0x10 (17) 0x11 (18) 0x12 (19) 0x13 (20) 0x14 (21) 0x15 (22) 0x16 (23) 0x17 (24) 0x18
last	5		RW	(0..23) 0x0 (24) 0x1

15.12.44 SCHED_DRR_Q[0..24] [0..7]

Defines the quantum of this group in reference to the other classes in deficit round robin. This parameter is only used in reference to other round robin classes of the same egress priority. There are 25 x 8 entries where 25 is the number of ports and 8 is the number of groups.

Table 300: SCHED_DRR_Q[0..24] [0..7]

Name	Bit	Description	Type	Default
Weight	23: 0		RW	0x800

15.12.45 SCHED_SHAPING_GROUP_CFG[0..24]

Maps per-port traffic classes to shaping groups

Table 301: SCHED_SHAPING_GROUP_CFG[0..24]

Name	Bit	Description	Type	Default
pri0	2: 0	Maps traffic class 0 to scheduling group [0..7]	RW	0x0
reserved0	3	Reserved.	RV	0x0
pri1	6: 4	Maps traffic class 1 to scheduling group [0..7]	RW	0x0
reserved1	7	Reserved.	RV	0x0
pri2	10: 8	Maps traffic class 2 to scheduling group [0..7]	RW	0x0
reserved2	11	Reserved.	RV	0x0
pri3	14: 12	Maps traffic class 3 to scheduling group [0..7]	RW	0x0
reserved3	15	Reserved.	RV	0x0
pri4	18: 16	Maps traffic class 4 to scheduling group [0..7]	RW	0x0
reserved4	19	Reserved.	RV	0x0
pri5	22: 20	Maps traffic class 5 to scheduling group [0..7]	RW	0x0
reserved5	23	Reserved.	RV	0x0
pri6	26: 24	Maps traffic class 6 to scheduling group [0..7]	RW	0x0
reserved6	27	Reserved.	RV	0x0
pri7	30: 28	Maps traffic class 7 to scheduling group [0..7]	RW	0x0
reserved7	31	Reserved.	RV	0x0

15.12.46 SWITCH_PRI_TO_CLASS_1

Table 302: SWITCH_PRI_TO_CLASS_1

Name	Bit	Description	Type	Default
pri0	2: 0	Maps switch priority 0 to traffic class [0..7]	RW	0x0
pri1	5: 3	Maps switch priority 1 to traffic class [0..7]	RW	0x1
pri2	8: 6	Maps switch priority 2 to traffic class [0..7]	RW	0x2
pri3	11: 9	Maps switch priority 3 to traffic class [0..7]	RW	0x3
pri4	14: 12	Maps switch priority 4 to traffic class [0..7]	RW	0x4
pri5	17: 15	Maps switch priority 5 to traffic class [0..7]	RW	0x5
pri6	20: 18	Maps switch priority 6 to traffic class [0..7]	RW	0x6
pri7	23: 21	Maps switch priority 7 to traffic class [0..7]	RW	0x7

15.12.47 SWITCH_PRI_TO_CLASS_2

Table 303: SWITCH_PRI_TO_CLASS_2

Name	Bit	Description	Type	Default
pri8	2: 0	Maps switch priority 8 to traffic class [0..7]	RW	0x0

Name	Bit	Description	Type	Default
pri9	5: 3	Maps switch priority 9 to traffic class [0..7]	RW	0x1
pri10	8: 6	Maps switch priority 10 to traffic class [0..7]	RW	0x2
pri11	11: 9	Maps switch priority 11 to traffic class [0..7]	RW	0x3
pri12	14: 12	Maps switch priority 12 to traffic class [0..7]	RW	0x4
pri13	17: 15	Maps switch priority 13 to traffic class [0..7]	RW	0x5
pri14	20: 18	Maps switch priority 14 to traffic class [0..7]	RW	0x6
pri15	23: 21	Maps switch priority 15 to traffic class [0..7]	RW	0x7

15.12.48 CM_GLOBAL_CFG

Global Congestion Management Configuration

Table 304: CM_GLOBAL_CFG

Name	Bit	Description	Type	Default
ifgPenalty	7: 0	The additional charge in bytes to apply per-frame for the purposes of ingress or egress rate-limiting and scheduling DRR	RW	0x14
sharedPauseInterval	15: 8	Defines the number of FH clocks to wait checking RX private watermark after this watermark as caused a change in the pause state. This is to avoid repetitive pause on/off if the RX queue quickly move around the private watermarks while the switch is congested.	RW	0x0
forcePauseOn	16	Force all ports to be paused	RW	0x0
forcePauseOff	17	Force all ports to be unpaused	RW	0x0

15.12.49 TRAFFIC_CLASS_TO_SCHED_PRI

FOR FULCRUM USE ONLY This register should be reprogrammed if the network designates particular Traffic-Classes as having a different priority order than the default order enforced during Frame Scheduling (higher Traffic-Classes preferred over lower numbered Traffic-Classes)

Table 305: TRAFFIC_CLASS_TO_SCHED_PRI

Name	Bit	Description	Type	Default
pri0	2: 0	Maps traffic class 0 to scheduler priority [0..7]	RW	0x0
pri1	6: 4	Maps traffic class 1 to scheduler priority [0..7]	RW	0x1
pri2	10: 8	Maps traffic class 2 to scheduler priority [0..7]	RW	0x2
pri3	14: 12	Maps traffic class 3 to scheduler priority [0..7]	RW	0x3
pri4	18: 16	Maps traffic class 4 to scheduler priority [0..7]	RW	0x4
pri5	22: 20	Maps traffic class 5 to scheduler priority [0..7]	RW	0x5
pri6	26: 24	Maps traffic class 6 to scheduler priority [0..7]	RW	0x6

Name	Bit	Description	Type	Default
pri7	30: 28	Maps traffic class 7 to scheduler priority [0..7]	RW	0x7
reserved7	31	Maps traffic class 7 to scheduler priority [0..7]	RV	0x0

This register must be configured to implement a 1-1 mapping. Failure to do this WILL result in improper operation. Scheduler Priorities may be placed in equipriority and bandwidth sharing groups by means of SCHED_GROUP_CFG. Moving scheduler priorities while frames are enqueued for that priority will result in improper operation. Changing this register should conform to the procedure for changing CM_SMP_MEMBERSHIP.

15.12.50 CN_STATS_CFG[0..1]

Table 306: CN_STATS_CFG[0..1]

Name	Bit	Description	Type	Default
tx	4: 0		RW	0x0
smp *(was tc)*	8		RW	0x0

15.12.51 CN_STATS[0..1] [0..7]

Statistics for congestion notification. The counters are:

- cnt[1..0][0]: CN frame sent Q_OFF between 1 and 8
- cnt[1..0][1]: CN frame sent Q_OFF between 9 and 16
- cnt[1..0][2]: CN frame sent Q_OFF between 17 and 64
- cnt[1..0][3]: CN frame sent Q_OFF greater than 64
- cnt[1..0][4]: CN frame sent Q_OFF between -1 and -8
- cnt[1..0][5]: CN frame sent Q_OFF between -9 and -16
- cnt[1..0][6]: CN frame sent Q_OFF between -17 and -64
- cnt[1..0][7]: CN frame sent Q_OFF smaller than -64

Table 307: CN_STATS[0..1] [0..7]

Name	Bit	Description	Type	Default
Count	31: 0		RW	0x0

15.12.52 SCHED_FH_GROUP_CFG[0..24]

Maps traffic class to scheduling sub-group This register is identical to register SCHED_GROUP_CFG and must contain the same value for normal operation.

Table 308: SCHED_FH_GROUP_CFG[0..24]

Name	Bit	Description	Type	Default
SchedulingGroupBoundary	7: 0	Defines groups of classes that use a common a deficit counter, for the purpose of delay-deficit	RW	0xff

Name	Bit	Description	Type	Default
		round-robin scheduling classes within a scheduling group have strict priority relative to each other. There is 1 bit per class. If this bit set to 0 then the class belongs to the same group as the class to the right (the class with higher index). If this bit is 1, then the class is at the head of a new group.		
Unimplemented	22: 8		RV	0x00
StrictPriority	23: 16	Defines class priority. There is one bit per class. If this bit set to strict (1), then all packets are drained unless throughput limit is achieved. If set to (0), then packets are drained according to weight. Note that this bit must be set to the same value for all classes within the set (all 0s or all 1s).	RW	0xff

15.12.53 QDM_CFG[0..1]

Permits configuration of up to two (tx,tc) pairs whose queues will be measured for average delay per-frame.

Table 309: QDM_CFG[0..1]

Name	Bit	Description	Type	Default
tx	4: 0	A value between 0 and 24 inclusive	RW	0x01
tc	8: 6	A value between 0 and 7 inclusive	RW	0x00
w	13: 10	The weight to use while updating the exponential weighted average. D_{n+1} = D_n + (FrameDelay-D_n) >> w	RW	0x01
mode	14	Exponential Weighted Average (0) or Peak (1)	RW	0x00

15.12.54 QDM_FRAME_CNT[0..1]

Permits configuration of up to two (tx,tc) pairs whose queues will be measured for average delay per-frame.

Table 310: QDM_FRAME_CNT[0..1]

Name	Bit	Description	Type	Default
cnt	15: 0	Number of frames to sample. 0xFFFF denotes infinite. 0x0000 disables the QDM.	RW	0x00

15.12.55 QDM_INSTRUMENT[0..1]

Number of FH Ticks Average Delay. Actual internal measurement is maintained in \frac{1}{2**12} tick increments. Delay measurement describes roughly the time spent in the TX queue plus some additional signaling delays. Latency due to parsing, routing, store-and-forward behavior, and other rx effects are not tabulated. Time-spent idle does not weight into the average. Delay measurement does not occur per-frame but rather each QDM Instrument measures one frame at a time. When that frame's latency has been

estimated, estimation begins on the next new frame to entry the queue. If a frame's latency exceeds the available representation, a bit denoting that this has occurred is set and the latency is taken to have been the maximum interval.

Table 311: QDM_INSTRUMENT[0..1]

Name	Bit	Description	Type	Default
ts	30: 0	Delay in FH ticks	RW	0x00
spill	31	Timestamp wrapped during Measurement	RW	0x00

15.13 Link Aggregation Registers (reserved 4KW) Details

15.13.1 LAG_CFG[0..24]

Table 312: LAG_CFG[0..24]

Name	Bit	Description	Type	Default
LagSize	3: 0	Defines the number of ports in the link aggregation group. The value 0 means 16.	RW	0x0
Index	7: 4	Index of this port in the link aggregation port, must be between 0 and LAG_CFG:LagSize-1.	RW	0x0
HashRotation	8	Selects hash rotation A (0) or rotation B (1) for filtering.	RW	0x0
InLAG	9	Defines if the port is currently part of a LAG group (1) or not (0).	RW	0x0

15.13.2 CANONICAL_GLORT_CAM[0..15]

Table 313: CANONICAL_GLORT_CAM[0..15]

Name	Bit	Description	Type	Default
LagGlort	15: 0	Each source glort will be masked and compared to this LAG glort. If it matches, the lower PortFieldSize bits will be cleared.	RW	0x0
MaskSize	19: 16	Comparison to LagGlort is performed by first AND-ing the source glort by ~(2**MaskSize - 1).	RW	0x0
PortFieldSize	22: 20	Number of low-order bits to clear in a source glort that matches this entry's LagGlort.	RW	0x0

15.14 Policer Registers (reserved 16KW) Details

15.14.1 POLICER_TABLE[0..3] [0..511]

The FFU specifies a 9-bit index in each of 4 counter banks to be counted or policed. Each counter bank can be configured to count or police by bytes or frames. Interrupts can be raised for indicies at or above a specified minimum.

Table 314: POLICER_TABLE[0..3] [0..511]

Name	Bit	Description	Type	Default
FrameCount	63: 0	Frame count when the entry is used as a counter	RW	0x0
PolicerCommittedCurrent	27: 0	Number of 1/16th bytes that the committed rate token bucket currently contains. Signed quantity.	RW	0x0
PolicerExcessCurrent	54: 28	Number of 1/16th bytes that the excess rate token bucket currently contains. Signed quantity.	RW	0x0
PolicerCommittedAction	55	Action if committed rate excedeed (0=drop,1=mark down)	RW	0x0
PolicerCommittedRateMantissa *(was PolicerCommittedIndex)*	63: 56	Determines the refill rate of the committed rate token bucket. Policer Rate = RateMantissa * 2 $^{-RateExponent}$ [Bytes/FH_CLK]. Internally, the byte counters have a precision of 1/16 th of a byte.	RW	0x0
ByteCount	127: 64	Byte count when the entry is used as a counter	RW	0x0
PolicerCommittedRateExponent	67: 64	Determines the refill rate of the committed rate token bucket. Policer Rate = RateMantissa * 2 $^{-RateExponent}$ [Bytes/FH_CLK]. Internally, the byte counters have a precision of 1/16 th of a byte.	RW	0x0
PolicerCommittedLimit	80: 68	Capacity of committed rate token bucket in 1024 byte units.	RW	0x0
PolicerExcessLimit	92: 81	Capacity of excess rate token bucket in 1024 byte units.	RW	0x0
PolicerExcessAction	93	Action if excess rate excedeed (0=drop,1=mark down).	RW	0x0
PolicerExcessRateMantissa *(was PolicerExcessIndex)*	101: 94	Determines the refill rate of the excess rate token bucket. Policer Rate = RateMantissa * 2 $^{-RateExponent}$ [Bytes/FH_CLK].	RW	0x0

Name	Bit	Description	Type	Default
		Internally, the byte counters have a precision of 1/16 th of a byte.		
PolicerExcessRateExponent	105: 102	Determines the refill rate of the excess rate token bucket. Policer Rate = RateMantissa * 2 $^{-RateExponent}$ [Bytes/FH_CLK]. Internally, the byte counters have a precision of 1/16 th of a byte.	RW	0x0
PolicerTimestamp	121: 106	Timestamp of the last update. Timestamp ticks at the Frame Handler frequency.	RW	0x0

The minimum configurable policer rate is 46Mb/s due to rounding truncation in hardware. Since the rate equation is:

rate = m*2^(-e)*FH_CLK (Bytes/sec)

The following equation must be satisfied:

rate = 8*m*2^(-e)*FH_CLK >= 46Mb/s

where:

- m is the mantissa and has a range of 0 to 255
- e is the exponent and has a range of 0 to 15

If the above is not satisfied, the resulting rate may be close or equal to 0 Mb/s.

For example, with FH_CLK = 371MHz the values: m=128, e=13 work, and result in a rate of slightly greater than 46Mb/s.

15.14.2 POLICER_REPAIR[0..3] [0..1]

Table 315: POLICER_REPAIR[0..3] [0..1]

Name	Bit	Description	Type	Default
REPAIR_ADDR	3: 0	Defines per-bank, per-half-entry which address to repair	RW	0x0

15.14.3 POLICER_CFG[0..3]

Table 316: POLICER_CFG[0..3]

Name	Bit	Description	Type	Default
IndexLastPolicer	9: 0	Defines the index of the last policer. Setting this value to 0 means that there are no policers in this bank.	RW	0x0

Name	Bit	Description	Type	Default
IndexLastCountNoInt	19: 10	Defines the index of the last counter without interrupts. If the index supplied through the FFU action is above this limit then the interrupt pending bit for this bank will be set when the counter is incremented	RW	0x0
IngressColorSource	21: 20	Defines what defines the color. The choices are 0=DSCP, 1=SwitchPRI, 2=NoColor (assume green).	RW	0x0
MarkDSCP	22	Defines if DSCP field is changed when color is changed or not.	RW	0x0
MarkSwitchPri	23	Defines if Switch Priority is changed when color is changed or not.	RW	0x0
TimestampWriteEnable	24	Defines whether the timestamp field of a policing entry in the policer table is overwritten with the value from the management write. Otherwise if set to 0, the current timestamp will be written when the management writes to the policer entry.	RW	0x0

15.14.4 POLICER_IP

Table 317: POLICER_IP

Name	Bit	Description	Type	Default
InterruptPending	3: 0	Defines if there is an interrupt pending for each bank (one bit per bank). Writing clears the register.	CW1	0x0

15.14.5 POLICER_IM

Table 318: POLICER_IM

Name	Bit	Description	Type	Default
InterruptMask	3: 0	Defines if the interrupts from each bank is masked (1) or passed through (0). There is one bit per bank.	RW	0x0f

15.14.6 POLICER_STATUS

Table 319: POLICER_STATUS

Name	Bit	Description	Type	Default
Timestamp	15: 0	Current policer timestamp value.	RO	0x0

15.14.7 POLICER_DSCP_DOWN_MAP[0..63]

Table 320: POLICER_DSCP_DOWN_MAP[0..63]

Name	Bit	Description	Type	Default
NewDSCP	5: 0	Defines new downgrade DSCP. Only used if the policer is configured to downgrade the DSCP.	RW	0x0

15.14.8 POLICER_SWPRI_DOWN_MAP[0..15]

Table 321: POLICER_SWPRI_DOWN_MAP[0..15]

Name	Bit	Description	Type	Default
NewSwitchPriority	3: 0	Defines new downgrade Switch Priority. Only used if the policer is configured to downgrade the switch priority.	RW	0x0

15.15 Triggers Registers (reserved 4KW) Details

15.15.1 TRIGGER_CONDITION_CFG[0..63]

Table 322: TRIGGER_CONDITION_CFG[0..63]

Name	Bit	Description	Type	Default
MatchSA	1: 0	Match frame's source MAC address against the SA_ID field in TRIGGER_CONDITION_PARAM.	RW	0x2
MatchDA	3: 2	Match frame's destination MAC address against the DA_ID field in TRIGGER_CONDITION_PARAM.	RW	0x2
MatchHitSA	5: 4	Match depending on whether a source MAC address lookup was performed and was found to exist in the table.	RW	0x2
MatchHitDA	7: 6	Match depending on whether the destination MAC address was found in the table.	RW	0x2
MatchHitSADA	9: 8	Match depending on whether the source *or* destination MAC address was found in the table (or both).	RW	0x2
MatchVlan	11: 10	Match depending on whether the TrigID from the VID_TABLE lookup matches VID_ID from TRIGGER_CONDITION_PARAM.	RW	0x2
MatchFFU	13: 12	Match depending on whether the TrigID from the FFU's SET_TRIG action matches FFU_ID, when both are ANDed with FFU_Mask (See TRIGGER_CONDITION_FFU register).	RW	0x2

Name	Bit	Description	Type	Default
MatchSwitchPri	15: 14	Match if the frame's switch priority matches SwitchPri from TRIGGER_CONDITION_PARAM.	RW	0x2
MatchEtherType	17: 16	Match if the frame's first non-VLAN EtherType matches the EtherType parameter in TRIGGER_CONDITION_TYPE when both are ANDed with EtherTypeMask.	RW	0x2
MatchDestGlort	19: 18	Match if the destination glort associated with the frame matches DestGlort when both are ANDed with GlortMask (See TRIGGER_CONDITION_GLORT register).	RW	0x2
MatchByPrecedence	20	If 0, begins a new precedence-evaluated trigger set, evaluated in parallel with all other precedence-evaluated sets.	RW	0x0
MatchRandomNumber	21	Specifies one of two random number generators to compare against.	RW	0x0
MatchRandomIfLess	22	When set to 1, matches if 'random is less than or equal to 2^MatchRandomThreshold'; otherwise, matches if 'random is greater than or equal to 2^MatchRandomThreshold'.	RW	0x1
MatchRandomThreshold	27: 23	Exponent used to calculate the threshold for rate-constrained matching. Threshold equals 2^MatchRandomThreshold.	RW	0x18

Each 2-bit condition field specifies one of three defined match cases:

- 0x0 - Associated frame property must *not* equal the corresponding trigger parameter in order to match.
- 0x1 - Associated frame property must equal the corresponding trigger parameter in order to match.
- 0x2 - Match unconditionally.

A configuration value of 0x3 is reserved and will cause undefined behavior if selected.

15.15.2 TRIGGER_CONDITION_PARAM[0..63]

Table 323: TRIGGER_CONDITION_PARAM[0..63]

Name	Bit	Description	Type	Default
SA_ID	5: 0	Source MAC address ID, compared against TrigID from the source address' entry in MA_TABLE.	RW	0x0
DA_ID	11: 6	Destination MAC address ID, compared against TrigID from the destination address' entry in MA_TABLE.	RW	0x0
VID_ID	17: 12	VLAN ID, compared against TrigID from the VID_TABLE lookup.	RW	0x0
SwitchPri	21: 18	Trigger parameter to compare against the frame's associated switch priority.	RW	0x0

Name	Bit	Description	Type	Default
FrameClassMask	24: 22	If bit 22 is set, the trigger will match against unicast frames. If bit 23 is set, the trigger will match against broadcast frames. If bit 24 is set, the trigger will match against multicast frames.	RW	0x7
RoutedMask	26: 25	If bit 25 is set, trigger will match on switched frames. If bit 26 is set, trigger will match on routed frames.	RW	0x3
FtypeMask	30: 27	FtypeMask[i]==1 implies Trigger will match frames with FTYPE==i.	RW	0x3
MatchDroppedFrames	31	When MatchDroppedFrame==0 then TRIGGER_CONDITION_TX is ANDed with the destination mask, and the trigger matches if the result is nonzero. When MatchDroppedFrame==1, the destination mask must be EQUAL to TRIGGER_CONDITION_TX. This allows matching dropped frames by setting TRIGGER_CONDITION_TX to 0.	RW	0x0

15.15.3 TRIGGER_CONDITION_FFU[0..63]

Table 324: TRIGGER_CONDITION_FFU[0..63]

Name	Bit	Description	Type	Default
FFU_ID	7: 0	ID to compare against the TrigID specified by an FFU SET_TRIG action. Note that 0 is a reserved value (it is the default value assigned by the FFU to the FFU_ID) and should not be used for normal use.	RW	0x0
FFU_Mask	15: 8	Mask to be ANDed with both FFU_ID and the TrigID specified by an FFU SET_TRIG action prior to comparison.	RW	0x0

15.15.4 TRIGGER_CONDITION_TYPE[0..63]

Table 325: TRIGGER_CONDITION_TYPE[0..63]

Name	Bit	Description	Type	Default
EtherType	15: 0	Type value to compare against the frame's first non-VLAN EtherType.	RW	0x0
EtherTypeMask	31: 16	Mask to be ANDed with both the EtherType trigger parameter and the frame's first non-VLAN EtherType prior to comparison.	RW	0x0

15.15.5 TRIGGER_CONDITION_GLORT[0..63]

Table 326: TRIGGER_CONDITION_GLORT[0..63]

Name	Bit	Description	Type	Default
DestGlort	15: 0	Glort value to compare against the frame's destination glort.	RW	0x0
GlortMask	31: 16	Mask to be ANDed with both the DestGlort trigger parameter and the frame's destination glort prior to comparison.	RW	0x0

15.15.6 TRIGGER_CONDITION_RX[0..63]

The source port mask is unconditionally ANDed with 2^{src_port}. The result must be nonzero in order for this condition to match.

Table 327: TRIGGER_CONDITION_RX[0..63]

Name	Bit	Description	Type	Default
SrcPortMask	24: 0	Source port mask. The bit corresponding to the frame's source port must be set in order for the trigger to match.	RW	0x0

15.15.7 TRIGGER_CONDITION_TX[0..63]

For frames that are not dropped, the configured destination port mask is unconditionally ANDed with the frame's destination mask. The result must be nonzero in order for this condition to match. For a dropped frame, the destination port mask is changed to 0 after the Action Resolution stage of the frame processing pipeline. As a result when it arrives at the trigger unit, to generate a match, set both triggerConditionTx = 0 and triggerConditionParam.matchDroppedFrames==1.

Table 328: TRIGGER_CONDITION_TX[0..63]

Name	Bit	Description	Type	Default
DestPortMask	24: 0	Destination port mask. At least one bit set in the frame's destination mask must also be set in DestPortMask in order for the trigger to match.	RW	0x1FFFFFF

15.15.8 TRIGGER_CONDITION_AMASK_1[0..63]

****** This register was named TRIGGER_CONDITION_AMASK ******

Table 329: TRIGGER_CONDITION_AMASK_1[0..63]

Name	Bit	Description	Type	Default
HandlerActionMask	31: 0	At least one bit from the frame's handler action mask must be set in HandlerActionMask in	RW	0x00000000

Name	Bit	Description	Type	Default
		order for the trigger to match. Corresponds to action cases 0 through 31. See chapter 2, Overview, for the list of action codes 0 - 31.		

15.15.9 TRIGGER_CONDITION_AMASK_2[0..63]

Table 330: TRIGGER_CONDITION_AMASK_2[0..63]

Name	Bit	Description	Type	Default
HandlerActionMask	12: 0	At least one bit from the frame's handler action mask must be set in HandlerActionMask in order for the trigger to match. Corresponds to action cases 32 through 44. See chapter 2, Overview, for the list of action codes 0 - 37. The log codes, 38 - 44, are as follows: • 38: Header Timeout - Header too long to parse. • 39: Log Ingress FFU - Frame logged to CPU due to ingress FFU action. • 40: Log Egress FFU - Frame logged to egress FFU action. • 41: Mirror FFU - Frame is Rx mirrored due to FFU action. • 42: Log L3Uni/L2Multi - Frame logged to CPU due to combination of a mcast L2 destination address with a L3 ucast destination address. • 43: Log ICMP Mcast TTL - ICMP Multicast frame logged to CPU because TTL is less than or equal to 1. • 44: Log Mcast TTL - IP Multicast frame logged to CPU because TTL is less than or equal to 1.	RW	0x30

15.15.10 TRIGGER_ACTION_CFG_1[0..63]

Table 331: TRIGGER_ACTION_CFG_1[0..63]

Name	Bit	Description	Type	Default
ForwardingAction	1: 0	Forwarding action (0: Leave as-is, 1: Forward, 2: Redirect, 3: Drop.)	RW	0x0
TrapAction	3: 2	Trap action (0: Leave as-is, 1: Trap, 2: Log, 3: Do not Trap or Log)	RW	0x0
MirroringAction	4	Mirroring action (0: Leave as-is, 1: Mirror.)	RW	0x0
SwitchPriAction	5	Switch Priority modification action (0: Leave as-is, 1: Reassign.)	RW	0x0

Name	Bit	Description	Type	Default
VlanAction	6	Egress VLAN modification action (0: Leave as-is, 1: Reassign.)	RW	0x0
LearningAction	8: 7	Learning action (0: Leave as-is, 1: Do not learn, 2: Force learning.) Note: an action value of '2' will force the SMAC address to be learned only if an SMAC lookup was performed and the lookup missed or returned a non-locked entry. The entry will be learned (or refreshed) with a state value of '2' ("young").	RW	0x0
RateLimitAction	9	Rate limiter action (0: Leave as-is, 1: Apply rate limit.)	RW	0x0

15.15.11 TRIGGER_ACTION_CFG_2[0..63]

Table 332: TRIGGER_ACTION_CFG_2[0..63]

Name	Bit	Description	Type	Default
CpuTruncate	0	Enable truncation on frames sent to the CPU due to the Trap or Log action.	RW	0x0
MirrorTruncate	1	Enable truncation on frames mirrored due to MirrorAction.	RW	0x0
NewSwitchPri	5: 2	New switch priority (applicable when SwitchPriAction is 1.)	RW	0x0
NewVlan	17: 6	New egress VLAN (applicable when VlanAction is 1.)	RW	0x0
RateLimitNum	21: 18	Rate limiter number (applicable when RateLimitAction is 1.)	RW	0x0

15.15.12 TRIGGER_ACTION_GLORT[0..63]

Table 333: TRIGGER_ACTION_GLORT[0..63]

Name	Bit	Description	Type	Default
NewDestGlort	15: 0	New destination glort when ForwardingAction is 1 or 2.	RW	0x0
NewDestGlortMask	31: 16	Only bits that are set to 1 will be overwritten by NewDestGlort.	RW	0x0

15.15.13 TRIGGER_ACTION_DMASK[0..63]

Table 334: TRIGGER_ACTION_DMASK[0..63]

Name	Bit	Description	Type	Default
NewDestMask	24: 0	New port destination mask when ForwardingAction is 2.	RW	0x0

Name	Bit	Description	Type	Default
FilterDestMask	25	Destination mask will be LAG-filtered when set to 1 when ForwardingAction is 2.	RW	0x1

15.15.14 TRIGGER_ACTION_MIRROR[0..63]

Table 335: TRIGGER_ACTION_MIRROR[0..63]

Name	Bit	Description	Type	Default
MirrorPort	4: 0	Port to send a mirrored copy of the frame to when MirroringAction is 1.	RW	0x0
MirrorGlort	20: 5	Destination glort to attach to the mirrored frame when MirroringAction is 1.	RW	0x0

15.15.15 TRIGGER_ACTION_DROP[0..63]

The drop mask is applied to the frame's destination mask when ForwardingAction in TRIGGER_ACTION_CFG_1 is set to 3 (drop).

Table 336: TRIGGER_ACTION_DROP[0..63]

Name	Bit	Description	Type	Default
DropMask	24: 0	Bits set to 1 will be cleared in the frame's destination mask.	RW	0x0

15.15.16 TRIGGER_RATE_LIM_CFG_1[0..15]

Table 337: TRIGGER_RATE_LIM_CFG_1[0..15]

Name	Bit	Description	Type	Default
Capacity	11: 0	Capacity of rate limiter token bucket in 1024 byte units. (Maximum burst allowance is 4 MB, or 2MB for near-maximal rate limits.)	RW	0x0
RateMantissa	23: 12	Floating point mantissa of the token bucket's refill rate. Amount of refill is (RateMantissa * 32) >> (RateExponent+2) per 16 FH clock cycles, in units of 1/16th bytes.	RW	0x0
RateExponent	25: 24	Floating point exponent of the token bucket's refill rate. Amount of refill is (RateMantissa * 32) >> (RateExponent+2) per 16 FH clock cycles, in units of 1/16th bytes.	RW	0x0

15.15.17 TRIGGER_RATE_LIM_CFG_2[0..15]

Table 338: TRIGGER_RATE_LIM_CFG_2[0..15]

Name	Bit	Description	Type	Default
DropMask	24: 0	Drop mask to apply to the frame when the rate limiter exceeds its configured rate.	RW	0x0

15.15.18 TRIGGER_RATE_LIM_USAGE[0..15]

Table 339: TRIGGER_RATE_LIM_USAGE[0..15]

Name	Bit	Description	Type	Default
Level	26: 0	Current rate limiter token bucket level in units of 1/16th bytes. Value is signed two's complement notation.	RO	0x0

15.15.19 TRIGGER_IP[0..1]

Table 340: TRIGGER_IP[0..1]

Name	Bit	Description	Type	Default
Pending	31: 0	Interrupt pending bits, set whenever the corresponding trigger fires.	CW1	(0..1) 0x0

15.15.20 TRIGGER_IM[0..1]

Table 341: TRIGGER_IM[0..1]

Name	Bit	Description	Type	Default
Mask	31: 0	Interrupt mask bits.	RW	(0..1) 0xffffffff

15.16 ARP Control and Table (128KW) Details

15.16.1 ARP_TABLE[0..16383]

Note that these registers apply to the FM3000 even though routing is not done in these devices. ARP is used for load balancing purposes. Also, ARP table entry 0 is reserved and cannot be used for routing.

Table 342: ARP_TABLE[0..16383]

Name	Bit	Description	Type	Default
DMAC	47: 0	Destination MAC address if type is MAC	RW	0x0
dstGlort	15: 0	Destination GLORT if type is GLORT	RW	0x0

Name	Bit	Description	Type	Default
MTU_Index	18: 16	Defines the MTU size for this VLAN through a 3-bit index into MTU_TABLE. This superseed the MTU size index defined in the EGRESS_VID_TABLE.	RW	0x0
VLAN	59: 48	Destination VLAN if type is MAC	RW	0x0
type	61: 60	Type of entry: 0=MAC, 1=GLORT, 2=From IPv6	RW	0x0
reserved	62	Reserved	RV	0x0
parityError	63	Parity error bit. Setting to 0 on writing forces the hardware to set the proper parity, setting to 1 on writing forces the hardware to set bad parity. Reading 0 indicates the parity was good, reading 1 indicates that the parity was bad.	RW	0x0
VRID	71: 64	Destination Virtual Router ID	RW	0x0

15.16.2 ARP_USED[0..511]

Table 343: ARP_USED[0..511]

Name	Bit	Description	Type	Default
used	31: 0	32 consecutive bits marking usage of corresponding ARP_TABLE entries.	CW1	0x0

Each time ARP_TABLE[index] entry is used to forward a frame, the chip sets the ARP_USED[index{13:5}]{index{4:0}} bit to 1. When software writes a word to this table, any bits that are 1 in the word will clear the corresponding bit in ARP_USED to 0. The software should read a word into x, record which entries have been used since the last sample, and then write x back to ARP_USED to clear the bits. This avoids any race between the hardware and software. The concept is similar to an interrupt register.

15.16.3 ARP_IP

Table 344: ARP_IP

Name	Bit	Description	Type	Default
Redirect	0	Indicates if there is a redirect event posted or not. This is logged only if the ingress VLAN is equal to the EGRESS VLAN for unicast IP frames only.	CW1	0x0

15.16.4 ARP_IM

Table 345: ARP_IM

Name	Bit	Description	Type	Default
Mask	0	Interrupt mask	RW	0x1

15.16.5 ARP_REDIRECT_SIP

Defines the source address of the IP frame that exit on the same VLAN it came from which can potentially be redirected.

Table 346: ARP_REDIRECT_SIP

Name	Bit	Description	Type	Default
SIP	127: 0	Source IP address.	RW	0x0

15.16.6 ARP_REDIRECT_DIP

Defines the destination address of the IP frame that exit on the same VLAN it came from which can potentially be redirected.

Table 347: ARP_REDIRECT_DIP

Name	Bit	Description	Type	Default
SIP	127: 0	Source IP address.	RW	0x0

15.17 Filtering and Forwarding Unit Registers (256KW address space) Details

The Filter and Forward Unit. Its primary purpose is to classify the header using programmable mappings followed by a ternary CAM lookup. Result is a set of actions to perform, such as a next-hop forwarding router or permit/deny. Also generates portmasks for egress ACL actions.

15.17.1 FFU_MAP_SRC[0..24]

Table 348: FFU_MAP_SRC[0..24]

Name	Bit	Description	Type	Default
MAP_SRC	3: 0	Resulting MAP_SRC per source port	RW	0x0
Routable	4	FM4000 devices only. Enable IP routing on this source port	RW	0x0

15.17.2 FFU_MAP_MAC[0..15]

Table 349: FFU_MAP_MAC[0..15]

Name	Bit	Description	Type	Default
MAC	47: 0	Ethernet MAC Key	RW	0x0
IgnoreLength	55: 48	Ignore this many LSB bits when matching	RW	0x0
validSMAC	56	Entry valid for MAP_SMAC	RW	0x0

Name	Bit	Description	Type	Default
validDMAC	57	Entry valid for MAP_DMAC	RW	0x0
MAP_MAC	61: 58	4-bit mapped MAC	RW	0x0
Router	62	FM4000 devices only. Is this a virtual router destination MAC?	RW	0x0

15.17.3 FFU_MAP_VLAN_REPAIR

Table 350: FFU_MAP_VLAN_REPAIR

Name	Bit	Description	Type	Default
REPAIR_ADDR	11: 0	Address to repair in VLAN sram	RW	0x0

15.17.4 FFU_MAP_VLAN[0..4095]

Table 351: FFU_MAP_VLAN[0..4095]

Name	Bit	Description	Type	Default
MAP_VLAN	11: 0	The mapped VLAN	RW	0x0
Routable	12	FM4000 devices only. Enable IP routing on this VLAN	RW	0x0
ParityError	13	Encodes parity error when read. If written with a 1, will write bad parity. OR'd with ParityError bits in FFU_SLICE_SRAM	RW	0x0

15.17.5 FFU_MAP_TYPE[0..15]

Table 352: FFU_MAP_TYPE[0..15]

Name	Bit	Description	Type	Default
TYPE	15: 0	EtherType Key	RW	0x0
MAP_TYPE	19: 16	4-bit mapped EtherType	RW	0x0

15.17.6 FFU_MAP_LENGTH[0..15]

Table 353: FFU_MAP_LENGTH[0..15]

Name	Bit	Description	Type	Default
LENGTH	15: 0	IP Packet Length Key	RW	0x0
MAP_LENGTH	19: 16	4-bit MAP_LENGTH	RW	0x0

15.17.7 FFU_MAP_IP_LO[0..15]

Table 354: FFU_MAP_IP_LO[0..15]

Name	Bit	Description	Type	Default
IP_LO	63: 0	Low 64 bits of IP address prefix key	RW	0x0

15.17.8 FFU_MAP_IP_HI[0..15]

Table 355: FFU_MAP_IP_HI[0..15]

Name	Bit	Description	Type	Default
IP_HI	63: 0	High 64 bits of IP address prefix key	RW	0x0

15.17.9 FFU_MAP_IP_CFG[0..15]

Table 356: FFU_MAP_IP_CFG[0..15]

Name	Bit	Description	Type	Default
IgnoreLength	7: 0	Ignore this many LSB bits when matching	RW	0x0
validSIP	8	Entry valid for MAP_SIP	RW	0x0
validDIP	9	Entry valid for MAP_DIP	RW	0x0
MAP_IP	13: 10	4-bit mapped IP address	RW	0x0

15.17.10 FFU_MAP_PROT[0..7]

Table 357: FFU_MAP_PROT[0..7]

Name	Bit	Description	Type	Default
PROT	7: 0	Input IP protocol number	RW	0x0
MAP_PROT	10: 8	3-bit mapped IP protocol number	RW	0x0

15.17.11 FFU_MAP_L4_SRC[0..63]

Table 358: FFU_MAP_L4_SRC[0..63]

Name	Bit	Description	Type	Default
L4SRC	15: 0	L4 source port key	RW	0x0
MAP_PROT	18: 16	3-bit MAP_PROT	RW	0x0
VALID	19	Valid mapping	RW	0x0
MAP_L4SRC	47: 32	Mapped result	RW	0x0

15.17.12 FFU_MAP_L4_DST[0..63]

Table 359: FFU_MAP_L4_DST[0..63]

Name	Bit	Description	Type	Default
L4DST	15: 0	L4 destination port key	RW	0x0
MAP_PROT	18: 16	3-bit MAP_PROT	RW	0x0
VALID	19	Valid mapping	RW	0x0
MAP_L4DST	47: 32	Mapped result	RW	0x0

15.17.13 FFU_INIT_SLICE

Table 360: FFU_INIT_SLICE

Name	Bit	Description	Type	Default
ActiveSlices	5: 0	Indicates the number of active slices. The number must be between 0 (no active slice) and 32 (max active slice).	RW	0x20
Repair1	11: 6	Indicates the first physical slice that will be skipped at initialization.	RW	0x3F
Repair2	17: 12	Indicates the second physical slice that will be skipped at initialization.	RW	0x3F

15.17.14 FFU_MASTER_VALID

Used to allow atomic and non-disruptive changes of ingress and egress ACL's. Writing this register atomically disables certain slices or chunks and enables other slices or chunks. Applies to all scenarios. Invalid slices pass all cascaded hit, index, raw hit, and actions through unmodified to the next slice. Invalid chunks do likewise.

Table 361: FFU_MASTER_VALID

Name	Bit	Description	Type	Default
SliceValid	31: 0	Bitmask indicating all valid FFU slices	RW	0x~~FFFFFFFF~~
ChunkValid	63: 32	Bitmask indicating all valid egress chunks	RW	0x~~FFFFFFFF~~

15.17.15 FFU_EGRESS_CHUNK_CFG[0..31]

Table 362: FFU_EGRESS_CHUNK_CFG[0..31]

Name	Bit	Description	Type	Default
DstPortMask	24: 0	Bitmask of destination ports to apply to	RW	0x0
StartCascade	31	Starts a priority hit cascade across chunks	RW	0x0

15.17.16 FFU_EGRESS_CHUNK_VALID[0..31]

Contains a valid bit for each of 32 possible scenarios. The FFU_MASTER_VALID bit for this chunk must also be true. If invalid for either reason, no rules in the chunk can hit, and any cascaded hit will ripple through unmodified.

Table 363: FFU_EGRESS_CHUNK_VALID[0..31]

Name	Bit	Description	Type	Default
Valid	31: 0	Valid bit-mask for each of 32 scenarios	RW	0x~~FFFFFFFF~~

15.17.17 FFU_EGRESS_ACTIONS[0..511]

Table 364: FFU_EGRESS_ACTIONS[0..511]

Name	Bit	Description	Type	Default
Drop	0	Drop on specified egress ports	RW	0x0
Log	1	Log to CPU on specified egress ports	RW	0x0
Count	2	Count bytes and frames on specified egress ports	RW	0x0

15.17.18 FFU_SLICE_TCAM[0..31] [0..511]

Configures an entry in the TCAM. The TCAM entries are readable but the key returned will be masked out according to the mask originally written along with the key.

Note that the case bit is OR'd with !Valid before being written into TCAM

A match occurs when the data searched is equal to the key and the entry is valid. Both the data and the key are ANDed with the mask prior to comparison. Note that a mask of 0 (ignore all bits) will cause the match to occur regardless of the value or key.

The highest index has precedence in case there is more than one match in the same slice.

Note: If the Valid bit is 0, the Case bit will read as 1.

Table 365: FFU_SLICE_TCAM[0..31] [0..511]

Name	Bit	Description	Type	Default
KeyLow	31: 0	TCAM key[31:0]	RW	0x0
MaskLow	63: 32	TCAM mask[31:0]	RW	0x0
KeyTop	67: 64	TCAM key[35:32]	RW	0x0
MaskTop	71: 68	TCAM mask[35:32]	RW	0x0
Valid	72	TCAM valid bit	RW	0x0
Case	73	TCAM case bit	RW	0x1

15.17.19 FFU_SLICE_SRAM[0..31] [0..511]

Table 366: FFU_SLICE_SRAM[0..31] [0..511]

Name	Bit	Description	Type	Default
RouteData	21: 0	Defines data when Command==Route as follows: • Bit 21: route type: 1=glort, 0=arp • if type ARP: o Bits 13:0: ARP_INDEX, defines the base index in the arp table for the first entry. o Bits 17:14: ARP_COUNT, defines the number of entries in the arp table for this route. A value of 0 means 16. Typical value of 1 for single route and different than 1 when there is more than one equivalent route (ECMP). o Bits 20:18: reserved • if type glort: o Bits 15:0: destination glort. o Bits 20:16: reserved	RW	0x0
ShortData	15: 0	16-bit data for Action A or Action B.	RW	0x0
ByteMask	7: 0	8-bit mask for FLAGS, TRIG, USR actions. Flags bits 0..4 are DENY, TRAP, LOG, NOROUTE, RX_MIRROR.	RW	0x0
VLAN	11: 0	Data used to set VLAN.	RW	0x0
DSCP	5: 0	Data used to set DSCP.	RW	0x0
ByteData	15: 8	8-bit value for FLAGS, TRIG, USR actions. Flags bits 0..4 are DENY, TRAP, LOG, NOROUTE, RX_MIRROR.	RW	0x0
PRI	15: 12	Data used to set PRI or VPRI.	RW	0x0
SubCommand	21: 16	Additional 6-bits of control when Command==BitSet: • 0x0 - Change FLAG field • 0x1 - Change TRIG field • 0x2 - Change USR field • 0x3 - Action A • 0x4 - Action B	RW	0x0
SetDSCP	16	Set DSCP field when Command==FieldSet.	RW	0x0
SetVLAN	17	Set VLAN field when Command==FieldSet.	RW	0x0
SetVPRI	18	Set VPRI field when Command==FieldSet.	RW	0x0
SetPRI	19	Set PRI field when Command==FieldSet.	RW	0x0

Name	Bit	Description	Type	Default
Command	23: 22	Command encoding: • 0x0 - Nop • 0x1 - Route • 0x2 - BitSet • 0x3 - FieldSet	RW	0x0
CounterIndex	33: 24	Counter index	RW	0x0
CounterBank	35: 34	Counter bank	RW	0x0
Precedence	38: 36	Precedence of actions	RW	0x0
ParityError	39	Encodes parity error when read. If written with a 1, will write bad parity.	RW	0x0

15.17.20 FFU_SLICE_VALID[0..31]

Contains a valid bit for each of 32 possible scenarios. The FFU_MASTER_VALID bit for this slice must also be true. If invalid for either reason, no rules in the slice can hit, and all comparisons or actions are cascaded through unmodified.

Table 367: FFU_SLICE_VALID[0..31]

Name	Bit	Description	Type	Default
Valid	31: 0	Valid bit-mask for each of 32 scenarios	RW	0x0

15.17.21 FFU_SLICE_CASE[0..31]

Table 368: FFU_SLICE_CASE[0..31]

Name	Bit	Description	Type	Default
Case	31: 0	Case bit-mask for each of 32 scenarios	RW	0x0

15.17.22 FFU_SLICE_CASCADE_ACTION[0..31]

Table 369: FFU_SLICE_CASCADE_ACTION[0..31]

Name	Bit	Description	Type	Default
CascadeAction	31: 0	CascadeAction bit-mask for each of 32 scenarios	RW	0x0

15.17.23 FFU_SLICE_CASE_CFG[0..31] [0..1]

Table 370: FFU_SLICE_CASE_CFG[0..31] [0..1]

Name	Bit	Description	Type	Default
Select0	5: 0	Selects byte 0 of the lookup key and mask	RW	0x0

Name	Bit	Description	Type	Default
Select1	11: 6	Selects byte 1 of the lookup key and mask	RW	0x0
Select2	17: 12	Selects byte 2 of the lookup key and mask	RW	0x0
Select3	23: 18	Selects byte 3 of the lookup key and mask	RW	0x0
SelectTop	29: 24	Selects top 4 bits of the lookup key and mask	RW	0x0
StartCompare	30	Starts a compare cascade	RW	0x0
StartAction	31	Starts an action cascade	RW	0x0

The following table lists the choices of keys controlled by the 5 Select fields. Some bytes are concatenated from two keys with {A,B}, where the bits of A are the most significant bits.

Select	TCAM[35:32] Top 4 bits	TCAM[31:24] Byte 3	TCAM[23:16] Byte 2	TCAM[15:8] Byte 1	TCAM[7:0] Byte 0
0	MAP_SIP[3:0]	DIP[31:24]	DIP[23:16]	DIP[15:8]	DIP[7:0]
1	MAP_DIP[3:0]	DIP[63:56]	DIP[55:48]	DIP[47:40]	DIP[39:32]
2	VLAN_VPRI[15:12]	DIP[95:88]	DIP[87:80]	DIP[79:72]	DIP[71:64]
3	DGLORT[15:12]	DIP[127:120]	DIP[119:112]	DIP[111:104]	DIP[103:96]
4	MAP_VLAN[3:0]	DMAC[31:24]	DMAC[23:16]	DMAC[15:8]	DMAC[7:0]
5	MAP_VLAN[7:4]	DMAC[47:40]	DMAC[39:32]	DMAC[31:24]	DMAC[23:16]
6	MAP_VLAN[11:8]	DMAC[15:8]	DMAC[7:0]	DMAC[47:40]	DMAC[39:32]
7	0	SGLORT[15:8]	SGLORT[7:0]	DGLORT[15:8]	DGLORT[7:0]
8	0	ISL[63:56]	ISLUSER[7:0]	SGLORT[15:8]	SGLORT[7:0]
9	0	DGLORT[15:8]	DGLORT[7:0]	ISL[63:56]	ISLUSER[7:0]
10	0	VLAN_VPRI[15:8]	VLAN_VPRI[7:0]	VLAN_VPRI[15:8]	VLAN_VPRI[7:0]
11	0	L4DST[15:8]	L4DST[7:0]	L4DST[15:8]	L4DST[7:0]
12	0	L4SRC[15:8]	L4SRC[7:0]	L4SRC[15:8]	L4SRC[7:0]
13	0	TOS[7:0]	TOS[7:0]	TOS[7:0]	TOS[7:0]
14	0	L4A[15:8]	L4A[7:0]	L4A[15:8]	L4A[7:0]
15	0	L4B[15:8]	L4B[7:0]	L4B[15:8]	L4B[7:0]
16	0	IP_LENGTH[15:8]	IP_LENGTH[7:0]	IP_LENGTH[15:8]	IP_LENGTH[7:0]
17	0	{MAP_DIP[3:0], MAP_SIP[3:0]}	{MAP_DIP[3:0], MAP_SIP[3:0]}	{MAP_DIP[3:0], MAP_SIP[3:0]}	{MAP_DIP[3:0], MAP_SIP[3:0]}
18	0	{VLAN_VPRI[15:12], MAP_VLAN[11:8]}	MAP_VLAN[7:0]	{VLAN_VPRI[15:12], MAP_VLAN[11:8]}	MAP_VLAN[7:0]
19-30	0	0	0	0	0
31	ANY	ANY	ANY	ANY	ANY
32	MAP_SRC[3:0]	SIP[31:24]	SIP[23:16]	SIP[15:8]	SIP[7:0]
33	MAP_TYPE[3:0]	SIP[63:56]	SIP[55:48]	SIP[47:40]	SIP[39:32]
34	MAP_SMAC[3:0]	SIP[95:88]	SIP[87:80]	SIP[79:72]	SIP[71:64]

Select	TCAM[35:32] Top 4 bits	TCAM[31:24] Byte 3	TCAM[23:16] Byte 2	TCAM[15:8] Byte 1	TCAM[7:0] Byte 0
35	MAP_DMAC[3:0]	SIP[127:120]	SIP[119:112]	SIP[111:104]	SIP[103:96]
36	MAP_PROT[3:0]	SMAC[31:24]	SMAC[23:16]	SMAC[15:8]	SMAC[7:0]
37	MAP_LENGTH[3:0]	SMAC[47:40]	SMAC[39:32]	SMAC[31:24]	SMAC[23:16]
38	0	SMAC[15:8]	SMAC[7:0]	SMAC[47:40]	SMAC[39:32]
39	0	TYPE[15:8]	TYPE[7:0]	TYPE[15:8]	TYPE[7:0]
40	0	MAP_L4DST[15:8]	MAP_L4DST[7:0]	MAP_L4DST[15:8]	MAP_L4DST[7:0]
41	0	MAP_L4SRC[15:8]	MAP_L4SRC[7:0]	MAP_L4SRC[15:8]	MAP_L4SRC[7:0]
42	0	PROT[7:0]	PROT[7:0]	PROT[7:0]	PROT[7:0]
43	0	{SCENARIO[4:0], MISC[2:0]}	{SCENARIO[4:0], MISC[2:0]}	{SCENARIO[4:0], MISC[2:0]}	{SCENARIO[4:0], MISC[2:0]}
44	0	L4C[15:8]	L4C[7:0]	L4C[15:8]	L4C[7:0]
45	0	L4D[15:8]	L4D[7:0]	L4D[15:8]	L4D[7:0]
46	0	TTL[7:0]	TTL[7:0]	TTL[7:0]	TTL[7:0]
47	0	0	IPv6Flow[19:16]	IPv6Flow[15:8]	IPv6Flow[7:0]
48	0	{MAP_TYPE[3:0], MAP_SRC[3:0]}	{MAP_TYPE[3:0], MAP_SRC[3:0]}	{MAP_TYPE[3:0], MAP_SRC[3:0]}	{MAP_TYPE[3:0], MAP_SRC[3:0]}
49	0	{MAP_DMAC[3:0], MAP_SMAC[3:0]}	{MAP_DMAC[3:0], MAP_SMAC[3:0]}	{MAP_DMAC[3:0], MAP_SMAC[3:0]}	{MAP_DMAC[3:0], MAP_SMAC[3:0]}
50	0	{MAP_LENGTH[3:0], MAP_PROT[3:0]}	{MAP_LENGTH[3:0], MAP_PROT[3:0]}	{MAP_LENGTH[3:0], MAP_PROT[3:0]}	{MAP_LENGTH[3:0], MAP_PROT[3:0]}
51-62	0	0	0	0	0
63	ANY	SrcPort[25:24]	SrcPort[23:16]	SrcPort[15:8]	SrcPort[7:0]

15.18 Statistics Details

15.19 Statistics Control Registers Details

15.19.1 STATS_CFG[0..24]

The EnableGroup *n* bits enable or disable the counters in the corresponding per-port counter groups. These counters may not be fully provisioned for a 360 Mpps frame rate when all groups are enabled. The resulting counter update drops will be recorded in the STATS_DROP_COUNT_RX and STATS_DROP_COUNT_TX counters. By disabling counter groups, the total required counter bandwidth will be reduced, allowing the counters to sustain peak frame rate without drops.

Table 371: STATS_CFG[0..24]

Name	Bit	Description	Type	Default
EnableGroup1	1	Enable/Disable updates to Group 1 (RX frame classification)	RW	0x1
EnableGroup2	2	Enable/Disable updates to Group 2 (RX frame length binning)	RW	0x1
EnableGroup3	3	Enable/Disable updates to Group 3 (RX octets)	RW	0x1
EnableGroup4	4	Enable/Disable updates to Group 4 (RX per-priority packets)	RW	0x1
EnableGroup5	5	Enable/Disable updates to Group 5 (RX octets by priority)	RW	0x1
EnableGroup6	6	Enable/Disable updates to Group 6 (RX forwarding action)	RW	0x1
EnableGroup7	7	Enable/Disable updates to Group 7 (TX frame classification)	RW	0x1
EnableGroup8	8	Enable/Disable updates to Group 8 (TX frame length binning)	RW	0x1
EnableGroup9	9	Enable/Disable updates to Group 9 (TX octet counters)	RW	0x0
EnableGroup11	11	Enable/Disable updates to Group 11 (VLAN packet counters)	RW	0x1
EnableGroup12	12	Enable/Disable updates to Group 12 (VLAN octet counters)	RW	0x1
EnableGroup13	13	Enable/Disable updates to Group 13 (trigger counters)	RW	0x1
EnableGroup14	14	Enable/Disable updates to Group 14 (egress ACL counters)	RW	0x1
PrioritySelect	15	When set to 1, frames will be classified by their associated traffic class in the per-priority counters (Groups 4 and 5). When zero, classification will be by ingress user priority. Note: Users should set this bit to 1. Counting by priority is known to be unreliable.	RW	0x0

15.19.2 STATS_DROP_COUNT_RX[0..24]

Table 372: STATS_DROP_COUNT_RX[0..24]

Name	Bit	Description	Type	Default
DropCount	31: 0	Number of times counter updates to groups 1 through 6 (RX counters) were dropped due to insufficient counter bandwidth.	CW	0x0

15.19.3 STATS_DROP_COUNT_TX[0..24]

Table 373: STATS_DROP_COUNT_TX[0..24]

Name	Bit	Description	Type	Default
DropCount	31: 0	Number of times counter updates to groups 7 through 9 (TX counters) were dropped due to insufficient counter bandwidth.	CW	0x0

15.20 Layer 2 Lookup Registers (reserved 128KW) Details

15.20.1 MA_TABLE[0..16383]

Table 374: MA_TABLE[0..16383]

Name	Bit	Description	Type	Default
MACAddress	47: 0	MAC Address	RW	0x0
FID	59: 48	FID. Learning Group. In shared spanning tree mode, FID = 0. In multiple spanning tree mode, FID = VID.	RW	0x0
state	61: 60	State of this entry: • 0: invalid • 1: old • 2: young • 3: locked	RW	0x0
parityError	62	Parity Error. Software must write 0 in this field for the hardware to compute the right parity. Rading 0 means the parity was correct, reading 1 means the parity was incorrect. Writing 1 will cause the hardware to computer the wrong parity and could be used for testing.	RW	0x0
reserved	63	Reserved. Write as 0.	RV	0x0
destMask	88: 64	Destination Mask. A bit mask of the destination ports to which this address corresponds.	RW	0x0
destGlort	79: 64	Destination GLORT.	RW	0x0
destType	89	Defines if the etnry contains a "destMask" (0) or a "glort" (1).	RW	0x0
trigId	95: 90	TRIG-ID. Each trigger has a TRIG-ID and a defined in TRIGGER_CFG. If the trigger calls for a single MAC address match, then of the 2 MAC address lookups, there must be one match for that trigger. If the trigger calls for a source address and destination address match, then both lookups must resolve to the same TRIGID as the trigger lookups.	RW	0x0

Name	Bit	Description	Type	Default
		Note that two trigger ID's are pre-assigned and may not be changed: 0x0 and 0x3F. See TRIGGER_CONDITION_PARAM.		

(1) The table is searched by MAC address and FID. That is, the same MAC address may exist once per VLAN in the table in multiple spanning tree mode. On a VLAN boundary violation, an address is not learned. (2) On a parity error, the line is considered invalid. (3) On power-up, all bits are zero by default. (4) MAC entries take 3 32-bit words to completely specify. The entries are aligned to 128 bit boundaries in address space, that is, one entry every four addresses.

15.20.2 INGRESS_VID_TABLE[0..4095]

****** This register was named VID_TABLE ******

Table 375: INGRESS_VID_TABLE[0..4095]

Name	Bit	Description	Type	Default
parityError	0	Parity Error. Software must write 0 in this field for the hardware to compute the right parity. Rading 0 means the parity was correct, reading 1 means the parity was incorrect. Writing 1 will cause the hardware to computer the wrong parity and could be used for testing.	RW	0x0
reflect	1	If this bit is set and PORT_CFG_2 destination mask include itself, then layer 2 packets associated with this VLAN are allowed to go back on the port they come from (subject to LOOPBACK_SUPPRESS registers). If this bit is not set, then layer 2 packets associated with this VLAN are not allowed to go back on the port they come from. Routed packets are always allowed to go back where they come from and this bit is not used in this case.	RW	0x0
CounterIndex *(was vcnt)*	7: 2	Specifies the Group 11 counter index to update as frames on this VLAN are received by the switch.	RW	0x0
membership	63: 14	Defines port membership for this VLAN. There are 2 bits per port and they are defined as: • 00: this not a member of this VLAN • 01: undefined • 10: this port is a member of this VLAN • 11: undefined	RW	0x0
FID	75: 64	Learning group for this VLAN.	RW	0x0

Name	Bit	Description	Type	Default
TrapIGMP	77	Indicates if the IGMP frames on this VLAN shall be trapped or switched normally. The IGMP frames are trapped to the CPU glort.	RW	0x0

15.20.3 EGRESS_VID_TABLE[0..4095]

Table 376: EGRESS_VID_TABLE[0..4095]

Name	Bit	Description	Type	Default
parityError	0	Parity Error. Software must write 0 in this field for the hardware to compute the right parity. Rading 0 means the parity was correct, reading 1 means the parity was incorrect. Writing 1 will cause the hardware to computer the wrong parity and could be used for testing.	RW	0x0
MTU_Index	3: 1	Defines the MTU size for this VLAN through a 3-bit index into MTU_TABLE. The MTU is checked only for routed frames and the checking is done against the packet length defined in the IP header and not the actual frame length.	RW	0x0
membership	28: 4	Defines if this port is member of this VLAN or not.	RW	0x0
FID	43: 32	Learning group for this VLAN.	RW	0x0
TrigID	49: 44	VLAN trigger match condition identifier.	RW	0x0

15.20.4 INGRESS_FID_TABLE[0..4095]

****** This register was named FID_TABLE ******

Table 377: INGRESS_FID_TABLE[0..4095]

Name	Bit	Description	Type	Default
parityError *(was parityError)*	0	Parity Error. Software must write 0 in this field for the hardware to compute the right parity. Rading 0 means the parity was correct, reading 1 means the parity was incorrect. Writing 1 will cause the hardware to computer the wrong parity and could be used for testing.	RW	0x0
STPState	49: 2	Two bits of spanning tree state per port 1 through 24. The states are: • 0 : DISABLE • 1 : LISTENING • 2 : LEARNING • 3 : FORWARD	RW	0x0

The spanning tree state for port 0 is always assumed to be FORWARD

15.20.5 EGRESS_FID_TABLE[0..4095]

Table 378: EGRESS_FID_TABLE[0..4095]

Name	Bit	Description	Type	Default
parityError	0	Parity Error. Software must write 0 in this field for the hardware to compute the right parity. Rading 0 means the parity was correct, reading 1 means the parity was incorrect. Writing 1 will cause the hardware to computer the wrong parity and could be used for testing.	RW	0x0
Forwarding	24: 1	Defines forwarding state per port for port 1 through 24. The port 0 is always assumed to be in forwarding mode.	RW	0x0

15.20.6 MA_TCN_FIFO[0..511]

Table 379: MA_TCN_FIFO[0..511]

Name	Bit	Description	Type	Default
Entry	95: 0	Entry. Same structure as MA_TABLE.	RO	0x0
Index	107: 96	Index in the table.	RO	0x0
Set	110: 108	Set.	RO	0x0
Type	113: 111	Type of entry. • 0: Learned. • 1: Aged. • 2: Full. Entry couldn't be learned because the set is full • 3: Parity Error. The referenced MA_TABLE entry had an error • 4: Security violation (new). • 5: Security violation (moved). • 6: Purge completed.	RO	0x0
ParityError	114	Reads 1 when a parity error is detected in this notification entry of the MA_TCN_FIFO.	RO	0x0

15.20.7 MA_TCN_PTR

Table 380: MA_TCN_PTR

Name	Bit	Description	Type	Default
Head	8: 0	Entry that the software will read next. Updated by the software.	RW	0x0

Name	Bit	Description	Type	Default
Tail	17: 9	Entry that the hardware will write next. Updated by the hardware.	RO	0x0

15.20.8 MA_TCN_IP

Table 381: MA_TCN_IP

Name	Bit	Description	Type	Default
PendingEvents	0	Set whenever MA_TCN_PTR.Tail is advanced and (512 + Tail(new) - Head) % 512 > InterruptThreshold or when the idle period since MA_TCN_PTR.Tail was advanced is greater than the InterruptTimeout.	CW1	0x0
EventAgedOverflow	1	Set whenever a aging event notification could not be enqueued to the MA_TCN_FIFO.	CW1	0x0
EventLearnedOverflow	2	Set whenever a learning event notification event could not be enqueued to the MA_TCN_FIFO.	CW1	0x0
EventParityErrorOverflow	3	Set whenever a parity error notification could not be enqueued to the MA_TCN_FIFO.	CW1	0x0
BinFullOverflow	4	Set whenever a bin full notification could not be enqueued to the MA_TCN_FIFO.	CW1	0x0
SecurityViolationOverflow	5	Set whenever a security violation notification could not be enqueued to the MA_TCN_FIFO.	CW1	0x0

Writing '1' into any bit clears the corresponding bit, writing 0 has no effect.

15.20.9 MA_TCN_IM

The interrupt mask register controls if the corresponding interrupt source is presented to the interrupt hierarchy (bit is set to 0) or masked out (bit is set to 1).

Table 382: MA_TCN_IM

Name	Bit	Description	Type	Default
PendingEvents	0	Mask PendingEvents interrupts (1) or not (0).	RW	0x1
AgedOverflow	1	Mask aged event overflow interrupts (1) or not (0).	RW	0x1
LearnedOverflow	2	Mask learned event overflow interrupts (1) or not (0).	RW	0x1
ParityErrorOverflow	3	Mask parity error overflow interrupts (1) or not (0).	RW	0x1
BinFullOverflow	4	Mask bin full overflow interrupts (1) or not (0).	RW	0x1
SecurityViolationOverflow	5	Mask security violation overflow interrupts (1) or not (0).	RW	0x1

15.20.10 MA_TABLE_STATUS_3

Table 383: MA_TABLE_STATUS_3

Name	Bit	Description	Type	Default
skipSACount	15: 0	Count the number of times a source address lookup or learning event was not done because it is best effort and there wasn?t time. (Learning events are always besteffort, source address lookup is only best-effort if the mode bit is set).	CW	0x0
skipLRNCount	31: 16	Count of the number of times a learning event was skipped because it is best effort and there wasn?t time.	CW	0x0

Writing any value reset this register to 0.

15.20.11 MA_TABLE_CFG_1

Table 384: MA_TABLE_CFG_1

Name	Bit	Description	Type	Default
hashRotation	2: 1	The hash function produces a 16 bit value. The hash address is only 12 bits. Which 4 bits are excluded is programmable. • 0x0 - Bits 15:12 are not used. • 0x1 - Bits 11:8 are not used. • 0x2 - Bits 7:4 are not used. • 0x3 - Bits 3:0 are not used.	RW	0x0
tablePartition	3	Defines if the table is used as one table of 16K address or 2 tables of 8K address. In the later case, hardware maintains the two tables content in sync and perform SA and DA simultaneously using both table using the first table for SA and the second table for DA. This would guarantee full line rate performance in all cases when exact SA is required on all frames for security reasons.	RW	0x0
RateSALookupCapacity	9: 4	Defines the capacity of the token bucket for relax learning.	RW	0x4
RateSALookupCurrent	15: 10	Returns the current value of the token bucket	RO	0x0
RateSALookupIncrement	21: 16	Defines the number of tokens to add to the token bucket periodically.	RW	0x1
RateSALookupInterval	27: 22	Defines the time period to increment the token bucket in clock cycles.	RW	0x0F

15.20.12 MA_TABLE_CFG_2

Table 385: MA_TABLE_CFG_2

Name	Bit	Description	Type	Default
FloodGlort *(was GlortFlood)*	15: 0	Defines the glort to use when flooding.	RW	0x0
XcastGlort *(was GlortBroadcast)*	31: 16	Defines the glort used for broadcast and multicast. Note that multicast in this case is defined as MAC addresses that resolve to a destination mask instead of a glort, which is how multicast is implemented in FM2000, but is deprecated in FM4000	RW	0x0

15.20.13 MA_TABLE_CFG_3

Table 386: MA_TABLE_CFG_3

Name	Bit	Description	Type	Default
softAging	0	Defines if the aging (actual deletion of entries from the MA table) is done by hardware (0) or software (1). When this bit is set and an MA table entry becomes old, then the event is reported in the TCN FIFO but the entry is actually not deleted from the MA table by the hardware.	RW	0x0
softLearning	1	Defines if the learning (actual addition of entries from the MA table) is done by hardware (0) or software (1). When this bit is set and a new MAC address is marked for learning, then the event is reported in the TCN FIFO but the entry is not written in the MA table by the hardware.	RW	0x0
TrigIdDefault	7: 2	Defines the trigger ID to use when loading an entry in the table when there is no security violation.	RW	0x0

15.20.14 MA_TCN_CFG_1

Table 387: MA_TCN_CFG_1

Name	Bit	Description	Type	Default
AgedEventsThreshold	8: 0	Defines the MA_TCN_FIFO level at or above which aged entries are not enqueued. If FlowControlEnable is set, then MA_TABLE aging will also be suspended when this threshold is reached.	RW	0x1ff
LearnedEventsThreshold	17: 9	Defines the MA_TCN_FIFO level at or above which learning events are not enqueued. If FlowControlEnable is set, then learning into	RW	0x1ff

Name	Bit	Description	Type	Default
		the MA_TABLE will also be suspended when this threshold is reached.		
ErrorEventsLimit	26: 18	Defines the maximum number of error events to allow into the FIFO. Error events are defined to be BinFull, ParityError and SecurityViolation types.	RW	0x1ff
FlowControlEnable	27	Defines whether learning and aging is skipped if an entry cannot be enqueued into the FIFO.	RW	0x0

15.20.15 MA_TCN_CFG_2

Table 388: MA_TCN_CFG_2

Name	Bit	Description	Type	Default
InterruptThreshold	8: 0	Defines the number of entries in the FIFO before posting a PendingEvents interrupt.	RW	0x1ff
InterruptTimeout	18: 9	Specifies a maximum time to allow from enqueueing an event to posting the PendingEvents interrupt. If this timeout period is reached without exceeding InterruptThreshold, the PendingEvents interrupt will be set. The time base is the frame handler clock divided by 4096.	RW	0x3ff

15.20.16 MA_PURGE

The register is used to force an accelerated aging (purging) of the MAC address table on some entries (per VLAN, per glort, per VLAN/glort or for entire table). A TCN FIFO of type "purge completed" is posted when the purge completes.

Table 389: MA_PURGE

Name	Bit	Description	Type	Default
glort	15: 0	Defines the glort to purge.	RW	0x0
FID	27: 16	Defines the VLAN to purge.	RW	0x0
matchFID	28	If set to 1, then only the entries that match the FID defined in this register are purged. If set to 0, then the FID is ignored.	RW	0x0
matchGlort	29	If set to 1, then only the entries that match the GLORT defined in this register are purged. If set to 0, then the glort is ignored.	RW	0x0
startPurge	30	Start purge when this bit is set. This bit will self clear once the purge is completed.	RW	0x0

15.20.17 MTU_TABLE[0..7]

Defines the list of MTUs supported by the switch.

Table 390: MTU_TABLE[0..7]

Name	Bit	Description	Type	Default
mtu	13: 0	MTU size in bytes.	RW	0x0

Chapter 16 - Electrical Specifications

16.1 Absolute Maximum Ratings

Table 391: Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Units
Core Voltage	V_{DD}	-0.3	2	Volts
SerDes Supply Voltage	V_{DDX}	-0.3	2	Volts
SerDes Bias Voltage	V_{DDA}	-0.3	2	Volts
Transmitter Termination Voltage	V_{TT}	-0.3	2	Volts
LVTTL Power Supply	V_{DD33}	-0.3	3.9	Volts
PLL Analog power supply	V_{DDA33}	-0.3	3.9	Volts
Case Temp under bias		-	+130	°C
Storage Temp		-65	+150	°C
ESD		-2000	+2000	Volts

16.2 Recommended Operating Conditions

Table 392: Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Units
Core Voltage	V_{DD}	-	1.25	-	Volts
SerDes Supply Voltage	V_{DDX}	1.14	1.2	1.26	Volts [1,3]
SerDes Supply Voltage (when using SGMII)	V_{DDX}	0.95	1.0	1.05	Volts [2,3]
SerDes Bias Voltage	V_{DDA}	1.14	1.2	1.26	Volts
SerDes Bias Voltage (when using SGMII)	V_{DDA}	0.95	1.0	1.05	Volts
LVTTL Power Supply	V_{DD33}	3.14	3.3	3.47	Volts
PLL Analog power supply	V_{DDA33}	3.14	3.3	3.47	Volts
Transmitter Termination Voltage	V_{TT}	V_{DDX}	1.5	1.8	Volts
Operating Temp (Case)					
Commercial		0		+85	°C
Extended		0		+105	°C

Parameter	Symbol	Min	Typ	Max	Units
Industrial		-40		+115	°C

1. Connect a 1.2K resistor from RREF to VDDX for 1.2V operation
2. Connect a 1.0K resistor from RREF to VDDX for 1.0V operation
3. Sharing VDD and VDDX is not recommended.

Table 393: DC Characteristics of 2mA LVTTL Outputs

LED_CLK, LED_DATA[1..3], LED_EN, SPI_CLK, SPI_CS_N, SPI_MISO, SPI_MOSI

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}=1.2V, 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 V	-	10	-	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			-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 394:

Table 395: DC Characteristics of 4mA LVTTL Outputs

FH_PLL_CLKOUT, RXEOT_N, RXRDY_N, TDO, TXRDY_N

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}=1.2V, 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 V	-	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 396: DC Characteristics of 6mA LVTTL Outputs

AUTOBOOT, DATA[0..31], DERR_N, DTACK_INV, DTACK_N, EEPROM_EN1, EEPROM_EN2, I2C_ADDR[0..2], DATA_HOLD, GPIO_15, IGNORE_PARITY, INTR_N, MDC, PARITY_EVEN, PAR[0..3], RW_INV

Parameter	Symbol	Test Conditions	Min	Typical	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 V	-	-26	-	mA

Parameter	Symbol	Test Conditions	Min	Typical	Max	Units
Output LOW Current	I_{ODL}	V_{DD}=1.2 V, V_{DD33}=3.3 V, V_O = 1.5 V	-	-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

Table 397: DC Characteristics of LVTTL Inputs

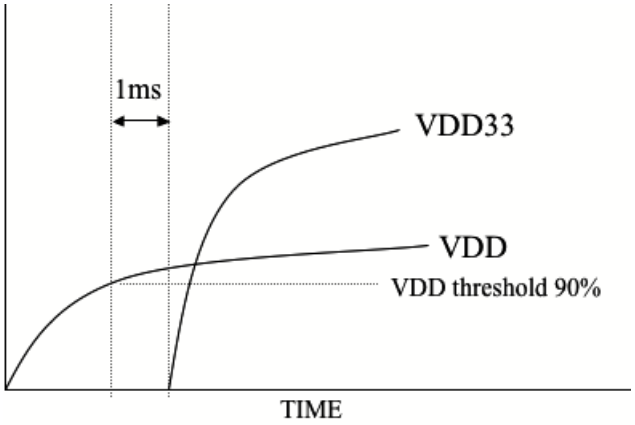
Table 398: Applies to all LVTTL inputs or input/outputs when in input mode.

Parameter	Symbol	Test Conditions	Min	Typical	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 Hysterisis	V_H	it0		5		mV
Input Hysterisis	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
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}=-18mA		-0.7	-1.2	V
Quiescent Power Supply Current	I_{DD33L}	V_{DD}=Max, V_{CC}=Max, V_{IN}=GND		0.1	10	µA

Parameter	Symbol	Test Conditions	Min	Typical	Max	Units
Quiescent Power Supply Current	I_{DD33H}	V_{DD}=Max, V_{CC}=Max, $V_{IN}=V_{DD}$		0.1	10	µA

16.2.1 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 1ms before VDD33 is powered on as shown in the figure below. In addition, it is recomened that VDD33 be ramped down prior to removing VDD. Also, RESET_N should be asserted at power down.



16.3 AC Timing Specifications

Table 399: 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]			.24	UI
	Random jitter component (RJ)			.12	
	Deterministic jitter component (DJ)			.12	
J_{TT}@3.125 Gb/S	Transmitter Total Jitter (Peak-Peak) [4]			.35	UI

Symbol	Parameter	Min	Typ	Max	Units
	Random jitter component (RJ)			.18	
	Deterministic jitter component (DJ)			.17	
Z_{OSE}	Single Ended Output Impedance	40	50	60	Ohms
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 400: XAUI and REFCLK Input 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 Gbps	Total jitter tolerance [1]			.71	UI
	Random jitter component (RJ)			.26	
	Deterministic jitter component (DJ)			.45	
J_{TT}@ 3.25 Gbps	Total jitter tolerance [1]			.65	UI
	Random jitter component (RJ)			.24	
	Deterministic jitter component (DJ)			.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]	-.5		1.6	V

1. CJPAT
2. Frequency range of 100MHz to 1.875GHz
3. Without damage to any signal pin

Table 401: 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

16.3.1 CPU Interface, General Timing Requirements

Figure 48: CPU Signal Timing

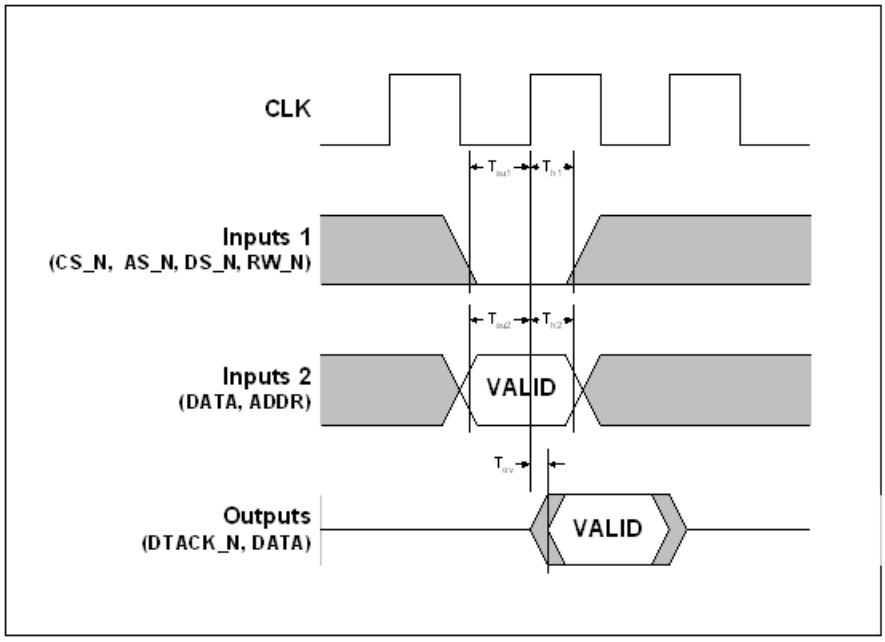


Table 402: CPU Interface Timing Constraints

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	Tsu1	2.0	-	-	ns	-

Parameter	Symbol	Min	Typ	Max	Units	Test Conditions
Hold time for CS_N, AS_N, DS_N and RW_N, to rising edge of clock	Th1	0.5	-	-	ns	-
Setup time for ADDR and DATA(in) to rising edge of clock	Tsu2	2.0	-	-	ns	-
Hold time for ADDR and DATA(in) to rising edge of clock	Th2	0.5	-	-	ns	-
Output valid for DTACK_N and DATA(out) to rising edge of clock	Tov	0	-	3.5	ns	

16.3.2 MDIO Interface, General Timing Requirements

The MDIO interface timing is shown in the figure below. 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.

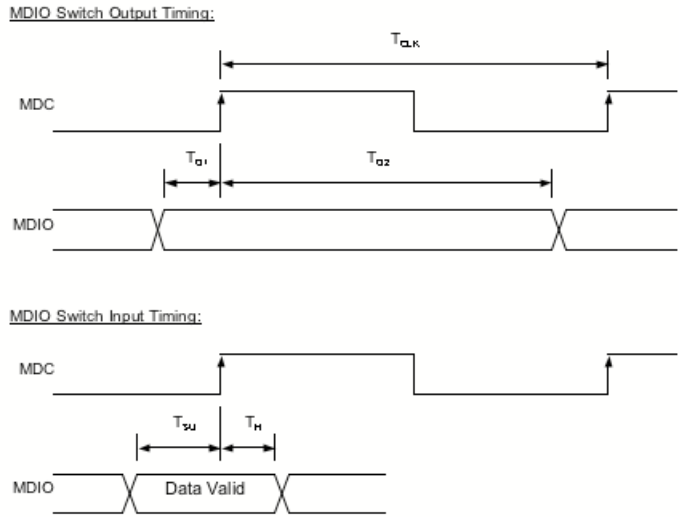


Table 403: MDIO Interface Timing Constraints

Parameter	Symbol	Min	Typ	Max	Units	Comments
MDC output clock period	TCLK	-	-	-		Programmable period using the MDIO_CFG DIVIDER register. Period = (CPU_CLK/2)/DIVIDER
MDIO switch output timing before MDC clock edge	TO1	-	-	-		8 times the CPU clock

Parameter	Symbol	Min	Typ	Max	Units	Comments
MDIO switch output timing after MDC clock edge	TO2	-	-	-		TCLK - TO1
MDIO input set-up time relative to MDC	TSU	10.0	-	-	ns	-
MDIO input hold time relative to MDC	TH	0.0	-	-	ns	-

16.3.3 JTAG Interface

The JTAG interface follows standard timing as defined in the IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture, 2001.

Note 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.

Chapter 17 - Mechanical Specification

This section provides a set of mechanical drawings for the FM3000 and FM4000 series of products. There are three package sizes used for these products as listed below.

Package	FM3000 Products	FM4000 Products
1433-Ball, 40mm	FM3224, FM3212	FM4224, FM4212
897-Ball, 32mm	FM3208, FM3112, FM3104, FM3103	FM4208, FM4112, FM4104, FM4103
529-Ball, 25mm	FM3410	FM4410

17.1 1433-Ball Package Dimensions

Figure 49: 1433-Ball Package Bottom View

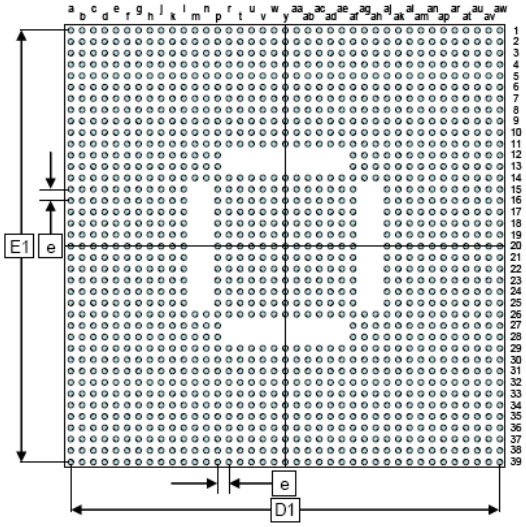


Figure 50: 1433-Ball Package Top View

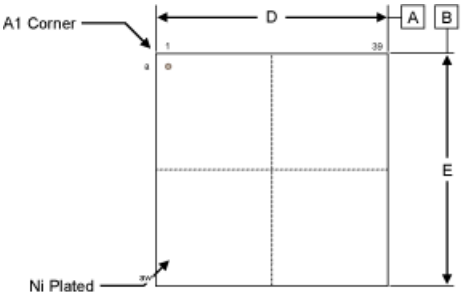


Figure 51: 1433-Ball Package Side View

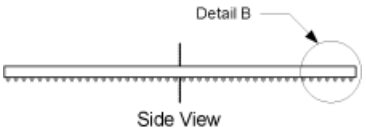


Figure 52: Detail B of 1433-Ball Package Side View

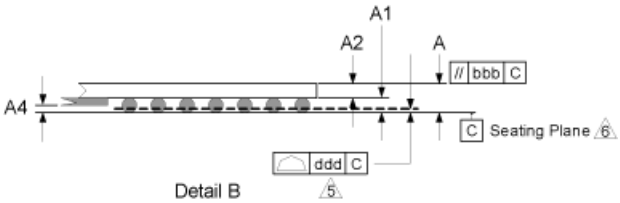


Table 404: 1433-Ball Package Demensions

Dimensional References			
Reference	Min	Nom	Max
A	2.67	3.07	3.47
A1	0.39	0.49	0.59
A2	2.18	2.58	2.98
D	39.80	40.00	40.20
D1	38.0 BSC		
E	39.80	40.00	40.20
E1	38.0 BSC		
e	1.00 BSC		
M	39		
N	1433		
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.
- Dimensioning and tolerancing per ASME Y14.5M 1994

17.2 897-Ball Package Dimensions

Figure 53: 897-Ball Package Bottom View

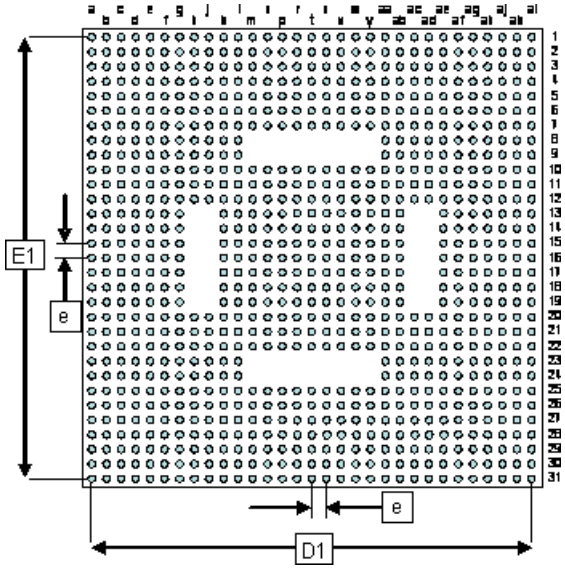


Figure 54: 897-Ball Package Top View

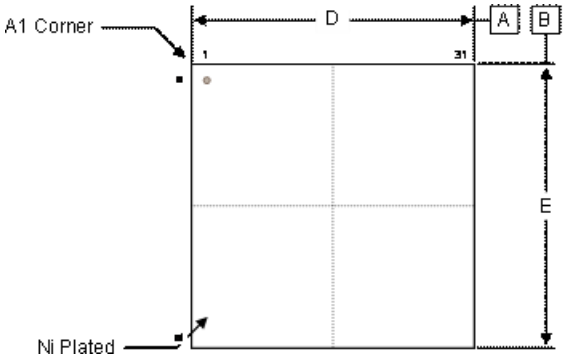


Figure 55: 897-Ball Package Side View

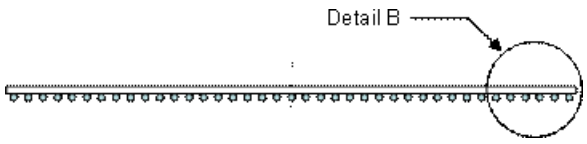


Figure 56: Detail B of 897-Ball Package Side View

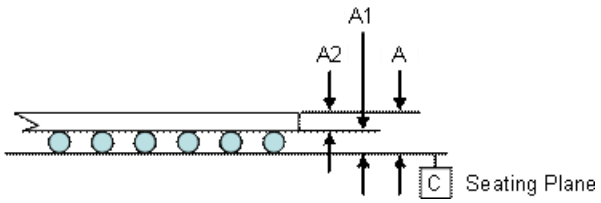


Table 405: 897-Ball Package Dimensions

Dimensional References			
Reference	Min	Nom	Max
A	2.67	3.07	3.47
A1	0.39	0.49	0.59
A2	2.18	2.58	2.98
D	31.80	32.00	32.20
D1	30.0 BSC		
E	31.80	32.00	32.20
E1	30.0 BSC		
e	1.00 BSC		
M	31		
N	897		
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.
- Dimensioning and tolerancing per ASME Y14.5M 1994

17.3 529-Ball Package Dimensions

Figure 57: 529-Ball Package Bottom View

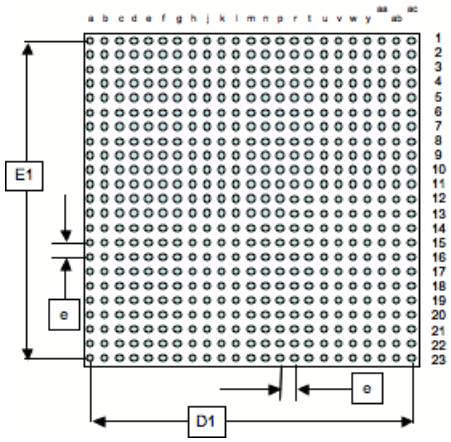


Figure 58: 529-Ball Package Top View

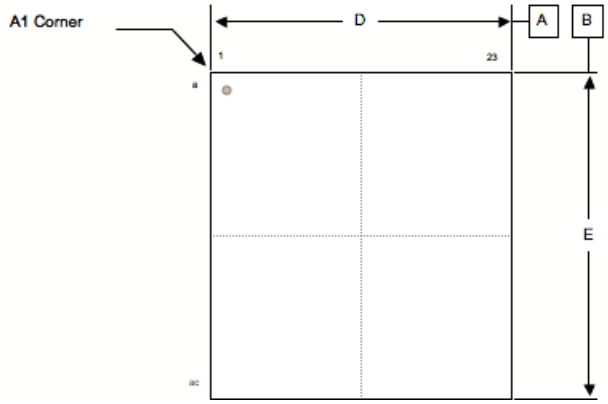


Figure 59: 529-Ball Package Side View

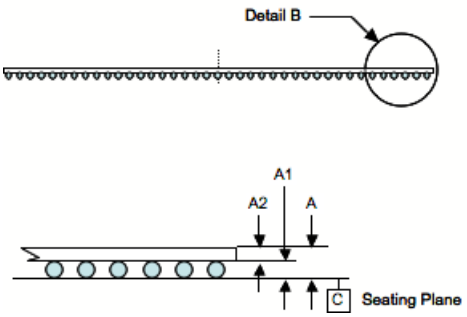


Table 406: 529-Ball Package Dimensions

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.)
- Dimensioning and tolerancing per ASME Y14.5M 1994

17.4 Heat Sinking

It is anticipated that a heat sink will be required for most FM2224 applications. Fulcrum has successfully used two copper heat sinks designed by Fulcrum 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 64mm x 64mm and is taller at just over 25mm in height. The FUL004/5 design occupies a larger footprint of 80mm x 100mm but is only 8.5mm in height. The thermal characteristics (thetaCA) 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 will also ensure that the junction stays below its maximum allowable temperature of 125 ºC. With a junction-to-case thermal resistance (thetaJC) of only 0.1 ºC/W, even the highest allowed value of case temperature, 115ºC, will keep the junction temperature below 125ºC. This is true even assuming a worst case power dissipation approaching 48W, which results in a 4.8ºC rise in junction temperature over case temperature (48W x 0.1ºC/W = 4.8ºC).

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 188.

Table 407: Thermal Parameters

Parameter	Description		Value
Theta-JC	Thermal resistance, junction to case		0.1 ºC/Watt
Psi-JB (1,2)	Thermal resistance, junction to board		3.5 ºC/Watt
Theta-CA	Thermal resistance, case to ambient with heat sink in place.	100 LFM	1.2 ºC/Watt (typ)
		150 LFM	0.9 ºC/Watt (typ)
		200 LFM	0.75 ºC/Watt (typ)
TCASE(max)	Maximum case temperature	Commercial, -C	85 ºC
		Extended, -E	105 ºC
		Industrial, -I	115 ºC

Notes:

(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, 105ºC 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 the Table of Paramters, this equates to a temperature rise of approximately 35ºC. With a maximum case temperature of 105ºC , a maximum ambient temperature of 70ºC 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.

17.4.1 Heat Sink Mounting Pressure

The maximum allowable pressure on the package surface when mounting a heat sink is 15 psi.

17.5 Pin Locations

Table 408: PIN Locations (Alphabethical Order)

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
ADDR[23..2]	ADDR[2]: T32	ADDR[2]: K29	
	ADDR[3]: T33	ADDR[3]: K25	
	ADDR[4]: T34	ADDR[4]: L25	
	ADDR[5]: T35	ADDR[5]: L26	
	ADDR[6]: T36	ADDR[6]: L27	
	ADDR[7]: T37	ADDR[7]: L28	
	ADDR[8]: T38	ADDR[8]: L29	
	ADDR[9]: T39	ADDR[9]: L30	
	ADDR[10]: U31	ADDR[10]: L31	
	ADDR[11]: U32	ADDR[11]: M25	
	ADDR[12]: U33	ADDR[12]: M26	
	ADDR[13]: U34	ADDR[13]: M27	
	ADDR[14]: U35	ADDR[14]: M28	
	ADDR[15]: U36	ADDR[15]: M29	
	ADDR[16]: U37	ADDR[16]: M30	
	ADDR[17]: U38	ADDR[17]: M31	
	ADDR[18]: U39	ADDR[18]: N26	
	ADDR[19]: V33	ADDR[19]: N27	
	ADDR[20]: V34	ADDR[20]: N28	
	ADDR[21]: V35	ADDR[21]: N29	
	ADDR[22]: V36	ADDR[22]: N30	
	ADDR[23]: V37	ADDR[23]: N31	
AS_N	W35	P30	
CHIP_RESET_N	N33	J27	M19
CPU_CLK	Y37	R29	T23
CONT_EN	N31	G23	

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
CS_N	W34	P29	
DATA[31..0]	DATA[0]: AB33	DATA[0]: V31	
	DATA[1]: AB34	DATA[1]: V25	
	DATA[2]: AB35	DATA[2]: V26	
	DATA[3]: AB36	DATA[3]: V27	
	DATA[4]: AB37	DATA[4]: V28	
	DATA[5]: AC31	DATA[5]: V29	
	DATA[6]: AC32	DATA[6]: V30	
	DATA[7]: AC33	DATA[7]: W25	
	DATA[8]: AC34	DATA[8]: W26	
	DATA[9]: AC35	DATA[9]: W27	
	DATA[10]: AC36	DATA[10]: W28	
	DATA[11]: AC37	DATA[11]: W29	
	DATA[12]: AC38	DATA[12]: W30	
	DATA[13]: AC39	DATA[13]: W31	
	DATA[14]: AD31	DATA[14]: Y25	
	DATA[15]: AD32	DATA[15]: Y26	
	DATA[16]: AD33	DATA[16]: Y27	
	DATA[17]: AD34	DATA[17]: Y28	
	DATA[18]: AD35	DATA[18]: Y29	
	DATA[19]: AD36	DATA[19]: Y30	
	DATA[20]: AD37	DATA[20]: Y31	
	DATA[21]: AD38	DATA[21]: AA26	
	DATA[22]: AD39	DATA[22]: AA27	
	DATA[23]: AE31	DATA[23]: AA28	
	DATA[24]: AE32	DATA[24]: AA29	
	DATA[25]: AE33	DATA[25]: AA30	
	DATA[26]: AE34	DATA[26]: AA31	

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	DATA[27]: AE35 DATA[28]: AE36 DATA[29]: AE37 DATA[30]: AE38 DATA[31]: AE39	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	
TEST_RESET_N	Y30	T26	
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
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_DATA0	R29	H24	K18
LED_DATA1	T29	J24	N18
LED_DATA2	U29	K24	P18

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
LED_EN	V29	K23	J17
LED_CLK	P29	G24	K17
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
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: W20 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_RCN: J35 P02_RCP: J36 P02_RDN: G35	P02_RAN: G27 P02_RAP: G28 P02_RBN: E27 P02_RBP: E28 P02_RCN: C27 P02_RCP: C28 P02_RDN: A27	P02_RAN: E22 P02_RAP: D22 P02_RBN: E21 P02_RBP: D21 P02_RCN: E20 P02_RCP: D20 P02_RDN: E19

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P02_RDP: G36	P02_RDP: A28	P02_RDP: D19
P02_T{A..D}{P,N}	P02_TAN: N38	P02_TAN: G30	P02_TAN: B22
	P02_TAP: N39	P02_TAP: G31	P02_TAP: A22
	P02_TBN: L38	P02_TBN: E30	P02_TBN: B21
	P02_TBP: L39	P02_TBP: E31	P02_TBP: A21
	P02_TCN: J38	P02_TCN: C30	P02_TCN: B20
	P02_TCP: J39	P02_TCP: C31	P02_TCP: A20
	P02_TDN: G38	P02_TDN: A30	P02_TDN: B19
	P02_TDP: G39	P02_TDP: A31	P02_TDP: A19
P03_R{A..D}{P,N}	P03_RAN: AR33	P03_RAN: AG19	P03_RAN: W17
	P03_RAP: AT33	P03_RAP: AH19	P03_RAP: Y17
	P03_RBN: AR35	P03_RBN: AG21	
	P03_RBP: AT35	P03_RBP: AH21	
	P03_RCN: AR37	P03_RCN: AG23	
	P03_RCP: AT37	P03_RCP: AH23	
	P03_RDN: AR39	P03_RDN: AG25	
	P03_RDP: AT39	P03_RDP: AH25	
P03_T{A..D}{P,N}	P03_TAN: AV33	P03_TAN: AK19	P03_TAN: AB17
	P03_TAP: AW33	P03_TAP: AL19	P03_TAP: AC17
	P03_TBN: AV35	P03_TBN: AK21	
	P03_TBP: AW35	P03_TBP: AL21	
	P03_TCN: AV37	P03_TCN: AK23	
	P03_TCP: AW37	P03_TCP: AL23	
	P03_TDN: AV39	P03_TDN: AK25	
	P03_TDP: AW39	P03_TDP: AL25	
P04_R{A..D}{P,N}	P04_RAN: E39	P04_RAN: E25	P04_RAN: E17
	P04_RAP: D39	P04_RAP: D25	P04_RAP: D17
	P04_RBN: E37	P04_RBN: E23	
	P04_RBP: D37	P04_RBP: D23	

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P04_RCN: E35 P04_RCP: D35 P04_RDN: E33 P04_RDP: D33	P04_RCN: E21 P04_RCP: D21 P04_RDN: E19 P04_RDP: D19	
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
P06_R{A..D}{P,N}	P06_RAN: L31	P06_RAN: E17	P06_RAN: E14

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P06_RAP: K31	P06_RAP: D17	P06_RAP: D14
	P06_RBN: L29	P06_RBN: E15	P06_RBN: E13
	P06_RBP: K29	P06_RBP: D15	P06_RBP: D13
	P06_RCN: L27	P06_RCN: E13	P06_RCN: E12
	P06_RCP: K27	P06_RCP: D13	P06_RCP: D12
	P06_RDN: L25	P06_RDN: E11	P06_RDN: E11
	P06_RDP: K25	P06_RDP: D11	P06_RDP: D11
P06_T{A..D}{P,N}	P06_TAN: H31	P06_TAN: B17	P06_TAN: B14
	P06_TAP: G31	P06_TAP: A17	P06_TAP: A14
	P06_TBN: H29	P06_TBN: B15	P06_TBN: B13
	P06_TBP: G29	P06_TBP: A15	P06_TBP: A13
	P06_TCN: H27	P06_TCN: B13	P06_TCN: B12
	P06_TCP: G27	P06_TCP: A13	P06_TCP: A12
	P06_TDN: H25	P06_TDN: B11	P06_TDN: B11
	P06_TDP: G25	P06_TDP: A11	P06_TDP: A11
P07_R{A..D}{P,N}	P07_RAN: AR25	P07_RAN: AG3	P07_RAN: W6
	P07_RAP: AT25	P07_RAP: AH3	P07_RAP: Y6
	P07_RBN: AR27	P07_RBN: AG5	P07_RBN: W7
	P07_RBP: AT27	P07_RBP: AH5	P07_RBP: Y7
	P07_RCN: AR29	P07_RCN: AG7	P07_RCN: W8
	P07_RCP: AT29	P07_RCP: AH7	P07_RCP: Y8
	P07_RDN: AR31	P07_RDN: AG9	P07_RDN: W9
	P07_RDP: AT31	P07_RDP: AH9	P07_RDP: Y9
P07_T{A..D}{P,N}	P07_TAN: AV25	P07_TAN: AK3	P07_TAN: AB6
	P07_TAP: AW25	P07_TAP: AL3	P07_TAP: AC6
	P07_TBN: AV27	P07_TBN: AK5	P07_TBN: AB7
	P07_TBP: AW27	P07_TBP: AL5	P07_TBP: AC7
	P07_TCN: AV29	P07_TCN: AK7	P07_TCN: AB8

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P07_TCP: AW29	P07_TCP: AL7	P07_TCP: AC8
	P07_TDN: AV31	P07_TDN: AK9	P07_TDN: AB9
	P07_TDP: AW31	P07_TDP: AL9	P07_TDP: AC9
P08_R{A..D}{P,N}	P08_RAN: E31	P08_RAN: E9	P08_RAN: E9
	P08_RAP: D31	P08_RAP: D9	P08_RAP: D9
	P08_RBN: E29	P08_RBN: E7	P08_RBN: E8
	P08_RBP: D29	P08_RBP: D7	P08_RBP: D8
	P08_RCN: E27	P08_RCN: E5	P08_RCN: E7
	P08_RCP: D27	P08_RCP: D5	P08_RCP: D7
	P08_RDN: E25	P08_RDN: E3	P08_RDN: E6
	P08_RDP: D25	P08_RDP: D3	P08_RDP: D6
P08_T{A..D}{P,N}	P08_TAN: B31	P08_TAN: B9	P08_TAN: B9
	P08_TAP: A31	P08_TAP: A9	P08_TAP: A9
	P08_TBN: B29	P08_TBN: B7	P08_TBN: B8
	P08_TBP: A29	P08_TBP: A7	P08_TBP: A8
	P08_TCN: B27	P08_TCN: B5	P08_TCN: B7
	P08_TCP: A27	P08_TCP: A5	P08_TCP: A7
	P08_TDN: B25	P08_TDN: B3	P08_TDN: B6
	P08_TDP: A25	P08_TDP: A3	P08_TDP: A6
P09_R{A..D}{P,N}	P09_RAN: AJ17	P09_RAN: AB9	P09_RAN: W1
	P09_RAP: AK17	P09_RAP: AA9	P09_RAP: Y1
	P09_RBN: AJ19		P09_RBN: W2
	P09_RBP: AK19		P09_RBP: Y2
	P09_RCN: AJ21		P09_RCN: W3
	P09_RCP: AK21		P09_RCP: Y3
	P09_RDN: AJ23		P09_RDN: W4
	P09_RDP: AK23		P09_RDP: Y4
P09_T{A..D}{P,N}	P09_TAN: AM17	P09_TAN: AE9	P09_TAN: AB1
	P09_TAP: AN17	P09_TAP: AD9	P09_TAP: AC1

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P09_TBN: AM19 P09_TBP: AN19 P09_TCN: AM21 P09_TCP: AN21 P09_TDN: AM23 P09_TDP: AN23		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
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_RAN: AB7 P11_RAP: AA7	

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P11_RDN: AR23 P11_RDP: AT23		
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_RAN: AG1 P13_RAP: AH1	P13_RAN: U5 P13_RAP: U4

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P13_RBP: AT11 P13_RCN: AR13 P13_RCP: AT13 P13_RDN: AR15 P13_RDP: AT15		
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_TAN: B1 P14_TAP: A1	P14_TAN: G2 P14_TAP: G1

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P14_TDP: A9		
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	
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_TAN: J2 P16_TAP: J1	

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P16_TCN: H11 P16_TCP: G11 P16_TDN: H9 P16_TDP: G9		
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	
P18_T{A..D}{P,N}	P18_TAN: N8	P18_TAN: N2	

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P18_TAP: N7 P18_TBN: R8 P18_TBP: R7 P18_TCN: U8 P18_TCP: U7 P18_TDN: W8 P18_TDP: W7	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: E3	P20_RAN: G5 P20_RAP: G4	P20_RAN: H5 P20_RAP: H4

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P20_RCP: D3 P20_RDN: E1 P20_RDP: D1		
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
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_RAN: L5 P22_RAP: L4	P22_RAN: K5 P22_RAP: K4

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P22_RBN: L5 P22_RBP: K5 P22_RCN: L3 P22_RCP: K3 P22_RDN: L1 P22_RDP: K1		
P22_T{A..D}{P,N}	P22_TAN: H7 P22_TAP: G7 P22_TBN: H5 P22_TBP: G5 P22_TCN: H3 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
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_TAN: U2 P23_TAP: U1	P23_TAN: N2 P23_TAP: N1

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	P23_TDN: AG2 P23_TDP: AG1		
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	RCK1AN: AD21 RCK1AP: AC21 RCK1BN: AD20 RCK1BP: AC20 RCK2AN: J21 RCK2AP: H21 RCK2BN: J20	RCK1AN: U16 RCK1AP: T16 RCK1BN: U15 RCK1BP: T15 RCK2AN: H16 RCK2AP: G16 RCK2BN: H15

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	RCK2BP: N26	RCK2BP: H20	RCK2BP: G15
	RCK3AN: AG13	RCK3AN: AD11	RCK3AN: U7
	RCK3AP: AF13	RCK3AP: AC11	RCK3AP: T7
	RCK3BN: AG14	RCK3BN: AD12	RCK3BN: U8
	RCK3BP: AF14	RCK3BP: AC12	RCK3BP: T8
	RCK4AN: P13	RCK4AN: J11	RCK4AN: H7
	RCK4AP: N13	RCK4AP: H11	RCK4AP: G7
	RCK4BN: P14	RCK4BN: J12	RCK4BN: H8
	RCK4BP: N14	RCK4BP: H12	RCK4BP: G8
RESERVED. Leave open.			P20, P21, P22, R21, R22, R23
RREF[24..1]	RREF01: AK36	RREF01: AH28	RREF01: Y23
	RREF02: K36	RREF02: D28	RREF02: D23
	RREF03: AT36	RREF03: AH22	RREF03: Y18
	RREF04: D36	RREF04: D22	RREF04: D18
	RREF05: AK28	RREF05: AH14	RREF05: Y15
	RREF06: K28	RREF06: D14	RREF06: D15
	RREF07: AT28	RREF07: AH6	RREF07: Y10
	RREF08: D28	RREF08: D6	RREF08: D10
	RREF09: AK20	RREF09: AD8	RREF09: Y5
	RREF10: K20	RREF10: H8	RREF10: D5
	RREF11: AT20	RREF11: AD6	
	RREF12: D20	RREF12: H6	
	RREF13: AT12	RREF13: AH2	RREF13: V4
	RREF14: D12	RREF14: D2	RREF14: F4
	RREF15: AK12	RREF15: V6	
	RREF16: K12	RREF16: P7	
	RREF17: AD10	RREF17: U6	
	RREF18: T10	RREF18: R7	

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	RREF19: AT4 RREF20: D4 RREF21: AK4 RREF22: K4 RREF23: AD4 RREF24: T4	RREF19: V7 RREF20: P6 RREF21: U7 RREF22: R6 RREF23: T7 RREF24: T6	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
TESTMODE	T31	L23	R17
TMS	AE30	AD25	U17
TRST_N	AD30	AC25	T20
TXRDY_N	Y34	P26	
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,	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

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	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		
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	AA24	T17
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,	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,	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,

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	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, 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,	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, 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	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

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	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		
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 VTT08: B26, B30, D26, D30 VTT09: AK18, AK22, AM18, AM22 VTT10: H18, H22, K18, K22	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 VTT08: B4, B8, D4, D8 VTT09: AE8 VTT10: G8 VTT11: AE6	VTT01: AB23 VTT02: B23 VTT05: AB15 VTT06: B15 VTT07: AB10 VTT08: B10 VTT09: AB5 VTT10: B5

Signal	FM4224/FM3224 Pin Location	FM4112/FM3112 Pin Location	FM4410/FM3410 Pin Location
	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	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	
VTT (no port association)			B16, B18, F2, J2, M2, R2, V2, AB16, AB18