NXP Semiconductors
Product Brief

Document Number: 2.0
Rev. 2.0 02/2023

S32G PFE Product Brief

Contents

1. Software Product Overview

The Packet Forwarding Engine (PFE) software (SW) consists of following SW components for the S32G Hardware (HW) Platform:

- PFE Firmware;
- Ethernet Driver;
- Configuration API - FCI (Fast Control Interface);
- Management daemon - CMM (Conntrack Monitor Module).

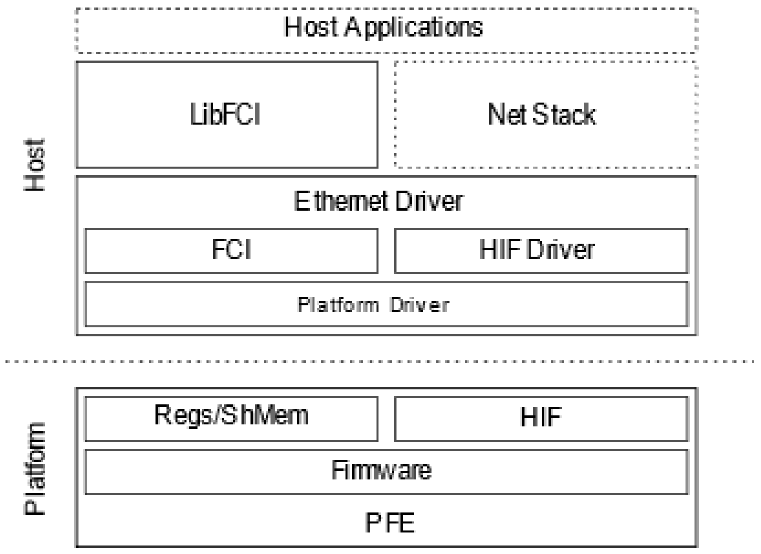

Figure 1. **SW Components**

All of the components or only particular ones are subject of integration with target environment represented by target HW platform, operating system (OS), OS-provided networking components (Networking Stack, firewall), and related user applications.

Firmware

Firmware is a software component running within the PFE. Each packet reaching the PFE (ingress/egress) is processed by the firmware, which is responsible for packet classification, routing, modifications and supporting functions. Firmware is under control of NXP and can be modified/extended as required by target use cases to add specific features related to Ethernet traffic processing. The firmware is delivered in a binary form.

Ethernet Driver

Ethernet driver is, in general, OS-specific component and consists of three main functional blocks:

- Platform driver

  This covers initial PFE HW bring-up, configuration, firmware upload, and management of PFE HW components. In general, this part can be described as low-level PFE driver providing interface to the hardware (including firmware).

- Data path management

  Tasks related to Ethernet data traffic management in terms of passing packets between PFE and the networking stack. This includes implementation of Host Interface (HIF) driver and connection with networking stack.

- Control path management

Functionality related to PFE engine configuration, FCI.

The driver runs within the target host OS environment and connects the PFE with networking stack. It provides access to physical Ethernet interfaces via exposing endpoints to the OS and implements control path domain in form of configuration and monitoring the PFE hardware and firmware resources. The driver can be used in its full form or in a form of just a simple HIF driver in case when host CPU does not need to run full driver but still requires access to the data path domain (multi-core environment).

- Master-Slave Runtime

The driver can run in multiple instances within the same system to allow sharing the PFE provided connectivity using dedicated host interfaces for dedicated CPU cores.

Configuration API

To let host applications, configure the PFE features, the PFE SW provides configuration API called, “Fast Control Interface (FCI).” It is io-ctl-like interface implemented in form of user space library and a kernel space part, running within the driver. Functions like fci_open(), fci_write(), fci_cmd() or fci_query() provided by the LibFCI are available to pass commands with arguments and receive responses. Configuration API implementation is closely coupled to platform driver and thus is OS dependent.

CMM

CMM is a host user space demo application developed to simplify configuration of some PFE features, like L3 Router. It runs within the host OS and monitors the networking stack. Once it detects an Ethernet flow which could be potentially fast-forwarded (routed by PFE instead of host CPU) it uses LibFCI to configure the PFE to do so. User then does not need to program own management application and configure the PFE but just configures the routing subsystem using native OS-provided interfaces. The CMM could be potentially extended to automatically configure IPsec offload or L2 bridging. For CMM availability please contact NXP representative.

2. Software Content

This chapter covers basic functional description of features supported by the PFE block and related SW stack (firmware, drivers, APIs).

Datapath Endpoint

This is standard data flow functionality provided by the Firmware and Ethernet driver via endpoints (PFEn) exposed to the OS and corresponding to particular physical interfaces (MACn) of the PFE. It covers:

- Initial PFE platform configuration (clocks, pins, memory, firmware, ...);
- MAC address filtering per physical interface;
- Optional CRC generation and verification offload (IPv4/IPv6 TCP/UDP/ICMP, FCS);
- Reception of packets via physical interfaces and providing them to host networking stack via endpoints;
- Transmission of packets originating from host networking stack via physical interfaces.

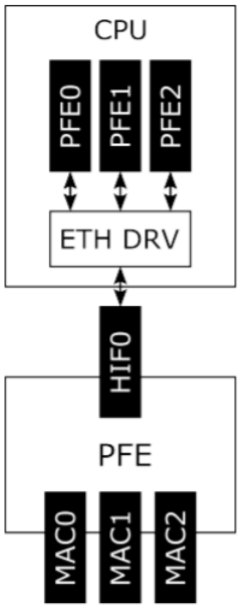

Figure 2. **Driver and Interfaces**

IPv4/IPv6 Router

The IPv4/IPv6 Router is a dedicated feature to offload the host CPU from tasks related to forwarding of specific IP traffic between two physical interfaces. Normally, the ingress IP traffic is passed to the host CPU running TCP/IP stack which is responsible for routing of the packets. Once the stack identifies that a packet does not belong to any of local IP endpoints it performs a lookup in the routing table to determine how to process such traffic. If the routing table contains an entry associated with the packet (5-touple search), the stack modifies and forwards the packet to another interface to reach its intended destination node.

The PFE can be configured to identify flows which do not need to enter the host CPU using its internal routing table, and to ensure that the right packets are forwarded to the right destination interface. The following features are implemented to support the forwarding functionality:

- Routing table lookup

  The PFE performs routing table lookup based on information parsed from the ingress packet header fields (5-touple: SRC/DST IP, SRC/DST port, protocol). The output of the lookup operation is an entry containing instructions programmed by the user specifying actions to be performed on the matching packet (destination interface, NAPT configuration, ...).

- NAPT

  If specified, the PFE performs packet modification according to NAPT setup. It updates address(es) and/or port(s) as requested by configuration given within routing table entry.

- TCP end of connection monitoring

  The PFE implements monitoring of fast-forwarded flows. If SYN, FIN, or RST packet is detected the PFE sends notification to the host application where it can be appropriately handled (remove the flow from local table, update firewall, ...).

- Packet forwarding to destination interface

Once packet is routed and modified it is forwarded to destination interface.

L2 Bridge (Switch)

The L2 Bridge functionality covers forwarding of packets based on MAC addresses. It provides the possibility to move bridging-related tasks from host CPU to the PFE and thus offloads the host-based networking stack. The L2 Bridge feature represents a network switch device implementing the following functionality:

- MAC table and address learning

  The L2 bridging functionality is based on determining to which interface and ingress packet shall be forwarded. For this purpose a network switch device implements a bridging table (MAC table aka FDB - Forwarding Data Base) which is searched to get target interface for each packet entering the switch. If received, interface with the packet has been received on, is added - learned. Destination MAC address of an ingress packet is then used to search the table to determine the target interface.

- Aging

  Each MAC table entry get default timeout value once learned. In time this timeout is being decreased until zero is reached. Entries with zero timeout value are automatically removed from the table. The timeout value is re-set each time the corresponding table entry is used to process a packet.

- Port migration

  When a MAC address is seen on one interface of the switch and the entry has been created, it is automatically updated when the MAC address is seen on another interface.

- VLAN awareness

  The bridge implements VLAN table. This table is used to implement VLAN-based policies like Ingress and Egress membership. Feature includes configurable VLAN tagging and un-tagging functionality per bridge interface.

The bridge utilizes PFE HW accelerators to perform MAC and VLAN table lookup thus this operation is highly optimized. Host CPU SW is only responsible for correct bridge configuration using the dedicated API.

Ingress QoS

Under heavy traffic conditions a network device can experience so called “congestion state.” This can be caused, for instance, by internal performance limitations (i.e., limited processing bandwidth like routing of big amount of small packets or too complex/time-consuming operations) and results in packet drop within the input stages of the device. Therefore, the input interface is where Ingress QoS takes place. Its main purpose is congestion avoidance and congestion control. To support this the Ingress QoS implements following mechanisms:

- Classification of ingress traffic to: Managed, unManaged and Reserved,
- Weighted Random Early Drop (WRED), and
- Ingress port-based rate shaper

  These mechanisms allow user to prioritize ingress traffic and ensure that high-priority traffic will be received preferentially at the expense of low-priority flows.

Classification

The classification is implemented via table where the user can program data flow parameters (L2/L3/L4) and specify which type of traffic (Managed, unManaged, Reserved) the flow belongs to. Optionally, certain flows can be classified as "To be dropped" to identify and terminate malicious flows just within the ingress block, without reaching next processing stages.

WRED

After the classification stage, the Ingress QoS implements the WRED algorithm to drop low-priority traffic when PFE is reaching the congestion state. There are multiple configurable zones with different thresholds and drop probabilities. The higher zone is identified, the higher drop probability is used by the WRED.

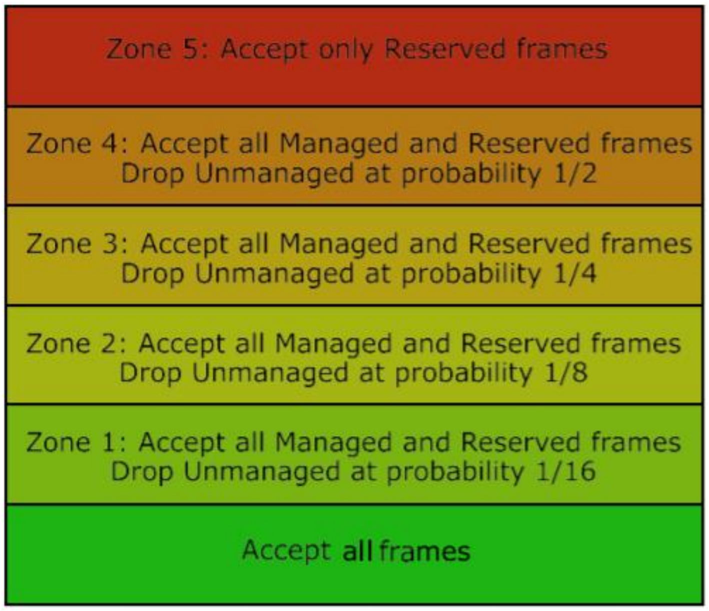
Zone 5: Accept only Reserved frames
Zone 4: Accept all Managed and Reserved frames Drop Unmanaged at probability 1/2
Zone 3: Accept all Managed and Reserved frames Drop Unmanaged at probability 1/4
Zone 2: Accept all Managed and Reserved frames Drop Unmanaged at probability 1/8
Zone 1: Accept all Managed and Reserved frames Drop Unmanaged at probability 1/16
Accept all frames

Figure 3. **WRED Zones**

Shaper

Frames passing the WRED are routed to the last Ingress QoS component: The port-based rate shaper. This shaper is implemented for the complete port traffic. It implements a standard token bucket algorithm to count the number of bytes to be admitted. The user can configure its parameters and leverage maximum data rate (64 kbps to 2.4 Gbps) to be received by the particular ingress interface.

The Ingress QoS also provides statistics to inform the user about the number of drops on the particular stages.

Egress QoS

The Egress QoS is located within the egress stage of the PFE packet processing pipeline. It is implemented on a per-interface basis in a form of set of queues (8x), schedulers (2x), and shapers (4x) with configurable topology.

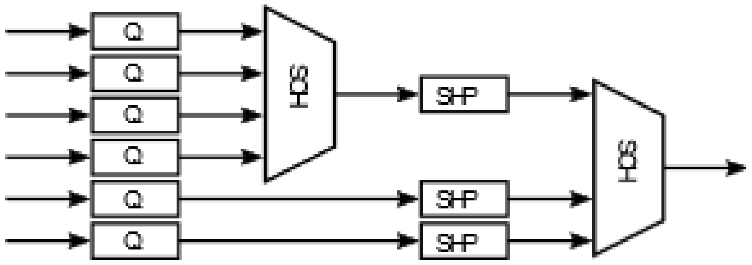

Figure 4. **Egress QoS Topology Example**

Queue

Queue is an entity representing input FIFO of the Egress QoS block used to temporarily hold packets to be transmitted. It can also be understood as an interface through which packets belonging to a certain traffic class are being transmitted. It has limited size and once this limit is reached, all attempts to enqueue additional packets into the full queue ends with so called “tail drop behavior” when the new packets are dropped. Packets are written to particular queues by the PFE classifier engine.

The mapping between traffic class and destination queue is given by configurable priority mapping table and the user can configure the PFE to map packets to queues based on TOS/TrafficClass value from IPv4/IPv6 header. Transmission of packets from queues is managed using scheduler(s) and various scheduling algorithms can be used to arbitrate the queues.

Scheduler

Scheduler is a component of the Egress QoS which is responsible for traffic aggregation from multiple queues. Its main task is to select the packet from a queue which will be transmitted next. For this purpose, the schedulers implement the following scheduling disciplines:

- Simple Round Robin (RR)
- Deficit Weighted Round Robin (DWRR)

  Every queue is configured with a specific percentage of link speed as the weight. Quota and deficit parameters are used to guarantee committed bandwidth and fair chance for each queue.
- Priority Queuing (PQ)

  Every queue has been assigned a priority. The high priority queue must be emptied before accessing the lower priority queue.

  The maximum number of inputs assigned to a single scheduler is 8.

Shaper

The Egress QoS implements 801.Q Credit Based Shapers. Its purpose is to space out the frames as much as possible, reduce bursting in case of data peaks, and control maximum egress data rate (in data rate or packet rate units). In general, shaper operates with configurable clock ticks. With every tick the internal credit value is updated according to configured parameters. When the input queue has data and the credit value is positive, the queue is serviced, and the credit value is decreased. When credit is negative the input queue will not be serviced. The user can configure position and particular parameters of shaper according to application needs.

Flexible Parser and Flexible Router

The PFE classifier does a lookup on standard L2/3/4 header fields to make a routing decision. The flexible parser allows proprietary fields to be part of the routing decision. A similar scheme is implemented in HW for the GMAC on S32G. For PFE this is a firmware-based solution.

Flexible Filter

Flexible Filter allows users to define a table of rules associated with certain actions. Each rule specifies data, data position, and mask to be matched with content of an Ethernet frame. When a frame payload matches the rule, the associated action (accept, reject, jump to another rule) is executed.**Can2Ethernet**

Ability to route CAN frames to Ethernet and vice-versa with zero host intervention. All running on M0+ LLCE core.

2.1. Premium Features

IDPS

Intrusion detection and prevention system extensions, in cooperation with Argus PFE Firmware API, for the customer library were created. This allows customer-specific classification of packet flows at the firmware level. It offers performance boost of customer applications and added flexibility. Please contact an NXP representative for more details on IDPS.

IPsec

The base firmware includes IPsec support to offload protected packets to the HSE via the utility PE within PFE. Base firmware does not make use of the additional IPsec protocol acceleration feature provided by the HSE (i.e., IPsec ESP encapsulation and decapsulation). Firmware modifications takes advantage of this feature and improve the performance of our solution.

3. Supported Targets

Please see the Release Notes document for release-specific details.

- Supported target: S32G2 and S32G3 processors

4. Quality, Standards Compliance and Testing Approach

The S32G PFE Firmware, MCAL and QNX SW products are developed according to NXP Software Development Processes that is Automotive-SPICE, IATF16949 and ISO 9001 compliant.

5. Document Information

Table 1. **Sample revision history**

Revision number	Date	Substantive changes
1.0	10/2021	Initial release
1.7	2/2022	Corrected Figure 1 and 4
2.0	2/2023	Updated of S32G

Document Number: 2.0
Rev. 2.
02/2023